Application Report

April 2001

BACK

Global Positioning System in a Digital Signal Processor
for the TMS320 DSP platform

Andreas Hansson, Per-Ludvig Normark
Andreas Rosenlind, Christian Ståhlberg
Fredrik Svensson, Dr. Dennis Akos

Luleå University of Technology

ABSTRACT

The satellite-based Global Positioning System (GPS) provides a radio frequency (RF) signal to enable users to compute: (1) unaided position to approximately 15 meters, and/or (2) absolute time to the sub microsecond level. Such capabilities can enable a wealth of applications not previously possible and a tremendous increase in the number of users of the system is expected in the near future.

The GPS signal structure utilizes a code division multiple access (CDMA) spread spectrum ranging. As a result, the fundamental transmission has a 2 MHz null-to-null bandwidth and requires a sampling frequency of at least 4 MHz. In addition, it is necessary to process transmissions from a minimum of four satellites to determine a user's three-dimensional position. Thus the computational requirements for processing the GPS signal are extremely demanding.

To date, GPS receivers have relied on an application specific integrated circuit (ASIC) in combination with a programmable processor (micro controller or microprocessor) to operate. The ASIC would be responsible for the high bandwidth signal processing of the CDMA signal such as the digital downconversion and correlation operation, while the programmable processor would accomplish lower bandwidth tasks. The limited processing power of most programmable processors has not been capable of the load for real time CDMA signal processing.

This report documents a fully GPS software receiver (gpSrx) implementation on the Texas Instrument (TI) TMS320C6201 digital signal processor (DSP). This implementation removes the need for any dedicated signal processing hardware and demonstrates the generic CDMA processing capabilities of the TI DSP processors using GPS as an example. The TMS320C6201 used operates at a clock speed of 160 MHz, yet still provides all the required signal processing to solve for a user's position or absolute time. In addition, the report discusses the feasibility of a gpSrx implementation on other TI TMS320 DSPs.

It is important to recognize that this development allows GPS functionality to be added to devices that have been designed around a TI DSP without dedicated signal processing hardware. If the device were to operate as a true "software radio", one such mode of operation could be as a GPS receiver and provide positioning/timing capabilities. In addition, if the device for which GPS functionality is being added provides wireless capabilities, the GPS processing can be aided to further improve position accuracy and speed at which that knowledge is obtained.

This document was an entry in the TI DSP Challenge 2000, an annual contest organized by TI to encourage students from around the world to find innovative ways to use DSPs. For more information on the TI DSP Challenge 2000, see TI's World Wide Web site at www.ti.com/sc/dsp_challenge.

Contents

Figures

1 Introduction

In this section the necessary background material is presented briefly to understand the development of the Global Positioning System software receiver (gpSrx). It begins with a brief overview of the Global Positioning System, then defines what is a “software radio”, describes the algorithms necessary to process code division multiple access (CDMA) spread spectrum signals, the traditional GPS receiver design is then discussed, and finally the design goals for gpSrx are presented. The details in this section cannot be considered comprehensive, rather they merely provide a background to better understand the gpSrx project. As such, a detailed set of references is included in applicable sections if a more detailed understanding of a particular area is desired.

1.1 Global Positioning System

The Global Positioning System (GPS) is a satellite-based system to provide positioning and timing information to users worldwide. It is a passive system in that users only need to receive and process the transmission to utilize the system. No users transmission or response is required for operation. Originally developed for the United States military, the system now provides a military specific component known as GPS Precise Positioning Service (GPS-PPS) and a civilian component known as GPS Standard Positioning Service (GPS-SPS). The focus of this report is on the publically available GPS-SPS and it will be referred to as simply GPS.

GPS uses code division multiple access (CDMA) spread spectrum modulation to enable all satellites to broadcast on the same 1575.42 MHz carrier frequency. Each satellite uses a specific 1023 chip pseudo random noise (PRN) code known as the Coarse-Acquisition code to distinguish itself. The chipping rate of GPS is 1.023 Mchip/s so each period of the PRN code is exactly 1 ms. The data rate for the system is 50 bit/s. The binary PRN codes and data are modula2 combined and modulated onto the carrier using Binary Phase Shift Keying (BPSK). The resulting null-to-null bandwidth, which contains over 90% of the broadcast energy, is twice the chipping rate or approximately 2 MHz.

GPS receivers must utilize CDMA signal processing (Section 1.3) to de-spread the wideband signal and extract the data bits. These data bits will then be utilized to obtain the exact position of the broadcasting satellites and timing information. Users can compute their precise location to approximately 15 m or absolute time to the sub microsecond level without any external aiding. In order to obtain a three-dimensional position solution, it is necessary to process the transmissions of at least four different satellites in parallel.

There are a number of outstanding references on the GPS system that can be used to obtain further information on GPS and satellite navigation/positioning/timing systems [1, 2, 3].

1.2 Software Radio

The software radio concept is built upon two basic principles: (1) Move the analog-to-digital converter (ADC) as close to the antenna as possible; and (2) Process the resulting samples using a programmable processor/logic. Both principles offer numerous advantages.

First, consider moving the ADC near the antenna. Conceptually, this architecture has tremendous potential. The analog signal conditioning components (antenna, amplifiers, and filters) are all very wide band and capture an exceptional wide frequency span containing many transmissions. The particular frequency band of interest could then be digitally filtered, the resulting samples decimated to accommodate the subset of the frequency band sampled. This design would provide a front-end design capable of processing a span of radio frequency (RF) signals and provide true multi-mode operation. However, current technology limits the effectiveness of such a design. ADCs for this type of implementation, to capture a wide range of signals of varying formats and power levels, would require an extremely high dynamic range. Also the computational requirements for a digital filter/decimator for samples streaming at rates on the GHz level would be excessively high. However, the ability to capture multiple distinct frequency bands using such a design that incorporates bandpass sampling has been demonstrated successfully. Although not fully applicable for the current application, technology is improving only moving closer to such an implementation and the advantages it will bring.

Second, consider using programmable logic to process the resulting samples. This has a number of advantages. First, it would allow for multi-mode radios, a design that is becoming more and more important particularly in the wireless area where a number of different standards continue to evolve. Rather than having a dedicated application specific integrated circuit (ASIC) designed to process a specific signaling format, software could be loaded into a programmable processor designed for a specific signal structure. If a different processing architecture would be desired, it would simply be a matter of a software change rather than replacing physical components. This will also provide a much more rapid implementation of theoretical signal processing algorithms since they need only be coded in software. In addition, any simulation of algorithms can be made to more easily match the actual signal processing that will be implemented as both can be done in software. The challenge of programmable logic is in the available processing power. Typically dedicated hardware, such as ASICs, will always be faster than a programmable processor. As such, wide bandwidth systems such as GPS and other CDMA spread spectrum system will require significant computational power. However, Moore's law (the doubling of processing power every 18 months) continues to hold true and processing power is now meeting the needs of such systems.

There are a number of references that describe the software radio, its potential, and experimental results in more detail. These include [4, 5, 6, 7].

1.3 Code Division Multiple Access Signal Processing

GPS is a CDMA system and utilizes traditional CDMA signal processing algorithms. These can be separated into the following three components: (1) Acquisition; (2) PRN Code Tracking; and (3) Carrier Tracking, Data Demodulation, and Processing.

A CDMA transmission is spread across a wide frequency band. As a result, the actual received power of the transmission may actually be below the noise floor of a receiver (this is the case for GPS). As such, the first step in processing the signal is to locate it or acquire it. This is a search for three parameters. The first is the specific PRN code. In the case of GPS, each satellite is broadcasting a unique PRN code. The specific transmitters, or satellites, in view must be determined by correlating the received signal with the possible PRN codes. The second parameter is the specific code phase. Even if it is known that a particular satellite is in view, which identifies the PRN code, a locally generated PRN code must be properly aligned to within $\pm\frac{1}{2}$ chip to declare acquisition of a signal. Determining the proper alignment provides the second parameter for acquisition – the code phase. The final parameter of interest is the exact carrier frequency. Although this is typically known for a CDMA system, a search is still required as line-of-sight dynamics can introduce an unknown frequency Doppler shift. Also oscillators are never perfect and can experience drift that must be compensated. As such, the third acquisition parameter that must be determined is carrier frequency and in the case of GPS knowledge of the frequency to within $\pm$ 250 Hz is adequate to proceed to tracking.

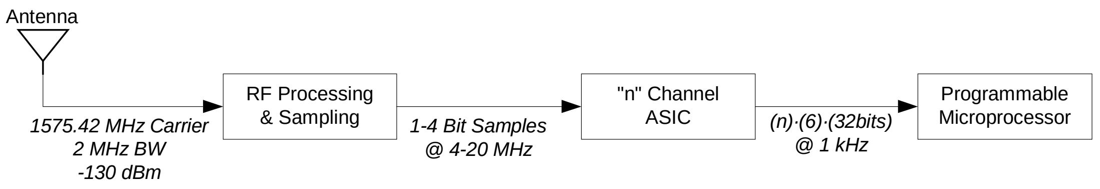
Once acquisition of a signal has been achieved, the processing can transition to a tracking mode in which two loops operate in parallel. One loop, the code tracking loop, typically uses an early-late closed loop system to track the PRN code and despread the signal. The second loop, typically a Costas loop for BPSK signals, will track the phase/frequency of the carrier, demodulate the data bits, and process the resulting data. The two loops are coupled in that the code tracking loop requires a carrier frequency/phase estimate in order to do code correlation while the carrier tracking loop can only function if the PRN code is removed from the incoming signal.

There are a number of references that described spread spectrum signal processing for CDMA systems, one particularly comprehensive reference is [8].

1.4 Traditional GPS Receiver Design

Commercially available GPS receivers are almost exclusively based on the block diagram depicted in Figure 1. An antenna is connected to an RF signal conditioning component (amplification, filtering, frequency downconversion, and sampling). The digital samples are passed to an ASIC which is responsible for high speed digital operations including: in-phase and quadrature mixing to baseband, correlation with early, late, and prompt versions of the PRN code, and accumulation of these results over a PRN code period. These accumulations are then passed, typically at a 1 kHz rate, to a programmable microprocessor that is responsible for closing the tracking loops (provide feedback to the ASIC to control carrier frequency/phase and PRN code rate) as well as decoding and processing the 50 bit/s data stream to determine position.

Figure 1. Traditional GPS Receiver Architecture



This has been a well-engineered partitioning of the required computations across fixed and programmable logic. However, it limited the flexibility of the GPS receiver architecture and further that architecture is often proprietary. This limited flexibility holds true for all aspects of the signal processing, but is particularly restrictive for signal acquisition. In such a design signal acquisition is limited to a slow serial search implementation since the microprocessor only has access to is the resulting accumulations rather than the pre-correlation data samples. Also this design does not take advantage of increasing programmable processing power that can reduce the required components for the implementation of a GPS receiver which would also imply cost and power savings.

1.5 gpSrx Design Goals

The abbreviation for receiver is RX, thus gpSrx is the designation for Global Positioning System Software Receiver, which is the focus of this report. The goal of the gpSrx implementation is to re-design the traditional GPS receiver architecture using the software radio design philosophy. With the recent advances in programmable processor power of the TI TMS320 DSP family, it is possible to remove the dedicated ASIC from the design of the GPS receiver and place all signal processing requirements within the DSP. Thus all that will be required is the analog signal conditioning and DSP components. Such an implementation will provide the means to address the shortcomings of the prevalent existing GPS receiver architecture listed in Section 1.4. Improvements gained by the gpSrx architecture will be illustrated in the prototype implementation described in Section 2.

In addition to revolutionizing the traditional GPS receiver architecture, the gpSrx application will provide a basis for the implementation of wide band CDMA signal processing in the TMS320 line of DSPs. CDMA signal processing has been adapted for the next generation wireless standard and is becoming increasing popular as a modulation format. This report will demonstrate that the extremely demanding computational requirements for GPS signal processing, a system in which multiple CDMA channels must be processed concurrently, can be met exclusively using a single TMS320 DSP. Thus a goal of the project is that gpSrx will set a foundation for the implementation of other CDMA applications within the family of TMS320 DSPs.

A GPS software receiver has been described from a post-processing perspective in [9, 10]. One of the first real time GPS software radio implementations has been achieved by the authors and is documented in [11]. This real time implementation utilized a high-end x86 compatible microprocessor for the implementation. Although fully successful, the x86 implementation utilized a high cost microprocessor with substantial power requirements to support the high clock rate. Transitioning this implementation to a lower cost/power DSP processor is a goal of this project.

2 Prototype gpSrx Implementation

In this section, the specifics of the gpSrx implementation are described along with test results. The section is subdivided into three components. The first focuses on the hardware aspect of the design, although the goal is for a software receiver, it is necessary to have some RF signal conditioning/sampling in order to get the data to process within the DSP. Also described in this hardware section is the particular DSP hardware used in the implementation of the prototype. The second section describes the software implementation of gpSrx. The software implementation is quite extensive and this section focuses on those aspects most important for implementation. Finally, performance results are presented.

2.1 Hardware

Even a software radio implementation must revolve around some core hardware components. This holds true for the gpSrx application. As such, both the front-end and the DSP requirements have been investigated to allow the development of a complete gpSrx prototype. The hardware components will be discussed in the reverse order, starting with the DSP working through the antenna.

The first aspect to be considered is the DSP requirement as this is the core of any software radio implementation. CDMA signal processing, particularly the concurrent multichannel processing required for GPS, is computationally demanding. As a result, it is necessary to select a part from the TMS320 DSP line capable of the processing requirements. There were a few possibilities, based on their characteristics described in the data sheets. Of particular interest were the latest C64x and C55x DSP families. The C55x looks to offer the required computational power yet provides low power performance. The C64x family offers tremendous computational capabilities with its VLIW architecture and specific broadband instructions while employing many of the power saving features of the C55x part. However, neither of these latest offering were available in an Evaluation Module (EVM) at the time of the development. EVM availability was an important consideration as a goal of the project was to take the design beyond the simulation results, porting the code to the target processor and address run time issues such as memory mapping. As such, the clear choice for the target DSP was the 160 MHz TMS320C6201 and corresponding EVM Bundle. Although this particular DSP utilizes a clock of less than a third of that used in the x86 GPS software radio implementation [11], prior experience with this part provided confidence that the multiple functional units from the VLIW architecture would supply the computational requirements necessary. Appendix A.1 provides a discussion as to the feasibility of the implementation of gpSrx on other members of the TMS320 family.

The next consideration was in the components for the RF front-end design, ADC, and interfacing logic. For a production level implementation, the obvious solution is to use one of the commercially available GPS RF chipsets. Such a design will be proposed in Appendix A.2 of this report. There are numerous third party vendors that supply plug-in modules for the various EVM board. The component used for this prototype was the Signalware AED-100 module [12]. The AED-100 interfaces directly to the TMS320C6201 EVM and provides dual 40 MHz TI TLC5540 8-bit ADCs, dual TI TLC7524 10 MHz 8-bit DACs, and interfacing logic between the converters and TMS320C6201.

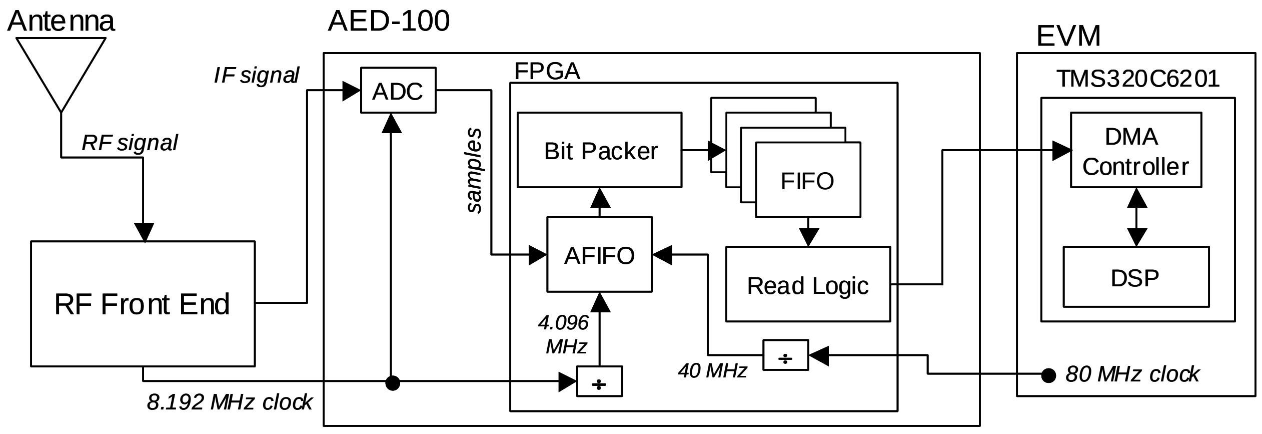
The AED-100 module was a perfect match for the gpSrx prototype and contained many more features than that required for the application. Of particular usefulness in the AED-100 module are two parts: (1) the high speed TLC5540 ADC that is used to sample the GPS intermediate carrier frequency (IF); and (2) programmable interface logic to connect the ADC to the DSP.

The modifications needed to use the AED-100 with the gpSrx application were focused on the programmable logic interface and DSP software. In keeping with the hardware trend of this section, the programmable logic interface will be discussed here while the software interfacing will be saved for Section 2.2. The programmable logic interface is based on a Xilinx XC4013XL Field Programmable Gate Array (FPGA).

The ADC clock signal is derived from the RF front-end oscillator and samples the IF signal at 8.192 MHz. This derivation of the sampling clock from the front-end is important since GPS signal processing is significantly improved when all “clocks” in the system (local oscillators and sampling clocks) are based on the same base oscillator. The FPGA skips every other sample in order to insert samples at 4.096 MHz into an asynchronous FIFO (the minimum clock rate of the TI TLC5540 ADC is 5 MHz). The asynchronous FIFO is needed since two clock domains are interfaced (the FPGA clock is derived from the EVM board and the ADC clock from the RF front-end). Two different FPGA configurations were developed, one that buffers the complete 8-bit data from the ADC, and one that only buffers the most significant bit (MSB) of the 8-bit data. This is acceptable for the gpSrx application as GPS signal processing suffers little, on the order of 3 dB, if the input samples are reduced from 8-bits to a 1-bit. Additional logic has been added to the latter configuration to pack the MSBs before buffering. This option provides an 8x improvement in the amount of data that can be transferred in the same memory block with a minimal degradation of the GPS signal processing. One drawback with this second option is that the bits must be unpacked within the DSP chip to be processed. However, this unpacking is a fairly simple operation with minimal computational overhead. As the FPGA buffer fills with the incoming samples, data is passed from the FPGA directly to the DMA controller on the DSP for efficient transfer without involving the processing unit.

Finally the overall hardware design, focusing on the FPGA configuration, for the gpSrx prototype is depicted in Figure 2.

Figure 2. Hardware Block Diagram



The final component in the hardware design is the RF front-end that provides the analog signal conditioning. In order to better understand the necessary signal conditioning, the front-end was constructed from discrete components and based on the design in [11]. It uses a single stage frequency downconversion plan with bandpass sampling to translate the 1575.42 GPS carrier frequency to an IF of 940 kHz. The final IF filter bandwidth is 2.0 MHz which captures the main lobe of the GPS signal spectrum (over 90% of the transmitted energy). This allows a sampling frequency of 4.096 MHz to be used to acceptably sample the analog signal without any distortion. It was necessary to clock the ADC at twice the desired sampling frequency, 8.192 MHz, and decimate the resulting samples within the FPGA to maintain the minimum sampling frequency requirement on the TLC5540. The resulting 4.096 MHz sampling frequency captures the required GPS information bandwidth yet is sufficient low enough to minimize computational complexity. A block diagram detailing the front-end design is shown in Figure 3.

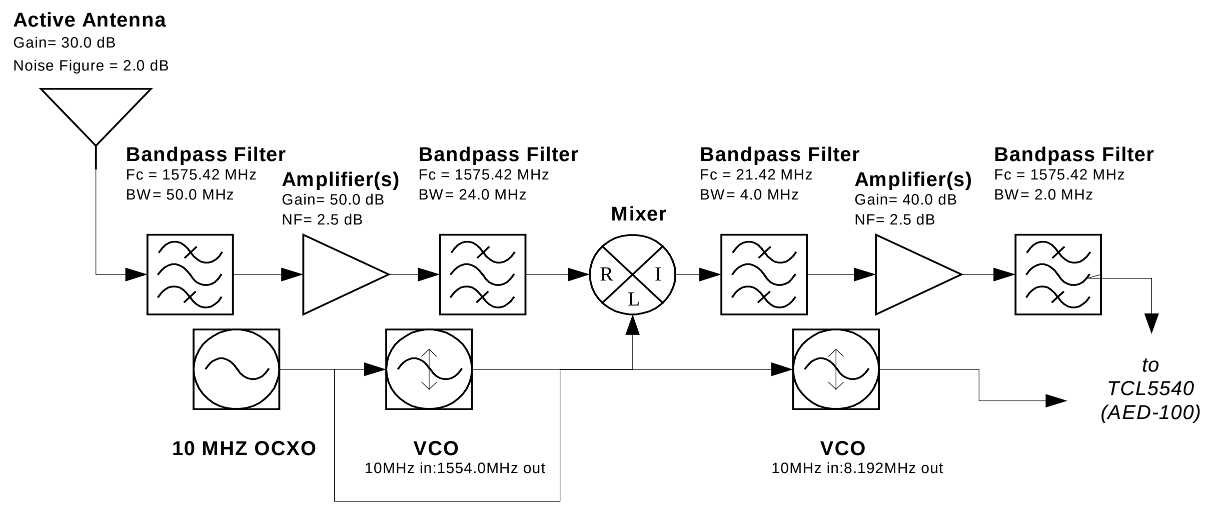
2.2 Software

The gpSrx application is been the focus of this report. It will be detailed extensively here describing its operation and mapping to the TMS320C6201. However, a secondary application has been developed to collect and store actual GPS data samples from the ADC to allow for postprocessing and testing. This application also provides the framework for an efficient means of transferring data from the ADC to the internal memory of the DSP processor. Both applications will be discussed in this section with the bulk of the material devoted to the gpSrx application.

2.2.1 rec2disk Application

The rec2disk application was developed to test the FPGA configuration and to be able to collect data for post-processing purposes. The application consists of three parts: a FPGA configuration, software running on the DSP, and software for the host PC.

Figure 3. Front-end Design



The main responsibility of the FPGA configuration is to buffer the incoming samples from the ADC before transferring them to the DSP. Although the FPGA is reconfiguration, the discussion of its functionality was presented in the hardware section of the report and the focus here is strictly on the software.

In order to transfer samples from the buffers to the DSP in an efficient manner, the FPGA configuration sends an interrupt to the DMA controller on the DSP. The interrupt is generated when the buffers are almost full. The DMA controller then moves the samples from the buffers to the internal memory of the DSP. This is done without disturbing the DSP and the rate of reads from the buffers are higher than the rate of writes to them. When the transfer has finished the DMA controller notifies the DSP that a block has been received.

The DSP software collects blocks of samples from the FPGA through the mechanism described above. When a block has been collected it sends an HPI message to the host PC. The software running on the host PC receives the message and reads the data directly from the DSP memory using evm6x_hpi_read() and saves the data to an output file. The throughput of the PCI interface with the application is about 1.3 Mb/s. At a sampling frequency of 4.096 MHz, this allows for continuously saving large amounts of packed 1 bit data to disc on the host PC (since this only requires a throughput of 512 kB/s).

2.2.2. gpSrx Application

The gpSrx DSP application is designed to work in two modes either in real-time with samples streaming from the front-end and the AED-100 board or in post-processing mode where data is provided from a recorded datafile on host. The application is divided into three components: *Acquisition*, *Tracking* and *Processing of Data*. These components have different requirements on timing, memory, there need for data. Each of these components will be described further in the following three sections with a performance overview in a fourth section.

2.2.2.1 gpSrx Acquisition

First the gpSrx application has to find, or acquire, at least four satellites. In a traditional receiver serial search is used to acquire the satellites. Serial search is utilized since only the accumulator outputs of the correlation function (which usually reside in an ASIC) can be accessed. In gpSrx there is access to the raw data samples and therefore a much faster FFT-based search can be used [9]. This search is on the order of 100x faster than serial search and provides sub-second signal acquisition. The FFT search is based on the mathematical property that convolution in the time domain is multiplication in the frequency domain. The FFT of the PRN code of the satellite is multipled with an FFT of a 1 ms block of IF data (4096 samples) from the ADC digital downconverted to baseband. The IFFT is done on the result and the absolute value provides a test of all possible code phases. If the result contains a distinct maximum, a satellite signal has been found. The index of the maximum indicates the start of the PRN code, or code phase, in the data set. A scheme using the FFT approach has been implemented and is described below.

The FFT is the most important algorithm in acquisition substage. A fixed point ANSI-C FFT has been developed, optimized to use short datatypes, and needs only 16 Kb memory to perform one 4096 point FFT (since it uses a very small sinusoid table). The internal memory can then be used efficiently to support the required two FFTs and one IFFT. Use of the TI Cx6201 dsplib version of the FFT was considered as it is faster, but the current ANSI C version has the advantage of being easy to port, it uses less memory, and is sufficiently fast for the application. The TI dsplib FFT uses a sinusoid table that is the same length as the FFT. The 4096 point FFTs/IFFTs in this application would result in the use of an extra 8 kB of data in the internal memory.

In the first stage of the gpSrx DSP acquisition, 1 ms of data is used to search for all 32 satellite PRN codes with a 6 kHz Doppler window in 500 Hz steps. The six strongest satellites together with their code phase estimate and Doppler frequency (accurate to 500 Hz) are saved for the next stage of processing.

In the second stage, a new 1 ms block of IF data is used and only the 6 satellites from stage one are tested in the search but now using only three frequency bins (frequency estimate from stage one and two additional offset by ± 250 Hz). The four strongest results from this search are saved together with their code phase and the frequency estimates. This information is then passed on to a frequency adjustment loop, which is the first part of the tracking algorithm.

It is important to note that the acquisition stage does not need continuous samples. It is sufficient to use 1 ms blocks of data for each of the two acquisition stages and simply count the number of buffers collected (the exact time that has passed) while acquisition processing occurs. This count allows a propagation ahead in time with the correct code phase for the transition to tracking. This makes it possible to divert the incoming samples beyond the first buffer to a scratch memory area in the SDRAM part of the EVM board. The FPGA design will not need extra logic to stop acquiring data and the internal data memory is not referenced by the DMA controller.

2.2.2.2 gpSrx Tracking

The tracking loop requires the code phase estimate to be accurate to approximately ± 2 samples and the frequency estimate (Doppler offset and local oscillator drift) to be accurate within ± 5 Hz. The code phase estimate is achieved from the acquisition stage, but the frequency estimate must be more precise than that provided via acquisition. The first part of the tracking algorithm uses a modified correlation subroutine to perform an implementation of a Frequency Lock Loop (FLL) which reduces the frequency uncertainty from ±250 Hz to ±5 Hz.

The tracking uses a noncoherent 2nd order delay lock loop (DLL) to track the PRN code and a phase lock loop (PLL) to track the carrier/doppler frequency and extract the data bits.

Tracking the GPS signal requires processing of continuous streaming samples. The DMA controller is used to implement a dual buffering (ping-pong) approach in the internal memory of the DSP to achieve this in real time. The buffer size used in the gpSrx DSP application is 16 Kb. The TMS320C6201 DSP chip has two blocks of 32 Kb of internal ram where a 16 Kb data buffer is allocated utilized half the available memory in each block. The variables needed for tracking (12 kB for 4 channels) utilizes a portion of the remainder of each 32k block. The DMA controller fills up one of the buffers with samples from the front-end/AED-100 board while the DSP is processing the other buffer. The DMA interrupts the DSP whenever the next buffer is ready, and they alternate buffers. This imposes hard timing requirements on the processing algorithms, if they are not finished before the next buffer is ready, it simply will not work.

The most computationally expensive task in the gpSrx application is the digital mixing of the IF signal to baseband in-phase and quadrate components combined with the correlation of an early, late, and prompt version of the PRN code. This function is efficiently implemented in fixed-point software and mapped to the Cx6201 architecture. However, in order to achieve acceptable performance two further processing optimizations have been implemented. First, the traditional early-prompt-late correlation signals has been modified to a prompt plus a combined early-and-late C/A code. This approach requires additional memory, but reduces processing requirements by approximately 25% in the most computational expensive function of the gpSrx application. A second optimization involves reducing the window of the standard correlation function from the full 1 ms of data (4096 samples) to a fraction of the full period. The integration time is proportional to the computation time and it can be reduced with a penalty in resulting Signal-to-Noise Ratio (SNR). For strong satellites, however, reducing the correlation time to 75% of the full period has a negligible impact on performance. This has been implemented and is utilized to further decrease the computational demands. In order to be able to track four satellites in parallel, it is necessary to perform the necessary mixing, correlation, and close/update the tracking loops over each 1 ms of data (4096 samples) in less then 0.25 ms in order to keep up with real time performance. This level of performance is not trivial to achieve, but it is feasible.

The prompt in-phase data is accumulated over 1 ms (or some suitable fraction) and passed into the data processing algorithm for data bit decoding. When enough data bits to solve for position are decoded, gpSrx stops tracking and solves for position, as described in the following section.

2.2.2.3 gpSrx Data Processing

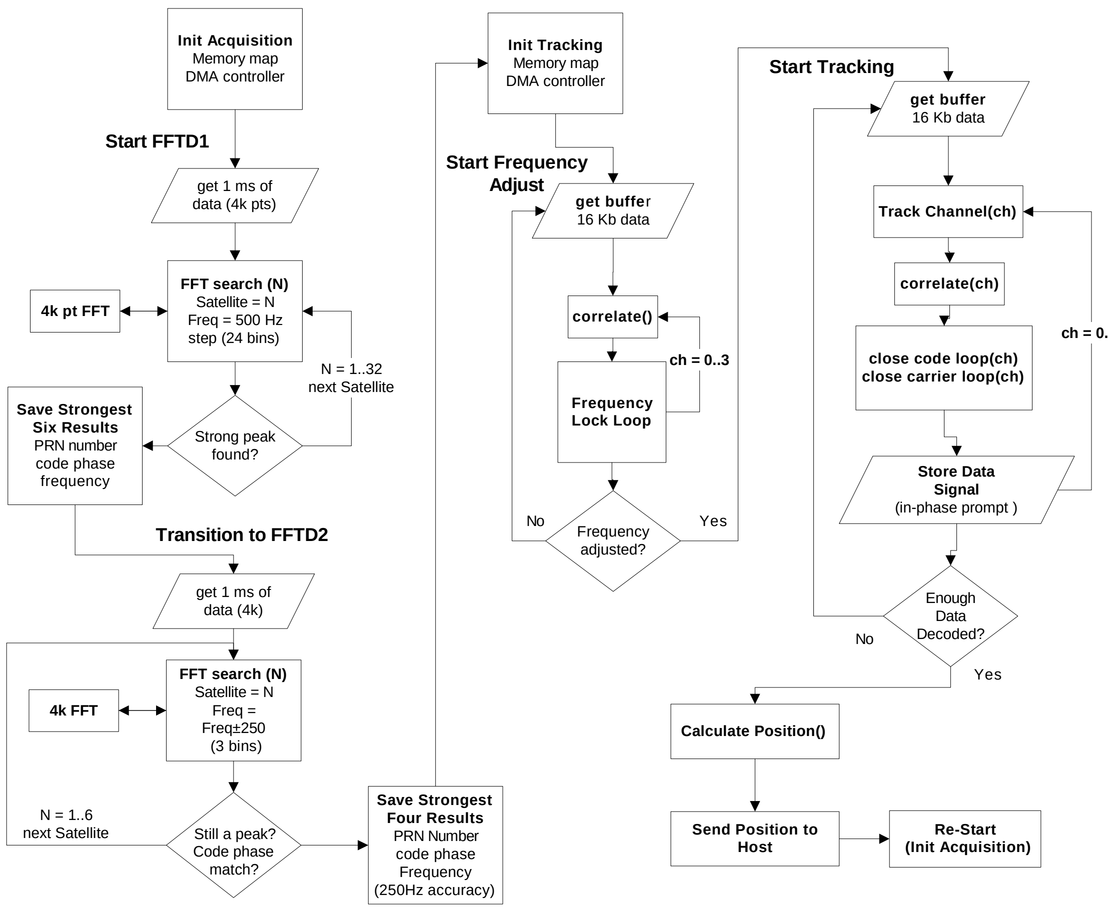
The incoming prompt in-phase accumulation from the tracking function is used to decode the navigation data bits. For GPS, the navigation data bit rate is 50 bits/s, which means that each data bit is 20 ms long, and consists of 20 prompt in-phase accumulations. First, a data bit transition must be found (bit-sync) in order to synchronize with the incoming data. When the bit sync is found, the decoding of the data bits can start. Then a word is decoded which consists of 30 data bits and parity errors are checked. Ten such words are collected into something called a navigation data subframe. GPS satellites transmit five subframes over and over again (each 300 bits, 6 seconds long) and for position solution subframe 1, 2 and 3 must be decoded. These subframes contain a time stamp (when they were generated on the satellite), ephemeris (where in space the satellite is) and clock data. With this information combined with a time stamp of when the subframe was received, it is possible to determine the distance, or range, to the satellite. Typically, all the needed data is received after tracking a satellite for between 18 to 36 seconds. In order to calculate the user position the range to at least at least four satellites must be determined (three unknowns: X, Y, Z positions and clock adjustment).

The position solution uses floating point calculations, and since the C6201 is a fixed point chip all floating point calculations are emulated in software by the compiler. This slows the calculations and the computation time requirement for tracking does not allow it to be done at every while tracking. Instead, once enough databits are decoded, the tracking operation halts and the position is calculated. A message is sent to the host PC with the calculated position and the process re-starts. Processing data collected from a surveyed antenna location using the gpSrx application results in a position solution accurate to within 35 m.

For future development, the position solution calculation has been spread out over five consecutive calls instead of one long call to make it faster to provide for continuously tracking all 4 satellites and deliver a calculated position solution at every fifth buffer.

This entire process is illustrated in the flowchart in Figure 4.

Figure 4. gpSrx Application Flow



2.2.2.4 gpSrx Performance

In the x86 implementation [11] developed previously the correlation subroutine was responsible for 98% of the computational load, thus it needs to be highly optimized if real time operation is desired. The main subpart in tracking consists of a correlation loop that has to run on each satellite being tracked. The faster this loop can be made, the more satellites that can be tracked which enable better accuracy in the position solution. Four satellites is the minimum acceptable number that can be tracked and still provide a three dimensional position solution. A high performance (600 MHz+) x86 processor can be supply the necessary performance for a four channel implementation. A DSP is often clocked at a much lower rate for power consumption issues. Instead, a DSP usually supplies specific features such as specialized instructions for this type of data processing together with a compiler that can use those special instructions.

The x86 gpSrx code (which was pure ANSI C) was brought into the Code Composer environment and the correlation timing was investigated. Since the target architecture is the C6x family it is possible to utilize the TI VelociTI compiler to optimized loops. With feedback supplied from compiler, a fully software-pipelined optimized correlation loop was developed. The effectiveness of the compiler removed the need for a linear assembler development as shown below from the compiler generated .asm file.

```
;*----------------------------------------------------------------------------*
;*   SOFTWARE PIPELINE INFORMATION
;*
;*      Known Minimum Trip Count         : 1
;*      Known Maximum Trip Count         : 1
;*      Known Max Trip Count Factor      : 1
;*      Loop Carried Dependency Bound(^) : 2
;*      Unpartitioned Resource Bound     : 3
;*      Partitioned Resource Bound(*)    : 3
;*      Resource Partition:
;*                                A-side   B-side
;*      .L units                     0        0
;*      .S units                     2        2
;*      .D units                     3*       2
;*      .M units                     3*       3*
;*      .X cross paths               1        3*
;*      .T address paths             3*       2
;*      Long read paths              0        0
;*      Long write paths             0        0
;*      Logical  ops (.LS)           0        0     (.L or .S unit)
;*      Addition ops (.LSD)          4        5     (.L or .S or .D unit)
;*      Bound(.L .S .LS)             1        1
;*      Bound(.L .S .D .LS .LSD)     3*       3*
;*
;*      Searching for software pipeline schedule at ...
;*         ii = 3  Schedule found with 6 iterations in parallel
;*      done
```

A cycle comparison between the optimized generic C version and the version used in the TMS320C6201 port shows that the DSP chip uses about 1/3 of the clock cycles required for the x86 version. One call to the correlation loop (full 1 ms of data) for one satellite requires about 46000 cycles on the TMS320C6201 DSP compared to 136000 cycles for the generic version on a x86 processor (AMD Athlon processor).

Since the implementation in this project was aimed towards a fixed-point processor, the correlation time versus total tracking time factor has decreased significantly compared to the x86 version. The correlation loop utilizes fixed point variables exclusively but floating point operations are used in updating the tracking loops. To date, a number of floating point calculations remain and the software emulated floating point calculation in the code produces a 25% overhead. On the x86 version 98% is spent on doing the correlation, on the DSP C6201 version 75% is spent on the correlation, the rest is spent on floating point emulation to close the tracking loops and such. Thus it should be possible to improve performance of the gpSrx application to allow a continuously updating position to be incorporated.

3 Conclusions

This report documents a software GPS receiver, referred to as gpSrx, on the TMS320C6201 DSP. Background has been provided on the necessary elements to understand the overall design with a complete reference list where addition details can be found if desired.

The gpSrx implementation represents the next generation in GPS receiver architectures in that all the signal processing is accomplished in software rather than the established design that uses a combination of an ASIC and microprocessor. The CDMA signal structure associated with GPS combined with the need to track multiple signals simultaneously provide demanding computational requirements. The current generation of TMS320 DSP processors provide the computational power to transition to a fully software GPS receiver architecture and gain the advantages of such a design.

Software and hardware have been developed to enable the collection of actual GPS signals via the TMS320C6201 EVM when combined with the AED-100 module from Signalware. This has allowed testing of the developed algorithms with actual GPS data to verify their performance.

The implementation has been developed and tested in both the software simulator and has been benchmarked on the physical TMS320C6201 using the EVM development tool. Results show the feasibility of such the design on this particular DSP clocked at 160 MHz (one of the lesser capable in the C6x family). As such, the design should be portable to other TMS320 DSPs chips, in particular the C64x and possibility the C55x. The full design has been tested with actual GPS samples from the initial acquisition phase, through signal tracking, and finally to a correct position solution.

Future gpSrx refinements have been suggested that will provide the basis for a transition from a prototype to more refined production level design. The application has been design to take advantage of a true software radio implementation where the DSP may be used concurrently for multiple signal processing operations. If one of these operations is wireless communications, techniques for improving the performance of the gpSrx application have been suggested.

The software radio approach enable many benefits associated with a flexible programmable architecture. The gpSrx application is likely to be the first of many such designs as available DSP computational power matches the processing requirements for GPS and similar broadband signals.

Appendix A Potential gpSrx Developments

This reports thus far provides the results of a prototyping environment designed to show the feasibility of implementing a GPS software receiver in the TMS320C6201 DSP. It is also important to consider the issues associated with transitioning this prototype design into one suited for manufacture and other considerations for later revisions.

In moving from a prototype implementation to a final design, the software providing the gpSrx implementation can be adapted to the target processor and take into account such issues as the best partitioning of the algorithms and the number of channels that can be operated in parallel. Such design issues need to first identify a target processor to be specific in the modifications. It is possible to draw some general conclusions about porting the gpSrx application to other members of the TMS320 family and this is documented in this section. Also included in this section are RF front-end hardware considerations. Any software radio will still need some analog signal conditioning and sampling before the data can enter the DSP processor. The discrete component-based front-end used for the prototype would not be applicable for a final design as a result of the cost, size, and power requirements. However details are provided on a more practical implementation using TI's TRF5001GPS front-end part. Finally, some insight is provided on how the gpSrx application could be integrated within a wireless environment and the potential such an implementation would offer.

A.1 Implementation in Other TMS320 DSPs

The TMS320C6201 DSP was used for the gpSrx implementation as a result of the available EVM hardware. However, the implementation should not be limited to that DSP. Characteristics of other members of the C6x family and also the power efficient C55x part have been investigated as to their potential to host the gpSrx application [13,14].

The multiple functional units of the VLIW architecture, low cycle count instructions, and an excellent software pipelining compiler provided for the gpSrx implementation in a C6201. However, there is still unutilized optimization potential within the code due to the signed char (or smaller) wordsizes of the GPS data samples.

Many processors provide single-instruction multiple-data (SIMD) instruction (under various acronyms such as MMX, SSE, 3DNow) to perform multiple operations in parallel. These allow packed data operations, even saturation arithmetic, with an acceptable precision level. The computationally intensive correlation loop of the gpSrx application has a majority of the computations done using 8-bit datatypes that are read linear from memory and similar operations are performed on these sequentially. Such characteristics make it indeed very suitable for packed data operations. The accumulators in the loop would be sufficient with 16-bit fields if saturation arithmetic is used. The C64x DSP family has significantly more support for such instructions than is provided in the C62x. Since the gpSrx application is written C and proven functional on one of the least capable DSPs in the C6x family, it should be even easier to port it to a C6x compatible part such as the C64x and obtain additional performance.

The only functional requirement for the gpSrx application is that the target DSP have fast (i.e., hardware) support for 32 bit data types. Since the C55x meets this requirement, it makes a port possible and the low power operation and 200 MHz clock rate make this a most interesting option. Speculating on real-time performance would require a much more detailed investigation of the optimal translation of the existing code components into target specific assembler to take advantage of the particular DSP features. These features include the numerous instructions suited for signal processing combined with the numerous data and address fetches on the different buses which are all possible in parallel. When these attributes are fully explored, the required performance could well be achieved.

A.2 Front-End/Hardware Interfacing

The RF front-end implemented for the gpSrx prototype is fully functional and provides the means to translate and amplify the GPS signal for sampling. However, the cost, size, and power requirements for such a design are not consistent with those desired in a final GPS software radio design.

There are, however, a number of integrated circuits (ICs) solutions that provide the functionality all the components and more of the design in Figure 2 (excluding the antenna). One such upcoming IC from TI is the TRF5001, which is a full GPS front-end, including 4-bit ADC and automatic gain control all in a footprint of 5mm on each side (with a depth of 1mm) and requires 47 mA at 2.7v [15]. Such a part, integrated with a TMS320 DSP running the gpSrx application, provides the means to a final production-level software GPS receiver suitable for a number of applications.

The integration of the ADC and DSP would typically require some amount of “glue logic” which is served by the FPGA in the prototype design. This element is critical to not to involve the processing unit of the DSP but rather directly interface with the direct memory access (DMA) controller for efficient transfer of the samples into the DSP memory. The TI DSPs offer direct serial access for external data to the DMA controller via the Multichannel Buffered Serial Port (McBSP). Further investigation is required to determine the feasibility of this interface that has the potential to completely eliminate the need for any glue logic between the DSP and the TRF5001.

There are still issues that must be addressed in regard to interfacing a GPS RF front-end IC with a TMS320 DSP. These include: the sampling frequency, GPS IF and bandwidth, and the number of bits desired. Such are engineering trade-offs that can be balanced to determine the most effective final implementation.

A.3 gpSrx with External Aiding

There remains significant potential for the gpSrx application that has not yet been discuss in this report. It has been shown that the gpSrx application can provide the functionality of a complete GPS receiver using only a TMS320 DSP and GPS front-end chip. However, the benefit of the gpSrx application will can be enhanced when it is used to add GPS functionality to an existing application using a TMS320 DSP. This will allow true software radio operation on the DSP. Depending on the processing power of the DSP, the various applications could either be called on demand as their processing is required or, given enough processing power, the DSP could do multitasking between operations, enabling all to operate, seemingly, concurrently. Many such applications will revolve around wireless application – perhaps mobile base stations that need an absolute time reference that can be provided by GPS or, more interestingly, mobile users that would like to take advantage of GPS positioning capabilities.

If a mobile user wanted to use gpSrx to determine their GPS position and that user was participating in a wireless network, it is possible to aid the gpSrx application with information from the wireless network to augment the GPS solution. This aiding can come in two formats which could also be combined: (1) in terms of reduced processing time where less time would be required to solve for the position; and/or (2) in terms of accuracy where the mobile users GPS position solutions could be improved.

In any wireless application, it is likely the user will have some coarse estimate of position based on the base station they will be communicating with. Over relatively small geographical areas, both the mobile and base station will see the same GPS satellite constellation. The permanent base station can provide the mobile user GPS satellites parameters, such as the ephemeris data, that would normally need to be decoded from the GPS transmission and take up to 30 seconds, or longer if the GPS receiver is using the traditional serial search acquisition. In providing this information, it will enable a mobile user to obtain a position fix in less than a second (multiple seconds could provide additional averaging and accuracy).

Another option is for the mobile user to compute only their GPS observable used in computing position (the ranges from the user to each of the satellites being tracked) using the gpSrx application. These observables could then be sent, via the wireless link, back to the base station where they could be adjusted for common mode errors (such as ionospheric delay) computed by the base station observing the same satellites at a fixed known location. The process could also work in the reverse with the base station using the wireless link to send the GPS observables correction factors to the mobile user so that these could be incorporated in gpSrx. Either approach would greatly improve the resulting GPS position solution.

References

[1] Kaplan, E. (ed.), *Understanding GPS: Principles and Applications*, ISBN: 0890067937, Artech House, MA., 1996.

[2] Parkinson, B., and Spilker, J.J. (eds.), *GPS: Theory and Applications*, Vol I & II, ISBN: 156347249X, American Institute Of Aeronautics and Astronautics, Inc., Washington DC, 1996.

[3] Enge, P., and Misra, P., *Global Positioning System: Signals, Measurements, and Performance*, ISBN: to be assigned, Ganga-Jamuna Press, 2001.

[4] *IEEE Communications Magazine*, Software Radio Issue, Volume: 33 Issue: 5 , May 1995.

[5] Buracchini, E., “The Software Radio Concept”, *IEEE Communications Magazine*, Volume: 38, Issue: 9, Sept. 2000.

[6] Mitola, J., *Software Radio Architecture: Object-Oriented Approaches to Wireless Systems Engineering*, ISBN: 0471384925, John Wiley & Sons, 2000.

[7] Akos, D.M.; Stockmaster, M.; Tsui, J.B.Y.; Caschera, J., “Direct Bandpass Sampling of Multiple Distinct RF Signals”, *IEEE Transactions on Communications*, Volume: 47 Issue: 7 , July 1999.

[8] Peterson, R.L., Ziemer, R.E., Borth, D.E*., Introduction to Spread Spectrum Communications*, ISBN: 0024316237, Prentice Hall, 1995.

[9] Tsui, J.B.Y., *Fundamentals of Global Positioning System Receivers: A Software Approach*, ISBN: 0471381543, John Wiley & Sons, 2000.

[10] Akos, D.M., *A Software Radio Approach to Global Navigation Satellite System Receiver Design*, Ph.D. Thesis, Ohio University, 1997.

[11] Akos, D.M., Normark, P.L., Hansson, A., Rosenlind, A., Enge, P., “Real-Time GPS Software Radio Receiver”, Institute of Navigation 2001 National Technical Meeting, Long Beach, CA, January 22-24, 2000.

[12] Signalware Corporation, *AED-100 High Speed Analog Expansion Daughterboard Documentation Package*, Version 1.0, Signalware Corporation, Colorado Springs, CO, April 2000.

[13] Texas Instruments, *TMS320C64x Technical Overview*, Texas Instruments, Dallas, TX, September 2000.

[14] Texas Instruments, *TMS320VC5510 Fixed-Point Digital Signal Processor Data Manual*, Texas Instruments, Dallas, TX, April 2001.

[15] Texas Instruments, *TRF5001 GPS RF Receiver – Product Preview Datasheet*, Texas Instruments, Dallas, TX, December 2000.

BACK