International Journal of Microwave and Wireless Technologies

cambridge.org/mrf

Verifying a concept of adaptive communication with LEO satellites using SDR-based simulations

S. Kozłowski[1], K. Kurek[1], J. Skarzyński[1], K. Szczygielska[1] and M. Darmetko[2]

[1]Warsaw University of Technology, Institute of Radioelectronics and Multimedia Technology, Warsaw, Poland and [2]Space Research Centre, Polish Academy of Sciences, Warsaw, Poland

Research Paper

Cite this article: Kozłowski S, Kurek K, Skarzyński J, Szczygielska K, Darmetko M (2019). Verifying a concept of adaptive communication with LEO satellites using SDR-based simulations. *International Journal of Microwave and Wireless Technologies* **11**, 602–608. https://doi.org/10.1017/S1759078719000552

Received: 8 October 2018
Revised: 8 April 2019
Accepted: 11 April 2019
First published online: 28 May 2019

Key words:
Software-defined radio; adaptive communication; low Earth orbit satellites

Author for correspondence:
S. Kozłowski, E-mail: S.Kozlowski@ire.pw.edu.pl

Abstract

The paper is related to an adaptive satellite communication system for data transmission from small, low cost, low Earth orbit satellites. Tests run in a set-up consisting of a number of software-defined radio (SDR) modules operating as a satellite, a ground station, and a satellite channel simulator, have shown that by changing modulation scheme and code rate one can obtain increase of amount of data which can be downloaded from a satellite during a single pass over a ground station approximately by a factor of 2. To determine data rates obtainable in an SDR system using a common personal computer as a digital signal processing device, execution times of particular processing steps involved in the reception process were measured.

Introduction

In an adaptive communication system, the transmission mode, i.e. modulation scheme and channel code, are changed depending on the quality of the received signal, which is evaluated in the receiver as either signal to noise ratio (SNR) or bit error rate (BER). In good conditions, when SNR is high, the transmission can be switched to a more efficient mode to increase the data rate. Under unfavorable conditions, i.e. when SNR is low and BER exceeds the predefined acceptable level, a more robust mode may be used to restore correct reception. To send the transmission mode change requests from the receiver to the transmitter, a kind of return channel is required, but in many modern communication systems, such channels are available. Actually, adaptation is already exploited in commercial systems such as DVB-S2 [1] or IEEE 802.11n [2]. The adaptive communication was proposed also for space applications [3, 4].

Switching between particular transmission modes can be easily realized in a system based on software-defined radio (SDR) concept. In an SDR module, a large part of signal processing is performed in the digital domain by an appropriate digital signal processing (DSP) device. A typical SDR-based transmitter or receiver is a combination of a hardware radio frequency front-end and a DSP part, in the simplest case a common personal computer (PC) carrying on operations involved in the transmission or reception process. In consequence, various modulation schemes and channel codes can be supported in such system if only the computer used is powerful enough to perform all required signal processing.

Approaches to the problem of adaptation included in standards [1, 2] are complicated and thus they would be hard to implement in systems in which factors such as power, size, and price are of great consideration. In papers dealing directly with low-cost solutions, like [3, 4], authors focus mainly on the problem how to design hardware capable of adaptation, with no propositions of adaptation mechanism and no analysis of its cons and pros. For these reasons, in [5] we proposed a system intended for providing an adaptive communication between a small, low cost, low Earth orbit (LEO) satellite and a ground station (GS). The laboratory model of the system was set up using the transmitter operating as a LEO satellite, the satellite channel simulator, and the receiver representing a ground station, described in details in [6]. This paper is focused on the reception process. Besides benefits from utilizing the adaptation mechanism under various conditions, which were partially discussed in the previous works, also computational complexity of the implemented SDRbased receiver was addressed. This factor is usually omitted by other authors, yet it is crucial for determining how fast transmission is possible to obtain or, contrary, what hardware is required for correct reception of the signal transmitted with the desired data rate.

The adaptation principle

Due to high orbital velocity and short orbit radius, a satellite placed in LEO is visible from GS only for a short time (several to approximately 15 min, depending on whether the satellite passes through the zenith or does not reach high elevation angles). This severely limits the total amount of data that can be downloaded from the satellite unless one has a network of

GSs located at various points of the Earth. It is also known that a LEO satellite is 3–4 times closer to GS when it is near the zenith compared with the situation when it is just above the horizon. This fact is exploited by the proposed adaptation mechanism which should result in increase of the total amount of data possible to download during a single satellite pass.

The set of available modulation schemes is assumed to include BPSK, QPSK, 8-PSK, and 16-APSK (after the DVB-S2 standard [7]), whereas the convolutional code rate can be selected from 1/2, 2/3, 3/4, 5/6, or 7/8 (after the DVB-S standard [8]). The receiver measures the current SNR level, compares it with a number of predefined thresholds, selects the best transmission mode to be used [6], and sends the transmission mode change request to the satellite via the return channel. The threshold levels were determined experimentally so as to provide BER not higher than 10^{-6} at the output of the channel decoder. Each communication session is started in the most robust mode, which is also forced after a prolonged transmission failure. Available transmission modes, together with the corresponding effective data rates (Rb) expressed in useful information bits per symbol duration, are presented in Table 1.

The transmission is performed using frames containing a constant number of symbols, the same for all modes. Each frame includes the training sequence, as well as the header and the payload protected with the CRC code. The maximum frame duration is limited by the Doppler effect caused by very high orbital velocity of a LEO satellite. The frequency shift may change rapidly, especially when the satellite is close to the zenith. Under such circumstances, a carrier recovery algorithm designed to return a single frequency value for the whole frame may introduce synchronization error. For a given symbol rate, this limits the maximum number of symbols in a single frame. On the other hand, the number of symbols in the frame cannot be too small due to the large overhead introduced by the preamble and header. The above-mentioned problems can be solved by using a synchronization algorithm tracing the phase or frequency of the received signal on a sample-by-sample manner, e.g. a software PLL [9], but at the expense of troublesome selection of the loop's bandwidth providing both wide capture range and sufficient robustness when the loop works in the locked state.

SDR model of the adaptive system

The block diagram of the proposed adaptive system model is shown in Fig. 1. The model includes the transmitter (SDR module Ettus Research USRP N210 with WBX radio interface [10] controlled by a common PC1), the satellite channel simulator (two N210 modules controlled by PC2), and the receiver (SDR module NI USRP 2920 [11] controlled by PC3). The connections between the SDR modules and corresponding PCs are provided by means of Gb Ethernet LAN. An auxiliary LAN connection between PC1 and PC3 represents the return channel from the GS to the satellite. The entire system operates at 2 GHz. The frequency was chosen to be close to the S Band allocated for the satellite – ground station communication, but less than the SDR modules limitation (2.2 GHz).

The signal processing performed in the transmitter involves relatively simple operations like frame forming and pulse shaping with the square-root raised cosine filter. The channel simulator processes the transmitted signal so as to provide SNR level and Doppler shift corresponding to the current orbit point. Its implementation was quite a challenging task, but it goes beyond the scope of this paper. The structure of the receiver is, however, crucial for the system performance and will be described in detail.

Table 1. Transmission modes

Mode no.	Modulation/code rate	Rb (bit/symbol)
1	BPSK 1/2	0.5
2	BPSK 2/3	0.67
3	QPSK 1/2	1
4	QPSK 2/3	1.33
5	8-PSK 1/2	1.5
6	16-APSK 1/2	2
7	16-APSK 2/3	2.67
8	16-APSK 3/4	3
9	16-APSK 5/6	3.33
10	16-APSK 7/8	3.5

The receiver application is implemented in C and is divided into three threads: the data acquisition thread, the baseband processing thread, and the soft-decision decoding thread, as it is shown in Fig. 2.

The data acquisition thread provides a sample stream from the SDR module. The received signal is sampled at the rate of four samples per symbol.

The task of the baseband processing thread is to provide a symbol stream for the soft-decision decoder. In the first step of processing, the coarse Doppler shift correction is carried out in the fine downconversion block. The out-of-date value of the Doppler shift from the previous frame is used. Next, the signal is filtered with the root raised cosine filter, the same as used in the transmitter. After filtering, the residual frequency offset is removed. To determine the offset, the carrier wave is recovered by subjecting the signal to a modulation-dependent nonlinear operation, and then the Fourier transform is computed to determine the position of a peak indicating the presence of the recovered carrier. To obtain frame synchronization and symbol timing, the part of the sample stream where the beginning of a frame is supposed to be located is interpolated to 32 samples per symbol in order to increase its time resolution, and then cross-correlated with the preamble. Once the symbol timing is recovered, the signal is resampled to obtain exactly one sample per symbol using polyphase filter. Channel estimation is carried out by comparing the transmitted preamble with the received one. The determined value of the channel impulse response is then used in the channel equalizer to correct amplitudes and phases of incoming symbol values.

The soft-decoding thread performs the most computationally demanding operations: soft-decision demapping, Viterbi decoding, and SNR estimation. It also decides whether and when to change the transmission mode. The decisions are based on the estimated SNR, averaged over 20 frames to mitigate problems resulting from the SNR estimation inaccuracy. Also, a 2 dB wide hysteresis is introduced to the decision algorithm to avoid frequent switching between two neighboring modes.

Tests of the adaptation mechanism

The performance of the proposed system was verified in a number of tests run in the set-up of Fig. 1. Various scenarios differing in

https://doi.org/10.1017/S1759078719000552 Published online by Cambridge University Press

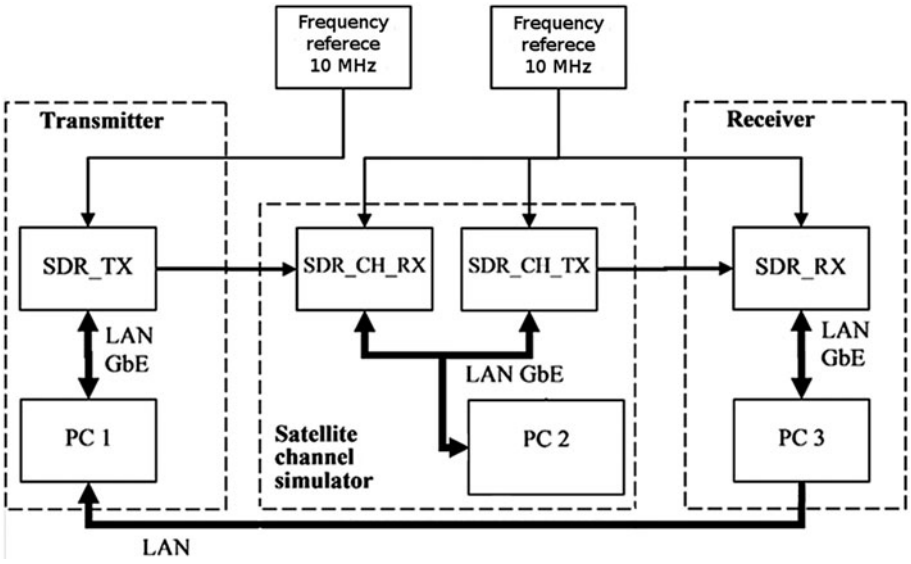

Fig. 1. Block diagram of the proposed system model.

orbit, SNR level at the beginning of the communication session, and frame duration were considered. The first two factors determine a set of attainable transmission modes. When the SNR is low or the ratio of the longest to the shortest distance between the satellite and GS is close to 1, not all modes will be used. For long frames, the overhead introduced by non-useful data, e.g. synchronization preamble and header, is low, but on the other hand changing Doppler shift is harder to eliminate in the reception process since implemented algorithm tries to correct the beginning and the end of a frame using the same frequency value, which may cause synchronization failure.

The results obtained from the tests are summarized in Table 2. Benefits of employing proposed adaptation mechanism are expressed as the amount of data downloaded during a single satellite pass over GS in the proposes system ($C_{adaptive}$) compared with the amount of data that would be downloaded in the reference system using only the most robust transmission mode ($C_{BPSK\ 1/2}$). One can notice that said ratio oscillates around 2, although significantly lower (1.46) and higher (4.9) values were also reported in certain extraordinarily bad or good transmission conditions.

Figure 3 presents an exemplary result obtained from a single test. For the selected orbit, the satellite was visible over the horizon line for about 11 min. When the distance to GS was the shortest, the transmission has switched to the 6th mode (16-APSK 1/2, 2 useful bites per symbol duration, see Table 2). Due to fast changing Doppler shift in this part of the satellite pass, two synchronization failures have occurred. The system has reverted to the basic mode for a short time needed to regain synchronization. It is also visible that between the 200th and 300th second the transmission has switched to a lower mode due to SNR measurement inaccuracy, although, as it was mentioned above, a hysteresis was introduced to the adaptation mechanism to prevent such events from being frequent.

Computational complexity

Several kinds of devices are commonly considered as suitable for signal processing in SDR-based systems: FPGA circuits, digital signal processors, graphics processing units (GPU), and general purpose processors. In the work presented in the paper, the last option was chosen, assuming that the general purpose processor is part of a PC provided with standard operating system. Employing PCs for providing computational resources for SDR modules is widespread for many reasons. PCs are easily available and relatively inexpensive. Numerous free compilers and Integrated Development Environments allow users to develop their own applications without the costs associated with purchasing dedicated hardware and specialized software tools, as is the case with FPGA and digital signal processors. Last but not least, programming, even using a low-level language, takes less time and requires less experience than preparing and debugging hardware description for an FPGA circuit.

On the other hand, a general purpose processor is relatively slow when compared with the other above-mentioned devices. This may cause problems, especially for a receiver, which is always more troublesome than a transmitter and typically involves a number of highly complex baseband operations. To identify them, execution times of particular blocks of Fig. 2 were measured. In this paper we do not address issues related to the channel decoding, which is computationally demanding, but also well described in some other works [12–14].

Processing device

The receiver application was run on a PC working under 64-bit Microsoft Windows 7 and equipped with Intel Core i7 4770 processor having a nominal clock frequency of 3.4 GHz. The selected processor supports SIMD extensions [15], frequency scaling [16], and Hyper-Threading (HT) technology. The threads were granted the highest priority: REALTIME_PRIORITY_CLASS/THREAD_PRIORITY_TIME_CRITICAL [17].

SIMD extensions enable the processor to perform certain arithmetic, logical or comparison operations on multiple variables simultaneously. To keep the portability of the receiver application, only SSE instructions were employed, since AVX instruction sets may be not available in older processors. Frequency scaling consists in automatic reduction of processor clock frequency in order to save energy when the processor usage is low. This option can be disabled by a user. In such a case, the processor operates with its nominal frequency at the expense of high power

https://doi.org/10.1017/S1759078719000552 Published online by Cambridge University Press

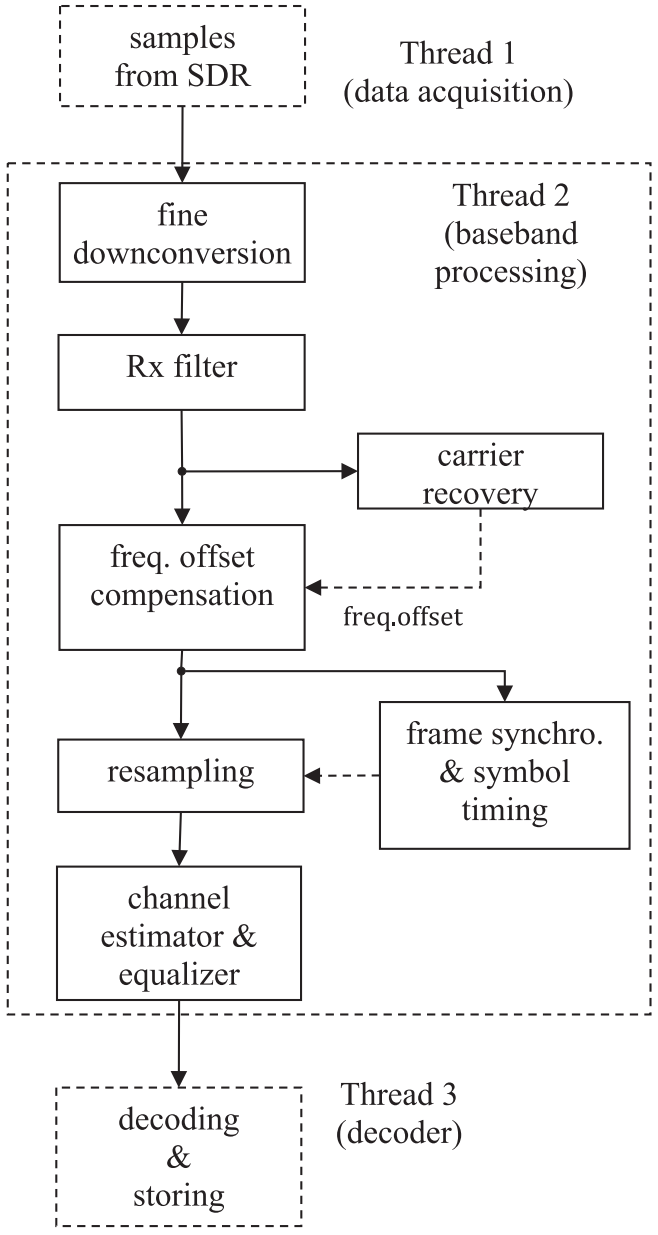

Fig. 2. Signal processing steps performed by the software receiver.

consumption. HT technology is based on duplicating processor parts responsible for storing architectural state without duplicating the execution units. Consequently, each processor core is seen by the operating system as two independent logical processors which can be scheduled to execute separate threads at the same time. When one thread is stopped due to temporary lack of data, the execution of the other one can be started immediately, which reduces idle time of execution units and increases average number of instructions performed per clock cycle. The selected processor includes four cores and supports eight parallel threads.

Measurement results

Figure 4 shows cumulative distribution functions (CDF) of the frame reception time. Each curve is plotted on a base of measurements carried out for approximately 33 000 frames (10 000 symbols per frame, four samples per symbol, 0.5 Msymbol/s). For particular CDFs the following options are enabled or disabled: HT ("HT" in the legend if enabled), frequency scaling ("F" if enabled), and storing decoded data on the hard drive ("W" if enabled). Although the latter operation is performed in separate thread, it is commonly known to be extremely time-consuming and therefore its influence on the entire receiving process was also investigated. The CDFs labeled with "none" correspond to measurements with all the above three options disabled.

Table 2. Benefits of using the adaptive system in various scenarios.

Scenario	Value of parameter	$C_{adaptive}/C_{BPSK\ 1/2}$
Various frame durations (orbit height = 300 km)	2.5 ms	2.65
	5 ms	2.6
	10 ms	2.4
	20 ms	2.2
	40 ms	1.8
Various orbit heights (frame duration = 5 ms)	300 km	2.6
	600 km	2.1
	800 km	2.0
Various starting SNR levels (orbit height = 300 km, frame duration = 5 ms)	20 dB	4.9
	15 dB	3.8
	11.5 dB	2.99
	8.5 dB	2.36
	7 dB	1.87
	5 dB	1.84
	3 dB	1.46

Figure 4 can be concluded as follows. Enabling the frequency scaling allows the processor to slow down in an uncontrolled manner. Some frames are received in the shortest possible time (1.6 ms), but there are also some frames received in 8.5 ms. Nevertheless, even in the worst case, the processor speed is high enough to perform the reception process in the time interval shorter than the frame duration (20 ms). When the frequency scaling is disabled, the reception time is substantially constant. For a few frames it is abnormally long, but never exceeds the mean value (1.8 ms) more than 1.5 times. Enabling or disabling options other than the frequency scaling practically does not change the reception time.

Table 3 presents execution time of particular steps involved in the reception process, expressed as a percent of the total frame reception time.

One can observe that the most time-consuming operation in this particular receiver application is frequency synchronization. The carrier recovery, i.e. determining the exact frequency offset value, takes 24% of the total reception time. The two-stage procedure of making the received signal strictly DC-centered (fine downconversion and frequency offset compensation) takes another 31%. Thus, the overall synchronization process takes more than 50% of the total time.

The carrier recovery comprises several sub-steps. The nonlinear operation, which in the case of QPSK modulation scheme used in the experiment is the 4th power, takes <2%. Decimation used to reduce the number of samples inputted to the Fourier

https://doi.org/10.1017/S1759078719000552 Published online by Cambridge University Press

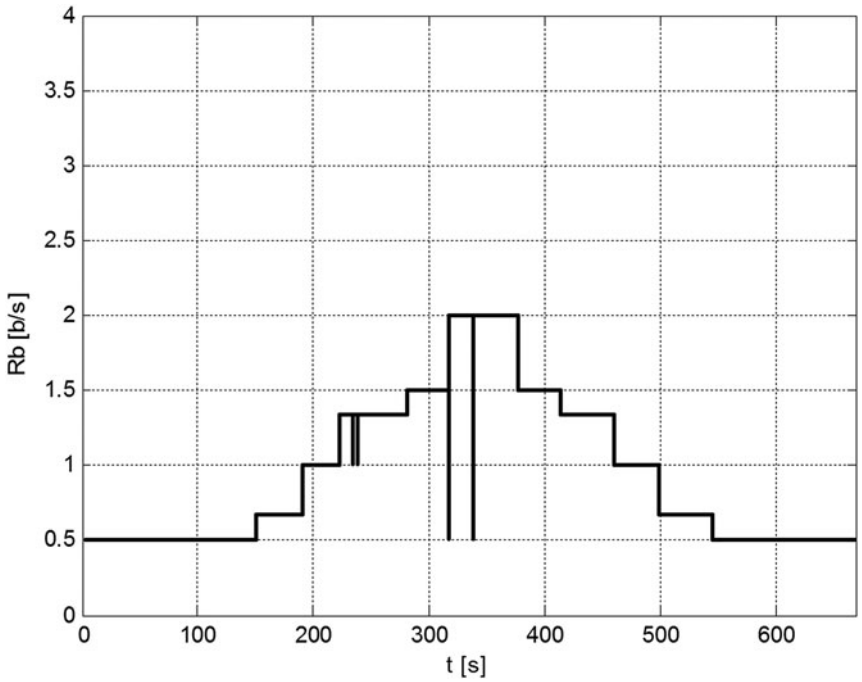

Fig. 3. An exemplary result for a single satellite pass over GS.

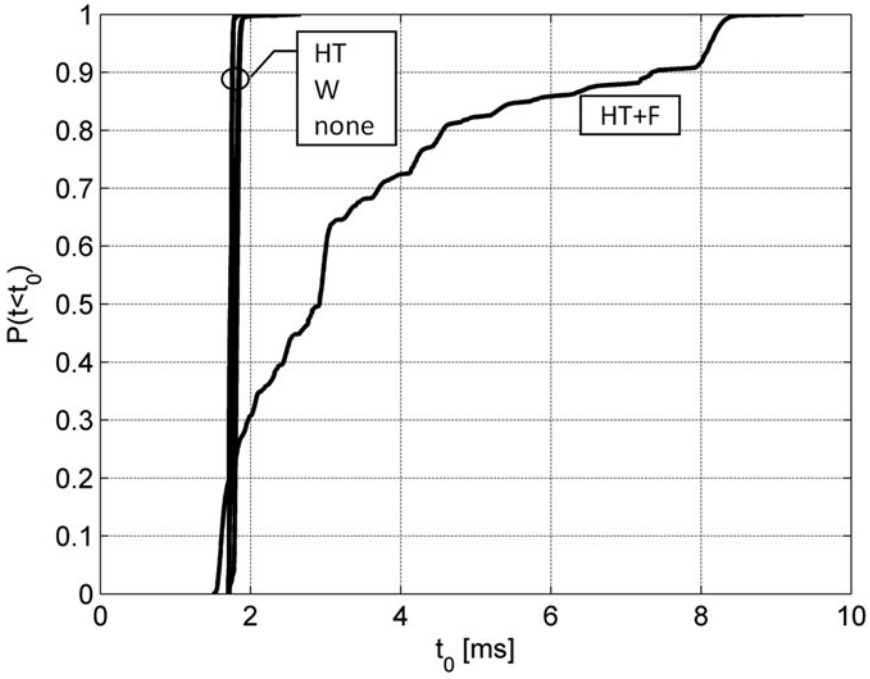

Fig. 4. CDFs of the frame reception time.

transform at the expense of narrower observed frequency range takes 17%, the Fourier transform takes 5%, and finally, power spectrum calculation and searching for a peak takes <1%. The first and the last sub-step have a little contribution to the total time. Algorithms implemented in the FFTW library employed in the receiver are highly optimized and developing faster FFT would be a very challenging task. However, as far as the decimation is concerned, the impulse response of the filter used is very long (301 elements) in order to assure narrow and steep transmittance characteristic. At this point, impulse response length/amount of aliasing trade-off can obviously be done to make the decimation much faster.

Filtering operation is the second one having a significant impact on the total reception time. It is already highly optimized using SSE instructions and therefore substantial reduction of its execution time would be required in employing some sophisticated programming techniques, such as splitting the whole filtering process into several threads or moving it to GPU. However, it is possible to shorten the impulse response of the filter with limited loss of performance as long as the roll-off factor is kept high.

https://doi.org/10.1017/S1759078719000552 Published online by Cambridge University Press

Table 3. Execution time of particular processing steps.

Step	Fine down-conv.	Rx filter	Carrier recovery	Freq. offset comp.	Frame synchro. &symbol timing	Resampling	Channel estimator & equalizer
% of total reception time	17	22	24	14	5	13	5

Conclusions

In the investigation being the subject of this paper, a concept of an adaptive system for communication with LEO satellites was verified using an SDR-based system. Proposed adaptation mechanism consists in changing the modulation scheme and convolutional code rate according to the current propagation channel conditions. The results have shown that switching the transmission mode when propagation conditions change works properly and gives about 2–2.5 times increase in system throughput compared with a non-adaptive system operating always with the transmission mode assuring stable communication in the worst case.

Execution time measurements have shown that, if only the receiver application is appropriately optimized, it is possible to achieve data rate of 0.5 Msymols/s with the oversampling factor of 4 using a general purpose computer. One can suppose that even higher rates are achievable since in the experiments performed, the average frame reception time was always significantly shorter than the frame duration. Furthermore, due to rapid progress in computer technology, more and more powerful processors are being available in the market.

However, it should be also stressed that in the studied cases the reception chain did not comprise some blocks often required in radio receivers, such as descrambler, deinterleaver, outer channel decoder or source decoder. Implementing those blocks, even in separate threads, will increase the total processor occupation, which may have an adverse effect on the overall receiver performance. A possible solution to this problem is using GPU to carry out the additional processing.

Author ORCIDs. S. Kozłowski, 0000-0002-0906-2625.

Acknowledgement. The presented work was a part of the SACC (Satellite Adaptive Communication Channel) project funded by the Government of Poland through an European Space Agency contract under the PECS (Plan for European Cooperating States).

References

1. **Digital Video Broadcasting (DVB)** (2005) DVB-S2 adaptive coding and modulation for broadband hybrid satellite dialup applications, ETSI TS 102 441 V1.1.1.
2. **IEEE Std 802.11n 2009 Part11** (2009) Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: enhancements for higher throughput, IEEE Standard 802.11n.
3. **Sims WH, Varnavas K and Eberly E** (2014) High speed low cost telemetry access from space development update on programmable ultra lightweight system adaptable radio (PULSAR). Presented at 28th Annual AIAAA/USU Conference on Small Satellites.
4. **Sauer C, Dickinson J and Epperly M** (2011) A reconfigurable, radiation tolerant S-Band radio for space use. In *Proc. IEEE Aerospace Conf.*, Montana: Big Sky, pp. 1–6.
5. **Kozłowski S, Kurek K, Skarzyński J, Szczygielska K and Darmetko M** (2018) Investigation on adaptive satellite communication system performance using SDR technique. *22nd International Microwave and Radar Conference (MIKON)*, Poznan, Poland, pp. 363–366. doi: 10.23919/MIKON.2018.8405226.
6. **Skarzynski J, Darmetko M, Kozlowski S and Kurek K** (2016) SDR implementation of the receiver of adaptive communication system. *Radio Science* **51**, 344–351. doi: 10.1002/2015RS005899.
7. **Digital Video Broadcasting (DVB)** (2009) Second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications (DVB-S2), ETSI EN 302 307 V1.2.1.
8. **Digital Video Broadcasting (DVB)** (1997) Framing structure, channel coding and modulation for 11/12 GHz satellite services. ETSI EN 300 421 V1.1.2.
9. **Kozłowski S** (2018) A carrier synchronization algorithm for SDR-based communication with LEO satellites. *Radioengineering* **27**, 299–306. doi: 10.13164/re.2018.0299.
10. **USRP N210 product details** (2018) [Online]. Accessed: Sept. 2018. https://www.ettus.com/product/details/UN210-KIT
11. **NI USRP 2920 product details** (2018) [Online]. Accessed: Sept. 2018. http://www.ni.com/pl-pl/support/model.usrp-2920.html
12. **Tseng S-M, Kuo Y-C, Ku Y-C and Hsu Y-T** (2009) Software Viterbi Decoder with SSE4 Parallel Processing Instructions for Software DVB-T Receiver. *2009 IEEE International Symposium on Parallel and Distributed Processing with Applications*, pp. 102–105.
13. **De Mesmay F, Chellappa S, Franchetti F and Puschel M** (2010) Computer generation of efficient software Viterbi decoders. *High Performance Embedded Architectures and Compilers* **5952**, 353–368.
14. **Li R, Dou Y, Li Y and Wang S** (2013) A fully parallel truncated Viterbi decoder for Software Defined Radio on GPUs. *2013 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 4305–4310.
15. **Intel Intrinsics Guide** (2018). [Online]. Accessed: Sept. 2018. https://software.intel.com/sites/landingpage/IntrinsicsGuide/
16. **Intel Support** (2018) Frequently asked questions for Enhanced Intel SpeedStep Technology for Intel Processors. [Online]. Accessed: Sept. 2018. https://www.intel.com/content/www/us/en/support/articles/000007073/processors.html
17. **Microsoft Developer Network (MSDN)** (2018) [Online]. Accessed: Sept. 2018. https://docs.microsoft.com/pl-pl/windows/desktop/ProcThread/scheduling-priorities

Sebastian Kozłowski received the M.Sc. and Ph.D. degrees in telecommunication from Warsaw University of Technology (WUT), Warsaw, Poland, in 2004 and 2011, respectively. Since 2011, he has been an Assistant Professor with the Institute of Radioelectronics and Multimedia Technology, WUT. His research interests include propagation in wireless systems, MIMO technique, software-defined radio, and cognitive radio.

Krzysztof Kurek received his Ph.D. degree in electronic engineering from Warsaw University of Technology (WUT), Faculty of Electronics and Information Technology, in 2002. Since 2002 he has been an Assistant Professor with the Institute of Radioelectronics and Multimedia Technology, WUT. His research interests include the areas of radio communications and microwave technology: propagation in wireless systems, radiofrequency engineering, mobile systems, satellite communications, and space technologies. He is a tutor of the Student Space Engineering Scientific Group. Dr. Kurek is a member of the Space Research Committee of the Polish Academy of Sciences since 2007.

Jacek Skarzynski received the B.Sc from Warsaw University of Technology (WUT), Faculty of Electronics and Information Technology, in 2015. He

https://doi.org/10.1017/S1759078719000552 Published online by Cambridge University Press

is currently a graduate student at the same university. His research interests include real-time digital signal processing in radiocommunication systems and low-level software optimization. He works at Sigma Connectivity Sp. z o. o. company.

Katarzyna Szczygielska received her M.Sc. from Warsaw University of Technology (WUT), Faculty of Electronics and Information Technology, in 2014. During studies, she was a member of the Student Space Engineering Scientific Group. Her research interests include radiocommunication systems, software applications, and digital signal processing.

Marcin Darmetko received his M.Sc. from Warsaw University of Technology (WUT), Faculty of Electronics and Information Technology, in 2011. His research interests include communication systems, signal processing, and FPGA devices. He works at the Space Research Centre of the Polish Academy of Sciences.

https://doi.org/10.1017/S1759078719000552 Published online by Cambridge University Press