Journal of Global Positioning Systems (2009)
Vol.8, No.1 : 75-84

Journal of Global Positioning Systems
2009
© CPGPS

Implementation of a Complete GPS Receiver on the C6713 DSP through Simulink

Gihan Gomah Hamza†, Abdelhaliem A. Zekry‡, and Mohamed N. Moustafa♀
†Research assistant in the National Institute of Standards, Guiza, Egypt.
‡Professor in the Communications and Electronics Department, Faculty of Engineering, Ain Shams University, Cairo, Egypt.
♀Assistant professor in the Computer and systems Department, Faculty of Engineering, Ain Shams University, Cairo, Egypt.

Abstract

In the past, the implementation of a complete GPS receiver was divided into two parts. The first part is implemented on ASIC or FPGA. This part includes the acquisition and tracking phases, where the algorithms of the both are written by the HDL programming language. The second part is the navigation solution part that is implemented on DSP by writing its algorithm by C/C++, FORTRAN, or assembly. This means that we have to deal with three environments to implement a complete GPS receiver. The three environments are the Simulation, FPGA, and DSP environments. Moreover, using a text programming languages in writing such long and complicated algorithms makes the process exhaustive and difficult to debug, modify, and learn. This article introduces the simulation and implementation of a complete GPS receiver on a DSP through a graphical programming language, which is SIMULINK. This makes every part in the receiver architecture very clear and easier to understand, follow, modify and debug. This can be considered a step added on the route of an open source GPS receiver. Using the same environment in both the simulation and implementation stages makes the designer's mind dedicated most of the time in developing and enhancing the algorithm through rapid prototyping and experimentation and less time on the coding. In general, this article can be considered as introducing a new look for designing, simulating, and implementing the most complicated parts of a typical GPS receiver using a graphical programming language, which is SIMULINK.

Keywords: GPS receiver, MATLAB, SDR, Simulink, System design, RTW, and RTDX.

1 Introduction

Any GPS receiver that deals with the Standard Positioning Service (SPS) transmitted on the L1 carrier of 1.57542GHz using the Coarse Acquisition (C/A) code has four phases to extract the position and time from the received signal. These phases are: the front-end, the acquisition phase, the tracking phase, and the navigation solution (calculations) phase.

Previously, and disregarding the front-end, to implement such receiver according to the SDR technology, it is required to deal with three environments; A simulation environment such as MATLAB or Simulink to simulate and verify each phase, FPGA environment to implement and verify both the acquisition and tracking phases by writing the algorithm in the HDL language, and a DSP environment to implement and verify the navigation solution (calculations) phase by writing its algorithm in C/C++ programming language. This process, for sure, is exhaustive and takes a long time. In this article we introduce the simulation and implementation of all the phases of a GPS receiver that deals with the SPS service through Simulink, which is a graphical programming language. Utilizing the same environment in the simulation and implementation stages makes the designer mind dedicated most of the time in developing and enhancing the algorithm and less time on the coding.

In fact, this article is a continuation to the efforts that have been exerted by Kai Borre, Dennis Akos, and others to facilitate the simulation of the GPS receiver. These efforts fruited a book called "A Software-Defined GPS and GALILEO Receiver: A Single-Frequency Approach" in which they introduced the simulation of a complete single frequency GPS receiver using the C/A code on the L1 carrier utilizing MATLAB as the coding and simulation environment. They introduced the receiver's algorithm in 39 m-files. The GPS signal that they used in verifying the algorithm was a real signal received by an ASIC-based front-end fabricated especially for Borre–Akos book.

In this article the Borre-Akos algorithm was ported to Simulink for the purpose of both the simulation and implementation on the C6713 DSP starter kit through Simulink. We added some modifications, which will be stated in the following sections, to make the Simulink models implementable on a DSP.

The implementation process is done through utilizing the Real Time Workshop (RTW) and the Target Support Package (TC6) softwares introduced by Matlab. Each phase was built as a separate Simulink model and verified on the Code Composer Studio (CCS3.3) C6713 Device Functional Simulator from Texas Instruments. The tracking phase was emulated on the C6713DSP starter kit. All the simulated and verified phases are combined in a single Simulink model to represent a complete GPS receiver, which is ready to be downloaded on a DSP by just one click. The GPS signal that was used in verifying our Simulink models was the same real signal that was used in verifying the Borre-Akos algorithm. This signal was accompanying the Borre-Akos book on a DVD. We used the CCS Functional simulator instead of the 6713 DSP starter kit for the acquisition and the navigation solution phases because the generated C-code from Simulink requires a RAM size more than the 16MB that was available on the DSP kit.

2 The Front-End

The real GPS signal that was used in verifying both the Borre-Akos algorithm and our Simulink models was received by the SE4110 ASIC-based front-end. The functional block diagram of this front-end is shown in Fig. 1.

The received GPS signal has the following parameters:-

- Sampling frequency (Fs): 38.192MHz,
- Intermediate Frequency (IF): 9.548MHz, and
- Eight-bit samples.

These parameters are acceptable in the Simulation stage and will not cause major problems. Because the hardware resources required for the simulation will be withdrawn from the computer (PC). When we intended to implement the simulated models we had to make another look to these parameters specially the sampling frequency. Using a high sampling rate has a direct impact on increasing both the processing time and the number of operations required by the processor platform whether it was DSP or FPGA. Because as the sampling frequency increases the number of samples to be processed increases. So, we have to either make another front-end that belongs to the low-IF type or change these parameters as a step prior to the acquisition phase. We chose the second solution.

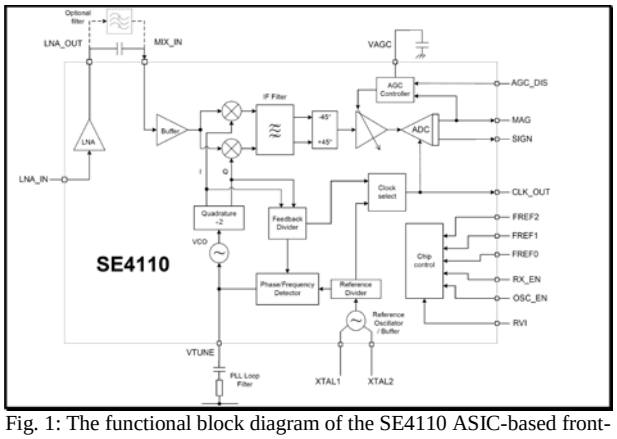

Fig. 1: The functional block diagram of the SE4110 ASIC-based front-end

Changing the IF to a value equals to double the code frequency, which amounts to 1.023MHz, and it is done by multiplying the input signal by a frequency value equal to the difference between the original IF and double the code frequency (i.e., 9.548MHz – 2*1.023MHz). After acquisition, the detected carrier frequencies for the visible satellites are referenced again to the original IF by adding to them the same difference.

In the acquisition phase we are using the parallel code phase search acquisition, which will be described in the next section, by performing Circular Correlation through Fourier Transform. The data size selected for acquisition was 1 ms (equivalent to one complete C/A code). This means that the number of samples per C/A code to be processed equals to $F_s \times 1$ms. So if the sampling frequency had the lowest possible radix-2 value, (i.e., $Fs=2^n$; n is a positive integer), the acquisition time will be decreased and the processor requirements will be relaxed.

The resampling process was executed by an unordinary manner. The number of samples in a complete C/A code, which will be processed in acquisition and equals to 38192 samples, was first interpolated to be 65536 samples and then each 8 or 16 consecutive samples were replaced with their mean. By this way we obtain a sampling frequency of 8.192MHz, which means 8192 samples per C/A code, or 4.096MHz, which means 4096 samples per C/A code, respectively. The detected code phase, for a visible satellite, after resampling by such a manner can be referenced again to the original number of samples per C/A code by multiplying the detected one by either (38192/8192) or (38192/4096) then rounding the result to the nearest integer. Fig. 2 and Fig. 3 show the process of stepping down both the IF and Fs.

We have to notice here that resampling by a fractional parameter using the “Resample” function in MATLAB gave worse simulation results as well as caused problems

during implementation.

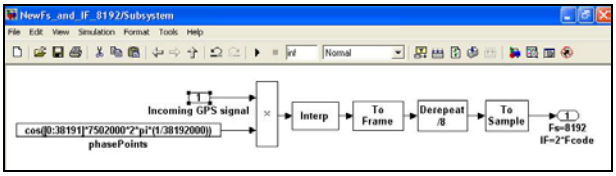

Fig. 2: Changing the IF to be double the code frequency and stepping down Fs to be 8.192MHz

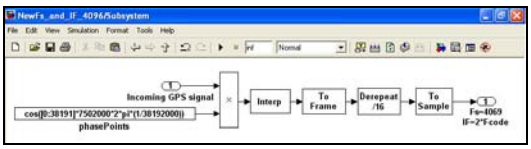

Fig. 3: Changing the IF to be double the code frequency and stepping down Fs to be 4.096MHz

Resampling here implies that we resample only the incoming GPS signal but both the locally generated demodulating frequency and dispreading PRN codes are sampled according to the new parameters from the start. In other words, changing the IF and Fs in such way is equivalent to processing a signal received from a low-IF front-end that has the new Fs and IF. This method is superior to the Averaging Correlator method because it decreases the simulation time significantly and doesn't imply sampling and resampling both the locally generated modulating frequency and dispreading codes.

Now we have two trials for making successful acquisition after reducing IF and Fs. The first trial uses the following parameters:

- Fs=8.192 MHz,
- IF=2*Code Frequency,
- No. of samples per C/A code=8192, and
- No. of samples per chip=8.
- The second trial uses the following parameters:
- Fs=4.096 MHz,
- IF=2*Code Frequency,
- No. of samples per C/A code=4096, and
- No. of samples per chip=4

3 Acquisition

The main purpose of acquisition is to determine the frequency, as a rough estimation, and the code phase of the PRN for each visible satellite. Acquisition was implemented by the parallel code phase search technique that is shown in Fig.4. In this technique we circularly correlate (in the frequency domain) the incoming GPS signal with both a number of the locally generated carrier frequencies and PRN codes of different code phases. The locally generated carrier frequencies cover a range determined according to the satellites rotation velocity and the receiver velocity. The minimum frequency range that corresponds to a fixed GPS receiver equals to IF±5KHz. The maximum range occurs for a receiver that has a very high velocity and this range equals to IF±10KHz. In general, we use a 500Hz frequency bin. The number of the locally generated PRN codes is 32, which corresponds to the number of all the working satellites in the space segment. We have a correlation peak if and only if both the generated carrier frequency and the phase of the generated PRN are perfect replicas for those of the received signal. The search through all the possible frequency bins is parallelized with the code phase search such that the total number of searches per satellite equals the number of frequency bins. The number of frequency bins in our case was 29 frequency bins for ±7 KHz search band. The parallel code phase search technique was implemented in Simulink two times, as shown in Fig. 5, to eliminate the effect of the data bit transition. In the Simulink model of acquisition the 29 frequency bins are searched at the same time where the incoming signal, after resampling, is multiplied by a matrix containing all the possible IF frequencies. This means that the total number of searches for the 32 satellites is 32. We ran the acquisition algorithm of Borre-Akos that was written as m-code for Fs=38.192MHz and IF=9.548MHz. Also we ran our Simulink model of acquisition two times. The first time for Fs=8.192MHz and IF=2.046MHz and the second time for Fs=4.096MHz and IF=2.046MHz. In the following section we will compare the acquisition performance in the three cases.

Simulation stage

Acquisition results are represented by four outputs: the detected PRNs, the Signal to Noise Ratio (S/N), the code phase of the PRN, and the carrier frequency (IF) for the visible satellites. The S/N represents the ratio between the highest correlation peak to the next peak.

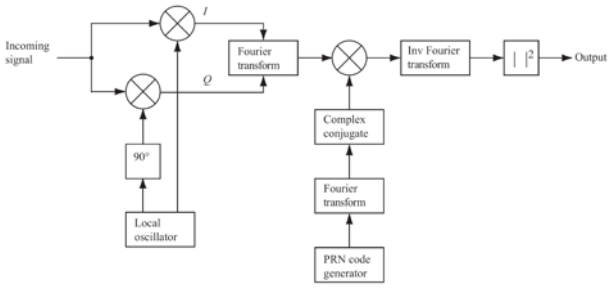

Fig. 4: The parallel code phase search algorithm

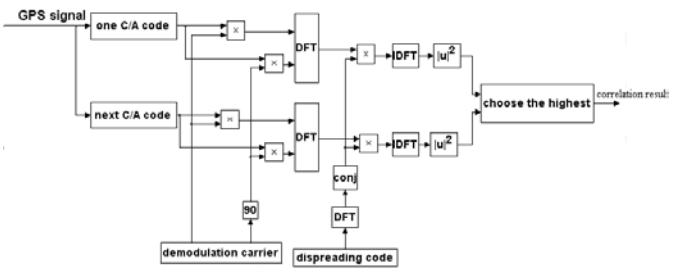

Fig. 5: The implementation of the parallel code phase search algorithm to avoid data bit transition

Table 1 shows a comparison between the acquisition results obtained from the Borre-Akos algorithm and those obtained from our Simulink model after modifying the Fs and IF. This comparison is made according to: 1- the number of detected PRNs in each trial; 2- the maximum percentage error ($\frac{original\ \mathrm{Result} - trial\ \mathrm{Result}}{original\ \mathrm{Result}}\%$) in the detected S/N, code phase, and IF for each trial. Our reference is the results obtained from Borre-Akos algorithm; 3- the data type used in the processing; 4- the simulation environment; 5- the acquisition time for each trial. Fig. 6a, 6b, and 6c show the acquisition plot using the original and modified IF and Fs.

From Fig. 6 and Table 1 it is clear that the percentage error in the results obtained at Fs=8.192MHz are negligibly small and we can say that we obtained nearly typical results as those of Borre-Akos but with much reduced acquisition time.

Although the maximum error in the S/N for the PRNs detected at Fs=4.096MHz exceeded 50%, the error in the detected code phase of the visible satellites is very small. This is because the code phase step in the parallel code phase search acquisition technique is one sample. This means that we can detect the code phase with a very high accuracy. In spite of missing three PRNs we could acquire more than four satellites with accurate code phase and carrier frequency. This means that a lower Fs can be used for stronger GPS signal reception.

Implementation stage

It was difficult to make the implementation stage at the original sampling frequency due to the huge number of samples per C/A code to be processed. So, the implementation was executed for the model that work according to the modified IF and Fs.

	Borre-Akos IF=9.548MHz Fs=38.192MHz (Reference)	**1st trial IF= 2.046MHz Fs=8.192MHz**	**2nd trial IF=2.046MHz Fs=4.096MHz**
Detected PRNs	8	8	5
Max. % error in S/N	0%	14.5%	50.4%
Max. % error in Code phase	0%	0.12%	0.25%
Max. % error in Detected IF	0%	7.3%	5.3%
Data type used	Double	Single	Single
Simulation environment	M-files	SIMULINK	SIMULINK
Acquisition time	183 sec	< 24 sec	< 17 sec

Table 1: Comparison between the acquisition results obtained before and after modifying the IF and Fs

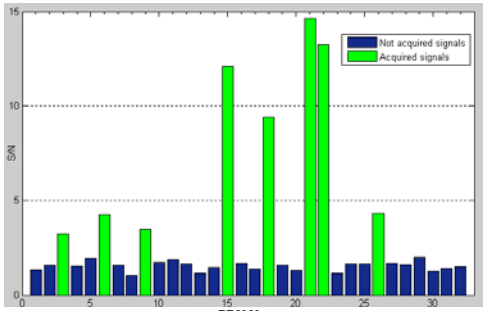

Fig. 6a: Acquisition plot at IF= 9.548MHz and Fs=38.192MHz

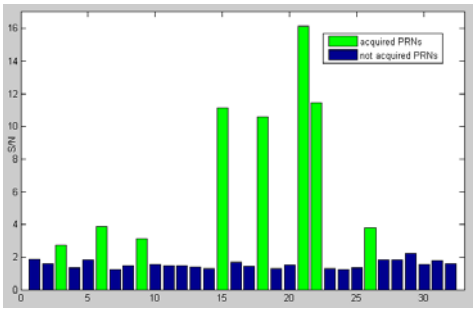

Fig. 6b: Acquisition plot at IF= 2.046MHz and Fs=8.192MHz

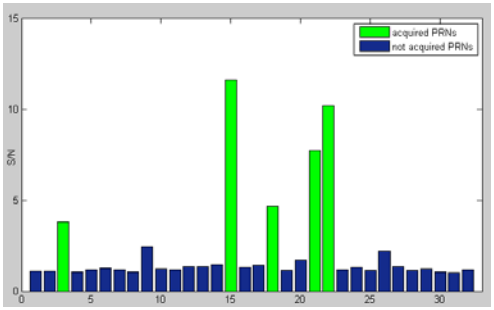

Fig. 6c: Acquisition plot at IF= 2.046MHz and Fs=4.096MHz

The simulated Simulink model isn't necessarily the same model that can use the Real Time Workshop (RTW) technology that automatically generates a C/C++ source code from it because:

1- Not all the Simulink blocks used in building the model are permitted to use the RTW. These blocks can be known by reviewing the help of each block. In this case we have to rebuild the functions done by such block by using another blocks that have a permission to use the RTW.

2- There may be a combination of blocks in the Simulink model that can't be understood by the RTW. The problem here is that when we try to Build (use the RTW and TC6) the model gives either pseudo errors or no errors and stop Building. This combination can be identified by partitioning the model into small sections. If the first section succeeded in using the RTW then we add to it the next section and try to use the RTW and so on till we fail after adding certain section. At this moment we have to write the function of that section as M-code using

the Embedded MATLAB Function block.

The two bottlenecks in the acquisition model that were modified to enable it to use the RTW successfully were: a combination of two Selector blocks used at the input to make acquisition for two consecutive C/A codes and a combination of blocks used in searching for the correlation peak, frequency bin, and code phase. The two combinations are reconstructed by writing their functions as m-code using the Embedded MATLAB Function.

Inserting m-code in the Simulink model makes it slower in the simulation stage, but enables it to use the TC6 that integrates Simulink with Texas Instruments Express DSP Development tools.

To verify that the generated C-code from the Simulink acquisition model gives the same results as in the simulation stage, Real Time Data Exchange (RTDX) blocks are inserted at the model's outputs before generating the C code to enable it from transferring the results from the C6000 target to the PC. We found that all the DSP results are the same as the simulated results shown in Fig.6.

The implementation of the acquisition phase is done by using the C6713 Device Functional Simulator existed in the Code Composer Studio (CCS 3.3) because the RAM size required by the generated code exceeded the 16MB that was available on the C6713 starter kit.

4 Tracking

The main purpose of tracking is to refine the acquisition results, track any changes that occur to these results with time, and demodulate and dispread the incoming signal to obtain the 50 bits/s navigation data. The data size that is being processed in each tracking cycle is a complete C/A code plus or minus the delay or lag that is being calculated during tracking. Tracking starts if the number of the detected PRN codes at the acquisition phase ≥ 4.

In general, tracking is implemented as a carrier tracking loop that works in consistency with a code tracking loop. The carrier and code tracking loops that are used usually in GPS receivers are the Costas Phase Lock Loop (PLL) and the Delay Lock Loop (DLL). Fig. 7a, and 7b show the block diagram of both Costas PLL and the DLL. Usually the two loops are combined in one loop to increase the control on the generated carrier frequency and the code phase and to reduce the number of multipliers. This helps in reducing the tracking time.

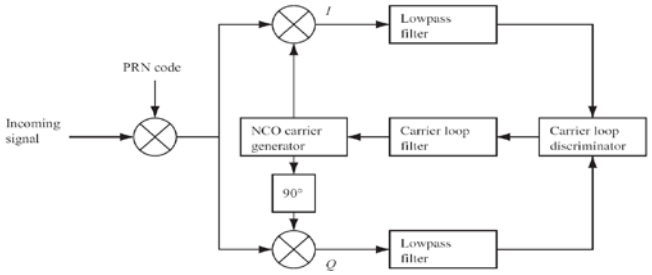

Fig. 7a: Costas phase locked loop

Simulation stage

A complete tracking channel was built in Simulink according to the functional block diagram of the combined Costas PLL and DLL that is shown in Fig.8. Fig. 9 shows the Simulink model of a complete tracking channel, which corresponds to the functional block diagram shown in Fig. 8. Now, to demonstrate how using a graphical programming language in simulation makes the simulated system very clear we will describe each block in Fig. 9.

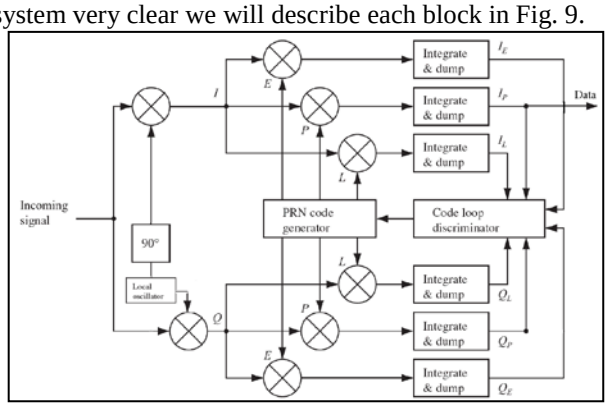

Fig. 7b: Delay Lock Loop has 6 correlators

The Carrier Loop Discriminator block was built as an arctan discriminator according to the following equation:

$$\phi = \tan^{-1}(\frac{Q}{I})$$

Where ϕ is the phase error. I and Q are the in-phase and quadrature signals of Costas loop. The implementation of this discriminator is shown in Fig. 10.

The Carrier Loop Filter was implemented as a second order filter in the carrier tracking loop that has the following parameters:

- Damping Ratio (ζ) =0.7,
- Noise Bandwidth (Bn) =25Hz,
- Natural frequency (w_n) = 8 ζB_n/ ($4\zeta^2+1$),
- Loop gain (K) = 0.25, and
- Filter coefficients (τ_1, τ_2), where $\tau_1=K/(w_n)^2$ and $\tau_2=2\zeta/ w_n$.

Fig. 11 shows the implementation of this filter in Simulink.

The NCO Carrier Generator block, which represents the numerically controlled oscillator, is implemented in Simulink as a center frequency, which was detected in the acquisition phase, its value is decreased or increased according to the filtered output of the discriminator. Sampling the locally generated carrier frequencies according to the original Fs, which amounts to 38192KHz, was implemented in Simulink as shown in Fig. 12.

The PRN Code Generator block generates three code replicas that have a spacing of ±½ chip. The three code replicas are correlated with the incoming signal. According to the correlation result we decide to lag or lead the P-code to follow the changes on the phase of the received code. The PRN Code Generator block was implemented as a look up table that retain all the 32 PRNs without sampling. According to the detected PRN from acquisition and the anticipated number of samples per C/A code that is detected from the previous tracking cycle, we sample this PRN. Fig. 13 shows the implementation of the PRN code generator in Simulink.

The Code Loop Discriminator was implemented as a normalized noncoherent discriminator according to the following equation:

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

Where I_E and I_L are the in-phase outputs of the early and late codes respectively. Q_E and Q_L are the outputs of the quadrature early and late codes respectively. Fig. 14 shows the implementation of that discriminator in Simulink.

The Code Loop Filter is the same as the Carrier Loop Filter except that the noise bandwidth (Bn) was taken to be 2Hz and the loop gain (K) was 1. The Integrate and Dump operation shown in Fig.8 is implemented as a summing block that sums all the samples values per C/A code.

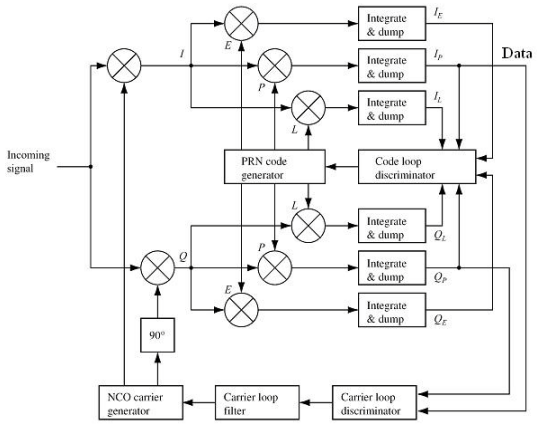

Fig. 8: A combined carrier and code tracking loop (complete tracking channel)

The real GPS signal that was used in verifying our models was saved in a binary file. The IncomingSignal block that is shown in Fig. 9 is responsible for feeding the tracking channel with the C/A codes. The number of samples to be fed each cycle from the file is calculated from the previous cycle and we start by 38192 samples. Reading data from that file was implemented by using the Level-2 M-file S-Function block as shown in Fig. 15.

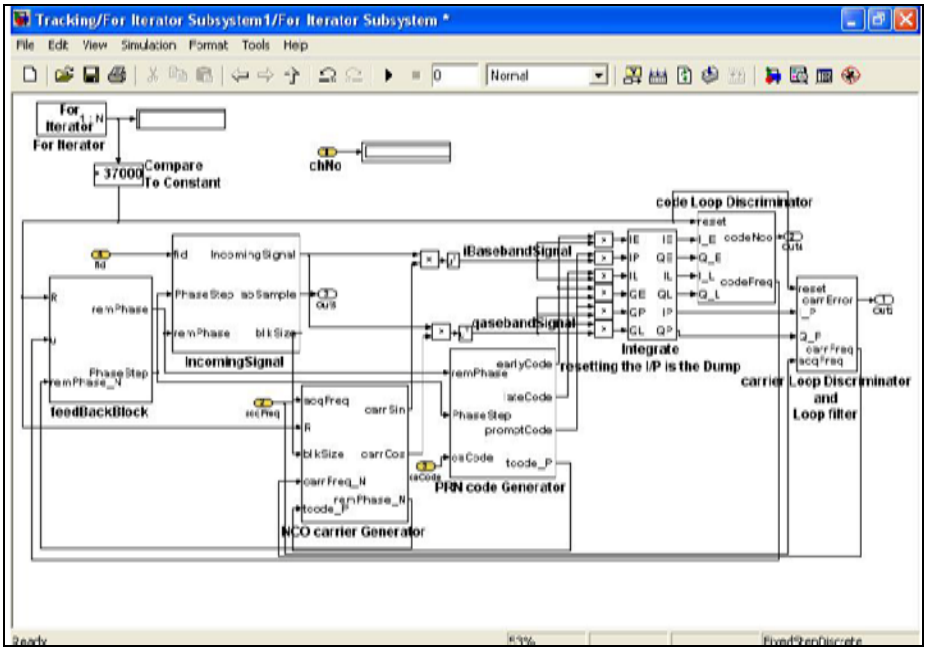

Fig. 9: The implementation of a complete tracking channel in Simulink

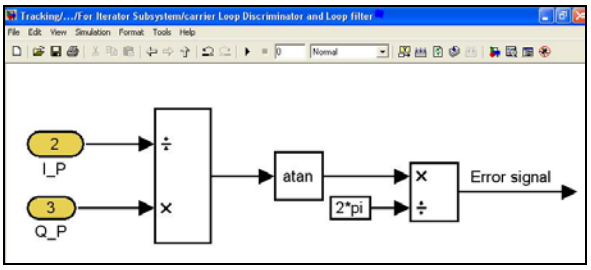

Fig. 10: The implementation of the carrier loop discriminator in Simulink

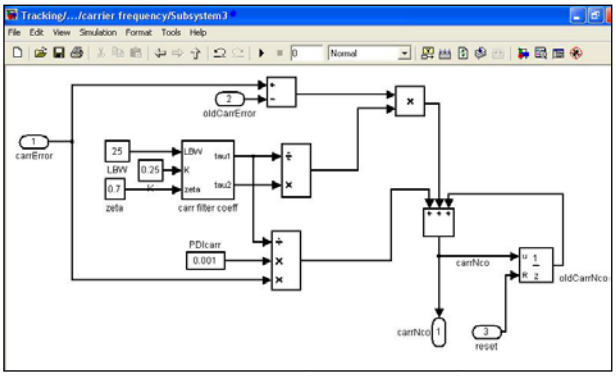

Fig. 11: The implementation of the carrier loop filter in Simulink

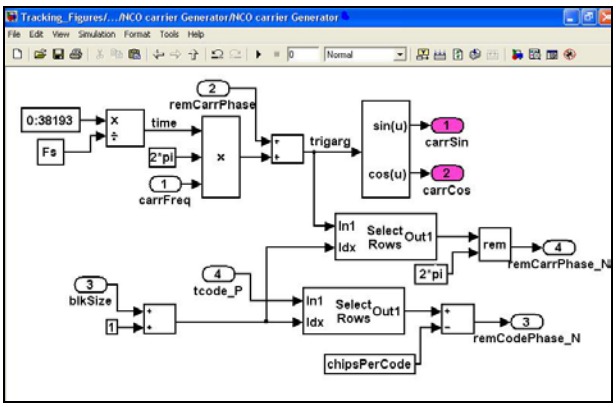

Fig. 12: Sampling the NCO output

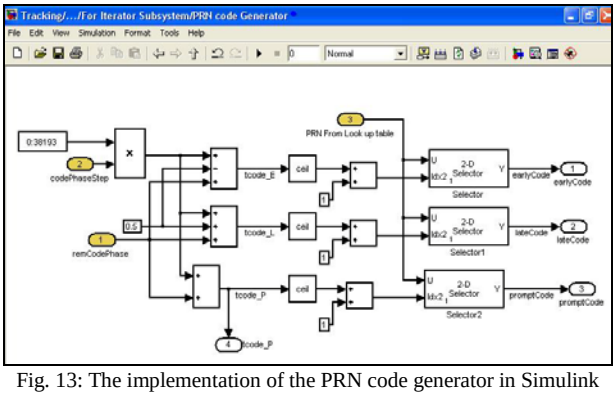

Fig. 13: The implementation of the PRN code generator in Simulink

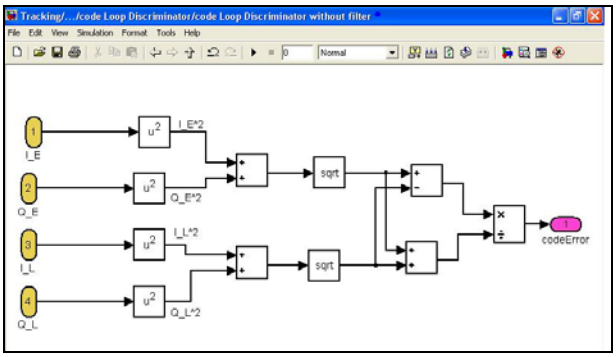

Fig. 14: The implementation of the code loop discriminator in Simulink

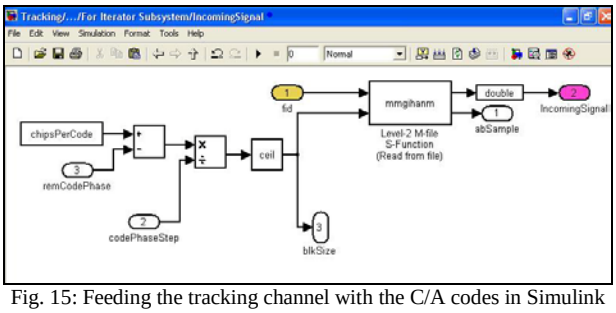

Fig. 15: Feeding the tracking channel with the C/A codes in Simulink

Tracking is done according to the original front-end main parameters, which are IF=9.548MHz and Fs=38.192MHz. Making tracking according to the modified IF and Fs will imply either resampling the data file that contains the incoming navigation message before starting tracking or resampling it step by step during tracking. In either of the two cases this will represent a time overhead added to the tracking given that to complete the tracking of only one page from the navigation message we need to process more than 36000 C/A codes.

Fig. 16a, 16b, and 16c show a part of the navigation data extracted from the tracking model for PRN number 3 after making acquisition using the original and modified parameters.

The data inversion for the same PRN that is shown in Fig. 16c returns to the ability of the carrier tracking loop (Costas loop) to track the signal with 180^{o} phase shift. This inversion doesn't affect the position solution because the navigation solution phase can distinguish between the data and its inverted version.

Implementation stage

To transform the Simulink model to a DSP implementable model that can use the RTW and TC6 to target a DSP we modified the following parts:

1- The file transfer section that is responsible for feeding the tracking algorithm with the required data size to be processed in each cycle was modified. The Level-2 M-file S-Function block

that is shown in Fig. 13 and the Level-2 M-file S-Function block that feeds the first one by the File Identifier (fid) were reconstructed as a C-file S-function. Because the "fopen", "fseek", "fread", and "ftell" m-functions are not permitted to use the RTW software.

2- The two multipliers that are used in wiping off the carrier in the I and Q branches are replaced by an Embedded MATLAB Function that do the same job.

The tracking phase isn't heavy as the acquisition phase, which is the heaviest and most time consuming phase in the GPS receiver. Tracking needed only less than 6.5MB of SDRAM. So, it could be implemented on the C6713DSP starter kit that has a maximum SDRAM size of 16MB.

The verification of the tracking algorithm that ran on the DSP starter kit was done through transferring the results through the RTDX channels from the target RAM to MATLAB. The results received from the DSP kit were the same as those obtained in simulation.

5 Navigation solution

In this phase the 50 Hz navigation bits are decoded according to ICD-GPS-200 (1991) to obtain the pseudorange, the receiver position, and the receiver clock offset. The navigation solution phase is simulated according to the block diagram shown in Fig. 17.

The first step in the navigation solution phase is the Bit Synchronization. In this step we find the time where bit transitions occur and then each 20ms are replaced by a single value that represent a navigation data bit. The second step is determining the beginning of each subframe that is consisting of an 8-bit long preamble and that is performed by correlating the tracking output with a locally generated preamble.

The third step is to determine the transmission time of the first detected subframe by the aid of decoded TOW word. The TOW corresponds to the transmission time of the next subframe. To obtain the transmission time of the first subframe we multiply the TOW by 6 and subtract 6s from the result. All the navigation bits are ready now to be decoded. The ephemeris parameters are decoded according to Table 2 and Table 3. Pseudorange calculations are accomplished on two steps: the first step is to find initial pseudoranges. The second step is to keep track of the initially calculated pseudoranges. The initial pseudoranges are used in calculating the receiver position (X Y Z) and the receiver clock offset (dt). The Least-Squares method is usually used in calculating the receiver position for ≥ 4 visible satellites.

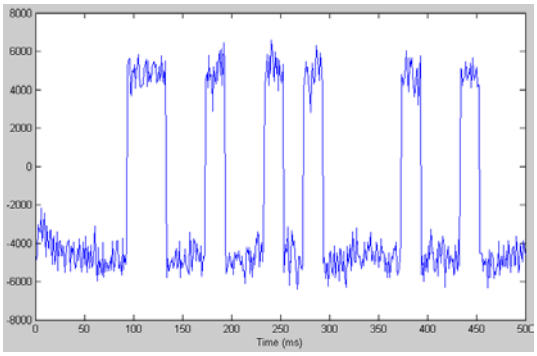

Fig. 16a: Part of the navigation data bits for PRN3 extracted after making acquisition by IF=9.548MHz and Fs=38.192MHz

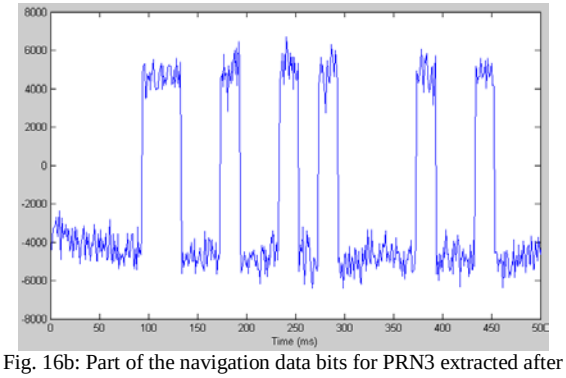

Fig. 16b: Part of the navigation data bits for PRN3 extracted after making acquisition by IF=2.046MHz and Fs=8.192MHz

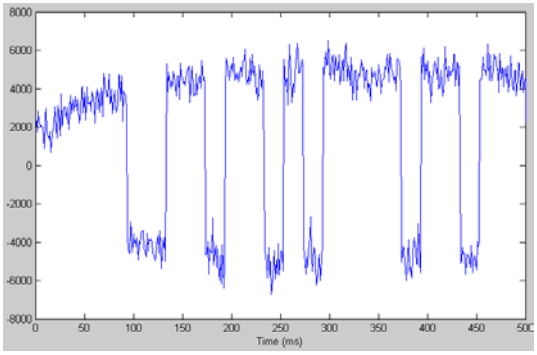

Fig. 16c: Part of the navigation data bits for PRN3 extracted after making acquisition by IF=2.046MHz and Fs=4.096MHz

IODE	issue of data, ephemeris
Δn	mean motion correction
μ_0	mean anomaly at t_{oe}
e	eccentricity
$\sqrt{a}$	square root of semi-major axis
t_{oe}	reference epoch of ephemeris
Ω_0	longitude of ascending node at t_{oe}
i_0	inclination at t_{oe}
ω	argument of perigee
$\dot{\Omega}$	rate of Ω_0
$\dot{i}$	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 ω

Table 2: The ephemeris parameters

Simulation stage

Although the navigation solution phase is the lightest phase in the GPS receiver; it is the most complicated phase. Building this phase using Simulink makes the compound and complicated calculations very clear and easy to follow. The Find Preamble and Pseudorange algorithms were built in Simulink as shown in Fig. 18 and Fig.19 respectively. Fig. 20 shows the results obtained for the receiver position and receiver clock offset.

Parameter	No. of bits	Scale factor (LSB)	Unit
IODE	8		
C_{rs}	16*	2^{-5}	m
Δn	16*	2^{-43}	semicircle/s
μ_0	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}	$m^{1/2}$
t_{oe}	16	2^{4}	s
C_{ic}	16*	2^{-29}	radian
Ω_0	32*	2^{-31}	semicircle
C_{is}	16*	2^{-29}	radian
i_0	32*	2^{-31}	semicircle
C_{rc}	16*	2^{-5}	m
ω	32*	2^{-31}	semicircle
$\dot{\Omega}$	24*	2^{-43}	semicircle/s
$\dot{i}$	14*	2^{-43}	semicircle/s

Table 3: The decoding scheme for the ephemeris parameters

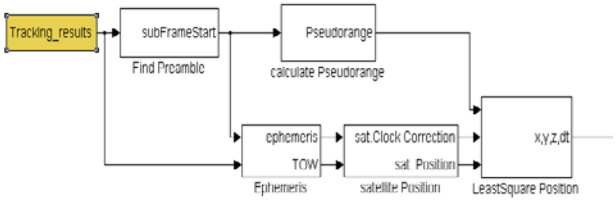

Fig. 17: Navigation solution block diagram

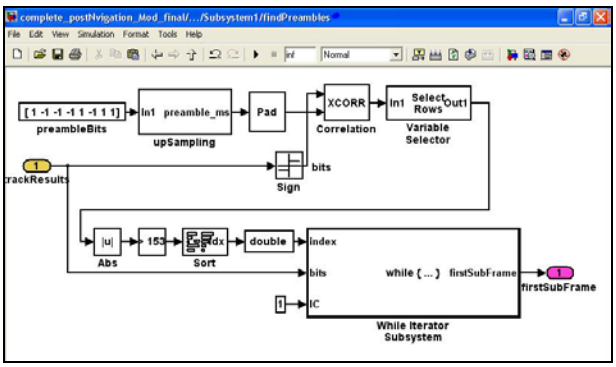

Fig. 18: The algorithm for finding the preamble in Simulink

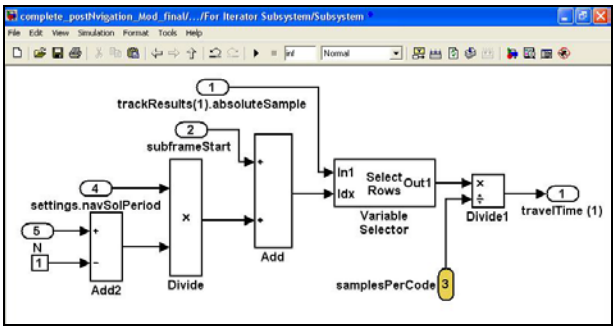

Fig. 19: The pseudorange calculations for one satellite in Simulink

Implementation stage

The navigation solution model could use the RTW and TC6 easily. But due to the large amount of the tracking results to be fed to the navigation solution algorithm, we couldn't run it on the C6713DSP starter kit and simulated its performance on the kit by using the C6713 Device Functional Simulator. The same results that are obtained from the Simulink model are obtained, through the RTDX channels, from the target Functional Simulator. We have to mention that the error in the navigation solution (with respect to that obtained from Borre-Akos algorithm) after making acquisition according to Fs=8.192MHz and IF=2.046MHz was as follows:

- The error in the x direction was less than 7.77e-13.
- The error in the y direction was less than 2.17e-13.
- The error in the z direction was less than 2.51e-13.
- The error in the receiver clock error (dt) was less than 1.34e-12.

The error in the navigation solution after making acquisition according to Fs=4.096MHz and IF=2.046MHz was as follows:
- The error in the x direction was less than 1.47e-05.
- The error in the y direction was less than 2.64e-06.
- The error in the z direction was less than 4.71e-06.
- The error in the receiver clock error (dt) was less than 4.12e-05.

6 Simulation and Implementation of a complete GPS receiver through Simulink

The Simulink models of acquisition, tracking, and navigation solution phases were combined in one model that represents the simulation stage of a complete GPS receiver. Constructing the receiver in such a way makes its architecture very clear and very easy to debug, modify, and separate any part of it and obtain immediate results.
The DSP Implementable models of acquisition, tracking, and navigation solution were also combined in one model. This model forms a complete GPS receiver that is ready, by just one click, to run on the C6713 Device Functional Simulator or any DSP kit that support Simulink and has enough RAM size.

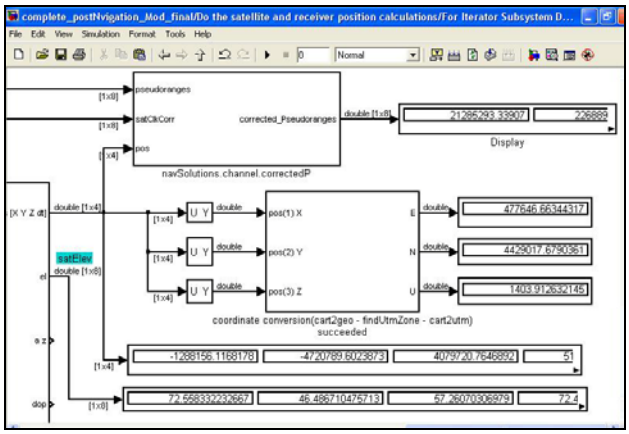

Fig. 20: The results obtained from the Navigation Solution model

7 Summary and Conclusion

In this article we introduced the simulation and implementation of a complete GPS receiver on a DSP through a graphical programming language, which is Simulink. Also we introduced the idea of modifying both the IF and Fs in the Simulink GPS receiver and the corresponding results can be easily analyzed. Using a graphical environment in both the simulation and implementation stages makes the designer mind dedicated most of the time in developing and enhancing the algorithm and less time on the coding. Moreover using a graphical programming language makes the algorithm very clear and can be easily modified and debugged. In general, the simulation and implementation of a complete GPS receiver through a graphical environment represent a new approach for the SDR technology.

References

K. Borre, D. Akos (2006), **A Software-Defined GPS and GALILEO Receiver – A Single-Frequency Approach**, Birkhauser Boston.

Akos, D. (1997), **A Software Radio Approach to Global Navigation Satellite System Receiver Design Approach,** Ph.D Dissertation, Ohio University.

Yi-Ran Sun. (2006), **Generalized Bandpass Sampling Receivers For Software Defined Radio,** Ph.D Dissertation, Royal Institute of Technology, Stockholm.

James Bao-Yen Tsui (2005), **Fundamentals Of Global Positioning System Receivers**, USA, John Wiley & Sons.

J. Starzyk and Z. Zhu, (2001), **Averaging Correlation for C/A code Acquisition and Tracking in Frequency Domain**, MWSCS, Fairborn, Ohio.

Kaplan, E. D. and C. J. Hegarty, (2006), **Understanding GPS: Principles and Applications, 2nd Edition**, Norwood

GPS Joint Program Office, (1995), **Global Positioning System Standard Positioning Service Signal Specification**.

Real-Time Workshop User's Guide, (http://www.mathworks.com/access/helpdesk/help/pdf_doc/rtw/rtw_ug.pdf)

Target Support Package TC6 3, User's Guide, (http://www.mathworks.com/access/helpdesk/help/pdf_doc/tic6000/tic6000.pdf)

Ziemer, R. E., Peterson, R. L. (1985), **Digital Communications and Spread Spectrum System**, Macmillan, New York.

E. Del Re, M. Ruggieri (2008), **Satellite Communications and Navigation Systems**, Springer Science+Business Media, LLC.

http://celestrak.com/GPS/icd200cw1234.pdf

SE4110L PointChargerTM GPS Receiver IC, Data Sheet.