# Implementation Methodologies of a Software Defined Navigator (SDN) allowing the Conception of a Real Time Robust Hybrid GPS/Galileo Receiver

**Bernard Dionne***, **René Jr. Landry****, **Aurelian Constantinescu*****

Ecole de Technologie Supérieure
Department of electrical engineering
1100 Notre-Dame street West
Montreal (Quebec), Canada, H3C 1K3
Tel.: +1 (514) 396-8506 Fax: +1 (514) 396-8684
e-mail: *bdionne@ele.etsmtl.ca, **rlandry@ele.etsmtl.ca, ***aconstan@ele.etsmtl.ca

**Keywords**: GPS, Galileo, Hybrid, receiver, software defined, real-time, DSP , FPGA, prototyping.

**ABSTRACT**

*This paper deals first with the implementation methodologies to design hardware navigation receivers using the new telecommunication SDR (Software Defined Radio) state-of-the-art concept. Those concepts and know-how with sufficient adaptations to the navigation domain allow reaching a generic version of the so-called Software Defined Navigator (SDN) well suited for fast real-time prototyping. In this study, a real-time architecture of a robust hybrid GPS/Galileo receiver will be presented.*

*In fact, the GPS and GNSS market is growing rapidly, and new services will be brought up soon. The need of an easy adaptable and reconfigurable architecture of a navigation receiver was proven to be important in order to follow the satellite system continuous evolutions and technological modifications. The gain of using the Galileo constellation is obvious considering the four(4) main problematics of the GPS itself which are the precision, the robustness, the continuity of service and the system reliability as discussed in Kaplan[1]. The advent of the European Galileo navigation system will increase the demand on highly robust and precise GNSS hybrid receiver. This situation will make the issue of "time-to-market" an important one. To achieve this, a software defined navigator concept will be used to develop quickly new kind of navigation technologies and technical services, allowing to take advantages of the new potential of navigation market. Our interest will focus on the implementation capabilities of a Matlab-Simulink GPS receiver model in a targeted hardware composed of a FPGA and a DSP. Many parameters will be taken into account for that kind of implementation. The high frequency data rate and the number of logical gates are among the characteristics needed to be evaluated.*

*Also, the compromise between FPGA and/or DSP real-time implementation for the various sections of the simulation model will be approached by a thorough examination of the model's characteristics. Upon this analysis, the navigation receiver model will be separated in two different modules, one for the FPGA and one for the DSP. Real-time code generation methodology will be presented. Final version of the software architecture will be properly presented. A noisy environment is considered to analyze the accuracy of the model and to evaluate correctly the performances of the navigation receiver. The hardware platform in which the receiver will be implemented is modular and allows flexibility and rapid prototyping. The evaluation of the prototype is done by the simulation of the constellation and receiver within the hardware prototype. Evaluation of the receiver prototype is done using different parameters such as PLL and DLL parameters and resolution accuracy. The results are compared to the present literature standards of a GPS receiver. Finally, to compare the performance of the implemented model with an existing receiver, a GPS antenna is connected into an already developed Direct RF Sampling board in order to obtain IF samples of the real GPS signal. The data will be treated in the FPGA and DSP and the calculated position of our prototype given according to conventional navigation algorithms will be analyse.*

## 1.0 INTRODUCTION

Since its birth, the Global Positioning System (GPS) has had a phenomenal success. People have now discovered a more precise way to determine their position, speed and time. With the advent of European Galileo system, a new satellite constellation, it will be possible to achieve an order of precision never seen before. Interest in precise navigational data will augment the need of a new breed of receiver that enables the capture of many signals to transform them into coordinates. But these new receivers will have to be flexible, robust and adaptable to meet the new environment in which they will work. Some studies have been made about it, like for a system-on-chip[2], or a DSP processor[7]. Also, with that in mind, new services will be brought up via the GPS and Galileo systems increasing the demand for the receiver development. Companies will have to adapt themselves to the new situation and the "time-to-market" factor will be an important one. For them to stay competitive in the market, they will have to provide high performance, compact and robust receiver to the clients. The radio-navigation industry is rapidly evolving and continues to modernize the satellite navigation systems (GPS, Galileo, and even Glonass). The demand for a receiver architecture who can ensure the complete implementation of a hybrid processing between the two systems (GPS and Galileo), is growing. The Software Defined Navigator concept presented in this paper will help to develop fast prototype receiver, with less costs, so that companies can adapt their products on the market. Others have developed similar strategies such as software defined radio[8] or for other applications[3][10].

## 2.0 THE CONCEPT OF SOFTWARE DEFINED NAVIGATOR (SDN)

Since its launch in December 1993, the GPS has attracted more and more researchers throughout the years. In parallel, many companies have surfaced and used the GPS technology to create products, offer services and develop many tools using this technology. As the years past, the evolution of product-based GPS has multiplied and new services have entered the market. It's not surprising that, with billions of dollars invested, the European Galileo project has created the same effervescence throughout the navigation community. As products have become more complex, the cost of developing such product has also gone up. Hence, a new concept must be developed to ensure a low-cost efficient rapid prototyping. At LACIME laboratory of the École de Technologie Supérieure in Montreal, a project was built to deliver a methodology that respected the new parameters in the navigation market. As shown in Figure 1, there are two ways to integrate new services. The first one is to change physically the hardware in the component, which is represented by the three upper blocks. The second one, which is at the base of SDN concept, is to have a software application that simulates the technology. It allows to make changes and to see the effect immediately by simulation. Once the results are obtained, a prototype can be created by configuring a hardware platform. Then the prototype is tested in a real environment to measure if it actually fits the demand of the user or client.

In reference to Figure 1, the combination of the software application, hardware platform and the prototype is the software defined concept; some developers may think they are doing SDR development. This paper will define SDN concept and will present an application of this concept in the field of navigation.

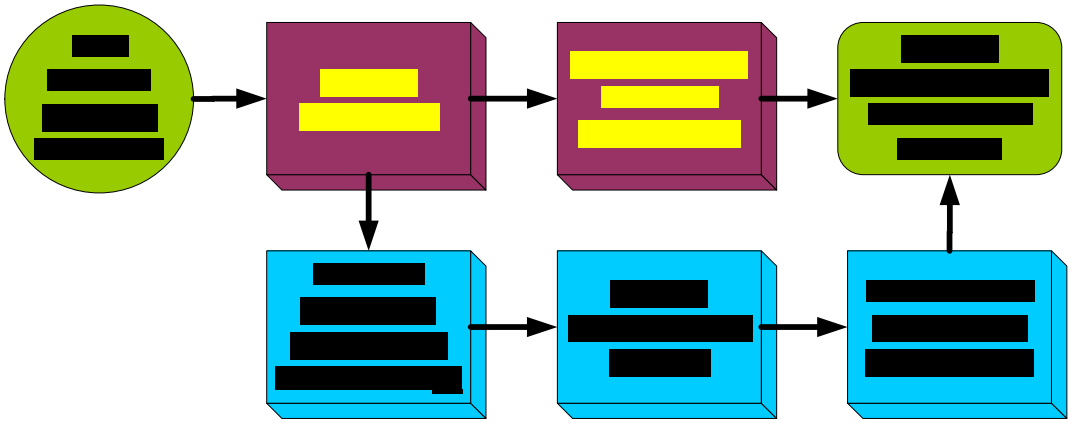

***Figure 1 -*** Steps for the integration of a new service.

### 2.1 Similarities of the approach with Software Defined Radio concept

The concept of a software defined platform is not new. The telecommunication community has already developed around it with the Software Defined Radio. At the origin, the name was Modular Multifunction Information Transfer System (MMITS), but since it was not clear enough to people, the telecommunication community later changed it to Software Radio and then to Software Defined Radio. It responded to the need of a faster and more flexible development tool. By definition, a SDR is "a collection of hardware and software technologies that enables reconfigurable system architectures for wireless networks and user terminals"[8]. It allows the development of multi-mode, multi-band and multi-functional wireless devices at a reduced cost and more rapidly. And since a great part of our economy is based on telecommunications, there is a diversity of interests, so the concept must be flexible.
The basic of communication revolve around three poles: source, medium and terminal. On a SDR basis, the diversity of technologies used world-wide makes the system really versatile. Navigation is one of the numerous applications that can benefit from the concept of SDR. The source is the satellite signal, the medium is the environment and the terminal is the receiver. So basically, SDN is a branch of SDR. The reconfiguration of the device used (e.g, platform, prototype) by the software allows developers to work with only one hardware to provide the service demanded by a client.

Just like SDR, the concept of SDN has many benefits:

- Standard architecture for every satellite communications products.
- Potential for a reduced cost development.
- Rapid prototyping of a hybrid GPS/Galileo receiver.
- Facilitate the conversion from analog to digital signal.
- Accessible update of new services for an end-user throughout the Internet.

### 2.2 Structure of the real-time demonstrator platform

In recognition of the characteristics needed to create the concept presented above, the hardware platform must be chosen in respect of those needs. The navigation system is a signal processing system with complex algorithms; hence, the platform must have digital signal processor with high MIPS (Multi Instructions per Seconds) and high memory. Also, the hardware must be reconfigurable to allow the flexibility needed and the development of a prototype. To achieve these goals, a combination of a Field Programmable Gate Array (FPGA)

and Digital Signal Processor (DSP) is an obvious choice. As it will be presented, both hardware are needed, each having a specific role, due to their characteristics, in the development of the hybrid navigation receiver.
Because of the high frequency of the GPS and Galileo signals, the selected hardware must have high speed and large bandwidth analog to digital converter in order to achieve Direct RF sampling and digital down-converter. Not many ADC on the market today can achieve that operation. The GPS carrier is at 1575.42 MHz which demands, to respect the Nyquist criteria, a converter of at least 3150.84 MHz of sampling period. To bypass that problem, the platform will include a DR Sampling board, developed at LACIME laboratory, to obtain a GPS signal at an intermediate frequency. Sundance Inc[13]. has developed a concept of module that satisfies all of our pre-requisite. The platform will be composed of a carrier board, which ensures communications between a computer and the modules, on which each module will be presented with a specific task. For the implementation of a hybrid GPS/Galileo receiver, three(3) modules are needed, one with the FPGA, another one for the DSP and the last one will be the data acquisition module. The separation of the modules responds to the flexibility factor needed in our development, and was look into before by Avrunin and al.[5]. Figure 2 shows the interaction between the modules and the computer. As it can be seen, each module can interact with the others, without being completely dependant of each other.

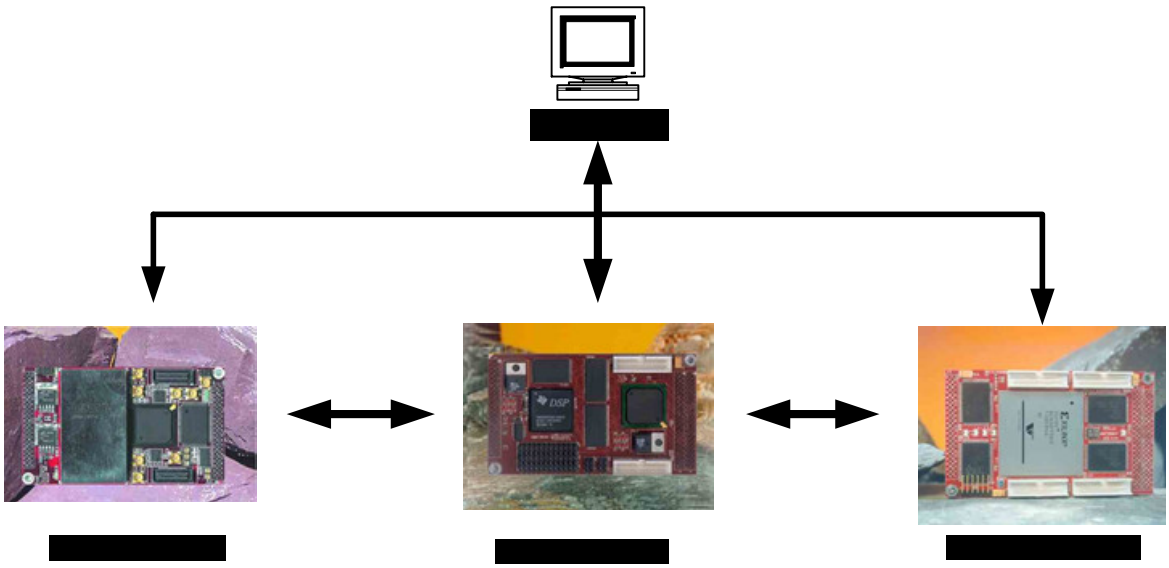

***Figure 2*** - Interaction between the hardware platforms

Communications between the module is done by comports within the carrier board. Also, some high-speed buses are available for data flow. This option is important in the development since navigational signals are higher that 1 GHz in frequency.

### 2.3 Description of software

Getting back to the concept, we need a software simulation of the existing technology. In the case of navigation, the Mathworks tool Simulink offers fast simulation and allows, through Real-Time Workshop, the generation of all the source codes needed to be implemented in the hardware. The detailed library Simulink offers allows dividing the process in many simple tasks,

which will become practical for the implementation. Also, the Matlab/Simulink environment is flexible, fast and accurate. Because of the complexity of some operations, the Code Composer Studio software tool, developed by Texas Instruments, is needed to implement the navigational equation and the Kalman filter in the DSP. C/C++ code can treat more complex algorithms faster than Hardware Defined Language (HDL), which is needed for the FPGA. The Xilinx FPGA comes with the ISE 5.2 system. An important part of the system is System Generator, which allows Simulink models to be transformed into VHDL project, as shown by Ownby and al.[11]. System Generator runs inside the Simulink environment produced by MathWorks, meaning that all the libraries provided in Simulink are available for the simulation. In addition, the results seen in simulation reflect those that will be seen in the hardware, which is the purpose of SDN. The signals simulated are not just bits, but they can change into appropriate signal types, reflecting at the utmost the reality. Figure 3 shows the interaction and the path of the software used in the methodology.

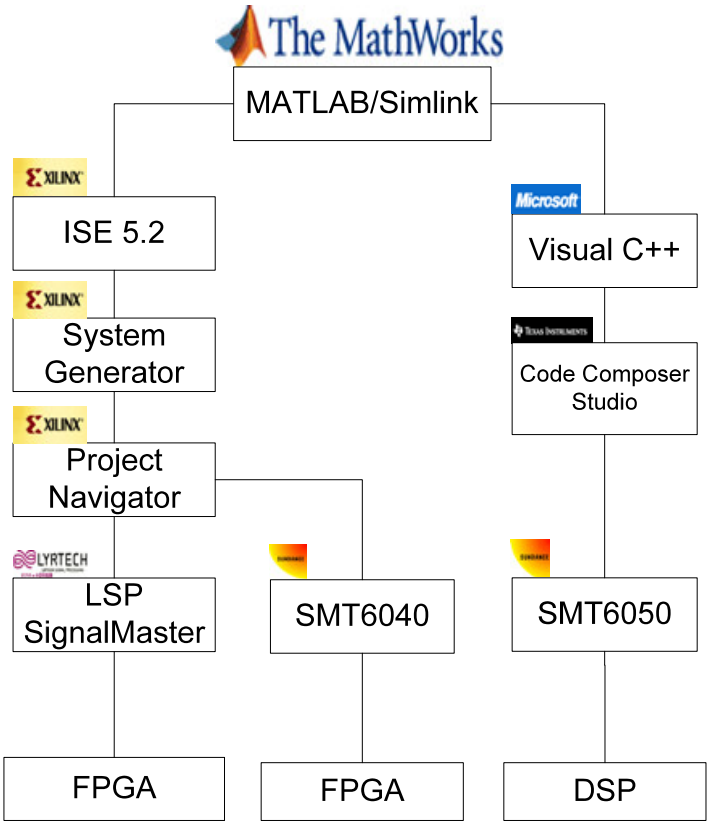


***Figure 3*** – Software Defined Navigation

## 3.0 METHODOLOGY DESCRIPTION

Dealing with such high frequency signals, we need to develop a methodology that will ensure to keep the properties of all signals. At the beginning of the project, it was obvious that we needed a concept such as SDN. Similar to SDR, the concept would make the development of a prototype easier. The first step was to find the type of hardware in which the prototype will be downloaded. The characteristics of the hardware would have an impact on the software we choose. Today, on the market, there is a lot of hardware available that satisfies many needs.

To achieve the best platform, the objectives must be analysed. The technical objectives are; Based on the communication link seen in Figure 4:

- Simulate and process in real-time the source (represented by the blocks before the satellite)
- Simulate and process in real-time the channel (between satellite and receiver antenna)
- Simulate and process in real-time the receiver (represented by the blocks after the antenna)

As we can see from the SDR definition, the SDN objectives are similar to a normal communication link. In order to achieve those objectives, there are some particularities that we need to look at. These characteristics of the model will help us to define the type of hardware to use:

- High frequency ( carrier modulation is 1.57542 GHz)
- Large bandwidth analog-to-digital converter (the source coding is 1.023 MHz)
- Analog inputs, Large memory, High speed processor
- Serial port (communication with the PC)

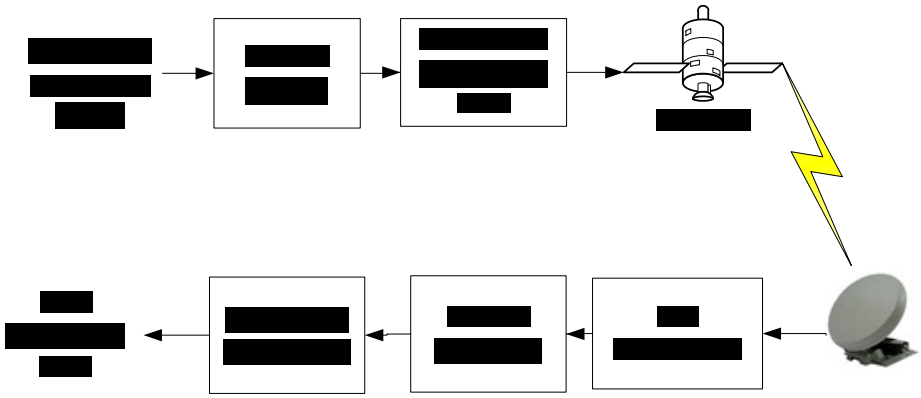

***Figure 4*** - Communication link of navigation process

Also, the interaction between many processes must be taken into account. Some tasks demand higher sampling rate (for instance, the Phase Locked Loop), as others demand more complex algorithms (e.g, the Kalman filter).

### 3.1 FPGA and DSP characteristics applied to SDN

Digital Signal Processor (DSP) and Field Programmable Gate Array (FPGA) have been the two majors digital processing units developed in the past few years[9]. Already used in SDR, their characteristics allow developers flexibility and fast-prototyping. To make the transition between simulation and real-time, DSP characteristics must be examined:

- Easy to program
- Good development tools
- Infrastructure for non-DSP tasks is difficult
- Fast time-to-market
- Low cost development
- High digital signal processing performance

DSP, developed by Texas Instruments[14], have matured and offer a lot of flexibility. Because it has small and rapid instructions, the DSPs can do a lot of complex operations in a small amount of time. Also, a sufficient memory is added to support wide process.

On the other hand, the FPGA has some capabilities not to be outlook:

- Architectural flexibility
- Re-use hardware for multiple tasks
- Low cost, low power and small size
- DSP-oriented tools are becoming more efficient.

For the project at LACIME laboratory, the model to be implemented is a hybrid GPS/Galileo satellite navigation system. Hence, each operation is divided in several tasks and those tasks are examined to implement them in the appropriate hardware. Depending on the characteristics of the task and the mentioned above characteristics of the hardware, each tasks will follow a specific path in order to generate the files, who are needed to implemented the system. More complex algorithms fit more the DSP but faster ones are bound to be in the FPGA. Examination of the algorithms is essential for the optimization of the hardware. Since, we have a determined memory size or gates, the files generated must be concise.

### 3.2 Step-by-step implementation

With the basic structure differences of the FPGA and the DSP, there are differences in the path used to achieve the implementation. Evidently, the tools needed for both hardware pieces will reflect those differences. In the Figure 5, the generation of the file to be implemented in the DSP is showed. First of all, a model simulating the service we have to implement must be created. The Matlab software family, from MathWorks, have many characteristics to help us to create our model in navigational communication. The flexibility and the various libraries of Simulink allow developing a GPS/Galileo satellite architecture to the utmost reality. The architecture is presented later in this paper.

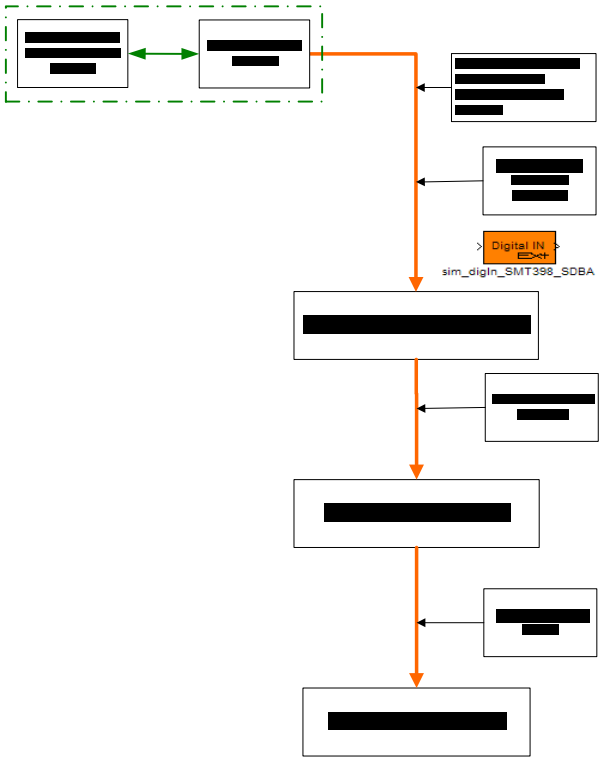


***Figure 5*** – DSP implementation method for real-time simulation

Once the model properly simulates the communication, some blocks, from Simulink libraries developed by Sundance, are needed to generate the files that will be executed in the DSP. The library, developed by Sundance, encloses all the basic elements required for the model such as the generation of sine wave, multipliers and buffers. The C++ files are then generated and Real-time Workshop build them on the base of the modified model. Many parameters must be taken into account when generating those files, such as the destination (what kind of DSP), the speed, and the format of the code, for instance. Each one has a specific effect on the generated code. The file is then processed through Code Composer Studio from Texas Instruments, which creates, from a project, the assembly file (.out) that the DSP read and then execute the tasks. The path is presented in Figure 5

To get files into the FPGA, another path is taken, since the tools used in the DSP branch are not fitted for generation of VHDL code and digital signal processing without a little transformation, as it was explored by [6]. The workload here is greater than before since every block used must be changed for a specific block of the Xilinx libraries. Because we will be working with binary numbers, each operation needs specification not available in Simulink libraries. Once the model is adequately changed, the software System Generator, also from Xilinx, is used to generate a project in VHDL, which is a variation of Hardware Descriptive Language (HDL). Once again using tools provided by Xilinx, the generation of the binary code files is done by Project Manager. Completed with the generation of the code are reports of time and space taken by the code inside the FPGA. These values are important to optimize the code. Of course, when you specify the number of bits you are using, the code adjust itself in that manner; a higher number of bits will increase the value of the precision but will also take a larger amount of memory. The tolerance of error that the user wants will determine the value of the number of bits, and consequently, the time of generation and length of the code. Also, the quantification of error and overflow has the same effect on the code. The choice of the FPGA in which the implementation will take place is specified at the very beginning of the code generation. Based on that information, it will be easily seen if the hardware is able to execute the file. The number of logical gates is the primary issue since it will determine the number of independent operations the FPGA will be able to do in one cycle.

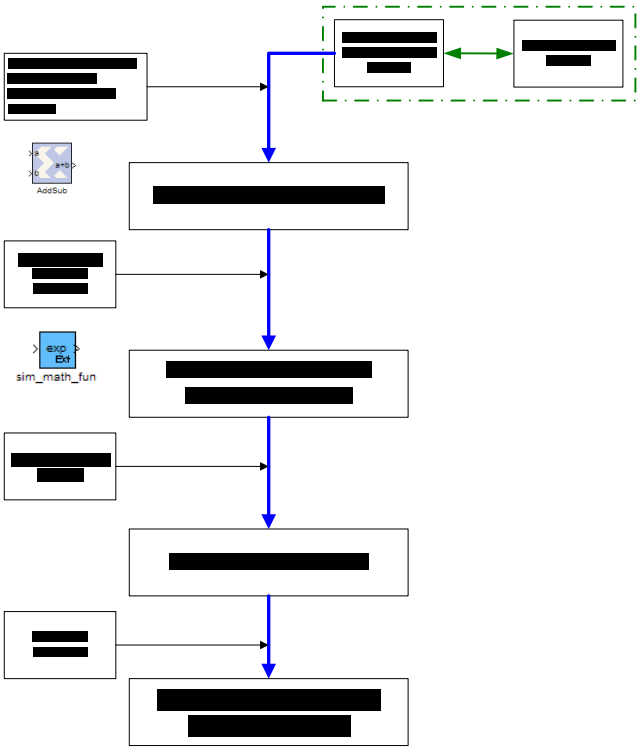


***Figure 6 -*** FPGA implementation method for real-time simulation

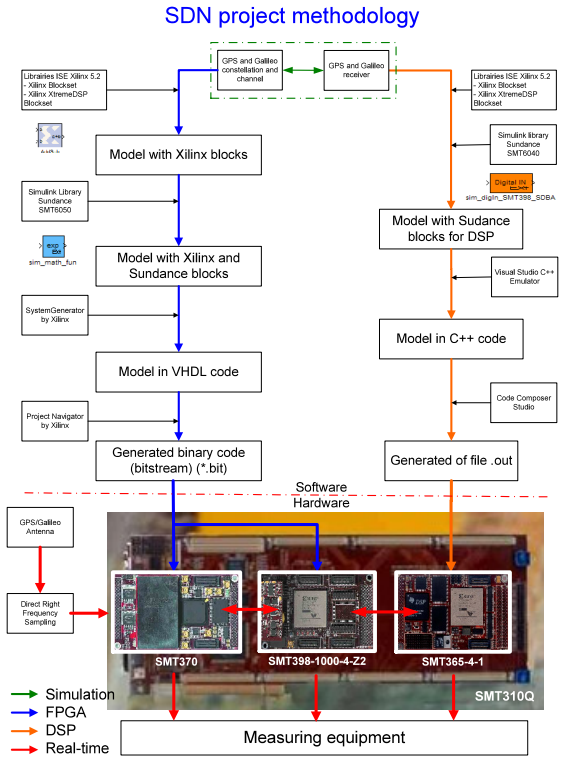


***Figure 7 –*** Implementation methodology

In Figure 7, the last step of the methodology is presented. The two different generated codes are uploaded into their respective hardware. Before the FPGA and DSP can treat any data, it is needed to sample the satellite signal. For that purpose, a third module, containing an FPGA for managing the module, serves as the signal acquisition module. As presented before, the analog-to-digital converter characteristics must enable it to sample the GPS signal. A converter of such high sampling frequency is uncommon and costly. During a project at LACIME Labs a Direct Right Frequency Sampling card has been developed, allowing a satellite signal to be converted to an intermediate frequency. Placing the card before the acquisition module (Sundance SMT370-DC) reduces the need of a high and large bandwidth ADC, thus lowering the cost of our prototype. Once the data is sampled, it is directed to the FPGA for treatment or re-directs in the DSP to be treated, in regards to the developed architecture of the receiver.

## 4.0 IMPLEMANTATION OF A GPS/GALILEO RECEIVER

### 4.1 Receiver description

To simulate a hybrid GPS/Galileo communication on Simulink requires some principles:

- The generation of the GPS and Galileo signals is made accordingly to the known or develop modulation schemes explained in Kaplan[1];
- Real-time digital signal processing needs to be done in IF;
- Dynamic disturbances (Doppler effect) must be included on the carrier frequency and on the code modulation directly in the source module;
- The receiver is composed of a PLL (Phase Locked Loop), DLL (Delay Locked Loop) and all other receiver modules of a generic GPS receiver and according to the known literature.

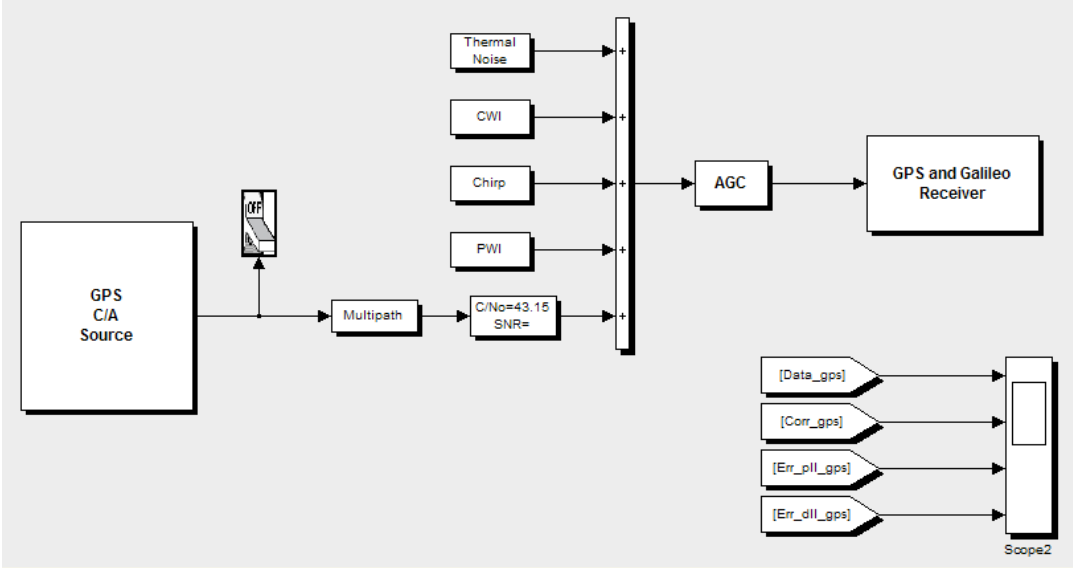


***Figure 8*** – Navigation link architecture

The simulator is composed of three(3) parts: source, channel and receiver; each having specific tasks to do. It is important to note how the simulator is divided, because it will affect the code generation and the way it will to be configured in the modules. The source module generates code, a Gold code sequence, modulated with the data and transposed on IF (for each satellite signal). The frequency and the phase of the transmitted signal vary according to the selected disturbances. The source module generates multiple signals according to the number of desired satellites or according to the desired system to be analyzed (GPS and/or Galileo). The main role of the channel is to attenuate the signal, to introduce the ionospheric model, to add the thermal noise, the jammers and the multipath in order to degrade the signal and to make the model as more realistic as possible as shown in Ilie and al.[4]

Within the framework of the present paper, the receiver is composed of four C/A channel (GPS) and two E2-L1-E1 channels (Galileo). The Galileo satellite source signal model is presented in Figure 10. The main elements in the receiver channel are the PLL and DLL. In general, the behaviour of the latter determines the performance of the receiver and the system robustness. Those digital loops are our main concern during the analysis of robustness, dynamics and performances of the receiver. In fact, without strong and precise carrier and code phase measurements, the navigation algorithm and the Kalman filtering will introduce positioning errors.

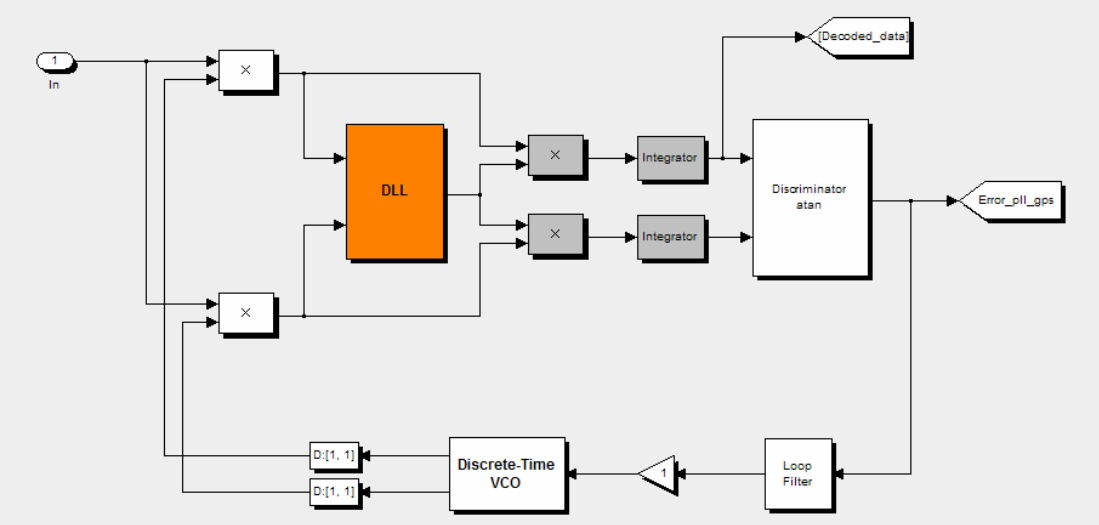


***Figure 9*** – Digital GPS receiver architecture

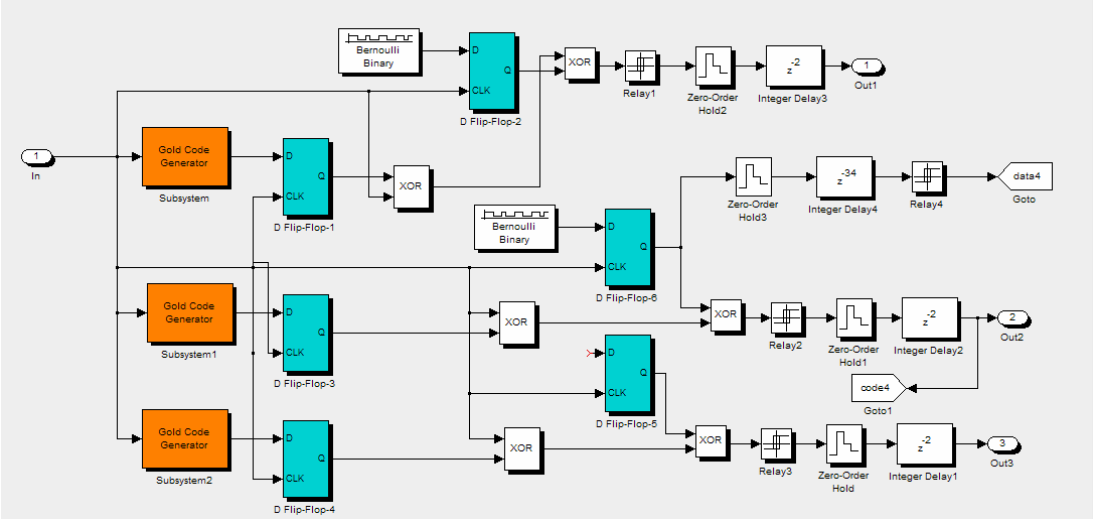


***Figure 10*** – Galileo code generation model

### 4.2 Modifications on Simulink model for real-time simulation

Once the validation of the model of the GPS/Galileo receiver is complete, an analysis is done to divide the model in two different parts, to go along with our methodology. Taking in account the characteristics of the hardware, in this case an FPGA and DSP, and the complexity of each operation done to the signals, modifications to the model are implemented. Due to the complexity of the algorithm, the Kalman filter is more fitted for the DSP than the FPGA. In fact, the use of matrices in the filter, and the operations going along with that, proves to be the deciding factor. For the two(2) loops (PLL and DLL) of the receiver, the amount of operations is high and time is the essence here. A faster clock in the FPGA proves to be an important factor. Because the simulator not only includes a receiver, the source as well should go in the FPGA for the same reasons as the PLL and DLL. Hence the implementation of the system in real time is done as presented in table 1.

| FPGA | DSP |
|---|---|
| Phase Lock Loop (PLL) | Other navigation processes |
| Delay Lock Loop (DLL) | Navigation channel |
| Satellite signal (source) | |

***Table 1 –*** Section of the SDN model implemented in hardware

Because of the division of the model, the accuracy of the simulator is reduced. The different discriminators used in the PLL and DLL are mathematical expressions that cannot be implemented without losing a significant amount of memory. In many discriminators, a division is needed to calculate the error on the synchronization signal. Unfortunately, such operation in binary mode creates a high latency, and cannot stay synchronized with the speed of the loop. Because of that, simpler discriminators are implemented thus reducing the accuracy of the receiver. But a higher number of bits in the discriminator are used to compensate, even though that solution takes a lot of memory also.

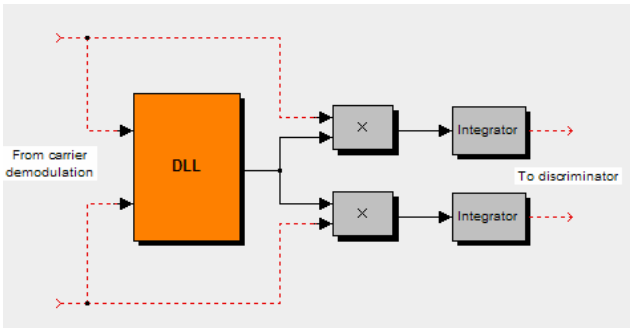


***Figure 11***– Part of model with Simulink blocks

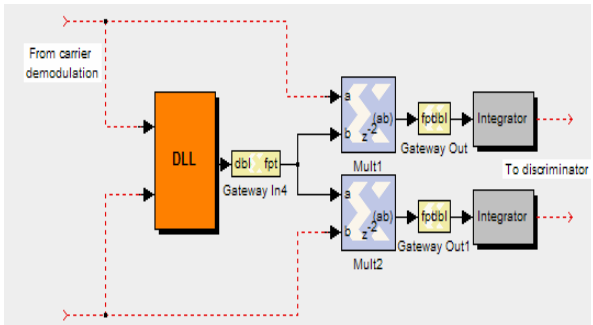


***Figure 12***– Part of model with Xilinx blocks

The two figures above (Figure 11 and Figure 12) show the difference in the model between Simulink blocks and the Xilinx ones. Some parameters taken in account for the changes are the latency, the tolerance on the error, the type of data and the number of bits. The value of those parameters is determined in a way to represent the reality as close as possible. A simulation incorporating those changes is done and the results are compared to the Simulink model results.

The Figure 13 and Figure 14 show the results of the decoded signal before it enters the navigation process. At that point, the signal is a binary code. The results are similar, proving that our model, even modified with Xilinx blocks, still reflects the communication between the satellite and a GPS receiver.

The same observation can be said about the Galileo architecture model developed at LACIME laboratory. Because of the latency due to using binary, the PLL in the model using Xilinx blocks takes more time to lock on the signal, as we can see in Figure 13 and Figure 14; based on both figures, the difference is 1 ms, this will be adjust in the future. That effect is translated in real-time also, so one might expect that our prototype will not be, at first, as fast as a known receiver would be.

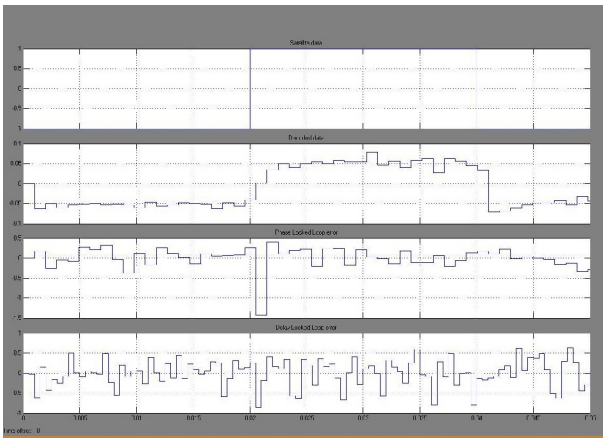

***Figure 13*** – Navigational data, decoded navigational data, phase error and delay error in normal basic function in Simulink

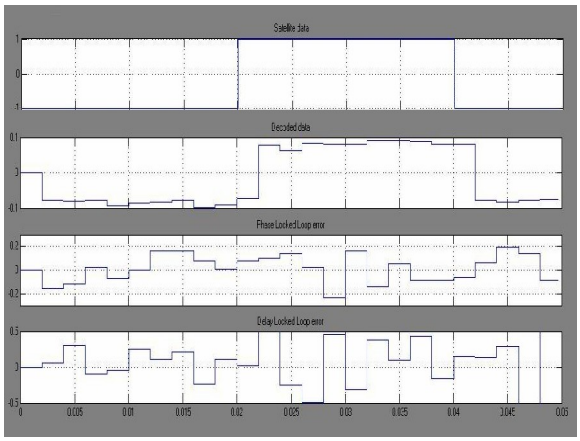


***Figure 14*** – Navigational data, decoded navigational data, phase error and delay error with Xilinx blocks in Simulink

### 4.3 Code generation

Upon modifying the different models, they are then processed through the path mentioned before in order to generate the respective code implemented in the hardware. Before generating any code, some parameters must be fixed, such as the number of bits used in the two(2) loops. Table 2 shows the effects of the maximum number of bits used in the receiver. The number of bits indicated reflects the maximum number found in one operation in the receiver. The destination is a Xilinx XC2V1000 FPGA.

| **Model of receiver** | **1-channel 32 bits** | | **1-channel 16 bits** | | **4-channels 16-bits** | |
|---|---|---|---|---|---|---|
| **Components** | Number of components used | % of available components used | Number of components used | % of available components used | Number of components used | % of available components used |
| **Slices** | 1181 | 23 | 918 | 17 | 3672 | 74 |
| **Slices Flips-Flops** | 331 | 3 | 311 | 3 | 1244 | 12 |
| **4 inputs LUT** | 2170 | 21 | 1652 | 16 | 6608 | 64 |
| **Bonded IOBs** | 40 | 9 | 40 | 9 | 136 | 31 |
| **BRAMs** | 1 | 2 | 1 | 2 | 5 | 10 |
| **GCLKs** | 1 | 6 | 1 | 6 | 6 | 6 |
| **Generation time** | 290 sec | | 152 sec | | 311 sec | |

***Table 2*** – Hardware availability for hybrid receiver

As expected, the receiver with the higher number of bits takes more memory and more time to generate then the one with less bits. The 16-bit limit is chosen to maximize the efficiency of the digital to analog converter, which has a dual 16-bit sampling. One of the objectives of SDN is fast prototyping, and the time of code generation gives us the expected result. An effective GPS receiver composed of minimum 4-channels takes approximately 5 minutes to implement.

The maximum frequency of the model is declared at 38,601 MHz, which is more than enough for our receiver. The direct right frequency sampling combined with the 14-bit ADC will sample the data at a frequency under 38,601 MHz.

The same analysis can be made for the source part which is the satellite signal. As explained above, the satellite signal consists of navigational data, C/A code and modulation. Generated

independently, those three aspects of the satellite signals required a small amount of the resources, as displayed in table 3.

| **Part of the source** | **C/A code (Gold code)** | | **Voltage Controlled-Oscillator (VCO)** | | **Satellite signal** | |
|---|---|---|---|---|---|---|
| **Components** | Number of components used | % of available components used | Number of components used | % of available components used | Number of components used | % of available components used |
| **Slices** | 27 | 0 | 591 | 11 | 1227 | 22 |
| **Slices Flips-Flops** | 44 | 0 | 48 | 0 | 130 | 2 |
| **4 inputs LUT** | 14 | 0 | 1143 | 11 | 2400 | 23 |
| **Bonded IOBs** | 11 | 0 | 24 | 5 | 59 | 10 |
| **BRAMs** | 0 | 0 | 0 | 0 | 5 | 12 |
| **GCLKs** | 1 | 6 | 1 | 6 | 1 | 6 |
| **Generation time** | 60 sec | | 120 sec | | 250 sec | |

***Table 3*** – Hardware availability for GPS source

Considering those numbers, a complete communication link between GPS satellite and GPS receiver fit in exactly two(2) Xilinx XC2V1000 FPGA, one simulating in real-time the source, and the other the receiver. Since the platform has three(3) FPGA, it is easily possible to fit the entire communication link in the chosen hardware.

These results prove that a prototype can be realised in less than 6 minutes, and tested nearly immediately with the adequate tools around it. In that way, it provides fast and efficient prototyping at a reduced cost. At that stage of the methodology, the prototype is ready to be tested with actual GPS signals.

## 5.0 PRELIMINARY RESULTS ANALYSIS

### 5.1 Real-time results

Real-time simulation is done after the files are implemented in the prototype. Validation of the model is made by simulating the communication link between the satellite and the receiver inside the prototype. Figure 15 to Figure 18 show the results in simulation and in real-time of a GPS source.

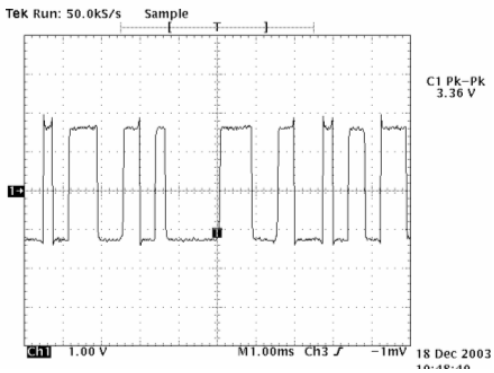


***Figure 15*** – Gold code in real-time from the FPGA

To optimize the testing, the navigational link is separated and tested individually. In Figure 15, the generation of the Gold code is made within the FPGA and observed through an oscilloscope. By calculation of the period, the validation of the model implemented to generate the Gold code is made. The code has a frequency of 1 MHz, which is what is observed in the figure. After the Gold code is validated, the rest of the satellite signal is implemented in the FPGA.

The two figures below (Figure 16 and Figure 17) show that the satellite simulator implemented in the platform has the same respond than the simulator itself. The period of the signal is respected at 2.046 MHz. This baseband signal is selected because, with higher frequency, sampling the signal with Matlab demands a significant amount of memory.

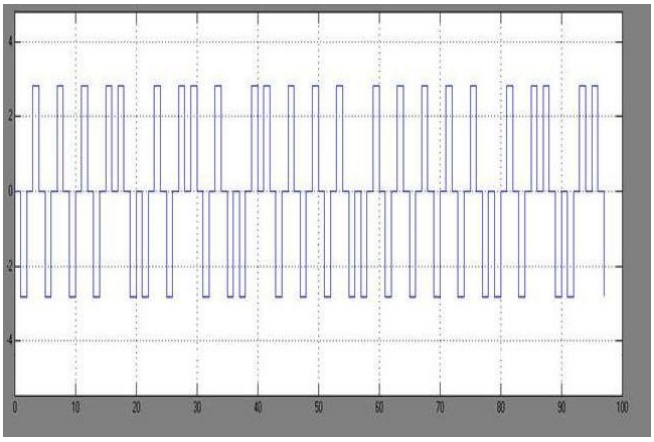

***Figure 16***- Satellite signal simulated in Simulink

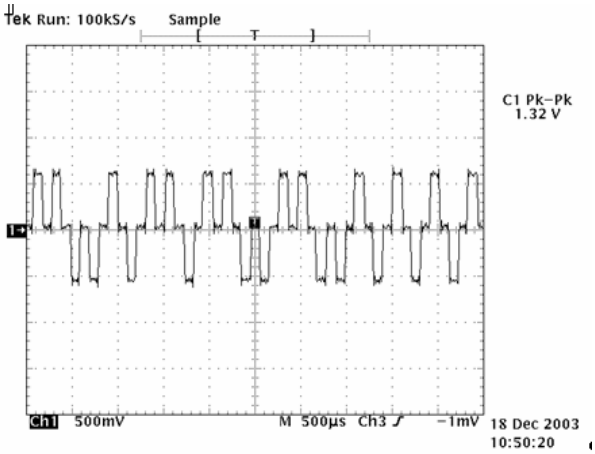


***Figure 17*** - Satellite signal simulated in real-time FPGA platform

A carrier frequency of 2.046 MHz is used and, to respect Nyquist criteria, a sampling frequency of 8.184 MHz enables us to correctly digitalize the signal. The amplitude of the signal from the FPGA is lesser than the one from the Matlab simulation in regards of the quantification from the digital to analog converter, and the number of bits used for the quantification. The first part of the model is coherent with the reality of the GPS signal and with today literature on GPS signal.

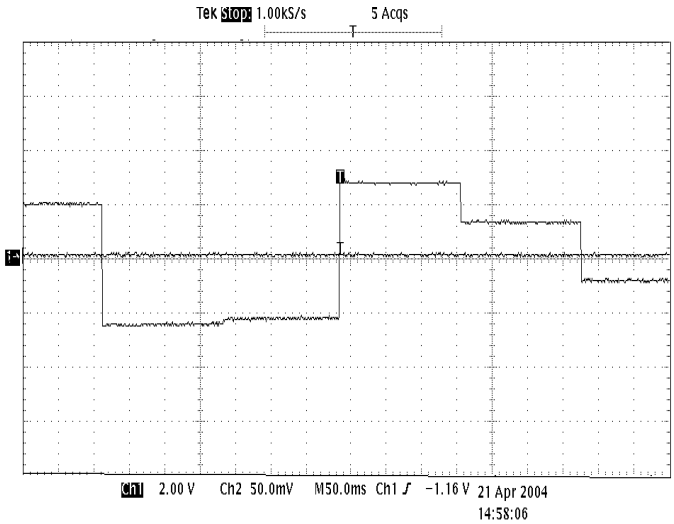


***Figure 18***– Decoded Navigational data in real-time

The decoded navigational data, before it is processed in the navigational equations, is shown in Figure 18. Once again, it is easy to validate that the implementation gives the result awaited for. With a frequency of 50 Hz, the limits of the oscilloscope are reached so that we cannot see the entire wave form but the wave form correspond to the sent data. The only discrepancy comes from the sampling frequency difference between the FPGA and the DAC. The FPGA has a higher frequency and the converter cannot sample all the data. Within the framework of this paper, the loss of information is not critical because the decoded navigational data is not used.

In the complete structure of the model, the decoded navigational data is driven to the DSP, which has the Kalman filtering process. The purpose here is to validate the process in real-time. From those results, the methodology has proven to be fast and efficient, generating a complete 4-channels receiver in less than 6 minutes, with a 16-bits quantification. The simulator will be available, in simulation and in real time, over the Internet, so that anyone will be able to validate their data. At the time of this paper, Kalman filter and navigational equations are being brought up in the implementation process so that the decoded navigational data will be used to have time and position form an actual GPS signal. Hence, making the model develop, a complete independent receiver which can, not only compute position and time through GPS signal but is also able, once the constellation will be active, to accept Galileo signals, reducing the error on precision, robustness, continuity of service and system reliability.

## 6.0 CONCLUSION

The concept of software defined navigator, even new, allows many possibilities, just as SDR did for the telecommunication market. The advent of Galileo is going to pushed further the demand for prototype embracing both technology, i.e GPS and Galileo. The methodologies develop at LACIME Labs showed that not only, it is cost efficient and flexible, but also fast prototyping can be achieve with the right tools. Further development around the GPS and Galileo can be foreseen, such as an acquisition module, which will bring the architecture receiver to a complete digital receiver. The modular approach makes validating and efficient coding an actual receiver an easy task in the sense that you get real-time results pretty fast. The methodology proposed is based on a thorough analysis of the GPS/Galileo technology and clearly adapts to the navigational community as it quickly validate the new requirements of the technologies.

## REFERENCES


[1] Kaplan, E., ***"Understanding GPS:Principles and Applications",*** 3rd edition, London, Artech House, 1996.

[2] Rabaeijs, A., Grosso, D., Huang, X., Qi, D.,**"*GPS receiver prototype for integration into system-on-chip",*** IEEE Transactions on Consumer Electronics, Vol 49, No. 1, February 2003.

[3] Turney, R. D., Reza, A. M., Delva, J. G. R. Issler, JL., Martin, J.C., Erhard P., Lucas-Rodriguez, R. , Pratt, T., **"*FPGA implementation of adaptive temporal kalman filter for real time video filtering",*** IEEE Publication 1999.

[4] Ilie, I., Landry, R.Jr., **"*Simulation of GPS and Galileo architectures for anti-jamming and multipath analysis",*** École de technologie superieure, 2003.

[5] Avrunin, G., Corbett, J. C., Dillon, L. K., ***"Analyzing Partially-Implemented Real-time systems"***, IEEE Transactions on software engineering, August 1998

[6] Attri, S., Sohi, B.S., Chopra, Y.C***., "Efficient design of application specific DSP cores using FPGAs",*** 2001

[7] Kibe, S.V., Shridara, K.A., Jayalalitha, M., ***"Software-based GIC/GNSS compatible GPS receiver architecture using TMS320C030 DSP processor",*** Satellite Systems for Mobile Communications and Navigation Conference, Publication no 424, 13-15th May 1996

[8] Software Defined Radio Forum, www.sdrforum.org.

[9] Berkeley Design Technology, Inc., ***"Comparing FPGAs and DSPs for Embedded Signal Processing"***, Presentation, October 2002

[10] Martina, M., Molino, A., Vacca, F., ***"FPGA System-on-chip soft IP design: A reconfigurable DSP"***, CERCOM, IEEE publication, 2002.

[11] Ownby, M., Mahmoud, W.H., ***"A design methodology for implementing DSP with Xilinx® System Genrator for Matlab®"***, IEEE Publication, 2002.

[12] Xilinx Corp., ***"Xilinx System Generator for Simulink®"***, Version 3.1.

[13] Sundance Inc.., www.sundance.com

[14] Texas Instruments Inc.., www.ti.com