FPGA-Based Architecture for High Throughput, Flexible and Compact Real-Time GNSS Software Defined Receiver

B. Sauriol, *Member*, *ION*, *Ecole de Technologie Supérieure*
R.Jr. Landry, *Ecole de Technologie Supérieure*

BIOGRAPHY

Bruno Sauriol received an Electrical Engineering Degree from the University of Sherbrooke (Canada) in 2006. He is currently undertaking postgraduate studies as part of a master degree in engineering at the Electrical Engineering Department of Ecole de Technologie Supérieure (ETS), Montreal (Canada). His research interests include Global Navigation Satellite Systems (GPS and Galileo), embedded systems design and real time digital signal processing. He is actually working on the development of a hybrid GPS and Galileo receiver and he is also playing an important role in the development of a GNSS constellation signal simulator.

René Jr. Landry received a PhD degree at SupAero / Paul-Sabatier University and a Post Doc in Space Science at the Centre National d'Etudes Spatiales (CNES), both at Toulouse, France, in 1997 and 1998 respectively. Since 1999, Professor Landry is involved in receiver design and robust navigation in severe environment for the Canadian Navigation and Communication Industries. One of his major interest concerns the development of new innovative mitigation techniques for GNSS receiver robustness design including those of electronic Inertial Navigation System based on low cost MEMS. He is actually working on several digital signal processing applications in high resolution navigation, receiver design, indoor navigation and inertial navigation systems.

ABSTRACT

The advent of Galileo, along with the GPS and GLONASS modernizations, will make available a tremendous number of signals by 2015 to civilians for precise and reliable positioning. Nonetheless, the real-time integration of all those Global Navigation Satellite System (GNSS) signals remains a challenge and only a few solutions have been proposed yet. In this paper, a Field Programmable Gate Array (FPGA) based Software defined GNSS Receiver (SGR) is presented as a promising solution for fast real-time integration and as a perfect tool for Research and Development (R&D) in the field of navigation.

The SGR is a compact L1 GPS C/A and Galileo hybrid receiver designed to incorporate up to 36 channels in the same FPGA. It has a sampling frequency of 60 MHz along with a maximal bandwidth of 24 MHz and it offers a real-time position solution updated at rates of at least 20 Hz. Yet, the receiver delivers centimetric position stability over short time periods and it demonstrates the capability to operate with signals Carrier-to-Noise-density ratios (C/N_0) as low as 20 dB-Hz.

First, a review of the SGR architecture is done in this paper. Then, resources evaluation and comparison with other available FPGA is achieved. Finally, real-time test results are presented and commented.

1. INTRODUCTION

In the past years, designers have mostly developed GPS and GLONASS receivers based on Application-Specific Integrated Circuits (ASIC) as a hardware approach. With the arrival of Galileo and other brand new GNSS signals, ASIC design stays an interesting option as it allows dense integration combined with high processing speed. However, ASIC design remains a tedious, expensive and non flexible process. It is therefore not well suited for today R&D that has to deal with reduced budgets and very fast design to market cycle times.

A more recent approach, namely Software Defined Radio (SDR), offers on the other hand the flexibility and short turnaround time needed by GNSS receiver developers and manufacturers. Originating from the telecommunication world, SDR is an emerging concept that has still a fuzzy definition. Yet no official consensus exists in the area of satellite navigation, resulting often in an erroneous classification of the multiple kinds of SGR. This paper makes first a short review of the SDR concept in order to

clarify this aspect and to provide a refined definition for the navigation community. A brief classification is also provided to facilitate the comprehension of what is a SGR.

Despite of its multiple advantages, the current SGR approach however suffers from low throughput. This problem is mainly due to the Intermediate Frequency (IF) signals being too often processed by an overloaded single processor. As a consequence, current Software defined GNSS Receivers (SGR) will be most likely limited to PC computer-based kind of platforms [1]. To help with this problem, this paper presents a FPGA-based receiver as a promising solution for real-time embedded SGR. In contrast with previous alternatives, FPGA-based design allows both high throughput and flexibility at the same time. As its name indicates, the FPGA is first of all a programmable device, so the design is made faster and more flexible. Second of all, FPGA allows the implementation of real-time parallel digital architectures similar to those encountered in ASIC. With operating clock frequencies nowadays ranging from 100 to 500 MHz [2], it therefore offers sufficiently high throughput in all situations. Moreover, as FPGA are getting larger and cheaper, precise and reliable SGR might be developed in a more cost-effective way.

As it is the case with most real-time GNSS receivers, the SGR presented here is subdivided in three fundamental sections: The Radio-Frequency (RF) front-end, the Intermediate Frequency (IF) and the baseband signal processing stage. Since the FPGA role lies mainly at the IF stage, the RF and baseband sections are therefore succinctly reviewed. The emphasis is instead placed onto the IF architecture in order to illustrate the benefits of using a FPGA within a GNSS receiver design. Amongst those benefits, a massive reduction of the baseband processor load via preprocessing and the creation of custom hardware accelerators is discussed. The availability of very large signal bandwidth (> 50 MHz) and flexible digital signal processing are also presented as other advantages.

The quality of a SGR is mainly characterized by its precision and its robustness. In that purpose, a series of experimental tests have been performed onto the developed receiver to validate its performances. Those tests could be realized real-time in static and dynamic scenarios with the use of a GNSS signal simulator. Later on, static tests could also been performed directly outside with real signals. However, this paper only presents the test results for 12 GPS-L1 C/A channels since the Galileo channels are still under development as this paper is being written.

This paper is organized as following: Section 2 presents a review of the SDR concept and a definition of the SGR is also proposed. In Section 3, a basic SGR classification is presented. Section 4 introduces the FPGA-based SGR general architecture and the IF stage is described more thoroughly in Section 5. The FPGA resources usages are studied in Section 6 and the receiver's performances are discussed in Section 7. Section 8 introduces the real-time experiments protocol and the results are displayed in Sections 9 and 10. The last section is dedicated to some conclusions and perspectives.

2. SGR DEFINITION

The interest for SDR has really started in the early 1990s with the initiation of the Joint Tactical Radio System (JTRS) program by the U.S. Department of Defense (DoD). Since then, the SDR concept made significant progresses and grew in popularity in many fields of application of the industry. Still, a lot of confusion surrounds the concept itself while no official definition truly exists. At best, the SDR Forum describes SDR as being "radios that provide software control of a variety of modulation techniques, wide-band or narrow-band operation … and waveform requirements of current and evolving standards over a broad frequency range" [3]. Others risk themselves with more general definitions, as an example seen in Wikipedia: "A software-defined radio (SDR) system is a radio communication system which can tune to any frequency band and receive any modulation across a large frequency spectrum by means of a programmable hardware which is controlled by software" [4].

As every domain perceives the SDR in a different way, a general definition losses most of its functional signification and so might be of less interest. Instead, this paper proposes a more generally accepted definition of what is a SDR receiver to the navigation community: a receiver in which signal's demodulation is done digitally by the means of reprogrammable devices.

3. SGR CLASSIFICATION

The SGR is often seen as a receiver in which digital processing is solely performed by a Digital Signal Processor (DSP) or a PC computer. Because FPGAs are quite different from DSPs and PCs (they are programmed in a different way and are said to be less flexible), FPGA-based SGRs are most of the time put in a different class. This is however a misconception as the FPGA is generally more versatile than most other devices. FPGAs are usually known for their ability to develop logic circuits, but they can also incorporate powerful processors at the same time. For example, the Xilinx Virtex II Pro FPGA can offer up to 2 Power-PC processors in the same device. Also, lots of microcontroller and DSP soft-cores are readily available to be implemented in most brands of FPGA using logic gates. Thus, FPGAs offers all the

processor benefits without the need of developing new hardware all the time. Conclusion: there is no valid reason to put the FPGA-based SGRs in a separated class.

Figure 1 presents the main SGR types. As seen in [1], two main categories are generally considered: Post-Processing and Real-Time Capable SGRs. The first category refers to all the programs and code lines developed for algorithm prototyping, signal analysis and other applications that are not time critical. The other category first regroups the PC-based real-time receivers, a new trend of receivers that are bound to run on laptop or desktop computers solely. The real-time category also regroups the embedded receivers which are dedicated to be run in the field with all it implies (reduced size, weight and power, harsh environments and more). The embedded software receivers usually rely on dedicated components such as DSPs and FPGAs for their development. The SGR presented in this paper is located halfway between the PC-based and embedded categories as it relies upon both a FPGA and a PC computer at the same time. However and as it will be discussed, the computations done in the PC can easily be running on a normal DSP or on a FPGA soft-core processor in order to implement a fully embedded SGR.

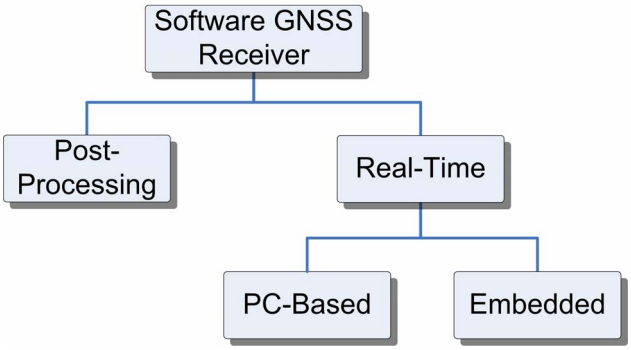

Figure 1: Main Categories of Software Receivers

4. GENERAL GNSS RECEIVER ARCHITECTURE

The developed GNSS receiver is based upon a Xilinx Virtex II 3000 FPGA, a medium size FPGA offering up to 3 MGates (28.7k Look-Up-Tables – LUT). The FPGA is mounted on a Lyrtech VHS-ADC development board equipped with eight 14-bits general purpose Analog to Digital Converters (ADC), 128 MB of onboard Synchronous Dynamic Random Access Memory (SDRAM) and is connected to a PC computer through a 20 MB/s Compact PCI bus [5]. Each ADC has a maximal sampling frequency of 105 MHz and is dedicated to a single GNSS frequency band, such as L1, L2 or L5. The L1 band is the only one to be processed at this moment as the L2 and L5 bands are scheduled for the next project phase.

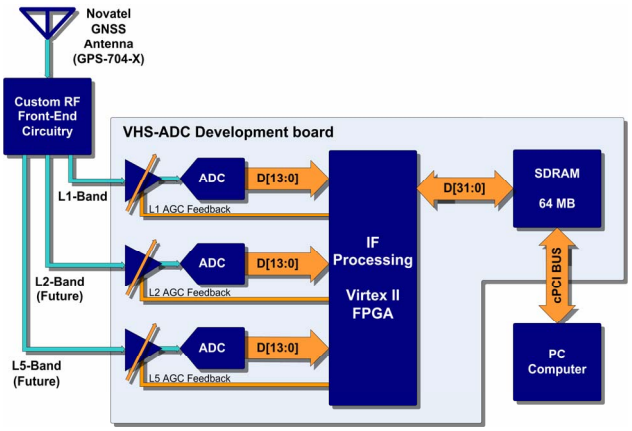

Figure 2: Proposed General GNSS Receiver Architecture

Because RF GNSS signals are much too weak and too high frequency to be digitized directly by the general purpose ADCs, an analog front-end built from Components-Off-The-Shelf (COTS) performs signal conditioning. The front-end consists of an antenna followed by a filter and a pre-amplification stage [6, 7]. Next, a two-step down-conversion stage with proper filtering is implemented in order to reduce the bandwidth and to reject the image introduced by the frequency shift from RF to IF. The local oscillators are two Agilent E4431B signal generators tuned at 1505.42 MHz and 55 MHz in order to down convert the signal from 1575.42 MHz to 70 MHz and 15 MHz respectively. At last, the IF signal is amplified by 72 dB to be digitized correctly by the ADC.

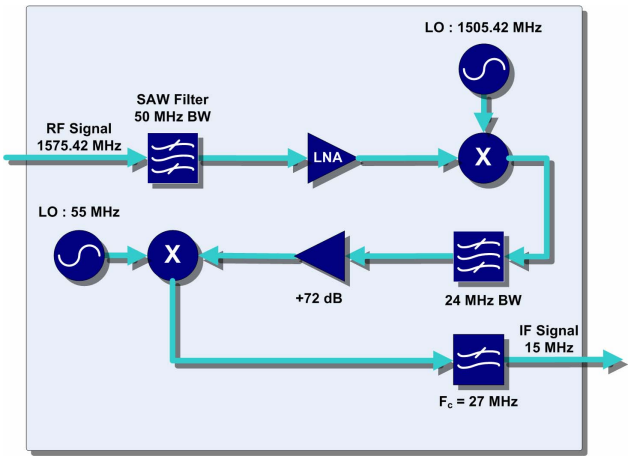

Figure 3: RF Front-End Circuitry of the SGR

The resulting IF signal is centered at 15 MHz and has a 24 MHz bandwidth to accommodate both GPS-L1 and Galileo-E1 full bandwidth signals. The IF signal amplitude is adjusted by a 18 dB Variable Gain Amplifier (VGA) included with each ADC and is digitized at a sampling rate of 60 MHz. Afterwards, the FPGA performs signals acquisition, tracking, demodulation and message decoding. It also stores the recovered navigation messages and measurements such as pseudo-ranges and

Dopplers on the onboard SDRAM. Finally, a PC-computer in charge of programming and configuring the FPGA harvests data from the onboard SDRAM and saves them on its hard disk drive. The PC has also the role of computing a real-time Position-Velocity-Time (PVT) solution since the FPGA does not have this capability for the moment (refer to section 5). At last, a User Graphical Interface (GUI) running on the PC allows the user to configure the receiver and to monitor any internal parameters in real-time.

5. GNSS RECEIVER IF ARCHITECTURE

Compared to PC-based software receivers where the entire signal processing is done in a computer, the developed SGR has its IF processing done in the FPGA (refer to Figure 4). In that purpose, a total of 12 GPS-L1 demodulation channels have been implemented in the FPGA (and 12 more Galileo-E1 channels are still under development). A single functional GPS channel was actually programmed in Very high speed integrated circuit Hardware Description Language (VHDL) as the 11 other ones are mainly replicas. The role of these channels is to demodulate the received signals by first removing the carrier and then despreading the Code Division Multiple Access (CDMA) signals. These operations are done with the use of classic digital Phase Locked Loops (PLL) and Delay Locked Loops (DLL) as seen in [8]. In that purpose, a total of 6 correlators (3 In-Phase and 3 Quadra-Phase) are implemented per channel in a Early-Prompt-Late fashion.

All 12 GPS-L1 channels are driven by the 60 MHz external clock. Though this frequency has been chosen in the early development stage of the receiver (to ensure the FPGA timing requirements could be met), it could be easily pushed up to the ADC limits, i.e. 105 MHz. Therefore, a maximal signal bandwidth of 52.5 MHz could be achieved, which is very interesting for the future Galileo E5a/E5b signals that are supposed to use very large bandwidth AltBOC modulation [9].

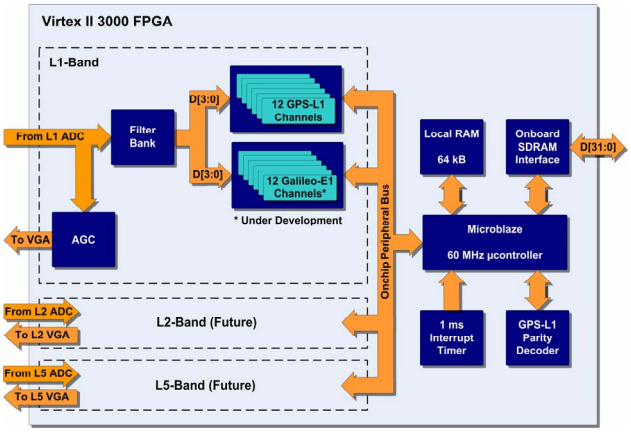

Figure 4: FPGA-Based SGR IF Architecture

Besides the 12 GPS channels, an Automatic Gain Control (AGC) has been implemented in order to drive the external VGA. A filter bank has also been developed (but is yet not tested) so each channel could adapt its bandwidth depending on the circumstances. For example, a large bandwidth should allow a more precise tracking of the correlation peak at the expense of a lowered Signal-to-Noise Ratio (SNR). On the other hand, a narrower bandwidth should increase the channels robustness to noise by enhancing the SNR. Once functional, this module will allow choosing between bandwidths of 2, 4, 12 and 24 MHz (and eventually 48 MHz for the Galileo AltBOC signals).

A Xilinx Microblaze soft-core microcontroller has been added to the FPGA and also operates at a 60 MHz frequency with interrupts generated every millisecond. Primarily dedicated to close all 12 GPS PLL and DLL loops, this soft microcontroller revealed itself enough powerful to carry out most of the baseband processing. As a result, acquisition, tracking and navigation message synchronization/decoding are directly performed inside the FPGA in real time. The microcontroller has however no double precision floating point capability at this time, so the PVT solution is left to an external processor (the PC computer) for simplicity. The use of a soft-core processor with double precision capability instead of the Microblaze is currently under study (this could eventually allow the FPGA to be a complete embedded SGR).

The Microblaze microcontroller has access to 2 different memory spaces: a local memory space of 64 KB and an external memory space of 64 MB. The local memory space is implemented directly inside the FPGA using Xilinx dedicated Random Access Memory Blocks (BRAM) to allow fast program execution. The external memory consists of the onboard SDRAM chip which can be also accessed through the compact PCI bus at the same time. Access to this memory space is slower than for local memory but it allows easy data transfers between the Microblaze and the PC computer.

The word flexibility takes all its sense here as a custom hardware accelerator has been attached to the Microblaze to decode the GPS-L1 messages parity. This hardware accelerator, a core implemented inside the FPGA and that requires a minimal amount of gates, allows the Microblaze to compute the parity in a single clock cycle while it would take more than 80 otherwise. This feature is specific to the FPGA design and is of great interest as it allows a general use soft processor core to be optimized for a specific application. Other hardware accelerator examples could be the add-on of specialized floating point units or of a Viterbi decoder, which can be seen as an economic way to enhance a simple processor instead of relying on more expensive specialized DSPs.

6. FPGA RESOURCE USAGE

The number of gates used by a VHDL design is of great importance in a FPGA as the resources are limited comparatively to an ASIC. The way a FPGA design is done and how much attention a designer pays to the available resources will impact directly onto the minimal FPGA size and speed grade that will be required. The use of bigger and faster FPGA might not be a problem in most general applications, but it really matters when it comes to design embedded systems. The designer should be aware that larger FPGA and increased clock frequencies generally imply higher device costs and power consumption.

Resource usage minimization has thus been a priority all along the GPS-L1 channels development. Because the channels precision impacts directly on the number of FPGA gates consumed, only the 4 Most Significant Bits (MSB) have been kept out of 14 available from the ADC for signal processing. By doing so, more channels can be packed inside a same FPGA without any risk of compromising the GNSS signals quality [10]. As a result shown in Table 1, the FPGA gate consumption is minimal: only 535 Look-Up Tables (LUT) per channel for a total of 6420 LUT for all 12 channels together. For the Microblaze (and all its peripherals), about 8957 LUT are used. Therefore, the entire SGR occupies 15 377 LUT, i.e. 54% of the total FPGA resources. Table 1 also presents these results for a 1 and 2 bits input signals resolution. Few tests also showed that only 40% of the Microblaze processing power was used, so enough gates and processing power are left to easily implement the 12 Galileo-E1 channels under development.

Table 1: LUT Distribution of the SGR (Virtex II 3000)

Input Signal Resolution (# bits)	1	2	4
GPS L1 Channel LUT usage	440 (1.5%)	471 (1.6%)	535 (1.9%)
12x GPS L1 Channels total LUT usage	5280 (18.4%)	5652 (19.7%)	6420 (22.4%)
Microblaze + Peripherals LUT usage	8957 (31.2%)	8957 (31.2%)	8957 (31.2%)
GNSS Receiver Total LUT usage	14 237 (49.7%)	14609 (51.0%)	15 377 (53.6%)

A short study has been done in order to compare the resource usages of the developed SGR to other FPGA available on the market. A large Virtex IV FX140 FPGA from Xilinx has been chosen along with two equivalent FPGAs from Altera: the Cyclone II 35 and the Stratix II 130. Each FPGA LUT usage for a 12 GPS-L1 SGR is shown in Table 2 [2, 11] (for this study, a 4 bits input signals resolution has been taken). The maximal number of GPS-L1 channels per FPGA has also been evaluated for various input signal resolutions (see Table 3). The evaluation has been done with the assumption of a single Microblaze per FPGA in all cases. Because the Microblaze is a Xilinx proprietary core, a NIOS II soft-core processor must be normally considered for implementation in Altera FPGAs. Since the resource consumptions are similar to both Microblaze and NIOS II cores, the results presented in Table 3 have therefore been approximated.

Table 2: Total LUT usage for a 4 bits Input Signal Resolution

FPGA	Available Look-Up Tables	12 GPS-L1 SGR LUT usage (%)
Virtex II (XC2V3000)	28 672	53,6
Altera Cyclone II (EP2C35)	33 216	46,3
Virtex IV (XC4VFX140)	126336	12,2
Altera Stratix II (EP2S130)	132 540	11,6

Table 3: Maximal number of implementable GPS-L1 Channels

Input Signal Resolution (# bits)	1	2	4
Max. # of Chanels for the Virtex II (XC2V3000)	44	41	36
Max. # of Chanels for the Altera Cyclone II (EP2C35)	55	51	45
Max. # of Chanels for the Virtex IV (XC4VFX140)	267	249	219
Max. # of Chanels for the Altera Stratix II (EP2S130)	281	262	231

Table 3 shows that up to 44 channels might be implemented in the current Virtex II FPGA. It is also interesting to remark that over 250 channels might be developed inside nowadays largest FPGA available on the market. Therefore, it proves that the development of the most complex real-time, multi-band and multi-channel GNSS receiver is made possible with the use of a single FPGA.

7. SGR RECEIVER PERFORMANCES

The developed SGR has proved itself to be quite competitive when compared to the PC-based software receivers currently being developed. Table 4 summarizes some of the currently available PC-based software front-ends characteristics [1] along with those of the developed receiver. The characteristics of another 12 channels FPGA-based SGR form the University of New South Wales (UNSW - Australia) are also shown [12, 13].

As shown in Table 4, FPGA-based SGRs usually enjoy a higher sampling rate (and the possibility of larger signal bandwidths) than their PC-based counterparts [1]. The possibility of implementing very sharp and flexible digital filters in the IF spectrum is also another feature of great

interests that is difficult to realize in real time in most current PCs. The developed SGR will however soon benefit of such a feature with a 2-24 MHz variable IF bandwidth achieved inside the FPGA.

Table 4: Overview of some available SGR Front-Ends

Source	Frequency band	Bandwidth (MHz)	Sample rate (MHz)
Accord	L1/L2	2	2-6
Fraunhofer	L1/L2	4-20	10-40
IfEN	L1	10	23
NordNav	L1	< 8	16
Univ. FAF Munich/ Fraunhofer	L1/L2/L5	18.5	20 or 40
UNSW	L1	4	40
ÉTS-SGR	L1	2-24	60-105

A last aspect not depicted in the table above is the maximal PVT update rate allowed by a SGR. PC-Based receiver usually achieves real-time PVT update rates on the order of 10 Hz [14] (at least 2 GHz computers are recommended in that purpose). However, a tremendously high update rate (virtually up to 1 kHz for 12 GPS channels) is allowed by the developed SGR. This feature is specific to the designed architecture which relies on a relatively tight integration of a FPGA and a PC computer. Simply speaking, ranging measurements (pseudo-ranges, Doppler and satellite vehicle time) are directly transferred from the FPGA to the PC computer at a 1 kHz rate. Yet the PVT solving algorithm is the only computational task left to the PC computer so the position update rate is mainly limited by the incoming data flow. Few tests have been done in order the measure the PVT algorithm computational load and they all lead to the following conclusion: a processor usage smaller than 1% is required to compute the PVT for update rates as high as 20 Hz (the PC is equipped with a 2.2 GHz Pentium 4 M processor).

8. REAL-TIME TESTING

A series of real-time tests have been performed in order to validate the developed 12 GPS-L1 channels. Those tests were first realized in a controlled environment with the use of a Spirent GSS7700 GPS constellation simulator. Multipaths were disabled and the signals C/N_0 were adjusted between 42 and 44 dB-Hz to simulate clear sky conditions. Static and dynamic scenarios have been explored in order to measure the receiver's accuracy as well as its response to acceleration and jerk stresses. Signal power has been decreased afterwards so the receiver's limits of operation due to noise could be measured (tests have been done for both acquisition and tracking phases).

A second series of tests has also been performed outside with the help of the Novatel GPS-704-X Antenna. As the tests have been done in static mode, the impact of multipaths on the receiver, along with unmodeled ionospheric and tropospheric errors could be directly gauged.

The dynamic scenario consisted of a rectilinear trajectory for which the receiver was 1) initially at rest, 2) subjected to an acceleration of 10 m/s^2 to reach a maximal speed of 100 m/s, 3) kept a constant velocity during 30 sec, 4) subjected to an acceleration of 20 m/s^2 in the opposite direction 5) to finally return to rest (see Figure 8). The receiver was subject to jerk stresses of 10 000 m/s^3 at the beginning and end of the first acceleration phase and jerks of up to 40 000 m/s^3 was associated to the last acceleration phase. The receiver heading was 45° of azimuth and 0° of pitch. All tests were realized with the parameters shown in Table 5.

Table 5: SGR Test Setup

IF Bandwidth	24 MHz
Correlator Spacing	1/16 chip
PLL Integration Time	1 ms
PLL Order	3rd
PLL Noise Bandwidth	10 Hz
DLL Integration Time	10 ms (non-coherent)
DLL Order	2nd
DLL Noise Bandwidth	1 Hz
DLL Aiding	None
PVT Computation	Kalman Filter
PVT Update Rate	20 Hz
Ionospheric Corrections	Broadcasted by GPS Satellites
Tropospheric Corrections	EGNOS Tropospheric Model [15]

In all tests, pseudo-range was smoothed prior to PVT computation using a carrier smoothing algorithm. This algorithm was developed on the PC computer side and is based on the following equation [16]:

$$\bar{\rho}(n) = \frac{1}{n}\rho(n) + \frac{n-1}{n}\left[\bar{\rho}(n-1) + \left(\Phi(n) - \Phi(n-1)\right)\right] \quad (1)$$

Where $\rho(n)$ is the code pseudo-range, $\Phi(n)$ is the carrier phase, $\bar{\rho}(n)$ is the smoothed pseudo-range and *n is an integer (*1, 2, 3, …). Because the first term of this equation becomes negligible over time (due to the 1/*n* factor), the following equation is rather used for *n* larger than 10 000:

$$\bar{\rho}(n) = \bar{\rho}(n-1) + \left(\Phi(n) - \Phi(n-1)\right) \quad (2)$$

Since the smoothed pseudo-range tends to drift with time, a correction is applied once every 20 seconds. This correction is applied independently to each channel from the first moment it locks down (so the overall position correction might seem asynchronous - refer to Figure 7 and Figure 9 for examples). The pseudo-range correction is computed in an "integrate and dump" fashion in order to reduce the computational burden:

$$\rho_{correct} = \frac{1}{M}\sum_{m=0}^{M}\left(\rho(n) - \bar{\rho}(n)\right) \quad (3)$$

Where $\rho_{correct}$ is the carrier pseudo-range adjustment and M is the number of integrated samples (M = 20 000 since the measurements received from the FPGA are updated every millisecond).

9. CONTROLLED ENVIRONMENT RESULTS

The receiver precision has been first verified in static mode by connecting the front-end circuitry to the Spirent GPS signals simulator. The channels status window from the developed GUI is presented in Figure 5 and shows the visible constellation and satellites C/N_0. A total of 10 satellites were included into the PVT solution presented in Figure 6 and Figure 7. The results are shown for a consecutive duration of 12 minutes (with a total of 14 400 points on each figures).

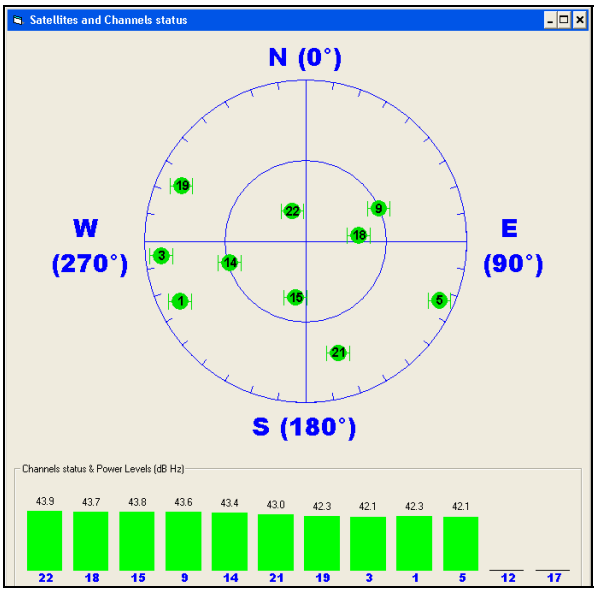

Figure 5: Simulated GPS Constellation and Signals C/N_o

The examination of the following figures reveals that the receiver solution is very stable over a short period of time (< 35 cm peak-to-peak on all 3 axes during 12 minutes). A closer look to the scatter plot in Figure 6 however shows that instead of a uniform distribution, the position is spread in smaller Gaussian agglomerations. This is actually caused by the pseudo-range smoothing algorithm which applies corrections at specific points of time instead of doing so continuously. This discrete behavior can be observed more easily in Figure 7 with corrections seen as a series of random position steps.

The mean positioning error is excellent with centimetric values on all 3 axes (5 mm east, 23 cm north and -38 cm height). This result was however expected as the ionospheric and clock errors introduced by the simulator were perfectly modeled. The residual error is solely due to the receiver tropospheric model which differs from the one used by the simulator (the STANAG model is used by the simulator instead of EGNOS).

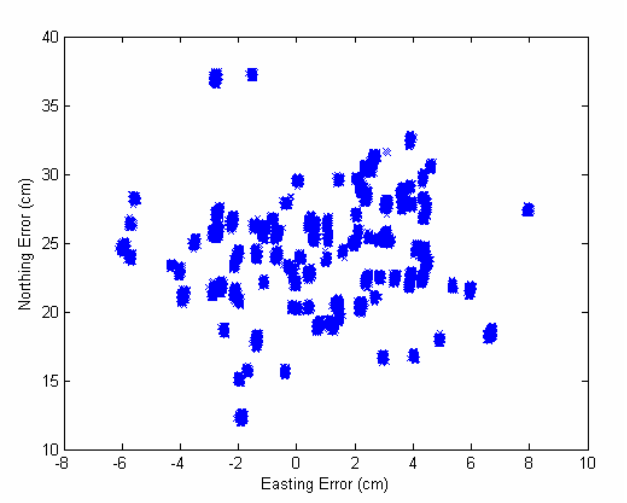

Figure 6: Absolute Horizontal Error at High C/N_o (42-44 dB-Hz)

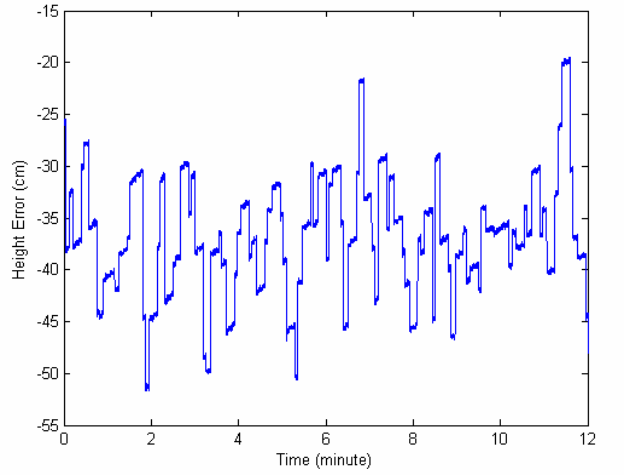

Figure 7: Absolute Vertical Error at High C/N_o (42-44 dBHz)

Tests were also done in dynamic mode so the receiver response to acceleration and jerk stresses could be observed. The acceleration displayed in Figure 8 was monitored by the receiver by taking the computed velocity derivative on all 3 axes. The position accuracy and stability happened to be quite similar to those seen in static mode for any velocity or acceleration level, as long as the jerk did not exceed 17 000 m/s^3. Larger jerks made however many channels' PLL to unlock (characterized by high acceleration spikes between seconds 48 and 55 in Figure 8 and by an acceleration error of about 5 m/s^2). As a consequence, position drifts up to several tens of meters were measured during the last acceleration period.

At last, the satellites signal strength has been manually reduced so the receiver robustness to noise could be measured. The positioning error is shown in Figure 9 and Figure 10 for signals C/N_0 ranging from 30 to 32 dB-Hz. A minimal C/N_0 of 30 dB-Hz has been measured for both acquisition and tracking phases to operate correctly.

Considering the fact that a large bandwidth has been used (24 MHz), a much lower C/N_0 (on the order of 20 dB-Hz) is however anticipated by reducing the GPS-L1 channels bandwidth down to 2 MHz. This is easily shown using the equation below:

$$C/N_0 = 10\log_{10}\left(P_c S_n B_n\right) \qquad (4)$$

Where P_c is the signal power, S_n is the noise spectral density and B_n is the noise bandwidth. Because the GPS L1 C/A signal has 95% of its total power concentrated in a 2 MHz frequency band, an improvement of at least 10 dB can be achieved by reducing the total bandwidth from 24 MHz down to 2 MHz. The minimal C/N_0 required by the receiver to operate is therefore of about 20 dB-Hz for a 2 MHz bandwidth.

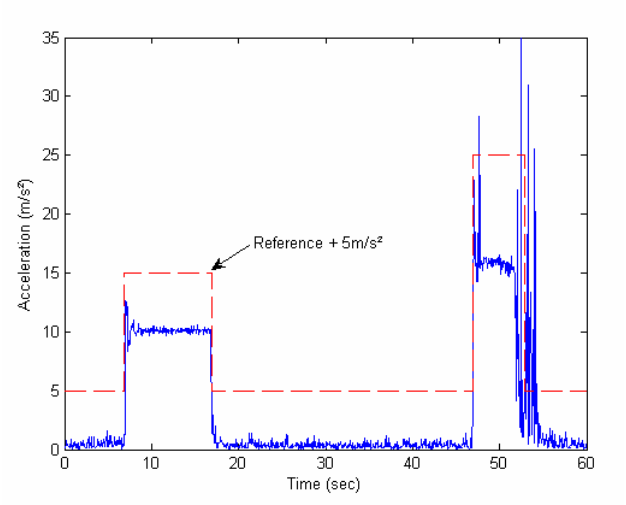

Figure 8: Receiver Measured Acceleration

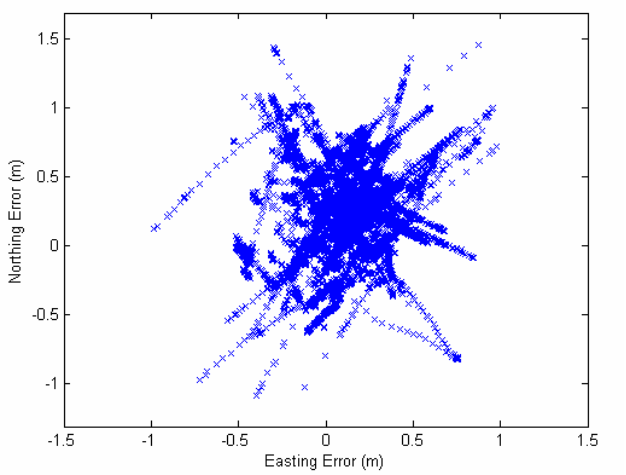

Figure 9: Absolute Horizontal Error at Low C/N_0 (30-32 dB-Hz)

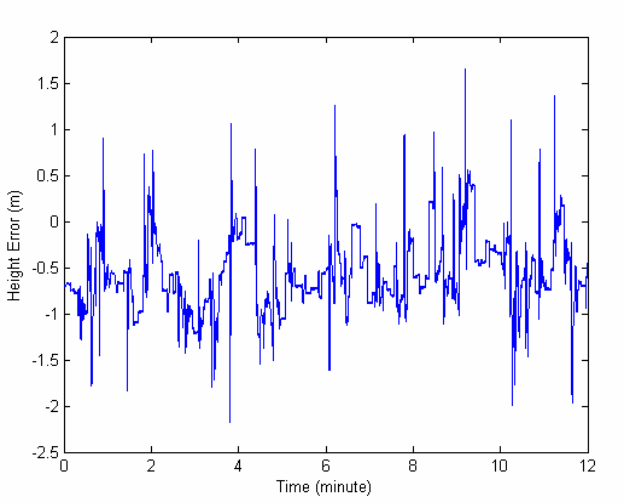

Figure 10: Absolute Vertical Error at Low C/N_0 (30-32 dB-Hz)

An inspection of both figures above shows a maximal peak-to-peak error of 3.5m on all 3 axes. Because the signals strength was very close to the receiver's limit of operation, spurious PLL unlock/relock events were monitored. The one meter spikes seen in Figure 10 (and the ray-like patterns seen in Figure 9) are the result of such spurious events (or more precisely of the pseudo-range algorithm convergence phase related to lost satellites reacquisition).

10. REAL ENVIRONMENT RESULTS

Tests in static mode have also been realized outside with real GPS signals. The antenna phase center was exactly located at WGS-84 N45° 29' 40.23" W73° 33' 44.5", which is on the roof of Ecole de Technologie Superieure, downtown Montreal (Canada). The height above sea level was roughly measured to 36 m from a Garmin GPS receiver using the Wide Area Augmentation System (WAAS) capability. Two 30 minutes tests were performed: one with the carrier smoothing algorithm activated and one without this feature. The absolute positioning errors are shown in Figure 12 to Figure 15 with a total of 36 000 points on each figure. The visible constellation and signals C/N_0 at the tests beginning are shown in Figure 11. A close examination of this figure reveals that signals C/N_0 are quite strong, ranging from about 37 to 47 dB-Hz. Notice that the satellite PRN #15 was not used in the PVT solution since its health was bad during the test interval (held on December 8th 2006). Up to 9 satellites were thus included into the PVT solution.

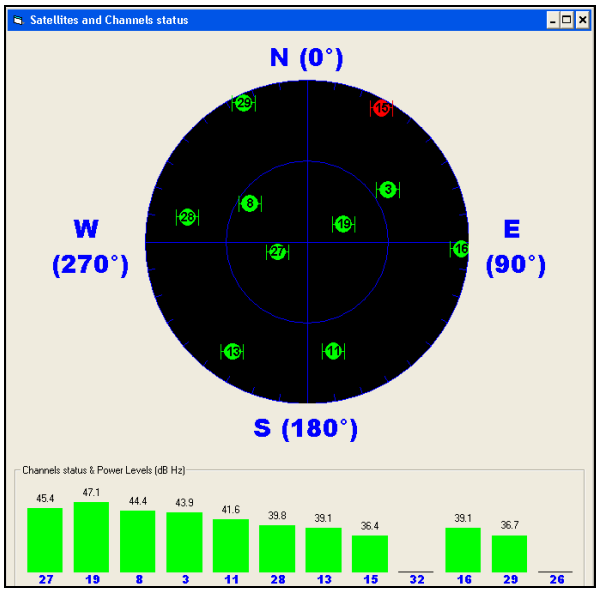

Figure 11: Real GPS Constellation and Signals C/N_0

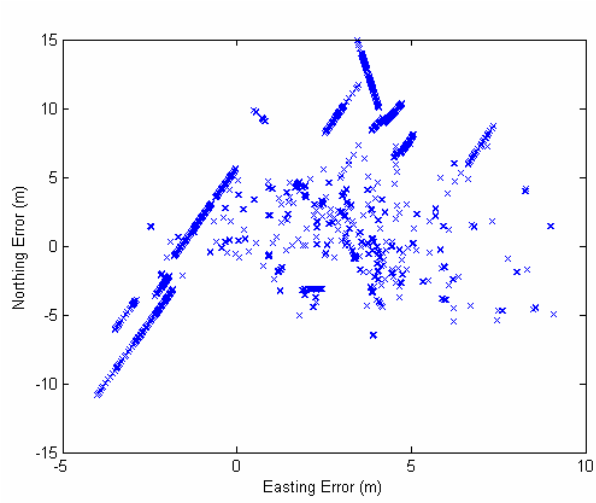

Figure 12: Absolute Horizontal Error (with Carrier Smoothing)

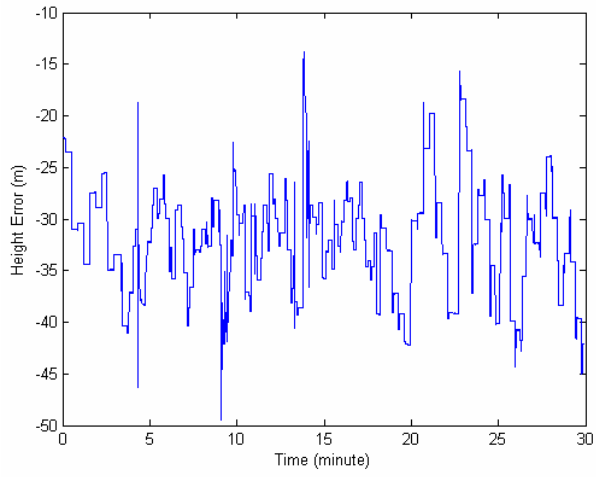

Figure 13: Absolute Vertical Error (with Carrier Smoothing)

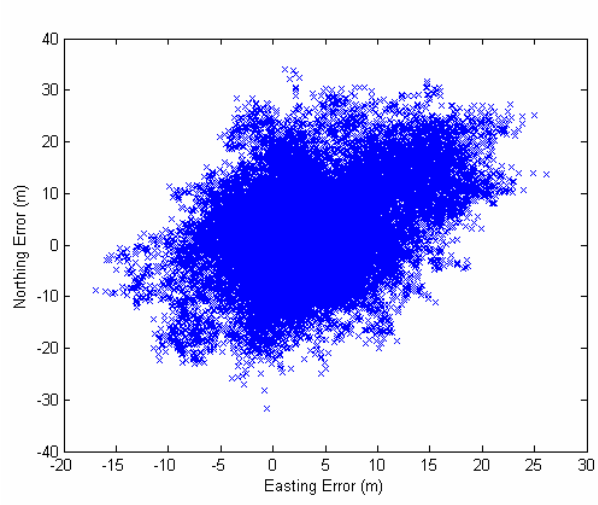

Figure 14: Absolute Horizontal Error (No Smoothing)

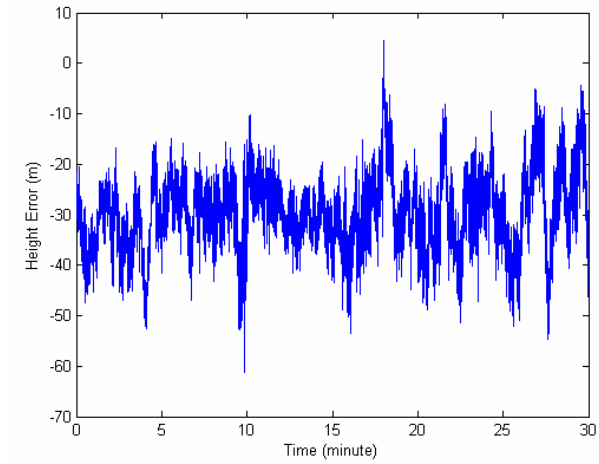

Figure 15: Absolute Vertical Error (No Smoothing)

Figure 12 and Figure 13 shows that the position computed in real-time from the real GPS signals is less stable than previously seen during simulation (15, 25 and 35 m peak-to-peak easting, northing and for height respectively). Because the signals were strong (36.4 dB-Hz being the smallest C/N_0), the thermal noise had little influence on the ranging measurements stability. Multipaths played however an important role in disturbing the PVT solution. The carrier smoothing algorithm was not able to correctly filter those multipaths due to their slow variations (easily monitored in Figure 15 as position variations taking place over several minutes). The smoothing algorithm did however reduce the maximal positioning error measured as the non-smoothed solution error presented in Figure 14 has a maximal value of 60 m peak-to-peak northing (thus about twice larger than the smoothed solution). At last, an absolute accuracy on the order of 2 to 4 meters horizontally and of about 30 meters vertically as been measured in both tests. Considering the fact that standalone GPS positioning accuracy is usually on the order of 10 m horizontally and greater than 20 m

vertically (depending on the satellites geometry), both tests therefore proved the ability of the developed SGR to operate correctly in real-time with real GPS signals.

CONCLUSION

In summary, the FPGA-based GNSS software defined receiver presented in this paper is of great interest as it allows good precision and high throughput without sacrificing signal bandwidth. Because of the FPGA flexibility, powerful designs and short turnaround times can be achieved, features which are very attractive to today rapid R&D. The possibility of integrating numerous GNSS channels inside a single component has also been discussed as a major advantage of using FPGAs in SGR designs. These aspects are therefore thought to bring significant contributions to future low-cost GNSS receivers design and to lead to ever more robust and precise real-time navigation solutions.

In perspective, tests shall soon be performed outdoor in dynamic scenarios in order to characterize furthermore the developed receiver. Twelve (12) Galileo-E1 channels should also be tightly integrated with the current GPS channels as interoperability is the key project driver. As part of the project, the migration toward a larger and faster Virtex IV FPGA is also scheduled so the implementation of GPS-L5 and Galileo-E5a/E5b channels could be achieved in a near future. Finally, the developed navigation solution is not yet optimized and some improvements could be added to the navigation solution, such as preselecting the GNSS satellites offering the best geometry and the lowest perturbations.

ACKNOWLEDGMENTS

The authors wish to thank Dr. Yacine Adane for its support during the development of the RF front-end. Also a special thank to Iurie Ilie and Dr. Aurelian Constantinescu who gave precious advices all along the project development. This work has been made possible by the National Sciences and Engineering Research Council of Canada (NSERC) and the Fonds Québécois de la Recherche sur la Nature et les Technologies (FQRNT).

REFERENCES

[1] J.-H. Won, T. Pany and G. Hein, "GNSS Software Defined Radio: Real Receiver of Just a Tool for Experts?," in *Inside GNSS*, vol. 1, 2004, pp. 48-56.
[2] Xilinx website, http://www.xilinx.com.
[3] SDR Forum website, http://www.sdrforum.org.
[4] Wikipedia online encyclopedia, http://www.wikipedia.com.
[5] Lyrtech website, http://www.lyrtech.com.
[6] F. Piazza and H. Qiuting, "A 1.57-GHz RF front-end for triple conversion GPS receiver," *Solid-State Circuits, IEEE Journal of*, vol. 33, pp. 202-209, 1998.
[7] D. K. Shaeffer, A. R. Shahani, S. S. Mohan, H. Samavati, H. R. Rategh, M. del Mar Hershenson, X. Min, C. P. Yue, D. J. Eddleman, and T. H. Lee, "A 115-mW, 0.5µm CMOS GPS receiver with wide dynamic-range active filters," *Solid-State Circuits, IEEE Journal of*, vol. 33, pp. 2219-2231, 1998.
[8] E. D. Kaplan, *Understanding GPS principles and applications*. Boston, Mass.: Artech House, 1996.
[9] "Galileo Open Service Signal In Space Interface Control Document Draft 0," European Space Agency / Galileo Joint Undertaking, May 2006, pp. 19-20.
[10] M. Cloutier, T. Varelas, C. Cojocaru, and F. Balteanu, "4-dB NF GPS receiver front-end with AGC and 2-b A/D," *Proceedings of the Custom Integrated Circuits Conference*, pp. 205-208, 1999.
[11] Altera website, http://www.altera.com.
[12] F. Engel, and G. Heiser, "An Open GNSS Receiver Platform Architecture," presented at International Symposium on GNSS/GPS, Sydney, 2004.
[13] F. Engel, P. Mumford and K. Parkinson. , "Namuru FPGA GPS tracking module datasheet," University of New South Whales, datasheet Issue 1.0, May 2006 May 2005.
[14] NordNav website, http://www.nordnav.com.
[15] N. Penna, Alan Dodson and Wu Chen, "Assessment of EGNOS Tropospheric Correction Model," *Journal of Navigation*, vol. 54, pp. 37-55, January 2001.
[16] P. Misra and P. Enge, *Global positioning system : signals, measurements, and performance*. Lincoln, Mass.: Ganga-Jamuna Press, 2004.