Simulink Implementation of a GPS Receiver

Alamjyot Parihar 260570908
Ahmed Sami 260568821

Project Supervisor: Prof. Haryy Leib

1 Abstract

Traditionally the algorithms that make up a complete software defined GPS are done in assembly, C/C++ or Matlab. This requires a large team to write and test algorithms, which is both costly and time consuming. This project is intended to provide a basis for clear, easy to follow and modify implementation of a L1 carrier GPS receiver. Our simulation employs the use of the graphical programming environment Simulink to design and test models.This project aims to make the fundamental functionality of a GPS receiver more visible and the inner workings simpler to examine and analyze. We hope that our implementation will help pave the way for other simpler methods of implementing and understanding intricate systems.

2 List of Abbreviations

CDMA - Code Division Multiple Access

BPSK - Binary Phase Shift Keying

LO - Local Oscillator

C/A - Coarse/Acquisition

DLL - Delay Locked Loop

PLL - Phase Locked Loop

SNR Signal to Noise Ratio]

3 Introduction

4 Doppler Shift

Due to the nature of satellite navigation systems, the point of signal generation is constantly moving overhead. This movement causes a doppler effect on the signal, leading to shifts in carrier frequency and code phase seen at the receiver. Typically, doppler shifts to code phase can reach up to ± 3 chips/s for stationary receivers.

5 Signal Generation

The signal generator for testing tracking loops is similar to the previous function generator used to test the acquisition loop[1]. Several additions and modifications made to the generator are explained below.

[1]Detailed explanation of acquisition function generator available in previous semesters report

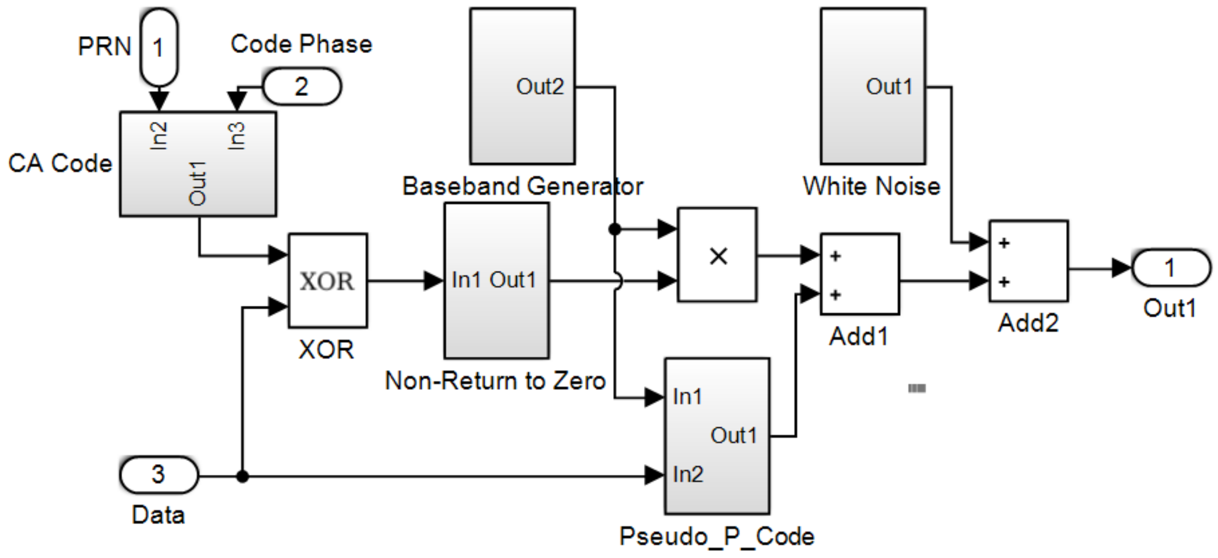


Figure 1: Signal Generator

5.1 Simulated Doppler Shift

In order to properly simulate effects of the Doppler shift, a counter controlled VCO was used to generate the carrier signals. The counter begins at 0 and counts to 2500 over a one second period. Setting the VCO input sensitivity to ± 0.08,0.04, or 0.02Hz/V, allows us to achieve frequency shifts of ± 200, 100, or 50Hz/s respectively. A counter was also used to shift the C/A code phase by increments of $\frac{1}{32}$ (made possible by up-sampling and shifting the original code). Adjusting the sampling period of the counter, the desired code phase shifts of ± 3 to 6 chips/s can be produced.

5.2 Pseudo P-Code and Noise

A pseudo P-code was generated using a square wave with the same frequency of the actual P-code. This code is then modulated on to a 90 degree phase shifted carrier and tuned 3db down to better approximate actual GPS signal generation.

The final addition to the signal generator was that of a white noise block. This allows us to simulate noise that may occur in actual signals. We chose white noise specifically as it has equal intensity at different frequencies, providing a fairly uniform PSD. The ratio of signal to noise if varied between 10dB and 20dB

6 Received Signal Demodulation

Once a signal has been determined to be in range by the acquisition stage, the data must be demodulated. To perform demodulation, it is required to multiply the incoming signal with a carrier-frequency matched local oscillator. This effectively wipes the carrier from the signal, bringing it back to the data baseband. The last step is to multiply the signal with a locally generated C/A code, effectively decoding the data.

The values found during acquisition are used to generate the local oscillator and C/A code, however this is not enough. As explained above, the code phase and carrier frequency undergo Doppler shifts and change over time. These changes must be tracked and fed back to the generators to provide stable, coherent demodulation. The process of signal tracking is explained in the following section.

7 Tracking Loops

7.1 Carrier Tracking

7.1.1 Costas Loop

7.1.2 Discriminators

7.2 Code Phase Tracking

7.2.1 Delay Locked Loop

7.3 Discriminators

7.4 Combined Tracking Loop

8 Simulink Implementation of Tracking Loop

8.1 Simulink Model

The tracking loops implemented in simulink are modeled after a combined costas loop and delay locked loop mentioned in the previous section. The tracking stage block contains 4 of these loops, which track signal simultaneously

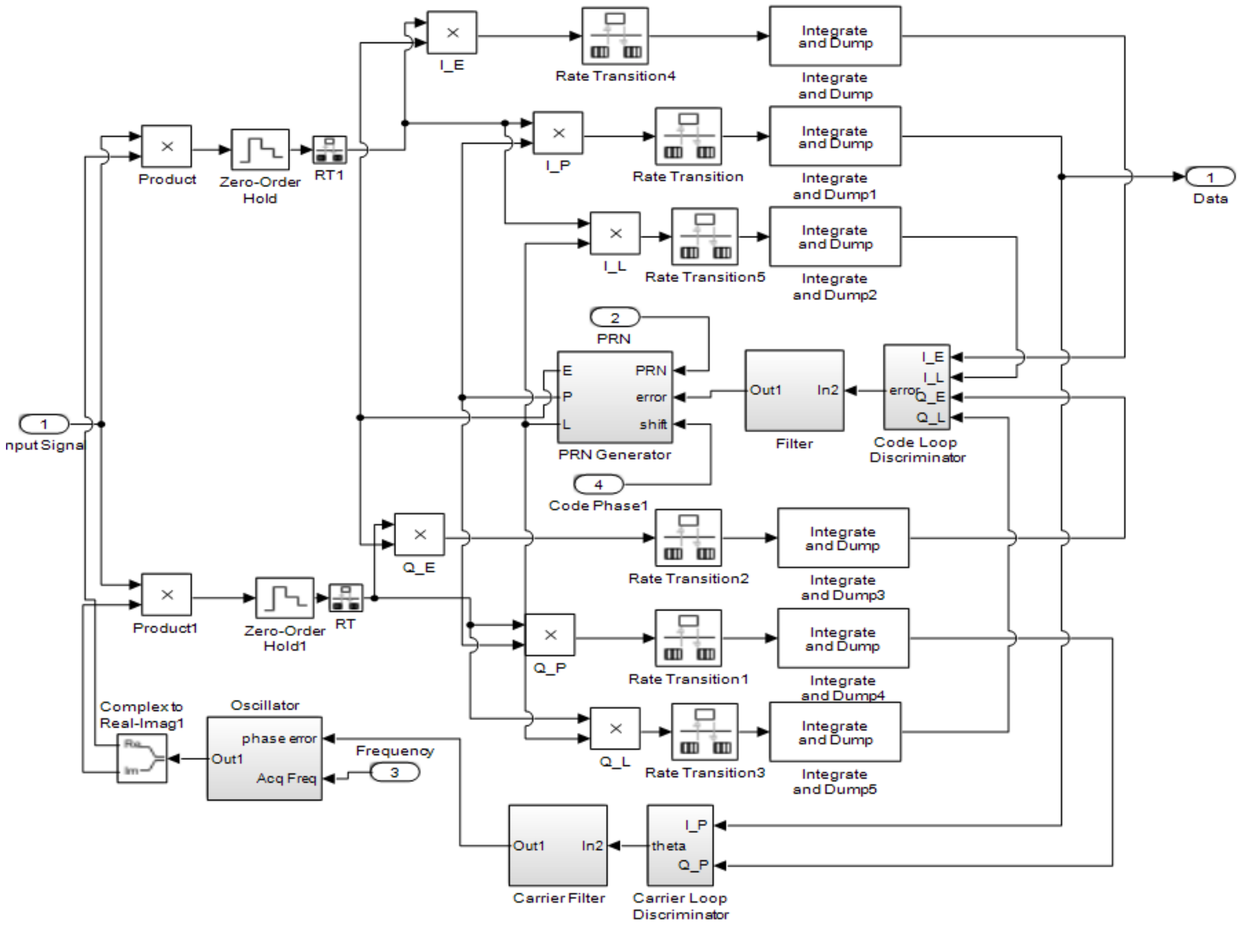


Figure 2: Combined tracking loop simulink implementation

8.2 Design Parameters

When transitioning from the block model to the simulink model, there are several design choices to be made. The error term used to drive the local oscillator for carrier tracking is an arctan discriminator. This provides indifference to the 180 deg phase shifts seen in BPSK signals. The error term used to drive the C/A code generator is the normalized early minus late power discriminator. This discriminator provides the best tracking when chip errors are larger than one. The integration period of the integrate and dump block is 1ms, equal to one full C/A code. In order the perform the necessary signal processing in discrete time, the continuous input signal was converted using a zero order hold block with a sampling rate equal to the sampling rate of our system (f_s=32.768MHz).

8.3 Controlled Oscillator

The controlled oscillator used to wipe the carrier from the signal is modeled after a numerically controlled oscillator. The formulas used to control the output of the NCO are provided below. Note that the accumulator word length (N) is 16 for our implementation.

The frequency of the output is $f_0 = \frac{\Delta\theta f_s}{2\pi 2^N}$

where $\Delta\theta[n] = 2\pi 2^N(\frac{f_c}{f_s} + e[n])$

and the frequency resolution is

Using this relationship between error term and NCO output, A simulink model for the local oscillator was created as shown below.

8.4 Controlled C/A Code Generator

The C/A code generator is modeled after that of the acquisition stage, with a few adjustments. To make use of our desired discriminator, we must produce an early and late version of the C/A code (spaced half a chip apart) in addition to the prompt code. To space the chips properly, the C/A code is up sampled 32 times and shifted 16 bits away from the correct code phase, in the appropriate direction. The error term of the code phase discriminator is then added to the detected code phase from acquisition, allowing tracking of the correct code phase to be input to the local C/A code generator.

8.5 Discriminator Filtering

In order to provide smooth changes to the local carrier generator, the error term must be filtered. The filter implemented in simulink consist of a simple second order IIR filter. The fixed parameters used to choose appropriate coefficients are

Bandwidth 30Hz

Damping Coefficient 0.707

Loop Gain $\frac{1}{4}$

The set values of these parameters were chosen based on resulting waveforms of the carrier discriminator for several test values. The actual coefficients used in the IIR fiter block can then be calculated by using the following formulas: $formula$

9 Tracking Loop Testing

To validate the full functionality of the tracking stage, we must first test each individual parameter. Certain parameters were tested as the loop was built, however this does not garaunteThis requires us to generate a few test signals to work with. A real GPS signal would use data bits of 50Hz, however, for the purpose of saving time on simulations we have set it to 100Hz. Data bit values are set according to a 5 bit repeating vector unique to each generated signal. The defining features of our test signals are listed below, where the value in brackets indicates the doppler shift.

Satellite Number	Carrier Frequency	Code Phase	Data Vector
1	9.209MHz (+50Hz/s)	0 (+3chips/s)	[1 0 0 1 1]
2	9.204Hz (-100Hz/s)	200 (+3chips/s)	[1 0 1 0 1]
3	9.207MHz (+200Hz/s)	512 (-3chips/s)	[0 1 1 0 0]
4	9.207MHz (+100Hz/s)	999 (+3chips/s)	[1 0 0 1 0]

Normally the satellite PRN number, code phase and carrier frequency would be passed to the tracking loop from the acquisition stage, however for the purposes of testing, these parameters are pre-set and loaded in from a matlab array.

The very first test involving the tracking stage was simply to demodulate the data from a single signal when no other signals are present. This is fairly straightforward as the tracking loop should be able to demodulate data in the begining without worrying about tracking changes. This begining test was successful in recovering the generated data, and we were able to move on to the next stage of testing.

9.1 Carrier Tracking Testing

The first carrier tracking test involved a single test signal, feeding the tracking loop with the exact values used to generate it. As the signal propagates, the frequency will changing, and we must look at a plot of the discriminator error term to determine whether it is being tracked accurately. A plot of the

The second carrier tracking test involved feeding the four different tracking loop with same signal, but with varying deviations to the initial carrier frequency.

The final carrier tracking test was carried out using an input signal consisting of a sum of all 4 generated signals. The purpose of this testing stage is to determine whether or not the loop can coherently demodulate data from a particular satellite out of a sum of signals.

9.2 Code Phase Tracking Testing

Code phase tracking was carried out in a manner similar to carrier tracking. The first simulations was a single signal test performed using correct initial parameters.

The second test was a single signal test performed with varying deviations to the initial code phase estimate. Results are shown plotted below, labeled with the differing initial values and the true value of the code phase.

The final code phase tracking test

9.3 Combined Signal Testing

10 Synthesis with Acquisition Stage

The final phase of this project was to test both the acquisition and the tracking stages together. The information used for tracking initializations must now come from the acquisition results instead of being pre-set (as in previous testing stages). To make this synthesis function as intended, some overhead logic is needed to control both models, as well as store and sort satellite parameters. Testing results and details of the adjustments required to successfully synthesize both models is provided in the following sections.

10.1 Adjustments to Acquisition and Tracking

To gather useful information from the correlation output, a matlab function block was written which stores the code phase of the highest correlation value at every frequency step. The largest value per C/A code is found in a similar manner, allowing us to locate both the code phase and frequency step of the highest correlation peak. If this peak is determined to be above our threshold ($2 * 10^7$, found through correlation data analysis) it is added to an array of acquired signals.

As the array is filled, the signals should be organized according to signal strength; This can be achieved by looking at either correlation peaks or the SNR value. The organized array is then output from the loop,

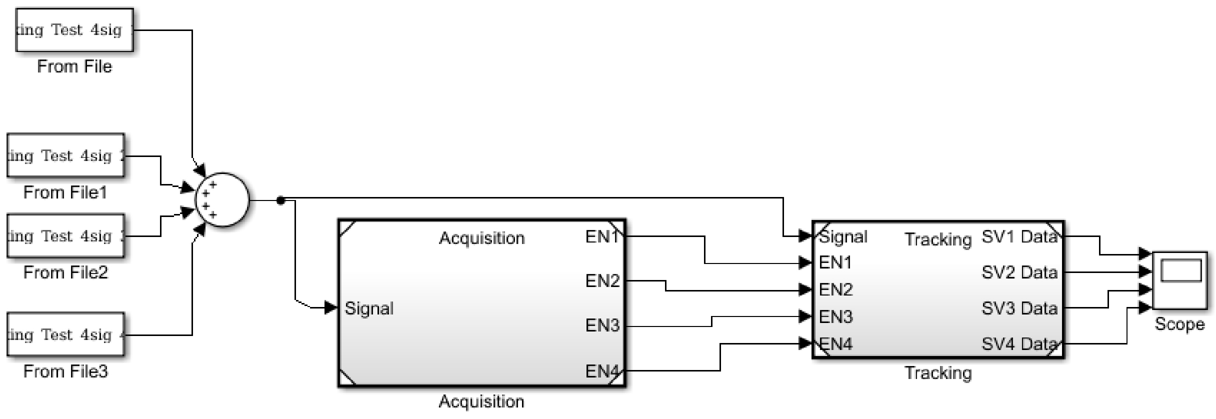


Figure 3: Synthesized receiver model

where it is written to a variable via a data set memory block. This shared memory block will then be read out during tracking and used to initialize the loop parameters, as shown below.

The acquisition block controls the start of tracking stage execution through a wired boolean enable line. The acquisition loop will also terminate itself when it is no longer needed. The tracking stage of a real GPS receiver would ordinarily not start until the best suited satellites are found, which requires a full sweep through the 24 possible satellites orbiting the earth. For the purposes of our testing however, the acquisition stage will enable the tracking of signals as they are found. This allows us to reduce the computation time required to view the results of tracking.

10.2 Final Test

The final test was performed on the sum of 4 generated signals. All of the signals have a data rate of 100Hz (real GPS uses 50Hz, changed to reduce required computation time) as well as different 5 bit repeating data vectors. The goal of this final stage testing is to simply provide an input signal and see if the recovered data bits match the generated ones.

11 Data Navigation Extraction Overview

12 Impact on Society and the Environment

13 Report on Teamwork

14 Conclusion

15 References

16 Appendix

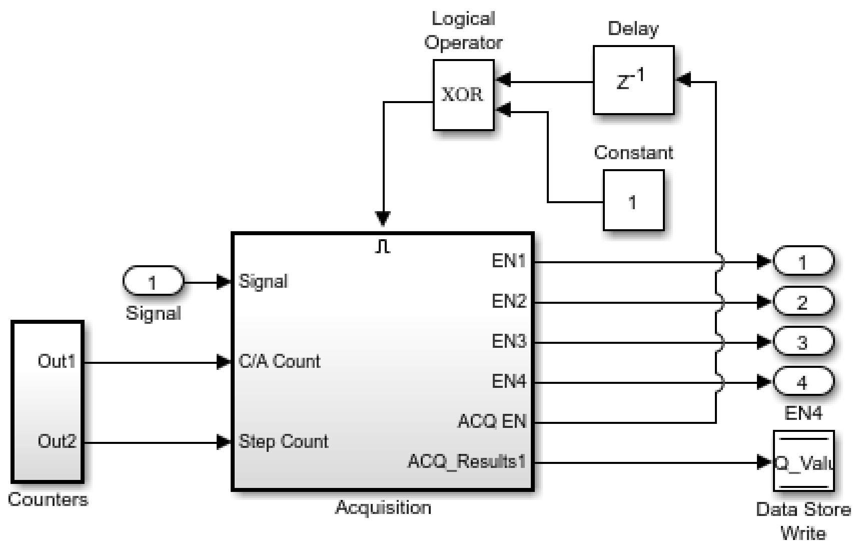
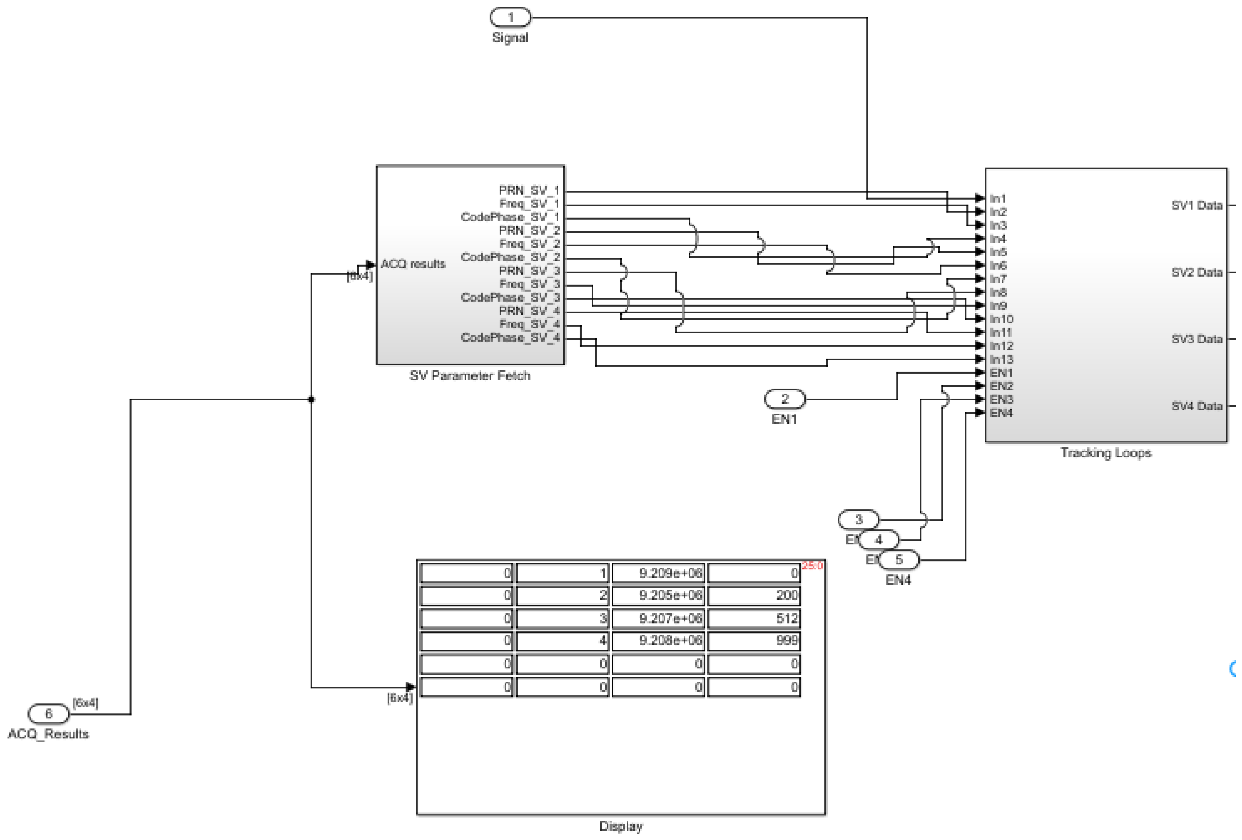


Figure 4: Overhead logic used to control acquisition model



0	1	9.209e+06	0
0	2	9.205e+06	200
0	3	9.207e+06	512
0	4	9.208e+06	999
0	0	0	0
0	0	0	0

Figure 5: Tracking loop initialization