THE IRNSS RECEIVER TRACKING IMPLEMENTATION AND DESIGN

A Project Report

submitted by

PRANAVKUMAR SHENDE, EE13B094

in partial fulfilment of requirements
for the award of the dual degree of

BACHELOR OF TECHNOLOGY AND MASTER OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY MADRAS

MAY 2018

THESIS CERTIFICATE

This is to certify that the thesis titled **The IRNSS Receiver Tracking, Implementation and Design**, submitted by **Pranavkumar Shende, EE13B094**, to the Indian Institute of Technology Madras, for the award of the dual degree of **Bachelor of Technology and Master of Technology**, is a bona fide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Dr. V. Kamakoti
Research Guide
Professor
Dept. of Computer Science and Engineering
IIT Madras, 600036

Dr. Arunkumar D. Mahindrakar
Research Guide
Associate Professor
Dept.of Electrical Engineering
IIT Madras, 600036

Place: Chennai

Date: 10th May 2018

ACKNOWLEDGEMENTS

I have learnt a lot and really enjoyed working on this project and I would take this opportunity to express a deep sense of gratitude towards everyone who extended their support to me during the entire timeline of this project

I am highly indebted to my project guide Prof V. Kamakoti for not only privileging me with an opportunity to work under him but also his constant guidance, inspiration and contribution without which, the project wouldn't have been a success.

My thanks and appreciations also go to Mr. Arjun Menon for his mentorship in the form of selfless support and constant encouragement, for being there whenever it mattered and for cheering me up in each and every phase of the project

I also want to thank my team-mates Mr.Karthik Jayachandran and Ms.Yashodhara Tarey for making such a wonderful and a co-operative team. Their help, contribution, important suggestions and helpful discussions proved to be invaluable for the project

Lastly, I would like to thank my parents, who have always supported me in my endeavours and have made me feel that I am capable of achieving what I dream to. They have always inspired me to aim higher and I owe everything to them.

ABSTRACT

This project is focussed on implementation and design of the tracking module in an IRNSS receiver. Execution of the tracking module is the second step after the Initial Signal Acquisition of the inputs to the Digital Signal Processing block (DSP). Tracking is executed after acquisition where each block of data is processed according to the measurements made during the processing of the previous block. The Acquisition block computes the initial estimates of the essential measurements required in the data demodulation. The tracking block computes the fine estimates and continuously keeps track of the measurements made above. Output of tracking module is input to the Lock Detector which checks the reliability of the demodulated data before finally passing it to the Navigation Processor

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

IITM	Indian Institute of Technology Madras
IRNSS	Indian Regional Navigation Satellite System
DSP	Digital Signal Processing
RF	Radio Frequency
IF	Intermediate Frequency
BSV	Bluespec System Verilog
BRAM	Block Random Access Memory
HLS	High Level Synthesis
PRN	PseudoRandom Noise
NCO	Numerical Controlled Oscillator
FLL	Frequency Locked Loop
PLL	Phase Locked Loop
CORDIC	COordinate Rotation DIgital Computer

CHAPTER 1

PRODUCT OVERVIEW

1.1 Introduction

The IRNSS Correlator is a crucial part of the DSP or the Baseband Processing module of the Generic Navigation Receiver Block also known as IRNSS Receiver Block. The navigation receiver acquires and demodulates the data transmitted to it from various satellite to find the location of receiver. As shown in the generic navigation receiver block diagram in Figure 1.1 the signals from all the visible satellites are received at the antenna in the Radio Frequency (RF) front end where they are down-converted to Intermediate Frequency (IF) and also converted from analog signal to a digital signal before being passed DSP block.

The DSP block takes the Digitized IF signal as the input, makes all the necessary measurements (Code Phase and Doppler) and demodulates the navigation data present in the digitized IF signal and outputs the demodulated data to the Navigation Processor.

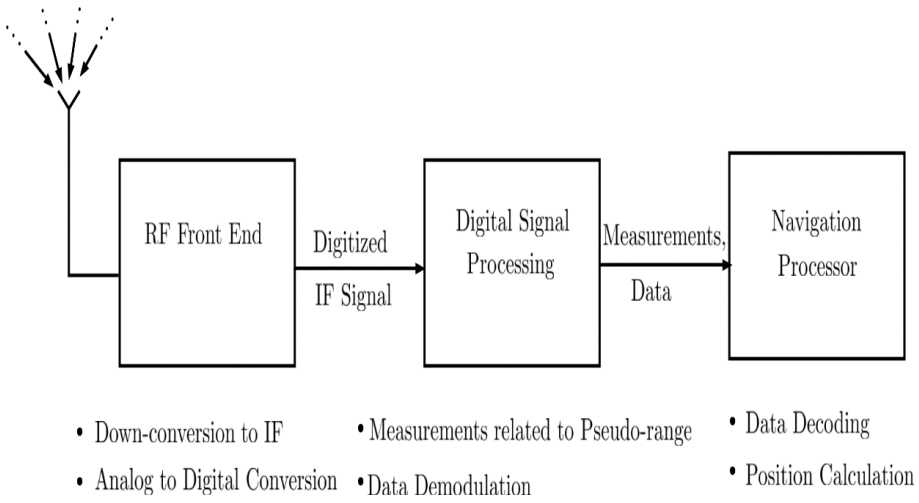

Figure 1.1: Generic Navigation Receiver Block

1.2 Implementation

The product is implemented by the Bluespec SystemVerilog (BSV) Hardware description language (HDL). BSVBSV, 2003 is chosen for the implementation of the Correlator because it supercedes all the other HDLs in various factors. Unlike most HDLs, BSV is based on circuit generation rather than merely circuit description and atomic transactional rules instead of a globally synchronous view of the world. Also, BSV is not like classical High Level Synthesis (HLS) where the source language (C/C++) has a quite different computation model (and algorithmic cost structure) from the hardware.

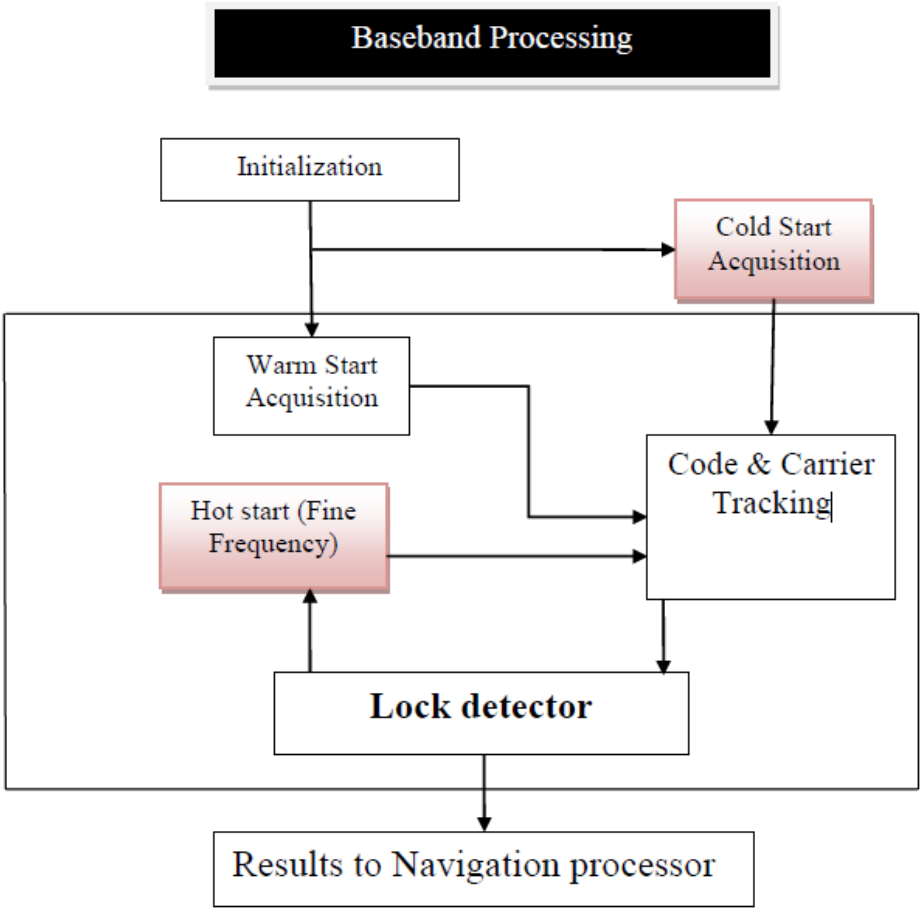

Figure 1.2: Description of Baseband Processing

1.3 Product Description

The digitized IF signal is read from a file in the Initialization part. The system can be started in 3 different ways namely cold start, warm start or hot start. The criteria for selecting the type of start is the availability of the previous measurements in the initialization. The Acquisition procedure is performed accordingly. In the beginning, the user has only 2 options namely the cold or warm start as it is assumed that previous estimates are either not available at all or only Doppler is known approximately. After acquisition, tracking is executed where each block of data is processed according to the measurements made during the processing of the previous block. The results from the tracking module are then passed on to the Lock Detector which determines the signal strength by approximately measuring Carrier to Noise Ratio (CNR). The decision on whether the lock sustains or there is requirement of a fine frequency search again (hot start) or there is no need to process for a particular satellite, is made by lock detector. The results are updated then and passed on to the navigation processor.

1.4 Product Requirements

The Product (a BSV based Software DSP) requires some initial settings to be made before starting the processing. The initial settings include signal properties (sampling frequency, IF frequency, data type, file location etc.) and system properties (selection of file reading parameters, acquisition and tracking parameter selection). A recorded or simulated signal digital IF file is the primary requirement. The tracking module requires the initial estimates acquired from the acquisition process (Code Phase and Doppler Frequency) and the digitized IF signal.

CHAPTER 2

ALGORITHM

2.1 Outline of the algorithm

A single channel of the baseband processor .the correlator receiver are shown in figure 2.1. There has to be at-least four channels that can demodulate the data for the minimum required four satellites as a minimum of four satellite co-ordinates are required. The two essential measurements required for data demodulation are measurement of the Code Phase and measurement of Doppler Frequency. After the initial estimates obtained from the acquisition block, the tracking process outputs their fine estimates and feeds it to the lock detector.

For tracking of the signal, along with Demodulated Data, the initial Code Phase and Doppler estimates are also required. Also, as these estimates and navigation data are different for the different satellites, independent processing is required for for every satellite.

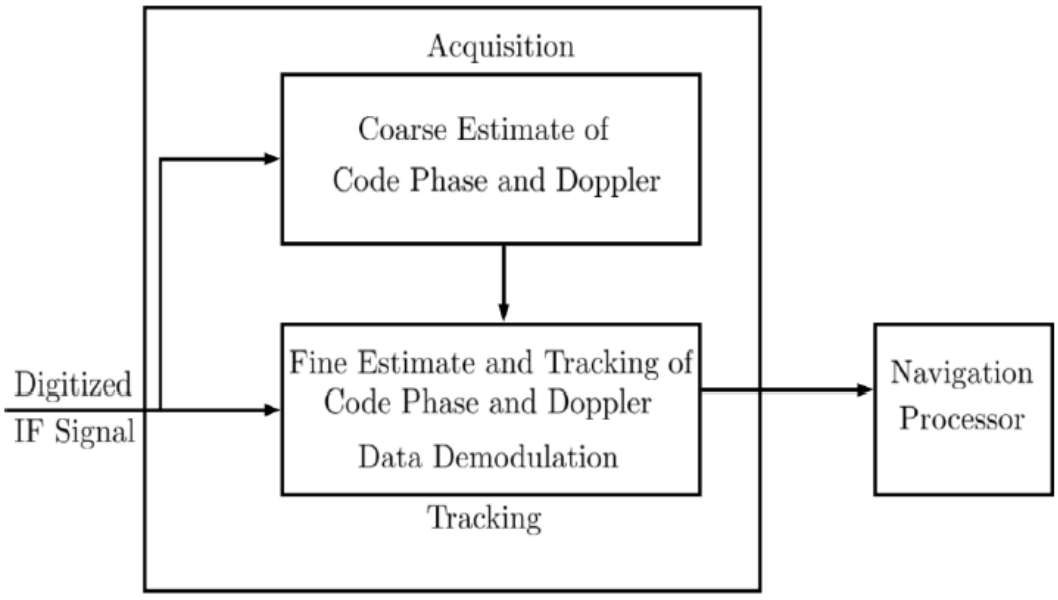

Figure 2.1: Correlator Receiver

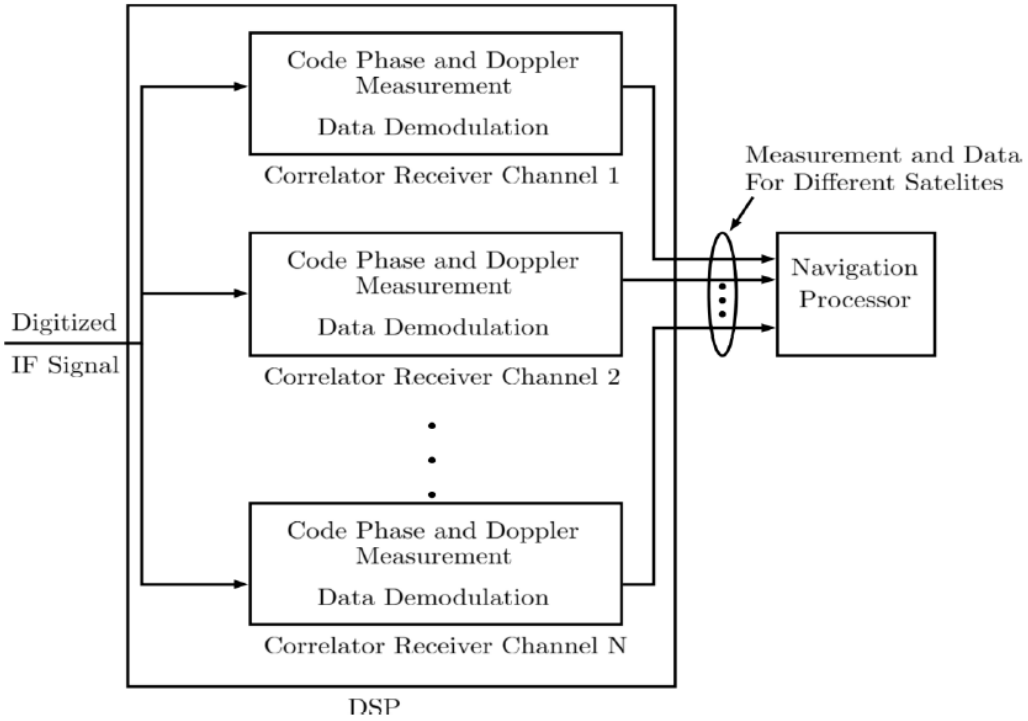

Figure 2.2: Basic outline

2.2 Structure of the input

2.2.1 Inputs to tracking and DSP

The input to the tracking module consists of the digitized IF signal and the initial estimates of the Code Phase and Doppler Frequency whereas the input to the DSP is the IF signal only.

2.2.2 Satellite Signal Structure

The digitized IF Signal or the Satellite Signal Structure(SPS) is a multiplication of the bipolar navigation data bits, bipolar PRN code chips and a high frequency carrier as shown in figure 2.3. Data rate for navigation data bits is 50 samples per second. Hence, time length of a single data bit equal to 20 ms. Chip rate for PRN code is 1.023 Mcps and the code length is 1023 chips. That implies time length of a code is 1ms. There are 20 full length codes in a single navigation data bit.

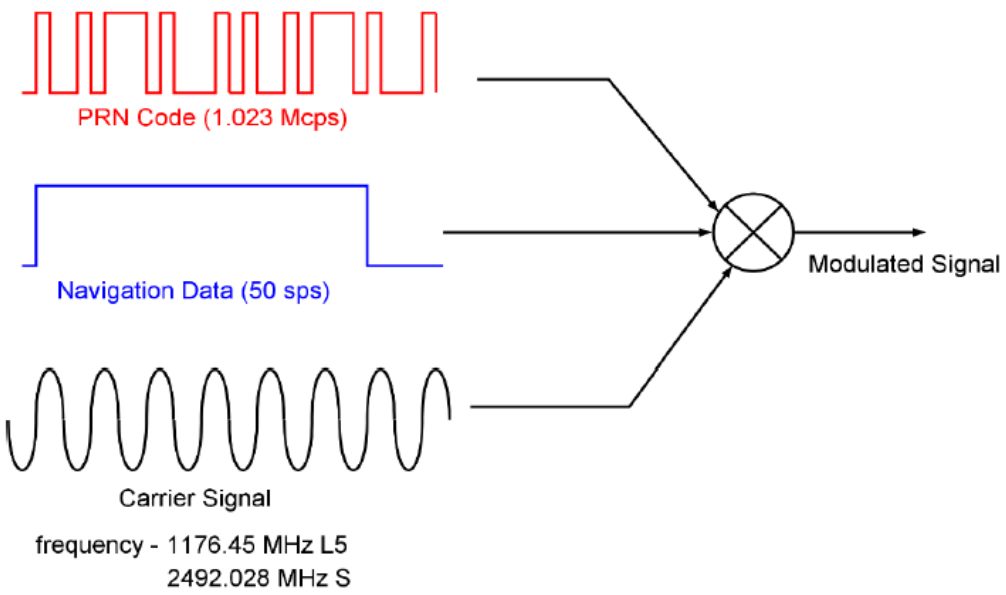

Figure 2.3: Satellite Input Structure

IRNSS SPS signals are transmitted at two carrier frequencies, one in L-Band and other in S-Band. In L-Band, L5 frequency i.e. 1176.45 MHz is used as carrier while in S-Band, 2492.028 MHz is used. Each chip transmitted at L5 modulates 1150 cycles of the carrier (1150 x 1.023 MHz = 1176.45 MHz) and each chip at S-frequency modulates 2436 cycles of the carrier (2436 x 1.023 MHz = 2492.028 MHz).

2.3 The process of Correlation

The process of Correlation is used in The Tracking and The Acquisition modules to determine the correlation values. The correlation values between two signals are calculated by sample to sample multiplication and accumulation. An example of the process of correlation is shown in figure 2.4. The replica is generated with different code phase shifts at the receiver, whenever the code phase of the generated replica and that of the incoming signal match, correlation results in a high value which is otherwise a very low value

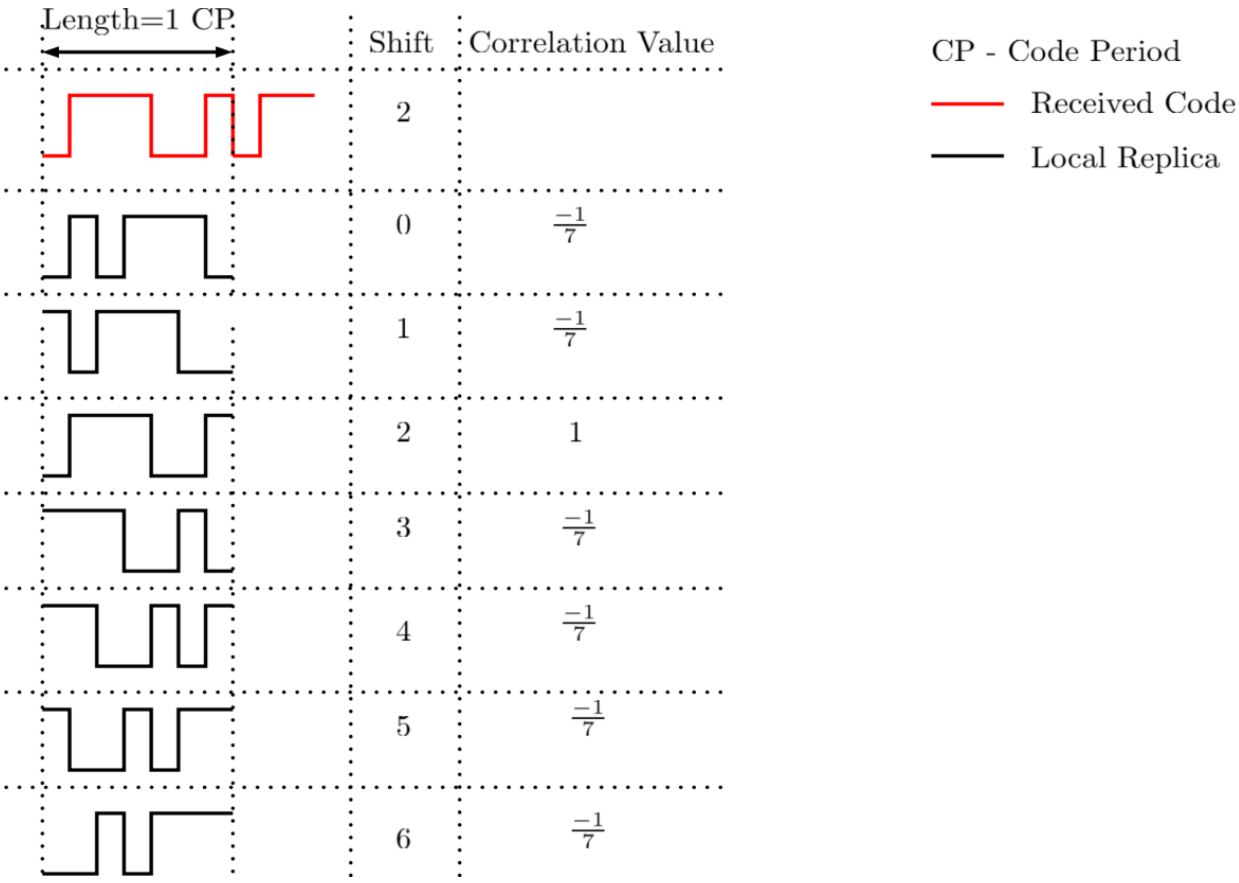

Figure 2.4: Process of correlation

CHAPTER 3

SIGNAL TRACKING

3.1 Theoretical Description

Tracking serves two purposes, to convert the coarse estimates in to fine estimates and to continuously keep track of the incoming code phase, carrier phase and Doppler frequency. A block diagram for tracking is shown in figure 3.1. Tracking module consists of two loops, the code tracking loop and carrier tracking loop. Code tracking loop estimates and tracks the code phase, while the carrier loop estimates and tracks the carrier frequency and carrier phase.

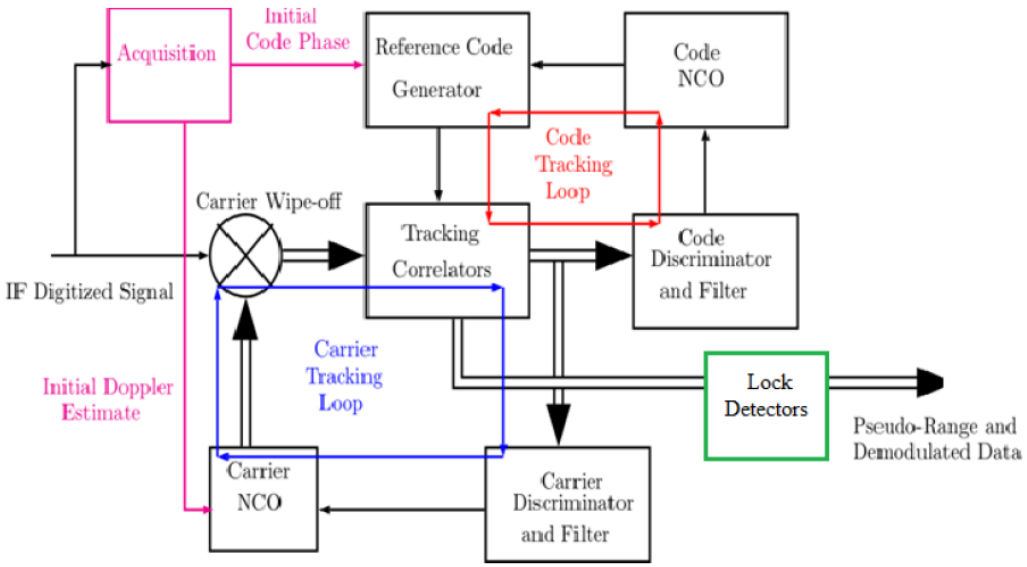

Figure 3.1: The process of tracking

3.1.1 The Carrier Tracking Loop

The function of Carrier tracking loop is to generate the appropriate frequency and phase required to wipe off the carrier from the digitized IF signal obtained as the input to obtain the in-phase component(I_P) and quadrature component(Q_P). Carrier tracking loop consists of

- Tracking Correlators
- Carrier Discriminator
- Carrier Loop Filter
- Carrier NCO

The component values are passed to the Phase and Frequency discriminators to determine the deviation between them and the values at the NCO. The output from the discriminators is passed through a first order loop filter for smoothening and the NCO is updated with the smoothened phase and frequency. This process continues until the receiver stops. The block diagram for Carrier tracking loop is shown in figure 3.2

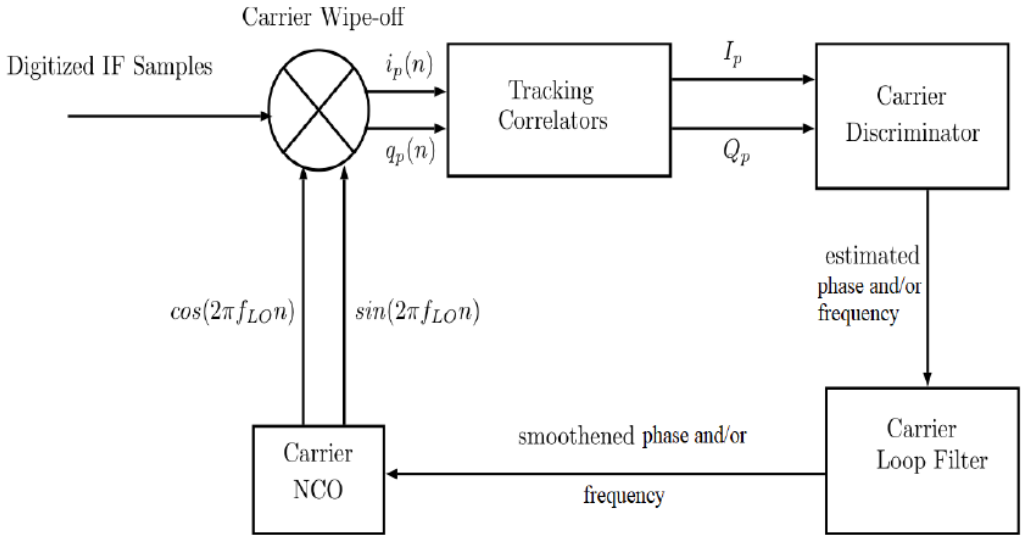

Figure 3.2: Carrier tracking loop

3.1.2 The Code Tracking Loop

The function of the code tracking loop is to determine the direction of shift of the Prompt code required to track the code phase. Code tracking loop consists of

- Tracking Correlators
- Code Discriminator
- Code Loop Filter
- Code NCO
- Reference Code Generator

Code phase is tracked using three reference codes named Early, Prompt and Late codes. Early and late codes are half chip shifted versions of the prompt code in either directions in standard receivers. The correlation values are used for finding out the direction in which the codes need to shift to track the code phase.

Figure 3.3 depicts the code tracking loop.

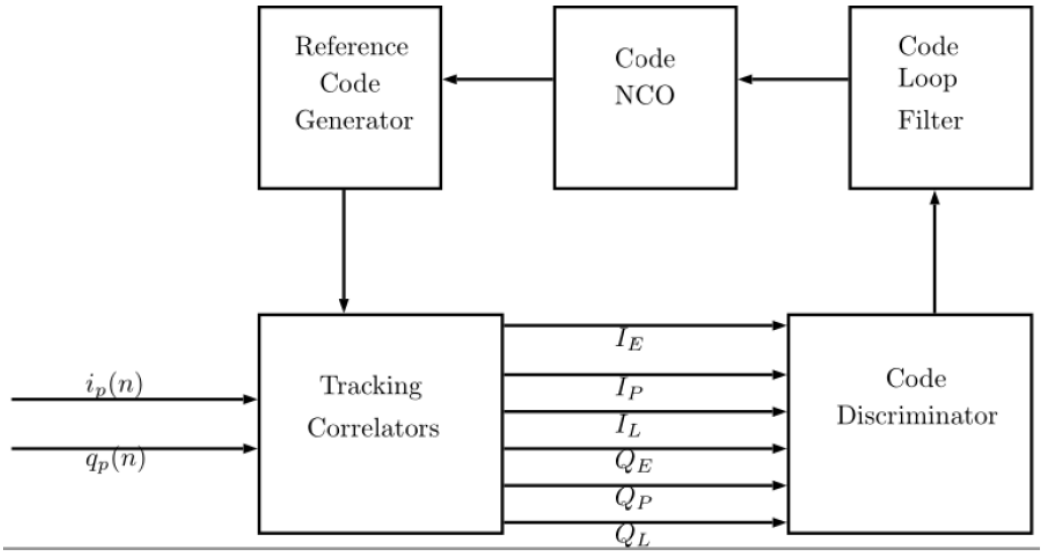

Figure 3.3: Code tracking loop

The Code Discriminator calculates the difference between Early and Late envelopes for determine the direction of shift which will lead to the correct code phase. The procedure described above is depicted in the figure.

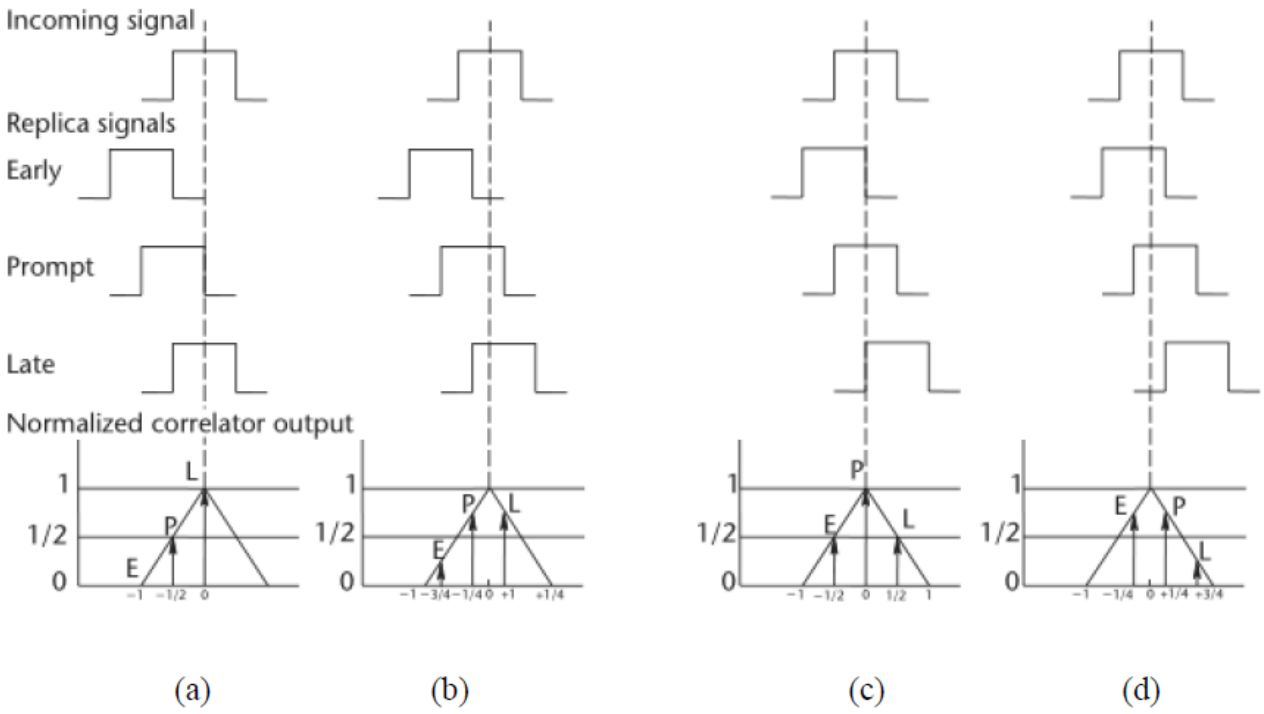

Figure 3.4: Code correlation Phases: (a) replica code 1/2 chip early,(b) replica code 1/4 chip early, (c) replica code align, (d) replica code 1/4 chip late.

From the figure 3.4 the following can be inferred,

If E-L > 0 Shift in Left

If E-L < 0 Shift in Right

If E-L $\approx$ 0 Code Phase is correct

Where E,P and L are early, prompt and late envelopes respectively.

3.2 Mathematical Description

If the initial estimates from the Acquisition block are Code Phase (τ_{in}) and initial Carrier Frequency (also Doppler estimate) (f_{in}), the signal received from carrier Numerical Controlled Oscillator (NCO) assuming initial phase of the signal is zero (i.e. θ_0 = 0) is

$$y_i = \cos(2\pi f_{in} n)$$

$$y_q = \sin(2\pi f_{in} n)$$

Carrier Discriminator is used to estimate phase and frequency using PLL and FLL.The discriminator used in the case of PLL is defined by $\theta = tan^{-1}\frac{I_P}{Q_P}$ where θ is the esti-

mated phase by phase discriminator. In case of FLL, the estimated frequency deviation is calculated as

$$\Delta f = (\theta_2 - \theta_1)/2\pi T_{loop}$$

where θ_2 - θ_1 is the difference between current and previous phase deviation estimates and T_{loop} is loop update time.

The phase deviation and frequency deviation estimates from the discriminators are fed to the loop filter (figure 3.5 which is used to smoothen the incoming phase and frequency.

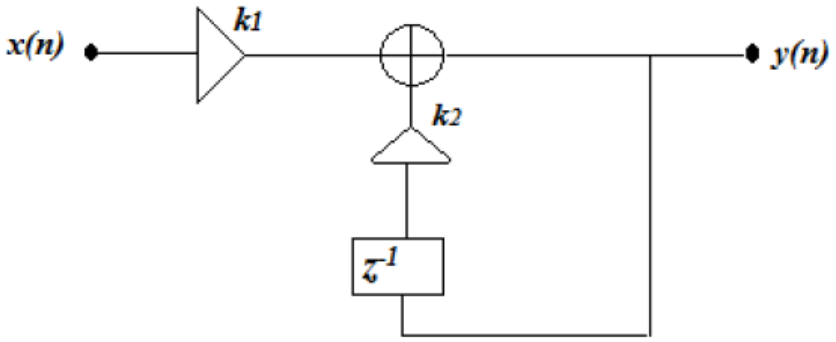

Figure 3.5: First Order Loop Filter

The time domain equation for a general first order loop is given below: Where $y(n)$ is output and $x(n)$ is input of the loop filter, k_1 and k_2 are the filter coefficients that determine the noise bandwidth and response time of the filter. A careful choice is required for the coefficients k_1 and k_2.

When the phase is given at input of loop filter the smoothen phase is obtained θ'_o and similarly for smoothen frequency $\Delta f'_o$. The smoothened phase and frequency deviations are applied to carrier NCO to generate updated values of carrier wipe off sines and cosines required. This process continues until receiver stops.

$$y_i = \cos((2\pi f_{in} + \Delta f'_o)n + \theta'_o)$$

$$y_q = \sin((2\pi f_{in} + \Delta f'_o)n + \theta'_o)$$

In Code Tracking, outputs of tracking correlators calculated using Early, Prompt and Late codes are used. Prompt code is generated using the initial code phase estimate acquired from the Acquisition block. The prompt correlator outputs are as follows

$$I_P = \frac{1}{N}\sum_{n=1}^{N} y_d[n]\cos(2\pi f_{in})C_{rep}(n-\tau)$$

$$I_Q = \frac{1}{N}\sum_{n=1}^{N} y_d[n]\sin(2\pi f_{in})C_{rep}(n-\tau)$$

$C_{rep}(n-\tau)$ represents the replica code generated at the receiver. Early and Late codes are half chip shifted versions of Prompt code generated by shifting the prompt code in either direction. The Early replica code generated would be $C_{rep}(n-\tau_{in}-\frac{el}{2})$ The Late replica code generated would be $C_{rep}(n-\tau_{in}+\frac{el}{2})$ where el is the spacing between the Early and Late codes which is fixed to 1 chip. The expression for Early and Late correlation values are as follows

$$I_E = \frac{1}{N}\sum_{n=1}^{N} y_d[n]\cos(2\pi f_{in})C_{rep}(n-\tau_{in}-\frac{el}{2})$$

$$Q_E = \frac{1}{N}\sum_{n=1}^{N} y_d[n]\sin(2\pi f_{in})C_{rep}(n-\tau_{in}-\frac{el}{2})$$

$$I_L = \frac{1}{N}\sum_{n=1}^{N} y_d[n]\cos(2\pi f_{in})C_{rep}(n-\tau_{in}+\frac{el}{2})$$

$$Q_L = \frac{1}{N}\sum_{n=1}^{N} y_d[n]\sin(2\pi f_{in})C_{rep}(n-\tau_{in}+\frac{el}{2})$$

The envelope values are calculated from the correlation values as given below

$$E = \sqrt{I_E^2 + Q_E^2}$$

$$P = \sqrt{I_E^P + Q_P^2}$$

$$L = \sqrt{I_L^2 + Q_L^2}$$

From the conditions discussed above,

If E-L > 0 Shift in Left

If E-L < 0 Shift in Right

If E-L $\approx$ 0 Code Phase is correct

CHAPTER 4

IMPLEMENTATION USING BSV

4.1 Acquisition Status

In the Tracking module, first the reference code is generated by the Reference Code Generator as shown in figure 3.3. The Acquisition status is checked and based on the status, the tracking process is started appropriately. The Acquisition status is determined by the need for warm/hot start. If there is no need for warm/hot start, the tracking process starts immediately

If warm start is required, the process of Acquisition needs to be performed again in the same way except it has to be done for a selected PRN Code and also the selected Doppler Frequency Search range is less than that in the Initial Acquisition.

If hot start is required, the reference code is generated for an already known estimate of Code Phase and the frequency space is searched finely around the known corresponding approximate frequency.

4.2 Tracking

When the warm/hot start is not required, the block of signal to be processed is selected from the recorded signal by downsampling it. Initial phase and frequency to be passed to the Carrier NCO is selected and passed to the NCO to generate the corresponding sine and cosine carriers. These carriers are mixed with the incoming signal for carrier wipe-off and generating the "In-phase" and "Quadrature" components.

The Prompt delay is determined to be delay of the codePhase. The Early delay and Late delay are computed by respectively, subtracting and adding half chip from the Prompt delay. The corresponding prompt code, early code and late code are generated from the reference codes using the delays computed above.

After the codes are generated, they are correlated with the carrier wiped-off signals in the Tracking Correlators to obtain the early, promt and late Correlation values

each of both the in-phase and quadrature components. These values are then passed to the tracking loops as shown in figure 3.2 and figure 3.3

4.2.1 Carrier Tracking Loop

In the Carrier Tracking loop, the Correlation values of the Prompt are used by the phase and frequency discriminators to calculate phase and frequency deviation between the incoming signal and the carrier generated at the NCO. These deviations are passed to the loop filter for smoothening and NCO is updated. The value of phase equals tan-inverse of inphase and quadrature of prompt.

Estimation of frequency deviation requires two values of phase as seen in the previous chapter. Hence the loop update time of Frequency Locked Loop (FLL) is twice as that of Phase Locked Loop (PLL). These values are inputs to respective first order loop filters. The equations for loop filters were described in previous chapters. The values of loop constants for PLL are $k_1 = 1$ and $k_2 = 0$. The values of loop constants for FLL are $k_1 = 0.305$ and $k_2 = 0.695$.

The value for loop filter update time of both the loops is $1ms$. In every millisecond, the filters get updated. Since the FLL requires 2 values, the FLL is updated every $2ms$ and consequently the PLL is also updated every $2ms$

4.2.2 Code Tracking Loop

The envelope values of Early, Prompt and Late codes are calculated from their corresponding In-phase and Quadrature components using the formulae described in the previous chapters. These envelopes are passed to the code discriminators to compute the code error $E - L$. The code error curve is checked for any zero-crossings to verify if the shift in the code phase is required or not.

A positive product of the code errors in the previous and current iteration indicates no zero crossing in the code error curve. In this case, if current code error is greater than zero, the code phase needs to be shifted to the left and to the right in case of negative code error. A zero code error indicates correct value of code phase. A negative product of the code errors in the previous and current iteration indicates there is a zero crossing in the code error curve. The code phase value in this case is correct as well.

The values of code phase are sent to a histogram based code phase filter for smoothening. The last 20 elements of code phase are selected and their maxima and minima are stored. The histogram is initialized of the dimensions equal to the maxima - minima + 1 and histogram indices being spread evenly with a gap of 1 from minima to maxima. Then every code phase is compared with all histogram indices. histogram value at that index is incremented if code phase equals index. The indices represent the frequency. The maximum frequency is passed as an output to the code NCO.

4.3 Programming of blocks

4.3.1 The NCO

The NCO generates appropriate sines and cosines needed for carrier wipe-off from the signal at its input. First the time vector is generated which contains the same number of indices as the number of samples. These vectors are then divided by the sampling frequency and arguments of the sines and cosines are generated for every time sample at a particular frequency. Arguments are given appropriate offset equal to the initial phase. Sines and cosines of these arguments are calculated and returned.

Generation of sines and cosines required arguments of data-type "Real" in BSV which are not synthesizable. Converting them to synthesizable variables was not very efficient. Hence the CORDIC Algorithm was used for this purpose owing to its efficient algorithms for implementing various mathematical functions and accuracy.

CORDIC, or Volder's Algorithm (Volder (1959)), is a mathematical algorithm that describes an efficiently implementable method to compute trigonometric functions and other mathematical operations purely using additions, subtractions, table lookups and shift operations. It runs iteratively on a basic set of equations obtained from the rotation matrix:

$$x_{i+1} = x_i cos\theta - y_i sin\theta \tag{4.1}$$

$$y_{i+1} = x_i sin\theta + y_i cos\theta \tag{4.2}$$

On taking a common factor of $cos\theta$, choosing $\theta = \tan^{-1}(2^{-i})$, where i is the itera-

tion count, we obtain the following final equations:

$$x_{i+1} = x_i - y_i d_i 2^{-i} \tag{4.3}$$

$$y_{i+1} = y_i + x_i d_i 2^{-i} \tag{4.4}$$

$$z_{i+1} = z_i - d_i tan^{-1}(2^{-i}) \tag{4.5}$$

where $d_i = \pm 1$, and i indicates the iteration count. Here, the sign of z_i determines d_i, which indicates the direction of rotation with respect to the previous x_i and y_i values to compute the $(i+1)^{th}$ iteration value. Based on the initial values chosen for x,y and z, various functions can be realized.

For example, in order to compute $sin\theta$ and $cos\theta$, the choice of initial values is $x_0 = 0.607253$, $y_0 = 0$, $z_0 = \theta$. The initial value of x_0 is chosen as so due to the appearance of a product of cosines of $\tan^{-1}(2^{-i})$ in the expression above. Choosing the iteration count to be 16 for a reasonable degree of accuracy, we get $K = \prod_{i=0}^{15} \cos(\tan^{-1}(2^{-i}))$. After the computation, we get $y = sin\theta$ and $x = cos\theta$.

For the case of the Doppler correlation, we require $y_i \cos\theta$ and $y_q \sin\theta$, so we run the CORDIC with a different initial value set, basically $x_0 = Ky_i$ or Ky_q, with $y_0 = 0$ and $z_0 = \theta$. The x and y components respectively give us the cosine and sine products required.

4.3.2 The Tracking Correlators

The tracking correlators perform element-wise dot product and accumulation of the replica PRN code at the receiver and the carrier wiped-off signal. A correlator block was made which computes the 4 multiplications and outputs addition of 4 of them. 100 instances of such correlators are called in parallel in a single clock cycle. This implies 400 correlation operations resulting in 100 sums. Another program completes this instantiation of the correlator blocks. This process is carried out until all the correlation values are computed.

4.3.3 Reference Code Generator

The Reference Code Generator is used for generating replica code for the selected PRN code. The PRN code is 1023 samples long. The replica code must contain the same amount of samples as the carrier wiped off input. The Reference Code Generator is used for this purpose.

The evenly distributed sample time indices are generated which contain the same number of samples as the carrier wiped-off code and the chip indices are determined for every time index. The PRN code is generated for a selected PRN which outputs a 1023 samples PRN code. This code is then sampled according to the indices generated and then down-sampled further and used for correlation.

REFERENCES

1. *BSV, 2003* (retrieved:). URL `http://www.bluespec.com/`.

2. **Volder, J. E.** (1959). The cordic trigonometric computing technique. *IRE Transactions on electronic computers*, (3), 330–334.