

國立成功大學
電機工程學系
博士論文

即時全球導航衛星系統軟體無線電接收器之設計與實現
及其於干擾與電離層閃焰情況下之應用

Design and Implementation of Real-Time GNSS Software
Receiver and its Applications in the Presence of
Interference and Ionospheric Scintillation

研究生：陳育暄

Student: Yu-Hsuan Chen

指導教授：莊智清

Advisor: Jyh-Ching Juang

Department of Electrical Engineering
National Cheng Kung University
Tainan, Taiwan, R.O.C.
November 2011

中華民國一百年十一月

授權書

本授權書所授權之論文為本人在 國立成功 大學 電機工程學 系 控制 組
一百 學年度第 一 學期所撰 博 士學位論文。

論文名稱：即時全球導航衛星系統軟體無線電接收器之設計與實現及其於干擾與電離層閃焰情況下之應用

☒ 同意 ☐ 不同意

本人具有著作財產權之論文提要，授予國家圖書館、本人畢業學校及行政院國家科學委員會科學技術資料中心，得重製成電子資料檔後收錄於該單位之網路，並與台灣學術網路及科技網路連線，得不限地域時間與次數，以光碟或紙本重製發行。

☐ 同意 ☒ 不同意

本人具有著作財產權之論文全文資料，授予行政院國家科學委員會科學技術資料中心，得不限地域時間與次數以微縮、光碟重製後發行，並得享該中心微縮小組製作之研究報告、獎勵代表作、博碩士論文三檔資料等值新台幣伍佰元之服務。本論文因涉及專利等智慧財產權之申請，請將本論文全文延後至民國 __ 年 __ 月後再公開。

☒ 同意 ☐ 不同意

本人具有著作財產權之論文全文資料，授予教育部指定送繳之圖書館及本人畢業學校圖書館，為學術研究之目的以各種方法重製，或為上述目的再授權他人以各種方法重製，不限時間與地域，惟每人以一份為限。

上述授權內容均無須訂立讓與及授權契約書。依本授權之發行權為非專屬性發行權利。依本授權所為之收錄、重製、發行及學術研發利用均為無償。

指導教授姓名：莊智清

研究生簽名：
(親筆正楷)

陳育暄

學號：N28931065

日期：民國 100 年 11 月 15 日

- 備註：
1. 上述同意與不同意之欄立若未鈎選，本人同意視同授權。
 2. 授權第二項者，請再交論文一本予承辦人員。
 3. 本授權書已於民國 85 年 4 月 10 日送請著委會修正定稿。

簽署人須知

1. 依著作權法的規定，任何單位以網路、光碟與微縮等方式整合國內學術資料，均須先得到著作財產權人授權，請分別在三種利用方式的同意欄內鉤選並填妥各項資料。
2. 所謂非專屬授權是指被授權人所取得的權利並非獨占性的使用權，授權人尚可將相同的權利重複授權給他人使用；反之即為專屬授權，如果您已簽署專屬授權書予其他法人或自然人，請勿簽署本授權書。

3. 授權人的權利與義務：

在美國授權博碩士論文予 U M I 公司(博碩士論文全文資料發行公司)製作發行，須交付美金 45 元的出版費，銷售年逾七件以上時得享收入 10% 的權利金約美金 20 元；在國內本計畫之經費全數由政府支應，收入亦應歸國庫，為答謝您的支持，科資中心特為您提供新台幣 500 元的等值資料服務(以研究報告、獎勵代表作、博碩士論文三檔為限)，請逕洽本案聯絡人，地址電話詳如第 5 項。義務方面唯一要注意是，著作人日後不可以主張終止本授權書，但您仍可以授權其他自然人或法人上述的行為。

4. 全國博碩士論文全文資料微縮片整合計畫的宏觀效益：

在個人方面，您的論文將可永久保存（微縮技術在理論上可保存八百年，實證已逾百年），也因為您的授權，使得後進得以透過電腦網路與光碟多管道檢索，您的論文將因而被充分利用。在國家總體利益方面，紙本容易因影印而造成裝訂上的傷害，圖書館中孤本的公開陳列與外借也有破損之虞，唯有賴政府全面性的整合，借助科技設備才能一舉完成保存與利用的全方位效益，回憶您過去尋找資料之不便經驗，學弟與學妹確實須要您的論文與授權書。

5. 本案聯絡電話：(02)7377746 江守田、王淑貞

地址：台北市和平東路二段 106 號 17 樓 1702 室

研究生姓名：陳育暄 聯絡電話：(07)3790960

地址：高雄市鳥松區澄明街 110 巷 3 號

國立成功大學電機工程學系
博士論文

即時全球導航衛星系統軟體無線電接收器之設計與實現
現及其於干擾與電離層閃焰情況下之應用

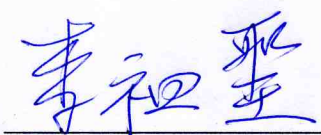
研究生:陳育暄

本論文業經審查及口試合格特此證明


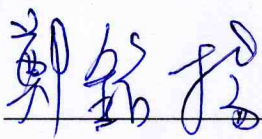
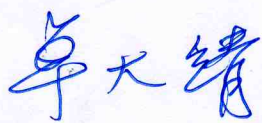
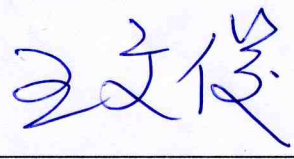
博士學位論文考試委員



張帆人



李仁聖



指導教授：



系主任：



中華民國一百年十一月十一日

Design and Implementation of Real-Time GNSS Software Receiver and its Applications in the Presence of Interference and Ionospheric Scintillation

by

Yu-Hsuan Chen

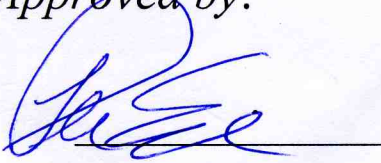
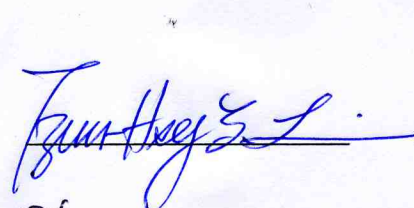
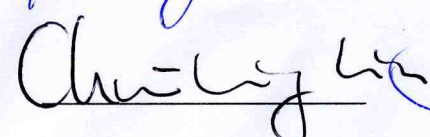
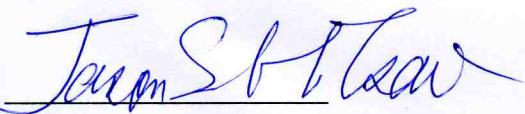

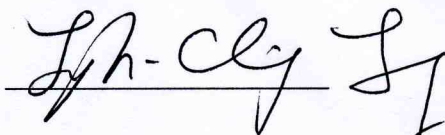
*A dissertation submitted to the graduate division in
partial fulfillment of the requirement for the degree of
Doctor of Philosophy*

at

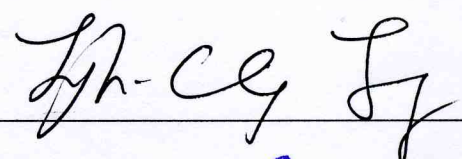
National Cheng Kung University
Tainan, Taiwan, Republic of China

November 2011

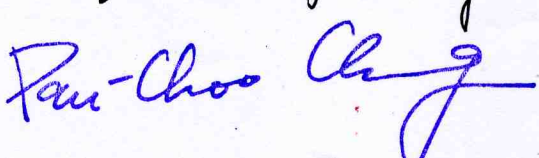
Approved by:

		Wen-June Wang
Jen-Ren Chang		
	Ming-Gang Chang	

Advisor:



Chairman:



即時全球導航衛星系統軟體無線電接收器之設計 與實現及其於干擾與電離層閃焰情況下之應用

陳育暄* 莊智清**

國立成功大學電機工程學系

摘要

隨著全球導航衛星系統的發展，衛星的數量隨之增加，其訊號也跟著變的多元與複雜。另外，全球導航衛星系統之訊號是相對微弱的，容易受到惡意或無意的干擾影響使之無法正常接收。再者，劇烈的太陽活動使得電離層極不穩定，導致訊號的振幅急遽的變化，此現象稱之為電離層閃焰。由於以上之原因，現今具有強健性的全球衛星系統接收器是難以設計的。軟體無線電是一種設計無線電系統的方法，其主要的目的在簡化硬體的元件，之後將其餘的運算用軟體來實現。因此，軟體無線電系統具有彈性與多元化的優點。在此篇論文中，將設計與實現全球導航衛星系統之軟體接收器，並將其應用在對二元偏置載頻訊號之電碼鑑別器設計、抗電離層閃爍效應、雙頻段、抗干擾。其挑戰在於如何達到即時性以及各應用的性能。對於電碼鑑別器設計，本文提出一種基於多相關器架構與最佳化編程之適應性法則，其作用在於降低錯誤鎖定在旁峰的機率，與達到最佳化的追蹤誤差以及多路徑效應的性能。此鑑別器將由 DSP/FPGA 的平台實現，並接收真實的 Galileo 訊號來驗證。對於電離層閃焰，本文提出一種基於粒子濾波器演算法的估測器來估測訊號振幅變動，此方法在劇烈的電離層閃爍環境下，提供強健的追蹤性能，此法則使用已知的電離層閃爍模型以及 MATLAB 程式來做模擬驗證。對於雙頻，本文實現首次使用民用 L5 訊號來定位的軟體接收器，並使用 L1/L5 雙頻組合去除虛擬距離中的電離層延遲，此雙頻接收器接收

GPS/WAAS 訊號作後處理驗證。對於干擾，本文實現了兩個接收器以控制天線接收增益來做波束合成，並撰寫平行運算程式，使之可達到即時性能。實驗結果顯示此接收器能夠抗強度高的干擾訊號以及多個干擾源。



* 研究生

** 指導教授

Design and Implementation of Real-Time GNSS Software Receiver and its Applications in the Presence of Interference and Ionospheric Scintillation

Yu-Hsuan Chen* Jyh-Ching Juang**

**Department of Electrical Engineering
National Cheng Kung University
Tainan, Taiwan, R.O.C.**

Abstract

The Global Navigation Satellite System (GNSS) has undergone continuous evolution as witnessed by the planned deployment of additional constellations and broadcasting of new signals for enhanced performance. Nevertheless, GNSS signals remain relatively weak and vulnerable to deliberate and/or unintentional interferences. In addition, severe sun activity will perturb the ionosphere and cause the scintillation effect which leads to rapid changes in amplitude and phase of GNSS signals. Thus, the design of a robust GNSS receiver against interferences and scintillation is essential. The software radio which is noted by its flexibility and diversity is an approach to design a radio system by reducing the hardware components and exploiting the computational power of processor. In this dissertation, techniques in code discriminator design for binary offset carrier (BOC) signals, detection scheme in accounting for data intermittency, and multithread implementation to meet real-time requirements are developed. The techniques are implemented in GNSS software-based receivers using DSP/FPGA-based and PC-based platforms. The software receivers are then applied to account for scintillation mitigation, dual-band signal reception, and interference rejection. For code discriminator design, an adaptive scheme based on multi-correlator architecture and optimization programming is proposed. Benefits of such

code discriminator include reducing the chance of false lock on side peaks and obtaining optimal performance of tracking error and multipath. This code discriminator is implemented by a DSP/FPGA-based software receiver and examined by receiving real Galileo signal. For scintillation, a particle filter based approach used to estimate the rapid change of signal amplitude is proposed. This approach leads to robust tracking on GNSS signal under sever scintillation situation. For dual-band signal reception, the first receiver that is capable of positioning using L5 signal is implemented. The dual-frequency L1/L5 combination is made to eliminate the ionosphere delay. The dual-band software receiver is examined by receiving GPS/WAAS signal and running in the post-processing mode. For interference rejection, two kinds of real-time software receiver implementations for controlled reception pattern antenna array processing (CRPA) are made to validate the beamforming algorithm. Experimental results show that the receiver is capable of rejecting multiple interferences with high interference-to-signal ratio (I/S).

* **The Student**

** **The Advisor**

Acknowledgement

I would like to express my sincere gratitude to my advisor, Professor Jyh-Ching Juang, for his encouragement and invaluable guidance on my studying in making this dissertation possible. I am greatly influenced by his rigorous attitude toward scientific research. I am also grateful to him for giving me much inspiration on the research. I sincerely appreciate Professor Per Enge for inviting me to visit GPS Lab at Stanford University and giving me a precious direction of research. Allow me to extend my work to diverse applications.

I would like to thank all those who give me many useful comments and suggestions to make my dissertation more complete, Professor Tzue-Hseng S. Li, Professor Wen-June Wang, Professor Fan-Ren Chang, Professor Chun-Liang Lin, Professor Jason Sheng-Hong Tsai, Professor Ming-Yang Cheng and Professor Dah-Jing Jwo.

I would like to express the deepest appreciation to all people of Mechatronic Lab in the department of EE, NCKU for their company and assistance during the course of the research work. Especially, Yung-Fu Tasi, Chiu-Teng Tsai and Tsai-Ling Kao help me a lot all my way. I also want to thank all people of GPS Lab at Stanford University, Dr. Sherman Lo, Professor Dennis M. Akos and Dr. Jiwon Seo, and Dr. David S. De Lorenzo. I received many advices and instructions from them when I was at Stanford. I also would like to thank my family and friends for their support and encouragement during my studying toward Ph.D. degree. Finally, I want to thank my wife, Chien-Ho Chen, for paying a lot of patient and giving the greatest encouragement to me.

Contents

摘要	i
Abstract	iii
Acknowledgement	v
Contents	vi
List of Figures	ix
List of Tables	xii
List of Abbreviation	xiii
Chapter 1 Introduction	1
1.1 Global Navigation Satellite System and Signals	1
1.2 Software Radio	3
1.3 Architecture of GNSS Software Receiver and Literature Survey	4
1.4 Motivation	5
1.5 Contributions	6
1.6 Organization	7
Chapter 2 Design and Implementation of an Adaptive Code Discriminator in a DSP/FPGA-Based Galileo Receiver	9
2.1 Code Tracking Design	10
2.2 Design Results	19
2.2.1 Discriminator Design I	20
2.2.2 Discriminator Design II	23
2.3 Adaptive Switching Logic	27
2.4 Implementation and Test	28
2.5 Test Results	30
2.6 Summary	32
Chapter 3 PC-Based Real-time Software Receiver	33
3.1 Software Architecture	34
3.1.1 Software Correlator	36
3.1.2 Tracking Procedure	39
3.2 Data Intermittency Issue	43
3.3 Solving the Data Intermittency	44
3.4 Experimental Results	46
3.5 Summary	50
Chapter 4 Robust GNSS Signal Tracking against Scintillation Effects by a Particle Filter Based Software Receiver Approach	51
4.1 Scintillation Effect	52
4.2 Introduction of Particle Filter	53

4.3	Using Particle Filter for Carrier Tracking in a Software Receiver.....	55
4.4	Simulation of Signal Generator and Receiver with Scintillation	58
4.5	Summary	62
Chapter 5 Dual-Frequency (L1/L5) GPS/WAAS Software Receiver.....		63
5.1	Signal Specification and Status of GPS/WAAS L5 Signal.....	64
5.2	Strategy of Positioning Using L5 Signal	66
5.3	Description of Signal Collection Hardware.....	67
5.4	Software Architecture.....	68
5.5	Assistance Mechanism between L1 and L5.....	71
5.6	Solving the Time Offset between GPS Time and WAAS Time	72
5.7	Solving the Time Offset between L1 and L5.....	74
5.8	Positioning Result of Dual Frequency Software Receiver	75
5.9	Summary	76
Chapter 6 Software Receiver for GPS Controlled Reception Pattern Antenna Array		
	Processing	79
6.1	Introduction.....	80
6.2	Beamforming Algorithm Used in the Software Receiver	82
6.3	Obtaining the Steering Vector without a Prior Calibration.....	86
6.4	Implementation I	88
6.4.1	Hardware Architecture	88
6.4.2	Software Architecture.....	90
6.4.2.1	Weight and Sum Code Example.....	92
6.5	Implementation II	93
6.5.1	Hardware Architecture	94
6.5.2	Software Architecture.....	96
6.5.2.1	Automatic Gain Control.....	98
6.5.2.2	Software Correlator	98
6.5.2.3	Acquisition/Tracking and Positioning.....	102
6.5.2.4	ICP Initialization and Calculation of Differential ICP	103
6.5.2.5	Adaptive MVDR Beamforming	104
6.5.2.6	Weight and Sum.....	107
6.5.2.7	Bias Calibration	107
6.5.2.8	Calculation of Angle-Frequency Response	108
6.5.2.9	Multi-Threaded Programming for Real-time Validation	109
6.6	Calibration of Antenna Array by Carrier Phase Precise Positioning.....	110
6.7	Analysis of Thread Activities and Timing Performance.....	112
6.7.1	Implementation I	112

6.7.2	Implementation II	114
6.8	Experimental Results	116
6.8.1	Implementation I	116
6.8.1.1	Experiment I – Enhancing C/No	117
6.8.1.2	Experiment II – Rejecting Interference.....	119
6.8.2	Implementation II	121
6.8.2.1	Scenario I – Single High I/S Interference	122
6.8.2.2	Scenario II – Multiple Interferences.....	125
6.9	Summary	127
Chapter 7 Conclusions.....		129
References		131
Publication List		138
Vita		140



List of Figures

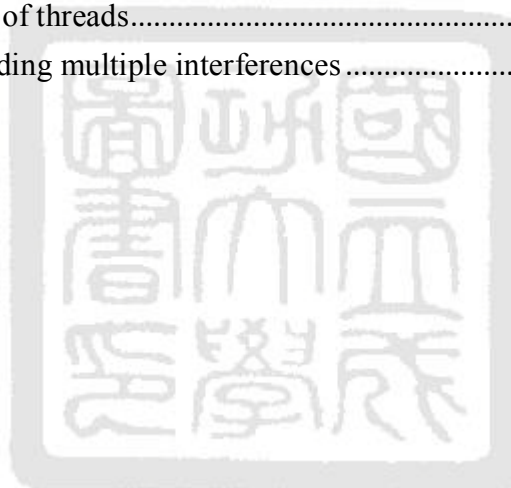
Figure 1.1 Architecture of GNSS software receiver	4
Figure 2.1 Multi-correlator code tracking loop architecture	12
Figure 2.2 Shape of the discriminator function.....	17
Figure 2.3 Block diagram of the linearized code tracking loop	18
Figure 2.4 Discriminator functions at the lock stage.....	22
Figure 2.5 Comparison of code tracking error variances	22
Figure 2.6 Comparison of multipath-induced code tracking errors.....	23
Figure 2.7 Comparison of running average multipath errors	23
Figure 2.8 Discriminator functions at the pull-in stage	25
Figure 2.9 Nonlinear simulation diagram.....	26
Figure 2.10 Comparison of transient responses	26
Figure 2.11 Comparison of regions of convergence	26
Figure 2.12 ROC with NELP discriminator and 1/4 spacing	27
Figure 2.13 DSP/FPGA based Galileo receiver prototype	29
Figure 2.14 Receiver state machine.....	30
Figure 2.15 I-Q plot of received GIOVE-A E1-B components.....	31
Figure 2.16 Analysis of initial delay and convergence performance.....	31
Figure 3.1 Hardware architecture of a PC-based GNSS software receiver.....	34
Figure 3.2 Flow chart of the GNSS software receiver.....	35
Figure 3.3 Structures of code table and carrier table	37
Figure 3.4 Example of counting the number of correlation results using SSE instructions	38
Figure 3.5 State machine of the tracking procedure	40
Figure 3.6 Carrier tracking loop in the software receiver.....	41
Figure 3.7 Data intermittency issue of RF front-end.....	44
Figure 3.8 State flow of the proposed tracking procedure	46
Figure 3.9 Effects of data intermittency on tracking	47
Figure 3.10 Comparisons of code phase, pseudorange and local time between original and proposed approach.....	49
Figure 4.1 Example of scintillation effect on the GNSS signal	53
Figure 4.2 Procedure of particle filter.	54
Figure 4.3 Carrier tracking employing particle filter in a software receiver.	55
Figure 4.4 Schematic diagram of particle filter for carrier tracking	56
Figure 4.5 Hypothesis testing for navigation data detector	57
Figure 4.6 Signal flow of simulation for scintillation.....	58
Figure 4.7 Simulation results of carrier tracking for comparison between particle filter and PLL under the condition of $S/N = -30$ dB and severe scintillation	59

Figure 4.8 Original β from scintillation as well as noise compared with estimated β by least squares estimator with different number of taps.	60
Figure 4.9 I & Q correlator outputs of receiver in the upper plot and accumulated Q correlator outputs and demodulation navigation data in the lower plot.....	61
Figure 5.1 Block diagram of the signal collection hardware	68
Figure 5.2 Block diagram of the software architecture	68
Figure 5.3 Screenshot of dual-frequency software receiver GUI	70
Figure 5.4 Results of time offset calculated by difference between true range and pseudorange (PDOP = 1.827)	74
Figure 5.5 EN plot of L1 positioning w/ and w/o WAAS time offset unknown respect to precise position (PDOP = 1.827).....	74
Figure 5.6 EN plot of positioning by L1 only, L5 only, and iono-free respect to precise position (PDOP = 8.876).....	76
Figure 6.1 Architecture of STAP with adaptive MVDR beamforming	84
Figure 6.2 Antenna geometry and direction of satellite showing calculation of $\Delta\phi_i^l$	86
Figure 6.3 Mechanism to obtain the steering vectors in a software receiver	87
Figure 6.4 Diagram of hardware set-up of CRPA software receiver	89
Figure 6.5 Procedure of IF data transfer from the USB interface to the circular queues	91
Figure 6.6 Planned execution flow of threads every msec	91
Figure 6.7 Example of the weight-and-sum operation using SSE instructions.....	93
Figure 6.8 Block diagram of the signal collection hardware	95
Figure 6.9 Software flow of software receiver in the implementation II	96
Figure 6.10 Structure of code table, carrier table, and IF data	99
Figure 6.11 Complex data format used for executing PMADDWD instruction.....	100
Figure 6.12 Buffer structure, correlation window and three cases for correlation	102
Figure 6.13 Architecture of carrier tracking loop of software receiver	103
Figure 6.14 Architecture of weight and sum	107
Figure 6.15 Plan of execution flow of software program within 1 msec	110
Figure 6.16 Block diagram of the calibration procedure of antenna array.....	112
Figure 6.17 Execution flow of the threads of the CRPA real-time software receiver collected from the Intel Thread Profiler collector program	113
Figure 6.18 Box plot of the execution time illustrating the real time capability of the CRPA.....	114
Figure 6.19 Execution flow of the software receiver collected from.....	115
Figure 6.20 Histogram of duration to illustrate the real time capability of software receiver	115
Figure 6.21 Hardware setup of the software receiver	116

Figure 6.22 Relative position of the antennas from the carrier phase precise positioning	117
Figure 6.23 Filtered C/No of the software receiver with and w/o CRPA beam steering...	118
Figure 6.24 Gain patterns of the composite antennas by the CRPA toward the specified satellite in a sky plot format.....	118
Figure 6.25 Diagram of the hardware set-up of the CRPA software receiver with the interference using GPS simulator	119
Figure 6.26 Power spectral density of IF signal with and without interference	120
Figure 6.27 Filtered C/No of the software receiver performing CRPA by MVDR and deterministic beamforming in the cases with CDMA and CW interference ..	121
Figure 6.28 Relative position of the antennas of data collection	122
Figure 6.29 Filtered C/No of all-in-view CRPA and WAAS GEOs.....	123
Figure 6.30 Gain patterns of the composite antennas without and with interference and sky plot showing the directions of the satellite and interference.....	124
Figure 6.31 Positioning result of the software receiver without and with interference.....	124
Figure 6.32 Sky plots, gain patterns and angle-frequency responses of four cases when adding multiple interferences	126
Figure 6.33 All-in-view CRPA filtered C/No when adding multiple interferences	127

List of Tables

Table 1.1 GNSS signals list	2
Table 1.2 Description of component in GNSS software receiver	5
Table 3.1 Description of component in software receiver	36
Table 5.1 Signal specification of GPS/WAAS L5	64
Table 5.2 Status of GPS/WAAS L5 before May 6 th 2011	65
Table 5.3 Issues and solutions of positioning using L5	67
Table 5.4 Methods of calculating position	70
Table 5.5 User range accuracy (URA) of satellites on collection day	73
Table 5.6 Time offset and antenna phase center difference between L1 and L5	75
Table 5.7 Positioning accuracy	76
Table 6.1 Component list of software architecture in the implement I	90
Table 6.2 Component list of software architecture in the implementation II	97
Table 6.3 Execution time of threads	114
Table 6.4 Sequence of adding multiple interferences	125



List of Abbreviation

ADC	Analog to Digital Converter
AltBOC	Alternative-BOC
APNT	Alternative Position Navigation and Timing
BOC	Binary Offset Carrier
bps	bit per second
BPSK	Binary Phase Shift Keying
C/A	Coarse/Acquisition
CDMA	Code Division Multiple Access
C/No	Carrier to Noise ratio
CPU	Central Processing Unit
CRC	Cyclic Redundancy Check
CRPA	Controlled Reception Pattern Antenna array
CW	Continuous Wave
DAQ	Data AcQuisition card
DLL	Delay Locked Loop
DoA	Direction of Arrival
DOP	Dilution of Precision
DSP	Digital Signal Processing
ENU	Earth North Up
EGNOS	European Geostationary Navigation Overlay System
FAA	Federal Aviation Administration
FDMA	Frequency Division Multiple Access
FEC	Forward Error Correction
FLL	Frequency Locked Loop
FPGA	Field Programmable Gate Array
GDOP	Geometric Dilution of Precision
GEO	Geostationary Orbit
GIOVE	Galileo In-Orbit Validation Element
GLONASS	GLObal NAVigation Satellite System
GNSS	Global Navigation Satellite System

GPS	Global Positioning System
GPU	Graphic Processing Unit
GUI	Graphical User Interface
ICP	Integrated Carrier Phase
IFB	Inter-Frequency Bias
IGSO	Inclined Geosynchronous Orbit
IF	Intermediate Frequency
I/S	Interference-to-Signal ratio
IOC	Initial Operational Capability
LAMBDA	Least-squares AMBIGUITY Decorrelation Adjustment
MBOC	Multiplexed Binary Offset Carrier
MC	Multi-Correlator
MSAS	Multi-functional Satellite Augmentation System
MVDR	Minimize Variance Distortionless Response
NELP	Noncoherent Early-minus-Late Power
NMC	Noncoherent Multi-Correlator
NCO	Numerically Controlled Oscillator
NH	Neuman-Hoffman
ROC	Region Of Convergence
PC	Personal Computer
PDOP	Position Dilution of Precision
PLL	Phase Locked Loop
PN	Pseudo Noise
PNT	Positioning, Navigation, and Timing
PRN	Pseudo-Random Noise
PRS	Public Regulated Service
QPSK	Quadrature Phase Shift Keying
RFI	Radio Frequency Interference
SBAS	Satellite Based Augmentation System
SFAP	Space-Frequency Adaptive Processing
SIMD	Single Instruction Multiple Data

S/N	Signal to Noise ratio
STAP	Space-Time Adaptive Processing
sps	symbol per second
SPS	Standard Positioning Service
SSE	Streaming SIMD Extensions
SVN	Satellite Vehicle Number
TOW	Time Of Week
WAAS	Wide Area Augmentation System
w/o	without
URA	User Range Accuracy
USB	Universal Serial Bus



Chapter 1

Introduction

1.1 Global Navigation Satellite System and Signals

Global Navigation Satellite System (GNSS) provides Position, Navigation and Time (PNT) for worldwide users. Several systems already exist or are under development. United States' Global Positioning System (GPS) [57,58,61] is most well-known and fully operational. GPS constellation consists of 24 satellites (31 satellites available in 2011) and its civilian used-widely signal is the L1 C/A. The modernized civilian signal of GPS is L2C/L5 signal which will be broadcasted in every next launching satellite. Russian Global Orbiting Navigation Satellite System (GLONASS) [54] is finishing its constellation of 24 satellites. Frequency division multiple access (FDMA) signal is the primary multiple access scheme for GLONASS. For being compatible to other system, code division multiple access (CDMA) signal will be broadcasted on L1 in next generation of satellite and has been broadcasted on L3 by GLONASS-K1 satellite. European's Galileo system [54] has two test beds GIOVE-A and GIOVE-B in orbit from 2008. Galileo plans to complete 30 satellites in orbit for fully constellation. China's Compass now has 3 geostationary orbit (GEO) and 4 inclined geosynchronous orbit (IGSO) satellites which are usable. Compass plans to complete a constellation of 35 satellites in 2020 [54]. In the future, there will be plenty of navigation satellites with different frequency bands and signal specifications. This provides advantages of signal diversity and better geometry. Thus, centimeter accuracy could be achieved by using civilian signal only. Table 1.1 lists

the specifications of the future available GNSS signals. Future GNSS receiver requires supporting multi-band and multi-constellation. Thus, this will increase the complexity of receiver design.

Additionally, Satellite Based Augmentation System (SBAS) including Wide Area Augmentation System (WAAS), European Geostationary Navigation Overlay System (EGNOS), and Multi-functional Satellite Augmentation System (MSAS) provide augment GPS for improving its accuracy, integrity, and available. SBAS transmit GPS-like signals and provide ranging information which could be used in the positioning. For WAAS, L1 and L5 signals are broadcasted from three WAAS GEOs in 2011 [62].

Table 1.1 GNSS signals list

	GPS	GLONASS	Galileo	Compass
Constellation	24(31)	24	30	35
Civilian Signal	L1(1575.42MHz) L1C(1575.42MHz) L2C(1227.60MHz) L5(1176.45MHz)	L1(1602MHz) L2(1246MHz) L3(1207MHz)	L1(1575.42MHz) E5a(1176.45MHz) E5b(1207.14MHz) E6(1278.75MHz)	E1(1589.74MHz) E2(1561.10MHz) E6(1268.52MHz) E5b(1207.14MHz)
Modulation Scheme	BPSK	BPSK	MSK, AltBOC	BPSK, QPSK
Multiple Access	CDMA	FDMA, CDMA	CDMA	CDMA

The GNSS signal is relatively weak and is vulnerable to interfered by unintentional or intentional signal. Especially, GNSS jammers are easily available in the internet and block the GNSS reception. These render vulnerable the use of GNSS for any applications of position, navigation and time [50]. Besides, the other effect degrading GNSS signals dramatically is ionospheric scintillation which is due to solar activity. Every 11 years, the sun enters a period of solar maximum and produce disruption to radio signal [47]. During this period, the amplitude and phase of GNSS signal will change rapidly. This causes

losing lock of the tracking loop in the receiver.

1.2 Software Radio

The definition of software radio borrowed from [12] is “software radio is a thought to build flexible radio systems, multiservice, multiband, multi-standard, reconfigurable and reprogrammable by software.” Its fundamental philosophy of design are to 1) put the Analog-to-Digital Converter (ADC) as near as possible to the antenna in the front-end and 2) process the resulting samples using programmable processor. The benefits of software radio are listed as follows.

1. Flexibility: being controlled, configured and programmed by software.
2. Reconfigurability: update radio functionality by download software without change of hardware.
3. Diversity: capable of implementing multiband and multi-standard radio systems.
4. Computational-complexity: use a processor with high computing power to implement complex algorithm.

As the progress of semiconductor and antenna, microprocessor and digital signal processor (DSP) are designed with multi-core and instructions of single-instruction-multiple-data. Antennas and front-end are designed to receive wideband signals. Therefore, the radio system design by software radio concept becomes emerging technology and is suitable to apply to complicate GNSS receiver. Plenty of software receivers are already implemented in the literature which architecture is described in the next section. GPS software receiver has become a solution in the cell phone which uses x86 or ARM processor in the phone [64]. Commercial products of PC-based GNSS software receiver have been announced in [35]

1.3 Architecture of GNSS Software Receiver and Literature Survey

Based on the software radio concept, the general hardware architecture of GNSS software receiver [1,2] is depicted in the Figure 1.1. And, its descriptions of each components and its possible realizations are listed in the Table 1.2.

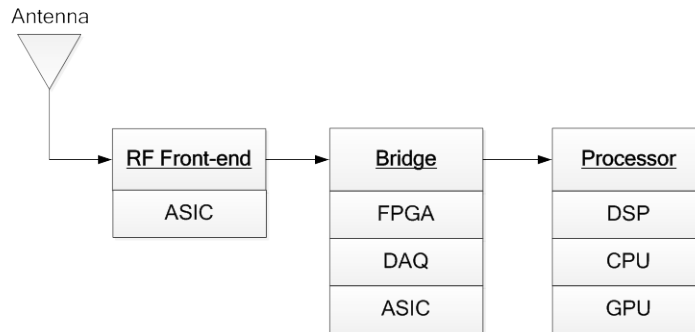


Figure 1.1 Architecture of GNSS software receiver

Software receivers with real-time capability are addressed in the literature. [1] firstly introduced the front-end design and software based signal processing of software radio GNSS receiver. [2] implemented the concept in [1] to complete 4 channels GPS L1 receiver using FPGA and DSP. [59] implemented a hybrid software GPS/Galileo receiver for 24-channel L1 band using USB interference front-end and 1.7 GHz PC. [3] proposed a bit-wise parallel algorithm for software correlation. They also implement a 12-channel software GPS L1 C/A receiver on a 1.73 GHz PC. [52] implemented a GPS 10-channel civilian L1/L2 software receiver by RF front-end, a data acquisition card (DAQ) and 3.2 GHz PC. It had a navigation accuracy of 2-5 meters. Besides, Graphics Processing Units (GPU) which is originally designed to parallel manipulate computer graphics is also extended to applications of science analysis and signal processing. [48] used GPU as a correlation unit and cooperate with CPU for processing GPS L1 C/A signal.

Table 1.2 Description of component in GNSS software receiver

Component	Description	Realization
Antenna	Right-hand circular polarization L band antenna	Patch, Helical, ...
RF Front-end	RF front-end consists of following modules 1. Amplifier : amplify signal to input range of ADC. 2. Down-converter : converter RF high frequency(1~2 GHz) signal to intermediate frequency (0~20 MHz). 3. Filter : filter out out-of-band signal. 4. Analog to digital converter (ADC) : sample the analog IF signal to digital IF signal.	ASIC
Bridge	The bridge serves as a buffer of digital IF signal and sends it to following processor. 1. FPGA : digital IF could be firstly correlated with local replica of code and carrier and then output the correlator results to following processor. 2. DAQ: ADC may be included in this stage. 3. ASIC: conventional IC is USB microcontroller.	FPGA DAQ ASIC
Processor	The processor runs the software to operate following functions. 1. Signal processing : signal acquisition/tracking, data demodulation/decode, and ephemeris attraction 2. Positioning : pseudorange measurement and position calculation.	DSP CPU GPU

1.4 Motivation

Accounting for complexity of future GNSS signal and its vulnerability, the motivation of the dissertation is to design and implement a real-time GNSS receiver by software radio concept. Then, use this software receiver to solve problems as follows.

1. False lock on side peaks for tracking Binary Offset Carrier (BOC) signal.
2. Enhance signal tracking performance including noise rejection and multipath mitigation for BOC signal.
3. Data intermittency issue resulted from buffer overrun while transmission of digital Intermediate Frequency (IF) data.
4. Losing tracking due to rapid change of amplitude and phase of GNSS signal in the scintillation environment.
5. Tracking and positioning using civilian GPS/WAAS L5 signal.
6. Processing multiband signal for eliminate the ionosphere delay.
7. Implementing controlled pattern reception antenna array.
8. Rejecting interferences including high power and multiple sources from different directions.

1.5 Contributions

The main contributions of this dissertation are listed as follows.

1. For Binary Offset Carrier (BOC) signal which is the modulation scheme of Galileo signal, an adaptive scheme on the design of code discriminator is proposed to reduce the chance to false lock on side peaks and get optimal tracking performance. A DSP/FPGA based software receiver is implemented to examine this scheme by receiving real Galileo signal.
2. By software radio concept, a PC-based real-time software receiver is implemented for the reception of GPS L1 C/A signals. Due to its computational complexity, a parallel programming coding method is used to speed up the operations of GNSS receiver. Further, a carrier tracking loop using pre-loading carrier table is built. Besides, a signal-processing logic is proposed to account for the signal data intermittency issue

which is caused by buffer overrun.

3. Ionospheric scintillation resulting from the activity of sun may affect the GNSS signals and lead to rapid changes of phase and amplitude. An open-loop carrier tracking using particle filter is investigated to track the GNSS signal under the scintillation environment. Because of its complexity, this signal tracking method is realized by software radio concept.
4. In the modernization of GPS/WAAS, a new civil signal L5 has been broadcasted for signal diversity. A dual-frequency L1/L5 GPS/WAAS software receiver is implemented by an assistance mechanism between L1 and L5. Positioning result of dual-frequency receiver is firstly obtained by combining the measurements of L1 and L5. Among this positioning procedure, time offsets between GPS and WAAS as well as L1/L5 are solved by adding variables into positioning calculation.
5. Radio Frequency Interference (RFI) is a great treat for weak GNSS signal. A real-time software receiver for controlled reception pattern antenna array is designed and implemented on a PC platform. In the receiver, Space-Time Adaptive Processing (STAP) structure is used and its weights are determined by Minimize Variance Distortionless Response (MVDR) algorithm. This makes the receiver capable of rejecting both narrowband and broadband interferences. A parallel programming approach for operation of receiver by Single Instruction Multiple Data (SIMD) is addressed to achieve real-time capability.

1.6 Organization

The dissertation is organized as follows. In Chapter 2, a DSP/FPGA-based Galileo receiver is described for implementing the proposed adaptive code discriminator. Analyses of multipath and tracking error for these code discriminator designs are provides. Testing

results of the developed Galileo receiver are shown to validation of the design. Chapter 3 describes the design of real-time PC-based software receiver for GPS L1 C/A signal and a signal-processing logic for accounting for data intermittency issue. The details of hardware and software are mentioned. Chapter 4 describes an open-loop tracking using particle filter under scintillation environment. Based on software receiver structure, the procedure of particle filter is designed to provide robust tracking for rapid change of phase and amplitude. Simulation results are provided to verify the benefit of this approach. Chapter 5 describes the development of a dual-frequency L1/L5 GPS/WAAS software receiver. The assistance mechanism between L1 and L5 is addressed to provide positioning ability using L5 signal. The estimation of time offset between L1 and L5 as well as GPS Time and WAAS Network Time are mentioned. The positioning results using three methods L1 only, L5 only and L1/L5 combination are provided to show its benefit of dual-band receiver. Chapter 6 describes the design of a GPS/WAAS software receiver for CRPA. The structure of STAP and the algorithm of MVDR used in this receiver are described. Two kinds of implementation are mentioned in detail including architectures of hardware and software. The results of experiments are shown to validate the interference rejection performance of receiver. Finally, a summary of the main results and some topics of further research are given as the conclusion.

Chapter 2

Design and Implementation of an Adaptive Code Discriminator in a DSP/FPGA-Based Galileo Receiver

In a Global Navigation Satellite System (GNSS) receiver, the code tracking loop is responsible for the synchronization of the local code and the incoming code in the presence of noise and multipath. The design requirements of the code tracking loop thus include small error variance, bounded multipath-induced error, and fast transient response. The design is traditionally realized by a delay-locked loop (DLL) in which a discriminator is used to provide error signal for the adjustment of the timing of the local code in the tracking loop. Several different discriminators including noncoherent early-minus-late power (NELP) discriminator, coherent early-minus-late discriminator, and dot-product discriminator have been discussed in [65,71]. In existing discriminators, the designer typically adjusts the code spacing and loop filter bandwidth to account for multipath-induced error and carrier-to-noise density ratio variation. Future GNSS including European Galileo and modernized GPS will employ binary offset carrier (BOC) modulation or its multiplexed variation to obtain improved performance in comparison with the legacy binary phase shifted keying (BPSK) modulation in GPS [23]. The BOC modulation, however, is subject to an ambiguity problem in the code tracking process. Such an ambiguity issue, together with existing conflicting concerns on steady-state error, multipath-induced error, and transient response, makes the design of the code tracking loop

complicated. Fortunately, parameters of the loop filter and discriminator design can be brought to bear. Several code discriminator designs have been addressed in [7,25,40,60]. In this chapter, a design approach of a noncoherent multi-correlator (NMC) discriminator is investigated. The code discriminator design is formulated as an optimization problem by regarding different code tracking performance requirements as either an objective function to be minimized or constraints to be satisfied. To account for different requirements at the pull-in and lock stages, an adaptive scheme is devised to tailor different discriminators. The adaptive code discriminator is then realized in a Digital Signal Processor/Field Programmable Gate Array (DSP/FPGA) platform for the reception of Galileo signals and experimentally verified by receiving signals from the GIOVE-A, the first Galileo in-orbit validation satellite. The test results reveal that the proposed NMC discriminator leads to an improved performance than the conventional NELP approach.

2.1 Code Tracking Design

In this section, a multi-correlator based code tracking loop for GNSS signal reception is analyzed and the design of the code discriminator is formulated. Figure 2.1 depicts the multi-correlator code tracking loop architecture, which differs from existing DLLs in the sense that a set of correlators is used. After filtering and down-converting in the front-end, the GNSS signal is modeled as [44,65] :

$$s(t) = \sqrt{2P}d(t-\tau)c_f(t-\tau)\cos(2\pi f_{IF}t - 2\pi f_{RF}\tau + \phi) + n(t) \quad (2.1)$$

where P is the power of the received signal, $d(t)$ is the data, $c_f(t)$ is the spreading code after filtering, τ is the code phase delay, f_{IF} is the intermediate frequency, f_{RF} is the radio frequency of the incoming signal, ϕ is the phase of the incoming signal, and $n(t)$ is the narrowband noise. The Doppler frequency is included in the $2\pi f_{IF}t - 2\pi f_{RF}\tau$

term [41]. It can be derived by

$$2\pi f_{IF}t - 2\pi f_{RF}\tau = 2\pi(f_{IF} + f_d)t \quad (2.2)$$

where f_d is Doppler frequency [12]. Substituting (2.2) into (2.1), the incoming signal becomes

$$s(t) = \sqrt{2P}d(t-\tau)c_f(t-\tau)\cos(2\pi(f_{IF} + f_d)t + \phi) + n(t) \quad (2.3)$$

Let $h(t)$ be the impulse response of the front-end filter and $c(t)$ be the original spreading code, the filtered spreading code $c_f(t)$ is related to $c(t)$ by

$$c_f(t) = c(t) * h(t) \quad (2.4)$$

where $*$ stands for the convolution integral operation. As we are primarily interested in the code tracking loop, the effects due to Doppler frequency error and data symbol are not considered for simplicity.

In the code tracking loop, the signal $s(t)$ is multiplied by local I-phase and Q-phase carrier replicas, $\cos(2\pi(f_{IF} + f_d)t + \hat{\phi})$ and $\sin(2\pi(f_{IF} + f_d)t + \hat{\phi})$, respectively, where $\hat{\tau}$ is the time delay estimate, and $\hat{\phi}$ is the phase estimate. The results are then correlated with the local code replica $c(t - \hat{\tau} - d_k)$ where d_k is the correlator position. A set of m correlators are used as depicted in the figure. The standard DLL typically employs a pair of early-late correlators whose positions are located at $d_1 = \frac{\delta}{2}T_c$ and $d_2 = -\frac{\delta}{2}T_c$, respectively, in which δ is the correlator spacing and T_c is the chip interval. The multi-correlator architecture can thus be regarded as a generalization of the traditional early-late DLL.

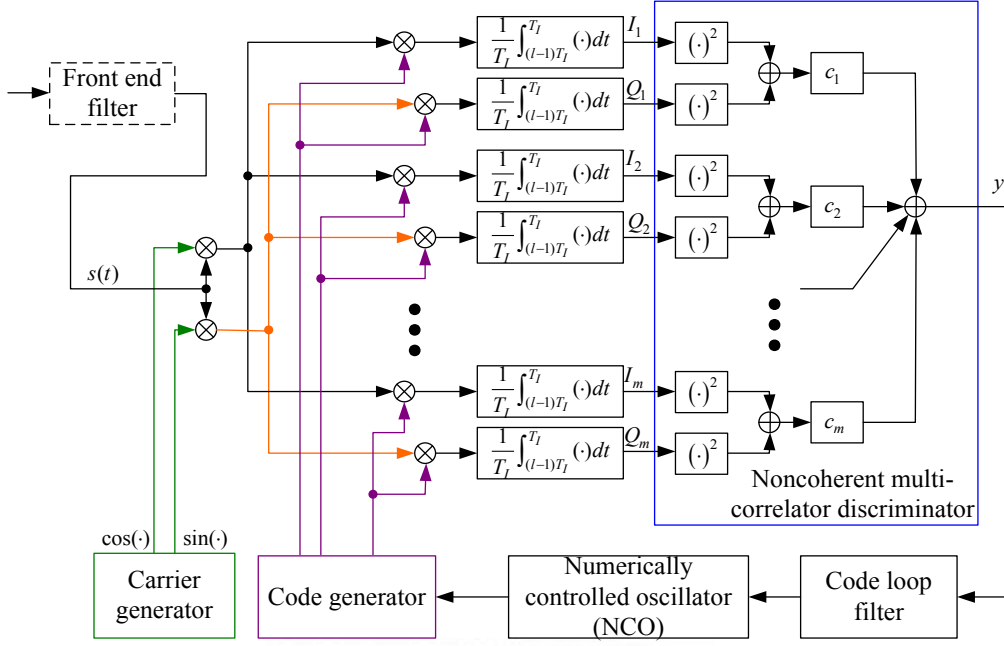


Figure 2.1 Multi-correlator code tracking loop architecture

After integrating for a period of T_I , the I-phase and Q-phase correlator outputs become [65,44]

$$I_k = \sqrt{\frac{P}{2}} \Lambda_f(\tilde{\tau} - d_k) \cos \tilde{\phi} + n_{I_k} \quad (2.5)$$

$$Q_k = \sqrt{\frac{P}{2}} \Lambda_f(\tilde{\tau} - d_k) \sin \tilde{\phi} + n_{Q_k} \quad (2.6)$$

where $\tilde{\phi} = \phi - \hat{\phi}$ is the carrier phase error, $\tilde{\tau} = \tau - \hat{\tau}$ is the code phase error, $\Lambda_f(\tilde{\tau} - d_k)$ is the autocorrelation function between the local code $c(t - \hat{\tau} - d_k)$ and the filtered spreading code $c_f(t - \tau)$, and n_{I_k} and n_{Q_k} are noise components after correlation in the I-phase and Q-phase arms, respectively. The autocorrelation function Λ_f can be expressed as

$$\Lambda_f(x) = \int_{-\infty}^{\infty} S(f) H(f) \exp(-j2\pi fx) df \quad (2.7)$$

where $S(f)$ is the power spectral density of the spreading code $c(t)$ and $H(f)$ is the frequency response of $h(t)$ [11,62]. When the sine-phased BOC(1,1) modulation is used, $S(f)$ is given as [9]

$$S(f) = \frac{1}{T_c} \left(\frac{\sin(\pi f T_c)}{\pi f} \frac{\sin(\pi f T_c / 2)}{\cos(\pi f T_c / 2)} \right)^2 \quad (2.8)$$

Both the noise components n_{I_k} and n_{Q_k} in (2.5) and (2.6) are of zero mean and the variances are [41]

$$E\{n_{I_k}^2\} = E\{n_{Q_k}^2\} = \frac{N_0}{4T_I} \int_{-\infty}^{\infty} S(f) |H(f)|^2 df \quad (2.9)$$

Further, the noise terms n_{I_k} and n_{Q_l} are uncorrelated. The cross-correlation between n_{I_k} and n_{I_l} (likewise, n_{Q_k} and n_{Q_l}) is given by

$$E\{n_{I_k} n_{I_l}\} = \frac{N_0}{4T_I} \int_{-\infty}^{\infty} S(f) |H(f)|^2 \exp(-j2\pi f(d_k - d_l)) df \quad (2.10)$$

The code discriminator combines I_k and Q_k , $k=1,2,\dots,m$, to form an error signal to control the operation of the code tracking loop. The MC discriminator and NMC discriminator bear the following form:

$$\begin{aligned} y_{MC} &= \sum_{k=1}^m c_k I_k \\ y_{NMC} &= \sum_{k=1}^m c_k (I_k^2 + Q_k^2) \end{aligned} \quad (2.11)$$

where c_k are coefficients to be determined. It is noted that when the phase information is available, a coherent discriminator can be used. A coherent discriminator which is not subject to the squaring loss can yield an improved performance compared to a noncoherent one. However, the realization of a coherent discriminator relies on the phase information which may not be available at the pull-in stage. The NMC discriminator output $y(\tilde{\tau})$ is a function of the code phase error $\tilde{\tau}$. Typically, the design objectives of the discriminator include

- (i) The discriminator is unbiased when the code phase error is zero.
- (ii) The discriminator output is sensitive to code phase error.

- (iii) The discriminator is non-ambiguous in the sense that the discriminator function has one and only one zero cross-over.
- (iv) The resulting code phase error in the presence of noise and multipath is as small as possible.

These objectives can be achieved by appropriately designing the coefficients c_k and the accompanying loop filter.

Substituting (2.5) and (2.6) into (2.11), ones have

$$\begin{aligned}
 y_{MC} &= \sqrt{\frac{P}{2}} \sum_{k=1}^m c_k \Lambda_f(\tilde{\tau} - d_k) + \sum_{k=1}^m c_k n_{Ik} \\
 y_{NMC} &= \frac{P}{2} \sum_{k=1}^m c_k \Lambda_f^2(\tilde{\tau} - d_k) + \sqrt{2P} \sum_{k=1}^m c_k \Lambda_f(\tilde{\tau} - d_k) [n_{Ik} \cos \tilde{\phi} + n_{Qk} \sin \tilde{\phi}] \\
 &\quad + \sum_{k=1}^m c_k [n_{Ik}^2 + n_{Qk}^2]
 \end{aligned} \tag{2.12}$$

In the absence of noise and when $\tilde{\tau}$ is zero, it is desired that $y(\tilde{\tau})$ is zero, which is the unbiased-ness condition. From (2.12), the discriminator is unbiased when the following equation is satisfied:

$$\begin{aligned}
 \sum_{k=1}^m c_k \Lambda_f(d_k) &= 0 && \text{(for MC)} \\
 \sum_{k=1}^m c_k \Lambda_f^2(d_k) &= 0 && \text{(for NMC)}
 \end{aligned} \tag{2.13}$$

Note that a special condition for unbiased-ness is that the NMC discriminator is constructed of several early-late gates.

The expected value of $y(\tilde{\tau})$ is

$$\begin{aligned}
 E\{y_{MC}(\tilde{\tau})\} &= \sqrt{\frac{P}{2}} \sum_{k=1}^m c_k \Lambda_f(\tilde{\tau} - d_k) \\
 E\{y_{NMC}(\tilde{\tau})\} &= \frac{P}{2} \sum_{k=1}^m c_k \Lambda_f^2(\tilde{\tau} - d_k) + \sum_{k=1}^m c_k \frac{N_0}{2T_I} \int_{-\infty}^{\infty} S(f) |H(f)|^2 df
 \end{aligned} \tag{2.14}$$

The sensitivity of the discriminator function can be evaluated by taking the derivative of $E\{y(\tilde{\tau})\}$ with respect to $\tilde{\tau}$ [35], giving

$$\begin{aligned}
K_{0,MC} &= \left. \frac{dE\{y_{MC}(\tilde{\tau})\}}{d\tilde{\tau}} \right|_{\tilde{\tau}=0} = \sqrt{\frac{P}{2}} \sum_{k=1}^m c_k \left. \frac{d\Lambda_f(\tilde{\tau}-d_k)}{d\tilde{\tau}} \right|_{\tilde{\tau}=0} \\
K_{0,NMC} &= \left. \frac{dE\{y_{NMC}(\tilde{\tau})\}}{d\tilde{\tau}} \right|_{\tilde{\tau}=0} = P \sum_{k=1}^m c_k \Lambda_f(d_k) \left. \frac{d\Lambda_f(\tilde{\tau}-d_k)}{d\tilde{\tau}} \right|_{\tilde{\tau}=0} = P \mathbf{c}^T \mathbf{g}_0
\end{aligned} \tag{2.15}$$

where the vector \mathbf{c} is defined as $\mathbf{c} = [c_1 \ c_2 \ \cdots \ c_m]^T$ and \mathbf{g}_0 is the $m \times 1$ vector whose k -th entry is $\Lambda_f(d_k) \left. \frac{d\Lambda_f(\tilde{\tau}-d_k)}{d\tilde{\tau}} \right|_{\tilde{\tau}=0}$. Although, in practice, the gain can be arbitrarily selected to meet the design needs, it is customary to compare the gain with that of the NELP discriminator which is [8,37,44]

$$K_{NELP} = 4\pi P \Lambda_f\left(\frac{\delta}{2}\right) \int_{-\infty}^{\infty} f S(f) H(f) \sin(\pi f \delta) df \tag{2.16}$$

Let $\mathbf{h}(\tilde{\tau})$ be the $m \times 1$ vector whose k -th entry is given by $\sqrt{\frac{P}{2}} \Lambda_f(\tilde{\tau}-d_k)$ for MC, and $\frac{P}{2} \Lambda_f^2(\tilde{\tau}-d_k) + \frac{N_0}{2T_l} \int_{-\infty}^{\infty} S(f) |H(f)|^2 df$ for NMC, then (2.14) becomes

$$E\{y(\tilde{\tau})\} = \mathbf{c}^T \mathbf{h}(\tilde{\tau}) \tag{2.17}$$

It can further be shown that in the presence of multipath, the expected discriminator output becomes

$$E\{y(\tilde{\tau})\} = \mathbf{c}^T (\mathbf{h}(\tilde{\tau}) + \mathbf{g}(\tilde{\tau}, \alpha, \beta, \theta)) \tag{2.18}$$

where α , β , and θ are related to the attenuation, delay, and phase of the multipath with respect to the direct line-of-sight signal. When there is one multipath, the k -th entry of

$$\begin{aligned}
\mathbf{g}(\tilde{\tau}, \alpha, \beta, \theta) \quad \text{is} \quad & \sqrt{\frac{\alpha P}{2}} \sum_{k=1}^m c_k \Lambda_f(\tilde{\tau} - \beta - d_k) \cos \theta \quad \text{for MC, and} \\
& \sqrt{\alpha P} \sum_{k=1}^m c_k \Lambda_f(\tilde{\tau} - d_k) \Lambda_f(\tilde{\tau} - \beta - d_k) \cos \theta + \alpha \frac{P}{2} \sum_{k=1}^m c_k \Lambda_f^2(\tilde{\tau} - \beta - d_k) \quad \text{for NMC.}
\end{aligned}$$

To avoid

ambiguities or multiple zero-crossovers, the following condition can be imposed:

$$\begin{cases} \mathbf{c}^T \mathbf{h}(\tilde{\tau}) > 0, & 0 < \tilde{\tau} \leq \bar{\tau} \\ \mathbf{c}^T \mathbf{h}(\tilde{\tau}) < 0, & 0 > \tilde{\tau} \geq -\bar{\tau} \end{cases} \tag{2.19}$$

where $\bar{\tau}$ is positive scalar. Further, to account for multipath-induced bias, the following condition can be used

$$\begin{cases} \mathbf{c}^T (\mathbf{h}(\tilde{\tau}) + \mathbf{g}(\tilde{\tau}, \alpha, \beta, \theta)) > 0, & \underline{\tau} < \tilde{\tau} \leq \bar{\tau} \\ \mathbf{c}^T (\mathbf{h}(\tilde{\tau}) + \mathbf{g}(\tilde{\tau}, \alpha, \beta, \theta)) < 0, & -\underline{\tau} > \tilde{\tau} \geq -\bar{\tau} \end{cases} \quad (2.20)$$

for all permissible multipath parameters with $\underline{\tau}$ being a bound on allowable multipath-induced error. To further shape the admissible discriminator function, some additional bounds can be imposed as illustrated in Figure 2.2 in which the shaded region depicts the forbidden region of the discriminator function. For example, to ensure that the gain is sufficiently high, the following condition can be used

$$|\mathbf{c}^T \mathbf{h}(\tilde{\tau})| > D_{up} \text{ or } |\mathbf{c}^T (\mathbf{h}(\tilde{\tau}) + \mathbf{g}(\tilde{\tau}, \alpha, \beta, \theta))| > D_{up}, \text{ for } \tau_{s1} \leq |\tilde{\tau}| \leq \tau_{s2} \quad (2.21)$$

In the above, τ_{s1} and τ_{s2} stands for the region of interest and D_{up} is a lower bound of the multipath-free or multipath-affected discriminator function. Further, the tail of the discriminator function is restricted when $\tilde{\tau}$ is large leading to

$$|\mathbf{c}^T \mathbf{h}(\tilde{\tau})| < D_{dn} \text{ or } |\mathbf{c}^T (\mathbf{h}(\tilde{\tau}) + \mathbf{g}(\tilde{\tau}, \alpha, \beta, \theta))| < D_{dn}, \text{ for } \tau_{l1} \leq |\tilde{\tau}| \leq \tau_{l2} \quad (2.22)$$

Here, the discriminator function is bounded in magnitude from above by D_{dn} for $\tilde{\tau}$ in the region between τ_{l1} and τ_{l2} . It is indeed possible to include a set of constraints of the form (2.21) and (2.22) to bound the discriminator function.

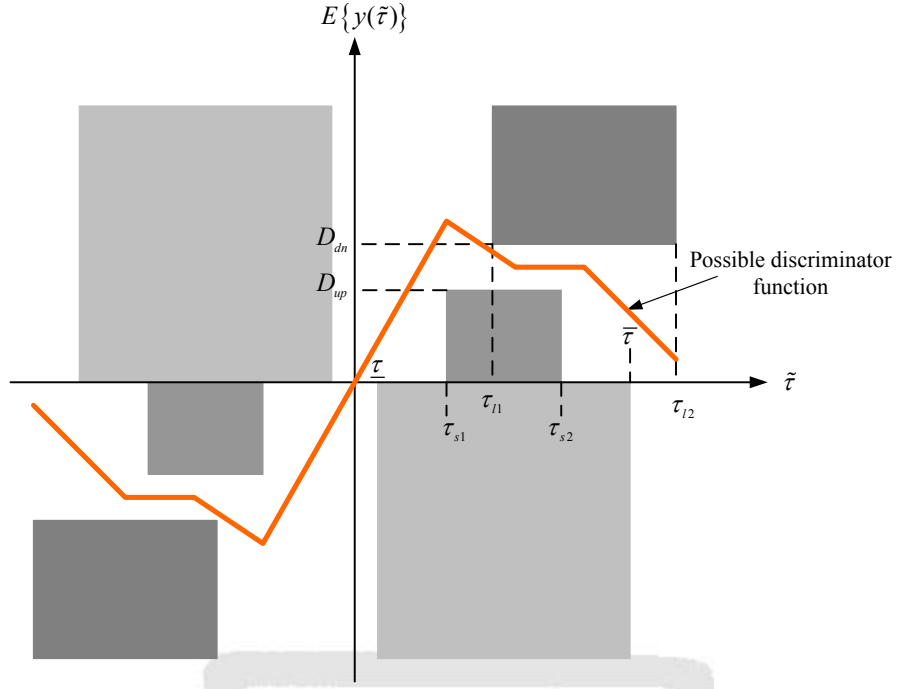


Figure 2.2 Shape of the discriminator function

As the code tracking loop is a feedback system that is affected by noise, the variance of the code tracking error is of concern. The characteristic of the discriminator function can be approximated as

$$y = K_0 \tilde{\tau} + n_y \quad (2.23)$$

where K_0 is $\sqrt{\frac{P}{2}} \sum_{k=1}^m c_k \frac{d\Lambda_f(\tilde{\tau} - d_k)}{d\tilde{\tau}} \bigg|_{\tilde{\tau}=0} + \sqrt{\frac{\alpha P}{2}} \sum_{k=1}^m c_k \frac{d\Lambda_f(\tilde{\tau} - \beta - d_k)}{d\tilde{\tau}} \bigg|_{\tilde{\tau}=0}$ for MC and

$$P \sum_{k=1}^m c_k \Lambda_f(d_k) \frac{d\Lambda_f(\tilde{\tau} - d_k)}{d\tilde{\tau}} \bigg|_{\tilde{\tau}=0} + \sqrt{\alpha P} \sum_{k=1}^m c_k \left(\Lambda_f(d_k) \frac{d\Lambda_f(\tilde{\tau} - \beta - d_k)}{d\tilde{\tau}} \bigg|_{\tilde{\tau}=0} + \Lambda_f(\beta + d_k) \frac{d\Lambda_f(\tilde{\tau} - d_k)}{d\tilde{\tau}} \bigg|_{\tilde{\tau}=0} \right) \cos \theta \quad \text{for NMC,}$$

$$+ \alpha P \sum_{k=1}^m c_k \Lambda_f(\beta + d_k) \frac{d\Lambda_f(\tilde{\tau} - \beta - d_k)}{d\tilde{\tau}} \bigg|_{\tilde{\tau}=0}$$

n_y is the equivalent noise, which is a zero mean process with the following variance

$$E\{n_y^2\} = \mathbf{c}^T \mathbf{Q} \mathbf{c} \quad (2.24)$$

where \mathbf{Q} is the $m \times m$ matrix whose (k, l) -th entry is

$$q_{kl,MC} = \frac{N_0}{4T_I} \int_{-\infty}^{\infty} S(f) |H(f)|^2 e^{-j2\pi f(d_k - d_l)} df$$

$$q_{kl,NMC} = 2P \left[\begin{aligned} &\Lambda_f(d_k) \Lambda_f(d_l) + \\ &\sqrt{\alpha} \left(\Lambda_f(d_k) \Lambda_f(d_l - \beta) + \Lambda_f(d_k - \beta) \Lambda_f(d_l) \right) \cos(\theta) + \\ &\alpha \Lambda_f(d_k - \beta) \Lambda_f(d_l - \beta) \end{aligned} \right] \left(\frac{N_0}{4T_I} \int_{-\infty}^{\infty} S(f) |H(f)|^2 e^{-j2\pi f(d_k - d_l)} df \right) + 4 \left(\frac{N_0}{4T_I} \int_{-\infty}^{\infty} S(f) |H(f)|^2 e^{-j2\pi f(d_k - d_l)} df \right)^2$$

The discrete-time linearized code tracking loop is depicted in Figure 2.3 in which the loop filter is represented in terms of $F(z)$ and the numerically controlled oscillator (NCO) is $\frac{1}{z-1}$ with z being the shift operator. The code tracking error variance can then be expressed as

$$E\{\tilde{\tau}^2\} = \frac{\|T(z)\|_2^2}{K_0^2} E\{n_y^2\} \quad (2.25)$$

where $\|T(z)\|_2$ is the 2-norm of $T(z)$ and $T(z)$ is the closed-loop transfer function which is given by $T(z) = \frac{K_0 F(z)}{z-1} \left(1 + \frac{K_0 F(z)}{z-1} \right)^{-1}$. In practice, as the gain K_0 is typically required to be greater than a given bound \bar{K} , i.e., $K_0 \geq \bar{K}$, an upper bound on the code tracking error variance is imposed:

$$E\{\tilde{\tau}^2\} \leq \frac{\|T(z)\|_2^2}{\bar{K}^2} \mathbf{c}^T \mathbf{Q} \mathbf{c} \quad (2.26)$$

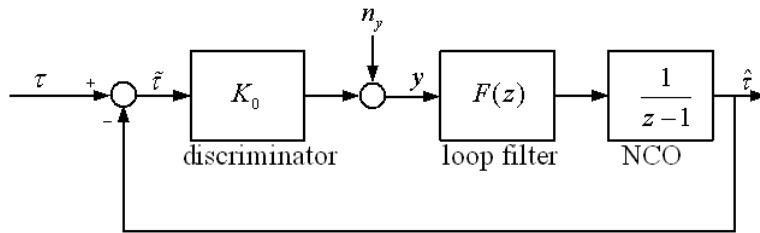


Figure 2.3 Block diagram of the linearized code tracking loop

Note that the requirements on unbiasedness, sensitivity, and shape are linear functions of the coefficient vector \mathbf{c} while the variance constraint is a quadratic function of \mathbf{c} . The design of the optimal coefficient vector can thus be formulated as the solving of a quadratic programming problem, that is, determination of \mathbf{c} such that a quadratic function, i.e, (2.26), is minimized while satisfying a set of linear equalities (2.13), (2.15) and linear inequalities (2.19)-(2.22).

2.2 Design Results

In this section, the proposed design method is applied to determine the coefficient vector of an NMC discriminator. As a Galileo receiver is to be realized, the BOC(1,1) modulation is considered and the power spectral density in (2.6) is used with a chip rate of 1.023 MHz. Ten correlators are used to construct the multi-correlator architecture. The delays of local code \mathbf{d} are set as:

$$\mathbf{d} = \left[-\frac{12}{8} \quad -\frac{7}{8} \quad -\frac{4}{8} \quad -\frac{2}{8} \quad -\frac{1}{8} \quad \frac{1}{8} \quad \frac{2}{8} \quad \frac{4}{8} \quad \frac{7}{8} \quad \frac{12}{8} \right]^T \quad (2.27)$$

in the unit of T_c . The resolution of the correlator position, $\frac{1}{8}T_c$, is due to hardware limitation in realizing the code generator and implementing the tracking loop. The difference between adjacent correlator positions is selected in accordance with the Fibonacci sequence. Recall that the tracking error due to noise and multipath depends heavily on the spacing between the narrow correlators. On the other hand, it is also desired that given a finite number of correlators, a wide operating range can be achieved by placing the correlator positions. The heuristic use of the Fibonacci sequence attempts to balance these two conflicting requirements in placing the correlator positions. It is noted that the vector \mathbf{d} can also be designed through an optimization process; see [39].

The process of GNSS signal reception undergoes several stages including acquisition, confirm, pull-in, and lock (or track). The closed-loop code tracking loop in Figure 2.3 operates during the pull-in and lock stages. At the pull-in stage, some apparent code error remains and the objective is to reduce the code tracking error to within certain threshold as fast as possible while avoiding locking into ambiguous side-peaks. In contrast it is desired to reduce the code tracking error as much as possible at the lock stage. In other words, the pull-in stage emphasizes on the transient response of the code tracking loop is subject to initial code tracking error, while the lock stage mainly addresses the steady-state response due to noise and multipath. Consequently, different loop filter and weighting coefficients can be used at the two stages to seek an overall performance enhancement in transient and steady-state responses. In the following, two NMC discriminators are designed in which one is used at the pull-in stage and the other is used at the lock stage. An adaptive decision rule is then developed to blend the two designs into one receiver architecture.

2.2.1 Discriminator Design I

The first code discriminator design is to minimize the tracking error variance as in (2.26) while treating the unbiased-ness (2.11), sensitivity (2.15), multipath (2.20), and shaping (2.21) requirements as constraints. Specific parameters that are used in the optimization process are as follows: $K_0 = 4P$. In Formula (2.21), $\tau_{l1} = 0.5T_c$, $\tau_{l2} = 1.5T_c$, $D_{dn} = 0.4$ at positive delays, and $D_{dn} = 0.2$ at negative delays by considering one multipath with parameters $0 < \alpha \leq 0.25$, $0 < \beta \leq 0.5T_c$, and $0 \leq \theta \leq 2\pi$. Formula (2.20) is exploited for in-phase and out-of-phase multipath. For in-phase case, $\theta = 0$, $\alpha = 0.25$, $\beta = 0.3T_c$, $\underline{\tau} = 0.08T_c$, and $\bar{\tau} = 0.2T_c$. For out-of-phase case, $\theta = \pi$, $\alpha = 0.25$, $\beta = 0.2T_c$, $\underline{\tau} = 0.08T_c$, and $\bar{\tau} = 0.2T_c$. After solving the quadratic program, the following coefficient vector is obtained

$$\mathbf{c} = \begin{bmatrix} 0.0021 \\ -0.0095 \\ 0.0193 \\ -0.1649 \\ 0.6954 \\ -0.6879 \\ 0.1733 \\ -0.0432 \\ 0.0050 \\ -0.0011 \end{bmatrix} \quad (2.28)$$

Figure 2.4 shows the NMC discriminator as a function of $\tilde{\tau}$. The NELP discriminators with spacing 1/4 chip and 1/2 chip are also provided for comparison. Due to the use of (2.22), the tail of the NMC discriminator is less pronounced than those of NELP discriminators. The code tracking error and multipath-induced error are further analyzed. Figure 2.5 shows the code tracking error as a function of C/N_0 . Note that the sensitivity of the NMC discriminator is close to that of the NELP discriminator with spacing 1/2 chip. The tracking error of the NMC discriminator, however, is close to that of the NELP discriminator with spacing 1/4 chip. The multipath envelope of Figure 2.6 shows that the NMC discriminator behaves similar to the NELP scheme with spacing 1/4 chip. In this analysis, a multipath which has a power that is half of the direct signal is considered. The figure of merit for multipath can also be assessed by using the running average multipath error (RAME) [34,45], which is expressed as

$$A(\tau) = \frac{1}{\tau} \int_0^\tau \frac{|\max(E_{mp}(x)) - \min(E_{mp}(x))|}{2} dx \quad (2.29)$$

where $E_{mp}(x)$ is the multipath envelope. As depicted in Figure 2.7, the proposed NMC discriminator appears to be superior to the NELP discriminator in accounting for multipath delays. In short, the sensitivity of the NMC discriminator is purposely limited to be close to an NELP discriminator with a wide spacing. Yet, through the optimization process, the

NMC discriminator renders steady-state error variance and multipath-induced error that are close to an NELP scheme with a narrow spacing. It goes without saying that if a higher sensitivity is specified as a design constraint, the performance can be further improved.

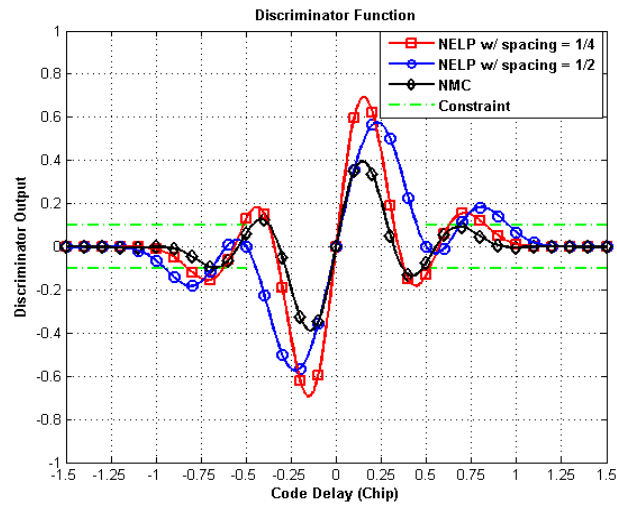


Figure 2.4 Discriminator functions at the lock stage

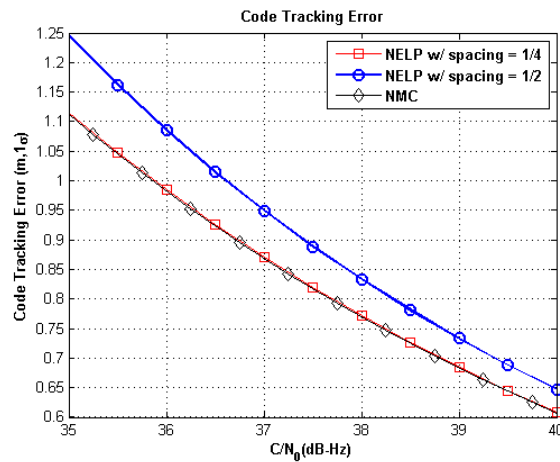


Figure 2.5 Comparison of code tracking error variances

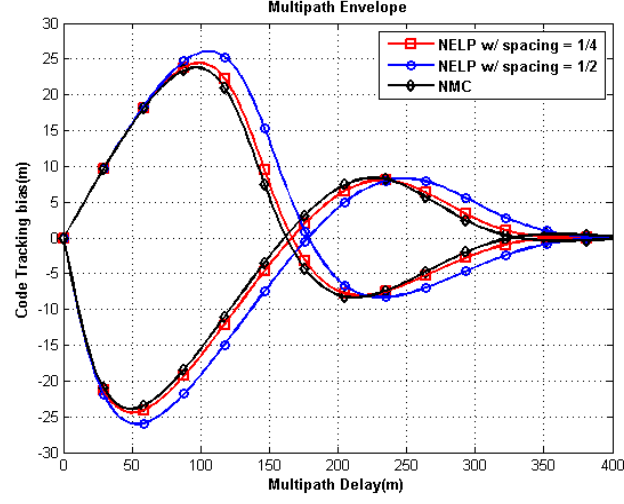


Figure 2.6 Comparison of multipath-induced code tracking errors

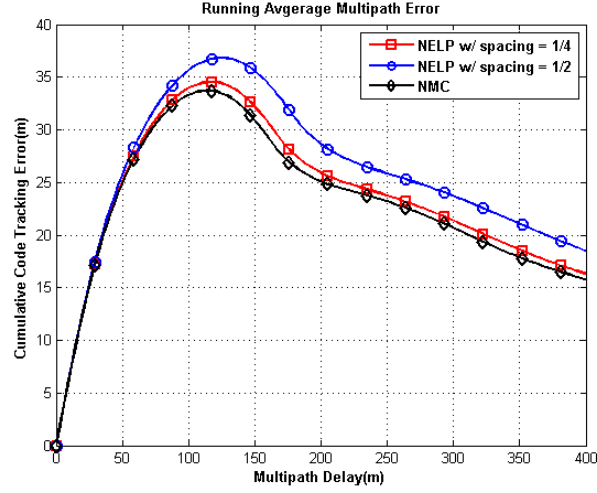


Figure 2.7 Comparison of running average multipath errors

2.2.2 Discriminator Design II

The second discriminator is designed with an aim to enhance the transient response and avoid ambiguities. The conditions for unbiased-ness, high sensitivity, ambiguity-free, and shaped discriminator are imposed as linear constraints on the coefficient vector. The following design parameters are selected: $\mathbf{g}_0^T \mathbf{c} = 8$, $\bar{\tau} = 1.5T_c$, $D_{up} = 0.5$, $\tau_{s1} = 0.05T_c$, $\tau_{s2} = 0.7T_c$, $D_{dn} = 0.25$, $\tau_{l1} = 0.8T_c$, and $\tau_{l2} = 1.5T_c$. As the code tracking error variance is

not of primary concern during the pull-in stage, the following objective function is considered

$$\int_{-\tau_0}^{\tau_0} |\mathbf{c}^T \mathbf{h}(\tilde{\tau}) - K_0 \tilde{\tau}|^2 d\tilde{\tau} \quad (2.30)$$

where τ_0 is a positive scalar. By minimizing (2.30), the discriminator function between $-\tau_0$ and τ_0 can be made as close to a linear curve as possible. By solving the quadratic program, the optimal coefficient vector is

$$\mathbf{c} = \begin{bmatrix} 0.0349 \\ -0.1506 \\ 1.5225 \\ 0.0755 \\ 1.5361 \\ -1.5361 \\ -0.0755 \\ -1.5225 \\ 0.1506 \\ -0.0349 \end{bmatrix} \quad (2.31)$$

Figure 2.8 shows the resulting discriminator function. To assess the transient response, a nonlinear simulation is used as the discriminator is subject to nonlinear behavior in the pull-in stage. The nonlinear simulation diagram is depicted in Figure 2.9 in which u_f is the filtered control signal that adjusts the NCO. In the simulation, a first-order loop filter is assumed and the noise bandwidth is set as 1 Hz. Figure 2.10 depicts the transient responses of the three designs when the initial delay error is -1 chip. The NMC discriminator results in a faster response than the NELP discriminators. The NELP discriminator with 1/4-chip-spacing, in this case, attempts to track a side peak, which is biased at 0.568-chip. As the code tracking loop is of second order, there are two state variables and the behavior can be characterized in terms of the phase plane. The boundaries of region of convergence (ROC) in the phase plane of different designs are further depicted in Figure 2.11. When the

initial state is outside the ROC, the state will eventually diverge, oscillate, or reach a false equilibrium. When the NELP discriminator is used, a larger spacing results in a slightly larger ROC. The ROC of the code tracking loop with the NMC discriminator is significantly larger than those of NELP discriminators. The performance of the NMC discriminator in transient response and ROC can be attributed to the use of multiple correlators so that the nonlinear characteristic of the discriminator function is made monotonic and the design of the discriminator in which only one zero-crossover is allowed. Recall from Figure 2.10 that there exist side peaks when the NELP discriminator with 1/4-chip spacing is used. When such an NELP discriminator is used, the sidepeaks at ± 0.568 -chips constitute two additional equilibrium points in the phase plane as depicted in Figure 2.11. Depending on the initial condition, three different regions of convergence with different equilibrium points are obtained. In contrast, the code tracking loop with the NMC discriminator is not subject to multiple equilibrium points.

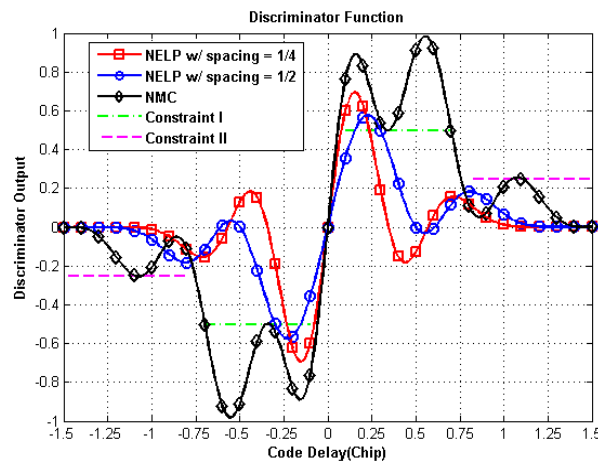


Figure 2.8 Discriminator functions at the pull-in stage

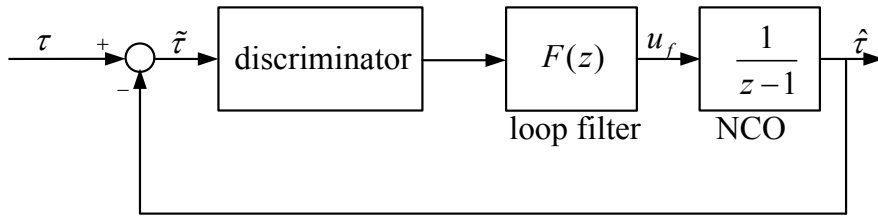


Figure 2.9 Nonlinear simulation diagram

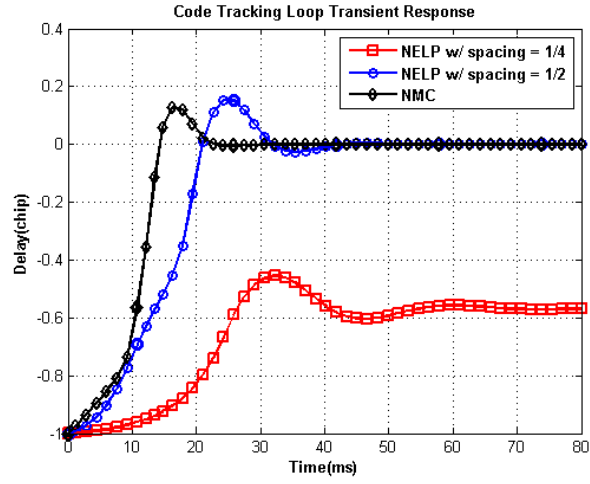


Figure 2.10 Comparison of transient responses

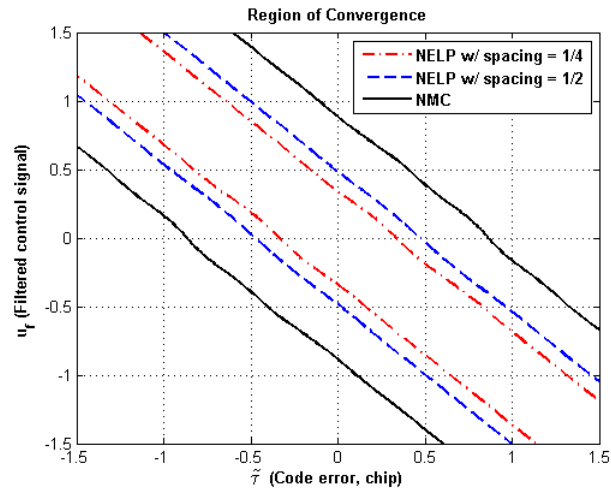


Figure 2.11 Comparison of regions of convergence

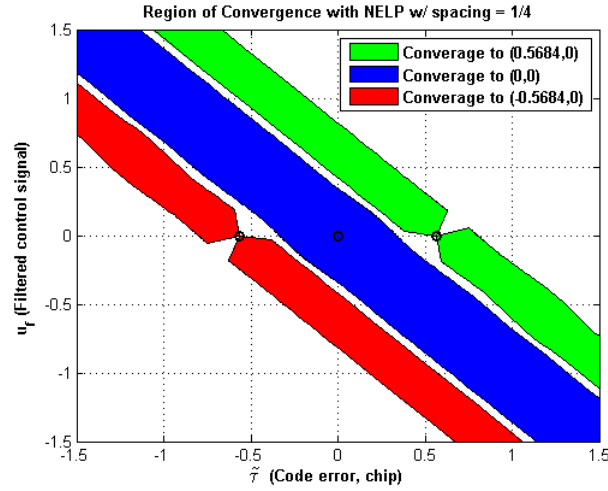


Figure 2.12 ROC with NELP discriminator and 1/4 spacing

2.3 Adaptive Switching Logic

The two discriminators discussed previously are designed with specific goals in mind and do not have to operate at the same time. During the pull-in stage, the initial error is often more significant and the objective is to ensure a successful and rapid convergence. In contrast, the lock stage is to address the error variance in the presence of noise and multipath. Therefore, the two discriminators are employed at different stages to ensure an overall performance enhancement. As the coefficient vector \mathbf{c} of either (2.28) or (2.31) is realized through software in the receiver, the switching from one coefficient vector to another can be realized rather easily.

The switching logic that adaptively engages different discriminators can further be examined by analyzing the behavior of the code tracking loop. Note that the ROC of the Discriminator I is smaller than that of Discriminator II. When switching from the latter to the former, one can then engage the switching when the state is within the ROC of the former to ensure convergence. This logic can be augmented to typical switching logics including the verification of tracking error and magnitude of the prompt correlator output in controlling the receiver operation.

2.4 Implementation and Test

A DSP/FPGA board together with an RF front-end is configured as a Galileo receiver prototype as in Figure 2.13 [13,46] to assess the effectiveness of the proposed adaptive NMC discriminator. The received signal at antenna is down-converted, filtered, and digitized in the front-end. The bandwidth of the front-end filter is about 4 MHz. At a sampling rate of 8.184×10^6 samples per second, the digitized I-phase and Q-phase baseband data are sent to the DSP/FPGA board for processing. More precisely, the interface (I/F) from the front-end de-serializes the input data into I and Q data which are then mixed with local carrier and code replicas. The mixed results are integrated and dumped in the multi-correlator output buffer. At every code period which is 4 msec for Galileo E1-B signal, the outputs of correlators are transferred to the DSP. In the DSP, discriminators and loop filters of the code tracking loop and carrier tracking loop are realized. The code slew rate and carrier frequency are then generated and used for the control of the NCOs in code/carrier generators. As far as the adaptive NMC discriminator is concerned, the multi-correlators are realized in the FPGA and the discriminator and loop filter are implemented in terms of DSP software.

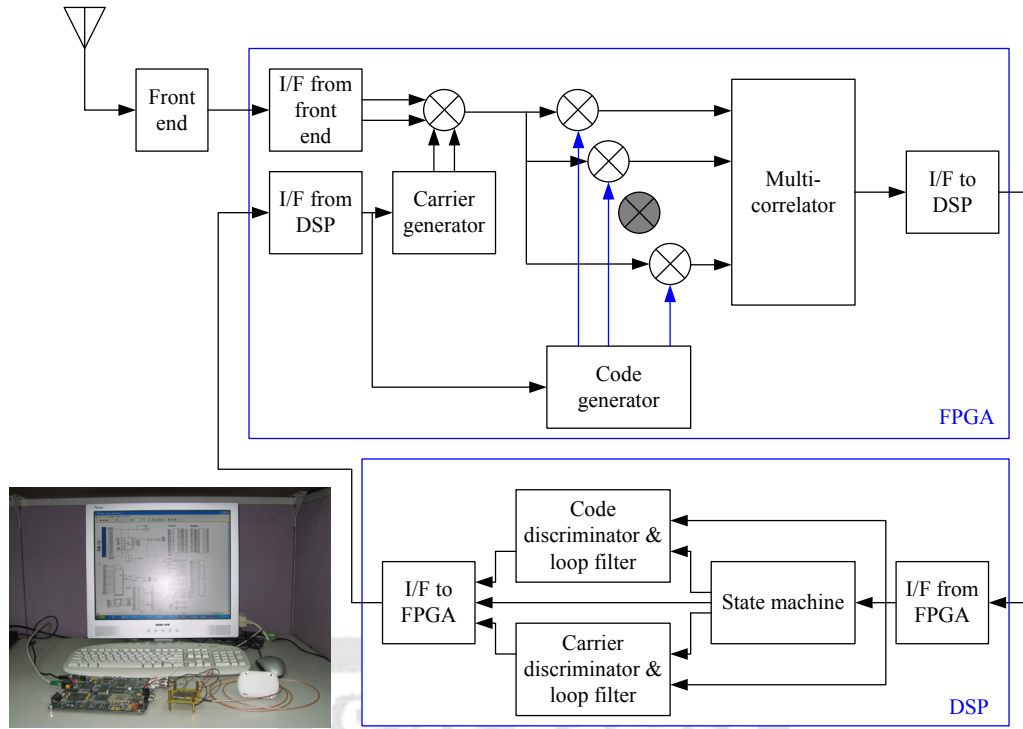


Figure 2.13 DSP/FPGA based Galileo receiver prototype

Figure 2.14 depicts the flow chart of the state machine in the receiver operation. In the *Acquire* state, code phase is slewed and carrier frequency is updated to search for the incoming signal. As long as the magnitude of any correlators is greater than a pre-determined threshold, the state machine moves to the *Confirm* state. Once the signal is detected and confirmed, the state machine enters the *Pull-in* state and the initial code phase and carrier frequency are set. In the *Pull-in* state, a carrier tracking loop and a code tracking loop are used to pull in the carrier frequency and code phase, respectively. The Discriminator II in (2.31) is applied at this stage. After a specific period of time, the adaptive switching logic is employed to detect whether the criteria on phase error, prompt correlator magnitude, and state within ROC are met. If affirmative, the state machine proceeds to the *Lock* state, in which code phase and carrier frequency are tracked steadily. The Discriminator I in (2.28) is adopted at this stage.

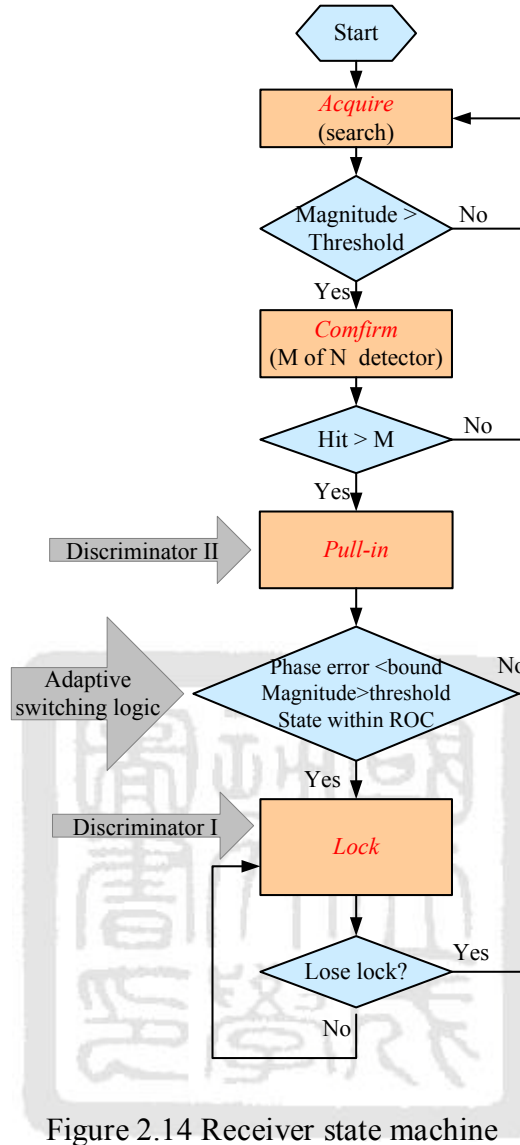


Figure 2.14 Receiver state machine

2.5 Test Results

An experiment was conducted for the reception of the GIOVE-A E1-B signal components to assess the adaptive NMC discriminator design. Figure 2.15 shows the I-Q plot of the received E1-B component, which is bi-phased modulated by navigation data. The relative 3σ value is about 0.28 and the navigation data can be easily demodulated. In order to assess the convergence performance, the collected data during the satellite pass were divided into segments. Each segment of data was then undergone the acquisition-tracking process. In the assessment, when the *Pull-in* stage was triggered, the

initial code phase was recorded and the operation moved through the *Pull-in* and *Lock* stages. When the code was eventually locked, a success track was marked. If, on the other hand, the operation failed to converge, a false track was reported. The statistics of success and false tracks are depicted in Figure 2.16 for the NELP and NMC discriminators. In the figure, the probability of false track is shown on the right-hand part and the distribution of the remaining success tracks is also depicted with the initial code error as the horizontal axis. It is evident that the region and probability of convergence of the proposed NMC is larger than that of the NELP.

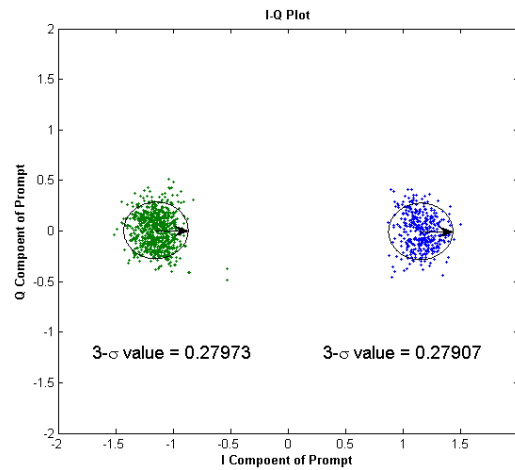


Figure 2.15 I-Q plot of received GIOVE-A E1-B components

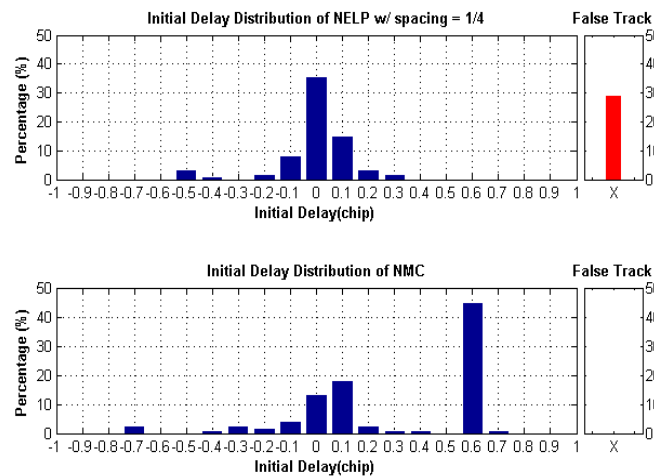


Figure 2.16 Analysis of initial delay and convergence performance

2.6 Summary

An adaptive code discriminator design approach is proposed, implemented, and tested. The design is formulated as an optimization problem and two discriminators are designed. The first one is to reduce the code tracking error for lock stage operation, while the second one is to enhance transient response and enlarge the ROC to yield a better performance during the pull-in stage. The operation of the two discriminators is governed by an adaptive decision logic. The result of ROC analysis is shown to be beneficial in ensuring that the switching from pull-in to track stages can be accomplished smoothly. The adaptive NMC discriminator is further realized and tested to show the improvement in tracking error and transient response.



Chapter 3

PC-Based Real-time Software Receiver

The function of a PC-based GNSS receiver can typically be divided into three portions: RF processing, baseband processing, and navigation processing. The RF processing is typically realized using an RF front-end module which is responsible for signal amplification, noise filtering, down conversion, automatic gain control, and analog-to-digital conversion. The front-end module renders digital IF data at a rate of several mega Hz even though the incoming GNSS signal is in L band (1~2 GHz). The digital IF data, typically 1 or 2 bits, are then processed by a baseband processor whose roles include correlation, signal acquisition, code and carrier tracking, and data demodulation. The demodulated data and resulting pseudorange measurements are then processed by a navigation processor to yield position, velocity, and timing information. Indeed, most existing consumer GPS receivers involve two chips for signal reception: one is the RF front-end chip for signal conditioning and conversion and the other is the baseband chip for the handling of correlation/tracking and navigation tasks. A software receiver differs from a conventional receiver in the sense that the functions of the baseband chip including correlation/tracking and navigation tasks are realized through software, leading to a more flexible design with potential savings in cost and power. Figure 3.1 depicts the hardware architecture of a typical PC-Based GNSS software receiver.

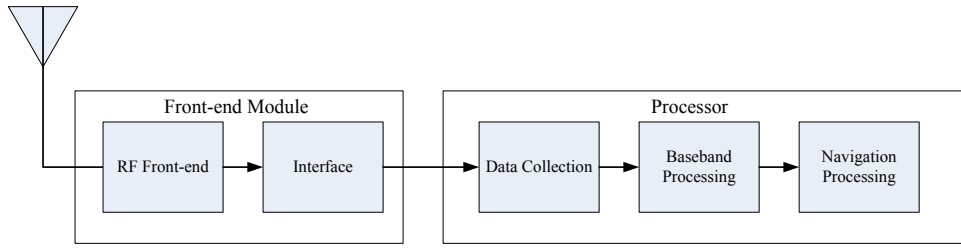


Figure 3.1 Hardware architecture of a PC-based GNSS software receiver

3.1 Software Architecture

The RF down-converter convert GNSS signal to low IF signal. The IF signal is then sampled and digitized into 2-bit or 1-bit data stream. Typically, the data buffer stores the digitized IF data temporarily before sending to the host computer through a USB interface for tracking and navigation processing. Then, the CPU in PC is used to realize the rest of the receiver functions. The host computer fetches the IF signal routinely and performs the acquisition, tracking, and navigation tasks. The software flow chart of the software GPS receiver is shown in Figure 3.2. The software starts from initializing parameters. The parameters include the filter coefficients, channel assignments, variables pertaining to the setting of tracking channel, initial estimates of position and time, almanac, and ephemeris, if possible. After the initialization, the software enters an indefinite loop in which a block of data from the front end is obtained and processed. Based on setting of driver, a block of data with length of several msec is loaded into the memory of PC. The signal data are then correlated with local replica in the software correlator. The correlation results which are generated every millisecond are fed to another part of receiver, the tracking-to-navbit procedure. And, the measurements are made every 100 msec to being fed to measure-to-position procedure. The descriptions of components in the flow chart are listed in the Table 3.1.

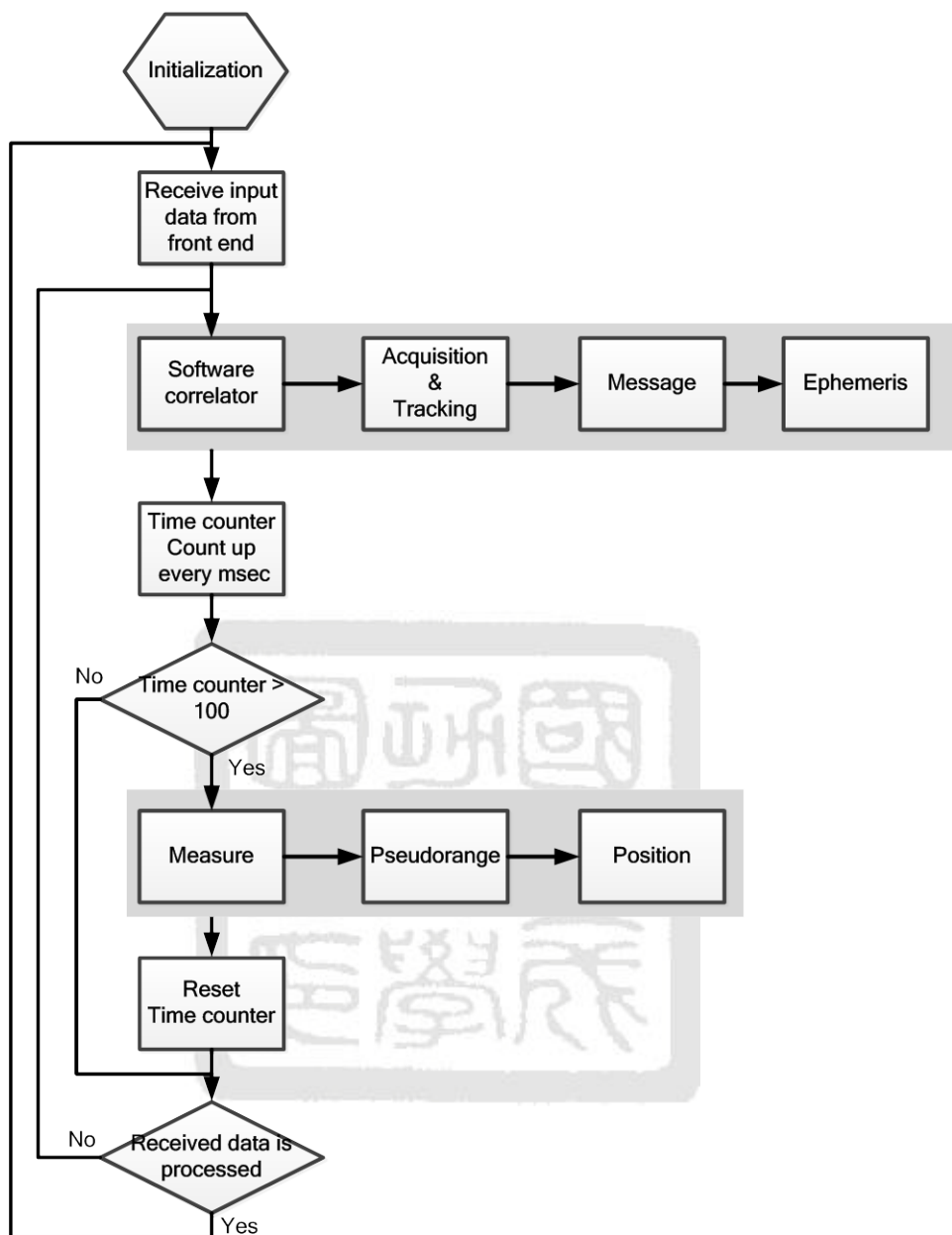


Figure 3.2 Flow chart of the GNSS software receiver

Table 3.1 Description of component in software receiver

Component	Description	Frequency
Initialization	Set the initial value of (1) Filter coefficients (2) Channel assignments (3) Acquisition/Tracking threshold (4) Initial time and position (5) Almanac and ephemeris	One time
Software correlator	Correlation between IF signal and local replica. Implement by bit-wise parallel algorithm [3] and SIMD instruction.	1 msec
Acquisition/Tracking	Acquisition: search for initial code phase and Doppler frequency. Tracking : fine track the code phase and Doppler frequency	1 msec
Message	Find the preamble of message, perform parity check and demodulate bits in the navigation data.	20 msec
Ephemeris	Get ephemeris of satellite from navigation data.	6 sec
Measure	Measure the code phase, nav. bit time, and carrier phase.	100 msec
Pseudorange	Use measurements to calculate the pseudorange with carrier-smoothing	100 msec
Position	Use pseudorange and ephemeris to calculate position of receiver.	100 msec

3.1.1 Software Correlator

In order to realize the correlator function in software, tables of code and carrier are provided and stored in the disk before execution [3]. The tables are made for 1 millisecond long and its number of phase depends on the number of samples in 1 millisecond. The code table is duplicated for each PRN number. The carrier table is made to span ± 10 kHz Doppler frequency with spacing of 10 Hz. For reducing the size of carrier table, each

replica of frequency component starts from zero phase. Figure 3.3 shows the structures of code table and carrier table where the size of 1 sample could be 1 bit for low-resolution or several bytes for high-resolution. The code phase and carrier frequency could be set before the correlation operation between the incoming data block and the data from the memory.

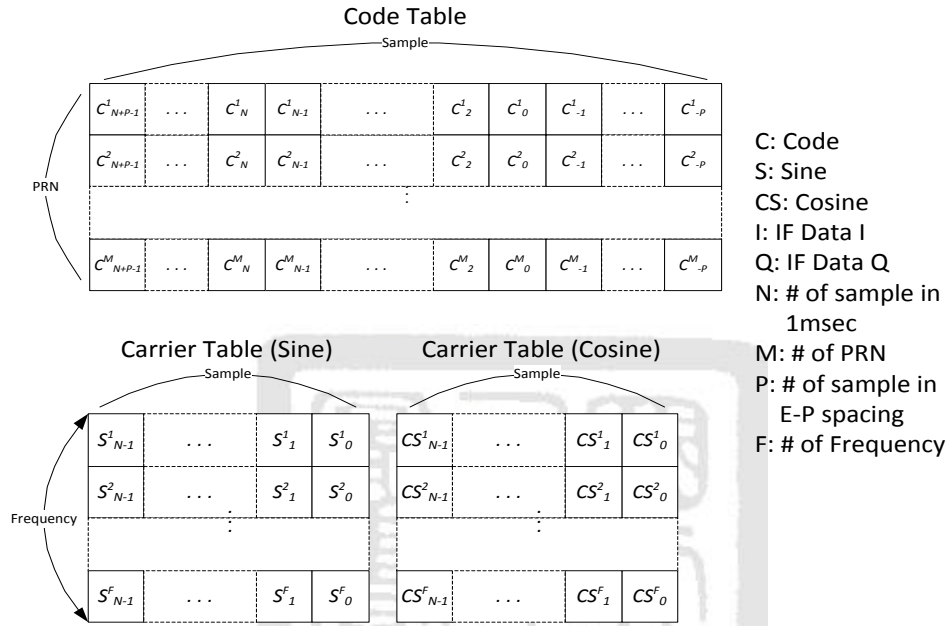


Figure 3.3 Structures of code table and carrier table

In order to reach real-time capability, the software correlator is often implemented by parallel approaches. The bit-wise parallel algorithm represents each sample of signal by bit and performs bit-wise operations to accomplish the correlation [3]. Additionally, Single Instruction Multiple Data (SIMD) instruction set is a group of instructions which can perform the same operations on multiple data. For current x86 processors, one of SIMD instruction set is Streaming SIMD Extensions (SSE) [39]. The SSE instructions can allocate the multiple samples in one 128-bit-wide XMM register and perform parallel operations in one instruction cycle. In the software receiver, the bit-wise parallel algorithm is implemented using the SIMD instructions. 128 samples can be processed at a

time using the 128-bit XMM registers. After the bit-wise operation, the results are stored within the bits of the register and an accumulation (counting the number of bit set to one) is required. Conventionally, this operation is performed by addressing a look-up table in the memory. However, with a CPU with SSE 4.2 can use POPCNT instruction [39] which can count the number of bits set to one for a 32-bit register in a 32-bit operating system. An example of counting the number of correlation results being six using SSE instructions shows in the Figure 3.4.

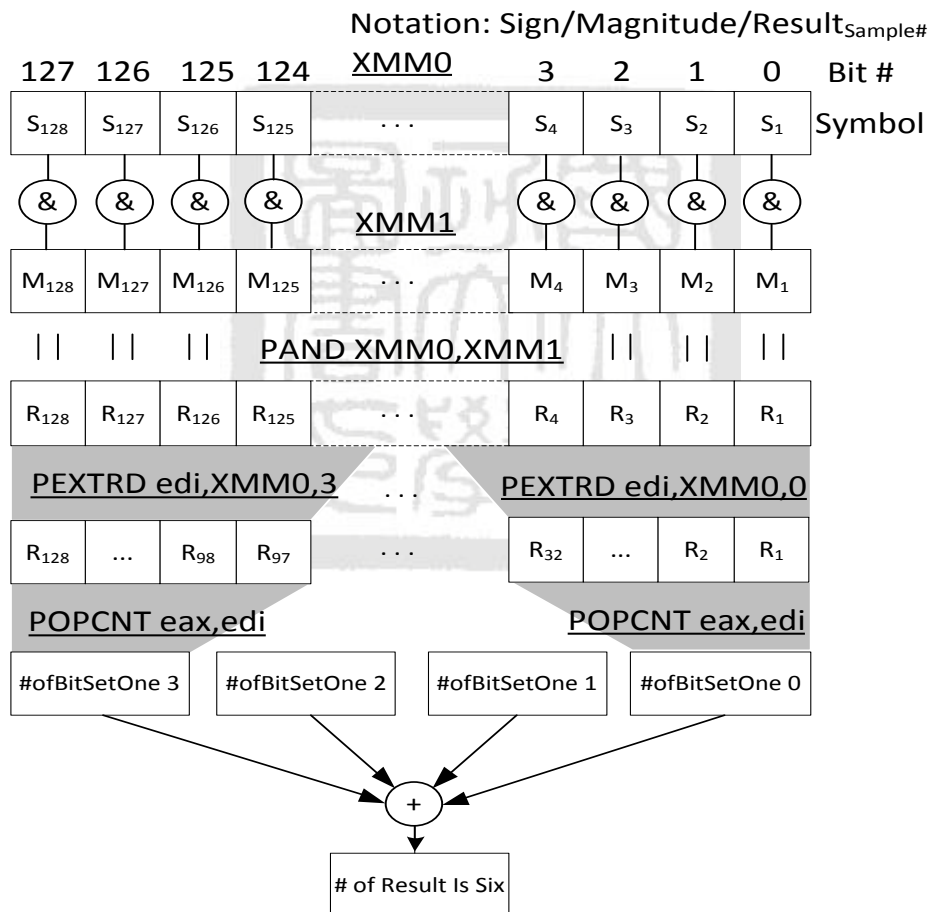


Figure 3.4 Example of counting the number of correlation results using SSE instructions

3.1.2 Tracking Procedure

The tracking procedure is controlled by a state machine as shown in Figure 3.5. In the un-modified software GPS receiver, the tracking operation is decomposed into the following four states: *acquire*, *confirm*, *pull-in*, and *lock*. The *acquire state* performs a search in the dimensions of Doppler frequency and code phase to detect the presence of GPS signals. As long as the magnitude of any correlators is greater than a pre-determined threshold, the operation then enters the *confirm state*. In the *confirm state*, an M of N detector is used to confirm whether the signal is present. Once the signal is detected, the operation then enters the *pull-in state*. In the *pull-in state*, a FLL (frequency locked loop) plus PLL (phase locked loop) carrier tracking loop and a DLL (delay locked loop) code tracking loop are used to pull in the carrier frequency and code phase, respectively. After going through the pull-in process for a specific time period, the software will check whether the variance of the phase error is under the accepted level and the magnitude of the correlation output of the prompt arm is greater than the threshold. If affirmative, the tracking operation enters the *lock state*. In the *lock state*, the PLL carrier locked loop has a narrower noise bandwidth and the DLL integrates for 20 ms to enhance the signal quality. Besides, for every 100 ms, it will check whether the signal is lost. If it is, the tracking procedure returns to the *acquire state*.

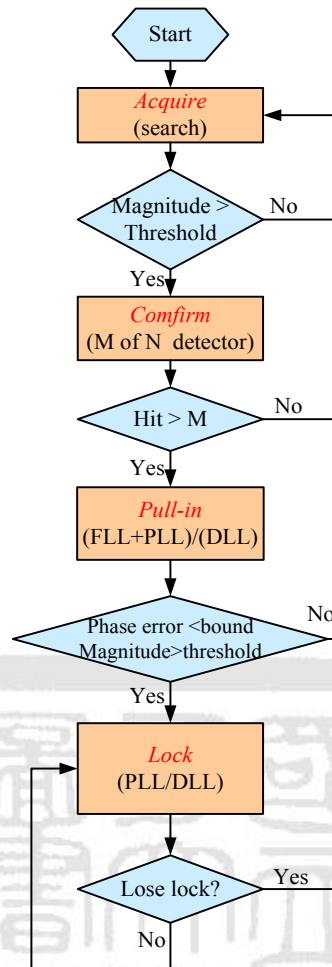


Figure 3.5 State machine of the tracking procedure

In the software GPS receiver, the carrier tracking loop is different from the traditional one because the carrier table is made for limited numbers of frequency and has zero phase in the beginning of table. The overall carrier tracking loop in the software receiver is depicted in Figure 3.6. The incoming data is mixed with the local carrier phase replica every ms to generate in-phase (I) and quadrature (Q) correlation outputs. A rotation operation is then employed to account for zero-phase tabular data. The amount of rotation depends on the frequency as well as integration interval. The rotated I and Q components are then passed to a phase discriminator and the phase error is obtained. The phase error is then filtered and used to adjust the frequency of the local oscillator in the look-up table.

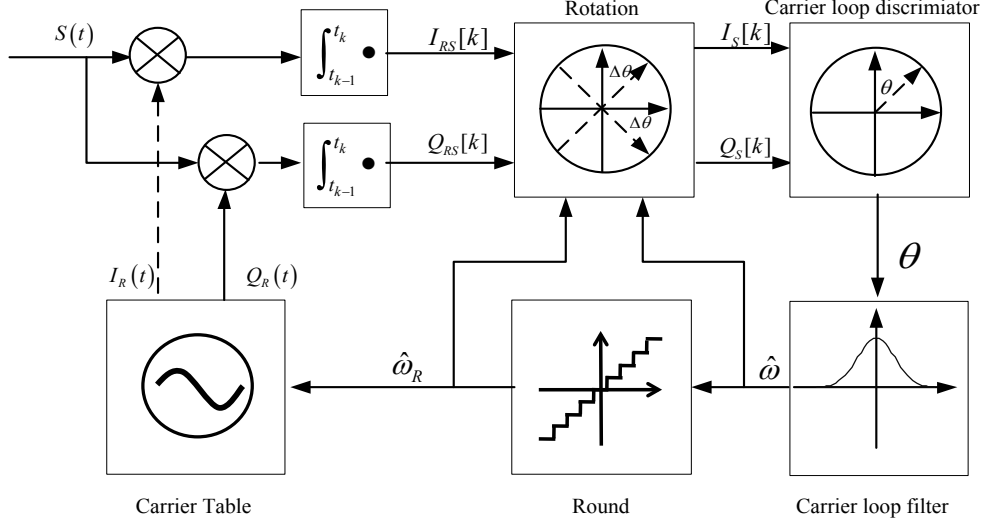


Figure 3.6 Carrier tracking loop in the software receiver

To explain the need of the rotation operation, consider, for simplicity, the following incoming carrier data

$$S(t) = \sqrt{2P} \cos(\omega t + \phi) \quad (3.1)$$

where P is the power, ω is the carrier frequency, and ϕ is the phase. When zero-phase carrier replica are used, the replica of the I and Q arms are given, respectively, by

$$I_R(t) = \sin(\hat{\omega}_R(t - t_{k-1})) \quad (3.2)$$

and

$$Q_R(t) = \cos(\hat{\omega}_R(t - t_{k-1})) \quad (3.3)$$

where $\hat{\omega}_R$ is the frequency after rounding the estimated frequency $\hat{\omega}$ to the nearest frequency grid of the carrier table. After mixing and integrating from t_{k-1} to t_k , the I and Q components are given, respectively, by

$$\begin{aligned} I_{RS}[k] &= \int_{t_{k-1}}^{t_k} \sqrt{2P} \cos(\omega t + \phi) \sin(\hat{\omega}_R(t - t_{k-1})) dt \\ &= \frac{\sqrt{2PT_i}}{2} \text{sinc}\left(\frac{(\hat{\omega}_R - \omega)T_i}{2}\right) \sin(\theta_{RS}[k]) \end{aligned} \quad (3.4)$$

and

$$\begin{aligned}
Q_{RS}[k] &= \int_{t_{k-1}}^{t_k} \sqrt{2P} \cos(\omega t + \phi) \cos(\hat{\omega}_R(t - t_{k-1})) dt \\
&= \frac{\sqrt{2PT_i}}{2} \text{sinc}\left(\frac{(\hat{\omega}_R - \omega)T_i}{2}\right) \cos(\theta_{RS}[k])
\end{aligned} \tag{3.5}$$

where $T_i = t_k - t_{k-1}$ is the integration time and sinc is the function given by $\text{sinc}(x) = \sin x / x$.

The phase angle $\theta_{RS}[k]$ is given by $\theta_{RS}[k] = \frac{(\hat{\omega}_R - \omega)T_i}{2} - \omega t_{k-1} - \phi$. Recall that when the local replica of continuous phase based on the estimated frequency, the I and Q replica are $I(t) = \sin(\hat{\omega}t)$ and $Q(t) = \cos(\hat{\omega}t)$, respectively. Under this condition, the I and Q components of the correlator output at t_i are represented as

$$\begin{aligned}
I_S[k] &= \int_{t_{k-1}}^{t_k} \sqrt{2P} \cos(\omega t + \phi) \sin(\hat{\omega}t) dt \\
&= \frac{\sqrt{2PT_i}}{2} \text{sinc}\left(\frac{(\hat{\omega} - \omega)T_i}{2}\right) \sin(\theta_S[k])
\end{aligned} \tag{3.6}$$

and

$$\begin{aligned}
Q_S[k] &= \int_{t_{k-1}}^{t_k} \sqrt{2P} \cos(\omega t + \phi) \cos(\hat{\omega}t) dt \\
&= \frac{\sqrt{2PT_i}}{2} \text{sinc}\left(\frac{(\hat{\omega} - \omega)T_i}{2}\right) \cos(\theta_S[k])
\end{aligned} \tag{3.7}$$

where the phase $\theta_S[k]$ is $\theta_S[k] = \frac{(\hat{\omega} - \omega)T_i}{2} + (\hat{\omega} - \omega)t_{k-1} - \phi$. Note the phase difference between $\theta_{RS}[k]$ and $\theta_S[k]$ is

$$\Delta\theta = \theta_S[k] - \theta_{RS}[k] = \frac{(\hat{\omega} - \omega_R)T_i}{2} + \hat{\omega}t_{k-1} \tag{3.8}$$

The phase difference is affected by the rounding error in the frequency table and an offset due to frequency estimate. When the frequency estimate is available, the phase error can be compensated through the rotation operation; for otherwise, the phase tracking loop may be subject to phase discrepancy between two integration intervals. When the phase rotator is employed, the traditional carrier tracking loop can be used to track the frequency and phase of the incoming signal.

3.2 Data Intermittency Issue

For PC-based GNSS software receiver, the digital IF data is first buffered in the memory of front-end. Upon request from the host computer, the buffered data are sent to the host computer for acquisition, tracking, and navigation processing. Since the host computer of a mobile device may be busy with some high-priority applications or the bus may be occupied by other data transfer flows, the data for GNSS applications are subject to burst-type errors in which a block of data is lost as depicted in Figure 3.7. Such a data intermittency phenomenon may affect the tracking procedure discussed in the previous section. Indeed, when a block of data is lost, the tracking operation cannot continue as the peak of the correlator output is no longer maintained. As a result, whenever a data intermittency occur, the tracking operation returns to the acquire state and the navigation capability is greatly compromised. In GPS signal tracking, the frequency, carrier phase, and code phase need to be locked. Typically, data intermittency may not lead to a significant variation on the frequency. However, both carrier phase and code phase are affected. In addition, as the data are decoded based on continuous tracking, the navigation data may then be subject to bit error. With the availability of A-GNSS techniques, the data decoding issue can be resolved. Therefore, the challenge lies in the seamless tracking of carrier phase and code phase.

Note that when a block of data is lost, all tracking channels which are tuned to lock different satellites are equally affected. The inter-channel information can then be explored for the detection of data intermittency. Once the phenomenon is detected, it is desired to resume the tracking as soon as possible without going through the states of acquire, confirm, and pull-in. To this end, the intra-channel information reveals a way for the correction in carrier phase and code phase. By exploring the inter-channel and intra-channel information, the lock state can be recovered rapidly.

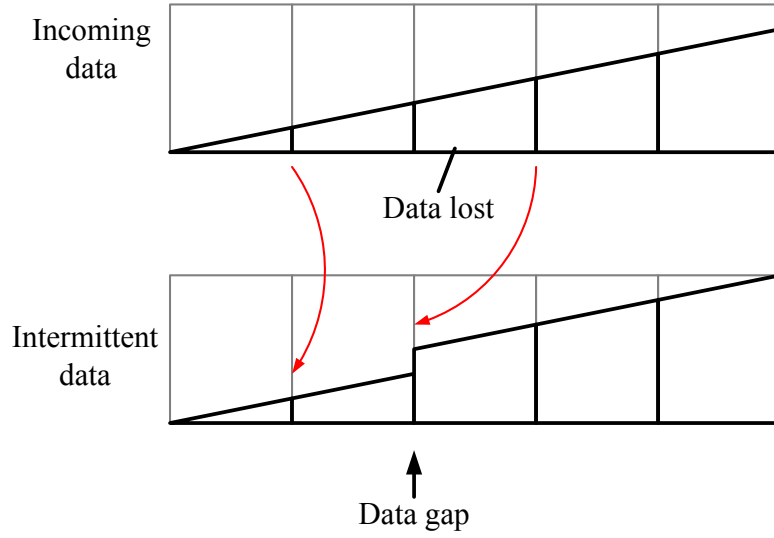


Figure 3.7 Data intermittency issue of RF front-end

3.3 Solving the Data Intermittency

The proposed tracking procedure that is capable of accommodating data intermittency is depicted in Figure 3.8. In the tracking procedure, a new *recover state* is included to recover the tracking performance in the presence of data gaps. The operations at the *recover state* include a detection scheme and a re-lock mechanism. When each channel has been visited, the tracking status of all channels that are in *lock state* can be assessed. When there are multiple channels that are subject to performance degradation such as drop of correlator output (of the prompt correlator) and increase of code/phase tracking error simultaneously, it is likely that the incoming signal is subject to either a signal blockage or data intermittency problem. The operation of the tracking software then enters the *recover state*. To verify whether the problem is due to data intermittency, a correlator operation is performed by placing the local code replica at d chips advance from the previous code phase. The advance in chips d is related to the size of data block and can be computed as follows. Suppose that the size of the data block is N bits and the sampling frequency is F_s Hz (or bit/sec), then each data block corresponds to a period of $T_g = \frac{N}{F_s}$ in sec. Let T_c

be the GNSS code chip length, f_L be the carrier frequency, and \hat{f}_D be the Doppler frequency of the channel, then the advance of chips of the code phase for re-lock is given by

$$d = \left(1 + \frac{\hat{f}_D}{f_L}\right) \frac{T_g}{T_c} \quad (3.9)$$

When a correlation operation is performed at the advanced position and a peak correlation value is obtained, it is likely that the tracking disruption is due to data intermittency. By cross-checking the results of different channels, the confidence can be increased and the tracking operation can then be returned to the *lock state* without having to endure the lengthy reacquisition process. It is also noted that the phase tracking loop may need to be modified accordingly. To this end, the phase change $\Delta\theta$ due to data intermittency can be estimated as

$$\Delta\theta = (f_{IF} + \hat{f}_D) T_g \quad (3.10)$$

where f_{IF} is the IF frequency. When this phase change is used to update the phase estimate, the phase tracking can be resumed more rapidly. Once data intermittency is detected, the epoch counter for the maintenance of local time is adjusted for the easiness of data demodulation and position determination.

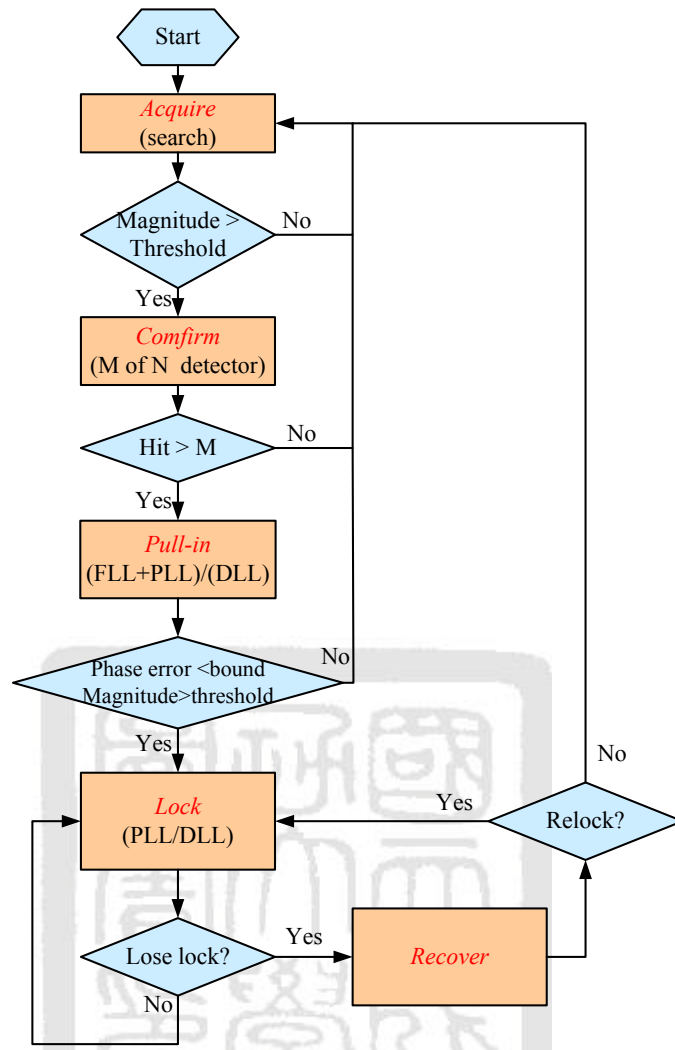


Figure 3.8 State flow of the proposed tracking procedure

3.4 Experimental Results

To assess the proposed approach in accounting for data intermittency, the software GPS receiver described in Section 3.1 is modified. The experimental results are reported in this section. The software receiver can indeed be used for real-time tracking and processing. However, for comparison, a set of data is saved and tracked by the unmodified and the proposed software.

When the unmodified software is used to process a set of data being collected, the resulting code phase estimate and correlator output value are depicted in Figure 3.9 with

respect to the GPS satellite of PRN 3. The code is tracked originally in the figure. At time instant about 32 msec, the data transmission is subject to intermittency and the tracking procedure returns to the *acquire*, *confirm*, and *pull-in* states before the satellite can be locked at around 158 msec. During the transient period, no pseudorange measurements can be obtained.

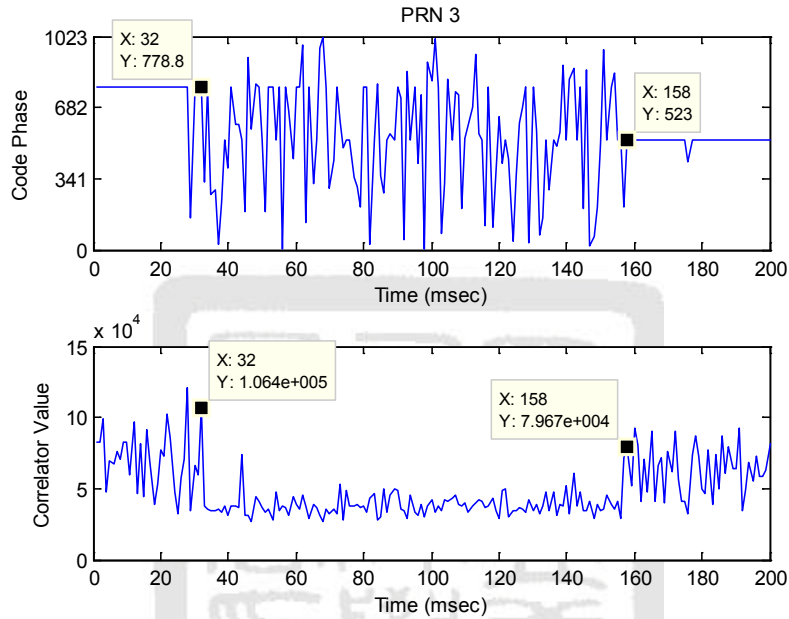


Figure 3.9 Effects of data intermittency on tracking

The situation is further elaborated in Figure 3.10 in which the code tracking results of four satellites with PRN 3, 13, 25, and 27, respectively, are depicted in (a) and (b) when the unmodified software is employed. The code phase is subject to jump when the buffered data are not successfully transmitted to the host computer. Although the software is capable of reacquire and re-lock the code phase, the data demodulation and pseudorange measurement capabilities are jeopardized. The pseudorange measurements are depicted in Figure 3.10 (c) and (d). Due to data intermittency and transient behavior of the tracking loop, the measurements are subject to irregular gaps and discontinuities and, consequently, the positioning capability is limited. The local time are depicted in Figure 3.10 (e) and (f).

It is observed from Figure 3.10 (a) that the code phase jumps occur at the same time. By exploring the inter-channel and intra-channel properties, the proposed software is capable of seamlessly tracking the incoming signals. When the proposed method is employed, the code phases, pseudorange measurements, and local time are shown in Figure 3.10 (b), (d) and (f), respectively. The code phases in Figure 3.10 (b) are subject to jump as before. Yet, the pseudorange and local time measurements are compensated as the transient behavior is rapidly detected and recovered. The effect of data intermittency is thus accounted for and the positioning capability becomes continuous.



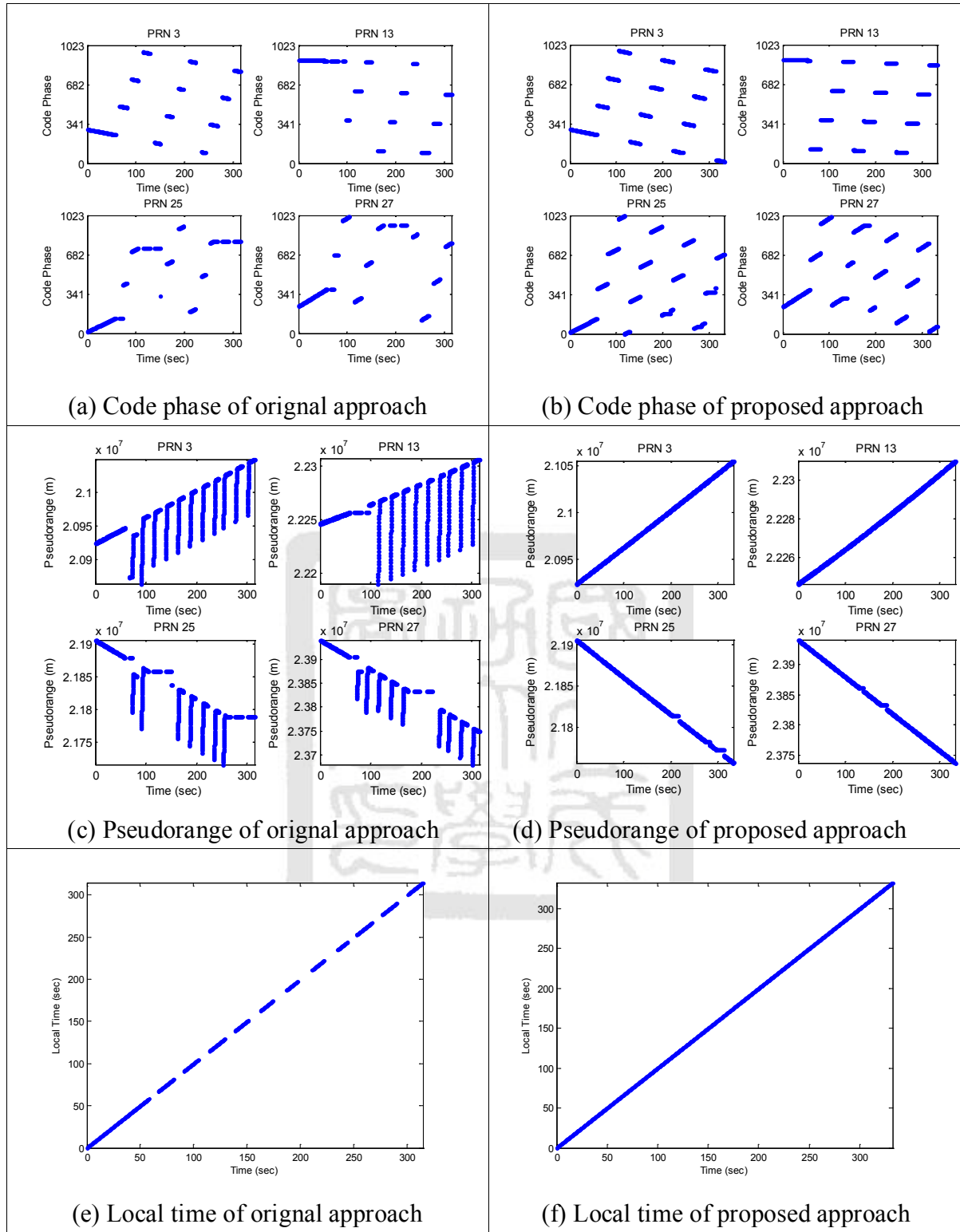


Figure 3.10 Comparisons of code phase, pseudorange and local time between original and proposed approach

3.5 Summary

This chapter demonstrates a real-time PC-Based GNSS software receiver. The hardware and software architecture are designed carefully to achieve real-time capability and are described in detail. In the implementation, parallel-programming techniques including bit-wise parallel algorithm and SIMD assembly are used to speed the computation of operation in software correlator. A specific carrier tracking loop for real-time software receiver with prior code table and carrier table is addressed in this chapter. Moreover, a method is developed to account for data intermittency in a GNSS software receiver. For other applications like mobile with a software GNSS capability, the issues of data intermittency cannot be neglected. As the impact is equally detected among different tracking channels, a consistency check can be invoked to detect the presence of data intermittency and the adjustments in code phase and carrier phase can be estimated based on known knowledge on data block size and Doppler frequency estimate. The proposed method is verified in a software GPS receiver and the capability of continuous navigation is demonstrated.

Chapter 4

Robust GNSS Signal Tracking against Scintillation Effects by a Particle Filter Based Software Receiver Approach

In a GNSS receiver, the carrier tracking loop is responsible for the synchronization of the local carrier and the incoming carrier in the presence of noise and perturbation in the atmosphere. Every 12 years, the sun enters solar maximum while ionosphere becomes irregular. The result which ionosphere interacts with radio wave produces deep amplitude fade and fast phase shift called as scintillation [66]. In the literatures, several structures have been adopted in the carrier tracking during scintillation including conventional phase-locked loop (PLL) [69,70] , frequency locked loop (FLL) assisted PLL [28,29] , and Kalman filter-based filter PLL [69]. However, in the severe scintillation condition, PLL may lose track because of deep amplitude fading coincide with great phase shift. Therefore, a more robust method is needed to cope with scintillation. The particle filter is a sequential Monte Carlo method which employs discrete samples associated with weights to represent concerned posterior density function. By the principle of Monte Carlo method, the particle filter has the capability to find an optimal solution for the nonlinear dynamics system and non-Gaussian noise. This section adopts particle filter in carrier tracking for accounting of scintillation. Because computational burden of particle filter is high, a software receiver approach is used to implement particle filter based carrier tracking.

4.1 Scintillation Effect

Ionosphere is deeply influenced by solar activity which is the reason why the day and night structure of ionosphere is different. Particularly, the sun has eleven years of solar maximum period while solar activity makes ionosphere irregular. Radio waves experience the irregular ionosphere that it leads to refraction and diffraction, are called as scintillation. The scintillation effect is a kind of multipath in the ionosphere and produces rapid phase shifts and amplitude perturbations to the GNSS signal. [62] has processed the scintillation data that are collected at Ascension Island during the last solar maximum year 2001. It is also shown in [62] that GNSS signal has deep fading which C/No drops exceeding 30 dB. Most GNSS receiver would lose tracking at the moment while C/No is lower than 25 dB-Hz. It is possible that the GNSS receiver will stop positioning for a few seconds. Therefore, it is important that the GNSS receiver has the ability of robust signal tracking against scintillation. For designing a GNSS receiver which still keep tracking during scintillation, one should acquire the scintillation data firstly. Without real scintillation data, [47] provides scintillation model by adding filtered white noise and scintillation intensity. In this model, one can simulate different scintillation data by selecting filter's bandwidth and scintillation index. Figure 4.1 illustrates results of modeling for a severe scintillation. The upper plot shows normalized signal power versus time. It is noted that amplitude of carrier quite randomly changes with time. Besides, there are many deep fading points that degrade more than 30 dB in power. A half-cycle phase change occurs at the same time with deep fading points as showed in the lower plot of Figure 4.1. For carrier tracking of GNSS signal, these points with deep fading and half-cycle phase change are hard to distinguish with navigation data transition in the weak-signal condition.

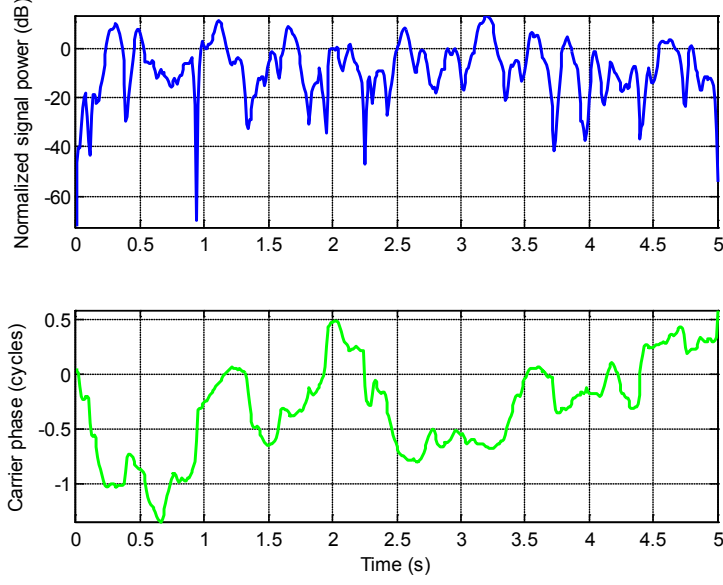


Figure 4.1 Example of scintillation effect on the GNSS signal

4.2 Introduction of Particle Filter

Particle filter is a sequential Monte Carlo method where employs discrete samples associated with weights to represent concerned posterior density function. For nonlinear system and non-Gaussian noise, particle filter provides a means to estimate optimal solution. In theory, the model of nonlinear systems with additive noise is given as follows [3].

$$x_k = f_{k-1}(x_{k-1}) + v_{k-1} \quad (4.1)$$

$$z_k = h_k(x_k) + w_k \quad (4.2)$$

where (4.1) is the state dynamics equation and (4.2) is the measurement equation. The particle filter takes N samples to approximate the posterior density of state x_k is given as

$$p(x_k | z_1, \dots, z_k) \approx \sum_{i=1}^N w_k^i \delta(x_k - x_k^i) \quad (4.3)$$

where w_k^i is the weight associated with x_k^i . It can be shown that as $N \rightarrow \infty$ the

approximation of (4.3) approaches to true posterior density. The procedure of particle filter is depicted as Figure 4.3 [3].

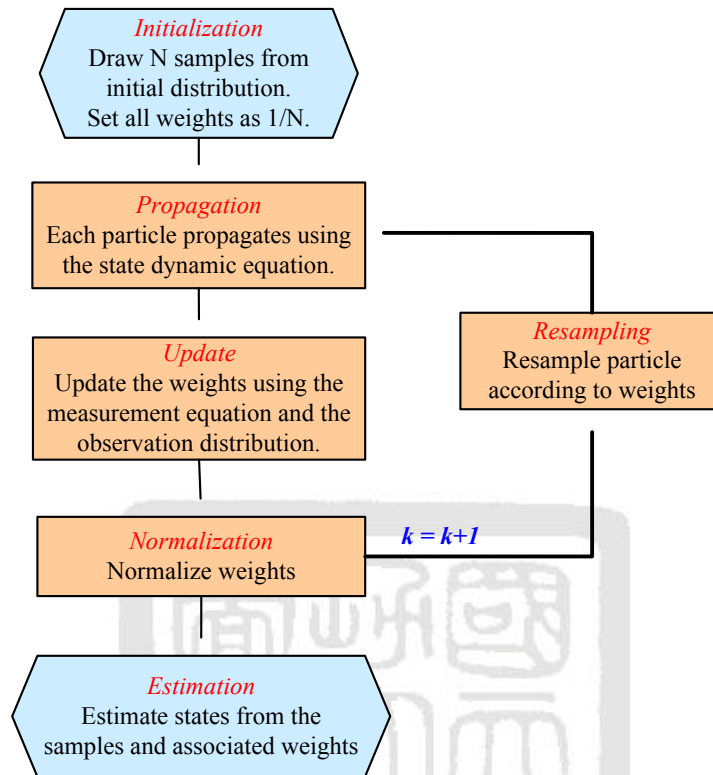


Figure 4.2 Procedure of particle filter.

At first, N samples are drawn from the initial importance density $q(x_0)$ which can be assumed as Gaussian distribution and its weights are uniformly distributed. In each discrete time, each particle propagates using the state dynamic equation to construct the prior importance density function $q(x_k|x_{k-1})$. Then, the weights are updated using the measurement equation and the observation distribution. Moreover, weights are needed to normalize to construct the posterior density function $p(x_k|z_1, \dots, z_k)$. One can estimate the state for discrete time k using samples and associated weights by taking the mean of x_k . In the beginning of next discrete time, resampling is needed to eliminate samples low weights and multiple samples with high weights. After resampling, all weights are reassigned as uniform distribution.

4.3 Using Particle Filter for Carrier Tracking in a Software Receiver

In order to overcome the nonlinear amplitude and phase variations due to scintillation, the particle filter which has the ability to estimate optimal solution for nonlinear system as well as non-Gaussian noise is employed for carrier tracking. The structure of carrier tracking employing particle filter is depicted in Figure 4.3. I and Q correlator outputs instead of phase and frequency error serve as the measurements for better estimating the amplitude. Three states of particle filter are amplitude, phase frequency, and of carrier. These three outputs of particle filter which are means of states and two measurements are the inputs to rotation operation. Besides, the frequency output of particle filter is then round to a nearest frequency that carrier table has.

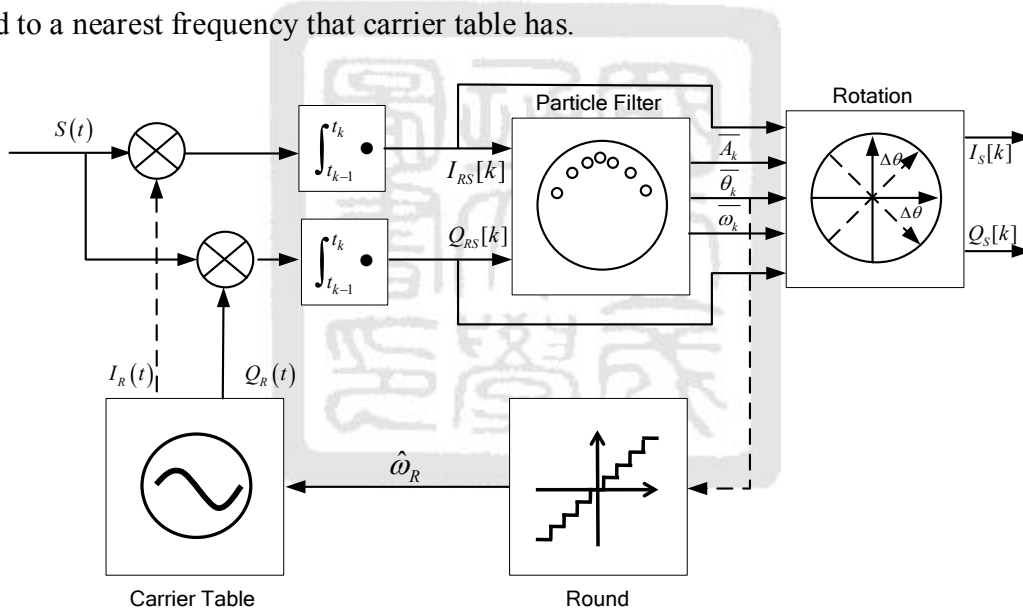


Figure 4.3 Carrier tracking employing particle filter in a software receiver.

According to the procedure of particle filter mentioned in previous section, the flowchart of particle filter for carrier tracking is depicted in Figure 4.4. Accounting for random change of amplitude and phase in scintillation, the Gauss-Markov model is employed in the state dynamic equation as follows [52]. The state is defined as $x_k = [A_k \quad \theta_k \quad \omega_k]^T$.

$$x_k = \begin{bmatrix} \beta & 0 & 0 \\ 0 & 1 & T \\ 0 & 0 & 1 \end{bmatrix} x_{k-1} + \begin{bmatrix} \varepsilon_A \\ \varepsilon_\theta \\ \varepsilon_\omega \end{bmatrix} \quad (4.4)$$

where β is the ratio between present and previous amplitude, ε_A is random noise of amplitude with distribution $N(0, \sigma_A^2)$, ε_θ is random noise of phase with distribution $N(0, \sigma_\theta^2)$, and ε_ω is random noise of frequency with distribution $N(0, \sigma_\omega^2)$. $\hat{\beta}$ can be estimated by least squares estimator in following equation:

$$\hat{\beta} = (A_{k-1}^T A_{k-1})^{-1} A_{k-1}^T A_k \quad (4.5)$$

where $A_k = [\overline{A_k} \dots \overline{A_{k-m+1}}]^T$. Determination of the number of taps depends on how dynamic the amplitude is and how smooth the estimation is. The measurement matrix is comprised of I and Q correlator outputs as follows:

$$h_k(x_k) = \begin{bmatrix} I(x_k) \\ Q(x_k) \end{bmatrix} = \begin{bmatrix} A_k \sin\left(\frac{(\hat{\omega}_R + \omega_k)T}{2} - \theta_k\right) \\ A_k \cos\left(\frac{(\hat{\omega}_R + \omega_k)T}{2} - \theta_k\right) \end{bmatrix} \quad (4.6)$$

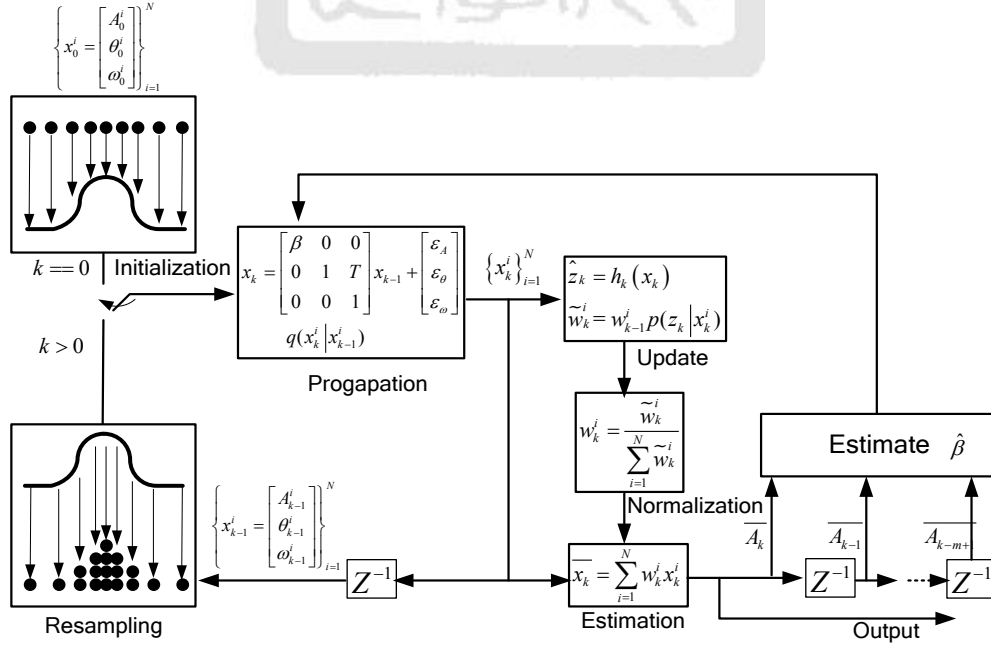


Figure 4.4 Schematic diagram of particle filter for carrier tracking

For carrier tracking, navigation data can be regarded as phase jump that degrades tracking performance, so a measure is needed to cope with navigation data. In the section, a navigation data detector is merged into particle filter by means of hypothesis testing. By adding navigation data term D_k into incoming signal, the measurement matrix is modified as

$$h_k(x_k) = \begin{bmatrix} I(x_k) \\ Q(x_k) \end{bmatrix} = \begin{bmatrix} A_k D_k \sin\left(\frac{(\hat{\omega}_R + \omega_k)T}{2} - \theta_k\right) \\ A_k D_k \cos\left(\frac{(\hat{\omega}_R + \omega_k)T}{2} - \theta_k\right) \end{bmatrix} \quad (4.7)$$

In the case that navigation data bit is positive, the measurement matrix denotes $h_k^+(x_k)$ where $D_k = 1$. Otherwise, if the navigation data bit is negative, the measurement matrix denotes $h_k^-(x_k)$ where $D_k = -1$. For update step of particle filter, one would use the measurement equation to update the weights which represent density function of states. The summation of all weights reflects how representative the density function is and provides a judgment whether the case is suitable. The hypothesis testing for navigation data detector is depicted in Figure 4.5. Two hypotheses which denote the sign of navigation data employ the same set of particles, but use different measurement equations to update weights. Then, sum all weights respectively and compare the results. The hypothesis with higher summation of weights is selected to output its individual weights.

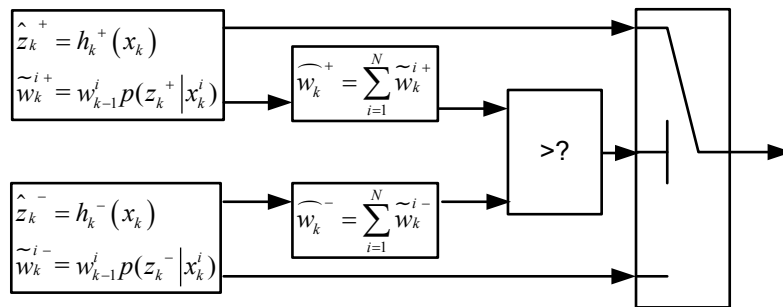


Figure 4.5 Hypothesis testing for navigation data detector

4.4 Simulation of Signal Generator and Receiver with Scintillation

In order to examine the proposed carrier tracking employing particle filter during scintillation, a simulating signal is generated and then fed into software receiver. The signal flow of simulation is depicted in Figure 4.6. There are four stages in this simulation from GNSS signal generator to receiver. In the GNSS signal generator stage, the baseband GNSS signal is generated by binary adder adding PN code and navigation data. Then, modulate baseband signal on an intermediate frequency carrier by BPSK modulator. In the atmosphere stage, the signal experiences scintillation effect where the amplitude and phase of signal has rapid change. In the channel stage, Gaussian white noise is added to signal. Finally, in the receiver stage, the resulted signal is processed in the GNSS software receiver in which carrier tracking employs PLL or particle filter.

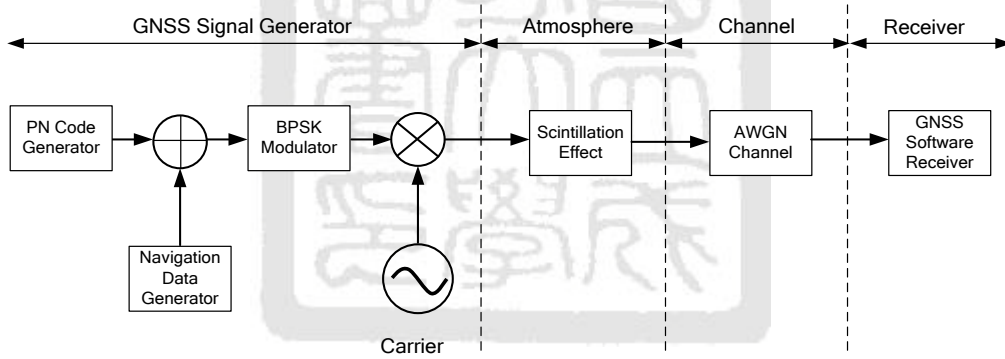


Figure 4.6 Signal flow of simulation for scintillation

For worse-case scenario, one could select lower S/N ratio and severe scintillation effect to examine the performance of receiver. In this section, a scenario is taken under the condition of S/N = -30 dB and severe scintillation. Figure 4.7 shows the simulation results of carrier tracking for comparing between PLL and particle filter. The first plot is the scintillation effect on the amplitude and phase of carrier. It is noted that there are two main deep amplitude fading points coincide with half-cycle phase change. The second, third, and fourth plots show tracking results of amplitude, carrier phase, and carrier frequency state respectively. In the second plot, estimated amplitude by particle filter follows the variation

of amplitude during scintillation; however, the PLL does not have amplitude estimation, only amplitude of correlator is shown. In the third plot, half-cycle phase estimation error in PLL occurs after encountering second deep fading point, but particle filter still can estimate phase accurately. In the fourth plot, estimated frequency of PLL is more sensitive to noise and also has a 100 Hz frequency error after second deep fading point, but the particle filter tracks the frequency robustly throughout simulation.

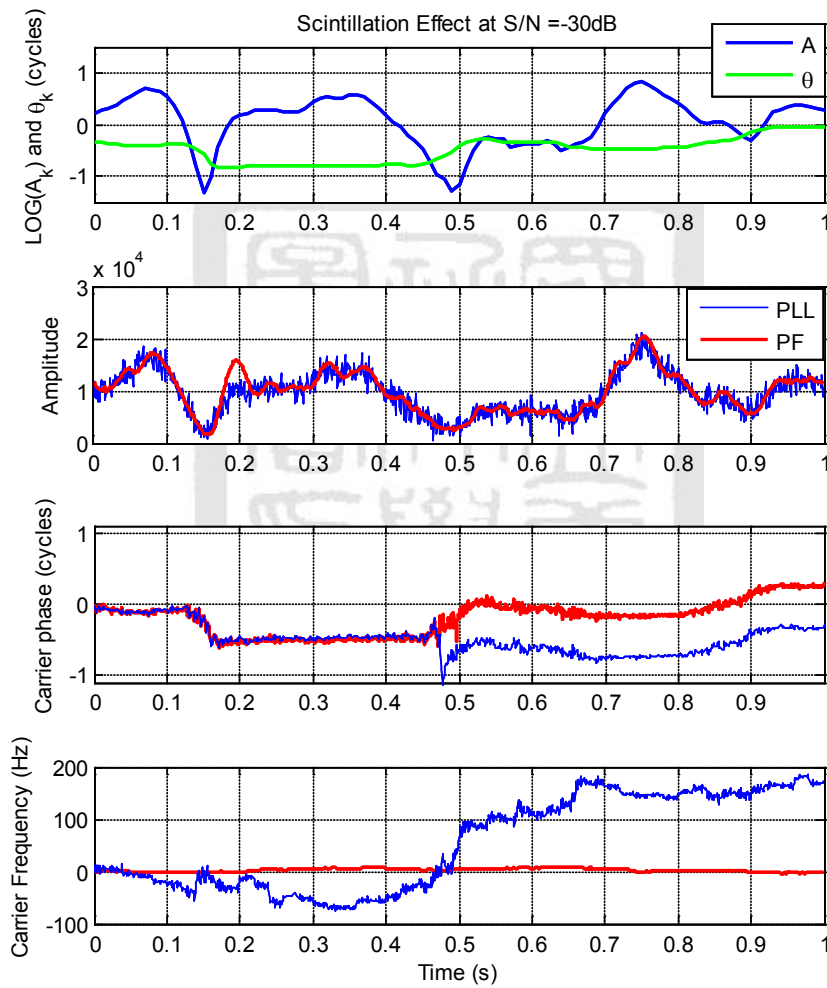


Figure 4.7 Simulation results of carrier tracking for comparison between particle filter and PLL under the condition of S/N = -30 dB and severe scintillation

For the parameter β of dynamic state equation (4.4), the number of taps of least squares estimator influences the performance of estimation. Figure 4.8 shows the original and estimated results of β . The first and second plots show the original value of β from scintillation and noise. The third plot shows estimated β by least squares estimator with different number of taps. It is noted that the using higher number of taps would get smoother the estimated results. And, estimating result by low number of taps is sensitive to noise. In the severe scintillation, fast variation of amplitude may cause estimator with high number of taps to fail because of slow response.

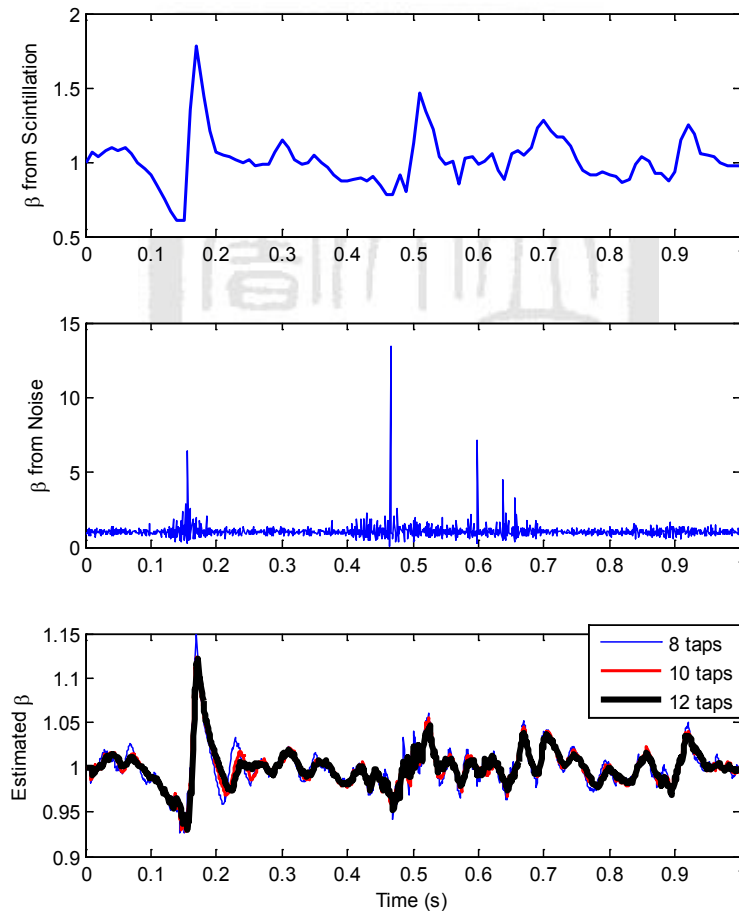


Figure 4.8 Original β from scintillation as well as noise compared with estimated β by least squares estimator with different number of taps.

For examining the proposed navigation data detector in the previous section, one configures a navigation data generator as square waveform with data rate 50 Hz like GPS. And, run the simulation in Figure 4.6 and the resulted receiver outputs are shown in the Figure 4.9. The upper plot shows the rotated I & Q correlator outputs, It is noted that the data is demodulated on the Q component even though the signal is degraded by scintillation and noise. The lower plot shows accumulated Q component for 20 milliseconds and represents the decision of navigation data where black and red circles stand for 1 and -1. Every navigation data is accurately demodulated by receiver.

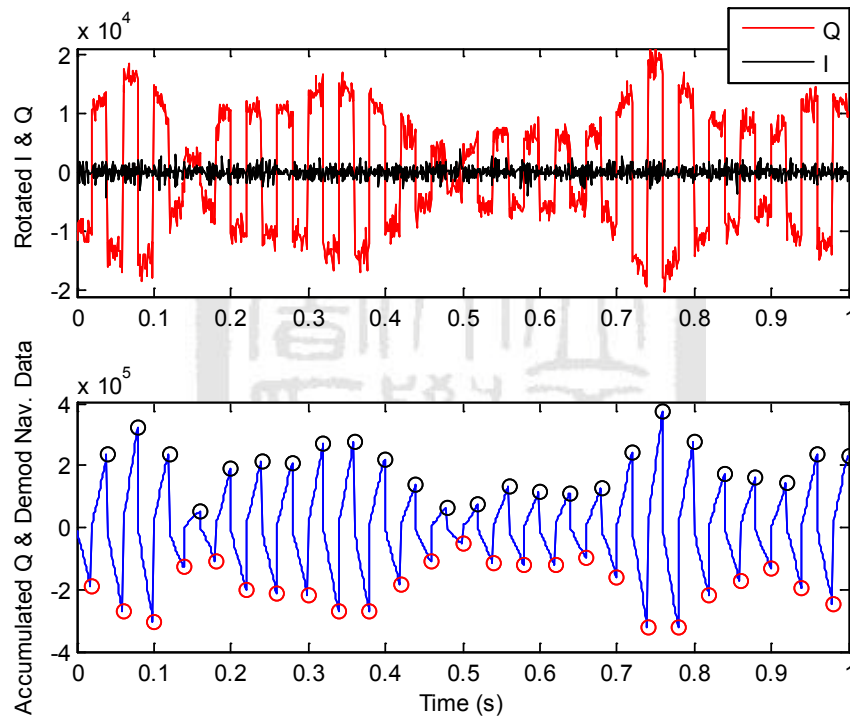


Figure 4.9 I & Q correlator outputs of receiver in the upper plot and accumulated Q correlator outputs and demodulation navigation data in the lower plot.

4.5 Summary

In this chapter, particle filter is employed to realize carrier tracking by a software receiver approach against scintillation effect. In the implementation, a Gauss-Markov model is used to formulate variation of amplitude and a navigation data detector is realized by merging into weight update stage of particle filter. For examining the proposed approach, a simulation from signal generator to receiver is constructed. In this simulation, a scintillation model is utilized to simulate scintillation effect on amplitude and phase of carrier and Gaussian white noise is also included in the channel. The GNSS software employs particle filter and PLL to realize carrier tracking. The simulation results show that the particle filter based is more robust than PLL during scintillation. Moreover, the estimation of β , the ratio between present and previous amplitude, and data demodulation results are shown to prove that the least squares estimator for β and navigation data detector are efficient.

Chapter 5

Dual-Frequency (L1/L5) GPS/WAAS Software Receiver

GPS is executing a modernization program to provide civil L5 signal. The spreading code on L5 signal has a ten times higher chipping rate than L1 C/A code. The signal decreases both tracking error and multipath-induced error. Future, using an iono-free combination of L1/L5 pseudoranges, positioning accuracy would improve under severe ionospheric conditions. Before May 6th 2011, there were two GPS satellites (PRN1/SVN49 and PRN25/SVN62) broadcasting the L5 signal. Additionally, the L5 signal payload will be included in all future GPS satellites. Wide Area Augmentation System (WAAS) geostationary earth orbit (GEO) satellites also provide user ranging information. Presently, all of WAAS GEO satellites (PRN133, PRN135, and PRN138) broadcast the L5 signal. In total, there were five GPS/WAAS satellites broadcasting L1 and L5 simultaneously. Hence, it was good practice to evaluate the positioning performance using L5 signal rather than only to assess signal tracking performance. The objective of this chapter is to implement a real-time software receiver capable of dual-frequency (L1/L5) processing for GPS/WAAS. In order to enable L5 positioning, one will provide the solutions for time offset problem. Moreover, use dual-frequency measurements to examine positioning methods including L1, L5, iono-free combination and analyze the resulting position.

5.1 Signal Specification and Status of GPS/WAAS L5 Signal

The GPS L5 signal is designed with two channels (I & Q) [58]. The chipping rate of the spreading code of both channels is 10.23 Mcps with 10230 code length. I channel is the data channel with 50 bit per second (bps). Forward error correction (FEC) code with rate $\frac{1}{2}$ convolution is encoded on I channel at 100 symbol per second (sps). It is further encoded by 10-bit Neuman-Hoffman (NH) code at 1 KHz. The Q channel is the data-free channel which is only encoded by 20-bit NH code at 1 KHz. The WAAS L5 signal has only one data channel with 250 bps which is encoded by the same FEC at 500 sps. It is further encoded by a 2-bit NH code (1,0) at 1000 sps [10]. Table 5.1 lists the signal specification of GPS/WAAS L5.

Table 5.1 Signal specification of GPS/WAAS L5

Signal Specification		GPS L5	WAAS L5
Spreading Code	Chipping rate	10.23 Mcps	10.23 Mcps
	Code length	10230	10230
Channel		I (data) Q (data-free)	I (data)
Neuman-Hoffman code		I (10-bit) Q (20-bit)	I (2-bit)
Forward error correction code		Rate $\frac{1}{2}$ convolution	Rate $\frac{1}{2}$ convolution
Data rate		50 bps on I	250 bps

Before May 6th 2011, the GPS PRN1/SVN49 was only broadcasting L5Q data-free channel and was set unhealthy. The SVN49 had internal multipath problem. To mitigate the multipath, one option proposed is to shift the antenna phase center of satellite 152 m above satellite [28], so there is about 517 nanoseconds time offset between L1 and L5. And, the

L5 signal power of SVN49 was about -173.5 dBW corresponding to an average C/No 30dB-Hz resulting in a very weak that is hard to track. The PRN25/SVN62 is broadcasting on both L5 channels. The data message on I channel of PRN25/SVN62 only contains preamble, time of week (TOW), PRN, and cyclic redundancy check (CRC). The other fields of message are filled with zeros. The data on L5 signal of WAAS GEO satellites are the same as L1 signal. But, timing information like TOW is not included in any available message type broadcasting by the WAAS GEO. Table 5.2 lists the current status of GPS/WAAS L5.

Table 5.2 Status of GPS/WAAS L5 before May 6th 2011

Current Status	GPS		WAAS GEO
PRN/SVN	PRN1 /SVN49	PRN25 /SVN62	PRN133 PRN135 PRN138
Channel	Q	I & Q	I
Time offset Between L1 and L5	517 nsec delay ¹	103 nsec adv. ¹	233 nsec 255 nsec 185 nsec adv. ¹
NAV Data	None	CNAV on I w/ limited content	WAAS Message
Signal Power	-173.5 dBW ²	-157.9 dBW[58]	-158.5 dBW [62]

¹The time offset between L1 and L5 is estimated by developed dual-frequency software receiver one time and may include ionosphere delay.

²The power is calculated by subtracting C/No difference between L1 and L5 which is received by Trimble Net-R9 receiver.

5.2 Strategy of Positioning Using L5 Signal

Before fully operation of GPS L5 signals, it is necessary to get assistance from the L1 broadcast. The ephemeris of GPS satellite which is unavailable on the L5 can be obtained from L1. The acquisition of the L5 signal would take longer than the L1 signal because the search space of code phase is ten times bigger. Moreover, the code phase synchronization of two frequencies can be used to provide code phase assistance from L1 to shorten the acquisition time of the L5 signal. Another issue is that timing information like TOW is not included in any available message type broadcasted by WAAS GEO satellites. According to [62], the start of every other 24-bit preamble (provided as 8 bits every second) of WAAS message is synchronized with a 6-second GPS sub-frame epoch. Because the altitude of the WAAS GEO satellite is roughly 15,000 kilometer more than that of the GPS satellites, the preamble of WAAS message is delayed by approximately 50 milliseconds relative to the preamble of GPS signals. Therefore, the TOW obtained from GPS signal can be used to align WAAS signal with millisecond-level accuracy once its preamble is found. Further, the WAAS time has a time offset to GPS time [62]. This time offset should be taken in account when using the WAAS GEO satellites ranging information. Moreover, there exists another time offset between L1 and L5. If one would like to use the ephemeris of L1 to serve for L5, the time offset should be considered. The time offset for two cases can be estimated by adding an unknown when positioning. However, we need better geometry in order to have higher confidence in the solution.

Table 5.3 Issues and solutions of positioning using L5

Issues	Solutions
The ephemeris of GPS satellite is unavailable on L5.	Use ephemeris of GPS satellite from L1 signal.
It takes longer to acquire L5 signal.	Take assistance from L1 to assign code phase.
The timing information like TOW is not included in any available message type broadcasted by WAAS GEO satellites.	Use the TOW obtained from GPS signal to align WAAS signal with millisecond-level accuracy once the preamble is found.
Time offset exists between WAAS time and GPS time.	Add an unknown into positioning to solve it.
Time offset exists between L1 and L5.	

5.3 Description of Signal Collection Hardware

The hardware used to collect dual-frequency dataset is depicted in Figure 5.1. The detail description of hardware is described in [19]. The hardware contains two signal collection systems including the USRP2 software radio system [23] and host computer. The Trimble L-band Zephyr antenna is used to receive the dual-frequency signal. The signal is divided into two branches by a 1-to-2 power splitter. Each one signal passes to a USRP2 board equipped with a DBSRX programmable mixing and down-conversion daughterboard. Individual USRP2 boards are synchronized by a 10 MHz external common clock generator. The USRP2 is controlled by host computer running Ubuntu distribution of Linux. The open-source GNU Radio software-defined radio block is used to configure USRP2 and collect dataset. One of USRP2 is configured to collect L1 (1575 MHz) signal and the other one is for L5 (1176 MHz). The signals are converted to near zero intermediate frequency (IF) and digitized to 14-bit complex outputs (I & Q). Its sampling rate is set as 20 MHz with given 24 MHz bandwidth for L5 and limitations due to the communication rate of Gigabit Ethernet from USRP2 to host computer. The host computer

uses a solid state drive to store the dataset because of high data streaming rate (80 Megabyte/s).

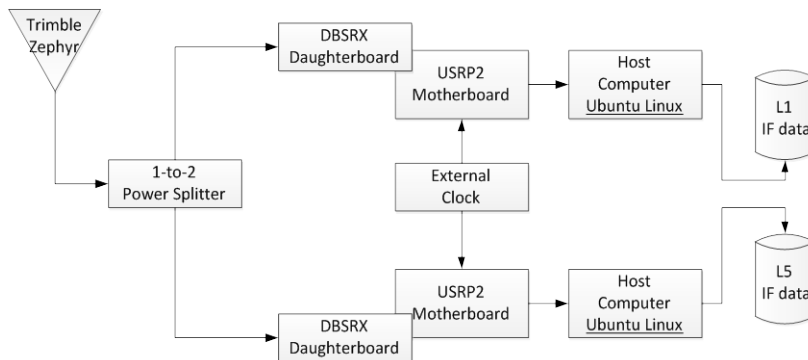


Figure 5.1 Block diagram of the signal collection hardware

5.4 Software Architecture

The software is developed with Visual Studio under Windows. The source code is mostly programmed using C++. Inline assembly is used to program the functions with high computational complexity such as correlation operations. The software architecture [14] of dual- frequency software receiver is depicted in Figure 5.2.

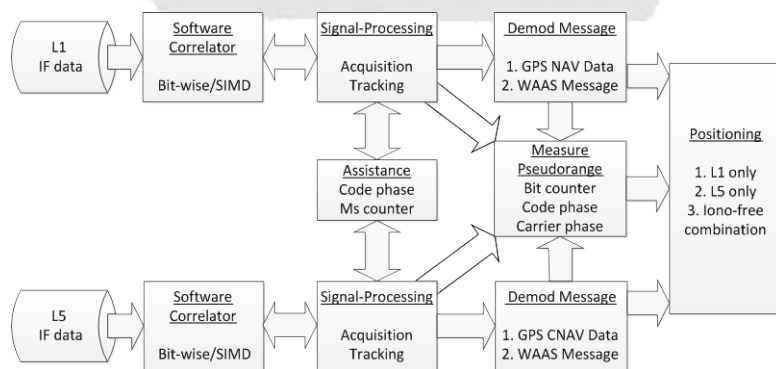


Figure 5.2 Block diagram of the software architecture

Basically, this architecture exploits two sets of parallel signal-processing engines to process L1/L5 signal and combine measurements in positioning. The individual IF data is

read from disk, quantized into 2-bit resolution, and stored in a queue buffer in terms of millisecond length. The data is processed by software correlator which adopts bit-wise parallel algorithm [3] and is implemented by signal instruction multiple data (SIMD) instructions. The software correlator is controlled by signal-processing function for setting PRN, code phase, and Doppler frequency. The signal-processing uses the correlator outputs to decide the state of signal-processing from acquisition to tracking. The details of the signal-processing for the software receiver are described in Section 3.1. An assistant mechanism between L1 and L5 for shortening acquisition time and increase sensitivity of tracking L5Q was developed. It is described in the next section. The tracking state of signal-processing would find the data transition point and decode the NH code. Signal-processing function outputs the navigation data sequence to demodulate messages. Several message formats are possible including GPS NAV, GPS CNAV, and WAAS Message. After demodulating the messages, the TOW is obtained and provides second-level accuracy. The preamble of message is used to find the header of message and provides millisecond-level accuracy. The bit counter, code phase, and carrier phase are measured in the tracking state of signal-processing function to calculate the pseudorange with carrier smoothing (where bit counter represents the number of navigation data bit from the start of week). In the end, the positioning function combines ephemeris and pseudoranges from previous stage to calculate position by three methods in Table 5.4. Figure 5.3 shows the GUI of dual-frequency software receiver including channel status of L1 and L5, sky plot, C/No plot and positioning results for three methods. The software receiver with 12 channels for individual frequency is tracking five satellites with L1 and L5 in the post-processing mode. The processing time represents for the execution time of software receiver. The data streaming time represents for how long IF dataset has processed. The status bar of GUI shows that the processing time is less than data streaming

time. This demonstrates that the developed dual-frequency software receiver can operate in real-time.

Table 5.4 Methods of calculating position

Method	Description
L1 only	Use ephemeris and pseudoranges from L1
L5 only	Use ephemeris from L1 and pseudoranges from L5
Iono-free combination	Use ephemeris from L1 and L1/L5 iono-free combined pseudoranges [66] by $\frac{f_{L1}^2 PR_{L1} - f_{L5}^2 PR_{L5}}{f_{L1}^2 - f_{L5}^2}$

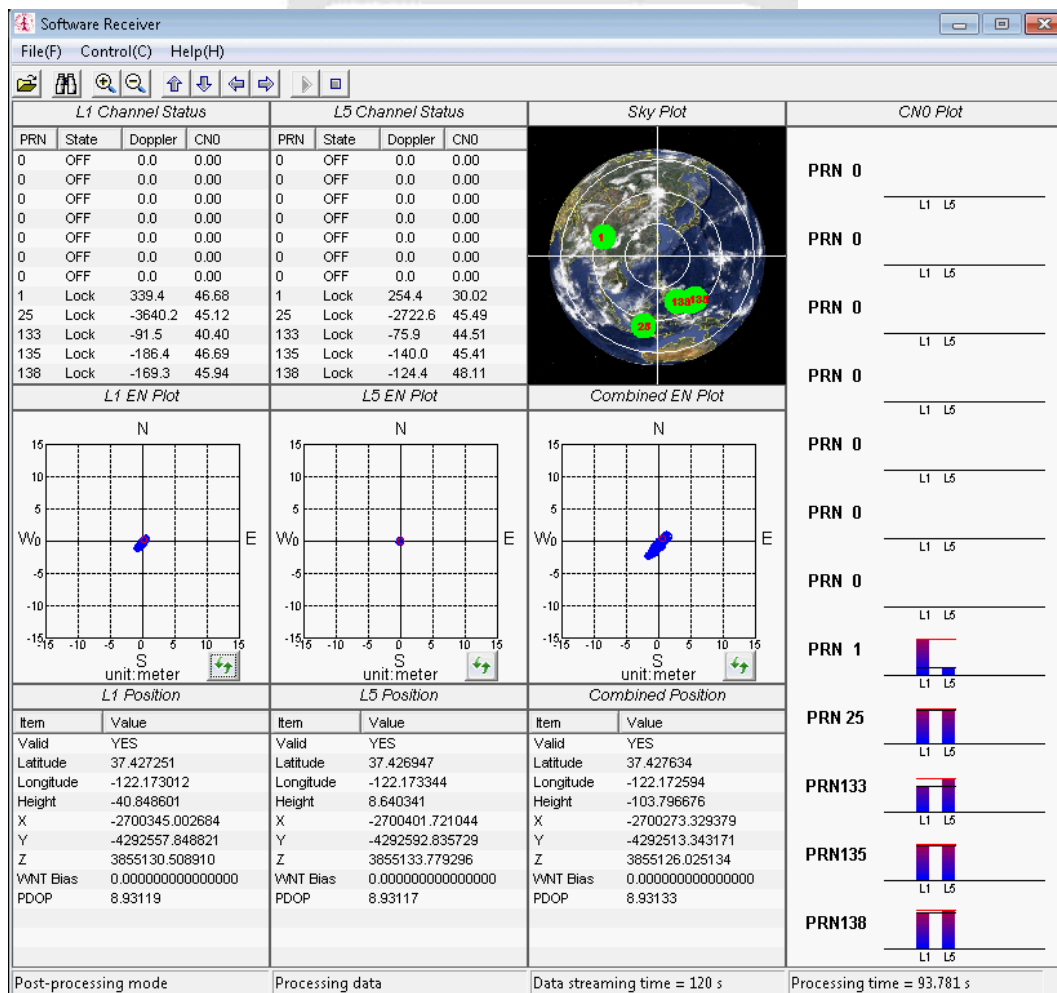


Figure 5.3 Screenshot of dual-frequency software receiver GUI

5.5 Assistance Mechanism between L1 and L5

In this dual-band software receiver, there are two assistance mechanisms between L1 and L5. First one is to shorten acquisition time. Once signal-processing function of one frequency enters tracking state, the assistance mechanism assigns its code phase to the other frequency which is still in the acquisition state. Typically, the L1 channel goes into tracking faster than L5 due to its smaller code phase search space. In case the L5 signal is weak as with SVN49, it needs to integrate several code periods to detect the signal. This would take significant time to search whole code space and frequency space. With this mechanism, the code phase obtained from L1 is fixed and it only needs to search frequency space. Moreover, frequency step for searching can be narrow down for increasing signal sensitivity.

The other assistance mechanism is to increase signal sensitivity for tracking L5Q data-free channel. The millisecond (MS) counter is used to indicate the index of millisecond in one navigation data period. For L1 and L5Q, a MS counter counts from 0 to 19. The assistance mechanism is made by sending the MS counter from L1 to L5Q for getting the NH_{20} code. With known NH_{20} code, one could perform coherent integration for whole period of NH_{20} code. The NH_{20} code can combine the L5Q spreading code to make 20 times longer length code. The coherent integration will benefit when L5 signal is weak like SVN49. However, it also increases the size of code table by 20 times. Alternatively, one can use the original one-millisecond code table to perform correlation, multiply the output by NH code sequence, and then sum over the NH_{20} code period. This approach is easy to implement using hardware correlator. But, for software correlator, the carrier table is made by starting from zero phase [14]. So, each correlator output should be rotated by a phase which is function of carrier frequency and integrated phase. The coherent integration for software correlator is written as follows.

$$\begin{aligned} \begin{bmatrix} I_S \\ Q_S \end{bmatrix} &= \sum_{i=0}^M \begin{bmatrix} I_R^i \\ Q_R^i \end{bmatrix} = \sum_{i=0}^M NH_i \begin{bmatrix} \cos(\Delta\theta^i) & \sin(\Delta\theta^i) \\ -\sin(\Delta\theta^i) & \cos(\Delta\theta^i) \end{bmatrix} \begin{bmatrix} I_Z^i \\ Q_Z^i \end{bmatrix} \\ \Delta\theta^i &= \frac{(\omega_r - \omega)T_I}{2} + \sum_{j=0}^{i-1} \omega T_I \end{aligned} \quad (5.1)$$

where I_S , Q_S denotes integrated correlator output. i is index of integration period. M is the number of integration. I_R^i , Q_R^i , denotes rotated correlator output of i th period. NH_i is i th code of NH_{20} sequence. $\Delta\theta^i$ is phase shift of i th integration. I_Z^i, Q_Z^i denote original zero-phase correlator output of i th period. ω is carrier frequency. ω_r is the frequency after rounding the ω . T_I is the time of code period.

5.6 Solving the Time Offset between GPS Time and WAAS Time

In order to measure the time offset between GPS time and WAAS time, a surveying-level receiver (NovAtel ProPak-G2) is used to calculate the precise location of a fixed antenna. Using the same antenna, IF data are then logged by the collection hardware for three minutes. The software receiver then post-processes the data to obtain the pseudoranges and compute the positions of the satellite. The time offset is calculated by the following equation without subtracting other error sources such as ionosphere delay.

$$t_{os} = \frac{\|\mathbf{x}^{(k)} - \mathbf{x}_r\| - \rho_r^{(k)}}{c} \quad (5.2)$$

where $\mathbf{x}^{(k)}$ is the position of k th satellite, \mathbf{x}_r is the precise position of antenna, $\rho_r^{(k)}$ is the pseudorange from k th satellite to antenna, and c is speed of light. Figure 5.4 shows the results of estimated time offset. The constellation geometry is shown in the upper left portion of the figure. The result corresponds to URA of satellites as Table 5.5. The WAAS GEO satellites have a greater time offset than GPS satellites with most of value resulting from the difference between GPS time and WAAS time.

For solving the time offset, one unknown is added to observation equation as follows [56].

$$\begin{bmatrix} \delta\rho^{(1)} \\ \delta\rho^{(2)} \\ \vdots \\ \delta\rho^{(k)} \\ \delta\rho^{(k+1)} \\ \vdots \\ \delta\rho^{(k+n)} \end{bmatrix} = \begin{bmatrix} (-1^{(1)})^T & 1 & 0 \\ (-1^{(2)})^T & 1 & 0 \\ \vdots & \vdots & \vdots \\ (-1^{(k)})^T & 1 & 0 \\ (-1^{(k+1)})^T & 0 & 1 \\ \vdots & \vdots & \vdots \\ (-1^{(k+n)})^T & 0 & 1 \end{bmatrix} \begin{bmatrix} \delta x \\ \delta b \\ \delta w \end{bmatrix} + \tilde{\varepsilon} \quad (5.3)$$

where $1^{(k)}$ is the unit vector from satellite to antenna, δx , δb , and δw are the unknown corrections for position, receiver clock bias and time offset between WAAS time and GPS time. k , n is the number of GPS satellite and WAAS GEO. Figure 5.5 shows the positioning result using L1 on the ENU coordination respect to precise position. The positioning results where one do not solve for the GPS/WAAS time offset are divided into four groups which are resulted from using different number of WAAS GEOs into positioning. As expected for this case, if one use more WAAS GEOs for positioning, one generally has greater the position bias. However, positioning results that solve for the GPS/WAAS time offset have smaller position bias and are less affected by the number of WAAS GEOs. The solution of WAAS time offset is about 52 nanoseconds which is close to the statement that WAAS time is maintained such that the offset from GPS is less than 50 nanoseconds in [62]. However, this result is only one time estimation and one would present more statistical result in the future.

Table 5.5 User range accuracy (URA) of satellites on collection day

PRN	URA(m)	PRN	URA(m)	PRN	URA(m)
5	2	21	2	30	2
15	2	25	2	133	4096
16	2	26	2	135	4096
18	2	29	2	138	2

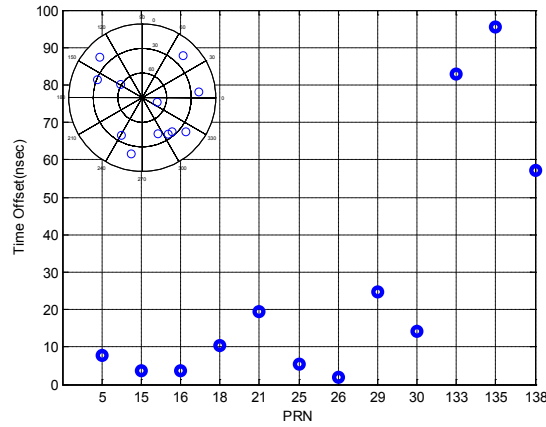


Figure 5.4 Results of time offset calculated by difference between true range and pseudorange (PDOP = 1.827)

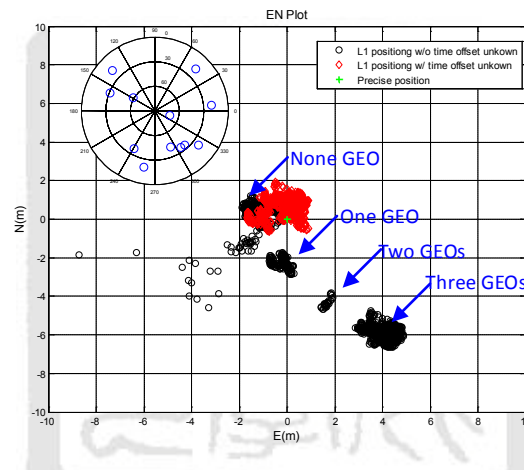


Figure 5.5 EN plot of L1 positioning w/ and w/o WAAS time offset unknown respect to precise position (PDOP = 1.827)

5.7 Solving the Time Offset between L1 and L5

The ephemeris of GPS L1 is used to serve as the ephemeris of GPS L5 for positioning. But, a time offset between L1 and L5 which is resulted from inter-frequency bias (IFB) is needed to subtract from L5 pseudorange. For solving this time offset, one unknown is added to observation equation and uses L5 pseudorange of target satellite instead of L1 pseudorange as follows [56].

$$\begin{bmatrix} \delta\rho_{L1}^{(1)} \\ \delta\rho_{L1}^{(2)} \\ \vdots \\ \delta\rho_{L1}^{(k-1)} \\ \delta\rho_{L5}^{(k)} \end{bmatrix} = \begin{bmatrix} \begin{pmatrix} -1^{(1)} \end{pmatrix}^T & 1 & 0 \\ \begin{pmatrix} -1^{(2)} \end{pmatrix}^T & 1 & 0 \\ \vdots & \vdots & \vdots \\ \begin{pmatrix} -1^{(k-1)} \end{pmatrix}^T & 1 & 0 \\ \begin{pmatrix} -1^{(k)} \end{pmatrix}^T & 0 & 1 \end{bmatrix} \begin{bmatrix} \delta x \\ \delta b \\ \delta v \end{bmatrix} + \tilde{\varepsilon} \quad (5.4)$$

where $1^{(k)}$ is the unit vector from satellite to antenna, δx , δb , and δv are the unknown corrections for position, receiver clock bias and time offset between L1 and L5. Each satellite with the L5 signal goes through this process. The solutions are averaged with time and listed in the Table 5.6. The time offset of PRN 1 is 517 nanoseconds (155 meters). This result corresponds to the current configuration for PRN 1 which places antenna phase center above 152 meters above the satellite [28].

Table 5.6 Time offset and antenna phase center difference between L1 and L5

PRN	L1/L5 Time offset (nsec)	L1/L5 Antenna Phase Center Difference (m)
1	517.219	155.058
25	-103.214	-30.942
133	-233.096	-69.881
135	-255.516	-76.602
138	-185.551	-55.718

5.8 Positioning Result of Dual Frequency Software Receiver

After solving time offset from previous two sections, some corrections can be made by adding the time offset between WAAS and GPS time to satellite time correction. Moreover, subtracting the time offset between L1 and L5 from L5 pseudorange allows us to align the L5 antenna phase center with L1. Applying the three positioning methods mentioned in Table 5.4, one generates the results in the Figure 5.6. Table 5.7 compares the positioning accuracy of three methods. The PDOP = 8.876 is poor due to the limited number of satellites with L1 and L5. The positioning results have some biases respect to

precise position and distribute in a line shape with 45 degree because of no satellite available in the upper right side. The (L5 only) method has smallest bias (1.311 m) and 2-D RMS error (0.711 m) which is much better than SPS service. There is no ionosphere activity on the collection day, so the iono-free combination method which combines the errors from the L1 and L5 has worst performance [66].

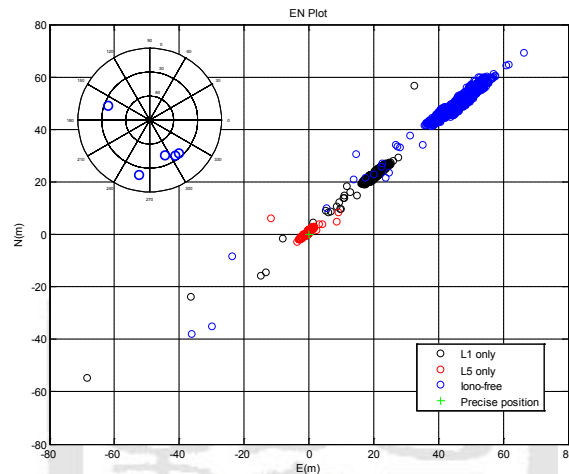


Figure 5.6 EN plot of positioning by L1 only, L5 only, and iono-free respect to precise position (PDOP = 8.876)

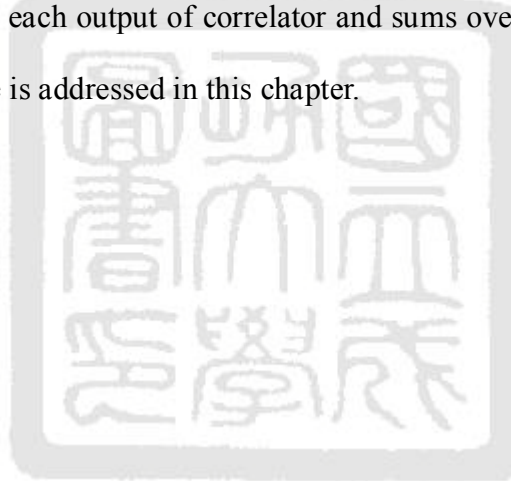
Table 5.7 Positioning accuracy

Method	Bias(m)	1-sigma 2D-RMS(m)
L1 only	30.804	3.103
L5 only	1.311	0.771
Iono-free combination	68.312	7.572

5.9 Summary

A real-time software receiver is implemented for GPS/WAAS dual-frequency (L1/L5) in a PC with a modern processor to demonstrate positioning using L5 signal and dual-frequency combination. This work is potentially the first GPS receiver capable of

L5-only positioning, and solves some significant issues with using the current non-homogeneous L5 constellation and signal broadcast. The solutions of time offset between L1 and L5 as well as WAAS time and GPS time are developed. The results show positioning accuracy of software receiver. While more and more GPS satellites with L5 capability are planned for the future, the software receiver can be used to test new satellite and positioning accuracy will be further improved with better geometry. This chapter also proposes an assistance mechanism between L1 and L5 for shortening acquisition time and increasing signal sensitivity of L5Q signal. For integrating correlation results using combination code consisting of spreading code and NH code in the software receiver, a method which rotates the each output of correlator and sums over NH code period without increasing code table size is addressed in this chapter.





Chapter 6

Software Receiver for GPS/WAAS Controlled Reception Pattern Antenna Array Processing

Adaptive antenna array processing is widely known to provide significant advantages within a GNSS receiver. A main challenge in the quest for such receiver architecture has always been the computational/processing requirements. Even more demanding would be to try and incorporate the flexibility of the software radio design philosophy in such an implementation. This chapter documents a feasible approach to a real-time software radio implementation of a beam steered GNSS receiver and validates its performance. Challenges of such a receiver include real-time capability and interference rejection performance. This research implements a fully software receiver on a widely-available x86-based multi-core microprocessor to process array signals in real time. In the documented implementation, two kinds of software receiver are demonstrated. First one uses 7-element antenna array with low-resolution (2 bits) Analog to Digital Converter (ADC) and steers a beam to single satellite. Second one can process 4 antennas and 12 channels all-in-view CRPA capable of rejecting more than 3 interferences. SIMD instructions assembly coding and multithreaded programming fully documented within the chapter are used to reduce computation complexity. Conventional antenna array system receivers use the geometry of antennas and cable lengths known in advance. This CRPA implementation is architected to operate without extensive set-up and pre-calibration enhancing its suitability for commercial users. By using Space-Time Adaptive Processing

(STAP), the software receiver can make notches both in the frequency and space domain. The performance of the design, both in terms of computational efficiency and interference rejection, is also demonstrated within the chapter. Several synthetic interference sources are added to a collected multiple antenna dataset with specified direction and signal power. The validation component of the chapter demonstrated that the developed software receiver has reliable high power/multiple source interference rejection performance and provides real-time capability.

6.1 Introduction

GPS provides 24 hour all-weather position, navigation, and timing (PNT) services worldwide. However, GPS and GNSS signals are relatively weak and thus vulnerable to deliberate and unintentional interference. An electronically-steered antenna array system provides an effective approach to mitigate interference by controlling the reception pattern and steering beams/nulls. As a result, so called controlled reception pattern antenna (CRPA) array have been deployed by organizations such as the US military which seek high levels of interference rejection. However, there is a tradeoff in increased cost and computational complexity which to-date has not been acceptable to commercial GNSS users. The research conducted in this chapter brings the directive gains and interference rejection benefits of electronically-steered antennas closer to commercial users by implementing CRPA processing in a software receiver. In the literature, the CRPA receivers have been implemented by different approaches. [67] used a 4-element antenna array and a CRPA system to perform spatial nulling adaptive array. The CRPA system used an analogue approach to combine the signals and adopt a correlation feedback to derive the weight vector. [48] implemented a MATLAB software receiver to assess the performance of the beamforming algorithms and found the steering vectors by direction of arrival (DoA)

algorithms. [52] proposed a beamforming architecture for real IF signals and calculated optimum weights from given GPS almanac. [30] developed a 4-element antenna array and front-ends for dual-band L1/L5. They also implemented a MATLAB software receiver for field test of adaptive beamforming algorithms with directions of satellites calculated from DoA estimation. [3] implemented a real-time hardware and software platform capable of the one-channel digital beamforming algorithm. They use Field Programmable Gate Array (FPGA) for digital beamforming up to eight antenna elements and deliver the resulting spatially-filtered digital signal to a PC by Ethernet bus. The signal can be stored for post-processing or processed by a real-time software receiver. However, these papers do not implement an all-in-view CRPA receiver with real-time capability. A bit-wise parallel software correlation algorithm [3] could be used to implement a real-time software receiver with 2-bit resolution. But, this does not have enough dynamic range of signal and is easy to saturate in case of high power interference.

Conventionally, antenna array system receivers perform CRPA with the geometry of the antennas and cable lengths known in advance. In the developed software receivers, the algorithm implemented allows for operation without such a priori knowledge. As the carrier phase difference is related to geometry of antenna as well as line biases of the cables, this can be used for constraints of adaptive beamforming algorithm.

In order to achieve real-time capability, some functions in software receiver are programmed by assembly using SIMD instructions. This chapter addresses a structure of SIMD parallel programming and its codes are included as an example. Additionally, multi-threading programming is adopted to fully exploit the multi-core resources of the processor. An execution flow is designed to distribute the tasks across multiple cores. For reducing the operations of the software correlator, local replicas of code and carrier at zero-phase are made prior execution, so phase shifting is needed to rotate the phase of

correlator outputs in the tracking loop. A technique for the software correlator to correlate the Intermediate Frequency (IF) with local replica for whole code period without crossing data transition is also described in this chapter.

For the 7-element antenna array, an experiment was conducted to demonstrate that the carrier to noise ratio (C/N_0) is enhanced by array processing. With an injected interference, the developed software receiver is also tested with low power CW and CDMA interferences. Comparisons are made between a single antenna, CRPA by deterministic beamforming and MVDR adaptive beamforming.

For the 4-element antenna array, one adds synthetic interferences to the collected dataset. Two scenarios are made to validate and demonstrate the interference rejection performance of the CRPA software receiver. The first one is with high Interference-to-signal ratio (I/S) interference. The second one is with multiple interferences of different types from different directions. The results are illustrated by angle-frequency responses.

6.2 Beamforming Algorithm Used in the Software Receiver

The primary goal of a controller reception pattern antenna is to enhance the carrier-to-noise ratio of desired signal and reject the interferences. Digital beamforming approaches are used to implement CRPA by combining the signals of antenna array. The architectures of combination are addressed in the literatures including frequency-processing, spatial-processing, STAP [24] and Space-Frequency Adaptive Processing (SFAP) [30]. The frequency-processing is primarily against narrowband interference. The spatial-processing is efficiently against both of broadband and narrowband interference, but only can cancel $N-1$ interferences, where N is the number of element in the antenna array. The STAP or SFAP places nulls both in the frequency and

spatial domain and can cancel more than $N-1$ interferences if some of interferences are narrowband. The computational complexity of STAP or SFAP is high, so implementing them in the real-time software receiver needs a powerful processor and parallel programming. Several algorithms are used to calculate weights. Some of them optimize certain conditions with known signal structure of the desired signal such as maximum signal-to-interference ratio, minimum mean square error and minimum output power. Others algorithms do not need prior knowledge of signal structure and minimize output power to certain constraints such as Minimum Variance Distortionless Response (MVDR) [3] and Constrained Least Mean-Squares [26]. The constraints can be set to form a beam in the direction of satellite or steer a null in the direction of interference. However, the steering vector associated with the direction of satellite needs to be obtained either from satellite ephemeris or carrier phase differences between elements of antenna array.

Due to low power of GNSS signals and existing multiple-source, k multi-type of interference, one adopts the STAP with adaptive MVDR beamforming algorithm to implement CRPA as shown in Figure 6.1.

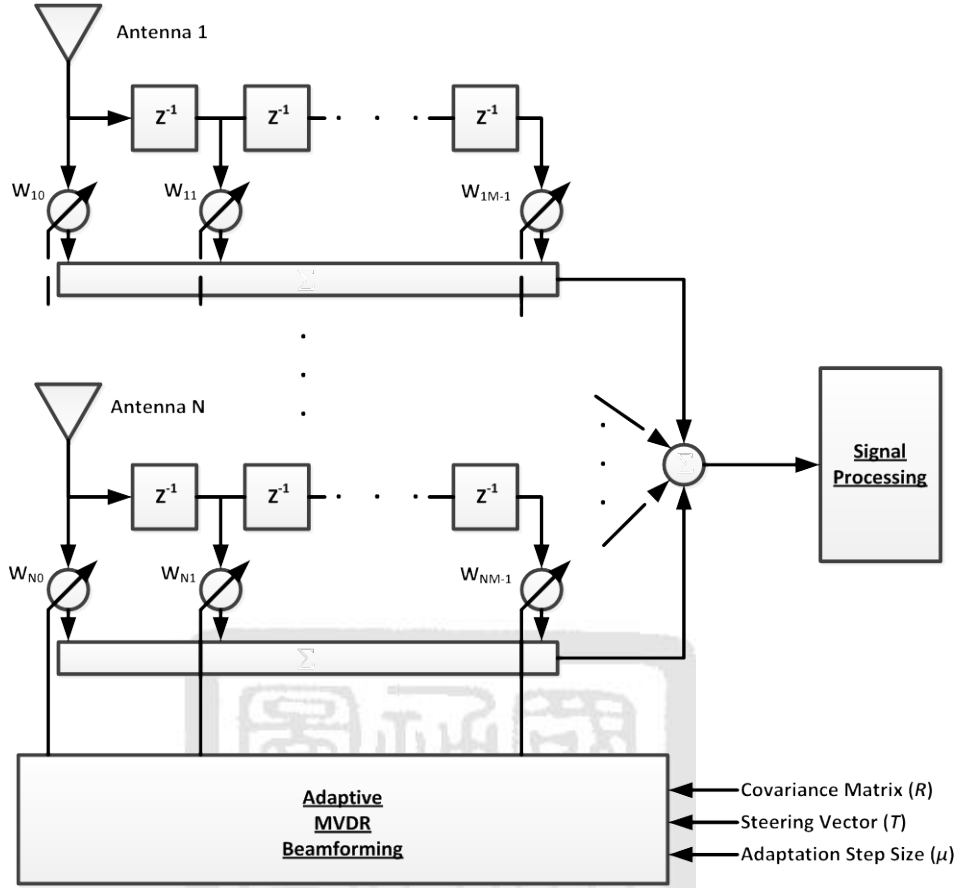


Figure 6.1 Architecture of STAP with adaptive MVDR beamforming

The beamforming combines the signal of antennas, multiplied by complex weights and then summed over all antennas as equation (6.1).

$$\begin{aligned}
 y[k] &= \sum_{n=1}^N \sum_{m=0}^{M-1} w_{nm} s_n[k-m] = W^T S[k] \\
 W &= [w_{10} \quad w_{20} \quad \cdots \quad w_{N0} \quad w_{11} \quad \cdots \quad w_{N1} \quad \cdots \quad w_{N0} \quad \cdots \quad w_{N(M-1)}]^T \\
 S[k] &= [s[k] \quad s[k-1] \quad \cdots \quad s[k-(M-1)]]^T \\
 \underline{s}[k] &= [s_1[k] \quad s_2[k] \quad \cdots \quad s_N[k]]
 \end{aligned} \tag{6.1}$$

where N is the number of element in the antenna array. M is the number of tap. $s_n[k-m]$ is the signal from n th antenna and m th tap. w_{nm} is the weight associated to the n th antenna and m th tap. The MVDR algorithm minimizes the output power and constraint the gain of the direction of desired signal to unity as (6.2) [26].

$$\begin{aligned} &\text{minimize } y^* y = W^* R W \\ &\text{subject to } W^T C = 1 \end{aligned} \quad (6.2)$$

where R is the covariance matrix of input signals and C is the constraint vector toward the target satellite. The covariance matrix R is estimated by computing sample covariance matrix with assuming the sample mean is zero as equation (6.3).

$$R[k] = \frac{1}{N_s} \sum_{t=0}^{N_s-1} S[k+t] S^*[k+t] \quad (6.3)$$

where N_s is the number of samples to compute the covariance matrix. The constraint vector C is composed of steering vector T in the first to N th component and zeros in the others represented in equation (6.4).

$$\begin{aligned} C &= \begin{bmatrix} t_1 & t_2 & \cdots & t_N & \underbrace{0 \cdots 0}_{(M-1)N} \end{bmatrix}^T \\ T &= [t_1 \quad t_2 \quad \cdots \quad t_N]^T \end{aligned} \quad (6.4)$$

Traditionally, the signal of first antenna is set as a reference and its component of steering vector is set to 1. The other components are set as phase shifts relative to the reference antenna represented in equation (6.5).

$$t_i = \exp(-j(\Delta\phi_i^l + \Delta\gamma_i)) \quad (6.5)$$

where $\Delta\phi_i^l$ is the phase difference based on antenna geometry and the direction of desired satellite. $\Delta\gamma_i$ is the phase resulting from the difference of cabling and RF chain including down converter and digitization among elements of antenna array. $\Delta\phi_i^l$ can be calculated as

$$\Delta\phi_i^l = \frac{2\pi \vec{p}_i \cdot \hat{r}^l(\phi, \theta)}{\lambda} \quad (6.6)$$

where \vec{p}_i is the baseline vector of the i th antenna and $\hat{r}^l(\phi, \theta)$ is the unit vector to satellite l as shown in the Figure 6.2.

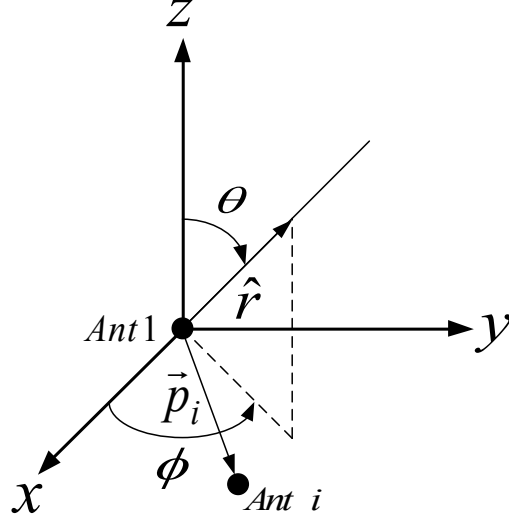


Figure 6.2 Antenna geometry and direction of satellite showing calculation of $\Delta\phi_i^l$

Assume \vec{p}_i is known and $\hat{r}_i(\varphi, \theta)$ can be obtained through positioning. Only $\Delta\gamma_i$ is needed to be re-calibrated whenever any part of hardware of antenna array is changed. From the derivation in [26], the solution of (6.2) is

$$W = R^{-1}C(C^T R^{-1}C)^{-1} \quad (6.7)$$

There are three matrix inversions in (6.7). Alternatively, [26] derived an adaptive approach which iteratively updates the weight listed by (6.8)

$$\begin{aligned} W[0] &= \frac{1}{N}C \\ W[k+1] &= \left(I - \frac{1}{N}CC^T\right)(I - \mu R[k])W[k] + \frac{1}{N}C \end{aligned} \quad (6.8)$$

where μ is the adaptation step size which can be constant or variable related to covariance matrix. In our software receiver, μ is calculated by equation (6.9).

$$\mu = \alpha / \text{trace}(R) \quad (6.9)$$

6.3 Obtaining the Steering Vector without a Prior Calibration

For adaptive MVDR beamforming, obtaining the steering vector is the key to implementing a CRPA. However, as mentioned in the previous section, some parameters of

the steering vector need to be calibrated. Some of them are obtained after positioning. A software receiver has the flexibility of implementing multiple channels to track multi-antenna signals. Integrated carrier phase (ICP) is one of the tracking outputs of the phase locked loop. The ICP is often used to smooth code pseudorange for improving accuracy of positioning. In our software receiver, ICP differences between different antennas are taken to build the steering vector instead of deriving the vector from the azimuth/elevation to the satellites and baseline vectors of antennas [21]. Figure 6.3 shows the block diagram of the approach used to obtain the steering vector in our software receiver.

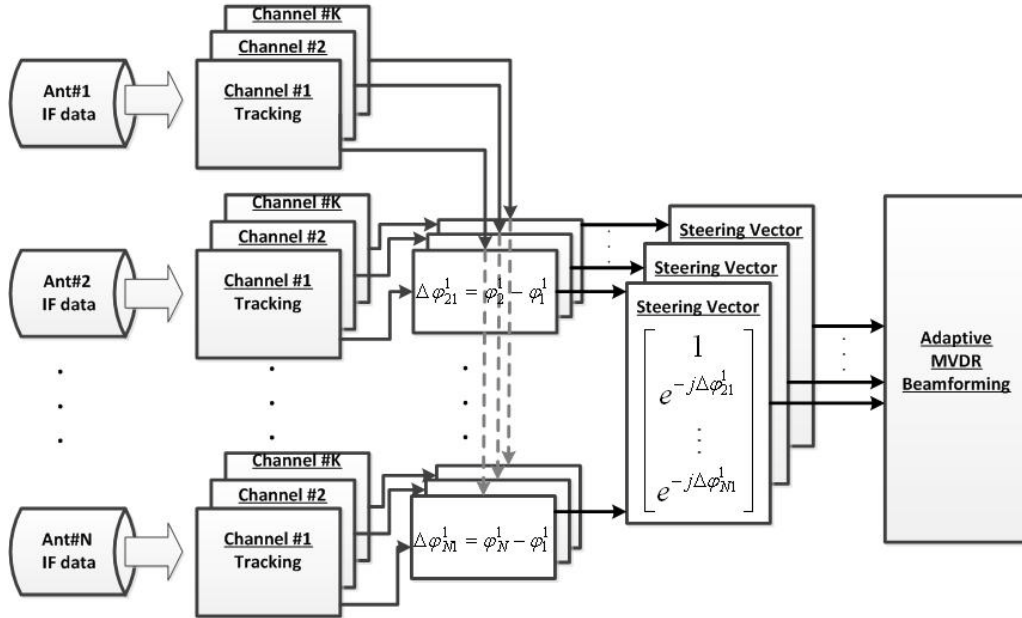


Figure 6.3 Mechanism to obtain the steering vectors in a software receiver

The IF signal of each antenna is processed in the K channels in which each channel is assigned to track one satellite. Once all of the channels from different antenna assigned to a given satellite are tracked, the ICPs are initialized by filtered phases of the in-phase and quadrature-phase components of correlator output. Then, the ICPs are continuously integrated and the phase difference between the reference antenna and others are computed

at runtime. Taking the difference of ICP between different antennas gives component of the steering vector. The phase difference includes the all the parts in (6.5) and also can be used to calibrate the cabling/RF chain part $\Delta\gamma_i$ after positioning with known baseline vectors of antennas.

6.4 Implementation I

The developed CRPA real-time software receiver runs on a PC platform and uses USB for data input. It currently operates on GPS L1 C/A signal. There are a total of 80 channels of which ten are allocated to process real data for each antenna as well as ten tracking channels to process complex data for the beam formed composite signal. ICP measurements from each channel assigned to track the same satellite are collected to build the steering vector. Only one satellite is selected for beam steering (single beam). The weight updating rate of MVDR algorithm is 1 KHz. Moreover, positioning is dedicated to a beam formed composite signal. There is a GUI to show ICP differences, weights and C/No of all channels to illustrate the beamforming performance as well as the positioning result.

6.4.1 Hardware Architecture

The 7-element antenna elements are arranged in circle with one wavelength between the each on a circular plane of aluminum. In the RF front-end, the L1 signal is down-converted to a 4.1304 MHz intermediate frequency and sampled at 16.3676 MHz. The front-end outputs 2-bit real IF data. The clocks of the front-ends must be perfectly synchronized for array signal processing. A function generator set as sinusoidal wave output at 16.3676 MHz. It is divided to two branches by a 1-to-2 splitter. One is used to drive the clock input of USB microcontroller. The other one is further divided by a 1-to-8

splitter to eight branches which are used as reference clock for each front-end. The USB microcontroller serves as a bridge from RF front-end to PC. Its 16-bit parallel digital interface I/O is connected to seven 2-bit RF front-ends outputs. The resulting data rate is 32 MByte/s close to the limit of USB 2.0 data rate of 40 MByte/s. It is necessary to have an efficient strategy to reach such a high data rate. The hardware including antenna array, RF front-end and USB microcontroller board was developed in [6]. Since the developed software receiver implements as many as 80 channels, it required a multi-core processor with multi-thread support. Further, the processor needs to support single instruction multiple data (SIMD) instructions to account for the high computational complexity. The implementation of SIMD instructions for Intel is called as MMX/SSE [39]. The 128-bit register of SSE can be divided to several items in terms of bytes, words, or double words and perform parallel mathematical and comparative operations. The used processor is the quad-core Intel Core i7 which runs eight threads at a time and supports SIMD instructions sets including MMX and SSE (1, 2, 3, 4, 4.1, 4.2). The hardware set-up of the CRPA software receiver is depicted in the Figure 6.4.

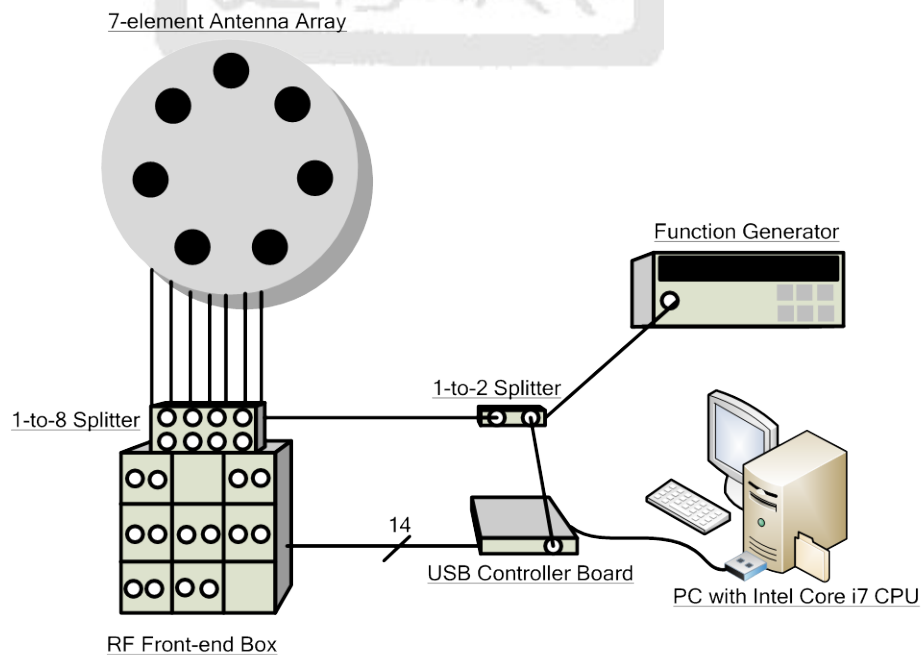


Figure 6.4 Diagram of hardware set-up of CRPA software receiver

6.4.2 Software Architecture

The software is developed with Visual Studio under 32-bit version of Microsoft Windows XP. Most of source code is programmed using C++. The functions with high computational complexity are programmed by inline assembly such as correlation operation and covariance matrix calculation. The components are listed in the Table 6.1.

Table 6.1 Component list of software architecture in the implement I

Component	Description
USB driver USB C++ library	Provided by chip manufacturer.
Software correlator	Hand-coded inline assembly using SSE instruction set based on bit-wise parallel algorithm [3].
Tracking Positioning	Use GPL-GPS [30] open source code and modify interface to software correlator.
MVDR adaptive beamforming -Covariance Calculation -Weight Update	Hand-coded inline assembly using SSE instruction set for calculating covariance. Hand-coded C++ code for weight update
Weight-and-Sum and quantization of composite signal	Hand-coded inline assembly using SSE instruction set.
System Program	Arrange threads to achieve real-time capability

The IF data transfer uses the C++ library provided by chip manufacturer for communicating with USB driver. The procedure of the IF data transfer is depicted in the Figure 6.5. The width of data transfer for one sample is 16-bits for the entire IF data stream from all antennas. The data is further separated into a circular queue of individual antennas. The size of the entry of the queue is C/A code period (one msec). New IF data is stored into the rear index of the queue and the processing of the data is started from front index of queue. In order to fully exploit the resources of Intel Core i7 CPU, multiple threads are

created and arranged in a way such that at most 8 threads need be executed. Figure 6.6 shows the planned execution flow of the threads.

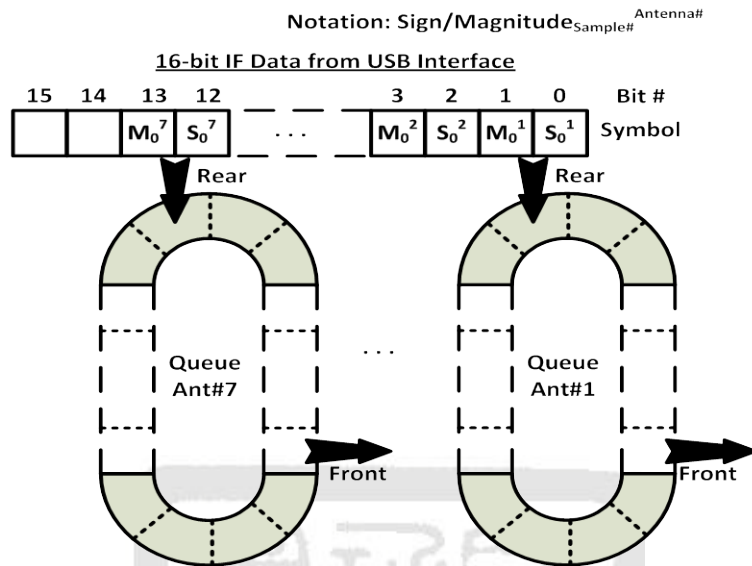


Figure 6.5 Procedure of IF data transfer from the USB interface to the circular queues

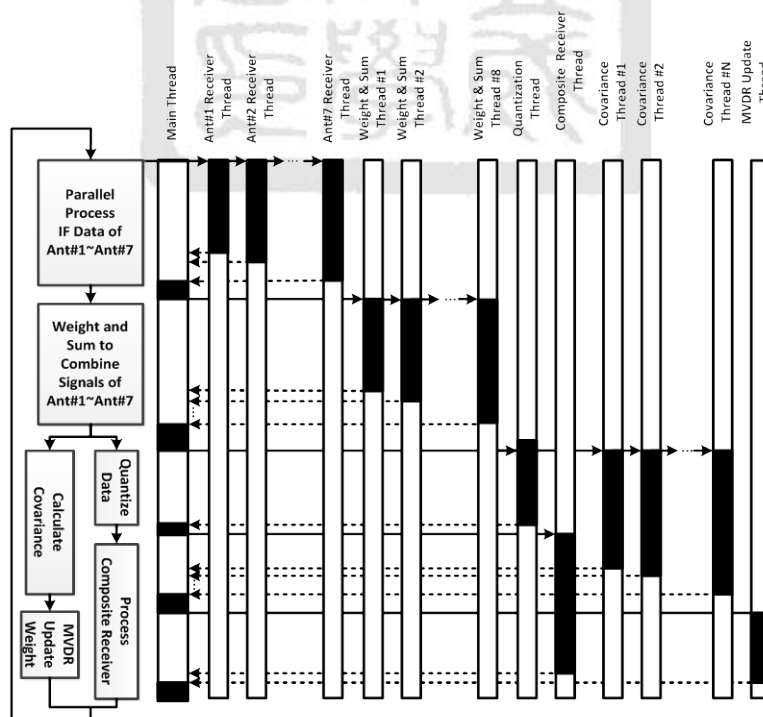


Figure 6.6 Planned execution flow of threads every msec

The main thread controls all the working threads. At first, seven receiver threads process the IF data obtained from the individual queue of antennas. Software correlation and acquisition/tracking are performed within each receiver thread. Next, the weight-and-sum operations for beam forming are separated into 8 threads and run simultaneously. The combined data is further quantized to complex 2-bit outputs, and then processed within a composite antenna receiver thread. In the composite receiver thread, not only is software correlation, acquisition and tracking performed, but the position solution is also calculated. In parallel, the covariance matrix is calculated by averaging over one msec within N threads where N depends on the number of elements in the covariance matrix. In addition, the MVDR weight update is performed in another thread. The whole procedure must be finished within one msec to achieve real-time capability.

6.4.2.1 Weight and Sum Code Example

As mentioned in the previous section, the weight-and-sum operation is to combine the IF data by multiplying the complex weights and summing over all antennas. This operation can be performed in parallel using SSE instructions. Figure 6.8 shows implementation of the weight-and-sum operation using SSE instruction. At first, the real inputs and complex weights are loaded into the XMM registers in terms of bytes. Then, the multiply packed signed integers and store low (PMULLW) instruction is performed to multiply real inputs with complex weights in parallel. Finally, a parallel addition is performed three times to obtain the summations of real and imagery components.

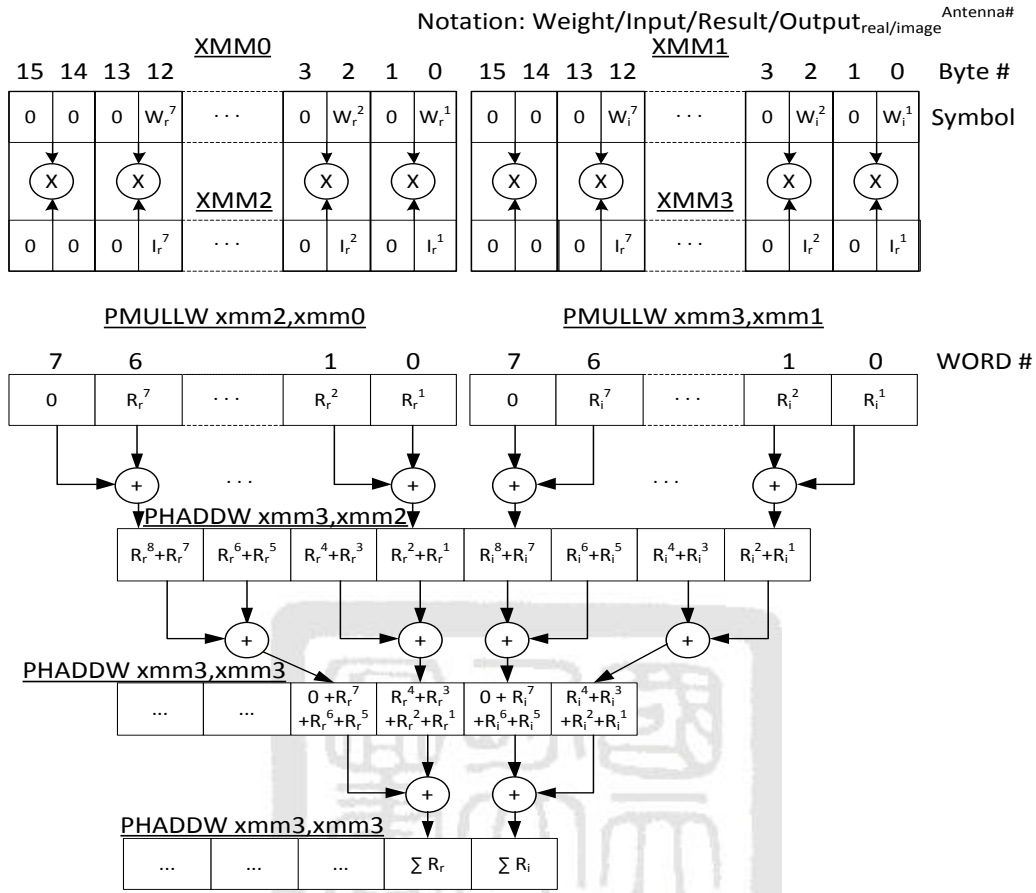


Figure 6.7 Example of the weight-and-sum operation using SSE instructions

6.5 Implementation II

The developed CRPA real-time software receiver runs on a PC platform and uses IF datasets as input. It currently operates on GPS/WAAS L1 C/A signal and uses 4-element antenna array. There are a total of 60 channels of which 48 channels are used to process complex data from antennas as well as 12 channels for the beam formed composite signal. Each of the beam formed channel use one set of weights to steer a beam toward satellite. The software receiver can track 12 signals of GPS/WAAS satellite and steer 12 beams simultaneously. The STAP structure with adaptive MVDR beamforming mentioned in the previous sections is implemented in our software receiver to enhance C/No and reject multiple types of interferences. The steering vectors as inputs of adaptive MVDR

beamforming are obtained from taking ICP difference without any calibration. Hence, the software receiver can form beams before positioning. The weight updating rate of MVDR algorithm is 1 kHz for adapting high dynamic interferences. The current platform can achieve real-time performance with up to 5 taps. There exists a trade space between numbers of antenna array, channel and tap as well as weighting updating rate. Given the 4-element antenna array, our software receiver can reject at most three broadband interferences. It can reject more than three interferences if some of them are narrowband. Moreover, positioning is dedicated to the beam formed composite signal. In order to illustrate the interference rejection performance, the cabling/RF chain biases are calibrated after positioning. Then, the results of biases combined with weights are used to calculate the accurate angle-frequency response which provides a useful visualization of the outputs of STAP.

6.5.1 Hardware Architecture

The developed CRPA software receiver can process any datasets with 16-bit resolution. The hardware depicted in Figure 6.8 is what one uses to collect multi-antenna datasets. The detail description of the hardware is described in [19]. The hardware contains four signal collection systems including the USRP2 software radio system and host computer. The Trimble L-band Zephyr antenna is used to receive the L1 signal. Each one signal passes to a USRP2 board equipped with a DBSRX programmable mixing and down-conversion daughterboard. Individual USRP2 boards are synchronized by a 10 MHz external common clock generator. The USRP2 is controlled by host computer running Ubuntu distribution of Linux. The open-source GNU Radio software-defined radio block is used to configure USRP2 and collect dataset. All of USRP2 are configured to collect L1 (1575 MHz) signal. The signals are converted to 0.42 MHz near zero IF and digitized to

14-bit complex outputs (I & Q) which would provide more than 40 dB dynamic range for interference rejection. The IF signals are sampled at 4 MHz and stored in the format of 2 bytes (16 bits). And then, the data is transferred to host computer by Gigabit Ethernet. The host computers use the solid state drives to store these IF data sets for post-processing.

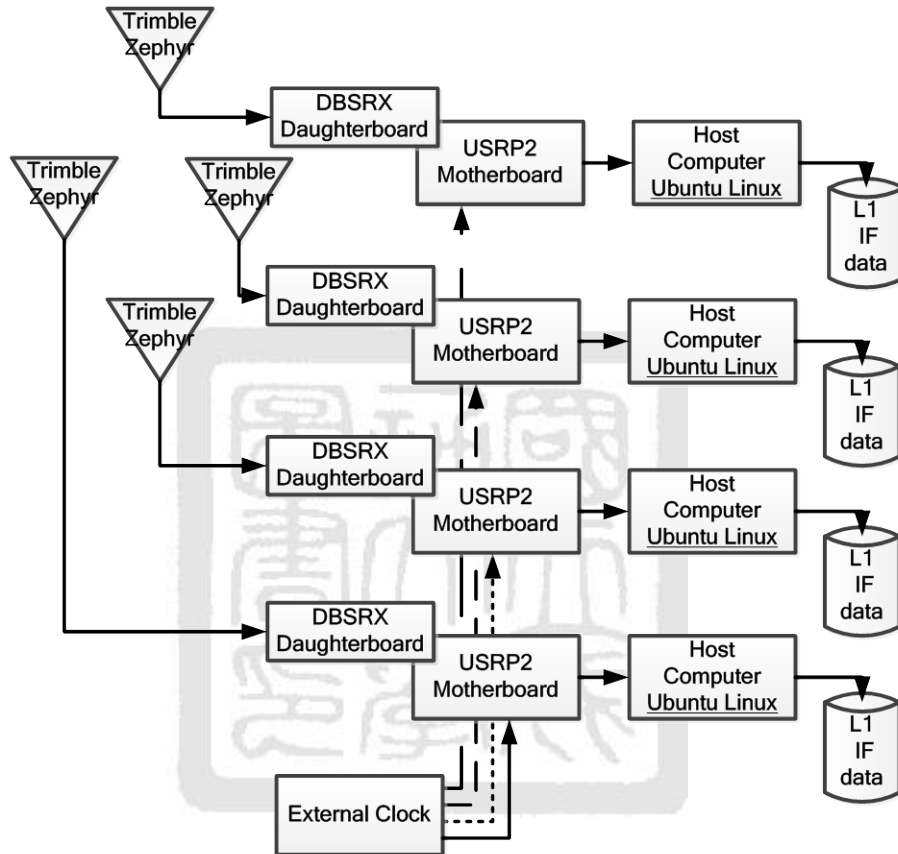


Figure 6.8 Block diagram of the signal collection hardware

Since the developed software receiver implements as many as 60 channels, a multi-core processor supporting multi-thread is required accounting for the high computational complexity. Further, the processor supporting SIMD instructions would accelerate the computation by parallel-computing programming. For 64-bit mode, there are sixteen 128-bit SSE registers which can be divided to several items in terms of bytes, words, or double words and perform parallel mathematical and comparative operations.

The instructions in the SSE, SSE2, SSE3, and SSE4.1 are used in our software receiver, so any CPU supporting all of these versions of SSE can be adopted. One adopts the quad-core Intel Core i7 which can run eight threads at a time and supports up to SSE4.2.

6.5.2 Software Architecture

The software is developed with Visual Studio under 64-bit version of Microsoft Windows 7. Most of source codes are programmed using C++. The functions with high computational complexity are programmed by assembly such as correlation operation and covariance matrix calculation. Software flow is shown in Figure 6.9. The used components are listed in the Table 6.2. Each component would be discussed in the following sections.

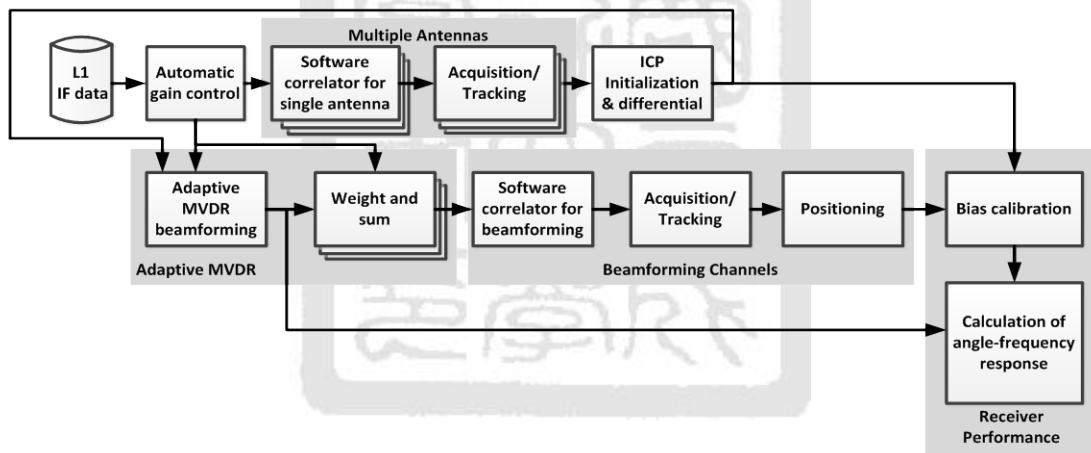


Figure 6.9 Software flow of software receiver in the implementation II

Table 6.2 Component list of software architecture in the implementation II

Component	Description	Frequency
Automatic gain control	Equalize noise power of all elements of antenna array to full 16-bit representation. Hand-coded assembly using SSE instruction set.	Call right after reading every 1 msec data from hard disk.
Software correlator	Perform 16-bit complex data correlation by software approach. Hand-coded assembly using SSE instruction set.	1 msec
Acquisition/Tracking Positioning	Signal processing and positioning function for GPS/WAAS satellite. Based on GPL-GPS open code [30], modify the interface to software correlator, and add the support of WAAS L1 signal.	1 msec for Acquisition/Tracking 100 msec for Positioning
ICP initialization and calculation of the differential ICP	Perform the initialization and difference calculation of ICP for determining the steering vectors. Hand-coded C++ code.	1 msec
Adaptive MVDR beamforming -Covariance calculation -Weight updating	Implement the adaptive MVDR beamforming including covariance matrix calculation and weight updating. Hand-coded assembly using SSE instruction set for covariance calculation. Hand-coded C++ code for weight updating.	1 msec
Weight and sum	Perform the signal combination as equation (6.1). Hand-coded assembly using SSE instruction set.	1 msec
Bias calibration	Perform the calibration of cabling/RF chain bias. Hand-coded C++ code.	One time after first positioning
Calculation of angle-frequency response	Perform the calculation of angle-frequency response. Hand-coded C++ code.	5 sec
System Program	Arrange threads to achieve real-time capability. Hand-coded C++ code.	1 msec

6.5.2.1 Automatic Gain Control

The software receiver starts from reading the IF data from the disk. Then, each of IF data is amplified by a gain which is set by automatic gain control function. The gain is updated iteratively by finding maximum value in a one millisecond length of data as shown in equation (6.10)

$$\begin{aligned} M &= \max_{i=1}^{N_s} |s_i| \\ N &= (2^{N_b} - 1) / M \\ G[k+1] &= G[k] + \alpha(N - G[k]) \end{aligned} \tag{6.10}$$

where s_i is i^{th} digitized sample of the IF signal, N_s is the number of samples in one millisecond, N_b is the number of bit for representing data, and α is updating step. The objective of the gain control is to equalize the noise power of all elements of antenna array due to different RF chains.

6.5.2.2 Software Correlator

In the software correlator, the IF signals are wiped off code and carrier. The 16-bit complex data (I/Q) is correlated with local code and carrier replica. Due to its high computational complexity, the software correlator needs to be programmed using the assembly language through SIMD instructions. Based on the SIMD library correlator in [32], an assembly code for the specific code and carrier table is developed to speed up the execution. The structures of code, carrier table, and IF data are depicted in Figure 6.10. All of tables are formatted as 16-bit short integer. The code table is made by $N+2P$ phase for each PRN code where P is the number of samples between Early-to-Prompt spacing and N is the number of samples in one msec. The carrier table is divided by sine and cosine tables and started with zero phase for each Doppler frequency. I and Q of IF data are bundled together to make a pair. The correlation operation is done between IF data with the nearest

sample to zero phase and local replicas Early/Prompt/Late components according to code phase measurement.

Code Table									
C_{N+P-1}^1	...	C_N^1	C_{N-1}^1	...	C_2^1	C_0^1	C_{-1}^1	...	C_{-P}^1
C_{N+P-1}^2	...	C_N^2	C_{N-1}^2	...	C_2^2	C_0^2	C_{-1}^2	...	C_{-P}^2
⋮									
C_{N+P-1}^M	...	C_N^M	C_{N-1}^M	...	C_2^M	C_0^M	C_{-1}^M	...	C_{-P}^M

Carrier Table							
S_{N-1}^1	...	S_1^1	S_0^1	CS_{N-1}^1	...	CS_1^1	CS_0^1
S_{N-1}^2	...	S_1^2	S_0^2	CS_{N-1}^2	...	CS_1^2	CS_0^2
⋮				⋮			
S_{N-1}^F	...	S_1^F	S_0^F	CS_{N-1}^F	...	CS_1^F	CS_0^F

IF Data							
Q_{N-1}	I_{N-1}	...	Q_1	I_1	Q_0	I_0	

C: Code
S: Sine
CS: Cosine
I: IF Data I
Q: IF Data Q
N: # of sample in 1msec
M: # of PRN
P: # of sample in E-P spacing
F: # of Frequency
Format: 16-bit short

Figure 6.10 Structure of code table, carrier table, and IF data

The correlation with complex IF data performs operation as (6.11)

$$\begin{aligned}
 CQ + jCI &= \sum_{k=0}^{N-1} C_k (CS_k + jS_k)(Q_k + jI_k) \\
 CQ &= \sum_{k=0}^{N-1} CS_k \times C_k Q_k - S_k \times C_k I_k = \sum_{k=0}^N CQ_k \\
 CI &= \sum_{k=0}^{N-1} CS_k \times C_k I_k + S_k \times C_k Q_k = \sum_{k=0}^{N-1} CI_k
 \end{aligned} \tag{6.11}$$

where CQ and CI are the real part and imaginary part of correlator output, respectively.

The following procedure describes the implemented software correlator with complex IF data.

1. Multiply I and Q of IF data by PRN code
2. Use Sine and Cosine table to make the complex data format

3. Perform complex multiplication by multiply-and-add
4. Accumulate the complex results

Using SIMD instructions and 128-bit-wide SSE registers, it is possible to operate eight 16-bit-wide words in an instruction in parallel. The complex data format used for executing the parallel multiply-and-add PMADDWD instruction is depicted in Figure 6.11.

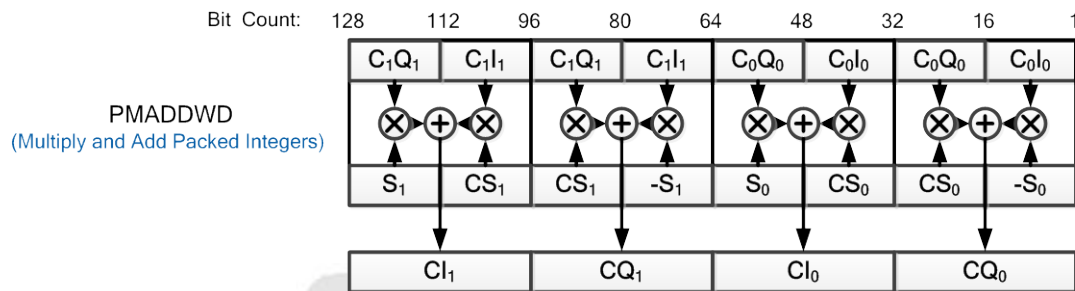


Figure 6.11 Complex data format used for executing PMADDWD instruction

Totally, 8 multiplications and 4 additions are performed in one instruction cycle, so it would reduce around 12 times computational load. Then, program following assembly code compatible to SSE, SSE 2, SSE3, and SSE4.1:


```

movupd xmm0, [r8]          ; load Code(C)
movupd xmm1, [r9]          ; load CarrSin(S)
movupd xmm2, [r10]         ; load CarrCos(CS)
movupd xmm3, [r11]         ; load InDataIQ
movupd xmm4, [r11+16]      ; load InDataIQ1
movdqa xmm5,xmm0           ; move Code to xmm5
movupd xmm6,xmm2           ; move CarrCos(CS) to xmm6
                           ;          7      6      5      4      3      2      1      0
punpcklwd xmm5,xmm5        ; xmm5 = C3 | C3 | C2 | C2 | C1 | C1 | C0 | C0
pmullw xmm5,xmm3           ; xmm5 = C3Q3| C3I3| C2Q2| C2I2| C1Q1| C1I1| C0Q0| C0I0
punpcklwd xmm6,xmm1        ; xmm6 = S3 | CS3 | S2 | CS2 | S1 | CS1 | S0 | CS0
pxor xmm7,xmm7            ; clear xmm7
psubw xmm7,xmm1           ; xmm7 = -S
movdqa xmm14,xmm7         ; move -S to xmm14
punpcklwd xmm7,xmm2        ; xmm7 = CS3 | -S3 | CS2 | -S2 | CS1 | -S1 | CS0 | -S0
movdqa xmm8,xmm5           ; xmm8 = xmm5
punpckldq xmm5,xmm5        ; xmm5 = C1Q1| C1I1| C1Q1| C1I1| C0Q0| C0I0| C0Q0| C0I0
punpckhdq xmm8,xmm8        ; xmm8 = C3Q3| C3I3| C3Q3| C3I3| C2Q2| C2I2| C2Q2| C2I2
movdqa xmm9,xmm7           ; xmm9 = xmm7
punpckldq xmm7,xmm6        ; xmm7 = S1 | CS1 | CS1 | -S1 | S0 | CS0 | CS0 | -S0
punpckhdq xmm9,xmm6        ; xmm9 = S3 | CS3 | CS3 | -S3 | S2 | CS2 | CS2 | -S2
pmaddwd xmm5,xmm7          ; xmm5 = CI1 | CQ1 | CI0 | CQ0
pmaddwd xmm8,xmm9          ; xmm8 = CI3 | CQ3 | CI2 | CQ2
paddq xmm5,xmm8            ; xmm5 = CI3 + CI1 | CQ3 + CQ1 | CI2 + CI0 | CQ2 + CQ0
movdqa xmm6,xmm5          ; mov xmm5 to xmm6
punpckhdq xmm6,xmm5        ; move bits 64..127 of xmm5 into 0..63 of xmm6
pmovsxdq xmm5,xmm5        ; xmm5 = CI2 + CI0 | CQ2 + CQ0
pmovsxdq xmm6,xmm6        ; xmm6 = CI3 + CI1 | CQ3 + CQ1
paddq xmm15, xmm5          ; accumulate xmm5 to xmm15
paddq xmm15, xmm6          ; accumulate xmm6 to xmm15

```

In order to avoid performing correlation cross data transition point, two msec long buffers are created to correlate zero-phase IF data with zero-phase code and carrier tables for whole code period. Figure 6.12(a) shows the architecture of IF data buffer. When new data comes in, the oldest data will pop out, shift the second buffer to first one, and copy the

new data to second buffer. Figure 6.12(b) shows the correlation window which is set by current code phase. However, code phase can shift forward or backward as a result of Doppler frequency. The developed process divides the effect of code phase movement into three cases which are shown in Figure 6.12(c)(d)(e).

1. Forward phase to zero (phase at sample N-1 of 1st buffer and sample 0 of the 2nd buffer): perform two times correlation for first and second buffers.
2. Backward phase to N-1 (phase sample 0 of 2nd buffer): perform a correlation for second buffer and skip next correlation.
3. Others: perform a correlation across two buffers.

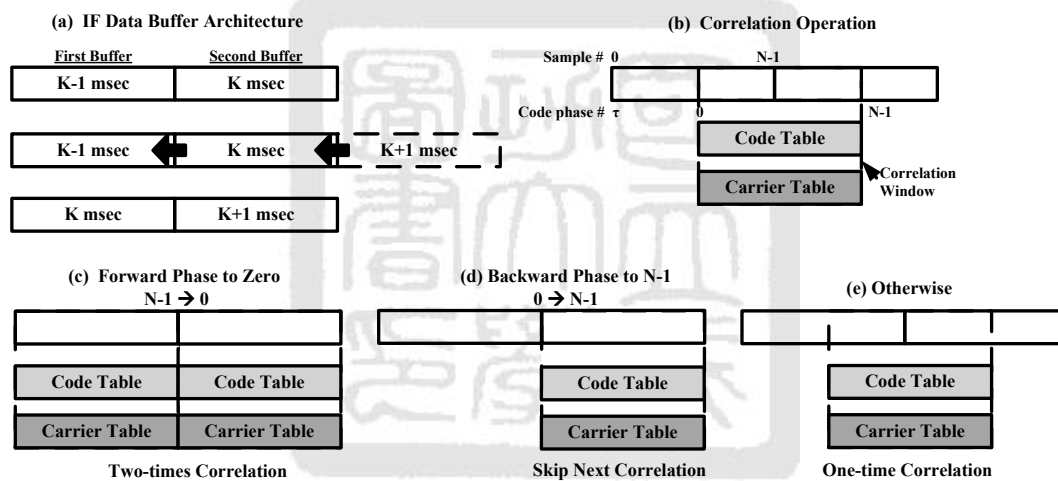


Figure 6.12 Buffer structure, correlation window and three cases for correlation

6.5.2.3 Acquisition/Tracking and Positioning

These functions adopt open source codes in [30] by replacing its interface which controls hardware correlator by our software correlator. Additionally, the WAAS signal processing and Viterbi decoder for messages are added to support positioning with the WAAS GEO. Figure 6.13 shows the structure of carrier tracking loop of software receiver for the specific formats of code and carrier table. First, the incoming data is shifted for setting correlation windows as Figure 6.12(b). Then, perform the correlation and its outputs

are rotated by a phase accounting for the zero-phase tables. The details of the loop are described in Section 3.1. The reference also derived the rotating phase only for zero-phase carrier table. Because zero-phase code table is also used in this architecture, there is a correction term to the rotating phase for code-phase changing. Equation (6.12) shows the rotation operation where $\Delta\tau$ stands for the correction term which is the difference time of shifting between previous and current correlation.

$$\begin{bmatrix} I_R \\ Q_R \end{bmatrix} = \begin{bmatrix} \cos(\Delta\theta) & \sin(\Delta\theta) \\ -\sin(\Delta\theta) & \cos(\Delta\theta) \end{bmatrix} \begin{bmatrix} I_{RS} \\ Q_{RS} \end{bmatrix} \quad (6.12)$$

$$\Delta\theta = \frac{(\omega_R - \omega)T_I}{2} + \omega(T_I - \Delta\tau)$$

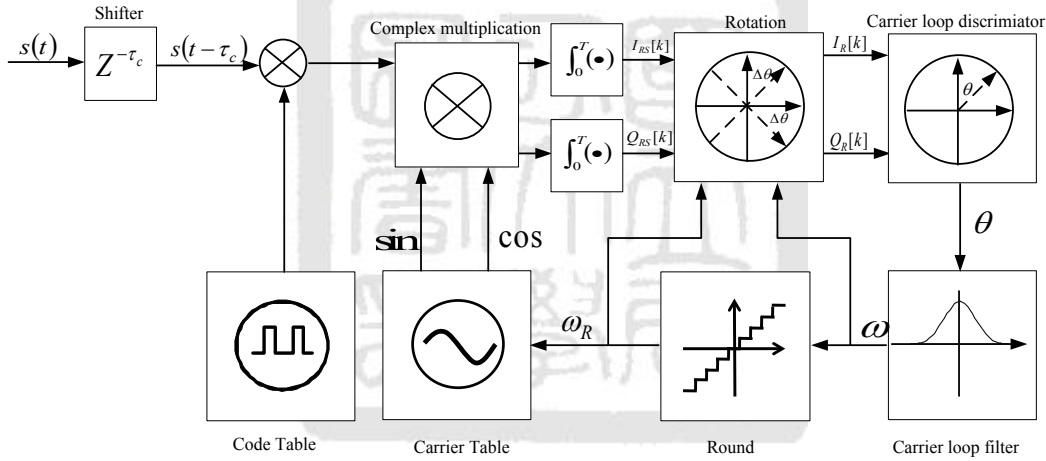


Figure 6.13 Architecture of carrier tracking loop of software receiver

6.5.2.4 ICP Initialization and Calculation of Differential ICP

The ICP is obtained by integrating product of IF carrier frequency and integration time shown in (6.13).

$$\phi[k] = \int_0^{t_k} \omega dt + \phi_0 \approx \sum_{i=0}^{k-1} \omega[i]T_I + \phi_0 \quad (6.13)$$

where ϕ_0 is the initial phase. From section 3.1, the phase of correlator output as (6.14)

$$\theta_{RS}[k] = \text{atan2}(I_{RS}[k], Q_{RS}[k]) = \frac{(\omega_R - \omega)T_I}{2} - \omega t_{k-1} - \phi_0 \quad (6.14)$$

Hence, the phase of incoming signal at time t_k is given by

$$\phi[k] = \omega t_{k-1} + \phi_0 + \omega T_I = -\theta_{RS}[k] + \frac{(\omega_R + \omega)T_I}{2} \quad (6.15)$$

The phase in (6.15) is subject to noise. A filter is used to get noise-free phase as shown in (6.16)

$$\phi[k] = \frac{k}{M}(\phi[k-1] + \omega[k-1]T_I) + \left(1 - \frac{k}{M}\right)\phi[k] \quad (6.16)$$

where M is the number of msec to perform the initialization. Afterwards, the ICPs are kept integrating phase and made the difference between antennas for every channel. The differential ICPs are averaged for several msec and then used to build the steering vectors as shown in (6.17).

$$\begin{aligned} \Delta\phi_n^l &= \sum_{i=1}^L (\phi_n^l[i] - \phi_1^l[i]) \\ t_n^l &= \exp(-j\Delta\phi_n^l) \end{aligned} \quad (6.17)$$

where L is the number of msec to average the differential ICP. l and n stand for l th channel and n th antenna. t_n^l is l th channel and n th antenna element of the steering vector.

6.5.2.5 Adaptive MVDR Beamforming

The adaptive MVDR beamforming function starts from computing the covariance matrix. The covariance matrix in (6.3) is a Hermitian matrix where $r_{ab} = \overline{r_{ba}}$, so only the elements of the upper triangle part need to be computed and the remaining elements can be established through the conjugation operation. For each element, the following operation is executed

$$r_{ab} = \frac{1}{N_s} \sum_{t=0}^{N_s-1} s_a[t] \cdot \overline{s_b[t]} \quad (6.18)$$

The computational complexity of (6.18) is high. Indeed, in the 4 antennas and 5 taps case, there are 210 elements to be computed. The computational load for each element is N_s complex 16-bit multiply-and-add operations. As the sampling rate of 4MHz, 4000

multiply-and-add operations need to be completed within 1 msec. To address this problem, two approaches are adopted to speed up the computation. First, use the multithreaded programming approach to separate operations to multiple threads for exploiting the resources of multi-core CPU. Second, program a SIMD assembly to accelerate the complex multiply-and-add operations. The following procedure describes the computation of covariance matrix.

1. Load the both of a and b components and store as a complex data
2. Perform complex multiplication by multiply-and-add
3. Accumulate the complex results

After computing the covariance matrix, the adaptation step size is decided by (6.9). Then, with inputs of covariance matrix, steering vector and adaptation step size, the adaptive procedure in equation (6.8) for each channel is performed to update its weight. The SIMD assembly code for calculating covariance matrix and compatible to SSE, SSE2, SSE3, and SSE4.1 is written as follows:

```

mov r10, -281466386841599;0xFFFF0001FFFF0001;

pinsrq xmm14,r10,0;

movddup xmm14,xmm14      ; xmm14 =  0xFFFF0001FFFF0001FFFF0001FFFF0001

movupd xmm0, [r8]        ; load A_IQ

movupd xmm1, [r9]        ; load B_IQ

movdqa xmm2,xmm0         ; move A_IQ to xmm4

                                ;          7      6      5      4      3      2      1      0
                                ; xmm2 =  Qa3 | Ia3 | Qa2 | Ia2 | Qa1 | Ia1 | Qa0 | Ia0

punpckldq xmm0,xmm0      ; xmm0 =  Qa1 | Ia1 | Qa1 | Ia1 | Qa0 | Ia0 | Qa0 | Ia0

punpckhdq xmm2,xmm2      ; xmm2 =  Qa3 | Ia3 | Qa3 | Ia3 | Qa2 | Ia2 | Qa2 | Ia2

movdqa xmm3,xmm1         ; move B_IQ to xmm3

                                ; xmm3 =  Qb3 | Ib3 | Qb2 | Ib2 | Qb1 | Ib1 | Qb0 | Ib0

pshufbw xmm3,xmm3,177    ; xmm3 =  Qb3 | Ib3 | Qb2 | Ib2 | Ib1 | Qb1 | Ib0 | Qb0

pshufbw xmm3,xmm3,177    ; xmm3 =  Ib3 | Qb3 | Ib2 | Qb2 | Ib1 | Qb1 | Ib0 | Qb0

psignw xmm3,xmm14        ; xmm3 = -Ib3 | Qb3 | -Ib2 | Qb2 | -Ib1 | Qb1 | -Ib0 | Qb0

movdqa xmm4,xmm1         ; move B_IQ to xmm4

punpckldq xmm4,xmm3      ; xmm4 = -Ib1 | Qb1 | Qb1 | Ib1 | -Ib0 | Qb0 | Qb0 | Ib0

punpckhdq xmm1,xmm3      ; xmm1 = -Ib3 | Qb3 | Qb3 | Ib3 | -Ib2 | Qb2 | Qb2 | Ib2

pmaddwd xmm4,xmm0        ;

pmaddwd xmm1,xmm2        ;

paddq xmm4,xmm1          ; xmm4 =      i3 + i1 | r3 + r1 | i2 + i0 | r2 + r0

movdqa xmm6,xmm4         ; mov xmm4 to xmm6

punpckhqdq xmm6,xmm4     ; move bits 64..127 of xmm4 into 0..63 of xmm6

pmovsxdq xmm4,xmm4       ; xmm4 =      i2 + i0          |      r2 + r0

pmovsxdq xmm6,xmm6       ; xmm6 =      i3 + i1          |      r3 + r1

paddq xmm15, xmm4         ; accumulate xmm4 to xmm15

paddq xmm15, xmm6         ; accumulate xmm6 to xmm15

```

6.5.2.6 Weight and Sum

The weight and sum function performs the combination of IF data from each antennas by equation (6.1). SIMD assembly is also used to implement the function within architecture shown in Figure 6.14 and the procedure of code is listed as follows.

1. Use the every weight value to make a complex format and duplicate two times
2. Load the IF data and also make the complex format for two samples
3. Perform complex multiplication by multiply-and-add
4. Accumulate the complex results

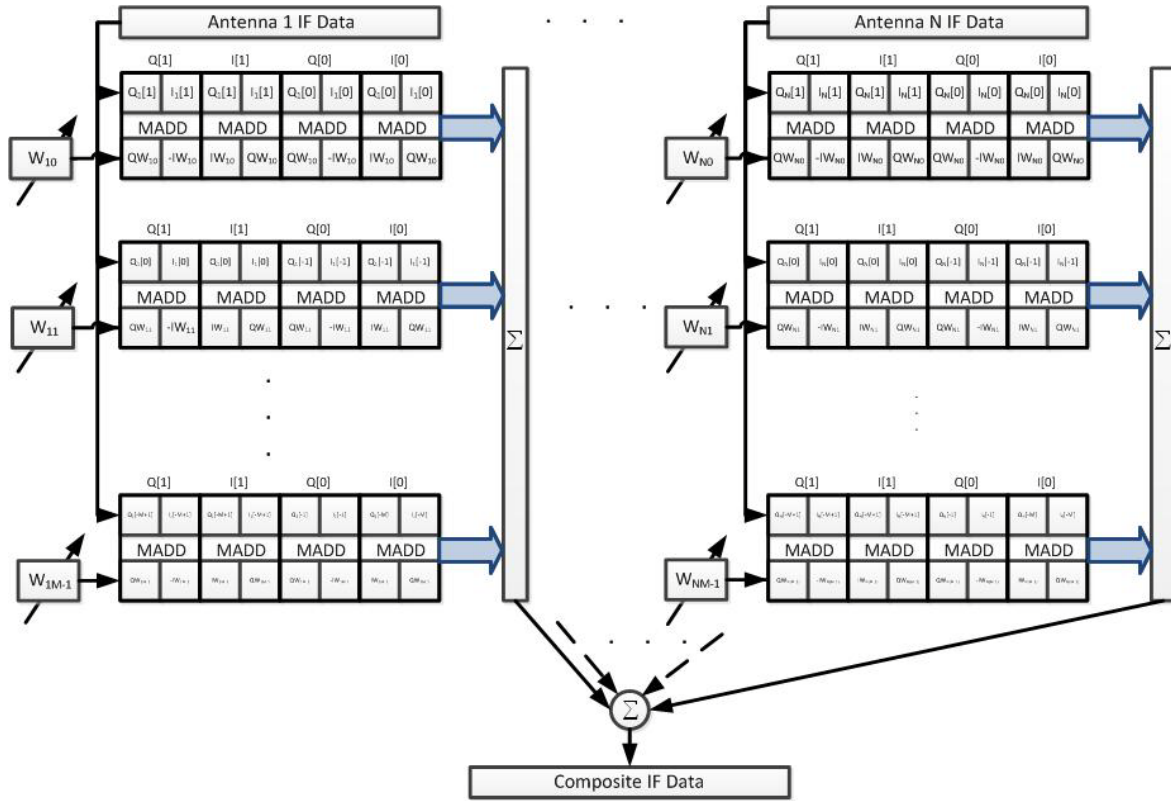


Figure 6.14 Architecture of weight and sum

6.5.2.7 Bias Calibration

In order to show the change of the angle-frequency responses with respect to different interference environments, the RF chain/cabling biases need to be calibrated before. The software receiver calibrates these biased after obtaining the differential ICPs and unit

vectors to satellites through positioning. The differential ICP for k th satellite between i th and first antenna is given by

$$\phi_i^k = \frac{2\pi\vec{p}_i \cdot \hat{r}^k}{\lambda} + \Delta\gamma_i + N_i^k + \varepsilon_i^k \quad (6.19)$$

where N_i^k is the integer associated to ϕ_i^k and ε_i^k is the phase error. With known baseline vectors, the fractional part of bias can be estimated by

$$fr(\Delta\gamma_i) = \frac{1}{K} \sum_{k=1}^K \text{mod}(\phi_i^k - \frac{2\pi\vec{p}_i \cdot \hat{r}^k}{\lambda}, 2\pi) \quad (6.20)$$

where K is the number of channels. Only the fractional part of bias is required to calculate the response because exponential function of integer part is equal to one.

6.5.2.8 Calculation of Angle-Frequency Response

The response of STAP in the direction ϕ, θ and frequency f is given by

$$GP(\phi, \theta, f) = \sum_{n=1}^N \sum_{m=0}^{M-1} w_{nm} \times \exp \left[j \left(\frac{2\pi\vec{p}_n \cdot \hat{r}(\phi, \theta)}{\lambda} + \Delta\gamma_n - 2\pi f m T_s \right) \right] \quad (6.21)$$

where T_s is the sample time. There are three dimensions of the response in which two of them for spatial domain and one for spectral domain. If one would like to show the spatial response such as the gain pattern, fix the frequency and illustrate the gain value of two direction angle on the polar plot. The gain pattern is used to show the spatial performance of beamforming where directional beams and nulls are obvious to see. Additionally, assuming that one of directional angle of interference is known, one fixes the directional angle and shows the gain of the other directional angle versus frequency called as angle-frequency response. The angle-frequency response is used to simultaneously show the spatial and spectral performance of STAP where narrowband notch filter and broadband null steering are obvious to see.

6.5.2.9 Multi-Threaded Programming for Real-time Validation

The system program schedules the threads to execute the software receiver for fully exploiting the resource of CPU. Figure 6.15 shows the plan of the execution flow for using 4 antennas and 5 taps. To reach real-time capability, the execution flow must be finished within 1 msec. The *MultiAntsThread* thread masters the whole flow to synchronize the threads using event objects. At first, *MultiAntsThread* sets events to receiver threads and adaptation thread for initiating the individual receiver operation and weight updating. For composite signal, the *RcvrThread* performs the weight-and-sum using weight from previous iteration, software correlator, acquisition/tracking and positioning. The positioning result will be given in this thread. For each single antenna, each *RcvrWoNavThread* only performs software correlator and acquisition/tracking. Only ICPs are measured in these threads. For adaptation, the *AdaptApplebaumThread* firstly sets events to multiple *CalculateCovarThread* threads to calculate the elements of covariance matrix, and then perform the MVDR algorithm to update the weights. Finally, the *MultiAntsThread* makes the ICPs differential form build the steering vectors.

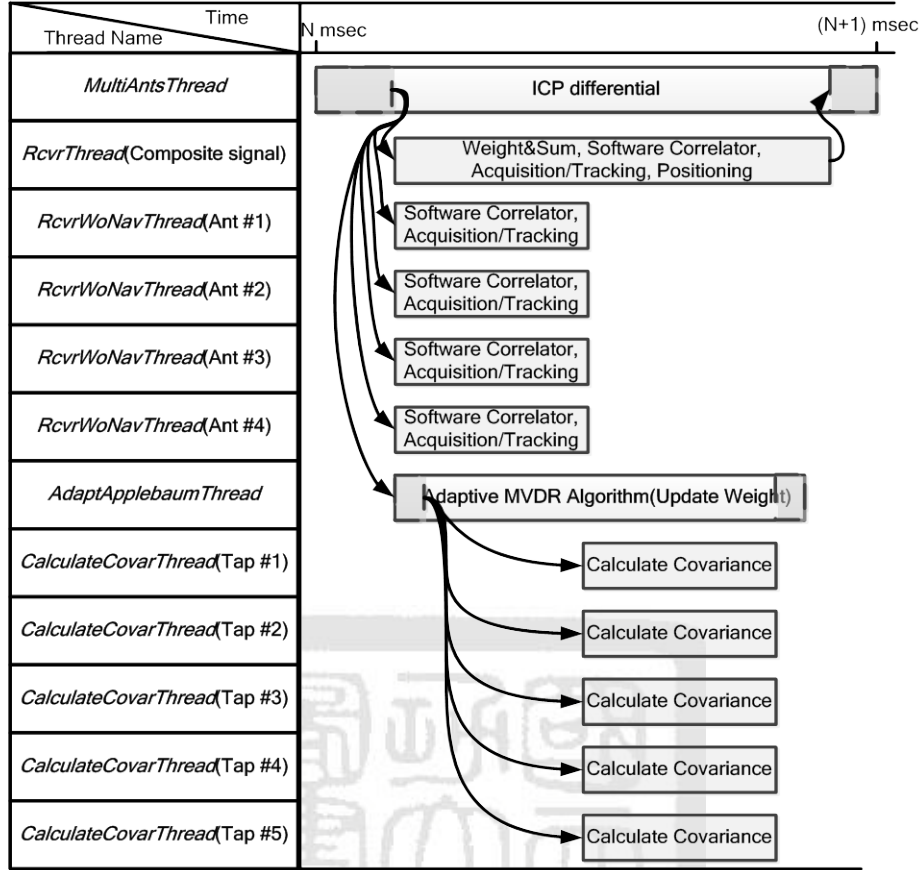


Figure 6.15 Plan of execution flow of software program within 1 msec

6.6 Calibration of Antenna Array by Carrier Phase Precise Positioning

Although the software receiver can act as a CRPA in real-time without any prior knowledge, the calibration of the antenna array in the post-processing mode will aid in examining the CRPA performance of receiver. The gain pattern of the composite antenna is critical to show the performance of CRPA. The gain is calculated using relative positions of antennas and weights as shown in equation (6.22).

$$GP(\varphi, \theta) = \sum_{i=1}^M w_i \times \exp \left[j \left(\frac{2\pi \vec{p}_i \cdot \hat{r}_i(\varphi, \theta)}{\lambda} + \Delta\gamma_i \right) \right] \quad (6.22)$$

where baseline vector \vec{p}_i and cabling bias $\Delta\gamma_i$ are obtained through calibration. Figure 6.16 shows the used calibration procedure of the antenna array. In the CRPA software receiver, the ICP measurements are collected from the channels of the individual antennas

for one minute. The azimuth/elevation of satellite is also obtained using the beam formed composite signal. The single difference ICP between signals of antennas and reference antenna j is represented as [56]:

$$\phi_{ij}^k = \lambda^{-1} r_{ij}^k + \delta L_{ij} + N_{ij}^k + \varepsilon_{ij}^k \quad (6.23)$$

where r_{ij} is differential range toward the k satellite between i th and j th antenna, N_{ij}^k is the integer associated to ϕ_{ij}^k , ε_{ij}^k is the phase error. The double difference ICP between satellites and reference satellite l is represented as:

$$\phi_{ij}^{kl} = \lambda^{-1} r_{ij}^{kl} + N_{ij}^{kl} + \varepsilon_{ij}^{kl} \quad (6.24)$$

The cable length difference term is subtracted in the double difference. Based on the distance of the antenna position close to one wavelength, equation (6.24) can be written as:

$$\phi_{ij}^{kl} = \lambda^{-1} \left(-\hat{r}^k - (-r^l) \right) p_{ij} + N_{ij}^{kl} + \varepsilon_{ij}^{kl} \quad (6.25)$$

where \hat{r}^k is the unit vector to satellite k , p_{ij} is baseline vector between i th and j th antenna. By combining all the double difference measurements of the pair ij th antennas, the observations equation is represented as:

$$\begin{bmatrix} \phi_{ij}^{21} \\ \phi_{ij}^{31} \\ \vdots \\ \phi_{ij}^{K1} \end{bmatrix} = \lambda^{-1} \begin{bmatrix} -\hat{r}^2 - (-r^1) \\ -\hat{r}^3 - (-r^1) \\ \vdots \\ -\hat{r}^K - (-r^1) \end{bmatrix} p_{ij} + I_{K-1} \cdot \begin{bmatrix} N_{ij}^{21} \\ N_{ij}^{31} \\ \vdots \\ N_{ij}^{K1} \end{bmatrix} + \begin{bmatrix} \varepsilon_{ij}^{21} \\ \varepsilon_{ij}^{31} \\ \vdots \\ \varepsilon_{ij}^{K1} \end{bmatrix} \quad (6.26)$$

$$\Gamma = \lambda^{-1} G p_{ij} + N + E$$

From the positioning results of composite channels, the azimuth and elevation of satellites are used to manipulate matrix G . Before solving the relative antenna positions, the integer vector N needs to be resolved. The LAMBDA method, specified in reference [6], is used. Finally, the cabling biases are calculated by (6.20).

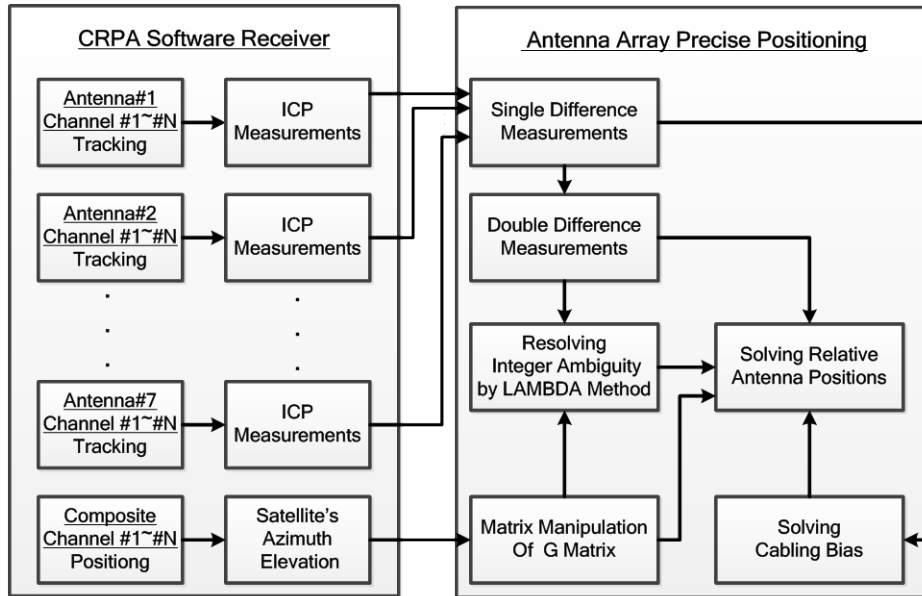


Figure 6.16 Block diagram of the calibration procedure of antenna array

6.7 Analysis of Thread Activities and Timing Performance

6.7.1 Implementation I

In order to analyze the activities of the threads of the CRPA real-time software receiver, the Intel Thread Profiler collector program is utilized. This is a thread analyzer application to understand the threading patterns in multi-threaded software. This collector program is executed with our software receiver and outputs a timeline diagram containing execution flow of all the threads as shown in the Figure 6.17. The resulting execution flow exactly follows the designed flow shown in Figure 6.6.

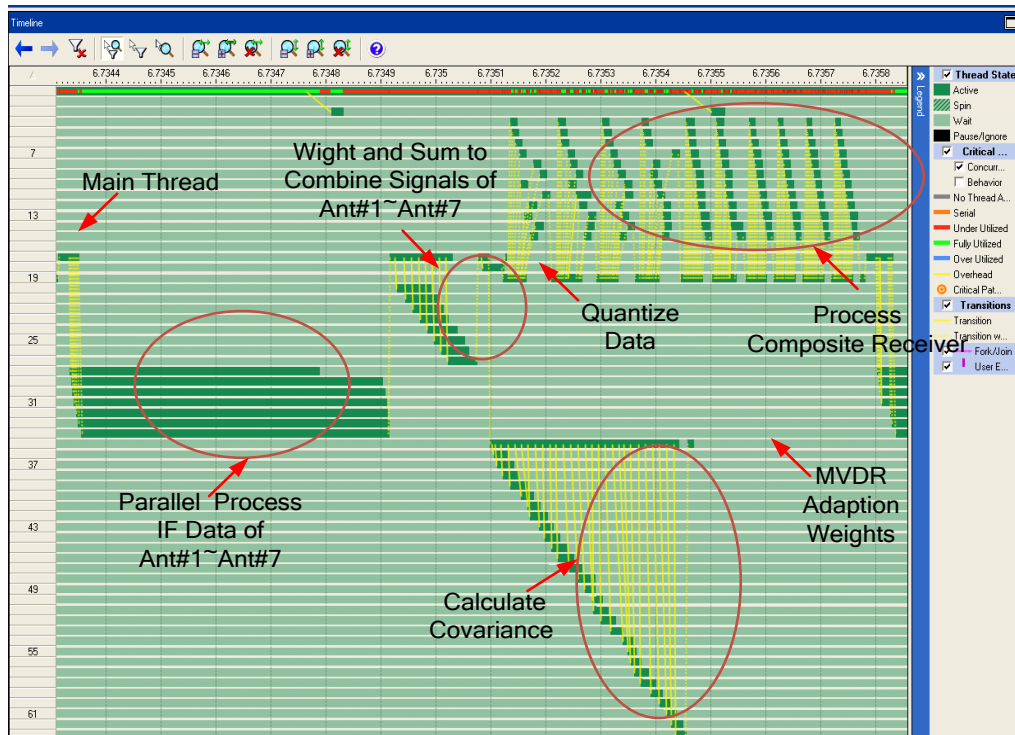


Figure 6.17 Execution flow of the threads of the CRPA real-time software receiver collected from the Intel Thread Profiler collector program

The execution time of the threads is measured by counting the clock cycles of CPU. The threads needed to execute in one msec are divided into three parts and execution times is measured 1000 trials. The results are represented as a box plot in the Figure 6.18. The mean execution times of each part are listed in the Table 6.3. The total mean execution time is less than one msec and shows that the CRPA software receiver can achieve real-time capability.

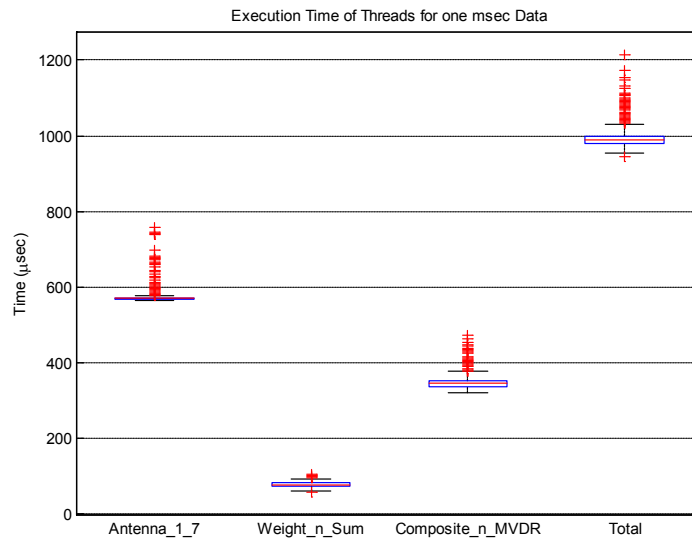


Figure 6.18 Box plot of the execution time illustrating the real time capability of the CRPA

Table 6.3 Execution time of threads

Thread Name	Mean Execution Time (μsec)
Process IF Data of Antenna #1 ~ #7	572.93
Weight and Sum	75.23
Composite Receiver and MVDR weight update	345.81
Total	993.98

6.7.2 Implementation II

For implementation II, the performance profiler “Intel VTune Amplifier XE” is utilized. Taking 4 antennas and 5 taps for example, the snapshot of the execution flow is shown in the Figure 6.19. This execution flow exactly follows the designed flow shown in Figure 6.15. The measured duration for processing one msec data is 0.9 msec.



Figure 6.19 Execution flow of the software receiver collected from Intel VTune Amplifier XE

The execution time of the threads is measured by counting the clock cycles of CPU. The duration for processing one msec data is measured for 100,000 trials. The results are represented as a histogram shown in Figure 6.20. The mean duration is about 0.8 msec. More than 99.9% of the duration is less than 1 msec. This shows that the CRPA software receiver can achieve real-time capability.

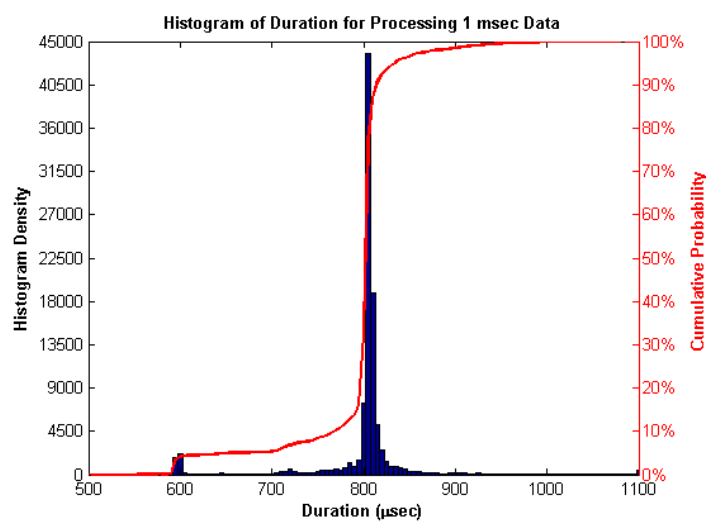


Figure 6.20 Histogram of duration to illustrate the real time capability of software receiver

6.8 Experimental Results

6.8.1 Implementation I

Two experiments are conducted in open field (low multipath environment) to examine the performance of CRPA. Figure 6.21 shows the hardware setup of the software receiver.



Figure 6.21 Hardware setup of the software receiver

At first, one runs the software receiver for 60 seconds for recording the ICP measurements and azimuth/elevation of satellites. Then, the calibration of antenna array using carrier phase precise positioning is performed and the result is shown in Figure 6.22. It is close to the physical antenna array geometry.

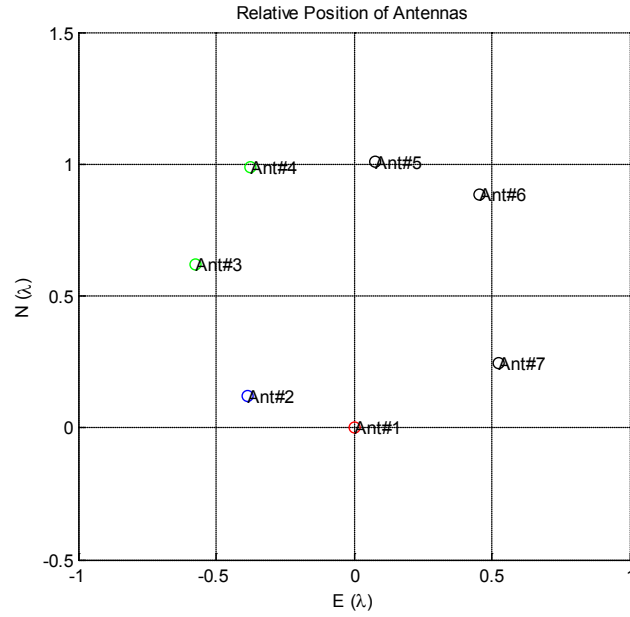


Figure 6.22 Relative position of the antennas from the carrier phase precise positioning

The first experiment is conducted to demonstrate that the C/No is enhanced by array processing. Then, the second experiment is to test the software receiver in the presence of low power CW and CDMA interferences by an injected interference source. Comparisons are made between a single antenna, CRPA by deterministic beamforming, and MVDR adaptive beamforming.

6.8.1.1 Experiment I – Enhancing C/No

In order to determine the beam formed antenna gain pattern for each satellite in view, the software receiver runs in post-processing mode. In each run, the receiver performs CRPA by MVDR adaptive beamforming toward one of satellites. Figure 6.23 shows the filtered C/No of all satellites in view. The CRPA begins to perform from 30th second. There is more than 6dB gain through the CRPA for all satellites tested.

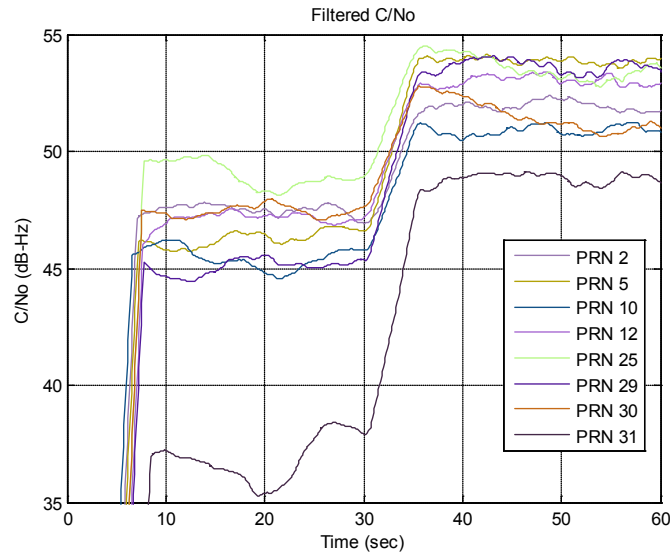


Figure 6.23 Filtered C/No of the software receiver with and w/o CRPA beam steering

Figure 6.24 shows the resulting gain patterns toward the selected satellite. The corresponding sky plot is shown in the center of the figure. There is a high gain in the direction of satellite that is the reason why the C/No is enhanced by the CRPA processing.

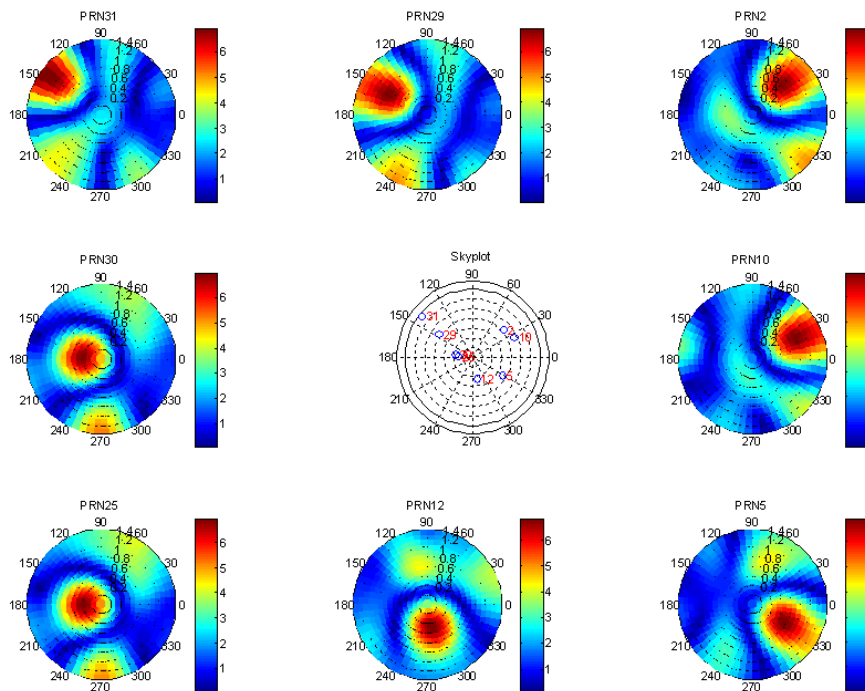


Figure 6.24 Gain patterns of the composite antennas by the CRPA toward the specified satellite in a sky plot format

6.8.1.2 Experiment II – Rejecting Interference

In order to examine the interference rejection performance of the CRPA software receiver, a single channel GPS simulator provides injected, via splitters, interference which is combined with received signal from four elements of the antenna array. The hardware set-up is depicted in Figure 6.25. The GPS simulator generates two types of interference. One is CDMA interference obtained by setting a PRN number which is currently unallocated. The other is CW interference by turning off C/A code spreading. Figure 6.26 shows the power spectral density of IF signal for three cases: w/o interference, with CDMA interference, and with CW interference. The spectrum of the signal with CDMA interference has a higher power lobe within 2 MHz bandwidth than no interference case. The spectrum of signal with CW interference has a peak in the center frequency of IF.

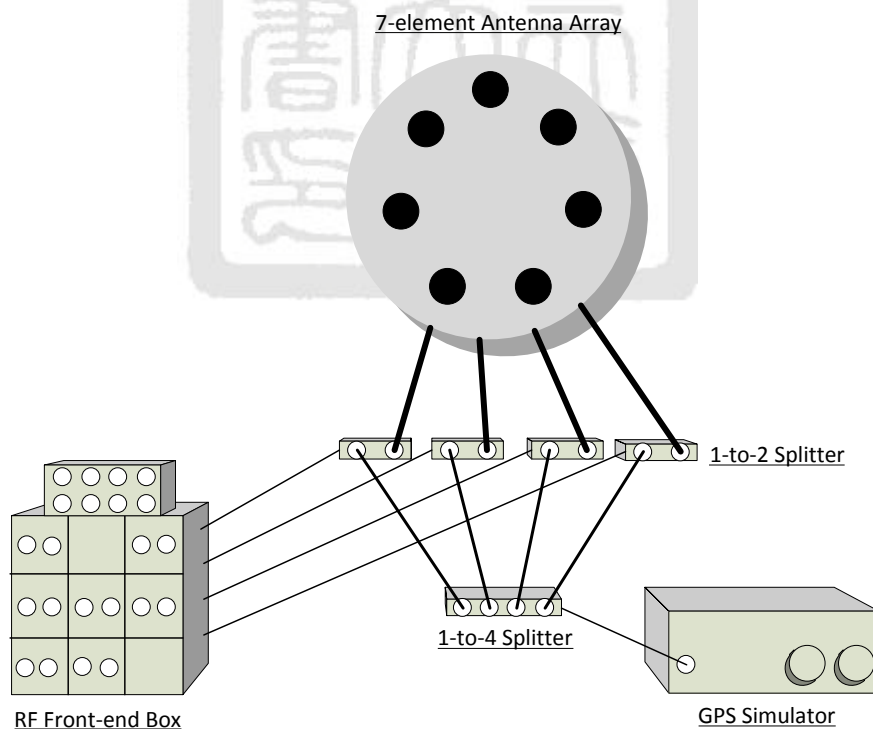


Figure 6.25 Diagram of the hardware set-up of the CRPA software receiver with the interference using GPS simulator

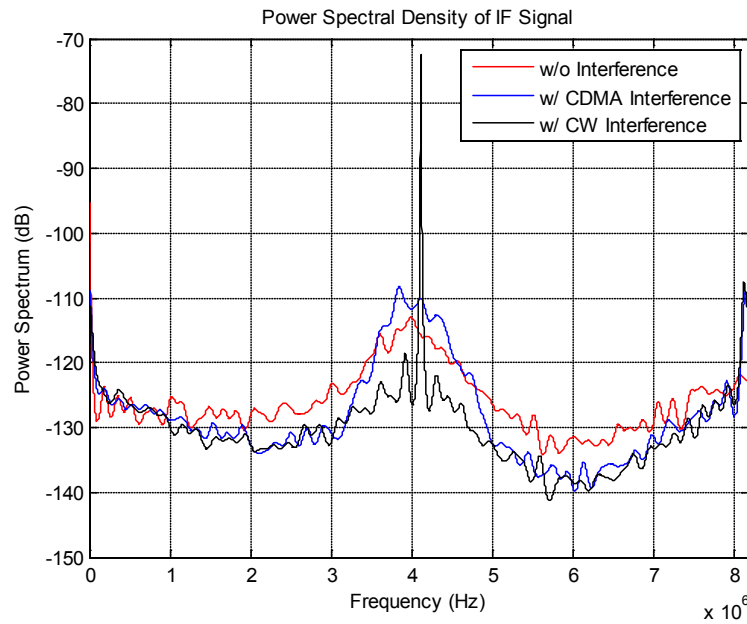


Figure 6.26 Power spectral density of IF signal with and without interference

The CRPA software receiver processes these signals in the post-processing mode and only combines the four signals antennas which are subject to the interference. Figure 6.27 shows filtered C/No of the PRN 2 in the presence of CDMA and CW interferences. In both cases, the interference starts 30 seconds into the run. Without CRPA, the software receiver will lose lock on the PRN 2 signal when interference is present. The software receiver performs implements the CRPA at 10th second and contributes over 5 dB gain on C/No. When interference is not present, the performance of MVDR is close to deterministic beamforming. However, after interference is present, MVDR has gain about 0.5 dB higher than deterministic beamforming. It should be noted that these are really simple tests of null steering and meant to verify that the algorithms performing correctly.

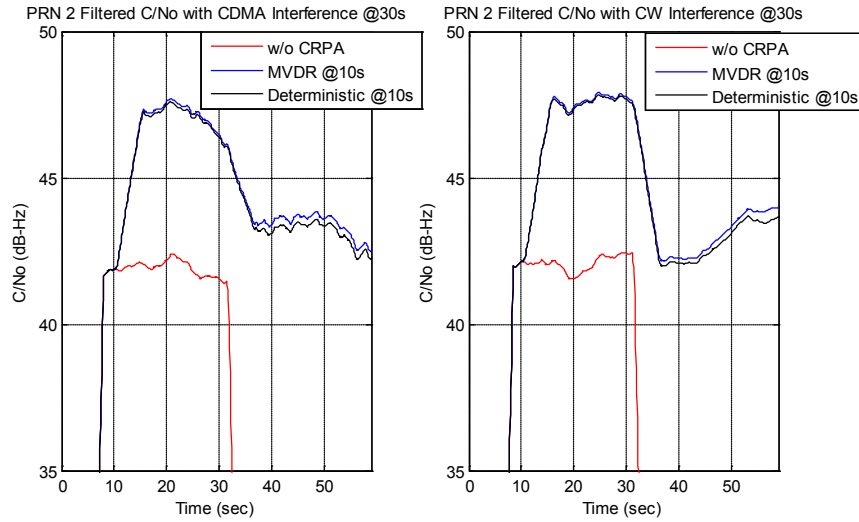


Figure 6.27 Filtered C/No of the software receiver performing CRPA by MVDR and deterministic beamforming in the cases with CDMA and CW interference

6.8.2 Implementation II

A dataset is collected by signal collection hardware in open field (low multipath environment) to examine the interference rejection performance of CRPA software receiver. Figure 6.28 shows the geometry of antenna array which is arranged as unitary rectangle array with about 1.3 wavelength for L1. The reason for setting the distance is that the physical size of Trimble Zephyr antenna is close to one wavelength. Because the distance is greater than one wavelength, there are multiple grating lobes in the gain pattern. The grating lobes will affect the interference rejection performance even though the direction of the satellite is not close to interference. This effect will be shown later in this section.

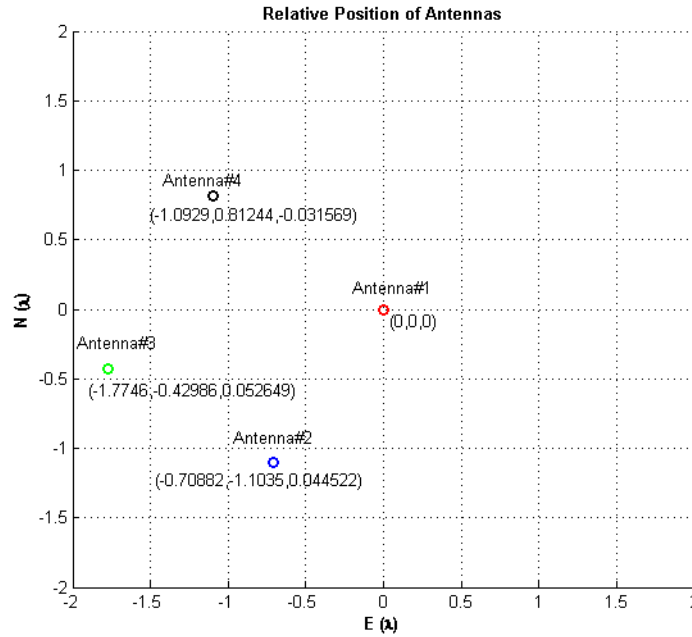


Figure 6.28 Relative position of the antennas of data collection

For the interference, a MATLAB script was written to generate the synthetic interferences and combine them with original dataset. The direction and signal specification of interferences can be specified in the script. Two scenarios are listed as follows.

1. Single broadband interference with high Interference to Signal ratio (I/S)
2. Multiple interferences containing combination of broadband and narrowband ones with moderate I/S

6.8.2.1 Scenario I – Single High I/S Interference

A single interference specified with 40 dB I/S, 5° elevation angle, and 0 ° azimuth angle is simulated to examine our software receiver performance. The interference is added into dataset starting from 41st second. The software receiver simultaneously tracks twelve GPS/WAAS satellite and steers twelve beams toward the each satellite. The left figure of Figure 6.29 shows the filtered C/No of all-in-view CRPA. Most of channels are still in

tracking besides PRN 7 and PRN 16 when high I/S ratio interference is on. And, the right figure of Figure 6.30 shows the filtered C/No of three WAAS GEOs used to compare between CRPA and single antenna. Before turning on the interference, there is about 5 dB gain in C/No of CRPA over single antenna. When interference is on, the C/No of single antenna drop more than 10 dB and the channels of CRPA keep tracking without losing much C/No.

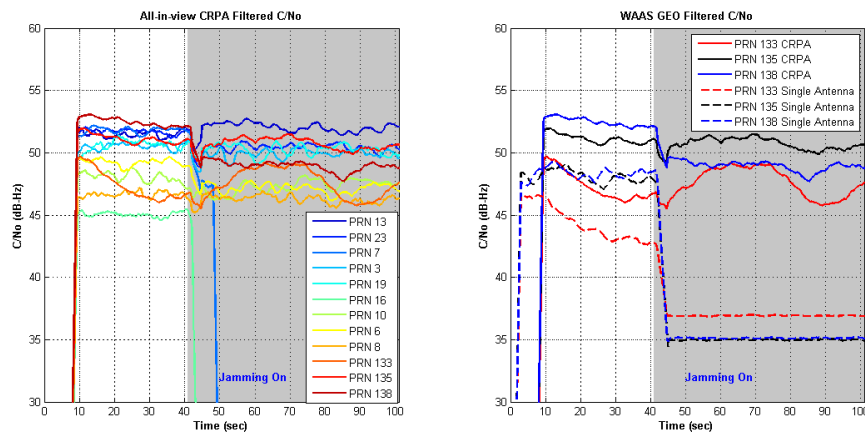


Figure 6.29 Filtered C/No of all-in-view CRPA and WAAS GEOs

The reason why the channels of PRN 7 and PRN 16 lose track is the effect of the grating lobes. This effect can be seen from the gain patterns of antenna array. Figure 6.30 shows the gain patterns for three satellites without and with interference as well as sky plot. The gain patterns of PRN 8 in two cases have the same beam in the direction of satellite and deep null in the direction of interference. However, for the PRN 7 and PRN 16, due to the grating lobes, there are beams toward the direction of the satellite and the interference which can be seen in the gain pattern without interference. When interference is on, the adaptive MVDR algorithm will make a null in the direction of interference and then squeeze the original beam in the direction of satellite to the other direction. The constraint of equation (6.2) still works, so the gain in the direction of satellite is maintained at one which can be seen in the gain patterns of PRN 7 and PRN 16 with interference. The

resulting beam has a gain more than 2 and toward the other direction instead of the direction of satellite. This effect will degrade the interference rejection performance especially for high I/S scenarios.

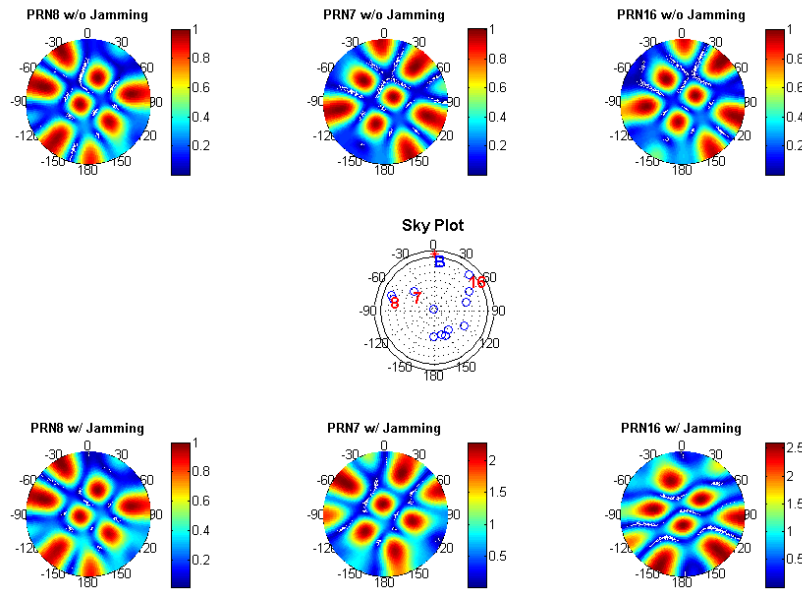


Figure 6.30 Gain patterns of the composite antennas without and with interference and sky plot showing the directions of the satellite and interference

Figure 6.31 shows earth-north (EN) plots with and without interference. There are two main clouds in the EN plot for interference one due to losing tracking in two channels.

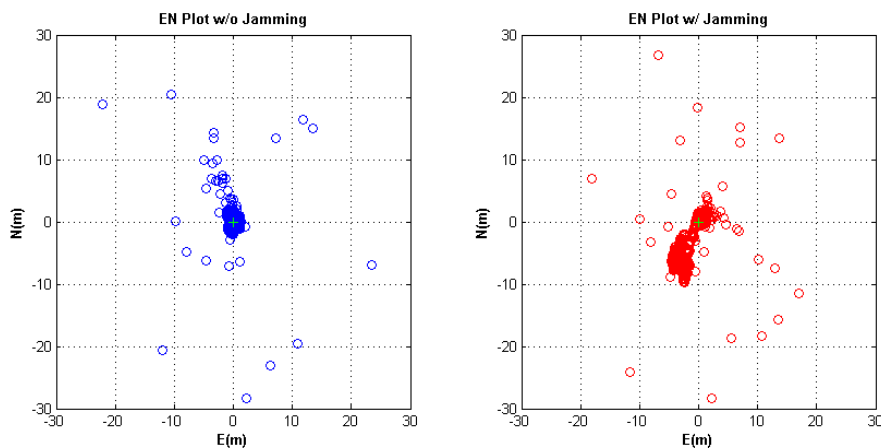


Figure 6.31 Positioning result of the software receiver without and with interference

6.8.2.2 Scenario II – Multiple Interferences

In order to examine the resistance of our software receiver to multiple interferences, several interferers are added to the data set as indicated by the sequence shown in Table 6.4.

Table 6.4 Sequence of adding multiple interferences

Sequence	Type	I/S	IF	Direction
1B	Broadband	30 dB	0.42 Hz	Azimuth: 0° Elevation: 5°
2B	Broadband	30 dB	0.42 Hz	Azimuth: 60° Elevation: 5°
3N	Narrowband	30 dB	0.42 MHz	Azimuth: 120° Elevation: 5°
4N	Narrowband	30 dB	1.42 MHz	Azimuth: 180° Elevation: 5°
5N	Narrowband	30 dB	-1.58 MHz	Azimuth: -120° Elevation: 5°
6N	Narrowband	30 dB	-0.58 MHz	Azimuth: -60° Elevation: 5°

Using a four-element array, the maximum number of interferences that can be nulled using spatial adaptation is three. In this scenario, two broadband and four narrowband interferences will be appearing in the end of dataset for examining the performance of STAP. Figure 6.32 shows the sky plots, gain patterns, and angle-frequency responses of four cases, where the gain patterns and the angle-frequency are averaged over all the 12 channels for representing common nulls in the plots. The angle-frequency responses are represented as azimuth versus frequency by fixing the elevation angle as 5°. Each column stands for each case when the interferences are shown in the sky plot. In the first column, there is no interference, so no obvious nulls are present. In the second column, when two broadband interferences are added, two deep nulls appear in the direction of the interferences in the gain pattern, respectively. And, there are two broadband notches in the azimuth angle 0 and 60° shown in the angle -frequency response. In the third column, when the narrowband interferences are added, there are no additional nulls in the direction of narrowband interferences. However, there are two narrowband notches in the

angle-frequency response. This results from adaptation using the time taps which makes notches in the frequency domain instead of steering nulls in the spatial domain. It is the reason why the number of interferences can be resisted can be more than the number of element in the antenna array minus one. In the end, when all the interferences are on, there are two broadband nulls and four narrowband notches shown in the gain pattern and response of the forth column. Figure 6.33 shows all-in-view CRPA filtered C/No when adding multiple interferences. Only the channel of PRN 16 loses track due to the effect of grating lobes. The other channels keep track throughout sequence. This shows our software receiver capable of resisting multiple interferences.

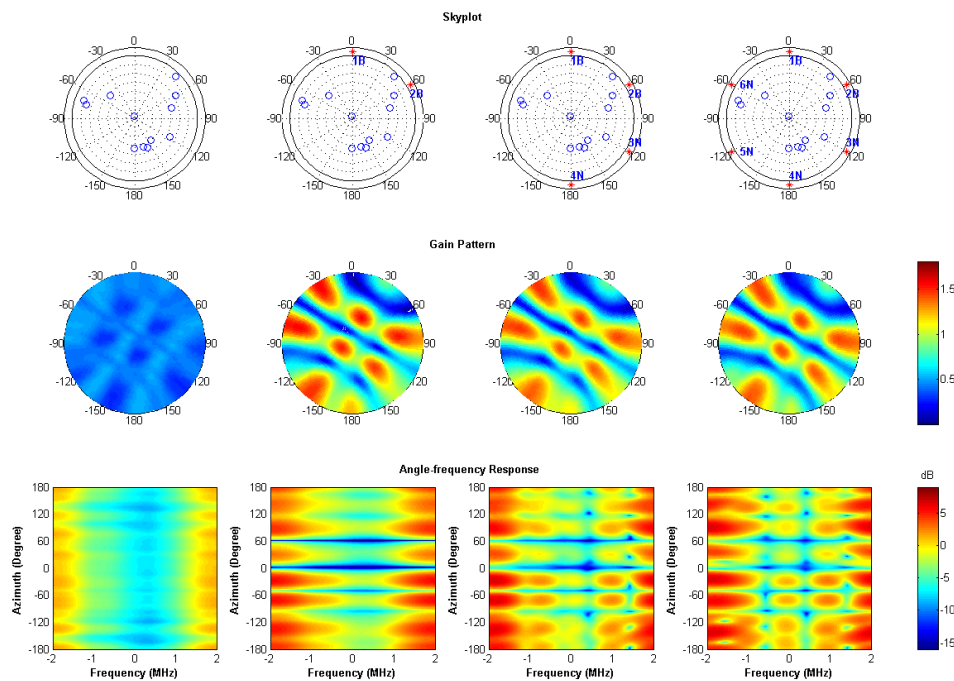


Figure 6.32 Sky plots, gain patterns and angle-frequency responses of four cases when adding multiple interferences

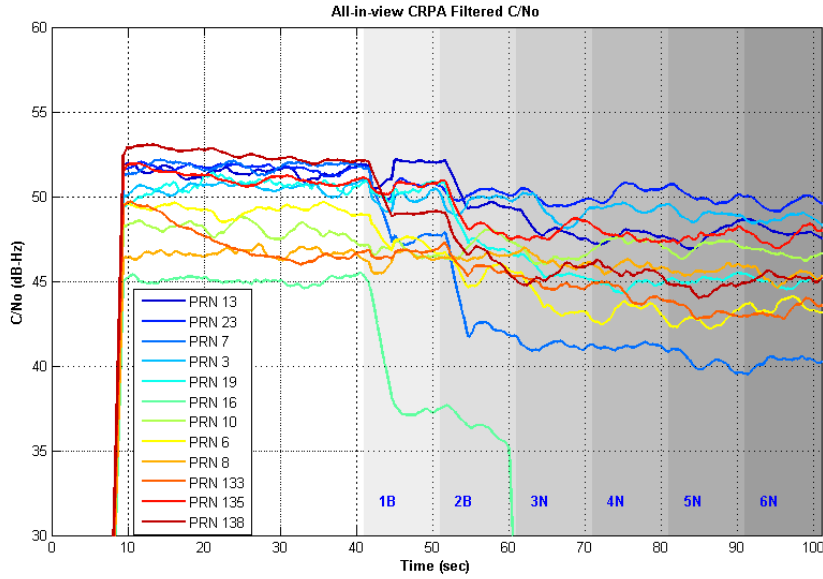


Figure 6.33 All-in-view CRPA filtered C/No when adding multiple interferences

6.9 Summary

This chapter describes two implementations of a software receiver for GPS/WAAS L1 C/A signals to demonstrate the feasibility of CRPA technology for civil applications. The developed approach performed complete CRPA processing without any prior knowledge. A mechanism is built to determine the steering vectors without prior knowledge of antenna geometry. The architecture of STAP is constructed by a software approach to implement CRPA. The optimum weights are determined by the adaptive MVDR algorithm to give the interference rejection performance capable of rejecting multiple interferences. The components of software receiver are implemented efficiently using the SIMD assembly and C++ multithread programming. In particular, one optimized SIMD assembly to develop fast correlation process of high resolution (16 bits) samples. This demonstrates the capability of the software receiver to perform all-in-view beamforming up to twelve channels in real-time. The angle-frequency response is also included to illustrate the interference rejection performance of receiver. The results of the experiments demonstrate interference rejection capability in the presence of 40 dB I/S or six interferences.



Chapter 7

Conclusions

In this dissertation, GNSS receiver is designed by software radio concept and is implemented by DSP/FPGA-based and PC-based approaches. Both approaches are examined by receiving real GNSS signals and the real-time execution capabilities are demonstrated. In order to achieve real-time capability, the functions of receiver are coded by parallel programming in two techniques. The tasks of receiver are decomposed in parts and the processes are implemented through multithreading. In addition, assembly codes are developed by using bit-wise and SIMD instructions to speed up the execution. Moreover, a method is developed to account for data intermittency in a GNSS software receiver. The developed software receiver is applied to cope with scintillation, dual-band processing, and interferences.

Based on multi-correlator architecture, an adaptive code discriminator is designed to reduce the risk of false lock on side peaks and achieve the optimal performance of tracking error and multipath. A DSP/FPGA-based software receiver implements this code discriminator and examines its performance by receiving real GIOVE-A signal.

A particle filter based estimator is designed to keep tracking signal against severe scintillation effect. In this estimator, a Gauss-Markov model is used to formulate variation of amplitude and a navigation data detector is realized by merging weight update stages of particle filter. To examine the proposed approach, an existing scintillation model and designed estimator are simulated. The result shows that the particle filter based approach has better performance than conventional PLL during scintillation.

A dual-band L1/L5 software receiver is implemented by PC-based approach. This work is potentially the first GPS receiver capable of L5-only positioning, and solves some significant issues with using the current non-homogeneous L5 constellation and signal broadcast. Two L1/L5 assistance mechanisms are used to shorten the acquisition time and increase sensitivity of L5 signal. This receiver uses three types of pseudoranges, L1-only, L5-only, and L1/L5 combination to perform positioning. The positioning results show that the L5-only type has the best performance in the regular days.

A CRPA software receiver is implemented by PC-based approach. Two kinds of implementation are made and both have real-time capability. The first one is 7-element, 2-bit resolution, 10-channel and single-beam beamforming and the second one is 4-element, 16-bit resolution, 12-channel and all-in-view beamforming. By using STAP structure and MVDR algorithm, the CRPA software receiver can steer a null in the spatial domain and make a notch in the frequency domain at same time. Experimental results show that the CRPA software receiver is capable of rejecting interferences with high interference-to-signal ratio (I/S) and multiple sources from different locations.

Future works is to develop a complete platform for CRPA software radio for the time transfer and time synchronization. The U.S. Federal Aviation Administration (FAA) Alternative Position Navigation and Timing (APNT) study is interested in the use of the CRPA with the L5 signal for robust time transfer [19]. This platform could be extended to realize multi-constellation and multi-band GNSS receiver which best fits the software radio concept. The other future work is to implement the particle filter based tracking by PC-based software and examine it under the scintillation environment when the next solar maximum will occur in 2013.

References

- [1] D. M. Akos, *A Software Radio Approach to Global Navigation Satellite System Receiver Design*, Ph.D dissertation, Ohio University, 1997.
- [2] D. M. Akos, P. L. Normark, A. Hansson, A. Rosenlind, C. Ståhlberg, and F. Svensson, "Global Positioning System Software Receiver (gpSrx) implementation in low cost/power programmable processor", in *Proceedings of the 14th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GPS 2001)*, pp. 2851-2858, 2001.
- [3] S. P. Applebaum, "Adaptive arrays," *IEEE Transactions on Antennas and Propagation*, Vol. 24, No. 5, pp. 585-598, 1976.
- [4] J. Arribas, D. Bernal, C. Fernández-Prades, P. Closas, and J. A. Fernández-Rubio, "A novel real-time platform for digital beamforming with GNSS software defined receivers," in *Proceedings of the 22nd International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS 2009)*, pp. 2329-2345, 2009.
- [5] M. S. Arulampalam, S. Maskell, N. Gordon and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Transactions on Signal Processing*, Vol. 50, No. 2, pp. 174-188, 2002.
- [6] S. Backén, D. M. Akos, and M. L. Nordenvaad, "Post-processing dynamic GNSS antenna array calibration and deterministic beamforming," in *Proceedings of the 21st International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2008)*, pp. 1311-1319, 2008.
- [7] P. A. Bello and R. L. Fante, "Code tracking performance for novel unambiguous M-code time discriminators," in *Proceedings of the 2005 National Technical Meeting of The Institute of Navigation (ION NTM 2005)*, pp 293-298, 2005.
- [8] J. W. Betz and K. R. Kolodziejwski, "Extended theory of early-late code tracking for a bandlimited GPS receiver," *Navigation: Journal of The Institute of Navigation*, Vol. 47, No. 3, pp 221-226, 2000.
- [9] J. W. Betz, "Binary offset carrier modulations for radionavigation," *Navigation: Journal of The Institute of Navigation*, Vol. 48, No. 4, pp 227-246, 2001.
- [10] D. Bobyn, A. J. Van Dierendonck, H. Kroon, M. Clayton, and P. Reddan, "A prototype WAAS (SBAS) L1/L5 signal generator," in *Proceedings of the 16th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2003)*, pp. 2760-2768, 2003.

- [11] M. S. Braasch and A. J. Van Dierendonck, "GPS Receiver Architectures and Measurements," *Proceedings of the IEEE*, Vol. 87, No. 1, pp. 48-64, 1999.
- [12] E. Buracchini, "Software radio concept," *IEEE Communications Magazine*, Vol. 38, No. 9, pp. 138-143, 2000.
- [13] Y. H. Chen, J. C. Juang, and T. L. Kao, "Implementation of BOC signal acquisition using a DSP/FPGA board," in *Proceedings of the 12th IAIN World Congress, 2006 International Symposium on GPS/GNSS*, Vol. 2, pp. 405-410, 2006.
- [14] Y. H. Chen and J. C. Juang, "A GNSS software receiver approach for the processing of intermittent data," in *Proceedings of the 20th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2007)*, pp. 2772-2777, 2007.
- [15] Y. H. Chen, J. C. Juang, and T. L. Kao, "Robust GNSS signal tracking against scintillation effects: a particle filter based software receiver approach," in *Proceedings of the 2010 International Technical Meeting of The Institute of Navigation (ION ITM 2010)*, pp. 627-635, 2010.
- [16] Y. H. Chen, J. C. Juang, D. S. De Lorenzo, J. Seo, S. Lo, P. Enge and D. M. Akos, "Real-time software receiver for GPS controlled reception pattern antenna array processing," in *Proceedings of the 23rd International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS 2010)*, pp. 1932-1941, 2010.
- [17] Y. H. Chen, J. C. Juang, D. S. De Lorenzo, J. Seo, S. Lo, P. Enge and D. M. Akos, "Real-Time dual-Frequency (L1/L5) GPS/WAAS software receiver," to be present in *Proceedings of the 24th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS 2011)*, 2011.
- [18] T. Y. Chiou, J. Seo, T. Walter, and P. Enge, "Performance of a Doppler-aided GPS navigation system for aviation applications under ionospheric scintillation," in *Proceedings of the 21st International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS 2008)*, pp. 1139-1147, 2008.
- [19] P. De Jonge and C. Tiberius, "The LAMBDA method for integer ambiguity estimation: implementation aspects," Publications of the Delft Geodetic Computing Centre, 1996.
- [20] D. S. De Lorenzo, S. C. Lo, J. Seo, Y. H. Chen, and P. Enge "The WAAS/L5 signal for robust time transfer," in *Proceedings of the 23rd International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS 2010)*, pp. 2106-2116, 2010.
- [21] D. S. De Lorenzo, S. C. Lo, P. K. Enge, and J. Rife, "Calibrating adaptive antenna arrays for high-integrity GPS," *GPS Solutions*, online first, 2011.

- [22] L. Dong, C. Ma, and G. Lachapelle, "Implementation and verification of a software-based IF GPS signal simulator," in *Proceedings of the 2004 National Technical Meeting of The Institute of Navigation (NTM 2004)*, pp. 378-389, 2004.
- [23] Ettus Research LLC, USRP2 motherboard and DBSRX programmable daughterboard, available on the web at <http://www.ettus.com>.
- [24] R. Fante and J. Vaccaro, "Wideband cancellation of interference in a GPS receive array," *IEEE Transactions on Aerospace Electronic System*, Vol. 36, pp. 549-564, April, 2000.
- [25] R. L. Fante, "Unambiguous tracker for GPS binary-offset-carrier signals," in *Proceedings of the 59th Annual Meeting of The Institute of Navigation and CIGTF 22nd Guidance Test Symposium (ION AM 2003)*, pp. 141-145, 2003.
- [26] O. L. Frost III, "An algorithm for linearly constrained adaptive array processing," *Proceedings of the IEEE*, Vol. 60, No. 8, pp. 926-935, 1972.
- [27] Galileo OS SIS ICD, *Galileo Open Service: Signal in Space Interface Control Document*, Draft 1, European Space Agency / European GNSS Supervisory Authority, 2008.
- [28] G. Gao, *Towards Navigation Based on 120 Satellites: Analyzing the New Signals*, Ph. D dissertation, Stanford University, 2008.
- [29] GPSW POC, SVN-49 Signal Anomaly, available on the web at http://www.navcen.uscg.gov/pdf/cgsicmeetings/49/Reports/%5B10%5DSVN_49_at_CGSIC.pdf.
- [30] A. Greenberg and T. Ebinuma, "Open source software for commercial off-the-shelf GPS receivers," in *Proceedings of the 18th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION NTM 2005)*, pp. 2820-2829, 2005.
- [31] I. J. Gupta and T. D. Moore, "Space-frequency adaptive Processing (SFAP) for radio frequency interference mitigation in spread-spectrum receivers," *IEEE Transactions on Antennas and Propagation*, Vol. 52, pp. 1611-1615, June, 2004.
- [32] G. W. Heckler and J. L. Garrison, "SIMD correlator library for GNSS software receivers," *GPS Solutions*, Vol. 4, No. 4, 2006.
- [33] M. V. T. Heckler, M. Cuntz, A. Konovaltsev, L. A. Greda, A. Dreher, and M. Meurer, "Development of robust safety-of-life navigation receivers," *IEEE Transactions on Microwave Theory and Techniques*, Vol. 59, No. 4, pp. 998-1005, 2011.

- [34] G. Hein, J. A. Avila-Rodriguez, S. Wallner, A. R. Pratt, J. Owen, J. L. Issler, J. W. Betz, C. J. Hegarty, L. S. Lenahan, J. J. Rushanan, A. L. Kraay, and T. A. Stansell, "MBOC: The new optimized spreading modulation recommended for GALILEO L1 OS and GPS L1C," in *Proceedings of IEEE/ION PLANS 2006*, pp. 883-892, 2006.
- [35] G. Heinrichs, M. Restle, and C. Dreischer, "NavX®-NSR—A novel Galileo/GPS navigation software receiver," in *Proceedings of the 20th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2007)*, pp. 1329-1334, 2007.
- [36] J. K. Holmes, "Code tracking loop performance including the effect of channel filtering and Gaussian interference," in *Proceedings of the IAIN World Congress and the 56th Annual Meeting of The Institute of Navigation (ION AM 2000)*, pp. 382-398, 2000.
- [37] J. K. Holmes, "Noncoherent late minus early power code tracking loop performance with front end filtering," in *Proceedings of Proceedings of the 10th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GPS 1997)*, pp. 583-591, 1997.
- [38] Information – Analytical Center, *GLONASS Interface Control Document*, Russian Space Agency, 2002.
- [39] Intel Corporation, "System programming guide, Part 2," *Intel® 64 and IA-32 Architectures Software Developer's Manual*, Vol. 3B, 2010.
- [40] J. C. Juang, "A multi-objective approach in GNSS code discriminator design," in *Proceedings of IEEE/ION PLANS 2006*, pp. 223-234, 2006.
- [41] J. C. Juang and Y. H. Chen, "Accounting for data intermittency in a software GNSS receiver," *IEEE Transactions on Consumer Electronics*, Vol. 55, No. 2, pp. 327-333, May 2009.
- [42] J. C. Juang and Y. H. Chen, "Phase/Frequency tracking in a GNSS software receiver," *IEEE Journal of Selected Topics in Signal Processing*, Vol. 3, No. 4, pp. 651-660, August 2009.
- [43] J. C. Juang, Y. H. Chen, T. L. Kao and Y. F. Tsai, "Design and implementation of an adaptive code discriminator in a DSP/FPGA-based Galileo receiver," *GPS Solutions*, Vol. 14, No. 3, pp. 255-266, 2010.
- [44] O. Julien, *Design of Galileo L1F Receiver Tracking Loops*, Ph.D. Dissertation, Department of Geomatics Engineering, University of Calgary, 2005.
- [45] O. Julien, C. Macabiau, J. L. Issler, and L. Ries, "1-bit processing of composite BOC (CBOC) signals and extension to time-multiplexed BOC (TMBOC) signals," in *Proceedings of the 20th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2007)*, pp. 227-239, 2007.

- [46] T. L. Kao, Y. H. Chen, and J. C. Juang, "A DSP/FPGA design for the acquisition and tracking of GIOVE-A signals," in *Proceedings of the 20th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2007)*, pp. 2250-2255, 2007.
- [47] P. M. Kintner, T. Humphreys, and J. Hinks, "GNSS and ionospheric scintillation," *Inside GNSS*, July/August 2009, pp. 22-31, 2009.
- [48] A. Knezevic, C. O'Driscoll, and G. Lachapelle, "Co-processor aiding for real-time software GNSS receiver," in *Proceedings of the 2010 International Technical Meeting of The Institute of Navigation (ION ITM 2010)*, pp. 667-678, 2010.
- [49] A. Konovaltsev, F. Antreich, and A. Hornbostel, "Performance assessment of antenna array algorithms for multipath and interferers mitigation," in *2nd GNSS Signals and Signal Processing Workshop*, 2007.
- [50] D. Last, "GPS forensics, crime & jamming," in *2nd GNSS Vulnerabilities and Solutions Conference*, Baška, Krk Island, Croatia, September 2-5, 2009.
- [51] B. M. Ledvina, S. P. Powell, P. M. Kintner, and M. L. Psiaki, "A 12-channel real-time GPS L1 software receiver," in *Proceedings of the 2003 National Technical Meeting of The Institute of Navigation (ION NTM 2003)*, pp. 767-782, 2003.
- [52] B. M. Ledvina, M. L. Psiaki, D. J. Sheinfeld, A. P. Cerruti, S. P. Powell, and P. M. Kintner "A real-time GPS civilian L1/L2 software receiver," in *Proceedings of the 17th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2004)*, pp. 986-1005, 2004.
- [53] M. Li, F. X. Wang, A. T. Balaei, A. G. Dempster, and C. Rizos, "A GNSS software receiver beamforming architecture," in *Proceeding of the International Symposium on GPS/GNSS*, pp. 904-909, Tokyo, Japan, November 25-28, 2008.
- [54] M. Lu and G. Gao, "Status of Compass Development," in *Stanford's 2010 PNT Challenges and Opportunities Symposium*, available on the web at http://scpnt.stanford.edu/pnt/PNT10/presentation_slides/7-PNT_Symposium_LUandGao.pdf, 2010.
- [55] K. M. Malladi, R.V. R. Kumar, and K. V. Rao, "Gauss-Markov model formulation for the estimation of time-varying signals and systems," *1998 IEEE Region 10 International Conference on Global Connectivity in Energy, Computer, Communication and Control*, Vol. 1, pp.166-169, 1998.
- [56] P. Misra and P. Enge, *Global Positioning System: Signals, Measurement, and Performance*, 2nd Edition, Ganga-Jamuna Press, Lincoln, MA, 2006.
- [57] Navstar GPS Joint Program Office, *IS-GPS-200D: Navstar GPS Space Segment/Navigation User Interfaces*, 2004.

- [58] Navstar GPS Joint Program Office, *IS-GPS-705, Navstar GPS Space Segment/User Segment L5 Interfaces*, 2002.
- [59] P. L. Normark and C. Ståhlberg, "Hybrid GPS/Galileo real time software receiver," in *Proceedings of the 18th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2005)*, pp. 1906-1913, 2005.
- [60] T. Pany, M. Irsigler, and B. Eissfeller, "S-curve shaping: A new method for optimum discriminator based code multipath mitigation," in *Proceedings of the 18th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2005)*, pp. 2139-2154, 2005.
- [61] B. W. Parkinson and J. J. Jr. Spilker, *Global Positioning System: Theory and Applications*, Volume 1, American Institute of Aeronautics and Astronautics, 1995.
- [62] Radio Technical Commission for Aeronautics Special Committee 159, *RTCA DO-229C, Minimum Operational Performance Standards for Global Positioning System/Wide Area Augmentation System Airborne Equipment*, 2006.
- [63] J. Seo, T. Walter, T. Y. Chiou, J. Blanch, and P. Enge, "Evaluation of deep signal fading effects due to ionospheric scintillation on GPS aviation receivers," in *Proceedings of the 21st International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2008)*, pp. 2397-2404, 2008.
- [64] S. Söderholm, T. Jokitalo, K. Kaisti, H. Kuusniemi, and H. Naukkarinen, "Smart positioning with Fastrax's software GPS receiver solution," in *Proceedings of the 21st International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2008)*, pp. 1193-1200, 2008.
- [65] A. J. Van Dierendonck, "GPS receivers," In: *Global Positioning System: Theory and Application*, Vol. I, B. W. Parkinson and J. J. Spilker, Jr., Eds, Washington, D.C., American Institute of Aeronautics and Astronautics, 1996.
- [66] T. Walter, J. Blanch, and P. Enge, "Coverage improvement for dual frequency SBAS," in *Proceedings of the 2010 International Technical Meeting of The Institute of Navigation (ION ITM 2010)*, pp. 344-353, 2010.
- [67] D. Williams, S. Clark, J. Cook, P. Corcoran, and S. Spaulding, "Four-element adaptive array evaluation for United States navy airborne applications," in *Proceedings of the 13th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GPS 2000)*, pp. 2523-2532, 2000.
- [68] K. C. Yeh and C. H. Liu, "Radio wave scintillations in the ionosphere," *Proceedings of the IEEE*, Vol. 70, No. 4, pp.324-360, 1982.

- [69] W. Yu, G. Lachapelle, and S. Skone, "PLL performance for signals in the presence of thermal noise, phase noise, and ionospheric scintillation," in *Proceedings of the 19th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2006)*, pp. 1341-1357, 2006.
- [70] L. Zhang and Y. T. Morton, "Tracking GPS signals under ionosphere scintillation conditions," in *Proceedings of the 22nd International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS 2009)*, pp.227-234, 2009.
- [71] R. E. Ziemer and W. H. Tranter, *Principles of Communications: Systems, Modulation, and Noise*, 5th Edition, John Wiley & Sons, 2002.



Publication List

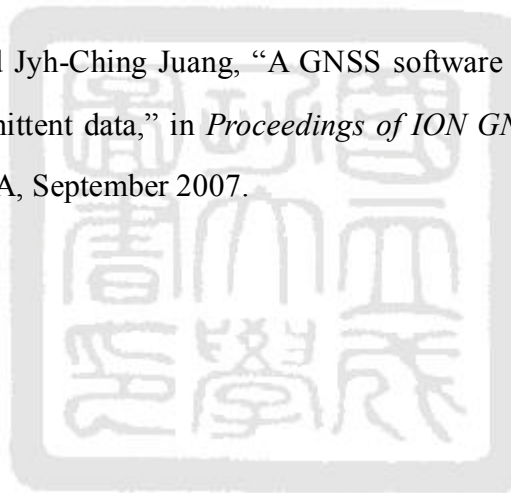
A. International Journal Papers

- [1] Jyh-Ching Juang and Yu-Hsuan Chen, "Accounting for data intermittency in a software GNSS receiver," *IEEE Transactions on Consumer Electronics*, Vol. 55, No. 2, pp. 327-333, May 2009. (SCI 2010 IF = 1.057)
- [2] Jyh-Ching Juang and Yu-Hsuan Chen, "Phase/Frequency tracking in a GNSS software receiver," *IEEE Journal of Selected Topics in Signal Processing*, Vol. 3, No. 4, pp. 651-660, August 2009. (SCI 2010 IF = 2.647)
- [3] Jyh-Ching Juang, Yu-Hsuan Chen, Tsai-Ling Kao and Yung-Fu Tsai, "Design and implementation of an adaptive code discriminator in a DSP/FPGA-based Galileo receiver," *GPS Solutions*, Vol. 14, No. 3, pp. 255-266, 2010. (SCI 2010 IF = 1.483)
- [4] Jyh-Ching Juang and Yu-Hsuan Chen, "Global navigation satellite system signal acquisition using multi-bit codes and a multi-layer search strategy," *IET Radar, Sonar & Navigation*, Vol. 4, No. 5, pp. 673-684, 2010. (SCI 2010 IF = 1.113)

B. International Conference Papers

- [1] Yu-Hsuan Chen, Jyh-Ching Juang and Ji-De Lin, "Design and Implementation of a Multicorrelator Tracking System for GNSS Multipath Mitigation," in *Processing of International Symposium on GPS/GNSS 2005*, Hong Kong, December 2005.
- [2] Yu-Hsuan Chen, Jyh-Ching Juang and Tsai-Ling Kao, "Implementation of BOC signal acquisition using a DSP/FPGA board", in *Proceedings of IAIN/GNSS 2006*, Vol. 2, pp. 405-410, Jeju, Korea, October 2006.
- [3] Jyh-Ching Juang and Yu-Hsuan Chen "Acquisition of GIOVE-A signals using multi-bit processing," in *Proceedings of ION NTM 2007*, pp. 947-956, San Diego, CA, USA, January 2007.

- [4] Yu-Hsuan Chen, Jyh-Ching Juang and Tsai-Ling Kao, "Robust GNSS signal tracking against scintillation effects: a particle filter based software receiver approach," in *Proceedings of ION ITM 2010*, pp. 627-635, San Diego, CA, USA, January 2010.
- [5] Yu-Hsuan Chen, Jyh-Ching Juang, David S. De Lorenzo, Jiwon Seo, Sherman Lo, Per Enge and Dennis M. Akos, "Real-time software receiver for GPS controlled reception pattern antenna array processing," in *Proceedings of ION GNSS 2010*, pp. 1932-1941, Portland, OR, USA, September 2010..
- [6] Yu-Hsuan Chen, Jyh-Ching Juang, David S. De Lorenzo, Jiwon Seo, Sherman Lo, Per Enge and Dennis M. Akos, "Real-time dual-frequency (L1/L5) GPS/WAAS software receiver," to be present in *Proceedings of ION GNSS 2011*, Portland, OR, USA, September 2011.
- [7] Yu-Hsuan Chen and Jyh-Ching Juang, "A GNSS software receiver approach for the processing of intermittent data," in *Proceedings of ION GNSS 2007*, pp. 2772-2777, Fort Worth, TX, USA, September 2007.



Vita

Name Yu-Hsuan Chen

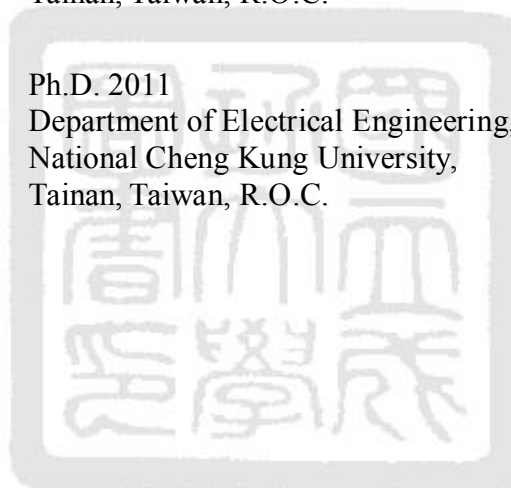
Date of Birth July 30, 1978

Place of Birth Kaohsiung, Taiwan, R.O.C.

Academic Record B.S. 2000
Department of Electrical Engineering,
National Sun Yat-sen University,
Kaohsiung, Taiwan, R.O.C.

M.S. 2002
Department of Electrical Engineering,
National Cheng Kung University,
Tainan, Taiwan, R.O.C.

Ph.D. 2011
Department of Electrical Engineering,
National Cheng Kung University,
Tainan, Taiwan, R.O.C.



著作權聲明



本論文同意部份影印



本論文同意全部影印



本論文不得影印

簽名

陳育昭