Hindawi Publishing Corporation
EURASIP Journal on Embedded Systems
Volume 2009, Article ID 543720, 10 pages
doi:10.1155/2009/543720

Research Article

Multicore Software-Defined Radio Architecture for GNSS Receiver Signal Processing

Heikki Hurskainen, Jussi Raasakka, Tapani Ahonen, and Jari Nurmi

Department of Computer Systems, Tampere University of Technology, P. O. Box 553, 33101 Tampere, Finland

Correspondence should be addressed to Heikki Hurskainen, heikki.hurskainen@tut.fi

Received 27 February 2009; Revised 22 May 2009; Accepted 30 June 2009

Recommended by Markus Rupp

We describe a multicore Software-Defined Radio (SDR) architecture for Global Navigation Satellite System (GNSS) receiver implementation. A GNSS receiver picks up very low power signals from multiple satellites and then uses dedicated processing to demodulate and measure the exact timing of these signals from which the user's position, velocity, and time (PVT) can be estimated. Three GNSS SDR architectures are discussed. (1) A hardware-based SDR that is feasible for embedded devices but relatively expensive, (2) a pure SDR approach that has high level of flexibility and low bill of material, but is not yet suited for handheld applications, and (3) a novel architecture that uses a programmable array of multiple processing cores that exhibits both flexibility and potential for mobile devices. We present the CRISP project where the multicore architecture will be realized along with numerical analysis of application requirements of the platform's processing cores and network payload.

1. Introduction

Global navigation has been a challenge to mankind for centuries. However, in the modern world it has become easier with the help from Global Satellite Navigation Systems (GNSSs). NAVSTAR Global Positioning System (GPS) [1] has been the most famous implementation of GNSS and the only fully operational system available for civilian users, although this situation is changing.

Galileo [2] is emerging as a competitor and complement for GPS, as they both are satellite navigation systems based on Code Division Multiple Access (CDMA) techniques. CDMA is a technique that allows multiple transmitters to use same carrier simultaneously by multiplying pseudorandom noise (PRN) codes to the transmitted signal. The PRN code rate is higher than data symbol rate which divides the energy of a data symbol to a wider bandwidth. The used PRN codes are unique to each transmitter and thus transmitter can be identified in reception when received signal is correlated with a replica of the used PRN code.

The Russian GLONASS system, originally based on Frequency Division Multiple Access (FDMA), is adding a CDMA feature to the system with GLONASS-K satellites [3]. China has also shown interest in implementing its own system, called Compass, during the following decade [4]. The GPS modernization program [5] introduces additional signals with new codes and modulation. Realization of the new navigation systems and modernization of GPS produce updates and upgrades to system specifications.

Besides changing specifications, GNSS is also facing challenges from an environmental point of view. The resulting multipath effects make it more difficult to determine exact signal timing crucial for navigation algorithms. Research around multipath mitigation algorithms is active since accurate navigation capability in environments with heavy multipaths is desired. Among interference issues multipath mitigation is also one of the biggest drivers for the introduction of new GNSS signal modulations.

Designing a true GNSS receiver is not a trivial task. A true GNSS receiver should be reconfigurable and flexible in design so that the posibilities of new specifications and algorithms can be exploited, and the price should be low enough to enable mass market penetration.

2. GNSS Principles and Challenges

2.1. Navigation and Signal Processing. Navigation can be performed when four or more satellites are visible to the receiver. The pseudoranges from receiver to satellites and navigation data (containing ephemeris parameters) are needed [1, 6, 7].

When pseudoranges (ρ) are measured by the receiver, they can be used to solve unknowns, the users location $(x, y, z)_u$ and clock bias b_u with known positions of satellites $(x, y, z)_i$. The relation between pseudorange, satellite position, and user position is illustrated in

$$\rho_i = \sqrt{(x_i - x_u)^2 + (y_i - y_u)^2 + (z_i - z_u)^2} + b_u. \quad (1)$$

The transmitted signal contains low rate navigation data (50 Hz for GPS Standard Positioning Service (SPS)), repeating PRN code sequence (1023 chips at 1.023 MHz for GPS SPS) and a high rate carrier (GPS SPS is transmitted at L1 band which is centered at 1575.42 MHz) [1]. For Galileo E1 Open Service (OS) and future GPS L1C it also contains a Multiplexed Binary Offset Carrier (MBOC) modulation [8, 9]. These signal components are illustrated in Figure 1.

The signal processing for GNSS can be divided into analog and digital parts. Since the carrier frequencies of the GNSS are high (>1 GHz) it is impossible to perform digital signal processing on it. In the analog part of the receiver, which is called the radio front-end, the received signal is amplified, filtered, downconverted, and finally quantized and sampled to digital format.

The digital signal processing part (i.e., baseband processing) has two major tasks. First, the Doppler frequencies and code phases of the satellites need to be acquired. The details of the acquisition process are well explained in literature, for example, [1, 7]. There are a number of ways to implement acquisition, with parallel methods being faster than serial ones, but at the cost of consuming more resources. The parallel methods can be applied either as convolution in the time domain (matched filters) or as multiplication in the frequency domain (using FFT and IFFT).

Second, after successful acquisition the signals found are being tracked. In tracking, the frequency and phase of the receiver are continously fine-tuned to keep receiving the acquired signals. Also, the GNSS data is demodulated and the precise timing is formed from the signal phase measurements. A detailed description of the tracking process can be found, for example, in [1, 7]. The principles for data demodulation are also illustrated in Figure 1.

2.2. Design Challenges of GNSS. The environment we are living in is constantly changing in topographic, geometric, economic, and political ways. These changes are driving the GNSS evolution.

Besides new systems (e.g., Galileo, Compass), the existing ones (i.e., GPS, GLONASS) are being modernized. This leads to constantly evolving field of specifications which may increase the frustration and uncertainty among receiver designers and manufacturers.

The signal spectrum of future GNSS signals is growing with the new systems. Currently the GPS L1 (centered at 1575.42 MHz) is the only commercially exploited GNSS frequency band. Galileo systems E1 OS signal will be sharing the same band. Another common band of future GPS and Galileo signals will be centered at 1176.45 MHz (GPS L5 and Galileo E5a).

The GPS modernization program is also activating the L2 frequency band (centered at 1227.60 MHz) to civilian use by implementing L2C (L2 Civil) signal [10]. This band has already been assigned for navigation use, but only for authorized users via GPS Precise Positioning Service (PPS) [1].

To improve the signal code tracking and multipath performance new Binary Offset Carrier (BOC) modulation was originally introduced as baseline for Galileo and modern GPS L1 signal development [11]. The later agreement between European and US GNSS authorities further specified the usage of Multiplexed BOC (MBOC) modulation in both systems. In MBOC modulation two different binary subcarriers are added to the signal, either as time multiplexed mode (TMBOC), or summed together with predefined weighting factors as Composite BOC (CBOC) [8, 9, 12].

Like any other wireless communication, satellite navigation also suffers from multipaths in environments prone to such (e.g., urban canyons, indoors). The problem caused by multipaths is even bigger in navigation than in communication since precise timing also needs to be resolved. The field of multipath mitigation is actively researched and new algorithms and architectures are presented frequently, for example, in [13–15].

Besides GNSS there are also other wireless communication technologies that are developing rapidly and the direction of development is driven towards multipurpose low cost receivers (user handsets) with enhanced capabilities [16].

3. Overview of SDR GNSS Architectures

In this section we present three architectures for Software-Defined Radio (SDR) GNSS receiver. A simplified definition of SDR is given in [17]. "*Radio in which some or all of the physical layer functions are software defined.*"

The root SDR architecture was presented in [18]. Figure 2 illustrates an example of GNSS receiver functions mapped on to this canonical architecture. Only the reception part of the architecture is presented since current GNSS receivers are not transmitting. Radio Frequency (RF) conversion handles the signal processing before digitalization. The Intermediate Frequency (IF) processing block transfers the frequency of the received signal from IF to baseband and may also take care of Doppler removal in GNSS. The baseband processing segment handles the accurate timing and demodulation, thus enabling the construction of the navigation data bits. The division into IF and baseband sections can vary depending on the chosen solution since the complex envelope of the received signal can be handled in baseband also. Desired navigation output (Position, Velocity, and Time (PVT)) is solved in the last steps of the GNSS receiver chain.

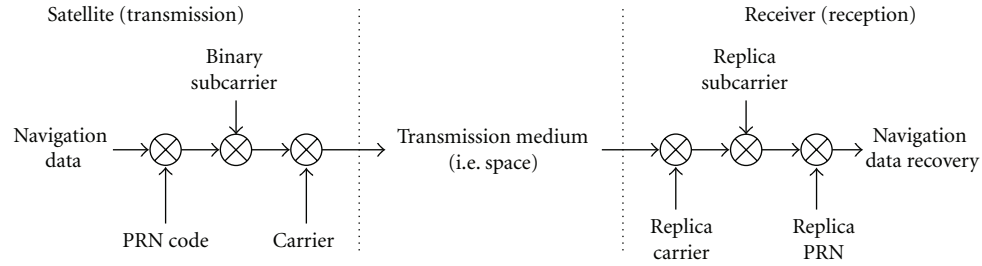


Figure 1: Principles for GNSS signal modulation in transmission and demodulation in reception.

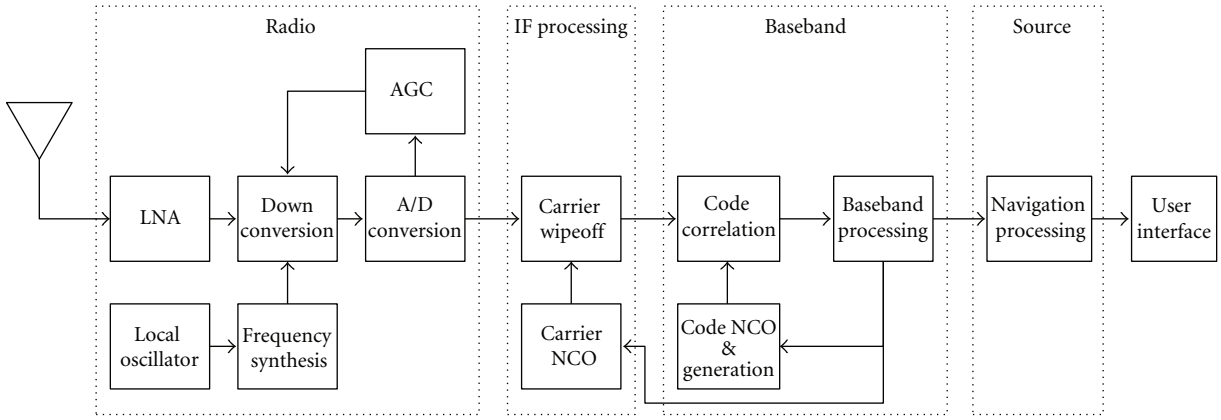


Figure 2: Canonical SDR architecture adapted to GNSS. It is modified from [18].

Current state-of-the-art mass market receivers are based on a chipset or single-chip receiver [19]. The chipset or single-chip receiver is usually implemented as an Application Specific Integrated Circuit (ASIC). ASICs have high Nonrecurring Engineering (NRE) costs, but when produced in high volumes they have very low price per unit. ASICs can also be optimized for small size and to have small power consumption. Both of these features are desired in handheld, battery operated devices. On the other hand, ASICs are fixed solutions and impossible to reconfigure. Modifications in design are also very expensive to realize with ASIC technology.

This approach has proven to be successful in mass market receivers because of price and power consumption advantages although it may not hold its position with growing demand for flexibility and shortened time to market.

3.1. Hardware Accelerated SDR Receiver Architecture. The first SDR receiver architecture discussed in this paper is the approach where the most demanding parts of the receiver are implemented on a reconfigurable hardware platform, usually in the form of a Field Programmable Gate Array (FPGA) progammed with a Hardware Description Language (HDL). This architecture, comprised of radio front-end circuit, reconfigurable baseband hardware, and navigation software is well known and presented in numerous publications, for example, [16, 20–22]. FPGAs have proved to be suitable for performing GNSS signal processing functions [23]. The building blocks for hardware accelerated SDR receivers are illustrated in Figure 3.

In this architecture the RF conversion is performed by analog radio. The last step of the conversion transforms the signal from analog to digital format. IF processing and baseband functionalities are performed in accelerating hardware. The source, PVT for the GNSS case, is constructed in navigation processing.

The big advantage for reconfigurable FPGAs in comparison to ASIC technologies is the savings in design, NRE and mask costs due to shorter development cycle. The risk is also smaller with FPGAs, since the possible bugs in design can be fixed by upgrades later on. On the other hand FPGAs are much higher in unit price and power consumption.

A true GNSS Receiver poses some implementation challenges. The specifications are designed to be compatible (i.e., systems do not interfere with each other too much) and the true interoperability is reached at receiver level. One example of interoperative design challenges is the selection of the number of correlators and their spacing for tracking, since different modulations have different requirements for the correlator structure.

3.1.1. Challenges with Radio Front End. Although the focus of this paper is mainly on baseband functions, the radio should not be forgotten. The block diagram for a GNSS single frequency radio front end is given on the left-hand side of Figure 3. In the radio the received signal is first amplified with the Low Noise Amplifier (LNA) and then after necessary filtering it is downconverted to low IF, for example, to 4 MHz [24]. The signal is converted to a digital format after downconversion.

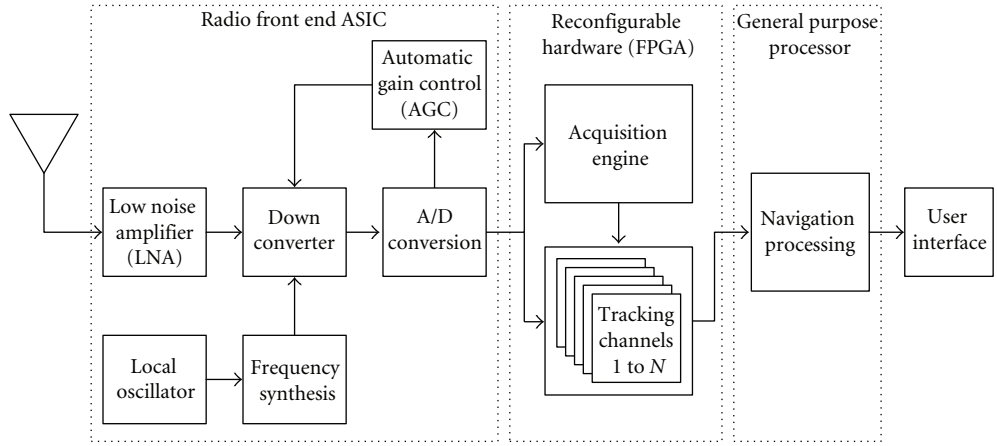


Figure 3: Hardware accelerated baseband architecture. From left to right: analog radio part, reconfigurable baseband hardware, and navigation software running on GPP.

The challenges for GNSS radio design come from the increasing amount of frequency bands. To call a receiver a true GNSS receiver and also to get the best performance, more than one frequency band should be processed by the radio front-end. Dual- and/or multifrequency receivers are likely choices for future receivers, and thus it is important to study potential architectures [25].

Another challenge comes from the increased bandwidth of new signals. With increased bandwidth the radio becomes more vulnerable to interference. For mass market consumer products, the radio design should also meet certain price and power consumption requirements. Only solutions with reasonable price and power consumption will survive.

3.1.2. Baseband Processing. The fundamental signal processing for GNSS was presented in Figure 1. The carrier and code removal processes are illustrated in more detail in Figure 4. The incoming signal is divided into in-phase and quadrature-phase components by multiplying it with the locally generated sine and cosine waves. Both phases are then correlated in identical branches with several closely delayed versions (for GPS; early, prompt, and late), of the locally generated PRN code [1]. Results are then integrated and fed to discriminator computation and feedback filter. Numerically Controlled Oscillators (NCOs) are used to steer the local replicas.

An example of the different needs for new GNSS signals is the addition of 2 correlator fingers (bump-jumping algorithm) due to Galileo BOC modulation [26]. In Figure 4 additional correlator components needed for Galileo tracking are marked with darker shade. In most parts the GPS and Galileo signals in L1 band are using the same components. The main difference is that due to the BOC family modulation Galileo needs additional correlators; it is very-early (VE) and very-late (VL) to remove the uncertainty of main peak location estimation [27]. The increasing number of correlators is related to the increase in complexity, measured by the number of transistors in the final design [13].

The level of hardware acceleration depends on the selected algorithms. Acquisition is rarely needed compared to tracking and thus it is more suitable for software implementation. FFT-based algorithms are more desirable for designer to implement in software since hardware languages are usually lacking direct support for floating-point number calculus. Tracking on the other hand is a process containing mostly multiplication and accumulation using relatively small word lengths. The thing that makes it more suitable for hardware implementation is that the number of these relatively simple computations is high, with a real-time deadline.

3.2. Ideal SDR GNSS Receiver Architecture. The ideal SDR is characterized by assigning all functions after the analog radio to a single processor [18]. In the ideal case all hardware problems are turned to software problems.

A fundamental block diagram of a software receiver is illustrated in Figure 5 [28]. The architecture of the radio front-end is the same that was illustrated in Figure 3. After radio the digitized signals are fed to buffers for software usage. Then all of the digital signal processing, acquisition, and tracking functions are performed by software.

In the literature, for example, [28, 29], the justification and reasoning for SDR GNSS is strongly attributed to the well-known Moores law which states that the capacity of integrated circuits is doubling every 18–24 months [30]. Ideal SDR solutions should become feasible if and when available processing power increases. Currently reported SDR GPS receiver implementations are working in realtime only if the clock speed of the processor is from 900 MHz [31] to 3 GHz [29], which is too high for mobile devices but not, for example, a laptop PC.

In the recent years, the availability of GNSS radio front ends with USB has improved, making the implementation of a pure software receiver on a PC platform quite straightforward. The area where pure software receivers have already made a breakthrough is postprocessing applications. Postprocessing with software receivers allows fast algorithm

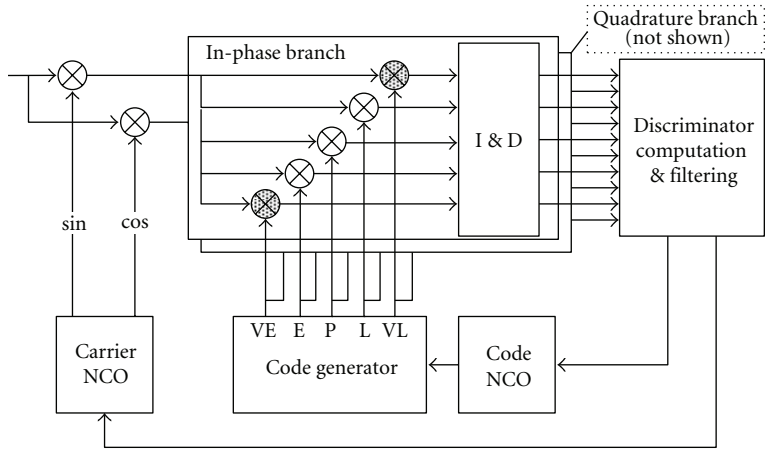


Figure 4: GPS/Galileo tracking channel.

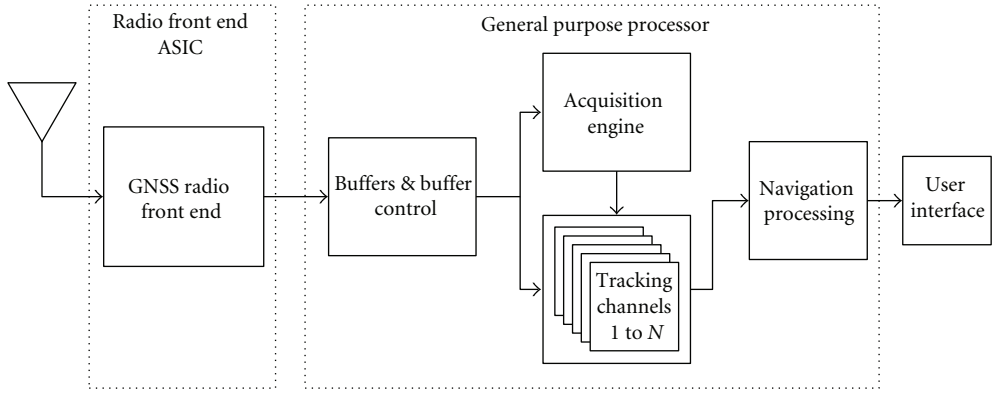


Figure 5: Software receiver architecture. On left-hand side: analog radio part, and on right-hand side: baseband and navigation implemented as software running on a GPP.

prototyping and signal analysis. Typical postprocessing applications are ionospheric monitoring, geodetic applications, and other scientific applications [21, 32].

Software is definitely more flexible than hardware when compared in terms of time to market, bill of materials, and reconfigurable implementation. But with a required clock frequency of around 1 GHz or more, the generated heat and battery life will be an issue for small handheld devices.

3.3. SDR with Multiple Cores. What about having an array of reconfigurable cores for baseband processing? In a multicore architecture baseband processing is divided among multiple processing cores. This reduces the clock frequency needed to a range achievable by embedded devices and provides an increased level of parallelism which also eases the work load per processing unit.

An example of the GNSS receiver architecture with reconfigurable baseband approach is illustrated in Figure 6. In this example one of the four cores is acting as an acquisition engine and the remaining three are performing the tracking functions. A fixed set of cores is not desirable since the need for acquisition and tracking varies over time. For example, when receiver is turned on, all cores should be performing acquisition to guarantee the fastest possible Time To First Fix (TTFF). After satellites have been found more of the acquisition cores are moved to the tracking task.

If (and when) manufactured in large volumes the (properly scaled) array of processing cores can be eventually implemented in an ASIC circuit. This lowers the per unit price and makes this solution more appealing for mass markets, while still being reconfigurable and having high degree of flexibility.

In the next section we present one future realization of this architecture.

4. CRISP Platform

Cutting edge Reconfigurable ICs for Stream Processing (CRISP) [33] is a project in the Framework Programme 7 (FP7) of the European Union (EU). The objectives of the CRISP are to research the optimal utilization, efficient programming, and dependability of a reconfigurable multiprocessor platform for streaming applications. The CRISP consortium is a good mixture of academic and industrial know-how with partners; Recore (NL), University of Twente (NL), Atmel (DE), Thales Netherlands (NL),

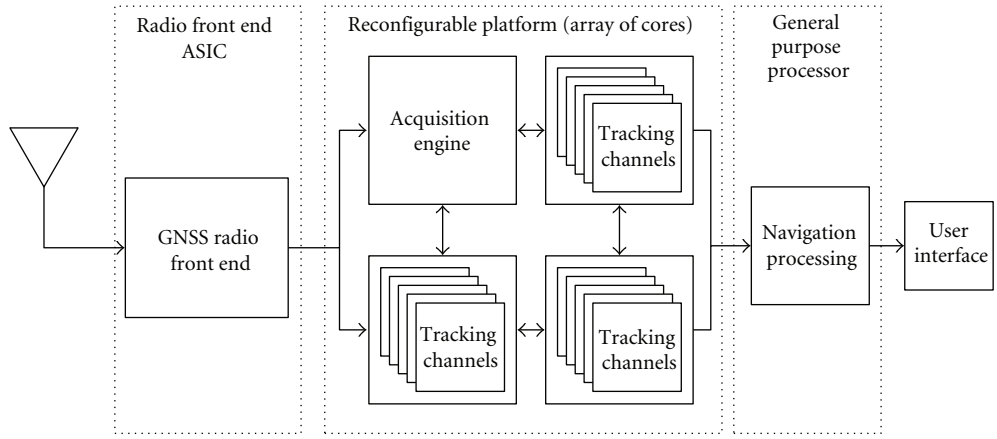


Figure 6: Software reconfigurable baseband receiver architecture. From left: analog radio part, baseband implemented on an array of reconfigurable cores, and navigation software running on GPP.

Tampere University of Technology (FI), and NXP (NL). The three-year project started in the beginning at 2008.

The reconfigurable CRISP platform, also called General Streaming Processor (GSP), designed and implemented within the project, will consist of two separate devices: General Purpose Device (GPD) and Reconfigurable Fabric Device (RFD). The GPD contains off-the-shelf General Purpose Processor (GPP) with memories and peripheral connections whereas the RFD consists of 9 reconfigurable cores. The array of reconfigurable cores is illustrated in Figure 7 [34], with "R" depicting a router.

The reconfigurable cores are Montium cores (it was recently decided to use Xentium processing tile as Reconfigurable Core in the CRISP GSP. The Xentium has at least similar performance to the Montium (with respect to cycle count), but is designed for better programmability (e.g., hardware supporting optimal software pipelining)). Montium [35] is a reconfigurable processing core. It has five Arithmetic and Logical Units (ALUs), each having two memories, resulting in total of 10 internal memories. The cores communicate via a Network-on-Chip (NoC) which includes two global memories. The device interfaces to other devices and outer world via standard interfaces.

Within the CRISP project the GNSS receiver is one of the two applications designed for proof of concept for the platform. The other is a radar beamforming application which has much higher demands on computation than a standalone GNSS receiver.

4.1. Specifying the GNSS Receiver for the Multicore Platform. In the CRISP project our intention is to specify, implement, and integrate a GNSS receiver application supporting GPS and Galileo L1 Open Service (OS) signals on the multicore platform. In this case, the restriction for L1 band usage comes from the selected radio [24], but in principle the multicore approach can be extended to multifrequency receivers if a suitable radio front-end is used.

4.1.1. Requirements for Tile Processor. The requirements of GNSS L1 application have been studied in [36]. The results, restated in Table 1, indicated that a single Montium core running at 200 MHz clock speed is barely capable of executing the minimum required amount of acquisition and tracking processes. This analysis did not take into account the processing power needed for baseband to navigation handover nor navigation processing itself. With this it is evident that an array of cores (more than one) is needed for GNSS L1 purposes. The estimations given in Table 1 are based on reported [35] performance of the Montium core. The acquisition figures are computed for a search speed of one satellite per second and the tracking figures are for a single channel.

Table 1: Estimation of GNSS baseband process complexity for Montium Tile Processor running at 200 MHz, max performance of 1 GMAC/s [36].

Process	Usage (MMAC/s)	Usage of TP (%)
Acquisition (GPS)	43.66	4.4
Acquisition (Galileo)	196.15	19.6
Tracking (GPS)	163.67	16.4
Tracking (Galileo)	229.14	22.9

The results presented in Table 1 reflect the complexity of the processes when the input stream is sampled at 16.368 MHz, which is the output frequency of the selected radio front end for CRISP platform [24]. This is approximately 16 times the navigation signal fundamental frequency of 1.023 MHz.

The GNSS application can also be used with a lower rate input stream without a significant loss in application performance. For this paper, we analyzed the effect of the input stream decimation to the complexity of the main baseband processes. The other parameters, such as acquisition time and number of frequency bins for acquisition and number of active correlators per channel for tracking, remained the same as in [36].

Figures 8 and 9 illustrate the effect of decimation by factors 1, 2, 4, 8, and 16 to the utilization of the Montium Tile processor. Decimation factor 1 equates to the case where no

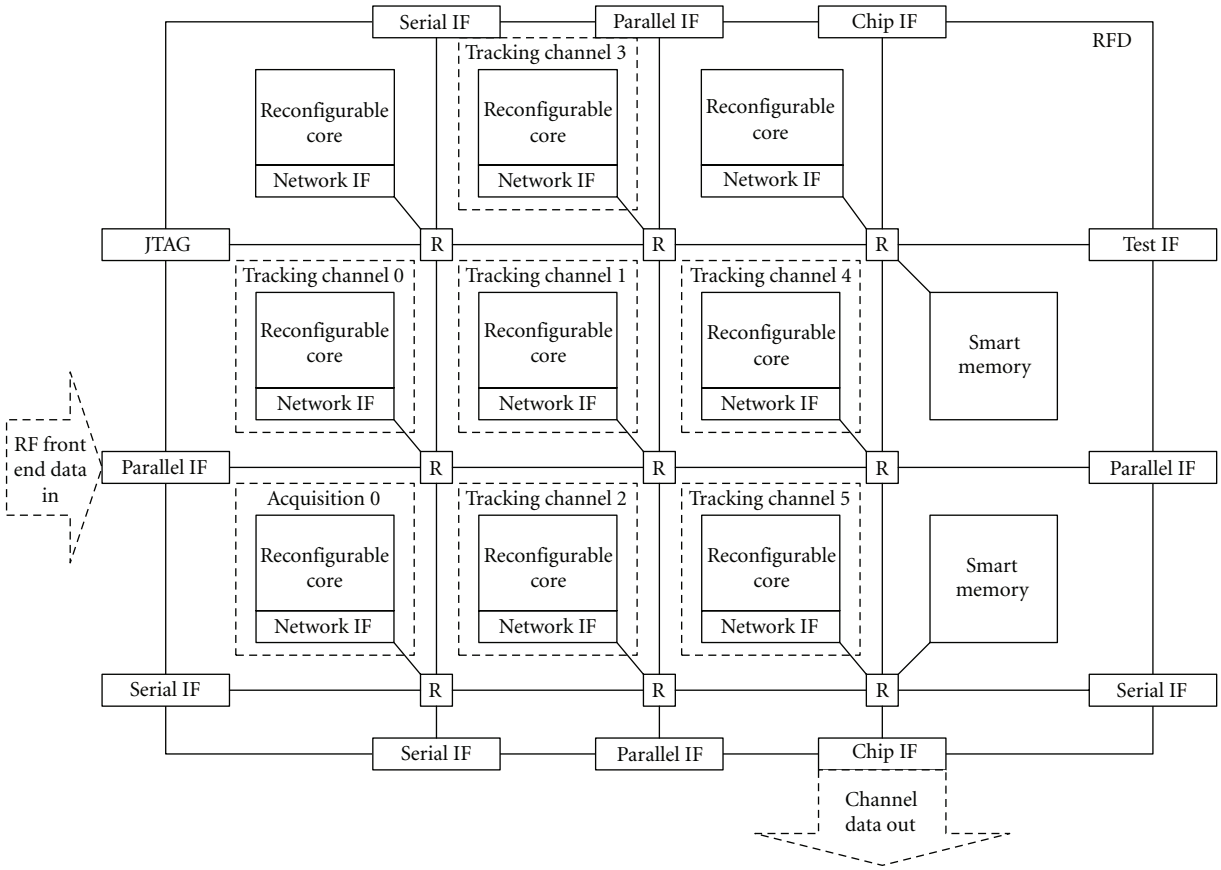


Figure 7: Array of 9 reconfigurable cores [34] with example mapping of GNSS application illustrated, the selection of cores is random. "R" depicts router and "IF" interface.

decimation is applied, that is, results shown in Table 1. The presented figures show how the complexity of both processes, measured as Montium Tile Processor utilization percentage, decreases exponentially as decimation factor increases. The behavior is the same for GPS and Galileo signals, except that utilization with Galileo signals is a bit larger than with GPS in all studied cases.

To ease the computational load of the Tile Processor the decimation of the input stream seems to be a feasible choice. The amount of decimation should be sufficient to effect meaningful savings in TP utilization without significantly degrading performance of the application. For the current GPS SPS signal, decimation by factor 4 (4.092 MHz) is feasible without significant loss in receiver performance. Factor of 8 (2.046 MHz) is equal to the Nyqvist rate for 1.023 MHz, which is the PRN code rate used in GSP SPS signal.

In the Galileo case, decimation factor 4 is the maximum decimation factor. This is because with a sampling frequency of approximately 4 MHz the BOC(1,1) component of the Galileo E1 OS signal can be still received with a maximum loss of only −0.9 dB, when compared with the reception of the whole MBOC bandwidth [12]. (This applies also to the modern GPS L1C signals, but they are not specified in our application [36].)

Table 2: Estimation of GNSS baseband process complexity with decimated (by factor 4) input stream. Montium Tile Processor running at 200 MHz, max performance of 1 GMAC/s.

Process	Usage (MMAC/s)	Usage of TP (%)
Acquisition (GPS)	9.57	0.96
Acquisition (Galileo)	43.66	4.37
Tracking (GPS)	40.92	4.09
Tracking (Galileo)	57.28	5.73

In the ideal case the decimation of the input stream should be changing with the receiver mode (GPS/Galileo). Since in CRISP the decimation of the radio stream will be implemented as hardware in FPGA, which is connecting the radio to the parallel interface of the final CRISP prototype platform, the run time configuration of the decimation factor is not feasible. For this reason, in the rest of the paper we will focus on the scenario where fixed decimation factor of 4 is used, resulting in a stream sample rate of 4.092 MHz.

Table 2 shows baseband complexity estimation for the case when input stream is decimated by a factor of four. When it is compared to the original figures of complexity shown in Table 2, it can be seen that the utilization of TP is over four times smaller.

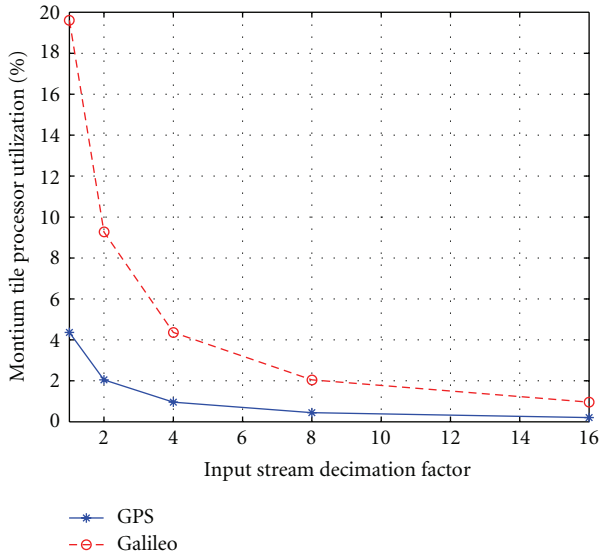


Figure 8: Acquisition process utilization of Montium Tile Processor resources as a function of the decimation factor of the input stream.

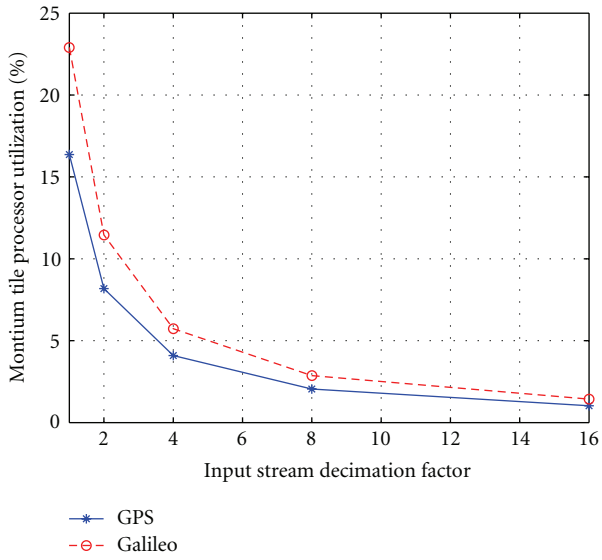
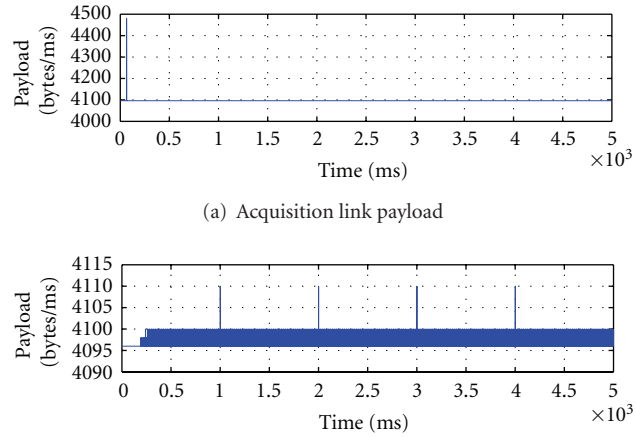


Figure 9: Tracking process utilization of Montium Tile Processor resources as a function of the decimation factor of the input stream.

4.1.2. Requirements for the Network-on-Chip. To analyze the multicore GNSS receiver application we built a functional software receiver with the C++ language, running on a PC. The detailed analysis of the software receiver will be given in substantial paper [37].

In our SW receiver each process was implemented as a software thread. With approximating one process per core this approach enabled us to estimate the link payload by logging communication between the threads.

We estimated a scenario where one core was allocated to perform acquisition and six cores were mapped for the tracking process. This scenario is illustrated in Figure 7. Digitized RF front-end data is input to the NoC via an interface.



(a) Acquisition link payload

(b) Average tracking link payload

Figure 10: Link payloads for GPS acquisition process (a) and average payload of GPS tracking processes (b).

A specific chip interface is used to connect the RFD to the GPD, and it is used to forward channel data (channel phase measurement data related to pseudorange measurements, and navigation data) to the GPD. The Selected mapping is a compromise between minimal operative setup (one acquisition and four tracking) and the needs of dependability testing processes, where individual cores may be taken offline for testing purposes.

The scenario was simulated with a prerecorded set of real GPS signals. Since signal sources for Galileo navigation were not available, the Galileo case was not tested. The link payloads caused by the cores communicating while the software was running for 5 seconds is illustrated in Figure 10.

The results show that, in GPS mode, our GNSS application causes a payload for each link/processing core with a constant baseline of 4096 Bytes/millisecond. This is caused by the radio front-end input, that is, the incoming signal. In this scenario we used real GPS front end data which was sampled at 4.092 MHz, each byte representing one sample. This sampling rate is also equal to the potential decimation scenario discussed earlier.

With a higher sampling rate the link payload baseline will be raised, but on the other hand one byte can be preprocessed to contain more than one sample, decreasing the traffic caused by radio front-end input.

The first peak in the upper part of Figure 8 is caused by the acquisition process output. When GNSS application starts, FFT-based acquisition is started and the results are ready after 60 milliseconds, which are then transmitted to tracking channels. This peak is also the largest individual payload event caused by the GNSS application.

After a short initialization period the tracking processes start to produce channel output. An Average of simulated GPS tracking link/processing core payloads is illustrated in Figure 10(b). Every 20 milliseconds a navigation data symbol (data rate is 50 Hz in GPS) is transmitted and once a second higher transmission peak is caused by the loop

phase measurement data, which is transmitted to GPD for pseudorange estimation.

In Galileo mode, the payload caused by incoming signal will be equal since the same radio input will be used for both GPS and Galileo. However, the transmission of data symbols will cause a bigger payload since data rate of Galileo E1 signals is 250 symbols per second [8]. Galileo phase measurement rate will remain the same as in GPS mode.

From the results it is seen that the link payload caused by the incoming RF signal is the largest one in both operating modes, and if the link payload needs to be optimized the reduction of it is the first thing to be studied. The results also indicate that when GNSS application is running smoothly the link payloads caused by it are predictable.

Note that this estimation does not contain any overheads caused by network protocol or any other data than navigation related (dependability, real-time mapping of the processes). These issues will be studied in our future work.

4.2. Open Issues. Besides the additional network load caused by other than the GNSS application itself, there are also some other issues that remain open. There may be challenges in designing software for a multicore environment. Power consumption as well as the final bill of materials (BOMs), (i.e., final price of the multicore product) remains an open issue at the time of this writing. In future these issues will be studied and suitable optimizations performed after the prototyping and proof of concepts have been completed successfully.

5. Conclusions

In this paper we discussed three Software-Defined Radio (SDR) architectures for a Global Navigation Satellite System (GNSS) receiver. The usage of flexible architectures in GNSS receiver was justified with the need for implementing support for upcoming navigation systems and new algorithms developed, and especially for multipath mitigation. The hardware accelerated SDR architecture is quite close to the current mass market solutions. There the ASIC is replaced with a reconfigurable piece of hardware, usually an FPGA. The second architecture, ideal (or pure) SDR receiver is using a single processor to realize all necessary signal processing functions. Real-time receivers remain a challenge, but postprocessing applications are already taking advantage of this architecture.

The third architecture, SDR with multiple cores, is a novel approach for GNSS receivers. This approach benefits in both having high degree of flexibility, and when properly designed and scaled, a reasonably low unit price in high volume production. In this paper we also presented the CRISP project where such a multicore architecture will be realized along with the analysis of GNSS application requirements for the multicore platform.

We extended the previously published analysis of processing tile utilization to cover the effect of input stream decimation. Decimation by factor four seems to offer a good compromise between core utilization and application performance.

We implemented a software GNSS receiver with processes implemented as threads and used that to analyze the GNSS application communication payload for individual links. This analysis indicated that the incoming signal represents the largest part of the communication in the network between processing cores.

Acknowledgments

The authors want to thank Stephen T. Burgess from Tampere University of Technology for his useful comments about the manuscript. This work was supported in part by the FUGAT project funded by the Finnish Funding Agency for Technology and innovation (TEKES). Parts of this research are conducted within the FP7 Cutting edge Reconfigurable ICs for Stream Processing (CRISP) project (ICT-215881) supported by the European Commission.

References

[1] E. D. Kaplan and C. J. Hegarty, Eds., *Understanding GPS, Principles and Applications*, Artech House, Boston, Mass, USA, 2nd edition, 2006.

[2] J. Benedicto, S. E. Dinwiddy, G. Gatti, R. Lucas, and M. Lugert, "GALILEO: Satellite System Design and Technology Developments," European Space Agency, November 2000.

[3] S. Revnivykh, "GLONASS Status and Progress," December 2008, http://www.oosa.unvienna.org/pdf/icg/2008/icg3/04.pdf.

[4] G. Gibbons, "International system providers meeting (ICG-3) reflects GNSS's competing interest, cooperative objectives," *Inside GNSS*, December 2008.

[5] U. S. Airforce, "GPS Modernization Fact Sheet," 2006, http://pnt.gov/public/docs/2006/modernization.pdf.

[6] M. S. Braasch and A. J. van Dierendonck, "GPS receiver architectures and measurements," *Proceedings of the IEEE*, vol. 87, no. 1, pp. 48–64, 1999.

[7] K. Borre, D. M. Akos, N. Bertelsen, P. Rinder, and S. H. Jensen, *A Software Defined GPS and Galileo Receiver—A Single-Frequency Approach*, Birkhäuser, Boston, Mass, USA, 2007.

[8] "Galileo Open Service, Signal in space interface control document (OS SIS ICD)," Draft 1, February 2008.

[9] "Interface Specification—Navstar GPS Space segment/User segment L1C Interfaces," IS-GPS-800, August 2007.

[10] R. D. Fontana, W. Cheung, and T. Stansell, "The modernized L2C signal—leaping forward into the 21st century," *GPS World*, pp. 28–34, September 2001.

[11] "Galileo Joint Undertaking—Galileo Open Service, Signal in space interface control document (OS SIS ICD)," GJU, May 2006.

[12] G. W. Hein, J.-A. Avila-Rodriguez, S. Wallner, et al., "MBOC: the new optimized spreading modulation recommended for GALILEO L1 OS and GPS L1C," in *Proceedings of the IEEE/ION Position, Location, and Navigation Symposium (PLANS '06)*, pp. 883–892, San Diego, Calif, USA, April 2006.

[13] H. Hurskainen, E. S. Lohan, X. Hu, J. Raasakka, and J. Nurmi, "Multiple gate delay tracking structures for GNSS signals and their evaluation with simulink, systemC, and VHDL," *International Journal of Navigation and Observation*, vol. 2008, Article ID 785695, 17 pages, 2008.

[14] S. Kim, S. Yoo, S. Yoon, and S. Y. Kim, "A novel unambiguous multipath mitigation scheme for BOC(kn, n) tracking in GNSS," in *Proceedings of the International Symposium on Applications and the Internet Workshops*, p. 57, 2007.

[15] F. Dovis, M. Pini, and P. Mulassano, "Multiple DLL architecture for multipath recovery in navigation receivers," in *Proceedings of the 59th IEEE Vehicular Technology Conference (VTC '04)*, vol. 5, pp. 2848–2851, May 2004.

[16] F. Dovis, A. Gramazio, and P. Mulassano, "SDR technology applied to Galileo receivers," in *Proceedings of the International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GPS '02)*, Portland, Ore, USA, September 2002.

[17] "SDR Forum," January 2009, http://www.sdrforum.org.

[18] J. Mitola, "The software radio architecture," *IEEE Communications Magazine*, 1995.

[19] P. G. Mattos, "A single-chip GPS receiver," *GPS World*, October 2005.

[20] P. J. Mumford, K. Parkinson, and A. G. Dempster, "The namuru open GNSS research receiver," in *Proceedings of the International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS '06)*, vol. 5, pp. 2847–2855, Fort Worth, Tex, USA, September 2006.

[21] S. Ganguly, A. Jovancevic, D. A. Saxena, B. Sirpatil, and S. Zigic, "Open architecture real time development system for GPS and Galileo," in *Proceedings of the International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS '04)*, pp. 2655–2666, Long Beach, Calif, USA, September 2004.

[22] H. Hurskainen, T. Paakki, Z. Liu, J. Raasakka, and J. Nurmi, "GNSS receiver reference design," in *Proceedings of the 4th Advanced Satellite Mobile Systems (ASMS '08)*, pp. 204–209, Bologna, Italy, August 2008.

[23] J. Hill, "Navigation signal processing with FPGAs," in *Proceedings of the National Technical Meeting of the Institute of Navigation*, pp. 420–427, June 2004.

[24] Atmel, "GPS Front End IC ATR0603," Datasheet, 2006.

[25] M. Detratti, E. Lopez, E. Perez, and R. Palacio, "Dual-frequency RF front end solution for hybrid Galileo/GPS mass market receiver," in *Proceedings of the IEEE Consumer Communications and Networking Conference (CCNC '08)*, pp. 603–607, Las Vegas, Nev, USA, January 2008.

[26] P. Fine and W. Wilson, "Tracking algorithms for GPS offset carrier signals," in *Proceedings of the ION National Technical Meeting (NTM '99)*, San Diego, Calif, USA, January 1999.

[27] H. Hurskainen and J. Nurmi, "SystemC model of an interoperative GPS/Galileo code correlator channel," in *Proceedings of the IEEE Workshop on Signal Processing Systems (SIPS '06)*, pp. 327–332, Banff, Canada, October 2006.

[28] D. M. Akos, "The role of Global Navigation Satellite System (GNSS) software radios in embedded systems," *GPS Solutions*, May 2003.

[29] C. Dionisio, L. Cucchi, and R. Marracci, "SOFTREC G3, software receiver and signal analysis fog GNSS bands," in *Proceedings of the 10th IEEE Internationl Symposium on Spread Spectrum Techniques and Applications (ISSSTA '08)*, Bologna, Italy, August 2008.

[30] G. E. Moore, "Cramming more components onto integrated circuits," *Proceedings of the IEEE*, vol. 86, no. 1, pp. 82–85, 1998.

[31] S. Söderholm, T. Jokitalo, K. Kaisti, H. Kuusniemi, and H. Naukkarinen, "Smart positioning with fastrax's software GPS receiver solution," in *Proceedings of the International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS '08)*, pp. 1193–1200, Savannah, Ga, USA, September 2008.

[32] J. H. Won, T. Pany, and G. W. Hein, "GNSS software defined radio: real receiver or just a tool for experts," *Inside GNSS*, pp. 48–56, July-August 2006.

[33] "CRISP Project," December 2008, http://www.crisp-project.eu.

[34] P. Heysters, "CRISP Project Presentation," June 2008, http://www.crisp-project.eu/images/publications/D6.1_CRISP_project_presentation_080622.pdf.

[35] P. M. Heysters, G. K. Rauwerda, and L. T. Smit, "A flexible, low power, high performance DSP IP core for programmable systems-on-chip," in *Proceedings of the IP/SoC*, Grenoble, France, December 2005.

[36] H. Hurskainen, J. Raasakka, and J. Nurmi, "Specification of GNSS application for multiprocessor platform," in *Proceedings of the International Symposium on System-on-Chip (SOC '08)*, pp. 128–133, Tampere, Finland, November 2008.

[37] J. Raasakka, H. Hurskainen, and J. Nurmi, "Modeling multicore software GNSS receiver with real time SW receiver," in *Proceedings of the International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GNSS '09)*, Savannah, Ga, USA, September 2009.