FPGA Implementation of Tracking Phase of the GPS Receiver Using XSG

M. El Hawary[1, 2], G. G. Hamza[1], Abdelhaliem Zekry[2], and I. Motawie[1]
[1] National Institute of Standards (NIS), Giza, Egypt
[2] Faculty of Engineering, Ain Shams University, Cairo, Egypt
Email: hawary5000@yahoo.com; gihan_gomah@yahoo.com; aaazekry@hotmail.com; imotawie10@gmail.com

***Abstract*—**GNSS systems are globally available systems used for positioning and navigation. GNSS receivers are widely used as a standard frequency source for time and frequency measurements. So, a lot of efforts have been made in the design, simulation and real-time implementation of the GNSS receivers. The efforts that were made were to transform the construction of the GNSS receivers from hardware to software by using SDR technology. This facilitates debugging and redesigning these complicated systems. In this paper, the tracking phase of the GPS receiver was completely simulated and implemented on FPGA using the same platform and through a graphical programming language, which is the Xilinx System Generator. So, this paper proves that large systems can be simulated and prototyped through the same platform.

***Index Terms*—**Global Navigation Satellite System (GNSS), GPS receiver, tracking phase, Software Defined Radio (SDR), Xilinx System Generator (XSG), FPGA Implementation

I. Introduction

There are different GNSSs in the world. These GNSSs are coming from different continents where we have the American GNSS (the Global Positioning System (GPS)), the Russian GNSS (the Global Navigation Satellite System (GLONASS)), the European GNSS (GALILEO), the Chinese GNSS (BeiDou), and the India's which is called (IRNSS). These systems target two main applications; which are navigation and timing applications. These systems have the same main building blocks in both transmitters and receivers.

In general, GNSS receivers consist of four phases; which are the front-end phase, the acquisition phase, the tracking phase and the navigation solution phase. The acquisition phase is the most resources consuming phase due to the heavy processing needed in it. The tracking phase is the phase that contains the largest number of building blocks.

Usually, the implementation of either the systems that imply heavy processing or the big systems using the SDR technology is not a straightforward process.

Applying the SDR technology on the GNSS receivers had gone through many phases starting from using the high-level open-source programming languages (such as C++) to the graphical programming languages (such as Simulink). As an additional step forward the simulation platform was integrated with the prototyping platform (like the Xilinx System Generator (XSG)) to ease the implementation process [1]-[10]. This means that we move a step forward toward easier and more transparent designing, simulation and prototyping through the same environment.

The purpose of the prototyping tools, like the Xilinx System Generator (XSG), and embedding it with the simulation tool to make the hardware designer to use the same environment in both simulation and implementation; which allows easier developing, upgrading and improving the performance of the receiver. The authors proved before the effectiveness of this concept when applied to the implementation of the acquisition phase of a GPS receiver [11].

In this paper, the efficiency of integrating the simulation and prototyping tools in the implementation of the large systems was proved. In this paper, the Simulink model of the tracking phase that was previously introduced by G. Hamza and A. Zekry was taken as guidance in building the tracking phase using XSG and implementing it on FPGA**.**

As a knowledge transfer, all the problems encountered during the conversion process of the Simulink model of the tracking phase to XSG model will be stated and accompanied by the used solutions. For example, one of these problems was the incompatibility between the various software versions that were used during building the system. Several attempts have been made to try several versions of the Matlab program with different versions of the Xilinx ISE 14.7 program. Until we reached to use Matlab R2013a and Xilinx ISE 14.7 program operated in operating system windows 7.

In this paper, Xilinx ISE 14.7, Xilinx System Generator 14.7, and Xilinx Virtex-6 (xc6vcx240t-2 FF1156) evaluation kit were used in the prototyping process [12].

II. The Tracking Phase of the GPS Receiver

GPS receiver consists of four phases, which are; the front-end phase, the acquisition phase, the tracking phase, and the navigation solution phase. The front-end phase is responsible for receiving the RF signal and down-convert it to baseband signal. The acquisition phase gives an estimated value for both the carrier frequency and the

Manuscript received September 26, 2019; revised March 5, 2020.
Corresponding author email: hawary5000@yahoo.com.
doi:10.12720/jcm.15.4.367-371

code phase for each visible satellite. While the tracking phase of the GPS receiver gives the exact value of the carrier frequency and the code phase to keep tracking of the received signal properties over time. These values change with time due to the continuous movement of the GPS satellites. This produces a shift in frequency between transmitter and receiver. The shift in frequency called a Doppler frequency shift. This shift in frequency effects on the incoming signal in acquisition and tracking phases. If the shift in frequency in the GPS receiver is not corrected, this will lead to loss of tracking the GPS satellites and no navigation data will be received. So, it must be done alignment between the locally generated PRN codes and the received PRN codes.

The main function of the tracking phase is to refine the output of the acquisition phase. The output from the acquisition phase input to the tracking phase. Through the tracking phase, the navigation data can be extracted from the received signal by demodulating and dispread it.

The tracking phase consists of two main loops, which are; the code tracking loop and the carrier-tracking loop.

A carrier tracking loop is used to generate the carrier replica. While a code tracking loop is used to generate the PRN code replica. The code and carrier tracking loops should work uniformity with each other. The two loops are combined into one loop to reduce the number of multipliers and to take control of the generated carrier frequency.

The verification of the implementation of the tracking loop model in XSG is verified with the Simulink model in [2]. Fig. 1 shows the block diagram of the tracking loop.

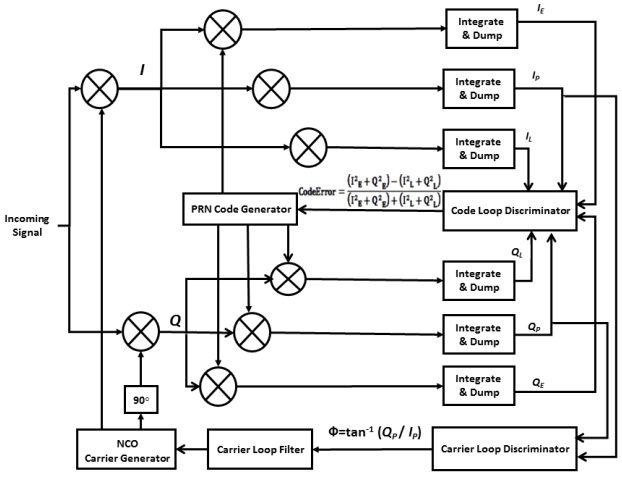

Fig. 1. Block diagram of a complete tracking loop of the GPS receiver.

A. Implimemtation of the Code Tracking Loop of the GPS Receiver

The code tracking loop is most implemented as a Delay Locked Loop (DLL). The function of the DLL is to generate three local codes (replicas) and correlated these replicas with the incoming signal. The DLL consists of a PRN code generator, code loop discriminator (code phase error), loop filter, local oscillator, six correlators and integrate and dump block. The purpose of this loop is to preserve the phase of the local generate PRN code stratify with the phase of the received signal. The nonconformity between the local generated PRN and the received PRN result from the influence of the Doppler frequency shift. So, use a Delay Locked Loop to prevent nonconformity between them. There are three replicas of the PRN code that are locally generated. These replicas called Early (E) PRN, Prompt (P) PRN and Late (L) PRN codes. The three replicas are correlated with the incoming signal. The incoming signal is multiplied with the local carrier signal and with the replicas of the PRN codes. After that, the result of multiplication is integrated and dump to getting the correlation values. These correlation values inform the correlation between the three replicas and the received PRN code. Fig. 2 shows the block diagram of the code tracking loop.

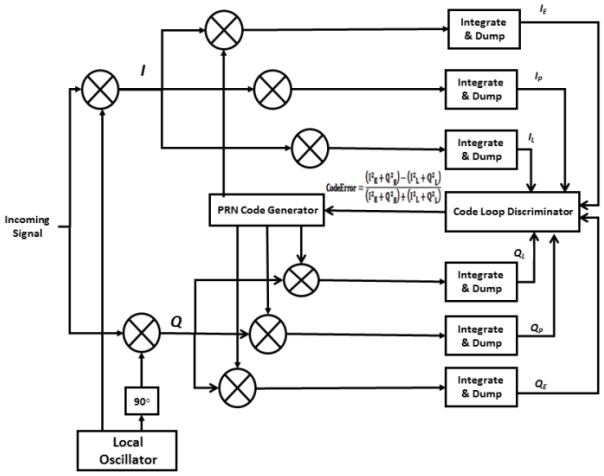

Fig. 2. Block diagram of the code tracking loop of the GPS receiver.

The incoming signal is the same signal that use in [2] and has the following parameters: sampling frequency Fs=38.192 MHz and Intermediate Frequency IF= 9.548 MHz. The following subsections show the implementation of the constriction blocks of the code tracking loop.

1) PRN code generator

The PRN code has a length of 1023 chips and a chipping rate of 1.023 MHz. The PRN code generator generates three code replicas. These replicas are separated by $\pm\frac{1}{2}$ chip. The P-code is separated from the E-code by $-\frac{1}{2}$ and from the L-code by $+\frac{1}{2}$. The three replicas are correlated with the incoming signal.

To get the correction on the code, it must determine it's late or prompt or early to follow its changes. If the incoming signal is in phase with the locally generated PRN code replica, the Early (E) and Late (L) values are equal and the local PRN code does not need any correction. If the local PRN code replica diverges in phase with the incoming PRN code, corrective action is generated.

For example, if the incoming PRN code phase is delayed from the local PRN code replica, the Early

correlation value becomes greater than the Late value. For that, a correction action must be taken. Fig. 3 shows the implementation of the PRN code generator in XSG. The three replicas of the PRN code are stored in RAM in XSG.

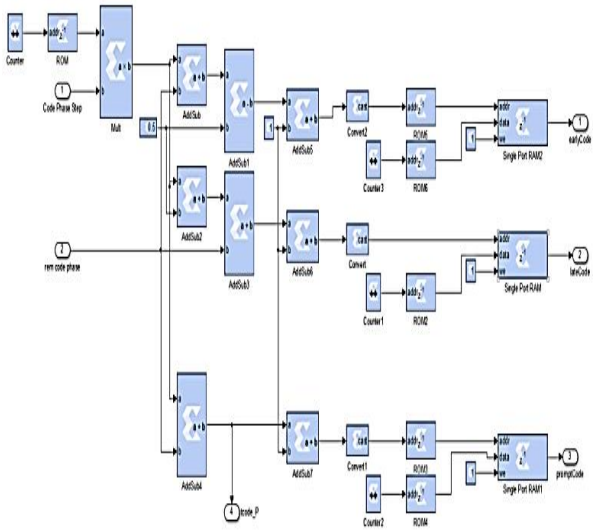

Fig. 3. Implementation of the PRN code generator in XSG.

2) Code loop discriminator

The purpose of the code loop discriminator is to make adjusting for code phase. It calculates the amount of error in the code replica and the direction early or late from the variation in the amplitudes of the early and late envelopes. This error is filtered and after that applied to the Numerically Controlled Oscillator (NCO), where the output frequency is increased or decreased as necessary to correct the replica code generator phase with respect to the incoming signal code phase.

This discriminator was implemented as non-coherent discriminator according to (1):

$$\text{CodeError} = \frac{(I^2_E+Q^2_E)-(I^2_L+Q^2_L)}{(I^2_E+Q^2_E)+(I^2_L+Q^2_L)} \quad (1)$$

where: I_E is the in phase output of the early code, I_L is the in phase output of the late code, Q_E is the quadrature output of the early code and Q_L is the quadrature output of the late code.

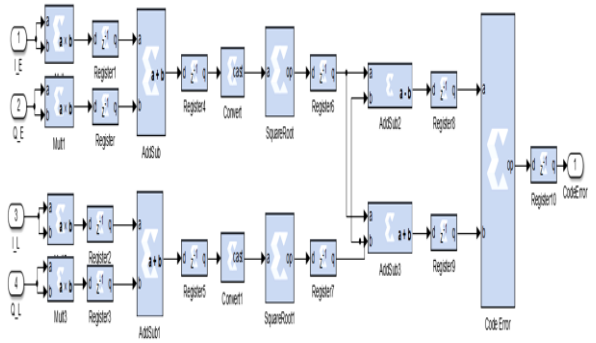

Fig. 4. Implementation of the code loop discriminator in XSG.

The implementation of code loop discriminator that represented in equation (1) in XSG shown in Fig. 4.

The code loop discriminator is consisting of a group of blocks of XSG which are; multiplication blocks, register blocks, square root blocks, add blocks and division block to get the code error.

3) Code loop filter

The function of the code loop filter is to decrease the noise in order to getting an accurate estimate of the original signal at its output. The code loop filter implemented as a second order filter which has the following parameters as shown in Table I. The order of the loop filter and the noise bandwidth determine the response of the filter to signal dynamics. The output from the loop filter is subtracted from the original signal to produce an error signal which is fed back into the filter input in a closed loop process. The implementation of code loop filter in XSG shown in Fig. 5.

TABLE I: PARAMETERS OF THE CODE LOOP FILTER

Parameter	Value
Damping ratio (ζ)	0.7
Noise Bandwidth (Bn)	2 Hz
Natural Frequency (Wn)	$\frac{8\zeta Bn}{4\zeta^2+1}$
Loop gain (K)	1
Filter Coefficients	$\tau_1 = \frac{K}{(Wn)^2}$ and $\tau_2 = \frac{2\zeta}{Wn}$

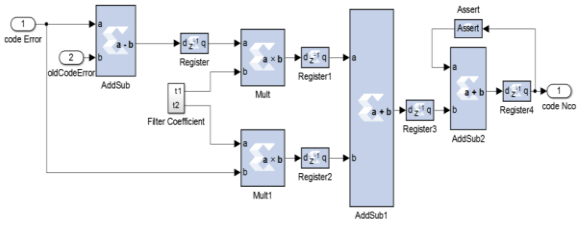

Fig. 5. Implementation of the code loop filter in XSG

B. Implimemtation of the Carrier Tracking Loop of the GPS Receiver

The carrier tracking loop is a Phase Locked Loop. It is consists of carrier generator, carrier loop discriminator (carrier phase error) and carrier loop filter to decrease the loop noise. The object of this loop is to generate a carrier frequency replica to that of the received signal. In this loop the incoming signal is multiplied with the local carrier signal to dispose of the carrier signal and the PRN code of the incoming signal. The output after multiplication is input to the carrier loop discriminator to determine the carrier phase error. The goal of the carrier loop filter is to filter the carrier phase error. Then the output from the filter is used as feedback to a NCO that generates a perfectly aligned carrier signal. A carrier frequency offset occurs when a signal arrives at a receiver with a different carrier frequency from the receiver's demodulation frequency. This occurs as a result of the Doppler frequency. The following subsections show the implementation of the constriction blocks of the code tracking loop.

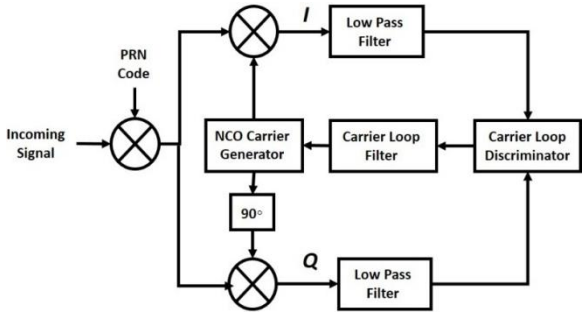

Fig. 6. Block diagram of carrier tracking loop.

The following subsections show the implementation of the constriction blocks of the code tracking loop.

1) Numerically Controlled Oscillator (NCO) carrier generator

NCO is a digital oscillator signal generator. It is use to generate the carrier signal. The function of the NCO carrier generator is to refined the value of the carrier frequency. Figure 7 shows the structure of NCO. The output frequency f_o of NCO is expressed as in (2) [11]:

$$f_o = \frac{\Delta\theta . f_S}{2\pi . 2^N} \quad (2)$$

where f_s is the sampling frequency, N is the word size of the phase accumulator and $\Delta\theta$ is the input phase increment. The input phase to the NCO is the carrier phase and the NCO phase of the previous sample. After that the loop will correct the phase error.

The NCO is implemented by using the Direct Digital Synthesizer (DDS) block in XSG to generate sine and cosine carriers. The implementation of NCO in XSG is showing in Fig. 8.

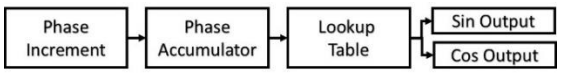

Fig. 7. Structure of NCO.

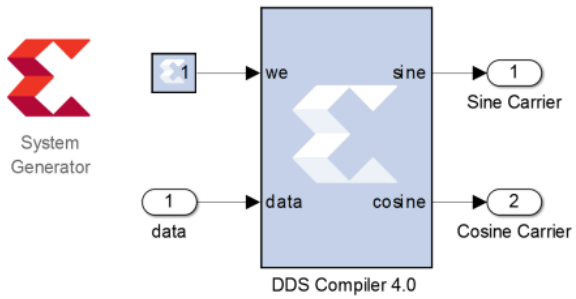

Fig. 8. Implementation of NCO in XSG.

2) Carrier loop discriminator

The carrier loop discriminator constructed as arctan discriminator according to (3).

$$\emptyset = \tan^{-1}(\frac{Q}{I}) \quad (3)$$

where: ∅is the phase error, Q is the quadrature signal of Costas loop and I is the in-phase signal of Costas loop. Fig. (9) shows the implementation of the carrier loop discriminator in XSG.

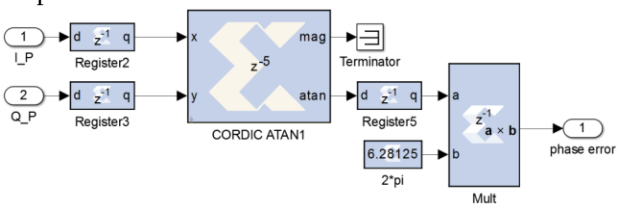

Fig. 9. Implementation of Carrier loop discriminator in XSG.

3) Carrier loop filter

After computing the phase error using the loop discriminator, it is filtered out by a carrier loop filter. The carrier loop filter implemented as a second order filter which have the following parameters that shown in table (II). Fig. 10 shows the implementation of the carrier loop filter in XSG.

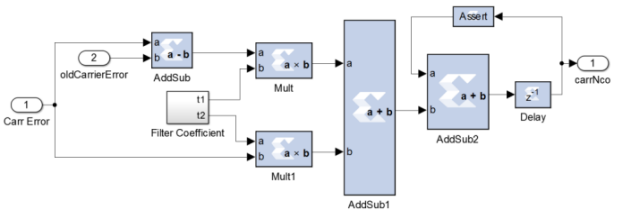

Fig. 10. Implementation of Carrier loop filter in XSG.

TABLE II: PARAMETERS OF THE CARRIER LOOP FILTER

Parameter	Value
Damping ratio (ζ)	0.7
Noise Bandwidth (Bn)	25 Hz
Natural Frequency (Wn)	$\frac{8\zeta Bn}{4\zeta^2+1}$
Loop gain (K)	0.25
Filter Coefficients	$\tau_1 = \frac{K}{(Wn)^2}$ and $\tau_2 = \frac{2\zeta}{Wn}$

III. IMPLIMENTATION OF A COMPLETE TRACKING LOOP OF THE GPS RECEIVER THROUGH XSG

The tracking phase of the GPS receiver is implemented using Xilinx virtex-6 evaluation kit. FPGA resources utilization summary of a complete tracking loop which indicates how many hardware components are used by the model is provided in Table III. It can be seen that the tracking loop model requires fewer hardware resources. Implementation results indicate that the tracking loop model in XSG has a simple design and avoids complexities. The summery of timing is provided in Table IV which display the total routing time is 1.223 ns, the total logic time is 1.643 ns and the total time is 2.86 ns.

TABLE III: SUMMARY OF THE FPGA RESOURCE UTILIZATION

	Used	Utilization
Number of Slice Register	568	1%
Number of Slice LUTs	1018	1%
Number of fully used LUT-FF pairs	380	32%
Number of DSP48E1s	28	3%

TABLE IV: SUMMARY OF TIMING

Clock frequency	348.949 MHz
Total Routing	1.223 ns
Total Logic	1.643 ns
Total Time	2.86 ns

IV. CONCLUSIONS

This paper presented the simulation and implementation of the tracking phase of the GPS receiver on FPGA through the Xilinx System Generator. The same platform is used in both the simulation and implementation. This means that we moved a step forward toward easier and more transparent designing, simulation and prototyping of the tracking phase of the GPS receiver through the same environment. The output result from the Xilinx System Generator model and the output from the Simulink model are close to each other. This expresses the successful conversion processing of the Simulink blocks to the Xilinx hardware blocks. Through the output from the tracking phase, navigation data can be extracted. Also, our future work is to use the output from the tracking phase to generate a timing pulse signal.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

REFERENCES

[1] K. Borre and D. Akos, "A software-defined GPS and GALILEO receiver – A single-frequency approach," *Birkhauser, New York*, 2006.

[2] G. Hamza, A. Zekry, and I. Motawie, "Implementation of a complete GPS receiver using simulink," *IEEE Circits Syst. Mag.*, 2009.

[3] M. E. Hawary, G. Gomah, A. Zekry, and I. Hafez, "Simulation of the E1 and E6 galileo signals using simulink," *Int. J. Comput. Appl.*, vol. 88, 2014.

[4] M. ElHawary, "Signal simulator for global navigation satellite system," Ain Shams University, 2014.

[5] G. G. Hamza, A. A. Zekry, *et al.*, "Implementation of a complete GPS receiver on the C6713 DSP through simulink," *J. Glob. Position. Syst.*, vol. 8, 2009.

[6] G. G. Hamza, "Enhancing the time measurement accuracy of GPS receiver," Ain Shams University, 2009.

[7] E. Schmidt, D. Akopian, and D. J. Pack, "Development of a real-time SDR GPS receiver exploiting a LabVIEW based instrumentation environment," *IEEE Trans. Instrum. Meas.*, vol. 99, pp. 1-15, 2018.

[8] F. S. Larosa, "Design, simulation, implementation and testing of search and tracking modules for a FPGA-Based GPS receiver," in *Proc. IEEE X Southern Conference on Programmable Logic (SPL)*, 2019.

[9] A. Manaandhar, "FPGA based tracking system for GNSS receivers," The Arctic University of Norway, 2017.

[10] I. Baumann, "Traceable time from GNSS signals," *Inside GNSS*, 2017.

[11] M. E. Hawary, G. G. Hamza, A. Zekry, and I. Motawie "FPGA implementation of the acquisition phase of the GPS receiver using XSG," *IJET*, 2019.

[12] System Generator for DSP Reference Guide, UG638, 14.5. March 2013.

Mohamed El Hawary is a researcher assistant in the National Institute of Standards (NIS), Egypt. He received his MSc degrees in Communications and Electronics from Faculty of Engineering, Ain Shams University, Egypt. He is a Ph.D. student at Faculty of Engineering, Ain Shams University, Cairo, Egypt. His research interests include GNSS receivers and FPGA.

Gihan G. Hamza received her BSc, MSc, and Ph.D. degrees in Communications and Electronics from Ain Shams University, Faculty of Engineering, Egypt. Now she is an Assoc. Prof. at the National Institute for Standards (NIS), Egypt. She is interested in the time and frequency dissemination through using GPS receivers.

Abdelhalim Zekry is a professor of electronics at faculty of Engineering, Ain Shams University, Egypt. He worked as a staff member on several universities. He published more than 300 papers. He also supervised more than 104 Master thesis and 28 Doctorate. Prof. Zekry focuses his research programs on the field of microelectronics and electronic applications including communications and photovoltaics. He got several prizes for his outstanding research and teaching performance.

Ibrahim Motawie was graduated from Cairo University Egypt. He was offered the MSc degree from Ain Shams University, Egypt. He got his Ph.D. at 1979 from Paul Seharie University, France. Now, he is a professor at the National Institute for Standards (NIS). He is interested in the time and frequency metrology and its applications