REAL-TIME SOFTWARE RECEIVER

Inventors: Brent M. Ledvina, Ithaca, NY (US); Mark L. Pslak, Brocktondale, NY (US); Steven P. Powell, Ithaca, NY (US); Paul M. Kintner, Jr., Ithaca, NY (US)

Assignee: Cornell Research Foundation, Inc., Ithaca, NY (US)

Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

Appl. No.: 10/753,927
Filed: Jan. 8, 2004

Prior Publication Data

Related U.S. Application Data
Provisional application No. 60/439,391, filed on Jan. 10, 2003.

Int. Cl. H03K 9/00 (2006.01)
Field of Classification Search 375/316, 375/147, 375/343, 455/130
See application file for complete search history.

References Cited
U.S. PATENT DOCUMENTS
FOREIGN PATENT DOCUMENTS

OTHER PUBLICATIONS

ABSTRACT
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.

27 Claims, 14 Drawing Sheets
OTHER PUBLICATIONS


* cited by examiner
FIG. 1

S/W Correlator w/ bit-wise parallel capability

Signal Tracking, Data Demodulation, Navigation Solution and User Interface

MICROPROCESSOR 16

Data Buffering and Acquisition System

Digitalizer

Mixer

Local Oscillator

RF FRONT END

Antenna
DATA BUFFERING AND ACQUISITION SYSTEM 17

FIG. 2A

SOFTWARE CORRELATOR 19

FIG. 2B
Quantity | Sequence
---|---
Sample Times | $t_0$, $t_1$, $t_2$, $t_3$, $t_4$, $t_5$, $t_6$, $t_7$
RF Signal | 1 1 -1 -1 -1 1 i 1
Word Representation of Signal | 1 1 0 0 0 1 1 1 73
PRN Code replica | 1 1 -1 -1 1 1 1 75
Word Representation of PRN Code replica | 1 0 0 0 1 1 1 1
Product of Signal and PRN Code replica | 1 -1 1 1 -1 1 1 1
Word Representation of Product | 0 1 0 0 1 0 0 0

FIG. 2C
FIG. 2D
FIG. 3A

- Baseband mixed sign HIMAG LOMAG 23A 23B 23C 23D
- Fully mixed prompt INT HIGH MAG LOW MAG 31A 31B 31C 31D
- Early-minus-late INT HIGH MAG LOW MAG ZERO MASK 33A 33B 33C 33D
- Early-minus-late PRN code 35A 35B
- Real-time over-sampled code generator 39A
- Pre-computed over-sampled code table 28

Connections:
- Signal MAG 21A XOR 21B
- Carrier replica 25A XOR 25B
- Prompt PRN code 29A XOR 29B
- XOR 31 33 35
FIG. 3B
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.

DE-SPREAD BY MIXING IN-PHASE AND QUADRATURE BASEBAND MIXED SIGNALS 23 WITH PROMPT PRN CODE 29 AND EARLY-MINUS-LATE PRN CODE 35: COMPUTE FULLY MIXED PROMPT INTEGRAND SIGN 31A = XOR (BASEBAND MIXED SIGN 23A, PROMPT PRN CODE SIGN 29A); COMPUTE FULLY MIXED PROMPT INTEGRAND MAGNITUDE 31B/C = \( f \) (BASEBAND MIXED MAGNITUDE 23B/C); COMPUTE FULLY MIXED EARLY-MINUS-LATE INTEGRAND SIGN 33A = XOR (BASEBAND MIXED SIGN 23A, EARLY-MINUS-LATE PRN CODE SIGN 35A); COMPUTE FULLY MIXED EARLY-MINUS-LATE INTEGRAND HIGH/LOW MAGNITUDE 33B/C = \( f \) (BASEBAND MIXED MAGNITUDE 23B/C); COMPUTE FULLY MIXED EARLY-MINUS-LATE ZERO MASK 33D = \( f \) (EARLY-MINUS-LATE PRN CODE ZERO MASK 35B); RESULT IS FULLY MIXED IN-PHASE AND QUADRATURE PROMPT INTEGRANDS 31 AND FULLY MIXED EARLY-MINUS-LATE INTEGRANDS 33.

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).

FIG. 4A
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 = \( f(\text{PROMPT INTEGRAND VALUE WORD 27}) \), COMPUTE EARLY-MINUS-LATE INTEGRAND VALUE-WORD ONE-BITS COUNTS = \( f(\text{EARLY-MINUS-LATE INTEGRAND VALUE WORD 37}) \). 

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 = \( \Sigma (\text{PROMPT VALUE-WORD ONE-BITS COUNTS}) \), SUM IS OVER ALL WORDS IN ACCUMULATION INTERVAL; COMPUTE EARLY-MINUS-LATE ACCUMULATIONS 49 = \( \Sigma (\text{EARLY-MINUS-LATE INTEGRAND VALUE WORD ONE-BITS COUNTS}) \), 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 = \( \Sigma (\text{VALUE 41A} \times [\text{PROMPT ACCUMULATION 41}]) \), SUM IS OVER ALL VALUES; SUMMED EARLY-MINUS-LATE ACCUMULATION 47 = \( \Sigma (\text{VALUE 41A} \times [\text{EARLY-MINUS-LATE ACCUMULATION 49}]) \), 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 $I$ Code Chips</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x(1)$</td>
<td>$\Delta t_{0\text{min}}$</td>
<td>$0\ 0 \ldots\ 0\ 0\ 0\ 0\ 0$</td>
</tr>
<tr>
<td>$x(2)$</td>
<td>$\Delta t_{0\text{min}}$</td>
<td>$0\ 0 \ldots\ 0\ 0\ 0\ 0\ 1$</td>
</tr>
<tr>
<td>$x(3)$</td>
<td>$\Delta t_{0\text{min}}$</td>
<td>$0\ 0 \ldots\ 0\ 0\ 0\ 1\ 0$</td>
</tr>
<tr>
<td>$x(4)$</td>
<td>$\Delta t_{0\text{min}}$</td>
<td>$0\ 0 \ldots\ 0\ 0\ 0\ 1\ 1$</td>
</tr>
<tr>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
</tr>
<tr>
<td>$x(2^L)$</td>
<td>$\Delta t_{0\text{min}}$</td>
<td>$1\ 1 \ldots\ 1\ 1\ 1\ 1\ 1$</td>
</tr>
<tr>
<td>$x(2^L+1)$</td>
<td>$\Delta t_{0\text{min}+1}$</td>
<td>$0\ 0 \ldots\ 0\ 0\ 0\ 0\ 0$</td>
</tr>
<tr>
<td>$x(2^L+2)$</td>
<td>$\Delta t_{0\text{min}+1}$</td>
<td>$0\ 0 \ldots\ 0\ 0\ 0\ 1\ 0$</td>
</tr>
<tr>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
</tr>
<tr>
<td>$x(2^L \cdot k_{\text{tot}})$</td>
<td>$\Delta t_{0\text{max}}$</td>
<td>$1\ 1 \ldots\ 1\ 1\ 1\ 1\ 1$</td>
</tr>
</tbody>
</table>

FIG. 5
FIG. 6
INITIALIZE TIME OFFSET INDEX $k_{fO}$ AND AUXILIARY TABLE INDEX $\mu_i$ BY EVALUATING EQUATIONS 51a-52b 205

COMPUTE $k_{fmin}$, $k_{fmax}$, $\eta_{f}$, $k_f$, $\mu$, $k_v$, and $i_v$ 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 $x_{p}$, $x_{emlzm}$, and $x_{eml2sv}$ 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 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 off-line. 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 must 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/-2 early-minus-late 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 software correlator computes 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 improved 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. 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
Positioning System Software Receiver (gpsrx) Implementa-
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 (gpsrx) Implementa-
tion 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
over-sampled signals by use of bit-wise parallelism.

A still further object of the present invention is to use very
long over-sampled 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 algo-
rithms, 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 over-
sampled 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 over-sampled versions of the prompt and early-
minus-late PRN codes in real-time through use of an over-
sampling function described herein. The values of the over-
sampling function can be located in a specially designed
table that can be generic across PRN codes. The length of the
specialy 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 indi-
ces 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 par-
allelism 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 math-
ematical correlation operations. If, for example, four accu-
mulations 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 embed-
iment, 33,500 bit-wise parallel operations are necessary per
PRN code period when the RF signal has a 2-bit represen-
tation. This operation count is further reduced to approxi-
ately 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 accumula-
tion 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,
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 inven-
tion 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 chipping 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 chipping
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 sys-
tems that must fuse GPS data with data from other types of
sensor systems.

The software correlator of the present invention can
advantageously be easily adapted to accept signals at any
frequency, new PRN codes, or even signals for 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 point 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 bit-wise parallel radio frequency signal and PRN code replica storage and mixing;

FIG. 2D 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 an 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 herein 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:

\[ y(t) = \sum_j A_j D_{jk} C_j \left[ 0.001 \left( \frac{t - \tau_{jk}}{\tau_{jk+1} - \tau_{jk}} \right) \right] \cos[\omega t + \phi_j(t)] + \sigma_j \]

where \( t \) is the sample time, \( A_j \) is the amplitude, \( D_{jk} \) is the navigation data bit, \( C_j \) is the C/A code, \( \tau_{jk} \) and \( \tau_{jk+1} \) are the start times of the received \( k^{th} \) and \( k+1^{st} \) C/A code periods, \( \omega \) is the intermediate frequency corresponding to the L1 carrier frequency, \( \phi_j(t) \) is the carrier phase perturbation due to accumulated delta range, \( n_i \) is the receiver noise, and the subscript \( j \) refers to a particular GPS satellite. The summation is over all visible GPS satellites. The negative sign in front of \( \phi_j(t) \) 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:

\[ y_p(t) = C_j \left[ 0.001 \left( \frac{t - \tau_{jk}}{\tau_{jk+1} - \tau_{jk}} \right) \right] \cos[\omega t + \delta \phi(t) + \delta \phi(t)] \]

\[ y_q(t) = C_j \left[ 0.001 \left( \frac{t - \tau_{jk}}{\tau_{jk+1} - \tau_{jk}} \right) \right] \sin[\omega t + \delta \phi(t) + \delta \phi(t)] \]

where equations (2) and (3) apply during the \( k^{th} \) C/A code period. In these equations \( \tau_{jk} \) and \( \tau_{jk+1} \) are the receiver’s
estimates of the start times of the kth and k+1st code periods, \( \phi_{ik} \) is the estimated carrier phase at time \( \tau_{ik} \), and \( \phi_{Dopp} \) is the estimated carrier Doppler shift during the kth code period.

A typical receiver computes the estimates \( \tau_{ik} \), \( \tau_{ip} \), \( \phi_{ik} \), and \( \omega_{Dopp} \) by various conventional means that are described in GPS Receivers, A. J. Van Dierendonck, Global Positioning System: Theory and Applications, B. W. Parkinson and J. J. Spilker, Jr., Eds., vol. I, American Institute of Aeronautics and Astronautics, 1996, Chapter 8, pp. 329-406

where \( \Delta \) is the time offset \( \Delta \) causes the replica PRN code to play back early if it 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

\[
I_{jk}(\Delta) = \sum_{i=0}^{N_k-1} \cos(\omega_0 t_i) \left[ \frac{1}{2} I_{jk} + \frac{1}{2} I_{jk} \right] 
\]

\[
Q_{jk}(\Delta) = \sum_{i=0}^{N_k-1} \sin(\omega_0 t_i) \left[ \frac{1}{2} I_{jk} - \frac{1}{2} I_{jk} \right] 
\]

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® GPS Architect soft ware ported to RTLINUX® (see A Coming of Age for GPS: A RTLINUX BASED GPS RECEIVER, Ledvina et al., Proceedings of the Workshop on Real Time Operating Systems and Applications and Second Real Time Linux 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±0.63)x10^6 Hz±1.4053968254 MHz and then performs analog-to-digital conversion. The resultant, digitized signal data 21 has a pre-determined 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 and 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 signal magnitude and sign data bits. Pertinent features of this card are the thirty-two digital output ports on the card.

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 RTLINUX®, 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 RTLINUX®, perhaps gotten from the suite of open Source 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™ proces-
A corollator 19 uses these frequency commands to reconstruct carrier replicas that are the most accurate versions of the 115 possible 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 replica tables are compensated for by post-processing calculations, as described below.

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

\[
I_{\text{in}}(\Delta) = \frac{1}{2} \sum_{n=-N}^{N} y[n]x[n]C_{\text{zero}} \left[ 0.001 \left( \frac{\Delta - \tau_{\text{in}}}{F_{x}} + \frac{\tau_{\text{in}}}{F_{x}} \right) \right] \cos \left[ \omega_{\text{in}}n \right]
\]

where the in-phase and quadrature, summed prompt accumulation and summed early-minus-late accumulation are equivalent to what would be produced by a continuously variable sine wave. These are provided to an application-specific code, 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 chip rate and can 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 a 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/CL 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:

\[
i_{\text{in}}(\Delta) = \frac{1}{2} \sum_{n=-N}^{N} y[n]x[n]C_{\text{zero}} \left[ 0.001 \left( \frac{\Delta - \tau_{\text{in}}}{F_{x}} + \frac{\tau_{\text{in}}}{F_{x}} \right) \right] \cos \left[ \omega_{\text{in}}n \right]
\]
where \( \omega_{g,\Delta} \) is the grid frequency that is closest to the estimated frequency \( \omega_{\text{upg}} \) and \( \Delta \omega \) is the time at which this carrier replica signal 25 (FIG. 3A) has zero carrier phase. Software correlator 19 rotates these accumulations 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:

\[
\begin{align*}
\Delta \Phi_{\text{avg}} &= \phi_{\text{avg}}(t) - \phi_{\text{avg}}(t - \Delta t), \\
\phi_{\text{avg}}(t) &= \phi_{\text{replica}}(t) + \phi_{\text{host}}(t).
\end{align*}
\]

where \( \Delta \Phi_{\text{avg}} \) is the average phase difference between the grid carrier phase and the estimated carrier phase averaged over the accumulation interval:

\[
\Delta \Phi_{\text{avg}} = \frac{\phi_{\text{replica}}(t) + \phi_{\text{host}}(t)}{2} - \frac{\phi_{\text{replica}}(t - \Delta t) + \phi_{\text{host}}(t - \Delta t)}{2} - \phi_{\text{host}}(t).
\]

Note that equations (8), (9), and (10) are an illustrative example of how software correlator (19) can rotate its I and Q accumulations in order to correct for phase and frequency errors in its table of pre-computed carrier replica signals. There exist other formulas that yield equivalent results, and this patent disclosure covers all such techniques.

The validity of equations (8) and (9) is dependent on the assumption that

\[
1 - \cos \left[ \frac{1}{2} (\omega_{g,\Delta} - \phi_{\text{upg}}(t) + \phi_{\text{upg}}(t - \Delta t)) \right] < 1
\]

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) and (9) can be derived from equations (4) and (5) as follows. First, the carrier phase of the grid signal in the arguments of the cosine and sine terms of equations (6) and (7) are added to and subtracted from the arguments of the cosine and sine terms in equations (4) and (5). Next, trigonometric identities are used to split the resulting cosine and sine terms into sums of products of cosine and sine functions. In each product, one of the terms involves an argument like the arguments in the trigonometric terms in equations (6) and (7). The other trigonometric terms are then approximated by either \( \cos(\Delta \Phi_{\text{avg}}) \) or \( \sin(\Delta \Phi_{\text{avg}}) \). These approximations are valid because of the inequality in equation (11) and because the average of \( \sin \left\{ \omega_{g,\Delta} - \omega_{\text{upg}} \right\} (t + \phi_{\text{upg}} + \phi_{\text{host}}) \) over the accumulation interval is zero.

A decrease in the carrier to noise ratio C/N, which characterizes the receiver’s sensitivity, is caused by the use of an inexact baseband mixing frequency. The worst-case decrease is expressed as a function of the frequency grid spacing \( \Delta f \) and is given by

\[
\Delta \text{SNR} = 20 \log_{10} \left( \frac{\sin(\pi \Delta f/T)}{\pi \Delta f/T} \right)
\]

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, loss of 0.11 dB for \( T=0.001 \) sec.

Referring now to 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-computed 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 computed based on, for example, the incoming RF signal data 21 (FIGS. 2A and 3A). Pre-computing can be most advantageously used when the PRN code is not very long. Generating PRN codes in real-time can be a more appropriate solution when the PRN codes are very long (and thus 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.

Continuing to refer primarily to FIGS. 2A, 2B, and 3A, in order to perform bit-wise parallel operations, software correlator 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 25 (FIGS. 2B and 3A) in terms of one, two, or more bits are suitable for using bit-wise parallelism to perform the calculations 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. Similarly, tables can store prompt PRN code 29 (FIG. 3A) and early-minus-late PRN code 35 (FIG. 3A), which are composed of prompt PRN code sign 29A (FIG. 3A), early-minus-late PRN code sign 35A (FIG. 3A), and early-minus-late PRN code zero mask 35B (FIG. 3A). The data words that comprise the bit-wise parallel representations of these three signal types, the original RF signal data 21 (FIGS. 2A and 3A), the carrier replica signal 25 (FIGS. 2B and FIG. 3A), and the de-spreading prompt PRN code 29 (FIG. 3A) and early-minus-late PRN code 35 (FIG. 3A), are the inputs to the calculations of software correlator 19 (FIG. 1).

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 time if microprocessor 16 (FIG. 1) has a bit-wise EXCLUSIVE OR command. Other bit-wise commands are used to perform additional software correlation operations in parallel on sets of two thirty-two samples.

At this point, the problem of over-sampling is introduced. Referring now to FIGS. 2C and 2D, the problem of over-sampling is illustrated with respect to bit-wise parallelism as follows. There is normally more than one RF data sample per PRN code chip. The three successive -1 values 73 (FIG. 2C) at sample times \( t_1 \) to \( t_3 \) all occur during the same PRN code chip as do the four successive +1 values 75 (FIG. 2C) at times \( t_4 \) through \( t_7 \). The difference in the number of samples for the two code chips arises because the PRN code chip period is not an integer multiple of the sample period. Analogously, referring to FIG. 2D, where sample interval \( \Delta t \) is less than actual PRN code chip length \( \Delta t_c \), over-sampling is indicated because the RF sampling frequency \( f_s = 1/\Delta t_c \) is greater than the PRN code chip frequency \( f_c = 1/\Delta t \). PRN codes for CDMA signaling are sequences of +1 and -1 values, the elements of which are chips. Over the time intervals of interest, a carrier replica progresses through its chips at a constant chipping rate of \( f_s = 1/\Delta t_c \) chips/second. The time interval \( \Delta t \) is the actual PRN code chip length 65 (FIG. 2D). Software correlator 19 (FIG. 1) normally receives PRN code and, attempts to align it with the prompt replica version of the code, prompt PRN code 29 (FIG. 2D). It makes use of the signal’s correlation with prompt PRN code 29 (FIG. 2D) and with early-minus-late PRN code 35 (FIG. 2D) in order to determine a chipping rate \( f_c \) that tends to align prompt PRN code 29 (FIG. 2D) as desired. Conventional methods for determining \( f_c \) are well-known in the art. Chips of early code 69B (FIG. 2D) start and stop 0.5\( \Delta t \) seconds before the corresponding chips of prompt PRN code 29 (FIG. 2D), and the chips of late code 69C (FIG. 2D) start and stop 0.5\( \Delta t \) seconds after prompt PRN code 29 (FIG. 2D). Early-minus-late PRN code 35 (FIG. 2D) is the difference between early code 69B (FIG. 2D) and later code 69C (FIG. 2D). Example segments of these four types of replica codes are depicted in FIG. 2D.

Referring to FIGS. 1, 2A, 2C, and 2D software correlator 19 (FIG. 1) receives, through conventional RF front end 13 and DAQ system 17, signal data 21 (FIG. 1), the raw data 12 (FIG. 1) source of which is sampled at the rate \( f_s = 1/\Delta t \) Hz. In order to process the resulting RF signal data 21, software correlator 19 (FIG. 1) needs prompt PRN code 29 (FIG. 2D) and early-minus-late PRN code 35 (FIG. 2D) replicas sampled at the same times as raw signal 12 (FIG. 1). FIG. 2D depicts sixteen sample times as vertical dash-dotted lines. Referring to FIG. 2D, prompt PRN code 29 (FIG. 2D) can be represented by its prompt PRN code sign 29A (FIG. 2D) at the sample times. The bit value one represents +1, and the bit value zero represents -1. Prompt PRN code sign 29A (FIG. 2D), shown at the sixteen sample times—starting with three 1s, continuing with ten 0s, and finishing with another three 1s—is a 16-bit word stored as the integer \( 2^5 \times 2^4 \times 2^3 \times 2^2 \times 2^1 \times 2^0 = 57351 \). Early-minus-late PRN code 35 (FIG. 2D) requires a 1.5-bit representation. A zero mask bit is set to zero if early-minus-late PRN code 35 takes on the value zero, and it is set to one if early-minus-late PRN code 35 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 \( 2^5 \times 2^4 \times 2^3 \times 2^2 \times 2^1 \times 2^0 = 57351 \). Early-minus-late PRN code sign 35A (FIG. 2D) is six 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 \( 2^4 = 4 \).

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 Signal, R. D. Fontana et al., Proceedings of the ION GPS 2001, Sep. 11-14, 2001, Salt Lake City, Utah, pp. 617-631). 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 can include integer values that constitute the bit-wise parallel representation of the PRN code for the sample times associated with the data word in question. Indices into each 1-dimensional table are functions of (a) the time offset between the first PRN code chip and the first sample time of the given data word, and (b) the bit pattern of the PRN code chips that span the sample times of the data word. The sizes of the tables are independent of the period of the PRN code that is being over-sampled. The tables can be re-used for multiple PRN codes in a multi-channel receiver. The computation and use of the tables are discussed in more detail later.

Continuing with the description of bit-wise parallelism with respect to the operations of software correlator 19 (FIG. 1), and continuing to refer to FIG. 3A, the specially-developed algorithms described herein make use of bit-wise parallelism so that a single programming language statement, such as a C code command, can partially process up to thirty-two samples at a time. Previously referred-to carrier replica signal 25 in the form of cosine and sine signals are stored as binary carrier replica sign 25A and carrier replica magnitude 25B. The format of this representation is defined in Table 2 and illustrated in FIG. 3C which is a reconstructed carrier and carrier replica in the form of representative sine signal 51 (FIG. 3C) shown in optimal 2-bit representation 53 (FIG. 3C) that has the minimum square error. The format of Table 2 assumes that the cosine and sine signals have an amplitude of approximately 2.4. Note that other representations, beyond 2-bit representation, are possible. In general, more bits yield a better SNR, but can also require a larger number of computations for the correlation operations.

TABLE 2

<table>
<thead>
<tr>
<th>Carrier Replica Sign 25A</th>
<th>Carrier Replica Magnitude 25B</th>
<th>Carrier Replica Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>-2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>+1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>+2</td>
</tr>
</tbody>
</table>

Continuing to refer to FIG. 3A, multiplication of the RF front-end output representation, the signal sign 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. Thus, these two magnitude bits are available without the need for computation. Baseband mixed sign 23A is the result of an EXCLUSIVE OR operation between signal sign 21A and carrier replica sign 25A. Notice how the relationship of the sign bit value with the actual sign gets reversed from that of Tables 1 and 2.

TABLE 3

<table>
<thead>
<tr>
<th>Prompt PRN Code Sign 29A</th>
<th>Prompt PRN Code Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>+1</td>
</tr>
<tr>
<td>0</td>
<td>-1</td>
</tr>
</tbody>
</table>

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.

TABLE 4

<table>
<thead>
<tr>
<th>Baseband Mixed Sign 23A</th>
<th>Baseband Mixed High Magnitude 23B</th>
<th>Baseband Mixed Low Magnitude 23C</th>
<th>Baseband Mixed 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>+2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>+3</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>

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 τ2=−τ′=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 \( (T_s + T_{pm}) / 2 \). 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 oversampled 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, \( W_5 \) (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, \( W_5 \) (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, \( W_5 \) (FIG. 6).

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. Fully mixed early-minus-late integrand sign 33A is an EXCLUSIVE OR between early-minus-late PRN code sign 35A and baseband mixed sign 23A. Fully mixed early-minus-late integrand high magnitude 33B and fully mixed early-minus-late integrand low magnitude 33C are, as above, baseband mixed high magnitude 23B and baseband mixed low magnitude 23C. Fully mixed early-minus-late integrand zero mask 33D is early-minus-late PRN code zero mask 35B. The resulting representation is given in Table 7. As in Table 5, each X entry in the table indicates that the corresponding bit can be either zero or one without affecting the corresponding integrand value.

### Table 6-continued

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

### Table 7

<table>
<thead>
<tr>
<th>Early-Minus-Late (EML) Sign 33A</th>
<th>EML High Magnitude 33B</th>
<th>EML Low Magnitude 33C</th>
<th>EML Zero Mask 33D</th>
<th>Early-Minus-Late Integrand Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>-2</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>-4</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>-6</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>-12</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>+2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>+4</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>+6</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>+12</td>
</tr>
</tbody>
</table>

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 computing in-phase and quadrature baseband mixed sign 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 value word 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 summing over each prompt integrand value word 27 and early-minus-late integrand value word 37 (FIG. 3B) the number of one bits (or zero bits) using one bits summation table 38 (FIG. 3B) or using a processor command if available (method step 111, FIG. 4B), and summing, over the 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 prompt accumulations 41 (FIG. 3B) and early-minus-late accumulations 49 (FIG. 3B) (method step 113, FIG. 4B). The method further includes the step of multiplying prompt accumulations 41 (FIG. 3B) and early-minus-late accumulations 49 (FIG. 3B) by corresponding values 41 and summing the results over the value words of each signal for an entire accumulation interval to yield in-phase and quadrature summed prompt accumulations 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 accumulations 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 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 steps 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.

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 MINUSSIX 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:

- MINUSONE=NOT(SIGN) AND NOT([HIGHMAG] AND NOT([LOWMAG])
- MINUSTWO=NOT(SIGN) AND [HIGHMAG] AND [LOWMAG]
- MINUSTHREE=NOT([SIGN] AND [HIGHMAG] AND [LOWMAG])
- MINUSSIX=NOT([SIGN] AND [HIGHMAG] AND [LOWMAG])
- PLUSONE=SIGN AND NOT([HIGHMAG] AND [LOWMAG])
- PLUSTWO=SIGN AND [HIGHMAG] AND [LOWMAG]
- PLUSTHREE=SIGN AND [HIGHMAG] AND NOT([LOWMAG])
- PLUSSIX=SIGN AND [HIGHMAG] AND [LOWMAG]

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 logic 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:

- MINUSONE=NOT([SIGN] AND NOT([HIGHMAG] AND NOT([LOWMAG])
- MINUSTWO=NOT([SIGN] AND [HIGHMAG] AND [LOWMAG])
- MINUSTHREE=NOT([SIGN] AND [HIGHMAG] AND [LOWMAG])
- MINUSSIX=NOT([SIGN] AND [HIGHMAG] AND [LOWMAG])
- PLUSONE=SIGN AND NOT([HIGHMAG] AND [LOWMAG])
- PLUSTWO=SIGN AND [HIGHMAG] AND [LOWMAG]
- PLUSTHREE=SIGN AND [HIGHMAG] AND NOT([LOWMAG])
- PLUSSIX=SIGN AND [HIGHMAG] AND [LOWMAG]

Also, a new 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:

- MINUSTWO=[ZEROMASK AND NOT(SIGN)] AND [NOT([HIGHMAG] AND NOT([LOWMAG])]
- MINUSFOUR=[ZEROMASK AND NOT(SIGN)] AND [NOT([HIGHMAG] AND [LOWMAG])]
- MINUSSIX=[ZEROMASK AND NOT(SIGN)] AND [HIGHMAG AND NOT([LOWMAG])]
- MINUSTWELVE=[ZEROMASK AND NOT(SIGN)] AND [HIGHMAG AND [LOWMAG]]
- PLUSTWO=[ZEROMASK AND SIGN] AND NOT([HIGHMAG] AND NOT([LOWMAG])]
- PLUSFOUR=[ZEROMASK AND SIGN] AND [NOT([HIGHMAG] AND [LOWMAG])]
- PLUSFOUR=[ZEROMASK AND SIGN] AND [HIGHMAG AND [LOWMAG]]
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 bit counts for each prompt integrand value word 27 (FIG. 3B) and for each early-minus-late integrand value words 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:

\[
I_{k1}(0) = \sum_{i=1}^{N_w} \text{ONESCOUNT}(-1)_{iet} + \sum_{i=1}^{N_w} \text{ONESCOUNT}(+1)_{iet} + \sum_{i=1}^{N_w} \text{ONESCOUNT}(+2)_{iet} + \sum_{i=1}^{N_w} \text{ONESCOUNT}(+3)_{iet} + \sum_{i=1}^{N_w} \text{ONESCOUNT}(+6)_{iet}
\]

where 1 is the index of successive bit-wise parallel data words in the accumulation interval, \(N_w\) is the total number of data words in the interval, and ONECOUNT(k)iet is the ones count for the corresponding value word 41 (FIG. 3B) associated with value k 41A (FIG. 3B) for the ith 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), \(\Delta t_s\), the sample interval 63 (FIG. 2D), \(\Delta t_{enl}\), the nominal early-to-late code delay 61 (FIG. 2D), \(\Delta t_{enl}\), the end time of the first code chip relative to the first sample time, or put another way, the time lag \(\Delta t_{enl}\), 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), \( \Delta t_i \), the nominal chip length, \( \Delta t_{\text{am}} \), the early-minus-late code delay 61 (FIG. 2D), \( \Delta t_{\text{am}} \), 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 \( \Delta t_i \) and the nominal chipping rate \( \Delta t_{\text{am}} \) (reciprocal of \( \Delta t_{\text{am}} \)) that is used for the above simplification can be accommodated by correcting time lag 67 (FIG. 2D), \( \Delta t_{\text{am}} \), for the average effects of Doppler shift, a procedure discussed later. Using the simplification, each look-up table has two variable inputs: the actual set of prompt code chips 91 (FIGS. 2D and 6) and time lag 67 (FIG. 2D), \( \Delta t_{\text{am}} \). 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 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 67 (FIG. 2D), \( \Delta t_{\text{am}} \), can take on an infinite number of values in the continuous range:

\[
-\frac{1}{2} \Delta t_{\text{end}} \leq \Delta t_{\text{am}} \leq \Delta t_i - \frac{1}{2} \Delta t_{\text{end}}
\]  

(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 67 (FIG. 2D) \( \Delta t_{\text{am}} \) 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 \( \Delta t_{\text{am}} \) would leave the late code 69C (FIG. 2D) at the first sample time undefined based on the available code chips.

Referring now to FIG. 5, to create an electronically processable table, the continuous range of \( \Delta t_{\text{am}} \) values can be replaced with a discrete grid having \( m \) equally spaced points per sample interval 63 (FIG. 2D), \( \Delta t_i \). The integer \( m \) is chosen to be large enough so that the granularity \( \Delta t_i/m \) gives sufficient PRN code timing resolution. In GPS applications \( m \) is usually chosen to be large enough so that \( c \Delta t_i/m \) 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:

\[
\Delta t_{\text{kth}} = \frac{k \Delta t_i}{m} \quad \text{for } k = k_{\text{min}}, \ldots, k_{\text{max}}
\]  

(32)

where the limits

\[
k_{\text{min}} = \text{floor}\left(-\frac{m \Delta t_{\text{end}}}{2 \Delta t_i}\right) - 2
\]  

(32a)

\[
k_{\text{max}} = \text{floor}\left(-\frac{m \Delta t_i}{2 \Delta t_{\text{end}}}\right)
\]  

(32b)

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 \( \Delta t_{\text{kth}} \) 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 \( \Delta t_{\text{am}} \) 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:

\[
l(\Delta t_{\text{am}}) = \text{floor}\left\{\frac{n_c - 1}{\Delta t_i} - \frac{1}{2} \cdot \frac{\Delta t_{\text{end}}}{\Delta t_i} + 2\right\}
\]  

(34)

where \( n_c \) 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 \( l(\Delta t_{\text{am}}) \) is a non-increasing function of \( \Delta t_{\text{am}} \).

Therefore, the maximum number of required chips occurs at the minimum value of \( \Delta t_{\text{am}} \):

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

(35)

The size of each table can be determined from the parameters \( k_{\text{min}}, k_{\text{max}}, \) and \( L \). The grid contains \( k_{\text{min}}, k_{\text{max}} + 1 \) 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 \( k_{\text{max}} \times 2^L \) 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 \( \Delta t_{\text{am}} = \Delta t_{\text{kth}} \) where the next \( 2^L \) entries correspond to \( \Delta t_{\text{am}} = \Delta t_{\text{am}}(k_{\text{min}}+1) \) and so forth. The tabulated bit sequences for a fixed \( \Delta t_{\text{am}} \) are ordered by interpreting the sequence as a binary index counter with the first chip being the most significant 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 \( \Delta t_{\text{kth}} \), 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 \( \Delta t_{\text{kth}} \) grid index \( k \) and its corresponding bit sequence 85. The corresponding bit sequence 85 associated with the array index consists of the chip values \( C(1), C(2), C(3), \ldots, C(L) \). 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:

\[
l(k, C(1), C(2), C(3), \ldots, C(L)) = 1 + (k - k_{\text{min}}) \times 2^L \sum_{j=1}^{L} C(j)2^{L-j}
\]  

(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:

\[
(\delta, k_{\text{min}}) = \text{floor}(L/2^L)
\]  

(37a)
where \( C_{(n,i)} \) is the over-sampled prompt PRN code and \( n \) is the index of the sample time within the over-sampled segment of the data word. These indices, in turn, can be used to determine the chip values that apply at sample times:

\[
\begin{align*}
\bar{c}(n, i) &= \text{floor}\left( \frac{(m+1)(n-1)}{2^s}\right) \quad \text{for } j = 1, 2, 3, \ldots, L
\end{align*}
\]  

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 of the three tables. Given \( i \), the corresponding code time offset \( \Delta t \) is computed from equation (37a) and is used to generate the \( x(i) \) table elements:

\[
\begin{align*}
\bar{c}(n, i) &= \text{floor}\left( \frac{(m+1)(n-1)}{2^s}\right) \quad \text{for } j = 1, 2, 3, \ldots, L
\end{align*}
\]  

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

where \( n \) is the index of the sample time within the over-sampled data word. The integer \( j(n,i) \) is the index of the PRN code chip that applies at sample \( n \) for the prompt PRN code \( \text{C}(n,i) \). The integers \( j(n,i) \) and \( k(n,i) \) are defined as follows for prompt PRN code \( \text{C}(n,i) \):

\[
\begin{align*}
\bar{c}(n, i) &= \text{floor}\left( \frac{(m+1)(n-1)}{2^s}\right) \quad \text{for } j = 1, 2, 3, \ldots, L
\end{align*}
\]  

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

where \( n \) is the index of the sample time within the over-sampled segment of the data word. These indices, in turn, can be used to determine the chip values that apply at sample times:

\[
\begin{align*}
\bar{c}(n, i) &= \text{floor}\left( \frac{(m+1)(n-1)}{2^s}\right) \quad \text{for } j = 1, 2, 3, \ldots, L
\end{align*}
\]  

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

where \( n \) is the index of the sample time within the over-sampled segment of the data word. These indices, in turn, can be used to determine the chip values that apply at sample times:

\[
\begin{align*}
\bar{c}(n, i) &= \text{floor}\left( \frac{(m+1)(n-1)}{2^s}\right) \quad \text{for } j = 1, 2, 3, \ldots, L
\end{align*}
\]  

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

where \( n \) is the index of the sample time within the over-sampled segment of the data word. These indices, in turn, can be used to determine the chip values that apply at sample times:

\[
\begin{align*}
\bar{c}(n, i) &= \text{floor}\left( \frac{(m+1)(n-1)}{2^s}\right) \quad \text{for } j = 1, 2, 3, \ldots, L
\end{align*}
\]  

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

where \( n \) is the index of the sample time within the over-sampled segment of the data word. These indices, in turn, can be used to determine the chip values that apply at sample times:

\[
\begin{align*}
\bar{c}(n, i) &= \text{floor}\left( \frac{(m+1)(n-1)}{2^s}\right) \quad \text{for } j = 1, 2, 3, \ldots, L
\end{align*}
\]  

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

where \( n \) is the index of the sample time within the over-sampled segment of the data word. These indices, in turn, can be used to determine the chip values that apply at sample times:

\[
\begin{align*}
\bar{c}(n, i) &= \text{floor}\left( \frac{(m+1)(n-1)}{2^s}\right) \quad \text{for } j = 1, 2, 3, \ldots, L
\end{align*}
\]  

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

where \( n \) is the index of the sample time within the over-sampled segment of the data word. These indices, in turn, can be used to determine the chip values that apply at sample times:

\[
\begin{align*}
\bar{c}(n, i) &= \text{floor}\left( \frac{(m+1)(n-1)}{2^s}\right) \quad \text{for } j = 1, 2, 3, \ldots, L
\end{align*}
\]  

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

where \( n \) is the index of the sample time within the over-sampled segment of the data word. These indices, in turn, can be used to determine the chip values that apply at sample times:

\[
\begin{align*}
\bar{c}(n, i) &= \text{floor}\left( \frac{(m+1)(n-1)}{2^s}\right) \quad \text{for } j = 1, 2, 3, \ldots, L
\end{align*}
\]  

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

where \( n \) is the index of the sample time within the over-sampled segment of the data word. These indices, in turn, can be used to determine the chip values that apply at sample times:

\[
\begin{align*}
\bar{c}(n, i) &= \text{floor}\left( \frac{(m+1)(n-1)}{2^s}\right) \quad \text{for } j = 1, 2, 3, \ldots, L
\end{align*}
\]  

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

where \( n \) is the index of the sample time within the over-sampled segment of the data word. These indices, in turn, can be used to determine the chip values that apply at sample times:

\[
\begin{align*}
\bar{c}(n, i) &= \text{floor}\left( \frac{(m+1)(n-1)}{2^s}\right) \quad \text{for } j = 1, 2, 3, \ldots, L
\end{align*}
\]  

where \( \text{mod}(y,z) = y - z \times \text{floor}(y/z) \) is the usual remainder function.
Applied Parallel Algorithms for Efficient Software Correlation

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 \( k_p, m, n, \Delta t, \) as implied by a comparison of equations (32) and (47). This relationship can

\[
\Delta t = \frac{\Delta t}{m_f}
\]

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

\[
k_p = \text{round} \left( \frac{\Delta t \Delta t}{m_f} \right)
\]

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

Many cases prompt code chips \( 99 \) (FIG. 6). (C), (C), (C), (C), . . . 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 \( W_e \) and that \( \mu_i \) is the auxiliary index that would have been used to determine \( \Delta_i \) from the \( \Delta(x) \) table had the table existed. In order to calculate \( \Delta_{i+1} \) for data word \( W_{e+1} \), \( \mu_{i+1} \) is computed (procedure defined herein), feedback shift register calculations that generate \( C(\mu_i), C(\mu_i+1), C(\mu_i+2), \ldots, C(\mu_i+) \) are iterated, and the resulting chip values are used to perform \( (\mu_{i+1} - \mu_i) \) iterations of equations (44b) or (44c).

Determination of the correct index into the \( x_i(i), x_{eml2}(i) \), and \( x_{eml2}(i) \) tables for data word \( W_e \) can be reduced to the determination of two quantities. One is the time offset index \( k_p \) that causes \( \Delta t \) from equation (32) to match the true time offset for data word \( W_e \) as closely as possible. The other quantity is the auxiliary index \( \mu_i \). It constitutes an index for the sequence of actual code chips that are associated with data word \( W_e \). Given these two quantities, the correct index for the three \( x(i) \) tables is

\[
\mu_{i+1} = \mu_i + k_p \Delta t
\]

The auxiliary index \( \mu_i \) is determined by the position of the \( W_e \) data word relative to the PRN code chip sequence. Once that position has been ascertained, the index \( k_p \) can be calculated from the position relative to the \( W_e \) samples of the L code chips that are associated with the index \( \mu_i \).

A time integer can keep track of the number of fine-scale time units in a given interval. The fine-scale time unit is a small fraction of the sample interval 63 (FIG. 2D), \( \Delta t \):

\[
\Delta t = \frac{\Delta t}{m_f}
\]

This computation requires the undefined chip values \( C(-L+1), C(-L+2), C(-L+3), \ldots, C(-1), \) and \( C(M+2), C(M+3), C(M+4), \ldots, C(M+L) \). The value zero can be used for each of these undefined chips because they can affect the over-sampled codes only for the first \( n_o \) samples of data word \( W_e \) (FIG. 6) or for the last \( n_o \) samples of data word \( W_e \) (99). None of which are part of the accumulation. The table of equation (43) can be constructed by using the following iterative procedure:

\[
\Delta t = C(0)
\]

\[
\Delta t = \text{mod}(2M, 2^2) + C(0) \quad \text{for} \quad \mu = 2, 3, 4, \ldots, (M+2)
\]

\[
\Delta t = \text{mod}(2M, 3) + C(0) \quad \text{for} \quad \mu = (M+3), (M+4), \ldots, (M+L+1)
\]

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

In many cases prompt code chips \( 99 \) (FIG. 6). (C), (C), (C), (C), . . . 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

\[
W_{e+1} = \text{round} \left( \frac{\Delta t \Delta t}{m_f} \right)
\]

where \( \text{round}() \) function rounds up or down to the nearest integer. The time lag 67 (FIG. 2D), \( \Delta t \), is the amount by which the end time of PRN code chip \( C(p,L) \) lags the first sample time of data word \( W_e \). 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 \( k_p, m_f, \Delta t, \) as implied by a comparison of equations (32) and (47). This relationship can
be used to determine \( k_x \) from an iteratively determined \( k_f \).

Several constants are required by the iterative procedure that determines \( k_f \), \( k_x \), and \( \mu_v \). The first five constants are used to account for the difference between the nominal chip length \( \Delta c_{nom} \) used to generate the \( x(i) \) tables, and the actual chip length (FIG. 2D), \( \Delta c \) used in the accumulation:

\[
k_f = \text{round}\left[ (n_x - 1) \frac{\Delta c_x}{2} \right]
\]

where the \( \text{round}(\cdot) \) function returns the nearest integer.

The iteration begins by initializing \( k_f \) and \( \mu_v \) for the first data word. The nominal initial values are:

\[
k_f = \text{round}\left( k_f \right)
\]

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

Five additional constants can be used to define the \( k_v \) and \( \mu_v \) iterations:

\[
L_{\mu} = \text{round}\left( \frac{n \cdot \Delta c}{\Delta c_{\mu}} \right)
\]

\[
\Delta k_{\mu} = \text{round}\left( \frac{n \cdot \Delta c}{\Delta c_{\mu}} \right)
\]

\[
\Delta k_{f} = \text{round}\left( \frac{m_f \cdot \Delta c_{f}}{\Delta c_{f}} \right) - n \cdot m_f
\]
The table look-up calculations finish with the computation of $k_{v}$, $i_{v}$, and the actual table look-ups:

$$k_{v} = \text{round} \left[ \frac{mk_{v,i}(p_{v})}{m_{v}} \right] \quad \text{for } v = 1, 2, 3, \ldots, N$$

The round( ) operation in equation (55) can be implemented by adding $m_{2}$ to $mk_{v,i}(p_{v})$ before the rightward bit shift that constitutes division by $m_{v}$. The result of the division will be the correct value of $k_{v}$ for any sign of $mk_{v,i}(p_{v})$ if the computer works with 2's complement notation for signed integers and if the rightward bit shift fills in the left with the 2's complement sign bit, i.e., with the left-most bit.

Given $k_{v}$ from equation (55) and $\mu_{v}$ from equation (54b), one can use equation (45) to compute $i_{v}$. This value, in turn, can be used to index into the tables to determine the Prompt PRN code sign 29A (FIGS. 2D and 3A), $x_{v,0}$, the early-minus-late PRN code zero mask 35B (FIGS. 2D and 3A), $x_{v,0,lz}$, and the early-minus-late PRN code sign 35A (FIGS. 2D and 3A), $x_{v,eml2}$, that correspond to data word $W_{v}$:

$$x_{v,0} = x_{0,i}(i_{v}) \quad \text{for } v = 1, 2, 3, \ldots, N$$

$$x_{v,0,lg} = x_{0,lz,i}(i_{v}) \quad \text{for } v = 1, 2, 3, \ldots, N$$

$$x_{v,eml2} = x_{em,i}(i_{v}) \quad \text{for } v = 1, 2, 3, \ldots, N$$

The reduced conditions in equations (54a) and (54b) can be replaced with the following computations:

$$\eta_{v} = \begin{cases} \min(0, \text{sign}(k_{v,i} - k_{v,0}) \Delta k_{v,0} < 0) & \text{for } v = 1, 2, 3, \ldots, N \\ 0 & \Delta k_{v,0} = 0 \\ \max(0, \text{sign}(k_{v,i} - k_{v,0}) \Delta k_{v,0} > 0) & \text{for } v = 2, 3, 4, \ldots, N \end{cases}$$

The min( ) and max( ) 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 $k_{v,0}$ and $\mu_{v}$ equal to $\mu_{v,0}$. The value of $\eta_{v}$ is $-1$ if $k_{v,i} > k_{v,0}$ and $k_{v,i} < k_{v,0}$, and $+1$ if $k_{v,i} > k_{v,0}$ and $k_{v,i} > k_{v,0}$. 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 $\Delta k_{v,0}$. 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 $k_{v,0} - k_{v,i}$. Similarly, $\eta_{v}$ for the third condition is equal to the sign bit of the 2's compliment representation of $k_{v,i} - k_{v,0}$. 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 $\Delta(i_{v})$ values. The method further includes the step of computing the auxiliary constants (method step 203, FIG. 7) in equations (48a)-(48e) and (50a)-(50e). The method further includes the step of initializing $k_{v}$ and $\mu_{v}$ (method step 205, FIG. 7) by evaluating equations (51a)-(52b). The method further includes the step of iterating equations (53a), (53b), (57a)-(57c), (49), (55), and (45) (method step 207, FIG. 7) to compute, for each iteration, $k_{v,0}$, $k_{v}$, $\mu_{v}$, $\mu_{v,0}$, and $i_{v}$. The method further includes the step of iterating equations (56a)-(56c) (method step 209, FIG. 7) to compute, for each iteration, $x_{v,0}$, $x_{v,eml2}$, and $x_{v,0,lg}$.
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 GLOMANS 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 55. At time TIC 56, the TIC function latches all of the PRN code 45, 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 \( \tau_{jk} \), the estimated code chipping rate of software receiver 10 for satellite j during its kth PRN code period, can be determined either by an acquisition search procedure, or if tracking, by a delay-locked loop. Likewise, \( \dot{\omega}_{Doppler} \), 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:

\[
\tau_{jk+1} = \tau_{jk} + \frac{1023}{f_{jk}} \quad (58)
\]

\[
\dot{\phi}_{jk} = \dot{\phi}_{jk} + \dot{\omega}_{Doppler}(\tau_{jk+1} - \tau_{jk}) \quad (59)
\]

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 \( \tau_{jk} \) and \( \dot{\phi}_{jk} \) 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 \( \tau_{jk} \) and \( \dot{\omega}_{Doppler} \) 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 Dienendorf.

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

\[
\dot{\psi}_{TIC} = 1023 \left( \frac{t_{TIC} - \tau_{jk+1}}{\tau_{jk+1} - \tau_{jk}} \right) \quad (60)
\]

where \( \dot{\psi}_{TIC} \) is the PRN code phase 55 (FIG. 8) in chips of signal j at TIC time \( t_{TIC} \) 56 (FIG. 8). The epoch counters, which are simply a running total of the number of code periods 57 (FIG. 8), are incremented at each code start/stop time.

The carrier phase calculation at time \( t_{TIC} \) 56 (FIG. 8) is similar to the PRN code phase 55 (FIG. 8) calculation:

\[
\dot{\phi}_{TIC} = \dot{\phi}_{jk} + \dot{\omega}_{Doppler}(t_{TIC} - \tau_{jk+1}) \quad (61)
\]

where \( \dot{\phi}_{TIC} \) is the carrier phase at time \( t_{TIC} \), 56. The Doppler shift that gets returned at time \( t_{TIC} \), 56 (FIG. 8) is \( \dot{\omega}_{Doppler} \).

With respect to the performance of the system and method of the present invention, a sample screen-shot from the illustrative embodiment of the present invention is provided in Table 8. This table illustratively shows the tracking of nine channels. The roof-mounted L1 antenna of the illustrative embodiment can have a preamp with 26 db of gain. The software correlator 19 (FIG. 1) of the present invention can provide positional accuracy on the order of 10-15 meters when working in conjunction with application-specific software 15 (FIG. 1).

**Table 8**

<table>
<thead>
<tr>
<th>Lat</th>
<th>42.44354</th>
<th>Spd</th>
<th>0.5</th>
<th>SVs</th>
<th>8</th>
<th>Ozark</th>
<th>FLL</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lon</td>
<td>-76.48143</td>
<td>Hdg</td>
<td>327.0</td>
<td>Nav</td>
<td>3D</td>
<td>ODOP</td>
<td>1.9</td>
<td>GPS 19:58:11</td>
</tr>
<tr>
<td>Alt</td>
<td>269.6560</td>
<td>ROC</td>
<td>-0.7</td>
<td>HI</td>
<td>DO</td>
<td>-393.0</td>
<td>OscErr</td>
<td>0.25</td>
</tr>
<tr>
<td>ELEV</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Numerical GPS tracking data output by the illustrative embodiment.
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 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 a 1.73 GHz microprocessor to implement a 6-channel GPS receiver when processing data from an RF front-end with a sampling rate of 1/2 the chip rate for this example use a code chip start time resolution of 1/2 the chip rate. What is claimed is:

1. A software receiver comprising:
   a receiver capable of receiving a radio signal;
   means for digitizing the radio signal; and
   means for using bit-wise parallelism to provide a software correlator capable of mixing the digitized radio signal to form a baseband signal.

2. The software receiver of claim 1 wherein said software correlator comprises:
   means for computing accumulations from the correlations;
   means for down-converting the radio signal to an intermediate frequency; and
   means for computing accumulations from the correlations using the bit-wise parallelism.

3. The software receiver of claim 2 wherein said software correlator further comprises:
   means for computing accumulations from the correlations using the bit-wise parallelism.

4. The software receiver of claim 3 further comprising:
   means for converting the radio signal to an intermediate frequency; and
   means for down-converting the radio signal to an intermediate frequency; and

5. The software receiver of claim 1 wherein said means for digitizing comprises:
   means for computing accumulations from the correlations using the bit-wise parallelism.

6. The software receiver of claim 5 wherein said means for digitizing comprises:
   means for computing accumulations from the correlations using the bit-wise parallelism.
produces at least one bit sample. Parallelism accommodates multi-bit signals.

A method for processing at least one radio frequency (RF) signal over an accumulation period comprising the steps of:

- receiving at least one RF signal;
- mixing the at least one RF signal to form a baseband mixed signal using bit-wise parallelism;
- computing a fully mixed prompt integrand as a function of the baseband mixed signal and a pre-selected pseudo-random number (PRN) code using the bit-wise parallelism;
- computing a fully mixed early-minus-late integrand as a function of the baseband mixed signal and the pre-selected PRN code using the bit-wise parallelism;
- computing in-phase summed accumulations and quadrature summed accumulations over the accumulation period, the in-phase summed accumulations and quadrature summed accumulations being functions of the fully mixed prompt integrand and the fully mixed early-minus-late integrand; and
- rotating the in-phase summed accumulations and quadrature summed accumulations to correct for effects of frequency and phase granularity of the baseband mixed signal.

A communications network comprising at least one node for carrying out the method according to claim 9.

The method of claim 9 wherein said step of computing a fully mixed prompt integrand comprises the step of:

- generating the pre-selected pseudo-random number (PRN) code using the bit-wise parallelism, said step of generating the pre-selected PRN code comprising 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; and
  - 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 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.

The method of claim 13 further comprising the step of:

- computing the time grid index as a function of a time offset index $k_e$ and an auxiliary table index $k_\mu$. 

16. A software receiver comprising:

- a front-end device capable of receiving a radio signal, said front-end device capable of converting the radio signal into signal data;
- a data acquisition device capable of receiving the signal data, said data acquisition device capable of providing the signal data to a microprocessor;
- at least one shift register capable of packing the signal data into at least one data word;
- a baseband mixer capable of computing at least one baseband mixed signal as a function of the at least one data word by using bit-wise parallel processing;
- a correlator capable of computing correlations between the baseband mixed signal and a pseudo-random number (PRN) code replica signal; and
- an accumulator capable of computing summed accumulations by accumulating the correlations.

17. The software receiver of claim 16 wherein the signal data further comprises a signal sign.

18. The software receiver of claim 16 wherein the signal data further comprises a signal sign and at least one signal magnitude.

19. The software receiver of claim 16 wherein said correlator uses the bit-wise parallel processing for computing the correlations.

20. The software receiver of claim 16 wherein said accumulator accumulates the correlations by an electronic mechanism.

21. The software receiver of claim 16 wherein the radio signal can be received from a global positioning source.

22. The software receiver of claim 16 wherein said correlator can be adapted to perform functions selected from a group consisting of accepting the radio signal at any frequency, accepting any PRN code, and accepting the radio signal from any device that generates a radio signal.

23. The method of claim 9 wherein said step of computing in-phase and quadrature summed accumulations comprises the steps of:

- representing a carrier replica signal from at least one channel from a plurality of channels as a carrier replica sign and a carrier replica magnitude;
- representing the at least one RF signal 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 the 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 and an early-minus-late PRN code; and
- representing the prompt PRN code as a prompt PRN code sign;
- computing a fully mixed prompt integrand sign as a function of the baseband mixed sign and the prompt PRN code sign;
- representing the early-minus-late PRN code 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 early-minus-late PRN code zero mask;
computing prompt in-phase and quadrature summed accumulations for the plurality of channels for 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.

24. The method of claim 23 further comprising the step of: selecting the significant bits from a group consisting of zeros and ones.

25. The method of claim 23 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.

26. The method of claim 23 further comprising the step 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.

27. The method of claim 26 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.

* * * * *