Real-Time Software Receiver Tracking of GPS L2 Civilian Signals using a Hardware Simulator

B. M. Ledvina,
Applied Research Laboratories, University of Texas at Austin
M. L. Psiaki,
Sibley School of Mechanical and Aerospace Engineering, Cornell University
S. P. Powell, and P. M. Kintner,
School of Electrical and Computer Engineering, Cornell University

BIOGRAPHY

Brent M. Ledvina is a Postdoctoral Associate at Applied Research Laboratories at the University of Texas at Austin. He received a B.S. in Electrical and Computer Engineering from the University of Wisconsin at Madison and a Ph.D. in Electrical and Computer Engineering from Cornell University. His research interests are in the areas of ionospheric physics, space weather, estimation and filtering, and GPS/GNSS technology and applications.

Mark L. Psiaki is an Associate Professor of Mechanical and Aerospace Engineering at Cornell University. He received a B.A. in Physics and M.A. and Ph.D. degrees in Mechanical and Aerospace Engineering from Princeton University. His research interests are in the areas of estimation and filtering, spacecraft attitude and orbit determination, and GPS technology and applications.

Steven P. Powell is a Senior Engineer with the Space Plasma Physics Group in the School of Electrical and Computer Engineering at Cornell University. He has been involved with the design, fabrication, testing, and launch activities of many scientific experiments that have flown on high altitude balloons, sounding rockets, and small satellites. He has M.S. and B.S. degrees in Electrical Engineering from Cornell University.

Paul M. Kintner, Jr. is a Professor of Electrical and Computer Engineering at Cornell University. His interests as a rocket scientist range from exploring the northern lights to understanding space weather. His recent work with GPS receiver design and scintillations led NASA to appoint him chair of the Living With a Star Geospace Mission Definition Team.

ABSTRACT

A dual-frequency civilian L1/L2 GPS receiver has been tested that runs 12 tracking channels in real time using a software correlator. This work is part of an effort to develop flexible receivers that can use the new GPS L2 CM/CL signals as they become available on L2 without the need for new correlator hardware. The receiver consists of an RF front end, a system of shift registers, a digital data acquisition (DAQ) system, and software that runs on a 3.4 GHz PC. An analog mixing RF front end composed of separate L1 and L2 signal channels converts the L1 C/A code and L2 CM/CL code signals into two 2-bit digital data streams sampled at 5.714 MHz. The shift registers parallelize the two 2-bit data streams, which the DAQ reads into the PC's memory using direct memory access. The PC performs baseband mixing and PRN code correlations in a manner that directly simulates a hardware digital correlator. It also performs the usual signal tracking and navigation functions, under the control of a real-time Linux operating system.

The main contributions of this work are a design for an inexpensive dual-frequency GPS civilian L1/L2 RF front end and demonstration of GPS civilian L1/L2 software receiver running on a GPS signal simulator. This RF front end uses two commercially-available GPS RF front ends designed for the L1 C/A code signal. To accommodate the L2 CM/CL code signal, a synthesizer and mixer are located upstream of the RF front end and are used to up-convert the L2 signal to the L1 frequency minus roughly 6 kHz. The successful operation of the receiver while connected to a hardware simulator indicates that the receiver should function when the first civilian L2 signals become available.

The GPS civilian L1/L2 software receiver tracks 12 channels in real time and has a navigation accuracy of 1-3 meters. It requires 81% of the processing capabilities of a 3.4 GHz Intel Pentium 4 PC.

I. INTRODUCTION

A real-time global navigation software receiver provides distinct advantages in an evolving signal and code environment.The current Global Positioning System (GPS) is slated to expand its capabilities to include new civilian codes on the L2 frequency and a new L5 frequency. In fact, the first GPS satellite to broadcast the civilian L2 signal will be launched shortly. A receiver that uses a hardware digital correlator will require hardware mod-

ifications in order to use these new signals. In the near term, a receiver designer will be faced with a complex trade-off in order to decide whether the extra complexity is worth the improved performance that will accrue only very slowly as new GPS satellites replace older models. A software receiver can use new signals without the need for a new correlator chip. Given a suitable RF front end, new frequencies and new pseudo-random number (PRN) codes can be used simply by making software changes. Thus, software receiver technology will lessen the risks involved for designers during the period of transition to the new signals. Furthermore, a software receiver can be reprogrammed to use the Galileo system, GLONASS, or both, which provides an added benefit from the use of a software radio architecture. Thus, there are good reasons to develop practical real-time software GPS receivers.

This work focuses on the development of a dual-frequency GPS civilian L1/L2 front end and the testing of a dual-frequency GPS software receiver that utilizes the L1 C/A code and the new L2 civilian CM/CL codes. This receiver will be useful for estimating the ionospheric delay and as an instrument to study the ionosphere.

A hardware dual-frequency GPS receiver can be broken down into various components. First, a dual-frequency antenna, possibly followed by a pre-amp, receives the two L-band GPS signals. After the antenna comes an RF section that filters and down converts the GHz GPS signals to intermediate frequencies in the MHz range. The RF section also digitizes the signal. The next section contains a correlator chip that separates the signal into different channels allocated to each satellite. A modern receiver has 12 or more channels. For each satellite and each carrier frequency, the correlator mixes the Doppler-shifted intermediate frequency signal to base-band and correlates it with a local copy of a PRN code. The final components of the receiver consist of software routines that track the signals by controlling carrier and code numerically-controlled oscillators in the correlator chip, that decode the navigation message, and that compute the navigation solution.

A software receiver differs from a hardware receiver in one important way. The functions of the correlator chip are moved to software that runs on a general-purpose microprocessor. This move is illustrated in Fig. 1, where the software correlator is shown running on a general-purpose microprocessor. The RF front end outputs a binary bit stream. A data buffering and acquisition system reads the bit stream into the microprocessor's memory. The bit stream is then processed by the software correlator. The dual-frequency correlator shown in Fig. 1 processes inputs from one or two bit

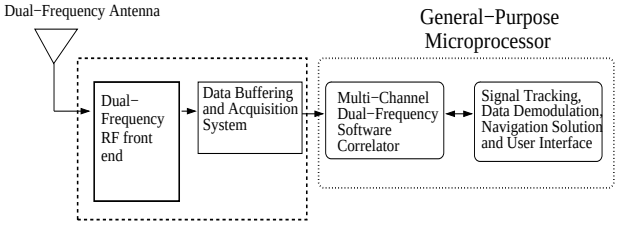

Fig. 1. A typical dual-frequency GPS software receiver showing the separation between special-purpose hardware and general-purpose hardware.

streams containing the signals at the two GPS frequencies.

GPS software receivers have received a lot of interest recently [1], [2], [3], [4], [5]. The work presented in this paper improves upon the software receiver of [5]. The receiver described here uses an analog mixing dual-frequency civilian L1/L2 RF front end and is tested on a hardware simulator. The receiver described here should be able to track and take advantage of the new civilian L2 signals as soon as they become available.

The remaining portions of this paper review the internal workings of a 12-channel real-time GPS civilian L1/L2 software receiver, describe the dual-frequency RF front end, and present experimental performance results for this system. The second section provides an overview of the components of the software receiver. Section III describes the system configuration including the data acquisition system and the use of previously-existing GPS software for tracking and navigation. Section IV describes the design and implementation of a dual-frequency civilian L1/L2 RF front end. Section V reviews the bit-wise parallel software correlation technique of [3] and [4]. Section VI reviews the real-time generation of oversampled PRN codes in [5] and [6]. Section VII discusses computational and memory requirements of the software correlator. Section VIII presents receiver performance results. Section IX discusses future work. The last section gives a summary and concluding remarks.

II. OVERVIEW OF THE SOFTWARE RECEIVER COMPONENTS

The software receiver described in this paper has a variety of hardware and software components that are explained in the next three sections. These components are described in broad terms here in order to present a coherent view of the different parts of the software receiver.

Fig. 1 illustrates the general functionality of the software receiver. A new dual-frequency civilian L1/L2 RF front end has been developed using commercial off-the-

shelf components and legacy GPS L1 C/A code RF front end hardware. This front end hardware provide the means to down convert and digitize the civilian L1 and L2 signals. The data buffering and acquisition hardware transfers the digital data stream into the PC for processing. One important aspect of the data buffering hardware is that it bit packs RF samples into shift registers before the data is read into the PC. This both reduces the bandwidth needed to read the data into the PC and sets up the software correlation process.

The remainder of the software receiver consists of software running on the 3.4 GHz PC. The digital data stream output from the RF front end is processed by the software correlator. This correlator processes 12 or more channels of the civilian L1 and L2 signals in real time. The correlator uses two technologies in order to process the RF front end data in an efficient manner. Without these two technologies, it would be very challenging to build a real-time 12-channel dual-frequency civilian L1/L2 software receiver using current microprocessors. The first technology, called bit-wise parallel signal processing, is a method for processing 32 RF samples in parallel. This method requires that the RF front end samples are bit packed into words prior to being read into the PC. Base-band mixing, code mixing, and accumulation are performed using bit-wise parallel processing of these bit-packed words. This speeds up the processing in the correlator by a factor of 2–4 as compared to using fixed-point multiply and accumulate (MAC) operations [3].

The second technology is a method for efficient real-time generation of over-sampled bit-packed PRN codes. This technology is necessary since the new civilian codes have longer periods than the L1 C/A code. In software receivers, it is computationally advantageous to pre-compute and store PRN code replicas, but the long length of the L2 civilian codes makes this approach difficult due to the required memory for storing the code replicas. For example, the receiver in [3] requires 960 kilobytes to store all the C/A code replicas. The new L2 CL code's period is 1500 times longer than that of the C/A code. Thus, it would require 1406 megabytes to store these code replicas in memory. The technology that allows efficient real-time generation of these PRN codes reduces the memory requirement to 81 kilobytes for the L1 C/A code, L2 CM and L2 CL codes. The down-side to this technology is that it adds a significant computational cost to the software receiver, increasing the required computational cost by roughly 40% [6].

The remaining components of the receiver are software for tracking the satellite signals, decoding the 50 bps navigation message, computing the navigation solution, displaying the receiver's operation, interacting with the user, etc. These components were available using previously-existing GPS software. Note that the computational cost of these functions is less than 5% of the computational cost of the software correlator.

III. SYSTEM CONFIGURATION

The fundamental component of the GPS software receiver is a personal computer. The current system uses a PC with a 3.4 GHz Intel Pentium 4 processor running the Real Time Application Interface (RTAI) operating system. RTAI is a hard real time variant of Linux implemented as a set of patches to the standard Linux kernel. Due to its real-time optimized design, RTAI provides low-latency interrupt responsiveness along with the ability to execute threads at regular intervals. This translates into a highly efficient and responsive operating system that reliably executes time critical code. An additional feature of RTAI is that it retains the functionality of Linux by running the kernel as the lowest priority thread. Thus, it is easy to develop, test, debug, and run real-time software.

The dual-frequency RF front end, which is described in the next section, outputs two digital data streams that are input into a data acquisition system. The data acquisition system reads the 2 pairs of digitized sign and magnitude bits from the RF front end into the PC. To make the process of reading data into the PC more efficient and to prepare for efficient correlation calculations, the DAQ card reads 32 bits of buffered samples at a time. A series of shift registers buffer the data, packing the L1 front end sign and magnitude bits and the L2 front end sign and magnitude bits into a 32-bit word. The data is formatted such that the most-significant 8 bits are the sign bits from the L2 front end, the next 8 bits are the sign bits from the L1 front end, the next 8 bits are the magnitude bits of the L2 front end, and the 8 least-significant bits are the magnitude bits from the L1 front end. A divide-by-8 counter converts the 5.714 MHz clock down to 714.30 KHz, which provides a signal indicating when the buffer is full.

The data acquisition system consists of a PC card and driver software. The card is a National Instruments PCI-DIO-32HS (6533) digital I/O card. Important features of this card are its 32 digital input lines, its direct memory access (DMA), and the availability of a driver for RTAI. The driver comes from a suite of open source drivers and application interface software for DAQ cards known as COMEDI (COntrol and MEasurement and Device Interface), which is freely available.

The software receiver is written entirely in ANSI C using tools available in standard Linux distributions. To promote portability of the software, no processor-specific assembly language or special instructions are

used.

Use of Existing Receiver Software

Existing receiver software is important to the implementation of this real-time GPS software receiver. The Mitel GPSArchitect GPS L1 C/A code receiver was ported to RT-Linux [7], subsequently ported to RTAI, and herein is referred to as Cascade. Since Cascade provides standard GPS functions such as signal tracking, data demodulation, and computation of the navigation solution, it is included as part of the real-time software receiver.

Three new developments to the Cascade software were required for it to operate with a dual-frequency correlator. First, a set of code and carrier tracking loops for the L2 CM and L2 CL signals were developed. The Cascade software already has a delay-locked loop (DLL) for code tracking and a frequency-locked loop (FLL) for carrier tracking. These loops are for tracking the L1 C/A code signal. Modified versions of these loops were implemented for the L2 CM and L2 CL signals. The modifications involved changing the nominal intermediate frequency and adjusting the integration interval to match the nominal 2-msec accumulation interval of the CM and CL codes.

Second, a method has been developed to remove the ionospheric delay from the L1 C/A code pseudorange measurement. The method involves computing an auto-regressive average of the high bandwidth difference of the L2 CM code and L1 C/A code start times to produce an estimate of the differential code delay. This estimate is output at 0.1 Hz. In the Cascade software, the estimated differential code delay is re-scaled to an equivalent range delay at the L1 carrier frequency. The L1 C/A code pseudorange is then corrected by this equivalent range delay.

A more accurate ionospheric delay estimation method uses both the differential code delay and the differential carrier phase advance. The appropriately-scaled differential code delay is an absolute measure of the of ionospheric delay but tends to be very noisy. The appropriately-scaled differential carrier phase advance is a relative measurement of the ionospheric delay and is much less noisy. An estimation method that uses the auto-regressive average of the differential code delay to produce a low-bandwidth estimate of the ionospheric delay and the differential carrier phase advance as a high-bandwidth estimate has been implemented.

Third, an acquisition method for detecting the L2 CM/CL signal has been developed. The Cascade software uses a brute-force code phase offset and Doppler frequency search routine for C/A code acquisition. Since the CL and CM code periods are respectively 20 and 750 times longer than the C/A code period, it is easier and more likely to acquire the C/A code first if the signal is strong. The modified Cascade software first acquires the L1 C/A code, and the L1 carrier Doppler shift is used to program the L2 carrier NCO. The correct L2 carrier frequency is determined via re-scaling of the acquired L1 Doppler shift by the ratio of the L2 to the L1 frequencies. This re-scaling must take into consideration receiver clock frequency error. This reduces the CM and CL acquisition search space to one dimension, the code start time dimension. The CM and CL acquisition then carries out a brute-force search for the start time. In the future a better acquisition strategy will be implemented, one that recognizes that the L2 CM code start time can fall only within a limited range of offsets from the L1 C/A code start time. The CL code acquisition can be sped up by decoding either the navigation message on the C/A code or the CM code signals. These messages contain time-of-week information that allow quicker acquisition of the CL code.

One missing component of the Cascade software is a data demodulation routine for the convolutionally-coded 50 symbol per second data stream on the CM code. This data stream provides ephemeris and clock information that is also available on the L1 C/A code 50 bit per second data stream. Because the receiver is configured to track the C/A code, it is not necessary to decode the CM code data stream.

The dual-frequency software correlator has been designed as an independent software module that interacts with other parts of the receiver according to well-defined interface specifications. This modular approach provides flexibility in the internal workings of the receiver. One benefit of this modularity is that the mixing methods and correlation routines are transparent to the other standard software modules. This enables quick changes in correlator design that do not significantly affect other parts of the GPS receiver. Another benefit is that the receiver can be reconfigured on the fly to operate as a single-frequency L1 C/A code receiver.

IV. DESIGN OF A DUAL-FREQUENCY RF FRONT END

The software receiver uses a dual-frequency RF front-end in order to down-convert, sample, and digitize the received L1 C/A-code spectrum and L2 CM/CL-code spectrum. The front-end has been designed to have good sensitivity while being simple to construct.

An initial design scheme considered using direct RF sampling because mixers are eliminated and because both frequency bands are captured in a single digitized data stream [8]. Unfortunately, the cost and size of the

required RF filters proved prohibitive. Therefore, it was decided to use a traditional mixing scheme in order to implement the new RF front-end.

The mixing design implements two separate RF front-ends, one for the L1 band and one for the L2 band. A common oscillator is used to drive the mixers and sample clocks of both front-ends. The two front-ends have been implemented using commercial off-the-shelf (COTS) parts as much as possible in order to reduce costs and minimize design risks.

The L1 front-end has been implemented using the existing Zarlink GP2015 chip [9]. This front-end uses three stages of off-chip filtering and one on-chip filter interspersed with three stages of on-chip mixing and 3 stages of on-chip amplification. The net bandwidth of the 4 filters is 1.9 MHz, which is wide enough to capture the main lobe of the C/A spectrum while enabling the front-end to sample at $40/7 = 5.714286$ MHz without picking up a significant amount of aliased out-of-band noise. The nominal L1 frequency gets down-converted to $4427/3150 = 1.405397$ MHz, but with a phase reversal that is characteristic of high-side mixing. The output samples are digitized using two bits, one a sign bit and the other a magnitude bit. These 2 bits represent the 4 possible output levels -3, -1, +1, and +3. The on-chip amplification includes an automatic gain controller that uses feedback in order to achieve optimal use of the two output bits.

The L2 front-end can be almost identical to the L1 front-end, except that it needs to work with a different carrier frequency. It can have the same sensitivity, the same net effective filter bandwidth, and the same sampling frequency. This is true because the L1 C/A code and the L2 CM/CL code power levels are similar and because their bandwidths are identical – their different pseudo-random number (PRN) codes both chip at 1.023 MHz. In short, the L2 front-end should look identical to the Zarlink GP2015 except that its filters and mixers should be modified in order to process a signal centered at 1227.6 MHz. Unfortunately, to the authors' knowledge, no such RF front-end is commercially available.

The approach taken for the L2 RF front-end design is to filter the L2 signal, up-convert it to the L1 frequency band, and then feed it into a second Zarlink GP2015 front-end. The up-conversion is accomplished by mixing the L2 signal with a signal whose frequency equals the difference between L1 and L2: 347.82 MHz. A filter must be used upstream of the up-converter in order to reject image signals centered at the frequency (L1 + 347.82 MHz) = 1923.24 MHz. The mixer down-converts this frequency to the L1 frequency, and the filter is needed in order to keep signals near this frequency from interfering with the up-converted L2 signal. A filter centered at L2 and having a bandwidth of about 20 MHz has been used to reject the image signals.

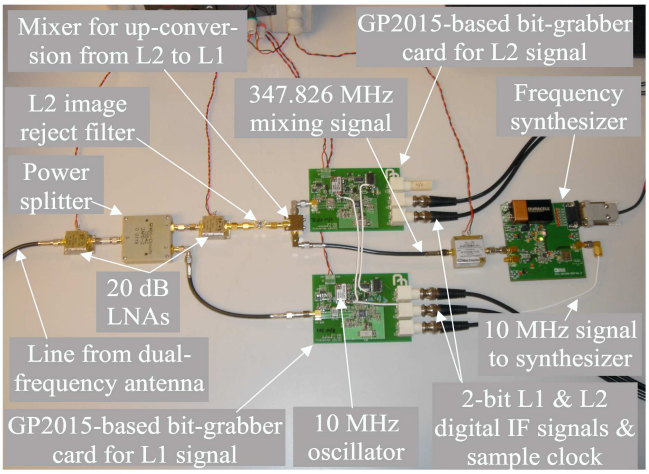

Fig. 2. Dual-frequency civilian L1/L2 RF front-end that uses 2 COTS L1 RF front-ends along with a mixing up-converter that maps the L2 signal approximately to the L1 band.

The required mixing signal needs to be phase-locked to the same 10 MHz reference oscillator that drives the L1 and L2 Zarlink GP2015 chips. A frequency synthesizer generates the mixing signal by multiplying the reference oscillator frequency by the factor 800/23. This results in a mixing signal with a frequency of 347.826087 MHz. The use of this signal to up-convert the L2 frequency results in a signal with a nominal frequency of 1575.426087 MHz, which is 6.087 kHz above the nominal L1 frequency. This difference can be tolerated because it is less than 0.33% of the effective filter bandwidth of the Zarlink GP2015. The nominal L2 frequency gets mapped to the frequency $10138/7245 = 1.399310$ MHz at the output of its GP2015 chip, and it has a carrier phase reversal due to high-side mixing. The software receiver implements L2 baseband mixing algorithms that are tailored to this modified nominal intermediate frequency.

A labeled photograph of the new dual-frequency RF front-end is shown in Fig. 2. The main engines of the RF front-end are the two bit-grabber cards. Each one has a Zarlink GP2015 as its heart. Each card also has various filters, a 10 MHz reference oscillator, a synthesizer that creates the 40/7 MHz sample clock from this oscillator, and other needed circuitry. Each card takes an RF signal in the L1 frequency band at its input, which is located on its left-hand side, and it outputs 3 digital signals on its right-hand side, a sample clock signal, a sign bit signal, and a magnitude bit signal. The lower bit-grabber card is used for the L1 signal, and the

upper bit-grabber card is used for the L2 signal. The two cards use a common oscillator and sample clock, which is provided by the L1 card. The L2 card's oscillator and sample clock are disabled. The two white wires that connect between the two cards transfer the L1 card's oscillator and sample clock signals to the L2 card. The up-converting mixer that translates the L2 signal into the L1 band is shown just to the left of the L2 bit-grabber card. It mixes the output of the L2 image reject filter, which lies to the mixer's left, with the 347.826087 MHz signal that comes out of the frequency synthesizer. This synthesizer is located at the far right-hand side of the figure, and its input is the signal from the L1 bit-grabber card's 10 MHz oscillator. The 20 dB low-noise amplifier (LNA) to the left of the L2 image reject filter provides a gain that compensates for the losses in the image reject filter and the mixer. The power splitter to the left of this LNA splits the L1 and L2 signals and sends them to their respective front-ends. The second 20 dB LNA to the left of the splitter helps to set the noise figures of both front-ends and to place the signal power levels within the allowable input ranges of the two GP2015 chips.

Care has been taken in designing the amplification, filtering, and mixing that is upstream of the L2 bit-grabber card. The mixer is a passive device (Mini-Circuits ZX05-25MH [10]). In order to function properly, the mixer requires the mixing signal from the synthesizer card (Analog Devices ADF4360-8 [11]) to have a certain prescribed power level. An amplifier with a gain of 19 dB has been added between the synthesizer and the mixer in order to give the mixing signal the correct power level. Although not labeled, this amplifier is shown in the figure. It is a Mini-Circuits ZFL-500HLN [12]. The L2 image reject filter upstream of the mixer is a SAW filter, part JRC NSVS1108 [13]. It has at least 20 MHz of bandwidth centered at L2, and it has at least 27 dB of attenuation (35 dB typical) at the image frequency of 1923 MHz. The 20 dB LNA upstream of the image-reject filter is a Mini-Circuits ZEL-1217LN 5. As previously mentioned, it counteracts the losses in the filter and the mixer. The power splitter is a Mini-Circuits ZAPD-2 [14].

There are a total of 5 independent output signals from the 2 RF front-ends. They are the sign and magnitude bits of the L1 and L2 bit-grabber cards and the sample clock of the L1 card. The L2 bit-grabber card's disabled sample clock is the unused coaxial output shown in Fig. 2.

The 5 output signals from the two RF front-ends are input to the software receiver computer through a custom-designed digital data packaging circuit that is interfaced to a COTS data acquisition card. The custom-designed circuit contains four 8-bit shift registers. One shift register holds 8 successive samples of the L1 bit-grabber's magnitude bits, the second shift register holds 8 samples of the L1 sign bits, the third holds 8 samples of the L2 magnitude bits, and the fourth holds 8 samples of the L2 sign bits. The shift registers are clocked by the L1 bit-grabber's sample clock. The custom-designed circuit also implements a divide-by-8 counter. This counter sends a pulse to the data acquisition card in order to inform it that the shift registers have a new 8 samples worth of data. The data acquisition card then reads all 32 bits from the 4 shift registers in one digital input operation. This read operation completes the connection between the dual-frequency antenna and the software receiver computer.

The new front-end design has been tested independently of the dual-frequency software receiver. The Spirent hardware simulator has been used to generate an L1/L2 civilian signal with known properties for input to the new RF front-end. The output data from the front-end has been input to the computer data acquisition system and recorded on disk for later post-processing. Post-processing has been accomplished using off-line software receiver functions that run in MATLAB. It involves acquisition of the L1 C/A code and the L2 CM code using the block acquisition techniques described in Refs. [15] and [16].

Two different GPS signals have been acquired, PRN 02 and PRN 09, and the acquired signals have been checked in four different ways. First, the code start times of the L1 and L2 signals have been compared. They match to within 1/38th of a code chip or better, which is within the resolution of the acquisition calculations. Second, the Doppler shifts detected on L1 and L2 have been checked. They have been compared with the simulator's "truth" Doppler shifts, and they have been compared with each other. The latter comparison factors in the ratio of the nominal L1 and L2 carrier frequencies. The acquired Doppler shifts match the simulator's "truth" values to within the frequency accuracy of the front-end's reference oscillator. The L1 and L2 shifts agree with each other to within 22 Hz when measured at L2. This level of agreement is within the resolution of the acquisition calculations.

In the third check, the signals have been re-acquired 3 seconds later in the data set, and the average code chipping rates have been determined. This calculation differences the acquired code start times at the beginning and end of this 3-second interval and divides the number of elapsed code chips by this difference. The detected Doppler shifts in the L1 and L2 PRN code chipping rates agree with the acquired carrier Doppler shifts to within the precision of this calculation. Note

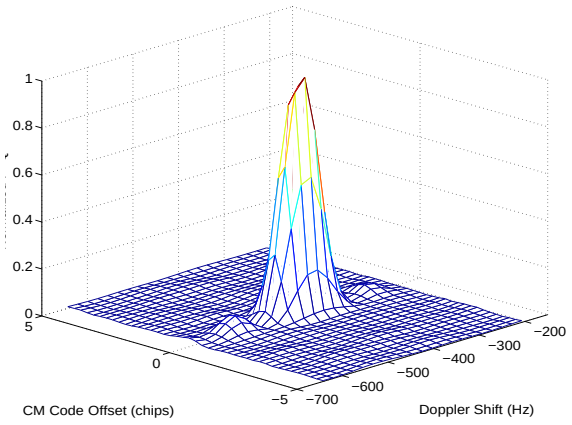

Fig. 3. Normalized L2 CM code acquisition statistic vs. code offset and carrier Doppler shift for a PRN 02 acquisition case.

that the check for agreement involves factoring by the ratios of the nominal code chipping rate to the nominal carrier frequencies. This test verifies that the RF front-end does not lose any samples during the transfer of data to the computer.

The fourth check has been to examine the carrier-to-noise ratios, C/N_0, of the acquired signals. The acquired L1 signals have $C/N_0 = 44.1$ dB-Hz for PRN 02 and $C/N_0 = 45.6$ dB-Hz for PRN 09. This is typical performance for the Zarlink GP2015 RF front-end. The acquire L2 signals are slightly stronger: $C/N_0 = 45.7$ dB-Hz for PRN 02 and $C/N_0 = 46.5$ dB-Hz for PRN 09. On average, the L2 C/N_0 values are about 1 dB higher than the corresponding L1 values.

The slightly better performance of the L2 channel is surprising given that the Spirent simulator was set to produce L2 CM/CL code that had 4.5 dB less power than the L1 C/A code. The cause of this counter-intuitive result may be the extra LNA located in the L2 chain between the power splitter and the L2 image reject filter. It may go beyond mere compensation for the losses in the filter and the mixer; it may also serve to improve the noise figure of the L2 path. This is a limited test, of course, and these results might change somewhat if averaged over many tests, but given that the L2 signal in the test was 4.5 dB weaker than the L1 signal, any modification to these results would probably continue to indicate that the L2 channel has the better noise figure.

The quality of the new RF front-end is illustrated by Fig. 3, which shows a typical L2 CM-code acquisition plot. This acquisition is for PRN 02. It uses a coherent integration interval of 0.010 sec and non-coherent integration over 0.040 sec, i.e., over 4 coherent intervals. The plot looks like a textbook example: Its correlation peak is 2 chips wide in the code offset direction. The main lobe in the frequency direction extends +/-100 Hz from the peak frequency. The frequency extent of the main lobe and the frequency extents and locations of the side lobes are consistent with the 0.010 sec coherent accumulation interval. Thus, the new RF front-end provides a sensitive and reliable means of mixing the L1 and L2 civilian signals down to intermediate frequencies, sampling them, digitizing them, and sending them to a computer so that they can be processed by the dual-frequency software receiver.

V. REVIEW OF BIT-WISE PARALLEL CORRELATION AND ACCUMULATION

The software receiver tested here uses bit-wise parallel Boolean logic computations to calculate correlations. This approach differs from other software receivers that perform integer-based computations in either the time domain or frequency domain. Bit-wise parallel correlation processes 32 RF output data samples at a time. This reduces the number of instructions by a factor of 2 or more [3]. Bit-wise parallel correlation is similar to how a hardware correlator functions. The bit-wise nature of this correlation method is aided by how the RF front end output data is read into the PC's memory. 32-bit words, representing 8 samples from the L1 front end and 8 from the L2 front end, are synchronously read into the PC's memory. There is no translation to an integer representation of the signal. Boolean logic operates on 32-bit words, each of which represents the sign or magnitude of various signals at 32 successive samples.

The software receiver requires both code and carrier replicas. The code replicas are computed in real-time using the method described in the next section. In order to reduce the computational load during correlation, the carrier replicas are pre-computed and stored in memory. In-phase and quadrature L1 and L2 carrier signals are computed on a coarse frequency grid with respective 175 Hz and 87.5 Hz spacings. The initial phase offset for these replicas is 0. These replicas require a total of 1328 kilobytes of storage. The correlator compensates for the use of carrier replica signals of the incorrect frequency and phase. The in-phase and quadrature accumulations are rotated by an angle that represents the average phase difference between the available base-band signal and the intended signal at the correct frequency and phase. This procedure is thoroughly explained in [3], [4], and [5].

Base-band mixing is performed by computing the bit-wise exclusive-or of the RF data sign words and carrier replica sign words over a C/A code period. For the

CM/CL codes, accumulations are computed over 1023-chip intervals which are nominally 0.002 secs. The RF data magnitude words and carrier replica magnitude words are not operated on, and are simply passed to the next stage. The resultant base-band signal has a 3-bit representation consisting of a 32-bit sign word, a 32-bit low magnitude word, and a 32-bit high magnitude word for 32 successive samples. Code mixing for the C/A code also involves a sequence of bit-wise exclusive-or operations. To mix the base-band signal with the C/A code, the base-band sign words are exclusive-ored with the code replica's sign words. The resultant signal also has a 3-bit representation. Code mixing for the CM/CL code is complicated by the fact that the codes are time multiplexed and that the CM code contains unknown data bits. Two replica CM/CL PRN codes are generated such that one has a +1 data bit on the CM code and the other has a –1 data bit. This is done for both the prompt and early-minus-late codes. All four of these codes are mixed with the in-phase and quadrature base-band L2 signal during code correlation. Accumulation involves summing the resultant signal over the accumulation interval, which is 1023 chips. Accumulation is performed by separating the eight different combinations of the 3-bit signal and then using a look-up table to count the instances of each of the eight combinations. Accumulation of the CM and CL codes is performed in a similar manner as the C/A code accumulation for each 2-msec interval. The resultant in-phase and quadrature accumulations for the +1 and –1 CM data bit cases are then added to produce the CL code accumulation or subtracted to produce the CM code accumulation. This procedure is described in detail in [5].

VI. REVIEW OF REAL-TIME GENERATION OF OVER-SAMPLED PRN CODES

This section reviews the method for real-time generation of bit-wise parallel representations of the over-sampled versions of the required PRN codes. The software correlator needs to generate bit-wise parallel representations of the prompt and early-minus-late over-sampled PRN codes, because they are too long to pre-compute and store in memory.

The real-time generation of over-sampled bit-wise parallel PRN codes requires several values: the length of a PRN code chip, the sample interval, the early-minus-late delay, the end time of the first code chip relative to the first sample time, and the actual +1/–1 values of the PRN code chips. This generation method of the bit-wise parallel PRN codes is dependent on the assumption that the sampling frequency, the chipping rate, and the early-minus-late code delay are all constant.

Over-sampling and translation into a bit-wise parallel representation are accomplished simultaneously through the use of three pre-computed tables, one for the prompt code, one for the magnitude of the early-minus-late code, and one for the sign of the early-minus-late code. The three tables' size is relatively small and is independent of the PRN code period. In fact, it can be reused for PRN codes in a multi-channel receiver. The PRN code chip length, the sample interval, the early-minus-late delay, and a desired code phase resolution are used to pre-compute the look-up tables. This tables contains the bit-packed 32-bit representations for all the possible PRN code combinations and initial phase offsets found in 32 successive samples.

The tables have 2 variable inputs: the sequence of code chip values that span a 32-bit data word and the end time of the initial prompt chip relative to the data word's first sample. A calculation is made using these values that gives an index into the 3 tables, which in turn outputs the correct bit-wise parallel representation of the prompt and early-minus-late codes. An exhaustive description of this technique is given in [6]. Its application to a GPS civilian L1/L2 software receiver is described in [5].

VII. COMPUTATIONAL LOAD AND MEMORY REQUIREMENTS

The computational load and memory requirements of a software receiver are two important factors. The software receiver described here has a significant computational load, demanding a high-end PC or DSP chip in order to track 12 or more channels. Its memory requirements are modest, requiring only a few megabytes of storage for both the code and data.

Computational Load

The bit-wise parallel correlation and accumulation calculations use mostly simple logic and table look-up operations in order to form the 4 accumulations for each PRN code. Generating the over-sampled PRN codes requires additional computations. A description of the computational requirements in terms of instructions per accumulation can be found in [5].

Fig. 4 shows the computational performance requirements for the software receiver to track 12 dual-frequency channels in real time. The figure demonstrates computational requirements for 32-bit and 64-bit software using 2-bit and 1-bit RF data streams. The 32-bit and 64-bit cases refer to a relatively recent development in desktop microprocessors. Conventional microprocessors process data in 32-bit chunks, but recently Intel and AMD have released new microprocessors that process data in 64-bit chunks. Current versions of these microprocessors can be run either 32-bit-complied software or 64-bit-complied software. The 32-bit and 64-bit test cases in Fig. 4 demonstrate bit-

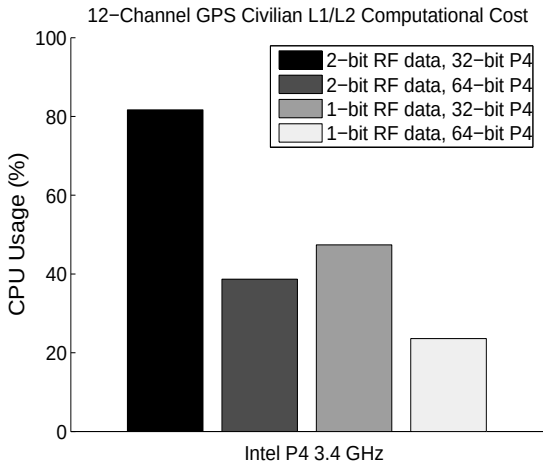

Fig. 4. Computational performance comparison using 32-bit and 64-bit code processing either 2-bit or 1-bit RF data streams running on a 3.4 GHz Intel Pentium 4 microprocessor.

wise parallel processing of 32-bit and 64-bit samples of data, respectively. The processing time is reduced by nearly 40% when moving from 32-bit software to 64-bit software. This is expected as the real-time generation of the PRN codes and bit-wise parallel correlation requires about half as many operations when processing 64-bit samples at a time.

The cases that show processing of 2-bit and 1-bit RF data streams demonstrate the reduction in complexity when going from 2 bits to 1 bit when computing the bit-wise parallel accumulations. A reduction of less than a factor of 2 is expected when reducing the RF front end bits from a 2-bit digitization to a 1-bit digitization and using the real-time generation of the oversampled PRN codes [5]. The computational requirements of the bit-wise parallel processing are dependent on the number of RF data bits, but the generation of over-sampled PRN codes is independent of the number of RF data bits. This translates into a decrease of 25%, as seen in Fig. 4, when going from 2-bit RF front end data streams to 1-bit RF front end data streams. Note that roughly a 2-dB loss in C/N_0 occurs when reducing the number of RF front end data bits from a 2 to 1 [17].

Note that this algorithm can be adapted to work with a different number of bits in the representation of the RF front end output data and of the cosine and sine mixing signals. Increases in the number of bits tends to increase the complexity of the bit-wise parallel correlation, and, in turn, the computational requirements.

Additional optimization of the correlation algorithms is possible. Using an Intel C compiler for Linux, off-line computational performance testing indicates that a significant speed-up on the Pentium 4 microprocessor is possible. Compared to the code produced by the GCC 3.3 compiler, the Intel compiler produced code that runs approximately 27% faster. This improvement in speed is most likely due to the Pentium-4-optimized code produced by the Intel compiler. The implementation of the software receiver reported on in this paper uses the GCC 3.3 compiler. Note that these performance values are dependent on the compiler version and how well it is suited to generating optimized code for a particular platform, or for example how well it generates 32-bit software versus 64-bit software.

Additional performance gains are possible by using dual microprocessors or dual-core microprocessors. To take advantage of these microprocessor architectures, the software correlator would need to be modified to run as a multi-threaded application. This modification is straightforward and could easily be implemented.

Memory Requirements

The pre-computed base-band mixing signals and PRN code over-sampling tables require a certain amount of memory. Each replica base-band mixing signal must occupy 180 32-bit words for the L1 C/A code signal and 359 32-bit words for the L2 CM/CL code signal. These sizes guarantee covering the full 5,714 and 11,429 RF front end samples for, respectively, the 1 ms C/A code accumulation and 2 ms CM/CL code accumulation intervals. Thus, 180×4=720 bytes are required for each bit of each carrier signal that must be stored for the C/A code signal, and 359×4=1436 bytes are required for each bit of each carrier signal that must be stored for the CM/CL code signal. The sine and cosine signals each have two-bit representations, which translates into a total storage requirement of 2880 bytes for the L1 carrier replicas and 5,744 bytes for the L2 carrier replicas. For the L1 C/A code signal, there are 115 Doppler shifts that must be stored in order to cover the –10 KHz to +10 KHz range with a 175 Hz grid spacing. For the L2 CM/CL signal, 179 Doppler shifts must be stored to cover the –8 KHz to +8 KHz range with a 87.5 Hz grid spacing. This translates into 1328 kilobytes of storage for all of the carrier replica signals.

The PRN code look-up tables require a modest amount of memory. The required memory is given by equation (56) in [5]. The equation is not repeated here because it requires explanation of a number of variables that are beyond the scope of this review. The required memory scales linearly with the code phase resolution and exponentially with the number of samples per data word. The memory requirements for the three PRN code tables have been calculated. These calculations assume that the nominal chipping rate is 1.023×10^6) chips/sec, the early-minus-late code spacing is one-half a chip, and

32 samples per data word. This software receiver uses the following parameters: an RF sampling interval of 175 nsec, and a range-equivalent code phase resolution of 3.75 m. These values yield at most 8 code chips per data word, 81 tabulated code start/stop times, and 6912 entries per PRN code table. The three PRN code tables together require 81 kilobytes of memory.

One method of simplifying the real-time generation of sampled codes is to make use of the fact that the C/A code is relatively short. It is possible to use pre-computed tables of the C/A code chips with this method. The advantage is that the computational requirements are slightly decreased in favor of a modest increase in the required memory. The total additional memory required to store the auxiliary tables for all 32 satellites is only 32 kilobytes.

Additional temporary memory is required to store the CM/CL code's auxiliary table that contains the code chips for the current processing interval. For $L \leq 8$ chips per data word the required memory equals $(M + 1 + L)$ bytes. For this receiver, $L = 8$ and $M = 2046$ and the auxiliary table requires 2 kilobytes.

The micro-computer uses two circular buffers to store the most recent 21 msec of RF front end data from each RF front end data. These buffers occupy 29.2 kilobytes of memory.

The total amount of memory required for storing the tables and data is 1472 kilobytes. The machine code for the software receiver requires additional storage on the order of roughly 1500 kilobytes.

VIII. RECEIVER PERFORMANCE RESULTS

Navigation and tracking performance has been tested using a hardware simulator. A simulator is required because the current GPS constellation does not include any GPS satellites that broadcast the new civilian L2 signal. The simulator is a Spirent GS7700 simulator, which is capable of generating signals containing the L1 C/A code and the L2 CM/CL code.

An interesting note is that the Spirent simulator needs to be coaxed into producing ionospheric delays that cannot be easily corrected using the broadcast ionospheric model. The simulator's default setting is to adjust its RF output using a model of the ionosphere specified by the eight broadcast model parameters. These same eight parameters are then broadcast in the navigation message. This allows a single-frequency receiver to correct for all of the ionospheric delay. Fortunately, it is possible to adjust either the parameters applied to the RF signal or those that are broadcast. The α_0 parameter of the broadcast model is essentially a bias term that

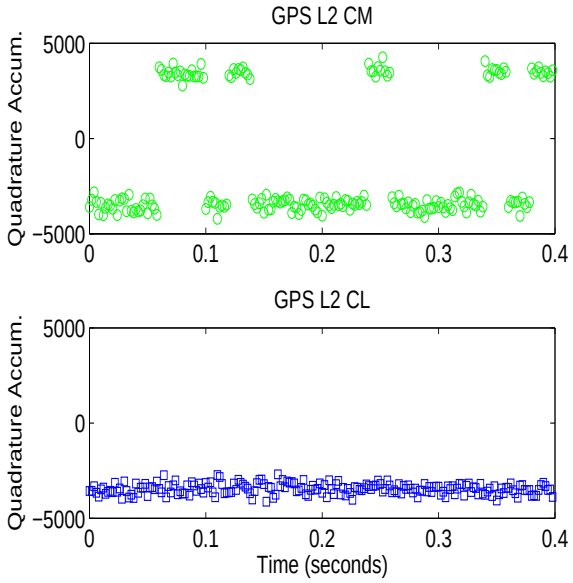

Fig. 5. Time history of 2-msec quadrature accumulations of the L2 CM and CL codes.

is minimum at zenith and increases as a function of the zenith angle [18]. The α_0 term for the RF-induced ionospheric model has been increased by 5 nsec over that of the broadcast model giving a larger and spatially variable ionospheric delay for all satellites. It is important that the additional ionospheric delay is not the same for the all satellites, otherwise it would be lumped into the receiver clock correction.

The presence of data bit transitions is tested by comparing the accumulations for the CM and CL codes. Fig. 5 shows the 2-msec CM and CL code quadrature accumulations for PRN 09. Since the Cascade software provides carrier tracking using an FLL, the in-phase and quadrature accumulations have been rotated after-the-fact to move the majority of the power into the quadrature component. This allows one to look for the phase transitions due to data bit transitions. It is clear from Fig. 5 that the accumulated CM code signal exhibits 180-degree phase transitions every so often. Closer inspection of Fig. 5 reveals that the phase transitions occur at multiples of 10 accumulations, or at an equivalent rate of 50 Hz, illustrating the presence of the 50 symbol per sec convolutionally-encoded data message. The CL code signal shows no phase transitions, which is expected, as it is a data-less signal.

An important aspect of a dual-frequency receiver is its ability to estimate the differential delay between the arrival of the L1 and L2 signals. The differential delay is linearly related to the ionospheric delay, which is used to improve the navigation solution or simply

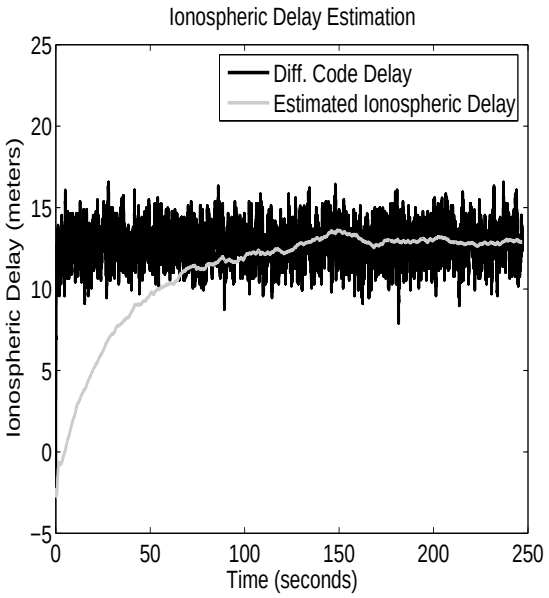

Fig. 6. Two methods that show the estimation of the ionospheric delay.

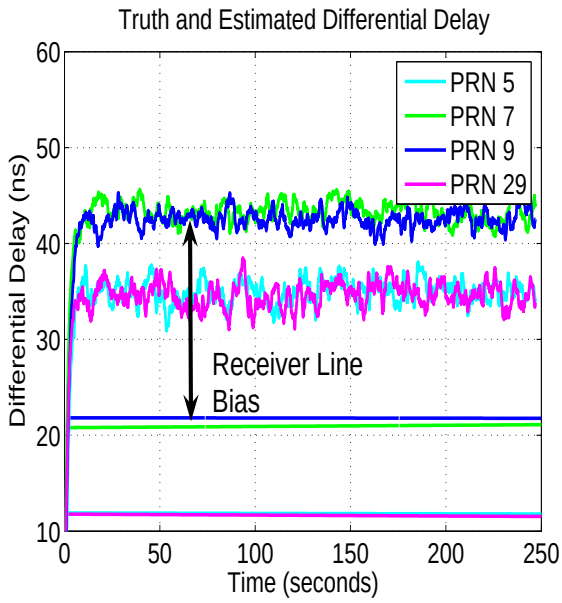

Fig. 7. Estimates of truth and differential code delay that illustrate the receiver line bias.

recorded as a measurement. The simplest way to estimate the differential delay is to measure the difference between the code arrival times of the L2 signal and the L1 signal. This estimate tends to be noisy because of the lack of resolution in determining the code phase. The black curve in Fig. 6 shows the 500-Hz differential code delay converted into an equivalent ionospheric delay measured on PRN 09. A more accurate method to estimate the differential delay is to use both the differential code delay and the differential carrier phase advance, as mentioned in Section III. The gray curve shows the estimate of the ionospheric delay that uses an auto-regressive average of the differential code delay as a low-bandwidth measurement of the delay along with the appropriately-scaled differential carrier phase advance providing the high-bandwidth variation in the ionospheric delay. Note that the first 100 seconds of the estimated ionospheric delay represent the transient stage of the estimator. This latter estimate of the ionospheric delay follows the same variations in the differential code delay estimate, but is less noisy.

Another important aspect of a dual-frequency receiver is the receiver line bias, which is the electronics-induced propagation delay that may be different for the L1 and L2 signal paths. This bias is not problematic for computing an accurate navigation solution, because it is common to all satellite signals. It is, however, important when estimating the ionospheric delay as a measurement, such as in an ionospheric monitoring system. Fig. 7 shows the truth and estimated differential code delays for one simulator run. The noisy curves are the auto-regressive averages of the differential code delay. Note that the initial 10 seconds contains a transient which is due to the estimator. Only four of nine satellites are shown to make the plot easier to view. The mean difference between the truth and the estimates is the receiver line bias. Two important points are shown in this figure. First, the receiver line bias is temperature-dependent and thus should be fairly constant over short time spans. Second, as already mentioned, the receiver line bias should be the same for each satellite. To the extent one can estimate it from this figure, the bias is in fact nearly the same for each satellite.

Fig. 8 shows the navigation solution residual error with respect to the simulator's truth position for a static receiver simulation. The figure shows the residual error from the software receiver running in two different modes. The first mode is as an L1-only receiver. The second mode is as an L1/L2 receiver. In the first mode, the L1-only receiver uses the broadcast ionospheric model to correct for the ionospheric delay. In the second mode, the receiver uses the broadcast ionospheric correction model for the first 250 secs shown in Fig. 8. At t=250 sec, the receiver switches from the using the broadcast ionospheric model to using the auto-regressive average of the appropriately-scaled differential code phase as an estimate of the ionospheric delay. At this time, an approximate 3-meter decrease in the residual error occurs, illustrating that the measured ionospheric delay is a more accurate estimate of the ionospheric delay than that of the broadcast iono-

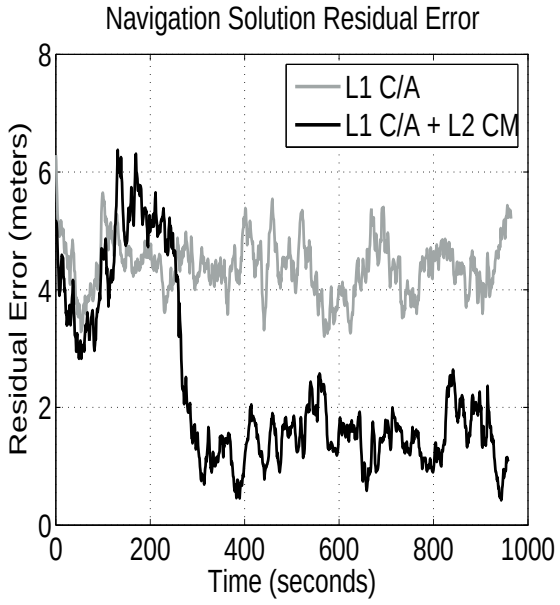

Fig. 8. Navigation solution residual error with respect to the truth position output by the simulator.

spheric model.

Fig. 8 shows after t=250 sec that the dual-frequency test run has a navigation solution residual error that is smaller than the single-frequency receiver. This improved dual-frequency residual error is what one would expect from a dual-frequency receiver with code-phase-based navigation. It was not possible to use the advanced ionospheric delay estimation method that utilizes the differential carrier phase advance to work for much longer than a few seconds. This may be a problem relating to the use of integrated carrier phase measurements produced from integrated-frequency estimates from an FLL as opposed to carrier phase measurements produced using a phase-locked loop (PLL).

IX. FUTURE WORK

Three significant improvements remain to be made to the current receiver. First, a more intelligent and faster signal acquisition method is needed for the CM/CL code. This may be either an FFT-based method or a traditional time-domain search method that makes better use of L1 C/A code tracking information and the navigation message.

Second, decoding has not been implemented for the CM code navigation data stream. It would be advantageous to add this feature to the software receiver. The primary benefit of this data stream is that error detection and correction capabilities are more robust than those present on the C/A code's navigation data stream.

Third, the ionospheric delay estimation method that uses both the differential code delay and the differential carrier phase advance needs to be tested and evaluated. Fixing this component of the receiver will likely reduce the navigation solution error and will certainly provide more precise estimates of the ionospheric delay.

X. SUMMARY AND CONCLUDING REMARKS

A 12-channel real-time GPS civilian L1/L2 software receiver that runs on a PC has been implemented and tested. The hardware consists of a two analog-mixing RF front ends, a data buffering and acquisition system, and a PC with a 3.4 GHz Intel Pentium 4 processor running RTAI. The software consists of a dual-frequency real-time software correlator and GPS software that provides the typical GPS functions such as signal acquisition, signal tracking, and navigation. The software correlator, running on the PC's processor, consumes about 81% of the CPU capacity. The navigation accuracy of this receiver is on the order of 1–3 m.

ACKNOWLEDGMENTS

This research was supported in part by ONR grant number N00014-92-J-1822.

REFERENCES

[1] D. M. Akos, P.-L. Normark, P. Enge, A. Hansson, and A. Rosenlind, "Real-Time GPS Software Radio Receiver," in *Proc. of the Institute of Navigation National Technical Meeting*, Long Beach, CA, January 22–24, 2001, pp. 809–816.

[2] J. Thor, P. Normark, and C. Ståhlberg, "A High-Performance Real-Time GNSS Software Receiver and its Role in Evaluating Various Commercial Front End ASICs," in *Proc. of the Institute of Navigation GPS*, Portland, OR, September 24–27, 2002, pp. 2554–2560.

[3] B. M. Ledvina, M. L. Psiaki, S. P. Powell, and P. M. Kintner, "A 12-Channel Real-Time GPS L1 Software Receiver," in *Proc. of the Institute of Navigation National Technical Meeting*, Anaheim, CA, January 22–24, 2003, pp. 767–782.

[4] B. M. Ledvina, M. L. Psiaki, S. P. Powell, and P. M. Kintner, "Bit-Wise Parallel Algorithms for Efficient Software Correlation Applied to a GPS Software Receiver," *IEEE Transactions on Wireless Communications*, 3(5) September, 2004.

[5] B. M. Ledvina, M. L. Psiaki, D. J. Sheinfeld, A. P. Cerruti, S. P. Powell, and P. M. Kintner, "A Real-Time GPS Civilian L1/L2 Software Receiver," in *Proc. of the Institute of Navigation GNSS 2004*, Long Beach, CA, September 21–24, 2004, pp. 986–1005.

[6] M. L. Psiaki, "Real-Time Generation of Bit-Wise Parallel Representations of Over-Sampled PRN Codes," *IEEE Transactions on Wireless Communications*, To Appear.

[7] B. M. Ledvina, F. Mota, and P. M. Kintner, "A Coming of Age for GPS: a RTLinux GPS Receiver," Workshop on Real Time Operating Systems and Applications, in *Proc. of the Workshop on Real Time Operating Systems and Applications and Second Real Time Linux Workshop* (in conjunction with IEEE RTSS 2000), Orlando, FL, Nov. 27–28, 2000.

[8] M. L. Psiaki, "Design and Practical Implementation of Multi-Frequency RF Front Ends Using Direct RF Sampling," *Proc. of the Institute of Navigation GPS*, Portland, OR, Sept. 9–12, 2003, pp. 90–102.

[9] Anon., "GP2015 Miniature GPS Receiver RF Front-End," Zarlink Semiconductor, Ottawa, Canada,

http://products.zarlink.com/product_profiles/GP2015.htm, 2005.
[10] eAnon., "Coaxial Frequency Mixers, ZX05-Series," Mini-Circuits, Brooklyn, New York, http://www.minicircuits.com/ZX05-SERIES.pdf, 2005.
[11] Anon., "ADF4360-8 - Integrated Integer-N Synthesizer and VCO - Output Frequency 65 to 400," Analog Devices, Norwood, Massachusetts, http://www.analog.com/en/prod/0,,770_850_ADF4360%252D8%2C00.html, 2005.
[12] Anon., "Low-Noise Amplifiers, 50Ω, Broadband, Linear 0.1 to 3000 MHz," Mini-Circuits, Brooklyn, New York, http://www.minicircuits.com/dg03-172.pdf, 2005.
[13] Anon., "JRC SAW Filter NSVS1108," Japan Radio Co. Ltd., Tokyo, Japan, http://www.jrc.co.jp/eng/product/saw/list/pdf/gps/NSVS1108.pdf, Oct. 2004.
[14] Anon., "Power Splitters/Combiners, 50 & 75Ω, 2 way-0° 2 kHz to 10 GHz," Mini-Circuits, Brooklyn, New York, http://www.minicircuits.com/dg03-120.pdf, 2005.
[15] Psiaki, M.L., "Block Acquisition of Weak GPS Signals in a Software Receiver," *Proc. Institute of Navigation GPS 2001*, Sept. 11–14, 2001, Salt Lake City, Utah, pp. 2838–2850.
[16] Psiaki, M.L., "FFT-Based Acquisition of GPS L2 Civilian CM and CL Signals," *Proc. Institute of Navigation GNSS 2004*, Sept. 21–24, 2004, Long Beach, CA, pp. 457–473.
[17] A. J. Van Dierendonck, "GPS Receivers," in *Global Positioning System: Theory and Applications, Vol. I*, B. W. Parkinson and J. J. Spilker Jr. , Eds., American Institute of Aeronautics and Astronautics, (Washington, 1996), pp. 329–407.
[18] J. A. Klobuchar, "Ionospheric Effects on GPS," in *Global Positioning System: Theory and Applications, Vol. I*, B. W. Parkinson and J. J. Spilker Jr. , Eds., American Institute of Aeronautics and Astronautics, (Washington, 1996), pp. 485–515.