The Witch Navigator - A Low Cost GNSS Software Receiver for Advanced Processing Techniques

Ondřej JAKUBOV, Pavel KOVÁŘ, Petr KAČMAŘÍK, František VEJRAŽKA

Dept. of Radio Engineering, Czech Technical University in Prague, Faculty of Electrical Engineering, Technická 2, 166 27 Prague 6, Czech republic

jakubon2@fel.cvut.cz, kovar@fel.cvut.cz

Abstract. *The developement of advanced GNSS signal processing algorithms such as multi-constellation, multi-frequency and multi-antenna navigation requires an easily reprogrammable software defined radio solution. Various receiver architectures for this purpose have been introduced. RF front-end with FPGA universal correlators on ExpressCard connected directly to PC was selected and manufactured. Such a unique hardware combination provides the GNSS researchers and engineers with a great convenience of writing the signal processing algorithms including tracking, acquisition and positioning in the Linux application programming interface and enables them to reconfigure the RF front-end easily by the PC program. With more of these ExpressCards connected to the PC, the number of the RF channels, correlators or antennas can be increased to further boost the computational power. This paper reveals the implementation aspects of the receiver, named the Witch Navigator, and gives the key test results.*

Keywords

GNSS, Universal Correlator, GPS, Galileo, GLONASS, Compass, SDR.

1. Introduction

Over the last decade the *global navigation satellite system* (GNSS) market has grown rapidly and has massively penetrated into various application fields such as military, civil aviation, car industry, geodesy, search and rescue, safety etc. The big demand on precise *position, velocity and time* (PVT) estimation motivates governments and companies all over the world to develope new systems (e.g. Galileo, Compass) or modernize the existing ones (GPS, GLONASS).

Receivers estimating distances to more satellites naturally yield improved PVT estimation, hence signal processing of GNSS signals from various systems, named as *multi-constellation* processing, is highly desired. This interoperability requirements formed an international agreement on *L1 and L5 radio frequency bands and CDMA multiplex* unification [4], which lowers the RF front-end costs and makes possible the employment of universal processing techniques.

The processing of GNSS signals can be accomplished by *traditional receivers* oriented on processing of particular signals - not (or not simply) reconfigurable, *pure software defined radio receivers* where the signal samples are directly processed by a general purpose microprocessor, and *offline processing techniques*.

The Witch Navigator (WitchNav) is a project with the aim of developing a low cost, high performance GNSS software receiver capable to process most of the present and future GNSS signals. The main design requirements are:

- low cost and high performance processing of most GNSS signals (*multi-constellation*),
- two frequencies in the first version (*multi-frequency*),
- separate antenna inputs (*multi-antenna*),
- free of charge development tools,
- available source code,
- operation examples and tutorials.

Our effort is devoted mainly to those studying or developing new GNSS processing algorithms which should be *fully programmed in C language* under Linux operating system (OS). Unlike current receiver architectures, the Witch Navigator receiver would provide a common flexible way of changing RF front-end parameters or acquisition, tracking and PVT algorithms. Undoubtedly, the Witch Navigator receiver becomes a sophisticated software defined radio example. Next, the device is intended to be employed in science and special applications, therefore it should be *portable* and of a *reasonable size*.

Despite the progress in satellite navigation, there are still problems such as *multipath propagation* or *indoor navigation* to be resolved. Many papers have been published on these topics, however a reasonably conceptual solution has not been found yet. Thus, a possibility of testing some of these methods is a challenge for the Witch Navigator project, as well.

Several GNSS software receiver architectures with similiar goals to ours have been introduced [5], [6], [7] (see

Subsec. 2.2), but their potential is either limited or their price is higher compared to the Witch Navigator receiver architecture.

In the following text we overview the available architectures and describe an approach to the proposed architecture with the final implementation. Our interoperability and flexibility requirements led us to the design of so called *universal correlator*, introduced in [1] and briefly reviewed here. Then, attention is turned to the implementation of a wideband data transfer between an FPGA with the universal correlators and the selected signal processing platform – PC, which became one of the major issues of our investigation. A short discussion on the proposed PC program implementation is delivered as well as references to the available test results of the receiver modules.

2. Overview of Available GNSS Receiver Concepts

2.1 Traditional Concept

An *example structure* of a traditional receiver concept is depicted in Fig. 1. The RF signal captured by the antenna is amplified in a low noise amplifier (LNA). The antenna with the LNA preselect the RF signal that is next mixed into the intermediate frequency (IF) band or down converted to the baseband. The signal is then filtered (FILTER) and amplified (G) in order to remove all the unwanted spectral components and to fit into the analog-to-digital converter's (ADC) dynamic range. The local oscillator signal (LO) can be fed to both the down converter and the sampler. As soon as enough data samples are available, the acquisition unit can begin the estimation of coarse satellite pseudoranges and Doppler frequency shifts in order to continue with precise tracking for each satellite channel. The precise pseudorange, Doppler shift, carrier phase, data symbol and SNR estimates are fed to the PVT processor.

The DSP channels for tracking, computing the correlation between the received signal and its generated replicas, can be implemented either as an ASIC or in an FPGA. If the acquisition is based on the serial search algorithm, the correlations are usually computed in the DSP channels for tracking. For parallel search algorithms, an extra acquisition unit must be included. The navigation processor can be quite simple, e.g. a single-chip microprocessor.

The advantage of the traditional concept is that the receiver can be implemented for a *high bandwidth GNSS signal* (51.15 MHz = reference bandwidth of the most complicated GNSS signal - Galileo E5 [14]). On the other hand, a rather complicated board design is necessary compared to the architectures described further in this section, which do not require any external FPGA or ASIC hardware for tracking and acquisition, but the PC. In case of the ASIC digital signal processors for tracking and acquisition, the solution misses a flexibility of any hardware change implementation. The RF front-end is usually with the fixed frequency of the conversion, the filter is not tunable and the correlators are implemented for a group of definite codes which all limit the receiver to *particular GNSS signals*.

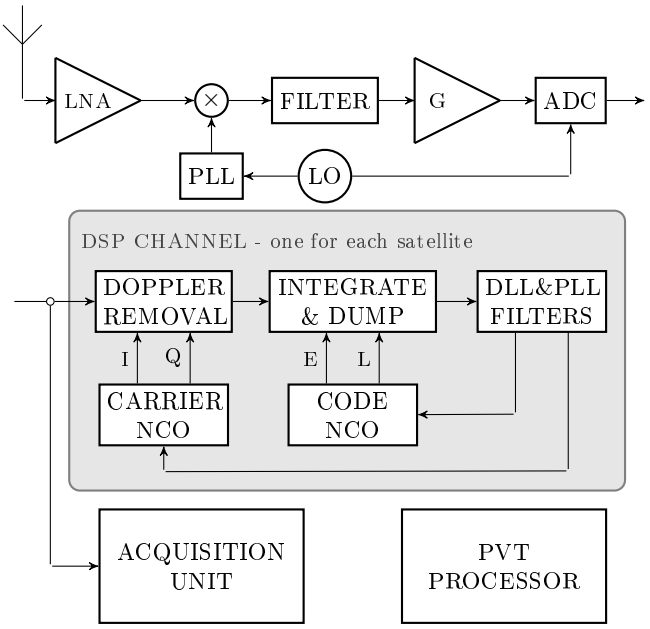

Fig. 1. Traditional GNSS receiver example structure.

2.2 Pure Software Defined Radio Concept

Another solution is to utilize the high clock frequency of today's personal computers and let one perform all signal processing computations following the ADC. Such receivers have been introduced in [5], [6], [7].

The *NordNav R30* receiver [5] can process up to 4-bit L1 signal samples with 16.3676 MHz sampling rate, transported to a PC via USB 2.0 bus. The real-time processing, here on Windows – a non-real time OS, is accomplished by storing a large amount of data (470 MB/min.) into the hard disk and processing them all at once (max. 24 satellite channels).

The *ipexSR* receiver [6] is capable to process both L1, L2 8-bit signal samples with 20 MHz sampling rate, which is enough for almost all of the GNSS signals. The data are transfered to the PC via PCI bus. The tracking and acquisition software uses the instruction set of 3.2 GHz Intel P4 CPU and can process up to 12 satellite channels in real-time mode.

A *real-time GPS civilian L1/L2 software receiver* introduced in [7] processes 2-bit data stream on a 3.4 GHz Intel Pentium 4 PC with a real-time Linux OS. PCI-DIO-32HS card connected to the PC sends the data snapshots to the PC using direct memory access (DMA). The system can process up to 12 satellite channels.

These solutions appear to be quite robust; however, the processing consumes a plenty of PC's resources. The advantage is its low price of the module and a possibility of the algorithm reprogramming up to the bearable computational load of the PC's CPU(s).

2.3 Offline Processing

The offline processing is similar to that from the previous section, but the signal samples are processed after the reception completion. A buffering method at the PC's side must be implemented for this purpose. Most of the pure software defined radio receivers implement offline processing.

The Matlab environment may simplify the algorithms' implementation. A universal program for processing of some GNSS signals for various input conditions has been developed [11]. However, for wideband signals and complex algorithms, the computations become tedious.

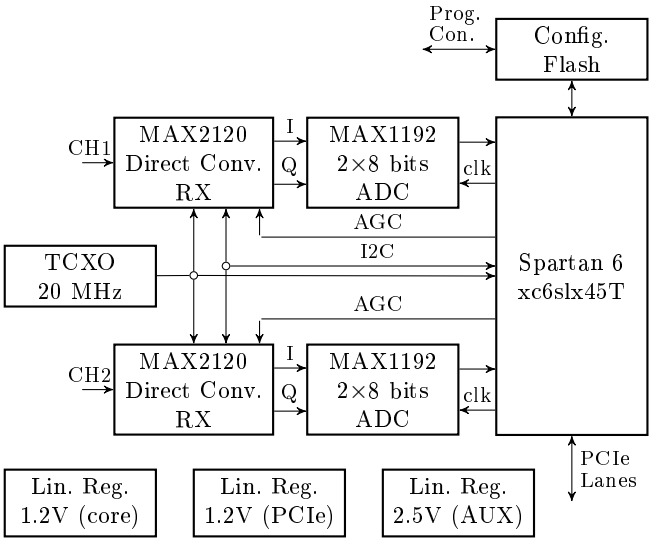

Fig. 2. The Witch Navigator receiver (v1.0) - block diagram.

3. Witch Navigator Concept

In this section, we pay attention to the concept of our GNSS receiver structure. The structure should follow the properties itemized in the introduction. We start our discussion from the RF front-end and copying the signal path we proceed up to the navigation processor. The implementation of the key blocks is provided as well.

3.1 Universal RF Front End

All the available GNSS signals are transmitted within the frequency range of $1 \div 2$ GHz, not exceeding the bandwidth of few tens of Megahertz (≤ 51.15 MHz). The power of the incoming signals at the receiver antenna typically ranges from -155 dBW to -165 dBW [15], [14].

To ensure the reception of any of these signals *zero IF DVB-S tuners* MAX2120 can be employed. Each tuner features *I2C programmable frequency synthesizer* ($925 \div 2175$ MHz), *baseband filter* ($4 \div 40$ MHz) and *two variable gain amplifiers* with dynamic range up to 75 dB for automatic gain control (AGC) realization. We implemented a pair of these tuners on a single board. This concept was introduced in [2], [8]. In the Witch Navigator receiver, 20 MHz highly stable quartz oscillator (TCXO) is used, which frequency equals the sampling frequency of the ADC. An active antenna must be connected to the RF channel input(s) (CH1(2)). The gain of the active antenna is expected not to be lower than 20 dB and the noise figure should be minimized (typically <2 dB). The block diagram is depicted in Fig. 2.

3.2 Universal Correlators in FPGA

As discussed in Subsec. 2.2, the so far available signal or universal processors have been able to perform multiplication and accumulation (MAC) of the incoming signal samples with exponentially rotated code replicas in a sequential program in real time for several satellite channels and for various GNSS signals, but with *consuming a lot of resources of expansive PCs*. Hence, a fast parallel ASIC or FPGA hardware preprocessing is needed. To meet the flexibility requirement and minimize the design costs we resort to the *FPGA implementation*.

Various GNSS signals have different modulations, however, they share common properties such as direct-sequence spread spectrum CDMA, code length and timing conventions. Thus, *universal correlator* - a generalized hardware structure accomplishing replica generation, complex exponential rotation and MAC of the signal with early and late replicas, was designed. If we implement as many of these correlators as possible into the FPGA and enable the user to interconnect a group of correlators with any radio channel, the multi-constellation tracking problem can then be *adaptively* resolved. We recall that the tuner frequency can be reprogrammed *online* in order to *select other frequency band* if desired.

3.3 Acquisition and Tracking on Desktop PC or Notebook

The acquisition of wideband GNSS signals introduces a high computational burden, since we search over long code periods and several kilohertz frequency shifts due to the Doppler effect. The serial algorithm is a brute force solution which takes a long time for the cold start of the receiver. On the other hand, if coarse estimates of the receiver position and satellite ephemeris are available, the warm start can be employed. The range of the Doppler shift is much smaller and the elapsed time by the acquisition turns to be much shorter.

To lower the time to first fix for the cold start, we have no choice but to employ parallel algorithms. Fast so-

lutions for the DSP acquisition with an FFT processor inside are available. The ASIC implementations are not flexible enough for the accommodation of new GNSS signals. The FPGA implementations consume quite a lot of resources. Employing an extra DSP or embedded PC requires an extra board design and expansive development tools. Hence, we resort to the acquisition on a desktop PC or a notebook, which would have enough computational power and provide a flexible way of reprogramming.

Before the Witch Navigator project started off, our group had implemented few GNSS receivers where the signal samples were sent via a gigabit Ethernet bus to a PC where the parallel acquisition took place. Coarse code phase and frequency shift estimates were transferred to the FPGA correlators as initial parameters to commence tracking. The data transfer between the PC and the FPGA and mainly the acquisition computation suffered from a variable latency. The feed-back loops were closed in the FPGA in an IP core processor (Microblaze, Power PC).

Due to the embedded processor in the FPGA, the maximum number of correlators is lower than with the processor outside the FPGA. The processor should be translated without the float-point unit to further save up with resources. Because of these two facts we started thinking of relocating the tracking processor out of the FPGA. Ethernet capacity does enable high-rate data transfers to the PC; tracking discriminator and filter output computations become easy for such a powerful processor. However, neither Ethernet interface nor ordinary operating systems guarantee the reaction of data transfer completion being sufficiently short ($\ll$1 ms = most of the GNSS system integrate&dump step). This implies that the closure of the feed backs over PC might not be prompt enough to ensure the stability of the loops.

3.4 ExpressCard Concept and Low Latency Linux OS

The standardization of *PCI Express Bus* (PCIe) - two wire packet-oriented bus featuring transfer rates of 4 Gbit/s has been released. Its comprehensive description is given in [13]. PCIe bus allows large data DMA transfers and unlike Ethernet does not suffer from high protocol layer latencies. PCIe cards (*ExpressCards*) can be added to the bus via connectors with no extra switch. Therefore, we decided to locate the RF front-end (except the LNA and antenna) with the FPGA to a single ExpressCard. To further increase the number of correlators, benefit from another antenna or receive signals in other frequency bands, one can connect another identical ExpressCard to the PC. The addition of cards can continue up to the limits of either the bus or the PC.

A research had been conducted on the topic of proper operating system. Although there exist quite a few open source real-time operating systems with exactly defined low latencies (RTLinux, RTAI, Neutrino, ...), potential Witch Navigator's clients - researchers, engineers and students developing new algorithms would not appreciate learning specific programming conventions which definitely come with any real-time system. An example of a PC real-time OS implementation is given in [12], [7]. In [7] the authors use RTAI where the critical part is implemented in the real-time application programming interface (API) and the kernel runs as the lowest priority thread.

A smart solution has been found by a group of *Linux kernel* developers led by Ingo Molnar. They developed a *real-time patch* which makes the kernel "more" preemptive [10]. Simply said, a real-time priority process can interrupt system calls being in the kernel mode at time instances sufficiently close to each other. Next, the real-time priority process cannot be further interrupted by any lower priority process and the scheduler always allocates resources for this process until it stops to sleep.

The prompt communication between the FPGA and the PC turns out to be feasible. The correlators can now be controlled by a *user space PC program* written in C language with the advantage of *float-point unit, standard compilers, debuggers etc.*

3.5 Other Communication Interfaces

Most FPGA vendors do not provide their customers with complete *PCIe Interface IP* (PCIeIIP). Hence, an extra IP had been written (in Verilog language). Similarily from the other side, *PCIe Linux driver* had to be developed, as well. To make the RF tuners online reconfigurable from the PC program, an *I2C controller* was implemented in the FPGA. The I2C data words created by the PC program are wrapped into the PCIe packets and forwarded by the I2C controller to the I2C bus. The FPGA processor is depicted in Fig. 3. The universal correlators are denoted as UCorIP. PLL 125 MHz is a standard Xilinx block for clock synthesis. It generates 125 MHz clock signal for PCIe IP from 100 MHz ExpressCard clock.

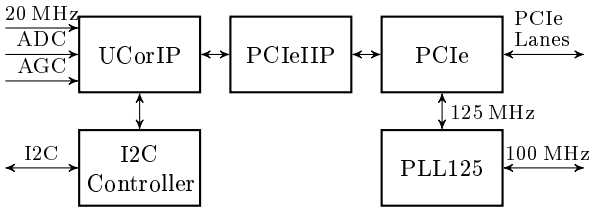

Fig. 3. Block diagram of FPGA processor.

4. Universal Correlator

The universal correlator was introduced in [1] with its main intention *not to prefer any GNSS system.* It has been successfully tested on E1b, E1c and E5 signals on an experimental receiver using a signal generator [3].

4.1 Details of the Implementation

The universal correlator is a realization of the Early/Late (E/L) correlator, such as a DSP channel in Fig. 1 without the filters, for processing of the BPSK signals. It's been equipped with a look up table for the code generation of maximum 10 230 chips. The chip length can be optionally shortened by the software.

The correlator is programmed in VHDL language and is highly optimized to minimize the hardware resources. The high performance and small hardware complexity was reached by the utilization of simple blocks, pipelining technique and serial-parallel implementation. The correlator is designed to process 8-bit baseband samples. The MAC operations are executed by DSP48 slices without signal resolution limitation. Tab. 1 documents FPGA resources required by the implementation of a single universal correlator. The employed FPGA (Fig. 2) can accommodate 12 of these correlators (including early/late, real/imaginary branches). FPGAs with larger capacity are planned to be implemented in future versions of the Witch Navigator receiver.

Resources	Quantity	Note
Slice Registers	144	General registers
Slice LUTs (Look Up Tab.)	50	General logic
RAM/FIFO [18 kb block]	1.65	Look up table for ranging code
DSP48E	1.65	Correlation calculation, phase accumulation in NCOs

Tab. 1. Xilinx Spartan 6 FPGA resources used for one universal GNSS correlator.

4.2 Non-BPSK GNSS Modulation Processing

The *binary offset carrier* (BOC) modulated signals can be processed as BPSK modulated signals with the chip rate corresponding to the doubled subcarrier frequency. Especially for high subcarrier rate BOC modulations, a suboptimal algorithm based on processing each spectral side lobe separately as a BSPK modulation (or QPSK for quadrature components) and then applying a special joint control of such correlates significantly lowers the sampling frequency and consequently simplifies the hardware complexity. The number of correlators is also reduced while the tracking performance approximately attains the optimal correlator [9] in example of Galileo E5 AltBOC(15,10). Its hardware resource allocation requirements in the FPGA and the tracking performance is already available in [1].

The universal correlator can process most of the known civil GNSS signals listed in Table 2. Reception of some signals like GPS L1 C/A, L5, GLONASS L1, L2 is fully supported; however, reception of the other signals like GPS L2 C/A, Galileo E1b and E1c is supported partially. The correlator is for example capable to process the GPS L2 C/A data signal. The pilot signal cannot be processed due to the very long ranging code. The signals with an MBOC or TMBOC modulation are processed as a BOC(1,1) signal only, the BOC(6,1) component is neglected.

System	Signal	Support.	No. of Correl.	Note
GPS	L1 C/A	Yes	1	
	L2 C/A	Partially	1	Data channel only
	L5	Yes	2	
	L1C	Partially	2	BOC(1,1) component only
GLONASS	L1	Yes	1	
	L2	Yes	1	
	L1 K	Not known	1÷4	GLONASS K sat. sig. not specified
	L5 K	Not known	2÷4	GLONASS K sat. sig. not specified
Galileo	E1b	Yes	1	BPSK method, BOC(6,1) sig. neglected, verified
	E1c	Yes	1	BPSK method BOC(6,1) sig. neglected, verified
	E5	Yes	4	Two QPSK subcarriers
	E5A	Yes	2	Verified
	E5B	Yes	2	Verified

Tab. 2. Universal GNSS correlator supported signals.

5. Data Transfer between FPGA and PC

The data transfer relations are depicted in Fig. 4. The data snapshots, correlator outputs, NCO actual states are transmitted at every TIC event distanced by 800 μs from one another. The total amount of one DMA data transfer is 32 536 B (= 40.67 MB/s). As soon as the DMA transfer finishes, MSI interrupt is handled in the Linux kernel. It is then the responsibility of the PCIe Linux device driver to handle the data and interface them with the user space process. The user space process must be set as real time and is woken up by the driver.

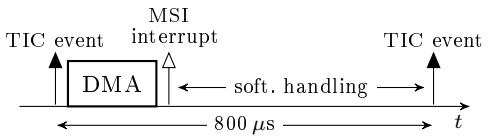

Fig. 4. FPGA and PC data transfer time relations.

This can be accomplished by a blocking `read()` function. If required by the acquisition (ACQ) process, snapshot data, which are delivered in all DMA transfers, can be stored into a shared memory between the processes, controlled by a semaphore mechanism. It is now time to perform a tracking loop step and save the results to the shared memory for the PVT process. When a new satellite becomes acquired, its coarse initial parameters can now be subtracted from the shared memory. The updates for the correlators, optionally tuner reconfiguration or PRN code data words, are send via PCIe bus using `write()` function. A standard memory write PCIe mechanism is employed here for simplicity, as we do

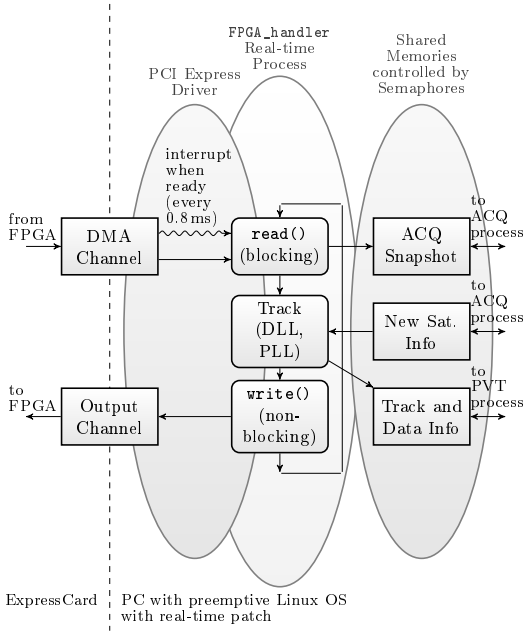

Fig. 5. PC real time process handling FPGA communication.

CPU(s)	2×Intel Core (TM) 2 T6320 @ 1.86 GHz
Memory	2 GiB
Kernel	Linux 2.6.33.7-rt29
Distribution	Fedora 12 (Constantine), GNOME 2.28.2
CPU load	4.7 %
Mem. load	<0.1 %

Tab. 3. PC test conditions for FPGA and PC data transfer related time histograms. The CPU and memory loads are for the application and the driver altogether.

not transport as much data as in the previous case. The situation is depicted in Fig. 5.

The times between two closest MSI interrupts have been measured and their histogram for 20 000 transfers is shown in Fig. 6. In the same experiment, we measured the elapsed times from the MSI interrupt beginning up to the write end, the MSI interrupt beginning to the read end and write beginning to the write end, which histograms are depicted in Fig. 6. The DMA bus transfer duration, the time between TIC event and MSI interrupt shown in Fig. 4, was measured to be approx. 100 μs.

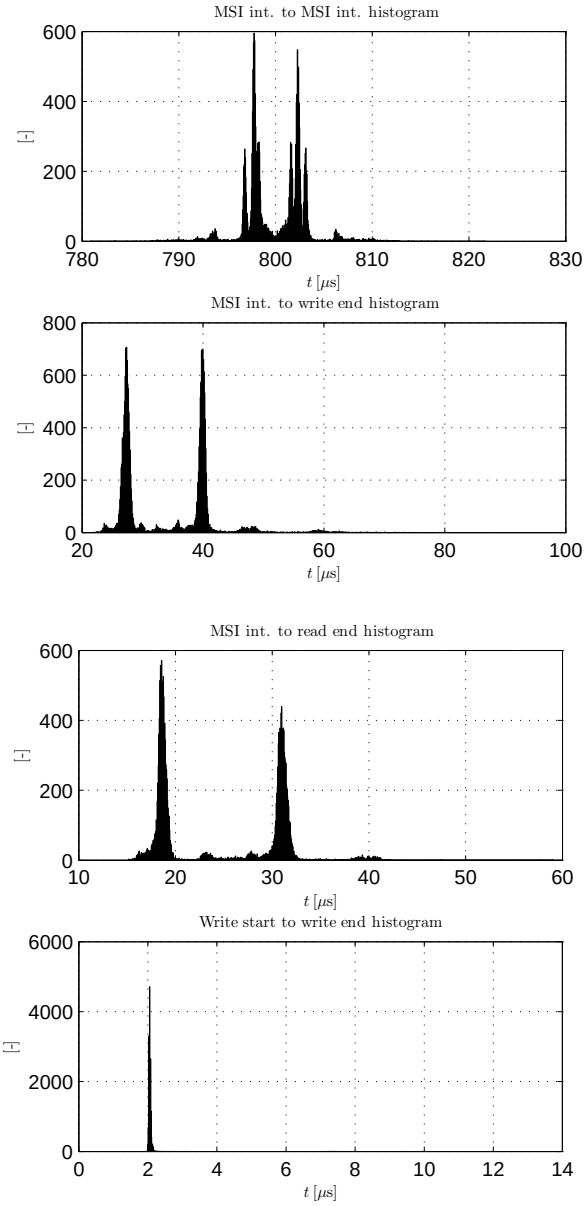

Fig. 6. FPGA and PC data transfer related time histograms with PC test conditions in Tab. 3.

6. Discussion

The available architectures, discussed in Subsec. 2.2, offer a flexible method of processing various GNSS signals. The high computational load caused by correlations evaluated in the PC has been removed by employing the universal correlators in FPGA, prompt PCIe bus and a PC with Linux OS with real-time preemptive patch. The Witch Navigator receiver architecture does not require a specific microprocessor, unless the user decides to process signal snapshots for complete tracking on the PC. It is now clear that our architecture can implement both the traditional concept and the pure software defined radio concept. The PCIe bus further does not limit the data throughput, unlike USB 2.0.

Due to the employment of the universal RF front-end cascade, FPGA programmed I2C controller and PC utility program, we remark that 1÷2 GHz *carrier frequency*, 4÷40 MHz *bandwidth* and 0÷15 dB *baseband gain* of the received signal can be *online reconfigured* by the PC program for each RF channel. Next, the universal correlators can process optimally *most of the GNSS signals* and some of them sub-optimally (Table 2). The RF channels and universal correlators can be interconnected dynamically by the PC program. The latency of the overall data transfer between the PC and FPGA was minimized down to maximum of $\approx 200\,\mu$s. The remaining 600 μs is a sufficient time for

the computation of DLL and PLL updates. *The acquisition, tracking and PVT algorithms* can be fully written in C language on PC using standard Linux API.

Summarizing our discussions, we list the properties of the Witch Navigator receiver in order to compare them with those wanted in the introduction. The Witch Navigator receiver

- is a *low cost* and *high performance* product if a low cost and powerful FPGA is used,
- is a *multi-constellation* and *multi-frequency* receiver with any number of systems limited by the number of correlators and two frequencies on one ExpressCard,
- is a *multi-antenna* receiver; two separate antennas can be connected to one ExpressCard,
- is *easy to reprogram*, and therefore flexible to develope new algorithms,
- is *portable* and of a *small size* when the ExpressCard(s) is/are connected to a notebook (Fig. 7).
- By adding more ExpressCards, the number of correlators and receiver's frequencies can be increased to further boost the performance.

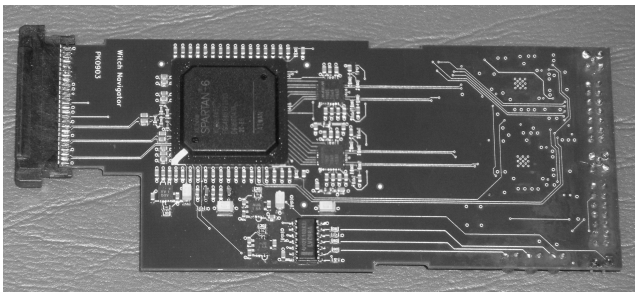

Fig. 7. Witch Navigator ExpressCard.

7. Conclusion

A prototype of the Witch Navigator receiver has been manufactured, universal correlators and RF front-end were successfully tested on Galileo E5, E1b and E1c signals. The communication between the FPGA and PC has been completely resolved and all the corresponding controllers and drivers were developed and tested. Although the interrupt latency on the PC has been measured to be sufficiently low, long term tests depending on CPU and memory loads are still to be done. Examples of tracking, acquisition, and PVT algorithms are being written to serve as tutorials for users new to the Witch Navigator. After that, we would like to research our own or verify the existing algorithms that have been tested only by a simulation.

So far one ExpressCard connected to a desktop PC has been tested (all the hardware modules and the PCIe driver). The functionality of the concept with more than one ExpressCards is to be verified.

Acknowledgements

This publication was supported by the grant MSM 6840770014 "Research of the Prospective Information Technologies" of the Ministry of Education of the Czech Republic.

References

[1] KOVÁŘ, P., KAČMAŘÍK, P., VEJRAŽKA, F. Low complex interoperable GNSS signal processor and its performance. In *Proceedings of IEEE/ION PLANS 2010 Position Location and Navigation Symposium* [CD-ROM]. Palm Springs (CA, USA), 2010, p. 947 – 951.

[2] KOVÁŘ, P., KAČMAŘÍK, P., VEJRAŽKA, F. Universal front end for software GNSS receiver. In *Proceedings of 13th IAIN World Congress* [CD-ROM]. Bergen (Norway), 2009, p. 1 – 6.

[3] KOVÁŘ, P., VEJRAŽKA, F., SEIDL, L., KAČMAŘÍK, P. Experimental software receiver of signals of satellite navigation systems. In *11th IAIN World Congress on Smart Navigation, Systems and Services.* Berlin (Germany), 2003, p.391 – 394.

[4] HEIN, G. GNSS interoperability: Achieving a global system of systems or "Does everything have to be the same?". [Online]. *Inside GNSS*, 2006, vol. Jan/Feb, p. 57 – 60. Available at: http://www.insidegnss.com/auto/0106_Working_Papers_IGM.pdf .

[5] NORMARK, P. L., STAHLBERG, C. Hybrid GPS/Galileo real time software receiver. In *Proceedings of ION GNSS 2005.* Long Beach (CA, USA), 2005, p. 1906 – 1913.

[6] PANY, T., FÖRSTER, F., EISSFELLER, B. Real-time processing and multipath mitigation of high-bandwidth L1/L2 GPS signals with a PC-based software receiver. In *Proceedings of ION GNSS 2004.* Long Beach (CA, USA), 2004, p. 971 – 985.

[7] LEDVINA, B., PSIAKI, M., SHEINFELD, D., CERRUTI, A., POWELL, S., KINTER, P. Real-time software receiver tracking of GPS L2 civilian signals using a hardware simulator. In *Proceedings of ION GNSS 2005.* Long Beach (CA, USA), 2005, p. 1598 – 1610.

[8] ŠPAČEK, J., PURIČER, P. Front-end module for GNSS software receiver. In *Proceedings of ELMAR 2006.* Zadar (Croatia), 2006, p. 211 – 214.

[9] KOVÁŘ, P., KAČMAŘÍK, P., VEJRAŽKA, F. High performance Galileo E5 correlator design. In *Proceedings of 13th IAIN World Congress.* Bergen (Norway), 2009, p. 1 – 8.

[10] Config preempt RT Patch (web page). [Online]. Cited 2010-10-09. Available at: https://rt.wiki.kernel.org/index.php/CONFIG_PREEMPT_RT_Patch

[11] BORRE, K., AKOS, D. M., BERTELSEN, N., RINDER, P., JENSEN, S. H. *A Software-Defined GPS and Galileo Receiver.* 1st ed. Boston: Birkhauser, 2007.

[12] KELLEY, C., CHENG, J., BARNES, J. Open source software for learning about GPS. In *Proceedings of 15th Int. Tech. Meeting of the Satellite Division of the U.S. Inst. of Navigation* [Online]. Portland (USA), 2002, p. 2524 – 2533. Available at: www.gmat.unsw.edu.au/snap/publications/opensourcegps.pdf

[13] MINDSHARE, INC., BUDRUK, R., ANDERSON, D., SHANLEY, T. *PCI Express System Architecture.* 4th ed. Boston (USA): Addison-Wesley, 2004.

[14] Galileo OS SIS ICD/D.0. *Galileo Open Service - Signal in Space Interface Control Document (OS SIS ICD)* [Online]. 1st issue. Cited 2010-10-02. Available at: http://ec.europa.eu/enterprise/policies/space/files/galileo/galileo_os_sis_icd_revised_2_en.pdf .

[15] ICD-GPS-200D. *Navstar GPS Space Segment/Navigation User Interfaces* [Online]. Revision D. Cited 2010-02-11. Available at: http://www.gps.gov/technical/icwg/IS-GPS-200D.pdf .

About Authors ...

Ondřej JAKUBOV was born in Liberec, Czech Republic, 1986. He received his M.Sc. (Ing.) from the CTU in Prague in 2010 and he is a postgraguate student at the same university at the Department of Radioelectronics. His research interests include GNSS signal processing algorithms and receiver architectures.

Pavel KOVÁŘ was born in Uherské Hradiště in 1970. He received his M.Sc. (Ing.) and Ph.D. (Dr.) degrees from the CTU in Prague in 1994 and 1998, respectively. Since 1997 to 2000 he worked in MESIT Instruments as a designer of avionics instruments. Since 2000 he has been with the Faculty of Electrical Engineering of the CTU in Prague as an assistant professor and since 2007 as an associate proffesor (Doc.). His interests are satellite navigation, digital communication and signal processing.

Petr KAČMAŘÍK was born in 1978. He received his M.Sc. (Ing.) degree from the CTU in Prague in 2002. During his postgraduate studies (2002 - 2006) he was engaged in the investigation of satellite navigation receiver techniques in hard conditions and wrote his Ph.D. thesis with its main topic on GNSS signal tracking using DLL/PLL with applicability to week signal environment. He defended the thesis in 2009. Since 2006 he works as an assistant professor at the Faculty of Electrical Engineering of the CTU in Prague. His main professional interests are satellite navigation, digital signal processing, implementations and simulations of signal processing algorithms.

František VEJRAŽKA was born in 1942. He received his M.Sc. (Ing.) degree from the CTU in Prague in 1965. He served as an assistant professor (1970) and associate professor (1981) at the Department of Radio Engineering. He is a full professor of radio navigation, radio communications and signals and systems theory since 1996. He was appointed the Head of the Department of Radio Engineering (1994 – 2006), vice-dean of the Faculty (2000 – 2001) and vice-rector of the CTU in Prague (2001 – 2010). His main (professional) interest is in radio satellite navigation where he participated on the design of the first Czech GPS receiver (1990) for the MESIT Instruments. He was responsible for the development of Galileo E5 receiver for Korean ETRI during 2008 and E1 and E5a receiver in 2009. Prof. Vejrazka has published 11 textbooks, more than 200 conference papers and many technical reports. He is the former president of the Czech Institute of Navigation, Fellow of the Royal Institute of Navigation in London, member of the Institute of Navigation (USA), vice-president of IAIN, vice-chairman of CGIC/IISC, member of IEEE, member of Editorial Board of GPS World, Editorial Board of InsideGNSS, etc.