A flexible implementation of a Global Navigation Satellite System (GNSS) receiver for on-board satellite navigation

A. Dion[1], E. Boutillon[2], V. Calmettes[1], E. Liegon[3]
[1]Universite de Toulouse, ISAE, DEOS, TESA, Toulouse, France
ISAE, 10 Av. E. Belin, BP54032, 31055 Toulouse Cedex 4, France. E-mail: arnaud.dion@isae.fr, vincent.calmettes@isae.fr
[2]Lab-STICC, CNRS, Universite Europenne de Bretagne, France
Centre de recherche, UBS, BP92116, 56100 Lorient, France. E-mail: emmanuel.boutillon@univ-ubs.fr
[3]Thales Alenia Space, France
26 Avenue Jean-Franois Champollion 31100 Toulouse, France. E-mail: emmanuel.liegeon@thalesaleniaspace.com

***Abstract*—In this paper, we present the implementation of a versatile Global Navigation Satellite System (GNSS) receiver for satellite applications. For versatility purpose, the choice of the receiver algorithms has been motivated by 1) their capability to fulfill the application requirements with a moderate complexity, 2) their capability of being factorized in a small set of elementary modules that can be configured and combined in various ways in order to process both GPS and Galileo current and future signals. These algorithms have been specified using a modeling language that can be common to hardware and software flow: C++ based SystemC. The use of a virtual platform for simulation allows us to identify bottleneck of the architecture and to propose algorithm modification to solve them. Thanks to this algorithm redesign, the time of acquisition (i.e. synchronization with a given GNSS satellite), for Low Earth Orbit (LEO) application, has been reduced by a factor 2 from 8.5 s to 4.25 s.**

I. Introduction

Actual positioning systems for satellites are mainly based on ground equipment, that means heavy infrastructures. Autonomous positioning and navigation systems using GNSS can then represent a great reduction in platform design and operating costs. In contrast with ground applications, GNSS constellations are not optimized for space applications. In [1], the authors presents a study to determine the application requirement of a GNSS receiver for satellite. Those requirements are mainly expressed in term of sensitivity, Doppler range of search and Time Of Acquisition (TOA). In [2], we have adapted the acquisition algorithm proposed by D. Akopian [3] to our application requirement, i.e., a GNSS receiver compatible for both GPS and Galileo signal and able to deliver positioning information for all type of orbits (Low, Medium and Geostationary Earth Orbit).

The resulting algorithm has a moderate complexity and a great flexibility. It can be factorized in a small set of elementary modules that can be configured and combined in various ways to process both GPS and Galileo current and future signals. In this paper, we present the final refinement of the Akopian's algorithms on the selected hardware platform. The architecture has been prototyped in a Xilinx FPGA and is composed of a softcore embedded processor and hardware computation modules.

After a brief recall on GNSS signals, the second section describes the acquisition process and the algorithm that will be implemented in order to acquire GNSS signals. This description shows the different stages involved in the acquisition process. The next section shows the necessary steps toward an effective implementation of the receiver. The design methodology and tools used during this project are also presented. The fourth section presents the design and implementation of the acquisition algorithm of the GNSS receiver. An example of early performance analysis and optimization shows the benefits of this design process. Finally, section five draws the conclusions and the perspectives.

II. Acquisition algorithm

A. The GNSS L1 signals

The algorithm is adapted to process the GPS-L1 Coarse Acquisition (C/A) signal as well as the Galileo-E1 Open Service (OS) signal. These are CDMA signals, data are therefore modulated by a spreading code. For both signals, the expression of the sampled complex signal at the input of the receiver can be defined by:

$$s_n = \sqrt{C}D(n-\tau)S_e(n-\tau)e^{j\varphi n} + n_n \qquad (1)$$

With:

- τ_n: the time delay due to the propagation of the signal.
- C: the power of the signal.
- D: the data signal.
- S_e is the spreading signal.
- φ_n: the carrier phase.
- n_n: an additive Gaussian white noise.

The GPS-L1 carrier frequency is $f_{L1} = 1575.42MHz$. The L1 C/A signal is modulated by a 1023-chip spreading code at a chip rate $f_{CA} = 1.023Mhz$, the pseudo-period is then 1ms [4]. This signal is also named BPSK(1), taking f_{CA} as reference. The data rate is 50 $b.s^{-1}$, one bit is therefore

modulated by 20 spreading code.
The Galileo-E1 signal is emitted at the same carrier frequency than GPS-L1. The modulation of the signal by a sub-carrier, $sc_s(n)$, permits a high degree of spectral separation from GPS-L1 signal (see figure 1). The analytic expression of the spreading code is presented equation 2.

$$S_e(n) = C(n)sign(\sin(2\pi f_{sc} n)) = C(n)sc_s(n) \quad (2)$$

The signal modulated by this sub-carrier is called BOC(n,m) (Binary Offset Carrier) with $n = f_{sc}/f_{CA}$ and $m = f_c/f_{CA}$, f_{sc} is the frequency of the sub-carrier and f_c is the frequency of the spreading code. The Galileo E1 OS signal is a combination of 2 BOC modulation and its power spectral density is presented equation 3 [5].

$$\Phi_{CBOC}(f) = \frac{10}{11}\Phi_{BOC(1,1}(f) + \frac{1}{11}\Phi_{BOC(6,1}(f) \quad (3)$$

It is called CBOC(6,1,1/11). It can be demodulated as a BOC(1,1), the correlation loss is then $L_C = 0.84\ dB$ [7]. The spreading code length is 4092 chips and the pseudo-period is then 4 ms. The data rate is 250 $b.s^{-1}$; thus each data value multiplies a single pseudo-period of the spreading sequence. Moreover, a pilot channel is provided. This dataless channel allows to increase the coherent integration time in the correlation process.

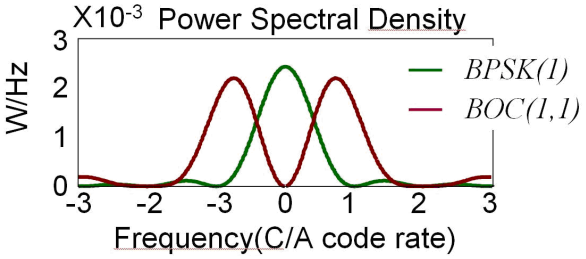


Fig. 1. Signals Power Spectral Density

B. Signal processing

The Doppler frequency f_d and the code delay τ are considered constant over the coherent integration time. The carrier phase is then:

$$\varphi_n = \varphi_0 + 2\pi f_d n T_s \quad (4)$$

And the signal expression is (from equation 1):

$$s_n = \sqrt{C}D(n-\tau)S_e(n-\tau)e^{j\varphi_0 + 2\pi f_d n T_s} + n_n \quad (5)$$

The purpose of the acquisition is to find a coarse estimation $\{\hat{f}_d, \hat{\tau}\}$ of the unknown value $\{f_d, \tau\}$. The received signal is correlated with a local replica $S_e(n-\hat{\tau})$ of the spreading code in order to search the correlation peak (see figure 2) along the time and frequency axis. The GPS C/A code is 1023-chip long. The sampling frequency is determined from the signal bandwidth. It has been chosen here as equal to twice the spectrum bandwidth: $Fs = 1/Ts = 4.096\ MHz$. Thus, the number of samples per chip is 4096/1023 and the search

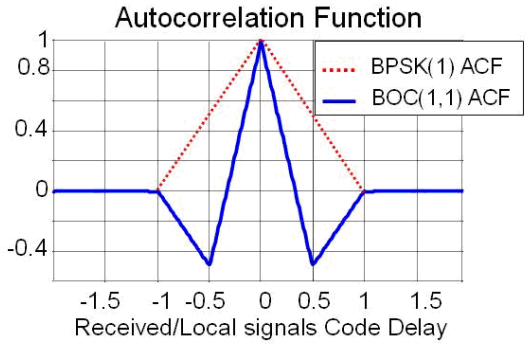


Fig. 2. ACF for BOC(1,1) and BPSK(1) signals

over the whole time axis will be performed for 4096 values of the code delay (τ).
For the Galileo OS signal, the spreading code is 4092-chip long and the sampling frequency is also chosen as equal to twice the spectrum bandwidth: $Fs = 1/Ts = 8.192\ MHz$. The number of samples per chip is 8192/1023 and the search over the whole time axis will be performed for 32768 values of the code delay.
Along the frequency axis, the search is related to the receiver application. The Doppler frequency range is 84 KHz for a LEO mission, whereas it is 16 KHz for a GEO mission [2]. For both signal the coherent integration time (T_{coh}) is 8 ms and the frequency step is chosen as ([1]):

$$\Delta f = \frac{1}{T_{coh}} = 125\ Hz$$

The search over the frequency axis will be performed for 84000/125=672 of the Doppler frequency (f_d) in case of LEO mission, 16000/125=128 in case of GEO Mission.
Thus, the search over the whole array $\{f_d, \tau\}$ is a time consuming process. The number of correlations to be performed is shown in the following table I.

Application	GPS C/A	Galileo OS
LEO Mission	4096 x 672	32768 x 672
GEO Mission	4096 x 128	32768 x 128

TABLE I
NUMBER OF SEARCH BINS

The threshold of the correlation decision, shown figure 3, is fixed to obtain a false probability equal to 10^{-3}. The value of the miss-detection probability depends on the C/N_0 ratio. In the frame of this study, the miss-detection probability must remain smaller than 10^{-1}. This value can be easily reached in the frame of LEO mission (see [1]). For GEO mission the integration process must be performed in two stages:

- A coherent integration over a duration of 8 ms which is adapted to the frequency step.
- A non-coherent integration adapted to the C/N_0 ratio in order to provide a miss-detection probability equal to

10^{-1}.

The correlation process is shown figure 3. A non-coherent square-law detector is added in order to eliminate the carrier signal after the coherent integration. A miss-detection probability of 0.1 is considered over the following study. The coherent integration is obtained from $N1 * N2$ samples where:

- $N1$: number of samples over the spreading code sequence.
- $N2$: number of spreading sequences over the coherent integration.
- $N3$: number of non-coherent integrations.

The value of $N3$ will be set to 1 for LEO mission. It will be set to 7 for GEO mission in order to acquire satellite with C/N_0 ratio equal to 30dB-Hz [1].

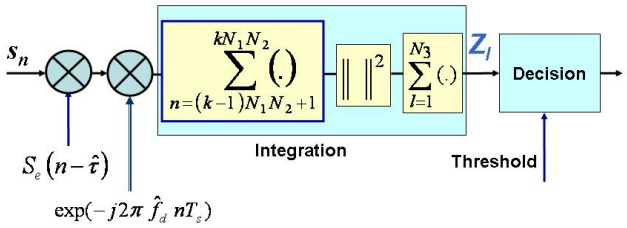


Fig. 3. Acquisition process

For GPS signal, the ACF has a single peak over a pseudo-period and this characteristic is exploited to find the estimate of the propagation delay (τ). For the acquisition of BOC signal, the process is more complex: the BOC ACF has multiple peaks and the risk of wrong peak selection should be considered (see figure 2). Several non ambiguous techniques have been developed to cope with this problem. The Sub-Carrier Phase Cancellation technique (SCPC) described in [6] has been implemented here (see figure 4). On this figure, the local

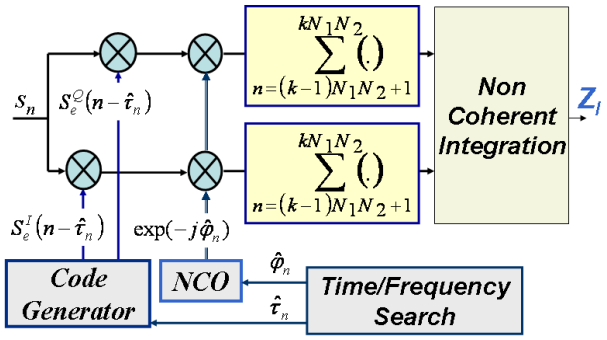


Fig. 4. Unambiguous SCPC technique

references have the following expression:

$$S_e^I = C(sign(\sin 2\pi f_{sc} n)) = C(n) sc_s(n) \quad (6)$$

$$S_e^Q = -C(sign(\cos 2\pi f_{sc} n)) = C(n) sc_c(n) \quad (7)$$

The resulting non-ambiguous ACF is represented on the figure 5. To summarize this section, the main parameters of the

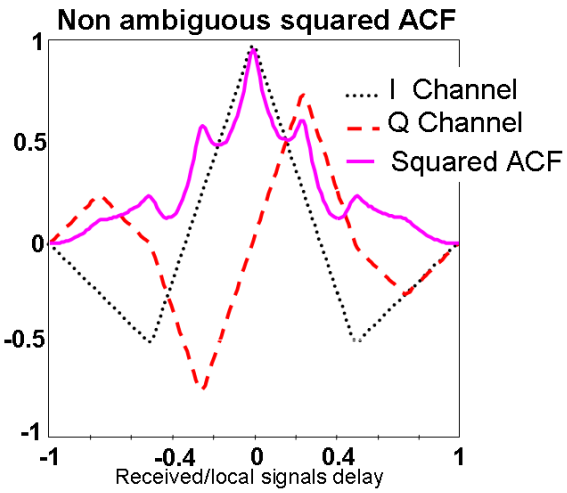


Fig. 5. Non ambiguous ACF

acquisition module are shown in the table II for each signal and satellite mission. With:

	GPS/LEO	GPS/GEO	Galileo/LEO	Galileo/GEO
N1	4096	4096	32768	32768
N2	8	8	2	2
N3	1	7	1	7
Na	2752512	524288	5505024	1048576
Nc	1	1	2	2

TABLE II
SEARCH ARRAY SIZE

- Na : Time/frequency Array size.
- Nc : Number of channels.

C. Acquisition algorithm

The previous section shows how complex is the acquisition process. Fast acquisition approaches, based on the correlation by Fast Fourier Transform (FFT), could be used in order to reduce the delay of the acquisition processing. These methods lead to reduce the number of arithmetic operations for the computation of the autocorrelation function. Thus, many designs use a one-dimensional FFT to explore the time axis whereas the frequency axis is swept by frequency translation. We propose here a technique, based on a size scalable FFT in order to explore the time and frequency axis at the same time, without frequency translation. The architecture is described in [3], it leads to a reduction of 50% of the number of butterfly computation for the search of 1 satellite in the case of LEO application, compared to the *classic* FFT method. The acquisition algorithm for 1 channel and 1 coherent integration is shown in the figure 6. Once the $N1 * N2 * N3$ samples have been collected over the integration time, the following process is applied:

1) A FFT is performed on the $N1 * N2$ samples.

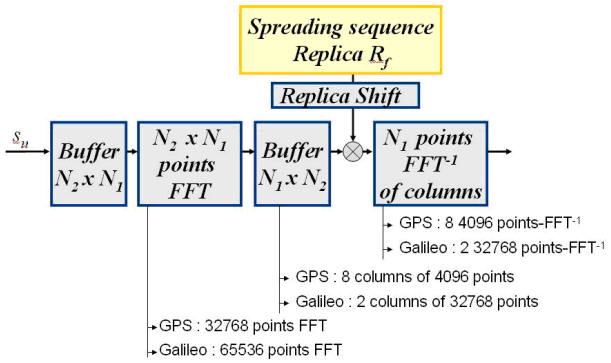


Fig. 6. Correlator unit

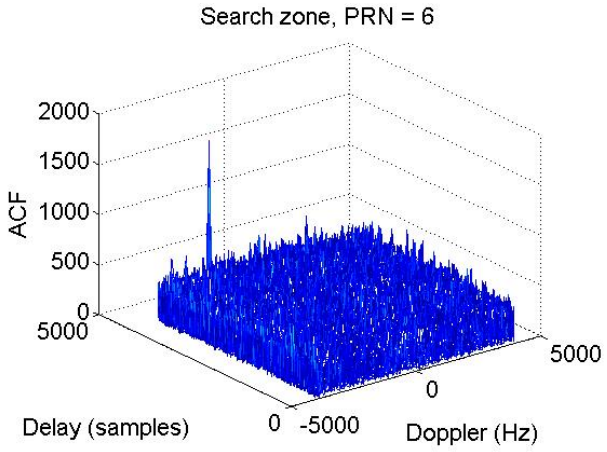


Fig. 7. Acquisition of GPS SVN36 BlockII-A (PRN6)

2) The output is organized to provide $N2$ columns of $N1$ samples. Each column corresponding to one pseudo-sequence period.
Let $r = r_1 r_2$ with $r_1 = 1..N2$ and $r_2 = 1..N1$.

$$x_2(r_1, r_2) = x_1(r1 + N2r_2) \tag{8}$$

3) The correlation is performed in the frequency domain on the $N2$ columns for $N2$ values of the Doppler frequency.
4) $N2$ inverse FFT are performed in order to explore the frequency domain over a range equal to $N2 * 125\ Hz$ (1KHz for GPS, 250Hz for Galileo).

These two last stages must be iterated several times in order to cover the whole frequency domain (see table III). And

Constellation	GPS		Galileo	
Case study	LEO	GEO	LEO	GEO
Doppler search zone (kHz)	84	16	84	16
Coarse frequency search zone	1 kHz	1 kHz	250 Hz	250 Hz
Number of iterations	84	16	336	64

TABLE III
NUMBER OF ITERATION OF THE LAST 2 STAGES

the whole process is iterated $N3$ times for non-coherent integration.
Figure 7 presents the result of the algorithm applied on real signal. The signal found is from GPS Satellite Vehicle Number (SVN) 36. The Pseudo-Range Number code is PRN6. It has been acquired with 8 ms of coherent integration and 1 non-coherent integration and the estimates $\{\hat{f}_d, \hat{\tau}_n\}$ are $\{-3.625kHz, 3024\}$.
The next section presents the design methodology that has been implemented in order to prototype this algorithm in a FPGA.

III. METHODOLOGY

A. Context

Core electronic components for satellite payloads are mainly ASICs and OTP (One Time Programmable) FPGA due to the constraints of space environment. One of the main drawbacks of these components is that they are not reconfigurable. Therefore, applications can't be upgraded during the life span of the satellite (from a few years up to 20 years). Embedded processors can provide reconfigurability and allow the designer to achieve high component integration. Although environments nowadays permit to perform transactional level modeling of hardware/software systems, none of them offers the possibility to easily modify the architecture, to easily migrate tasks from hardware to software (and vice versa) without recoding (due to the different languages) and to completely refine up to a hardware-software embedded product. The two main aims of this project are to design a receiver for space application and to study and implement a comprehensive and integrated co-design flow, from algorithm specification to on-chip implementation.

B. Versatility

The algorithms have been studied in order to find commonalities that could be factorized. Thus, we have reduced the navigation algorithms into a set of basic processing algorithms that can be configured and combined in various ways to build a signal processing chain for signal acquisition and tracking. Each of these processing modules can be configured with a small set of parameters, such the length of a FFT, the noise-band of a PLL or the downstream module to hand on the results. The idea is to implement a *pool* of modules that can be configured in order to form a processing chain. The real-time flexibility of the receiver will be ensured by *on-the-fly* configuration of signal processing chains during mission. A configuration arbiter will select and configure available modules in order to build a chain for a specific signal. We have also identified the parts that would have to be re-programmable in order to keep up with future signals. The recoding of software modules will permit long-term flexibility.

C. design methodology and tools

One of the goals of this project is to study and implement a comprehensive and integrated co-design flow, from algorithm specification to on-chip implementation. In order to fulfill this

objective, the design language should be common to hardware and software flow. Thus, tedious language conversions and time consuming validation phases would be avoided. The higher level of abstraction would also lead to a reduction of design time. We have chosen SystemC, a set of C++ classes and macros, which is an emerging standard for the description of hardware models. SystemC has been approved by the IEEE Standards Association as IEEE 1666-2005. It is then independent of the tools and it can be used with several ESL flow. High level synthesis tools, supporting C-based languages, are now available from EDA providers. Some tools support directly SystemC and, for the others, a SystemC file can be parsed into C/C++ for hardware synthesis or for software compilation.

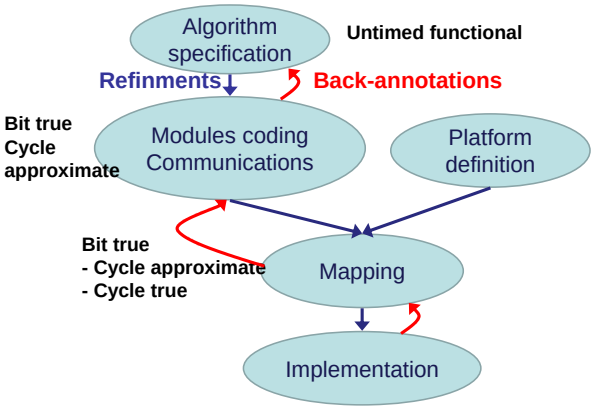


Fig. 8. Y approach

We have adopted the well-known 'Y' approach which separates the application from the architecture (see figure 8). First, the receiver algorithms have been specified with SystemC on a software development tool : Microsoft VisualC++ express, which is the free version of Microsoft Visual Studio.
For architectural exploration down to prototyping, we have used Space Studio, a tool from Space Codesign Systems. This tool allows the designer to partition the algorithm into modules and to map these modules onto an architecture [8]. This architecture, also named virtual platform, consists of a set of IPs modeled at the transactional level. The aim of the methodology is to delay the choice of a partition as late as possible in the design flow. Simulation and implementation analysis are then available in order to help the designer to choose an architecture.
Once validated, the user modules have been synthesized with Catapult, a high-level synthesizer from Mentor Graphics. Space Studio generates a Catapult project for each module, including all the dependencies. Catapult offers the possibility to analyze and explore the micro-architecture of hardware modules in order to converge on the optimal solution in terms of power, performance, and area. The VHDL generated has been synthesized with Precision from Mentor Graphics. Thanks to the flow, several architectures can be easily generated and compared.
The receiver has been prototyped on a Xilinx FPGA. Space Studio generates the Xilinx EDK project, which is then used to place/route and to program the component.
This design flow can be followed top-down for implementation refinement or bottom-up for detailed simulations and design space exploration. The next section presents the details of the implementation of the acquisition algorithm.

IV. Implementation of the Acquisition Algorithm of the GNSS Receiver

A. Algorithm modeling

During the system analysis and specification phases, the signal processing algorithms have been studied with Matlab. It is a standard tool for fast analysis of signal processing algorithms but it isn't suited for implementation. The algorithms have then been modeled with SystemC. Due to the C++ syntax and the templates provided, SystemC allows the designer to achieve fast implementation of the algorithms. The purpose of this untimed functional (UTF) model, also named Programmer view (PV) model, is to validate the algorithms as well as to analyze the data flow between the various tasks. SystemC provides also data types, such as fixed point, the effects of the quantization on the computations can then be easily studied. Morover, the high level of abstraction of the models enables fast exploration of the algorithms. Unlike Matlab, the time can also been modeled with SystemC, as in VHDL, Verilog or Simulink. Thus, time-dependent effects, such as the Doppler, can be precisely modeled. The algorithms has been validated using real GNSS signals captured by our L1 wide band radiofrequency front-end. We have also used signals generated with Matlab and with a Spirent generator belonging to the CNES (French space agency) in order to simulate the specifities of space applications, such as the International Space Station (ISS) or the satellite Spot5. Once validated, the algorithms has been divided into modules in order to be mapped on an architecture. At early stage of the design, Space Studio allows the designer to partition an algorithm into modules for fast architectural exploration. It provides debug and monitoring capabilities to validate the functionality as well as the partition and early analysis of the performances.
The most difficult task was then to decompose the sequential application into modules in order to exhibit enough parallelism to exploit the concurrency offered by the architecture. The modules can be untimed (PV) or approximately timed, Programmer View Timed (PVT), for more realistic assessment of performances. The modules should not be too small because each will be implemented with at least one communication interface, to the bus or to another module. Then, many modules lead to many interfaces and a larger physical implementation. The aim was then to decompose the algorithm into large coherent modules with common configuration parameters. The micro-architecture of the modules would be optimized later with the high-level synthesis tool.
Communication between SystemC user modules should be provided here, either by message-passing based on rendezvous

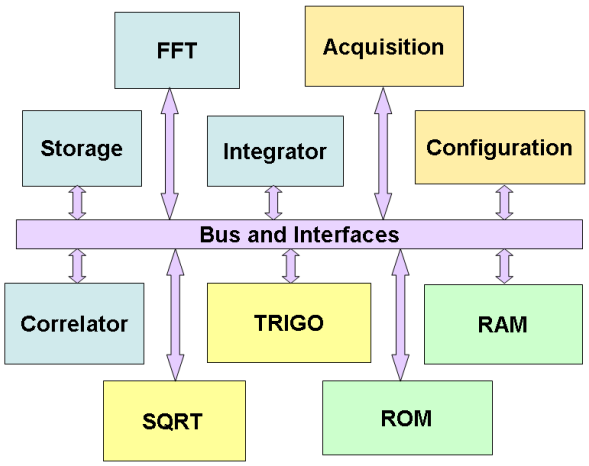


Fig. 9. Module partitioning

principles, or by using shared memory to exchange data. The communication protocol can be based on simple untimed crossbar or more complex cycle approximate model of standard busses, such as ARM AMBA or IBM CoreConnect. The communication is provided by simple functions, ModuleRead and ModuleWrite, that are independent of the channel or the implementation, software or hardware.

The acquisition algorithm is relatively straightforward, the 3 main modules are the variable-size *FFT*, the *Correlator* and the non-coherent *Integrator*. Due to the large number of data to process, up to 262144 (2^{18}) points, the FFT has no local memory. The results of the internal computations are stored in the external memory. The samples have a dynamic of 16 bits for real and imaginary parts, thus 32 bits for a complex number. The trigonometric function is stored in a local memory, the first quadrant is divided into 2048 samples with a 6-bit dynamic. The trigonometric module is extern to the FFT module because it is also used by the *Storage* module in order to remove the intermediate frequency of 9.54MHz. The local replicas of the spreading code are stored in an external ROM memory, these codes are processed by the *FFT* module at the power-up of the system in order to generate to replicas used by the *Correlator* module. The non-coherent *Integrator* module is able to process directly GPS signals or to integrate In-Phase and Quadrature Galileo signals for SCPC technique (see section II-B). The SCPC computation is only processed when a BOC signal is detected in order to remove the ambiguity. The *Acquisition* module controls the flow of the acquisition algorithm, such as : the Doppler search zone, the number of integrations, the acquisition threshold... The computations between modules are scheduled by using blocking or non-blocking transactions.

As the commonalities of the acquisition algorithms have been factorized during the previous design phase, each module have to be configured with a small number set of parameters, such as the number of samples to process or the memory mapping. So, a global set of parameters is defined for each GNSS constellation, GPS or Galileo, and type of signal. A control module configures the computation modules for a given application.

The same sets of signals, used in the previous design phase, have been used to validate the partitioned algorithm. This model is bit-true and cycle-approximate (BTCA), and the purpose is to validate the partition, the communication and the synchronization between the modules.

Thanks to the debug and analysis tools, bus traffic can

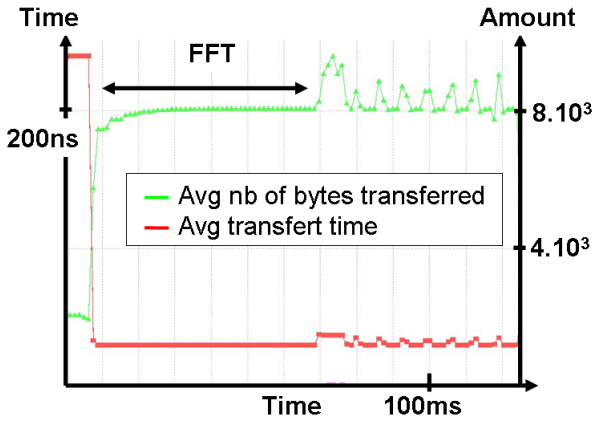


Fig. 10. Bus traffic analysis

be analyzed early in this phase in order to find possible congestion. The early performance assessment allows the designer to modify the function or the architecture long before implementation phase, thus reducing the costs and the design time. One module was particularly time consuming, as shown in figure 10. A detailed analysis of the transactions on the bus shows that the FFT module spend a lot more time on transactions than on computations, unlike other modules such as the non-coherent integrator module. On this example of C/A GPS signal acquisition, the FFT has to process 32768 complex samples, it means 32768 reads and 32768 writes for one stage and a total of 15 stages, i.e. 983040 bus transactions. As the FFT is versatile, it has only one butterfly module and the intermediate samples are stored in the external memory.

We have then adopted a multi-level FFT architecture in order to reduce the number of accesses to the external memory. The FFT can be decomposed into successively smaller FFTs and multiplications with phase factors [9]: a N-point FFT ($N = n1 * n2$) can be decomposed into successively n2 $n1 - points$ FFT and n1 $n2 - points$ FFT.

Thus, the optimized *FFT* module is based on one flexible FFT, able to process up to 512 (2^9) points in a local memory, with a phase rotator and a configuration module. for example, a 32768 (2^{15}) points FFT can then be decomposed into a first level of 64 FFT512 (2^9), followed by a phase rotation and a last level of 512 FFT64 (2^6), i.e. 131072 bus accesses. The number of bus accesses performed by the *FFT* module is then reduced and the computation time of the FFT is divided by

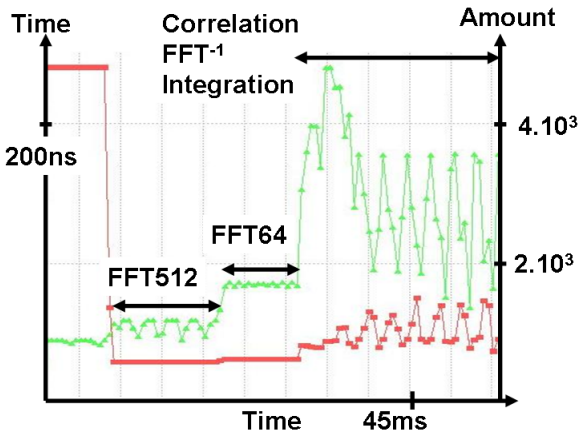


Fig. 11. Bus traffic analysis

3, in the case of a 32768 points FFT. The *FFT* module is composed of an address generator, a butterfly, a 512x32-bit local memory and a complex multiplier for phase rotation.
The next phase was to define a virtual platform and to map the modules onto it.

B. Architecture definition and mapping

Due to the constraint of the space environment, the receiver will have to be implemented in an ASIC. However, in order to completely validate the design flow, the receiver will be prototyped in a FPGA. Space Studio propose an IP library for Xilinx FPGAs, we have used then the MicroBlaze and the OPB bus.

The goal was to identify the blocks that could evolve during

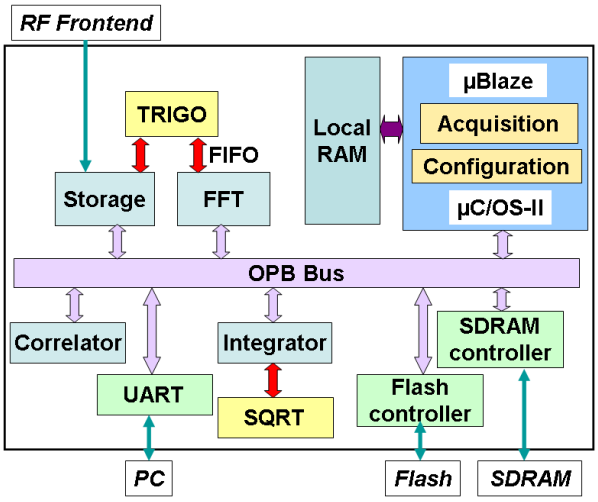


Fig. 12. Virtual Platform mapping

the life of the satellite. These blocks would benefit the software flexibility and should be implemented on processors. The most computationally intensive blocks should be implemented on hardware and optimized with the high level synthesis tool. As shown on figure 12, the low level modules, *Trigo* and *SQRT*, are not connected to the bus, but directly to the modules with bi-directional FIFOs. They don't interact with the memories and the direct links are faster than the bus transactions. High computation modules, *FFT*, *Correlator* and *Integrator* are synthesized on hardware whereas control modules, *Acquisition* and *Configuration* will be compiled on software and scheduled by real time OS : μC/OS-II. As there is no differences in communication for a hardware implementation and software one, it is easy to change and test various mapping on the platform. We have opted for the platform shown in figure 12. The next section presents the results of the implementation on a FPGA.

C. Implementation and FPGA prototyping

The acquisition module have been prototyped on Xilinx Virtex2Pro FPGA : 2VP30. Space Studio generates the Catapult project files in order to synthesize the hardware modules. The implementation of each module can be then precisely controlled and analyzed. Once the modules synthesized, the latency of each modules can be interpreted from Catapult schedule view and then back-annotated in the Space Studio SystemC model. After this design phase, the simulations are cycle accurate, even for the hardware modules. The clock frequency of the receiver is 100MHz, table IV presents some timing for the complete search for all Doppler bins and all GNSS satellites. Real acquisition time is significantly lower because the search process stops once a satellite is found. As an example, the acquisition duration for ground application is also presented.

Constellation	GPS			Galileo	
Application	LEO	GEO	Ground	LEO	GEO
Doppler search zone (kHz)	84	16	8	84	16
Non coherent integration	1	7	1	1	7
Max. acquisition time	2'15"	3'	13"	18'	

TABLE IV
SEARCH ACQUISITION TIME

The time to process Galileo signals is much greater due to the narrower coarse frequency step (250Hz instead of 1kHz for GPS) and the SCPC computation which increase the computation workload. These computation times can be divided by reducing the sampling frequency, from 4.096MHz to 2.048MHZ for GPS C/A and from 8.192MHz to 4.096MHZ for Galileo OS. As the receiver is entirely configurable, this change is easily done by the configuration module.
The acquisition system occupies 79% of the slices of a 2VP30 FPGA, 10837 out of 13696. As shown on table V, the interfaces between user module and busses occupies 25% of the FPGA. This is mainly due to the 4 interfaces to the OPB bus of the main modules : *Storage*, *FFT*, *Correlator* and *Integrator*. As exposed in section IV-A, the granularity of user modules should not be too small in order to implement as few bus interface as possible. Instead, the micro-architecture of the modules should be optimized with the high level synthesis

	Slices	DFF	Multipliers	BlockRAM
Available	13696	27392	136	136
User modules	4744	4723	14	9
Interfaces	2780	2644	0	8
Platform IPs	3313	3592	3	80
Total	10837	10959	17	97

TABLE V
AREA OF THE ACQUISITION SYSTEM

tool. The platform accounts for 30% of the area occupied. It is composed of a lot of small modules, such as the *Programmable Interrupt Controler*, *the UART*... and the bigger *μBlaze* (1407 slices). It can't be optimized a lot further, only few small modules could be removed, like the *Debugger*.
On the other hand, user modules account for 44% of the area. These modules are clearly not optimized, more than 1200 slices are used for the *Correlator* module. The micro-architecture has not been studied yet, this is the on-going phase of the project. As an example, multipliers has been implemented on DFF when a lot of hardware multipliers are still available.

V. CONCLUSION

This study presents a technique to acquire GNSS-L1 signals, in a receiver designed for space environment. The GPS-L1 C/A signal and the Galileo-E1 OS signal, the LEO mission and the GEO mission are considered. Acquisition algorithms are proposed in order to process GPS and Galileo signals. The architecture of an acquisition system, based on 1 scalable FFT core, is described. This architecture can be configured for the two missions and the two signals.
We have presented a coherent co-design flow for MPSOC. The acquisition algorithm has been modeled on a virtual platform and implemented in a FPGA using this tool flow. Thanks to the high level language, common to hardware and software flow, as well as the methodology, the design time has been reduced to approximately 3 months, from SystemC specification down to implementation, without prior knowledge of the tools.
We have also implemented a stand-alone flexible component that can be self reconfigurable. Moreover, the hardware/software codesign permits long term versatility for applications that can have a life span of 20 years.
The on-going step of the is the optimization of the micro-architecture. Thus, we need to explore the functionalities of the high level synthesis tool, Catapult. We will also experiment and compile rules for coding for hardware implementation, we have shown that the coding style of a same function can have a great impact on the implementation, as in VHDL or Verilog. Several architectures and module implementations will be compared for power, latency and area.

APPENDIX A
REFERENCES

[1] V. Calmettes, A. Dion, E. Boutillon, E. Liegeon, "Fast Acquisition Unit for GPS/GALILEO Receivers in Space Environment", *proc. of ION National Technical Meeting*, San Diego, Jan., 2008.
[2] A. Dion, E. Boutillon, V. Calmettes, "Reconfigurable GPS-Galileo Receiver for Satellite-based Applications", *proc. of ION Global Navigation Satellite Systems*, Fort Worth, Texas, Sept., 2007.
[3] D. Akopian, "FFT based satellite acquisition methods", *IEEE Proc. Radar, Sonar and navigation*, Vol. 152, No 4, August 2005, Pages 277-286.
[4] Navstar GPS Space Segment/Navigation User Interfaces (IS-GPS-200D), 2004.
[5] Galileo Open Service Signal in Space Interface Control Document (OS SIS ICD) Draft 1, 2008. Available: http://www.gsa.europa.eu/go/galileo/os-sis-icd
[6] V. Heiries, D. Roviras, L. Ries, V. Calmettes, "Analysis of Non Ambiguous BOC Acquisition Performance", *proc. of ION Global Navigation Satellite Systems*, Long Beach, Sept., 2004.
[7] N. Hoult, L. E. Aguado, R. Crescimbeni," Performance comparison of TMBOC and CBOC signals", *Proc. of CNES workshop on Galileo signals and signal processing*, Toulouse, 2006,
[8] L. Filion, M-A. Cantin, L. Moss, G. Bois, E. M. Aboulhamid, "Space Codesign: A SystemC Framework for Fast Exploration of Hardware/Software Systems", *proceedings of DVCON*, San Jose, Feb.,2007.
[9] G. Zhong, F. Xu, and A. N. Willson, Jr., "A Power-Scalable Reconfigurable FFT/IFFT IC Based on a Multi-Processor Ring", *IEEE Journal of Solid-State Circuits*, Vol. 41, pp. 483-495, Feb. 2006.