Design and Analysis of Reconfigurable Embedded GNSS Receivers Using Model-Based Design Tools

Shankararaman Ramakrishnan, Grace Xingxin Gao, David De Lorenzo, Todd Walter and Per Enge
Stanford University
Dennis Akos
University of Colorado - Boulder

BIOGRAPHY

Shankararaman Ramakrishnan received the B.E. (Hons.) degree in Electrical and Electronics Engineering from the Birla Institute of Technology and Science – Pilani, India in 2006. He is currently a Ph.D. student in the Department of Aeronautics and Astronautics at Stanford University where he is a member of the Global Positioning System Group. His research interests include signal integrity analysis and dual frequency software receiver design.

Grace Xingxin Gao is a Research Associate at Stanford University. She received her B.S. in Mechanical Engineering in 2001 and her M.S. in Electrical Engineering in 2003 both from Tsinghua University, Beijing, China. She received her Ph.D. in Electrical Engineering from Stanford University in 2008. Her current research interests include Galileo signal and code structures, GNSS receiver architectures, and GPS modernization.

David De Lorenzo is a member of the Stanford University GPS Laboratory, where he is a Research Associate. He received his Ph.D. in Aeronautics and Astronautics from Stanford University in 2007. His research interests include studying space-time adaptive antenna array processing and GNSS software receivers. David has previously worked for Lockheed Martin and for the Intel Corporation.

Dennis Akos is currently an Assistant Professor of Aerospace Engineering at the University of Colorado – Boulder. He completed the Ph.D. degree in Electrical Engineering at Ohio University conducting his graduate research within the Avionics Engineering Center. His research interests include GPS/CDMA receiver architectures, RF design, and software radios.

Todd Walter is a Senior Research Engineer in the Department of Aeronautics and Astronautics at Stanford University. Dr. Walter received his Ph.D. from Stanford University in 1993. His current research interests are developing WAAS integrity algorithms and analyzing the availability of the WAAS signal. He is a Fellow of the ION.

Per Enge is a Professor of Aeronautics and Astronautics at Stanford University, where he is the Kleiner-Perkins, Mayfield, Sequoia Capital Professor in the School of Engineering. He is also the Director of the GPS Research Laboratory, which works with the Federal Aviation Administration, U.S. Navy and U.S. Air Force to pioneer systems that augment the Global Positioning System (GPS). Prof. Enge has received the Kepler, Thurlow and Burka Awards from the Institute of Navigation for his work. He is a Member of the National Academy of Engineers (NAE), a Fellow of the ION, and a Fellow of the IEEE.

ABSTRACT

Over the next decade, civilian users will have access to multiple GNSS signal frequencies and constellations. This drastic increase in signals and their frequencies creates substantial opportunities and requirements for analysis and validation. Such analysis and validation is of significant importance from an aviation integrity perspective. The ultimate goal of our research efforts is to develop a standalone reconfigurable platform capable of tracking and monitoring multiple constellations and frequencies as and when they commence transmission. This platform will also be utilized to test and verify new receiver processing algorithms currently under development. It must be capable of running in real-time to ensure signals can be monitored on a 24x7 basis. As a first step in this direction, we designed and validated a standalone L1 C/A receiver which runs in real-time on a Xilinx University Program Virtex-II Pro Development Board. This board features a Virtex-II Pro FPGA with two

on-chip PowerPC 405 32-bit RISC hardcore processors. This feature of the FPGA fabric facilitates the development of reconfigurable embedded GNSS receivers which do not require the resources of a Host PC or a dedicated DSP processor. The entire system was designed using the Xilinx System Generator for DSP, a modeling and implementation tool for high-performance DSP systems. Such a model-based design approach facilitates rapid system development and prototyping thereby enabling system modifications and upgrades to be implemented in a short time span.

INTRODUCTION

Present day GNSS civilian users have unrestricted access to only the GPS L1 C/A and Glonass Standard Precision (SP) ranging signals. Over the course of the next decade, users will be able to access multiple frequencies and constellations. In addition to new signals on GPS, GNSS constellations such as the European Union's Galileo and China's Compass along with a modernized Russian Glonass system will provide multiple ranging sources to GNSS users. Some of the proposed frequencies and constellations are shown in Figure 1.

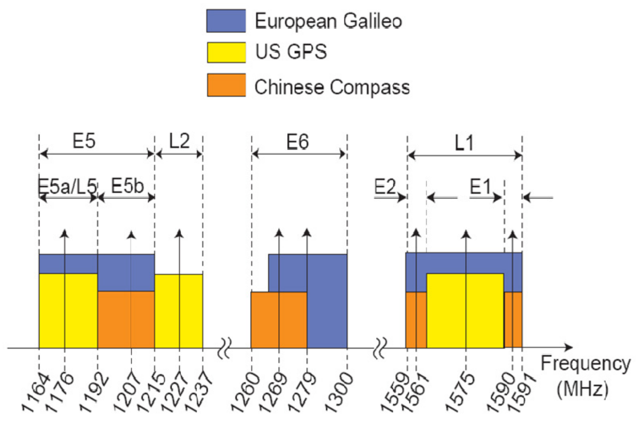

Figure 1: Future Global Navigation Satellite System Signals

Most of these systems will be interoperable since they will be modulated by a common set of carrier frequencies. The possibility of obtaining dual-frequency measurements will help improve accuracy for civilian users while enhancing system integrity, availability, and continuity for aviation users. A few of the proposed new GNSS frequencies and constellations currently transmit test signals which may not necessarily be their final signal specifications [1, 2, 3].

SRAM-based Field Programmable Gate Arrays (FPGA) based receivers can be easily reprogrammed making them an ideal choice to acquire, track, and validate new signals whose specifications may not have been finalized. Ease of reprogramming the device makes it an ideal choice for rapid prototyping. FPGAs help overcome the limitations of Application Specific Integrated Circuits (ASICs) and pure software defined radios. While ASICs are optimized for computational efficiency, they cannot be reconfigured to incorporate changes in the Signal in Space (SIS) specifications of the signals for which they are specifically designed for. Software defined radios provide flexibility in incorporating system changes but are computationally expensive.

A FPGA device is an integrated circuit with a central array of logic blocks that can be connected through a user configurable interconnect routing matrix. The periphery of logic-array is comprised of a ring of I/O blocks that can be configured to support different interface standards. This flexible architecture can be exploited to implement a wide range of synchronous and combinational digital logic functions. A simplified representation of a FPGA block diagram is shown in Figure 2.

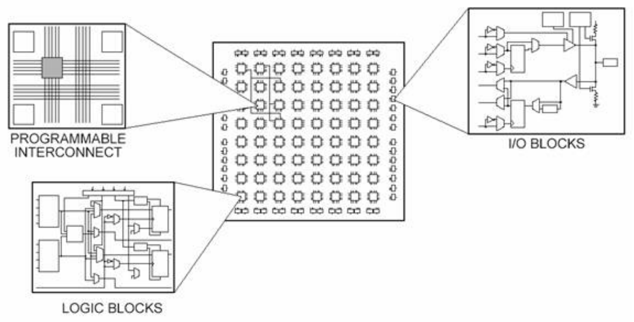

Figure 2: Simplified FPGA Block Diagram (Courtesy Xilinx, Inc)

A digital design can utilize one or more of three basic types of devices: Logic, Memory and Processors. Several of the current high-end FPGA families incorporate all three of these devices within a single integrated circuit (IC). The ability to implement hardcore or softcore processors within the FPGA makes them well suited for developing reconfigurable embedded systems. Further details about hardcore and softcore embedded processors are provided in a subsequent section of this paper. These FPGA families feature millions of equivalent gates of functionality and high-speed interfaces capable of supporting a broad range of engineering solutions including nontraditional applications. Today, FPGAs are capable of implementing complex functionality such as the correlation process in a GNSS receiver which traditionally is performed on a dedicated ASIC.

Apart from the basic three digital components mentioned in the previous paragraph, many newer generation FPGA families also feature dedicated components specifically designed to perform DSP functions. These components accelerate algorithms and enable higher levels of DSP integration and lower power consumption within the device. These dedicated DSP components support over 40 dynamically controlled operating modes including:

multiplier, multiplier-accumulator, multiplier-adder/subtractor, three input adder, barrel shifter, wide bus multiplexers, wide counters, and comparators. They also incorporate efficient adder-chain architectures for implementing high-performance filters and complex mathematical operations efficiently. Some FPGA manufacturers also provide users with Intellectual Property (IP) blocks which help implementation of popular DSP algorithms and functions in an optimal manner. Of particular importance for software GNSS receiver development include Math Functions such as the COordinated Rotation DIgital Computer (CORDIC) algorithm used to compute trigonometric functions and specialized DSP algorithms/designs such as the Fast Fourier Transform (FFT) and Finite Impulse Response (FIR) filter design. [4, 5]

In this paper, we present some preliminary results of our current efforts in developing a reconfigurable embedded GNSS receiver platform capable of tracking multiple frequencies and constellations. This platform will be used for a variety of applications which are discussed in a subsequent section of this paper. As a first step, we designed a real-time reconfigurable embedded GPS L1 receiver and analyzed its performance. We now describe the details of the receiver design and implementation process and compare its performance to that of a pure software defined receiver executed on a Host PC.

MOTIVATION & BACKGROUND

GNSS software radio receivers have evolved notably over the past few years. It was originally developed in 1997 as a tool for post processing of collected GPS data. Its implementation in Matlab made it computationally expensive [8]. In order to reduce computational expense, pure software receivers were implemented in a high level language such as C/C++ running on a programmable microprocessor. Over the years, through the use of novel processing techniques combined with technology improvements in microprocessor capabilities have enabled implementation of multi-channel real-time GPS L1 C/A software receiver [9, 10]. Such receivers utilize the processing resources of the microprocessor of a host PC and their performance is directly related to the resources available on the host PC.

To overcome the computational limitations placed by resource availability on a host PC, various implementations have been proposed which utilize the signal processing capabilities of a dedicated Digital Signal Processing Chip (DSP) and/or the reconfigurable parallel processing capabilities of a FPGA [11, 12, 13] to develop real-time GNSS software receivers. These implementations were designed using hardware descriptive languages (HDL) such as Verilog or Very-High-Speed Integrated Circuits HDL (VHDL) and were executed on hardware platforms custom designed by the authors.

Our desire to use a ubiquitous commercial off the shelf FPGA development board influenced the choice of the hardware platform selected for our current work. After comparing the cost and processing capabilities of several development boards, we decided to adopt the Xilinx University Program Virtex-II Pro Development System. In fact, this board is used by over 2000 universities the world over for digital design courses. We are of the opinion that this board can also be utilized as an attractive educational tool for GNSS software receiver design courses. The development board is distributed by Xilinx Inc., through its Xilinx University Program (XUP) to universities affiliated with the program. It features a Virtex II-Pro FPGA chip along with onboard external memory modules, I/O ports and other peripherals. By using this particular development board, we could avoid the time and costs associated with designing a custom hardware board. A picture of the board along with the external peripherals it supports is shown in Figure 3.

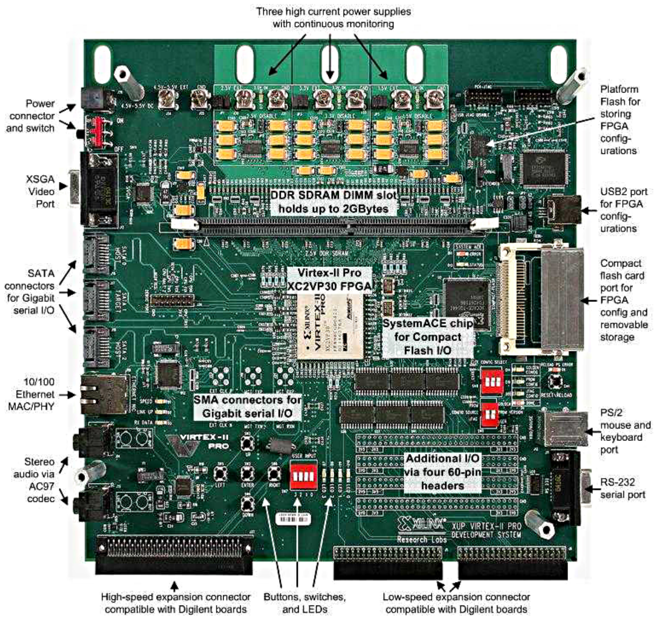
Three high current power supplies with continuous monitoring
Power connector and switch
XSGA Video Port
SATA connectors for Gigabit serial I/O
10/100 Ethernet MAC/PHY
Stereo audio via AC97 codec
Platform Flash for storing FPGA config-urations
USB2 port for FPGA config-urations
Compact flash card port for FPGA config and removable storage
PS/2 mouse and keyboard port
RS-232 serial port
DDR SDRAM DIMM slot holds up to 2GBytes
Virtex-II Pro XC2VP30 FPGA
SystemACE chip for Compact Flash I/O
SMA connectors for Gigabit serial I/O
Additional I/O via four 60-pin headers
High-speed expansion connector compatible with Digilent boards
Buttons, switches, and LEDs
Low-speed expansion connector compatible with Digilent boards

Figure 3: Xilinx University Program Development System

The Virtex series of FPGAs feature built-in hardcore 32-bit IBM processors known as PowerPCs which can be utilized as microprocessors within the same FPGA chip. We intended to exploit this feature while designing our standalone embedded GNSS software receivers. Thereby we could also avoid the necessity to use either a host PC or a dedicated DSP chip to perform the complex baseband receiver signal processing. Further, rather than designing the system using Hardware Descriptive Languages, we decided to implement the design using Model-based design tools available from Xilinx Inc. The System

Generator for DSP is an optimized modeling and implementation tool which can be used to design high-performance DSP systems. We believe such an approach is the most suitable since it provides us with the flexibility to utilize a single FPGA platform to design and implement multiple applications within a short time span. Such a design methodology does not require tedious manual coding and hence design teams need not worry about coding and commenting styles which vary amongst programmers. Such a design approach makes it easy for multiple individuals to work on small modules which can be integrated together to obtain the required larger design.

RECONFIGURABLE EMBEDDED GNSS SOFTWARE RECEIVER DESIGN

FPGA manufacturers use two different implementations to include an embedded CPU core within a FPGA chip. The first known as a "Softcore" is a processor written by the user as a parameterisable function along with code for the FPGA's logic. Such implementations include the MicroBlaze and the Nios-II from Xilinx and Altera respectively. Certain FPGAs from Xilinx and a few other manufactures also include dedicated CPU known as a "Hardcore" processor. A hardcore processor is implemented directly in IC transistors achieving maximal performances, while a softcore processor is an IP core which is implemented on the FPGA's logic cells. As a result, dedicated hardware processors do not use any of the FPGA's programmable resources. In contrast, using a dedicated processor external to the FPGA fabric requires hundreds of additional interface pins, which degrades system performance and significantly increases FPGA I/O requirements and overall board costs. Though typically the cost of including a softcore processor is lower than a hardcore processor, it exhibits significantly lower performance measured in terms of clock speed and MIPS (million instructions per second) count [14].

The Xilinx Virtex series of devices have up to 4 built-in IBM PowerPC microprocessors within the FPGA chip. These processors provide 32-bit fixed-point embedded applications with high performance at low power consumption. Also, PowerPC processors can provide floating-point support either in hardware or software. While the legacy Virtex-II Pro and Virtex-4 series of FPGAs featured the PowerPC 405 CPU which could clock a maximum speed of 450 MHz and execute over 700 DMIPS, the latest Virtex-5 FXT series of FPGAs feature a PowerPC 440 CPU which can clock a maximum speed of 550 MHz and execute over 2000 DMIPS.

The XUP development board previously described features a Virtex-II Pro FPGA (XC2VP30) with two dedicated PowerPC 405 processor blocks. This provides us with the means to design GNSS receivers based on the time-tested architecture of commercial GNSS receivers. In commercial receivers, ASICs are used to carry out massively parallel correlation operations while microprocessors such as an ARM processor are utilized for baseband signal processing. In our reconfigurable embedded GNSS receiver implementation, we leverage the parallel processing capabilities of the FPGA logic cells to perform simple but high frequency receiver processing functions such as the correlation processing during acquisition and code, carrier wipeoff in the tracking loops. The hardcore PowerPC is used to perform complex but low frequency functions such as the baseband signal processing in the tracking loops and navigation solution computation. The proposed use of the FPGA's logic cells and an embedded hardcore processor results in an optimized hardware/software partitioning that maximizes FPGA utilization while minimizing hardware costs. This implementation strategy does not require the processing resources of a Host PC or additional DSP chips. Thereby, our proposed software receiver implementation strategy results in a system design whose performance is not limited by the capabilities of a host PC nor does it involve the additional costs of a DSP chip.

SYSTEM DESIGN USING MODEL-BASED DESIGN TOOLS

Traditionally, pure software receivers running on a PC are coded in higher level language such as Matlab or C/C++. Design of customizable logic hardware requires coding the necessary logic using dedicated hardware description languages such as Verilog or VHDL. A GNSS software receiver utilizing a DSP processor would require the designer to program the processor using proprietary programming tools provided by the DSP manufacturer.

A typical software receiver implemented in a FPGA would require the user to be familiar with vendor provided simulation and synthesis tools such as the Integrated Simulation Environment (ISE) toolkit from Xilinx Inc. or third party tools such as ModelSim for simulation and Simplicity for design synthesis. In order to popularize greater use of their FPGAs manufacturers have started to offer Model-based design toolkits. Such toolkits do not require the designer to be familiar with Hardware Description Languages thereby resulting in faster system development time. Predefined IP blocks are made available to users for a variety of applications. One such toolkit of particular interest to us has been the Xilinx System Generator for DSP.

This toolkit runs within Simulink and is accessible as an additional blockset along with the Simulink blocksets provided by Mathworks. System Generator communicates with both synthesis tools such as the Xilinx ISE and also embedded system design tools such as the Xilinx Embedded Development Kit (EDK) in the background while abstracting the designer from such processes. It also facilities hardware in the loop co-simulation of the design thus ensuring the design works flawlessly in hardware. Each individual module of a larger design can be separately verified in hardware and then combined to obtain the required design. Figure 4 below shows the functionality of System Generator as a Model-based design toolkit.

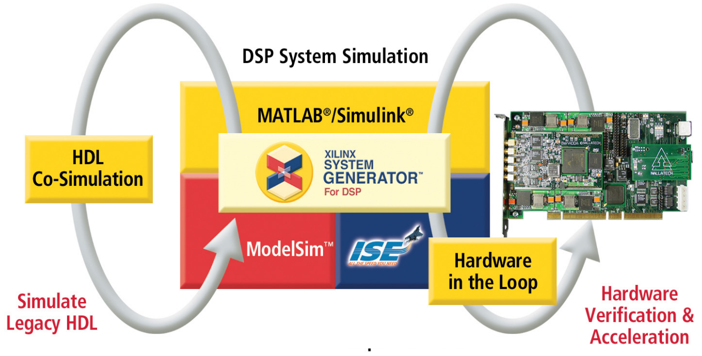

Figure 4: System Generator and its Functionality (Courtesy Xilinx Inc.)

Figure 5 below shows a block diagram of the parallel code phase search algorithm used for acquisition of signals in a software receiver. In order to obtain a functional software receiver such block diagrams will have to be coded and debugged using the appropriate design tools. Model-based design tools such as the Xilinx System Generator for DSP facilitate implementation of the block diagrams in an easy and intuitive manner. Once the necessary functional blocks required to perform the desired logic are included in the Simulink design window, the design can be implemented at the "push of a button". Figure 6 below shows the equivalent representation of the block diagram using predefined IP blocks. Such an implementation eliminates the tedious task debugging coding errors that appear at runtime. A visual description of the design also helps facilitate easy modifications and upgrades at a later stage.

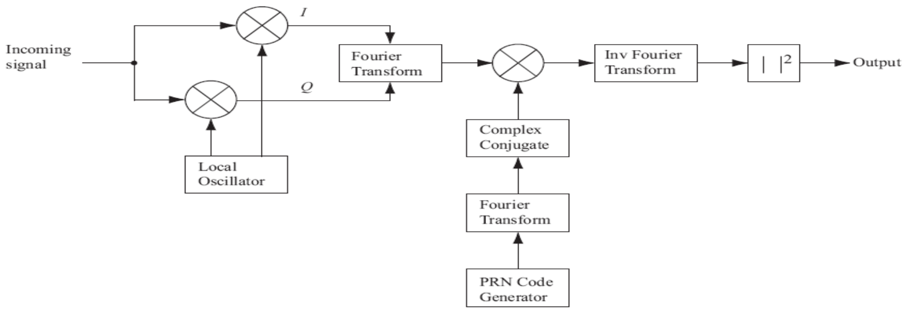

Figure 5: Block Diagram of Parallel Code Phase Search Acquisition

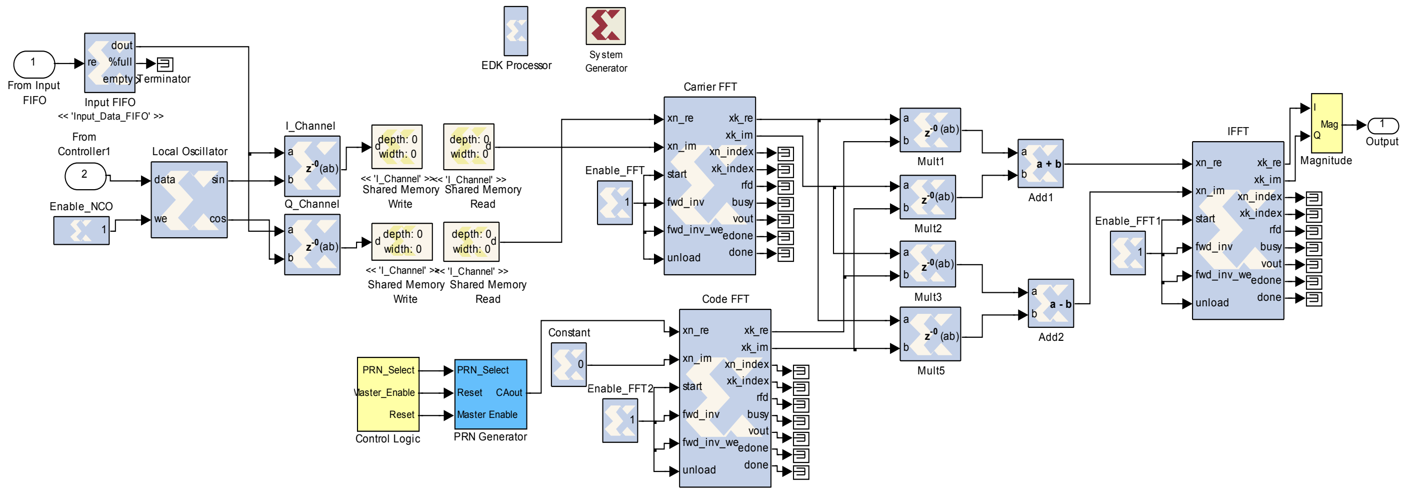

Figure 6: Implementation of Parallel Code Phase Search Acquisition using Model-Based Design Tools

RECEIVER IMPLEMENTATION

To validate the feasibility of designing reconfigurable embedded GNSS receivers, we designed and implemented a multi-channel GPS L1 C/A receiver. Since the larger objective of developing this reconfigurable platform is to monitor new signals and to implement new algorithms, we only implemented the acquisition and tracking modules of the receiver. Herein we present the implementation techniques for the acquisition and tracking loops of the receiver followed by its performance and amount of resources utilization on the FPGA.

I. ACQUISITION

Acquisition is the process by which a receiver determines coarse values of carrier Doppler frequency/residual IF and code phase of the signals transmitted by satellites visible. Acquisition is performed when a receiver is first turned on. Signals need to be reacquired when a receiver loses lock of signals it was tracking. These coarse estimates must be sufficiently accurate for convergence of the subsequent tracking loops. Signals transmitted by GNSS satellites are differentiated based on the PRN sequence assigned to each satellite. Code phase, is the time alignment of the PRN code in the current block of data. It is necessary to know the code phase in order to generate a local PRN code that is perfectly aligned with the incoming code. Only then can the incoming code can be removed from the signal. Carrier frequency is the center frequency of the RF signal used to modulate the PRN sequence and navigation data transmitted by GNSS satellites. In case of down-conversion at the receiver front end, the carrier frequency corresponds to the local IF. This value is obtained based on the carrier frequency and from the mixers used in the down-converter. However, this frequency can deviate from the expected value. The line-of-sight velocity of the satellite causes a Doppler effect resulting in a higher or lower frequency. In the worst case, this frequency deviation can be as large as ±10 kHz.

If the receiver performs a cold start (i.e. no a priori information is available), a serial search for the correlation power of all possible combinations of code phase and Doppler frequency bins must be performed. This results in an extremely large search space based on the resolution requirements for the code phase and Doppler frequency bins. However it is easy to realize that correlation in time-domain corresponds to a convolution process and that convolution in the time domain corresponds to multiplication in the frequency domain. Hence an acquisition algorithm based on Fourier Transforms of the incoming signals and locally generated replica greatly reduces the acquisition search space either in the Doppler frequency or code phase dimension.

We decided to adopt the *parallel code phase search* algorithm in our receiver implementation. Since the code phase dimension is significantly larger than the Doppler frequency dimension, parallelism in the code phase dimension can greatly reduce the size of the acquisition search space. The block diagram for this algorithm was shown in figure 5.

The down-converter used in our receiver down-converted the carrier frequency to an IF frequency of 4.1304 MHz with an A/D sampling frequency of 16.3676 MHz. The FFT IP block used in the implementation shown in figure 6 was implemented using a Radix-4 FFT configured as a Burst I/O operation. The incoming sampled data was padded with additional zeros to obtain the necessary data size of 4^7 required to perform FFT on the data. A 1 ms integration time was used in the implementation with a code phase size of 1 chip. This corresponds to a Doppler frequency bin of 500 Hz spread over a ± 5 KHz frequency deviation range. It is important to note that the entire FFT operation was performed in hardware unlike a pure software receiver wherein the FFT algorithm is implemented in software.

II. TRACKING

The principle function of the tracking module in a GNSS receiver is to refine the code phase and carrier frequency/IF residual frequency obtained through acquisition, to keep track and to demodulate the navigation data of the satellite being tracked.

Each tracking loop available in a receiver is referred to as a receiver channel and is capable of tracking a particular satellite at any given time. Signal processing within a tracking loop can be divided into Signal Demodulation also referred to as Code and Carrier Wipe off followed by baseband signal processing to dynamically update the code and carrier tracking loops [6, 7].

Code Tracking Loop:

The purpose of a code tracking loop is to keep track of the code phase of a specific code in the signal. The output of such a code tracking loop is a perfectly aligned replica of the incoming code. The code tracking loop used in GPS receivers is a delay lock loop (DLL) called an Early-Late (E-L read Early minus Late) tracking loop. The DLL discriminator provides the necessary feedback required to ensure the replica signal is always aligned with the incoming signal. In our implementation, we used a normalized coherent dot product discriminator with a 1-chip E-L correlator spacing. This discriminator requires that the carrier loop remains in phase lock. However such a discriminator requires low computational resources. The normalized coherent dot product discriminator is computed as:

$$\frac{1}{4}\frac{(I_E - I_L)}{I_P} \qquad (1)$$

where: I_E, I_L and I_P are the Early, Prompt and Late versions of the in-phase sampled data

Carrier Tracking Loop:

Successful demodulation of the navigation data requires an exact replica of the carrier wave to be generated at the receiver. The incoming carrier wave is tracking using either Phase Lock Loops (PLL) or Frequency Lock Loops or a combination of the two. While the use of PLLs is referred to as coherent tracking, FLL based tracking is also referred to as Non-coherent tracking. PLL or FLL discriminators blocks are used to find the phase or frequency error between the incoming signal and the local carrier wave replica. This output phase or frequency error is then filtered and used as a feedback to the Numerically Controlled Oscillator (NCO) which adjusts the frequency of the local carrier wave generated. This feedback process in the carrier tracking loop ensure that the local carrier wave could be an almost precise replica of the input signal carrier wave.

Since a pure PLL is sensitive to bit transitions in the navigation data, a Costas PLL is preferred. Costas PLL are inherently insensitive to the presence of data modulation in the incoming signal. Our implementation uses just the Two-Quadrant arctangent Costas loop discriminator. This discriminator is optimal at both high and low SNR and provides the actual phase error and not a function of the phase error. The discriminator algorithm is given by:

$$\mathrm{ATAN}(Q_P/I_P) \qquad (2)$$

where I_P and Q_P are the prompt versions of the in-phase and quadraphase sampled data.

The tracking loop integration time used in our receiver is dependent upon its mode of operation. The receiver implemented had three distinct tracking modes determined based on how long the receiver had been tracking the signal post acquisition. The three modes and the corresponding integration times used were:

1. Pull-In Mode: 1ms Integration Time
2. Transition Mode: 5ms Integration Time
3. Fine Tracking Mode: 20 ms Integration Time

RESULTS

I. RECEIVER PERFORMANCE

The GPS L1 receiver described in the previous section was streamed with sampled IF data. The data was processed using the reconfigurable hardware setup previously described. To validate the results obtained, the same data was processed using a pure software receiver executed in Matlab. Results for both the acquisition search and tracking operations were compared to ensure integrity of the data was maintained while being processing on the hardware platform.

A total of seven satellites were acquired using the FFT based acquisition algorithm. Figure 7 shows the 2-D correlation plot for PRN 14. This satellite had the lowest CPPR value of 2.87 amongst all the satellites acquired. A CPPR threshold of 2.5 was used in the logic circuitry to determine if a PRN was indeed acquired. The complete 3-D output for this PRN is shown in figure 8. The results of the acquisition and tracking modules were written to the Matlab workspace to generate the necessary plots. The code phase and carrier frequency computed using the Hardware platform was in close agreement to those obtained using a pure software receiver.

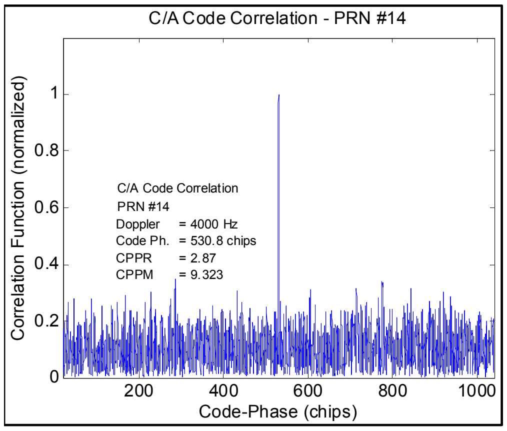

Figure 7: Two-Dimensional Correlation Plot for PRN 14

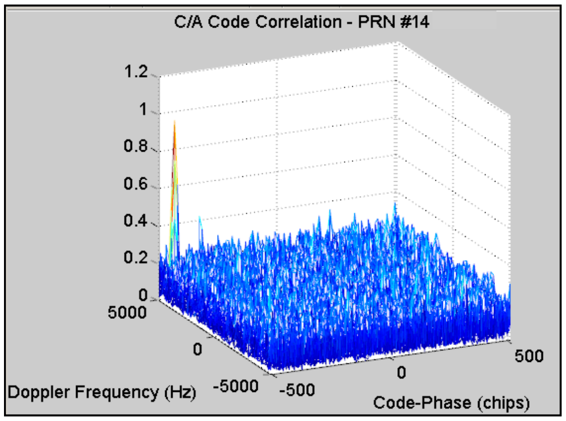

Figure 8: Output from parallel code phase search acquisition

What is of importance is the time required to acquire the signals. For test purposes, we first used the test platform to only acquire signals during a cold start operation. A single FFT based correlator was used for this purpose. The performance can be summarized as follows:

- *FPGA Clock Speed:* ***100 MHz***
- *# of PRNs checked:* ***32***
- *# of PRNs acquired: 7*
- *Total Clock Cycles:* ***62938944***
- *Time to Acquire: ~* ***0.63 seconds***

The same data was also acquired using a pure software receiver implemented in Matlab running on a 2.2 GHz Intel Core 2 Duo Processor based PC with 2 Gb of RAM. The time to acquire was about 47 seconds. This clearly illustrates the significant improvement in processing time required to acquire signals. The ability to perform parallel processing on a FPGA enables implementation of multiple correlators which can run in parallel during each clock cycle. The use of 4 correlators would reduce the acquisition time to a little over 0.15 seconds.

For test purposes, the receiver was initially implemented as a single channel receiver to track PRN 14. The tracking results for the first 2000 ms of tracking post acquisition of PRN 14 are shown in figure 9. The PLL discriminator clears shows the convergence in the carrier phase offset as the tracking loop progressed from the Pull-in mode to the Fine-tracking mode. No loss of lock was detected in the PLL operation as can be verified from the continuous Doppler frequency plot. Since the carrier loop was always in lock, the normalized coherent dot product based DLL discriminator was continuous able to determine the code phase difference between the incoming code phase and the locally generated replica of the incoming signal. Further, the incoming satellite signal for PRN 14 had a healthy C/N$_0$ of approximately 45 dB-Hz during the entire 2000 ms for which it was tracked. Such a high C/N$_0$ enables the two-quadrant Arctangent Costas PLL to function in an optimal manner.

The FPGA required a total of 75398 clock cycles to track each millisecond of incoming data. The GPS L1 C/A code exhibits a chipping rate of 1.023 MHz with a corresponding code period of 1ms. The receiver will be able to operate in real-time if the execution time required to process a single code period is below 1ms. Since our FPGA runs at a clock frequency of 100 MHz, we could clearly meet the goal of designing a real-time reconfigurable embedded GPS L1 receiver. The stated clock cycles also take into account the time required to perform baseband signal processing using the embedded hardcore PowerPC 405 processor which runs at a clock speed of 300 MHz.

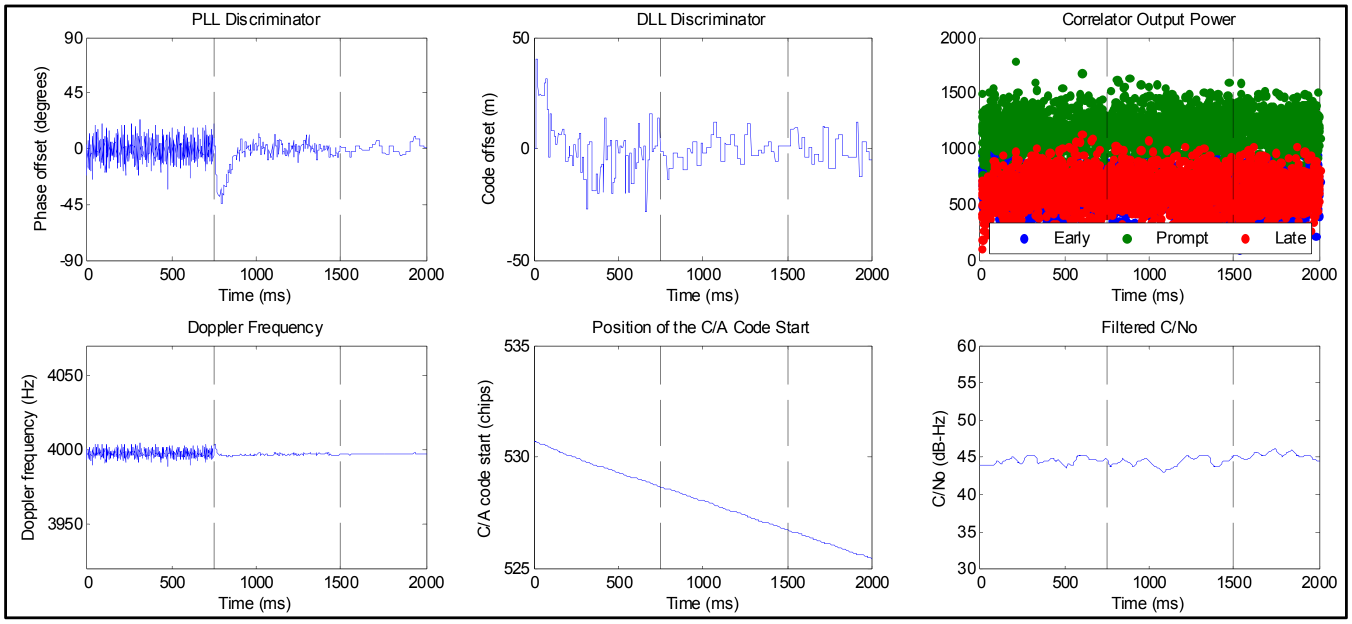

Figure 9: Tracking loop performance of receiver while tracking PRN 14

II. FPGA RESOURCE UTILIZATION

A FFT-based acquisition requires relatively large resources. However, FPGAs have sufficient resources available to enable multiple correlators and channels to be implemented. Figures 10 and 11 show the resources utilized to acquire signals using a single correlator. As the graph clearly shows, despite the FFT operation being computationally demanding, only about 10 % of the FPGA's resources were utilized. The use of 19 BRAMs, each with a capacity of 18 Kb, can be traced to the implementation strategy used to perform the FFT operation as shown in figure 6. The FFT was implemented as a Radix-4 FFT configured as a Burst I/O operation. This necessitated the need to buffer data samples before being transferred into the FFT IP block for performing the FFT operation.

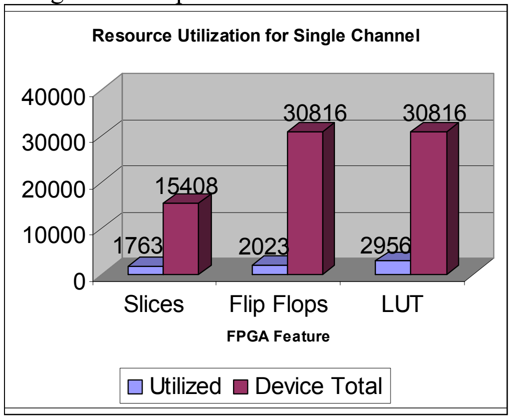

Figure 10: FPGA Resources Utilized to Implement a Single FFT-Based Correlator

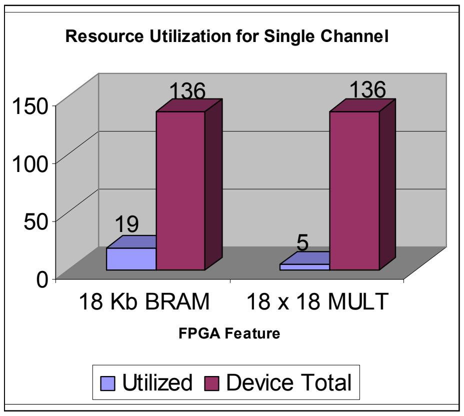

Figure 11: FPGA Resources Utilized to Implement a Single FFT-Based Correlator

The resources utilized to implement a single tracking channel are substantially lower compared to those utilized by a FFT-based correlator. This clearly justifies the reason for using ASICs for performing massively parallel serial search based acquisition in commercial receivers. Figure 12 and 13 illustrate the resources utilized to implement a single tracking channel.

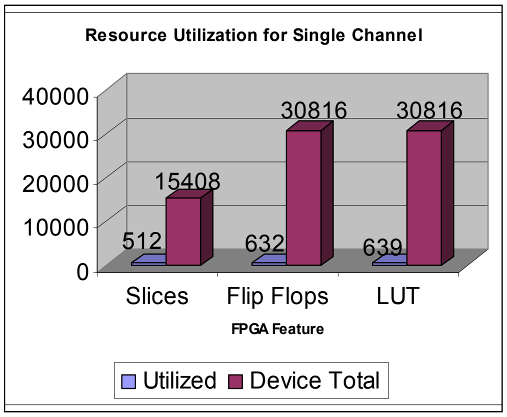

Figure 12: Resources utilized to implement a single tracking channel

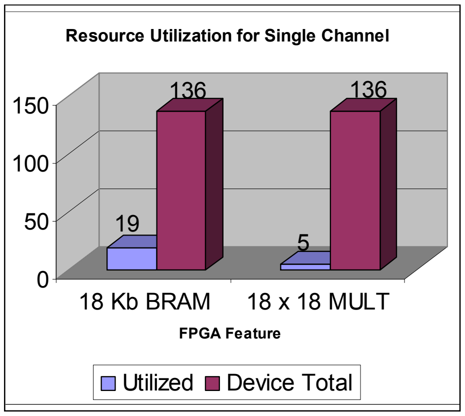

Figure 13: Resources utilized to implement a single tracking channel

Based on the resources utilized to implement a single FFT-based correlator and a single tracking channel, we decided to extend the design to implement a multi-channel receiver. Since synthesis tools are designed to optimize implementation of designs, the resources used to implement a multi-channel receiver with multiple correlators cannot be directly computed based on the resources utilized to implement a single correlator or tracking channel. This can only be determined through trail and error based on the total resources available on the FPGA under consideration and the extent of optimization the synthesis tool is able to perform.

Using the Virtex-II Pro FPGA, we were able to implement a 12 channel receiver comprising of 6 FFT-based correlators. This receiver could track the incoming signals in real-time thereby meeting our goal of implementing a reconfigurable embedded multi-channel GPS L1 receiver capable of running in real-time.

Several of the proposed new signals have a much larger chipping rate compared to the 1.023 MHz chipping rate used in the GPS L1 C/A signal. As a result, the IF and sampling rate required to process such signals would be higher resulting in increased data sizes. Preliminary analysis indicates that upto 3 FFT-based correlators can be implemented to acquire a GPS L5 signal. This signal has a chipping rate of 10.23 MHz. The present top of the line FPGAs such as the Virtex-5 series consists of over 6 – 7 times greater logic resources compared to those in the Virtex-II Pro. Hence, such FPGAs can be used to implement embedded reconfigurable multi-channel, multi-frequency GNSS receivers. Futher, designs can be easily transferred from one FPGA platform to the other. This is possible because the type of FPGA being used only influences the synthesis stages of translate, mapping and place and route. All other design stages involved in developing an embedded GNSS receiver are independent of the choice of the actual FPGA hardware.

APPLICATIONS

As previously stated, the purpose of our current research efforts it to develop a reconfigurable platform which can be used for a variety of applications. Most of our applications will be from an aviation signal integrity analysis perspective. Some of the research applications we intend to use this platform include:

1. DME/ TACAN Interference and Mitigation

The GPS L5 and Galileo E5 signals lie within the Aeronautical Radio Navigation Services band. Hence they are subject to signal interference from existing nav-aids such as the Distance Measuring Equipment (DME) and the Tactical Air Navigation (TACAN). Figures 12 and 13 show the frequency and time-domain plots for the GPS L5 signal collected at Stanford University subject to interference from six different DME/TACAN locations around the vicinity of Stanford University. Of particular importance is the sharp inference peak visible at a frequency of 1173 MHz resulting from the DME transmitter located at Woodside, CA.

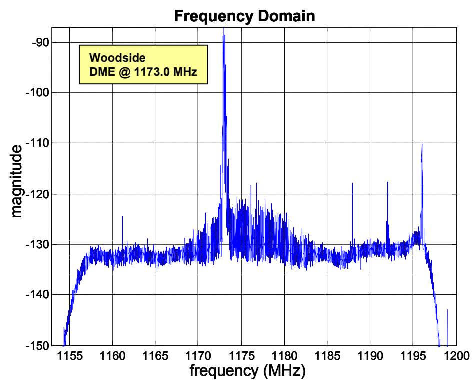

Figure 14: Frequency Spectrum of GPS L5 signal subject to DME/TACAN interference

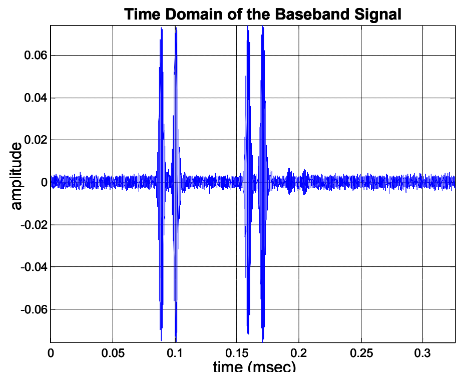

Figure 15: Time-Domain plot of GPS L5 baseband signal subject to DME/TACAN interference

Such interference results in repeated receiver lock of loss. Extensive research is being carried out by Stanford University researchers to identify optimal interference mitigation techniques [15]. The proposed reconfigurable platform can be used to test the performance of such new mitigation techniques.

2. Ionospheric Scintillation Analysis

Ionospheric scintillation is the phenomenon of deep signal fadings observed in GNSS signals. Such scintillation is usually not observed in the mid-latitude region, but it is frequently observed in the equatorial region during solar maximum. Signal to noise ratio or more precisely carrier to noise density ratio (C/No) of a certain satellite channel remains almost constant when no scintillation is observed. However, strong scintillation causes the signal C/No to fluctuate rapidly. The fluctuation can be more than 25 dB. Figure 16 compared the impact of scintillation on signal C/No [16, 17, 18].

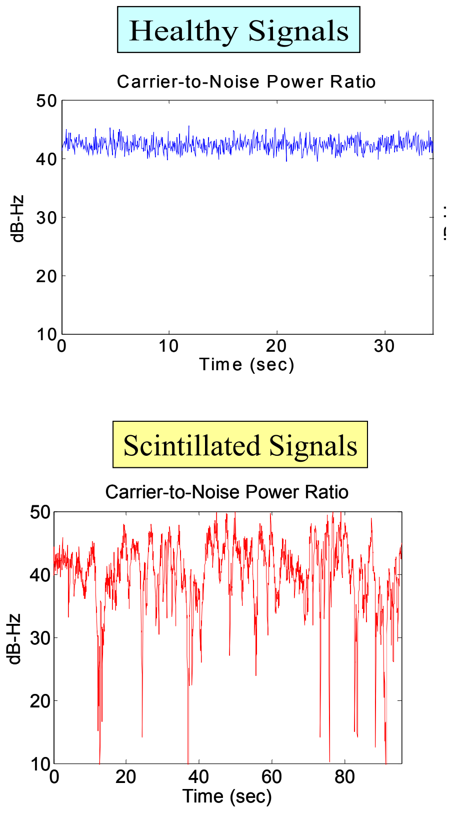

Figure 16: Impact of Ionosphere Scintillation on Signal C/No Ratio (Courtesy Dr. Tsung Yu Chiou, Stanford University)

Such deep signal fads result in repeated GNSS receiver loss of lock and are of extreme concern to aviation users. Since no past scintillated data is available for the GPS L5/ Galileo E5 signals, its effect on such signals cannot be empirically established. We intend to use the proposed reconfigurable platform to collect and analyze the effects of ionospheric scintillation on the L5 band of GNSS signals during the upcoming Solar maximum in the year 2011.

SUMMARY

As a first step in developing a Reconfigurable Platform capable of tracking multiple frequencies and constellations, we proposed and validated the feasibility of designing a reconfigurable embedded software receiver which can function in Real-Time. The complete receiver was designed using Model-Based Design Tools optimized for DSP functions. The small form factor of the hardware combined with the fact that no external DSP or a Host PC is required makes it an ideal test platform. We presented a few of the applications this reconfigurable platform can be utilized for. These applications can be developed in quick time using the appropriate Model-based design tools. Lastly, the fact that this particular hardware board is accessible to students at over 2000 universities makes it a perfect educational tool for teaching and learning GNSS receiver design.

ACKNOWLEDGEMENTS

This work was funded by the Federal Aviation Administration (FAA) through grant # CRDA 08-G-007. Their support is greatly appreciated. The authors also acknowledge the training support provided by Xilinx Inc. at their San Jose, CA based learning center. The opinions expressed and results presented in this paper are solely those of the authors. They do not necessarily reflect the views of the FAA, Xilinx Inc. or any of its vendors.

REFERENCES

1. Grace Xingxin Gao, David De Lorenzo, Alan Chen, Sherman Lo, Dennis Akos, Todd Walter and Per Enge, "Galileo GIOVE-A Broadcast E5 Codes and their Application to Acquisition and Tracking," *Proceeding of the ION National Technical Meeting 2007*, San Diego, CA, January 2007
2. Grace Xingxin Gao, Jim Spilker Jr., Todd Walter, Per Enge and Anthony Pratt, "Code Generation Scheme and Property Analysis of Broadcast Galileo L1 and E6 Signals", *Proceeding of the ION Global Navigation Satellite Systems Conference 2006*, Fort Worth, TX, September 2006.
3. Gao, Grace *et. al.* , "GNSS over China: the Compass MEO Satellite Codes", *Inside GNSS Magazine*, July-August 2007
4. Xilinx UG 190 Virtex-5 FPGA User Guide. Retrieved September 26, 2008 from www.xilinx.com/support/documentation/user_guides/ug190.pdf
5. Altera Stratix III Device Handbook. Retrieved September 18, 2008 from http://www.altera.com/literature/hb/stx3/stratix3_handbook.pdf
6. P. Misra, P. Enge, *Global Positioning System: Signals, Measurements and Performances*. Ganga-Jamuna Press, Lincoln, MA, 2006
7. E. Kaplan, C. Hegarty, *Understanding GPS Principles and Applications.* Artech House, Boston, MA, 2006
8. Akos, Dennis M., "A Software Radio Approach to GNSS Receiver Design", Ph.D. Dissertation, Ohio University, 1997.
9. Akos, Dennis M., *et al.*, "Real-Time GPS Software Radio Receiver," *Proceedings of ION National Technical Meeting 2001*, Long Beach, CA, Jan. 22-24, 2001, pp. 809 - 816.
10. Ledvina, B. M. *et. al.*, "Performance Tests of a 12-Channel Real-Time GPS L1 Software Receiver", *Proceedings of the ION GPS/GNSS Conference 2003,* Portland, OR, September 9-12, 2003, pp. 679 - 688.
11. Gold, Kenn., Brown, Alison "A Software GPS Receiver Applicable for Embedding in Software Defined Radios", *Proceedings of the ION GPS/GNSS Conference 2003*, Portland, OR, September 9-12, 2003.
12. Parkinson, K., "A real time multi-channel GPS positioning system architecture", *Proceedings of the IGNSS 2007 Symposium on GPS/GNSS*, Sydney, Australia, December 4-6, 2007.
13. Mumford, P.J. *et. al.*, "The Namuru Open GNSS Research Receiver", *Proceedings of the ION Global Navigation Satellite Systems Conference 2006*, Fort Worth, Texas, September 26 – 29, 2006, pp. 2847-2855
14. Xilinx Embedded System Tools Reference Manual, Retrieved September 19, 2008 from http://www.xilinx.com/support/documentation/sw_manuals/edk10_est_rm.pdf
15. Grace Xingxin Gao, "DME/TACAN Interference and its Mitigation in L5/E5 Bands", *Proceeding of the Global Navigation Satellite Systems Conference 2007*, Fort Worth, TX, September 25-28, 2007.
16. Seo, Jiwon, *et. al.*, "Characteristics of Deep GPS Signal Fading Due to Ionospheric Scintillation for Aviation Receiver Design", *Proceedings of the 12th International Ionospheric Effects Symposium*, Alexandria, VA, May 2008
17. S. Basu and S. Basu, *Equatorial Scintillations – A Review*, Journal of Atmospheric and Terrestrial Physics, Vol. 43, No. 5, June 1981.
18. J. Aarons, *Global Morphology of Ionospheric Scintillations*, Proceedings of the IEEE, Vol. 70, No. 4, April 1982.