

**DIRECT GLOBAL POSITIONING SYSTEM P-CODE ACQUISITION  
FIELD PROGRAMMABLE GATE ARRAY PROTOTYPING**

A Dissertation Presented to the Faculty of the  
Fritz J. and Dolores H. Russ  
College of Engineering and Technology  
Ohio University

In Partial Fulfillment  
of the Requirement for the Degree  
Doctor of Philosophy

by

Jing Pang

August, 2003

THIS DISSERTATION ENTITLED

**“DIRECT GLOBAL POSITIONING SYSTEM P-CODE ACQUISITION  
FIELD PROGRAMMABLE GATE ARRAY PROTOTYPING”**

by Jing Pang

has been approved

for the School of Electrical Engineering and Computer Science  
and the Russ College of Engineering and Technology

---

Janusz Starzyk, Professor  
School of Electrical Engineering and Computer Science

---

Dennis Irwin, Dean  
Fritz J. and Dolores H. Russ  
College of Engineering and Technology

## ACKNOWLEDGEMENTS

First of all, I would like to express my gratitude to my advisor, Dr. Janusz Starzyk, for his support, patience, encouragement throughout my graduate studies, and his invaluable advice during the whole work with this dissertation. He also has taught me innumerable lessons and insights on the workings of academic research in general.

Special thanks to Dr. Frank Van Graas, for his financial support of my graduate study, his inspiring and encouraging way to guide me to a deeper understanding of GPS research.

I would like to thank the rest of my dissertation committee: Dr. Chris G. Bartone offered much-appreciated advice to start this dissertation research work. Dr. Robert Curtis helped me on electronics when I worked on interfacing FPGA with other hardware. Dr. Surender Jain spent his valuable time to attend my presentation, read my dissertation draft, and gave valuable comments.

I am very grateful to all students in VLSI and software radio research groups. I appreciate all their friendships and their collective encouragement to finish this dissertation. I specially thank Abdulqadir A. Alaqeeli and Sanjeev Gunawardena for sharing their technical wisdom and research ideas.

Furthermore, I would like to acknowledge the rich blessings of my sister, and my parents. Their love is instrumental for me to complete this dissertation.

# TABLE OF CONTENTS

Acknowledgement.....	iii
Table of Contents.....	iv
List of Tables.....	viii
List of Figures.....	x
List of Abbreviations.....	xv
Chapter 1.....	1
Introduction.....	1
Chapter 2.....	9
GPS Background .....	9
2.1 GPS Signal Structure.....	9
2.2 GPS CDMA Principle.....	12
2.3 GPS Receiver Front Ends.....	14
2.4 GPS P-code Acquisition.....	15
2.4.1 Sequential Search.....	17
2.4.2 Parallel Code Phase Search.....	18
2.5 Software and Hardware Receiver.....	19
Chapter 3 .....	21
Direct GPS P-code Acquisition Modeling.....	21
3.1 Introduction.....	21
3.2 P-code Property.....	22

3.3	FFT Search and Zero Padding.....	26
3.4	Comparison of Different Acquisition Methods.....	27
3.5	Direct Average Method.....	30
3.6	Effect of Code Phase Shift.....	36
3.7	Overlap Average Method.....	39
3.8	Noise Effect.....	49
3.9	Example.....	63
3.10	Conclusion.....	65
Chapter 4.....		67
GPS P-code Generator Architecture.....		67
4.1	P-code Generator Architecture.....	69
4.2	GPS P-code Generator Tuning Model.....	71
4.3	P-code Generator Design Verification.....	76
Chapter 5.....		84
Direct GPS P-code Acquisition Design.....		84
5.1	Introduction.....	84
5.2	Architecture of Direct GPS P-code Acquisition Processor.....	85
5.3	Local Reference Generation Unit.....	87
5.4	Local Reference FFT Processor.....	90
5.4.1	Xilinx 1024-Point FFT/IFFT Core.....	93
5.4.2	Controller FFT_MACH.....	95
5.4.3	Loop Counter.....	97
5.4.4	RAM2 and RAM_FFT.....	98
5.4.5	Debug Setup.....	98
5.5	Complex Conjugate Multiplication Processor.....	100

5.6	IFFT Processor.....	105
5.6.1	Controller IFFT_MACH.....	108
5.7	Correlation Amplitude Square Unit.....	111
5.8	Correlation Peak and Peak Location Processor.....	112
5.8.1	Correlation Peak Processor.....	114
5.8.2	Correlation Peak Location Processor.....	114
5.9	Maximum Selection Unit.....	116
5.10	Hardware Design Cost.....	118
5.11	Conclusion.....	119
Chapter 6.....		121
FPGA Design Verification.....		121
6.1	Introduction.....	121
6.2	Verification of P-code Average With Scaling.....	122
6.3	Verification of FFT.....	123
6.4	Verification of Complex Conjugate Multiplication.....	128
6.5	Verification of IFFT.....	130
6.6	Verification of Correlation Square.....	134
6.7	Final Verification Results .....	135
6.8	Conclusion.....	139
Chapter 7.....		140
Summary and Future Work.....		140
7.1	Summary.....	140
7.2	Future Work.....	143

References.....	144
Appendix A.....	154
The Strathnuey Card.....	154
Abstract.....	156

## LIST OF TABLES

Table 3.1 Performance comparison of different acquisition methods.....	29
Table 3.2 C/No ratio for different acquisition methods with acquisition detection probability of around 90%.....	29
Table 3.3 Correlation and acquisition margin characteristics using different schemes without noise.....	50
Table 3.4 Correlation and acquisition margin characteristic using different schemes with added noise.....	58
Table 4.1 Solution table for example 1.....	75
Table 4.2 P-code Vector States [ICD-GPS-200].....	77
Table 4.3 Case 1 P-code generator initialization table.....	78
Table 4.4 Case 2 P-code generator initialization table.....	79
Table 4.5 P-code Reset Timing at the end of a week.....	83
Table 5.1 VirtexE FPGA design cost.....	118
Table 6.1 Correlation square peak and code phase shift over 10 ms in verification of case 1.....	135
Table 6.2 Correlation square peak and code phase shift over 10 ms in verification of case 2.....	136



Table 6.3 Correlation square peak and code phase shift over 10 ms in verification of case 3.....	137
Table 6.4 Correlation square peak and code phase shift over 10 ms in verification of case 4.....	138

## LIST OF FIGURES

Figure 2.1 BPSK diagram.....	10
Figure 2.2 GPS signal structure.....	12
Figure 2.3 GPS signal acquisition two-dimensional searching area.....	16
Figure 2.4 Sequential search direct GPS P-code acquisition diagram.....	17
Figure 2.5 Parallel code phase search direct GPS P-code acquisition diagram.....	19
Figure 3.1 Autocorrelation of P-code over 50 ms.....	23
Figure 3.2 Direct acquisition autocorrelation in 1ms.....	24
Figure 3.3 Direct acquisition: acquisition margin distribution over 1s.....	25
Figure 3.4 Direct average autocorrelation result.....	31
Figure 3.5 Direct average method: acquisition margin distribution over 1s.....	32
Figure 3.6 Direct average acquisition example.....	34
Figure 3.7 Correlation resolution improvement.....	35
Figure 3.8 Direct average method: correlation results with a code phase shift of 64 samples.....	36
Figure 3.9 Direct average method: acquisition margin distribution over 1s with a code phase shift of 64 samples.....	37
Figure 3.10 Direct average method: detected correlation peak value change, case 1....	38
Figure 3.11 Direct average method: detected correlation peak value change, case 1....	39

Figure 3.12 Correlation result using overlap average method.....	40
Figure 3.13 Statistical acquisition margin result for overlap average method.....	41
Figure 3.14 Correlation result for overlap average method with a code phase shift of 64 samples.....	42
Figure 3.15 Statistical acquisition margin result for overlap average method with a code phase shift of 64 samples.....	43
Figure 3.16 Overlap average method: two reference overlap, case 1.....	44
Figure 3.17 Overlap average method: two reference overlap, case 2.....	44
Figure 3.18 Peak correlation location detection difference using two different references.....	45
Figure 3.19 Peak correlation location detection pattern change.....	46
Figure 3.20 Overlap average method: two reference overlap, case 3.....	47
Figure 3.21 Overlap average method by using three references.....	48
Figure 3.22 Noise effect on direct autocorrelation result of p-code up samples over 1 ms.....	51
Figure 3.23 Direct acquisition: noise effect on acquisition margin result.....	52
Figure 3.24 Direct average method: correlation result with added Gaussian noise.....	53
Figure 3.25 Direct average method: noise effect on acquisition margin.....	54

Figure 3.26 Direct average method: noise effect on detected correlation peak change with code phase shift.....	55
Figure 3.27 Overlap average method: correlation result with added Gaussian noise.....	56
Figure 3.28 Overlap average method: noise effect on acquisition margin.....	57
Figure 3.29 Overlap average method: noise effect on detected correlation peak change with code phase shift.....	57
Figure 3.30 Direct average method: misdetection probability with different carrier to noise ratio when signal code phase shift is 64 samples.....	59
Figure 3.31 Overlap average method: misdetection probability with different carrier to noise ratio when signal code phase shift is 64 samples.....	60
Figure 3.32 Direct average method misdetection probability distribution in 1 s with carrier to noise ratio equal to 39 dB.....	61
Figure 3.33 Overlap average method misdetection probability distribution in 1 s with carrier to noise ratio equal to 39 dB.....	62
Figure 3.34 Acquisition by direct average method.....	63
Figure 3.35 Acquisition by overlap average method.....	64
Figure 3.36 Resolution improvement by correlating 128 samples.....	65

Figure 4.1 GPS P-code Signal Generator.....	68
Figure 4.2 X1A LFSR Diagram.....	70
Figure 4.3 X1B LFSR Diagram.....	70
Figure 4.4 X2A LFSR Diagram.....	71
Figure 4.5 X2B LFSR Diagram.....	71
Figure 4.6 Case 2 simulation waveform.....	80
Figure 4.7 The final X2AQ chip generation at the end of a week.....	81
Figure 4.8 The final X1BQ chip generation at the end of a week.....	81
Figure 4.9 Resume of X1AQ, X1BQ, X2AQ and X2BQ at the start of week.....	82
Figure 5.1 Direct GPS P-code Acquisition Processor.....	86
Figure 5.2 Local Reference generation unit diagram.....	88
Figure 5.3 Local reference FFT processor diagram.....	91
Figure 5.4 Xilinx 1024-point FFT/IFFT core single memory space configuration.....	94
Figure 5.5 Debug Use Block RAM diagram.....	100
Figure 5.6 Complex conjugate multiplication processor (part 1).....	102
Figure 5.7 Complex conjugate multiplication processor (part 2).....	103
Figure 5.8 IFFT processor diagram.....	106
Figure 5.9 The controller IFFT_MACH flow chart.....	109
Figure 5.10 Controller IFFT_MACH interface FFT/IFFT core diagram.....	110
Figure 5.11 Correlation amplitude square unit diagram.....	111
Figure 5.12 Correlation peak and peak location processor diagram.....	113

Figure 5.13 Maximum selection unit diagram.....	117
Figure 6.1 2nd ms GPS binary P-code average result.....	123
Figure 6.2 FPGA FFT real part values in the 2 <sup>nd</sup> ms.....	124
Figure 6.3 FPGA FFT imaginary part values in the 2 <sup>nd</sup> ms.....	125
Figure 6.4 FPGA and Matlab FFT real part difference in the 2 <sup>nd</sup> ms.....	126
Figure 6.5 FPGA and Matlab FFT imaginary part difference in the 2 <sup>nd</sup> ms.....	127
Figure 6.6 FPGA complex multiplication real part in the 2 <sup>nd</sup> ms.....	128
Figure 6.7 FPGA complex multiplication imaginary part in the 2 <sup>nd</sup> ms.....	129
Figure 6.8 FPGA IFFT real part in the 2 <sup>nd</sup> ms.....	130
Figure 6.9 FPGA IFFT imaginary part in the 2 <sup>nd</sup> ms.....	131
Figure 6.10 FPGA and Matlab IFFT real part difference in the 2 <sup>nd</sup> ms.....	132
Figure 6.11 FPGA and Matlab IFFT imaginary part difference in the 2 <sup>nd</sup> ms.....	133
Figure 6.12 FPGA correlation square values in the 2 <sup>nd</sup> ms.....	134
Figure A-1 Strathnuey FPGA board architecture.....	155

## LIST OF ABBREVIATIONS

ADC:	Analog to Digital Converter
AS:	Anti-Spoofing
ASIC:	Application Specific Integrated Circuits
BPSK:	Binary Phase Shift Keying
C/A:	Coarse Acquisition
CDMA:	Code Division Multiple Access
CLB:	Configurable Logic Block
C/No:	Carrier-to-Noise Density Ratio
DFT:	Discrete Fourier Transform
DIF:	Decimation-In-Frequency
DOD:	Department of Defense
DSP:	Digital Signal Processing
DSSS:	Direct Sequence Spread Spectrum
FFT:	Fast Fourier Transform
FPGA:	Field Programmable Gate Array
GPS:	Global Position System
GRM:	Generic Routing Matrix
HDL:	Hardware Description Language

HOW:	Hand-Over-Word
IDFT:	Inverse Discrete Fourier Transform
IF:	Intermediate Frequency
IFFT:	Inverse Fast Fourier Transform
IOB:	Input/Output Block
IP:	Intellectual Property
LFSR:	Linear Feedback Shift Registers
LNA:	Low Noise Amplifier
NF:	Noise Figure
NCO:	Numerically Controlled Oscillator
P-code:	Precision Code
PPS:	Precise Positioning Service
PRN:	Pseudorandom Noise
RF:	Radio Frequency
RTL:	Register-Transfer-Level
SNR:	Signal-to-Noise Ratio
SPS:	Standard Positioning Service
Y-code:	Encrypted P-code



# Chapter 1

## Introduction

The Global Positioning System (GPS) is a worldwide, satellite-based navigation system. It is being funded and operated by the U.S. Department of Defense (DOD). The concept of GPS started in the early 1970's, and it was originally intended for military applications. Later, the U.S. government made the system available for civilian use. The increased performance of GPS has fuelled the continued growth of the GPS market (El-Rabbany, 2002 and Pace, 1995). It has been used in agriculture (Bauer, 1994), mining (Flinn, 1999 and Jensen, 1992), transportation (Drane, 1998), surveying (Leick, 1995), land (Hada, 2000), marine (Trimble, 1998), airborne navigation (Hayward, 1998) and many other applications (Parkinson, vol. II, 1996).

GPS provides specially coded satellite signals that can be processed in a GPS receiver. A receiver measures the elapsed time between the emission and the reception of the GPS signal. This time is then multiplied by the speed of light to obtain the distance between the satellite and the receiver, which is referred to as pseudorange, not range (Langley, 1993), because the measurements contain errors and biases coming from the synchronization error between the satellite and receiver clocks, multipath errors and other error sources. GPS provides the pseudorange and carrier phase measurements. While

GPS signal acquisition gives coarse estimates of these measurements, carrier phase tracking provides a precise measurement of change in the satellite user pseudorange over a time interval, and the estimates of its instantaneous rate, or Doppler. Estimations of position and velocity are based on these measurements (Braasch, 1999 and Ward, 1995). A minimum of four measurements made on four different satellites can precisely determine position, velocity, and time (Kaplan, 1990).

The GPS satellites transmit two carrier frequencies: L1 (1575.42 MHz) and L2 (1227.60 MHz). The pseudorandom noise (PRN) codes (Misra, 2001) and navigation data message are added to the carriers as binary phase modulations (Rappaport, 1995). The navigation data message is a 50 bits/second digital data stream, which contains GPS satellite coordinates, the satellite health status, the satellite clock correction, the satellite almanac, and other satellite system parameters (Kaplan, 1990). The PRN codes include C/A (coarse acquisition) code on L1 carrier and P (precise) code on L1 and L2 carriers. The codes are called chips instead of bits to emphasize they do not carry data. Compared with the C/A code, which is 1023 chips long and is broadcast at 1.023 Mega-chips per second, the P-code is much longer (seven days) with a faster chipping rate 10.23 MHz, offering much higher accuracy (Misra, 1999 and Hoffmann-Wellenhof, 1994). The P-code is encrypted and the resulting code is called P(Y)-code. The encrypted P(Y)-code is for use only by authorized users with cryptographic keys.

Every GPS receiver design involves GPS signal acquisition, code tracking loop, and carrier tracking loop circuit design (Kaplan, 1990). The acquisition circuit searches

over a three-parameter space: satellite PRN code, code phase, and carrier frequency to acquire the signal. Acquisition can provide a “coarse” estimation of the pseudorange. The code tracking loop tracks the locally generated PRN code to be synchronized with the received code in order to despread the signal, derive navigation data and timing information.

Traditionally, the GPS receiver locks onto the *C/A* code to extract the navigation data message. The Hand-over-word (HOW) is the second word in each subframe/page of a navigation message. A HOW occurs every six seconds in the data frame (ICD-GPS-200, 1991). It provides the necessary timing information to tell where the *P(Y)* code pattern is in its whole sequence.

Then, starting from the HOW, acquisition is transferred to the *P*-code acquisition part. As a result, *P*-code acquisition relies on the hand over from an acquired *C/A* code (Lin, 1999). This is because the *C/A* code repeats every millisecond so that synchronizing to the *C/A* code is easy. A brute force chip-by-chip search of *P*-code is formidable in terms of the number of search trials and the time required to carry it out. But if the receiver can accurately predict the satellite signal transmit time and the satellite location, direct *P(Y)*-code acquisition is faster than handover from the *C/A* code search. During certain military operations, the *C/A*-code signal could be degraded so that the use of the open Standard Positioning Service to adversaries is denied and the use of the encrypted *P(Y)*-code is still sustained. Consequently, direct acquisition of the *P* code is the only option available. Newer designs attempt to achieve acquisition without first acquiring the

C/A code and use the P(Y)-code for direct acquisition.

The direct P (Y)-code acquisition potential is a more robust functionality for GPS receivers to provide the ability to acquire the P(Y)-code without the aid of the C/A code (Zyfer, 2002). Application of P(Y)-code rate sequences can also be used on pseudolites that implement wideband code formats. In situations with limited or no visibility of the GPS satellites, ground transmitters that emulate the signal structure of the GPS satellites (pseudolites) (Cobb, 1998) can be used as additional or replacement signal sources (Wang, 2000 and Stone, 1999). Pseudolites are useful for a variety of applications, including differential GPS implementation in aircraft precision approach in the Category II and III for universal availability. Pseudolites have the capability to augment the GPS constellation, providing better geometry for enhanced positioning accuracy, reliability, availability, continuity, and integrity monitoring. In addition they can be used to speed up integer ambiguity resolution in differential carrier phase applications, due to the large geometry change possible with the pseudolite signal (Ndili 1994).

The overall objective of this dissertation is to implement the direct GPS P-code acquisition algorithm using FPGA (Field Programmable Gate Array) technology. The challenges are how to reduce the code phase search time for the extremely long period of the P-code, how to avoid large size Fast Fourier Transform (FFT), and how to design the P-code generator, which can rapidly produce the P-code at any specific time of a GPS week. The exploration of the direct GPS P-code acquisition algorithm presents a good starting point for meeting these objectives. To produce a local code, which can

continuously duplicate the incoming GPS signal code phase, the initial unknown code phase has to be determined by searching through a set of possible phases around an initial time estimate. If a search over 10 ms is needed, this can be translated into a location uncertainty of  $3 \times 10^6$  meters. This location uncertainty may be caused by the uncertainty of the satellite location, the satellite clock offset, the receiver clock offset or other uncertainty factors from the time of transmission to the time of reception. The fast P-code phase search using the direct average and overlap average method proposed in this dissertation may be a tradeoff of the carrier to noise ratio performance. In addition, the proposed methods use a small size FFT/IFFT, which greatly facilitates the FPGA hardware design.

After the direct P-code acquisition algorithm is proposed and simulated, it is possible to have a hardware design to map the algorithm. A FPGA is a good choice to do hardware design. FPGA offers the advantage of short design cycles, rich resources, and programmability (Trimberger, 1994). In the past, FPGAs were primarily used for prototyping and lower volume applications. Custom ASICs were used for high volume, cost sensitive designs. Today's deep submicron fabrication technologies enable design engineers to implement an impressive number of components like microprocessors, memories, and interfaces in a single microchip (Langen, 2002). With today's deep submicron technology, it is possible to deliver over several millions usable system gates (Xilinx White Paper, 2002) and system speeds up to several hundreds MHz in an FPGA. In addition, the average ASIC design operating at 50-100 MHz can be implemented in an

FPGA using the same RTL (Register-Transfer-Logic) synthesis design methodology as ASICs. As device densities keep increasing, not only new levels of systems are integrated onto a single chip, but also a wide variety of features and capabilities are allowed for reprogrammable technology. Moreover, several companies, such as LSI Logic (Matsumoto, 1999), ADI (Ohr, 2000), Lucent (Lucent, 2000), QuickLogic (Merritt, 2000), Actel (Tanurhan, 2001), some startup companies (Matsumoto, 2000) and so on, already provide pre-designed and pre-verified blocks, often called cores or intellectual property (IP) blocks (Wilton, 2001). Fixed hardware designs called hardcores are incorporated into the FPGA architecture such as PowerPC/ARM processor cores, Booth multipliers for DSP (Digital Signal Processing) data paths, serial transceivers and so on (Xilinx, 2003). On the other hand, softcores are flexible IP building blocks that take full advantage of an efficient and flexible implementation in the FPGA including PCI-cores for I/O busses, processor cores, Viterbi decoders for DSP functions and many other examples (Hellmich, 2000 and Erdogan, 2003). These programmable logic cores are utilized by other companies and combined onto a single chip, and in a design process called System-on-a-Chip methodology (Savage, 2000 and Vladimirova, 2000).

This dissertation's major objectives are twofold: (1) to develop and analyze a direct GPS P-code acquisition algorithm which targets on FPGA design, and (2) to design FPGA architecture implementing the proposed algorithm.

The difficulty of direct GPS P-code acquisition is the extremely long period of the P-code, which makes the search for the correct carrier frequency and code phase very

slow. The promising methods to improve acquisition speed proposed by other researchers focus more on software simulation performance. They require large Fast Fourier Transform size. This is prohibiting in FPGA design because of large hardware resource requirement and design complexity. As a result, an appropriately scaled direct P-code acquisition design algorithm, especially for FPGA implementation would be desirable.

In addition, the design of a P-code generator is also a challenging task in this work. The basic properties of P-code are described in the literature, but there is no complete detailed hardware implementation of a P-code generator. Another necessary feature will be the availability of a P-code starting from any time of a week. This fits into the situation that there is approximate time information so that acquisition starts from some specific time range.

In order to have a successful implementation of the whole direct GPS P-code acquisition system in FPGA, Nallatech FPGA board interface and Xilinx VirtexE FPGA architecture features must be studied and utilized in the design. The whole system design should be partitioned into different processing parts. Each processor must be properly designed and the signal flows between different parts must be properly handled. It is very beneficial to use the Xilinx 1024-point FFT/IFFT core to achieve a fast system design cycle. A controller which is designed to meet the Xilinx 1024-point FFT/IFFT timing specification is critical for the integration of the core into the whole design system. A exhaustive test is also necessary to see how stable the FFT/IFFT results are and to verify

if the controller is right. At last, all synthesizable VHDL codes need to be downloaded on the FPGA board. Each partitioned processor must be verified. The final overall results should give the right direct P-code acquisition code phase shift and acquired maximum correlation information.

This dissertation is organized as follows. Chapter 2 covers the necessary GPS principles of operation. Chapter 3 models the direct GPS P-code acquisition algorithm and its modifications. The comparisons of different direct GPS P-code acquisition techniques are also made in this chapter. Then, Chapter 4 presents the GPS P-code generator architecture, design strategy and test results of P-code generator design. Next, Chapter 5 gives details on direct GPS P-code acquisition design implementation. In addition, Chapter 6 verifies each step and the overall FPGA designs. At last, Chapter 7 summarizes the whole dissertation and provides direction for future research.



## Chapter 2

### GPS Background

This chapter deals with some important concepts of GPS such as GPS signal structure, GPS code division multiple access (CDMA) principle, GPS receiver front ends, GPS P-code acquisition, software and hardware receivers. GPS signal structure and CDMA are the basic underlying components for GPS. A GPS receiver receives GPS signals by its front end, processes them in analog domain, and then outputs digital signals from an A/D converter for further acquisition process. Both sequential and parallel code phase search methods are explained. Furthermore, software and hardware receiver implementation are compared.

#### 2.1 GPS Signal Structure

Each GPS satellite transmits a unique navigational signal centered on two L-band frequencies of the electromagnetic spectrum: L1 at 1575.42 MHz and L2 at 1227.60 MHz. PRN codes and navigation message modulate the L1 and/or L2 carrier phase using the binary phase shift keying (BPSK) modulation technique (Kaplan, 1990). The binary codes are directly multiplied with the carrier, which results in a 180-degree phase shift of

the carrier every time the code changes its state. A BPSK diagram is shown in Figure 2.1.

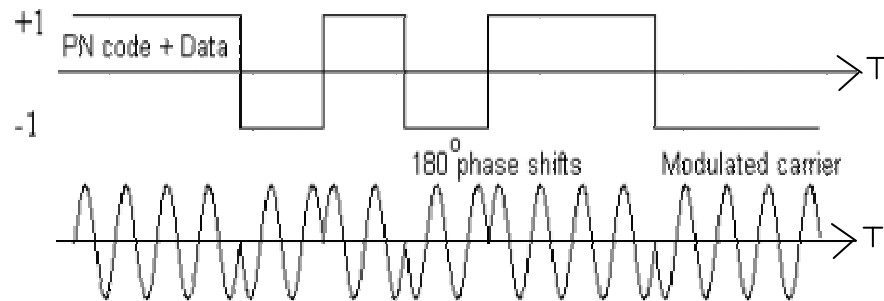


Figure 2.1 BPSK diagram (Braasch, 1999)

The C/A code is a repeating 1.023 MHz PRN code. There is a different C/A code PRN for each satellite. C/A sequences belong to Gold codes, which have very good autocorrelation and cross correlation properties. The P-code is a very long (seven days) 10.23 MHz PRN code. In the Anti-Spoofing (AS) mode of operation, the P-code is encrypted into the P(Y)-code. The encrypted P(Y)-code requires a classified AS module for each receiver channel and is used only by authorized users with cryptographic keys.

The C/A code is the basis for Standard Positioning Service (SPS) and the P-code offers Precise Positioning Service (PPS). According to the 1999 Federal Radio Navigation Plan, the SPS predictable accuracy is around 100-meter horizontal accuracy, 156-meter vertical accuracy and 340-nanosecond time accuracy. The PPS predictable accuracy is around 22-meter horizontal accuracy, 27.7-meter vertical accuracy and 200- nanosecond time accuracy.

The L1 signal is modulated by both the C/A code and the P(Y)-code, which are at

90 degrees to each other, so that one is in phase and the other is in quadrature. The L2 signal is modulated by the P(Y)-code.

The navigation message modulates both the L1 and L2 signals. The GPS navigation message contains parameters that describe the location of the GPS satellites, their clock offsets, and various other system parameters. Each satellite transmits a navigation message at 50 bits per second. The navigation message is composed of five subframes. Each subframe contains 10 words of 30 bits.

Figure 2.2 shows the GPS signal structure modeled in equation (2.1).

$$\begin{aligned}
 S_{L1i}(t) &= A_p P_i(t) D(t) \sin(2\pi f_{L1} t) + A_c G_i(t) D(t) \cos(2\pi f_{L1} t) \\
 S_{L2i}(t) &= B_p P_i(t) D(t) \sin(2\pi f_{L2} t)
 \end{aligned}
 \tag{2.1}$$

where  $A_p$ ,  $A_c$  and  $B_p$  are signal amplitudes

$P_i(t)$  is the P code for satellite i

$G_i(t)$  is the C/A code for satellite i

$D(t)$  is the navigation data for satellite i

$f_{L1}$  is the L1 carrier frequency equal to 1575.42 MHz

$f_{L2}$  is the L2 carrier frequency equal to 1227.60 MHz.

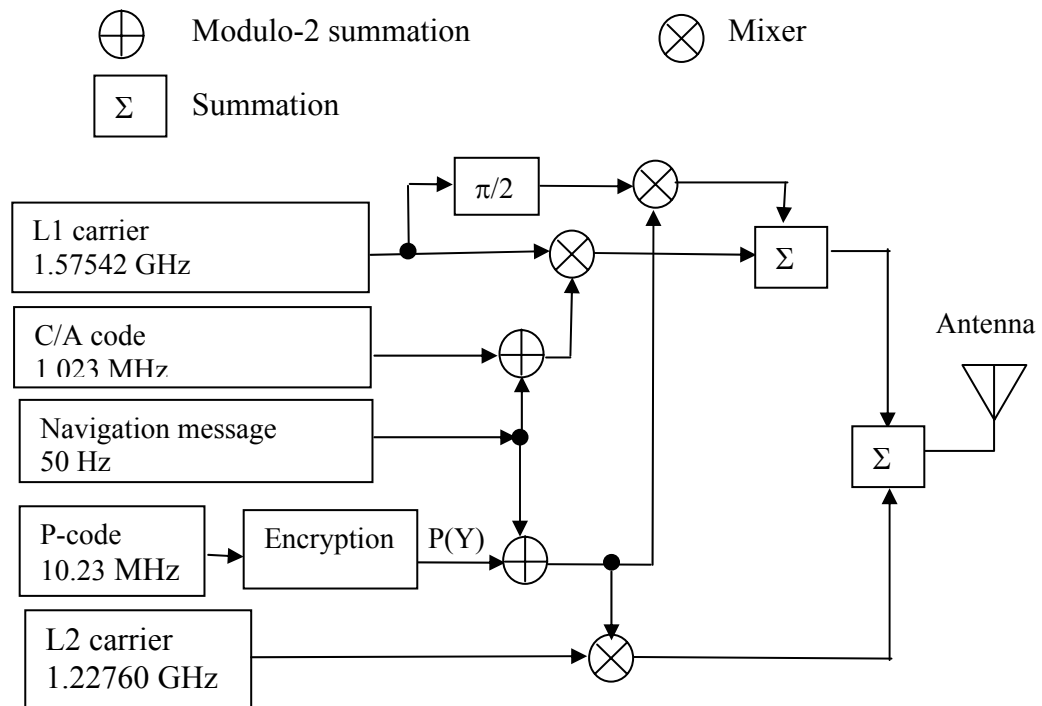


Figure 2.2 GPS signal structure (Kaplan, 1990)

## 2.2 GPS CDMA Principle

CDMA is a form of spread-spectrum, a digital communication technique that has been used in military applications for many years. Now it is one of the driving forces behind the rapidly advancing personal communications industry. CDMA uses unique noise-like spreading codes to spread the baseband data before transmission (Glisic, 1997 and Prasad, 1996). The signal can then be transmitted in a channel below the noise level. The receiver uses a correlator to despread the signal. One type of CDMA technique is

direct sequence spread spectrum (DSSS) (Viterbi, 1995). GPS uses the BPSK DSSS technique. When the user receives the GPS signals, the receiver will mix the received signal with a locally generated PRN-code. The locally generated PRN-code must be synchronous with that generated from the satellite. After the receiver is fully correlated with the GPS data, a correlation peak is generated. Because GPS signals have very wide spread spectrum, they are transmitted at a much lower spectral power density than narrowband transmitters (Braasch, 1999).

One important signal quality parameter is the signal-to-noise ratio (SNR), which is a measure of the signal strength relative to background noise. The ratio is usually measured in decibels (dB) for a specified bandwidth. The noise can be approximated by

$$N = kT_E B \quad (2.2)$$

where  $k$  is Boltzmann's constant ( $1.3806 \times 10^{-23} JK^{-1}$ )

$B$  is the bandwidth in Hz

$T_E$  is the effective noise temperature in Kelvin.

A normalized SNR to a 1-Hz bandwidth is called carrier-to-noise density ratio ( $C/N_o$ ). It is bandwidth-independent. The  $C/N_o$  is defined as equation 2.3.

$$C/N_o = (SNR)(B)[ratio - Hz] \quad (2.3)$$

where SNR is signal to noise ratio at certain stage of a receiver

$B$  is the bandwidth of a receiver at the same stage.

The received satellite signal power varies with the user antenna gain, the satellite

elevation angle, and the satellite age. The typical  $C/N_o$  value is from 35 to 55 dB-Hz (Braasch, 1999).

## 2.3 GPS Receiver Front Ends

The input to the GPS receiver is the analog GPS signal from an antenna via a low-noise pre-amplifier (LNA). The GPS signals are down converted by the radio frequency into the intermediate frequency (IF) signals, which are further converted by an A/D converter into the digital signals. These digital signals are taken by the GPS channels for the subsequent GPS acquisition and tracking process.

The RF front ends deal with the analog signal processing including filtering, amplification, and down conversion (Shaeffer, 1998). The main design parameters that have to be taken into consideration are antenna gain, LNA gain, LNA intermodulation, noise figure (NF), and power consumption (Shahani, 1997, and Piazza, 1998). Sharp cutoff filters must suppress out-of-band interference. Hard-limiting architecture (1-bit A/D conversion) results in slightly degraded performance compared to that of a multibit converter, but it allows the design of a simpler, lower power receiver without employing automatic gain control. Downconversion is performed either in single or multiple stages (Braasch, 1999). Multistage architectures allow for adequate image suppression and general bandpass filtering with the final IF placed close to the baseband

single-stage. The final conversion to baseband involves converting the IF signal to the in-phase and quadrature components of the signal envelope (Dierendonck, 1996).

## 2.4 GPS P-code Acquisition

A GPS receiver performs a three-dimensional search to acquire a GPS satellite signal: the GPS satellite PRN code, code phase, and carrier frequency offset (Parkinson, vol. I, 1996). If one has no knowledge on which satellites are in space, the acquisition must be performed on all satellites. However, usually there is information on the rough location and the approximate time of day, so acquisition is only necessary for those satellites which are known to be available to receivers. For each satellite, a two dimensional search must be performed: code phase and carrier frequency.

In Figure 2.3,  $\delta\tau$  is typically 1/2 code chip and  $\delta f$  is Doppler bin. Carrier frequency uncertainties are due to unknown Doppler shift and local oscillator drift. The span of code phase to be searched depends on uncertainties in indicated clock time and receiver position (Brown, 2001). Due to the extremely long period of the P-code, a search over a one-week long P-code is challenging. The FPGA design requirement in this dissertation is to search the P-code code phase over 10 ms. In realistic stand-alone receivers, the usual search is within 1 s. Satellite induced Doppler frequency uncertainty is less than 10 Hz, and stationary receiver oscillator frequency offset is within 1 kHz.

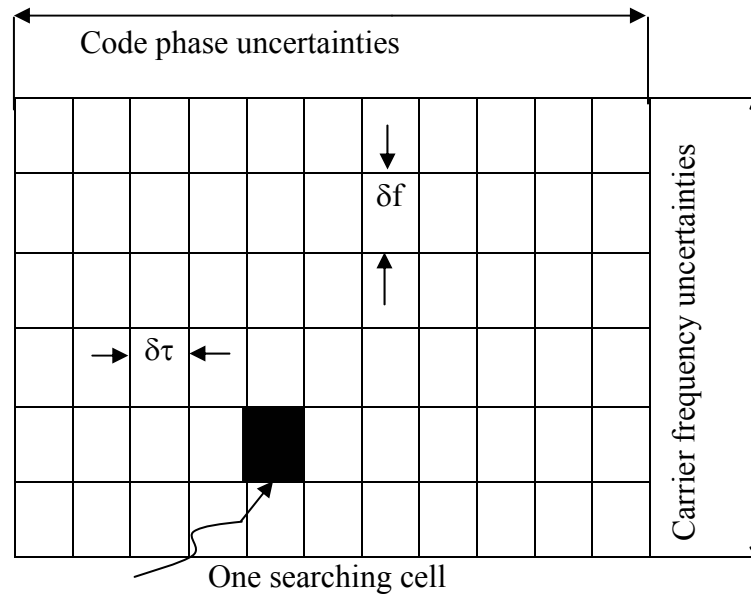


Figure 2.3 GPS signal acquisition two-dimensional searching area

In situations with limited or no visibility of the GPS satellites, ground transmitters that emulate the signal structure of the GPS satellites (pseudolites) (Cobb, 1998) can be used as additional or replacement signal sources (Wang, 2000 and Stone, 1999). In pseudolite application, satellite introduced Doppler is less than 1 Hz, and the receiver oscillator frequency offset is less than 10 Hz if a Rubidium oscillator is used.

Traditionally, a C/A code can be quickly and easily acquired, and is used as a hand over which provides accurate time-tick information for P-code acquisition. Direct P-code acquisition is desirable when a C/A code is not available under jamming conditions but it is still possible to acquire P(Y)-code.



## 2.4.1 Sequential Search

In Figure 2.4, the acquisition is based on a measurement of the correlator output. The correlators provide a measurement of the total I and Q channel correlation energy over the integration time. The In-phase and Quadrature components I and Q respectively are formed by stripping off the reference code and the carrier from the received signal. When the replica and reference signals are aligned with the incoming signal, the amplitude of the recovered correlation energy is at a maximum. The amplitude is compared with a threshold. When the amplitude is at or above the threshold (Brown, GPS Solutions 2000), it is detected as the presence of the signal. In the presence of noise, one must set the threshold based upon an acceptable probability that a noisy measurement that does not contain the signal will appear to match the replica (Kaplan, 1996).

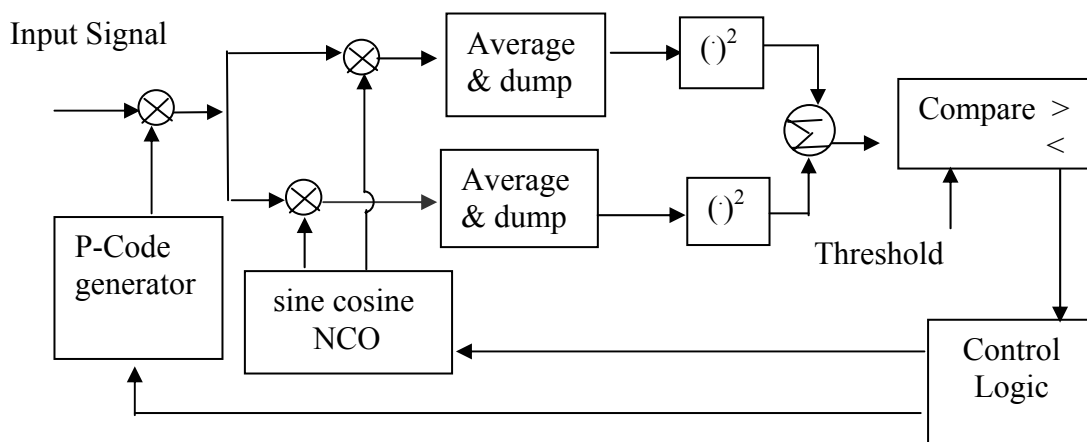


Figure 2.4 Sequential search direct GPS P-code acquisition diagram

## 2.4.2 Parallel Code Phase Search

In time domain, a massive parallel P-code generator with 1,024 or even 2,048 correlator channels searching for a code match has been reported (Trimble, 2002).

General GPS acquisition parallel search techniques include the parallel frequency space search, and the parallel code-phase search. They limit the search space to carrier frequency space and code phase space respectively. When the number of Doppler bins to be searched is far less than that of the code phases, the parallel code-phase search is the most efficient approach.

In Figure 2.5, the GPS acquisition scheme is shown implemented in the digital storage receiver using frequency domain correlation. In this architecture, the GPS signals are first buffered in memory to allow them to be accessed by hardware for processing. Since the GPS signals do not have to be processed in real-time, enhanced signal processing algorithms can be applied that allow the digital signals to be optimally reprocessed, maximizing the probability of acquiring the GPS signals in a challenging environment.

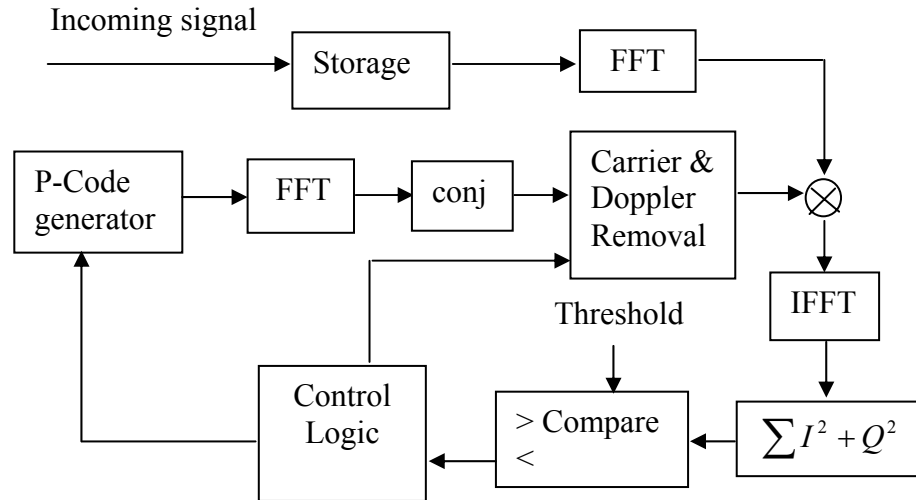


Figure 2.5 Parallel code phase search direct GPS P-code acquisition diagram

This approach takes advantage of the Fourier Transform correlation theorem which states that the frequency transform of the correlation function in the time domain is the product of the signals' transforms in the frequency domain. The FFT algorithm provides a convenient and computationally efficient method of performing correlations in the digital storage receiver architecture.

## 2.5 Software and Hardware Receiver

Development and operation of advanced GPS systems are provided through a variety of software (Brown, ION 2000 and Psiaki, 2001) and hardware solutions (BAE,

1998 and McCullagh, 1999). Except for the RF front ends, the software receiver implements the other receiver functions in software in a microprocessor (Kelley, 2002 and Krumvieda, 2001). The software based implementation of the receiver blocks allows the new acquisition architecture, improved tracking loops and other new features to be implemented, tested and simulated. New frequencies and new PRN codes can be used simply by making software changes. The software based approach can be used for the simulation of hardware receiver design and development.

Usually, the software receiver depends on a host processor that has limited capability to provide real time performance for GPS systems because of a required high dynamic range, a large bandwidth, and a large computation load. However, the use of pre-processors such as FPGA or DSP correlators (for correlation processing), FPGA or DSP processors for FFT processing (Masella, 1999 and Gerein, 2001) are available and provide real time capabilities.

## Chapter 3

# Direct GPS P-code Acquisition Modeling

### 3.1 Introduction

In order to have a good direct P-code acquisition hardware design, a good understanding of P-code acquisition property and a good searching algorithm are needed. The efficiency of a searching algorithm greatly affects the P-code acquisition time. Although the general parallel code-phase search is used due to its efficient searching speed, it is necessary to explore the advanced searching algorithm because of the extremely long period of P-code. Even just searching a segment of the P-code for several ms or 1 s, the searching task is still huge. A good acquisition algorithm is absolutely necessary by limiting the searching segment to several ms or 1 s, the searching task is still huge. A good acquisition algorithm is absolutely necessary.

In this chapter, new algorithms are introduced in this dissertation called direct average and overlap average methods. The performance of new algorithms is analyzed. The P-code property is studied. In addition, different acquisition techniques reported by researchers are compared.

## 3.2 P-Code Property

Each satellite uses unique P-codes to implement the CDMA technique. The autocorrelation function (Proakis, 2000) of a P-code is

$$R_p(\tau) = \int_{-\infty}^{\infty} P_i(t)P_i(t + \tau)dt \quad (3.1)$$

where  $P_i$  is the P-code from the  $i$ th satellite and  $\tau$  is the time phase shift of the P-code.

The correlation peaks repeat after one P-code period of one week. The property of the autocorrelation function is used to synchronize the receiver-replicated code with the received signal. It is important that the cross-correlation of any two P-codes is minimum for any phase or Doppler shift over the entire code period. The ideal cross-correlation is defined by

$$R_{ij}(\tau) = \int_{-\infty}^{\infty} P_i(t)P_j(t + \tau)dt = 0 \quad (3.2)$$

where  $P_i$  is the P-code from the  $i$ th satellite and  $P_j$  from the  $j$ th satellite and  $i \neq j$ .

The autocorrelation plot in Figure 3.1 has a big peak value. The plot has relatively small side lobes.

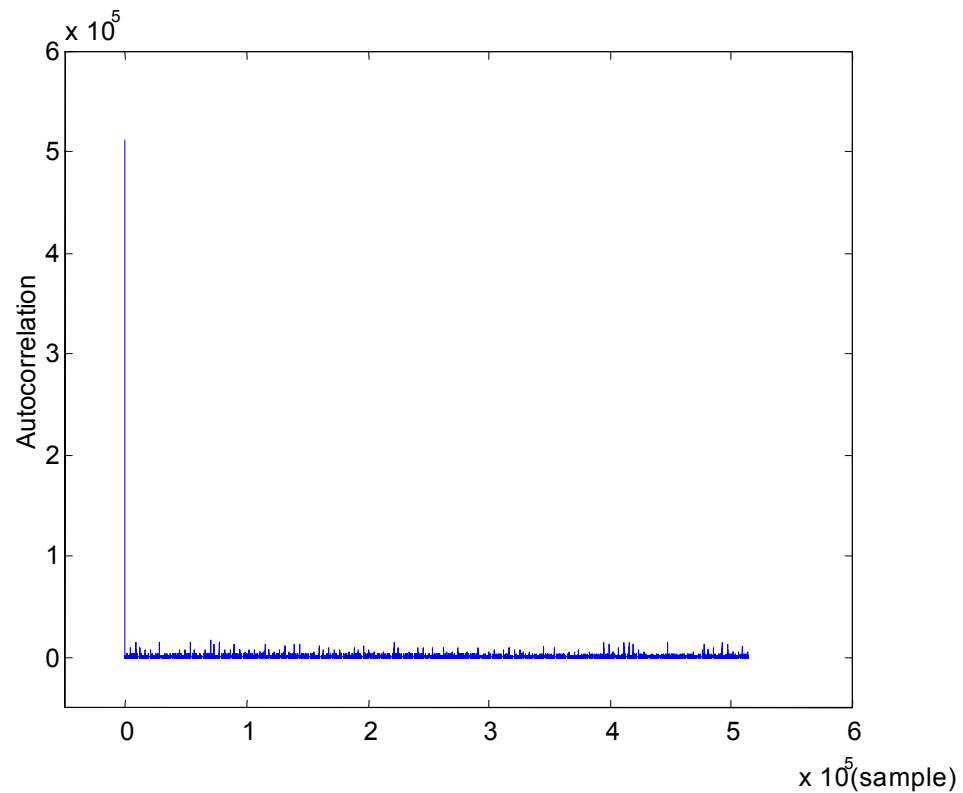


Figure 3.1 P-code Autocorrelation over 50 ms

First, we define the correlation peak in Figure 3.1 as the largest correlation peak. Then a terminology of acquisition margin can be defined as the ratio of the largest correlation peak divided by the second largest correlation peak

$$ACQUISITION\ MARGIN = \frac{LARGEST\ CORRELATION\ PEAK}{SECOND\ LARGEST\ CORRELATION\ PEAK} \quad (3.3)$$

Of course, the bigger the acquisition margin value, the better. As for the long period P-code, a careful statistical analysis is necessary. Suppose the IF for P-code

acquisition is 12.5 MHz, and the GPS signal is up sampled to 65.536 MHz. The signal is finally converted to the base band. Figure 3.2 shows the autocorrelation result of the up sampled P-code in 1 ms and Figure 3.3 illustrates the acquisition margin distribution in 1 s.

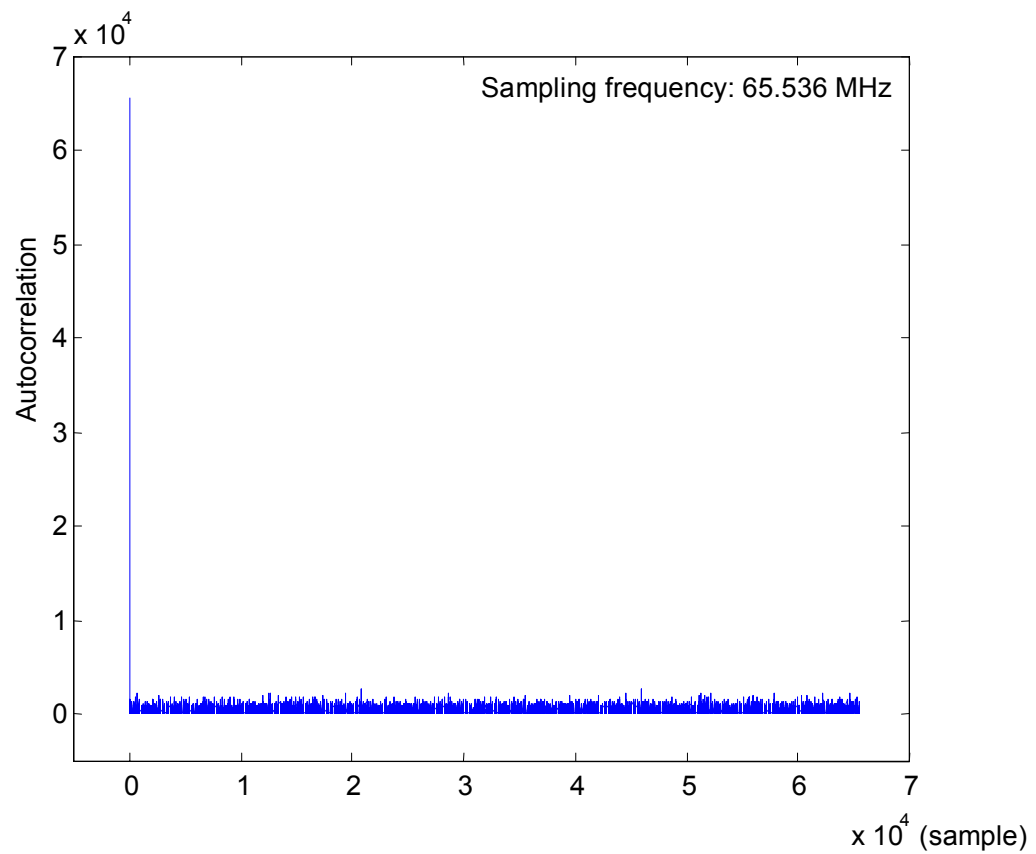


Figure 3.2 Direct acquisition autocorrelation in 1 ms



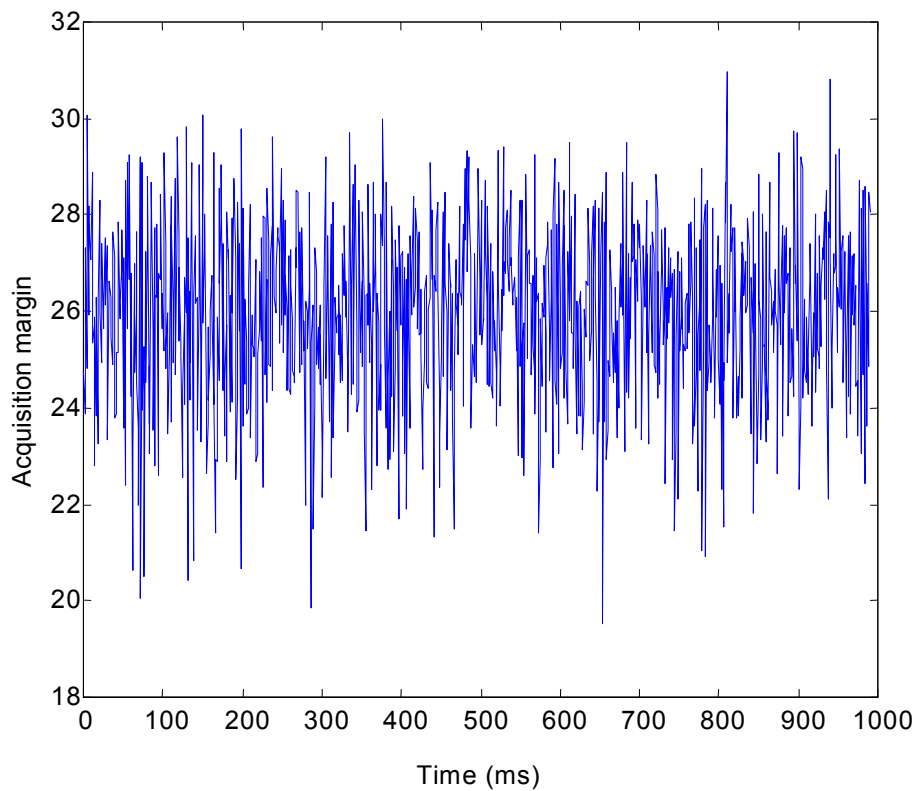


Figure 3.3 Direct acquisition: acquisition margin distribution over 1 s

In Figure 3.3, the mean value of the acquisition margin is 25.954 and the standard deviation is 1.841. So statistically, the acquisition margin value is stable when doing direct acquisition. From Figures 3.1, 3.2, and 3.3, it is interesting to observe that segments of the P-codes or up sampled P-codes are essentially orthogonal to each other. This is the basis for the direct average and the overlap average methods described later. This orthogonal property can be further tested if P-codes at any time interval of a week are available. One efficient way to generate the P-code will be described in Chapter 5.

### 3.3 FFT Search and Zero Padding

The linear correlation of periodic code can be performed by circularly shifting the replica code (Tsui, 2000). This resembles the circular convolution, which is a multiplication in the frequency domain. It can be expressed as

$$R[m] = \sum_{n=0}^{L-1} x[n]^* x[n+m] = x[n] \otimes x[-n] = F^{-1} \left( F(x[n]) \bullet F(x[n])^* \right) \quad (3.4)$$

The Discrete Fourier Transform (DFT) and its inverse are used to calculate the correlation value  $R$  in the above equation. The incoming signal in-phase and quadrature components are used as the real and imaginary inputs when calculating the DFT. The result is multiplied by the complex conjugate of the DFT of the periodic code. The FFT algorithm (Brigham, 1974) is used to implement the DFT and the IDFT, which greatly reduce computation compared with the serial search method (van Nee, 1991).

For non-periodic codes, such as a segment of the P-code, the above calculation method won't give correct correlation results. A zero padding method is used to solve this problem.

First, construct signal A by taking the 1<sup>st</sup> ms of P-codes and pad the 2<sup>nd</sup> ms with zeros. Then, get signal B by taking the 1<sup>st</sup> and the 2<sup>nd</sup> ms of P-codes. Next, calculate correlation according to the following equation:

$$R[m] = F^{-1} \left( F(B[n]) \bullet F(A[n])^* \right) \quad (3.5)$$

Finally, the first half of the correlation results are kept while the second half are discarded. The final correlation results correspond to the correlation of the 1<sup>st</sup> ms P-codes with the 1<sup>st</sup> ms of incoming signal.

### 3.4 Comparison of Different Acquisition Methods

The direct P-code acquisition in the time domain needs massive physical correlators in parallel for code search (Wolfert, 1998), which require a significant amount of resources for hardware implementation. Other attempts on direct P-code acquisition using a software radio algorithm are described below.

#### A. Circular correlation by partition and zero padding (Lin, 2000)

When doing circular correlation, two blocks of signals are taken. One block of reference is taken with another block padded by zeros. Then, take only the first half circular correlation results and discard the other half. Such design considers the almost non-periodic property of the P-code. In addition, the frequency search can be done by performing a FFT on the corresponding correlation results from several different blocks. The number of blocks used for frequency search corresponds to the frequency range covered.

#### B. Circular correlation by partition and superposition (Lin, 2000)

Similar to method A. Method B adds two neighboring blocks into one.

### C. Non-coherent circular correlation by partition (Lin, 1998)

Similar to method A. Method C adds amplitude of correlation results from different blocks to find the initial code phase. Then, do FFT to find the carrier frequency.

### D. Delay and multiplication (Lin, 1998)

Multiply the input signals with delayed signals. Do the same operation on local codes. Then correlate them to find the initial code phase. Next, do FFT to find frequency. Delay should be properly designed so that  $2\pi f_s \tau \approx \pi$ , where  $f_s$  is the sampling frequency, and  $\tau$  is the time delay.

### E. Extended replica folding (Yang, 1999 and Yang 2000)

First, in order to implement Doppler removal, the incoming signal spectrum is shifted by a given number of frequency bins along the frequency axis. Then do circular correlation with a locally extended replica folded reference, which covers the entire time uncertainty interval. At last, the folded correlations are coherently integrated over time with several consecutive incoming signal segments in the same way. A more detailed sequential block search technique is illustrated in other research (Yang, 2001).

Table 3.1 gives the performance comparison of different acquisition methods. As illustrated in Table 3.1, methods A~C need larger size FFT than method E for direct P-code acquisition, so they consume more hardware resources. Method D needs huge size FFT, so it is good for hardware design. Method E only uses small size FFT, but it requires a larger size of data.

Table 3.1 Performance comparison of different acquisition methods

Speed	A(1ms)	B(1ms)	C(1ms)	D(10ms)	E(16ms)
Multiplication ( $\times 10^7$ )	7.517	3.781	0.478	1.665	0.26
Addition ( $\times 10^7$ )	15.021	7.445	0.786	3.133	0.49
Bandwidth	1kHz	1kHz	1kHz	100Hz	1kHz
FFT size	8192	4096	4096	655360	1024

Table 3.2 compares different acquisition methods when acquisition probabilities are the same for all methods at around 90%. Methods A, C and D have better performance than methods B and E.

Table 3.2 C/No ratio for different acquisition methods with acquisition detection probability of around 90%

Methods	A (1ms)	B (1ms)	C (1ms)	D (10ms)	E (16ms)
C/No (dB/Hz)	41	44	42	41	47

From the above comparisons, when a large size FFT/IFFT hardware design is available, and also very good carrier to noise ratio performance is required, methods A and C are recommended for hardware design. When only a small size FFT/IFFT is available, also a fast acquisition is a major concern, method E is better for hardware design when GPS signals have a high carrier to noise ratio.

### 3.5 Direct Average Method

Due to the long period of the P-code, a fast correlation method has to be found. An average method can speed up the correlation procedure. First, let's see how the direct average method affects the autocorrelation function.

Step 1: There are 131,072 P-code samples in 2 ms if a sampling frequency is 65.536 MHz and suppose an IF of 12.5 MHz. Average every 128 samples to generate 1024 samples every 2 ms, and call them target 1.

Step 2: Pick up the first 512 points from target 1, pad them with 512 point zeros at the end to generate target 2 signals.

Step 3: Calculate the conjugate of 1024-point FFT for target 2 signals.

Step 4: Calculate 1024-point FFT for target 1 signals.

Step 5: Multiply the results from Step 3 and Step 4, and then do 1024-point IFFT.

Step 6: Select the first 512 elements from the results of Step 5, and discard the other 512 elements. The final results correspond to the autocorrelation function of the 1<sup>st</sup> ms P-code up samples.

Step 7: Shift the P-code samples by 65,536 samples, repeat Steps 1 to 6, get the autocorrelation function of the 2<sup>nd</sup> ms.

Step 8: Similarly, get all autocorrelation functions of 15 ms and sum all correlation results in 15 ms.

Figure 3.4 shows that averaged chunks of data are basically orthogonal to each other.

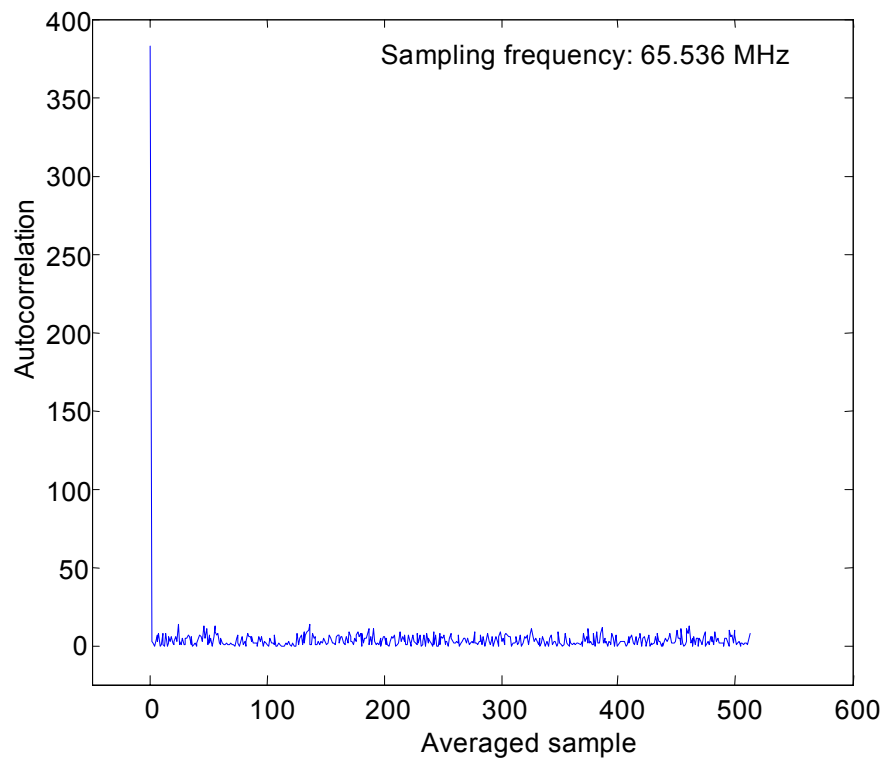


Figure 3.4 Direct average autocorrelation result for P-code

In Figure 3.5, the acquisition margin for a well aligned signal and reference over 1 s is calculated. The mean value is around 26.882 and the variation value is 2.676. As a result, the direct average method is statistically valid for segments of P-code up samples in different milliseconds. Compared with the direct acquisition margin, which is equal to 25.954 when no average is used for acquisition, the direct average method has the similar acquisition margin. This is the case when there is no noise added. However, the direct average method avoids using the extremely large size FFT hardware for 50 ms samples.

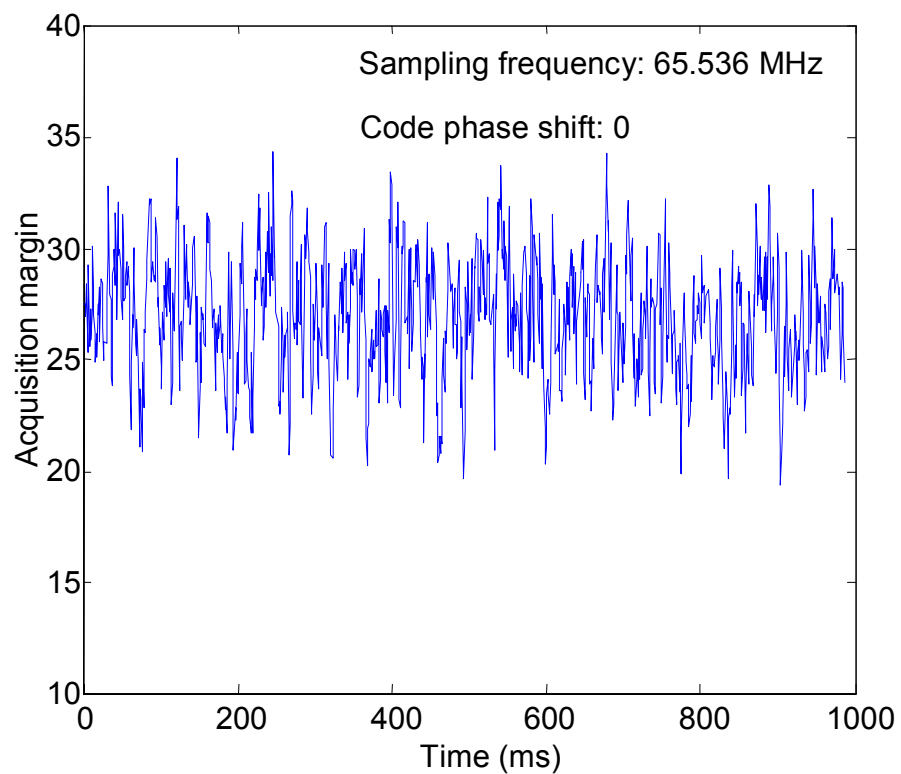


Figure 3.5 Direct average method: acquisition margin distribution over 1 s



Based on the results from Figure 3.4 and Figure 3.5, the direct average method can be used to reduce the time for acquiring P-code signals. The block processing technique (Haag, 1999) can be used to suppress noise by processing correlations in several blocks and then adding them together. After GPS signals are demodulated, the further acquisition procedure is given as follows:

Step 1 to Step 8 are similar to those described above for autocorrelation calculation except the target 1 reference is locally generated P-code samples rather than incoming signals.

Step 9: If a correlation peak is not detected, shift P-code samples by another 1 ms as reference. Repeat Steps 1 to 9 until a correlation peak is acquired.

Step 10: If a correlation peak is detected at location  $m$ , the peak location has a code phase resolution of 128 samples. If a more accurate peak location is needed, go to step 11.

Step 11: Shift the demodulated GPS signals by  $(m-1)*128$  samples. Then use the first 1024 signal samples to do a 1024-point FFT.

Step 12: Take the first 512 samples from the reference and pad them with 512 zeros. Obtain the conjugate of a 1024-point FFT for this new reference.

Step 13: Multiply results from Steps 11 and 12, and then do a 1024-point IFFT.

Step 14: Select the first 512 elements from the results of Step 13, and discard the other 128 elements. The final results should show the correlation peak location at  $n$ . Then use  $(m*128+n-128)$  to get the exact sample location.

Example 1: GPS signals are 42 samples in advance of reference. This example is given to show the simulation results using the above acquisition procedure.

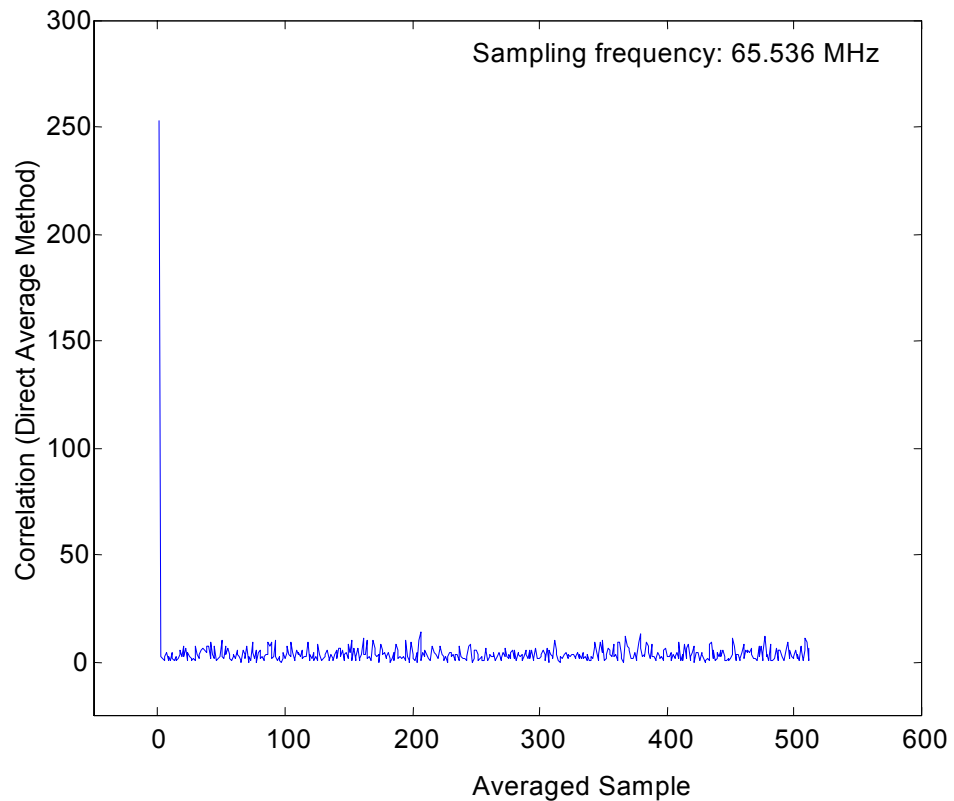


Figure 3.6 Direct average acquisition example

Figure 3.6 is a result of Step 1 to Step 8 in the above acquisition procedures using the direct average method. The acquisition correlation peak is located in the zero shift position, which is within the 128-sample resolution.

The result of Step 9 to Step 12 is illustrated in Figure 3.7. The correlation peak is located at the 86-sample shift position. Since  $(0 \cdot 128 + 86 - 128)$  is equal to  $-42$ , the acquired location is 42 samples in advance of the reference.

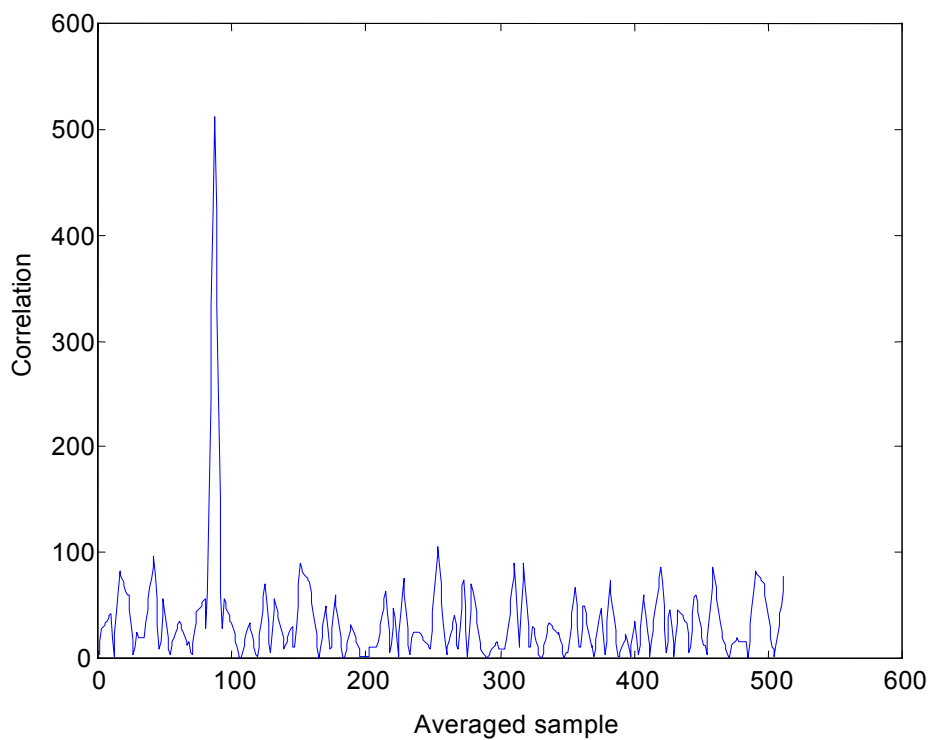


Figure 3.7 Correlation resolution improvement

### 3.6 Effect of Code Phase Shift

The acquisition results are optimum when the reference samples have a zero code phase shift relative to the incoming signal samples. Obviously, it is very important to study how the code phase shift affects the maximum correlation peak value and the acquisition margin. Figures 3.8 and 3.9 correspond to a code phase shift of 64 samples.

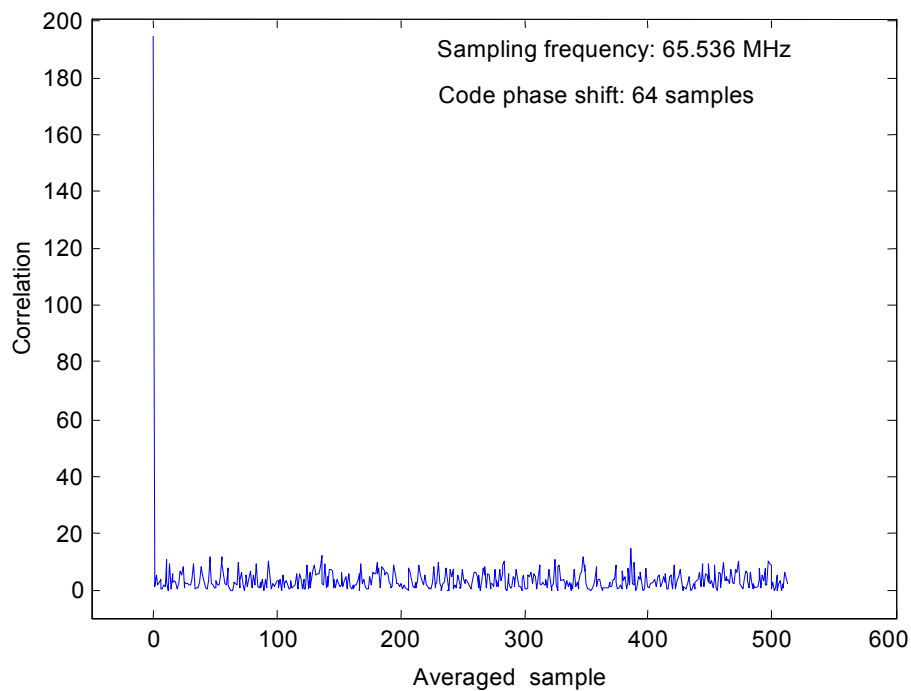


Figure 3.8 Direct average method: correlation results with a code phase shift of 64 samples

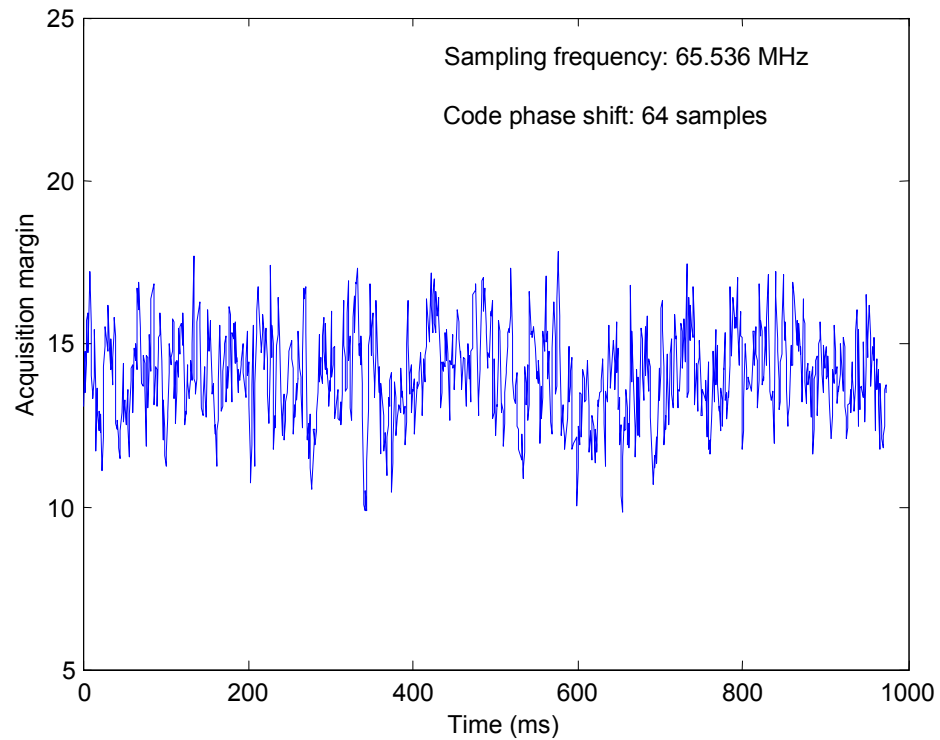


Figure 3.9 Direct average method: acquisition margin distribution over 1s with a code phase shift of 64 samples

Compared with Figures 3.4 and 3.5, the correlation peak value and the acquisition margin decrease a lot as can be observed from Figures 3.8 and 3.9. The mean of Figure 3.9 is 14.042, and the standard deviation is 1.424.

Figure 3.10 is a two dimensional plot of the relation between the correlation peak value and the code phase shift in samples. The correlation peak decreases almost linearly within the 128-sample shift. When there is a shift of 64 samples, there is almost a half correlation peak value loss, or a 3 dB loss.

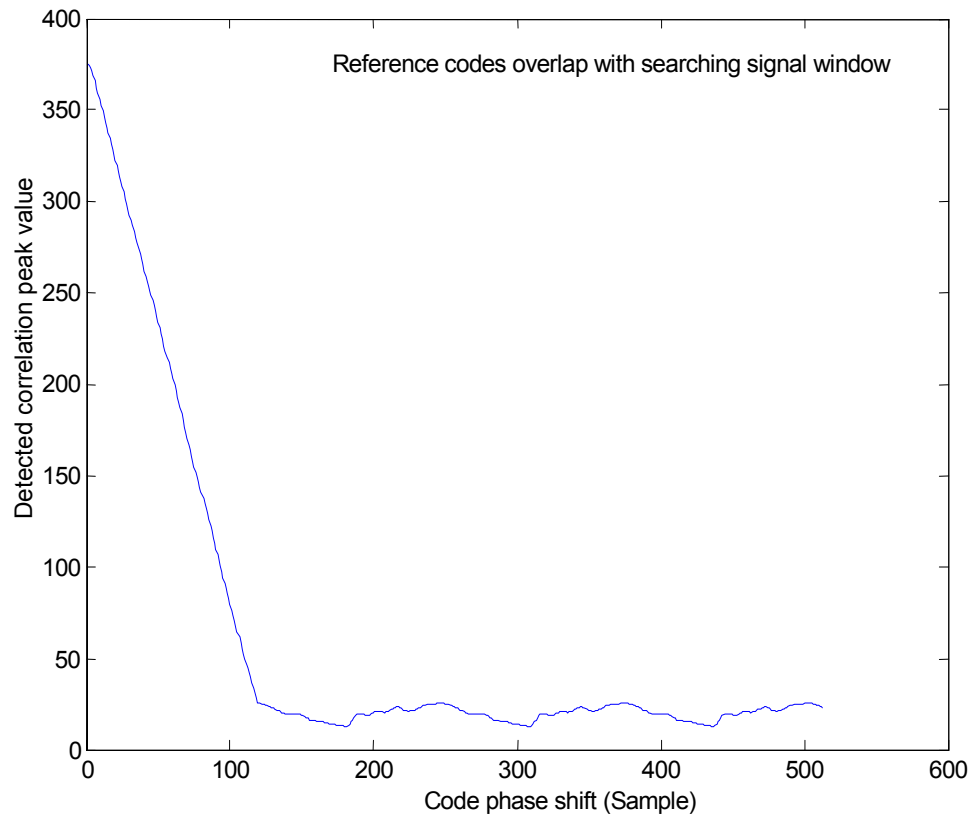


Figure 3.10 Direct average method: detected correlation peak value change, case 1

The triangle in Figure 3.11 repeats for 1 ms with a cycle of 128 samples, which means that if reference codes are hidden inside the first half of the 2 ms searching window, correlation peak can always be detected.

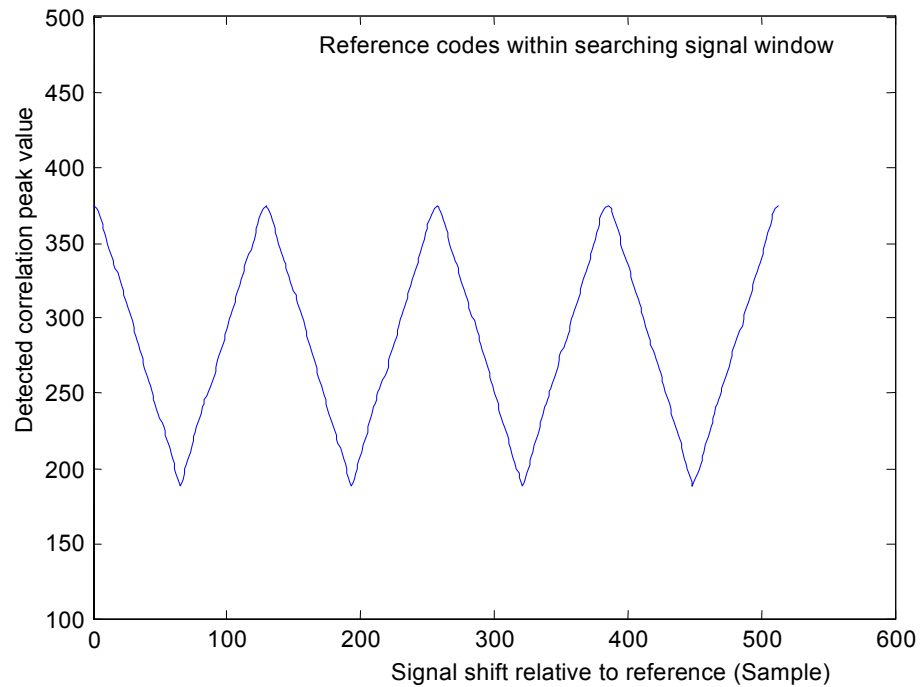


Figure 3.11 Direct average method: detected correlation peak value change, case 2

In order to make up the correlation energy loss in the dip region in the above figures, a new algorithm is presented in the next section to improve the acquisition performance.

### 3.7 Overlap Average Method

Since the detected correlation peak value changes almost linearly within the 128-sample shift range, a linear combination algorithm may gain more correlation energy and

make Figures 3.10 and 3.11 as flat as possible. Those are the goals of the overlap average method. The basic operation of this method is similar to the direct average method except the reference is different. Here, a second reference is used which is overlapped with the old reference by 64 P-code up samples. Averaging two reference codes will produce the new reference used as the first experiment in the overlap average method. In Figures 3.12 to 3.15, the correlation value and the statistical acquisition margin distribution are calculated without a code phase shift and with a code phase shift of 64 samples.

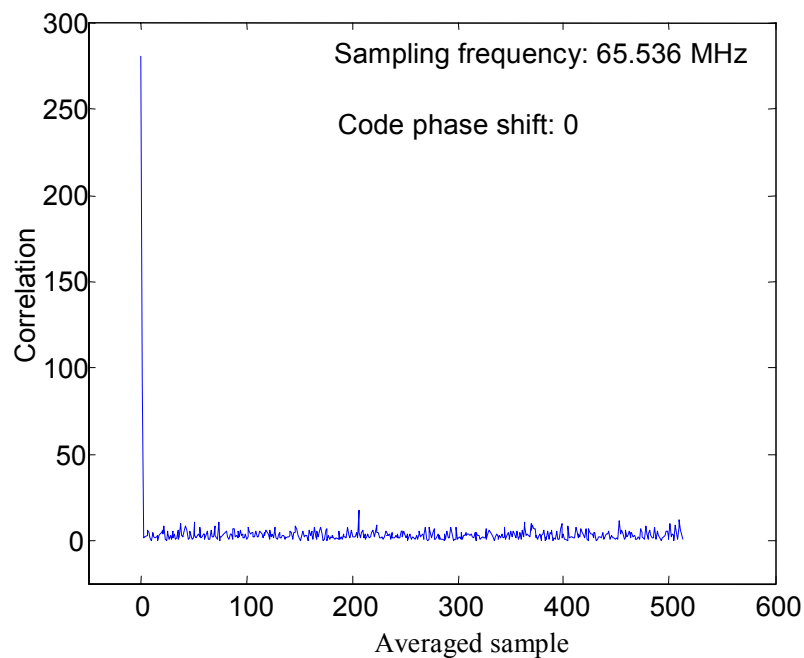


Figure 3.12 Correlation result using overlap average method

When there is no code phase shift, the optimum correlation value and acquisition margin value are achieved. Figures 3.13 and 3.14 show that the overlap average method



has a small decrease of these values. The mean of Figure 3.13 is 23.596, and the standard deviation is 2.653.

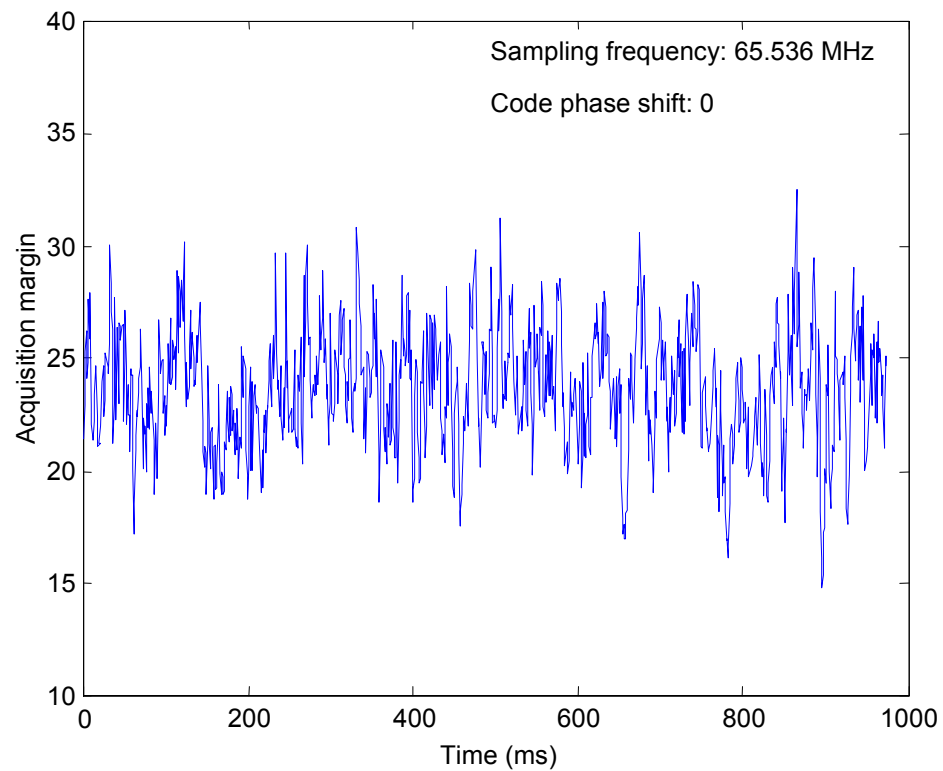


Figure 3.13 Statistical acquisition margin result for overlap average method

The advantage of the overlap average method shows up when there is a code phase shift. Figure 3.14 illustrates a worse case with a code phase shift of 64 P-code up samples. The correlation value remains similar to that without the code phase shift.

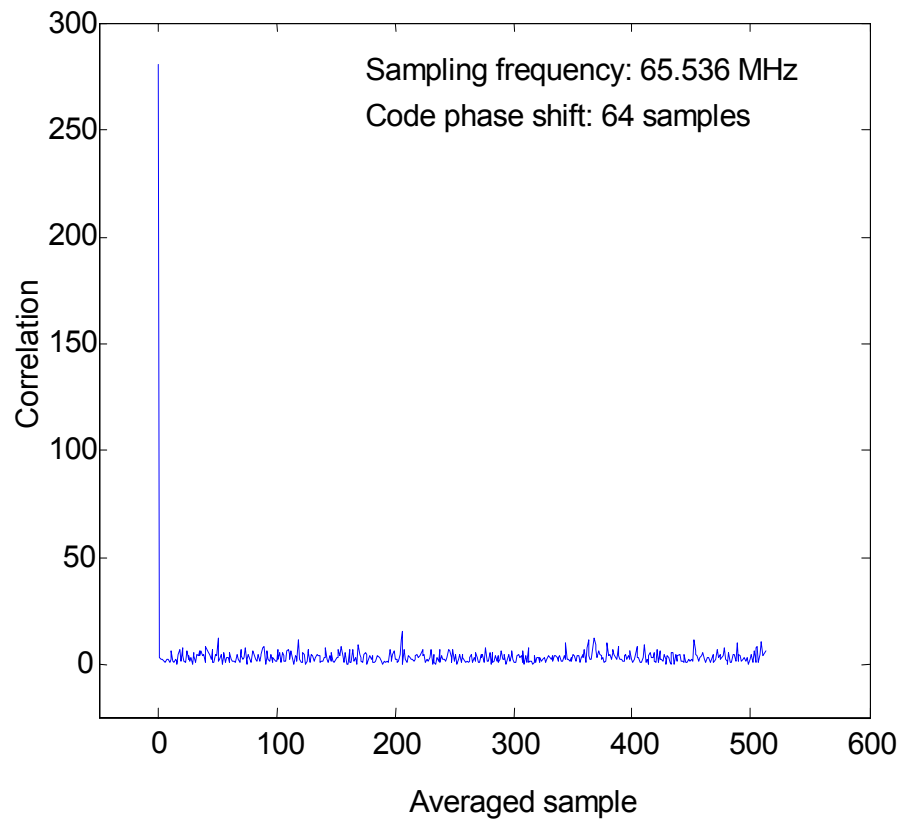


Figure 3.14 Correlation result for overlap average method with a code phase shift of 64 samples

In Figure 3.15 the acquisition margin in case of a code phase shift of 64 P-code up samples remains similar to the results without the code phase shift. The energy compensation is around 2 to 3 dB compared with the direct average method. The mean of Figure 3.15 is 23.885, and the standard deviation is 2.487.

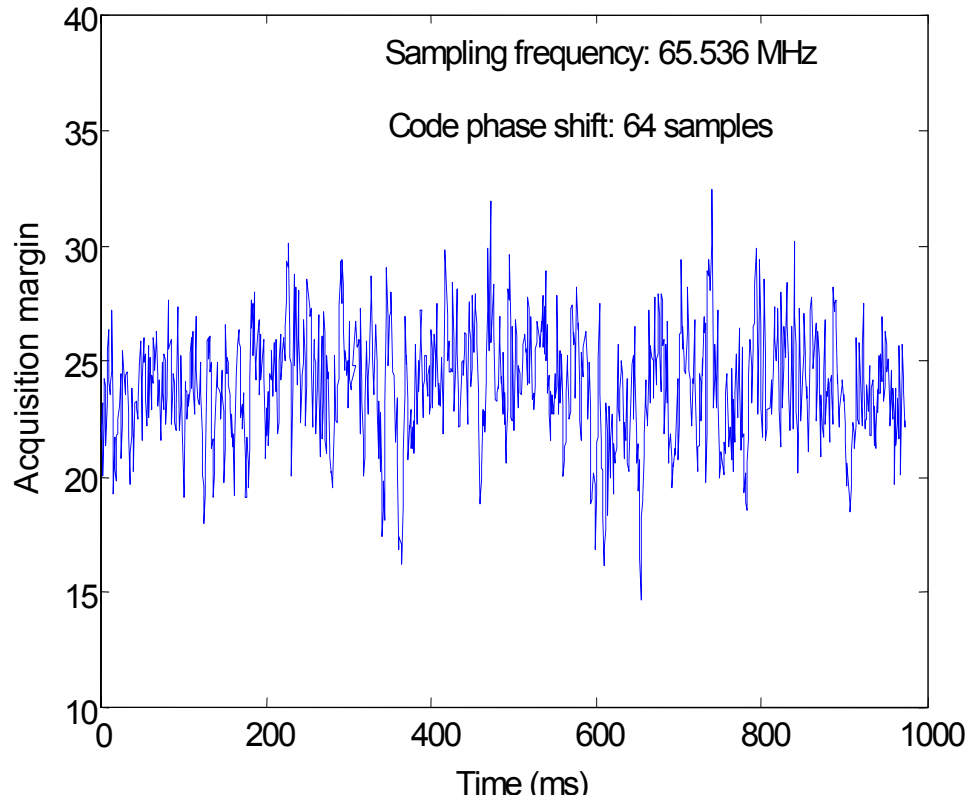


Figure 3.15 Statistical acquisition margin result for overlap average method with a code phase shift of 64 samples

In addition, the detected correlation peak changes are shown in Figures 3.16 and 3.17. As can be seen, the maximum correlation value curve becomes relatively flat and correlation energy loss, in the worse case, is decreased. The flat parts in Figures 3.16 and 3.17 prove the linear combination really works. In order to find out why there are still dips in Figure 3.17, it's necessary to explore the correlation peak location detection difference by using two different references.

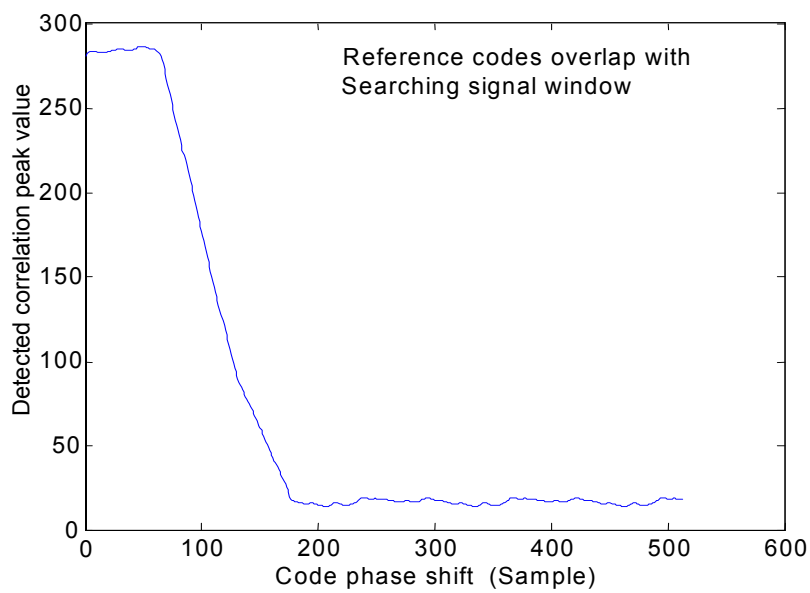


Figure 3.16 Overlap average method: two reference overlap, case 1

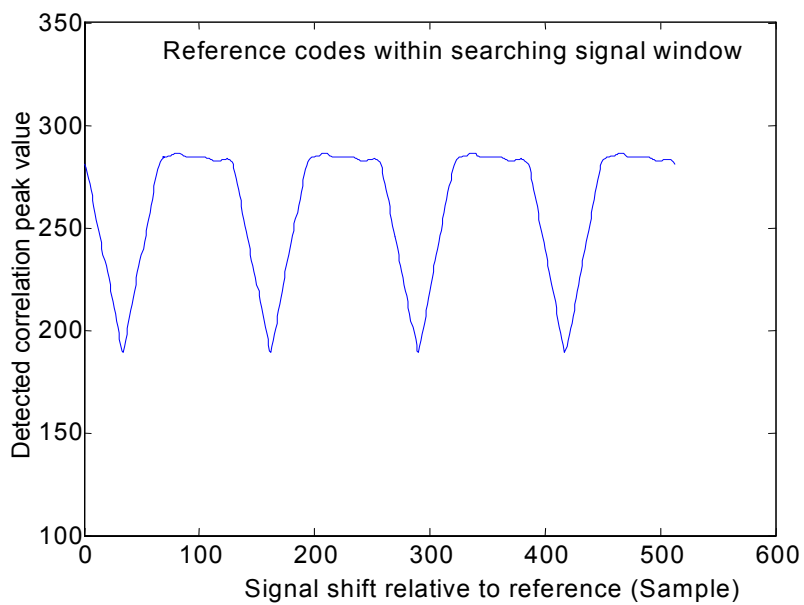


Figure 3.17 Overlap average method: two reference overlap, case 2

Suppose grouping the signal by 128 samples. In Figure 3.18, when the code phase shift is between 0 to 64 samples, reference 1 detects the correlation peak location in the 1st group, but reference 2 indicates the 2<sup>nd</sup> group. As a result, two correlation peaks don't have a chance to be added up. That's why dips happen in Figure 3.16. However, when the code phase shift is between 64 to 128 samples, two references detect the same correlation peak location so that the two correlation peaks are summed up into a bigger value, which results in the flat part in Figure 3.17.

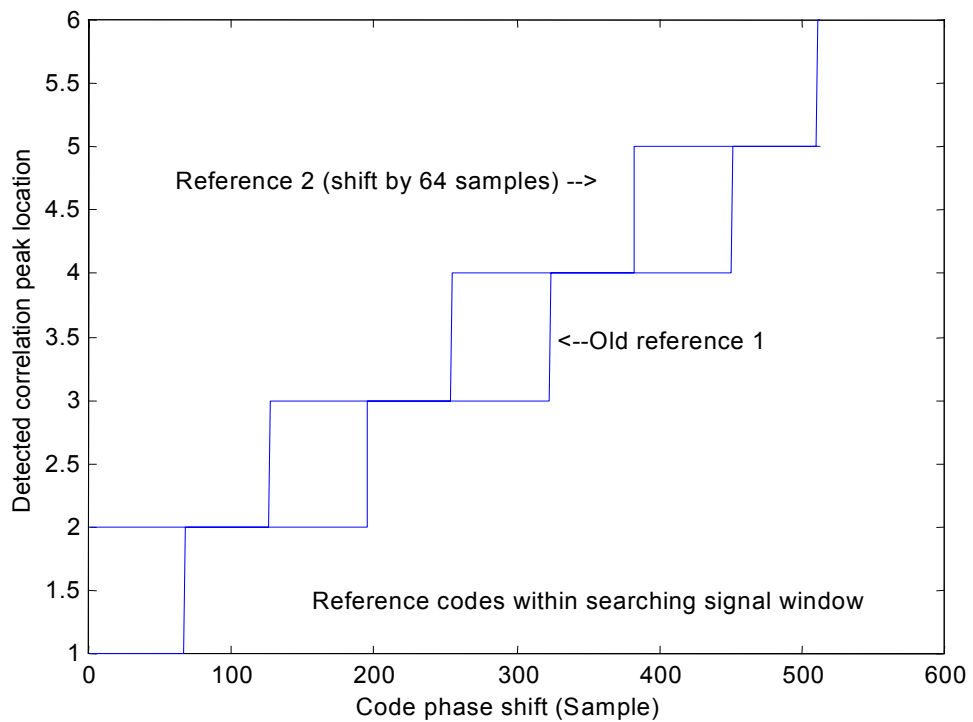


Figure 3.18 Peak correlation location detection difference using two different references

Theoretically, since actually we only utilize the positive part of linear correlation peak curve property, the negative part is not used. We can also use the cycling property showed in Figure 3.11. Accordingly, a new procedure is described next.

First, let's take two new references. One has a shift of 64 samples, and another one has a shift of 128 samples. These two references will shift the pattern in Figure 3.18 right by 64 samples as illustrated in Figure 3.19. Next combine these two into a new reference, and then the corresponding curve in Figure 3.17 will also shift right by 64 samples to generate a new curve showed in Figure 3.20.

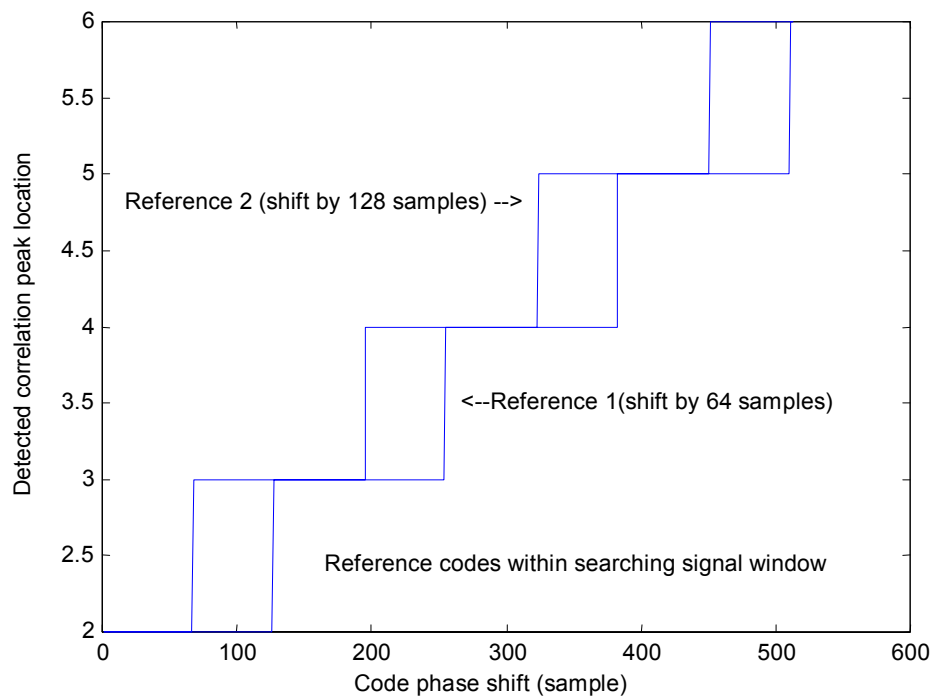


Figure 3.19 Peak correlation location detection pattern change

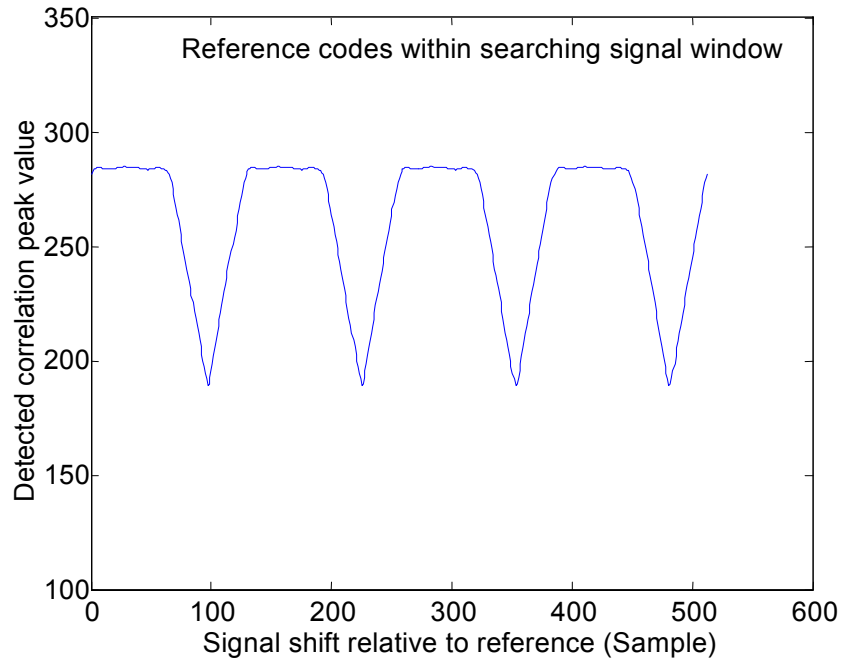


Figure 3.20 Overlap average method: two reference overlap case 3

Suppose the reference is within the signal searching window. Then the correlation peak value satisfies the following equation:

$$\begin{aligned}
 \text{correlation peak} &= \max \{ \sum [(x_m + x_{m+128}) * (y_0 + y_{64} + y_{64} + y_{128}) / 4] \} \\
 &= \max \{ \sum [x_m y_0 + 2 * (x_m y_{64} + x_{m+128} y_{64}) + x_m y_{128} + x_{m+128} y_0 + x_{m+128} y_{128}] / 4 \} \\
 &= \max \{ \sum [(x_m y_0 + x_m y_{128}) + 2 * (x_m y_{64} + x_{m+128} y_{64}) + (x_{m+128} y_0 + x_{m+128} y_{128})] / 4 \} \\
 &= \max \{ \sum [x_0 y_0 + 2 * x_0 y_0 + x_0 y_0] / 4 \} \\
 &= \max (\sum x_0 y_0)
 \end{aligned} \tag{3.6}$$

where  $x_0$  is the original signal without any shift,  $x_m$  and  $x_{m+128}$  are the signals with a shift of  $m$  samples and  $(m+128)$  samples;  $y_0$ ,  $y_{64}$  and  $y_{128}$  are the references with zero shift, 64-sample shift and 128-sample shift relative to signal  $x_0$  respectively.

There are pairs in Equation 3.6, which complement each other on correlation peak values. These pairs are  $(x_m y_0, x_m y_{128})$ ,  $(x_m y_{64}, x_{m+128} y_{64})$ , and  $(x_{m+128} y_0, x_{m+128} y_{128})$ . As a result, the maximum correlation peak value has no relation with the shift variable  $m$  and it should be the maximum correlation peak value. The correlation peak curve in Figure 3.21 has values a little less than the maximum value because the linear relationship described before is not perfect.

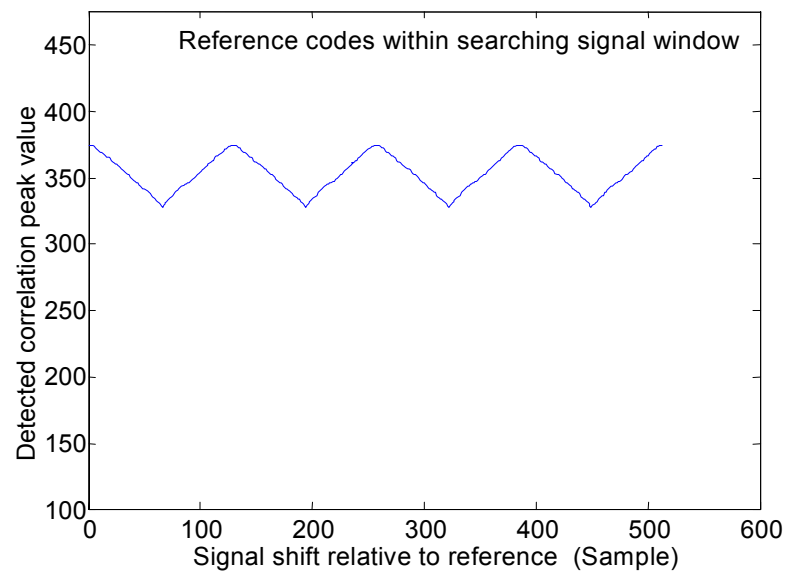


Figure 3.21 Overlap average method by using three references



Actually the final overlap average method uses three references. The first reference has no code phase shift, the second has a code phase shift of 64 samples, and the third has a code phase shift of 128 samples. Add one copy of the first reference, two copies of the second reference, and one copy of the third reference, and then average them to get a new reference. Also add incoming signals with signals having 128-sample code phase shift to obtain new signals. At last, correlate new signals with the new reference. In this way, almost optimum acquisition performance can be achieved.

### 3.8 Noise Effect

The above simulation results are ideal since noise is not considered. Table 3.3 lists characteristics of different acquisition schemes without noise added. Case 1 corresponds to the best situation and case 2 to the worse situation.

First, the direct acquisition has a big mean acquisition margin. But it needs a huge size FFT, which is almost impossible in the hardware design currently. Second, in the best case, the direct average method has a big correlation peak and mean acquisition margin. However, when there is a relative code phase shift such as 64 samples, the correlation peak value decreases a lot and the 2<sup>nd</sup> peak value doesn't change. As a result, the mean acquisition margin becomes much lower. Compared with the direct average method, the overlap average method is not so sensitive to the code phase shift. In

conclusion, the overlap average method has overall good performance in different situations. Also a small size FFT is available by averaging and it is good for hardware implementation.

Table 3.3 Correlation and acquisition margin characteristics using different schemes without noise

*Case 1: no shift  *Case 2: a relative shift of 64 samples		No Noise				
		Direct	Direct Average		Overlap Average	
			Case 1	Case 2	Case 1	Case 2
Correlation (1 ms)	True Peak	$6.5536 \cdot 10^4$	382.848	194.477	288.662	285.549
	Second Peak	$2.6500 \cdot 10^3$	13.919	14.824	13.452	13.178
Acquisition	Mean	25.954	26.882	14.042	23.596	23.885
Margin	Standard Deviation	1.841	2.676	1.424	2.653	2.487

In the real world, noise always exists in GPS signals. So it is important to analyze the above algorithms with noise added.

In Figure 3.22, when noise is added, the noise floor increases a lot in direct acquisition.

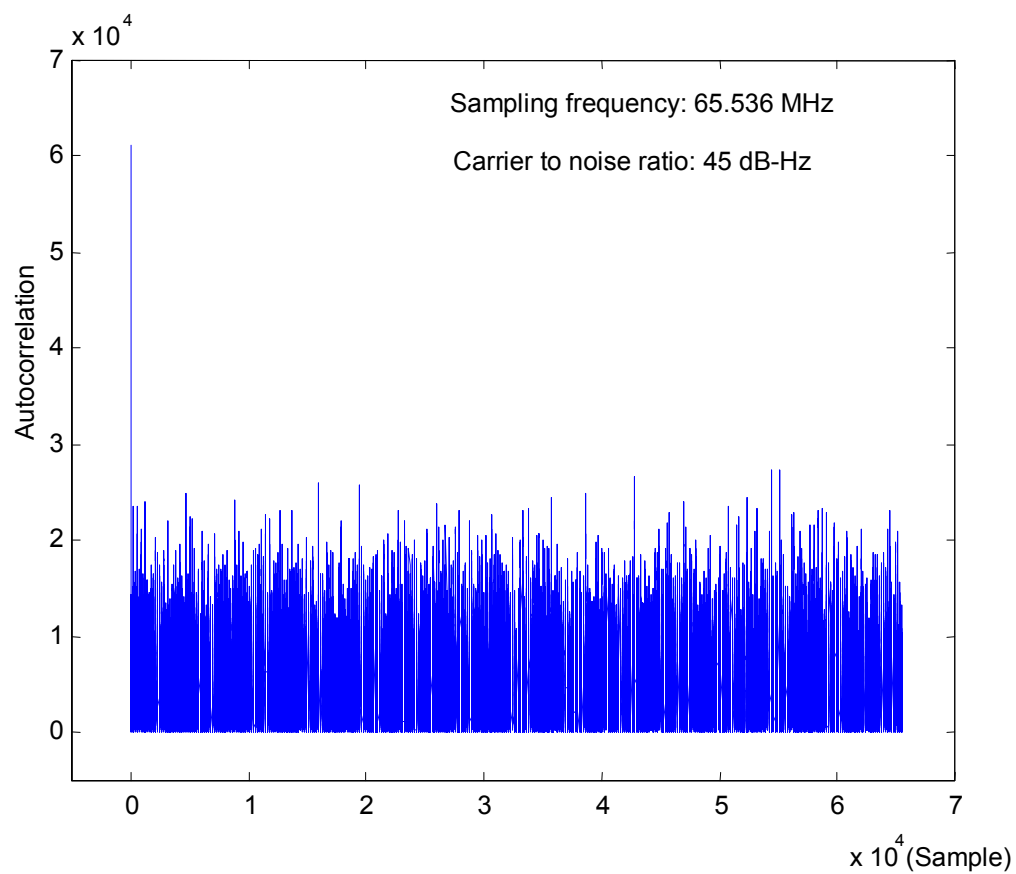


Figure 3.22 Noise effect on direct autocorrelation result of P-code up samples over 1 ms

In Figure 3.23, when noise is added, the mean acquisition margin decreases.

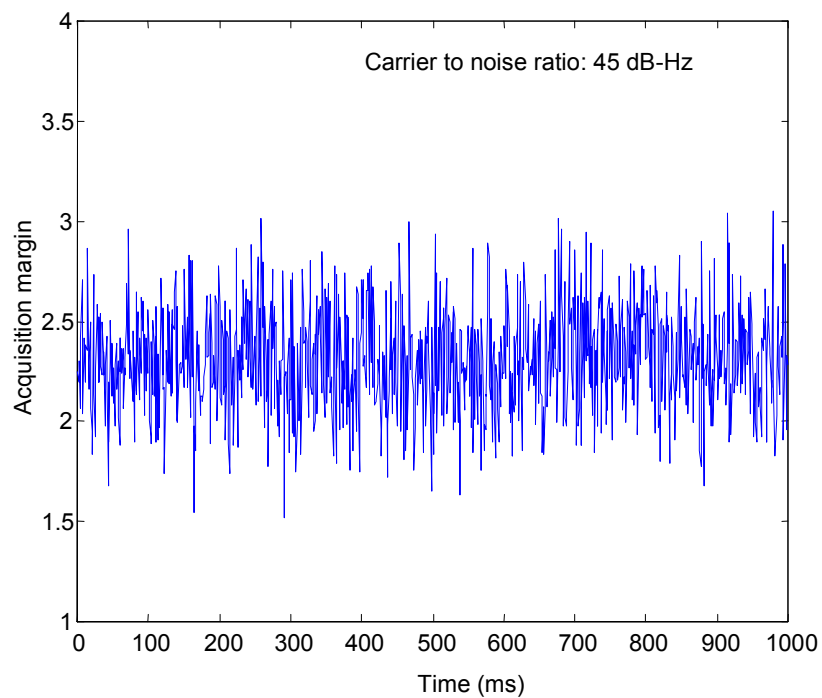


Figure 3.23 Direct acquisition: noise effect on acquisition margin result

As illustrated in Figure 3.24, the direct average method doesn't perform well when there's a relative big code phase shift such as 64 samples when noise is added. The detected correlation peak value decreases a lot.

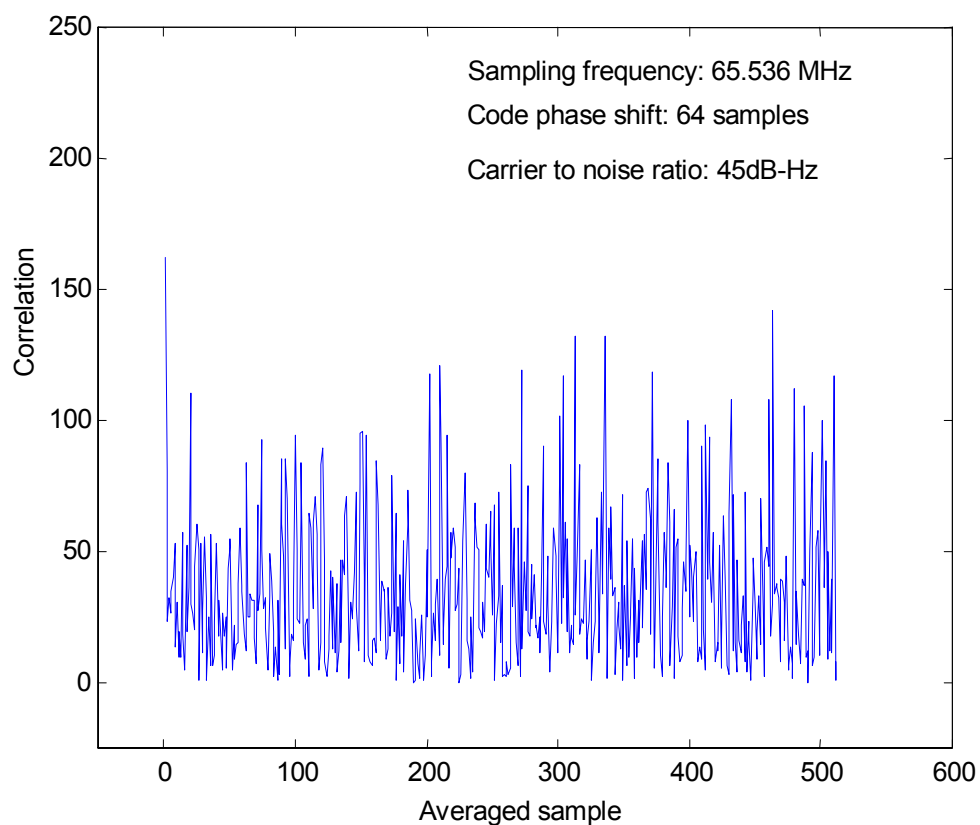


Figure 3.24 Direct average method: correlation result with added Gaussian noise

When there's a relative big code phase shift such as 64 samples, Figure 3.25 shows the acquisition margin decrease a lot in case of added noise.

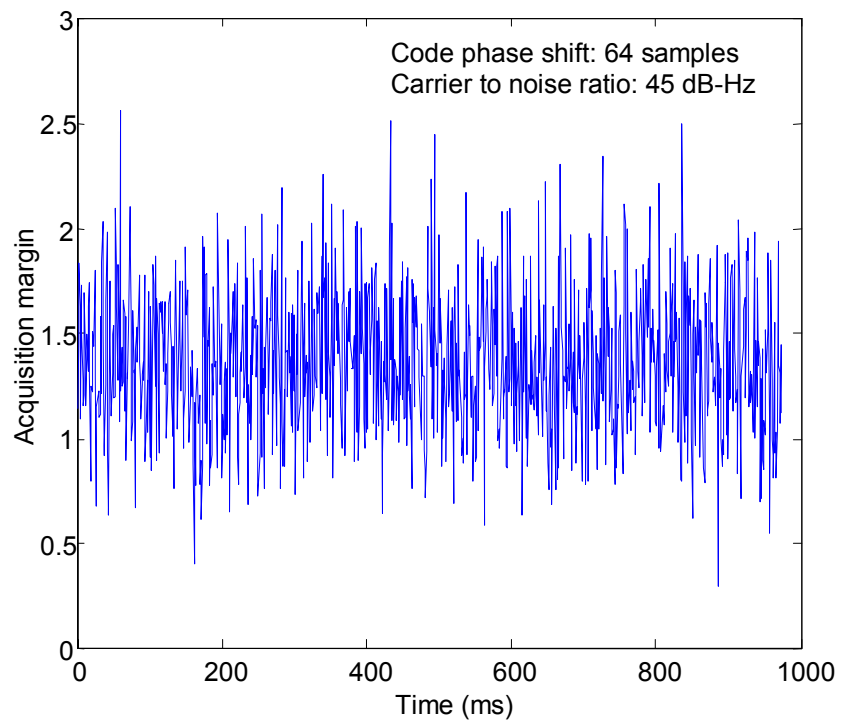


Figure 3.25 Direct average method: noise effect on acquisition margin

Figure 3.26 shows that the detected correlation peak still decreases almost linearly with code phase shift in case of added noise using the direct average method.

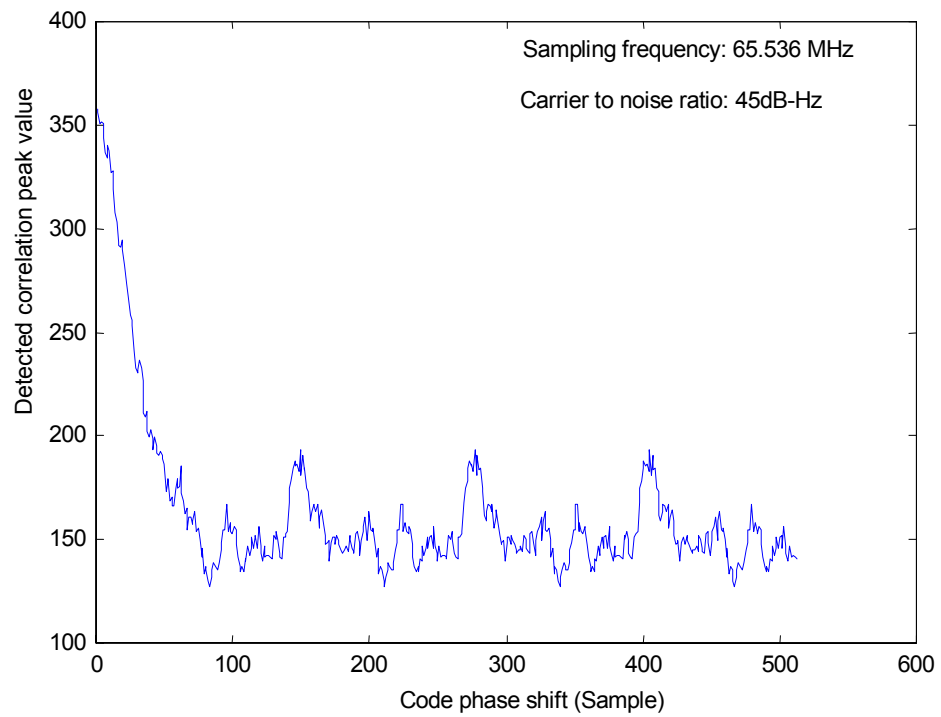


Figure 3.26 Direct average method: noise effect on detected correlation peak change with code phase shift

When noise is added, the overlap average method doesn't decrease detected correlation peak and acquisition margin much in Figures 3.27~3.29. So it performs much better than the direct average method. If reference codes are within the searching signal window, the detected correlation peak value curve keeps relatively flat with different code shifts.

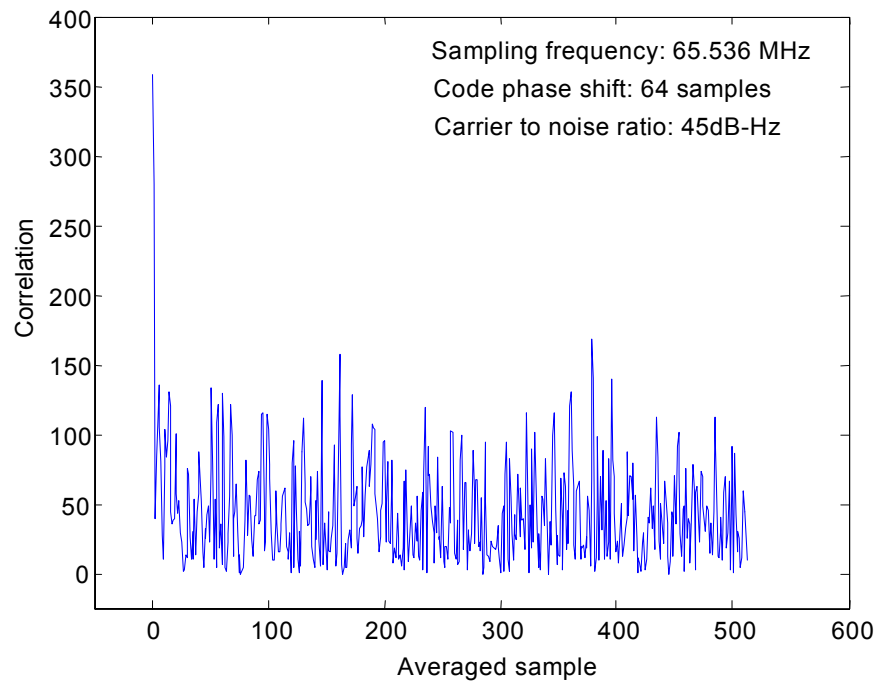


Figure 3.27 Overlap average method: correlation result with added Gaussian noise



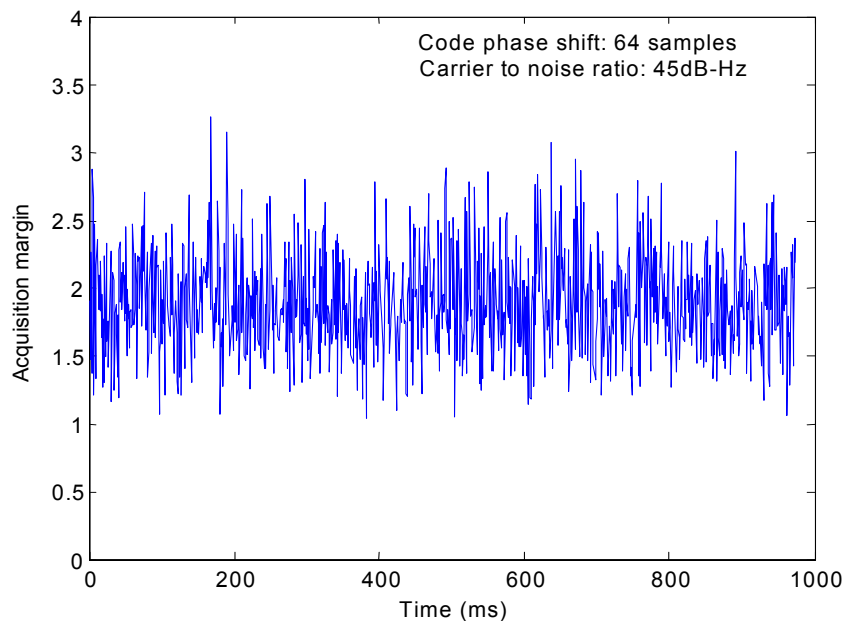


Figure 3.28 Overlap average method: noise effect on acquisition margin

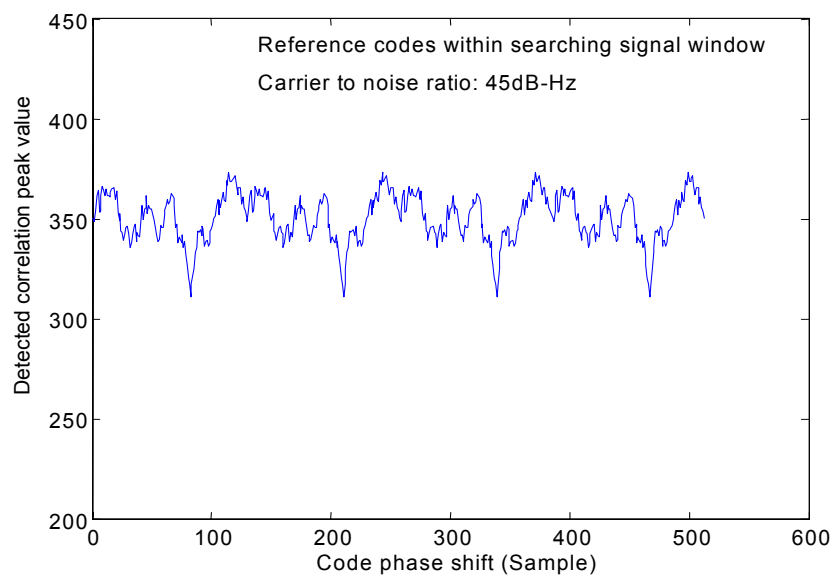


Figure 3.29 Overlap average method: noise effect on detected correlation peak value

Table 3.4 quantifies the correlation and acquisition margin using different acquisition schemes when noise is added.

Table 3.4 Correlation and acquisition margin characteristics using different schemes with added noise

*Case 1: no shift  *Case 2: a relative shift of 64 samples		Added Gaussian Noise (carrier to noise ratio=45dB-Hz)				
		Direct	Direct Average		Overlap Average	
			Case 1	Case 2	Case 1	Case 2
Correlation	True Peak	$6.125 * 10^4$	358.04	162.18	350.258	358.696
	Second Peak	$2.726 * 10^4$	140.89	142.00	162.963	168.395
Acquisition	Mean	2.308	2.722	1.364	2.186	1.897
Margin	Standard Deviation	0.255	0.409	0.341	0.409	0.366

Furthermore, 16 ms signals are taken to do acquisition using the direct average and the overlap average method with the same misdetection probability but a different carrier to noise ratio. The direct average method requires around a 3 dB-Hz less carrier to noise ratio compared with the overlap average method according to Figures 3.30 and 3.31.

In these figures, although the carrier to noise ratio using both methods is relatively high, these two methods have the advantages of reducing the time for the code phase

search and using a small size FFT. So there is a tradeoff between the code phase search speed and the carrier to noise ratio.

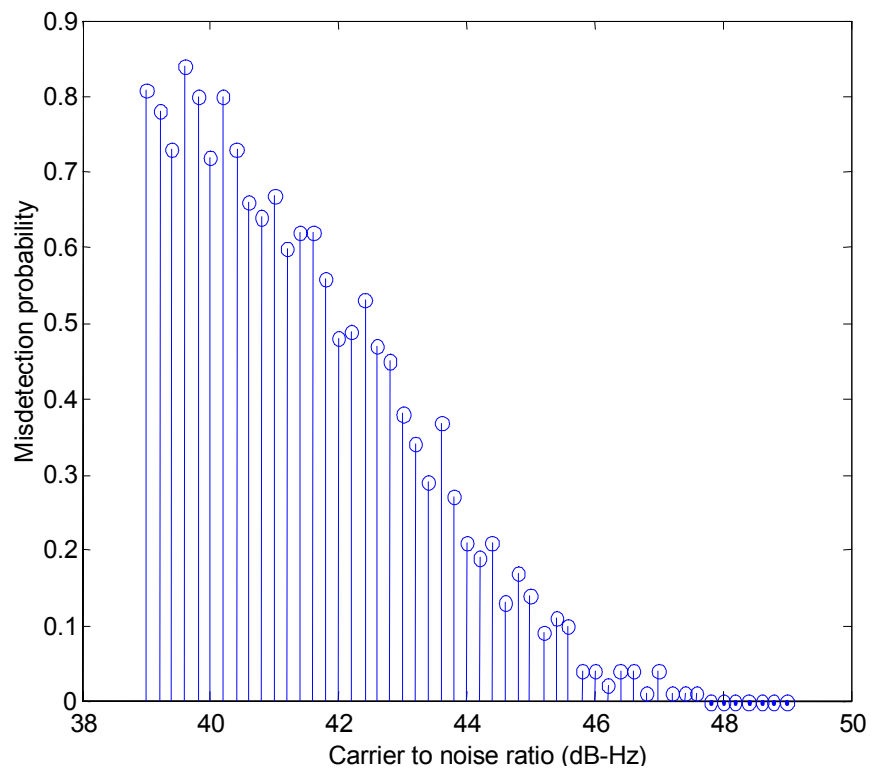


Figure 3.30 Direct average method: misdetection probability with different carrier to noise ratio when the signal code phase shift is 64 samples

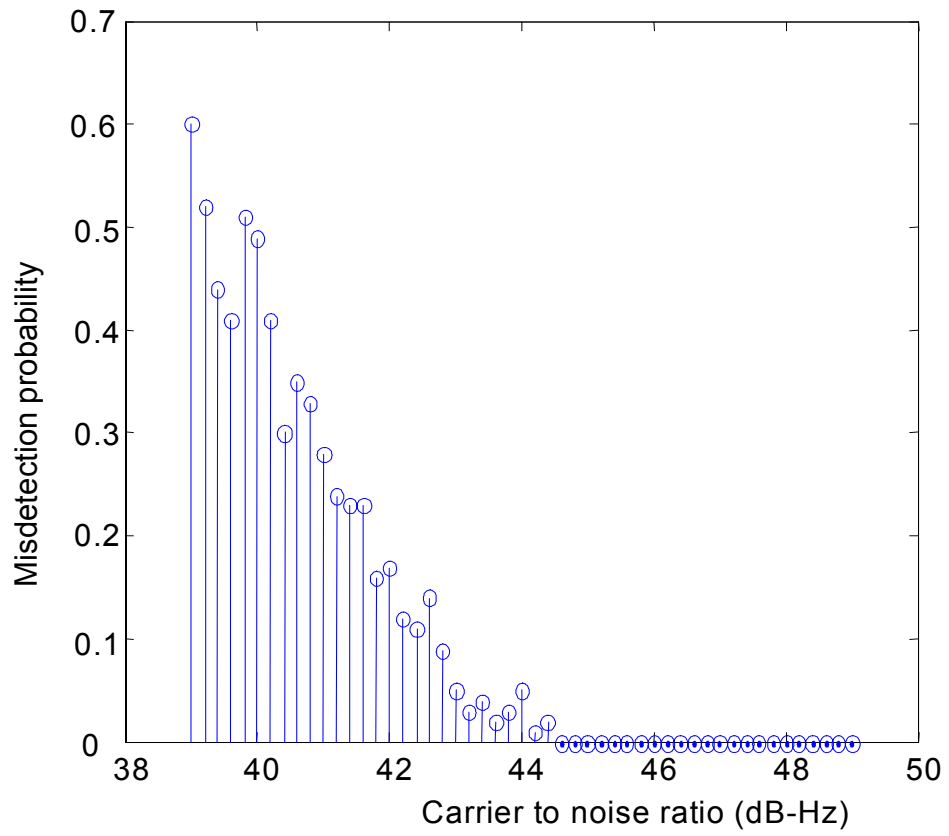


Figure 3.31 Overlap average method: misdetection probability with different carrier to noise ratio when the signal code phase shift is 64 samples

A statistical analysis was made to compare two acquisition methods. A study of the misdetection probability statistical distribution in 1 s using the direct average method illustrated in Figure 3.32 indicates that the misdetection probability has a mean of 0.5318 and a standard deviation of 0.055 when the carrier to noise ratio is equal to 42 dB-Hz.

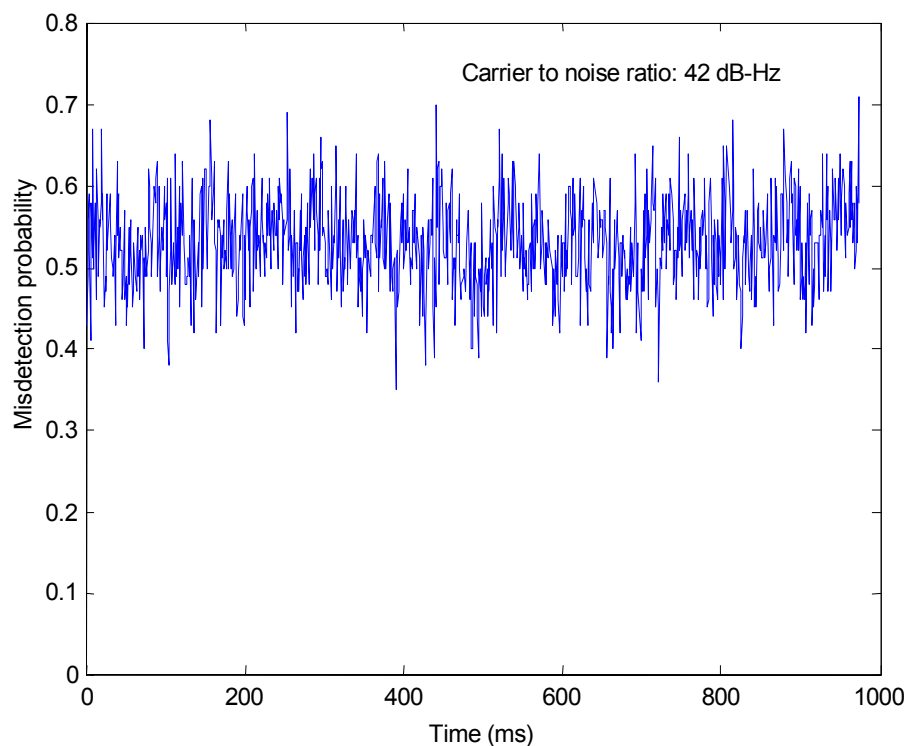


Figure 3.32 Direct average method misdetection probability distribution in 1 s with carrier to noise ratio equal to 42 dB-Hz

In case of using the overlap average method illustrated in Figure 3.33, the misdetection probability has a mean of 0.5658 and a standard deviation of 0.0512 when the carrier to noise ratio is equal to 39 dB-Hz. So with the almost the same misdetection probability, the overlap average method has a gain of around a 3 dB-Hz carrier to noise ratio higher than the direct average method statistically. In conclusion, the overlap average method has overall good performance with or without noise added.

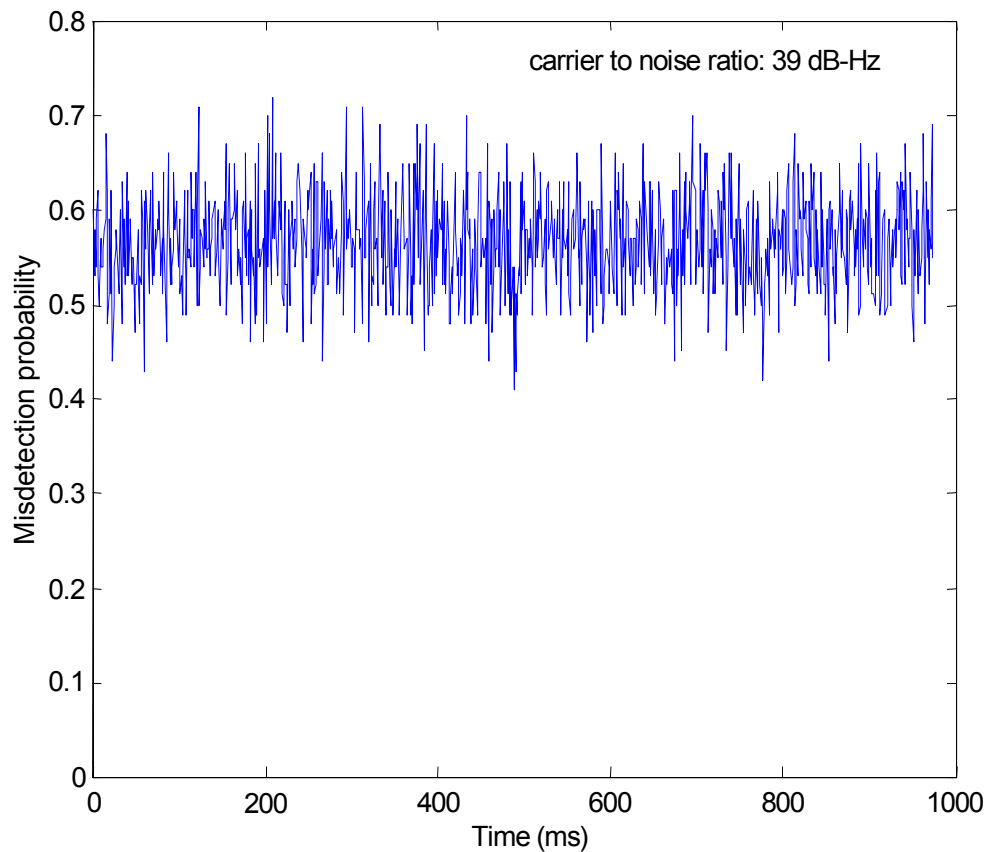


Figure 3.33 Overlap average method misdetection probability distribution in 1 s with carrier to noise ratio equal to 39 dB-Hz

## 3.9 Example

Example: Suppose reference codes are within 2 ms searching signal window. The relative code phase shift between signal and reference is 61,746 samples.

Both direct average and overlap average methods detect the correlation peak at a code phase shift equal to 482 in Figures 3.34 and 3.35. The result has a resolution equal to 128 samples. Searching at least 256 samples near the correlation peak location can refine the resolution. Since a 1024-point FFT is used, 1024 samples are taken for the searching purpose.

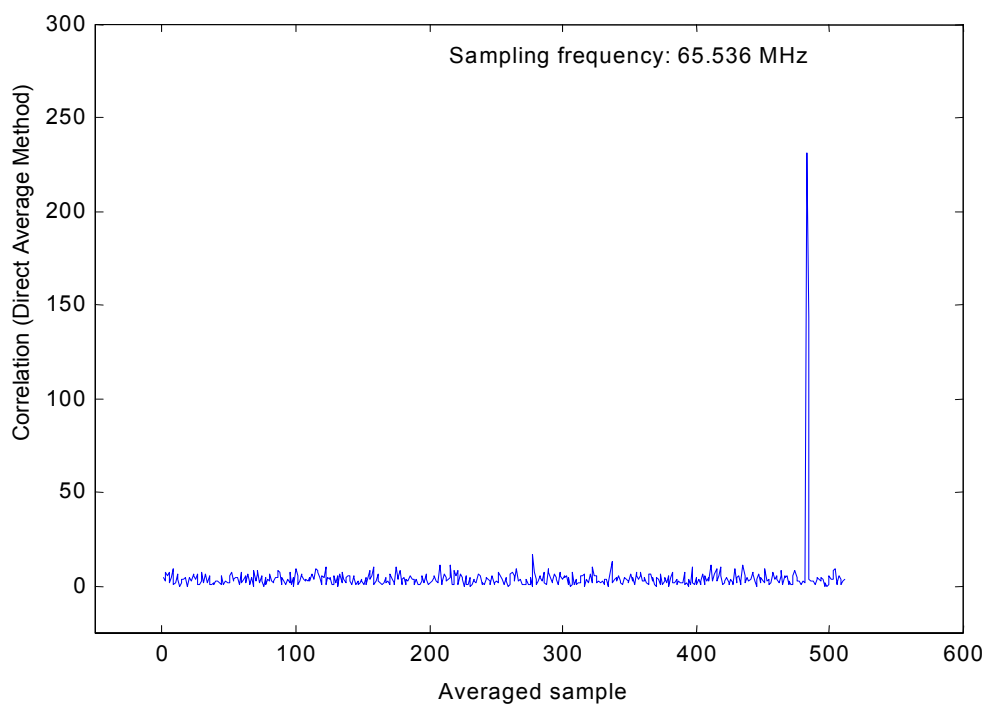


Figure 3.34 Acquisition by the direct average method

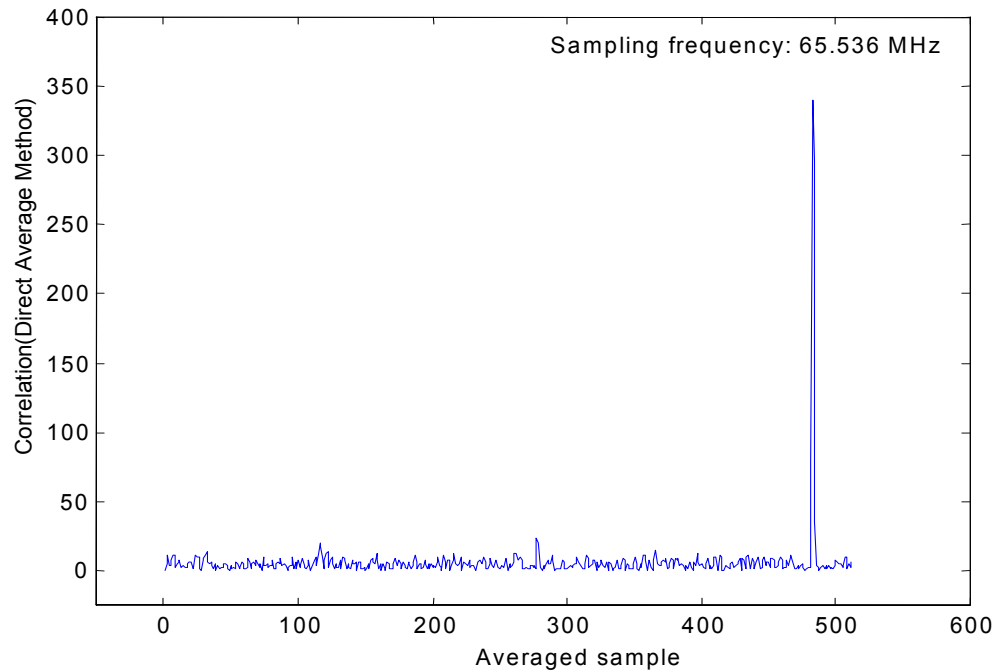


Figure 3.35 Acquisition by the overlap average method

Shift 2ms-signals by  $481 \times 128$  samples. Then take the first 1024 points to do a 1024-point FFT. Next take the first 512 points from reference and pad them with 512 zeros to get a new reference. Use the new reference to get the conjugate of the 1024-point FFT, and multiply them with previous 1024-point FFT results. Then do a 1024-point IFFT. At last, keep the first 512 results from the IFFT and discard the other half.

The final results are illustrated in Figure 3.36. The peak location has a shift of 178 samples.



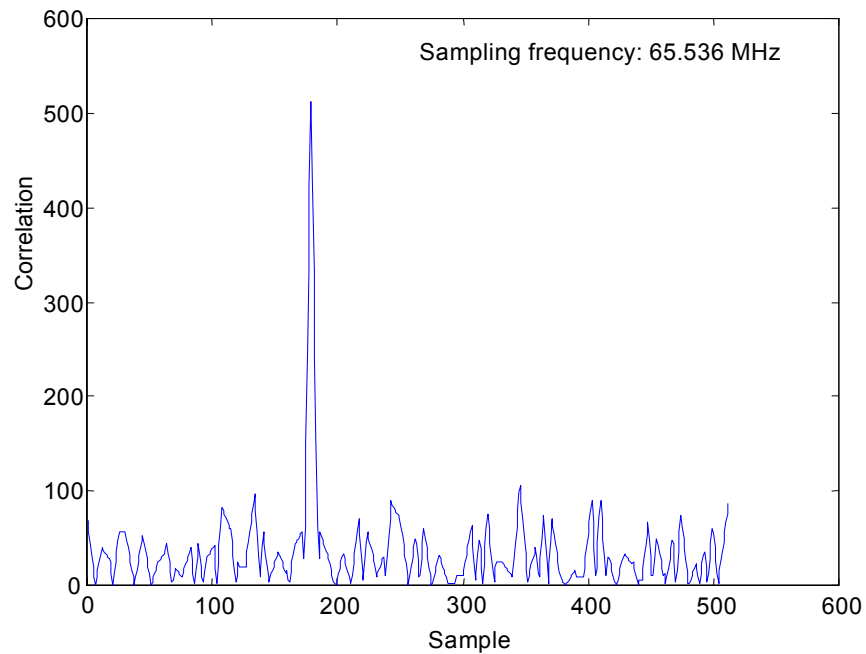


Figure 3.36 Resolution improvement by correlating 128 samples

As a result, the final shift is located at  $482 \cdot 128 + 178 - 128 = 61,746$  samples in advance of reference.

### 3.10 Conclusion

The P-code or even segments of the P-code have good autocorrelation and cross-correlation properties. Because a segment of the P-code is considered as nonperiodic, its correlation function is computed by the FFT search and zero padding method. Many

reported direct P-code acquisition algorithms listed in this dissertation need a large size FFT/IFFT, which requires large hardware resources and adds hardware design complexity. In this chapter, the direct average method is proposed to reduce the direct P-code acquisition code phase searching time using 1024-point FFT/IFFT. This method is based on the symmetrical linear P-code correlation curve discovered by using averaging.

Statistical analysis was performed to show the validness of the direct average method. Another overlap average method was described in this chapter to make up the correlation energy loss when there is a large code phase shift between the GPS incoming signals and the locally generated reference. Even with the added Gaussian noise, the overlap average method has better carrier to noise ratio performance than the direct average method statistically when the reference has a code phase offset relative to the GPS signal. Finally, one example is provided to illustrate how to use both methods.

## Chapter 4

### GPS P-code Generation

The P-code generator as shown in Figure 4.1 is a critical component in P-code acquisition. A clear understanding of the P-code generator architecture is important for a proper design implementation. In addition, a good design strategy is necessary due to the extremely long P-code period.

In this chapter, the GPS P-code generator tuning model is developed to facilitate the generation of the P-code from any specific time of a week in a FPGA chip using hardware description language (HDL). The model is used to determine the initial vector for each linear feedback shift register (LFSR), the initial value of the z-counter, and the initial values for different division circuits in the P-code generator.

The P-code sequences listed in ICD-GPS-200 are considered as truth and reference in this dissertation. The successful verification of each single LFSR design in the P-code generator is deducted. The LFSR halting phenomena at the end of each LFSR short cycle and at the end of a GPS week are verified successfully. All of these results are consistent with those listed in ICD-GPS-200.

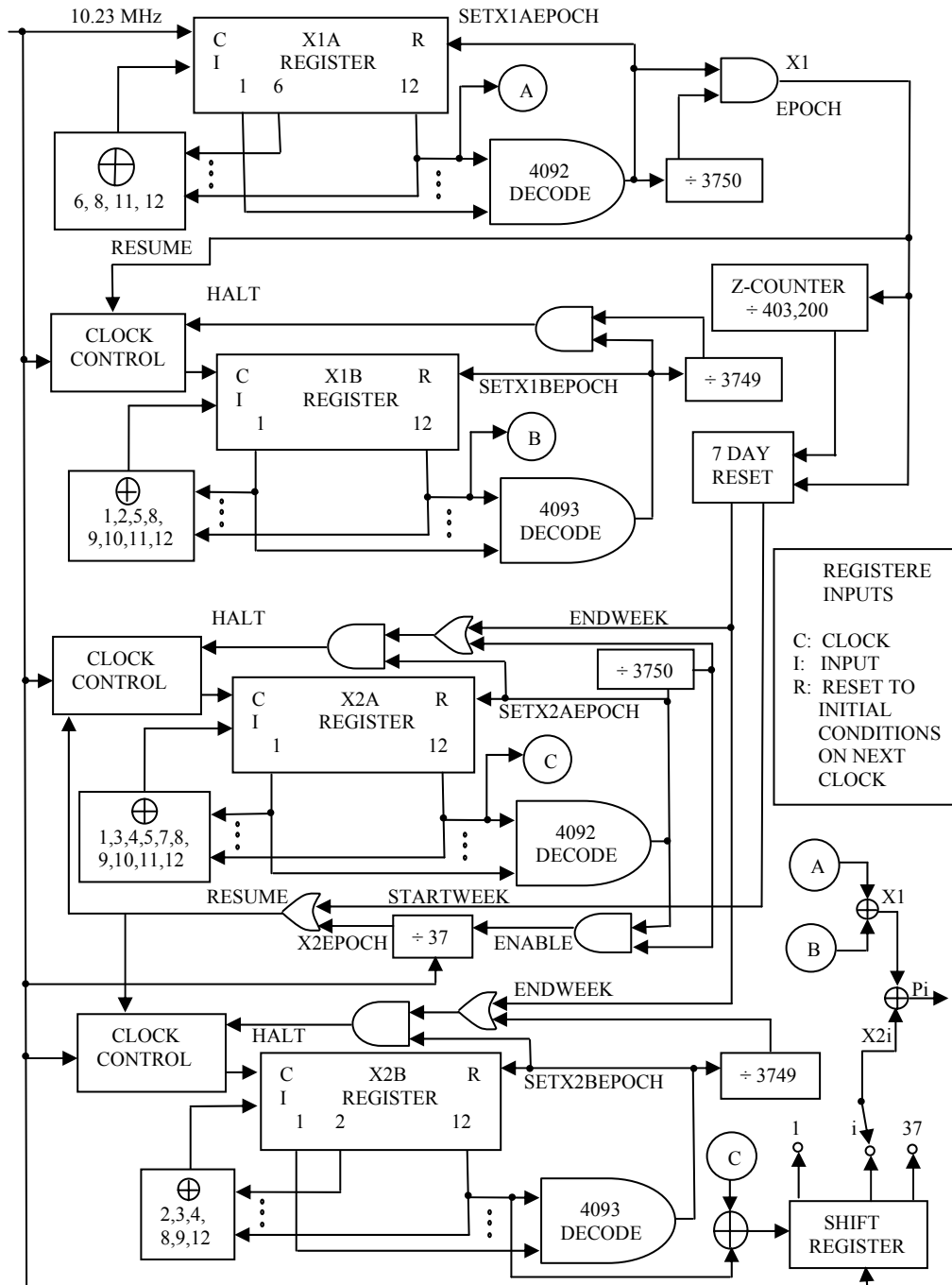


Figure 4.1 GPS P-Code Signal Generator (ICD-GPS-200, 1991)

## 4.1 P-code Generator Architecture

In the P-code generator, the main clock frequency is 10.23 MHz. As shown in Figure 4.1, there are four 12-stage linear feedback shift registers (LFSRs), which are X1A, X1B, X2A and X2B. X1A and X2A are each shorted to 4092 chips. X1B and X2B are each shorted to 4093 chips.  $X_1$  is generated by the Modulo-2 sum of the outputs of X1A and X1B.  $X_2$  sequence is produced by the Modulo-2 sum of the outputs of X2A and X2B, and then it is delayed by a selected integer number of chips,  $i$ , ranging from 1 to 37, which results in the  $X_{2i}$  sequences. Each  $P_i(t)$  is the Modulo-2 sum of  $X_1$  and  $X_{2i}$  (ICD-GPS-200, 1991). When the X1B short cycles are counted to 3749, X1B LFSR is halted. After X1A short cycles are counted to 3750, the X1 epoch is generated and it resumes X1B LFSR. Similarly, when the X2B short cycles are counted to 3749, X2B LFSR is halted. The difference is that after X2A short cycles are counted to 3750, X2B LFSR needs additional 37 clock cycles to be resumed. Then X2 epoch is generated, which accumulates 37 clock cycle delays for each epoch compared with X1 Epoch. At the beginning of the GPS weekly period, X1A, X1B, X2A and X2B shift registers are initialized to produce the first chip of the GPS week.

The processing of the shift registers with respect to X1A continues until the last X1A period of the GPS week interval. During this particular X1A period, X1B, X2A and X2B are held when reaching the last state of their respective cycles until X1A cycle is completed. At this point, all four shift registers are initialized and provide the first chip of the new GPS week.

The polynomials for X1A, X1B, X2A and X2B LFSRs can be written as:

$$X1A: 1 + X^6 + X^8 + X^{11} + X^{12} \quad (4.1)$$

$$X1B: 1 + X^1 + X^2 + X^5 + X^8 + X^9 + X^{10} + X^{11} + X^{12} \quad (4.2)$$

$$X2A: 1 + X + X^3 + X^4 + X^5 + X^7 + X^8 + X^9 + X^{10} + X^{11} + X^{12} \quad (4.3)$$

$$X2B: 1 + X^2 + X^3 + X^4 + X^8 + X^9 + X^{12} \quad (4.4)$$

The block diagrams of the four LFSRs are illustrated in Figures 4.2 to 4.5.

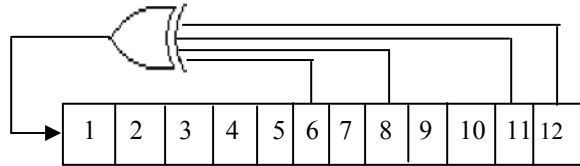


Figure 4.2 X1A LFSR Diagram

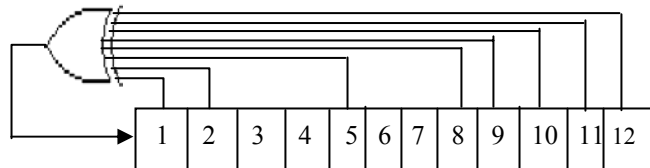


Figure 4.3 X1B LFSR Diagram

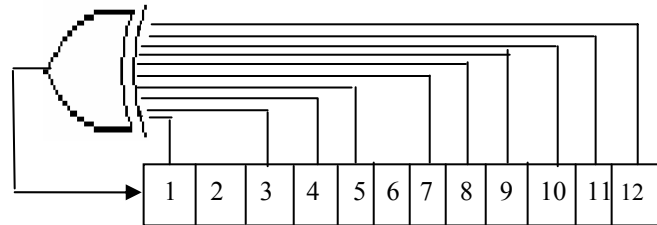


Figure 4.4 X2A LFSR Diagram

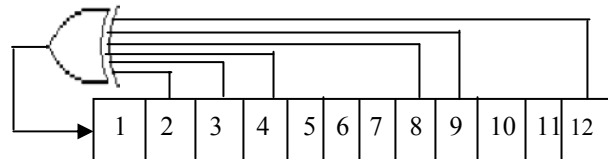


Figure 4.5 X2B LFSR Diagram

## 4.2 GPS P-Code Generator Tuning Model

HDL offers a very short and efficient design cycle involving simulation, synthesis and testing (Chang 1997 and Navambi 1993). They are increasingly replacing schematic oriented design entry methods. From the synthesis perspective, the most appealing benefits of using HDLs are the ability to parameterize modules in which designs can be created in the technology independent manner (Klupsch, 2002). The customizing module parameter method is convenient and powerful for designers to save design effort (Luk,

1998 and Mencer, 2003). This concept is used in this dissertation. The following describes the tuning model development details.

After 4092 chips, X1A finishes a cycle and generates a pulse called  $setx1aepoch$ . X1B needs to produce 4093 chips in a cycle to generate a pulse called  $setx1bepoch$ . After 3749 X1B cycles, X1B is halted and it will be resumed when X1A reaches 3750 clock cycles. At that time, another pulse is generated called  $x1epoch$ . So totally X1B is halted by X1A for  $(4092-3749)=343$  chip in one  $x1epoch$  cycle. X2A has similar relationship corresponding to X2B as that of X1A versus X1B. The difference is that X2A and X2B are halted 37 extra chips in each  $x2epoch$  (Kaplan, 1996). The last week final halting is specified in ICD200 table. At any specific time during a week, the number of chips generated by X1A generator can be easily calculated because the clock frequency is equal to 10.23 MHz. Then the number of chips can be described as Equation 4.5.

$$N = \text{Time [s]} * 10.23 \text{ [MHz]} \quad (4.5)$$

In order to initialize the P-code generator in a specific time during a week, integer  $y1a$  is used to load the z-counter. Integers  $x1a$  and  $x2a$  are used to load the 3750 division counters in X1A and X2A blocks. Integers  $x1b$  and  $x2b$  are used to load the 3749 division counters in X1B and X2B blocks. Integers  $z1a$ ,  $z1b$ ,  $z2a$  and  $z2b$  are used to initialize four linear feedback shift registers. Integer  $dv$  is used to load the delay by 37 chips block. Integers  $y2a$  and  $m$  are only needed to compute register values. In addition, two constants are defined as



$$C1 = 4092 * 3750 \quad (4.6)$$

$$C2 = 4093 * 3749 \quad (4.7)$$

Several equations involving the chip number can be formulated to relate those integers. For instance,  $z1a$ ,  $x1a$  and  $y1a$  can be obtained from

$$N = C1 * y1a + 4092 * x1a + z1a \quad (4.8)$$

where

$$z1a = REM[N \text{ mod } 4092] \quad (4.9)$$

$$x1a = REM[(N - z1a) / 4092] \text{ mod } 3750 \quad (4.10)$$

$$y1a = (N - z1a - 4092 * x1a) / C1 \quad (4.11)$$

Similarly,

$$N = C1 * y1a + 4092 * x1b + z1b \quad (4.12)$$

$$N = (C1 + 37) * y2a + 4092 * x2a + z2a + dv \quad (4.13)$$

$$N = (C1 + 37) * y2a + 4093 * x2b + z2b \quad (4.14)$$

The solutions of equations (4.12), (4.13) and (4.14) are listed in the following equations.

$$\begin{aligned} z1b &= 4092 \text{ when } (N - C1 * y1a) \geq C2, \\ &= REM[(N - C1 * y1a) \text{ mod } 4093], \text{ otherwise} \end{aligned} \quad (4.15)$$

$$\begin{aligned} x1b &= 3748 \text{ when } (N - C1 * y1a) \geq C2, \\ &= (N - z1b - C1 * y1a) / 4093, \text{ otherwise} \end{aligned} \quad (4.16)$$

$$m = REM[N \text{ mod } (C1 + 37)] \quad (4.17)$$

$$y2a = (N - m) / (C1 + 37) \quad (4.18)$$

$$dv = m - C1 \text{ when } m \geq C1 \\ = 0, \text{ otherwise} \quad (4.19)$$

$$z2a = REM[(m - dv) \text{ mod } 4092] \quad (4.20)$$

$$x2b = 3748 \text{ when } m \geq C2 \\ = (m - z2b) / 4093, \text{ otherwise} \quad (4.21)$$

One example is given below using the above tuning model.

Example 1: Suppose the required P-code starting time is Monday 11:30 a.m. of a GPS week.

Time in equation (4.5) is computed with reference to the beginning of every GPS week, which starts on Sunday at 0:00 a.m. The number of chips N can be computed by using 1 day 11 hours and 30 minutes with the clock frequency 10.23 MHz.

$$N = 10.23 * 10^6 * (24 * 3600 + 11 * 3600 + 30 * 60) \\ = 1.289,000,000,000 * 10^{12} \quad (4.22)$$

Then use the GPS P-code generator tuning model to solve x1a, z1a, y1a, x1b, z1b, y2a, z2a, x2b, z2b, and dv. The solutions and the corresponding P-code generator initialization set up are in Table 4.1.

Table 4.1 Solution table for example 1

	<b>LFSR</b>	Division by 3750 or 3749 block	Division by 37 block	z-counter
X1A	4092 <sup>nd</sup> vector (z1a=0)	0 (x1a=0)	0	85200
X1B	4093 <sup>rd</sup> vector (z1b=0)	0 (x1b=0)	(dv=0)	(y1a=85200)
X2A	2569 <sup>th</sup> vector (z2a=2569)	2979 (x2a=2979)		
X2B	3683 <sup>rd</sup> vector (z2b=3683)	2978 (x2b=2978)		

According to ICD-GPS-200, the x1epoch pulse appears every 1.5 s. At the end of every 1.5 s, X1A LFSR generates its last vector in its short cycle and X1B LFSR is held in its last vector. On Monday 11:30 a.m., time is  $(24*3600 + 11*3600 + 30*60) = 127800$  seconds and  $127800/1.5 = 85200$ . This also means 85200 x1epoch pulses are generated.

In the above solutions,  $y1a=85200$ .  $y1a$  is used to set up the z-counter to count the pulses of x1epoch.  $z1a=0$  and  $z1b=0$  mean X1A LFSR and X1B LFSR are in their last states. As a result, X1A LFSR is loaded with its 4092<sup>nd</sup> vector in its short cycle. X1B LFSR is loaded with its 4093<sup>rd</sup> vector correspondingly. In conclusion, the P-code generator set up solutions for the z-counter, X1A LFSR, X1B LFSR, and their corresponding division blocks are correct.

Again, according to ICD-GPS-200, each x2epoch pulse has an extra 37-chip delay relative to each x1epoch pulse. So we know that x2epoch will happen after additional  $85200*37=3152400$  chips are generated. Also both X2A LFSR and X2B LFSR will be

in their last states of their short cycles, which correspond to  $x_{2a}=0$ ,  $z_{2a}=0$ ,  $x_{2b}=0$ , and  $z_{2b}=0$ . This will happen after  $3152400/(10.23 \cdot 10^6)=0.3082$  second. So the resulting time for a new  $x_{2epoch}$  pulse is Monday 11'oclock 30 minutes 0.3082 seconds in the morning. New solutions for the variables in the GPS P-Code generator tuning model can be solved using the new time. The following solutions are obtained:

$x_{1a} = 770$ ,  $z_{1a}=1560$ ,  $y_{1a}=85200$ ,  $x_{1b}=770$ ,  $z_{1b}=790$ ,  $x_{2a}=0$ ,  $z_{2a}=0$ ,  $y_{2a}=85200$ ,  $x_{2b}=0$ , and  $z_{2b}=0$ .

$y_{2a}$  corresponds to the  $x_{2epoch}$  pulse. The result shows exactly 85200  $x_{2epoch}$  pulses are generated. The solutions of  $x_{2a}$ ,  $z_{2a}$ ,  $x_{2b}$ , and  $z_{2b}$  are the same with the expectation. So the solutions are in agreement with the ICD-GPS-200 document.

### 4.3 P-code Generator Design Verification

In order to verify the P-code generator design, apply the input vectors to generate the 1<sup>st</sup> chip, then observe the 4091<sup>st</sup>, 4092<sup>nd</sup> or 4093<sup>rd</sup> output vectors and compare them with those in Table 4.2.

Table 4.2 P-Code Vector States (ICD-GPS-200, 1991)

Code	Chip Number	Vector State (HEX)	Vector State for 1 <sup>st</sup> Chip following Epoch (HEX)
X1A	4091	100010010010 (892)	001001001000 (248)
	4092	000100100100 (124)	
X1B	4092	100101010101 (955)	010101010100 (554)
	4093	001010101010 (2AA)	
X2A	4091	111001001001 (E49)	100100100101 (925)
	4092	110010010010 (C92)	
X2B	4092	000101010101 (155)	010101010100 (554)
	4093	001010101010 (2AA)	

The verification of each single LFSR design satisfies the above table. Other verification concerns are the halting and the resuming of X1B, X2A and X2B at each epoch and also the chip generation at the end of a GPS week. They are verified in the following two cases.

Case 1: Verification of code halting after one cycle of X1A, X1B, X2A and X2B.

Assume  $y1a=0$ ,  $y2a=0$ ,  $x1b = 3748$  and  $z1b = 4092$ .

Use the algorithm described before to solve tuning models. The solutions are  $x1a=3749$ ,  $z1a=3748$ ,  $x2a=3749$ ,  $z2a=3748$ ,  $x2b=3748$ , and  $z2b=4092$ . Use these

parameters to setup the P-code generator as that shown in Table 4.3.

After one cycle of X1A, X1B, X2A and X2B, the simulation should show that X2AQ is halted for 37 chips, X1BQ is halted for 343 chips, and X2BQ is halted for 380 chips. X1AQ, X1BQ, X2AQ, and X2BQ are the 12<sup>th</sup> bit of the output vectors of X1A, X1B, X2A, and X2B LFSR.

Table 4.3 Case 1 P-code generator initialization table

	<b>LFSR</b>	Division by 3750 or 3749 block	Division by 37 block	z-counter
X1A	3748 <sup>th</sup> vector	3749	0	0
X1B	4092 <sup>nd</sup> vector	3748		
X2A	3748 <sup>th</sup> vector	3749		
X2B	4092 <sup>nd</sup> vector	3748		

Case 2: Verification of code halting at the end of the GPS week.

At the end of the GPS week, assume  $y1a=403199$ ,  $x1a=3748$ , and  $z1a=3022$ .

Solve tuning models to get  $x1b=3747$ ,  $z1b=3367$ ,  $x2a=102$ ,  $z2a=4091$ ,  $x2b=102$ ,  $dv=0$  and  $z2b=3989$ . Use these parameters to setup the P-code generator as those in Table 4.4.

Table 4.4 Case 2 P-code generator initialization Table

	<b>LFSR</b>	Division by 3750 or 3749 block	Division by 37 block	z-counter
X1A	3022 <sup>nd</sup> vector	3748	0	403199
X1B	3367 <sup>th</sup> vector	3747		
X2A	4091 <sup>st</sup> vector	102		
X2B	3989 <sup>th</sup> vector	102		

The waveforms in Figure 4.6 show the simulation results. 10 ns clock period is used for the functional verification. Signals X1ADIVINIT, X1BDIVINIT, X2ADIVINIT, and X2BDIVINIT are initialized with values 3748, 3747, 102, and 102 respectively for each division by 3750 or 3749 block in four LFSRs. z-counter is initialized with signal ZNTINIT equal to 403198 because it starts counting from zero. Signals ENDWEEK and STWEEK represent the end and the start of the GPS week. According to the timing shown in Figure 4.6, the ENDWEEK pulse occupies the time interval between two SETX1AEPOCH pulses at the end of the GPS week. The STWEEK pulse occupies the time interval between two SETX1AEPOCH pulses at the start of the new GPS week. Signals X1AQ, X1BQ, X2AQ and X2BQ correspond to the outputs of four LFSRs.

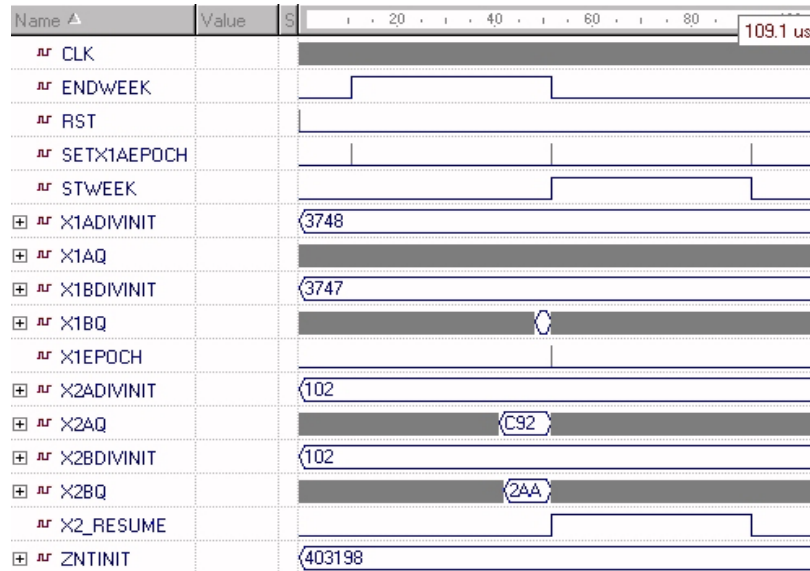


Figure 4.6 Case 2 simulation waveform

Figures 4.7, 4.8 and 4.9 show the timing of signals X2AQ, X1BQ, and X2BQ at the end of the GPS week. At 40.925 us, the X2A LFSR generates its 4092<sup>nd</sup> chip, and then it is halted. At 48.185 us, the X1B LFSR generates its 4093<sup>rd</sup> chip, and then it is halted. Similarly, at 51.625 us, the X2B LFSR generates its 4092<sup>nd</sup> chip, and then it is halted. As a result, at the end of the GPS week, the signal X1BQ holds a heximal value 2AA, the signal X2AQ holds a heximal value C92, and the signal X2BQ holds a heximal value 2AA. These values agree with those values listed in Table 4.1.



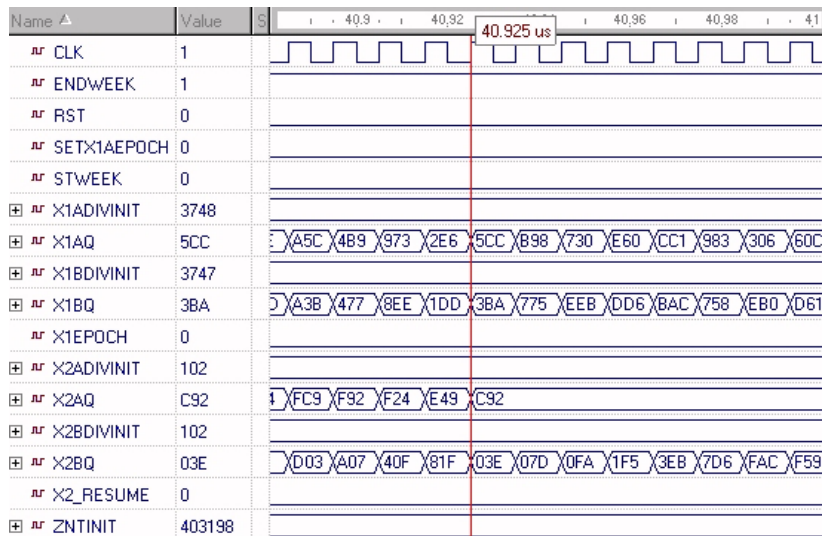


Figure 4.7 The final X2AQ chip generation at the end of a GPS week

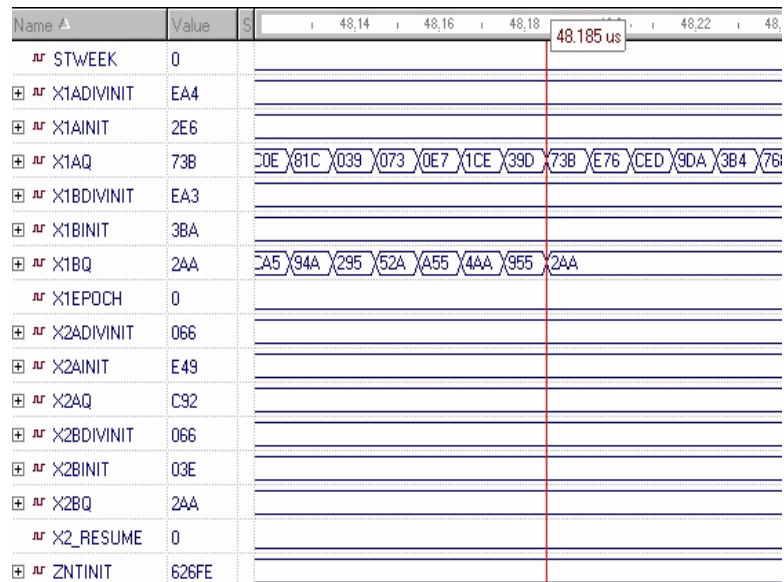


Figure 4.8 The final X1BQ chip generation at the end of a GPS week

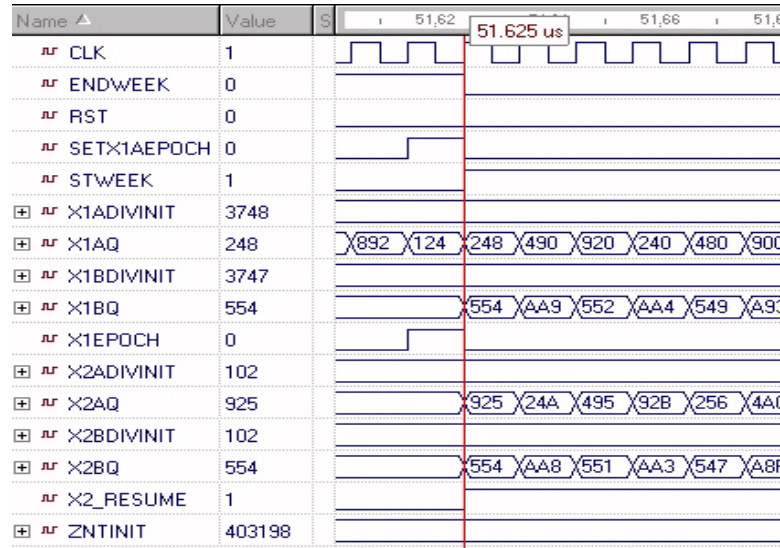



Figure 4.9 Resume of X1AQ, X1BQ, X2AQ and X2BQ at the start of a GPS week

From the above simulations, at the end of the GPS week, X1BQ is halted for  $(51625-48185)/10 - 1 = 343$  chips; X2AQ is halted for  $(51625 - 40925)/10-1 = 1069$  chips, and X2BQ is halted for  $(51625-41965)/10-1=965$  chips.

Table 4.5 lists the timing sequence generated at the end of the GPS week according to ICD-GPS-200. It shows that when the X1A LFSR generates the 3022<sup>nd</sup> chip, the X1B LFSR, the X2A LFSR and the X2B LFSR generates the 3367<sup>th</sup>, the 4091<sup>st</sup> and the 3989<sup>th</sup> chip respectively. Consequently, the signal X1BQ is halted for  $(4092-3749) = 343$  chips, the signal X2AQ is halted for  $(4092-3023) = 1069$  chips, and the signal X2BQ is halted for  $(4092-3127) = 965$  chips. In conclusion, the simulation results are the same with those expected from Table 4.5.

Table 4.5 P-code Reset Timing at the end of a GPS week (ICD-GPS-200, 1991)

	Code Chip			
	X1A-Code	X1B-Code	X2A-Code	X2B-Code
Time 	1	345	1070	967
	...	...	...	...
	3023	3367	4092	3989
	...	...	...	...
	3127	3471	4092	4093
	...	...	...	...
	3749	4093	4092	4093
	...	...	...	...
	4092	4093	4092	4093

## Chapter 5

# Direct GPS P-code Acquisition Design

## Implementation

### 5.1 Introduction

The architecture of a direct GPS P-code acquisition processor is presented in this chapter. The operation of each sub block is explained. Important diagrams are illustrated with some critical signals listed. The assumption made here is that the GPS signals have been demodulated first.

Great efforts have been spent on the modeling of the direct GPS P-code acquisition algorithm, and on the correct use of the Nallatech platform (Nallatech, 2002) and the Xilinx 1024-point FFT/IFFT core (Xilinx, 2000). A stable FFT/IFFT core controller has been written after numerous experiments and has been tested one-million times with stable output results. The Xilinx 1024-point transform engine employs a Cooley-Tukey radix-4 decimation-in-frequency (DIF) FFT (Cooley, 1965) to compute the DFT of a complex sequence. It uses the successive dragonfly with the proper scaling to accommodate the dynamic range expansion in each dragonfly pass. A full explanation

of scaling strategies and their implications can be found in literature (Knight, 1979 and Rabiner, 1975).

In addition, different features of Xilinx VirtexE FPGA are studied thoroughly (Xilinx, 2002). In the FPGA design, the direct average method is used. Only one block of the local P-code reference is averaged and processed for the code phase search in each loop.

## 5.2 Architecture of Direct GPS P-code Acquisition

### Processor

Some preprocessing procedures are done by the PC first. Averaging and FFT are taken on 2 ms demodulated GPS signals in the PC. These preprocessed data sequences are then sent to an FPGA board. The local generated P-code reference sequences are the averaged binary P-code. The FFT is taken on the averaged local reference block data. Then the GPS signal FFT results are multiplied by the complex conjugate of the local reference FFT results. This is called the complex multiplication in this dissertation. Next, the IFFT is taken to get the complex correlation results. Then the square of the correlation amplitude is calculated. The peak of the correlation amplitude square and the peak location are searched in each block. This procedure continues until 10-ms data sequences have been processed. The maximum and the second largest peak and peak location

values are the outputs of the whole processing hardware design shown in Figure 5.1.

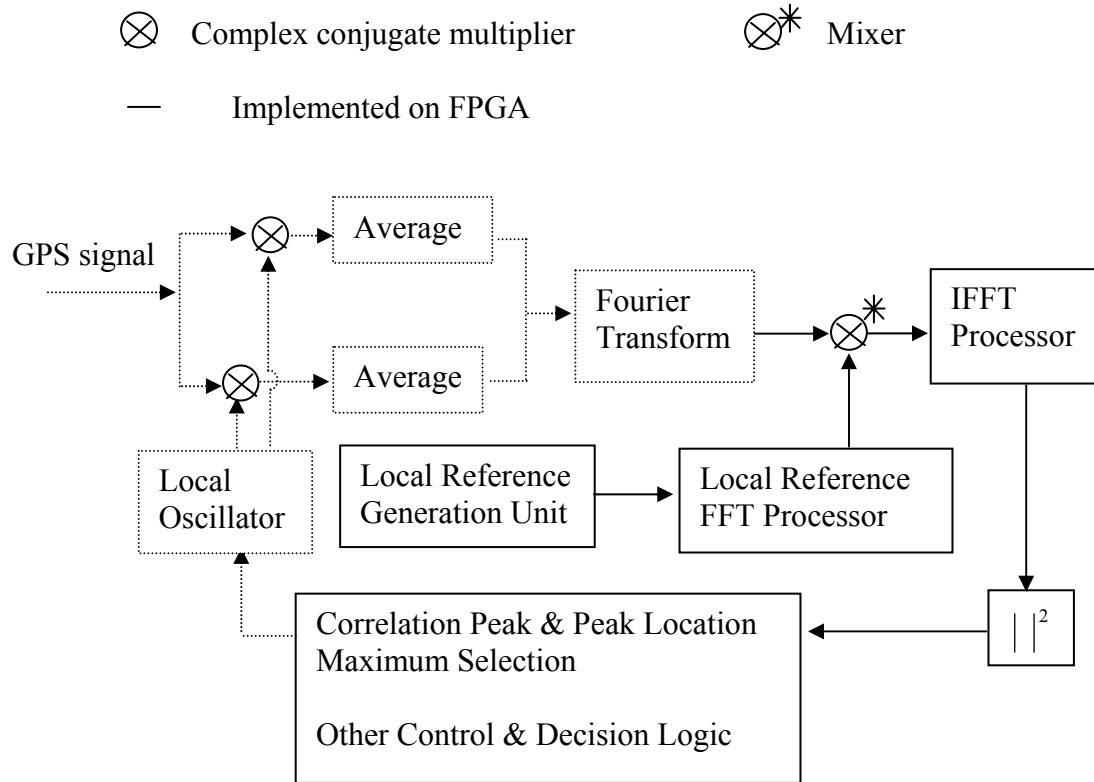


Figure 5.1 Direct GPS P-code Acquisition Processor

In Figure 5.1, the GPS P-code acquisition processor architecture consists of a local reference generation unit, a local reference FFT processor, a complex conjugate multiplication processor, an IFFT processor, a correlation amplitude square unit, a correlation peak unit, a correlation peak location unit, and a maximum selection unit.

### 5.3 Local Reference Generation Unit

The local reference generation unit consists of a P-code generator, a P-code binary converter, an AVERAGE unit, a state machine PCODE\_MACH, and a RAM address generator.

First, a RAM is preloaded with zeros and then its address is initialized to zero. After FFT results of the GPS signal are sent from the PC to the FPGA, the P-code generator is started to generate unary P-code in “0” and “1” format in each clock cycle. Then the P-code is converted to “-1” and “1” format by the binary converter. Next, every 20 chips of P-code are averaged into 1 chip by the AVERAGE unit. The averaged result is saved in a RAM. After 1 ms P-code generation, 10220 chips are averaged by 20 and saved in the RAM. The last 10 chips at the end of every 1 ms are averaged by 10 and saved in the RAM. As a result, 10230 chips in 1 ms are averaged into 512 points. The other 512 points are zeros which are preloaded into the RAM. As a result, 1024 averaged points are prepared for further FFT processing. At last, the RAM address is set to zero again. This process is repeated. The controller is the state machine PCODE\_MACH.

In Figure 5.2, the signal rst goes low when the FPGA chip is reset. All RAMs and Registers are cleared with zeros. The common clock and rst signals are used for different blocks in the Local Reference Generation Unit to synchronize the clock control and the reset operation. After the results of the PC preprocessed GPS signals are completely

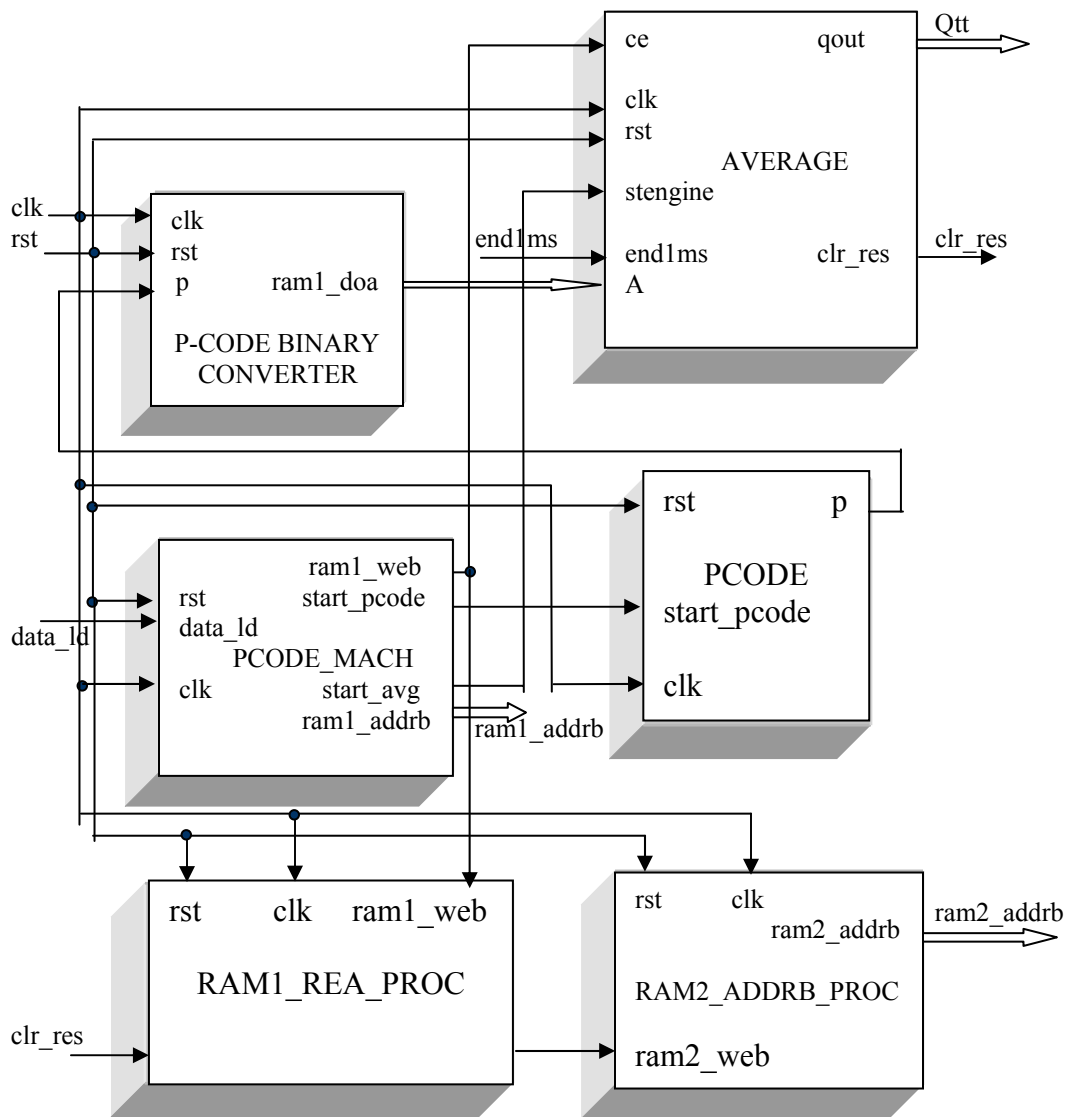


Figure 5.2 Local Reference generation unit diagram



sent into one RAM on the FPGA board, the signal `data_ld` is logically low. This triggers a state machine `PCODE_MACH` to generate a logic high control signal `start_pcode`. Then the P-code generation is started by a component `PCODE`. The P-code signal `p` is converted into a signal `ram1_doa` with -1 and 1 format by a component `P-CODE BINARY CONVERTER`. In addition, the `start_avg` pulse is also generated by the state machine `PCODE_MACH`, which is used to activate the `AVERAGE` block. The signal `ram1_doa` is averaged and scaled up by 2048 in the `AVERAGE` block. The resulting signal is `Qtt`. Scaling is necessary for the FFT operation, which will be explained in the later section. The signal `end1ms` pulse is produced after every 10230 chips of P-code have been generated, and then the `AVERAGE` block is cyclically cleared after every 512 averaging points are produced. A twenty-clock cycle waveform signal `clr_res` is used in the averaging process. Furthermore, the RAM address signal `ram2_addrb` and RAM write enable signal `ram2_web` are controlled by the controller `PCODE_MACH` in order to load the averaged results into a RAM for FFT processing.

## 5.4 Local Reference FFT Processor

The local reference FFT processor in Figure 5.3 consists of a RAM saving the sequentially averaged P-code signal  $Q_{tt}$  value, a RAM loading the FFT intermediate results, a 1024-point FFT/IFFT core doing the forward FFT, a loop counter for debugging FFT results use, and the whole process controller is implemented by a state machine `FFT_MACH`.

When the signal `end1ms` is logically high, the averaged and scaled signal  $Q_{tt}$  has been loaded into a RAM called `RAM2_COMP`. At this point, a controller `FFT_MACH` sets the clock enable signal `ce` to be logically high. Next, it sets the data write strobe signal `mwr` high for one clock cycle. A 1024-point FFT/IFFT core is in the forward FFT mode when the signal `fwd_inv` is equal to logic '1'. Signals `io_mode0` and `io_mode1` are set to '0' and '1' respectively in this dissertation. This indicates single memory configuration mode is used, which means one memory block `RAM_FFT_COMP` is necessary for loading input data into the FFT core sequentially and also for saving the intermediate results during the FFT computation process. The final FFT results can be read out from the FFT core sequentially. The FFT core operation timing sequence is specified by Xilinx company. The signal `addr_x` and the signal `io` are generated by the FFT core to control the RAM port B address signal `addrb` and the RAM port B write enable signal `web` of `RAM_FFT_COMP`. The data output from `RAM2_COMP` is

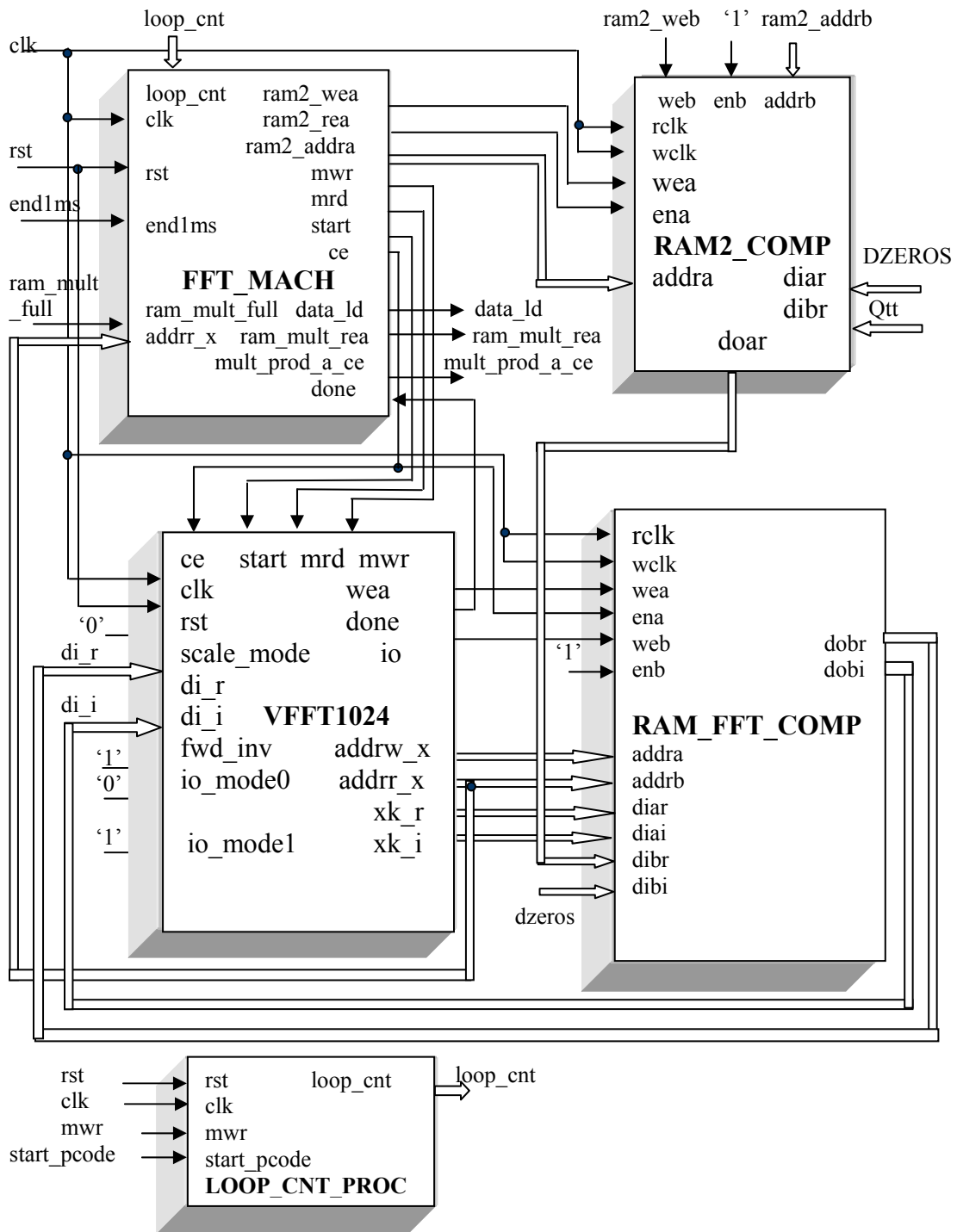


Figure 5.3 Local reference FFT processor diagram

connected with the port B data bus of RAM\_FFT\_COMP. 1024 clock cycles are needed to load the complex data sequences into RAM\_FFT\_COMP completely. The port B complex data outputs of RAM\_FFT\_COMP are connected with the FFT core data bus signals di\_r and di\_i. After the data loading process is finished, the FFT processing strobe signal start is set by the controller FFT\_MACH to be logic high for one clock cycle. Then the FFT computation is started. When FFT computation process is completed, the FFT results are in the interleaving order. At the same time, the signal done is set by the FFT core to be logic high for one clock cycle, which triggers the controller FFT\_MACH to set FFT result read strobe signal mrd logically high for one clock cycle. Consequently, after another 1024 clock cycles, all 1024 complex FFT results are available on the data bus signal di\_r and di\_i sequentially in the correct order. The signal loop\_cnt is used to count how many FFT loops have been completed.

Since the next step after FFT operation is the complex conjugate multiplication, three additional control signals ram\_mult\_full, mult\_prod\_a\_ce and ram\_mult\_rea are generated by the controller FFT\_MACH. When the PC preprocessing GPS results are completely loaded into a RAM on the FPGA board, the signal ram\_mult\_full is logic high. These results are used in the complex conjugate multiplication processor. Signals mult\_prod\_a\_ce and ram\_mult\_rea are the clock enable signal and the read enable signal for the RAM used on complex multiplication respectively.

### 5.4.1 Xilinx 1024-Point FFT/IFFT Core

When migrating between technologies, the intellectual property (IP) technology can be more easily used and reused among designers to help reduce critical development time in the production cycle. In this dissertation, a Xilinx 1024-point FFT/IFFT core is used.

The Xilinx FFT/IFFT Core computes a 1024-point complex forward FFT or inverse FFT (IFFT). The input data sequences are 1024 complex values represented as 16-bit 2's complement numbers – 16 bits for each of the real and imaginary component of a data sample. The 1024 output elements include 16-bit real and 16-bit imaginary parts. The final output sequences are scaled by a factor 1/1024 when a control signal SCALE\_MODE is equal to '0' and 1/2048 when SCALE\_MODE is equal to '1'.

The general DFT is defined by Equation 5.1.

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-jnk2\pi/N}, k = 0, \dots, N-1 \quad (5.1)$$

where  $N$  is the transform size.

The single-memory-space configuration provides the simplest memory and I/O interface to the FFT/IFFT core. A diagram of this configuration is shown in Figure 5.4.

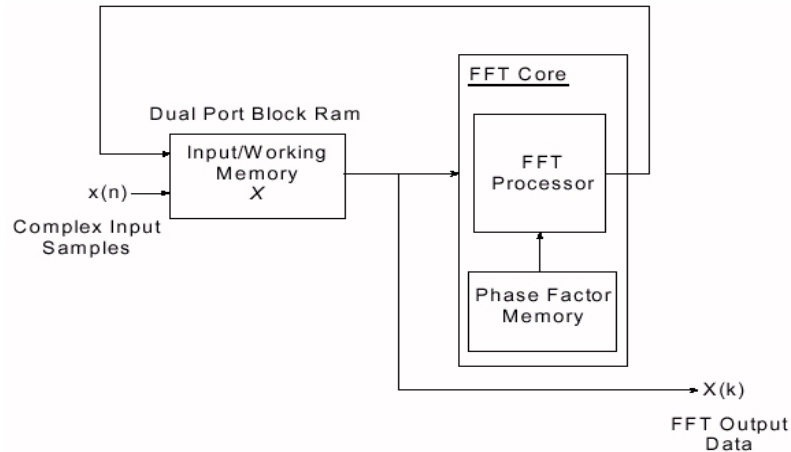


Figure 5.4 Xilinx 1024-point FFT/IFFT core single memory space configuration (Xilinx, 2000)

In Figure 5.4, the Xilinx 1024-point FFT/IFFT core needs three-stage processing: the input vector loading, the FFT computation, and the result unloading. The input loading requires 1024 clock cycles. After that, the signal start needs to be asserted in order to start the FFT computation. The first FFT output sample is written to memory 3121 clock cycles after the signal start is asserted. The final sample is written to memory 4145 clock cycles following the start pulse. The result unload operation also requires 1024 clock cycles.

## 5.4.2 Controller FFT\_MACH

FFT\_MACH is a finite state machine, which has 9 states: ft\_pc\_ld, ft\_rst, ft\_init, ft\_load, ft\_wadr, ft\_calc, ft\_wait1, ft\_wait2, ft\_rdffft. Three additional states can be added for debugging after the state ft\_rdffft: ft\_dbg, ft\_rd, and ft\_stop. The functions of all states are described in detail next.

- 1). State ft\_pc\_ld: load the preprocessed GPS signal from PC to the FPGA board.

A design bit file is downloaded on the FPGA board. The reset signal rst has a short pulse, which is used to set all signals to their initial values in this dissertation, and then trigger the state machine to enter the first state ft\_pc\_ld. The state machine stays at this state until a signal RAM\_MULT\_FULL triggers it to the next state ft\_rst. The signal RAM\_MULT\_FULL is set to logic high when all of the preprocessed GPS signals are loaded into a RAM.

- 2). State ft\_rst: implement zero padding.

The zero padding technique is needed in the P-code correlation process as described earlier. It is implemented by feeding a RAM (RAM2 in this dissertation) with 1024 zeros. After RAM2 is filled with all zeros, the state machine comes to the state ft\_init.

- 3). State ft\_init: wait until 512 points of the averaged P-code are available.

Twenty clock cycles are needed to obtain one averaged point. 512 averaged points

are produced and saved in RAM2. The other 512 points are zeros saved in RAM2 in advance. After 512 points are available, the signal `end1ms` will be logically high to activate the FFT/IFFT core clock enable signal `ce`. In addition, the signal `mwr` is asserted. Then the state machine is transferred to the state `ft_load`.

4). State `ft_load`: read data out of RAM2 port A into FFT/IFFT core continuously.

The port A of RAM2 is set to the read mode. The RAM2 port A address is continuously increased. After all 1024 points are read out of the RAM2 port A, the RAM2 port A is disabled from the reading mode. The state machine is then transferred to the state `ft_wadr`.

5). State `ft_wadr`: assert signal `start` after the FFT loading process is done.

When all 1024 points are loaded into the FFT/IFFT core, the FFT/IFFT core Block RAM read address bus `addr_r_x` is equal to a heximal value `3ff`. The signal `start` is asserted by the state machine. Next, the state machine is transferred to the state `ft_calc`.

6). State `ft_calc`: calculate FFT.

The state machine is kept in this state until the FFT calculation is finished. When the signal `done` is logically high, the state machine will be transferred to the state `ft_wait1`.

7). State `ft_wait1`: assert signal `mrd` to start reading FFT/IFFT results.

The signal `mrd` is asserted for one clock cycle, and then the state machine is transferred to the next state `ft_wait2`.

8). State `ft_wait2`: assert signal `RAM_MULT_REA` in order to start reading ram



RAM\_MULT.

RAM\_MULT saves the FFT results of. The averaged demodulated GPS signals are saved in RAM\_MULT. The signal RAM\_MULT\_REA is set to high so that RAM\_MULT port A is in the reading mode. The data saved in address zero is immediately available. Since the FFT/IFFT core results are read out one clock cycle after signal mrd is asserted, two reading process are synchronized in this way. This state lasts only for one clock cycle. The next state is the state ft\_rdfft.

9). State ft\_rdfft: wait until FFT/IFFT core results reading is complete.

The signal MULT\_PROD\_A\_CE is set to logic high to enable the complex multiplier. When FFT/IFFT core reading is completed, the signal addr\_x is equal to a heximal value 3ff. Then RAM\_MULT reading mode is disabled and the state machine is transferred to the 3<sup>rd</sup> state ft\_init to start a new loop.

### 5.4.3 Loop Counter

The loop counter is activated after the signal start\_pcode is logic high. Since the signal mwr is only valid once for one clock cycle in a whole FFT process, it can be used to count how many FFT processes have been implemented. The counter value is saved in signal loop\_cnt. This signal is useful when designers want to stop the FFT state machine after some FFT processes have been done.

#### 5.4.4 RAM2 and RAM\_FFT

RAM2 is used to provide the input data for the FFT/IFFT core. RAM2 contains 512 averaged P-code points and the other 512 points are zeros. Zeros are loaded first into the port A of RAM2 controlled by the state machine FFT\_MACH. Later, averaged P-code signal  $Q_{tt}$  values are written into the port B of RAM2. At last, all results are read out from the port A of RAM2 and loaded into the FFT/IFFT core.

RAM\_FFT is used to save the intermediate results from the FFT/IFFT core. So most of its input ports are controlled by the FFT/IFFT core output signals. The final FFT results are available from the port B of RAM\_FFT. The real part is signal  $di\_r$ , and the imaginary part is signal  $di\_i$ .

#### 5.4.5 Debug Setup

For debugging purpose, when FFT results are read out of the FFT/IFFT core, three additional states can be added. Also the state  $ft\_rdfft$  has to be modified accordingly.

- 9). State  $ft\_rdfft$ : wait until FFT/IFFT core results reading is completed.

When the FFT/IFFT core reading is completed, the signal `addr_x` is equal to a heximal value 3ff. Then, the `RAM_MULT` is disabled from the reading mode. The state machine is transferred to the next state `ft_dbg`.

10). State `ft_dbg`: make decision on whether to enter debugging mode or not

The signal `loop_cnt` is used to count how many FFTs have been processed. If the signal `loop_cnt` is equal to a preset value, then the state machine will be transferred to the next state `ft_rd` and the RAM3 port A will be set to the read enable mode. Otherwise, it will be transferred to the state `ft_init`.

11). State `ft_rd`: read out values saved in RAM3.

The signal `RAM4_WEB` is asserted so that the RAM4 port B is set to the write mode. Data streams are read out of the port A of RAM3 into the port B of RAM4. When reading is completed, the state machine is transferred to the last state `ft_stop`.

12). State `ft_stop`: stop the state machine.

All necessary signals are set to their initial state. The debug values are saved in RAM4.

Figure 5.5 is a block diagram of two RAMs used for debugging.

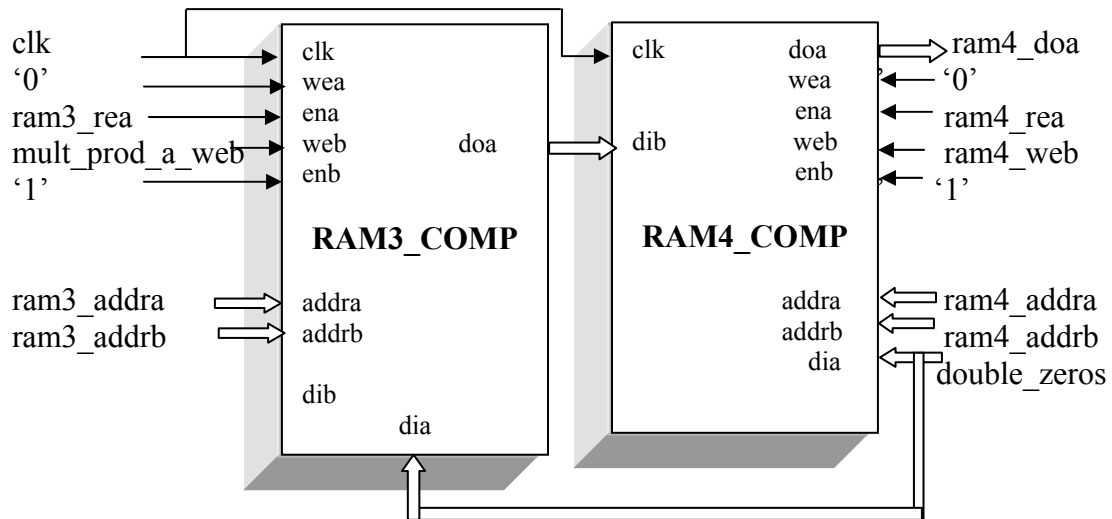


Figure 5.5 Block RAMs for debugging use

## 5.5 Complex Conjugate Multiplication Processor

The complex conjugate multiplication processor consists of a ram called RAM\_MULT, a RAM\_MULT address generators, several registers, three 16-bit by 16-bit multipliers, a summer and a subtractor.

The general complex conjugate multiplication of two complex numbers  $(ar + j \cdot ai)$  and  $(dr + j \cdot di)$  is done in the following equation:

$$(ar + j \cdot ai) \cdot (dr + j \cdot di)^* = ar \cdot dr + ai \cdot di + j \cdot (ai \cdot dr - ar \cdot di) \quad (5.2)$$

Here, four multiplications, one subtraction and two additions are needed to implement one normal complex conjugate multiplication.

The complex conjugate multiplication can also be modified in another way:

$$A0 = (ar + ai) \cdot dr \quad (5.2)$$

$$A1 = (dr - di) \cdot ai \quad (5.3)$$

$$A2 = (ai - ar) \cdot di \quad (5.4)$$

$$A0 - A1 = dr \cdot ar + di \cdot ai \quad (5.5)$$

$$A1 + A2 = dr \cdot ai - di \cdot ar \quad (5.6)$$

In the above calculations, three multiplications, two additions, and three subtractions are needed. Since multiplications are expensive for FPGA design, which use much more hardware resources than additions and subtractions, the modified complex conjugate multiplication is better than the normal one.

In Figure 5.6, the signal `write_en` is first set by the PC to be logically high to enable the loading of the input data `D` into `RAM_MULT` port B. These are 1024 FFT results for the demodulated and the averaged GPS signals. When all of them are written into `RAM_MULT`, the signal `ram_mult_full` is switched low to trigger the state machine `FFT_MACH` to go to the next state. After FPGA FFT/IFFT core finishes the FFT calculation, the FFT results real and imaginary parts are available as the signal `di_r` and the signal `di_i`. In addition, the signal `RAM_MULT_REA` is set by the state machine `FFT_MACH` to be logic high, and data values saved in `RAM_MULT` are read out from the port A of `RAM_MULT`.

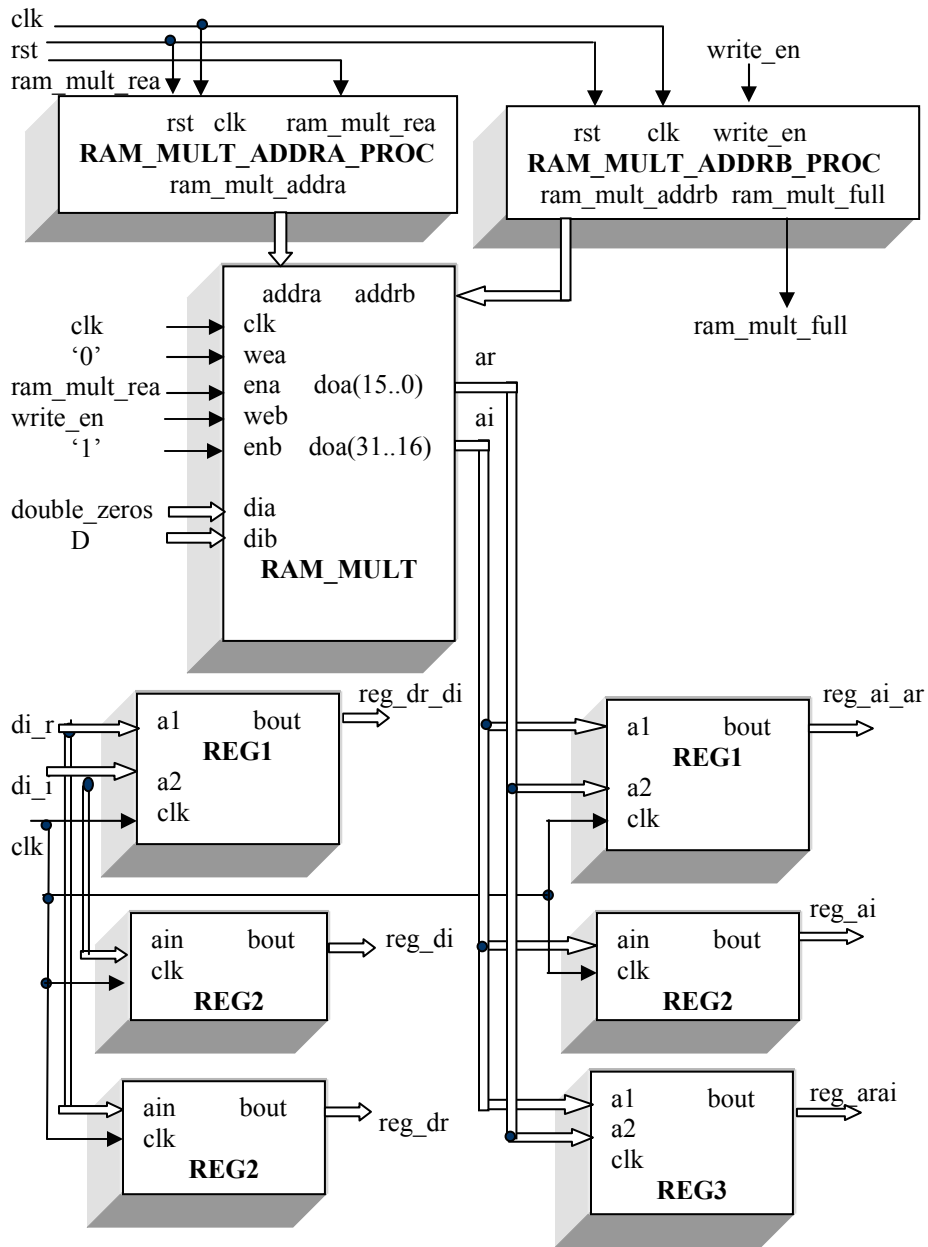


Figure 5.6 Complex conjugate multiplication processor (part 1)

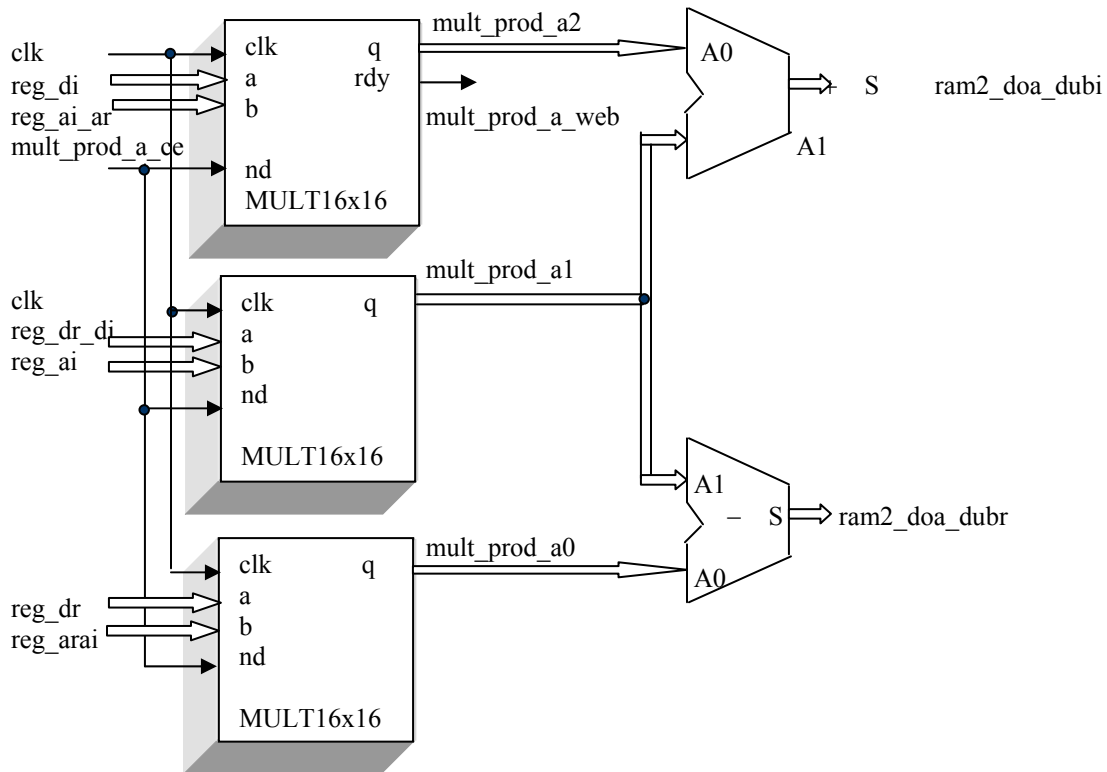


Figure 5.7 Complex conjugate multiplication processor (part 2)

In Figure 5.7, the signal `mult_prod_a_ce` is controlled by the state machine `FFT_MACH`. It is high only after the FFT process is finished and the final FFT results are available. Next, the complex multiplication is implemented according to equations 5.2 to 5.6. When the multiplication results are available, the signal `mult_prod_a_web` is high. The real and imaginary parts of the complex conjugate multiplication results are the signal `ram2_doa_dubr` and the signal `ram2_doa_dubi` respectively.

A brief description of the important signals and ports used in the complex conjugate multiplication processor is given below.

write_en:	write enable
ram_mult_rea:	RAM_MULT read enable
D:	data written into RAM_MULT port B
ram_mult_full:	low when RAM_MULT is full
di_r:	FFT/IFFT core final output data real component
di_i:	FFT/IFFT core final output data imaginary component
reg_dr_di:	registers value (dr-di) at the rising edge of clock
reg_di:	registered value di at the rising edge of clock
reg_dr:	registered value dr at the rising edge of clock
reg_ai_ar:	registered value (ai-ar) at the rising edge of clock
reg_ai:	registered value ai at the rising edge of clock
reg_arai:	registered value (ar+ai) at the rising edge of clock
mult_prod_a_ce:	high when multiplicands are available
mult_prod_a_web:	high when multiplication result is available
ram2_doa_dubi:	complex conjugate multiplication result imaginary part
ram2_doa_dubr:	complex conjugate multiplication result real part



## 5.6 IFFT Processor

The IFFT processor consists of a 1024-point FFT/IFFT core to do the IFFT, a loop counter for debugging the IFFT results, a RAM called RAM\_IFFT saving IFFT intermediate results and the whole process is controlled by a state machine IFFT\_MACH. The core also uses the single memory configuration mode. Since the FFT/IFFT core has the same interface timing control except signal fwd\_inv is switched to logic '0' value, the whole IFFT processing architecture is similar to that of the FFT process. Here, the controller IFFT\_MACH outputs more control signals than the controller used in the FFT processor, such as signals max\_en, cnt\_eni, and mult\_prod\_a\_ce in Figure 5.8. This is because IFFT\_MACH also needs to control the complex conjugate multiplication and the correlation peak search which are implemented after the IFFT process.

A brief description of the important signals and ports used by the IFFT processor in Figure 5.8 is given below.

max\_en:            when this signal is high, correlation peak searching block will be enabled

io\_mode0, io\_mode1:    when io\_mode0='0' and io\_mode1='1', the type of memory interface is in the single memory space configuration

fwd\_inv:            low when FFT/IFFT core is set to inverse FFT mode

ram2\_doa\_dubi:    complex conjugate multiplication result imaginary part

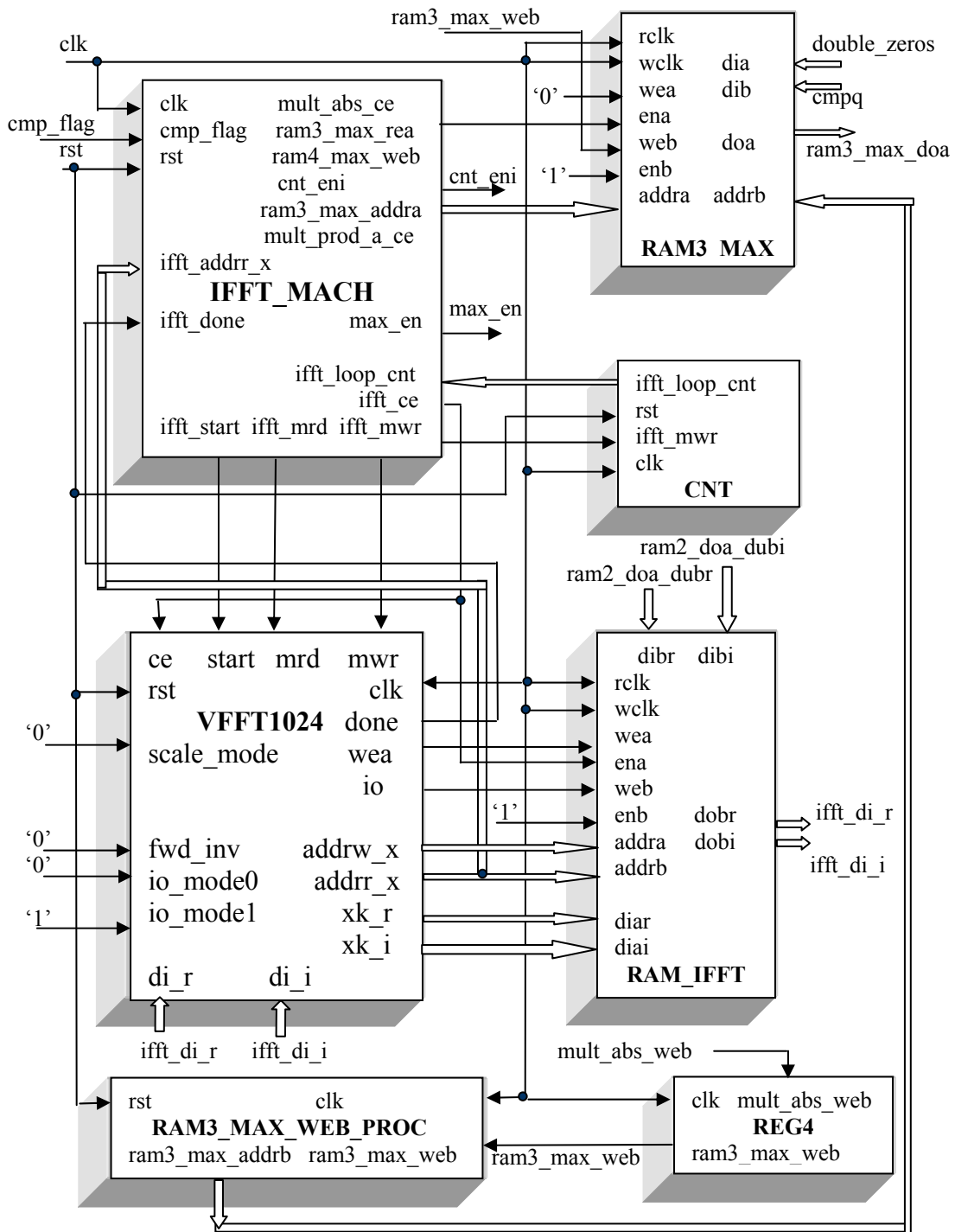


Figure 5.8 IFFT processor diagram

ram2\_doa\_dubr: complex conjugate multiplication result real part  
ifft\_di\_r: IFFT real component of the input data vector  
ifft\_di\_i: IFFT imaginary component of the input data vector  
mult\_abs\_ce: multiplication enable signal  
ram3\_max\_rea: ram3\_max port A read enable  
ram4\_max\_web: ram4\_max port B write enable  
cnt\_eni: clock enable signal used in correlation peak processor  
ram3\_max\_addra: ram3\_max port A address  
mult\_prod\_a\_ce: clock enable for the RAM used on complex multiplication  
cmp\_flag: high when the comparison is finished

Here, RAM3\_MAX and RAM4\_MAX are used for debug. They function similar with those two RAMs used in section 5.4.5.

### 5.6.1 Controller IFFT\_MACH

IFFT\_MACH is a finite state machine, which has 13 states: `ifft_init`, `ifft_load`, `ifft_wadr`, `ifft_calc`, `ifft_wait1`, `ifft_wait2`, `ifft_rdfft`, `ifft_wait3`, `ifft_dbg`, `ifft_cmp`, `ifft_loop_db`, `ifft_rd`, and `ifft_stop`.

Compared with the FFT state machine `FFT_MACH`, `IFFT_MACH` doesn't need to control any data loading from the PC to the FPGA. So it saves the first two states that are in `FFT_MACH`. But `IFFT_MACH` has extra states `ifft_wait3` and `ifft_cmp` to assert signal `max_en`, `cnt_eni`, and `loc_eni`. When signal `max_en` is high, the correlation peak searching process is started. Signals `cnt_eni` and `loc_eni` are used to enable the correlation absolute value square unit, correlation peak comparison process and the peak location searching process.

The flow chart for the controller `IFFT_MACH` is illustrated in the Figure 5.9 and the controller `IFFT_MACH` interface FFT/IFFT core diagram is shown in Figure 5.10.

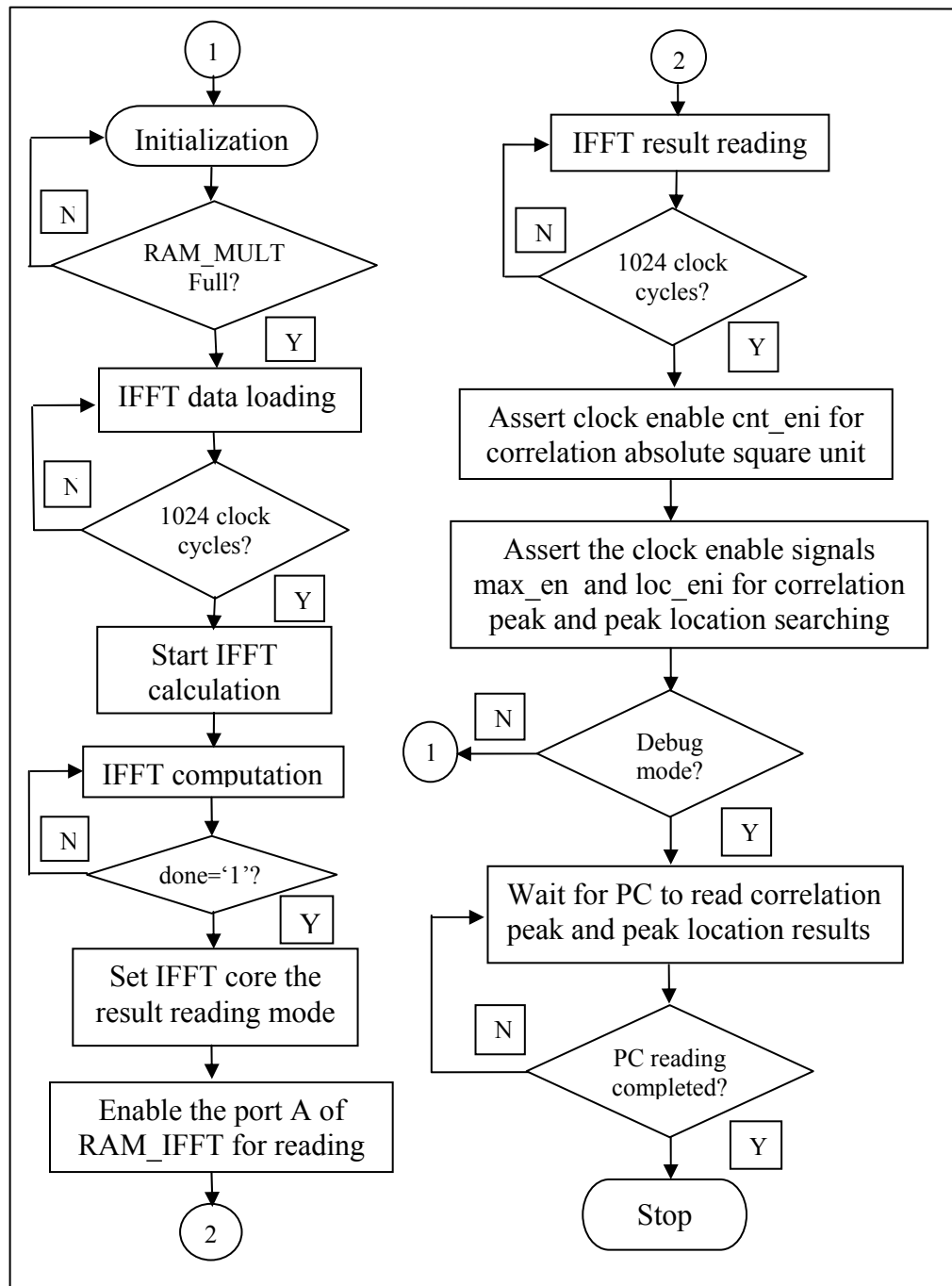


Figure 5.9 The controller IFFT\_MACH flow chart

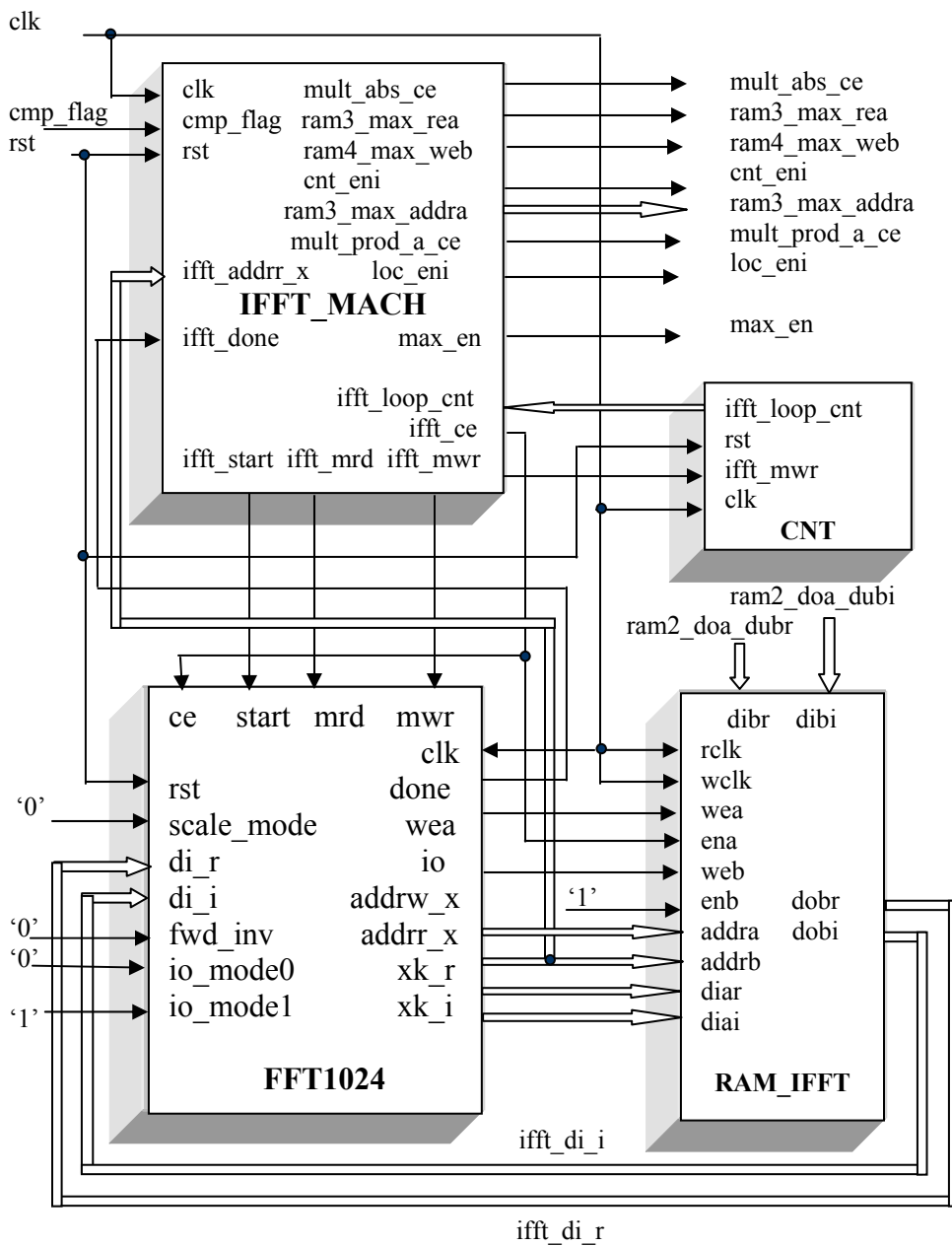


Figure 5.10 Controller IFFT\_MACH interface FFT/IFFT core diagram

## 5.7 Correlation Amplitude Square Unit

The correlation results are complex numbers. Suppose there is a complex number  $a + j \cdot b$ , then

$$|a + j \cdot b|^2 = a^2 + b^2 = a \cdot a + b \cdot b \quad (5.7)$$

The design diagram of this unit is in Figure 5.11.

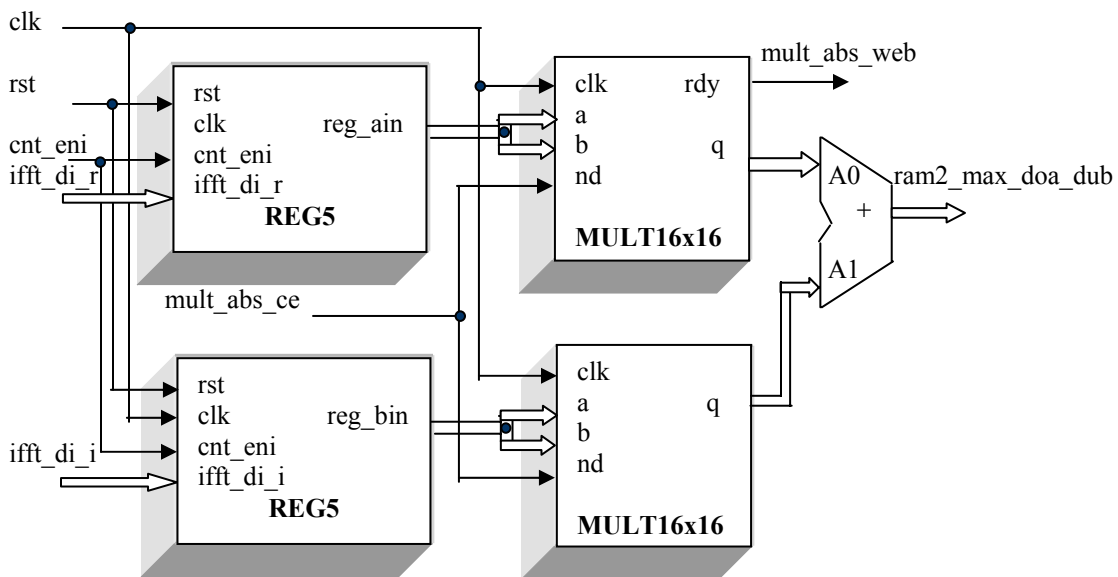


Figure 5.11 Correlation amplitude square unit diagram

In Figure 5.11, signals  $\text{ifft\_di\_r}$  and  $\text{ifft\_di\_i}$  are real and imaginary parts of correlation complex results. The signal  $\text{cnt\_eni}$  is controlled by the IFFT process state machine  $\text{IFFT\_MACH}$ . When the signal  $\text{cnt\_eni}$  is high, the correlation results are

available and they are latched by two registers at the rising edge of the clock. Next multiplications are done by two mult16x16 blocks. Their outputs are added together to generate the square of the correlation amplitude, which is signal ram2\_max\_doa\_dub. When signal mult\_abs\_web is high, it indicates multiplication results are available.

## 5.8 Correlation Peak and Peak Location Processor

The correlation peak and peak location processor consists of a peak-processing unit, a peak location-processing unit, and a maximum selection unit. The recorder unit provides the result of the largest correlation peak, the 2<sup>nd</sup> largest correlation peak and their locations.

In Figure 5.12, after one IFFT loop, correlation amplitude square values obtained as signal ram2\_max\_doa\_dub are sequentially sent to the port A of a comparator CMP32. Since only the first half of the IFFT results are kept for the peak searching processing, only 512 comparisons are made in one loop. The correlation amplitude square peak and peak location results in one loop are in two registers: PEAK\_PROC and PEAK\_LOC\_PROC.

The above procedures continue in loops. There are new peak and peak location results from other new loops. The peak and peak location results from different loops are always sent to the maximum selection unit PEAK\_LOC12 for further comparison. As a



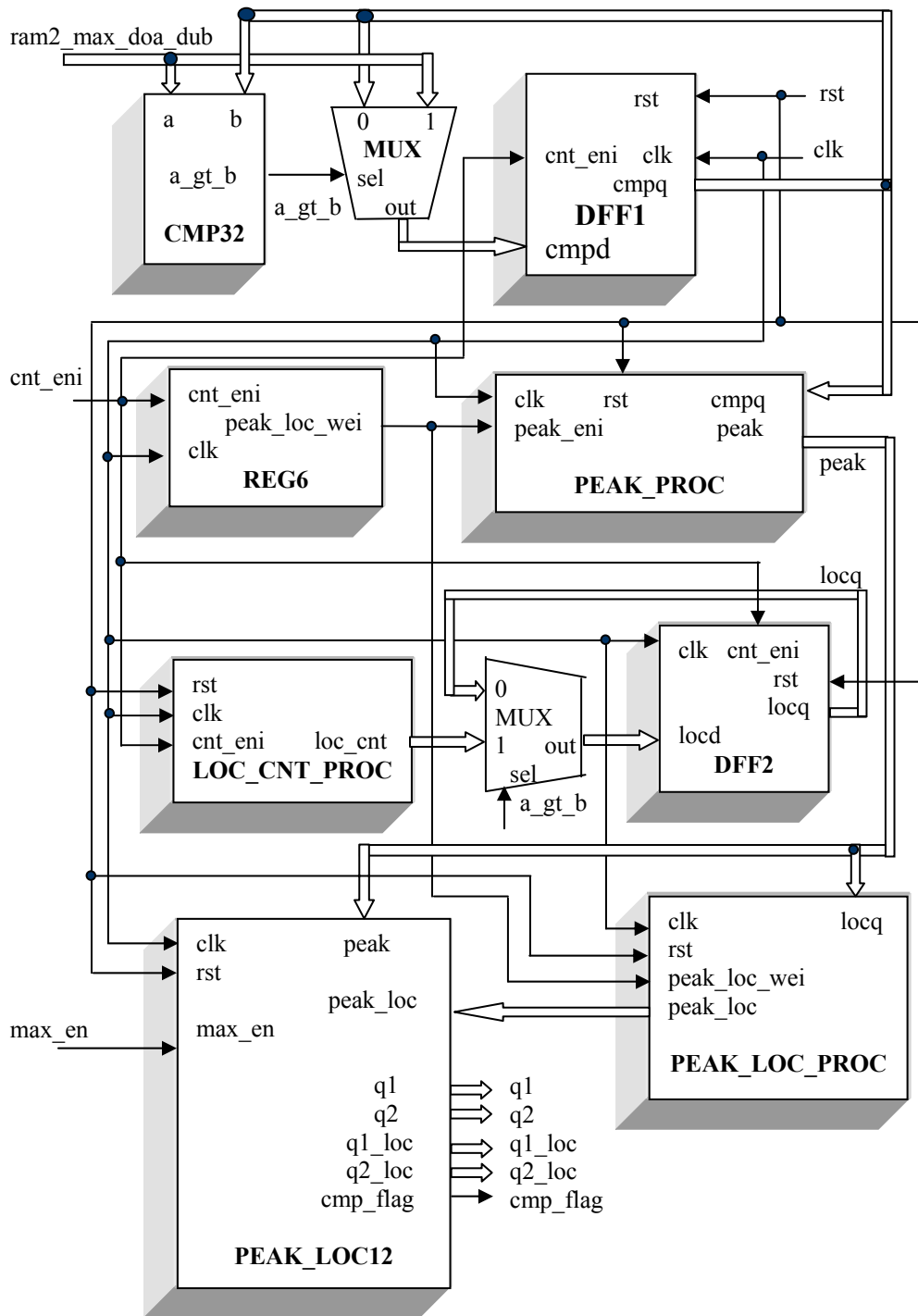


Figure 5.12 Correlation peak and peak location processor diagram

result, the biggest correlation amplitude square peak and peak location are the outputs of PEAK\_LOC12.

### 5.8.1 Correlation Peak Processor

The correlation peak processor is used to find a maximum value among 512 correlation amplitude square results. In Figure 5.12, when the signal `cnt_eni` is valid, the correlation results `ram2_max_doa_dub` are available. The peak searching process is similar to “bubble up.” The peak value is saved in a D flip flop as a signal `CMPQ`. The signal `CMPQ` is initialized with value zero. Then it is compared with a new signal value `ram2_max_doa_dub`. A comparator compares the value of `ram2_max_doa_dub` and `CMPQ`. The comparison logic result is represented by a signal `a_gt_b`. Next, a multiplexer selects the bigger one as the input to the D flip-flop. At the rising edge of the next clock cycle, the signal `CMPQ` keeps the bigger value of the comparison results. After 512 clock cycles, the signal `CMPQ` has the largest correlation peak square results in one loop.

### 5.8.2 Correlation Peak Location Processor

The correlation peak location processor is used to find a maximum value location

among 512 correlation amplitude square results. In Figure 5.12, the peak value location is saved in a D flip flop as a signal locq. First, the signal locq is initialized with value zero. Then it is compared with a sequential counter value loc\_cnt. The location counter is started at the same time when the peak value comparison process is being implemented. The comparison result signal a\_gt\_b is used as a selector for a multiplexer to select the signal loc\_cnt or the signal locq as an input for the D flip flop. As a result, after 1024 clock cycles, the signal locq will have the largest correlation peak square location result.

The important signals and ports used in the Figure 5.12 are given below.

ram2\_max\_doa\_dub: correlation amplitude square value

cnt\_eni: clock enable signal, and logically low when it is active

max\_en: clock enable signal for component PEAK\_LOC12

peak\_loc: a correlation peak location in every 1024 points being processed

peak: a correlation amplitude square peak value in every 1024 points processed

a\_gt\_b: high when the comparator CMP32 input signal value a is greater than b

q1, q1\_loc: the largest correlation amplitude square peak value and its location among all data points being processed

q2, q2\_loc: the second largest correlation amplitude square peak value and its location among all data points being processed

cmp\_flag: high when the comparison is finished

## 5.9 Maximum Selection Unit

The maximum selection unit is PEAK\_LOC12 block in Figure 5.12. The detailed design diagram is drawn in Figure 5.13. The initial values of the maximum, the second and the third maximum correlation results are zeros. The signal max1 keeps track of the previous maximum comparison result. Signals max2 and max3 correspond to the old second and the third maximum comparison results. Signals max1\_loc, max2\_loc and max3\_loc are corresponding correlation peak locations. When the signal max\_en is high, a new value of the signal peak is available and the comparison process is started by the maximum selection unit. The values of max3 and max3\_loc are updated by the values of peak and peak\_loc respectively. Next, the values of max1 and max2 are compared with the value peak. If the signal max31\_gt is high, the signal peak will have a bigger value than the signal max1. If the signal max32\_gt is high, the signal peak will have a bigger value than the signal max2. These comparison results affect the values of max\_ind and mid\_ind, which are used to select the maximum and the 2<sup>nd</sup> maximum values among the values of peak, old max1 and old max2. Then the selected new values are used to replace the values of max1 and max2. Their locations are also switched correspondingly. The results are registered as signals cp1, cp2, cp1\_loc and cp2\_loc respectively. A state machine called MAX12\_PROC is used to control the whole procedure. The maximum and the second maximum correlation peak amplitude square values and their locations

are the final outputs of the maximum selection unit. The signal `cmp_flag` is high when the comparison is finished. The signal `done_cmp` is high when the comparison is finished or the comparison is not started yet.

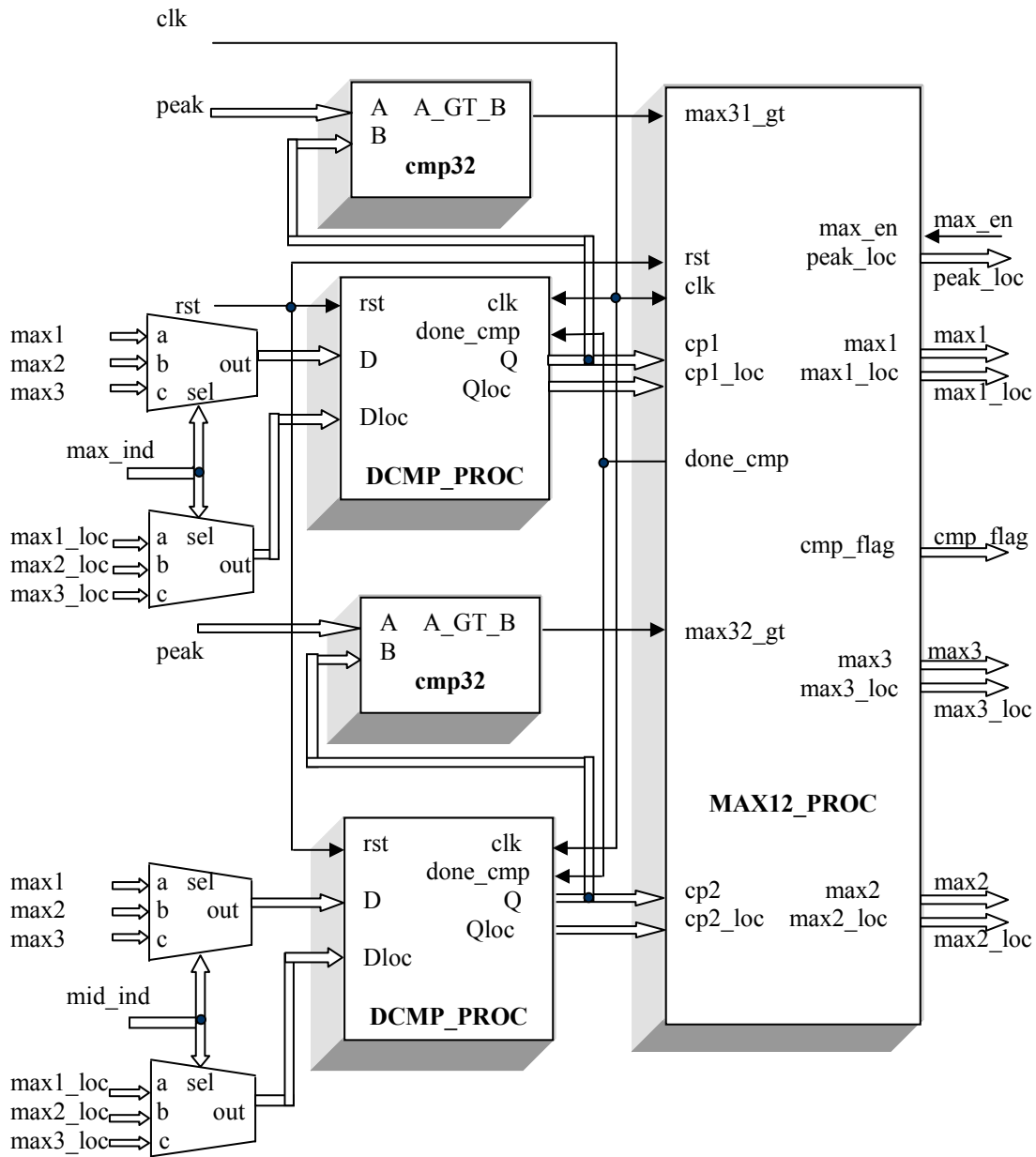


Figure 5.13 Maximum selection unit diagram

## 5.10 Hardware Design Cost

The architecture is implemented on a Xilinx FPGA VirtexE xcv1600E chip. The implementation cost is listed in Table 5.1.

Table 5.1 VirtexE FPGA design cost

	1	2	3	4	5	6	7	8
CLB slices	68*	182	25	1866	1866	559	386	55
Block Rams	0	0	0	16	16	0	0	0

Note:

1. NCO    2. P-code generator    3. Average    4. FFT    5. IFFT

6. Complex conjugate multiplication

7. Correlation amplitude square

8. Peak selection and decision logic

Total available CLB slices: 15552

Total available Block Rams: 144

\*: Cost Estimation

The preprocessing of the demodulated GPS signals on the PC includes averaging the demodulated GPS signals and taking FFT on them. The design implemented in this dissertation uses one P-code generator, one average unit, one FFT unit, one IFFT unit, one complex conjugate square unit, one correlation amplitude square unit, one peak selection and decision unit listed in Table 5.1. The total cost is a little more than one third of the FPGA hardware resources.

## 5.11 Conclusion

The direct GPS P-Code acquisition architecture presented in this chapter consists of eight units: the local reference generation unit, the local reference FFT processor, the complex conjugate multiplication processor, the IFFT processor, the correlation amplitude square unit, the correlation peak unit, the correlation peak location unit, and the maximum selection unit. The block diagram and major signal flows in each unit are illustrated in different sections of this chapter. The necessary design flow chart is also given. The design technical details are described. The implemented VirtexE FPGA design cost is estimated. Since only a little more than one third of the FPGA hardware resources is consumed, this is good for routing the final design successfully.

The whole design has a sequential architecture. Two FFT/IFFT cores are used. One is for the forward FFT and another one is for the inverse FFT. Another possible

architecture for acquisition speedup may exist for more parallel P-Code generators. But this also means more FFT/IFFT cores are needed. The upper limit is set by the hardware resources available on one FPGA chip.



## Chapter 6

# FPGA Design Verification

### 6.1 Introduction

In this chapter, all hardware design steps are verified for 10 ms GPS P-code acquisition processing. The same data sets are used both by FPGA and Matlab for the verification of each design step. Then the differences between them are compared. Comparison follows the order of the P-code average with scaling, FFT, complex multiplication, IFFT, correlation peak square, maximum peak, noise floor and their code phase shift.

Four cases are tested. One is for GPS signals without either noise or code phase shift relative to the reference signals. The second is for GPS signals without noise but with code phase shift. The third and the fourth cases correspond to GPS signals having no code phase shift and having code phase shifts. Moreover, noise is added into GPS signals. The  $C/N_0$  is 55 dB-Hz. The Matlab FFT and IFFT results are rounded before comparisons are made because FPGA works only with integer numbers.

First, a one second GPS P-code is generated using a C program based on the behavior of the GPS P-code generator. The results are saved and then used by Matlab

programs. The FPGA generated results can be read from FPGA to the PC. Next, the comparisons can be made. The comparisons show very promising results. Most comparisons give zero errors. Both FFT and IFFT operations have real part and imaginary part differences within a  $-3$  to  $3$  range. This is only the difference of integer data and rounded floating data used. The final results illustrate exactly the same code phase shift and approximately similar maximum peak and noise floor values.

## 6.2 Verification of P-code Average With Scaling

GPS P-code chips with 0 or 1 values are first generated by the FPGA chip. Then these values are converted to the binary format with values equal to 1 or -1. Every 20 chips are added together for 10220 chips. The the last 10 chips are added. So totally 512 averaged points are generated for each ms GPS P-code reference. These results are padded with 512 zeros to obtain 1024 input data for the further FFT processing. The FFT core requires the input data width of 16 bits. The absolute summation value is not greater than 32. In this dissertation, the 1024 input data values are scaled up by 2048. This is equivalent to shift the input data value left by 11 bits in the binary operation. The results are within the 16-bit range. These results are read back from the FPGA to the PC and they are scaled down by 2048 before they are compared with the Matlab summation results. The results are illustrated in Figure 6.1.

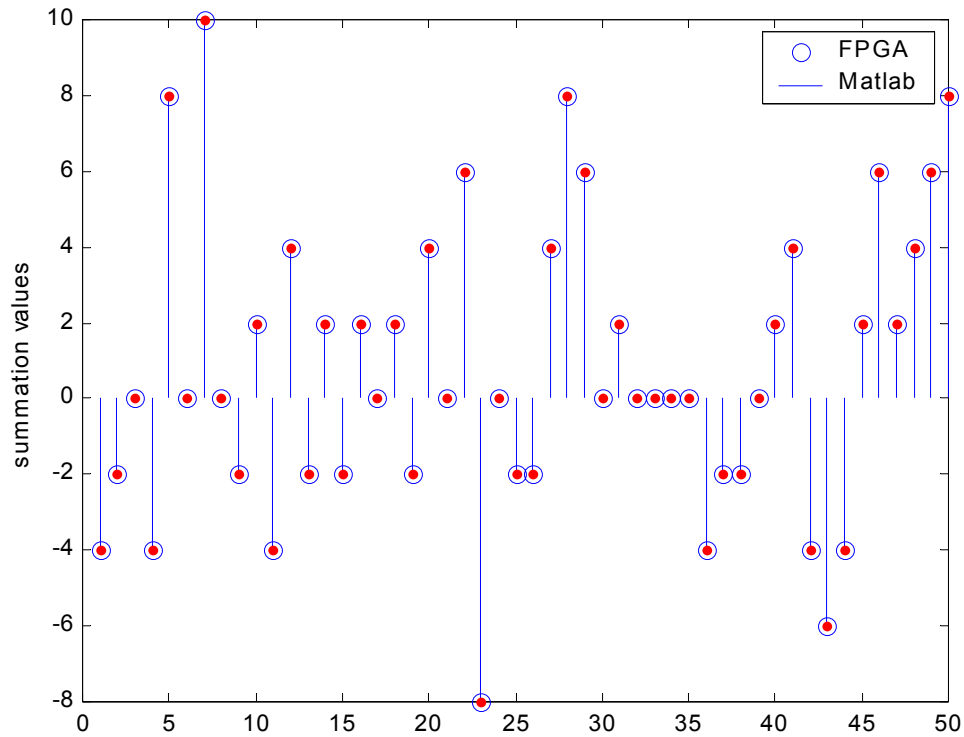


Figure 6.1 The 2nd ms GPS binary P-code averaging results

The Figure 6.1 shows that the Matlab simulation results are the same with that of the FPGA design.

### 6.3 Verification of FFT

The FPGA results from the verification in Section 6.2 correspond to the

scaled summation results. These are used by the Matlab function `fft`, which does the floating point operation internally. The FFT results are scaled by  $1/1024$  and then rounded for comparison with the FPGA FFT results. The FPGA FFT results are shown in Figures 6.2 and 6.3. The comparisons between FPGA and Matlab results are made in Figures 6.4 and 6.5.

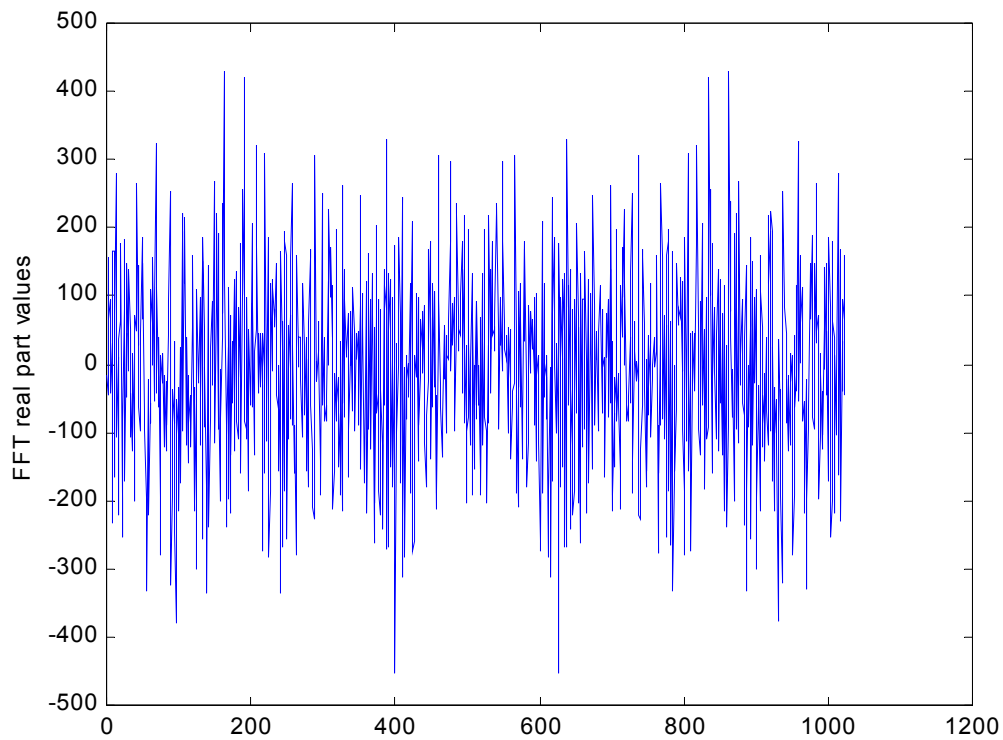


Figure 6.2 FPGA FFT real part values in the 2<sup>nd</sup> ms

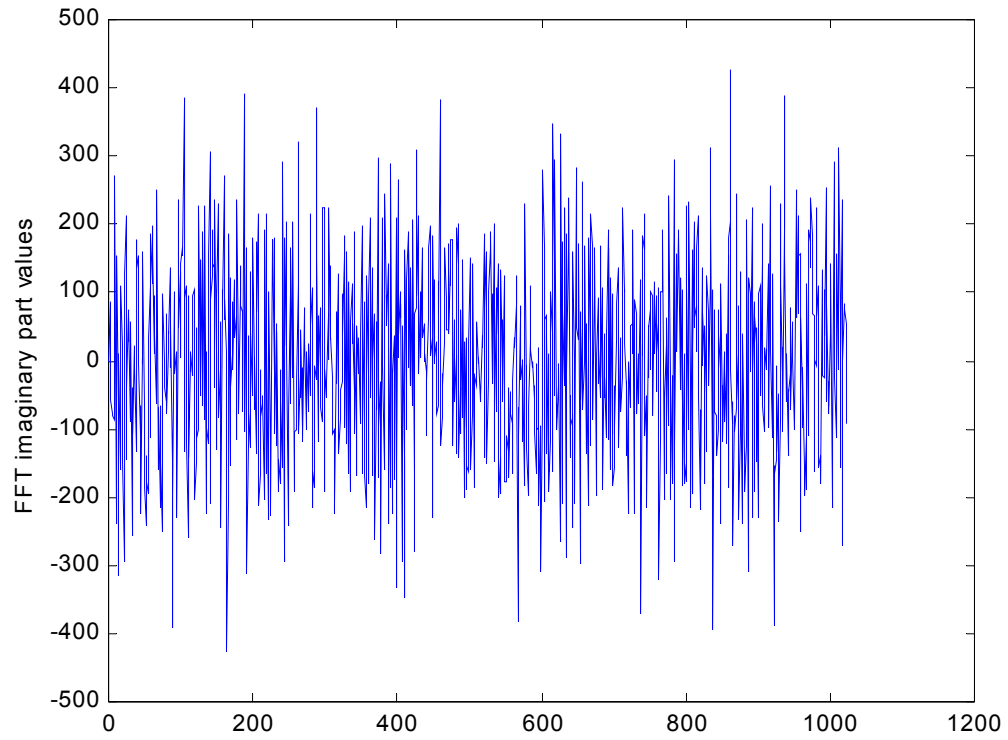


Figure 6.3 FPGA FFT imaginary part values in the 2<sup>nd</sup> ms

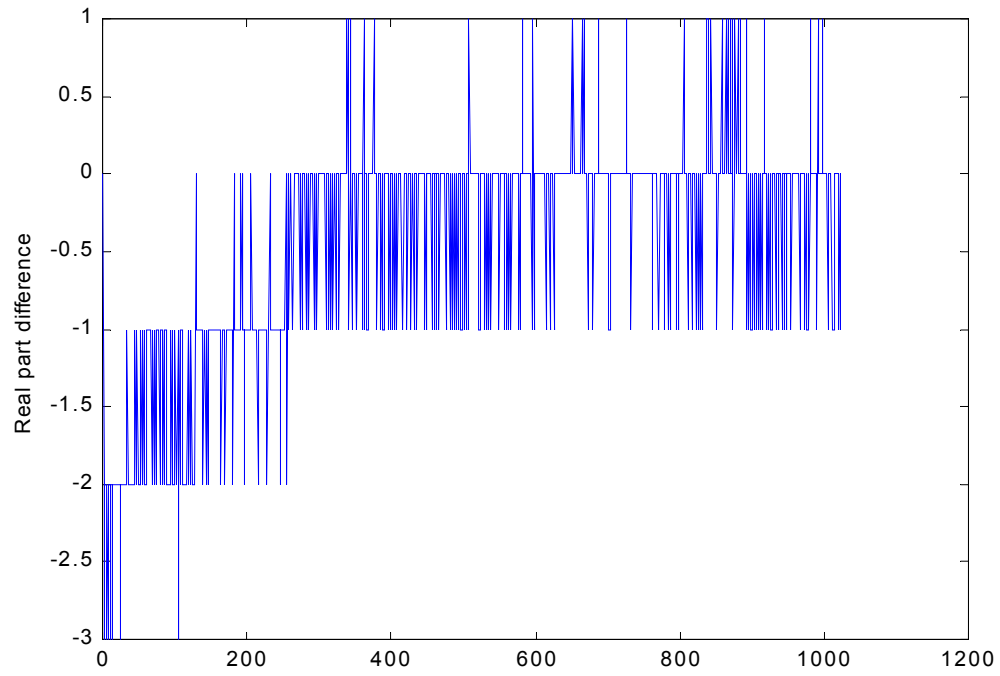


Figure 6.4 FPGA and Matlab FFT real part difference in the 2<sup>nd</sup> ms

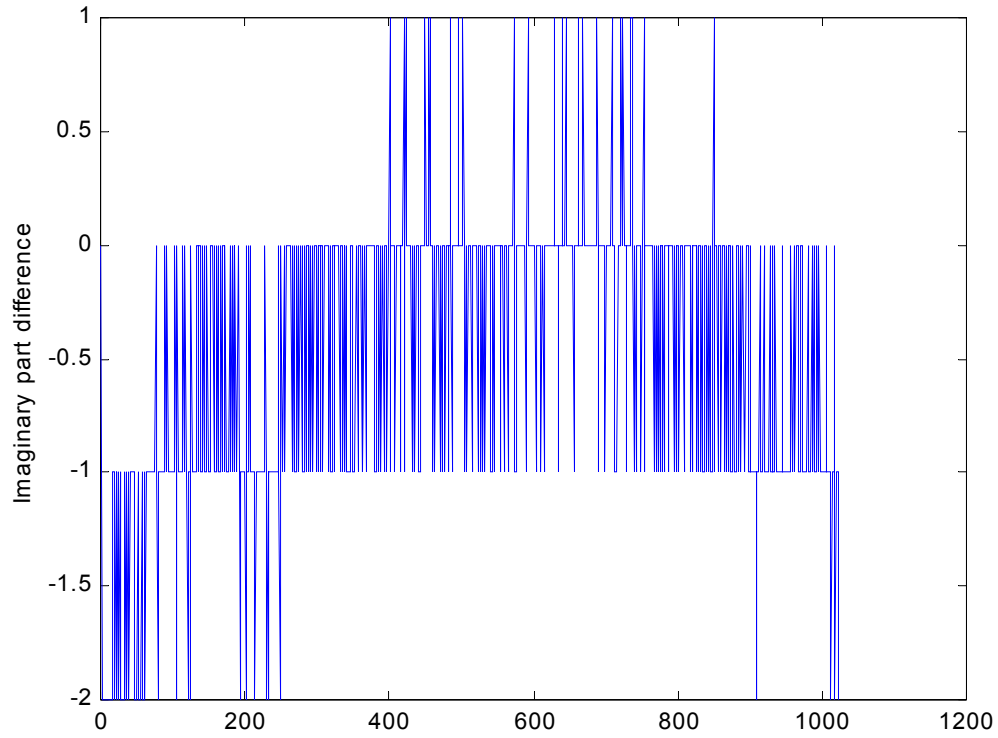


Figure 6.5 FPGA and Matlab FFT imaginary part difference in the 2<sup>nd</sup> ms

The results in Figures 6.4 and 6.5 show the small amount of random difference between  $-3$  to  $3$  exists on the real and the imaginary part of the FFT operation.

## 6.4 Verification of Complex Conjugate Multiplication

The 2ms GPS signal FFT results are precalculated using Matlab. The results are rounded and sent to the FPGA board for the complex multiplication. The GPS signal sampling rate is 65.536 MHz used in Figures 6.6 and 6.7. The complex multiplication, the real and imaginary part data value ranges are set to less than 16 bits, every 128 GPS signals are averaged and Matlab function `fft` is called. The FPGA results from Section 6.3 are used to do complex conjugate multiplication for the verification in this section.

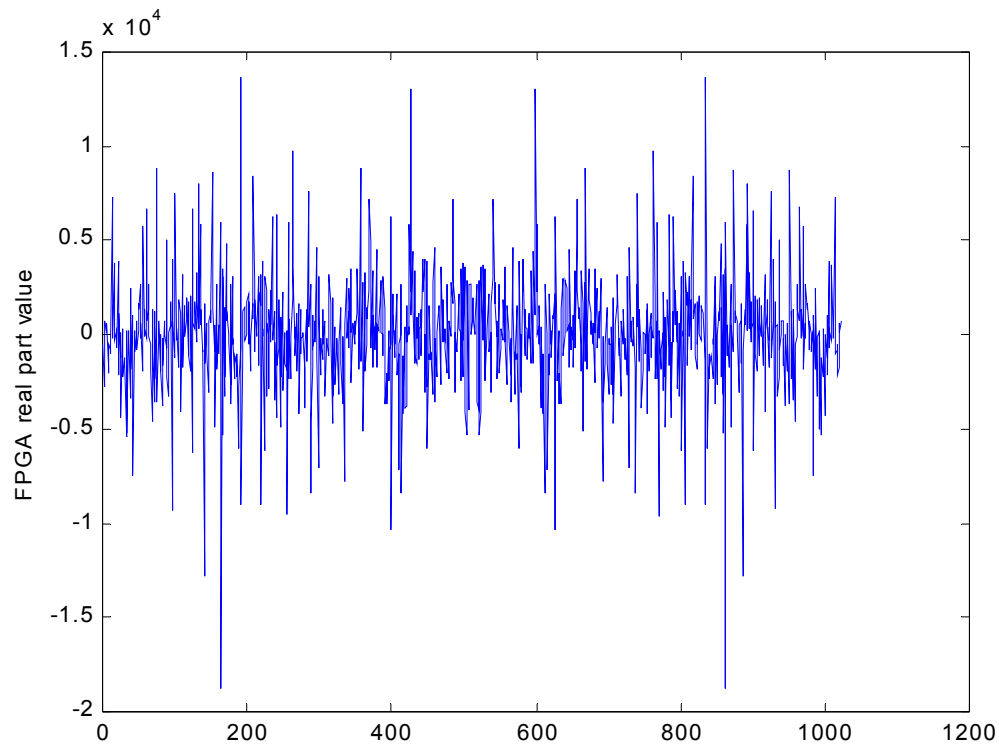


Figure 6.6 The FPGA complex conjugate multiplication real part in the 2<sup>nd</sup> ms



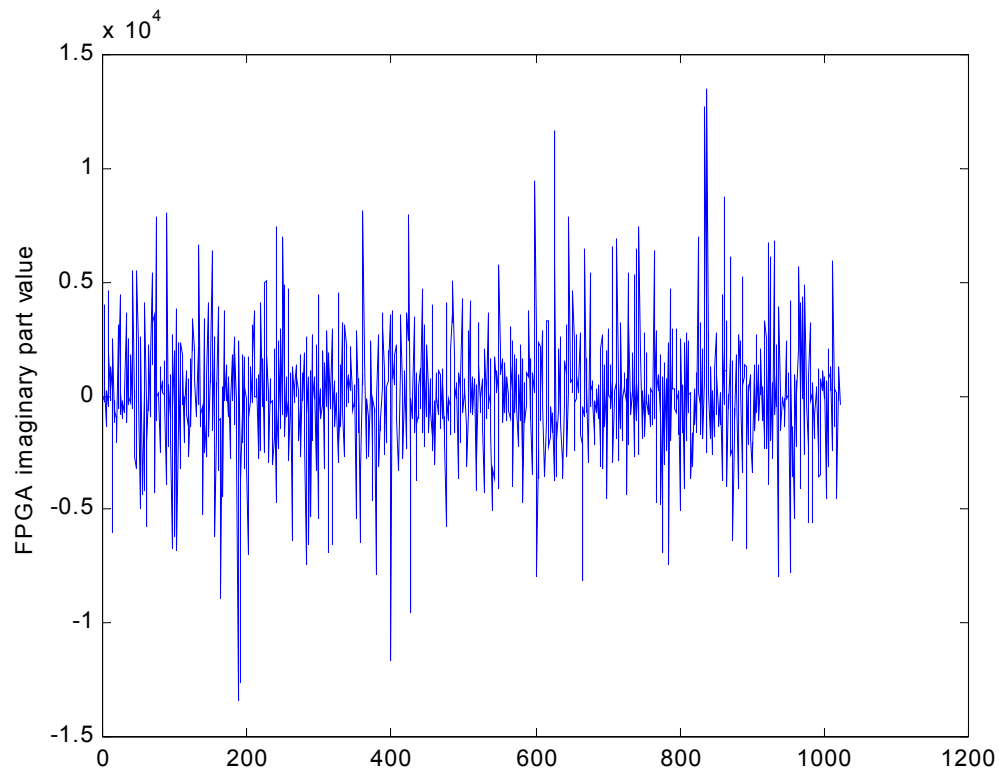


Figure 6.7 The FPGA complex conjugate multiplication imaginary part in the 2<sup>nd</sup> ms

The FPGA and Matlab complex conjugate multiplication results have no difference. From Figures 6.6 and 6.7, we can see the real and the imaginary part data values are within 16-bit range including the sign bit. These are specially chosen for the correlation square calculation in the next section.

## 6.5 Verification of IFFT

The Xilinx 1024-point FFT/IFFT core doesn't scale the IFFT results. So the Matlab program only needs to call function `ifft` and then do rounding. Figures 6.8 and 6.9 illustrate the FPGA IFFT results real and imaginary components in the 2<sup>nd</sup> ms.

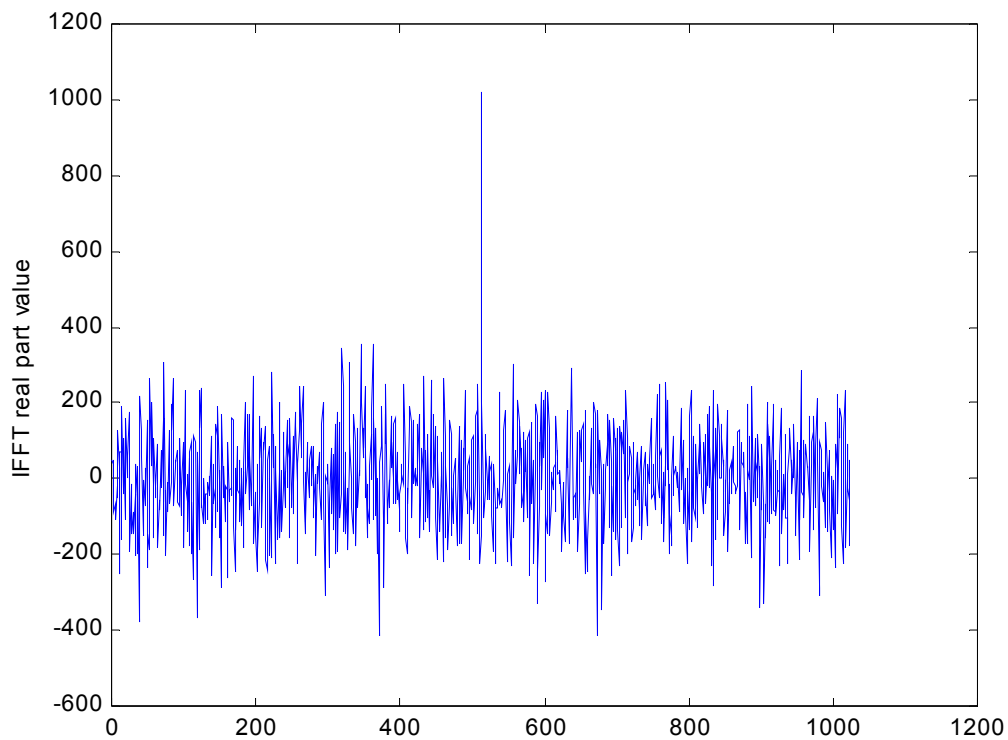


Figure 6.8 FPGA IFFT real part in the 2<sup>nd</sup> ms

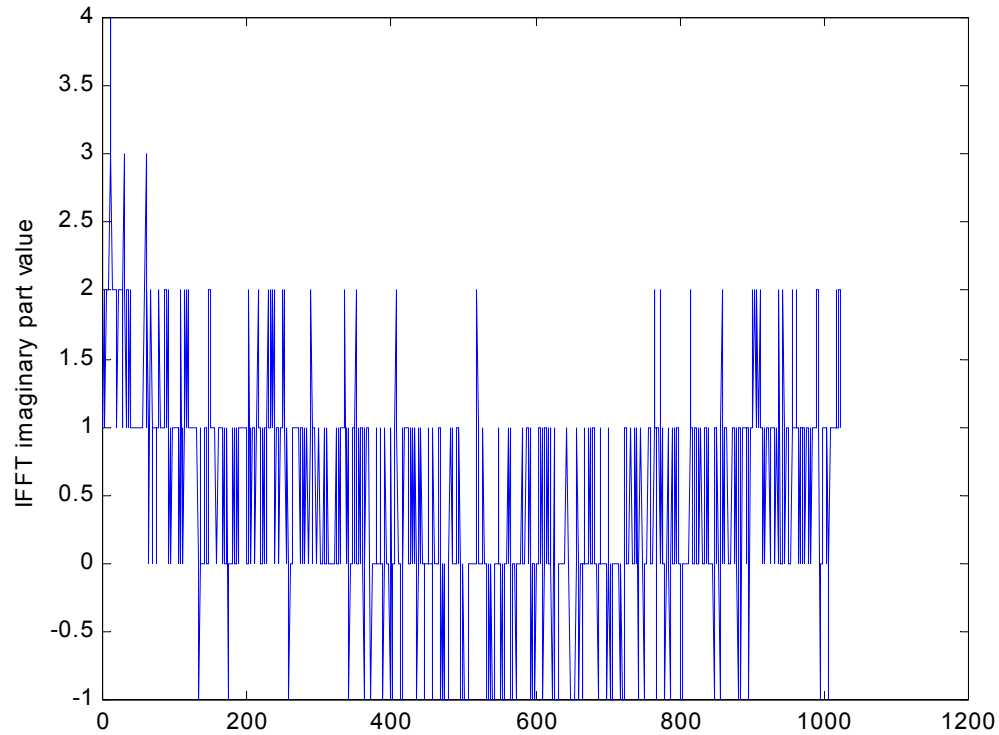


Figure 6.9 FPGA IFFT imaginary part in the 2<sup>nd</sup> ms

Although 1024 values are displayed in Figures 6.8 and 6.9, only the first half is useful. The 2<sup>nd</sup> half is discarded in the further correlation peak amplitude square calculation process.

Figures 6.10 and 6.11 illustrate the real and imaginary part difference between the FPGA and Matlab IFFT results in the 2<sup>nd</sup> ms. The real parts have big absolute values and the imaginary parts have small absolute values.

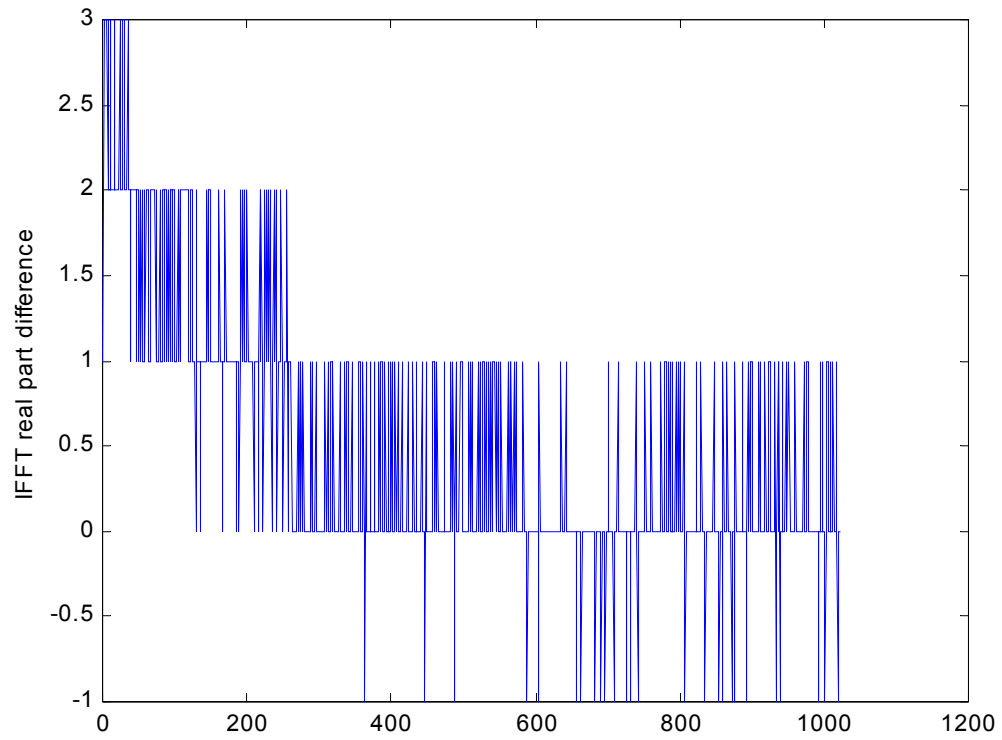


Figure 6.10 FPGA and Matlab IFFT real part difference in the 2<sup>nd</sup> ms

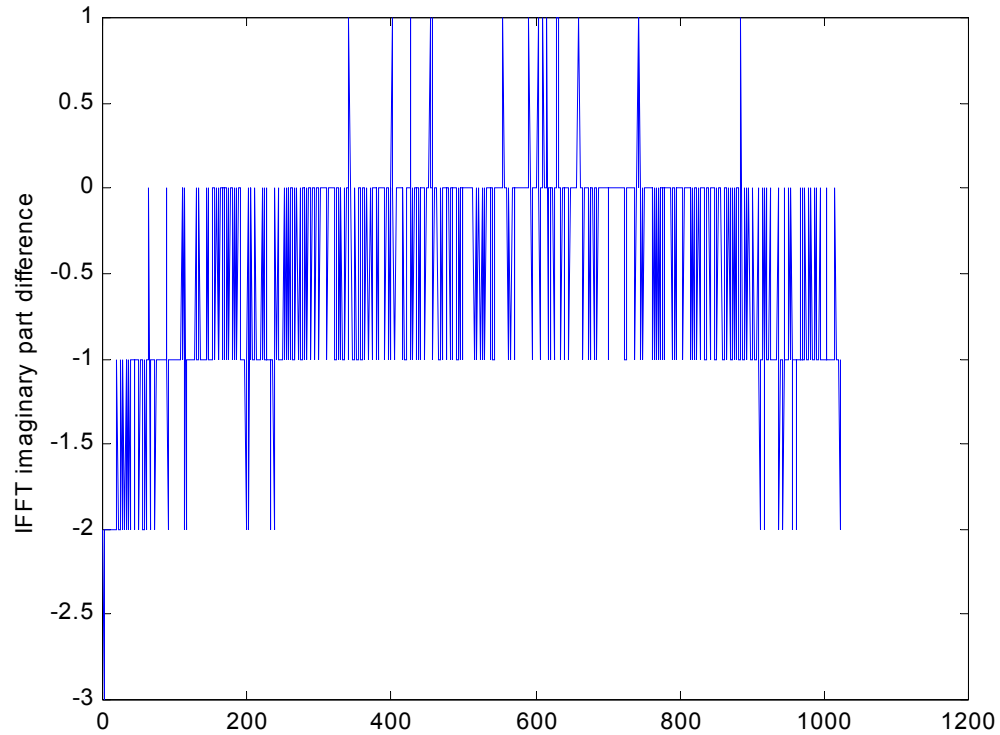


Figure 6.11 FPGA and Matlab IFFT imaginary part difference in the 2<sup>nd</sup> ms

In both Figures 6.10 and 6.11, the difference is still within a -3 to +3 range. So the conclusion is that the difference is not dependent on the magnitude of the absolute values. The difference is caused by the truncation of floating point data into integer values.

## 6.6 Verification of Correlation Square

The previous FPGA complex multiplication results are used as inputs to the Matlab program to calculate the correlation square values. Then the Matlab and the FPGA correlation square output values are compared. They are the same. Both FPGA and Matlab generate exactly the same results. Figure 6.12 only takes the first half of the correlation square values. The other half values are discarded as a result of the noncircular correlation calculation using the FFT and the zero padding method.

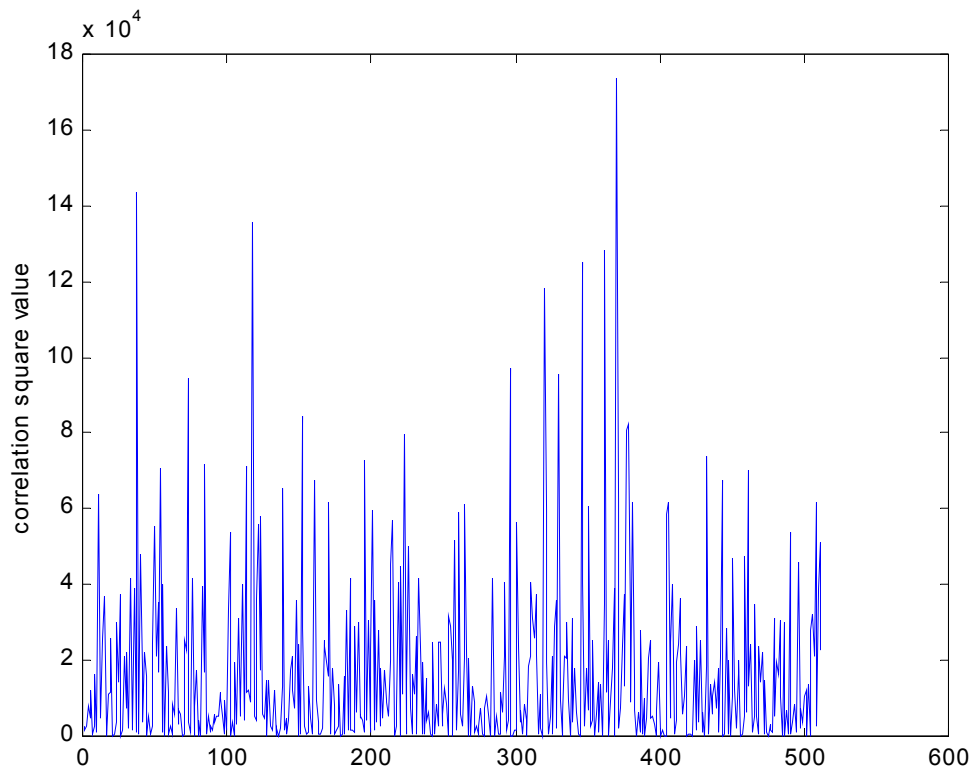


Figure 6.12 FPGA correlation square values in the 2<sup>nd</sup> ms

## 6.7 Final Verification Results

Case 1: GPS signals have no noise and no code phase shift relative to the reference.

Table 6.1 Correlation square peak and code phase shift over 10 ms in the verification of case 1

	1 <sup>st</sup> ms	2 <sup>nd</sup> ms	3 <sup>rd</sup> ms	4 <sup>th</sup> ms	5 <sup>th</sup> ms
Correlation square peak	556516	30276	17161	21316	19044
Code phase shift	0	218	280	455	301
	6 <sup>th</sup> ms	7 <sup>th</sup> ms	8 <sup>th</sup> ms	9 <sup>th</sup> ms	10 <sup>th</sup> ms
Correlation square peak	22500	15129	21904	25281	19881
Code phase shift	335	367	99	489	246 and 258

In Table 6.1, the code phase shift of case 1 corresponds to 128-sample code phase shift due to the averaging operation.

The maximum correlation square value of 556,516 with a code phase shift of 0 is obtained from the Matlab simulation. The second maximum correlation square value is 30,276 with code phase shift of 218. FPGA shows the maximum correlation square value 550,568 with a code phase shift of 0 is obtained from the FPGA design. The second maximum peak value is 30,277 with a code phase shift of 218. FPGA and Matlab have the similar maximum and the second maximum correlation square values with the same

code phase shift values. This conclusion is true for all following three cases.

Case 2: GPS signals have no noise, but with a code phase shift of  $(1 + 65536*7 - 128*8 + 10)$  samples relative to the reference in Table 6.2.

Table 6.2 Correlation square peak and code phase shift over 10 ms in the verification of case 2

Matlab Results					
	1 <sup>st</sup> ms	2 <sup>nd</sup> ms	3 <sup>rd</sup> ms	4 <sup>th</sup> ms	5 <sup>th</sup> ms
Correlation square peak	16129	17689	19321	<b><u>41209</u></b>	21316
Code phase shift	133	177	21	<b><u>39</u></b>	413
	6 <sup>th</sup> ms	7 <sup>th</sup> ms	8 <sup>th</sup> ms	9 <sup>th</sup> ms	10 <sup>th</sup> ms
Correlation square peak	29929	24336	<b><u>763876</u></b>	24336	30976
Code phase shift	270	273	<b><u>8</u></b>	286	366
FPGA Results					
	Maximum		Second maximum		
Correlation square	<b><u>760388</u></b>		<b><u>42026</u></b>		
Code phase shift	<b><u>8</u></b>		<b><u>39</u></b>		



Case 3: GPS signals with added noise: C/No ratio 55 dB-Hz. There is no code phase shift relative to the reference in Table 6.3.

Table 6.3 Correlation square peak and code phase shift over 10 ms in the verification of case 3

Matlab Results					
	1 <sup>st</sup> ms	2 <sup>nd</sup> ms	3 <sup>rd</sup> ms	4 <sup>th</sup> ms	5 <sup>th</sup> ms
Correlation square peak	<b><u>702244</u></b>	270400	301401	173889	208849
Code phase shift in each ms	<b><u>0</u></b>	369	444	383	77
	6 <sup>th</sup> ms	7 <sup>th</sup> ms	8 <sup>th</sup> ms	9 <sup>th</sup> ms	10 <sup>th</sup> ms
Correlation square peak	330625	<b><u>389376</u></b>	248004	201601	216225
Code phase shift in each ms	196	<b><u>500</u></b>	485	416	260
FPGA Results					
	Maximum		Second maximum		
Correlation square	<b><u>700570</u></b>		<b><u>390626</u></b>		
Code phase shift	<b><u>0</u></b>		<b><u>500</u></b>		

Case 4: GPS signals with added noise: C/No ratio 55 dB-Hz. There is a code phase shift relative to the reference of  $(1 + 65536*5 - 128 * 8 + 10)$  samples in Table 6.4.

Table 6.4 Correlation square peak and code phase shift over 10 ms in the verification of case 4

Matlab Results					
	1 <sup>st</sup> ms	2 <sup>nd</sup> ms	3 <sup>rd</sup> ms	4 <sup>th</sup> ms	5 <sup>th</sup> ms
Correlation square peak	233289	142129	177241	<b><u>272484</u></b>	231361
Code phase shift in each ms	238	304	418	<b><u>51</u></b>	267
	6 <sup>th</sup> ms	7 <sup>th</sup> ms	8 <sup>th</sup> ms	9 <sup>th</sup> ms	10 <sup>th</sup> ms
Correlation square peak	<b><u>824464</u></b>	179776	245025	254016	263169
Code phase shift in each ms	<b><u>8</u></b>	367	13	11	496
FPGA Results					
	Maximum		Second maximum		
Correlation square	<b><u>822650</u></b>		<b><u>274585</u></b>		
Code phase shift	<b><u>8</u></b>		<b><u>51</u></b>		

## 6.8 Conclusion

Verification is an important procedure in FPGA design. The Nallatech FPGA design platform provides the facility for designers to communicate between the PC and the FPGA board. This is very convenient for designers to test their work. A strategy in this dissertation is to insert some debugging RAMs in different parts of the design for verification. The design is partitioned into blocks which can be tested sequentially. Other systematic tools, such as Microsoft C++, Matlab, Aldec HDL and so on, can be used for the functional simulation to be compared with the FPGA design results. In this dissertation, it is guaranteed that the Matlab simulation results are the same with the FPGA design blocks whenever integer operations are used in 10 ms verification. For the FFT/IFFT operations, floating point operations are done by Matlab. The difference between Matlab and FPGA results for FFT/IFFT is within a -3 to +3 range. Xilinx provides the 1024-point FFT/IFFT behavioral library which uses the integer operation. The behavioral simulation using Xilinx library yielded exactly the same results as the FPGA hardware design results. In conclusion, the FPGA designs for this dissertation are successfully verified.

## Chapter 7

### Summary and Future Work

#### 7.1 Summary

The GPS P-code has higher chipping rate, better accuracy and anti-jamming property than the C/A code. Traditionally, GPS P-code acquisition depends on a handover from the C/A code. The P-code modulates both GPS L1 and L2 carrier frequencies. When L1 carrier signal is not available, direct GPS P-code acquisition is the only acquisition method that can be used. The acquisition techniques described in this dissertation can also be expanded to the pseudolite application. Pseudolite only broadcasts the wideband code, which is similar to the P-code.

The major objectives of this dissertation are to propose a direct GPS P-code acquisition algorithm and implement the direct GPS P-Code acquisition code phase search on the Xilinx FPGA chip. The direct GPS P-Code acquisition algorithm was proposed and its FPGA hardware design and verification issues were analyzed. The obstacles faced by researchers in dealing with extremely long period of P-code were also highlighted. The major contributions of this dissertation are described below.

First, the P-Code correlation property was studied and the acquisition margin was

defined. FFT with the zero padding method was presented to calculate the correlation function of non-periodic code such as a segment of the P-code.

Then, different direct GPS P-code acquisition methods recently reported by other researchers were compared. The strength and weakness of different methods were pointed out. Most of them need a large size FFT which is difficult for hardware implementation.

Next, the direct average method was proposed in this dissertation to achieve fast direct GPS P-code acquisition code phase search. The symmetrical and linear correlation curve as the effect of averaging was discovered. Furthermore, the overlap average method was presented to make up the correlation energy loss during the process of direct averaging.

The methods proposed in this dissertation achieve fast code phase search. They use small size FFT, which is good for hardware implementation, especially for FPGA prototyping. This meets the objective of algorithmic study set for this dissertation. Another aspect not studied in this dissertation is frequency search. The acquisition process implemented in this dissertation assumed the GPS signals were first demodulated before doing code phase search for acquisition.

In hardware design, mathematical equations were developed to design the GPS P-code generator to facilitate P-code generation starting from any time of a week. This yields an important facilitation of P-code acquisition when there is knowledge of the approximate time information. The P-code generator simulation results matched the

ICD-GPS-200 descriptions.

The whole direct GPS P-code acquisition hardware design architecture was partitioned into eight units. All these units were sequentially linked. The interface and signal flows between different units were designed. Besides, efforts were spent on the study of the Nallatech FPGA board and Xilinx VirtexE FPGA features. Especially, the Xilinx 1024-point FFT/IFFT core was integrated into the system design for this dissertation. A controller was designed to satisfy the timing specification of Xilinx datasheet. The FFT/IFFT controller design passed a million times verification successfully. The whole design was downloaded on FPGA and the results were read back by the PC. Each major unit in the whole architecture was verified successfully. The final FPGA acquisition correlation peak and code phase were correct in a 10 ms search process. A 10 ms search process can be transferred to a position uncertainty of  $3 \times 10^6$  meters. This uncertainty can be resolved in 0.333 ms by using a 33 MHz P-code generator as implemented in this dissertation.

This work is challenging due to the extremely long period of the P-code. The proposed overlap average method needs large hardware storing resource, so it was not implemented in this dissertation. Instead, the direct average algorithm was implemented on FPGA design successfully. Since there are lots of applications related to GPS P-code, the work reported in this dissertation is meaningful for future research, especially after a connection of my design with GPS P-code acquisition RF front end, which is now being developed by the Ohio University Avionics Engineering Center.

## 7.2 Future Work

The direct average, overlap average model and FPGA design developed in this work are useful and versatile tools for direct GPS P-code acquisition research. While the focus of this research has been on the direct average code-phase search of GPS P-code acquisition FPGA design, an extension of this work is to the field of carrier phase search. The carrier phase search usually consists of a NCO (Numerically Controlled Oscillator) whose operation principle is well known. It is important to choose an accurate oscillator available on the market, which fits into NCO carrier immediate frequency design with a small round off error. The overlap average method requires larger storing hardware resources. Although this can be achieved by using the ZBTRAM resource on the Nallatech board, more efforts are still needed to develop a ZBTRAM controller.

One potential use of this work is to connect the design with GPS P-code acquisition front end. Another possible extension is to add a tracking loop to get a full receiver design. A faster acquisition speed can be achieved using the parallel P-Code generators for parallel processing. The design techniques used in this dissertation can also be applied to the acquisition of pulsed pseudolites that transmit wideband (shifted P-code) signals.

## REFERENCES

- [BAE, 1998] Precision Landing System Receiver PLSR AN/ARN-155, BAE SYSTEMS' CNI Division product datasheet, 1998.
- [Bauer, 1994] Bauer, W. D., and M. Schefcik, "*Using Differential GPS to Improve Crop Yields*", GPS World, Vol. 5, No. 2, February 1994.
- [Braasch, 1999] Braasch M. and Van Dierendonck A., "*GPS Receiver Architectures and Measurements*", Proc. of the IEEE, Vol. 87, No.1, January 1999.
- [Brigham, 1974] Brigham, E. Oran, **The Fast Fourier Transform**, Prentice-Hall, Englewood Cliffs, NJ, 1974.
- [Brown, GPS Solutions 2000] Brown A., May M., and Tanju B., "*Benefits of Software GPS Receivers for Enhanced Signal Processing*", GPS Solutions, 4(1) Summer, 2000.
- [Brown, ION 2000] Brown A., Neil Gerein, and Keith Taylor, "*Modeling and Simulation of GPS Using Software Signal Generation and Digital signal Reconstruction*", Proc. of ION Technical Meeting, Anaheim, CA, January 2000.
- [Brown, 2001] Brown A. and Gerein N., "*Direct P(Y) Code Acquisition Using An Electro-Optic Correlator*", Proc. of ION National Technical Meeting 2001, Long Beach, CA, January 2001.
- [Chang 1997] Chang K., **Digital Systems Design with VHDL and Synthesis**, IEEE Computer Society Press, 1997.



- [Cobb, 1998] Cobb, S., and M. O'Connor, "*Pseudolites: Enhancing GPS with Ground-Based Transmitters*", GPS World, Vol. 9, No. 3, March 1998, pp. 55-60.
- [Cooley, 1965] Cooley J. and Tukey J., "*An Algorithm for the Machine Calculation of Complex Fourier Series*", Mathematics of Computation, Vol. 10, pp. 297-301, April 1965.
- [Dierendonck, 1996] Dierendonck A., "*GPS receivers*", in **Global Positioning System: Theory and Application, vol. I**, B. W. Parkinson and J. J. Spilker, Jr., Eds. Washington, DC: American Institute of Aeronautics and Astronautics, 1996, ch. 8, pp. 329-407.
- [Drane, 1998] Drane, C., and Rizos C., **Positioning Systems in Intelligent Transportation Systems**, Norwood, MA, Artech House, 1998.
- [El-Rabbany, 2002] El-Rabbany A., **Introduction to GPS: the Global Positioning System**, Artech House, 2002.
- [Erdogan, 2003] Erdogan A. and Arslan T., "*Low Power Block-based FIR Filtering Cores*", 2003 IEEE International Symposium on Circuits and Systems (ISCAS 2003), Bangkok, Thailand, May 2003.
- [Flinn, 1999] Flinn, J. A., Waddell C., and Lowery M. A., "*Practical Aspects of GPS Implementation at the Morenci Copper Mine*", Proc. of ION GPS 1999, Nashville, TN, September, 1999, pp. 915-919.
- [Gerein, 2001] Gerein N. and Brown A., "*Modular GPS Software Radio Architecture*", Proc. of ION GPS 2001, Salt Lake City, UT, September 2001.

- [Glisic, 1997] Glisic S., Vucetic B., **Spread Spectrum CDMA Systems for Wireless Communications**, Artech House, March 1997.
- [Haag, 1999] Haag M., “An Investigation Into the Application of Block Processing Techniques For the Global Positioning System”, Dissertation, August 1999.
- [Hada, 2000] Hada, H., et al., “*The Internet, Cars, and DGPS: Bringing Mobile Sensors and Global Correction Services On Line*”, GPS World, Vol. 11, No. 5, May 2000, pp. 38-43.
- [Hayward, 1998] Hayward R. C., Gebre-Egziabher and D., Powell J. D., “*GPS-Based Attitude For Aircraft*”, The Int. Conf. on Integrated Navigation Systems in St. Petersburg Russia, May 1998.
- [Hellmich, 2000] Hellmich H., Erdogan A. and Arslan T., “*Re-Usable Low Power DSP IP Embedded in an ARM based SoC Architecture*”, IEE Coloquim on Intellectual Property, Edinburgh, UK, July 2000.
- [Hoffmann-Wellenhof, 1994] Hoffmann-Wellenhof, B., Lichtenegger H., and Collins J., **Global Positioning System: Theory and Practice**, 3rd ed., New York, Springer-Verlag, 1994.
- [ICD-GPS-200, 1991] ICD-GPS-200, NAVSTAR GPS Space Segment/Navigation User Interfaces (Public Release Version), ARINC Research Corporation, Fountain Valley CA, 1991.
- [Jensen, 1992] Jensen, M. H., “*Quality Control for Differential GPS in Offshore Oil and Gas Exploration*”, GPS World, Vol. 3, No. 8, September 1992, pp. 36-48.

- [Kaplan, 1990] Kaplan, E., **Understanding GPS: Principles and Applications**, Norwood, MA, Artech House, 1990.
- [Kelley, 2002] Kelley, C., Cheng, J., and Barnes, J., “*Open source software for learning about GPS*”, 15th Int. Tech. Meeting of the Satellite Division of the U.S. Inst. of Navigation, Portland, Oregon, 24-27 September, 2002.
- [Klupsch, 2002] Klupsch St., Ernst M., Huss SA, Rumpf M., and Strzodka R., “*Real Time Image Processing based on Reconfigurable Hardware Acceleration*”, Proc. Workshop on Heterogeneous reconfigurable Systems on Chip, April 2002.
- [Knight, 1979] Knight W. and Kaiser R., “*A Simple Fixed-Point Error Bound for the Fast Fourier Transform*”, IEEE Trans. Acoustics, Speech and Signal Proc., Vol. 27, No. 6, pp. 615-620, December, 1979.
- [Krumvieda, 2001] Krumvieda, K., Cloman C., Olson E., Thomas J., Kober W., Madhani P., and Axelrad P., “*A Complete IF Software GPS Receiver: A Tutorial About the Details*”, ION GPS-2001, Salt Lake City, UT, pp. 789-829, September 2001.
- [Langen, 2002] Langen D., et al., “*Implementation of a RISC Processor Core for SoC Designs – FPGA Prototype vs. ASIC Implementation*”, Proc. of the IEEE-Workshop: Heterogeneous reconfigurable Systems on Chip (SoC), Hamburg, Germany, April 2002.
- [Langley, 1993] Langley, R. B., “*The GPS Observables*”, GPS World, Vol. 4, No. 4, April 1993, pp.52-59.
- [Leick, 1995] Leick, A., **GPS Satellite Surveying**, 2<sup>nd</sup> ed., New York, Wiley, 1995.

[Lin, 1998] Lin D. and Tsui J., “*Acquisition Schemes for Software GPS Receiver*”, Proc. of ION GPS’98, Part 1, pp. 317-326, September 1998.

[Lin, 1999] Lin D., Tsui J., Howell D., “*Direct P(Y)-Code Acquisition Algorithm for Software GPS Receivers*”, The Proc. of ION International Technical Meeting 1999, Nashville, Tennessee, September 1999.

[Lin, 2000] Lin D., and Tsui J., “*Comparison of Acquisition Methods for Software GPS Receiver*”, Proc. of ION GPS, 2000.

[Lucent, 2000] “Lucent Technologies introduces hybrid systems-on-a-chip for 10 Gigabit Ethernet over optical networks”, Lucent Technologies, November 2000.

[Luk, 1998] Luk W., P. Andreou, Derbyshire A., Dupont-De-Dinechin F., Rice J., Shirazi N., and Siganos D., “A Reconfigurable Engine for Real-Time Video Processing”, in **Field-Programmable Logic and Applications 1998**, Tallinn, Estonia, pp. 169-178.

[Masella, 1999] Masella, E. , “Achieving 20cm positioning accuracy in real time using GPS - the Global Positioning System”, GEC Review, 14(1), pp. 20-28, 1999.

[Matsumoto, 1999] Matsumoto C., “LSI Logic ASICs to add Programmable Logic Cores”, E.E. Times, August 29, 1999.

[Matsumoto, 2000] C. Matsumoto, “Startup Puts a Fresh Spin on Programmable Cores”, E.E. Times, September 15, 2000.

[McCullagh, 1999] McCullagh M., “*A Single Chip Silicon Bipolar Receiver for GPS/GLONASS Applications*”, IEEE International Solid-State Circuits Conference, 1999.

- [Mencer, 2003] Mencer O. and Luk W., **Parameterized High Throughput Function Evaluation for FPGAs**, The Journal of VLSI Signal Processing, Special Issue on Reconfigurable Computing, Kluwer 2003.
- [Misra, 1999] Misra P., Burke B., and Pratt M., “*GPS Performance in Navigation*”, Proc. of the IEEE, Vol. 87, No.1, January 1999, pp. 65–85.
- [Merritt, 2000] Merritt R., “QuickLogic Steps up Merger of FPGA with IP Cores – DSP First Target” , E.E. Times, August 9, 2000.
- [Misra, 2001] Misra P. and Enge P., **Global Positioning System: Signals, Measurements, and Performance**, Lincoln, Mass., Ganga-Jamuna Press, 2001.
- [Nallatech, 2002] Strathnuey PCI Card User Guide, NT 107-0076, Issue 9, Nallatech datasheet, 2002.
- [Navambi, 1993] Navambi Z., **VHDL Analysis and Modeling of Digital Systems**, Mc. Graw Hill, 1993.
- [Ndili, 1994] Ndili A., “*GPS Pseudolite Signal Design*”, Proc. of ION GPS 1994, Salt Lake City, Utah, September 1994.
- [Ohr, 2000] Ohr S., “ADI Taps Systolix Processor Array”, E.E. Times, April 21, 2000.
- [Pace, 1995] Pace S., Frost G., et al., **The global positioning system: assessing national policies**, Santa Monica, CA, RAND, 1995.
- [Parkinson, vol. I, 1996] Parkinson B. and Spilker J., **Global Positioning System: Theory and Applications, volume I**, of Progress in Aeronautics and Astronautics, AIAA, 1996.

- [Parkinson, vol. II, 1996] Parkinson B. and Spilker J., **Global Positioning System: Theory and Applications, Volume II**, American Institute of Aeronautics and Astronautics, 1996.
- [Piazza, 1998] Piazza F. and Huang Q., “*A 1.57-GHz RF Front-End for Triple Conversion GPS Receiver*”, IEEE Journal of Solid-State Circuits, Vol. 33, No. 2, February 1998.
- [Prasad, 1996] Prasad R., **CDMA for Wireless Personal Communications**, Boston, Artech House, 1996.
- [Proakis, 2000] Proakis J., **Digital Communications**, Fourth Edition, McGraw Hill, 2000.
- [Psiaki, 2001] Psiaki M., “*Smoother-Based GPS Signal Tracking in a Software Receiver*”, Proc. of ION GPS, Long Beach, California, 2001.
- [Rabiner, 1975] Rabiner L. and Gold B., **Theory and Application of Digital Signal Processing**, Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1975.
- [Rappaport, 1995] Rappaport T., **Wireless Communications: Principles & Practice**, Prentice Hall, 1995.
- [Savage, 2000] Savage W., Chilton J., and Camposano R., “*IP Reuse in the System on a Chip Era*”, 13th International Symposium on System Synthesis (ISSS'00), Madrid, Spain, September 2000.

[Shaeffer, 1998] Shaeffer D., Shahani A., Mohan S., et al., “*A 115-mW, 0.5-um CMOS GPS Receiver with Wide Dynamic-Range Active Filters*”, IEEE Journal of Solid-State Circuits, Vol. 33, No. 12, December 1998.

[Shahani, 1997] Shahani A., Shaeffer D., and Lee T., “*A 12mW Wide Dynamic Range CMOS Front-End for a Portable GPS Receiver*”, Proc. IEEE Int. Solid-State Circuits Conf., vol. 40, pp. 368-69, February 1997.

[Stone, 1999] Stone J., LeMaster E., Powell J., and Rock S., “*GPS Pseudolite Transceivers and their Applications*”, Proc. of ION National Technical Meeting 1999, San Diego, CA.

[Tanurhan , 2001] Tanurhan Y., “SOCs and Embedded Reprogrammable Logic”, E.E. Times, March 29, 2001.

[Trimberger, 1994] Trimberger S., **Field-Programmable Gate Array Technology**, Kluwer Academic Publishers, 1994.

[Trimble, 1998] MS750 Dual-Frequency RTK Receiver for Marine Applications, Trimble Navigation Limited Corporation product data sheet, 1998.

[Trimble, 2002] GRAM compliant, SAASM based Force™ 22 Miniature Ruggedized Unit (MRU), Trimble Navigation Limited Corporate Product Data sheet, 2002.

[Tsui, 2000] Tsui J., **Fundamentals of Global Positioning System Receivers, A Software Approach**, John Wiley & Sons Publishers, 2000.

[van Nee, 1991] van Nee, D. J. R., Coenen, A. J. R. M., “*New fast GPS code acquisition technique using FFT*”, Electronics Letters, vol.27, pp. 158-160, January 1991.

[Viterbi, 1995] Viterbi A., **CDMA: principles of spread spectrum communication**, Addison-Wesley Pub Co, 1995.

[Vladimirova, 2000] Vladimirova T., Tiggeler H., and Zheng D., “*A System-on-a-chip for Small Satellite Data Processing and Control*”, The 3<sup>rd</sup> annual Military and Aerospace Programmable Logic Device International Conference (MAPLD), Maryland, September, 2000.

[Wang, 2000] Wang, J., et al., “*Integrating GPS and Pseudolite Signals for Position and Attitude Determination: Theoretical Analysis and Experiment Results*”, Proc. of ION GPS 2000, Salt Lake City, UT, September, 2000, pp. 2252-2262.

[Ward, 1995] Ward P., “*The natural measurements of a GPS receiver*”, Proc. ION 51st Annual Meeting, 1995, pp. 67–85.

[Wilton, 2001] Wilton S. and Saleh R., “*Programmable Logic IP Cores in SoC Design: Opportunities and Challenges*”, the 2001 IEEE Custom Integrated Circuits Conference, San Diego, CA, May 2001.

[Wolfert, 1998] Wolfert R., Chen S., Kohli S., Leimer D., and Lascody J., “*Rapid Direct P(Y)-Code Acquisition In a Hostile Environment*”, Proc. of IEEE PLANS, April 1998.

[Xilinx, 2000] High-Performance 1024-Point Complex FFT/IFFT V1.0.5, Xilinx Product Specification datasheet, July 2000.



[Xilinx, 2002] Virtex™-E 1.8 V Field Programmable Gate Arrays, DS022-1 (v2.3), Xilinx Product Specification, July 2002.

[Xilinx White Paper, 2002] IBM Licenses Embedded FPGA Cores from Xilinx for Use in SoC ASICs (v1.0), Xilinx White Paper, June 2002.

[Xilinx, 2003] Virtex-II Pro Datasheet, Xilinx Databook, 2003.

[Yang, 2000] Yang C., Chaffee J., Abel J., and Vasquez J., “*Extended Replica Folding for Direct Acquisition of GPS P-Code and Its Performance Analysis*”, Proc. of ION GPS 2000, Salt Lake City, Utah.

[Yang, 1999] Yang C., Vasquez J., and Chaffee J., “*Fast Direct P(Y)-Code Acquisition Using XFAST*”, Proc. of ION GPS, 1999.

[Yang, 2001] Yang C., “*Sequential Block Search for Direct Acquisition of Long Codes under Large Uncertainty*”, ION NTM 2001, January 2001, Long Beach, CA.

[Zyfer, 2002] “SAASM and Direct P(Y) signal acquisition, a better way of life for the military GPS user”, Zyfer Inc. SAASM GPS White Paper, Anaheim, CA, April 2002.

## Appendix A

### The FPGA Board

In this Appendix, the introduction of the Strathnuey FPGA board is presented. Nallatech is a technology leader in the design of reconfigurable computers. Based on Nallatech's award-winning Dime standard, its Strathnuey motherboard is an FPGA-centric modular systems platform used for DSP, imaging, telecommunication and aerospace applications. This Strathnuey card features two Xilinx FPGAs and one DIME module expansion slot. FPGAs contain a Xilinx PCI Logic Core interface together with Nallatech's own PCI Bridge interface. This bridge provides the FPGA designer with a method to implement registers and blocks of memory that are easily accessible by the PC. The Spartan-II PCI FPGA is supplied pre-configured with Nallatech firmware for PCI interfacing/board control. The VirtexE FPGA is available exclusively for user design and applications. Besides, it has three on-board programmable clock sources, A/D converter, D/A converter, status LEDs, and JTAG configuration headers. It also has a 32 bit PCI connector. Moreover, the DIME Software DLL Library facilitates user with high level functions.

The internal structure of a Xilinx VirtexE is a Configurable Logic Block (CLB) matrix surround by Input/Output Blocks. Each CLB has two SLICES. Each VirtexE SLICE has two 4 input lookup tables and two flip-flops. There are also carry logics for the fast carry line. CLBs are connected through Generic Routing Matrix (GRM). There are three-state buffers associated with each CLB that drive dedicated segmentable horizontal routing resources. There are dedicated block memories of 4096 bits each. Moreover, there are clock DLLs for clock-distribution delay compensation.

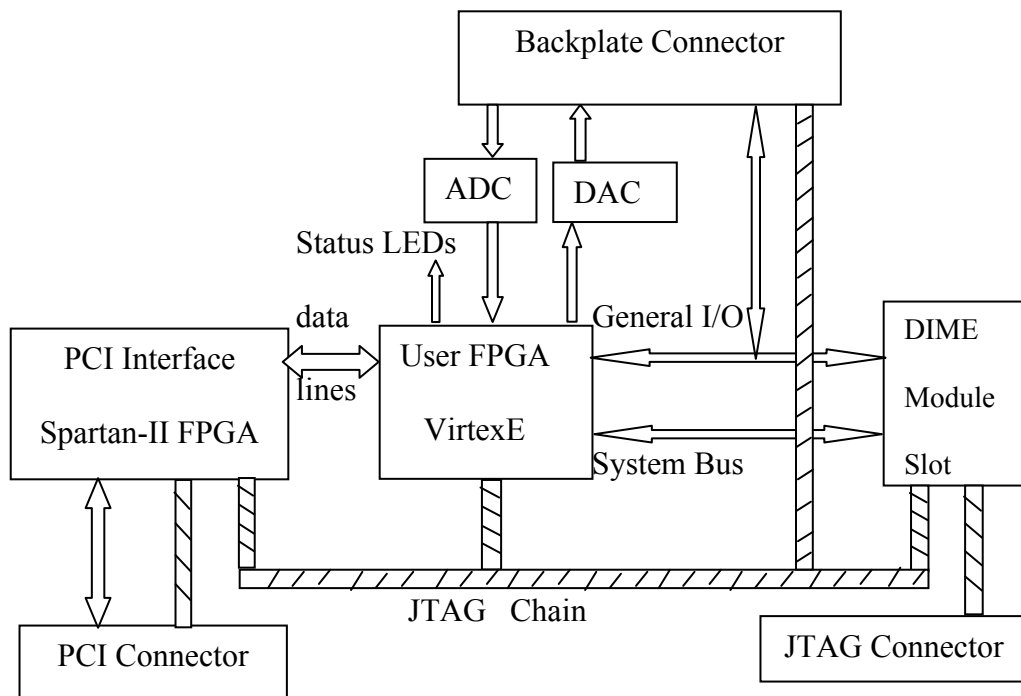


Figure A-1 Strathnuey FPGA board architecture

## ABSTRACT

Jing Pang. Ph.D. August 2003

Electrical Engineering and Computer Science

Direct Global Positioning System P-Code Acquisition FPGA Prototyping (157 pp.)

Director of Dissertation: Professor Janusz Starzyk

Conventional P-code acquisition relies on the hand over from an acquired C/A code because the amount of search from a C/A code is finite and bounded. When the C/A code is not available, direct acquisition of P(Y)-code is the only option available. The GPS P(Y)-code offers improved cross-correlation property and higher precision than C/A code. The direct P(Y)-code acquisition improves the robustness of the GPS receiver. The design techniques used for direct P(Y)-code acquisition can also be applied to the acquisition of pulsed pseudolites that transmit wideband (shifted P-code) signals.

In order to achieve a fast acquisition code phase search, the direct average method was proposed. Both of them use a small size FFT and IFFT to facilitate the hardware design in FPGA. Due to averaging, the signal detection probability is decreased. Both methods require high carrier to noise ratio. The code phase search of 10 s signals only needs around 3 s.

A group of mathematical equations were presented to initialize the different registers and counters in the P-Code generator, which was capable to start P-Code generation at any time of a week. This is very useful when there is approximate time and satellite location information.

The synthesizable design direct GPS P-Code acquisition design was downloaded to the Xilinx VirtexE chip on the Nallatech FPGA board. The design results were verified. The final FPGA results showed the correct correlation peak amplitude value and code phase shift result.

Approved: \_\_\_\_\_