SOFTWARE GPS IN SB3500 PROCESSOR

Daniel Iancu (Sandbridge Technologies Inc., Tarrytown, NY 10591 USA, Tampere University of Technology, Tampere, Finland, diancu@sandbridgetech.com) Mayan Moudgill (Sandbridge Technologies Inc., Tarrytown, NY 10591 USA, mmoudgill@sandbridgetech.com), Yuri Pogudin (Sandbridge Technologies Inc., Tarrytown, NY 10591 USA, ypogudin@sandbridgetech.com), Hua Ye (Sandbridge Technologies Inc., Tarrytown, NY 10591 USA, huaye@sandbridgetech.com), Andrei Iancu (Sandbridge Technologies Inc., Tarrytown, NY 10591 USA, aiancu@sandbridgetech.com), Helena Leppäkoski (Tampere University of Technology, Tampere, Finland, helena.leppakoski@tut.fi) , JarmoTakala (Tampere University of Technology, Tampere, Finland, takala@cs.tut.fi) , Emanoil Surducan (National Institute of Research and Development for Isotopic and Molecular Technologies, Cluj-Napoca, Romania, esurducan@gmail.com), Vasile Surducan (National Institute of Research and Development for Isotopic and Molecular Technologies, Cluj-Napoca, Romania, vsurducan@gmail.com) and John Glossner (Sandbridge Technologies Inc., Tarrytown, NY 10591 USA, glossner@sandbridgetech.com)

ABSTRACT

Global Positioning Satellite (GPS) receivers have entirely changed the way were used navigating. They are part of our day to day life and we find them in the most common as well as the most unexpected areas of applications ranging from less accurate E911 to the most accurate receivers employed in monitoring the tectonic plates movements. Due to the computational complexity, implementing the GPS receivers in low cost low power digital signal processors targeting smart phones and PDAs is still prohibitive. GPS receivers are implemented in HW, employing multiple parallel processing channels. Each channel is responsible for tracking and demodulating one satellite. For economic reasons, in some applications, one channel can be time shared for more than one satellite or, in most expensive receivers thousands of parallel processing channels are employed to improve the time to first fix. Also, in multi protocol communication systems the HW implementation becomes less attractive due to extra chip cost and PC board area consumed. In this paper we present a pure SW implementation of the GPS receiver, implemented in SB3500 DSP, with time to first fix comparable if not better then the existing, technologically most advanced GPS receivers with the same positioning accuracy.

1. INTRODUCTION

The received GPS signal can be viewed as a superposition of N_s DS-CDMA signals coming from N_s visible satellites. Each satellite has its unique signature. The receiver sees different carrier frequency, for each satellite, due to the Doppler effect. The GPS composite signal, for the CA cod [1], can be represented:

$$s(t)=\sum_{i=0}^{N_s-1}\sum_{n=-\infty}^{+\infty}\sum_{k=-\infty}^{+\infty}A_i d_i[k]g\left(t-k\frac{N_p}{f_i}\right)g\left(t-n\frac{1}{f_i}\right)\times \\ \times P_i[(n+n_i^{'})\%N_p]\cdot\cos(2\pi f_i t+\varphi_i) \quad (1)$$

The derivation of (1) is illustrated in the APPENDIX.
Without losing generality, in (1), the noise term has been deliberately ignored for commodity. In order to extract the payload information embedded in the composite signal, for one satellite, we perform the following operations:
Multiply equation (1) by

$$g\left(t-k^{'}\frac{N_p}{f}\right)\cdot g\left(t-m\frac{1}{f}\right)\cdot\cos(2\pi f t)$$

Multiply equation (1) by

$$-j\cdot g\left(t-k^{'}\frac{N_p}{f}\right)\cdot g\left(t-m\frac{1}{f}\right)\cdot\sin(2\pi f t),$$

Add the two expressions together

$$\chi(t)=s(t)\cdot g\left(t-k^{'}\frac{N_p}{f}\right)\cdot g\left(t-m\frac{1}{f}\right)\times \\ \times\cos(2\pi f t)-s(t)\cdot j\cdot g\left(t-k^{'}\frac{N_p}{f}\right)\times \\ \times g\left(t-m\frac{1}{f}\right)\cdot\sin(2\pi f t)=$$

$$= \sum_{i=0}^{N_s-1} \sum_{n=-\infty}^{+\infty} \sum_{k=-\infty}^{+\infty} \xi_{i,k,n} \cdot g\left(t - k' \frac{N_p}{f}\right) g\left(t - k \frac{N_p}{f_i}\right) \times$$
$$\times g\left(t - m \frac{1}{f}\right) g\left(t - n \frac{1}{f_i}\right) \cdot \cos(2\pi f_i t + \varphi_i) e^{-j2\pi f t} \quad (2)$$

In (2), $j = \sqrt{-1}$, f is the carrier frequency at the transmitter, the same for all satellites, and

$$\xi_{i,k,n} = A_i d_i[k] P_i[(n + n_i')\%N_p]$$

Without losing generality, f can be viewed as the Local Oscillator (LO) frequency, equal to the IF frequency in the receiver. In equation (2) the product of pulse functions

$$g\left(t - k' \frac{N_p}{f}\right) \cdot g\left(t - m \frac{1}{f}\right) \cdot g\left(t - k \frac{N_p}{f_i}\right) \cdot g\left(t - n \frac{1}{f_i}\right)$$

will be nonzero only if $k=k'$ and $m=n$. For specified $k=k'$ and $m=n$ there will be one single chip selected on the time scale, as illustrated in Figure 1.
Equation (2), after summation, becomes

$$\chi(t) = \sum_{i=0}^{N_s-1} A_i d_i[k] P_i[(n + n_i')\%N_p] \cdot g\left(t - n \frac{1}{\max(f, f_i)}\right) \times$$
$$\times \cos(2\pi f_i t + \varphi_i) \cdot e^{-j2\pi f t} \quad (3)$$

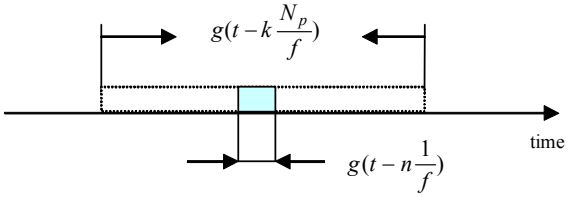

Figure 1 The highlighted part is the only non zero part in equation (2).

The integral of $\chi(t)$ over the entire time axis represents the Fourier transform of $\cos(2\pi f_i t + \varphi_i)$ scaled by a constant:

$$\int_{-\infty}^{+\infty} \chi(t)dt =$$
$$= \int_{n \cdot T}^{(n+1)\max(T,T_i)} \chi(t)dt =$$
$$= \int_{n \cdot T}^{(n+1)\cdot\max(T,T_i)} dt \sum_{i=0}^{N_s-1} A_i d_i[k] P_i[(n + n_i')\%N_p] \cdot \cos(2\pi f_i t + \varphi_i) \cdot e^{-j2\pi f t} =$$
$$= \int_{n \cdot T}^{(n+1)\cdot\max(T,T_0)} dt \quad A_o d_0[k] P_0[(n + n_0')\%N_p] \cdot \cos(2\pi f_0 t + \varphi_0) \cdot e^{-j2\pi f t} +$$
$$+ \int_{n \cdot T}^{(n+1)\cdot\max(T,T_1)} dt \quad A_1 d_1[k] P_1[(n + n_1')\%N_p] \cdot \cos(2\pi f_1 t + \varphi_1) \cdot e^{-j2\pi f t} + \ldots$$
$$+ \int_{n \cdot T}^{(n+1)\cdot\max(T,T_{N_s-1})} dt \quad A_{N_s-1} d_{N_s-1}[k] P_{N_{s-1}}[(n + n'_{N_s-1})\%N_p] \times$$
$$\times \cos(2\pi f_{N_s-1} t + \varphi_{N_s-1}) \cdot e^{-j2\pi f t} \quad (4)$$

After integration, keeping only the positive frequencies, for a single satellite Equation (4) becomes:

$$D_i(k, n + n_i') \int_{nT}^{(n+1)\cdot\max(T,T_i)} dt \cdot \cos(2\pi f_i t + \varphi_i) \cdot e^{-j2\pi f t} =$$
$$D_i(k, n + n_i') \cos\varphi_i \int_{n \cdot T}^{(n+1)\cdot\max(T,T_i)} dt \cdot \cos(2\pi f_i t) \cdot e^{-j2\pi f t} -$$
$$- D_i(k, n + n_i') \sin\varphi_i \times$$
$$\times \int_{n \cdot T}^{(n+1)\cdot\max(T,T_i)} dt \cdot \sin(2\pi f_i t) \cdot e^{-j2\pi f t} =$$

$$= D_i(k, n + n_i') \cos\varphi_i \int_0^T dt \cdot \cos(2\pi f_i t) \cdot e^{-j2\pi f t} -$$
$$- D_i(k, n + n_i') \sin\varphi_i \int_0^T dt \cdot \sin(2\pi f_i t) \cdot e^{-j2\pi f t} +$$
$$+ D_i(k, n + n_i') \cos\varphi_i \int_T^{T_i} dt \cdot \cos(2\pi f_i t) \cdot e^{-j2\pi f t} -$$
$$- D_i(k, n + n_i') \sin\varphi_i \int_T^{T_i} dt \cdot \sin(2\pi f_i t) \cdot e^{-j2\pi f t} =$$

$$= \frac{D_i(k,n+n_i')\cos\varphi_i}{2}\delta(f-f_i)+$$
$$j\frac{D_i(k,n+n_i')\sin\varphi_i}{2}\delta(f-f_i)+E_i \qquad (5)$$

Where:

$$E_i = D_i(k,n+n_i')\cos\varphi_i \int_T^{T_i} dt\cdot\cos(2\pi f_i t)\cdot e^{-j2\pi ft} -$$
$$- D_i(k,n+n_i')\sin\varphi_i \int_T^{T_i} dt\cdot\sin(2\pi f_i t)\cdot e^{-j2\pi ft}$$

is the total error due to windowing.
For all visible satellites:

$$\sum_{i=0}^{N_s-1}\int_{-\infty}^{+\infty}\chi(t)dt = \sum_{i=0}^{N_s-1}\frac{D_i(k,n+n_i')\cos\varphi_i}{2}\delta(f-f_i)+$$
$$+ j\frac{D_i(k,n+n_i')\sin\varphi_i}{2}\delta(f-f_i)\sum_{i=0}^{N_s-1}E_i \qquad (6)$$

From the above expression follows that to minimize the detection error for a particular satellite *i*, both conditions, $f-f_i=0$ and $\varphi_i=0$ need to be enforced. In other words, one has to correct for the Doppler shift as well as for the phase shift for each satellite. Coarse Doppler shift correction value is calculated through Fourier transform while the zero phase is enforced through a proportional integral differential (PID) controller for example.
In HW implementation the carrier is tracked by advancing or retarding the LO frequency and conforming to the output of a Phase Locked Loop (PLL) circuit. The integrals in (5) are performed in parallel, each by a separate channel. Each channel must have its own LO, PLL and Pseudo Number (PN) generator.
In SW, the LO is a sin-cosine real time function generator with the frequency established, for each satellite, by the Doppler search engine and the phase information is provided by the proportional integral differential (PID) type PLL as shown in Figure 2. The PN sequences are stored in memory.

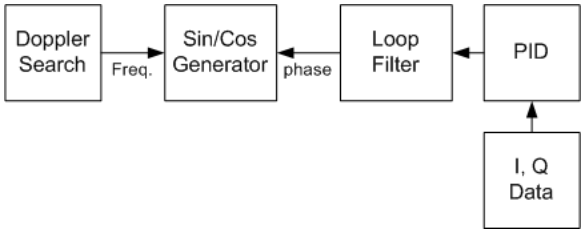

Figure 2 Sin – Cosine generation

In a multithreaded environment, all integrals in (5) are executed in parallel. The number of virtual channels is dynamically allocated depending on different technical or economical criteria.

2. IMPLEMENTATION

The 1575.42 MHz L1 carrier is down converted to 4.092MHz intermediate frequency and digitized, two bits per sample, at 16.368MHz sampling rate yielding a four times oversampled digital data. The digital samples are transferred to the digital signal processor through one of the high speed parallel interfaces.

The HW platform used for implementation is shown in Figure 3.

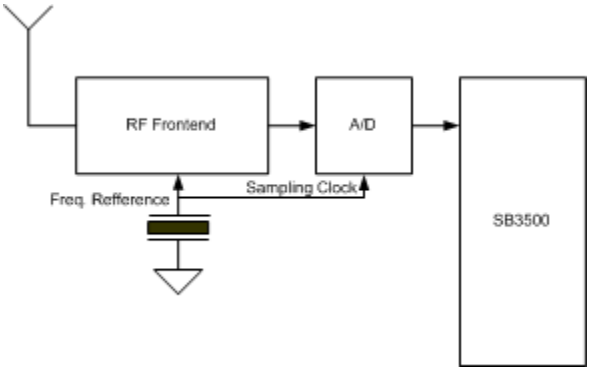

Figure 3 HW block diagram

The based band processing is described by a state machine partitioned in three major states as illustrated in Figure 4. The non real time satellite search processing blocks (state 1 and 2 in the state machine flow chart) are shown in Figure 5 while the tracking is illustrated in Figure 6.

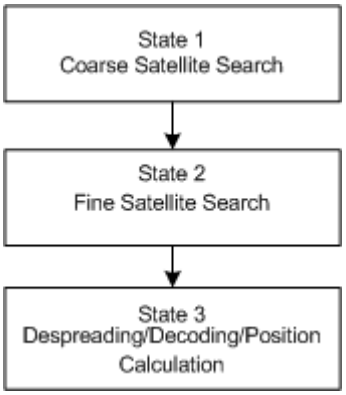

Figure 4 Receiver state machine flow chart

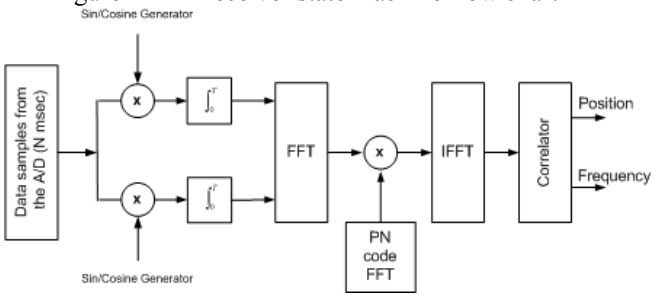

Figure 5 Simplified block diagram of the satellite search engine.

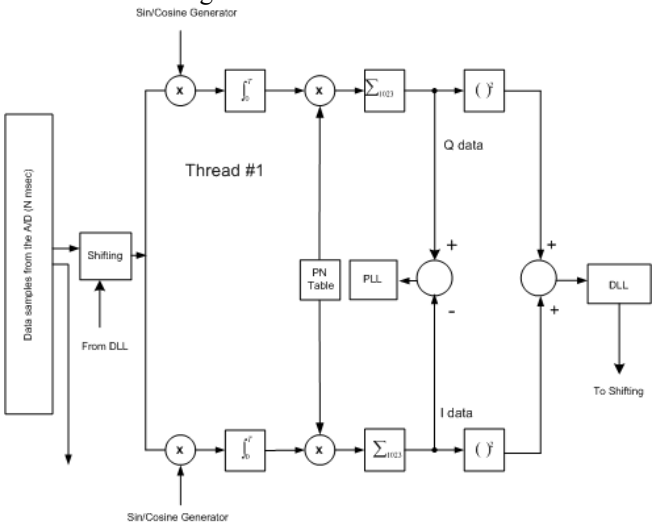

Figure 6 State 3 processing blocks for one of the early, late or prompt branch

2.1. Satellite Search

Initial satellite search is performed on 7ms of collected ADC samples (1023*16*7 samples, 2bits/sample) at sampling rate Fs=16.368MHz. For every satellite in the search list, a search of Doppler frequency is performed starting from an initial Doppler frequency range of -8000Hz to 9000Hz, with an initial Doppler search bin of 1000Hz. Each time when a search iteration is completed, the Doppler bin is reduced to 1/10 of the previous bin, the search for that satellite is completed when the Doppler bin is reduced to within 1Hz (+/- 20Hz from the true Doppler frequency guaranties correct decoding). Each 4096 point FFT/IFFT is performed on one millisecond worth of data and zero delay cross correlated over 7 milliseconds [6] .

The initial coarse search will locate the satellites that have the highest energy and will report for each satellite the Doppler frequency shift and an approximate sample position in one millisecond range. Since the initial search will only provide a coarse estimate of the initial sampling position that may not be accurate enough for the despreading operation, a fine sampling position search, in time domain, is performed for each detected satellite ranging a few samples left and right from the position reported by the initial coarse search.

2.2. Satellite Tracking

Real time satellite tracking is performed in State 3. The receiver State 3 is the steady state that performs the despreading operation, decoding operation and satellite position calculation. For each active satellite, the despreading operation is performed for 3 branches: early, prompt and late branches concurrently. Assume that the prompt branch takes samples from sampling position x, the early branch will take samples from sampling position x-8, while the late branch will take samples from sampling position x+8. Figure 6 shows the despreading operation for one of the three branches. The prompt branch's I/Q outputs are used by PLL to adjusts the LO's phase to track the Doppler shifts, while the DLL requires I/Q outputs from all the early, prompt and late branches. The DLL adjusts the sampling position for the despreading operation in order to compensate for any sampling clock drift. The prompt branch's output is also fed into the preamble detection and bit decoding.

The preamble detection will search for 2 preambles that are 6 seconds apart. Once found, the bit decoding can be started. The decoded bits at 50bits/sec will be collected to check CRC. For each frame of decoded bits, if all the CRC's are correct, the decoded frame will be used in the satellite position calculation. Four active satellites are used for the satellite position calculation. The final position estimation is made through Best Linear Unbiased Estimation (BLUE) [5] .

3. SW IMPLEMENTATION

A software GPS receiver has been implemented on the SB3500 SoC [4] . The SB3500 has three fixed-point DSPs implementing the Sandblaster 2.0 architecture. Of these, only one DSP is used for the GPS receiver. Each DSP is 4-way multithreaded with a 16-way 16-bit SIMD unit and 256KB of local memory. Among other features, the SIMD unit can do 4 radix-2 FFT butterflies in a single instruction.

The software GPS receiver is, in principle, a fairly direct fixed-point implementation of the algorithm described in the previous section. However, there were several challenges that had to be overcome during the implementation process. In this section, we shall describe some of them.

3.1. Memory vs. Parallelism

The SB3500 DSPs are 4-way multi-threaded. To fully utilize these threads, we have to split the task into 4 different components that can be executed in parallel. There are two models that can be used: independent or co-operative. In the independent model, the task is split into

disjoint portions, and each thread processes these disjoint portions. For instance, in the case of the coarse search, each thread could have picked 8 (i.e. ¼) of the satellites to process for the initial Doppler bin. Alternatively, the four threads could co-operate so that all 4 threads would work on the same satellite.

Each model of parallelization has advantages and disadvantages. The co-operative model requires more co-ordination between threads, leading to cycles being used for co-ordination rather than actually doing work. The independent model uses more memory. For instance, if we have four threads each doing a 4K point FFT instead of four threads working on the same FFT, we will need roughly four times the memory.

For states 1 & 2 of the algorithm, memory utilization concerns compel us to use a co-operative model of parallelization, For instance, the 4K point FFT in state 1 is implemented by having each of the 4 threads do a 1K point FFT, arrive at a synchronization barrier and them do the remaining 2 radix-2 FFT stages before coming to another synchronization barrier.

We have added special hardware on the SB3500 to decrease the cost of co-ordination; in particular, it is possible to implement a synchronization barrier so that all threads exit the barrier within a few cycles of the arrival of the last thread at the barrier.

In state 3 of the algorithm, the memory utilization is not very high. Consequently, we have chose to have 3 threads each track upto 4 satellites independently.

3.2. SIMD Sin/Cosine Generation

The sin/cosine generation routines must produce the sequence $e^{-j\theta n}$ for n=0…16383 and $|\theta| < 0.0123$ radians. To do this efficiently, we need to be able to use the SIMD unit. The SIMD unit uses 16 element vectors of 16-bit fixed point numbers. These 16 element vectors can also be treated as 8 element vectors of 16-bit complex fixed point numbers. Among other operations, the SIMD unit supports point-wise multiplication of 2 vectors of fixed-point complex numbers, yielding a vector of fixed-point complex numbers

The sin/cosine generation routine can be vectorized by initially computing the vectors $v_0=[e^{-j\theta 0},...,e^{-j\theta 7}]$ and $m=[e^{-j\theta 8},...,e^{-j\theta 8}]$. Then by repeatedly point-wise multiplying v by m, we can compute the necessary sequence. Specifically, after the first multiplication $v_8=[e^{-j\theta 8},...,e^{-j\theta 15}]$, after the second $v_{16}=[e^{-j\theta 16},...,e^{-j\theta 23}]$, and so on.

One problem that arises is that for larger n, the accumulated error from the repeated multiplications starts becoming significant. To compensate for this, we initially compute $M=[e^{-j\theta 256},...,e^{-j\theta 256}]$. Instead of computing v_{256} as $m*v_{248}$, we compute $v_{256} = M*v_0$. Similarly, we use M to compute all v_{k*256}.

When 4 threads co-operate to perform sin-cosine generation, each thread is used to compute ¼ of the sequence. For instance, to compute a sequence of length 1024, thread 0 computes $e^{-j\theta n}$ for n=0..255, thread 1 for n=256..511 and so on. There is an small non-parallel component where the starting vector for each thread is computed (in this case, v_0, v_{256}, v_{512}, and v_{768}). After that each thread proceeds independently to produce the 256 elements in its sub-sequence. They finish the sequence generation by synchronizing at a barrier.

3.3. 3rd State Implementation

In the implementation of the 3rd state, one thread does the ephemeris calculation while the other 3 threads are used to do the despread/satellite tracking/DLL-lock. Each of the 3 tracking threads indpendently process 1/3rd of the satellites detected after states 1 & 2.

The DLL-lock process requires 2 additional despreading steps for early and late. In practice it is not necessary to do the DLL-lock process for every sampling position. Consequently, to save compute, at each sampling position each thread will only do the DLL-lock process for one of the satellites it is tracking.

The thread that is doing the ephemeris computation uses floating point so as to obtain the high accuracy needed for the position computations. Since the SB3500 DSP has no hardware floating point, this is done using software emulation of floating point arithmetic.

4. PERFORMANCE

The GPS receiver was tested in the lab using the 12-channel Spirent 4500 as well as in the field. It can do a full sky-search in under 2s, locating upto 12 satellites. It can further decode the ephemeris and get a first fix in a worst case of 36s. The positioning precision of less then 10 meters with the decoded ephemeris data and less then 3 meeters with precise ephemeris. The total receiver sensitivity with an 18dBi active antenna is -147dBm.

5. CONCLUSIONS

In this paper, we have shown how a software defined GPS receiver has been implemented on the SB3500 platform. Experiments have shown that this receiver is competitive with custom hardware solutions in terms of time-to-first-fix, performance under adverse noise conditions and in terms of accuracy.

6. APPENDIX

6.1. GPS Signal

For one satellite, the L_1 BPSK modulated signal described in [1],[2] can be written as:

$$L^{i}{}_{1}(t) = A[P_i(t) \oplus D_i(t)]\cos(2\pi f_1 t) + \\ + \frac{1}{\sqrt{2}} A[Y_i(t) \oplus D_i(t)]\sin(2\pi f_1 t) \qquad \text{(A1)}$$

Where: A is the amplitude, P_i is the C/A code, 1023 long, for satellite i, f_1 is the L_1 carrier frequency (1.57542 MHz), D_i is the data associated with the satellite i, Y_i is the $P(Y)$ code 10230 long and $\oplus$ means XOR operation.

In the time domain, for the C/A code, the composite signal received from N_s visible satellites can be modeled using the pulse function for windowing the chips. One full code length has 1023 chips:

$$s(t) = \sum_{i=0}^{N_s-1}\left\{\sum_{k=-\infty}^{+\infty} g\left(t - k\frac{N_p}{f_i}\right) A_i d_i[k] \times \right. \\ \times \left\{\sum_{n=-\infty}^{+\infty} g\left(t - n\frac{1}{f_i}\right)\left[P_i[(n+n_i')\%N_p]\,\mathrm{Re}\left[e^{j2\pi f_i t + \varphi_i}\right] + \right.\right. \\ \left.\left. + \frac{1}{\sqrt{2}} Y[n]\cdot \mathrm{Im}\left[e^{j2\pi f_i t + \varphi_i}\right]\right]\right\}\Bigg\} + \eta(t) \qquad (A2)$$

Where: $g(t-mT_i)=\sigma(t-mT_i)\ \sigma[(m+1)T_i-t)]$, $f_i=1/T_i$ is the carrier frequency,

$\sigma(t) = \begin{cases} 1\ for & t \geq 0 \\ 0\ for & t < 0 \end{cases}$ is the unit step function, $d[k]$ is the data in the k^{th} millisecond, $P_i[n]$is the n^{th} chip in the k^{th} millisecond, A_i is the amplitude for the i^{th} satellite, % means modulus operation and, $\eta(t)$ is the thermal noise.

Equation (A2) can be rewritten as:

$$s(t) = \sum_{i=0}^{N_s-1}\sum_{n=-\infty}^{+\infty}\sum_{k=-\infty}^{+\infty} d_i[k] g\left(t - k\frac{N_p}{f_i}\right) g\left(t - n\frac{1}{f_i}\right) + \\ + \left[P_i[(n+n_i')\%N_p]\cdot \mathrm{Re}\left[e^{j2\pi f_i t + \varphi_i}\right] + \right. \\ \left. + \frac{1}{\sqrt{2}} Y[n]\cdot \mathrm{Im}\left[e^{j2\pi f_i t + \varphi_i}\right]\right] + \eta(t) \qquad (A3)$$

Since we are interested only in the C/A code we assume that the P[Y] code contribution will act as random noise and it will be incorporated in the noise term. For the C/A code only, (A3) will become:

$$s(t) = \sum_{i=0}^{N_s-1}\sum_{n=-\infty}^{+\infty}\sum_{k=-\infty}^{+\infty} d_i[k] g\left(t - k\frac{N_p}{f_i}\right) g\left(t - n\frac{1}{f_i}\right) \times \\ \times P_i[(n+n_i')\%N_p]\cdot\cos(2\pi f_i t + \varphi_i) + \eta(t) \qquad (A4)$$

7. REFFERENCIES

[1] Elliot D. Kaplan, "Understanding GPS Principles and Applications", Artech House Inc. 1996.

[2] Global Positioning System Standard Positioning Service Signal Specification, GPS NAVSTAR 2nd Edition, June 2, 1995

[3] E. O. Bringham, "The Fast Fourier Transform And Its Applications", Prentice-Hall Inc. 1988.

[4] M. Moudgill, J. Glossner, S. Agrawal, and G. Nacer, "The Sandblaster 2.0 Architecture and SB3500 Implementation", in Proceedings of the Software Defined Radio Technical Forum (SDR Forum '08), Washington DC, October, 2008.

[5] Steven M. Kay, "Fundamentals of Statistical Signal Processing Estimation Theory", PTR Prentice-Hall, Inc., 1993

[6] Kai Borre, Dennis M. Akos, Nicolay Bertelsen, Peter Rinder, Soren Holdt Jensen, "A Software-Defined GPS and Galileo Receiver", Birkhauser, Boston 2007.