A real-time software receiver that executes on a general purpose processor. The software receiver includes data acquisition and correlator modules that perform, in place of hardware correlation, baseband mixing and PRN code correlation using bit-wise parallelism.

26 Claims, 14 Drawing Sheets


* cited by examiner
FIG. 1

Signal Tracking, Data Demodulation, Navigation Solution, and User Interface

S/W Correlator w/ bit-wise parallel capability

MICROPROCESSOR 16

Data Buffering and Acquisition System

Mixer

Local Oscillator

RF FRONT END

Antenna
DATA BUFFERING AND ACQUISITION SYSTEM

FIG. 2A

SOFTWARE CORRELATOR

FIG. 2B
<table>
<thead>
<tr>
<th>Quantity</th>
<th>Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample Times</td>
<td>$t_0$ $t_1$ $t_2$ $t_3$ $t_4$ $t_5$ $t_6$ $t_7$</td>
</tr>
<tr>
<td>RF Signal</td>
<td>1 1 -1 -1 -1 1 1 1</td>
</tr>
<tr>
<td>Word Representation of Signal</td>
<td>1 1 0 0 0 1 1 1</td>
</tr>
<tr>
<td>PRN Code replica</td>
<td>1 f1 -1 -1 1 1</td>
</tr>
<tr>
<td>Word Representation of PRN Code replica</td>
<td>1 0 0 0 1 1 1 1</td>
</tr>
<tr>
<td>Product of Signal and PRN Code replica</td>
<td>1 -1 1 1 -1 1 1 1</td>
</tr>
<tr>
<td>Word Representation of Product</td>
<td>0 1 0 0 1 0 0 0</td>
</tr>
</tbody>
</table>

**FIG. 2C**
FIG. 2D
FIG. 3C PRIOR ART
REPRESENT SAMPLE SIGNAL DATA 21 FROM AT LEAST ONE CHANNEL AS SIGNAL SIGN 21A AND, IF PRESENT, SIGNAL MAGNITUDE 21B AND SELECT CARRIER REPLICA 25 BASED ON ITS FREQUENCY'S PROXIMITY TO A DESIRED CARRIER REPLICA FREQUENCY, REPRESENT CARRIER REPLICA 25 AS CARRIER REPLICA SIGN 25A AND CARRIER REPLICA MAGNITUDE 25B.

MIX SIGNAL DATA 21 TO BASEBAND \( \Rightarrow \) COMPUTE BASEBAND MIXED SIGN 23A = XOR (CARRIER REPLICA SIGN 25A, SIGNAL SIGN 21A), COMPUTE BASEBAND MIXED MAGNITUDE 23B/C = \( f \) (CARRIER REPLICA MAGNITUDE 25B, SIGNAL MAGNITUDE 21B).

SELECT PRN CODE FROM PRN CODE TABLE 28 OR COMPUTE IT USING REAL-TIME OVER-SAMPLED PRN CODE GENERATOR 30A; REPRESENT PROMPT PRN CODE 29 AS PROMPT SIGN 29A; REPRESENT EARLY-MINUS-LATE PRN CODE 35 AS EARLY-MINUS-LATE PRN SIGN 35A AND EARLY-MINUS-LATE PRN ZERO MASK 35B.


COMPUTE IN-PHASE AND QUADRATURE PROMPT INTEGRAND VALUE WORDS 27 AND EARLY-MINUS-LATE INTEGRAND VALUE WORDS 37: PROMPT INTEGRAND VALUE WORDS 27 = \( f \) (FULLY MIXED PROMPT INTEGRAND SIGN 31A, FULLY MIXED PROMPT INTEGRAND MAGNITUDE 31B/C), COMPUTE EARLY-MINUS-LATE INTEGRAND VALUE WORDS 37 = \( f \) (FULLY MIXED EARLY-MINUS-LATE INTEGRAND SIGN 33A, FULLY MIXED EARLY-MINUS-LATE INTEGRAND MAGNITUDE 33B/C, FULLY MIXED EARLY-MINUS-LATE INTEGRAND ZERO MASK 33D).
SUM OVER EACH INTEGRAND VALUE WORD 27/37 THE NUMBER OF ONE BITS (OR ZERO BITS) BY USING THE ONE BITS SUMMATION TABLE 38 OR A PROCESSOR COMMAND: COMPUTE PROMPT INTEGRAND VALUE-WORD ONE-BITS COUNTS = f (PROMPT INTEGRAND VALUE WORD 27), COMPUTE EARLY-MINUS-LATE INTEGRAND VALUE-WORD ONE-BITS COUNTS = f (EARLY-MINUS-LATE INTEGRAND VALUE WORD 37)

SUM OVER ACCUMULATION INTERVAL THE NUMBER OF ONE BITS (OR ZERO BITS) IN EACH PROMPT INTEGRAND VALUE WORD 27 AND EARLY-MINUS-LATE INTEGRAND VALUE WORD 37 TO PRODUCE ACCUMULATIONS 41 AND 49: COMPUTE PROMPT ACCUMULATIONS 41 = Σ (PROMPT VALUE-WORD ONE-BITS COUNTS), SUM IS OVER ALL WORDS IN ACCUMULATION INTERVAL; COMPUTE EARLY-MINUS-LATE ACCUMULATIONS 49 = Σ (EARLY-MINUS-LATE INTEGRAND VALUE WORD ONE-BITS COUNTS), SUM IS OVER ALL WORDS IN ACCUMULATION INTERVAL

MULTIPLY VALUE WORD ONES ACCUMULATIONS 41 AND 49 BY CORRESPONDING VALUES 41A AND SUM OVER ALL VALUE WORDS FOR IN-PHASE AND QUADRATURE PROMPT WORDS AND EARLY-MINUS-LATE WORDS: SUMMED PROMPT ACCUMULATION 45 = Σ (VALUE 41A*[PROMPT ACCUMULATION 41]), SUM IS OVER ALL VALUES; SUMMED EARLY-MINUS-LATE ACCUMULATION 47 = Σ (VALUE 41A*[EARLY-MINUS-LATE ACCUMULATION 49]), SUM IS OVER ALL VALUES, RESULTS ARE IN-PHASE AND QUADRATURE SUMMED PROMPT ACCUMULATION 45 AND SUMMED EARLY-MINUS-LATE ACCUMULATION 47

ROTATE THE IN-PHASE AND QUADRATURE PROMPT ACCUMULATIONS 45 AND EARLY-MINUS-LATE ACCUMULATIONS 47

FIG. 4B
<table>
<thead>
<tr>
<th>Table Element</th>
<th>Code Time Offset</th>
<th>Bit Sequence of L Code Chips (first is left-most, last is right-most)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x(1)$</td>
<td>$\Delta t_{0\text{min}}$</td>
<td>0 0 ... 0 0 0 0</td>
</tr>
<tr>
<td>$x(2)$</td>
<td>$\Delta t_{0\text{min}}$</td>
<td>0 0 ... 0 0 0 1</td>
</tr>
<tr>
<td>$x(3)$</td>
<td>$\Delta t_{0\text{min}}$</td>
<td>0 0 ... 0 0 0 1 0</td>
</tr>
<tr>
<td>$x(4)$</td>
<td>$\Delta t_{0\text{min}}$</td>
<td>0 0 ... 0 0 0 1 1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$x(2^L)$</td>
<td>$\Delta t_{0\text{min}}$</td>
<td>1 1 ... 1 1 1 1</td>
</tr>
<tr>
<td>$x(2^L+1)$</td>
<td>$\Delta t_{0\text{kmin}+1}$</td>
<td>0 0 ... 0 0 0 0</td>
</tr>
<tr>
<td>$x(2^L+2)$</td>
<td>$\Delta t_{0\text{kmin}+1}$</td>
<td>0 0 ... 0 0 0 1 0</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$x(2^L \times k_{IOT})$</td>
<td>$\Delta t_{0\text{kmax}}$</td>
<td>1 1 ... 1 1 1 1</td>
</tr>
</tbody>
</table>

**FIG. 5**
FIG. 6
COMPUTE TABLE OF CANDIDATE INTEGERS NECESSARY TO COMPUTE AN ARRAY INDEX (EQUATION (36)) BY ITERATING EQUATIONS (44a)-(44c) 201

COMPUTE DATA WORD MIDPOINT INDEX $k_{mfmid}$, CODE DOPPLER CORRECTION CONSTANTS $a_{bs}$ AND $b_{bs}$, NOMINAL CHIPS PER DATA WORD $L_{nn}$, NUMBER OF FINE TIME INCREMENTS PER CHIP $A_{fs}$, NOMINAL CHANGE IN FINE TIME OFFSET $A_{fpp}$, AND MINIMUM AND MAXIMUM FINE TIME OFFSET $k_{fmin}$ and $k_{fmax}$ USING EQUATIONS 48a-48e AND 50a-50e THESE ARE USED TO DETERMINE THE INDEX INTO PROMPT SIGN, EML ZERO MASK, AND EML SIGN TABLES 203

INITIALIZE TIME OFFSET INDEX $kp$ AND AUXILIARY TABLE INDEX $mu$ BY EVALUATING EQUATIONS 51a-52b 205

COMPUTE $k_{fmom}$, $mu_{mom}$, $eta$, $k_{fs}$, $mu$, $k$, and $i$, FOR $v = 1, ..., N$ BY ITERATING EQUATIONS 53a, 53b, 57a-c, 49, 55, 45 TO COMPUTE INDICES INTO PROMPT SIGN, EARLY-MINUS-LATE ZERO MASK, AND EARLY-MINUS-LATE SIGN TABLES 207

COMPUTE $x_{ps}$, $x_{emls}$, AND $x_{emlsz}$ FOR $v = 1, ..., N$ BY EVALUATING EQUATIONS 56a-c TO COMPUTE THE PROMPT SIGN, EARLY-MINUS-LATE ZERO MASK, AND EARLY-MINUS-LATE SIGN DATA WORDS 209

END

FIG. 7
FIG. 8
REAL-TIME SOFTWARE RECEIVER

CROSS REFERENCE TO RELATED APPLICATIONS

The present application is a divisional application of U.S. patent application Ser. No. 10/753,927, filed Jan. 8, 2004 now U.S. Pat. No. 7,010,060. The present application claims priority to U.S. Provisional Application No. 60/439,391 filed Jan. 10, 2003 entitled REAL-TIME SOFTWARE RECEIVER which is incorporated herein in its entirety by reference.

STATEMENT OF GOVERNMENT INTEREST

This invention was made with United States Government support from the Office of Naval Research (ONR) under contract number N00014-02-J-1822 and from the National Aeronautics and Space Administration (NASA) under contract numbers NCC5-563, NAG5-11819, and NAG5-12089. The United States Government has certain rights in the invention.

BACKGROUND OF THE INVENTION

This invention relates generally to software radio receivers, and more specifically to a software receiver for positioning systems.

A typical positioning system receiver, such as is used in the Global Positioning System (GPS), includes an antenna, a radio frequency (RF) section, a correlator, a signal tracking and demodulation component, and a component to compute the navigation solution. The antenna, which is possibly followed by a pre-amplifier, receives L-band GPS signals. The RF section filters and down converts the GHz GPS signal to an intermediate frequency in the MHz range. The RF section also digitizes the signal. The correlator separates the down-converted signal into different channels (ten or more in modern receivers) allocated to each satellite. For each satellite, the correlator mixes the Doppler-shifted intermediate frequency signal to baseband by correlating it with a local copy of the carrier replica signal and it distinguishes the particular satellite by correlating the signal with a pseudo-random number (PRN) code. Software routines cause the carrier replica and PRN replica signals to track the actual received signal, extract the navigation message, and compute the navigation solution.

Baseband mixing is a multiplication of an input signal by a complex exponential where the frequency of the complex exponential approximately matches that of the input signal. The resultant signal is centered at baseband. A complex signal can be broken down into cosine and sine signal components, resulting in separate in-phase and quadrature components. The frequency of the baseband mixed signal must be controllable to within a few millihertz in the case of a phase-locked loop for use in a precision navigation system, and the baseband mixed signal must have a continuously varying phase. In a hardware correlator, local oscillators generate cosine and sine signal components that have precise frequency control and a continuous phase. Generating cosine and sine signal components on the fly with the correct frequency and phase is too time consuming to be feasible for a software correlator. Instead, the software correlator generates cosine and sine signal components on a grid of frequencies offset. These signal components must be stored on a time grid of points sampled at the RF front-end sampling frequency, for example, at 5.714 MHz for one particular RF front-end hardware configuration, and the signals last for a typical accumulation period, e.g., for a 0.001 second coarse/acquisition (C/A) PRN code period when working with GPS L1 civilian signals. It takes tens of gigabytes of memory or more in order to brute-force store all frequencies on a one mHz grid ranging from -10 KHz to +10 KHz, which is the needed frequency range when tracking GPS satellites from a terrestrial receiver, and additional storage is required to store a grid of possible starting phases at each frequency point.

PRN code mixing is a multiplication of a baseband mixed signal by a prompt +1/-1 PRN code or by a ±2/0 PRN code, where the code timing and frequency approximately match that of the input signal. The resultant signal is a constant in the case of prompt PRN code mixing, and an approximately linear function of the code timing error in the case of early-minus-late mixing. A receiver accumulates both of these correlation outputs. The magnitude of the prompt accumulation indicates signal strength and whether a signal has been detected, and its in-phase (real) and quadrature (imaginary) components are used to measure carrier phase and Doppler shift. The magnitude of the early-minus-late accumulation measures the code timing error; it will be zero when the timing error is zero.

The code phase of the baseband mixing signal must be controllable to within a percent or less of a PRN code chip for use in a precision navigation system. In a hardware correlator, local oscillators generate the prompt and early-minus-late PRN code replicas. A software correlator can either compute and store PRN code replicas, or compute them in real-time.

The current Global Positioning System is slated to realize expanded capabilities that include new civilian codes on the L2 frequency, a new L5 frequency, and new codes (M-code, CL and CM codes) on the L2 frequency. Some of these upgrades are slated to start within one to three years. A hardware correlator requires hardware modifications 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 increased performance that will accrue only very slowly as new GPS satellites replace older models. One way to avoid the complex trade-off is to use a software receiver that can receive and process new signals without the need for a new correlator chip set.

A software receiver is flexible because its software components can be easily modified. One application of a software receiver is to merge together numerous devices that use wireless digital communication protocols to form a single device. For example, a cell phone, GPS receiver, and Personal Data Assistant (PDA) could become a single device that plays the role of all three. Another use of a software receiver is to shorten development and to-market times for new wireless devices. For example, as new frequencies and codes are added to GPS, a software receiver having a software correlator simply needs to be reprogrammed, while a hardware approach would require a brand new correlator chip design. New PRN codes can be used simply by making software changes. Thus, software receiver technology lessens the risks involved for designers during the period of transition to the new signals. Furthermore, a software receiver could be reprogrammed to use the Galileo system (European GPS) or GLONASS (Russian GPS).

In the recent past, GPS software receivers have been developed that either post-process stored signals or operate in real-time. Previous real-time software receivers function
with a limited number of channels (4-6) or require high-end computer speeds or digital signal processor (DSP) chips such as are disclosed in Real-Time GPS Software Radio Receiver, Akos et al., ION NTM 2001, 22-24 Jan. 2001, Long Beach, Calif., pp. 809-816 (Akos 2001a), and Global Positioning System Software Receiver (gpsRxs) Implementation in Low Cost/Power Programmable Processors, Akos et al., ION GPS 2001, 11-14 Sep. 2001, Salt Lake City, Utah, pp. 2851-2858, both incorporated herein in their entireties by reference.

Therefore, it is an object of the present invention to create a software receiver that operates in real-time and is not restricted to a severely limited number of channels or to a very fast processor.

Another object of the present invention is to minimize the number of sine and cosine signal components that must be stored.

A further object of the present invention is to process incoming signals through bit-wise parallelism.

A still further object of the present invention is to process oversampled signals by use of bit-wise parallelism.

A still further object of the present invention is to use very long oversampled PRN codes efficiently in a bit-wise parallel software receiver.

SUMMARY OF THE INVENTION

The objects set forth above as well as further and other objects are addressed by the present invention. The solutions and advantages of the present invention are achieved by the illustrative embodiment described herein below.

The software receiver system and method of the present invention enable the efficient execution of a set of algorithms, that perform software correlation on data sampled from incoming channels, on a general purpose processor. The system and method of the present invention provide for either PRN code storage or computation of PRN codes in real-time. PRN code storage is appropriate for PRN codes that have short periods, such as the GPS coarse acquisition codes, which are 1023 chips long. In this case, the system and method of the present invention pre-compute oversampled replicas of entire PRN code periods and store them for orderly and efficient retrieval, such as in a table. This table can include a selection of code start times as measured relative to the sample times at which RF data are available from the receiver front end. There is a separate table for each unique PRN code.

The system and method of the present invention can also generate oversampled versions of the prompt and early-minus-late PRN codes in real-time through use of an oversampling function described herein. The values of the oversampling function can be located in a specially designed table that can be generic across PRN codes. The length of the specially designed table can be independent of the length of the PRN code whose replica is being used to process a given received signal. The system and method of the present invention include techniques for efficiently calculating indices into the specially designed table that enable rapid, real-time table look-up.

The system of the present invention includes a software correlator that can mix the received signal to baseband, compute baseband/PRN correlations through bit-wise parallelism and look-up tables using either the tabulated or real-time-generated PRN codes, and compute accumulations through bit-wise parallelism and processor instructions or look-up tables. Bit-wise parallelism allows the processing of multiple data samples simultaneously as the multiple bits of a given word of computer data. For example, for 32-bit words, the software correlator can process up to 32 samples at a time. Bit-wise parallelism can optimally operate when each signal in question can be represented by only a few bits, which is normally the case in RF digital signal processing of navigation signals.

The bit-wise parallel operations of the present invention can save computation time in comparison to integer mathematical correlation operations. If, for example, four accumulations are required per sample, integer mathematics requires six multiplications and four additions per sample (except for the last sample). At a sampling rate of, for example, 5.714 MHz this translates into 57,140 integer operations per PRN code period. In the illustrative embodiment, 33,500 bit-wise parallel operations are necessary per PRN code period when the RF signal has a 2-bit representation. This operation count is further reduced to approximately 16,750 bit-wise parallel operations per PRN code period when the RF signal has a 1-bit representation. Thus, there can be a savings of almost a factor of two to almost a factor of four in the operation count.

The system and method of the present invention also include a table of pre-computed baseband mixing sine waves, algorithms that can produce correlation accumulation outputs that are equivalent to what would be produced by a continuously variable sine wave, and a method of use of the table and algorithms. Thus, in the present invention, a relatively small set of sine wave values need to be pre-computed and saved, which can conserve computer memory and processing time.

The present invention also includes a system and method for tracking the phase of PRN code replicas in software in order to track the timing of any given "chip" of the PRN code replica as measured with respect to a pre-specified set of sample times at which the basic raw data comes out of the RF front end (a chip is an element of a PRN code). The PRN code phase is kept track of via a variable for each channel, that indicates the PRN code start time with respect to the RF sample times. The system and method of the present invention allow for the synchronization of the measurements of PRN code phase, carrier phase, and carrier frequency for each satellite relative to these sample times.

The method for tracking the phase of each PRN code replica and the phase of each carrier replica includes the steps of latching all the C/A code phases, carrier phases, epoch counters, and carrier frequencies for each satellite at a pre-specified time, and computing the pseudo range to each satellite using the C/A code phase and epoch counters.

The method also includes the step of tracking and updating code and carrier phases by estimating code chiping rate and carrier Doppler shift inputs. The method further includes the step of computing the code phase at the pre-specified time for each satellite as a function of the updated code chiping rate and the pre-specified time. The method further includes the step of computing the carrier phase at the pre-specified time as a function of the updated carrier phase, the Doppler shift, and the pre-specified time. The timing of the PRN code phase (or chip location) is the most fundamental of GPS measurements for use in navigation data processing. The monitoring of these times in software allows complete control of the precision with which they can be measured, and it allows precise synchronization of these times with the measurement times of data from other sensors, such as inertial measurement units. This feature gives an enhanced ability to develop what are known as deeply coupled systems that must fuse GPS data with data from other types of sensor systems.
The software correlator of the present invention can advantageous be easily adapted to accept signals at any frequency, new PRN codes, or even signals for different types of devices. Thus, the same processing hardware could use the software correlator to implement such devices as a GPS receiver, a cell phone, or both. To allow for new codes, new frequencies, and new types of functionality, small changes can be made in the software correlator, or different versions of the software correlator can be run on the same processor. Hardware-correlator-based receivers of the prior art can deal only with frequencies and PRN codes that are hard-wired into their designs. Also, the system and method of the present invention could be implemented within systems such as GLONASS receivers, cell phones and cell base stations, pagers, wireless Ethernet (e.g. 802.11x standards), Bluetooth®, Blackberry® wireless internet devices, and satellite radio/phones (e.g. INMARSAT®). In fact, the system and method of the present invention are applicable to any sort of telecommunication system/device that uses spread spectrum, code division multiple access (CDMA) PRN codes for the transmission of information, either wired or wireless.

For a better understanding of the present invention, together with other and further objects thereof, reference is made to the accompanying drawings and detailed description. The scope of the present invention is pointed out in the appended claims.

DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWING

FIG. 1 is a schematic block diagram of the hardware environment of a typical software receiver;

FIGS. 2A and 2B are schematic diagrams of bit-wise mappings of signal and carrier replica sign and magnitude bits to computer data words;

FIG. 2C is a graphic representation of a plot of sections of prompt, early, late, and early-minus-late PRN code signals and 16-bit word representations of their over-sampled equivalents;

FIGS. 3A and 3B are data flow diagrams illustrating the bit-wise parallelism process (replicated twice, once for the in-phase carrier replica and once for the quadrature carrier replica) of the present invention;

FIG. 3C is a graphic representation of a plot of a prior art optimal 2-bit representation of a sine wave presented to enhance the reader’s understanding of the present invention;

FIGS. 4A and 4B are flowcharts of the method for computing correlation accumulations through bit-wise parallel computations of the present invention;

FIG. 5 is a schematic diagram of a look-up table layout as a function of code time offset and chip bit pattern;

FIG. 6 is a graphic representation of a plot illustrating the timing relationship between data sample words and the sequence of prompt code chips that defines an accumulation interval;

FIG. 7 is a flowchart of the method for computing bit-wise parallel representations of the over-sampled prompt PRN code replica and the over-sampled early-minus-late PRN code replica for an entire accumulation interval using the real-time over-sampled PRN code generation algorithm.

FIG. 8 is a graphic representation of a plot that illustrates the location in time at which the code phase of each signal is computed; and

FIGS. 9A and 9B are graphic representations of plots of correlations of the true sampled code with prompt (FIG. 9A) and early-minus-late (FIG. 9B) versions of the true and table look-up codes, the latter being generated by the new real-time over-sampled PRN code generator.

DETAILED DESCRIPTION OF THE INVENTION

The present invention is now described more fully hereinafter with reference to the accompanying drawings, in which the illustrative embodiment of the present invention is shown. The following configuration description is presented for illustrative purposes only. Any computer configuration satisfying the speed and interface requirements herein described may be suitable for implementing the system of the present invention. The equations herein are stated in general terms, but have parameters that are specific to the GPS L1 C/A signal for illustrative purposes only. For example, the 0.001 sec. accumulation interval seen in many of the equations is the nominal C/A code period. Also, the C/A PRN code of the illustrative embodiment can be replaced by the PRN code of any other CDMA signaling system.

By way of introductory explanation, RF signal processing equations and terms are herein provided. The time-domain L1 C/A signal received from, for example, a satellite, is represented by:

\[
y(t) = \sum_{j} A_j D_k C \left[0.001 \left( \frac{t - \tau_k}{T_{k+1} - \tau_k} \right) \right] \cos[\omega_1(t) - \phi_1(t)] + n_j
\]  

where \( t \) is the sample time, \( A_j \) is the amplitude, \( D_k \) is the navigation data bit, \( C_j[t] \) is the C/A code, \( \tau_k \) and \( \tau_{k+1} \) are the start times of the received \( k \)th and \( k+1 \)st C/A code periods, \( \omega_1 \) is the intermediate frequency corresponding to the L1 carrier frequency, \( \phi_1(t) \) is the carrier phase perturbation due to accumulated delta range, \( n_j \) is the receiver noise, and the subscript \( j \) refers to a particular GPS satellite. The summation over all visible GPS satellites. The negative sign in front of \( \phi_1(t) \) comes from the high-side mixing that occurs in the RF front-end that has been used in the illustrative embodiment. The signal in equation (1) is the output of a typical RF front-end.

A GPS receiver works with correlations between the received signal and a replica of it. The correlations are used to acquire and track the signal. The replica is composed of two parts, the carrier replica and the C/A PRN code replica. Two carrier replica signals are used, an in-phase signal and a quadrature signal. When mixed with the received signal from the RF front-end they form the in-phase and quadrature baseband mixed signals represented by:

\[
y_{lg}(t) = C \left[0.001 \left( \frac{t - \tau_k}{T_{k+1} - \tau_k} \right) \right] \cos[\omega_{IF}(t) - \phi_{IF}(t)]
\]  

where \( \omega_{IF} \) is the intermediate frequency corresponding to the L1 carrier frequency, \( \phi_{IF}(t) \) is the carrier phase perturbation due to accumulated delta range, and \( \tau_{IF} \) is the receiver’s estimate of the start time of the \( k \)th C/A code period. In these equations \( \tau_k \) and \( \tau_{k+1} \) are the receiver’s estimates of the start times of the \( k \)th and \( k+1 \)st code periods,
\( \hat{\phi}_k \) is the estimated carrier phase at time \( t_k \), and \( \omega_{\text{Doppler}} \) is the estimated carrier Doppler shift during the \( k \)th code period.

A typical receiver computes the estimates \( t_k \), \( t_{k+1} \), \( \hat{\phi}_k \), and \( \omega_{\text{Doppler}} \) by various conventional means that are described in GPS Receivers, A. J. Van Dierendonck, Global Positioning System: Theory and Applications, 1996, Chapter 8, pp. 329-406 (Dierendonck), incorporated herein in its entirety by reference. These include open-loop acquisition methods and closed-loop signal tracking methods such as a delay-locked loop to compute \( \hat{\phi}_k \) and \( t_{k+1} \) and a phase-locked loop or a frequency-locked loop to compute \( \omega_{\text{Doppler}} \). The software receiver developed herein uses conventional techniques for forming these estimates.

Both prompt and early-minus-late correlations are needed to track the carrier frequency, carrier phase, and code phase in a GPS receiver. A typical receiver uses the PRN code and carrier replicas to compute the following in-phase and quadrature correlation accumulations:

\[
I_k(\Delta) = \sum_{i=0}^{N_s} y(i) \cos\left[\omega_c t_i + \hat{\phi}_k + \omega_{\text{Doppler}}(t_i - t_k)\right]
\]

\[4\]

\[
Q_k(\Delta) = \sum_{i=0}^{N_s} y(i) \sin\left[\omega_c t_i + \hat{\phi}_k + \omega_{\text{Doppler}}(t_i - t_k)\right]
\]

\[5\]

where \( i_k \) is the index of the first RF front-end sample time that obeys \( \tau_{i_k} \leq t_k \) and \( N_s+1 \) is the total number of samples that obey \( \tau_{i_k} \leq t_k < \tau_{i_k+1} \). The time offset \( \Delta \) causes the replica PRN code to play back early if \( \Delta \) is positive and late if \( \Delta \) is negative. The prompt correlations are defined by equations (4) and (5) with \( \Delta = 0 \). The early-minus-late correlations are \( I_k(\Delta_{\text{len}}) - I_k(\Delta) \) and \( Q_k(\Delta_{\text{len}}) - Q_k(\Delta) \) where \( \Delta_{\text{len}} \) is the spacing between the early and late PRN carrier replicas. The present invention described herein is an efficient technique for the receiver to accumulate \( I_k \) and \( Q_k \) in software.

Referring now to FIG. 1, the operational platform of the software receiver 10 of the present invention includes an antenna 11, conventional RF front-end 13, a data acquisition (DAQ) system 17, a microprocessor 16, a software correlator 19, and application-specific code 15. Conventional RF front-end 13 interfaces with antenna 11 and with (DAQ) system 17. DAQ system 17 includes a system of shift registers and a data buffer. Microprocessor 16 executes software correlator 19, which includes a set of specially developed bit-wise parallel algorithms, and application-specific code 15, such as the GPS navigation and tracking functions. In the illustrative embodiment, conventional GPS software functions (signal tracking, data extraction, navigation solution, etc.) are provided by the MITEL®/GPSArchitect software ported to RT LINUX® (see A Coming of Age for GPS: A RT LINUX® BASED GPS RECEIVER, Ledvina et al., Proceedings of the Workshop on Real Time Operating Systems and Applications and Second Real Time

Linus Workshop (in conjunction with IEEE RTSS 2000) Nov. 27-28, 2000, but can be provided by any equivalent configuration.

Continuing to refer to FIG. 1, conventional RF front-end 13 can, for example, be a MITEL® GP2015 RF front-end, which down converts the nominal 1.57542 GHz GPS signal 12 to an intermediate frequency of (88.54/63)x10^6 Hz-1.4053968254 MHz and then performs analog-to-digital conversion. The resultant, digitized signal data 21 has a predetermined number of bits/sample, such as two binary bits/sample, a sign bit and a magnitude bit, or one bit/sample. The shift registers in the DAQ system 17 parallelize the magnitude and sign data bit streams into separate words, which the DAQ system 17 reads into the memory of microprocessor 16 using DMA. To make the process of reading data into the microprocessor 16 more efficient to prepare for efficient correlation calculations, DAQ system 17 can read a pre-specified number of bits of buffered samples, such as thirty-two bits, at a time. The exemplary thirty-two bits include sixteen sign bits and sixteen magnitude bits.

Referring now to FIGS. 1 and 2A, the shift registers in DAQ system 17 (FIG. 1) buffer signal data 21 (FIGS. 1 and 2A) and pack signal sign 21A (FIG. 2A) and signal magnitude 21B (FIG. 2A) into separate words, that represent the integer values \( \pm 1 \) and \( \pm 3 \) as is shown in Table 1. In the case of a 1-bit signal, the bit stream representing the samples is packed into successive words to prepare the signal for bit-wise parallel processing. DAQ system 17 also provides for accurate timing by, for example, synchronizing signal sign 21A and signal magnitude 21B to a (40/7)x10^6 Hz-5.714 MHz clock signal, which can be, for example, a third output from conventional RF front-end 13 (FIG. 1). DAQ system 17 can convert the 5.71424 MHz clock signal down to 357.14 KHz by use of, for example, a divide-by-16 counter for a 16-bit word, which can provide a signal indicating when the buffer is full. DAQ system 17 can use any method for providing a buffer full indication.

<table>
<thead>
<tr>
<th>Signal Sign 21A</th>
<th>Signal Magnitude 21B</th>
<th>RF Signal Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>-3</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>+1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>+3</td>
</tr>
</tbody>
</table>

With further reference to FIG. 1, in the illustrative embodiment, the DAQ system 17 can consist of an interface card and driver software that can be compatible with, for example, a 1.73 GHz AMD ATHLON® processor running RT LINUX®, but could be compatible with any operating system and any processor that can accommodate real-time operations. The interface card can, for example, be a NATIONAL INSTRUMENTS® PCI-DIO-32HS digital I/O card. Pertinent features of this card are the thirty-two digital input lines, DMA, and availability of a driver for RT LINUX®, perhaps gotten from the suite of open source drivers and application interface software for interface cards known as COMEDI (Control and MEasurement and Device Interface). Modifications to the conventional COMEDI driver for the PCI-DIO-32HS card include increasing the
number of input bits from sixteen to thirty-two, enabling DMA, and modifying the driver to support continuous interrupt-driven acquisition.

With still further reference to FIG. 1, microprocessor 16 can be, for example, a 1.73 GHz AMD ATHLON® processor running the RTLinux® operating system, but any operating system and processor that can accommodate real-time operations can be used. Low latency interrupt responsiveness, the ability to execute threads at regular intervals, with the kernel having a possibility of being the lowest priority thread, and reliable execution of time-critical code are among features of an operating system that could enhance the performance of the system of the present invention. The use of RTLinux® is presented herein for illustrative purposes only.

Continuing to refer to FIG. 1, analogous to a hardware correlator that takes input directly from the RF front end in serial fashion, software correlator 19 reads from a shared memory buffer that both software correlator 19 and DAQ system 17 can access, the former to read data, and the latter to write data. The shared memory buffer can be implemented as a DMA memory space and a circular buffer. In the illustrative embodiment in which the system and method of the present invention are used in a GPS (or similar) environment, microprocessor 16 can store the most recent twenty-one milliseconds of signal data 21 (FIGS. 1 and 2A) in the circular buffer, but could store more or less. The present invention does not fix the size of the circular buffer, nor the amount of RF data that can be stored there. The circular buffer allows the processing of code periods that start and stop at different times for different satellites during different iterations of a regularly scheduled program thread. DMA memory space can be written to directly by DAQ system 17 using a DAQ software driver, which fills the circular buffer. Communication between software correlator 19 and application-specific code 15 can be performed using operating system-provided shared memory capability. For example, the mbuff driver, included with RTLinux®, can be used to create and manage this shared memory space. Any memory management system that accommodates real-time processing can be used. If the mbuff driver is used, kernel modules can share memory and the kernel can be restricted from swapping the shared memory space to long-term storage.

Continuing with the analogy to hardware correlation, and still referring primarily to FIG. 1, in hardware correlation, the correlator receives frequency and phase information from tracking and acquisition loops that are part of application code, and Numerically Controlled Oscillators (NCOs) generate signals that correspond to the written frequencies and phases. In contrast, software correlator 19 includes simulated carrier and code NCOs that receive their frequency commands from application-specific code 15. Software correlator 19 uses these frequency commands to reconstruct carrier replica signal 25 (FIG. 3A) and prompt PRN code 29 and early-minus-late PRN code 35 (FIG. 3A) which mixes with the signal data 21 (FIG. 2A) resulting in fully mixed prompt integrand 31 and fully mixed early-minus-late integrand 33 (FIG. 3A).

To further continue the analogy, a hardware correlator generates in real-time a particular C/A code replica at the correct Doppler shifted frequency and phase. In contrast, software correlator 19 can generate C/A codes of line and store them in a memory table, the pre-computed over-sampled PRN code table 28 (FIG. 3A). The pre-computed over-sampled PRN code table 28 is used to select PRN codes with the correct timing relationship to the sample times of signal data 21 (FIG. 3A). The codes are then used to form correlations with baseband mixed signals 23 (FIG. 3A), the result from which is summed to produce the standard in-phase and quadrature, summed prompt accumulation 45 (FIG. 3B) and summed early-minus-late accumulation 47 (FIG. 3B) that are equivalent to what would be produced by a continuously variable sine wave. These are provided to application-specific code 15, such as conventional GPS software that executes signal tracking and navigation functions. In a second approach, software correlator 19 can generate the PRN carrier replicas on-line at the code chipping rate and use tabulated functions to re-sample the code at the sample rate of the RF front-end for purposes of calculating accumulations. Real-time over-sampled PRN code generator 30A (FIG. 3A) is used in place of pre-computed over-sampled PRN code table 28 (FIG. 3A) in this latter approach. This latter method can be used with longer PRN codes, such as the new civilian GPS L2 C1 codes.

With still further reference to FIG. 1, since the received L1 raw signal 12 can have an uncertain carrier phase, software correlator 19 computes both in-phase (I) and quadrature (Q) accumulations, as defined in equations (4) and (5). Software correlator 19 begins the accumulation process by using carrier replica signal 25 (FIG. 3A), which it gets from pre-stored carrier replica table 30 (FIG. 3A). The carrier replicas in this table fall on a rough frequency grid, and they all start with a particular phase, for example a phase of zero. The baseband mixing process involves selecting a carrier replica signal 25 (FIG. 3A) from carrier replica table 30 (FIG. 3A) that is at the frequency that is as close to "ideal" as possible. In the case of a 175 Hz grid spacing, the baseband mixing process selects a signal that is maximally within ±87.5 Hz of the ideal signal. The rough frequency grid can have a spacing of, for example, 175 Hz but could be larger or smaller depending on (a) the frequency range needed to cover, for example, ±10 KHz, (b) the amount of space available for storing pre-computed signals, and (c) other design decisions. The pre-computed signals in carrier replica table 30 (FIG. 3A) each may occupy 180 32-bit words in order to be guaranteed to cover the full 5,714 RF front-end samples that occur in one PRN code period for any possible code period start time within the thirty-two samples of the initial word. Thus, 180*4=720 bytes could be required for each bit of each pre-computed carrier replica signal 25 that is stored in the table. The sine and cosine waves of carrier replica signals 25 (FIG. 3A) each have 2-bit representations, which translates into a storage requirement of 2880 bytes for the carrier replica signals 25 at a given Doppler shift. There are 115 Doppler shifts that may be stored in order to cover the -10 KHz to +10 KHz range with a 175 Hz grid spacing. This translates into 323 Kbytes of storage for all of the carrier replica signals 25. This approach avoids the need to pre-compute sine waves with a prohibitively large number of possible frequencies and phase offsets and it avoids the need to compute sine waves in real-time. Instead, the errors created by using pre-defined sine wave replicas are compensated for by post-processing calculations, as described below.

In any case, and continuing to refer to FIG. 1, the resulting accumulations are

$$I_{nk}(\Delta) = \sum_{i=1}^{N_k} y[i] \cdot C[i] \cdot \left[ \cos(\omega_{nk} \cdot t - \phi_{nk}) \right]$$

where $t$ is the time index, $N_k$ is the number of samples in the period, $y[i]$ is the received signal, $C[i]$ is the code value, and $\omega_{nk}$ and $\phi_{nk}$ are the carrier and phase offsets for the $nk$-th code period.
terms in equations (6) and (7). The other trigonometric terms which this carrier replica signal 25 (FIG. 3A) has zero
this patent disclosure covers all such techniques.

Note that equations (9), and (10) are an generation can entail an additional computational cost. Both
where $\omega_{grid}$ is the grid frequency that is closest to the estimated frequency $\omega_{grid}$ and where $t_{grid}$ is the time at
which this carrier replica signal 25 (FIG. 3A) has zero
carrier phase, Software correlator 19 rotates these accumu-
ations in order to create accurate approximations of what
would have been computed had the estimated carrier phase
time history in equations (4) and (5) been used: Referring now to FIGS. 2A, 2B, and 3A, PRN codes

where $\Delta \phi_{avg}$ is the average phase difference between the grid carrier phase and the estimated carrier phase averaged
over the accumulation interval is zero. 1).

$\Delta \phi_{avg} = \frac{1}{2} \phi_{grid} - \phi_{grid} \frac{t_{grid} + t_{grid}}{2} + \omega_{grid} t_{grid}$

Note that equations (8), (9), and (10) are an illustrative example of how software correlator (19) can rotate its l and
Q accumulations in order to correct for phase and frequency
errors in its table of pre-computed carrier replica signals.

Continuing to refer primarily to FIGS. 2A, 2B, and 3A, in
order to perform bit-wise parallel operations, software cor-
relator 19 (FIG. 1) stores pre-computed carrier replica sign
25A (FIGS. 2B and 3A) and carrier replica magnitude 25B
(FIG. 3A) in data words. Simple representations of
signal data 21 (FIGS. 2A and 3A) and carrier replica signal
25A (FIGS. 2B and 3A) and carrier replica signal 25B (FIGS. 2B and 3A) in terms of one, two, or more bits are
suitable for using bit-wise parallelism to perform the calcula-
tions described herein. Bit-wise parallel operations work
with representations of the data that store successive samples in successive bits of a word. For example, thirty-
two samples (bits) of the RF front-end output are stored in
two N-32-bit words, signal sign 21A (FIGS. 2A and 3A) and
signal magnitude 21B (FIGS. 2A and 3A), or simply a single
32-bit word if signal data 21 consists of a single data bit.

Carrier replica sign 25A (FIGS. 2B and 3A) and carrier replica magnitude 25B (FIGS. 2B and 3A) are stored, for
example in tables, in separate words, with each 32-bit word
storing thirty-two sign or magnitude bits that tabulate to thirty-two successive samples of the corresponding cosine or
sine wave.

Continuing to refer primarily to FIGS. 2A, 2B, and 3A, in
to perform bit-wise parallel operations, software correla-
tor 19 (FIG. 1) stores pre-computed carrier replica sign
25A (FIGS. 2B and 3A) and carrier replica magnitude 25B
(FIGS. 2B and 3A) in data words. Simple representations of
signal data 21 (FIGS. 2A and 3A) and carrier replica signal
25A (FIGS. 2B and 3A) and carrier replica signal 25B (FIGS. 2B and 3A) in terms of one, two, or more bits are
suitable for using bit-wise parallelism to perform the calcula-
tions described herein. Bit-wise parallel operations work
with representations of the data that store successive samples in successive bits of a word. For example, thirty-
two samples (bits) of the RF front-end output are stored in
two N-32-bit words, signal sign 21A (FIGS. 2A and 3A) and
signal magnitude 21B (FIGS. 2A and 3A), or simply a single
32-bit word if signal data 21 consists of a single data bit.

For example, a 175 Hz grid spacing and a nominal C/A PRN
code period of 0.001 sec yields a value on the left-hand side
of inequality (11) of 0.04, which respects the assumed limit.

For example, a 175 Hz grid spacing and a nominal C/A PRN
code period of 0.001 sec yields a value on the left-hand side
of inequality (11) of 0.04, which respects the assumed limit.

Note that equations (8), (9), and (10) are an illustrative example of how software correlator (19) can rotate its l and
Q accumulations in order to correct for phase and frequency
errors in its table of pre-computed carrier replica signals.

continuing

where $\Delta f$ is in units of Hz, and T is the integration period.
Thus, a $\Delta f$ of 175 Hz causes a worst-case $C/N_0$ loss of 0.11
dB for T=0.001 sec.

Referencing now FIGS. 2A, 2B, and 3A, PRN codes
(composed of prompt PRN codes 29 (FIG. 3A) and early-
minus-late PRN codes 35 (FIG. 3A)) are either pre-com-
puted or generated in real-time. Pre-computing involves, for
each satellite, computing an entire PRN code, storing the
PRN code appropriately for easy retrieval, and referencing
the PRN code, possibly by means of indices that are com-
puted based on, for example, the incoming RF signal data 21
(FIGS. 2A and 3A). Pre-computing can be most advan-
tageously used when the PRN code is not very long. Gener-
atating PRN codes in real-time can be a more appropriate
solution when the PRN codes are very long (and thus would
require an unacceptable amount of storage), or perhaps when
too many PRN codes are required for the amount of storage
available, or for any other reason, but real-time PRN code
generation can entail an additional computational cost. Both
pre-computing and real-time determination of PRN codes
are described herein with respect to a bit-wise parallel
implementation.
Further continuing primarily to refer to FIGS. 2A, 2B, and 3A, many intermediate calculated quantities and at least three types of intermediate signals are also stored in bit-wise parallel format. First there are the in-phase and quadrature baseband mixed signals 23 (FIG. 3A), whose 3-bit representations for the illustrative embodiment are stored as baseband mixed sign 23A (FIG. 3A), baseband mixed high magnitude 23B (FIG. 3A), and baseband mixed low magnitude 23C (FIG. 3A). The second bit-wise parallel signal type is the fully mixed integrand, of which there are four signals: in-phase and quadrature fully mixed prompt integrand 31 (FIG. 3A) and in-phase and quadrature fully mixed early-minus-late integrand 33 (FIG. 3A). The former are stored as 3-bit representations in the illustrative embodiment as fully mixed prompt integrand sign 31A (FIG. 3A), fully mixed prompt integrand high magnitude 31B (FIG. 3A), and fully mixed prompt integrand low magnitude 31C (FIG. 3A). The latter are stored as 3.5-bit representations in the illustrative embodiment as fully mixed early-minus-late integrand sign 33A (FIG. 3A), fully mixed early-minus-late integrand high magnitude 33B (FIG. 3A), fully mixed early-minus-late integrand low magnitude 33C (FIG. 3A), and fully mixed early-minus-late integrand zero mask 33D (FIG. 3A). This representation is called a 3.5-bit representation because the sign, high-magnitude, and low-magnitude bits are ignored if the corresponding zero mask bit has the value zero. The third bit-wise parallel signal type is a value word, of which there are two types: prompt integrand value words 27 (FIG. 3B) and early-minus-late integrand value words 37 (FIG. 3B). Each fully mixed integrand is used to construct value words, one word for each possible value that the integer integrand can take on. There are eight possible values for the integrands of the illustrative embodiment: −1, −2, −3, −6, 1, 2, 3, and 6 for the in-phase and quadrature fully mixed prompt integrands 31 (FIG. 3A) and −2, −4, −6, −12, 2, 4, 6, and 12 for the in-phase and quadrature fully mixed early-minus-late integrands 33 (FIG. 3A). Each bit-wise parallel value word contains a one bit for each sample time when the integrand value equals the value of the value word, but it contains a zero bit for all other sample times. The storage of raw data and intermediate results in bit-wise parallel format allows the EXCLUSIVE OR operations that are involved in mixing to operate on thirty-two samples at a 2's sign bit is zero if early-minus-late PRN code zero mask bit is set to zero if early-minus-late PRN code takes on the value zero, and it is set to one if early-minus-late PRN code equals +2 or −2. Early-minus-late PRN code zero mask 35B (FIG. 2D) at sixteen sample times shown in FIG. 2D is equivalent to 2^{13} \times 2^{12} = 2^{25} = 32768. A 2's sign bit is set to one if early-minus-late PRN code equals +2 at the same time, and it is set to zero if the code equals −2. The 2's sign bit is irrelevant if the corresponding early-minus-late PRN code zero mask 35B (FIG. 2D) bit is zero. Early-minus-late PRN code sign 35A (FIG. 2D) for sixteen sample times contains X values that indicate bits whose values are irrelevant because the corresponding early-minus-late PRN code zero mask 35B (FIG. 2D) bits are zero. In an illustrative embodiment, all the X values become zero, thus the equivalent integer for early-minus-late PRN code sign 35A (FIG. 2D) is 2^{24}.

Continuing to refer to FIG. 3A, an alternative to taking the prompt PRN code 29 and early-minus-late PRN code 35 from pre-computed over-sampled PRN code table 28 is to generate prompt PRN code sign 29A, early-minus-late PRN code sign 35A, and early-minus-late PRN code zero mask 35B using real-time over-sampled PRN code generator 30A (FIG. 3A). Shown in FIG. 3A are two circles and a loose arrow with a quarter circle pointer. These are the symbols for a switch and indicate the ability of the system to choose possible alternate sources of PRN code. Using the real-time over-sampled PRN code generator 30A includes a step of generating the PRN code chips in real-time by conventional means. For example, the GPS civilian L2 CL and CM codes are generated by a 27-bit feedback shift register (see The New L2 Civil Signals, R. D. Fontana et al., Proceedings of the 1991 IEEE National Symposium on Communications and Information Systems, 1991, pp. 875-880, FIG. 25). The GPS civilian L2 CL and CM codes are generated by a 27-bit feedback shift register (see The New L2 Civil Signals, R. D. Fontana et al., Proceedings of the 1991 IEEE National Symposium on Communications and Information Systems, 1991, pp. 875-880, FIG. 25). The rates of the GPS civilian L2 CL and CM codes are generated by a 27-bit feedback shift register (see The New L2 Civil Signals, R. D. Fontana et al., Proceedings of the 1991 IEEE National Symposium on Communications and Information Systems, 1991, pp. 875-880, FIG. 25).
The method further includes the steps of choosing chip values from the PRN code, where the chip values correspond to a data interval that contains the samples of a data word and where the chips have a known timing relative to the data interval, transforming the relative timing into a time grid index, and translating the PRN code chip values and the time grid index for the data interval into the PRN code’s over-sampled bit-wise parallel format. These latter steps can be carried out efficiently by using a table look up function. One table each for prompt PRN code sign 29A, early-minus-late PRN code sign 35A, and early-minus-late PRN code zero mask 35B, and early-minus-late PRN code 35, on the other hand, is stored in a two-bit representation (actually a 1.5 bit representation); early-minus-late PRN code sign 35A and early-minus-late PRN code zero mask 35B, as denoted in Table 5. Note that the X in the first column of Table 5 indicates that zero or one can be placed in this location without affecting the corresponding code value. The X signifies a lack of effect of the sign bit on the code value when the zero mask bit equals zero. This is why the early-minus-late PRN code 35 representation is referred to as a 1.5 bit representation. This X value will affect the corresponding fully mixed early-minus-late integrand sign 33A, but it will not affect any of the early-minus-late value words because the zero value in the corresponding zero mask location will null out the corresponding bit of all early-minus-late value words.

Continuing to refer to FIG. 3A, and continuing to describe the bit-wise parallel algorithms, the required amount of storage for tables of pre-computed prompt PRN code 29 and early-minus-late PRN code 35 can be greatly reduced by making two simplifications. First, the prompt PRN code 29 is stored as prompt PRN code sign 29A. This representation is shown in Table 4. The early-minus-late PRN code 35, on the other hand, is stored in a two-bit representation (actually a 1.5 bit representation); early-minus-late PRN code sign 35A and early-minus-late PRN code zero mask 35B, as denoted in Table 5. Note that the X in the first column of Table 5 indicates that zero or one can be placed in this location without affecting the corresponding code value. The X signifies a lack of effect of the sign bit on the code value when the zero mask bit equals zero. This is why the early-minus-late PRN code 35 representation is referred to as a 1.5 bit representation. This X value will affect the corresponding fully mixed early-minus-late integrand sign 33A, but it will not affect any of the early-minus-late value words because the zero value in the corresponding zero mask location will null out the corresponding bit of all early-minus-late value words.

Continuing to refer to FIG. 3A, multiplication of the RF front-end output representation, the sign signal 21A and signal magnitude 21B, of Table 1 by the sine wave representation, carrier replica sign 25A and carrier replica magnitude 25B, of Table 2 yields baseband mixed signals 23, consisting of baseband mixed sign 23A, baseband mixed high magnitude 23B, and baseband mixed low magnitude 23C, that can take on the values -6, -3, -2, -1, +1, +2, +3, and +6, as shown in Table 3. Baseband mixed high magnitude 23B is simply signal magnitude 21B, and baseband mixed low magnitude 23C is carrier replica magnitude 25B.
Another simplification in the pre-computed over-sampled PRN code table 28, and continuing to refer to FIG. 3A, can be to ignore code Doppler shift variations. All signals in the table are assumed to have zero Doppler shift; i.e., all C/A codes in the table assume that $T_{2A+1} - T_{2A} = 0.001$ sec. Note that the period of 0.001 is applicable for accumulations that use the full 1023 chips of the C/A code only. Any other type of code or accumulation interval may have a different period.

The code phase errors due to this assumption can be eliminated by choosing a replica code from the pre-computed over-sampled PRN code table 28 whose midpoint occurs at the desired midpoint time $(T_{2A+1} + T_{2A})/2$. The only other effect of this assumption can be a small correlation power loss, which is no more than 0.014 dB if the magnitude of the Doppler shift is less than 10 KHz.

The pre-computed over-sampled PRN code table 28 should include a selection of different phases, for example fourteen, as measured relative to a signal sample spacing of, for example, 175 nsec. This translates into a code phase spacing of, for example, 12.5 nsec, which equals a pseudo range measurement digitization level of 3.8 m, or a maximum measurement error of 1.9 m. The number of phases in the pre-computed over-sampled PRN code table 28 is dependent upon the design of the system and no set number of phases is required by the present invention. Referring to FIG. 6, suppose that pre-computed over-sampled PRN code table 28 stores over-sampled bit-wise parallel representations of chips $C(1)$ through $C(M)$. The table must allow for the retrieval of over-sampled bit-wise parallel code replicas for a range of start times of code chip $C(1)$ that span the entire first data sample word in the accumulation interval $W$, 95 (FIG. 6). The table may contain code replicas whose different phases yield start times that span only a single sample interval of data word $W$, 95 (FIG. 6), which is only $1/n$, of the required number of start times. In this case the software correlator may apply bit shift operations to a tabulated PRN code replica from that sample interval in order to generate the over-sampled bit-wise parallel PRN code replica that applies when chip $C(1)$ starts in a different sample interval of data word $W$, 95 (FIG. 6).

Continuing to refer to FIG. 3A, and further continuing to describe the bit-wise parallel algorithms, prompt PRN code 29 and early-minus-late PRN code 35 replicas can be mixed with the baseband mixed signals 23 to form fully mixed prompt integrand 31 by an EXCLUSIVE OR operation and bit re-definitions. An EXCLUSIVE OR between prompt PRN code sign 29A and baseband mixed sign 23A produces fully mixed prompt integrand sign 31A given in Table 6. The fully mixed prompt integrand high magnitude 31B and fully mixed prompt integrand low magnitude 31C are baseband mixed high magnitude 23B and baseband mixed low magnitude 23C, also given in Table 6. Note that the Table 6 representation is identical to that of Table 3 except for the inversion in the meaning of the sign bits. The number of magnitude bits is dependent upon the design of the system and no set number of magnitude bits is required by the present invention. A change in the number of magnitude bits will cause a change in the number of entries of the equivalent of Table 6 and it will affect the possible values of the integrand.

<table>
<thead>
<tr>
<th>Fully Mixed Prompt Integrand Sign</th>
<th>Fully Mixed Prompt Integrand High Magnitude</th>
<th>Fully Mixed Prompt Integrand Low Magnitude</th>
<th>Fully Mixed Prompt Integrand Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>-3</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>-6</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>+1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>+2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>+3</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>+6</td>
</tr>
</tbody>
</table>

Still continuing to refer to FIG. 3A, the mixing of the early-minus-late PRN code 35 with the baseband mixed signals 23 forms fully mixed early-minus-late integrands 33.

Referring now to FIGS. 3A, 3B, 4A, and 4B, the method for computing in-phase and quadrature accumulations for every accumulation period, for example every millisecond for GPS C/A code, by use of bit-wise parallelism includes the steps of selecting carrier replica signal 25 (FIG. 3A) according to the proximity of its frequency to the desired frequency, and representing sample signal data 21 (FIG. 3A) and carrier replica signal 25 (FIG. 3A) from at least one channel as bits in signal sign 21A (FIG. 3A) and, if present, signal magnitude 21B (FIG. 3A) and carrier replica sign 25A (FIG. 3A) and carrier replica magnitude 25B (FIG. 3A) (method step 101, FIG. 4A). Note that carrier replica signal 25 (FIG. 3A) is chosen so that its frequency is close to the correct signal frequency. The method also includes the step of mixing signal data 21 (FIG. 3A) to baseband by comput-
ing in-phase and quadrature baseband mixed signal 23A (FIG. 3A) and in-phase and quadrature baseband mixed high and low magnitude 23B/C (FIG. 3A) (method step 103, FIG. 4A). The method further includes the steps of selecting PRN code from pre-computed over-sampled PRN code table 28 (FIG. 3A) or of computing it using real-time over-sampled PRN code generator 30A (FIG. 3A), representing prompt PRN code 29 (FIG. 3A) as prompt PRN code sign 29A (FIG. 3A), and representing early-minus-late PRN code 35 (FIG. 3A) from as early-minus-late PRN code sign 35A (FIG. 3A) and early-minus-late PRN code zero mask 35B (FIG. 3A) (method step 105, FIG. 4A). The method further includes the step of de-spreading in-phase and quadrature baseband mixed signal 23 (FIG. 3A) by mixing it with prompt PRN code 29 (FIG. 3A) and early-minus-late PRN code 35 (FIG. 3A), resulting in in-phase and quadrature fully mixed prompt integrands 31 (FIG. 3A), and fully mixed early-minus-late integrands 33 (FIG. 3A) (method step 107, FIG. 4A). The method further includes the step of using prompt value word logic 27A (FIG. 3B) to compute prompt integrand value words 27 (FIG. 3B) from the in-phase and quadrature fully mixed prompt integrands 31 (FIG. 3A). The method further includes the step of using early-minus-late integrand value logic 37A (FIG. 3B) to compute early-minus-late integrand value words 37 (FIG. 3B) from the fully mixed early-minus-late integrands 33 (FIG. 3B) (method step 109, FIG. 4A). The method further includes the steps of multiplying prompt integrations 41 (FIG. 3B) and early-minus-late integrations 49 (FIG. 3B) by corresponding values 41A and summing the resulting values over the words of each signal for an entire accumulation interval to yield in-phase and quadrature summed prompt integrations 45 (FIG. 3B) and summed early-minus-late accumulations 47 (FIG. 3B) (method step 115, FIG. 4B) that are stored for use by acquisition techniques or tracking loops. The method further includes the step of rotating the in-phase and quadrature summed prompt integrations 45 (FIG. 3B) and summed early-minus-late accumulations 47 (FIG. 3B) (method step 117, FIG. 4B) to simulate a condition in which baseband mixing had been performed using cosine and sine signal replicas with the correct frequency and phase. If there are more channels to process (decision step 119, FIG. 4B), the method includes the step of repeating the previous steps beginning at method step 101, FIG. 4A. If there are no more channels to process (decision step 119, FIG. 4B), the method includes the step of setting parameters for the next accumulation period, including storing current C/A code phases, epoch counters, carrier phases, and carrier Doppler shifts (method step 121, FIG. 4B). If the time period to wait until the next accumulations need to be calculated has not expired (decision step 123, FIG. 4B), the method includes the step of sleeping until the expiration of the time period (method step 125, FIG. 4B). If the time period has expired (decision step 123, FIG. 4B), the method includes the step of repeating the previous steps beginning at method step 101, FIG. 4A. The length of the time period depends on the nominal accumulation period. It is set to be less than this period, normally between 50% to 90% of this period, to reduce the possibility that accumulations are missed for any channels.

Referring again to FIGS. 3B and 4A, method step 109 (FIG. 4A) calls for computing value words. This computation starts by performing bit-wise parallel Boolean for each of the possible values in the right-hand column of the prompt integrand representation in Table 6. A 32-bit prompt integrand value word 27 (FIG. 3B) is computed for each thirty-two samples and each row of Table 6. The prompt integrand value word 27 (FIG. 3B) contains ones for the sample times when the actual integrand equals the corresponding value in the right-hand column of Table 6, and zeros for the remaining times when the actual integrand does not equal this value. The prompt integrand value words 27 (FIG. 3B) corresponding to the possible Table 6 values are formed by method step 109 (FIG. 4A) as follows:

\[
\begin{align*}
\text{MINUSTWO} &= \not(\text{SIGN}) \text{ AND } \not(\text{HIGHMAG}) \text{ AND NOT(LOWMAG)} \\
\text{MINUSFOUR} &= \not(\text{SIGN}) \text{ AND } \not(\text{HIGHMAG}) \text{ AND NOT(LOWMAG)} \\
\text{MINUS SIX} &= \not(\text{SIGN}) \text{ AND } \not(\text{HIGHMAG}) \text{ AND NOT(LOWMAG)} \\
\end{align*}
\]

Continuing to refer to FIGS. 3A, 3B, 4A, and 4B, method steps 109 (FIG. 4A), 111 (FIG. 4B), and 113 (FIG. 4B) call for operations for the fully mixed early-minus-late integrands 33 (FIG. 3A) that are similar to those for the fully mixed prompt integrands 31 (FIG. 3A). Early-minus-late integrand value words 37 (FIG. 3B) correspond to values that are double those of the prompt integrand value words 27 (FIG. 3B), i.e., the MINUS SIX word becomes the MINUSTWELVE word. Also, an additional AND operation must be performed with the zero mask bits of Table 7 in order to mask out sample times when the early and late PRN codes cancel each other. Possible formulas for the method step 109 (FIG. 4A) computation of these early-minus-late integrand value words 37 (FIG. 3B) are as follows:

\[
\begin{align*}
\text{MINUSTWO} &= \text{ZEROMASK AND NOT(SIGN)} \text{ AND } \not(\text{HIGHMAG}) \text{ AND NOT(LOWMAG)} \\
\text{MINUSFOUR} &= \text{ZEROMASK AND NOT(SIGN)} \text{ AND } \not(\text{HIGHMAG}) \text{ AND NOT(LOWMAG)} \\
\text{MINUS SIX} &= \text{ZEROMASK AND NOT(SIGN)} \text{ AND } \not(\text{HIGHMAG}) \text{ AND NOT(LOWMAG)} \\
\end{align*}
\]

US 7,305,021 B2
Additional zero masking can occur in the first and last words of an accumulation interval. This is true because the start and stop times of an accumulation interval do not normally fall at the boundaries of data words. Therefore, the bits in the first word that precede the accumulation interval may need to get zero masked as might the bits in the last word that come after the end of the accumulation interval.

Referring primarily to FIGS. 3B and 4B, the one bits counting operations of method step 111 (FIG. 4B) form the count of the number of one bits in each of the eight value words. If there are no such counting operations in the instruction set of microprocessor 16 (FIG. 1), the counting can be accomplished using a table look-up. In the case of a table look-up, prompt integrand value words 27 and early-minus-late integrand value words 37 (FIG. 3B) can be used as addresses in one bits summation table 38 (FIG. 3B), and one bits summation table 38 (FIG. 3B) can output the number of one values (or zeros) in the address. For example, if the table look-up operation is called BITSUM, the following computations can be performed to compute one-bits counts:

\[ \text{ONESCOUNT} = \text{BITSUM} (\text{VALUEWORD}) \]  

where the output of the table ONECOUNT is the number of one bits in the word VALUEWORD. This operation is repeated for each of the prompt integrand value words 27 (FIG. 3B) and early-minus-late integrand value words 37 (FIG. 3B) in order to accomplish method step 111 (FIG. 4B). Selection of table width, for example 16-bit or 32-bit, depends on the amount of memory available and other design decisions. If the table width is smaller than the number of bits in a value word, then multiple calls of the table are used in order to sum up the total number of one values in a given value word. Each call takes as input only a portion of the bits in the value word.

Continuing to refer primarily to FIGS. 3B and 4B, the accumulation operations of method steps 113 (FIG. 4B) and 115 (FIG. 4B) sum the one bits count for each prompt integrand value word 27 (FIG. 3B) and for each early-minus-late integrand value word 37 (FIG. 3B) over the entire accumulation interval, multiply each result by the value 41A (FIG. 3B) that is associated with the value word, and sum all of these scaled value accumulations to form the accumulations of equations (6) and (7), summed prompt accumulation 45 (FIG. 3B) and summed early-minus-late accumulation 47 (FIG. 3B). For example, the following computations can be performed to compute the in-phase summed prompt accumulation 45 in equation (6) as follows:

\begin{equation}
I_{ph}(0) = -\sum_{k=1}^{N_v} \text{ONESCOUNT}(-1)_{47k} + \sum_{k=1}^{N_v} \text{ONESCOUNT}(-2)_{47k} + \sum_{k=1}^{N_v} \text{ONESCOUNT}(-3)_{47k} + \sum_{k=1}^{N_v} \text{ONESCOUNT}(-6)_{47k} + \sum_{k=1}^{N_v} \text{ONESCOUNT}(-1)_{47k} + \sum_{k=1}^{N_v} \text{ONESCOUNT}(+2)_{47k}
\end{equation}

where 1 is the index of successive bit-wise parallel data words in the accumulation interval, \( N_v \) is the total number of data words in the interval, and \( \text{ONESCOUNT}(k)_{47k} \) is the ones count for the corresponding value word 41 (FIG. 3B) associated with value k 41A (FIG. 3B) for the lth data word interval and the in-phase summed prompt accumulation 45 (FIG. 3B). The quadrature summed prompt accumulations 45 (FIG. 3B) and the in-phase and quadrature summed early-minus-late accumulations 47 (FIG. 3B) are calculated in a similar manner. The only difference is in the actual ONECOUNT values used and, for the case of early-minus-late signals, the set of k values 41A (FIG. 3B).

Continuing to refer primarily to FIGS. 4A and 4B, the method of the present invention can be adapted to work with a different number of bits in the representation of the RF front-end output and of the baseband mixed signals. An increase above two bits can make the logic more complex and may decrease the time savings over straight integer arithmetic. A decrease to a 1-bit representation can have the opposite effect. For example, if the RF front-end uses 1-bit digitization rather than 2-bit digitization while carrier replica signal 25 (FIG. 2B) retains its 2-bit digitization, then the operation count can decrease by a factor of almost two for the 1-bit method, which can make the logic execute about 4.2 times faster than straight integer arithmetic.

Returning to the discussion of determining PRN code, and now referring again FIGS. 2C, 2D, and 3A, the real-time generation of bit-wise parallel over-sampled prompt PRN code sign 29A (FIGS. 2D and 3A), early-minus-late PRN code sign 35A (FIGS. 2D and 3A), and early-minus-late PRN code zero mask 35B (FIGS. 2D and 3A) can be carried out by real-time over-sampled PRN code generator 30A (FIG. 3A). The inputs to this calculation are the actual PRN code chip length 65 (FIG. 2D), \( \Delta t_r \), the sample interval 63 (FIG. 2D), \( \Delta t_{enb} \), the nominal early-to-late code delay 61 (FIG. 2D), \( \Delta t_{enb} \), the time lag 67 (FIG. 2D) from the first RF sample time to the end time of the first prompt PRN code chip, and prompt code chips 91 (FIGS. 2D and 6). The outputs are the three integers that store the prompt PRN code sign 29A (FIGS. 2D and 3A), early-minus-late PRN code zero mask 35B (FIGS. 2D and 3A), and early-minus-late PRN code sign 35A (FIGS. 2D and 3A), which are all in bit-wise parallel format.

Referring again to FIGS. 2C and 2D, table look-ups can be used to translate a PRN code and its timing information to bit-wise parallel representations of its over-sampled prompt and early-minus-late versions. The required table look-ups can be simplified by recognizing that the following parameters are substantially constant, for the purposes of this calculation: sampling interval 63 (FIG. 2D), \( \Delta t_r \), the nominal chip length, \( \Delta t_{enb} \), the early-minus-late code delay 61 (FIG. 2D), \( \Delta t_{enb} \), used by software correlator 19 (FIG. 1), and the maximum number of chips that span a data word of microprocessor 16 (FIG. 1). The difference between the actual chipping rate \( \Delta t_r \), (reciprocal of \( \Delta t_r \)) and the nominal chipping rate \( \Delta t_{enb} \), (reciprocal of \( \Delta t_{enb} \)) that is used for the above simplification can be accommodated by correcting time lag 67 (FIG. 2D), \( \Delta t_{enb} \), for the average effects of Doppler shift, a procedure discussed later. Using the simplification,
The number of code chips is required in order for the prompt code chips 91 (FIGS. 2D and 6) and time lag l67 (FIG. 2D), \( \Delta t_u \). A table look-up procedure for each signal component yields a single integer result for prompt PRN code sign 29A (FIGS. 2D and 3A), another single integer result for early-minus-late PRN code zero mask 35B (FIGS. 2D and 3A), and yet another single integer result for early-minus-late PRN code sign 35A (FIGS. 2D and 3A).

Time lag l67 (FIG. 2D), \( \Delta t_u \), can take on an infinite number of values in the continuous range:

\[
\frac{1}{2} \Delta t_{\text{rest}} < \Delta t_u < \frac{1}{2} \Delta t_{\text{rest}}
\]  
(31)

This range's lower limit guarantees that the end time of the first late chip occurs no earlier than the first sample time. A lower time lag l67 (FIG. 2D) value would make the first chip irrelevant to the prompt PRN code 29 (FIG. 2D), early code 69B (FIG. 2D), and late code 69C (FIG. 2D) at all of the sample times. The upper limit in equation (31) guarantees that the start time of the first late chip occurs no later than the first sample. A larger value of \( \Delta t_u \) would leave the code sign 29A (FIGS. 2D and 3A), and yet another single integer result for early-minus-late PRN code sign 35A (FIGS. 2D and 3A).

Referring now to FIG. 5, to create an electronically processable table, the continuous range of \( \Delta t_u \) values can be replaced with a discrete grid having m equally spaced points per sample interval 63 (FIG. 2D), \( \Delta t_s \). The integer m is chosen to be large enough so that the granularity \( \Delta t_s / m \) gives sufficient PRN code timing resolution. In GPS applications m is usually chosen to be large enough so that \( c \Delta t_s / m \) is on the order of several meters or less, where \( c \) is the speed of light, but reasonably sized because the table sizes are usually proportional to m. Given a choice of m, the grid of relative end times of the first prompt code period is:

\[
\Delta t_{\text{grid}} = \frac{k \Delta t_s}{m} \quad \text{for } k = k_{\text{min}}, \ldots, k_{\text{max}}
\]  
(32)

where the limits

\[
k_{\text{min}} = \text{floor}(\frac{m \Delta t_{\text{rest}}}{2 \Delta t_s}) - 2
\]  
(33a)

\[
k_{\text{max}} = \text{floor}(\frac{m (\Delta t_s - \frac{1}{2} \Delta t_{\text{rest}})}{\Delta t_s})
\]  
(33b)

provide full coverage of the interval defined in equation (31). The floor ( ) function rounds to the nearest integer in the direction of \( -\infty \). This \( k_{\text{min}} \) value can cause the minimum \( \Delta t_{\text{grid}} \) to fall slightly below the lower limit in equation (31), which can cause memory inefficiency, but this value is advantageous because it may simplify some further computations.

The size for each table can be a function of the maximum number of code chips that may fall within a data word's sample range. Given \( \Delta t_{\text{grid}} \), bit information for the following number of code chips is required in order for the prompt PRN code 29 (FIG. 2D), early code 69B (FIG. 2D), and late code 69C (FIG. 2D) to be fully specified at all of the data word's sample times:

\[
l(\Delta t_{\text{grid}}) = \text{floor}
\left\{ \frac{[n_e - 1] \Delta t_e - \Delta t_0 + \frac{1}{2} \Delta t_{\text{rest}}}{\Delta t_e} \right\} + 2
\]  
(34)

where \( n_e \) is the number of data samples that can be stored in bit-wise parallel format in each word. It is clear from equation (34) that \( l(\Delta t_{\text{grid}}) \) is a non-increasing function of \( \Delta t_u \). Therefore, the maximum number of required chips occurs at the minimum value of \( \Delta t_u \):

\[
L = l(\Delta t_{\text{min}})
\]  
(35)

The size of each table can be determined from the parameters \( k_{\text{min}}, k_{\text{max}} \), and L. The grid contains \( k_{\text{min}} \times L \) different time offsets of the first code chip. At each of these grid points there are \( 2^L \) possible combinations of the code chips. Thus, each table optimally contains \( k_{\text{min}} \times L \) entries, and each entry is optimally an unsigned integer in the range from 0 to \( 2^L - 1 \).

Continuing to refer to FIG. 5, each table can be stored as an array with a single index. The first \( 2^L \) entries correspond to the \( 2^L \) different possible chip sequences that can occur at \( \Delta t_u = \Delta t_{\text{grid}} \), the next \( 2^L \) entries correspond to \( \Delta t_u = \Delta t_{\text{grid}} + \Delta t_s \), and so forth. The tabulated bit sequences for a fixed \( \Delta t_{\text{grid}} \) are ordered by interpreting the sequence as a binary index counter with the first chip being the most significant counter bit and the \( L \)th chip being the least significant bit. The integer elements of the table can be the \( x(i) \) table elements 81 with corresponding code time offset 83 \( \Delta t_{\text{grid}} \) and corresponding bit sequence 85 of the chips. The array index of a given \( x(i) \) table element 81 can be computed based on its code time offset 83 \( \Delta t_{\text{grid}} \) and grid index k and its corresponding bit sequence 85. The corresponding bit sequence 85 associated with the array index consists of the chip values C(1), C(2), C(3), . . . , C(L). The C(j) chip values are either zero or one, with zero representing a -1 PRN code value and one representing a +1 PRN code value, and they are listed in order of increasing time. The corresponding array index of the \( x(i) \) table element 81 is:

\[
[i(k, C(1), C(2), \ldots, C(L))] = 1 + (k - k_{\text{min}}) \times 2^L + \sum_{j=1}^{L} C(j) \times 2^{L-j}
\]  
(36)

This equation can be inverted to give the code time offset 83 grid index k and the corresponding bit sequence 85 as functions of the \( x(i) \) table element 81 index i:

\[
h(i) = k_{\text{min}} + \text{floor}(i/2^L) - 1
\]  
(37a)

\[
C(j, i) = \text{mod} \left( \text{floor}(\frac{\text{mod}(i - 1, 2^L)}{2^{L-j}}), 2 \right) \quad \text{for } j = 1, 2, 3, \ldots, L
\]  
(37b)

where \( \text{mod}(y,z) = y - z \times \text{floor}(y/z) \) is the usual remainder function.

Continuing to refer to FIG. 5, the following computations generate the \( x(i) \) table elements 81 entries of the three tables. Given i, the corresponding code time offset 83 grid index k(i) is computed from equation (37a) and is used to generate three sequences of chip indices:
where \( n \) is the index of the sample time within the oversampled data word. The integer \( \lfloor j_p(n,i) \rfloor \) is the index of the PRN code chip that applies at sample \( n \) for the prompt PRN code 29 (FIG. 2D). The integers \( j_e(n,i) \) and \( j_z(n,i) \) are, respectively, the early code \( 69B \) (FIG. 2D) and late code \( 69C \) (FIG. 2D), respectively. The formulas in equations (38a)-(38c) amount to time measurements of each sample given in units of chip lengths past the first chip. These integers, in turn, can be used to determine the chip values that apply at the sample times: past the first sample of data word.

The integers \( j_e(n,i) \) and \( j_z(n,i) \) are defined similarly for the early code \( 69B \) (FIG. 2D) and late code \( 69C \) (FIG. 2D), respectively. The formulas in equations (38a)-(38c) amount to time measurements of each sample given in units of chip lengths past the first chip. These integers, in turn, can be used to determine the chip values that apply at the sample times: past the first sample of data word. The integers \( j_e(n,i) \) and \( j_z(n,i) \) are defined similarly for the early code \( 69B \) (FIG. 2D) and late code \( 69C \) (FIG. 2D), respectively. The formulas in equations (38a)-(38c) amount to time measurements of each sample given in units of chip lengths past the first chip. These integers, in turn, can be used to determine the chip values that apply at the sample times: past the first sample of data word.
with the sample time, but it will not get correlated with that chip if its sample time coincides exactly with the end time of the code chip.

Continuing to refer to FIG. 6, efficiently determining the correct $x_{p}(i), x_{em\text{zzm}}(i),$ and $x_{em\text{zzs}}(i)$ bit-wise parallel code representations for the $N$ data words $W_{v}$ (FIG. 6) through data word $W_{v+1}$ (FIG. 6) involves making an efficient determination of the correct table index $i,$ that corresponds to data word $W_{v+1}$ for $v=0,1,2,\ldots, N,$ where the table index $i,$ is a function of start lag $\Delta_{start}.$ (FIG. 6), actual PRN code chip length $\Delta_{s},$ (FIG. 6), $v,$ and prompt code chips $C(0),$ $C(1),$ $C(2),\ldots,$ $C(M+1)$ (FIG. 6). The chip value $C(0)$ (FIG. 6) is needed in order to specify the late code 69C (FIG. 2D) at the initial few samples of the accumulation, and the chip value $C(M+1)$ (FIG. 6) is needed to specify the early code 69B (FIG. 2D) at the final few samples. Additional constants that can be used in order to determine the $i,$ indices are $\Delta_{s}, n_{v}, n_{t}, k_{v}, k_{s},$ and nominal chip length $\Delta_{tau},$ which has been used to generate the three $x(i)$ tables.

The first step of the index calculation procedure precomputes and stores a table of candidate integers for the final determination of the correct index into the $x(i)$ tables. Given these two quantities, the correct index for the three $x(i)$ tables is:

$$i_{c}=\lfloor(k_{v}n_{v}+M+6)\rfloor$$

where $n_{v}$ is the integer number of fine-scale time intervals per sample interval 63 (FIG. 2D), $\Delta_{tau}.$ This number is chosen large enough, for example $n_{v}>2mN,$ to preclude any significant build-up of timing errors during an accumulation interval due to the finite time resolution $\Delta_{tau}.$ $N$ is the number of data words in the accumulation interval. The calculation of the $k_{v}$ values over one accumulation interval involves approximately $N$ iterative time increments, each of which has a resolution of $\Delta_{tau}.$ If $m_{v}$ obeys the inequality given above, then the cumulative timing errors due to the finite precision $\Delta_{tau}$ will be less than the timing error caused by the finite timing precision of the $x(i)$ tables. Normally it is possible to make $m_{v}$ much larger than $2mN$ and still keep all of the relevant calculations within the size limits of a 32-bit signed integer. If $m_{v}$ is a power of two, a rightward bit shift operation can be used to implement integer division by $m_{v}.$

Time unit $\Delta_{tau}$ can be used to define an integer that approximately keeps track of the code/sample time offset $\Delta_{tau}$ for data word $W_{v}$:

$$k_{p}=\text{round}\left(\frac{\Delta_{tau}}{\Delta_{tau}}\right) = \text{round}\left(\frac{m_{v}\Delta_{tau}}{\Delta_{tau}}\right)$$

where the round() function rounds up or down to the nearest integer. The time lag $67$ (FIG. 2D), $\Delta_{tau},$ is the amount by which the end time of PRN code chip $C(p_{v}-L)$ lags the first sample time of data word $W_{v}.$ The algorithm that iteratively determines $k_{p}$ tries to keep the relationship in equation (47) exact, but using only integer operations can allow small errors to build up. Note that $k_{p}, k_{p}, k_{v},$ as implied by a comparison of equations (32) and (47). This relationship can be used to determine $k_{p}$ from an iteratively determined $k_{p}.$

Several constants are required by the iterative procedure that determines $k_{v}, k_{p}$ and $k_{p}.$ The first five constants are used to account for the difference between the nominal chip length $\Delta_{tau}$ used to generate the $x(i)$ tables, and the actual chip length 65 (FIG. 2D), $\Delta_{tau},$ used in the accumulation:

$$k_{p}=\text{round}\left(\frac{\Delta_{tau}}{\Delta_{tau}}\right) = \text{round}\left(\frac{m_{v}\Delta_{tau}}{\Delta_{tau}}\right)$$

Note that the $\text{mod}(2x,2^{j})$ operation in the latter two equations can be replaced by a single truncated leftward bit shift.

In many cases prompt code chips 91 (FIG. 6) $C(0),$ $C(1),$ $C(2),\ldots,$ can be generated as the output of a feedback shift register or a system of such registers. For example, the new GPS civilian L2 signals can be generated this way. In this case, each iteration of equation (44b) can be interleaved with an iteration of the shift register calculations. Shift-register generation of PRN codes is well-known in the art.

An alternative to building up the previously-described table is to calculate the index component only for one data word at a time. Suppose that $\Delta_{i}$ is the correct index component for data word $W_{v}$ and that $\mu_{v}$ is the auxiliary index that would have been used to determine $\Delta_{i}$ from the $\Delta_{i}$ table had the table existed. In order to calculate $\Delta_{i}$ for data word $W_{v+1},$ $\mu_{v+1}$ is computed (procedure defined herein), feedback shift register calculation that generates $C(\mu_{v}), C(\mu_{v}+1), C(\mu_{v}+2),\ldots,$ $C(\mu_{v}+\tau),$ are iterated, and the resulting chip values are used to perform $(\mu_{v}+\tau)-\mu_{v}$ iterations of equations (44b) or (44c).
\begin{align*}
\lambda &= \Delta t - \Delta t_{\text{nom}} \\
\sigma_{f,0} &= \text{cei}\left[\left(k_{f,0} - \frac{m_f}{2}\right)\right] \text{sign}(\lambda) \\
b_{f,0} &= \begin{cases} 
1 & \text{if } \Delta t = \Delta t_{\text{nom}} \\
\text{round}\left(k_{f,0} + \frac{m_f}{2}\right) & \text{if } \Delta t \neq \Delta t_{\text{nom}}
\end{cases}
\end{align*}

where the sign() function returns +1 if its input argument is positive, zero if the argument is zero, and -1 if the argument is negative. The index \(k_{f,0}\) is approximately half the length of a data word as measured in units of \(\Delta t\) seconds. During an accumulation, the rational factor \(\sigma_{f,0}/b_{f,0}\) gets multiplied by the time offset between the end time of the first code chip and the midpoint of the data word. The result is a time perturbation that removes the average effect of the difference between the actual and nominal PRN code chipping rates. The time perturbation can be used to compute a corrected \(k_{f,0}\) value:

\[k_{f,0}(k_{f,0}) = k_{f,0} + \text{round}\left[k_{f,0} - k_{f,0} \frac{\sigma_{f,0}}{b_{f,0}}\right]\]

Equation (48d) picks \(b_{f,0}\) to equal a power of two so that the integer division by \(b_{f,0}\) in equation (49) can be accomplished using a rightward bit shift operation. The round() operation in equation (49) can be accomplished as part of the division if one first adds sign\(b_{f,0}\)\(b_{f,0}/2\) to the quantity \(k_{f,0}-k_{f,0}\)\(a_{f,0}\) before performing the rightward bit shift that constitutes division by \(b_{f,0}\). This approach can give the correct \(k_{f,0}\) because the signs of \((k_{f,0}-k_{f,0})\) and \(b_{f,0}\) are both positive and because the rightward bit shift has the effect of rounding the signed division result towards zero. An alternative implementation of the round function could be used for applications that do not guarantee \(k_{f,0} > k_{f,0}\). Such applications are normally associated with \(L \leq 2\) PRN code chips per data word. Each additional constant can be used to define the \(k_{f,0}\) and \(\mu_{v}\) iterations:

\begin{align*}
L_{\mu_0} &= \text{round}\left[\frac{m_f\Delta t}{\Delta t}\right] \\
\Delta k_{f,0} &= \text{round}\left[m_f\Delta t_{\text{nom}}/\Delta t\right] \\
\Delta k_{f,0} &= \text{round}\left[m_f\Delta t_{\text{nom}}/\Delta t\right] \\
k_{f,\text{min}} &= \text{round}\left[\frac{m_f\Delta t_{\text{nom}}}{m_f}\right] \\
k_{f,\text{max}} &= \text{round}\left[\frac{m_f\Delta t_{\text{nom}} - 1}{m_f}\right]
\end{align*}

The constant \(L_{\mu_0}\) is the typical number of code chips per data word. It is the nominal increment to \(\mu_{v}\) per data word. The constant \(\Delta k_{f,0}\) equals the number of fine-scale time intervals per PRN code chip. The constant \(\Delta k_{f,0}\) is used to adjust \(k_{f,0}\) up or down if \(k_{f,0}\) falls outside of the limits: \(k_{f,\text{min}} \leq k_{f,0} \leq k_{f,\text{max}}\). The constant \(\Delta k_{f,0}\) is the nominal increment to \(\mu_{v}\) per data word. The limits \(k_{f,\text{min}}\) and \(k_{f,\text{max}}\) are approximately the limits \(k_{f,\text{min}}\) and \(k_{f,\text{max}}\) from equations (33a) and (33b) re-scaled to the new fine time scale and adjusted for the difference between the nominal code chipping rate of the \(x(i)\) and the actual chipping rate of the accumulation. The extra \(-2\) term on the right-hand side of equation (33a) is compensated for by the increment to \(k_{f,\text{min}}\) on the right-hand side of equation (50d) and the decrement to \(k_{f,\text{max}}\) on the right-hand side of equation (50e). The original \(-2\) term and the increment and decrement have been included because they ensure that \(k_{f,0}\) values which respect the limits in equation (50d) and (50e) are transformed into \(k_{f,0}\) values that respect the limits in equations (33a) and (33b).

The iteration begins by initializing \(k_{f,0}\) and \(\mu_{v}\) for the first data word. The nominal initial values are:

\begin{align*}
k_{f,0} &= \text{round}\left[\frac{\Delta t_{\text{nom}}}{2}\right] + 1 + \text{floor}\left(\frac{\Delta t_{\text{nom}}}{2}\right)
\mu_{v} &= \text{floor}\left(\frac{2}{\Delta t}\right) + 1 + L
\end{align*}

It is possible that \(k_{f,\text{nom}}\) from equation (51a) can violate its upper limit \(k_{f,\text{max}}\). Therefore, the following conditional adjustment can be implemented in order to finish the initialization:

\begin{align*}
k_{f,1} &= \begin{cases} 
k_{f,\text{nom}} & \text{if } k_{f,\text{nom}} \leq k_{f,\text{max}} \\
k_{f,\text{nom}} - \Delta k_{f,0} & \text{if } k_{f,\text{nom}} < k_{f,\text{nom}}\end{cases} \\
\mu_{v} &= \begin{cases} 
\mu_{v,\text{nom}} + 1 & \text{if } k_{f,\text{nom}} < k_{f,\text{min}} \\
\mu_{v,\text{nom}} & \text{if } k_{f,\text{nom}} \geq k_{f,\text{min}} \text{ and } k_{f,\text{nom}} \leq k_{f,\text{max}} \text{ for } v = 2, 3, 4, \ldots, N
\end{cases}
\end{align*}

Given this initialization, the calculation of \((k_{f,0}, k_{f,1}), (k_{f,0}, k_{f,2}), \ldots, (k_{f,0}, k_{f,0})\) proceeds according to the following iteration:

\begin{align*}
k_{f,\text{nom}} &= k_{f,\text{nom}} - \Delta k_{f,0} + \Delta k_{f,0} \text{ for } v = 2, 3, 4, \ldots, N \\
\mu_{v} &= \mu_{v,\text{nom}} + 1 & \text{for } v = 2, 3, 4, \ldots, N
\end{align*}

The table look-up calculations finish with the computation of \(k_{f,0}\) and \(\mu_{v}\), and the actual table look-ups:

\[k_{f,0} = \text{round}\left[m_f\mu_{v}(k_{f,0})/m_f\right] \text{ for } v = 1, 2, 3, \ldots, N\]
The round( ) operation in equation (55) can be implemented by adding \( m/2 \) to \( k_{n0}(k_{p}) \) before the rightward bit shift that constitutes division by \( m \). The result of the division will be the correct value of \( k_{n} \) for any sign of \( k_{n0}(k_{p}) \) if the computer works with 2’s complement notation for signed integers and if the rightward bit shift fills in from the left with the 2’s compliment sign bit, i.e., with the left-most bit.

Given \( k_{n} \) from equation (55) and \( \mu \) from equation (54b), one can use equation (45) to compute \( \nu \). This value, in turn, can be used to index into the tables to determine the Prompt PRN code sign 29 (FIGS. 2D and 3A), \( x_{\nu} \), the early-minus-late PRN code sign 29 (FIGS. 2D and 3A), \( x_{\nu_{t}} \), and the early-minus-late PRN code sign 35 (FIGS. 2D and 3A), \( x_{\nu_{t}} \), that correspond to data word \( W \).

The conditional statements in equations (54a) and (54b) can be reduced to a single conditional per word data normal to improve efficiency. This can be done because the sign of \( \Delta k_{p} \) in equation (53a) is fixed for a given accumulation interval. (Normally the sign of \( \Delta k_{p} \) does not vary from accumulation interval to accumulation interval or from channel to channel for a given receiver because the only variable quantity that affects \( \Delta k_{p} \) is actual chip length \( 65 \) (FIG. 2D), \( \Delta f_{c} \) which normally does not vary significantly.) If \( \Delta k_{p} > 0 \), then the proper formula for determining \( k_{n} \) and \( \mu \) can be chosen by considering the inequality \( k_{n0} < k_{f_{p}} \). Conversely, if \( \Delta k_{p} < 0 \), then the proper formula can be determined by considering the inequality \( k_{n0} > k_{f_{p}} \). The decision about which condition to check can be made at the beginning of the accumulation because \( \Delta k_{p} \) is calculated prior to execution of the iteration in equations (53a)-(56).

When using a processor that creates instruction pipelines, “if” statements can disrupt the pipeline. In this case equations (54a) and (54b) can be replaced with the following computations:

\[
\eta_{v} = \begin{cases} 
\min[0, \text{sign}(k_{\text{from}} - k_{\text{fin}})] \Delta k_{p} & < 0 \\
0 & = 0 \\
\max[0, \text{sign}(k_{\text{from}} - k_{\text{fin}})] \Delta k_{p} & > 0 
\end{cases}
\]

for \( v = 2, 3, 4, \ldots, N \) (57a)

\[
\nu_{v} = k_{\text{from}} + \eta_{v} \Delta k_{p} \text{ for } v = 2, 3, 4, \ldots, N \text{ } (57b)
\]

\[
\mu_{v} = \nu_{v} - \nu_{\text{from}} \text{ for } v = 2, 3, 4, \ldots, N \text{ } (57c)
\]

The \( \min(\cdot) \) and \( \max(\cdot) \) functions return, respectively, the minimum or maximum of their two input arguments. The variable \( \eta_{v} \) is normally zero, in which case equations (57b) and (57c) leave \( k_{v} \) equal to \( k_{\text{from}} \) and \( \mu_{v} \) equal to \( \mu_{\text{from}} \). The value of \( \eta_{v} \) is \(-1\) if \( \Delta k_{p} < 0 \) and \( k_{\text{from}} < k_{\text{fin}} \) and \(+1\) if \( \Delta k_{p} > 0 \) and \( k_{\text{from}} > k_{\text{fin}} \). In both of these cases \( \eta_{v} \) causes equation (57b) and (57c) to perform the necessary adjustments to \( k_{v} \) and \( \mu_{v} \). Note that efficient code may not execute the conditional in equation (57a) once per data word. Instead, its accumulation iterations could be performed in one of three different iterative loops, depending on the value of \( \Delta k_{p} \). Additional economies can be had in the first and third conditional clauses of equation (57a). The value of \( -\eta_{v} \) for the first condition is equal to the sign bit of the 2’s compliment representation of \( k_{\text{from}} \). Similarly, \( +\eta_{v} \) for the third condition is equal to the sign bit of the 2’s compliment representation of \( k_{\text{from}} \). In either case, \( \eta_{v} \) (or its negative) can be computed in two operations.

Summarizing real-time over-sampled PRN code generator 30A (FIG. 3A) and referring now to FIG. 7, to compute prompt PRN code 29 (FIG. 3A) and early-minus-late PRN code 35 (FIG. 3A) for an entire accumulation interval, the method includes the steps of iterating equations (44a)-(44c) (method step 201, FIG. 7) to construct the table of \( \Delta(\mu) \) values. The method further includes the step of computing the auxiliary constants (method step 203, FIG. 7) in equations (48a)-(48b) and (50a)-(50c). The method further includes the step of initializing \( \eta_{v} \) and \( \mu_{v} \) (method step 205, FIG. 7) by evaluating equations (51a)-(52b). The method further includes the step of iterating equations (53a), (53b), (53c), (49), (55), and (45) (method step 207, FIG. 7) to compute, for each iteration, \( k_{\text{from}} \) \( \mu_{\text{from}} \) \( \eta_{v} \) \( \mu_{v} \) \( k_{v} \) and \( \nu \). The method further includes the step of iterating equations (56a)-(56c) (method step 209, FIG. 7) to compute, for each iteration, \( \nu_{v} \) \( \mu_{\text{from}} \) \( \eta_{v} \) and \( \nu_{v} \mu_{v} \). As mentioned above, it may prove efficient to interleave the equations (44a-c) and the accompanying shift register iterations between the iterations that compute \( k_{\text{from}} \) through \( \nu_{v} \). In this scenario \( \mu \) can be computed from equation (57c). Afterwards, the shift register iterations that generate code chips \( C(\mu_{v},-1) \) though \( C(\mu_{v},-1) \) can be performed, and these chip values can be used to iterate equations (44a-c) from \( \mu_{v} \) to \( \mu_{v} \) in order to determine \( \Delta(\mu_{v}) \) from \( \Delta(\mu_{v}) \).

The software correlator 19 (FIG. 1) of the present invention can advantageously be easily modified to work with signals at different frequencies, new PRN codes, or even signals for different types of devices. Thus, the same hardware could use the software correlator 19 (FIG. 1) to implement such devices as a GPS receiver, a cell phone, or both. To allow for new codes, new frequencies, and new types of functionality, small changes can be made in the software correlator 19 (FIG. 1), or different versions of the software correlator 19 (FIG. 1) can be run on the same processor. The changes involve using a different baseband mixing frequency and a different PRN code in the correlation, and perhaps changes that would provide the new signals of interest to the software correlator 19 (FIG. 1). In order for the present invention to work with signals at different frequencies, new PRN codes, or signals for different devices, two fundamental changes need to be made. First, the baseband mixing frequency must be tailored to that of the signal data 21, which also involves pre-computing and storing sine and cosine tables at this new frequency. Second, new pre-computed over-sampled PRN code tables 28 (FIG. 3A) must be constructed. The size of the new tables should match the over-sampled accumulation period, or at least one over-sampled period of the PRN code. As an alternative to generating new pre-computed over-sampled PRN code tables 28 (FIG. 3A), the new PRN codes can be generated in real-time by over-sampled PRN code generator 30A (FIG. 3A). Also, the system and method of the present invention could be implemented within systems such as GLONASS receivers, cell phones and cell base stations, pagers, wireless
Ethernet (e.g. 802.11x standards), Bluetooth™, BlackBerry® wireless internet devices, and satellite radio/phones (e.g. INMARSAT®). In fact, the system and method of the present invention are applicable to any sort of telecommunication system/device that uses spread spectrum, code division multiple access (CDMA) pseudo random number codes for the transmission of information, either wired or wireless.

Referring now to FIG. 8, navigation calculations require measured values of the PRN code phase 55, carrier phase, and carrier frequency. The measurements for all tracked satellites must be taken at exactly the same time. A time interval counter (TIC) function provides a periodic timing scheme to synchronize these measurements at time TIC 56.

At time TIC 56, the TIC function latches all of the PRN code phase 55, carrier phases, and carrier frequencies along with the code epoch counters, and software correlator 19 (FIG. 1) makes these available to application-specific code 15 (FIG. 1), for example, GPS receiver software. GPS receiver software uses the code phase and epoch counters to compute the pseudo range to each satellite. Software correlator 19 (FIG. 1) keeps track of the code and carrier phase of each signal as determined by the code chipping rate and the carrier Doppler shift inputs. The quantity ϕJk is the estimated code chipping rate of software receiver 10 for satellite j during its kth PRN code period, which can be determined either by an acquisition search procedure, or if tracking, by a delay-locked loop. Likewise, ϕ0, ϕDopp, the associated carrier Doppler shift, can be defined by an acquisition procedure or, if tracking has commenced, by a phase-locked loop or a frequency-locked loop. These determinations are made by application-specific code 15 (FIG. 1). Software correlator 19 (FIG. 1) can use these two frequencies to update quantities that keep track of its code and carrier phases according to the formulas:

\[ \hat{\varphi}_{J,k+1} = \hat{\varphi}_J + \frac{1023}{f_{code}} \]  

where is the carrier phase at time t, is the carrier phase at time t, is the code phase at time t, and \( f_{code} \) is the code chipping rate.

The epoch counters, which are simply a running total of the number of code periods TIC 55, are incremented at each code start/stop time. The carrier phase calculation at time TIC 56 (FIG. 8) is similar to the PRN code phase 55 (FIG. 8) calculation:

\[ \hat{\varphi}_{J,k} = \hat{\varphi}_{J,k-1} + \hat{\varphi}_{Dopp,k} + 1023(TIC - TIC_{k-1}) \]

where \( \hat{\varphi}_{J,k} \) is the carrier phase at time t, and \( \hat{\varphi}_{Dopp,k} \) is the Doppler shift that gets returned at time TIC 56 (FIG. 8). The epoch counters, which are simply a running total of the number of code periods TIC 55, are incremented at each code start/stop time.

Two comparison tests illustrate the performance of the system and method of the present invention. In the first test, a first configuration includes a MITEL® GP2021 hardware correlator, but is in all other ways identical to a second

<table>
<thead>
<tr>
<th>CH</th>
<th>SV</th>
<th>ELV</th>
<th>AZI</th>
<th>DOPP</th>
<th>NCO</th>
<th>UERE</th>
<th>SF</th>
<th>PRRr</th>
<th>PRerr</th>
<th>LOCK</th>
<th>SNR</th>
<th>iS4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>67</td>
<td>327</td>
<td>-1644</td>
<td>-528</td>
<td>-927</td>
<td>4</td>
<td>1</td>
<td>9.0</td>
<td>0.7</td>
<td>CCBF</td>
<td>16.9</td>
<td>-1.000</td>
</tr>
<tr>
<td>2</td>
<td>64</td>
<td>47</td>
<td>-650</td>
<td>-2045</td>
<td>4</td>
<td>1</td>
<td>14.3</td>
<td>0.6</td>
<td>CCBF</td>
<td>18.0</td>
<td>-1.000</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>59</td>
<td>14</td>
<td>2174</td>
<td>1550</td>
<td>2</td>
<td>1</td>
<td>6.5</td>
<td>1.0</td>
<td>CCBF</td>
<td>18.4</td>
<td>-1.000</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>22</td>
<td>106</td>
<td>-2650</td>
<td>-3050</td>
<td>2</td>
<td>1</td>
<td>42.9</td>
<td>0.2</td>
<td>CCBF</td>
<td>13.2</td>
<td>-1.000</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>17</td>
<td>1</td>
<td>170</td>
<td>1331</td>
<td>1</td>
<td>1</td>
<td>7.5</td>
<td>-0.4</td>
<td>CCBF</td>
<td>8.1</td>
<td>-1.000</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>81</td>
<td>2278</td>
<td>1887</td>
<td>2</td>
<td>1</td>
<td>5.2</td>
<td>-0.7</td>
<td>CCBF</td>
<td>7.4</td>
<td>-1.000</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>27</td>
<td>205</td>
<td>2969</td>
<td>2575</td>
<td>0</td>
<td>1</td>
<td>0.0</td>
<td>1.0</td>
<td>CCBF</td>
<td>8.0</td>
<td>-1.000</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>34</td>
<td>303</td>
<td>1856</td>
<td>1866</td>
<td>2</td>
<td>0</td>
<td>0.0</td>
<td>0.0</td>
<td>CCB</td>
<td>14.9</td>
<td>-1.000</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>22</td>
<td>185</td>
<td>3860</td>
<td>3464</td>
<td>2</td>
<td>1</td>
<td>1.0</td>
<td>-0.5</td>
<td>CCBF</td>
<td>15.9</td>
<td>-1.000</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>18</td>
<td>1200</td>
<td>1311</td>
<td>1</td>
<td>1</td>
<td>1.0</td>
<td>-0.5</td>
<td>CCBF</td>
<td>15.9</td>
<td>-1.000</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>18</td>
<td>1200</td>
<td>1311</td>
<td>1</td>
<td>1</td>
<td>1.0</td>
<td>-0.5</td>
<td>CCBF</td>
<td>15.9</td>
<td>-1.000</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>4</td>
<td>219</td>
<td>-3086</td>
<td>-3483</td>
<td>2</td>
<td>1</td>
<td>27.5</td>
<td>0.3</td>
<td>CCBF</td>
<td>9.6</td>
<td>-1.000</td>
<td></td>
</tr>
</tbody>
</table>

Software correlator 19 (FIG. 1) can keep a running track of these quantities and can initialize these iterations as part of the signal acquisition calculations that it carries out in conjunction with application-specific code 15 (FIG. 1). The quantities \( \hat{\varphi}_J \) and \( \hat{\varphi}_{Dopp} \) are either sent to software correlator 19 by application-specific code 15, or they are initialized arbitrarily by software correlator 19 and application-specific code 15 executes feedback control of \( \hat{\varphi}_J \) and \( \hat{\varphi}_{Dopp} \) to force the sequences defined by equations (58) and (59) to converge to appropriate values. Information about the previously-described conventional method can be found in Dierendorf.

The TIC time TIC 56 (FIG. 8) can occur at, for example, the millisecond boundaries of the receiver clock. At each time TIC 56, the PRN code phase 55 (FIG. 8) of each signal is computed in the following manner:

\[ \hat{\varphi}_{J,k} = 1023(TIC - TIC_{k-1}) \]

where \( \hat{\varphi}_{J,k} \) is the PRN code phase 55 (FIG. 8) in chips of signal j at TIC time TIC 56 (FIG. 8). The epoch counters, which are simply a running total of the number of code periods TIC 55, are incremented at each code start/stop time.

The quantity \( \hat{\varphi}_{J,k} \) is computed in the following manner:

\[ \hat{\varphi}_{J,k} = 1023(TIC - TIC_{k-1}) \]

(FIG. 8).
configuration that includes the software correlator 19 (FIG. 1) of the present invention. The two configurations differ in SNR by less than 1 dB and in navigation solutions by no more than 5-10 meters. In the second test, timing studies using the system of the present invention show that processing six channels uses only about 20% of the processor’s capacity, while Akos 2001a report a real-time software GPS receiver that would require 100% of the capacity at 1.73 GHz microprocessor to implement a 6-channel GPS receiver when processing data from an RF front-end with a sampling frequency of 5.714 MHz.

Referring now to FIGS. 9A and 9B, among other indicators that could assess the accuracy of the PRN code generated by real-time over-sampled PRN code generator 30A (FIG. 3A), which includes prompt PRN code sign 35A (FIG. 3A), early-minus-late PRN code sign 35A (FIG. 3A), and early-minus-late PRN code zero mask 35B (FIG. 3A), is the low distortion of the generated codes versus the true codes. Fig. 9A, generated for prompt code comparisons, shows juxtaposed plots of the autocorrelation function 80 (FIG. 9A) of the sampled true code and the cross correlation function 82 (FIG. 9A) between the sampled true code and the sampled code as generated from the new \( \chi \) \text{e} table. FIG. 9B shows a similar comparison for early-minus-late PRN code 35 (FIG. 3A). In either case, the nominal chipping frequency \( f_{\text{ch}} \) = 1.023 MHz, the sampling frequency \( f_s \) = 12.199 MHz, and the code timing resolution of the table is \( m=12 \), i.e., \( \frac{1}{12} \text{s} \) of a sample. This resolution translates into approximately \( \frac{1}{12} \text{s} \) of a code chip. The actual chipping rate differs from the nominal by \( f_s/f_{\text{ch}} = 1.5 \text{ Hz} \). This Doppler shift of the code chipping rate corresponds to a significant non-zero range rate between the receiver and the transmitter, 438 m/sec. The correlations are accumulated over 10230 code chips, and the code is a time-multiplexed version of a pair of the new GPS civil 1.2 CL and CM codes. The data word indexing calculations for this example use a code chip start time resolution of \( \Delta t/10^3 \), i.e., \( m=10^3 \).

It is obvious from FIGS. 9A and 9B that prompt PRN code 29 (FIG. 3A) and early-minus-late PRN code 35 (FIG. 3A) distortion is very small. The correlations produced using the new tables are virtually identical to those produced from the exact code. The low distortion of the new approach is best characterized by two parameters: the amount by which the correlation peak of the prompt PRN code sign 35A (FIG. 3A) drops below one and the offset of the zero-crossing time of the early-minus-late PRN code 35 (FIG. 3A). The former metric characterizes the power loss of the new approach, and the latter metric characterizes the net timing error. The drop of the prompt peak is only 0.3% of the nominal amplitude, which translates into a 0.03 dB loss. The timing distortion of the new code is less than \( 4 \times 10^{-4} \) code chips. This distortion is very small; it translates into about 0.1 m of GPS range measurement error.

It may seem paradoxical that the code timing error is only \( 4 \times 10^{-4} \) code chips when the code timing granularity of the \( \chi(\text{i}) \) tables is \( \Delta t/(\text{m} \Delta t) = 7 \times 10^{-3} \) code chips. The resolution of this paradox lies in the averaging effect of the accumulations. The length of a code chip equals 143.099269 code offset time grid intervals for the example shown in FIGS. 9A and 9B. The non-integer nature of this number causes the code offset errors of the \( \chi(\text{i}) \) tables to get dithered as the accumulation works its way through successive data words. This dithering tends to average out the table granularity errors, and this averaging can reduce the net timing error by an order of magnitude or more, as shown in FIG. 9B.

Although the invention has been described with respect to various embodiments, it should be realized this invention is also capable of a wide variety of further and other embodiments.

What is claimed is:

1. A method for computing prompt and early-minus-late in-phase and quadrature summed accumulations for a plurality of signals from a plurality of channels comprising the steps of:
   - representing a carrier replica signal from the at least one channel from the plurality of channels as a carrier replica sign and a carrier replica magnitude;
   - representing signal data from the at least one channel of the plurality of channels as at least one signal word;
   - computing a baseband mixed sign as a function of the carrier replica sign and at least one signal word;
   - computing a baseband mixed magnitude as a function of the carrier replica magnitude;
   - selecting a pseudo-random number (PRN) code having a prompt PRN code sign and an early-minus-late PRN code sign;
   - computing a fully mixed prompt integrand sign as a function of the baseband mixed sign and the prompt PRN code sign;
   - computing the early-minus-late PRN code sign as an early-minus-late PRN code sign and an early-minus-late PRN code zero mask;
   - computing a fully mixed early-minus-late integrand sign as a function of the baseband mixed sign and the early-minus-late PRN code sign;
   - computing at least one set of prompt integrand value words as a function of the fully mixed prompt integrand sign and the baseband mixed magnitude;
   - computing at least one set of early-minus-late integrand value words as a function of the fully mixed early-minus-late integrand sign, the baseband mixed magnitude, and the early-minus-late PRN code zero mask;
   - computing prompt in-phase and quadrature summed accumulations for the plurality of channels as an accumulation interval as functions of the number of significant bits in the at least one set of prompt integrand value words and as functions of the values associated with the at least one set of prompt integrand value words; and computing early-minus-late in-phase and quadrature summed accumulations for the plurality of channels as functions of the number of significant bits in the at least one set of early-minus-late integrand value words and as functions of the values associated with the at least one set of early-minus-late integrand value words; and
   - supplying the prompt and early-minus-late in-phase and quadrature summed accumulations to a software receiver to compute navigation data.

2. The method as in claim 1 further comprising the step of:
   - retrieving the carrier replica signal from a carrier replica table, the carrier replica table representing a coarse grid of frequencies.

3. The method as in claim 1 further comprising the steps of:
   - representing the signal word from the at least one channel as a signal sign and a signal magnitude; and
   - computing at least one baseband mixed magnitude as a function of the carrier replica magnitude and the signal magnitude.
4. The method as in claim 3 further comprising the step of: retrieving the carrier replica signal from a carrier replica table, the carrier replica table representing a coarse grid of frequencies.

5. The method as in claim 1 further comprising the steps of:
   - receiving at least one radio frequency (RF) signal from the at least one channel from the plurality of channels;
   - digitizing the at least one RF signal; and
   - mixing the at least one RF signal to form signal data using bit-wise parallelism.

6. The method as in claim 5 wherein the at least one RF signal is a multi-bit signal.

7. The method as in claim 5 further comprising the steps of:
   - down-converting the at least one RF signal to an intermediate frequency; and
   - digitizing the intermediate frequency.

8. The method as in claim 5 further comprising the step of: receiving the at least one RF signal from a global position source.

9. The method as in claim 1 wherein said step of computing a fully mixed prompt integrand sign is performed using bit-wise parallelism.

10. The method as in claim 1 wherein said step of computing fully mixed early-minus-late integrand sign and is performed using bit-wise parallelism.

11. The method as in claim 1 further comprising the step of:
    - rotating the in-phase and quadrature summed accumulations to correct for effects of frequency and phase granularity of the signal data.

12. The method as in claim 1 further comprising the step of: computing navigation data using the prompt in-phase and quadrature summed accumulations and the early-minus-late in-phase and quadrature summed accumulations.

13. The method as in claim 1 further comprising the step of:
    - retrieving the carrier replica signal from a carrier replica table, the carrier replica table representing a coarse grid of frequencies.

14. The method as in claim 1 wherein said step of computing a baseband mixed magnitude comprises the steps of:
    - representing the at least one signal word as a signal sign and a signal magnitude; and
    - computing the baseband mixed magnitude as a function of the carrier replica magnitude and the signal magnitude.

15. The method as in claim 1 further comprising the step of:
    - generating the PRN code using the bit-wise parallelism according to the steps of:
      - formulating a tabulated function for use in translating code chip and timing values into PRN code using the bit-wise parallelism;
      - generating at least one prompt PRN code in real-time; choosing at least one chip value from the at least one prompt PRN code, the at least one chip value corresponding to at least one data interval that contains at least one sample of a data word, the at least one chip value having a known timing relative to the at least one data interval;
      - transforming the known timing into a time grid index; and
      - translating the at least one chip value and the time grid index in the at least one data interval into the PRN code using the bit-wise parallelism for the at least one data interval, said step of translating resulting from the use of the tabulated function.

16. The method as in claim 15 further comprising the step of: computing the time grid index as a function of a time offset index and an auxiliary table index.

17. The method as in claim 15 further comprising the step of:
    - computing the time grid index iteratively as a function of a previously-computed time grid index, the at least one prompt PRN code, and the timing values associated with the at least one prompt PRN code.

18. A node in a computer network capable of carrying out the method according to claim 1.

19. A communications network comprising at least one node for carrying out the method according to claim 1.

20. The method of claim 1 wherein said computing prompt and early-minus-late in-phase and quadrature summed accumulations for a plurality of signals from a plurality of channels is performed by a computer system receiving electromagnetic signals traveling over a computer network conveying information capable of causing a computer system in the network to perform said computing of prompt and early-minus-late in-phase and quadrature summed accumulations for a plurality of signals from a plurality of channels.

21. A computer readable medium having instructions embodied therein for the practice of the method of claim 1.

22. A method for generating over-sampled prompt and early-minus-late pseudo-random number (PRN) codes in a bit-wise parallel format comprising the steps of:
    - formulating a tabulated function for use in translating code chip and timing information into over-sampled prompt and early-minus-late PRN code in the bit-wise parallel format;
    - generating at least one prompt PRN code in real-time;
    - choosing at least one chip value from the at least one prompt PRN code, the at least one chip value corresponding to at least one data interval that contains at least one sample of a data word, the at least one chip value having a known timing relative to the at least one data interval;
    - transforming the relative timing into a time grid index; and
    - translating the at least one chip value and the time grid index during the at least one data interval into the PRN code using the bit-wise parallelism for the at least one data interval, said step of translating resulting from the use of the tabulated function; and
    - distinguishing a signal and computing its PRN code phase by correlating the signal with the over-sampled prompt and early-minus-late PRN codes in bit-wise parallel format for the at least one data interval, said step of translating resulting from the use of the tabulated function; and
    - transforming the relative timing into a time grid index; and
    - translating the at least one chip value and the time grid index during the at least one data interval into the over-sampled prompt and early-minus-late PRN codes in bit-wise parallel format for the at least one data interval, said step of translating resulting from the use of the tabulated function; and
    - distinguishing a signal and computing its PRN code phase by correlating the signal with the over-sampled prompt and early-minus-late PRN codes in bit-wise parallel format.

23. The method as in claim 22 further comprising the step of:
    - computing the time grid index as a function of a time offset index and an auxiliary table index.

24. The method as in claim 22 further comprising the step of:
    - computing the time grid index iteratively as a function of a previously-computed time grid index, the at least one prompt PRN code, and the timing values associated with the at least one prompt PRN code.
25. A method for using over-sampled prompt and early-minus-late pseudo-random number (PRN) code replica data words that are stored in a bit-wise parallel representation in a pre-computed table consisting of the steps of:

- selecting the over-sampled prompt and early-minus-late PRN code based on over-sampled prompt and early-minus-late PRN code start time as measured relative to an RF data sample time, said step of selecting substantially matching the midpoint of the over-sampled prompt and early-minus-late PRN code with a desired PRN code midpoint; and
- bit-shifting the over-sampled prompt and early-minus-late PRN code data words, said step of bit-shifting insuring that the over-sampled prompt and early-minus-late PRN code start time corresponds with a pre-selected sample interval; and
- distinguishing a signal associated with the RF data and computing its PRN code phase based on a correlation between the signal and the over-sampled prompt and early-minus-late PRN code.

26. The method of claim 1 further comprising the step of:

- tracking the phase of the PRN code to track the timing of its chips including the steps of:
  - latching PRN code phase, carrier phase, epoch counters, and carrier frequencies at a pre-specified time;
  - computing a pseudo range using the PRN code phase and the epoch counters;
  - tracking and updating the PRN code phase and the carrier phase by estimating code chipping rate and carrier Doppler shift inputs; and
  - computing the PRN code phase at the pre-specified time as a function of the updated code chipping rate and the pre-specified time.

* * * * *