GNSS-Receiver with open interface for deeply coupling and vector tracking

Matthias Overbeck, Dr. Fabio Garzia, Christian Strobel, Christian Nickel, Muhammad Saad, Daniel Meister, Dr. Wolfgang Felber,
Fraunhofer Institute for Integrated Circuits IIS, Erlangen/Nuremberg, Germany

BIOGRAPHY

Matthias Overbeck received his Dipl.-Ing. (MSc) degree in Electrical Engineering from the University of Erlangen-Nuremberg, Germany, in 1999. Since September 1999 he is working at the Fraunhofer Institute for Integrated Circuits (IIS) in the Power Efficient Systems Department on the topic hardware development for global navigation satellite system (GNSS) receivers. Within this scope he developed digital baseband hardware like signal conditioning, correlators and microprocessor for field-programmable gate array (FPGA) and for application-specific integrated circuits (ASIC). In 2009 he has taken over for Fraunhofer the technical leadership of the GSA project Galileo Receiver for Mass Market Applications in the Automotive Area (GAMMA-A). Since 2013 he is the Group Manager of Precise GNSS Receivers at Fraunhofer IIS.

Dr. Fabio Garzia received his MSc degree as Electronics Engineer in March 2005 from the University of Bologna, Italy. After that he worked for a few months at ARCES Labs in Bologna. In 2006 he moved to Tampere (Finland) to carry out his PhD studies at the Tampere University of Technology. He received his PhD degree in December 2009. In October 2011 he moved to Germany, where he joined the Power Efficient Systems Department at Fraunhofer IIS. His main task is the development of digital hardware and HW/SW interfaces for GNSS receivers targeting both ASIC and FPGA technologies.

Christian Strobel received his Dipl.-Ing. (MSc) degree in Engineering Computer Science from the University of Technology, Ilmenau, Germany, in 2012. From July 2009 to June 2012 he worked at the Fraunhofer IIS in the Power Efficient Systems Department as Student Assistant on GUI programming with Qt as well as Linux kernel driver and firmware development. Since June 2012 he is working the Fraunhofer IIS in the Power Efficient Systems Department as Research Assistant. His main tasks are Software design and development for high precise GNSS receivers in C/C++ and Python™ as well as HW/SW interfaces for GNSS receivers.

Christian Nickel received his MEng degree in Electronic and Mechatronic Systems from Georg Simon Ohm University of Applied Sciences, Nuremberg, Germany, in 2015. From October 2012 to April 2015 he worked as Student Assistant in the Multisensor Systems Group, Power Efficient Systems Department at the Fraunhofer IIS. His main tasks were to develop and evaluate algorithms for motion detection and indoor localization of pedestrians based on inertial sensors and wifi fingerprinting. Since May 2015 he is mainly working on state estimation and sensor fusion algorithms in the same group as Research Assistant.

Muhammad Saad received his Bachelor (BSc) degree as Mechatronics Engineer in October 2012 from University of Engineering and Technology, Peshawar, Pakistan. In April 2014 he moved to Germany, where he started his Masters (Msc) in Embedded Systems Design from Hochschule Bremerhaven. In September 2015 he started internship in Navigation Group, Power Efficient Systems Department at Fraunhofer IIS. During his internship he was involved in designing and testing of some dedicated hardware modules in FPGA. In May 2016 he started his Master Thesis titled "Implementation of robust vector tracking for open-interface GNSS receiver" in the same group. Currently he is working on the algorithm and implementation part.

Daniel Meister received his MSc degree in Computer Science from Georg Simon Ohm University of Applied Sciences, Nuremberg, Germany in 2015. From October 2009 to December 2015 he worked as Student Assistant in the Power Efficient Systems Department at Fraunhofer IIS. In January 2016 he joined the group as Research Assistant. His main tasks are software design, software development and performance optimization in the area of sensor fusion, state estimation and localization.

Wolfgang Felber received his Dipl.-Ing. (MSc) degree in electrical engineering in 2002 and his doctoral degree Dr.-Ing. in 2006 from Helmut-Schmidt-University – University of Federal armed Forces Hamburg. Since 2014 he is the head of department Power Efficient Systems in Nuremberg. The main topics in his department are energy harvesting and low power technologies, hardware development of

satellite navigation receivers and sensor fusion in positioning applications.

ABSTRACT

Each system using deeply coupling of sensors demands its own parameterized algorithm regarding geometry, Inertial Measurement Unit (IMU) and e.g. vehicle data. The present GNSS-receiver market is not flexible enough to support each application and each special market. Nowadays deeply coupling solution is always a fixed combination of a high performance IMU with a high performance GNSS receiver. This paper describes a GNSS receiver development platform with an open interface to allow deeply coupling of inertial sensors and vector tracking on user side.
In this work, two different approaches are adopted for development of an open interface for GNSS receiver. First approach uses standard ZeroMQ library, while in second case dedicated Application Programming Interface (API) is developed for external loop closure.
Furthermore, for testing of first approach, three example implementations are presented along with results and comparison with standard tracking loops. Results are presented for second case and compared with standard tracking loops, on single-board computer (SBC).

INTRODUCTION

GNSS receivers are used to calculate positions by measuring differential time of arrival of satellite signals in many different applications and also in harsh environments for electromagnetic waves. Machine steering (e.g. autonomous driving) demands high availability also in scenarios with shadowing and multipath. The idea of vector tracking is to raise the availability of satellites by bridging signal outages on some of them exploiting the geometry and movement of the satellites in the line of sight. With this procedure more satellites stay in track even if the satellite signal is shadowed for some time and reacquisition time, which takes up to several seconds, can be spared. The same principle is used by deep coupling of sensor information. The position estimation is fed back into the receiver and aids the tracking loops to stay in track even if the signal to all satellites is lost.
For deep coupling commercial hardware solutions can be found which are based on one professional multi-frequency GNSS-receiver [1]. In the referenced paper the receiver is used with a combination of a commercial grade fiber optical gyro (FOG) and a microelectromechanical systems (MEMS)-based accelerometer building up the IMU. The positions and carrier-phases of GPS are used to update the Inertial Navigation System (INS)-filter which is a so called tight coupling. In the other direction the inertial position and velocity is used to assist the tracking loops in a deep coupling. About half a dozen IMUs are ready to support the described receiver. Other IMU or sensor data accept wheel speed are not supported. This and the costs restricts the useful application scenarios.
Beside this there is a lot of research activity in the field of vector tracking [2, 3] and deep coupling [4]. In some approaches of the shelf signal recorder like the Ettus USRP N210 are used as a front end and the processing is done completely in post-processing. In other setups an off-the-shelf front-end for digitizing the satellite signals in combination with a high performance personal computer is used, running MATLAB®, Python™ or C++ executables.
In this paper we employ for this purpose the GNSS-receiver platform developed by Fraunhofer IIS in the GOOSE project (see [5] and [6]). The platform is characterized by three separate components: an analog-front end board, a baseband board and the processor system. The low-IF signals generated by the analog front end are processed in the baseband board. The baseband board is based on a FPGA, where different GNSS dedicated HW modules are implemented. These include a FFT-based acquisition core and 60 flexible data-pilot channels. The FPGA modules are controlled by the processor system, which can read and write the control and status registers. Additionally it is also possible to close the FLL/PLL/DLL loops in the tracking, i.e., reading the E/P/L values and writing back the carrier and code NCO control values. The PCIe interface has been employed for this purpose.
The GNSS-receiver platform has been designed in order to support both the deployment in a commercial PC as well as an embedded platform. In the PC version, a PCIe riser card allows to plug the receiver platform in the PCIe connector of a motherboard. For the embedded version, a SBC can be directly plugged-in in the baseband board.

This paper is organized as follows. Firstly we present the GNSS-receiver development chain based on the GOOSE platform. Then we describe the open interface, which can be used to collect measurements from the receiver as well as to close the tracking loops. Afterwards we describe how a deep coupling algorithm could be mapped on the platform. It follows the current implementation of vector tracking loops and related results. Finally we draw some conclusions.

GNSS-RECEIVER DEVELOPMENT CHAIN

In order to allow the users to develop new algorithms, methods and applications in the GNSS field, we provide with the Open-GNSS receiver a complete development chain which is presented in the following figures.

Each stage of the chain is characterized by a different run-time environment and debugging possibilities, but it has a common Hardware Abstraction Layer (HAL) and open software interface (Open GNSS Receiver Protocol (OGRP)), depicted as central elements in figures 1, 2 and

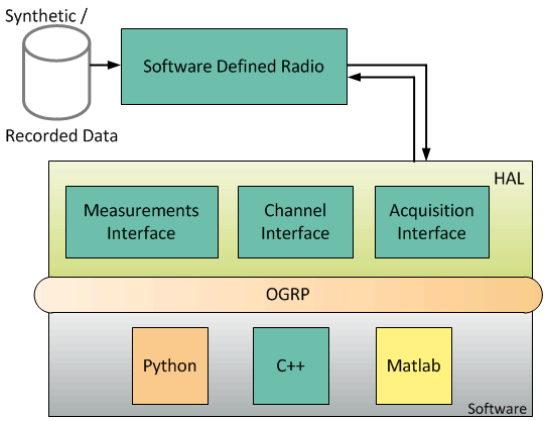

Figure 1. Data processing with Software Defined Radio

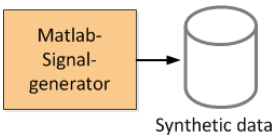

Figure 2. Generation of synthetic data with MATLAB®

5. The user can develop his/her software in different languages and communicate with the HAL through the open interface, described in Section 12. Because of the common interface, it is easier to port user software from one development stage to next one.

Software-defined radio receiver

The first stage of the development chain corresponds to software-defined radio (SDR)-receiver implemented in C++ shown in figure 1. This models the signal conditioning, acquisition and channel modules which are typically built in hardware. The input signal is parsed from a binary file which contains either a synthetic signal, generated e.g. with MATLAB® like in figure 2 , or recorded data with the help of the Open GNSS-Receiver (see figure 3). The main advantage of the SDR approach is that processing intensive algorithms can be implemented and analyzed as the system does post-processing on sampled files in a simulated time frame. Moreover breakpoints can be set for debugging purposes. With this complex functions can be monitored and validated which would not be possible in a real-time environment.
The combination of SDR plus the hardware abstraction

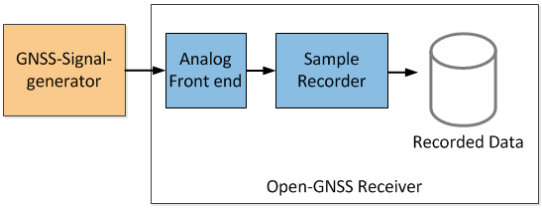

Figure 3. Recording of GNSS-signals with Open-GNSS Receiver

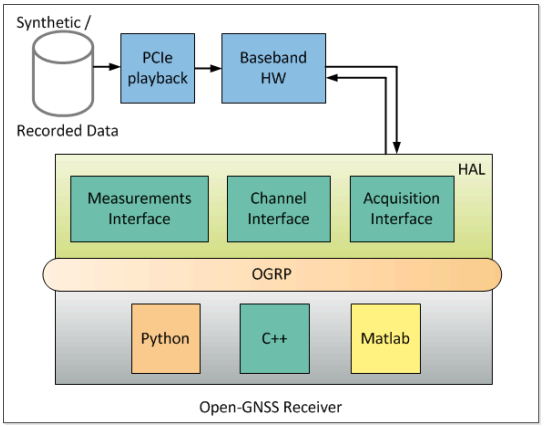

Figure 4. Data processing with Open-GNSS Receiver

layer (HAL) and the user software allows the user to track the synthetic signal and to calculate a PVT out of it. The user software can be implemented either in MATLAB®, Python™ or C++ without any hard real-time constraints. In particular, MATLAB® debugging with breakpoints has already been validated.

Hardware Receiver with PCIe playback input

The next stage of the development chain is based on the Open-GNSS Receiver developed in GOOSE project ([5], [6]). In this stage the baseband HW receives the digital signal through the PCIe interface instead of the analog-front end (see figure 4). The PCIe interface is also used to access the control register and perform the loop closure.
By using the acquisition core mapped on the FPGA, it is possible to detect the presence of a satellite, its Doppler and code phase. The detected signal can be processed in one of the available channels. Each channel can track any of the signals supported by the front-end. For L2C signal an additional time-multiplexed mode has been implemented to allow the separate tracking of data and pilot components. Regarding AltBOC signals, a dedicated subchip generator produces the sine and cosine subcarriers needed for their tracking.
Differently from the SDR receiver, in this case the signal is processed in real-time. The advantage for the users is that they can use the same test signals as in the previous step to have a reference behavior and reproducible results for easier debugging. In addition, it is possible to analyze how the user algorithm performs in a constrained real-time environment and what performance can be expected.

Hardware Receiver with analog front-end

In this third step of the development chain, the Open-GNSS Receiver gets its input signal from a analog-front end board (see figure 5).

The analog-front end is composed of three channels for separate GNSS bands. In each channel, the satellite signal is mixed down to a low intermediate frequency and con-

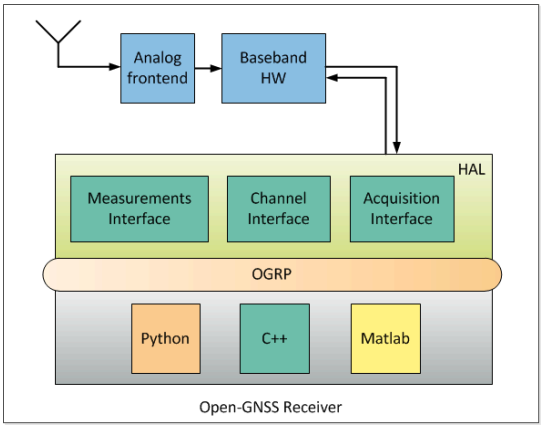

Figure 5. Schematic of Smart Antenna build up with Open-GNSS Receiver

verted to digital by a 81-Msps 8-bit ADC. The first channel supports GPS L1, Galileo E1, GLONASS G1 and BeiDou B1, the second one the L2/L2C-band and GLONASS G2 and the third one E5 AltBOC, E5a, E5b, L5 and B2.
This development step is characterized by different hardware versions which allow to extend the development chain. In the first version, the receiver is plugged into a PC. The user can develop its software by processing a real signal coming from the antenna, but using exactly the same development environment as before. In the embedded version, the baseband board is connected to a single-board computer instead of the PC through the same PCIe interface. In this case, the user is challenged by additional performance constraints related to the embedded processor. Finally, the smart antenna version provides a portable receiver. The user can finalize the software with field tests.

Figure 6. Smart Antenna build up with Open-GNSS Receiver

PROVIDED OPEN INTERFACE

In order to modify the conventional tracking loops of GNSS receiver program at tracking loops level, in an external environment, an open interface needs to be defined and developed, via which the receiver platform communicate/interacts with external environment and vice versa. This approach enables users to perform different techniques for loop closure, i.e. deeply coupling with IMU or vector tracking etc.. In addition, a comparative study and performance improvements scope in two cases, those are, conventional loops closure and loops closure with an open interface can also be performed. Based on the application requirements, two different strategies are adopted and tested, for the development of an open interface.
1) Open Interface via ZeroMQ:
ZeroMQ is a distributed-messaging library for inter-process communication which is available for the most common programming languages (C++, Python™, Java, etc.) and operating systems. It provides sockets that carry atomic messages using different transport protocols and communication patterns. The most common transport protocols are supported, among the others TCP, inter-process, in-process or multicast. The available patterns include request-reply, publisher-subscriber, push-pull and fan-out. The proposed interface is used in order to provide a generic which is flexible for future applications and needs. The advantage of this interface is the flexibility for the user. Additionally a ZeroMQ to MATLAB® module (MEX) is provided. Already implemented algorithms in many programming languages and MATLAB® can be tested. The disadvantage is the performance lose because of inter-process communication over the TCP layer on the SBC. Therefore a second interface option is provided for mobile applications on the SBC.
In order to take the benefits of this approach, three example implementations are presented in different programming languages, for external loop closure. In first case, external program is written in Python™ to close the loops, second example shows loops closure via open interface using C++ and for third case MATLAB® is used as an external environment. First two examples are real time, while MATLAB® is used in connection with SDR for development of vector tracking and deep coupling applications. Results are provided for each case, which shows good agreement of tracking performance.
2) API in C++:
This approach is adopted if the receiver program and external program needs to be run on SBC, for which first approach does not meet requirements because of the library and parsing overheads.
The OGRP is associated with the GNSS receiver platform described above. OGRP is an open interface protocol available at [7] and is foreseen to be extended and used by the GNSS-community. The goal of OGRP is to offer a well-defined and self-describing format for the available receiver

measurements, while to be vendor neutral at the same time. One of the challenges of the protocol is to support real-time message exchange between software modules as well as logging of the internal receiver status and measurements. To fulfill these requirements and to guarantee at the same time a human readable format, JavaScript Object Notation (JSON) has been used to implement the protocol. This way several libraries based on different programming languages can be used to parse the data. The usage of JSON simplifies the protocol extension through definition of new message types or extension of the defined ones.

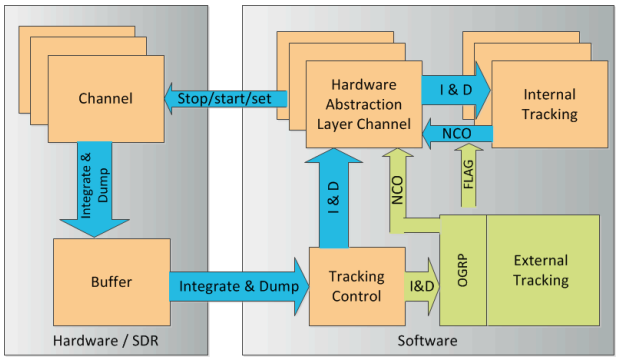

Figure 7. Concept of interfacing the external tracking

For deeply coupling the OGRP definition is extended by eight new messages:

1. Integrate & Dump (I&D) values from the hardware channels
2. Increment values for Code and carrier numerical controlled oscillator (NCO) to control the hardware channels by an external tracking
3. Acceleration and angular rate from IMUs
4. Wheel rotation speed and steering angle from vehicle
5. Ephemeris data from receiver platform to compute satellite position, velocity and clock information
6. User PVT result from receiver platform
7. Channel measurements from receiver platform
8. Satellite information from receiver platform containing satellite's azimuth, direction and elevation etc.

The concept of steering the channels by an external tracking is shown in figure 7. Each channel is started using the internal tracking. If a stable tracking state is reached a flag is set and the NCO values of the internal tracking are ignored, while the channel is steered by the IMU aided external tracking. The latter does not have to consider bit synchronization and integration time switching e.g. for GPS L1CA as this is already done before by the internal tracking. This reduces the effort to implement an aided external tracking. The internal tracking runs in background without steering the hardware channel, observes the tracking state and passes the navigation bits to the message decoder.

DEEP COUPLING AND VECTOR TRACKING DEVELOPMENT ENVIRONMENT

As described above a post-processing environment is useful for the development of vector tracking or a deep coupling filter for loop closure. This is provided in the first stage of the development chain. The Figure 8 shows the components used to close the loops on the one hand without using IMU (vector tracking) on the other hand with IMU (deep coupling). The 'IMU to ZMQ' component takes an IMU

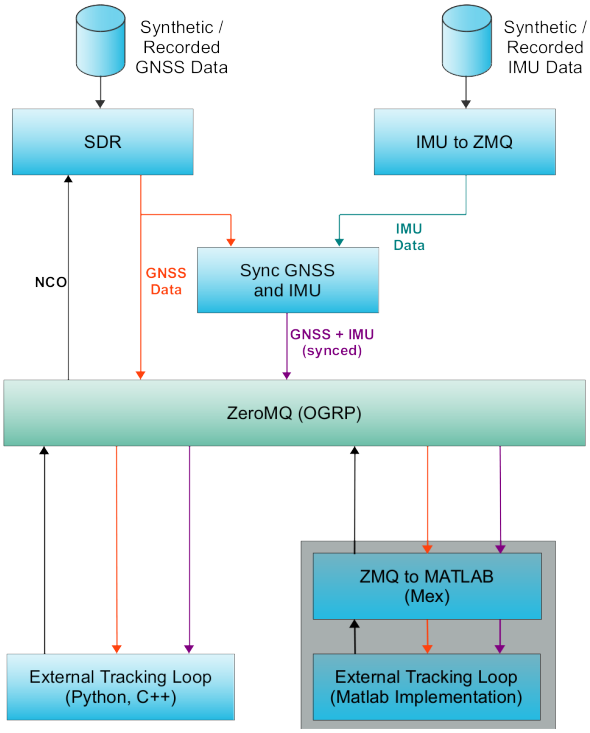

Figure 8. Development framework for loop closure

log file (e.g. generated by Spirent), converts each entry into an OGRP message and sends it via ZeroMQ. Another component (Sync GNSS and IMU) takes the GNSS data stream (I&D) as well as the IMU data stream (acceleration and angular rate) to synchronize to each other. Synchronization is done via a sample counter and sampling rate on GNSS side and via a time-stamps on IMU side. Synchronization has to be performed at the beginning of both the recorded GNSS data and the IMU data. At external loop implementation it is then possible to choose between the GNSS only and the synchronized (GNSS and IMU) stream. The NCO values will be send back in both cases via ZeroMQ.

The SDR-receiver environment helps developing vector tracking or deep coupling since the SDR waits until a NCO message is received to send the next I&D message. Also the synchronization component will wait for an I&D message to synchronize it with IMU data. Result generation, breakpoints, plotting of data or modifying of variables is possible at any time in execution of the external tracking loop. After development in MATLAB® the code can be ported to C++ or Python™ and easily be verified as the same ZeroMQ interface can be used. Afterwards it can be tested in an real-time environment.

EXAMPLE VECTOR TRACKING IMPLEMENTATION

A common way to improve the robustness and/or the precision of the GNSS position estimation is to use a coupled receiver or vector tracking. Typical issues with pure GNSS positioning are false signals or signal outages (e.g. multipath propagation or shadowing).
Deep coupling augments the GNSS tracking loops with the directional vector from an inertial navigation system (INS) to keep the tracking loops locked also in harsh environment. But also other information may be incorporated, e.g. vehicle specific data like wheel rotation speed or the steering angle. The coupling algorithm calculates its own corrections and controls the loops to stay locked with the GNSS signals, even if the reception of this signals is disturbed.
Mass-market as well as specialized GNSS receivers (e.g. in aerospace or defense systems) are black boxes and insofar do not allow to intervene in their tracking loops control. Therefore there is no possibility to get a custom deeply coupled or a vector tracking system working with such commercial receivers.
Presently, the required IMUs for deeply coupling are still somewhat expensive, since they need to be at least within the tactical grade class. However since these kinds of sensors are also increasingly available as MEMS, costs are dropping as well as the housings are getting slimmer.
The open interface with an open receiver enables such sensors and own customized algorithms to be used and opens therefore a complete new market. In the following, one example implementation of vector tracking (see figure 9) is described.

As depicted in the architecture concept above, the conventional scalar nature of code tracking loops is replaced by Vector Delay Lock Loop (VDLL). The navigation solution obtained from all channels is feedback to each of them. Thus weak ones benefit from strong ones [8]. Our navigation processor is based on a Kalman filter having eight states, which include errors of user's position ($\delta\hat{\mathbf{p}}^a_{e,k}$), velocity ($\delta\hat{\mathbf{v}}^a_{e,k}$), clock bias ($\delta\hat{c}^a_k$) and drift ($\delta\hat{\dot{c}}^a_k$) at time epoch k. Earth Centered Earth Fixed (ECEF) coordinate system has been used as shown by subscript e. Initially these errors

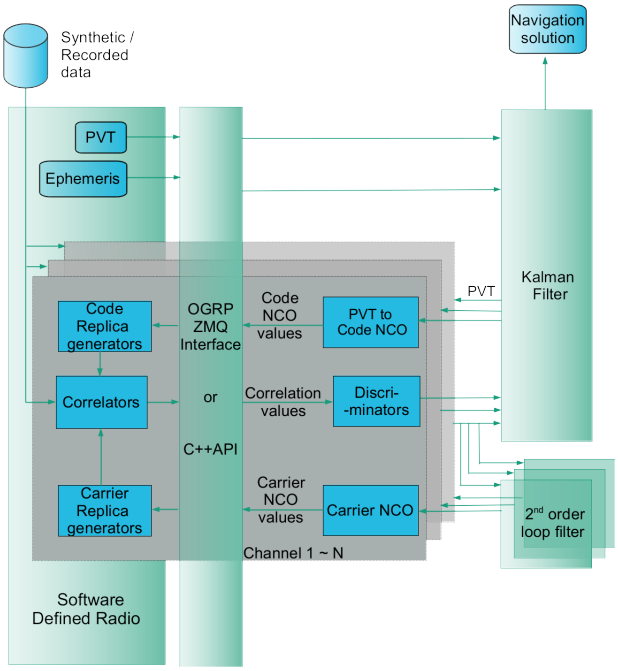

Figure 9. Architecture concept of vector tracking

are taken as zero. The estimated error states of the filter are used to correct the nominal states which correspond to the navigation solution. After every subsequent correction the error states are reset to zero. The nominal state vector and error state vector are given as

$$\hat{\mathbf{x}}_k = \begin{bmatrix} \hat{\mathbf{p}}^a_{e,k} \\ \hat{\mathbf{v}}^a_{e,k} \\ \hat{c}^a_k \\ \hat{\dot{c}}^a_k \end{bmatrix}_{8\times1} \quad \delta\hat{\mathbf{x}}_k = \begin{bmatrix} \delta\hat{\mathbf{p}}^a_{e,k} \\ \delta\hat{\mathbf{v}}^a_{e,k} \\ \delta\hat{c}^a_k \\ \delta\hat{\dot{c}}^a_k \end{bmatrix}_{8\times1} \tag{1}$$

The receiver starts with standard tracking running with the internal software until a stable state is reached. At this point it switches to the external tracking loops. The loops are initially closed using scalar tracking allowing us to get a position, velocity and time component (PVT) solution which is used to initialize the nominal states of the Kalman filter. In this approach a non coherent architecture is used, which means that the navigation filter uses the discriminator outputs of the correlation values as an input to update the navigation solution. When the Kalman filter has reached a stable state, the navigation solution of the filter is used to close the code NCO loops, by computing the new code Doppler.

In this approach the inherent relationship between the code phase discriminator output and the user position and clock bias is used. It can be represented in mathematical form as:

$$\delta\hat{z}^{\varphi}_{j,k} = (\hat{\mathbf{u}}^{as}_{e,j,k})^{\mathrm{T}}.\delta\hat{\mathbf{p}}^a_{e,k} + \delta\hat{c}^a_k + w^{\varphi}_{j,k} \tag{2}$$

where, $\delta\hat{z}^{\varphi}_{j,k}$ is the estimated code phase error in meters, $\hat{\mathbf{u}}^{as}_{e,j,k}$ shows the unit line of sight vector from the the user

to the j-th satellite and $w^{\varphi}_{j,k}$ represents the white Gaussian noise errors. So the system measurement model can be represented as

$$\begin{bmatrix} \delta\hat{z}^{\varphi}_{1,k} \\ . \\ . \\ . \\ \delta\hat{z}^{\varphi}_{j,k} \end{bmatrix}_{2j\times 1} = \begin{bmatrix} (\hat{\mathbf{u}}^{as}_{e,1,k})^{\mathrm{T}} & \mathbf{0}_{1\times 3} & 1 & 0 \\ \cdots & \cdots & . & . \\ \cdots & \cdots & . & . \\ \cdots & \cdots & . & . \\ (\hat{\mathbf{u}}^{as}_{e,j,k})^{\mathrm{T}} & \mathbf{0}_{1\times 3} & 1 & 0 \end{bmatrix} \begin{bmatrix} \delta\hat{\mathbf{p}}^{a}_{e,k} \\ \delta\hat{\mathbf{v}}^{a}_{e,k} \\ \delta\hat{c}^{a}_{k} \\ \delta\hat{\dot{c}}^{a}_{k} \end{bmatrix} \tag{3}$$

For the code phase error, we employ a normalized Delay Locked Loop (DLL) discriminator stated as

$$\begin{aligned} \delta\tilde{z}^{\varphi}_{j} &= \frac{1}{2}.\frac{E-L}{E+L} \\ &= \frac{1}{2}.\frac{\sqrt{I_E^2+Q_E^2}-\sqrt{I_L^2+Q_L^2}}{\sqrt{I_E^2+Q_E^2}+\sqrt{I_L^2+Q_L^2}}, \end{aligned} \tag{4}$$

where $\delta\tilde{z}^{\varphi}$ is the measured code phase error; I_E, I_P and I_L represent sum of the in-phase Early, Prompt and Late correlator outputs respectively between time epoch $k-1$ and k; Q_E, Q_P Q_L represent the sum of quadrature Early, Prompt and Late correlation results between $k-1$ and k epoch.

The code NCO command for the jth signal is

$$\hat{f}^{\varphi}_{NCO,k+1,j} = f^{\varphi}\left[1 - \frac{\hat{\rho}^{a}_{R,k+1,j-} - \hat{\rho}^{a}_{R,k,j-}}{c\tau_N}\right], \tag{5}$$

where $\hat{\rho}^{a}_{R,k+1,j-}$ and $\hat{\rho}^{a}_{R,k,j-}$ represents the pseudo-ranges for epoch k+1 and k, τ_N is the NCO command update interval [9].

RESULTS

The provided open GNSS receiver interface and the developed algorithms were tested with synthetic data created from the Spirent simulator [10]. The Spirent® GNSS signal generator connected to the Flexiband [11], creates a binary file which can be processed by the SDR. As a first step of the development chain, the proposed VDLL algorithm is implemented in MATLAB® and subsequently external loop closure via ZeroMQ is performed. In order to keep the complexity of a first test as low as possible, a stationary scenario without atmospheric errors was used. Furthermore, the receiver clock drift was removed by using the same clock for Spirent and Flexiband. In this way, the user's position, velocity and clock drift are known at any time, which simplifies the evaluation of the interface and the developed algorithm.

As an example, figure 10 shows the computed code Doppler frequency which was sent back to the SDR for

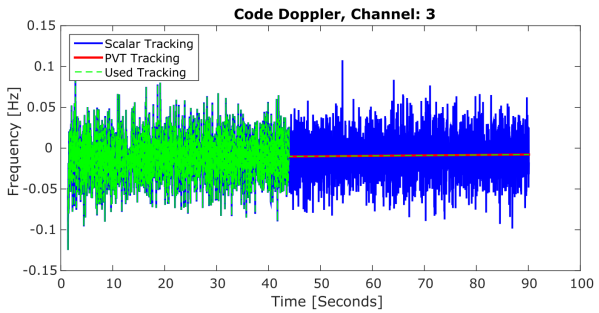

Figure 10. Computed Doppler frequency of code tracking loop

closing the code tracking loop. The duration of the scenario is 90 seconds in total. During the first 45 seconds, the code and carrier tracking loops are closed by scalar tracking until a PVT solution and the ephemeris data for all satellites in view are available. Then the Kalman filter based navigation processor is initialized and the external tracking switches from scalar to PVT tracking (VDLL). This means that the PVT solution is used together with the satellite position to compute the code Doppler frequency. The carrier tracking loops are still closed by scalar tracking. As long as VDLL is used, tracking loops are in lock, giving a good indication of the implemented approach. In addition, the code Doppler frequency computed by VDLL is approximately equivalent to the mean of the frequency that is computed by a scalar tracking.

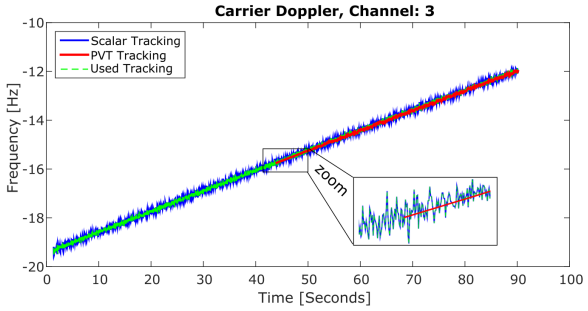

Figure 11. Computed Doppler frequency of carrier tracking loop

As mentioned earlier, the carrier tracking loops are still closed by scalar tracking. At the moment, work is in progress to extend the existing algorithm for the computation of carrier Doppler frequency. Figure 11 shows the results of the computed carrier Doppler frequency and its comparison with the mean of its counter scalar tracking, showing a good approximation as it was the case for VDLL. Nevertheless, further investigations are necessary to extend the VDLL to a long-term stable Vector Delay and Frequency Lock Loop (VDFLL).

CONCLUSION

With this approach an innovative GNSS receiver with an open access to the receiver core - the tracking loops - is

presented. This provides the GNSS community a possibility to integrate all kind of relative positioning sensors like INS, wheel rotation speed sensors, laser scanners, cameras and so on with user customized deeply coupling and vector tracking solutions.
A new market is developed where deeply coupling and vector tracking enables a high performance and more robust position solution also in harsh environment. Furthermore it opens a new research area for Small and Medium-Sized Enterprise (SME)s, universities and research institutes and makes the application market more independent from the receiver manufacturers.

The functionality of the provided open GNSS receiver interface and the MATLAB® VDLL implementation was tested successfully with synthetic data. As a next step we want to extend the VDLL to a VDFLL and use measurements from IMU to develop a deeply coupled GNSS/IMU algorithm.

EXECUTIVE SUMMARY

We present a versatile hardware assisted software receiver with an open interface enabling user customized deeply coupling and vector tracking solution.

ACKNOWLEDGMENT

The research is conducted under the GalileoOnline project (GO!) which is funded by the "Bundesministerium für Wirtschaft und Energie" (German Federal Ministry for Economic Affairs and Energy).

REFERENCES

[1] S. Kennedy and J. Rossi, "Performance of a Deeply Coupled Commercial Grade GPS/INS System from KVH and NovAtel Inc.," *InsideGNSS; E-library*, 2008.

[2] Y. Ng and G. X. Gao, "Advanced Multi-receiver Position-information-aided Vector Tracking for Robust Gps Time Transfer to Pmus," in *28th international technical meeting of the Satellite Division of The Institute of Navigation, ION GNSS+ 2015. Proceedings*, 2015.

[3] Y. Ng and G. X. Gao, "Advanced multi-receiver vector tracking for positioning a land vehicle," in *28th international technical meeting of the Satellite Division of The Institute of Navigation, ION GNSS+ 2015. Proceedings*, 2015.

[4] M. Adeel, C. Xin, Y. Wenxian, Y. Rendong, and L. Peilin, "Performance Analysis of Deeply Coupled INS Assisted Multi-Carrier Vector Phase Lock Loop for High Dynamics," in *28th international technical meeting of the Satellite Division of The Institute of Navigation, ION GNSS+ 2015. Proceedings*, 2015.

[5] M. Overbeck, F. Garzia, A. Popugaev, O. Kurz, F. Förster, W. Felber, A. Ayaz, S. Ko, and B. Eissfeller, "GOOSE - GNSS receiver with an open software interface," in *28th international technical meeting of the Satellite Division of The Institute of Navigation, ION GNSS+ 2015. Proceedings*, pp. 3662–3670, 2015.

[6] F. Garzia, C. Strobel, M. Overbeck, N. Kumari, S. Joshi, F. Foerster, and W. Felber, "A multi-frequency multi-constellation GNSS development platform with an open interface," in *Proceedings of the 2016 European Navigation Conference (ENC 2016)*, (Helsinki, Finland), pp. 1–7, Mai 2016.

[7] "OGRP - the Open Gnss Receiver Protocol," 2016. `https://github.com/Fraunhofer-IIS/ogrp`.

[8] H.-S. Kim, S.-C. Bu, G.-I. Jee, and C.-G. Park, "An Ultra-Tightly Coupled GPS/INS Integration Using Federated Kalman Filter," in *Proceedings of the 16th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2003), Portland, OR, USA, September 2003;*, pp. 2878–2885, 2003.

[9] P. Groves, *Principles of GNSS, Inertial, and multisensor integrated navigation systems*. Artec House, 2013.

[10] Spirent Communications plc, "GSS8000 brochure."

[11] A. Ruegamer, F. Foerster, M. Stahl, and G. Rohmer, "A flexible and portable multiband gnss front-end system," in *25th International Technical Meeting of the Satellite Division of the Institute of Navigation, ION GNSS 2012. Vol.3*, pp. 2378–2389, 2012.