# Implementation of Digital Baseband Modulation for GPS Receiver

*A Project Report*

*submitted by*

**YASHODHARA TAREY**

*in partial fulfilment of the requirements*

*for the award of the degree of*


**DEPARTMENT OF ELECTRICAL ENGINEERING**
**INDIAN INSTITUTE OF TECHNOLOGY MADRAS**

**May 2018**

# THESIS CERTIFICATE

This is to certify that the thesis titled **Implementation of Digital Baseband Modulation for GPS Receiver**, submitted by **Yashodhara Tarey**, to the Indian Institute of Technology, Madras, for the award of the degree of **Master of Technology**, is a bona fide record of the research work done by her 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
IIT-Madras, 600 036

Place: Chennai

Date:May 2018

# ACKNOWLEDGEMENTS

I take this opportunity to acknowledge all the people who have supported, guided and motivated me in this project.

Foremost, I would like to express my sincere gratitude towards my guide Dr. V. Kamakoti for his immense knowledge, valuable guidance and inspiration which made this project possible.

I would like to thank Dr. R. K. Ganti and Dr. Nitin Chandrachooran for their constant support and guidance.

My heartfelt thanks to Zaid and Arjun who have helped me at each step in this project which would not have been possible without them.

I thank my fellow lab mates and friends who have constantly encouraged me to do better and alleviated stress in difficult times.

# ABSTRACT

In this work, we prototype a digital baseband modulation for GPS data using the Virtex-7 VC707 FPGA board. The prototype's modulation scheme consists of identification of Doppler phase and timing synchronization of the transmitted data from the GPS satellite. Further phase and code locking algorithms are implemented using a phased locked loop (PLL) and delay locked loop (DLL). The implementation was tested using live feed from GNSS network of satellites and proper demodulation was observed.

KEYWORDS: GPS, Acquisition, Tracking, Phase Locked Loop, Delay Locked Loop, Correlation, Doppler phase shift, Code Phase

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# CHAPTER 1

# INTRODUCTION

Global Positioning System is a satellite based navigation system which was first introduced in 1970s by the US military. Since then it has highly evolved into a system which is being used extensively for navigation around the world.

There are three segments of a GPS system-

1. Space Segment - It comprises of a constellation of 32 satellites, signals from which give information about the satellite's position.
2. Control Segment - It consists of control stations all over world which maintain the satellites in their proper orbits.
3. User Segment - This is the GPS receiver equipment which converts the RF signals to baseband signals and calculates user position.

The user location computation is done using navigation data. Navigation data is a very slowly varying signal with a frequency of 50Hz. GPS signal transmits multiple navigation data simultaneously using Binary Phase Shift Key(BPSK) carrier modulation and C/A codes.

The navigation signal is spectrum spread using C/A(coarse/acquisition). C/A codes are derived from Gold codes with 1023 chips transmitting at 1.023MHz, meaning 1 C/A code in 1ms. C/A codes strongly autocorrelate only when they are perfectly aligned with an exact replica.Each satellite uses a unique C/A code which is already known to the receiver.

The GPS signal has two frequency components:L1(Link1) and L2(Link2).The center frequency of L1 band is 1572.42MHz and of L2 band is 1227.6MHz. L1 band is used for transmission of civilian data.

Due to relative motion between satellites and receiver, a Doppler frequency is introduced in GPS signal. For a slowly moving receiver, the Doppler frequency lies in the range of -5KHz to +5KHz.

The beginning of received signal might not coincide with the beginning of C/A code. In order to obtain baseband signal, Doppler frequency and shift in the C/A code need to be determined.This is done using acquisition and tracking blocks.

# CHAPTER 2

# RF Front End

Antenna receives RF signal of frequency 1572.42MHz.The design requires digital In-phase(I) and Quadrature(Q) signals. RF front end down converts incoming signal to an intermediate frequency and digitizes it.
The RF front end chain can be represented as-

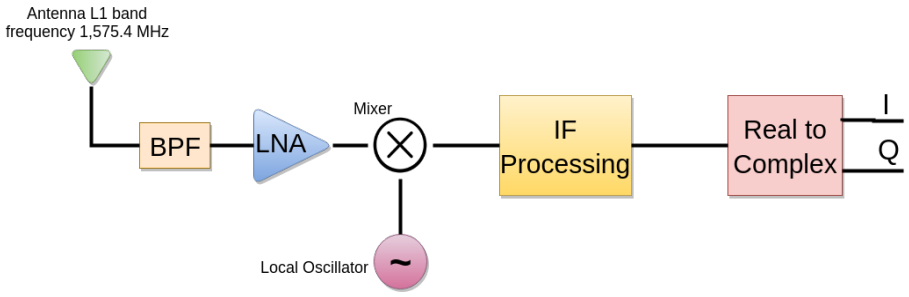


Figure 2.1: RF Front End

## 2.1 Antenna

The antenna used in this design is model no TTSLNA001 from Thiagarajar Telekom Solutions.This antenna operates at L1, L5 and S bands of frequency. It is a high performance antenna with technique for multipath signal suppression making it suitable for GPS signals.

Its frequency range is 1575±10MHz for L1 band, 1176±12 MHz for L5 band and 2491±8.5MHz for S band. It is mountable on M18 threaded pole and its operating temperature lies from -30° to 80°C.

It is an active antenna and requires a DC supply of 5V. A bias-tee is used for this purpose. It has three ends: RF, RF+DC and DC. The RF+DC end is connected to analog front-end, DC end is connected to power supply and RF end is connected to antenna. Thus, the antenna cable is used to inject power to antenna as well as to receive signals.

## 2.2 Filter

Filter is the first component in RF chain. A bandpass filter is used to provide additional frequency selectivity, such that any high-power out of band signals are eliminated. Few of the parameters used to characterize a filter include: insertion loss and bandwidth. Insertion loss is the attenuation of desired frequency component and should be as low as possible. Typical bandwidth specification used for filters is 3dB.

## 2.3 Amplifier

Amplifier is needed to increase signal magnitude to practical levels for analog to digital conversion. However a byproduct is amplification of noise levels along the signal. Basic parameters used to describe an amplifier are: gain, specified frequency range and noise figure. Noise figure indicates the amount of noise being added to the signal. Amplifier is an active component and needs power supply.

## 2.4 Local Oscillator

A local oscillator or mixer translates the input 1575.42MHz RF carrier to a lower intermediate frequency(IF) while preserving the modulated signal structure. This is done to bring the signal within operable ranges for analog to digital conversion. The frequency conversion to IF allows higher frequency selectivity with less costly/complex components.

## 2.5 Analog to Digital Converter

ADC is the final component in the RF front end chain. The key parameters to consider in an ADC are:resolution, maximum sampling frequency, analog input bandwidth, and analog input range. GPS being a CDMA signal, requires very little dynamic range from the sampled signal. The maximum frequency should accommodate the bandwidth of desired signal.

## 2.6 AD9361 Fmcomms3 Evaluation Board

In this design, AD9361 Fmcomms-3 evaluation board is used for RF front end processing. Its a high performance, programmable RF Transceiver. The receiver local oscillator operates from 47MHz to 6GHz range making it suitable for receiving L1 frequency band signals. Each receiver subsystem includes Automatic Gain Control, DC offset correction and digital filtering. Two ADCs per channel digitize the received I and Q signals and pass them through FIR filters to produce 12-bit output signal at the appropriate sample rate.

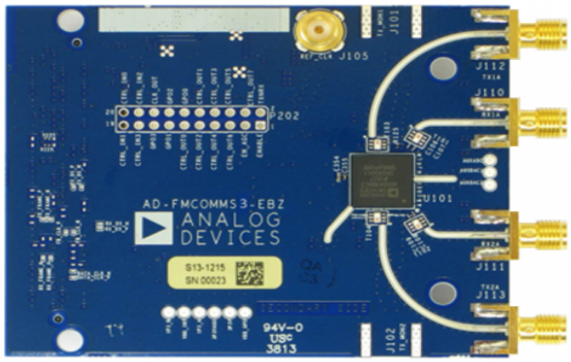


Figure 2.2: Fmcomms3 Evaluation Board

- RF 2 x 2 transceiver with integrated 12-bit DACs and ADCs TX band: 47 MHz to 6.0 GHz
- RX band: 70 MHz to 6.0 GHz
- Supports TDD and FDD operation
- Superior receiver sensitivity with a noise figure of 2 dB at 800 MHz LO
- RX gain control
- Real-time monitor and control signals for manual gain

### Clock Inputs

The AD9361 reference clock that can be provided by two different sources:a dedicated crystal with a frequency between 19 MHz and 50 MHz connected between the XTALP and XTALN pins or through an external oscillator/clock distribution device to the XTALN pin. This reference clock is used to supply the synthesizer blocks that generate all data clocks, sample clocks, and local oscillators inside the device.

### Control/Monitor

AD9361 allows real-time control via dedicated pins. These pins are connected to FPGA using FMC connector. The parameters which can be programmed include:gain, synchronization, state machine etc. It also allows real-time monitoring of some signals via another dedicated set of pins. Please refer the data sheet for more details.

### HDL Design

This project uses a reference design provided by Analog Devices for interfacing fmcomms-3 board with FPGA board, making control of parameters and obtaining digital I and Q data possible. The reference design is processor based embedded system. The functional block diagram is given in Figure-2.3. The device interface is a self-

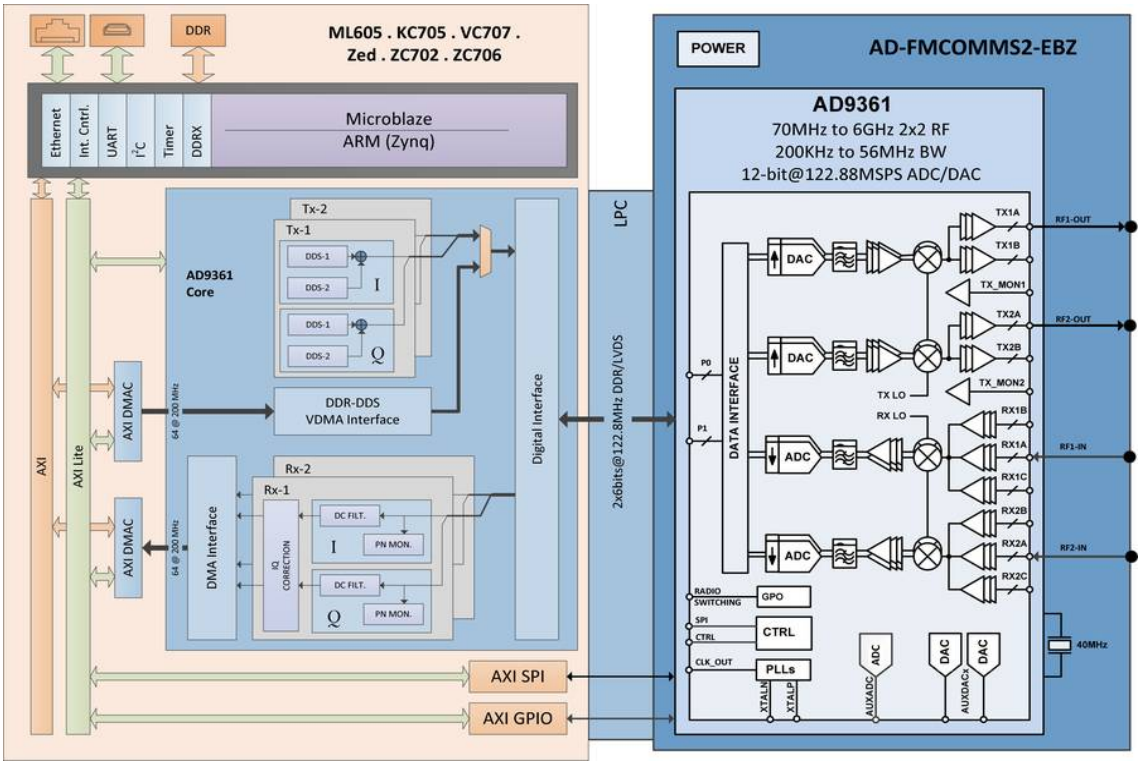


Figure 2.3: Reference HDL Design

contained peripheral which could be added as an IP in design.An AXI-lite interface is used to program core. The received data path includes a DMA interface.

An API is also provided by Analog Devices to interact with the AD9361 on systems without OS and provides all the necessary functions to control it.

In this project, a no-OS design was implemented for simplicity and easy inter platform porting.

A short description of some of the functions provided in the API:

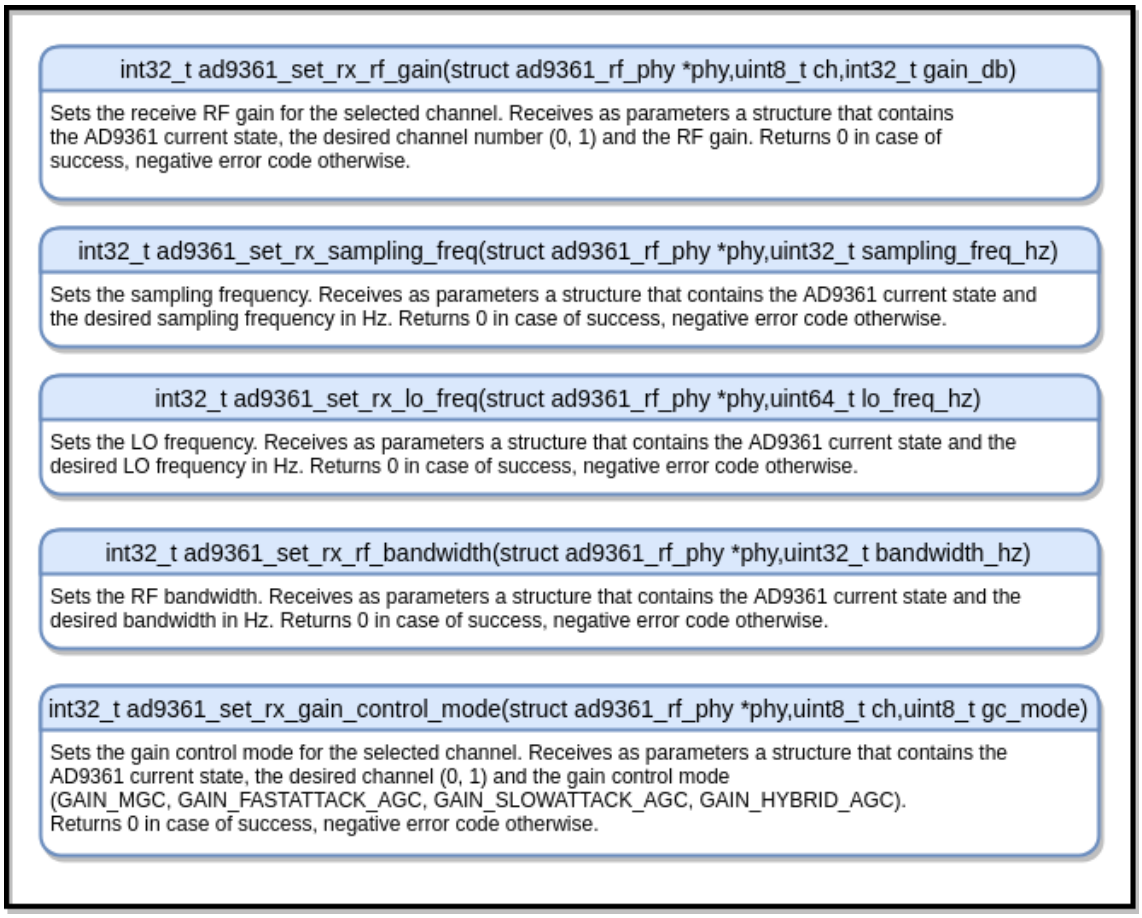


Figure 2.4: No-OS API functions

# CHAPTER 3

# Acquisition

Acquisition is done to determine which all satellites are visible to the user. If ascertained a satellite is visible, following parameters must be found:

- Doppler Frequency: Caused due to relative motion between satellite and user
- Code phase: Denotes the point in data block where the C/A code begins

There are three standard algorithms of acquisition-

1. Serial Search Acquisition
2. Parallel Frequency Space Search Acquisition
3. Parallel Code Phase Search Acquisition

### Serial Search Acquisition

This algorithm iteratively searches for a Doppler frequency, code phase pair which gives maximum correlation for a particular satellite. The incoming signal is multiplied with locally generated frequency. PRN codes are also locally generated for different satellites. Shifted versions of these codes are correlated with frequency removed signals.These correlated signals are squared and added to find correlation magnitude. Con-

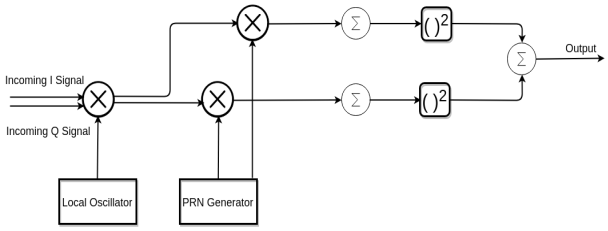


Figure 3.1: Serial Search Acquisition

sidering sampling frequency as 4MHz, 4000 samples of input data will be obtained in 1ms. PRN code will be up-sampled to 4000 bits in 1ms. Doppler frequency lies in -5KHz to +5KHz considering search step of 50Hz, the number of combinations will be-

$$4000\left(\frac{10,000}{50}\right) = 800000$$

Implementation of serial search acquisition is straightforward. However, the number of combinations to be searched for is very large and hence, it is time consuming.

**Parallel Frequency Space Search Acquisition**

This algorithm is based in frequency domain and used FFT to perform transformation from time domain to frequency domain.

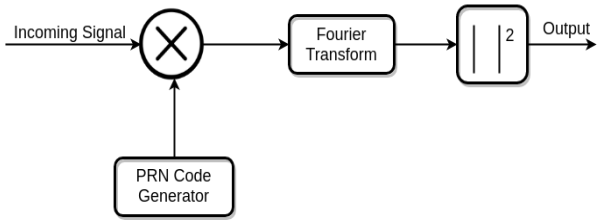


Figure 3.2: Parallel Frequency Space Search

When incoming signal is multiplied with a perfectly aligned PRN code, the result is a continuous wave signal. Fourier transform of this wave will show a peak at the IF plus Doppler frequency. Thus this algorithm parallelizes frequency search.

**Parallel Code Phase Search Acquisition**

As seen in the above calculation of number of combinations, the number of search steps in code phase is much larger as compared to frequency. In this algorithm, instead of parallel search of frequency, parallel search of code phase is performed.

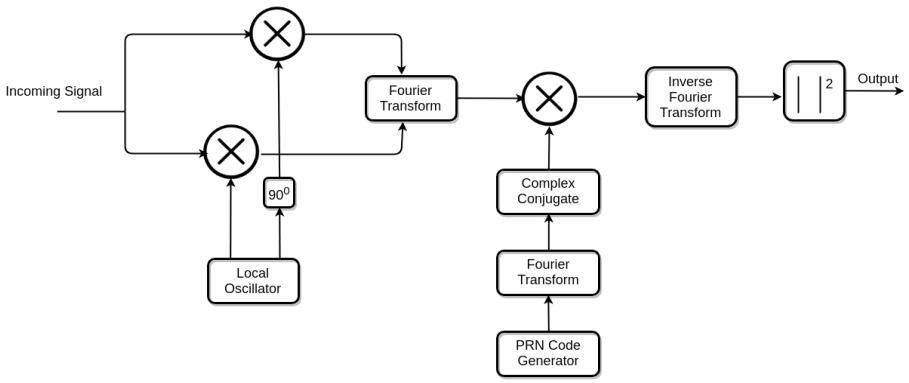


Figure 3.3: Parallel Code Phase Space Search

The Fourier transform of the input is multiplied with the Fourier transform of the PRN code. The result is transformed into time domain by an inverse FFT.If a peak is present in the correlation, the index of this peak marks the PRN code phase of the incoming signal. Thus this algorithm parallelizes code search.

In this design serial search algorithm is implemented. Incoming signal from a satellite k can be roughly modelled as -

$$s^k(t) = \sqrt{2P_c}C^k(t - t_k)D^k(tt_k)exp(j2\pi f_{L1}t)$$

where $P_c$ is the power of the signal,$C^k$ is the C/A code and $D^k$ is the navigation data sequence.The signal from ADC can be modelled as-

$$s^k(n) = \sqrt{2P_c}C^k(n - n_i)D^k(n - n_i)exp(jwn)$$

After this signal is passed through CORDIC rotors, the signal becomes-

$$s^k(n) = \sqrt{2P_c}C^k(n - n_i)D^k(n - n_i)exp(j(w - w_on)$$

where $w_o$ is the frequency being searched for. This signal is correlated with all shifted versions of PRN sequence of k satellite. The output of a correlator with a shift of j in PRN can be given as-

$$C_j = \sum \sqrt{2P_c}C^k(n - n_i)C^k(n - n_j)D^k(n - n_i)exp(j(w - w_on)$$

Due to property of PRN codes, this sum will be maximum when $n_i$ is equal to $n_j$.

The number of computations to be performed in this algorithm depends on the sampling frequency, range of Doppler frequency and frequency search step.

Here, sampling frequency is taken as 4MHz so in 1 ms will contain 4000 samples. 1 ms is of importance because C/A code is 1 ms long. It should be enough to do acquisition on 1 ms long data block, but there could a navigation data transition in that duration, which is why two consecutive data blocks of 1 ms are considered. Out of the two, only one block will have a transition. The maximum Doppler frequency shift for a slowly moving receiver is in the range of -5KHz to +5KHz and the number of frequency bins searched for are 208, making each search step 48Hz.

The schematic of complete complete acquisition block is gives in Figure-3.4

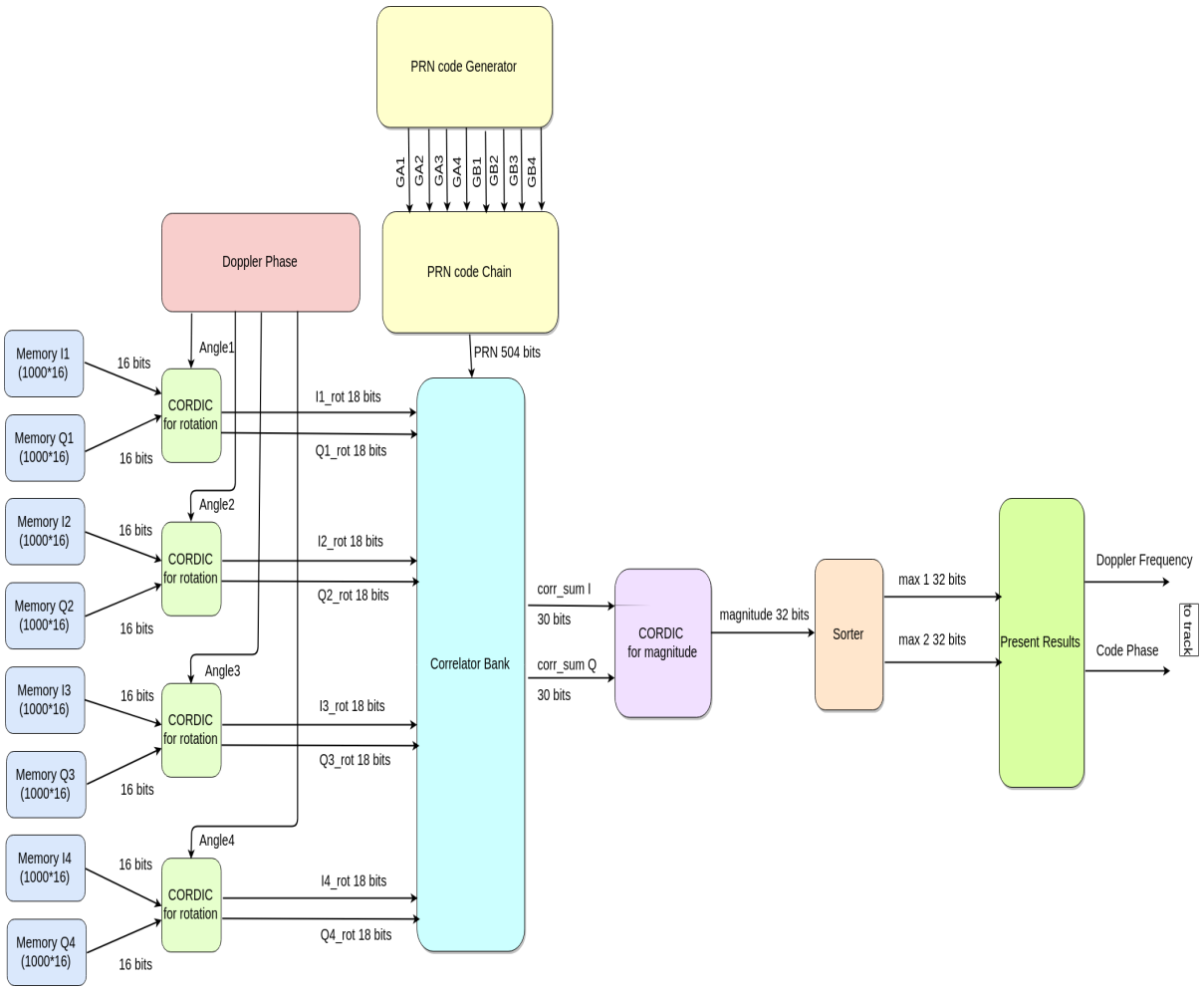


Figure 3.4: Acquisition Block Diagram

## 3.1 CORDIC

CORDIC or COordinate Rotation DIgital Computer is an efficient algorithm to compute trigonometric values in digital domain. It is based on simple look-up-table, add/subtract and shift operations. In this design, CORDIC is used to rotate the incoming signal through different pre-stored angles. These angles correspond to the various Doppler frequency shifts to be searched for.

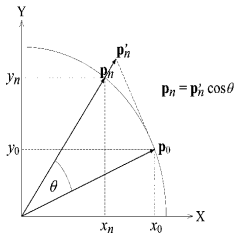


Figure 3.5: Vector Rotation

The rotation of a two-dimensional vector P through an angle $\alpha$ can be given as $P_\alpha$=$PR_\alpha$, where $R_\alpha$ is a rotation matrix-

$$R_\alpha = \begin{bmatrix} cos\alpha & -sin\alpha \\ sin\alpha & cos\alpha \end{bmatrix}$$

which can also be written as-

$$R_\alpha = [(1 + tan^2\alpha)^{\frac{-1}{2}}] \begin{bmatrix} 1 & -tan\alpha \\ tan\alpha & 1 \end{bmatrix}$$

The sum $(1 + tan^2\alpha)^{\frac{-1}{2}}$ can be approximated to 1.647. Rotation through any angle can be broken into iterations of rotation through predefined angles $\alpha_i = arctan(2^{-i})$. Thus rotation equations can be given as-

$$x_{i+1} = x_i - \sigma_i.y_i.2^{-i} \tag{3.1}$$

$$y_{i+1} = y_i + \sigma_i.x_i.2^{-i} \tag{3.2}$$

$$z_{i+1} = z_i - \sigma_i.\alpha_i \tag{3.3}$$

This mode of operation of CORDIC is called as rotation mode, there is another mode called as vectoring mode.In vectoring mode, given a vector CORDIC finds its angle of rotation from x-axis and magnitude of $\sqrt{x^2 + y^2}$. Vectoring mode of CORDIC is used in design later. The equations for vectoring mode remain the same as rotation mode. The value of $\sigma$ given in above equations is calculated as-

$$\sigma_i = \begin{cases} sign(z_i), & \text{if rotation mode.} \\ -sign(y_i), & \text{vectoring mode.} \end{cases} \tag{3.4}$$

The CORDIC module used in this design computes 16 iteration in all, 4 iterations each cycle. So it takes 4 cycles for completing all iterations. In addition, module is pipelined, so new input can be fed each cycle. For this, computed values after each 4 cycles are stored in registers. Latency is 5 cycles.

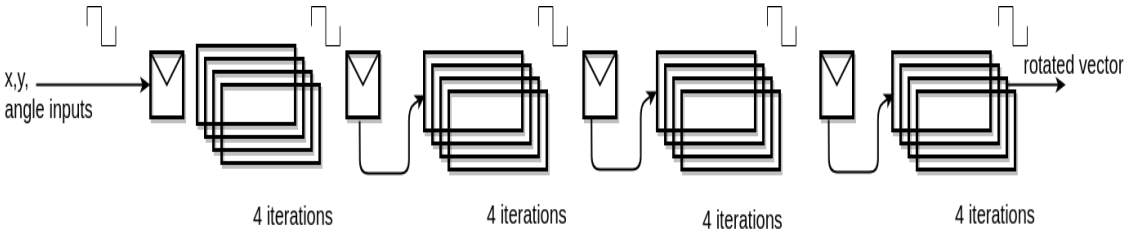


Figure 3.6: Pipelining in CORDIC

## 3.2 Correlator

Correlation in this design can be considered as a MAC(Multiply and Accumulate) unit. In each cycle, the incoming samples are multiplied with PRN data bits and the sum is accumulated until all 4000 samples are over.

The number of computations required for a satellite are equal to 208*4000, 208 frequencies and 4000 samples. Correlator takes rotated data samples and PRN bits as input. If the PRN bit is 1, sample is added and if it is 0, sample is subtracted from the overall sum. This is done for 4000 samples. Each correlator will have different shifted version of PRN code.

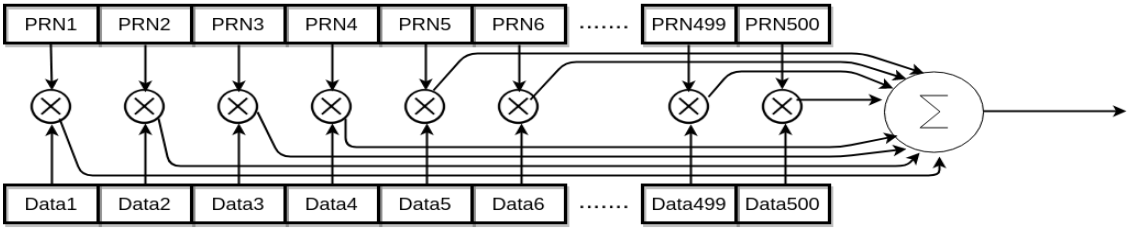


Figure 3.7: Correlation over one iteration

To speed up the process, 500 correlators work in parallel and each correlator does 4 sums per cycle. Therefore, it takes 1000 cycles to sum 4000 samples. Each 1000 cycle is known as an iteration. In one iteration, 500 shifts of PRN codes are covered. 8 iterations cover all the shifts.

## 3.3 PRN Code Generation

PRN(Pseudo Random Numbers) are used as a part of CDMA scheme in GPS. PRN code is statistically random and doesn't repeat itself at any point in its sequence, thus has very good auto-correlation properties. Each satellite has a different code to ensure near 0 cross-correlation.

PRN codes also called as Gold codes as generated using a pair of shift registers with feedback. The length of shift registers is 10 bits and they are right shifted after feedback calculation. Below diagram explains it-

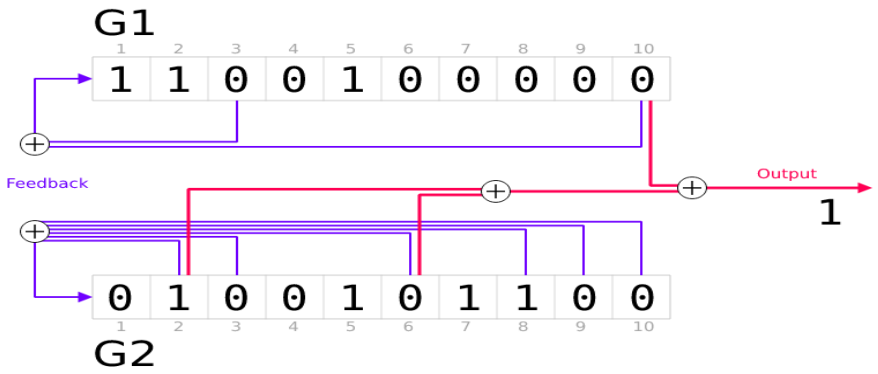


Figure 3.8: PRN Code Shift Registers

The feedback paths given in blue and purple lines are fixed, however, the paths given in pink lines are decided by satellite number which are as given in table 1 in appendix.

In this design, initial values of G1 and G2 registers are stored for 500 correlators in a header file. The next bits of PRN codes are generated using those initial values by using shift registers as shown in Figure-3.7. This module gives out 4 bits for 500 correlators in each clock cycle.

## 3.4 PRN Code Chain

There is a pattern of how PRN bits are used by different correlators. It is shown in the Figure-3.9. Correlator 1 uses PRN bits 4 to 7 in clock cycle 2, which were already used by correlator 5 in clock cycle 1. Similarly, bits 5 to 8 are used in clock cycle 2 by correlator 2 which were already used by correlator 6 in clock cycle 1. These values are stored in registers and reused.

| | CC1 | CC2 | CC3 | CC4 | ..... | CC1000 |
|---|---|---|---|---|---|---|
| Corr1 | PRN[0:3] | PRN[4:7] | PRN[8:11] | PRN[12:15] | ..... | PRN[3996:3999] |
| Corr2 | PRN[1:4] | PRN[5:8] | PRN[9:12] | PRN[13:16] | ..... | PRN[3997:1] |
| Corr3 | PRN[2:5] | PRN[6:9] | PRN[7:13] | PRN[14:17] | ..... | PRN[3998:2] |
| Corr4 | PRN[3:6] | PRN[7:10] | PRN[8:14] | PRN[15:18] | ..... | PRN[3999:3] |
| Corr5 | PRN[4:7] | PRN[8:11] | PRN[9:15] | PRN[16:19] | ..... | PRN[3999:3] |
| . | . | . | . | . | ..... | . |
| . | . | . | . | . | ..... | . |
| Corr500 | PRN[499:502] | PRN[503:506] | PRN[507:600] | PRN[601:604] | ..... | PRN[495:498] |

Figure 3.9: PRN Code Chain

The values of register G1 and G2 can change under three cases-

1. Reset or code reset: The registers should be given initial values which are stored in a header file.
2. Strobe: Strobe denotes on-going iteration. The registers will get previously stored values as explained in PRN code chain fig(3.8).
3. Change Offset: Change offset signal denotes end of an iteration and start of next. So if correlators 1 to 500 were functioning in first iteration, 501 to 1000 will function in next. Correlator 501 will need PRN bits 500 to 503. These values were available to correlator 1 in the 125th clock cycle. This point is noted as cp_match and values of G1, G2 registers are stored for future correlators.

## 3.5 Design Optimization

The number of bits in correlation sum will be 18+$\log_2 4000$=30 as output samples from CORDIC are 18 bits long and sum is computed for 4000 such samples. A straightforward implementation of correlator would involve extending all inputs to 30 bits and adding, however a more efficient way would be adding in a tree structure as shown in Figure-3.10.

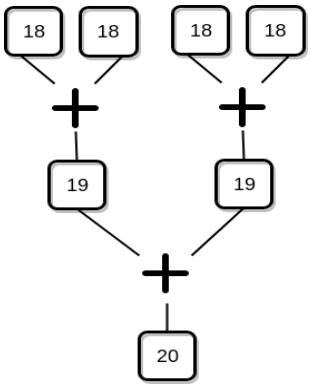


Figure 3.10: Tree Adder

Major optimization implemented is in terms of number of computations done in each cycle. In correlator design for this project, instead of using only adders, multiplexers and fewer adders were used.

The frequency of PRN codes is 1.023MHz, it is up-sampled by 4MHz because input data frequency is 4MHz as well. Since 4/1.023=3.91, each bit of PRN code will be repeated either 3 or 4 times. Therefore, number of combinations of 4 bits of PRN code will be limited. As 4 bits of PRN code are input to each correlator in one clock cycle, we look at the possible combinations of these 4 bits in Table-3.1. Here, a,b,c,d are the

| PRN Code | Input Sample Arithmetic | PRN Code | Input Sample Arithmetic |
|---|---|---|---|
| 0000 | -a-b-c-d | 1111 | a+b+c+d |
| 0001 | -a-b-c+d | 1110 | a+b+c-d |
| 0011 | -a-b+c+d | 1100 | a+b-c-d |
| 0111 | -a+b+c+d | 1000 | a-b-c-d |

Table 3.1: Possible Combinations of 4 bit PRN Code

input samples received from CORDIC. It is evident in the above table, that right side group is simply negative of the first group.

The right side PRNs can be converted into left side PRNs by XORing all bits with the MSB bit, for example, 1100 on XORing with 1 gives 0011, which is its corresponding value in left group.So a 4x1 MUX can be used to select a combination out of these 8 and negated, if needed.

Output from CORDIC block is fed to adder module, which computes all four possible combinations of sum. One of these values is selected by a 4x1 MUX, the selection lines of which are decided by 4 bits of PRN codes. There are 500 such MUX for 500 correlators. Another 2x1 MUX selects whether the sum or its negative is to be added. The selection line for this MUX is MSB of PRN code. Finally, correlator keeps adding the output of 2x1 MUX for 1000 cycles.

This optimization is based on the fact that the number of XOR operations in case of 30-bit adders is more than in case of design with multiplexers. The number of full adders in one cycle in first case can be calculated as 250*30*8=60,000, where 250 is number of correlators running in parallel, 30 is sample size and 8 is the number of samples. Each full adder is equivalent to 2 XORs, so number of XOR operations in first case equal to 1,20,000.

Whereas, in second case, the number of XOR operations will be (19*2+20)*4*3 due to adder module and (8*20*500) due to MUXs, each 4x1 MUX is equivalent to three 2x1 MUX, 20 is the sample size and 500 is the number of correlators, which makes number of XOR gates equal to 80,696.

The schematic for this design is given in Figure-3.11.

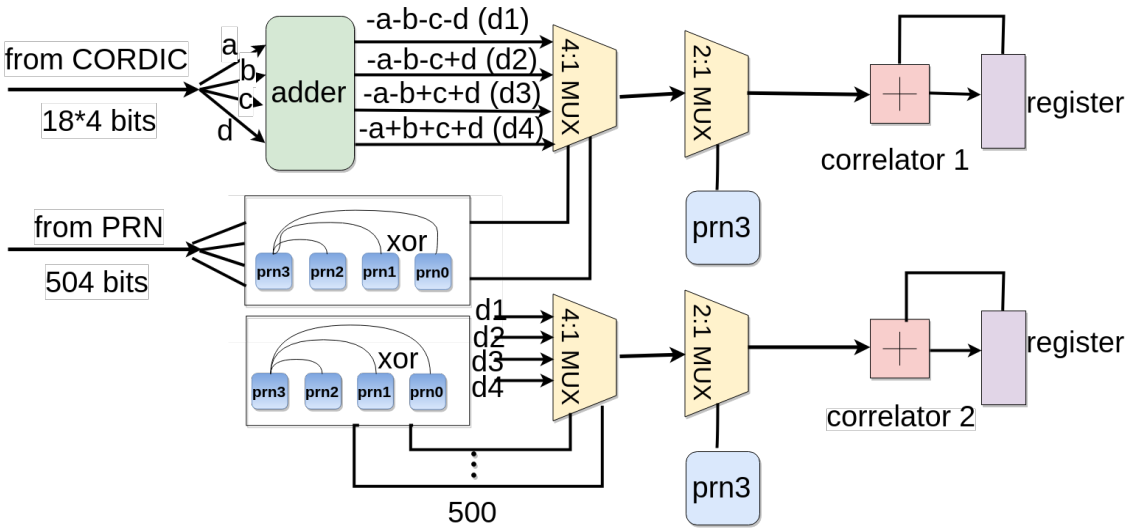


Figure 3.11: Correlator Bank Schematic

The percentage utilization summary for optimized design vs original design can be seen in Figure-3.12

| Name | Slice LUTs (303600) | Slice Registers (607200) | F7 Muxes (151800) | F8 Muxes (75900) | Slice (75900) | LUT as Logic (303600) | LUT as Memory (130800) |
|---|---|---|---|---|---|---|---|
| system_top | 226041 | 124017 | 4021 | 1416 | 68111 | 217641 | 8400 |
| dbg_hub (dbg_hub_CV) | 465 | 622 | 0 | 0 | 313 | 441 | 24 |
| i_system_wrapper (system_wrapper) | 225577 | 123395 | 4021 | 1416 | 67987 | 217201 | 8376 |
| system_i (system) | 225577 | 123395 | 4021 | 1416 | 67987 | 217201 | 8376 |
| Acq_hier (Acq_hier_imp_16IKVES) | 143432 | 42733 | 2240 | 988 | 41429 | 143077 | 355 |
| Acq_4M_250Shifts_8Adds_10kDop_0 (system_Acq_4M_250Shifts_8Adds_1... | 141937 | 40110 | 2171 | 960 | 40683 | 141931 | 6 |

| Name | Slice LUTs (303600) | Slice Registers (607200) | F7 Muxes (151800) | F8 Muxes (75900) | Slice (75900) | LUT as Logic (303600) | LUT as Memory (130800) |
|---|---|---|---|---|---|---|---|
| system_top | 217189 | 154263 | 5919 | 2128 | 66884 | 210619 | 6570 |
| dbg_hub (dbg_hub_CV) | 441 | 604 | 0 | 0 | 252 | 417 | 24 |
| i_system_wrapper (system_wrapper) | 216748 | 153659 | 5919 | 2128 | 66754 | 210202 | 6546 |
| system_i (system) | 216748 | 153659 | 5919 | 2128 | 66754 | 210202 | 6546 |
| acq_hier (acq_hier_imp_NERXZW) | 100590 | 78000 | 4540 | 1876 | 29941 | 100062 | 528 |
| Acq_IP_500_shifts_v2_0 (system_Acq_IP_500_shifts_v2_0_0) | 97951 | 73818 | 4463 | 1860 | 28878 | 97945 | 6 |

Figure 3.12: Utilization Reports of Original and Optimized Designs

# CHAPTER 4

# Tracking

The purpose of tracking is to fine tune the values of code phase and Doppler frequency shift found by acquisition module, keep track and demodulate navigation data.

Since positions of at least four satellites are needed, this design has four separate tracks working for four different satellites. Demodulation requires exact replicas of Doppler phase shift and PRN code. Feedback loops are used for obtaining exact replicas. A DLL(Delay Locked Loop) is used for code tracking and PLL(Phase Locked Loop) is used for Doppler phase tracking.

## 4.1 Phase Locked Loop

To track a carrier wave or Doppler wave, phase locked loops are used.To demodulate navigation data, an exact carrier wave is required.

A basic PLL can be represented as given in Figure-4.1

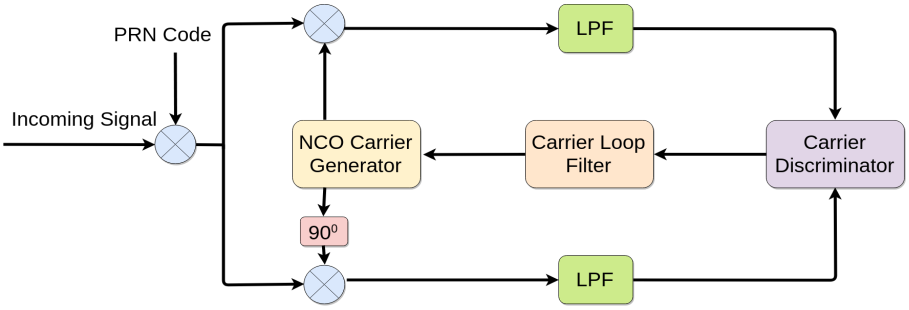


Figure 4.1: Phase Locked Loop

The incoming signal is stripped off of code phase and then carrier frequency(or Doppler frequency). To wipe off code phase prompt output from DLL is used. Carrier loop discriminator finds out the phase error on the local wave replica. The output of carrier loop discriminator is filtered using a carrier loop filter to smooth-en the output of discriminator. the filter's output is fed as an input to the NCO(Numerically Controlled Oscillator) Carrier Generator, which adjusts the frequency of carrier wave.

The problem with ordinary PLL is thet it is sensitive to 180° phase shifts. There could be 180° phase shifts in GPS data because of transitions in navigation data, hence a loop which is insensitive to 180° phase shifts needs to be used.

The loop shown in fig(4.1) is insensitive to 180° phase shifts. There are two multiplications in this loop: first between input signal and local carrier wave, and second between 90° phase shifted carrier wave and the input signal. The goal of PLL is to keep all energy in I-arm of multiplication. Feedback is used for this purpose.

The multiplication in the I arm yields the following sum:

$$D_k(n)cos(\omega_{IF}n)cos(\omega_{IF}n+\phi) = \frac{1}{2}D_k(n)cos(\phi) + \frac{1}{2}D_k(n)cos(2\omega_{IF}n+\phi)$$

where $\phi$ is the phase difference between the phase of the input signal and the phase of the local replica of the carrier phase. The multiplication in the quadrature arm gives the following:

$$D_k(n)cos(\omega_{IF}n)sin(\omega_{IF}n+\phi) = \frac{1}{2}D_k(n)sin(\phi) + \frac{1}{2}D_k(n)sin(2\omega_{IF}n+\phi)$$

The two signals are passed through a lowpass filter,the terms with the double intermediate frequency are filtered and the following two signals remain:

$$I_k = \frac{1}{2}D_k(n)cos(\phi)$$
$$Q_k = \frac{1}{2}D_k(n)sin(\phi)$$

A suitable term to feed back to the carrier phase oscillator to find phase error of the local carrier phase replica can be:

$$\frac{Q_k}{I_k} = \frac{\frac{1}{2}D_k(n)cos(\phi)}{\frac{1}{2}D_k(n)sin(\phi)}$$
$$\phi = \arctan(\frac{Q_k}{I_k})$$

Phase error is minimum when $I_k$ is maximum and $Q_k$ is minimum. Thus, this is the discriminator used for PLL.

## 4.2 Delay Locked Loop

Delay locked loop is used to keep track of the code phase in the signal. The output of a code tracking loop is a perfectly aligned replica of the code. The code tracking loop is also called as early-late tracking loop. The principle behind DLL is to correlate the input signal with three replicas of code.

A basic delay locked loop is shown in Figure-4.2

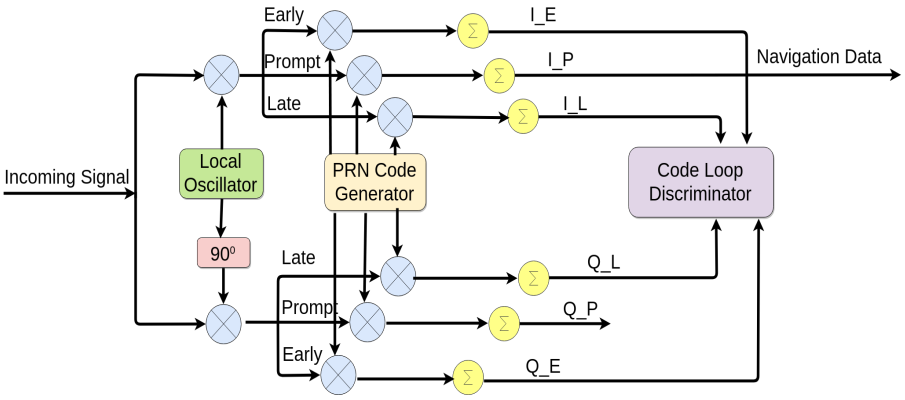


Figure 4.2: Delay Locked Loop

First carrier frequency is removed by multiplying with a perfectly aligned copy of carrier wave. After that the signal is multiplied with three copies of code:early, prompt and late. These codes have spacing of $\pm\frac{1}{2}$. The three multiplications are integrated over the period of 1ms or over the length of 4000 data samples. The output is indicative of how much each code correlates with the incoming signals. In the left side figure-4.3,

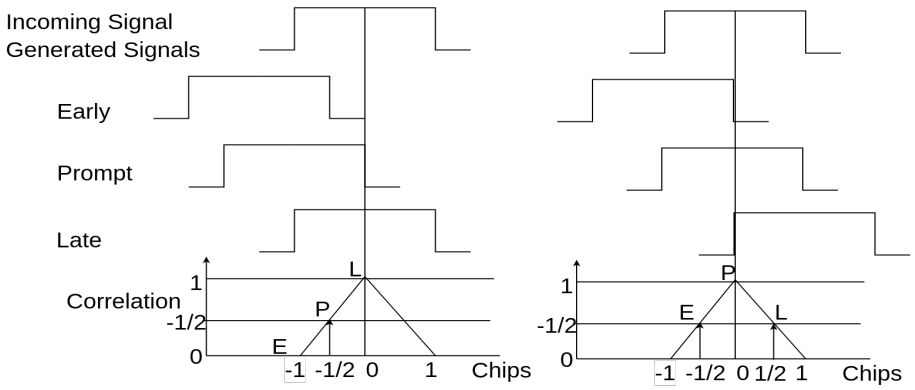


Figure 4.3: Correlation with Early, Prompt and Late Codes

highest correlation is obtained with late code and with prompt code in case of right side figure.

The DLL needs to use both I and Q arms for discriminator because the carrier wave might not be the exact replica, in which case there will be more noise, making it difficult for the DLL to lock on a code. If the local carrier wave is in phase with the input signal, all the energy will be in the in-phase arm. But if the local carrier phase drifts compared to the input signal, the energy will switch between the in-phase and the quadrature arm. If the code tracking loop performance has to be independent of the performance of the phase lock loop, the tracking loop has to use both the in-phase and quadrature arms to track the code.

The type of discriminator used for tracking in this design is non-coherent and given by-

$$\frac{(I_E^2 + Q_E^2) - (I_L^2 + Q_L^2)}{(I_E^2 + Q_E^2) + (I_L^2 + Q_L^2)}$$

In actual design, both PLL and DLL work together and there combined schematic is shown in Figure-4.4

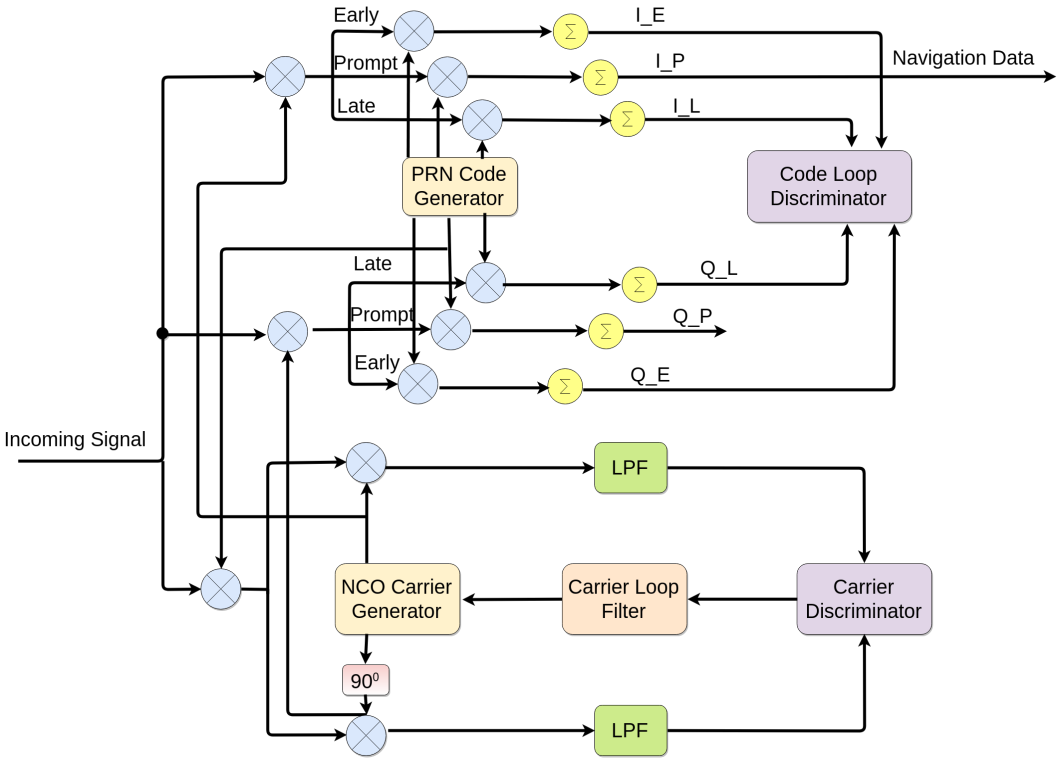


Figure 4.4: DLL and PLL Combined Loops

## 4.3 Memory Optimization

Acquisition takes 208*1000*8*2=33,28,000 clock cycles. Tracking can not start unless acquisition has completed. In the time acquisition is in process, the incoming samples should not be lost. For this memory is needed to store the data.

In the first version of design, a memory was placed inside each track module. This was done because each track starts reading data from code phase of that particular satellite. But this design could not work as BRAM utilization was over 100 percent. Instead a common memory design was devised.

As the data stored in all the tracks memories is same, it made sense to have one memory and state machine to manage reading from it for various tracks. The memory size is be finite, so track cannot keep reading forever from it. Instead, FIFOs are placed in all the tracks for storing further data.

Data continuously gets written to common memory. Once track starts reading, a counter keeps record of how many samples are remaining, when this counter reaches 4000, track stops reading from common memory and starts reading from FIFO. There are four modules related to common memory design-

1. Common memory write interface - This module is responsible for writing to common memory, finding samples remaining and start address for individual tracks.
2. Common memory read interface - This module is responsible for reading from common memory, finding out how many samples are remaining and marks stop address. Track works on the assumption that incoming data has no code phase shift, thus common memory is read from samples after code phase.
3. FIFO write interface - This module is responsible for writing to track FIFO.
4. FIFO read interface - This module is responsible for reading from track FIFO and selecting whether data has to be read from common memory or FIFO.

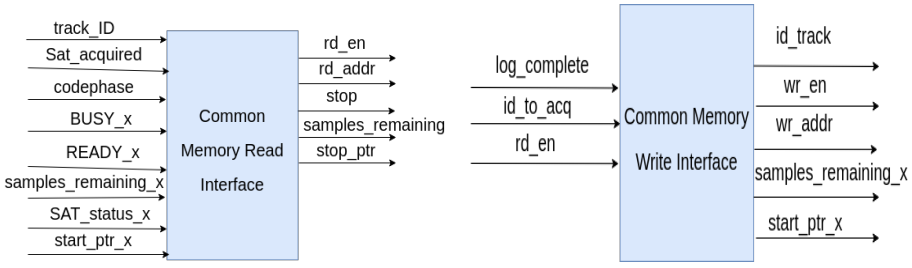


Figure 4.5: Common Memory Read and Write Interface Modules

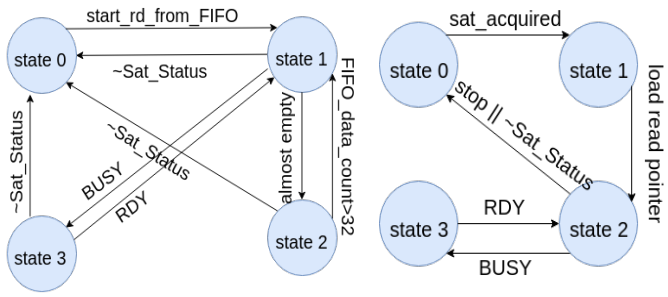


Figure 4.6: State Machines of FIFO and Memory Read Interfaces

Common memory read interface description-

1. track_ID: Denotes the track number
2. Sat_acquired: Goes high when a satellite is acquired
3. codephase: Denotes the code phase for a particular track
4. BUSY_x: Denotes that track is in middle of a computation.x denotes the track number.
5. READY_x: Denotes that track has completed computation.x denotes the track number.
6. samples_remaining_x: Denotes the number of samples remaining to be read from common memory for track'x'
7. SAT_status_x: Status of satellite 'x' becomes 1 once its computation is completed, remains 1 unless satellite is not acquired or track fails.
8. start_ptr_x: Denotes the address from which 'x' track should start reading.
9. rd_en_ID: Read enable for memory
10. rd_addr: Address from which data should be read for a track
11. stop: Denotes the time when no further data should be read from memory
12. samples_remaining: Number of samples remaining for particular track
13. stop_ptr: Address till which data should be read from memory

Common memory write interface description-

1. log_complete: Denotes completion of data logging of 8000 samples for acquisition
2. id_to_acquire: Track ID
3. rd_en: Read enable for common memory
4. id_track: Track ID registered at log_complete

5. wr_en: Write enable for common memory
6. wr_addr: Address in which data is to be written
7. samples_remaining_x: Described above
8. start_ptr_x:Described above.

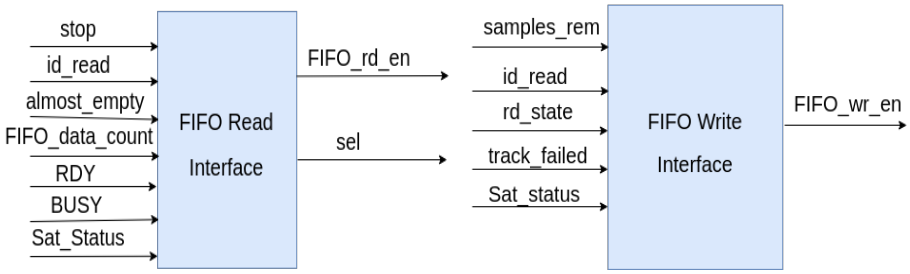


Figure 4.7: Track FIFO Read and Write Interface Modules

FIFO read interface description-

1. stop: Denotes when reading from common memory should stop
2. id_read: Track ID for which data is being read
3. almost_empty: Denotes when very few samples are remaining in FIFO. Reading should resume only after sufficient samples are written to FIFO again
4. FIFO_data_count: Count of number of samples in FIFO
5. RDY: Described above
6. BSY: Described above
7. Sat_status: Described above
8. FIFO_rd_en: Denotes when data reading from FIFO should start
9. sel: Selects whether data has to be read from common memory or FIFO. 1 means read from memory, 0 means read from FIFO.

FIFO write interface description-

1. samples_rem: Described above
2. id_read: Track ID for which data is being read
3. rd_state: Denotes the state of read state machine
4. track_failed: Goes high when track is no longer able to track
5. Sat_status: Described above
6. FIFO_wr_en: Denotes write enable for FIFO

# CHAPTER 5

# User Location Computation

The user position calculation is based on the principle of triangulation. The three-dimensional location is found using these four equations-

$$\rho_1 = \sqrt{(x_1 - x_\mu)^2 + (y_1 - y_\mu)^2 + (z_1 - z_\mu)^2} + b_\mu$$
$$\rho_2 = \sqrt{(x_2 - x_\mu)^2 + (y_2 - y_\mu)^2 + (z_2 - z_\mu)^2} + b_\mu$$
$$\rho_3 = \sqrt{(x_3 - x_\mu)^2 + (y_3 - y_\mu)^2 + (z_3 - z_\mu)^2} + b_\mu$$
$$\rho_4 = \sqrt{(x_4 - x_\mu)^4 + (y_4 - y_\mu)^2 + (z_4 - z_\mu)^2} + b_\mu$$

where ($x_\mu$,$y_\mu$,$z_\mu$) are user co-ordinates,
($x_1$,$y_1$,$z_1$) ,($x_2$,$y_2$,$z_2$), ($x_3$,$y_3$,$z_3$) and ($x_4$,$y_4$,$z_4$) are co-ordinates of four satellites,
$\rho_1$,$\rho_2$,$\rho_3$, and $\rho_4$ are pseudo-ranges and $b_\mu$ is the user clock bias error expressed in distance.
The pseudo-range can be found using-

$$\rho_i = c(t_\mu - t_{si})$$

where, $c$ is the speed of light, $t_\mu$ is the time of signal received and $t_{si}$ is the transmission time, which is received as time stamp with satellite data.

The output from the tracking loop is the in-phase arm value truncated to the values 1 and -1. One navigation bit durates 20 ms. The bit rate of the navigation data is 50 bps. The output from the tracking block comes at 1000 sps.Thus, the signal from the tracking block must be converted from 1000 sps to 50 bps. That is, 20 consecutive values must be replaced by only 1. This conversion procedure is referred to as bit synchronization.

First the time where bit transition in a sequence occurs is found, this is called as zero crossing. Consecutive bit transitions are located 20ms apart from the beginning one. After bit transitions are located, 20 samples of track output are replaced by 1.

The first task in navigation data decoding is to find preamble. The beginning of a

sub-frame is called a preamble and has the pattern 10001011 or 01110100. Since these patterns can occur elsewhere in data too, an additional check has to be performed which confirms whether the same preamble is repeated every 6s or not. 6s is the time between transmission of two consecutive sub-frames.After preamble if found, data from each sub-frame can be extracted. If the preamble is inverted, the entire sequence must be inverted.

Preamble marks the beginning of a sub-frame. Each of the sub-frames contains 300 bits divided into 10 30-bit words. So each sub-frame has 10 words and each frame consists of 5 sub-frames. The navigation message consists of 25 frames. The complete navigation message lasts for 12.5 minutes. The structure is shown in Figure-5.1.

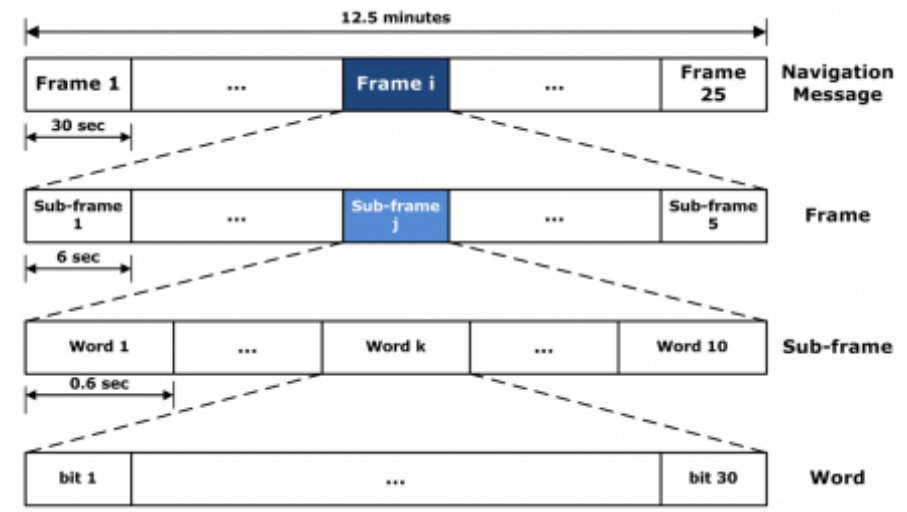


Figure 5.1: GPS Data Structure

The first two words of a sub-frame are HOW(Hand Over Word) and TLM(Telemetry).

## 5.1 Parity Check

Each word contains parity. Parity is 6-bits long. It is used to check for misinterpretation bits in the navigation data. When navigation data is received, a parity check must be performed to test if the received data is interpreted correctly.

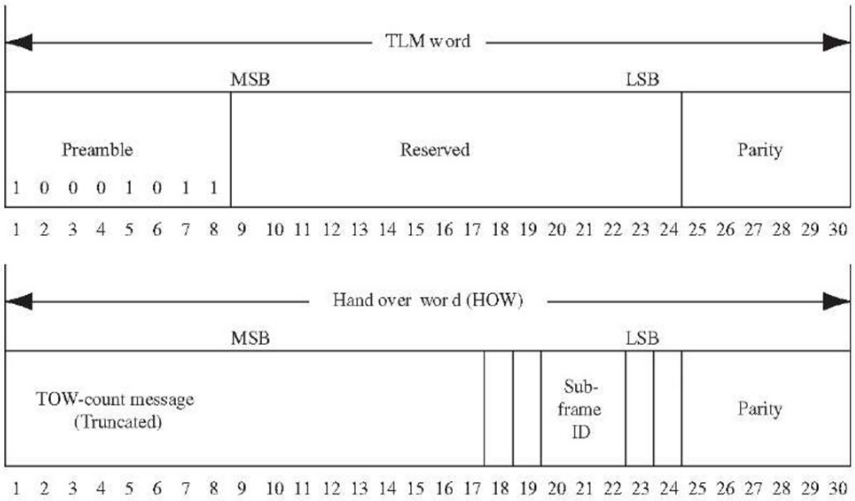


Figure 5.2: GPS Data Structure

## 5.2 Time of Transmission

To decode the navigation data sequence, scheme from ICD-GPS-200 (1991) is used, which gives details of every word. Another important parameter is to determine the time when the current sub-frame was transmitted from the GPS satellite.

The second word of each sub-frame is HOW that includes a truncated version of TOW. This number is called the z-count and it is the number of seconds that have passed since the last GPS week rollover in units of 1.5s. The rollover happens at midnight between Saturday and Sunday. The Z-count in HOW is a truncated version containing only the 17 MSBs.This truncation makes the Z-count increase in 6-s steps corresponding to the time between transmission of two consecutive navigation sub-frames.

The truncated Z-count value in the HOW corresponds to the time of transmission of the next navigation data sub-frame. Time of transmission of the current sub-frame is obtained by multiplying the truncated Z-count by 6 and subtracting 6 s from the result.

## 5.3 Remaining Parameters

The remaining parameters of navigation data like the ephemeris parameters are decoded according to ICD-GPS-200 (1991) as well. Ephemeris parameters and its decoding is listed in appendix.

# CHAPTER 6

# Shakti Processor

Shakti is a family of 6 processors which aim at breaking the barrier between academia and industry by providing open-source processors. These processors are based on the RISC-V ISA form developed by UC Berkeley which is open and patent free and has good support by OS and toolchains.

The processors are being developed at CAS Lab, IIT Madras. ISA remains standard across all flavors of processor only differing in their micro-architecture. Different variants of Shakti processors are developed to cater to a vast range of functionality. The variants will also have sub-variants, which differ from one another in terms of sets of functional blocks.

Features of C class processor include:

1. 32-bit 5 stage processor
2. Equipped with branch prediction
3. Targeted at 50-200 MHz micro-controller variants
4. Very low power, static design
5. Unified L1 cache
6. Includes hardware multiply,divide
7. Supports all integer and single precision float instructions
8. Can be interfaced with AHB or AXI-lite bus

**AHB Protocol**

The ARM Advanced Microcontroller Bus Architecture(AMBA) facilitates interfacing designs with large number of controllers and peripherals. It is open standard, on-chip interconnect for blocks on a Syastem-on-Chip(SoC) designs. It is being widely used on a range of ASIC and SoCs.

AHB is a bus protocol and it was introduced in AMBA version 2 published by the ARM company. There are two phases in a transaction in AHB bus:address phase, data

phase. There are no wait-states between the phases, only two bus cycles. AHB implements the following features required for high-performance, high frequency systems:

1. Burst transfers
2. Split transactions
3. Single-cycle bus master handover
4. Single-clock edge operation
5. Non-tristate implementation
6. Wider data bus configurations (64/128 bits).

The operation of AHB can be described as follows: the master drives the address and control signals to start a transfer. These signals contain information about the address, direction, width of the transfer. Transfers can be:

- Single
- Incrementing bursts that do not wrap at address boundaries
- Wrapping bursts that wrap at particular address boundaries

The control signals also contain information about whether the transfer forms part of a burst or not. The write data bus moves data from the master to a slave, and the read data bus moves data from a slave to the master.Every transfer consists of:address phase-address and control cycle, data phase-one or more cycles for the data. HRESP is used by the slave to indicate whether a transfer failed or succeeded.

# CHAPTER 7

# System Setup

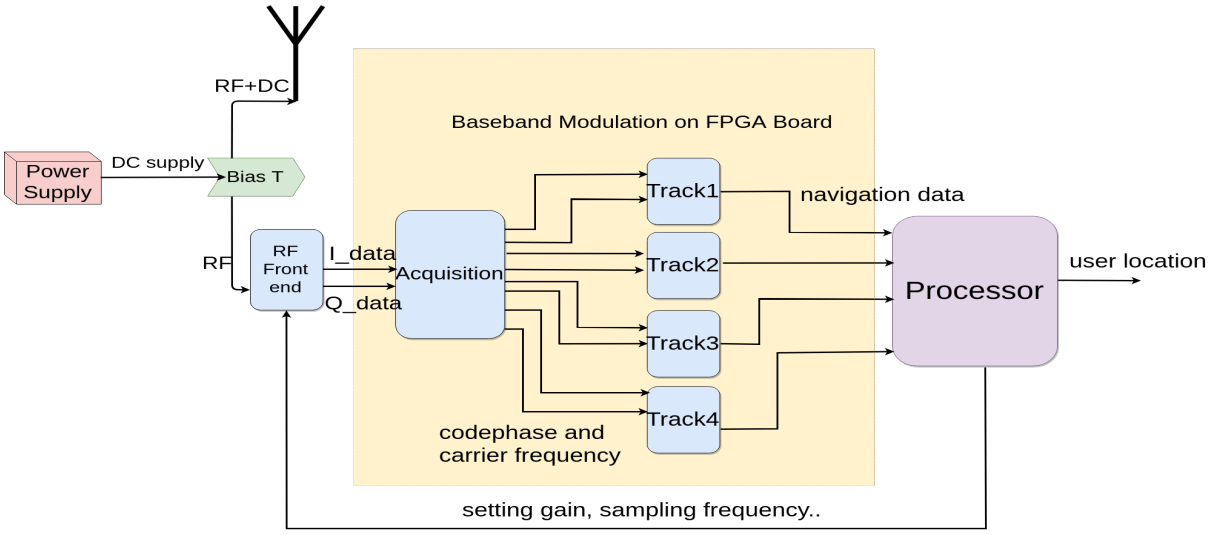


Figure 7.1: System Block Chain

A brief about various components of the system is as follows-

- Antenna : Model no TTSLNA001 from Thiagarajar Telekom Solutions is an active antenna
- DC Power Supply: GwINSTEK model number GPS-4303 is used to supply DC power to the antenna. Current limit is kept at 0.5 A and voltage supplied is 10V
- Bias Tee: Mini Circuits model number zfbt-4r2g-ft+ is a diplexer used to supply DC voltage to antenna and allows only RF signal on receiver side
- RF Front End: Board Fmcomms-3 EBZ by Analog Devices is used as a transceiver. The daughter board on it is AD9361. It is mounted on FPGA with FMC connectors
- FPGA Board: Board used for prototyping is Virtex-7 vc707 with 1 GB DDR3 memory, USB 2.0 ULPI Transceiver, 128 MB flach memory, USB JTAG through Digilent module, two FMC connectors and multiple other features
- Processor: Microblaze soft core processor with 32-bit RISC microprocessor configurations is used. It is a part of the Embedded Processor Development Kit

To setup the system, the components are connected as shown in fig(7.1). A cable from the antenna is connected to RF+DC port of bias tee using SMA connectors. Power

supply is connected to GND and DC ports on the supply side of bias tee. Lastly, RF end is connected to RX1 channel of fmcomms-3 card using SMA connectors. The fmcomms-3 card is connected to FPGA using FMC connector 1 embedded on the board.

For interfacing AD9361 with FPGA, Analog Devices has provided a reference HDL design which is used to set parameters in AD9361 and get the In-phase(I) and Quadrature(Q) phase data. The data is sampled at 4MHz and carrier removal is done in AD9361 itself.

Once bitstream is generated, the FPGA can be programmed. To run Microblaze, the soft-core processor, SDK tool is needed. First, the hardware files including bitstream are exported to project location and then Microblaze is programmed with the no-OS API files provided by Analog Devices to interact with AD9361.

The navigation bits produced from tracks are observed on ILA(Integrated Logic Analyzer).

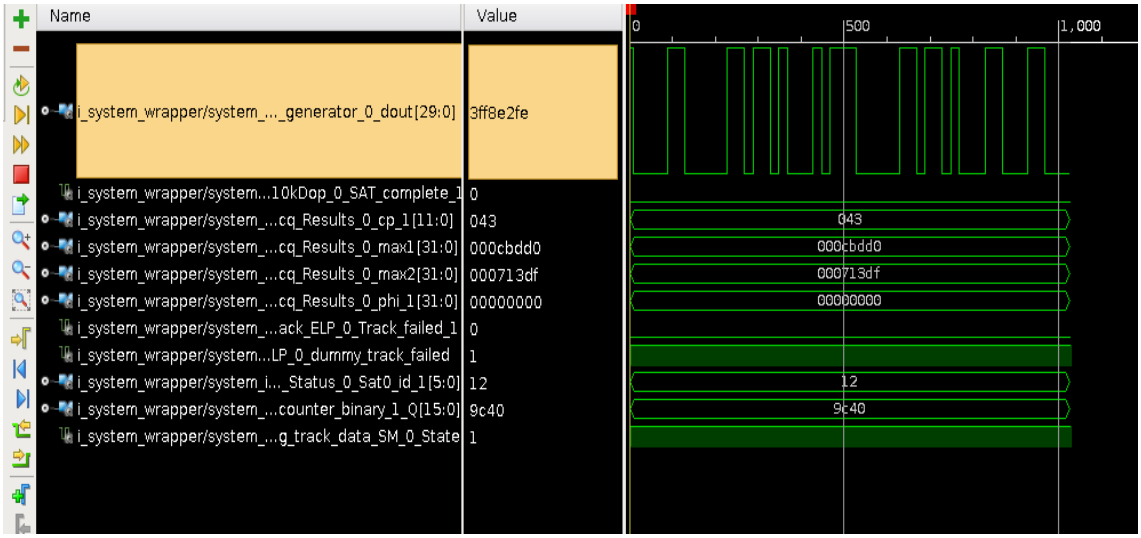


Figure 7.2: Navigation Bits Output from Track

# CHAPTER 8

# Conclusion and Future Work

Various blocks of algorithm, such as, acquisition, tracking, common memory modules were tested and verified for correct functionality through simulations and checking against standard MATLAB codes at each step. To run the simulations, huge number of data samples were logged from antenna using DDR3 memory and a DMA controller built on FPGA board. After some bug-fixing, baseband modulation was successfully achieved on live GPS data received from an antenna. Navigation bits for various satellites were obtained and observed. The acquisition algorithm was well optimized for power consumption.

Future work includes integration of Shakti processor in this design. The processor could control filter coefficients in PLL and also the discriminator spacing of DLL. The advantage of doing this is that if the signal-to-noise ratio suddenly decreases, the receiver uses a wider spacing in correlators to handle more noisy signals. In addition, acquisition algorithms using FFT could be tested for performance.

The utilization summary of design on Virtex-7 vc707 board is shown in Figure-8.1

**Frequency of operation:60MHz**

| Name | Slice LUTs (303600) | Slice Registers (607200) | F7 Muxes (151800) | F8 Muxes (75900) | Slice (75900) | LUT as Logic (303600) |
|---|---|---|---|---|---|---|
| system_i (system) | 216748 | 153659 | 5919 | 2128 | 66754 | 210202 |
| acq_hier (acq_hier_imp_NERXZW) | 100590 | 78000 | 4540 | 1876 | 29941 | 100062 |
| AD_ifc (AD_ifc_imp_G9WBTY) | 34 | 57 | 0 | 0 | 18 | 34 |
| axi_ad9361 (system_axi_ad9361_0) | 5767 | 13508 | 64 | 24 | 3580 | 5697 |
| axi_ad9361_adc_dma (system_axi_ad9361_adc_dma_0) | 484 | 560 | 2 | 0 | 199 | 484 |
| axi_cpu_interconnect (system_axi_cpu_interconnect_0) | 406 | 139 | 4 | 0 | 192 | 406 |
| axi_ddr_cntrl (system_axi_ddr_cntrl_0) | 18162 | 13225 | 485 | 64 | 6812 | 15848 |
| axi_gpio (system_axi_gpio_0) | 210 | 509 | 0 | 0 | 199 | 210 |
| axi_gpio_lcd (system_axi_gpio_lcd_0) | 87 | 137 | 0 | 0 | 49 | 87 |
| axi_iic_main (system_axi_iic_main_0) | 440 | 377 | 4 | 1 | 162 | 430 |
| axi_intc (system_axi_intc_0) | 242 | 201 | 0 | 0 | 98 | 242 |
| axi_linear_flash (system_axi_linear_flash_0) | 771 | 355 | 132 | 66 | 366 | 507 |
| axi_mem_interconnect (system_axi_mem_interconnect_0) | 6605 | 6514 | 13 | 0 | 2579 | 5929 |
| axi_spi (system_axi_spi_0) | 379 | 524 | 0 | 0 | 161 | 362 |
| axi_timer (system_axi_timer_0) | 323 | 216 | 1 | 0 | 111 | 323 |
| axi_uart (system_axi_uart_0) | 114 | 90 | 0 | 0 | 40 | 104 |
| clk_reset (clk_reset_imp_11P4Z4L) | 20 | 28 | 0 | 0 | 14 | 19 |
| common (common_imp_A3G8A8) | 3768 | 4013 | 270 | 88 | 1862 | 3273 |
| ila_0 (system_ila_0_2) | 848 | 1196 | 2 | 0 | 535 | 736 |
| ila_1 (system_ila_1_0) | 718 | 1033 | 2 | 0 | 418 | 624 |
| sys_concat_intc (system_sys_concat_intc_0) | 0 | 0 | 0 | 0 | 0 | 0 |
| sys_dlmb (system_sys_dlmb_0) | 0 | 0 | 0 | 0 | 0 | 0 |
| sys_dlmb_cntlr (system_sys_dlmb_cntlr_0) | 6 | 2 | 0 | 0 | 4 | 6 |
| sys_ilmb (system_sys_ilmb_0) | 0 | 0 | 0 | 0 | 0 | 0 |
| sys_ilmb_cntlr (system_sys_ilmb_cntlr_0) | 2 | 2 | 0 | 0 | 2 | 2 |
| sys_lmb_bram (system_sys_lmb_bram_0) | 0 | 0 | 0 | 0 | 0 | 0 |
| sys_mb (system_sys_mb_0) | 3597 | 3368 | 148 | 1 | 1275 | 3352 |
| sys_mb_debug (system_sys_mb_debug_0) | 183 | 173 | 0 | 0 | 78 | 167 |
| sys_rstgen (system_sys_rstgen_0) | 24 | 33 | 0 | 0 | 15 | 23 |
| track_hier0 (track_hier0_imp_1HE0FM9) | 18192 | 7205 | 55 | 2 | 5508 | 17769 |
| track_hier1 (track_hier1_imp_SBAA3P) | 18184 | 7205 | 55 | 2 | 5656 | 17761 |
| track_hier2 (track_hier2_imp_ODVTX4) | 18188 | 7205 | 55 | 2 | 5682 | 17765 |
| track_hier3 (track_hier3_imp_1V12DFW) | 18185 | 7205 | 55 | 2 | 5363 | 17762 |
| util_ad9361_adc_fifo (system_util_ad9361_adc_fifo_0) | 18 | 63 | 0 | 0 | 18 | 18 |
| util_ad9361_adc_pack (system_util_ad9361_adc_pack_0) | 162 | 437 | 32 | 0 | 85 | 162 |
| util_ad9361_divclk (system_util_ad9361_divclk_0) | 0 | 0 | 0 | 0 | 0 | 0 |
| util_ad9361_divclk_reset (system_util_ad9361_divclk_rese... | 21 | 28 | 0 | 0 | 9 | 20 |
| util_ad9361_divclk_sel (system_util_ad9361_divclk_sel_0) | 1 | 0 | 0 | 0 | 1 | 1 |

Figure 8.1: Utilization report

# APPENDIX A

## Tables for Reference

| Satellite Number | Code Phase Selection from G2 |
|---|---|
| 1 | 2,6 |
| 2 | 3,7 |
| 3 | 4,8 |
| 4 | 5,9 |
| 5 | 1,9 |
| 6 | 2,10 |
| 7 | 1,8 |
| 8 | 2,9 |
| 9 | 3,10 |
| 10 | 2,3 |
| 11 | 3,4 |
| 12 | 5,6 |
| 13 | 6,7 |
| 14 | 7,8 |
| 15 | 8,9 |
| 16 | 9,10 |
| 17 | 1,4 |
| 18 | 2,5 |
| 19 | 3,6 |
| 20 | 4,7 |
| 21 | 5,8 |
| 22 | 6,9 |
| 23 | 2,6 |
| 24 | 4,6 |
| 25 | 5,7 |
| 26 | 6,8 |
| 27 | 7,9 |
| 28 | 8,10 |
| 29 | 1,6 |
| 30 | 2,7 |
| 31 | 3,8 |
| 32 | 4,9 |

Table A.1: Code Phase Selection

| Parameter | No. of Bits | Scale Factor(LSB) | Unit |
|---|---|---|---|
| IODE | 8 | | |
| $C_{rs}$ | 16* | $2^{-5}$ | m |
| $\Delta n$ | 16* | $2^{-43}$ | semicircle/s |
| $\mu_o$ | 32* | $2^{-31}$ | semicircle |
| $C_{uc}$ | 16* | $2^{-29}$ | radian |
| e | 32 | $2^{-33}$ | dimensionless |
| $C_{us}$ | 16* | $2^{-29}$ | radian |
| $\sqrt{a}$ | 32 | $2^{-19}$ | $m^{0.5}$ |
| $t_{oe}$ | 16 | $2^{4}$ | s |
| $C_{ic}$ | 16* | $2^{-29}$ | radian |
| $\Omega_o$ | 32* | $2^{-31}$ | semicircle |
| $C_{is}$ | 16* | $2^{-29}$ | radian |
| $i_o$ | 32* | $2^{-31}$ | semicircle |
| $C_{rc}$ | 16* | $2^{-5}$ | m |
| $\omega$ | 32* | $2^{-31}$ | semicircle |
| $\dot{\Omega}$ | 32* | $2^{-31}$ | semicircle/s |
| $\dot{i}$ | 14* | $2^{-43}$ | semicircle/s |

Table A.2: Decoding Scheme for GPS Ephemeris Parameters

| | |
|---|---|
| IODE | issue of data, ephemeris |
| $\Delta n$ | mean motion correction |
| $\mu_o$ | mean anomaly at $t_{oe}$ |
| e | eccentricity |
| $\sqrt{a}$ | square root of semi-major axis |
| $t_{oe}$ | reference epoch of ephemeris |
| $\Omega_o$ | longitude of ascending node at $t_{oe}$ |
| $i_o$ | inclination at $t_{oe}$ |
| $\omega$ | argument of perigee |
| $\dot{\Omega}$ | rate of $\Omega_o$ |
| $\dot{i}$ | rate of i |
| $C_{rs}$,$C_{rc}$ | correction coefficients for sine and cosine terms of r |
| $C_{is}$,$C_{ic}$ | correction coefficients for sine and cosine terms of i |
| $C_{us}$,$C_{uc}$ | correction coefficients for sine and cosine terms of $\omega$ |

Table A.3: Ephemeris Parameters Definition

# REFERENCES


1. "A Software-Defined GPS and Galileo Receiver A Single-Frequency Approach" by Kai Borre, Dennis M. Akos, Nicolaj Bertelsen, Peter Rinder, SÃÿren Holdt Jensen

2. "Fundamentals of Global Positioning System Receivers", by James Bao-Yen Tsui

3. PRN code fundamentals: https://natronics.github.io/blag/2014/gps-prn/