A Novel Software Defined Research Receiver Architecture

Staffan Backén, Magnus L. Nordenvaad, Dennis M. Akos *†

BIOGRAPHIES

Staffan Backén completed his M.Sc. in electrical engineering at Luleå University of Technology in 2004, and his Licentiate degree in Electrical Engineering in 2007 from Luleå University of Technology, Luleå, Sweden. He is currently pursuing a PhD degree in electrical engineering at Luleå University of Technology. His research focus is on GNSS array processing and receiver architecture.

Magnus L Nordenvaad completed the PhD degree in Signal Processing at Chalmers University of Technology, 2003. He is currently a senior lecturer at the CSEE department where he specializes in statistical signal processing. He also holds a senior research position at the Swedish defense research agency where he develops antenna array solutions for underwater communications and underwater surveillance systems. Dr Nordenvaads primary research field is the estimation of complex covariance matrices and its applications in adaptive beam forming.

Dennis M. Akos completed the PhD. degree in Electrical Engineering at Ohio University within the Avionics Engineering Center. Currently he is an Associate Professor with the Aerospace Engineering Science Department at University of Colorado at Boulder and holds a visiting professor appointment at Luleå University of Technology and a consulting professor appointment with Stanford University.

ABSTRACT

Software defined receivers (SDR) are an increasingly important tool within the GNSS research community as the high level of flexibility offer a significant advantage over traditional hardware implementations. Over the last decade, software receivers have been used to investigate techniques as diverse as bi-static radar (additional correlators), multipath mitigation techniques, GPS/INS integration and array processing.

Mentioned above are only a few examples of features that could be required of an SDR, other include support for new signals (Galileo, GPS L5), multiple data file formats, high sensitivity and support for very long data sets. The large number of available features should ideally be coupled with program simplicity (such that other people can understand the program) and efficiency. This paper discusses these issues and proposes several solutions such as generalized data buffers (that is trivial to extend for new data formats) and a unified tracking structure (regardless of signal modulation). Examples are given using a Matlab implementation based on the Borre/Akos book Software-Defined GPS and Galileo Receiver", however with significant modifications. Where

*This work was supported by the Swedish Defence Research Agency (FOI) and in part by the Swedish National Graduate School of Space Technology.

†S. Backén, M.L. Nordenvaad and D. M. Akos are with the Department of Electrical Engineering and Computer Science, Luleå University of Technology, Luleå, 97187 Sweden e-mail: staffan@csee.ltu.se

critical, Java is used to increase performance while maintaining cross platform compatibility. Near real-time operation is available under optimal circumstances and the receiver currently supports GPS C/A- and GPS P-code signals.

1 INTRODUCTION

Satellite navigation, being both an impressive technical achievement and a solution to a difficult problem, attracts significant attention from researchers worldwide. The explosion in civilian use of GNSS over the last decades is mainly due to the advances in microelectronics. Due to the high throughput required from the signal processing algorithms, hardware based solutions such as ASICs (application specific integrated circuits) have historically been favored. SDR;s are an alternative, where generic processors are used for the computations. From a research perspective, this approach offers several advantages. The only required hardware is a radio front-end (as opposed to the design of an ASIC), and an interface to the processing engine. Everything else is performed in software, allowing for rapid development and implementation of new techniques.

Although software GNSS radio have been investigated since the mid nineties (see for example [1]), it is during the last six or seven years that research into SDR have taken off. In 2007, Borre et al. [2] published a book accompanied by an open source implementation of an SDR in Matlab. Several research institutions/universities have modified this code (for example [3]) or developed their own (such as [4] or [5]). With the GPS and GLONASS modernization programs and the launch of two new systems, Galileo and Compass, GNSS receivers will potentially have access to a wide range of different signals. Further, several extensions to stand-alone satellite navigation are actively investigated, most notably array processing and IMU integration.

An SDR targeted at research applications should ideally be efficient, customizable and simple to understand, and thus is no different from other large software projects. However, with the exception of the book from Borre et al., not much has been published on the architectural details of complete software defined receivers.

In this paper, key architectural decisions for a feature-rich SDR targeted at research applications is presented. In the next section, an overview of the software is presented, followed by more detailed discussions about select parts of the receiver.

A case study is also presented, where the effect of CW interference on C/A- and P-code is briefly investigated. Finally, the paper is concluded.

2 OVERVIEW

This section provides a brief summary of front-end architectures and GNSS signals. The receiver functionality is briefly summarized and receiver settings are explained. It is intended to provide sufficient background to understand more detailed descriptions in later sections. For readers new to GNSS receivers, I recommend the book "A Software-Defined GPS and Galileo Receiver" [2].

2.1 Front-end architectures

A GNSS consists of a number of satellites (24-30 is typically required for global coverage) transmitting signals in the L-band (around $1 - 2\,\mathrm{GHz}$) from a medium or geostationary orbit. The signals have low power (on the order of $100\,\mathrm{W}$) but are transmitted continuously. The signals propagate through space and eventually impinge on the antenna connected to the receiver. In order to relax the requirements on the ADC and subsequent processing elements, the received signal is ampli-

fied, filtered and mixed down. There are two main front-end architectures; superheterodyne and direct conversion. In the former, the received high-frequency signal is mixed to a lower frequency (between 2 and 20 MHz). After this, an ADC converts the analog signal to a digital replica. The second main architecture, direct conversion, use two mixers offset 90° to generate both I (inphase) and Q (quadrature) components. These are treated as the real and imaginary part of the signal. With this latter architecture, the signals may be mixed to nominal baseband.

A research receiver, such as the one presented in this paper, should be capable of processing data output from either architecture. Further, as signals are transmitted on several frequencies, front-ends capable of receiving multi-frequency data should be supported. This may require support for data in several files, a feature that is also needed for processing bi-static radar or data from array antennas.

2.2 Signal model

If we assume a perfect sampling clock, K GNSS signals can, after being received by the antenna and conditioned (amplified, filtered, mixed and sampled) in the front-end, be generally modeled as

$$s[n] = \omega[n] + ... + \sum_{k=0}^{K-1} \sqrt{P_k} \cdot M_k \left(t_0 + \frac{n}{f_s} + \delta t_k \right) \cdot e^{j\omega_k n + \phi_k} \quad (1)$$

where $s[n]$ is the received signal at sample n, ω is the noise, P is the received signal powers, M is the signal modulations. Further, t_0 is the time at sample 0 and δt is the transmit time. ω and ϕ are the angular frequencies and phases of the carriers respectively.

The modulations M contain two components; the first is information about the satellites time and the satellites position as a function of time (ephemeris). This, together with estimates of the relative time of arrival between different satellites at a sample n allows solving for the 4 unknowns, 3D position and time. The second component consists of a code, typically CDMA-SS (code division multiple access spread spectrum). Older signals are usually implemented using BPSK (binary phase shift keying) while modern signals (such as the E1 or E5 Galileo signals) usually have a slightly more complicated structure such as BOC (binary offset carrier) or MBOC, AltBOC etc. As a simple example, we can consider the GPS C/A signal. This BPSK signal is 1 ms long and have 1023 chips per code period. The code is a gold code, chosen for its good auto- and cross-correlation properties. There is also a 12.5 minute's long data message overlaid on the code with a bit rate of 50 Hz. Let TOW denote the number of chips since the start of the week, then

$$TOW = \left(t_0 + \frac{n}{f_s} + \delta t_k \right) \quad (2)$$

This gives, when the signal is a GPS C/A code signal

$$M_{k_{C/A}}(TOW) = CA(\lfloor (TOW \cdot f_r) \mod 1023 \rfloor) \cdot DB(\lfloor (TOW \cdot f_r) \mod (20 \cdot 1023) \rfloor) \quad (3)$$

where CA is the C/A code, DB is the data bits and f_r is the code rate (nominally 1.023 MHz). The principal interest is measuring the flight time δt_k, and this is achieved by estimating the start of a code. Please note that the above explanation is somewhat simplified (the sample clock is not identical, for example), but should provide sufficient background for the rest of the paper.

2.3 Receiver overview

The first stage of the receiver is the initialization. Here, the environment is prepared; sub directories are added to the Matlab search

path, settings are loaded etc. After the initialization, the processing is started. During processing, the receiver executes the following functions continuously:

Acquisition
Perform acquisition on signals that should be processed, but whose code locks are `false`. For each acquired signal, set code lock to `true`.

Tracking
Track all signals where code lock is `true`.

Data decoding and lock detection
Decode data bits and update lock status.

(Pseudo-range computation)
Compute pseudo-ranges. Not implemented yet.

(Position solution)
Compute a position solution. Not implemented yet.

Write data
Write information (such as correlator values) to disk.

2.4 SDR Settings

All settings for the receiver are defined in the function `initSettings.m`. The output is a data structure with many different fields. Receiver wide settings, such as filenames and how long time to process, is defined first. More specialized settings, such as acquisition and tracking settings, are defined using subfields. For example, `settings.fs` denotes the receiver wide sampling frequency, while `setting.acq` is in itself a data structure with several fields, all of them related to acquisition.

The settings file also defines which signals the receiver will attempt to acquire and track, as well the configuration of the IF data plug-ins.

3 FUNCTIONALITY AT THE IF LEVEL

When samples are requested from disk, the first step is to check whether this data have been requested before and thus resides in memory. If this is the case, data do not need to be read from disk again. If not, the data on disk are processed in four distinct stages. First, they are read from disk and converted to double precision floating point numbers, either real or complex. If the data source is a direct conversion front-end, the data is stored as complex numbers, otherwise it is at this stage stored as real numbers.

The second stage consists of applying the IF plug-ins defined in `initSettings.m` to the data. A wide variety of functionality is supported through this interface. For example, CW generation, 1-bit ADC and PSD (Power Spectral Density) export have currently been implemented. The plug-in is implemented as a simple Matlab function, whose input arguments are the data (a double precision M by N matrix where M is the number of data files and N is the number of samples) and the first sample number. The output should also be an M by N matrix. Future IF plug-ins may include other noise sources (AWGN, pulsed interference), AGC, IF export, resampling, a GNSS simulator, jitter generator and an adaptive beamforming algorithm.

In the third stage, the data is mixed to nominal baseband. The benefit of this approach is that the data from now on are always complex and at nominal baseband. Thus, the only data dependent parameter that acquisition or tracking need to consider is sampling frequency (there is however one additional parameter, spectrum mirroring, that may have to be taken into account). Although this introduce a small performance penalty, the resulting simplification in receiver design is beneficial for a research receiver.

The fourth and final stage consists of a data

buffer as mentioned previously. The default setting is three large data buffers, each capable of storing around 1 second of data. Whenever data are requested that are not stored in memory, 1 second of data is read from disk, IF plug-ins are applied and the data is mixed to nominal baseband as described above. The data resulting from this operation is stored in the oldest data buffer, thus discarding that information. This reduce the amount of data read from disk to a minimum. Currently, data may only be requested that fits in a maximum of two buffers.

4 SIGNAL DATA STRUCTURE

Currently, many different GNSS are transmitting different signals. The American GPS are currently operational while the Russian GLONASS is rapidly approaching a full constellation. The European system Galileo as well as the Chinese Compass have launched test satellites. In addition, numerous regional systems (such as the augmenting systems WAAS and EGNOS) are in operation or development. This section describes how the receiver is designed to support as many different signals as possible, requiring a minimum of changes.

Despite the wide array of available or planned signals, the techniques used to acquire and track these signals are similar (see section 2.2. The signals consist of a carrier multiplied with a CDMA (code division multiple access) modulation. However, the type of modulation differs; BPSK (binary phase shift keying) is typically used for generic signals while modernized signal tend to prefer a BOC (binary offset carrier) modulation.

In the receiver, all signals are described using a data structure, defining all parameters for the codes. The design of this structure is currently not finished, however the concept is straight forward. A shortened example of the signal for the C/A and P-code definition for PRN1 is shown in table 1.

Entries in the table surrounded by parenthesis denote possible vectors, and entries suffixed by the "@" signal is a function handle (for example, the field "codeDiscr" points to a function implementing the early power minus late power discriminator function). Using this approach, all proposed GNSS signals can be modeled, and integration of new signals should be straight forward.

There is one additional parameter in the signal data structure, `Correlator Matrix`, which has interesting properties. The first row contains correlator spacing's (in units of chips), and the second row is a file index. The default setting for BPSK signals are

$$\begin{bmatrix} -0.5 & 0 & 0.5 \\ 1 & 1 & 1 \end{bmatrix} \tag{4}$$

while BOC signals may require more correlators (the BOC modulation introduce a peak ambiguity). This matrix may be overridden in the `settings` file, and it is flexible enough to allow for bi-static and array processing with relative ease.

5 STATE DATA STRUCTURE

The receiver processes data in blocks, as mentioned in section 2.3. In other words, the tracking function will track a signal for a short time (around 1 second) and then exit to the parent function. This means that the current state of the channel must be preserved between function calls. The data structure containing this information is called `state` in the code, and its layout is as follows:

- `ID`
- `Current sample`
- `End of file`
- `Code`
- `Carrier`
- `Lock`
- `TOW`

Field Index	C/A 1	P 2	Explanation
Name	'GPSCA'	'GPSPL1'	Name of system and signal
ID	1	1	PRN number
Modulation	'BPSK'	'BPSK'	Type of modulation
Coderate	1.023M	10.23M	Code rate
Symbolrate	50	50	Data bit rate
Code	[-1, 1, 1, ...]	@generatePcode	Code
acquireable	1	0	Is acquisition possible
codeDiscr	@discr__EML	@discr__EML	Discriminator function
sibling	[2]	[1]	Link to other signals
siblingPhase	[90]	[-90]	Phase offset to
...	...	...	

Table 1: Example of the layout of the signal data structure

The latter four are in turn data structures. The fields `Code` and `Carrier` contain information regarding the code- and carrier tracking, such as filter settings, frequency estimates etc. The field `Lock` contains information about the current code lock and phase lock, and the field `TOW` describes the time of week as a function of sample number.

6 PERFORMANCE

Critical parts of the receiver code have been implemented in Java in order to improve performance. These include the core mixing and correlation functions, but also P-code generation (that involves a large number of bitwise operations). The mixing and correlation function also include a general LUT (look up table) for a sinusoid that can, if enabled, be used in other parts of the receiver. Using a modern desktop PC (AMD Phenom II X4 3.2GHz, 8GB RAM, XP64, MATLAB2009a) the receiver is capable of processing 10 C/A signals at 14% of real-time performance when $f_s = 2.048\,\text{MHz}$. For the P-code, 1 signal can be processed at 10% of real-time when $f_s = 30\,\text{MHz}$. Please note that the receiver is currently only utilizing a single core, and if this is addressed, near real time operation is feasible for low sampling frequencies.

7 IMPACT OF CW INTERFERENCE ON C/A- AND P-CODE

In this section, a small illustration into the impact of CW interference on the pseudo range accuracy of C/A- and P-code will be presented. A data set, recorded from a simulator capable of generating simultaneous C/A- and P-code, have been provided to the authors from FOI. Two test will be investigated, each using around 5 minutes of IF data. A reference pseudo-range has been computed using carrier aided P-code tracking. In all tests, PRN 32 has been processed.

Figure 1 shows the results from the first test. The topmost subplot show the pseudo-range accuracy of C/A- and P-code tracking that was not carrier aided. The second subplot show the performance when CW interference was enabled, and the third subplot shows the performance with both CW interference and 1-bit quantization. The bottom subplot shows the settings for the CW generation. From 10 seconds until 150 seconds the amplitude remains constant at 500, while the

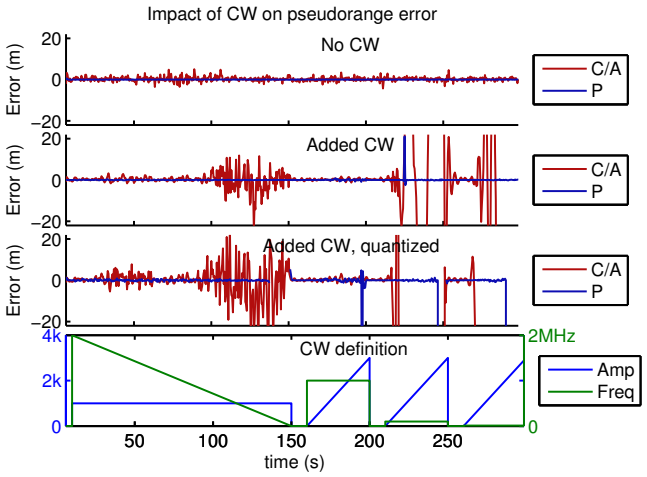

Figure 1: Impact of high frequency CW interference on pseudo range accuracy for C/A and P-code signals.

frequency starts at 2 MHz and ends at 0 MHz. During the second half of the data set, the frequency is kept constant at 1 MHz, 100 kHz and 10 kHz while the amplitude is ramped up from 0 to 3000.

The JNR (jammer to noise ratio) over the entire 30 MHz band can be computed as

$$JNR = 10 \log_{10} \left(\frac{A^2}{2\mu} \right) \tag{5}$$

where A is the amplitude of the CW signal and μ the variance of the noise. The variance for this particular data set is around 157 k giving a JNR of around 15 dB when the amplitude is 3000.

Let us first consider the case of only CW interference. It is evident that the C/A code is severely affected when the frequency of the interferer is smaller than 1 MHz. The P-code is largely unaffected regardless of frequency within the tested amplitude and frequency span. This is because the P-code signal does not repeat (as opposed to the C/A code).

It is also interesting to consider the impact of quantization with CW present. Typically, a GNSS receiver have around $1 - 8$ bits of resolution. In interference free environments, a 1-but quantization introduce only a 2 dB loss in C/N_0. In the presence of strong CW interference however, the sinusoid will capture the ADC, and thus reduce the performance. This effect can be seen in the lower subplot of figure 1. When 1-bit quantizing is enabled the impact is, as expected, more severe. The noise on the C/A code is increased and the P-code is also losing lock in several instances.

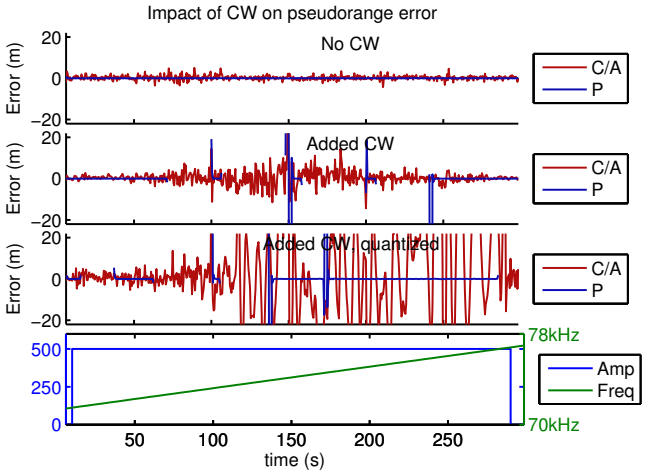

Figure 2: Impact of low frequency CW interference on pseudo range accuracy for C/A and P-code signals.

The results from the second test are shown in figure 2. Work by Balaei et al. [6], suggest that the impact of CW is most severe at the frequency of the signals maximum peak in the frequency domain. For this C/A signal (PRN32), the highest frequency component is at ± 74 kHz. This suggests that the impact of CW should be largest at either of these frequencies relative to the frequency of the signal itself. A CW interference with constant amplitude was generated, and the frequency was swept between 72.5 kHz and 75.5 kHz relative to the signal frequency. As expected, the impact is most severe at 150 seconds when the relative frequency is 74 kHz. The P-code also loses lock on several occasions, but it should be noted that phase lock on the C/A code is required for the P-code tracking to commence, i.e. no P-code acquisition have been imple-

mented. This explains the rather long P-code outages (between 105 and 145 seconds for example).

When quantization is enabled, the situation is worse. The accuracy of the C/A code is degraded, and the P-code loses lock at several intervals. Please note that the JNR in this test (with an amplitude of 500) is only around -1 dB. Although the performance could be improved by higher dynamic range, longer integration time or interference mitigation techniques, GNSS signals are sensitive to CW interference.

8 CONCLUSIONS

A research GNSS software defined receiver has been presented, together with motivations around key architectural decisions. The receiver is written primarily in Matlab (critical parts are implemented in Java), and designed to be flexible, feature-rich and reasonably efficient. It has been designed to support all known front-end architectures and many different signals, as well as provide support for bi-static and array processing. Several novel features have been presented, such as the nominal baseband mixing and the IF plug-ins. A case study have also been presented, where the impact of CW interference on C/A- and P-code data have been investigated.

Please not that the receiver is currently a work in progress, however no major changes are anticipated regarding the specific areas highlighted in this paper. The receiver code will eventually be made publicly available under the GPLv2 license.

References

Akos, D.M. (1997). A Software Radio Approach to Global Navigation Satellite System Receiver Design. *PhD thesis, Ohio University*

Borre, K. and Akos, D.M. and Bertelsen, N. and Rinder, P. and Jensen, S.H.(2007). A Software-Defined GPS and Galileo Receiver. *Birkhuser*

Gleason, S. and Quigley, M and Abbeel, P. (2009). An Open Source AGPS/DGPS Capable C-coded Software Receiver. *Proceedings of the ION GNSS*

Mark G. Petovello, M.G. and ODriscoll, C. and Lachapelle, G. and Borio, D. and Murtaza, H. (2008). Architecture and Benefits of an Advanced GNSS Software Receiver. *Journal of Global Positioning Systems*

Humphreys, T.E. and Bhatti, J.A. and Pany, T. and Ledvina, B.M. and O'Hanlon, B.W. (2009). Exploting Multicore Technology in Software-Defined GNSS Receivers. *Proceedings of the ION GNSS*

Balaei, A. T. and Dempster, A.G. and Barnes, J. (2006). A novel approach in detection and characterization of CW interference of GPS signal using receiver estimation of CNo. *Proceedings of the IEEE/ION PLANS*