2012 IEEE 18th International Symposium on Asynchronous Circuits and Systems

# A Low Power Asynchronous GPS Baseband Processor

Benjamin Z. Tang, Stephen Longfield, Jr., Sunil A. Bhave, Rajit Manohar
School of Electrical and Computer Engineering
Cornell University. Ithaca, NY, 14853, U.S.A.
Email: {bt48, slongfield}@csl.cornell.edu, sunil@ece.cornell.edu, rajit@csl.cornell.edu

***Abstract*—We present the design and implementation of an asynchronous Global Positioning System (GPS) baseband processor architecture designed with a combination of Quasi-Delay-Insensitive (QDI) and bundled-data techniques, with a focus on minimizing power consumption. All subsystems run at their natural frequency without clocking and all signal processing is done on-the-fly. Transistor-level simulations show that our system consumes only 1.4mW with position 3-D rms error below 4 meters, comparing favorably to other contemporary GPS baseband processors.**

***Index Terms*—GPS, QDI, Bundled-Data, Asynchronous Circuits, Low-Power Receiver**

## I. Introduction

### A. Motivation

Our lives are increasingly being affected by the use of GPS technology. We rely on GPS to navigate from one place to another, to locate a person or an object, to provide time synchronization in our telecommunication networks and power grids, and in many other every day applications.

Today, high power consumption of existing GPS receiver chips can cause overheating issues, and can limit continuous GPS operation in mobile devices. It is clear that the high power consumption in GPS receivers must be addressed to pave the way for advances in areas such as location-aware applications and micro robotics navigation.

Asynchronous techniques enable very low-power designs, especially in systems where the rate of required throughput may vary over time [1], [2], [3]. As a GPS system involves several different components, each of which compute at a different natural frequency, an asynchronous design could lead to benefits in power consumption for baseband processing.

A typical GPS receiver consists of an RF front end and a digital signal processor (DSP). The RF front end receives the GPS signal from the satellites, mixes it down to an intermediate frequency, and samples it. The DSP acquires a lock to multiple GPS satellite signals present in the front end samples and tracks variations in the signals over time. While the DSP tracks variations in the signal, it also extracts information from it that can be used to compute the current position and time—the "navigation solution."

### B. Related Work

Significant research effort has been devoted to reducing the power consumption of the RF front end, with current designs having power consumption of less than 10mW [4]. However, more work needs to be done to lower the power consumption of the GPS baseband processor.

A powerful DSP can perform all baseband processing in software [5]. This approach is highly reconfigurable and easy to develop and debug but not suitable for low power applications. An alternative is to use a hardware correlation engine to handle the fast correlation operations and a microprocessor to handle the rest of the signal processing tasks [6].

In applications where the user only requires infrequent position updates, a possible solution is proposed in [7], where the receiver is only powered intermittently. This approach does not continually track satellites, but instead focuses on rapid re-acqusition.

This work focuses on addressing the power consumption issue in the baseband processor when continuous position information is required. Instead of using a single DSP or a correlation engine with software support to perform baseband processing, we designed a hybrid architecture that decouples crucial GPS receiver operations from other post-processing that can either be managed locally by a co-processor or be separately managed by the cloud or base station.

Our system performs all baseband processing on-the-fly in hardware, leaving only initialization and navigation solution computation to software. This approach not only provides an optimized hybrid hardware and software solution but is also ideal for applications that need to deploy ultra-low-power mobile GPS receivers that transmit just enough information back to a base station to compute its position and time information. Moreover, our clockless data-flow driven baseband processor can be paired with any conventional GPS L1 RF front end. It is programmable to support different front end sampling and intermediate frequencies and mixing scheme.

## II. GPS Signal Structure

The GPS uses spread-spectrum signaling to modulate a carrier with ranging codes. This technique allows a GPS receiver to use Code Division Multiple Access (CDMA) to uniquely identify the signal from each satellite by the satellite's unique ranging code. Modern GPS satellites transmit signals at the L1, L2 and L5 carrier frequencies with civil and military codes. Since our system is designed to process the L1 civil signal, we will describe its signal structure here.

1522-8681/12 $26.00 © 2012 IEEE
DOI 10.1109/ASYNC.2012.20

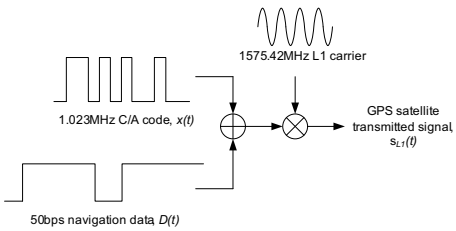


Fig. 1. GPS L1 C/A signal structure

The GPS L1 Coarse/Acquisition (C/A) signal consists of a 1575.42MHz L1 carrier signal modulated by a periodic 1023-chip C/A ranging code at 1.023Mbps, which in turn is modulated by the 50bps navigation data as shown in $Fig.$ 1. The C/A ranging code has a period of 1ms and is selected from a special class of pseudorandom noise (PRN) sequences known as Gold codes. Each satellite can be identified by its unique Gold code. The navigation data is a sequence of bits that carries satellite orbital information, satellite time and error correction parameters [8]. The L1 C/A signal, transmitted with an average power of $P_{L1}$, from a GPS satellite can be represented mathematically as

$$s_{L1}(t) = \sqrt{2P_{L1}}D(t)x(t)\cos\left(2\pi f_{L1}t + \theta_{Tx}\right) \qquad (1)$$

where $D$ is the navigation data; $x$ is the C/A code and $f_{L1}$ the L1 carrier frequency.

The signal from a particular satellite reaches the receiver's RF front end after some transmission delay. This difference in time of flight of the signals from different satellites to the receiver forms the fundamentals of radio-navigation on which the GPS system is based. Additional uncertainties in the received signal are introduced by the receiver's front end oscillator error, and the Doppler frequency shift due to the relative movement of the satellite and the receiver. The receiver must compensate for these uncertainties so that a correct navigation solution can be computed.

The GPS baseband processor's role is to first acquire a rough estimate of the transmission delay and Doppler frequency shift for each available satellite and then refine the estimates through tracking loops. A delay-locked loop (DLL) tracks deviations in the estimate of the transmission delay whereas a frequency-locked loop (FLL) or a phase-locked loop (PLL) tracks deviations in the Doppler frequency estimate.

It is important to realize that the signal in $(1)$, after conditioning by the front end, consists of several signals with different intrinsic frequencies, the fastest of which is the RF front end ADC sampling rate, followed by the chipping rate of 1.023MHz with a 1ms PRN code period in $x$, and finally the 50Hz navigation data rate in $D$. In subsequent sections, we will describe how our system exploits these properties when processing the signal to produce navigation solution, with the objective of minimizing power consumption while optimizing performance.

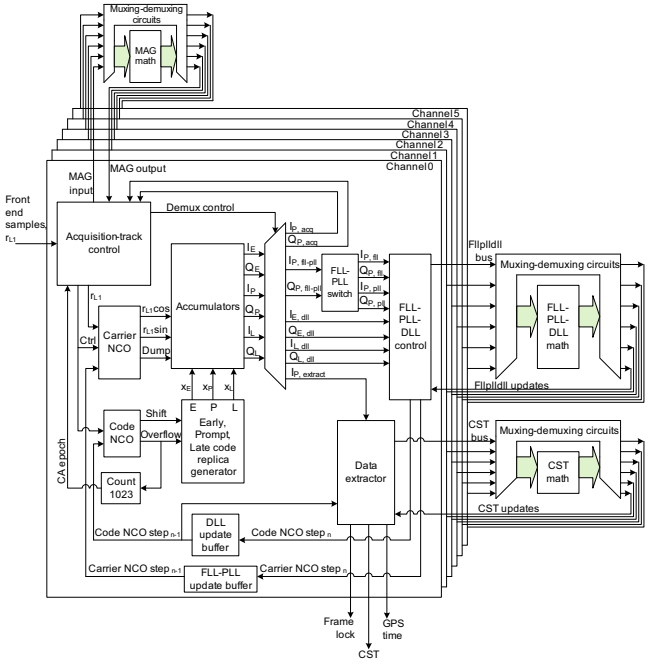


Fig. 2. System block diagram

## III. System Overview

Our dataflow-driven system expects 1-bit samples from an RF front end which it uses to acquire, track, and extract crucial data. It then sends the extracted data to a co-processor or base station for navigation solution computation. In order to solve the four unknowns corresponding to the receiver time, X, Y and Z coordinates, a GPS receiver needs to track at least four satellites. To allow for redundancy and the flexibility to compute an over-determined least-squares solution, our system is capable of tracking up to six satellites simultaneously. This is accomplished with six GPS *channels*, where each channel is responsible for processing the signal from one particular satellite. We optimize the system to have all channels share a single tracking loop.

A comprehensive description of GPS baseband processor architectures can be found in [9], [10], [11]. In what follows, we summarize the choices we have made in our implementation of a GPS baseband processor.

### A. Acquisition

Before a receiver can begin to track a satellite, it needs to know which satellite to track, and an estimate of the Doppler frequency and code offset of the signal for that particular satellite. Therefore, during signal acquisition, a typical receiver searches the expected Doppler frequency space and code offset space of candidate satellites.

First, for each candidate satellite, the input signal from the RF front end undergoes a carrier-wipeoff with a candidate Doppler frequency $f_D$ and a code-wipeoff using a locally-generated code replica, $X_P$, with a candidate code offset, $\hat{\tau}$.

This $(\hat{f}_D, X_P, \hat{\tau})$ candidate hypothesis is tested by correlating the anticipated signal with the received signal over a 1ms code period ($N$ front end samples) to yield an in-phase accumulation, $I$ and the quadrature accumulation, $Q$. Denoting the $k^{th}$ sample from the RF front end by $r_{L1}(t_k)$, and the intermediate frequency to which the satellite signal has been mixed down by the RF front end by $f_{IF}$, we can write:

$$I = \sum_{k=1}^{N} r_{L1}(t_k) X_P(t_k - \hat{\tau}) \cos\left\{2\pi\left(f_{IF} + \hat{f}_D\right) t_k\right\} \quad (2)$$

$$Q = \sum_{k=1}^{N} r_{L1}(t_k) X_P(t_k - \hat{\tau}) \sin\left\{2\pi\left(f_{IF} + \hat{f}_D\right) t_k\right\} \quad (3)$$

If the correlation power, $I^2 + Q^2$, exceeds a threshold, then we have validated our $(\hat{f}_D, X_P, \hat{\tau})$ hypothesis and completed acquisition for one satellite. Our system implements an assisted acquisition approach where the system is provided with the satellite number to acquire as well as their respective correlation power threshold and slowly-varying Doppler frequency estimate $\hat{f}_D$. We will describe this feature further in Section IV.

In each channel shown in $Fig.\ 2$, two 32-bit numerically controlled oscillators (NCO) are driven by the data-flow from the front end. The code NCO controls the chipping rate of the code replica generator. Every time the code NCO overflows, a new code replica is generated and every 1023$^{\text{rd}}$ overflow of the code NCO marks the end of a 1ms code period, forming our accumulation interval. The carrier NCO consumes data bit whenever one is available, and correlates this against replicated sinusoids to create a 3-bit output for both the in-phase and quadrature components. Together, these signals form the correlator summands from $(2)$ and $(3)$.

### B. Tracking

After acquisition, the receiver channel has enough knowledge of the code offset and Doppler frequency to roughly align its code and carrier replicas to the received signal. The receiver channel then enters tracking mode to continuously track deviations in code offset and Doppler frequency.

Besides the prompt correlators that produce the prompt in-phase accumulation in $(2)$ and quadrature accumulation in $(3)$, each channel uses early correlators with the replica PRN code advanced by a $\frac{1}{2}$ chip to produce the early in-phase and quadrature accumulations, and likewise, late correlators with the replica PRN code delayed by a $\frac{1}{2}$ chip to produce the late in-phase and quadrature accumulations. These early and late accumulations are needed to implement a DLL which is used to update the chipping rate for the channel's code NCO to correct for misalignments in phase between the replica PRN code and the received PRN code.

Other than tracking the code phase, the system uses an FLL and a PLL to track the Doppler frequency. The FLL is more robust to noise, has better dynamic performance, and has a wider pull-in range than the PLL but its measurements are noisier. Hence, the FLL is only used to obtain a more accurate estimate of the Doppler frequency immediately after acquisition before handing the task over to the PLL.

The transition from FLL tracking to PLL tracking happens after a programmable delay (400ms was used in testing). The PLL locks the phase of the in-phase portion of the carrier replica to the incoming signal. It is important to realize that all DLL, FLL and PLL tracking loops are only called into action at the end of an accumulation period which occurs once every 1ms in each channel. More details on our tracking loops can be found in Appendix A.

### C. Data Extraction

From $(1)$, the GPS satellite's signal contains a 50bps navigation data stream. This data stream is modulated with the satellite's PRN code which has a period of 1ms. As a result, 20 periods of the satellite's PRN code contain information for the same data bit. Since a PLL is used for tracking, the accumulation power is concentrated in the in-phase accumulation. Hence, at the end of a 1ms accumulation period, we can extract one raw data bit from just the sign of the prompt in-phase accumulation.

To remove the redundant data bits, we down-sample the raw data bits by 20 to 1. From the down-sampled data, the system attempts to lock onto the GPS message frame. Each 300-bit subframe begins with the preamble sequence of "10001011", so the system initially simply searches for this pattern.

However, as our PLL discriminator has a 180-degree phase ambiguity, the extracted data bits may be inverted. Thus, a search for both the normal and inverted preamble bit sequence is required. To ensure that the detected bit sequence is indeed the preamble bits instead of a string of navigation data message bits that happens to match the preamble bit sequence, the system extracts the GPS satellite time information and checks that it correctly increments from one subframe to the next before announcing frame lock.

Our system uses a code phase accumulator for each channel, hereafter called Code Start Times (CST), as time stamps in units of sample counts to mark the start of the C/A PRN code period. Every time the DLL updates (approximately 1ms), the CST is incremented by the number of samples in that approximate 1ms time lapse. At an interval of 1s, determined by counting 1000 raw data bits from the start of a subframe, each channel reports its frame lock status, GPS satellite time and its corresponding 64-bit CST time stamp.

With these three pieces of information from at least four satellites, an external co-processor or a base station can compute the navigation solution. First, relative pseudoranges are computed from the CST values of each channel using the first channel as the reference [9]. From the relative pseudoranges and corresponding GPS times, we can then compute the X, Y and Z receiver position and receiver clock error [9], [10].

## IV. Optimizations

### A. Natural Frequency Operation of All Sub-systems

The structure of the GPS signal allows for signal processing at different rates. By designing the system with asynchronous

circuit techniques, we can enable each sub-system–fast and slow–to run at its natural frequency.

Tokens are passed down through handshakes from fast sub-systems to slower sub-systems to enable on-the-fly signal processing without the use of memory. The correlators comprising accumulate and dump modules, code and carrier NCOs and their controls operate at the RF front end sampling frequency of several MHz. This is the fastest rate the system is expected to handle. The next signal processing step involving the tracking loops is three orders of magnitude slower as each channel's correlators only dump their accumulations at the end of a 1ms accumulation period.

Furthermore, only 1 in every 20 raw data bits extracted from the sign of the 1KHz stream of prompt in-phase accumulations is used for navigation message extraction. From this 50bps data stream, the system derives frame lock and satellite time information and finally outputs CST measurements at 1Hz. Other than the complicated fixed-point arithmetic circuits implemented with bundled data techniques [12], we implement all modules in the system using QDI techniques.

### *B. Adder and Increment-Decrement Accumulators*

Accumulators operate at the RF front end sampling frequency and are one of the more power-hungry components in the whole system. Moreover, there are six accumulators per channel to produce early, prompt and late pairs of in-phase and quadrature accumulations.

The 3-bit accumulator inputs which are represented by the summands in $(2)$ or $(3)$ and their early and late variants, are accumulated over one accumulation period to produce 16-bit sum outputs. It follows from $(2)$ and $(3)$ that when the system is tracking well, the iterative cross-correlation of the replica and received signals will either trend towards $+\infty$ or $-\infty$, depending on the sign of the encoded navigation data bit.

As the 3-bit summand is iteratively added to the 16-bit sum, the bits of higher significance do not switch frequently. Based on these observations, we introduce an increment-decrement-based accumulator design to reduce transistor switching activity and hence the dynamic power consumption of this critical module.

Instead of using a standard 16-bit accumulator consisting of an adder with its output fed back to one of its inputs via a register, we implement a standard 3-bit accumulator coupled with an increment-decrement-dump unit (IDD). Only a minimal number of low order bits (three because of the 3-bit input) are required to cycle in an out of the adder in the standard 3-bit accumulator. The rest of the higher order bits are handled by the IDD.

Referring to $Fig.$ 3, an encoder issues the command to increment or decrement based on the polarity of the input summand and the carry out from the adder. If the input to the accumulator is positive, the encoder issues an increment command if there is a carry out from the standard accumulator. Note that the negative of a 2's complement number $a$ ($a >$0) represented with $n$ bits is written as: $-a = 2^n - a$, where a "borrow" is obtained from the virtual $(n+1)^{th}$ bit. Therefore,

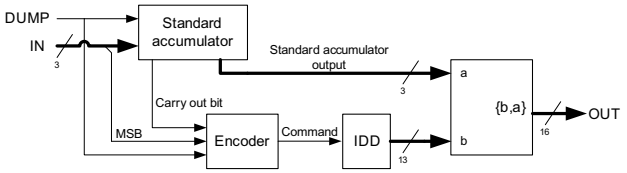


Fig. 3. Increment-decrement based accumulator

if the input to the accumulator is negative, the encoder issues a decrement command if there is no carry out from the standard accumulator as it would otherwise cancel out the "borrow" loaned from the higher order bits in the IDD. If neither of these cases hold, the encoder issues no command and triggers no transistor switching activity at all in the entire IDD.

We design the 13-bit IDD with 13 cells, each of which implements a need-based command propagation structure. If the IDD receives a command to increment, the LSB cell will increment its own value and will only propagate the increment command to the next higher-order cell if the LSB cell has a carry out; otherwise the higher-order cells experience no switching activity at all. Other than the MSB cell which does not need to propagate any commands further, all higher-order cells have the same need-based command propagation behavior as the LSB cell. Likewise, each IDD cell only propagates the decrement command if it needs to "borrow". The dump command which the accumulator receives once every 1ms is propagated throughout the IDD from the LSB to the MSB cell. The full accumulation result is just a concatenation of the result from the standard 3-bit accumulator and the result from the 13-bit IDD.

### *C. Memoryless Code Phase Acquisition*

Acquisition is typically done either using frequency domain search techniques with an FFT engine or using time domain search techniques with a bank of parallel correlators. For our assisted system that is provided with a PRN number of a satellite, a rough Doppler frequency estimate, and acquisition power threshold at start up, each channel proceeds to search the code offset space on-the-fly without the use of memory. This design exploits the fact that the PRN code sequence in the signal is periodic with a period of 1ms, and that the Doppler frequency is relatively stable over short time periods.

Each channel scans the in-coming front end samples for the correct code offset by performing 1ms-long accumulations and dropping one front end sample and incrementing an offset counter by 1 at the end of each accumulation period. When the acquisition power exceeds the threshold, that channel has found the correct code offset, $\hat{\tau}_{0,m}$, where $m$ is the channel number. The value of $\hat{\tau}_{0,m}$ is indicated by the value in the offset counter.

As different satellite signals have different code offsets, different channels will finish acquisition at different times. To handle this problem, the channels re-synchronize at strategic reference points determined by their respective code offsets, acquisition time lapses and Doppler frequency estimates.

Specifically, each channel ignores front end samples until the sample count equals the code start index $CST_{0,m}$ shown in (5). Upon doing so, the channel initializes its CST value as $CST_{0,m}$, resets its NCOs and begins tracking. The time needed for the $m^{th}$ channel to acquire, rounded to the next second, is simply:

$$t_m = \left\lceil \frac{\hat{\tau}_{0,m}}{1000} \right\rceil \quad (4)$$

$CST_{0,m}$ is calculated by adding $\hat{\tau}_{0,m}$ with the Doppler-adjusted number of samples over $t_m$ seconds as follows:

$$CST_{0,m} = N_0 \left(1 + s_m \frac{f_{D,m}}{f_{L1}}\right) t_m + \hat{\tau}_{0,m} \quad (5)$$

where $N_0$ is the nominal number of front end samples per second; $s_m$ is the front end mixing sign where $s_m = 1$ corresponds to a low-side mixing scheme and $s_m = -1$ high-side; $f_{D,m}$ is the rough Doppler frequency estimate for the $m^{th}$ channel and $f_{L1}$ is the L1 carrier frequency.

Our asymmetric acquisition design precludes the need for a bank of correlators dedicated to acquisition. The same code and carrier NCO's and accumulators are used both in acquisition mode and tracking mode. As shown in $Fig.$ 2, a demultiplexer in each channel routes the outputs of the accumulators either to the acquisition threshold detector in acquisition mode or to the tracking loops and data extractor in tracking mode.

### D. Bundled Data Math

Each channel needs to be able to carry out the math functions involved in the FLL, PLL and DLL tracking loops and in CST computation. To acquire the requisite levels of accuracy, these arithmetic circuits involved have wide datapaths. Although these arithmetic functions are highly complicated, each channel only performs such computations once every 1ms. Based on these observations, and to save design time, we implement these math functions in single-rail combinational logic blocks using fixed-point arithmetic, and do so in such a way that they are synthesizeable by commercial tools. All of the channels share these blocks using arbitrated muxing and de-muxing circuits. Data is communicated back and forth between the asynchronous QDI modules and the combinational logic blocks using a hybrid dualrail-bundled data interface.

$Fig.$ 4 illustrates how six receiver channels share a single combinational logic math block. Through arbitration, the muxing circuit sends data from the asynchronous QDI domain through its dualrail channel $M$ into the combinational logic math block. The true rails of channel $M$ drive the single-rail inputs of the combinational logic block.

The arbitration control signal derived from the validity of $M$ drives a matched delay line larger than the computation delay of the combinational logic block. Using asymmetric C-elements, the normal and inverted forms of the outputs from the combinational logic block are gated with the output of the delay line and the acknowledge of the output dualrail

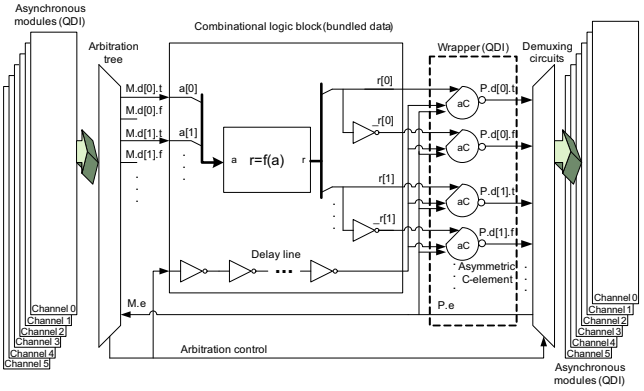


Fig. 4. Sharing of a combinational logic block and the use of a hybrid dualrail-bundled data interface

channel $P$ to generate the requisite true and false rails. Once the signal is back in the asynchronous QDI domain, the data is distributed to the corresponding receiver channel through de-muxing circuits.

### E. Optimized Tracking Loops

At the end of a 1ms accumulation period, a channel dumps its accumulations and uses the shared FLL, PLL and DLL tracking loops to update its code and carrier NCOs step sizes. Due to the complexity of the math involved, the tracking loops cannot have high levels of throughput as simple combinational blocks. The slowness of the tracking loop affects the entire asynchronous system because the NCOs will not obtain their step size updates fast enough to process the next data sample received from the RF front end.

To meet the system throughput requirements, the channel defers the application of the tracking loops outputs to the next accumulation period. At the end of the $n^{th}$ accumulation period, the NCOs update their step sizes immediately with the tracking loops outputs computed from the (n - 1)$^{th}$ accumulation period. By doing so, we not only can afford to implement slower and more power efficient tracking loops, but also can afford to allow all channels to share the same tracking loops. The tracking loops parameters are analyzed and tuned accordingly.

Some software GPS systems use floating point math to get the accuracy required for their tracking loops. However, since floating point arithmetic circuits would consume more power and increase circuit complexity, we implemented all math circuits in the bundled-data blocks with fixed point arithmetic. The bit widths of the computation sub-blocks were designed in such a way that the transformation does not degrade the final position accuracy significantly.

As shown in (7) in Appendix A, the DLL implements a discriminator with computationally expensive vector magnitude functions, involving both square and square-root operations. To reduce the complexity of the this operation, we apply a modified version of the Robertson approximation [13] where

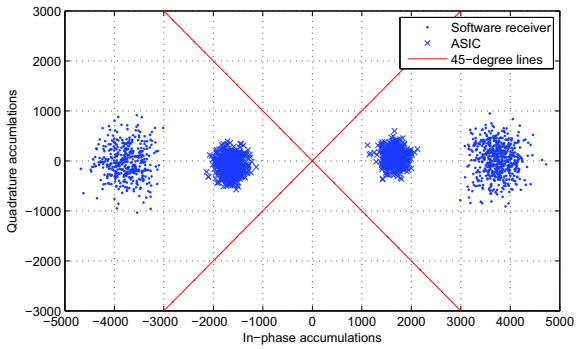


Fig. 5. In-phase and quadrature accumulations phasor compared to software receiver's

$$A = \sqrt{I^2 + Q^2} \approx \max\left(|I| + \tfrac{1}{4}|Q|, |Q| + \tfrac{1}{4}|I|\right) \quad (6)$$

Though the largest approximation error is about $10\%$ at a phasor angle of 45 degrees, the typical approximation error is less than $3\%$ because the system's PLL drives the phasor angle to zero, as shown in $Fig.\ 5$.

Additionally, the arctangent functions shown in $(8)$ and $(9)$ can be simplified with a Taylor series small angle approximation where $tan^{-1}\theta \approx \theta$. This arctangent approximation is valid for our application because the FLL and PLL discriminator values are small and the carrier phase errors alone only correspond to centimeter level errors in the navigation solution.

For the sake of modularity, the system's programmable hardware tracking loops allow them to be tailored to different tuning parameters and RF front end frequency plans. Therefore, the tracking loops parameters must be initialized from an external source on power-on.

## V. Simulation Results

### A. Receiver Performance Simulations

We first described the GPS system in the CHP language, a variant of CSP that is widely used when describing QDI asynchronous circuits. The CHP simulation was verified against a GPS software receiver written in MATLAB. Through the process of Martin synthesis [1], this high-level CHP description was translated into a gate-level description of our system. There are about 240K transistors per receiver channel and about 100K of which are in the fast-rate modules; the shared bundled-data math modules on the other hand are made up of about 580K transistors.

The correctness of the gate-level implementation is verified with a co-simulation involving Synopsys VCS and an in-house asynchronous circuit simulator. The detailed simulation results—sequences of values transmitted on all communication channels in the system—from the gate-level co-simulation were found to match that of the CHP simulation.

We simulated the system with 60 seconds of satellite signals generated from the Spirent GPS simulator without

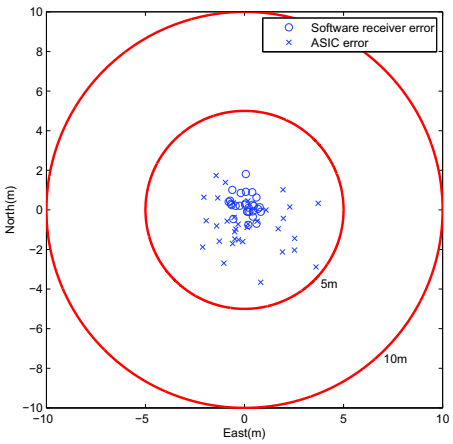


Fig. 6. Position accuracy using 6 satellites

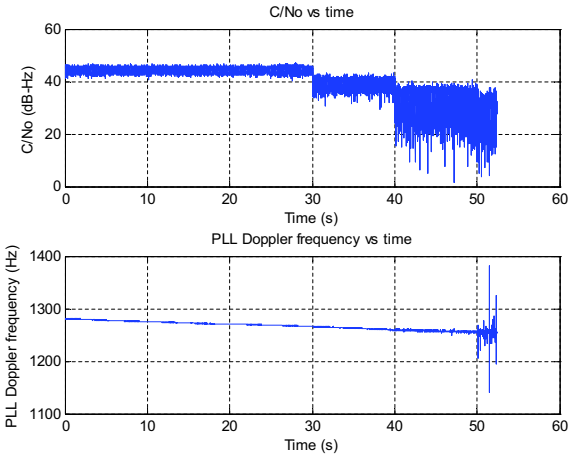


Fig. 7. Tracking sensitivity test

atmospheric, ionospheric or multipath errors. The signal from the Spirent GPS simulator is fed into the Zarlink GP2015 front end providing digital samples with a sampling period of 175ns. These datapoints are recorded in a binary file that is subsequently fed into the CHP and gate-level simulations of our system. The receiver position is computed from 6 satellites and compared to the actual simulated position.

A comparison of position error between our system and a reference Matlab software receiver developed by a leading GPS research group is shown in $Fig.\ 6$. Our system has a larger error spread because of the use of a single-bit RF front end samples, longer acquisition, fixed point arithmetic and the arithmetic approximations.

We tested tracking sensitivity with the Spirent GPS simulator using a graded reduction in carrier to noise ratio (C/No) every 10 seconds starting from the $30^{th}$ second. $Fig.\ 7$ shows that for C/No above 35dB-Hz, the PLL experiences no loss of lock. Hence, the PLL performs well in normal conditions

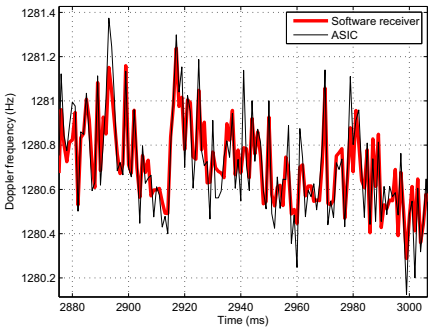


Fig. 8. Tracking of Doppler frequency by the PLL compared to software receiver's

but in weak-signal environment, longer accumulation intervals would be needed to boost tracking sensitivity. If we were to increase our accumulation interval, we would need to change the width of the accumulators, and the design of the bundled-data math. As widening the accumulators corresponds to adding additional bits to the IDD unit, this will only marginally increase switching power. Additionally, as the frequency of the bundled-data arithmetic circuits is low, we can increase the width of this with only minimal cost to power.

The front end's ADC quantization noise also plays a part in tracking sensitivity. $Fig.$ 5 shows the in-phase and quadrature accumulations distribution when tracking with the PLL where the correlation power is concentrated in the in-phase portion of the signal and the data bit flips in the signal contribute to the 2 blobs on each side of the vertical axis. The Matlab software receiver which uses 2-bit ADC samples has a higher correlation power than ours which uses only 1 bit. Despite using fixed-point arithmetic and approximations, our PLL tracks Doppler frequency relatively well, albeit noisier compared to the Matlab software receiver as shown in $Fig.$ 8.

### B. Power Simulations

We performed HSIM simulations on our system in a 90nm (Vdd=1V and T=25$^o$C) technology. To account for wire capacitance, we added wire load in the simulations to provide a more realistic account of our power numbers. The added wire capacitance is based on estimates from post-layout simulations of a portion of our system. Table I shows the power consumption of various modules in the system for 1 and all 6 receiver channels in acquisition and continuous tracking modes.

Note that since the acquisition and tracking modes in our architecture share almost all of the fast-rate modules that make up the correlators, the power consumption in both modes are close to each other. Since the implementation of a 6 receiver channel system involves duplicating the channel-dependent modules in a single channel six times, we can derive the power consumption of the full 6-channel system by adding the power consumption of the multiplexed shared math modules with 6 times the power consumption of the channel-dependent

TABLE I
POWER BREAKDOWN BY MODULE FOR A SINGLE CHANNEL AND FOR SIX CHANNELS

| | Acq ($\mu$W) (1 Chan) | Track ($\mu$W) (1 Chan) | Acq ($\mu$W) (6 Chan) | Track ($\mu$W) (6 Chan) |
|---|---|---|---|---|
| Acq control | 12.8 | 10.1 | 77.0 | 61.0 |
| Code Generator | 6.96 | 6.66 | 41.8 | 39.9 |
| Carrier NCO | 79.6 | 73.8 | 477 | 443 |
| Code NCO | 73.2 | 66.7 | 439 | 400 |
| 6 Accumulators | 61.2 | 59.9 | 367 | 360 |
| Bundled-Data | 3.95 | 4.09 | 5.90 | 6.38 |
| Other | 14.1 | 17.2 | 81.6 | 102.8 |
| Total | 252 $\mu$W | 238 $\mu$W | 1.49 mW | 1.41 mW |

TABLE II
COMPARISON WITH STATE-OF-THE-ART

| Name | This Work | [6] | [14] | [15] |
|---|---|---|---|---|
| Process (nm) | 90 | 110 | 180 | 90 |
| Voltage (V) | 1.0 | 1.2 | 1.6 | 1.4 |
| Number of Channels | 6 | 22 | 12 | - |
| System Power (mW) | 1.4 | 34 | 56 | 84 |
| RF Power (mW) | - | 19.5 | 20 | - |
| Baseband Power (mW) | 1.4 | 14.5 | 36 | - |
| Baseband Power/Channel (mW) | 0.2 | 0.7 | 3 | - |
| 3-D rms Error (m) | 3.9 | - | 3 | - |

modules in a single channel. The total power consumed by our 6-channel system is 1.5mW during acquisition and 1.4mW in continuous PLL tracking mode.

Table II provides a comparison of our system with other contemporary GPS receivers. [6], [14] and [15] are system on chip (SOC) GPS receivers with integrated RF front end and digital baseband processing. To compare the per channel baseband processing power of our system with these SOCs, we subtract the power consumed by their RF front ends from their system power, the result of which is then divided by the number of channels tracked. Note that these SOCs have the ability to compute position estimates whereas our system stops short of doing that but provides measurements that can be used to derive position estimates in a host processor or base station. Nevertheless, the power consumed in position estimate computation is negligible for typical position update rates on the order of 1Hz. Taking all these into consideration, our per channel baseband processing power is about 3 times lower than [6] and about 12 times lower than [14]. Our position estimates, computed offline, have a 3D-RMS error below 4m which is comparable to [14] and is typical for receivers used in mobile devices. The higher number of satellites tracked by [14] however does improve its position accuracy due to the effects of Dilution of Precision (DOP) [10].

## VI. CONCLUSION

The design and implementation of an asynchronous low power GPS baseband processor has been presented. The system implements an assisted acquisition scheme with sequential code phase search, a continuous tracking mode optimized for low power consumption and a data extraction scheme to provide synchronized Code Start Time measurements. The system achieves low power by allowing each subsystem to

operate at its natural frequency, by reducing switching activity in the accumulators with an increment-decrement based design and by employing shared tracking loops with deferred updates, fixed-point arithmetic and mathematical approximations. Our system consumes 1.4mW during continuous tracking mode with position 3-D rms error below 4 meters. Synchronous designers could emulate and benefit from our approach by breaking the full system down into subsystems based on their data rates and driving them with fast and slow clocks. However, synchronous designs will have more difficulty implementing completely data-driven behavior, because the system has clock ratios that are 40000000:7161000:7000:350:7.

We have begun work on laying out the system for physical manufacture and testing. Once it has been manufactured, we intend to integrate the system with a low-power RF front end and a microcontroller. Additionally, the work presented here can be extended to increase potential position accuracy and tracking sensitivity in weak-signal environments without significantly increasing the power consumption.

## APPENDIX A

Our system approximates a non-coherent normalized Early-Minus-Late (EML) DLL discriminator which gives a delay error estimate, in units of Gold code chips, at the end of the $n^{th}$ accumulation period, in the form

$$\tau_{err,n+1} = \frac{1}{2}\left(\frac{\sqrt{I_{E,n}^2 + Q_{E,n}^2} - \sqrt{I_{L,n}^2 + Q_{L,n}^2}}{\sqrt{I_{E,n}^2 + Q_{E,n}^2} + \sqrt{I_{L,n}^2 + Q_{L,n}^2}}\right) \quad (7)$$

where $I_{E,n}$ and $Q_{E,n}$ are the early in-phase and quadrature accumulations and $I_{L,n}$ and $Q_{L,n}$ are the late in-phase and quadrature accumulations. The DLL uses a carrier-aided first order filter loop [11].

Our system approximates a four-quadrant arctangent FLL discriminator, $\delta\theta$ with the objective of minimizing the Doppler frequency estimate error which is directly proportional to the rate of rotation of the prompt in-phase and quadrature accumulations' phasor. The rotation angle of the phasor that occurs in one code period at the end of the $n^{th}$ accumulation period is computed as

$$\delta\theta_{n+1} = \tan^{-1}\left(\frac{Q_{P,n}I_{P,n-1} - I_{P,n}Q_{P,n-1}}{I_{P,n}I_{P,n-1} + Q_{P,n}Q_{P,n-1}}\right) \quad (8)$$

where $I_{P,n}$ and $Q_{P,n}$ are the prompt in-phase accumulations for the $n^{th}$ code period and $I_{P,n-1}$ and $Q_{P,n-1}$ are the prompt in-phase accumulations for the $(n-1)^{th}$ code period. This discriminator is not affected by the sign flip in the prompt in-phase and quadrature accumulations caused by data bit transitions in the signal. The FLL uses a second order filter loop [11].

As for the PLL, our system approximates a two-quadrant arctangent Costas Loop discriminator, $\phi_D$.The PLL maximizes the in-phase correlation power while minimizing the quadrature correlation power, which is equivalent to minimizing the phasor angle, $\phi_{D,n+1}$ of the quadrature accumulation vector from the in-phase accumulation vector at the end of the $n^{th}$ accumulation period. The PLL uses a second order loop filter [11].

$$\phi_{D,n+1} = \tan^{-1}\left(\frac{Q_{P,n}}{I_{P,n}}\right) \quad (9)$$

## ACKNOWLEDGMENT

The authors would like to thank the late Paul Kintner for his advice on GPS systems. This material is based upon work supported in part by the National Science Foundation Graduate Research Fellowship under grant No. DGE-0707428.

## REFERENCES

[1] J. A. Brzozowski and C.-J. H. Seger, *Asynchronous circuits*, ser. Monographs in computer science. Springer, 1995.

[2] B. Sheikh and R. Manohar, "An operand-optimized asynchronous ieee 754 double-precision floating-point adder," in *Asynchronous Circuits and Systems (ASYNC), 2010 IEEE Symposium on*, may 2010, pp. 151 –162.

[3] A. Martin, "Remarks on low-power advantages of asynchronous circuits," in *Proc. European Solid-State Circuits Conference (ESSCIRC)*, 1998.

[4] K.-W. Cheng, K. Natarajan, and D. Allstot, "A 7.2mw quadrature gps receiver in 0.13um cmos," in *Solid-State Circuits Conference - Digest of Technical Papers, 2009. ISSCC 2009. IEEE International*, feb. 2009, pp. 422 –423,423a.

[5] B. O'Hanlon, T. Humphreys, M. Psiaki, and P. Kintner, Jr., "Development and field testing of a dsp-based dual-frequency software gps receiver," in *Proc. ION GNSS 2009*, 2009, pp. 317–325.

[6] J.-M. Wei, C.-N. Chen, K.-T. Chen, C.-F. Kuo, B.-H. Ong, C.-H. Lu, C.-C. Liu, H.-C. Chiou, H.-C. Yeh, J.-H. Shieh, K.-S. Huang, K.-I. Li, M.-J. Wu, M.-H. Li, S.-H. Chou, S.-L. Chew, W.-L. Lien, W.-G. Yau, W.-Z. Ge, W.-C. Lai, W.-H. Ting, Y.-J. Tsai, Y.-C. Yen, and Y.-C. Yeh, "A 110nm rfcmos gps soc with 34mw -165dbm tracking sensitivity," in *Solid-State Circuits Conference - Digest of Technical Papers, 2009. ISSCC 2009. IEEE International*, feb. 2009, pp. 254 –255,255a.

[7] W. Namgoong, S. Reader, and T. Meng, "An all-digital low-power if gps synchronizer," *Solid-State Circuits, IEEE Journal of*, vol. 35, no. 6, pp. 856 –864, jun 2000.

[8] B. Parkinson and J. Spilker, *The global positioning system: theory and applications*, ser. Progress in astronautics and aeronautics. American Institute of Aeronautics and Astronautics, 1996, no. v. 1; v. 163. [Online]. Available: http://books.google.com/books?id=lvI1a5J_4ewC

[9] K. Borre, D. Akos, N. Bertelsen, P. Rinder, and S. Jensen, *A Software-Defined GPS and Galileo Receiver: A Single-Frequency Approach*. Boston, MA: Birkhuser, 2006.

[10] P. Misra and P. Enge, *Global Positioning System: Signals, Measurements, and Performance*. Lincoln, MA: Ganga-Jamuna Press, 2006.

[11] E. Kaplan and C. Hegarty, *Understanding GPS: Principles and Applications, Second Edition*. Norwood, MA: Artech House, 2005.

[12] C. L. Seitz, "System timing," in *Introduction to VLSI Systems*, C. A. Mead and L. A. Conway, Eds. Addison Wesley, 1980, ch. 7.

[13] B. K. Levitt and G. A. Morris, "An improved digital algorithm for fast amplitude approximations of quadrature pairs," DSN Progress Report 42-40, pp. 98–101, 1977.

[14] G. Gramegna, P. Mattos, M. Losi, S. Das, M. Franciotta, N. Bellantone, M. Vaiana, V. Mandara, and M. Paparo, "A 56-mw 23-mm2 single-chip 180-nm cmos gps receiver with 27.2-mw 4.1-mm2 radio," *Solid-State Circuits, IEEE Journal of*, vol. 41, no. 3, pp. 540 – 551, march 2006.

[15] D. Sahu, A. Das, Y. Darwhekar, S. Ganesan, G. Rajendran, R. Kumar, B. Chandrashekar, A. Ghosh, A. Gaurav, T. Krishnaswamy, A. Goyal, S. Bhagavatheeswaran, K. M. Low, N. Yanduru, S. Dhamankar, and S. Venkatraman, "A 90nm cmos single-chip gps receiver with 5dbm out-of-band iip3 2.0db nf," in *Solid-State Circuits Conference, 2005. Digest of Technical Papers. ISSCC. 2005 IEEE International*, feb. 2005, pp. 308 –600 Vol. 1.