jet 933INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2019, VOL. 65, NO. 4, PP. 739-744
Manuscript received July 7, 2019; revised September, 2019. DOI: 10.24425/ijet.2019.130257

# FPGA Implemetation of Acquisition Phase of the GPS Receiver Using XSG

Mohamed Ibrahiem El Hawary, Gihan Gomah Hamza, Abdelhalim Zekry, and Ibrahiem Mohamed Motawie

***Abstract*—In the past it was usual to exert a huge effort in the design, simulation, and the real time implementation of the complicated electronic and communication systems, like GNSS receivers. The complexity of the system algorithms combined with the complexity of the available tools created a system that is difficult to track down for debugging or for redesign. So, the simulation and educational tools was different from the prototyping tools. In this paper the parallel search acquisition phase of a GPS receiver was simulated and implemented on FPGA using the same platform and through a graphical programming language. So this paper introduces the fruit of integrating the prototyping tools with the simulation tools as a single platform through which the complicated electronic systems can be simulated and prototyped.**

***Keywords*—GPS receiver, acquisition phase, SDR, Xilinx System Generator XSG, FPGA implementation**

## I. Introduction

FOR the time being the Global Navigation Satellite systems (GNSS) are used in many important and vital applications. Initially these systems were being dedicated for military applications. When they became available for the civilian community starting from 1980, they had a reduced accuracy compared to the accuracy available to the military applications. At 1990 the service operator, the US department of defense, started to add intentionally a noise to the GPS signals that are available for the civilian community. This action was denoted by the Selective Availability (SA). In May 2000 the SA was turned off and the accuracy of the positioning and timing services were considerably improved [1]. Turning off the SA Opened the door for more applications the GNSS receivers can be imbedded in. Beside the navigation and timing services, the GNSS receivers are used also, for example, in safety applications. So, today the GNSS are irreplaceable.

GNSS receivers are complicated electronic systems that have to pass through many stages of signal processing to accomplish their missions. The importance of using such systems in many daily and vital applications made the scientists to continue working on simplifying the design and implementation through trying different platforms. The emergence of the Software Defined Radio (SDR) technology was a breakthrough in simplifying the design and implementation of the complicated communications systems. Applying the SDR techniques on the GNSS receivers had passed through many forms.

The high-level open source programming languages, such as C++, were used in building the GNSS receivers. The complexity of the system combined with the complexity of the tool (C++ language) created a system that is difficult to track down for debugging or for redesign. This, for sure, affects the rate of developing such systems. It is noteworthy to mention that Clifford Killy and Douglas Baker initiated at 1995 an open source GPS project through which they designed a GPS receiver by using the C programming language [2].

On the way toward more simplification for such systems simpler tools and platforms were used such as MATLAB m-Language. Although both the m-language and C-language are classified as a text programming language, there is a substantial difference between them. The main advantage of the C++ is that it is based on a compiler that maps the whole algorithm to the machine language in one step but the m-language is an interpreter based language that translates the code step by step. So C++ is more complex but more efficient than Matlab when it is used in building embedded systems. So Matlab was initially used in only the simulation of the GNSS receivers and to implement that design one has to move to another platform. Kai Borre and Dennis Akos introduced a book titled "A Software Defined GPS and Galileo Receiver A Single - Frequency Approach" through which they introduced the theoretical background behind GPS receivers. Then they applied it on the simulation of a complete GPS receiver using the m-language [2]. They wrote about 39 m-files for all the 3 stages; acquisition, tracking, and navigation solution, of the GPS receiver. In general using a text programming language in simulating large systems, even if an easy language, it has drawbacks. The large number of m-files makes the system not transparent enough. This has an impact on the difficulties in the redesign and debugging errors [3].

It is well known that the graphical programming languages, like SIMULINK, are easier and more transparent than the text programming languages [4] [5]. Although Simulink is more suitable for the top level functions while Matlab is better for the low level functions, G.Hamza and A.Zekry succeeded in converting the 39 m-files wrote by Kai Borre and Dennis Akos(et al) to a Simulink model that represents a complete GPS receiver [6]. After that the code generation tools accompanied to Simulink was exploited in generating the C-code for that design to implement it on a DSP for the educational purposes [7] [8]. If it is required for that design to be real time implemented then both the acquisition and tracking phases should be implemented on FPGA due to the heavy processing in these stages.

A. Zekry is with Electronics and Electrical Communications Engineering Department, ASU University, Cairo, Egypt (e-mail: aaazekry@hotmail.com).

M. ELHawary, G. G. Hamza and I. Motawie are with Time and Frequency Department, National Institute of Standards (NIS), Cairo, Egypt (e-mail: hawary5000@yahoo.com,, gihan_gomah@yahoo.com, imotawie10@gmail.com).

At 2015 Didier Siboniyo designed, simulated and implemented the acquisition phase of a GPS receiver using the serial search acquisition technique through SIMULINK. The code generation tools inside SIMULINK, the HDL coder, enabled him from converting the simulated model to VHDL code that could be implemented on FPGA [9].

The advent of the prototyping tools, like the Xilinx System Generator (XSG), and embedding it with the simulation tool make the hardware designer to use the same environment in both simulation and the real time implementation.

In this paper, the Simulink model of the acquisition phase using the parallel search that is previously introduced by G.Hamza and A.Zekry was taken as a guidance in building the acquisition phase using XSG and implementing it on FPGA**.** This means that we move a step forward toward easier and more transparent designing, simulation and prototyping through the same environment.

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 [10:14].

This paper arranged as follows: Section II will describe the implementation of the acquisition phase of the GPS receiver which is divided into seven subsections. Subsection A will explain the interface between the Simulink blocks and the Xilinx System Generator model to feed the GPS signal to Xilinx System Generator model. Subsections B, C and D will show the implementation of correlation stage, peak detection stage and fine detection stage of the acquisition phase, respectively. Subsection E will display the implementation of a complete acquisition phase at sampling frequency 38.192 MH. Subsection F will focus on the implementation of a complete acquisition phase at sampling frequency 8.192 MHz. Subsection G will be devoted to a comparison between the two acquisition systems at sampling frequency of 32.768 MHz and 8.192 MHz. Section III concludes the paper.

## II. THE ACQUISITION PHASE OF THE GPS RECEIVER

The main function of the acquisition phase of the GPS receiver is to detect the visible satellites and estimate roughly both the code phase and the carrier frequency of the visible satellites. Acquisition has three standard methods which are: The serial search acquisition, the parallel frequency space search acquisition and the parallel code phase space search acquisition. The first two methods are implemented in ASIC because of the complex of the floating Fast Fourier Transform that used in the technique. While the third method used in Software Defined Radio (SDR) technology [2] [15].

So, the parallel code phase space search technique is used to implement the acquisition phase of GPS receiver as shown in figure 1.The parallel code phase space search technique searches for 29 frequency bins for each satellite in 14 KHz search band with 500Hz frequency step.

The acquisition phase of the GPS receiver has been implemented using Xilinx System Generator by dividing it into three stages as shown in Fig.1. The three stages are; the correlation stage, the peak detection stage and the fine detection stage. The incoming signal will fed the correlation stage from the collected data of the GPS signal in Simulink to Xilinx System Generator blocks through using interface Simulink blocks. These interface blocks will be displayed in detail in section A. The goal of fragmentation is to determine the implemented resource area of each stage. Also, to compare the results output from Xilinx System Generator with the output results that comes out from the Simulink model in [3] for validation. The output of our stages is verified with the corresponding output results of the simulated model in [3] and [6]. Then the three stages are combined to one Xilinx System Generator model. The compilation in Xilinx System Generator converts the Xilinx System Generator model to a hardware / software co-simulation block. This block represents an FPGA chip used in SIMULINK model to verify the output results from Xilinx System Generator and the output from this new block which is running in Xilinx Virtex-6 evaluation kit. In the following subsections the implementation of the three stages will be introduced. Then connecting the three stages will construct a complete acquisition phase that is implemented in Xilinx System Generator.

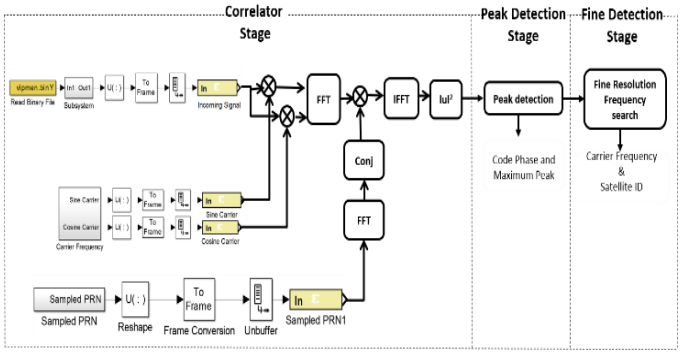


Fig. 1 Block diagram for the Implementation of Acquisition phase in Xilinx System Generator.

### *A. Feeding the Xilinx System Generator model with the GPS signal.*

The incoming signal from the front end of GPS receiver is the same collected data used in references [2] and [3]. These collected data has a sampling frequency of 38.192 MHz and Intermediate Frequency (IF) of 9.548MHz. Before inputting these data to the Xilinx System Generator its sampling frequency is reduced from 38.192 MHz to 32.768 MHz using the method in [3]. After that, this data is input through the gateway in block of the Xilinx System Generator. But we need to make interface before inputting this incoming signal from the SIMULINK blocks to the gateway in block of the Xilinx System Generator blocks. That's because the input signal is M-by-N array that is not consistent with Xilinx System Generator blocks. It deals with the data in serial format. So, one has to add some SIMULINK blocks before the Xilinx System Generator blocks to make an interface between them as depicted in Fig. 2. These interface blocks are added to carry out the following tasks: input the incoming GPS signal, add the sine carrier and cosine carrier as well as generate the sampled PRN. The incoming GPS signal is loaded from a binary file in Simulink, while the sine carrier, the cosine carrier and the PRN are stored in matlab workspace. The SIMULINK blocks added are: reshape block which converts the two dimensional matrix to one dimensional matrix for serial format, frame conversion block which converts the output from reshape block to frame and the last block is the unbuffer block which converts the input frame to sequence of scalar output. Then the output from the unbuffer block is input

to the Xilinx gateway in block which converts data received from Simulink in the floating point format to fixed point format which can be used inside the hardware system modeled using Xilinx System Generator. Also, in case of output data from the Xilinx System Generator a gateway output block is used to convert the output data from Xilinx System Generator model form of fixed point format to floating point format to show it in SIMULINK. Figure 2 shows the detailed block diagram of the input interface block from Simulink to the system generator. Figure 3 shows the histogram of collected 1048576 signal samples in Simulink and Fig. 4 shows the histogram of same collected 1048576 signal samples in Xilinx System Generator after the Xilinx gateway in block.

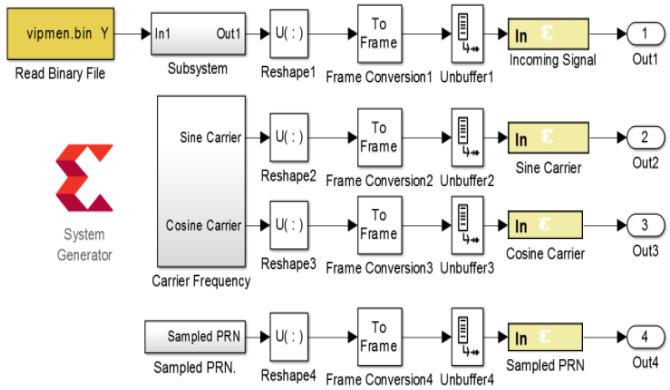


Fig. 2 Input data from MATLAB/SIMULINK to Xilinx System Generator model.

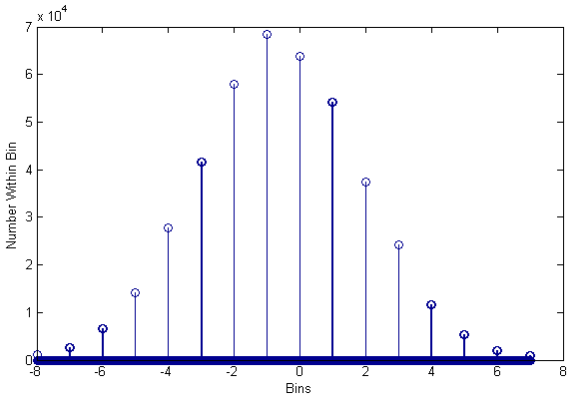


Fig. 3. Histogram of collected 1048576 samples in Simulink.

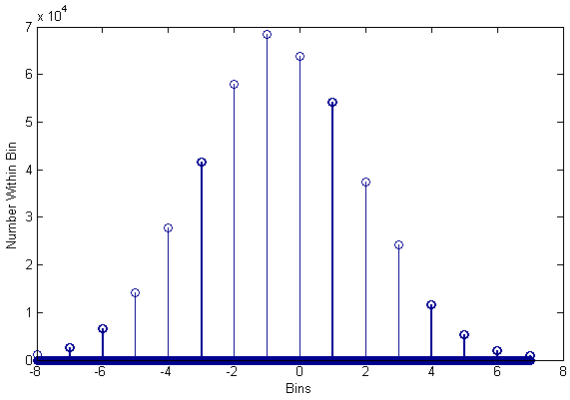


Fig. 4 Histogram of collected 1,048,576 samples inXilinx System Generator at out1.

### *B. Implementation of the correlator stage of the acquisition through Xilinx System Generator*

The purpose of this stage is to obtainment the absolute value that result from the correlation process. Detection the absolute value of the acquisition signal to determine the visible and nonvisible satellites. The implementation block diagram utilizing the Xilinx system generator of the correlator block and its corresponding testing Hardware Co-Simulation JTAG is given in Fig. 5. First, we apply the incoming signal and multiply it with a locally generated sine and cosine carrier signals which are stored in the MATLAB workspace. This gives two signal components, the in phase component, I-signal and the Quadrature component, Q-signal. Fast Fourier Transform is then performed on I and Q components to transfer them to frequency domain. The Xilinx FFT block has two inputs components which are the real component and the imaginary component. The I signal component is input to the real component of the Xilinx FFT block while the Q component is input to the imaginary component in Xilinx FFT block. The transform size of the FFT is 32768 samples. The configuration selected for the FFT is the pipelined streaming input/output option in Xilinx FFT block. The FFT can be considered the bottleneck for the GPS receiver because it consumes long time to load the data and processing them. So, the larger time consumed in the GPS receiver occurs in the acquisition phase. This consumed time depends on the length of FFT. Also, the increase in the length of the FFT will lead to increase in the hardware resources.

The function of the PRN generator is to generate the C/A code for each GPS satellite. There are 32 different C/A codes for the 32 GPS satellites. The C/A code is unique for each satellite. Each C/A code has a length of 1023 chips. The sampling frequency that is used is 32.768 MHz. So, the generated PRN codes are upsampled from 1023 samples to 32768 samples. There are two ways that can be used to perform the FFT of PRN code and conjugate the output.

The first way is to store the generated PRN codes in a lookup table. Then, perform the FFT for the PRN code to transform it to frequency domain and perform the conjugate operation on the FFT output. The other way is to perform the FFT of PRN code and conjugate it in SIMULINK and after that store the output result in RAM of FPGA. After that the outputs from the two FFT blocks are multiplied and the multiplication result is input to the inverse Fast Fourier Transform block (IFFT) to transform it to time domain. The output$u$of IFFT block is a complex signal and has R and I components. The usual method that used to detect whether the satellite is visible or invisible is to calculate the absolute value of correlation as in (1) and compare it with a threshold value [16].

$$|u|^2 = R^2 + I^2 \qquad (1)$$

Then the output values are stored in memory. After implementing the correlator stage using Xilinx System Generator, one chooses a Hardware Co-Simulation from compilation menu in the system generator block configuration to generate the model of Hardware Co-Simulation and test it through the virtex-6 Evaluation kit by using the JTAG cable. This cable is connected between the Evaluation kit and the host computer. The lower section of Fig. 5 shows the hardware co-Simulation block of the correlator stage.

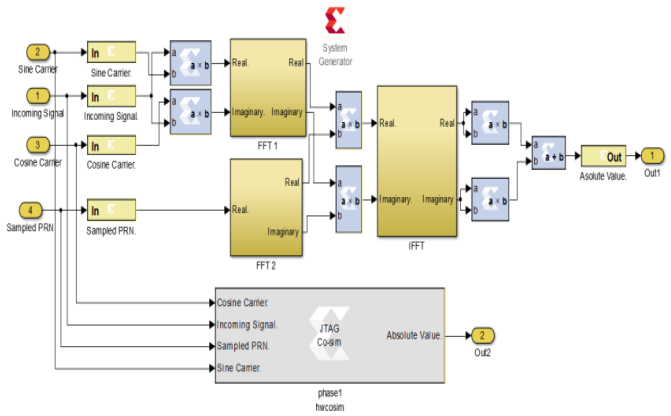


Fig. 5 Implementation of the correlator stage of acquisition phase using JTAG hardware CO-Simulation.

### *C. Implementation of the peak detection stage of acquisition through Xilinx System Generator*

The absolute value of the correlator stage is stored in memory for feeding to the peak detection stage. The purpose of this stage is to determine the maximum peak and the code phase of the detected signal. The technique used to determine the detection of the maximum peak and the code phase is the binary tree based logic. The stored samples stored in memory are compared to each other. Each sample is compared with the adjacent sample until getting the maximum amplitude and its index [17] [18] [19].

Figure 6 shows the implementation of the peak detection stage of the acquisition phase in the Xilinx System Generator and its corresponding testing Hardware Co-Simulation JTAG that connecting the outputs to the workspace. Figure 7 shows the acquisition results from the Xilinx System Generator for one of the visible satellites which have a peak value exceeding a threshold level of 2.5. This means that the ratio between the peak size and the second peak size must be greater than the value of 2.5. Figure 8 shows the acquisition results from the Xilinx System Generator for nonvisible satellite having values less than the predefined threshold value. This means that the ratio between the peak size and the second peak size less than the value of 2.5.

TABLE I
THE CODE PHASE OBTAINED FROM THE SIMULINK MODEL AND XILINX MODEL.

| Satellite ID | Code Phase from SIMULINK Model | Code Phase from Xilinx System Generator Model |
|---|---|---|
| **21** | 13404 | 13411 |
| **22** | 6288 | 6298 |
| **15** | 36321 | 36332 |
| **18** | 20725 | 20735 |
| **9** | 4696 | 4706 |
| **26** | 26826 | 26836 |
| **32** | 1489 | 1499 |
| **6** | 28202 | 28211 |

Table I shows the output code phase from the SIMULINK model and also the output code phase from the Xilinx System Generator model. The satellite 21 as it is in table I has a code phase of 13404 that output from the Simulink model and has a code phase of 13411 that output from the Xilinx System Generator model. The difference in the code phase between the two models is small. This means that the results obtained from the output code phase of the Xilinx System Generator model is close to the output code phase that output from the Simulink model. This difference, as a result of using hardware blocks instead of Simulink blocks.

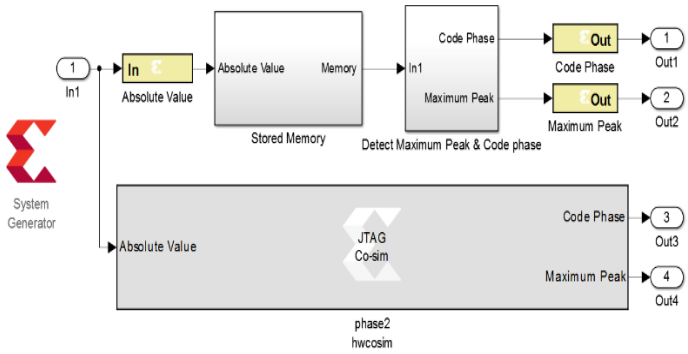


Fig. 6 Implementation of the peak detection stage of acquisition phase using JTAG hardware CO-Simulation.

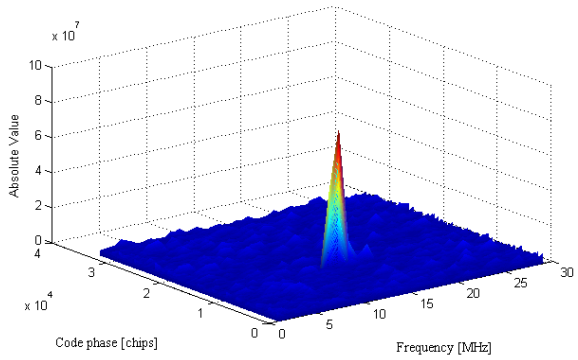


Fig. 7 Acquisition plot for a visible satellite.

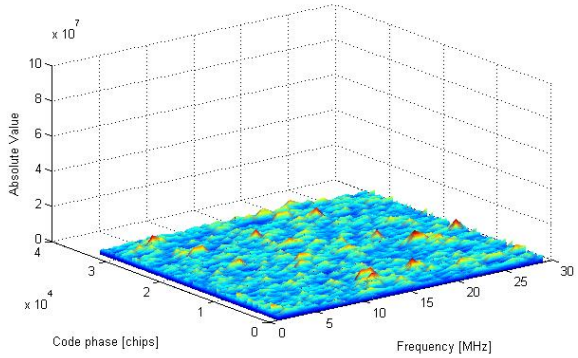


Fig. 8 Acquisition plot for nonvisible satellite.

### *D. Implementation of the fine detection stage in Xilinx System Generator*

The maximum value and its index that output from the peak detection stage is fed to the fine detection stage. The goal of this stage is to determine the value of the carrier frequency. In this stage the maximum value is compared to a predetermined threshold value. If the maximum value exceeds the threshold value, the satellite is considered visible. If not, then repeat the correlation and peak detection while inserting new PRN code. Figure 9 shows the implementation of the fine detection stage in the Xilinx System Generator and its corresponding testing Hardware Co-Simulation JTAG.

Table II shows the output results of the satellite number and its carrier frequency for SIMULINK model and for Xilinx System Generator model. From Table II it is clear that the output carrier frequency of the satellite ID 21 from the Xilinx System Generator model is 9.547420 MHz and also, the carrier frequency that output from the Simulink model is 9.547429 MHz. This difference between them is about 9 Hz. Also, as the case of the satellite ID 22 the difference between the carrier frequency that output from the Xilinx System Generator and the output from the Simulink model is about 9 Hz. This result from transforming the Simulink blocks to hardware blocks.

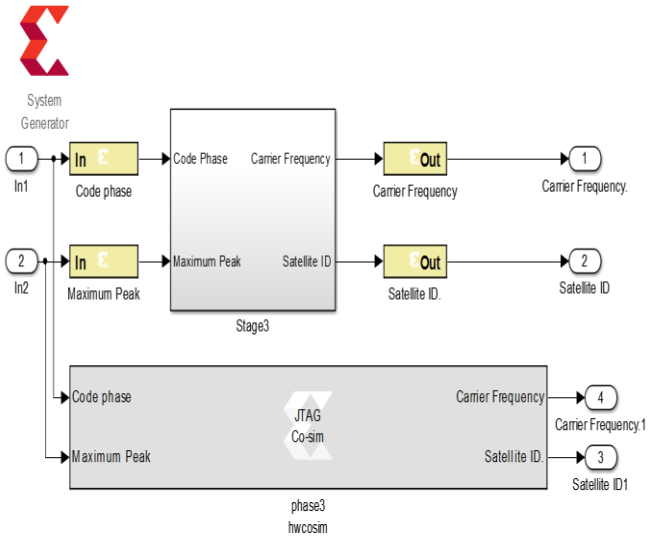


Fig. 9 Implementation of the fine detection stage using JTAG hardware Co-Simulation.

TABLE II

THE RESULTS OBTAINED FROM THE SIMULINK MODEL AND THE XILINX SYSTEM GENERATOR MODEL

| Satellite ID | Code Phase from SIMULINK Model | Code Phase from Xilinx System Generator Model |
|---|---|---|
| **21** | 9.547429 MHz | 9.547420 MHz |
| **22** | 9.549695 MHz | 9.549686 MHz |
| **15** | 9.549921 MHz | 9.549912 MHz |
| **18** | 9.548250 MHz | 9.548241 MHz |
| **9** | 9.550843 MHz | 9.550834 MHz |
| **26** | 9.545015 MHz | 9.545006 MHz |
| **32** | 9.316436 MHz | 9.316427 MHz |

### *E. Implementation of a complete Acquisition phase in Xilinx System Generator with sampling frequency =32.768 MHz*

Now, the correlator, peak detection and fine detection stages of the acquisition phase are combined into one stage as shown in Fig. 10 and Fig. 11 to test the whole system operation.

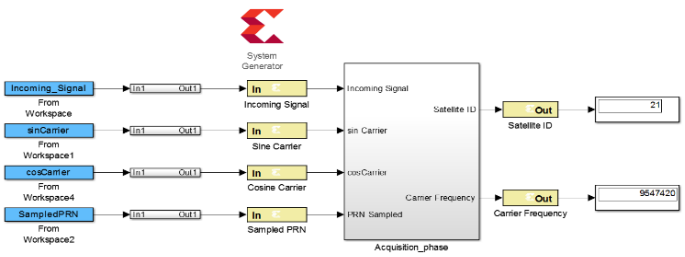


Fig. 10 Implementation of the complete acquisition phase in Xilinx System Generator.

Table III depicts the results from the Xilinx System Generator model of the whole system. One sees that the results from the Xilinx System Generator model agree well with that of the SIMULINK model. The output from this complete acquisition phase will be input to the tracking phase of the GPS receiver.

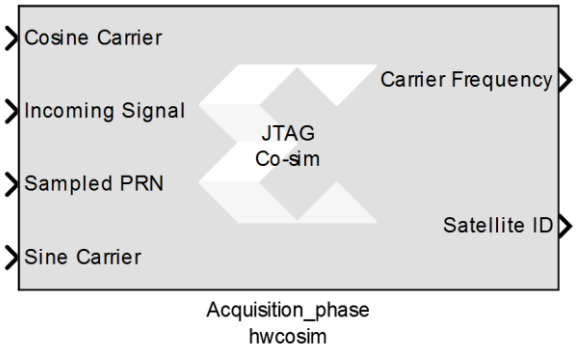


Fig. 11 Implementation of complete Acquisition phase using JTAG hardware Co-Simulation.

### *F. Implementation of a complete Acquisition phase in Xilinx System Generator model at sampling frequency =8.192 MHz and IF=2.046 MHz*

The aim of this section is to display the implementation of the acquisition phase of the GPS receiver using sampling frequency 8.192 MHz. The length of the FFT and IFFT are reduced to 8192 samples. Also, the sampled PRNcode is sampled by sampling frequency 8.192 MHz. So, the numbers of samples in C\A code are 8192 samples. This model also is divided into three stages as the previous model. After that, the three stages are connected to one model. The output results of this model are close to the output result of the previous model that use a sampling frequency of 32.768 MHz.

### *G. Comparison between the two acquisition systems at sampling frequency of 32.768 MHz and 8.192 MHz*

This section introduces a comparison between the two models with different sampling frequencies. The comparison contains the following items: the resources of the hardware implementation of the two systems and their processing time. In contrast to digital signal processing, FPGAs implementations are genuinely parallel in nature. So, various processing operations do not have to contain the same resources. Each freelance processing function is specified to a dedicated part of the chip. Also, it can work self sufficient with no impact from other logic blocks. Therefore, the execution of one part of the application isn't influenced when adding more processing operations.

Table III and table IV depict the hardware resources utilized in the FPGA implementations of the two designs. From the table III and table IV, it is illustrated that the hardware resources decrease with reduction the sampling frequency. Because the number of samples that is used is decreased.

Table V and table VI shows the processing times in the two designs. The design with higher sampling frequency has consumed slightly more resources from the FPGA chip. It also needs a larger time to process the signals. Surprisingly, the differences are not small.

TABLE III
SUMMERY OF THE RESOURCES USING SAMPLING FREQUENCY FS= 32.768 MHZ

| | Used | Utilization |
|---|---|---|
| Number of Slice Register | 102744 | 34% |
| Number of Slice LUTs | 97474 | 65% |
| Number of fully used LUT-FF pairs | 88475 | 86% |
| Number of Block RAM/FIFO | 305 | 73% |
| Number of DSP48E1s | 497 | 64% |

TABLE IV
SUMMARY OF THE RESOURCES USING SAMPLING FREQUENCY FS= 8.192 MHZ

| | Used | Utilization |
|---|---|---|
| Number of Slice Register | 91656 | 30% |
| Number of Slice LUTs | 87307 | 57% |
| Number of fully used LUT-FF pairs | 79725 | 80% |
| Number of Block RAM/FIFO | 100 | 24% |
| Number of DSP48E1s | 412 | 53% |

TABLE V
SUMMARY OF TIMING USING FS=32.768MHZ

| Clock frequency | 39.193 MHz |
|---|---|
| Total Routing | 15.511 ns |
| Total Logic | 10.004 ns |
| Total Time | 25.515 ns |

TABLE VI
Summery of Timing using Fs=8.192MHz

| Clock frequency | 39.193 MHz |
|---|---|
| Total Routing | 15.11 ns |
| Total Logic | 8.703 ns |
| Total Time | 23.819 ns |

## III. CONCLUSION

This paper introduced the simulation and implementation of the acquisition phase of the GPS receiver on FPGA through the Xilinx System Generator and hardware/ software co-simulation. In this paper the same platform is utilized in simulation and implementation. This means that we moved a step forward toward easier and more transparent designing, simulation and prototyping of the of the acquisition phase of the GPS receiver through the same environment. The output result from the Xilinx System Generator model and output from the Simulink model is close to each other. This express the successful conversion processing of the Simulink blocks to the Xilinx hardware blocks.

## REFERENCES

[1] GuochangXu and YanXu, *GPS: THEORY, ALORITHMS AND APPLICATIONS*, 3rd ed. 2016.
[2] K. Borre and D. Akos., "A Software-Defined GPS and GALILEO Rceiver – A Single-Frequency Approach," *Birkhauser, New York*, 2006.
[3] G. Hamza, AbdelhaliemZekry, and IbrahimMotawie, "Implementation of a Complete GPS Receiver using Simulink," *IEEE Circits Syst. Mag.*, 2009.
[4] M.ElHawary, G.Gomah, A.Zekry, and I.Hafez, "Simulation of the E1 and E6 Galileo Signals using SIMULINK," *Int. J. Comput. Appl.*, vol. 88, 2014.
[5] M.ElHawary, "Signal Simulator for Global Navigation Satellite System," Ain Shams University, 2014.
[6] G. G. Hamza, Abdelhaliem A.Zekry, and M. M. N., "Implementation of a Complete GPS Receiver on the C6713 DSP through Simulink," *J. Glob. Position. Syst.*, vol. 8, 2009.
[7] Gihan Gomah Hamza, "Enhancing the Time Measurement Accuracy of GPS receiver," Ain Shams University, 2009.
[8] "http://www.ti.com/tool/TMDSDSK6713#technicaldocuments."
[9] D. Siboniyo, "FPGA-based data acquisition system for GNSS receiver for LEO-satellites application," Arctic University of Norway, 2017.
[10] *System Generator for DSP Reference Guide, UG638*, 14.1. April 2012.
[11] *System Generator for DSP User Guide, UG640*, 14.1. April 2012.
[12] *System Generator for DSP User Guide, UG638*, 14.2. July 2012.
[13] *System Generator for DSP User Guide, UG640*, 14.3. October 2012.
[14] *System Generator for DSP Reference Guide, UG638*, 14.5. March 2013.
[15] J.Tian,W.Ye,S.Lin, and Z.Hua, "Software defined radio GNSS receiver design over single DSP platform," *in Proc. 10th Int. Symp. Spread Spectrum Techniques and Applications (ISSSTA08)*, 2008, pp.37-41.
[16] PabloE.Leibovich, JuanG.Díaz, and P. R. JavierG.García, "Dedicated hardware for FFT based fast acquisition of GNSS signals," *IEEE 6th Lat. Am. Symp. Circuits Syst.*, 2015.
[17] A. Shukla, "Hardware Implementation of Real time ECG Analysis algorithms," Hawaii University, 2008.
[18] PirajFozoonmayeh, "A practical approach to DSP algorithms using FPGA devices," Simon Fraser University, 2011.
[19] R. T. Bone, "FPGA design of a hardware efficient pipelined FFT processor," Wright State University, 2008.