100 Gbps data link layer – from simulation to FPGA implementation

L. Lopacinski, M. Brzozowski, R. Kraemer, S. Buechner, and J. Nolte

Abstract — in the paper, simulation and hardware implementation of a data link layer for 100 Gbps Terahertz wireless communication is presented. The overhead of protocols and coding should be reduced to a minimum. This is especially important for high-speed networks, where a small degradation of efficiency will degrade the user data throughput by several Gbps. The following aspects are explained: an acknowledge frame compression, the optimal frame segmentation and aggregation, Reed-Solomon forward error correction, an algorithm to control the transmitted data redundancy (link adaptation), and FPGA (field programmable gate array) implementation of a demonstrator. The most important conclusion is that changing the segment size influences the uncoded transmissions mostly, and the FPGA memory footprint can be significantly reduced when the hybrid automatic repeat request type II is replaced by the type I with a link adaptation. Additionally, an algorithm for controlling the Reed-Solomon redundancy is presented. Hardware implementation is demonstrated, and the device achieves net data rate of 97 Gbps.

Keywords — aggregation, ARQ, FEC, HARQ, Reed-Solomon, segmentation, link adaptation

1. Introduction

Within the last two years, a few new approaches for 100 Gbps wireless communication have been proposed. Research on physical transceivers and baseband processing changed the state of the art in the targeted area. Blocks required to modulate the 100 Gbps wireless signal in the Terahertz band are close to release in engineering samples. In [1] a 100 Gbps baseband signal has been sent over a 237.5 GHz link. Similar results are shown in [2]. More THz communication activity on the physical layer is documented in [3, 4, 5, 6]. In this paper, we design a data link layer for a wireless 100 Gbps system. The proposed solution is 14 times faster than the state of the art 802.11ac (5 GHz) and 802.11ad (60 GHz) WLANs [7]. Even if the achievement in 100 Gbps wireless communication is impressive, the PHY, baseband, and data link layer have not been integrated yet. To the authors best knowledge, the fully functional data link layer dedicated for 100 Gbps wireless THz application has not been shown anywhere in the world.

2. Related Work

Many research efforts have been addressed to highly efficient wireless protocols. A data link layer goodput analysis is a very popular topic, especially for WLAN. Our methodology for a frame segmentation is very similar to efforts presented by T. Li et al. [8], where segmentation is deeply investigated. T. Li proves that a frame fragmentation may increase a protocol efficiency. There are many authors, who publish papers similar to work of T. Li, for example: [9, 10, 11]. They consider possible improvements for the WLANs, mostly by using fragmentation and aggregation. The main difference is that we are strongly focused on ad-hoc connections for short distances with the highest possible efficiency (95% and more), and data rate of 100 Gbps.

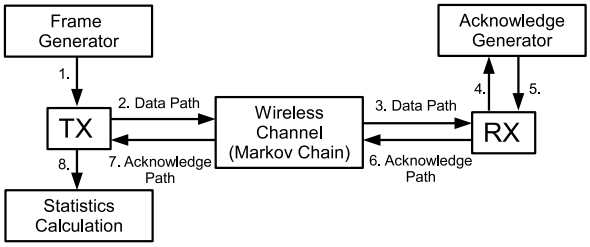

Fig. 1. A Matlab model used to generate transmission statistics. The receiver uses an acknowledge generator to build the ACK-frame. The transmitter uses the frame for retransmissions and statistic calculations.

Another deeply investigated topic is an automatic repeat request (ARQ). Similar work can be found in [12, 13]. We have focused our work on the ARQ concatenated with forward error control codes (FEC) [14]. Such technique is called hybrid-ARQ (HARQ) [15].

There are only a few wireless transceivers working at high-speed data rates. For example, ref. [16] introduces a system for wireless communication working at the 60 GHz frequency band. However, the supported data rate of 4 Gbps is still much lower than our goal: 100 Gbps wireless.

The core task of this paper is to test adaptation algorithms for forward error correction. This allows controlling the redundant data in view of the channel quality.

3. Work details

In this paragraph, we introduce some details of the research. Firstly, we explain how we generate the results. The employed simulation environment and the emulated wireless channel are explained. After that, we describe all implemented techniques used in the research. At the end, our FPGA prototype is presented.

3.1. Simulation Model

We performed Matlab simulations of the planned system, before the real demonstrator was implemented. The simulations are using the same algorithms to the solutions implemented in the hardware. We use field programmable gate arrays (FPGAs) for the final demonstrator.

Fig. 1 explains the simulation model. Two devices are communicating by an emulated wireless channel. The devices are exchanging data frames (the data path) and confirmation

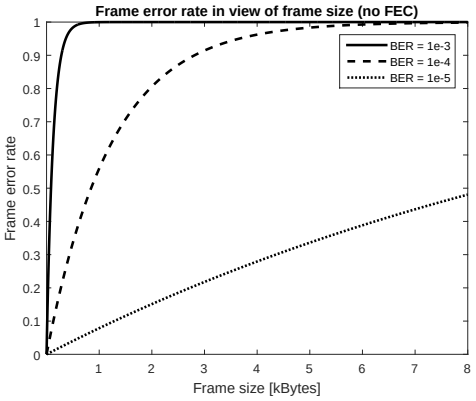

Fig. 2. A frame error rate in view of the frame size. If frame is longer, then higher is the probability that an error will occur during the transmission and the frame will be lost. Due to this aspect, shorter frames are preferred in a noisy wireless environment.

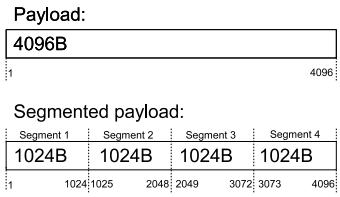

Fig. 3. An example of segmentation for a frame payload. The example payload is chopped to four segments of equal length. The shorter segments are more efficient during a transmission in a noisy channel.

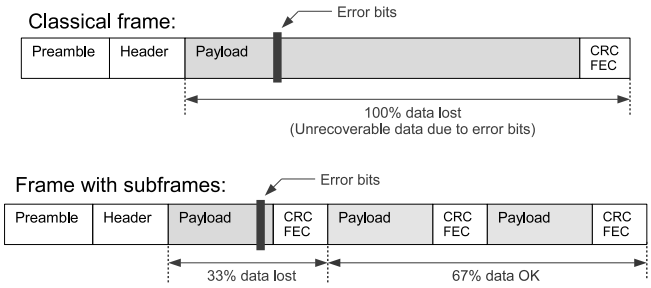

Fig. 4. An explanation how the segmented frame can improve the total efficiency. In a case of bit errors in a classical frame, the whole payload has to be retransmitted. If the segmented frame is used, then only invalid data part is rejected and there is no need to retransmit the whole frame, but just only the defected segment.

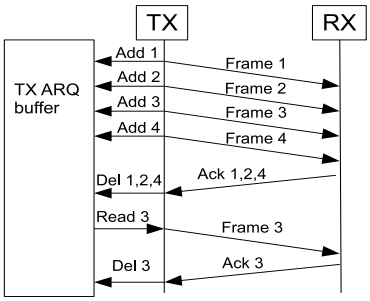

Fig. 5. An automatic repeat request process (ARQ). All transmitted frames are copied to the temporary TX ARQ buffer. If any frame will be lost during the transmission, then the transmitter reads the lost data from the buffer and starts the retransmission.

messages (the acknowledge path). Every successfully received data frame is confirmed by the receiver device (RX). That makes the data exchange process reliable, because the transmitter (TX) can repeat all lost frames. This process is called an automatic repeat request (ARQ). The core function of the ARQ process is generation of the acknowledge frame (ACK) and sending it to the transmitter device. Additionally, the TX device can calculate communication statistics. That allows estimating the efficiency of the implemented algorithm.

3.2. Wireless channel emulation

In this subsection, we introduce the implementation of the wireless channel used in the simulation (according to Fig. 1). We use a two state Markov Chain for errors emulation. This solution requires two transition statistics, which defines the channel. The probabilities of the transition define a bit error rate and error length in bits. It does not use any physical aspects of the wireless transmission. For testing the data link layer it is acceptable. We need to know the characteristic and distribution of the errors. The cause is unimportant, until the parameters describe the channel moreover correctly. A detailed description of the Markov Chain can be found in [11].

3.3. Frame segmentation and aggregation

A frame size and a bit error rate (BER) have a significant impact on the efficiency of the wireless communication. When the payload is longer in the frame, then less overhead is generated by the headers and checksums. Transmission is more efficient. Unfortunately, long frames are more vulnerable to transmission errors. This is explained in Fig. 2. If the frames become longer, then higher is the probability that some bits in the frame will be corrupted. The frame can be split to independent segments, to improve the robustness and efficiency of the communication. The splitting process is explained in Fig. 3. In the example, a single 4 kB frame is split to four 1 kB segments. Now, the individual segments are acting like sub-frames (frame fragmentation). Every segment is using an individual header and checksum, but the preamble is shared (frame aggregation). It means that the errors in one segment do not influence the payload in the other segments. That improves the communication efficiency (Fig. 4). In case of a bit error, only the defected part must be repeated but not the complete frame. In our case, the default frame size is 64 kB, and is segmented to 64 fragments. In a single ARQ session 64 frames are transported (4 MB). The FPGA implementation allows changing the frame settings in the fly, and only the on-chip memory buffers are limiting the flexibility of the frame format.

3.4. Automatic repeat request process

We have already mentioned that the TX and RX devices are working in a closed feedback loop. This loop is called ARQ. Every frame sent by the TX device is locally copied to the TX ARQ buffer (Fig. 5). If the RX will not acknowledge all of the sent frames, then the TX reads the lost frame from the buffer and makes retransmission. The retransmission process repeats until the positive ACK for the frame is received. If the ACK frame is lost, then the transmitter sends an ACK-request frame

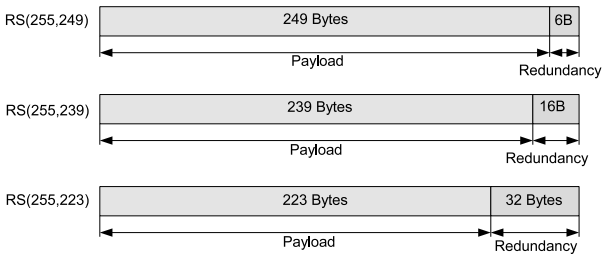

Fig. 6. The Reed-Solomon (RS) blocks. The algorithm is building the blocks of size of 255 bytes in our case. The redundancy is adjustable. If more redundancy bytes are used, then less payload is carried by the segments. More redundancy bytes allow to correct more errors after the transmission.

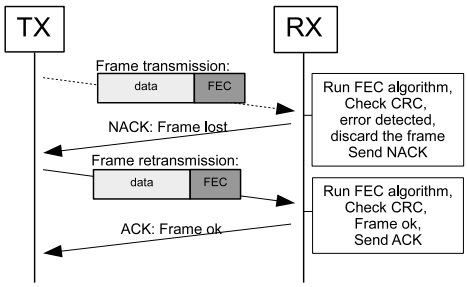

Fig. 7. The HARQ-I scheme. The transmitter always sends the frame with a forward error correction data. The retransmitted frame is a mirror copy of the original frame.

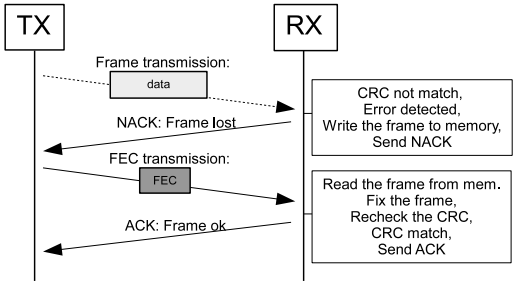

Fig. 8. The HARQ-II scheme. The transmitter usually sends the frame without forward error correction data. The standard frame is not extended by the FEC field. In a case when the frame is lost, then the transmitter sends the FEC only. The frame data is not retransmitted. The HARQ-II reduces the retransmission overhead in compare to the HARQ-I.

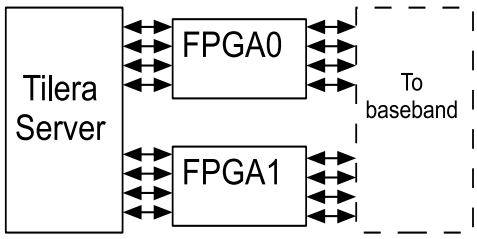

Fig. 9. The demonstrator overview.

after a predefined timeout. In our case, we have adopted this procedure to our implementation. Instead of acknowledging of full frames, every single segment (sub-frame) is acknowledged. It means that the ARQ process works on frame fragments but not on full-frames. For our FPGA prototype, an additional future is used. The implementation uses a zero-copy approach. The transmitted data is not copied to a dedicated buffer, but a pointer to a memory segment is requested from a higher layer in a case of retransmission. This saves energy and reduces memory footprint for the FPGA.

3.5. Forward error correction

The FEC algorithms are reducing the number of retransmitted frames in the ARQ process. That significantly improves the transmission efficiency. The transmitter is sending the data with some redundant bytes. In our work, we are using Reed-Solomon (RS) codes. We have selected the RS codes due to relatively high throughput. We will skip a detailed introduction to the FEC. This topic has many complicated aspects. More details can be found in [18, 19]. We will just explain how the RS is building the blocks. It is important to understand results of our paper. We use in the simulation three RS flavours: RS(255,249), RS(255,239), and RS(255, 223). The numbers are defining the RS block size (255 Bytes in our case) and the payload size (249, 239 or 223 Bytes). It means that the redundant information is 6, 16, or 32 Bytes long. This is explained in Fig. 6. The redundant bytes are used for error corrections. If more redundant data is produced, then more error symbols can be corrected. The RS(255,249) can correct up to 3 Bytes in the block, RS(255,239) 8 Bytes, and the RS(255, 223) 16 Bytes [18]. Our task is to find a trade-off between the redundancy and the payload, so the transmission process is efficient. The VHDL implemented FEC engine for the FPGA is more flexible, and more RS flavours is available. The implemented FPGA FEC engine is supporting any coding in a range of 2-18 redundancy bytes per a single RS block. It means that the following coding schemes are supported: (255,237), (255,239), (255,241), (255,243), (255,245), (255,247), (255,249), (255,251), and (255,253). Coding can be adjusted on the fly, and this feature is used by the proposed adaptation algorithm to choose the optimal coding for the current wireless channel condition. In our case, the higher coding granularity improves the overall performance.

The RS calculation is the most calculation demanding operation performed in the FPGA logic. The encoders and decoders occupy 55% of the FPGA logic resources. To support the targeted 100 Gbps stream, eighty encoders and eighty decoders are in use.

3.6. Hybrid ARQ

Any combination of the ARQ and FEC is called Hybrid-ARQ (HARQ). Two mainly investigated in the paper HARQ methods are HARQ type I and II. The HARQ-I adds error detection code and FEC to every packet at every condition. The HARQ-II sends the FEC data during the retransmission only. In such case, the error correction data is not overloading the link during the regular transmission (Fig. 7 and Fig. 8). This can introduce some improvements in efficiency. We answer in the next paragraph, which strategy is better for our protocol. A detailed description of the HARQ-I and II can be found in [18] and [20].

Fig. 10. The FPGA demonstrator.

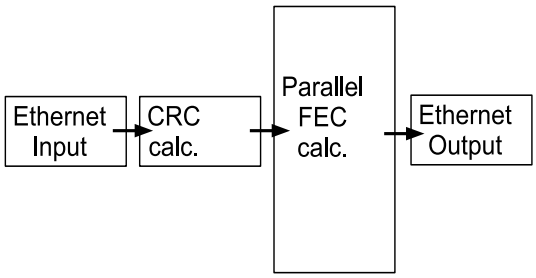

Fig. 11. A single processing lane (logical pipeline).

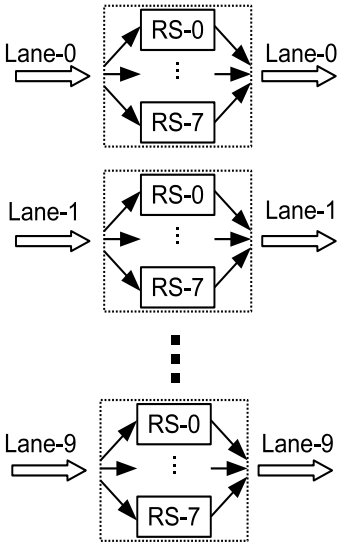

Fig. 12. The parallel FEC calculation array implemented in the FPGA logic.

3.7 FPGA demonstrator

The hardware demonstrator consists of two hardware parts (Fig. 9). The Tilera server is a dedicated 72 cores processor employed for frames segmentation and fast memory access. The FPGA is a calculation coprocessor supporting CRC, FEC calculations, and frames aggregation. The main state machine responsible for data link layer is run on the Tilera server. The FPGAs and sever are connected with 10G Ethernet optical fibers. For now, the architecture supports up to 80 Gbps with two FPGA boards (interfaces constraints). Generally, the Virtex7 FPGA can process up to 100 Gbps in a back-to-back connection (Fig. 10). The baseband processor is not finished yet. Thus, we can test our processor only in a loopback mode. A single, logical FPGA processing pipeline ('lane') is shown in Fig. 11.

3.8 Parallel FPGA processing

There is no possibility to process the 100 Gbps stream in a single processing pipeline (lane) [14]. Even if one of the fastest FPGA developments kit is used, the stream processing have to be divided and calculated in parallel. For that purpose, a parallel calculation array is implemented. The array calculates 640 bits @ 156.25 MHz. Internally the 640-bits-word is organized in ten sub-words processed by ten calculation lanes (Fig. 12). Every lane runs at 10 Gbps, and is connected to two 10G Ethernet ports (data input and data output). Such processor uses 294115 lookup-tables and 239019 flip-flops. It is respectively 65% and 27% of the total resources available in the Virtex7-690T FPGA. The slices occupation is equal to 80%.

4. Results

4.1. Transmission limiting factors

We have performed transmission experiments and recorded the most important parameters (the overall efficiency, the percentage of successfully received segments, the percentage of successfully received frame headers, the total number of acknowledge frames, the number of timeouts, and the total number of physical layer turnarounds). That allows us to investigate which factors reduce throughput in our system. Additionally, the retransmission segment size can be adjusted in a range of 32 to 65536 Bytes. A following assumption can be done after analysis of the results. The ACK-frame has to be as short as it is possible and always encoded with robust coding. Practically it means that the ACK-frame should be encoded with a code rate lower that the code rate of the data segments (a lower code rate means improved error correction). This reduces the total number of lost ACK-frames, timeouts, and PHY turnarounds. After that, only the loss of the data segments limits the throughput. Intensive FEC coding and segmentation for the data segments makes no sense without improved reliability of the ACK-frame. Fig. 13 and Fig. 14 demonstrate the used methodology for un-coded and encoded transmissions. In both cases the throughput is limited by lose of the data segments but not by the ACK-frames. The total

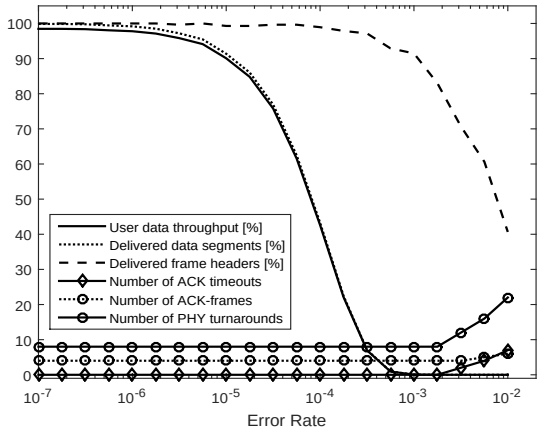

Fig. 13. Limiting factors of the transmission. The data segments are uncoded. The frame headers are delivered with a relatively low error rate. The goodput is limited by lose of the data segments.

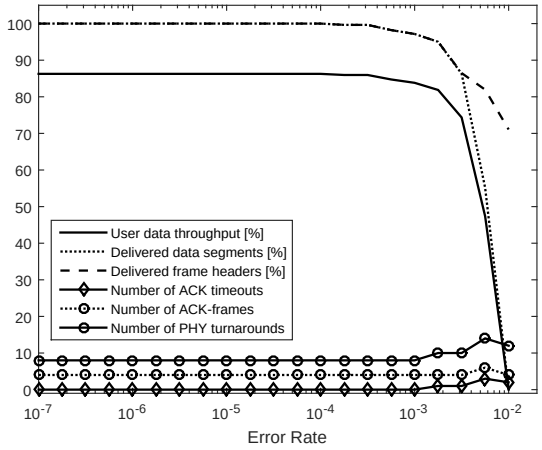

Fig. 14. Limiting factors of the transmission. The data segments are coded with RS(255, 223). The error rate of the data segment is strongly reduced as compared to uncoded transmission simulation.

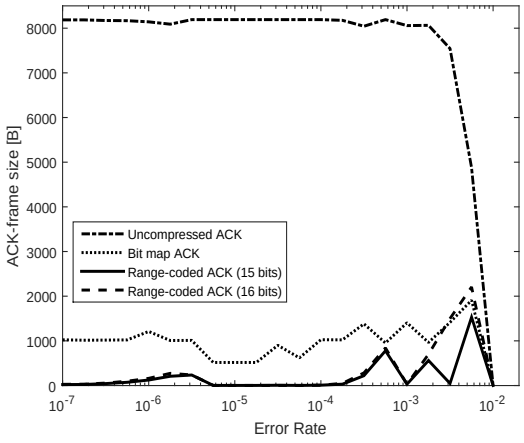

Fig. 15. The maximal ACK-frame sizes during the simulation. Three types of the ACK-frame compression methods are presented. The compressed ACK-frame is significantly shorter and is much more robust during the transmission.

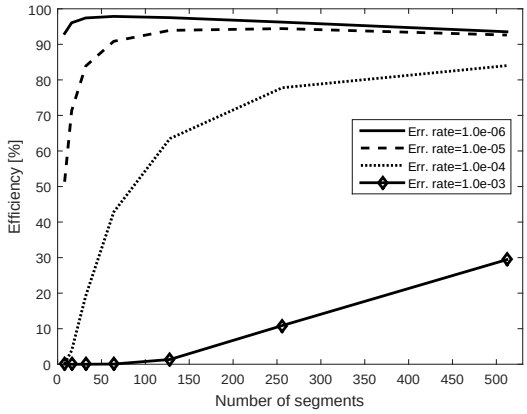

Fig. 16. The data link layer efficiency vs. the data segment size vs. an error rate. The data segments are uncoded. If the error rate increases, then smaller segments are preferred.

number of timeouts and the PHY turnarounds are relatively low during the simulation.

The ACK-frame length is depended from the total number of successfully received segments in a single ARQ session (positive acknowledgment). If the data frame segmentation is increased, then many small parts have to be sent and acknowledged. That increases the ACK-frame size. Unfortunately, too long ACK-frames cannot be delivered errorless and are limiting the throughput. Instead of the efficiency improvement, a degradation is observed. The ideal solution is to keep the ACK-frame size smaller than the size of the data segment. An ACK-frame compression is needed to achieve that in our case. We considered three solutions: a bit map coding, and two versions of a sequence number range coding. A single uint16 value and a bit map are sent in the bit map scheme. The uint16 value defines the first acknowledged segment number, and the bit map defines all next values. The bit position defines an offset and the bit value defines if the segment is acknowledged or not.

The second and third methods send only a range of addresses of the acknowledged segments. In some cases that may lead to an extended frame size. All three methods were investigated, and the results are shown in Fig. 15.

4.2. Optimal segment size

If the problem of the disadvantageous ACK-frame size is reduced, then additional improvements for the data segments can be done. First of all, we consider influence of the segment size. By reducing the segment size, the efficiency can be improved on “bad” channels. From the other side, more segments have to be sent to transmit the same data. Every segment is equipped with an individual header and checksum. This induces overhead. Additionally, enabling the FEC introduces some additional issues. This happens because block codes are used (in our case the RS block size is equal to 255 bytes). This introduces additional indirect-segmenting. The errors in each RS block are corrected individually, and each RS block acts like an independent sub-segment. In Fig. 16 the data segment size is investigated. It can be observed that the optimal segment size for error rates below 1e-6 is in the range

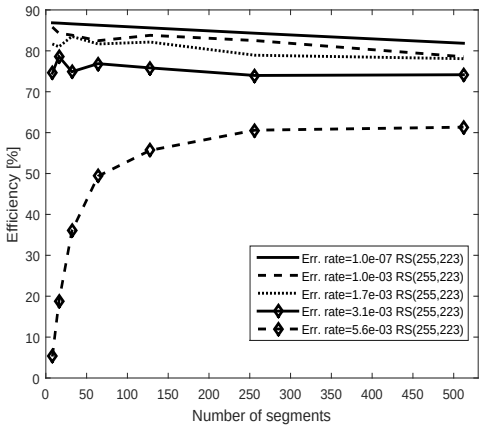

Fig. 17. The data link layer efficiency vs. the data segment size vs. an error rate. The data segments are coded with the RS(255,223). The RS encoded frames are less sensitive to the segment size that the uncoded frames.

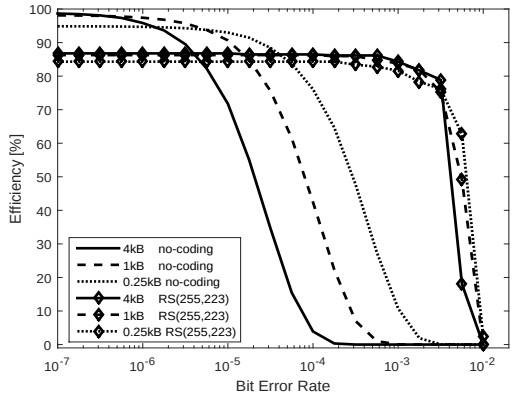

Fig. 18. The data link layer efficiency vs. the data segment size vs. an error rate. The uncoded and RS coded transmissions are plotted in one figure. The RS encoded frames are less sensitive to the segment size than the uncoded frames.

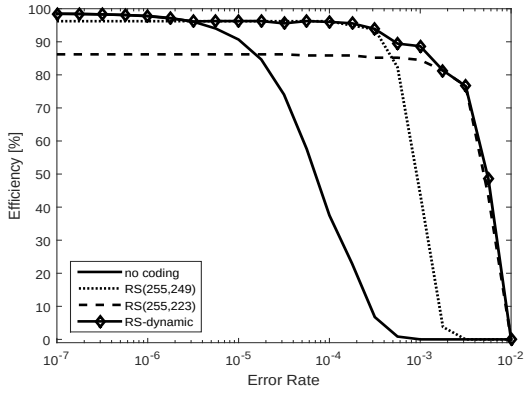

Fig. 19. The dynamic FEC algorithm results. The probability P(C) is equal to 0.5. The adaptive algorithm chooses the optimal coding and maximizes the goodput.

of 2 to 4 kB (16 to 32 segments for a 64 kB frame size). When the error rate increases, then the segment size should be reduced. Five hundred and more segments are required for

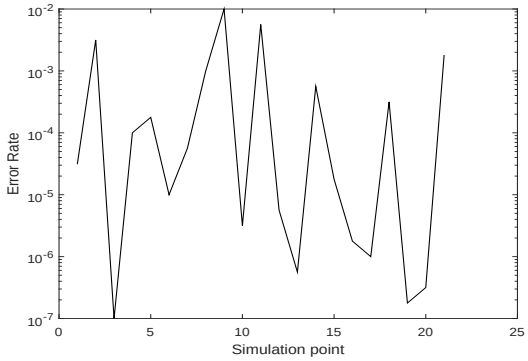

Fig. 20. An example characteristic used for evaluation of adaptive algorithms. The error rate is a permutation in a range of [1e-7; 1e-2].

links with an error rate higher than 1e-5. It means that the transmission without coding is very sensitive to the segment size. Dynamic change of this parameter can introduce some significant improvements to the efficiency. Slightly different situation can be observed, when RS coding is used. This situation is shown in Fig. 17. The transmission with RS coding is less sensitive to the segment size. That means that, advantages of the variable segment size can be reduced after enabling the coding. In our FPGA demonstrator, we skip the implementation of this feature in the first iteration. We presume that the block-FEC can be a good substitute of the variable segment size. To get better feeling of this observation, more simulations were performed (Fig. 18). The improvement of the variable segment size for the RS-coded transmissions is marginal.

4.3. Dynamic FEC redundancy

In this paragraph, a dynamic algorithm to find a trade-off between the FEC coding and the demanded error correction performance is proposed. The algorithm analyses the number of successfully delivered data segments and the number of corrected errors in the RS blocks. If the efficiency is degraded by loses of the data segments, then the algorithm increases the FEC coding. This solution is uncomplicated, but it is important to define a threshold, when the FEC mode should be changed. In this paper, we set the thresholds to 249/255≈97.6%, 239/255≈93.7%, and 223/255≈87.5%. If the data delivery efficiency is below the given values, then the corresponding RS code is used. It tries to find a compromise between the RS overhead and the rate of the lost segments. The thresholds correspond to the code rates and define upper bounding of the goodput. In our solution, we calculate error statistics of all decoded RS blocks, and we categorize all corrupted segments to some groups. Every error category can be corrected by a different RS code. If the statistic is known, then the best RS code can be chosen for all future transmissions. Results of the algorithm are shown in Fig. 19. It may happen that the channel changes so rapidly that this solution will work too slowly. To minimize this factor, HARQ-II can be applied. After that, any mistake of the adaption algorithm can be corrected by the FEC data sent in the next ARQ session. We performed an additional simulation

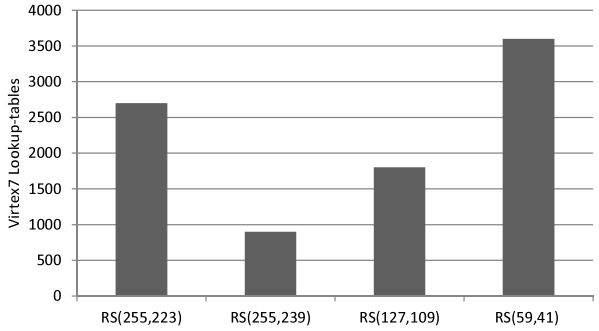

Fig. 21. Consumed logic area by the proposed solutions.

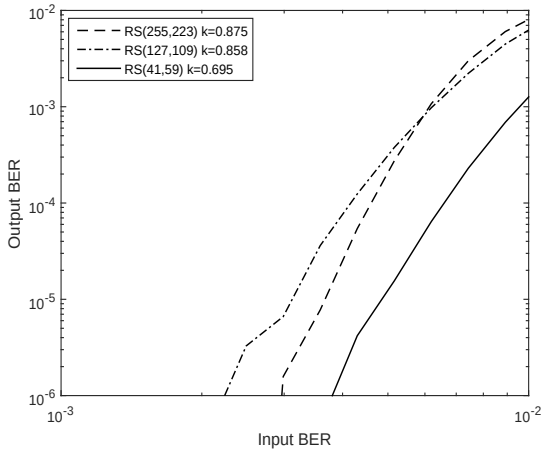

Fig. 22. Error correction performance of the proposed coding schemes.

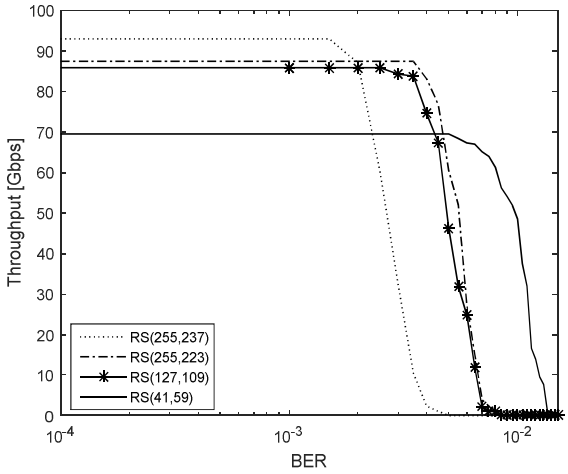

Fig. 23. Simulated throughput of the FPGA demonstrator with the improved FEC engine.

to test how the algorithm performs in rapidly changing environment (Fig. 20). Results of this simulation are presented in Table 1. The algorithm achieves quite high efficiency. It is possible to improve the switching logic in the future. For example, a proportional-integral-derivative (PID) or a fuzzy logic controller can be employed. These controllers can rely not only on the instantaneous value, but can track more parameters on a longer period.

TABLE I

PERFORMANCE FOR THE RAPIDLY CHANGING CHANNEL

Algorithm	**Average efficiency [%]**	**Peak efficiency [%]**
1. No coding (ARQ)	54.72	98.47
2. RS(255,249) (HARQ I)	74.69	96.23
3. RS(255,239) (HARQ I)	79.84	92.43
4. RS(255,223) (HARQ I)	79.19	86.20
5. Adaptive RS (HARQ I)	79.66	98.33
6. HARQ II with RS (255,223)	74.82	98.33
7. Adaptive RS with HARQ II	79.57	98.13
8. Adaptive RS (modified)	83.05	96.28
9. Adaptive RS with HARQ II (modified)	82.46	96.10

Tab. 1. Different algorithms vs. the rapidly changing channel (Fig. 16). Nine algorithms were tested. The best performance is achieved by using adaptive redundancy with the HARQ-I scheme. This algorithm is relatively easy to implement in the FPGA hardware. The HARQ-II scheme is giving similar results, but the complexity of the HARQ-II is higher.

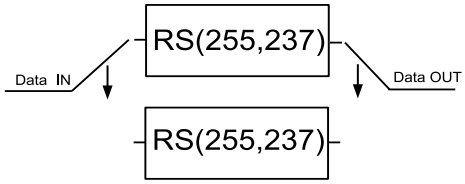

Fig. 24. Implemented code shortening.

4.4. Performance of the FPGA implementation

The back-to-back connected FPGAs and the implemented wireless channel emulator are used to test the algorithms in a real hardware. Our FPGA-implementation accepts a BER up to 2×10^{-3}. Above this value, the RS engine cannot fix errors in the stream, and the performance rapidly drops. In some cases, the wireless channel may produce BER higher than 2×10^{-3}. The hardware-implemented data link layer cannot operate in such conditions, and the device will lose the link. To improve the error correction results, we propose an extended version of our FEC engine.

In the simulation, we presume that the engine must support at least the RS(255,223) with code rate $R\approx0.875$. The used RS VHDL-implementation cannot support a lower coding than the RS(255,237) with code rate $R\approx0.929$. Thus, the achieved FPGA results are worse than simulated. There is a possibility to use shortened RS codes to decrease the code rate of the produced stream. That is the easiest approach to deal with the problem. The second solution is to redesign the implemented RS entity, that it can natively support the RS(255,223) [21]. We compare the both approaches in terms of consumed logic area (Fig. 21) and error correction performance (Fig. 22 and Fig. 23).

The implementation of the RS(127,109,8-bit symbol) is realized by shortening the RS(255,237) by removing 128 symbols from the message part of the codeword. Practically, it is achieved by using two hardware entities of the default code, and by multiplexing/switching the data input and output interfaces (Fig. 24). Every coder calculates half of the data block, and the rest of the symbols are filled with zeroes.

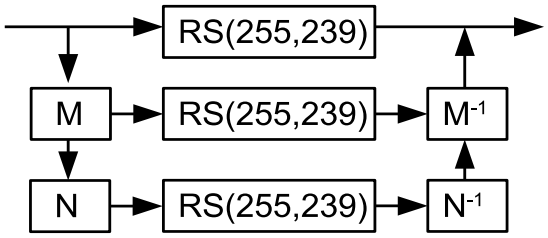

Fig. 25. Experimental RS decoder.

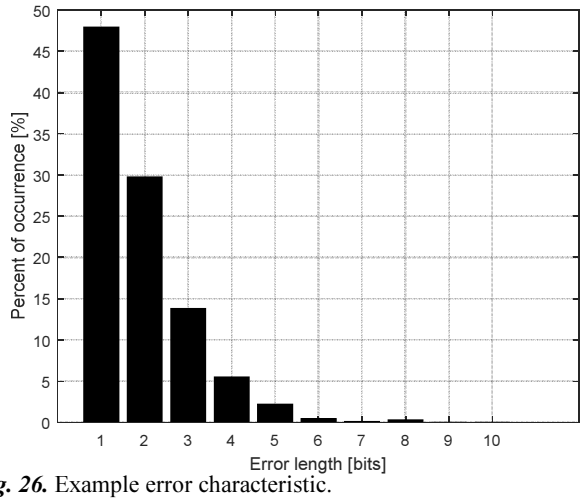

Fig. 26. Example error characteristic.

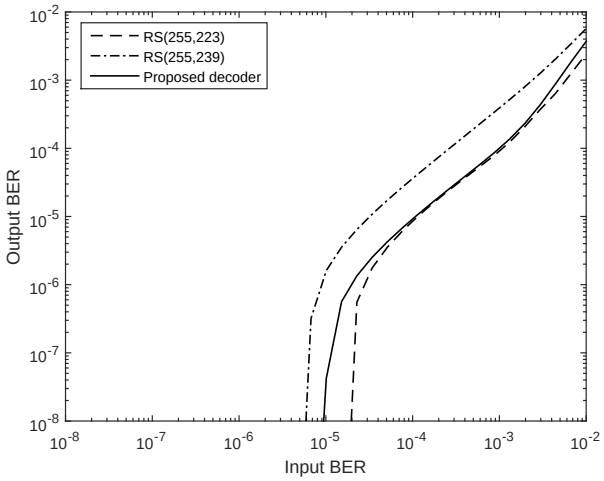

Fig. 27. Error correction results of the proposed decoder.

The RS(59,41,8-bit symbol) is a shortened version of the RS(255,237) by removing 196 data symbols. Practically those are four calculation entities switched four times during a single codeword coding. These uncomplicated operations improve the error correction performance without any significant complications of the system. Especially, the resources used for implementation of the RS(255,223) can be reduced by around 33%, when the coding is replaced with the proposed RS(127,109). This simplification causes a small loss of the error correction performance. The redundancy symbols

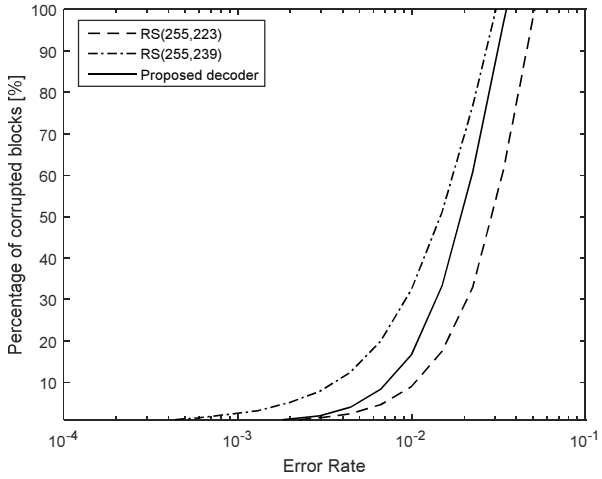

Fig. 28. Block correction results of the proposed decoder.

are spread more uniformly over the frame and the redundancy cannot be used as flexible as during processing of the full-length codewords. The error correction process is focused on shorter blocks, and some of the redundant information is not used efficiently.

5. Future work

We experiment with interleaving and multiplexing matrixes to increase error correction performance of our implementation (Fig. 25). The assumption is to improve decoding performance of the RS(255,239) and to achieve error correction performance similar to the RS(255,223). The proposed decoder must be mathematically analysed and it have to be proven, that the proposed structure requires less calculation operations than the RS(255,223). Up to now, achieved by us results are disappointing. The solution is inefficient against single, uniformly distributed bit errors (e.g. AWGN channel). In such a case, we cannot tune the structure to get any optimistic results. Usually, we consume more power than a single RS(255,239) decoder, and the increase of the error correction performance is marginal.

Better results are achieved if the structure is run against burst errors. In Fig. 26, an example error characteristic is presented. We use the proposed characteristic to test our structure (Fig. 27). The solution achieves very good BER performance, but this is not the most important statistic. Number of bit errors in individual blocks is significantly reduced, but the total number of fully recovered blocks is lower than after the RS(255,233) decoding (Fig. 28).

An additional disadvantage is that the *M* and *N* matrixes (Fig. 25) are dependent from the error characteristic, and are not universal for all burst error lengths. If the length of the typical error produced by a channel is changing, then also the matrixes have to be adopted.

The proposed scheme can be run iteratively. We do not consider an iterative mode of operation due to energy consumption and latency. For now, the presented solution cannot be considered as a substitute of the typical RS decoder.

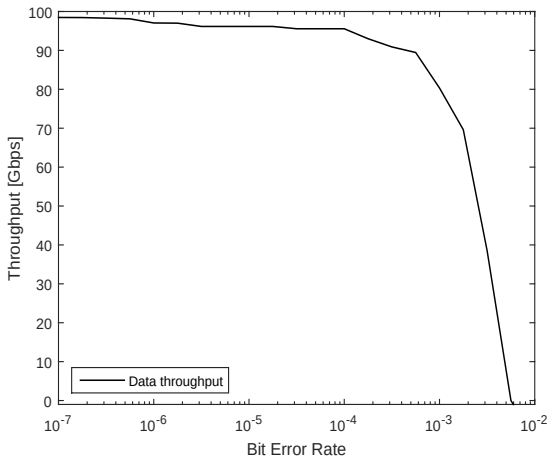

Fig. 29. Performance of the FPGA implementation in view of a bit error rate.

6. Conclusion

In the paper, three major aspects of the 100 Gbps data link layer are explained. Firstly, we analysed the limiting factors of the implementation. The data link layer robustness is improved after introducing the ACK-frame compression and coding. This reduces the total number of timeouts in the system simulation. After that, the data segmentation can be investigated. The most important observation is that the segmentation has more influence on the un-coded transmissions, than for the transmissions coded with the RS block codes. Because of this reason, we skip the implementation of a variable segment size in the first iteration. Instead of it, we focus on the FEC algorithms and a solution to manage the FEC overhead against the transmission requirements. The goal is to use as little overhead as possible and maximize the efficiency.

Link adaptation used with the HARQ-I simplifies the FPGA design, and it is a good substitute of the more complicated HARQ-II method. That allows to remove buffers from the design, due to the fact that broken frames do not have to be buffered.

All presented results are validated on the Xilinx VC709 Virtex7 FPGA platform. The implementation supports a net data rate of 97 Gbps on the real FPGA-hardware (Fig. 29).

Acknowledgements

This paper is related to the End2End100 project and cooperates with other proposed projects of the DFG Special Priority Program 1655 (SPP1655) on “Wireless 100 Gbps and beyond”, e.g. the Real100G.COM and Real100G.RF. This group of projects will investigate a complete wireless 100 Gbps system at ultra-high frequencies (240 GHz).

References

[1] S. Koenig, D. Lopez-Diaz, J. Antes, F. Boes, R. Henneberger, A. Leuther, A. Tessmann, R. Schmogrow, D. Hillerkuss, R. Palmer and others, "Wireless sub-THz communication system with high data rate," *Nature Photonics,* vol. 7, no. 12, pp. 977-981, 2013.

[2] F. Boes, T. Messinger, J. Antes, D. Meier, A. Tessmann and I. Kallfass, "Ultra-broadband MMIC-based wireless link at 240 GHz enabled by 64GS/s DAC," in *Infrared, Millimeter, and Terahertz waves (IRMMW-THz), 2014 39th International Conference on*, 2014.

[3] H. Wang, W. Yuan, B. Zhang, H. Li, Z. Zhang, X. Yang and W. Shi, "The design, test, and application of the front end in 0.3 THz wireless communication systems," in *Selected Proceedings of the Photoelectronic Technology Committee Conferences held June-July 2015*, 2015.

[4] T. Nagatsuma, K. Kato and J. Hesler, "Enabling Technologies for Real-time 50-Gbit/s Wireless Transmission at 300 GHz," in *Proceedings of the Second Annual International Conference on Nanoscale Computing and Communication*, 2015.

[5] I. T. Monroy, "Photonic Techniques for Sub-Terahertz Wireless Data Transmission," in *Photonic Networks and Devices*, 2015.

[6] K. KrishneGowda, T. Messinger, A. C. Wolf, R. Kraemer, I. Kallfass and J. C. Scheytt, "Towards 100 Gbps Wireless Communication in THz Band with PSSS Modulation: A Promising Hardware in the Loop Experiment," in *Ubiquitous Wireless Broadband (ICUWB), 2015 IEEE International Conference on*, 2015.

[7] *Wireless LAN medium access control (MAC) and physical layer (PHY) specifications: Enhancements for Very High Throughput in the 60 GHz Band,* IEEE Std, 12.2012.

[8] T. Li, Q. Ni, D. Malone, D. Leith, Y. Xiao and T. Turletti, "Aggregation with fragment retransmission for very high-speed WLANs," *IEEE/ACM Transactions on Networking (TON),* vol. 17, no. 2, pp. 591-604, 2009.

[9] D. Qiao, S. Choi and K. G. Shin, "Goodput analysis and link adaptation for IEEE 802.11 a wireless LANs," *Mobile Computing, IEEE Transactions on,* vol. 1, no. 4, pp. 278-292, 2002.

[10] D. Skordoulis, Q. Ni, H.-H. Chen, A. P. Stephens, C. Liu and A. Jamalipour, "IEEE 802.11 n MAC frame aggregation mechanisms for next-generation high-throughput WLANs," *Wireless Communications, IEEE,* vol. 15, no. 1, pp. 40-47, 2008.

[11] E. H. Ong, J. Kneckt, O. Alanen, Z. Chang, T. Huovinen and T. Nihtil, "IEEE 802.11 ac: Enhancements for very high throughput WLANs," in *Personal Indoor and Mobile Radio Communications (PIMRC), 2011 IEEE 22nd International Symposium on*, 2011.

[12] S. Choi and K. Shin, "A class of adaptive hybrid ARQ schemes for wireless links," *Vehicular Technology, IEEE Transactions on,* vol. 50, no. 3, pp. 777-790, 2001.

[13] L. Badia, N. Baldo, M. Levorato and M. Zorzi, "A Markov framework for error control techniques based on selective retransmission in video transmission over wireless channels," *Selected Areas in Communications, IEEE Journal on,* vol. 28, no. 3, pp. 488-500, 2010.

[14] M. A. Ingale, "Error correcting codes in optical communication systems," 2003.

[15] S. Falahati and A. Svensson, "Hybrid type-II ARQ schemes with adaptive modulation systems for wireless channels," in *Vehicular Technology Conference, 1999. VTC 1999-Fall. IEEE VTS 50th*, 1999.

[16] M. Ehrig and M. Petri, "60GHz broadband MAC system design for cable replacement in machine vision applications," *AEU-International Journal of Electronics and Communications,* 2013.

[17] E. Esteves, P. J. Black and M. I. Gurelli, "Link adaptation techniques for high-speed packet data in third generation cellular systems," in *European Wireless Conference*, 2002.

[18] S. Lin and D. Costello, Error Control Coding: Fundamentals and Applications, Prentice-Hall Series in Computer Applications In Electrical Engineering, 1983.

[19] L. Lopacinski, M. Brzozowski, R. Kraemer and J. Nolte, "100 Gbps Wireless - Challenges to the data link layer," in *ICTF 2014*, 2014.

[20] H. Chen, R. G. Maunder and L. Hanzo, "A Survey and Tutorial on Low-Complexity Turbo Coding Techniques and a Holistic Hybrid ARQ Design Example," *IEEE Communications Surveys \& Tutorials,* 2013.

[21] M. Marinkovic, M. Krstic, E. Grass i M. Piz, „Performance and complexity analysis of channel coding schemes for multi-Gbps wireless communications," w *Personal Indoor and Mobile Radio Communications (PIMRC), 2012 IEEE 23rd International Symposium on*, 2012.

Brandenburg University of Technology Cottbus-Senftenberg
Platz der Deutschen Einheit 1
03046 Cottbus, Germany
E-mail: lukasz.lopacinski@b-tu.de