Hindawi
Wireless Communications and Mobile Computing
Volume 2017, Article ID 3560521, 11 pages
https://doi.org/10.1155/2017/3560521

*Research Article*

# Data Link Layer Considerations for Future 100 Gbps Terahertz Band Transceivers

**Lukasz Lopacinski, Marcin Brzozowski, and Rolf Kraemer**

*IHP, Im Technologiepark 25, 15236 Frankfurt (Oder), Germany*

Correspondence should be addressed to Lukasz Lopacinski; lopacinski@ihp-microelectronics.com

Received 16 June 2016; Revised 14 September 2016; Accepted 5 October 2016; Published 10 January 2017

Academic Editor: Leyre Azpilicueta



This paper presents a hardware processor for 100 Gbps wireless data link layer. A serial Reed-Solomon decoder requires a clock of 12.5 GHz to fulfill timings constraints of the transmission. Receiving a single Ethernet frame on a 100 Gbps physical layer may be faster than accessing DDR3 memory. Processing so fast streams on a state-of-the-art FPGA (field programmable gate arrays) requires a dedicated approach. Thus, the paper presents lightweight RS FEC engine, frames fragmentation, aggregation, and a protocol with selective fragment retransmission. The implemented FPGA demonstrator achieves nearly 120 Gbps and accepts bit error rate (BER) up to $2e-3$. Moreover, redundancy added to the frames is adopted according to the channel BER by a dedicated link adaptation algorithm. At the end, ASIC synthesis results are presented including detailed statistics of consumed energy per bit.

## 1. Introduction

The fastest wireless technology available, based on wireless LAN 802.11ac (5 GHz) and 802.11ad (60 GHz), achieves data rates of "only" 7 Gbps [1]. Our goal is, to achieve wireless transmission of 100 Gbps, which is a great breakthrough: 10x faster than any other wireless communication. This paper is related to End2End100 project and cooperates with other proposed projects of the DFG SPP1655 [2]. This group of projects will investigate a complete wireless 100 Gbps system at terahertz frequencies (~240 GHz). Figure 1 depicts the demonstrator setup as investigated in the DFG. This paper describes research on the data link layer part of the demonstrator. The baseband and PHY layer are investigated under Real100G.COM activity [3]. More information on the dedicated PHY layer and baseband processing can be found in [4–6].

Within the last three years, a few new approaches for 100 Gbps physical layer (PHY) have been proposed. Table 1 summarizes selected transmission experiments performed in the terahertz band at the PHY layer.

From the data link layer point of view, the research presented in [7] is especially interesting. The authors consider a hybrid-ARQ approach for nanonetworks operating in 300 GHz band with OOK modulation. The simulation uses Hamming (15, 11) channel coding with automatic repeat request (ARQ). The uncomplicated solution is considered due to limited power for the targeted application and cannot be considered as a solution of general purpose 100 Gbps radio transceivers. Our paper proposes significantly more powerful error correction techniques, but also the consumed energy per processed bit is approx. $10^4$ times higher (comparing to the results estimated in [7]).

To perform 100 Gbps transmissions, more than a fast PHY and baseband is required. We focus on the overall transmission efficiency and reduce the overall overhead induced by protocols. As a result of our work an FPGA based data link layer processor is presented. The implementation uses link adaptation methods and process ~117 Gbps of user data.

## 2. Challenges to the Wireless 100 Gbps Data Link Layer Implementation

This section discusses major challenges of implementing data link layer for 100 Gbps wireless communication. A state-of-the-art Virtex7 FPGA is considered as the validation platform and IHP 130 nm technology as the final ASIC implementation.

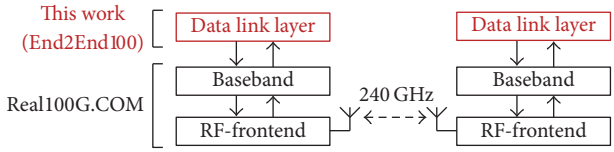


Figure 1: Terahertz demonstrator setup as investigated in the DFG-SPP1655 projects.

Table 1: Summary of reported transmission experiments in the Terahertz band (source: [10]).

| Frequency [GHz] | Data rate [Gbps] | BER | Distance [m] | Reference |
|---|---|---|---|---|
| 120 | 10 | $1e-10$ | 200 | [11] |
| 87.5 | 100 | $1e-3$ | 1.2 | [12] |
| 220 | 30 | $1e-8$ | 20 | [13] |
| 300 | 48 | $1e-10$ | 1 | [14] |
| 237.5 | 100 | $1e-3$ | 20 | [15] |
| 100 | 100 | $3.8e-3$ | 0.7 | [16] |
| 400 | 40 | $1e-3$ | 2 | [10] |

*2.1. Ultra-Short Processing Time.* To achieve 100 Gbps data transmission, a frame of 1500 bytes must be processed within 120 ns. Therefore, an FPGA with 200 MHz clock must process 63 bytes of the frame in each clock cycle. A single Viterbi decoder implemented in Virtex7 FPGA requires approximately 60000 ns to process the data [8], but as mentioned before, the complete processing must be finished in approx. 500 times shorter period. Moreover, transmitting the shortest Ethernet frame (64 bytes) requires ~5 ns. Thus, RAM memory with latency $\ll$5 ns is required. Additionally, at 100 Gbps data rate, the transceivers need 12.5 GB of the memory to store the transmitted data over the last second. State-of-the-art computers and FPGA kits have a few GB of DDR3 or DDR4 memory available. However, the access time to such memory is too slow. The estimated access time of DDR3 memory is around 45 ns [9], but as mentioned before, $\ll$5 ns is required.

*2.2. Bit Errors and Forward Error Correction (FEC).* Today there are various FPGA implementations that support 100 Gbps in wired networks, for example, Ethernet 802.3ba based on optical fiber cables running on Altera FPGA platform [17]. In theory, these high-speed implementations might be used, with some adaptations, as the data link layer of 100 Gbps wireless systems. However, it will work inefficiently because the data link layer of wireless systems must cope with unpredictable bit error rates (BER), leading to more complex solutions [18]. The BER in wireless communication can vary by several orders of magnitude. For example, the BER of high-speed wireless RF frontends presented in [10] achieves BER in range $1 \times 10^{-10}$ to $4 \times 10^{-3}$. Therefore, the FEC has to be adopted to the channel conditions. This increases the average code rate and compensates the unpredictable BER on wireless links.

An additional difference between the optical and wireless communication is duplex switching. Optical communication can use two separated fibers to perform uplink and downlink transmissions. Wireless transceivers are limited in this aspect. In most cases, half-duplex communication takes place. A radio can be in receive or transmit mode, but usually never in both states at the same time. Furthermore, some of the communication time is wasted because of switching the PHY between receiving (RX) and transmitting (TX) modes. Due to the mentioned factors, the data link layer for the wireless 100 G communication should be considered as new research. In the other case it is not possible to achieve high efficiency.

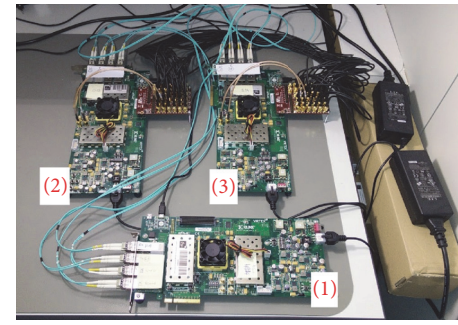


Figure 2: FPGA demonstrator with back-to-back cable connection. The setup consists of test data generator (1), data link layer transmitter with channel emulator (2), and data link layer receiver (3).

*2.3. Forward Error Correction Complexity.* Forward error correction (FEC) allows reducing the effective BER on the data link layer. That significantly improves robustness and efficiency of the system. Unfortunately, the FEC gain is very expensive in meaning of the processing effort. In [19] we introduce logic area consumed by Reed-Solomon, Viterbi, and LDPC decoders. The 1/2-rate Viterbi decoder with 5-bit soft coding presented in [20] requires a logic area of approx. 23 Virtex7 FPGAs to deal with the targeted 100 G stream.

## 3. Work Details

This chapter explains all necessary optimizations required for the implementation of the FPGA data link layer processor (Figure 2). One of the main objectives is algorithms comparison according to the computational complexity, hardware resources, and processing latency.

*3.1. Frames Fragmentation.* Frame error rate depends mainly on BER and frame length. For a longer frame, the probability that at least one of the bits will be altered during transmission is higher, due to the channel impairments. Fewer bits in a frame reduce the number of possibilities for bit errors to occur. Thus, shorter frames are preferred in a noisy channel. This observation leads to a frame-fragmentation concept. Long frames can be split into several shorter frames [22] (Figure 3). This operation improves frame error rate and data goodput (Figure 4). This is especially important for wireless 100 Gbps implementations, where the frame length has to be maximally extended to achieve high transmission goodput and to reduce idle time of the RF-frontend.

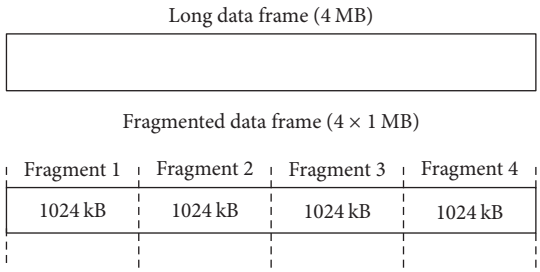


Figure 3: Frames fragmentation: long frames are split into shorter frames. Therefore, the frame error rate is reduced (source: [21]).

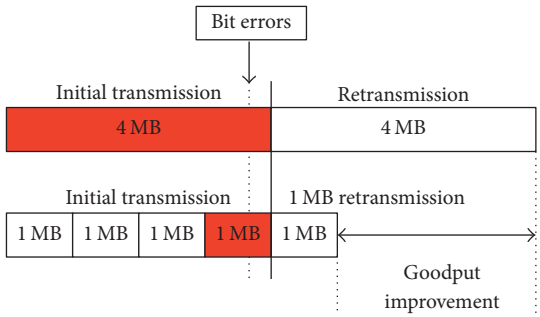


Figure 4: Explanation of improved transmission goodput achieved by fragmented frames. If a frame is fragmented and a bit error occurs in one of the fragments, then only the defected fragment must be retransmitted but not the entire frame.

If the retransmission process is taken into consideration (ARQ), then the probability of successful transmission of a payload encapsulated into smaller frames is higher than probability of transmission of the same payload encapsulated into longer frames. The probability can be calculated by the following equation:

$$P(n) = \sum_{k=1}^{n} \left( \frac{n! \times \left(1 - (1 - \mathrm{BER})^{l}\right)^{n-k} \times (1 - \mathrm{BER})^{lk}}{(n-k)! \times k!} \right), \quad (1)$$

where $P(n)$ is probability of successful frame delivery after $n$ transmissions, $l$ is frame length in bits, and BER is bit error rate.

Figure 5 compares the probability of a successful payload delivery of 1 MB and 4 MB frames as a function of transmission time. Shorter frame achieves significantly higher probability of successful reception.

*3.2. Frames Aggregation and Selective Fragment Retransmission.* Frames fragmentation improves transmission goodput by decreasing frames length in a noisy environment. This reduces frame error rate, but there are also negative aspects of this process. Increased number of frames requires more preambles generated on the PHY level. Each frame is extended by a PHY preamble to set correct RF-gain, synchronize the center frequency, and recover the data clock on the receiver side. In this time, user data is not transmitted

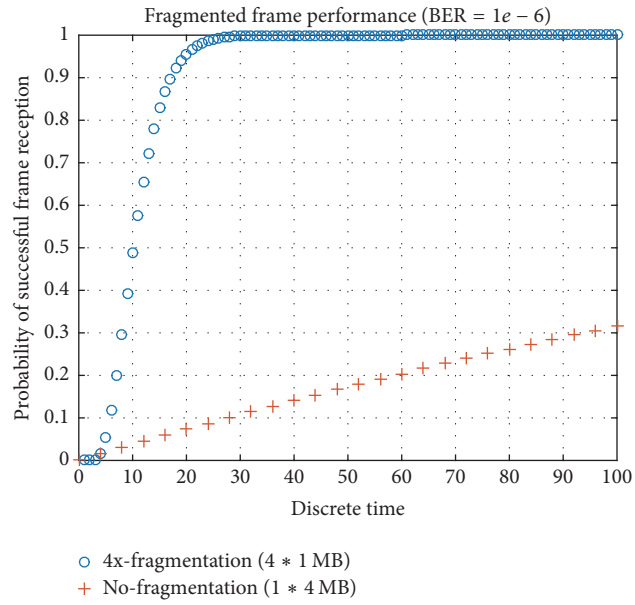


Figure 5: Probability of successful data transmission for 4 MB- and 1 MB-fragmented frames as a function of discrete time.

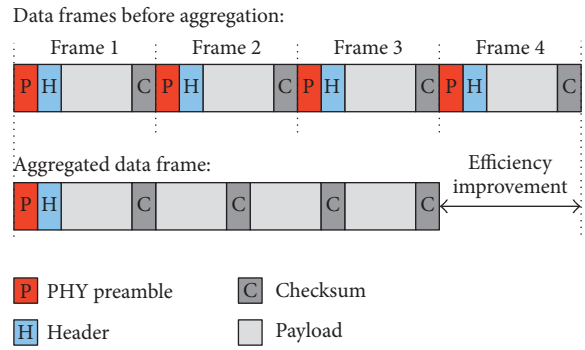


Figure 6: Frames aggregation. The aggregation method reduces the number of transmitted PHY preambles and headers. Thus, protocol goodput is increased.

and the preamble is reducing the goodput. Therefore, the number of transmitted preambles and headers should be reduced. The only way to do it is to extend the frame length as much as possible. This reduces the number of transmitted preambles and headers, but very long frames are not preferred in a noisy RF-environment due to increased frame error rate. This causes an impasse, but there is a possibility to reduce the number of transmitted preambles as well as reduce the logical frame size using frames aggregation and selective fragment retransmission [23] (Figure 6).

The most important aspect of an aggregated frame is resistance to bit errors and reduced preamble-overhead. The fragments of the frame share a common preamble and header, but CRC fields are separate. The CRCs are recalculated for each fragment independently, which allows detection and retransmission of the defected parts individually (Figure 7). Additionally, fragments length can be controlled on the fly according to the channel BER.

Table 2: Required hardware resources for selected forward error correction implementations.

| Implementation | Reference | Max. Clk. [MHz] | LUT area | FPGA/speed grade |
|---|---|---|---|---|
| Xilinx Viterbi decoder | [8] | 286 | 2525 | Virtex7/-1 |
| Xilinx Viterbi decoder | [8] | 403 | 2525 | Virtex7/-3 |
| Creonic Viterbi decoder | [28] | 250 | 2984 | Virtex6/-1 |
| Creonic Viterbi decoder | [28] | 142 | 3054 | Spartan6/-2 |
| IHP Viterbi decoder | [20] | 170 | 12000 | Virtex7/-2 |
| Xilinx RS decoder | [8] | 294 | 765 | Virtex7/-1 |
| Xilinx RS decoder | [8] | 410 | 765 | Virtex7/-3 |
| Xilinx RS encoder | [24] | 388 | 260 | Virtex7/-1 |
| Xilinx RS encoder | [24] | 598 | 260 | Virtex7/-3 |
| IHP RS decoder | [20] | 285 | 2585 | Virtex7/-2 |
| IHP RS encoder | [20] | 457 | 200 | Virtex7/-2 |
| IHP RS decoder | [20] | 270 | 5554 | Virtex7/-2 |

Table 3: Forward error correction algorithms, normalized decoding throughput.

| Implementation | Max. Clk. [MHz] | LUT area | Data throughput/LUT [Mbps] |
|---|---|---|---|
| Xilinx Viterbi decoder $R = 1/2$ (5-bit quantization) | 403 | 2525 | 0,16 |
| Xilinx RS decoder $R = 239/255$ (1-bit quantization) | 410 | 765 | 4,02 |
| IHP LDPC decoder $R = 3/4$ (5-bit quantization) | 160 | 21682 | 0,0074 |

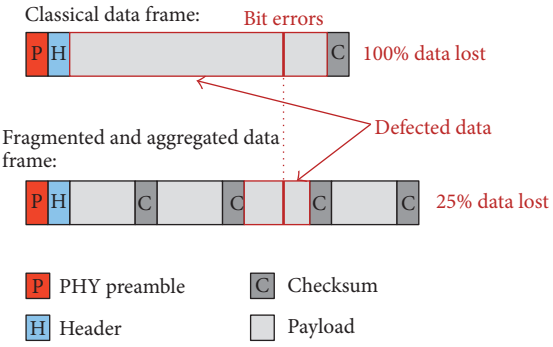


Figure 7: Frames aggregation and fragmentation. In case of bit errors, only the defected frame-fragment is retransmitted instead of the entire frame (selective fragment retransmission).

*3.3. Estimation of Hardware Resources Required for 100 Gbps FEC Decoder.* Tables 2 and 3 compare some common convolutional and RS coders implemented for FPGA technology. In the compared group of algorithms, RS consumes usually fewer hardware resources than the Viterbi. It means that RS obtains higher throughput per a single logic cell (Table 3). This comparison shows the advantages of RS decoders, but the analysis is not correct in this case. Both algorithms are compared in typical configurations. The code rate of the Viterbi decoder is defined as $R = 1/2$ and the code rate of the RS as $R = 239/255$. Thus, the algorithms have different error correction capabilities and such a study is not reliable. RS algorithm requires long decoding pipeline in most implementations and shortening RS (255, 239) codes to obtain $R = 1/2$ is usually not possible for FPGA IPcores [24–27]. Additionally, changing the code rates of the codes may significantly influence the decoding efficiency, and such modified algorithms may be ineffective in terms of obtained coding gain and consumed hardware resources. Moreover, decoding performance depends on the error type on the decoder input, and both solutions prefer different channel types. Thus, comparison of the hardware resources is difficult and can lead to wrong conclusions. Therefore, in this paragraph, a different benchmark is used. The goal is to estimate if 100 Gbps FEC engine based on typical FEC codes fits into a single high-end FPGA (e.g., Virtex7), in other words, if implementation of 100 Gbps FEC engine in an FPGA is achievable. Due to this reason, Table 3 gives an approximation of average throughput per single LUT of a half rate soft decision Viterbi decoder in comparison to a hard coding RS (255, 239) decoder. The normalized decoding throughput of the RS is 25 times higher than for the Viterbi. When the Viterbi solution is scaled for a 100 Gbps system, then the overhead is so high that it is not possible to fit the Viterbi decoder into the targeted xc7vx690tffg1761-2 FPGA. Due to this reason, the soft decision decodable Viterbi decoder cannot be considered for FPGA implementation of the 100 Gbps FEC engine. Error correction performance of the RS is limited but allows communicating over channels with BER $\approx 2e-3$ for single errors.

Table 3 compares LDPC [20] and RS decoders in a similar way. The RS (255, 239) requires less hardware resources than the soft decision decodable LDPC (1536, 1152) to achieve the same decoding throughput. Thus, RS coding is one of assumed solutions to build a FEC engine for the targeted 100 Gbps application.

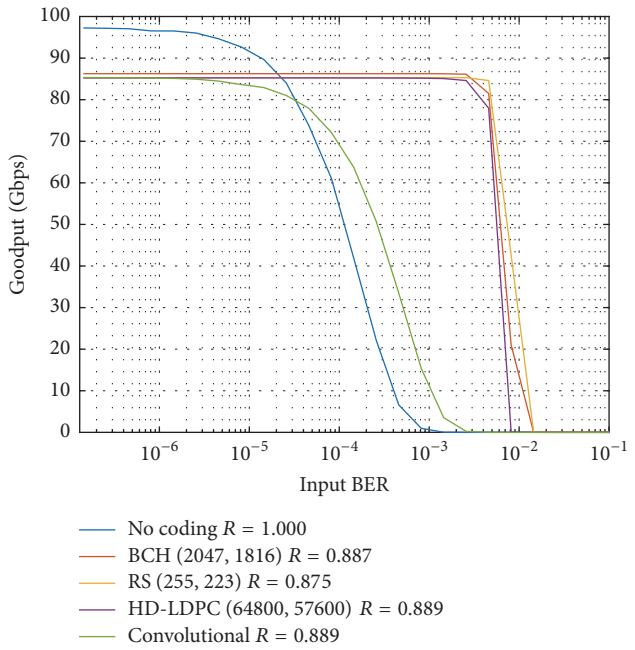


Figure 8: Comparison of selected hard decision decodable FEC algorithms against single errors.

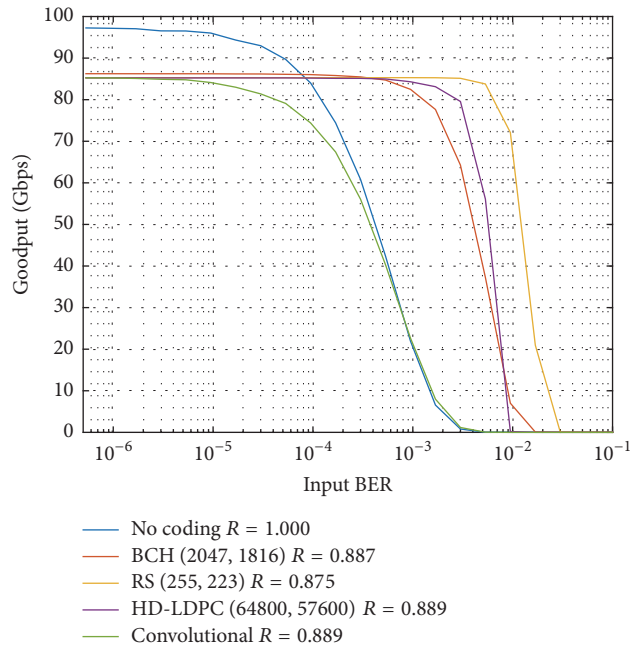


Figure 9: Comparison of selected hard decision decodable FEC algorithms against burst errors.

*3.4. Error Correction Performance of Selected FEC Codes.* The previous paragraph clearly shows that the soft decodable FEC methods require very high computation power. Moreover, we do not have a possibility to realize ADC that supports multibit quantization at the targeted data rate. Thus, error correction performance of hard decision decodable Viterbi, LDPC, and BCH decoders is compared with RS. Firstly, a complete Matlab model of the targeted ASIC/FPGA is implemented, and a simulation against single errors is performed. In the presented case, the selected RS (255, 223), BCH (2047, 1816), and HD-LDPC (64800, 57600) decoders achieve more or less comparable results (Figure 8). The Viterbi decodable convolutional code (with $R = 8/9$) obtains poor error correction performance. Thus, the code should not be considered for the targeted ASIC/FPGA implementation. The LDPC code corrects ~15% higher BER than the RS. The BCH decoder provides the best results and corrects ~30% higher BER than the RS. If a channel with burst errors is considered (Figure 9), then the RS decoder achieves significantly higher error correction performance than the other algorithms. It is important to emphasize the fact that the LDPC and Viterbi decoders use hard-decoded data input, and it is an untypical way of using the codes. In most applications, Viterbi and LDPC decoders use soft decision decoded bit values.

The goodput to BER notation shown in Figures 8 and 9 is not suited for coding papers, but for practical implementations allows estimating the required BER on the PHY layer. For all hardware-in-the-loop experiments, such notation is more preferable, because the overall system performance can be estimated without any additional SNR/EbN0 recalculations (e.g., goodput of the system can be easily estimated when one of the reported RF frontends in Table 1 is considered).

Due to relatively low decoding effort of RS codes, remarkable performance against single errors, and high performance against burst errors, RS codes are selected for the targeted FPGA/ASIC implementation. The performance against burst errors is required, because a single ripple on the RF-frontend or baseband power supply may disturb the decoding of tens consecutive bits (transmission of a single bit at 100 Gbps takes 10 ps). Additionally, data bits are packed into 60-bit symbols in the considered baseband [4]. If a single symbol decoding fails due to noise or synchronization, then a burst error up to 60 bits is expected on the output of the PSSS baseband.

*3.5. Parallel Processing Array of Interleaved RS Decoders.* The presented FPGA design cannot run at faster frequency than ~250 MHz on a Virtex7 FPGA. Thus, throughput of a single RS entity is limited to ~2 Gbps. It means that at least 50 parallel decoders are required to achieve the targeted 100 Gbps data rate. Therefore, an interleaved calculation array of RS decoders has to be employed. We have already published a dedicated article about Interleaved RS for 100 Gbps applications in [29]. In this paper we introduce the most important aspects of the implemented IRS FEC engine.

IRS architecture has two advantages. Firstly, robustness against long-burst errors is improved (Figure 10). Secondly, throughput of the coder is multiplied, so the Interleaved RS array can be easily scaled for 100 Gbps operations.

Due to the hardware issues of Virtex7 FPGA, there are advantages to keep the internal processing data buses 64-bit wide. The main reason of it is hardware multiplexing supported by the FPGA hardware. The multiplexers located at the input stage of the FPGA logic deserializes the input data to 64 bits in most cases (e.g., 10 G Ethernet [30] and GTH

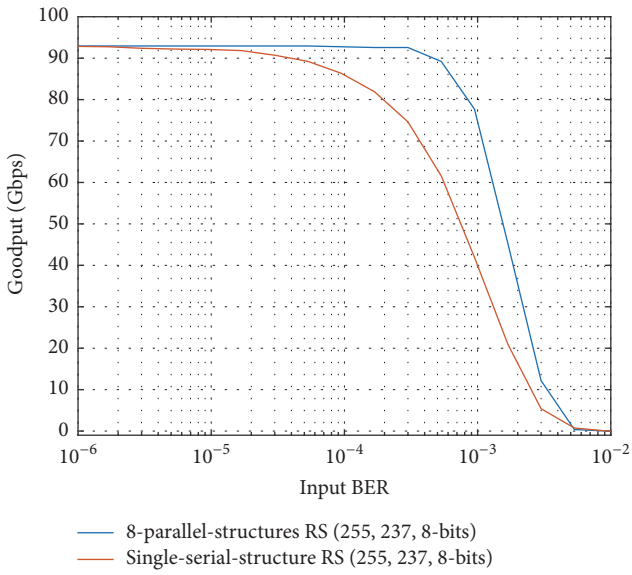


Figure 10: Comparison of burst error correction performance obtained by a single RS decoder and an array of Interleaved RS decoders.

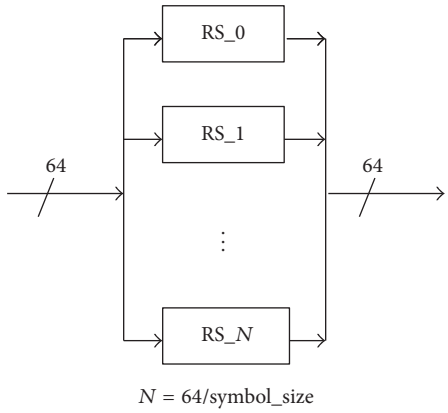


Figure 11: Parallel Reed-Solomon structures used for 100 Gbps IRS coding (optimized for Virtex7 GTX/GTH).

transceivers [31]). Each single RS entity calculates 8 bits from the 64-bit word (Figure 11). In the FPGA, ten such structures are employed to support the required 100 Gbps. Therefore, the FPGA calculates $10 \times 64$ bits in each clock cycle.

*3.6. Link Adaptation.* It is possible to design an algorithm that finds a trade-off between the coding overhead and the demanded error correction performance. The algorithm analyses the number of successfully delivered data fragments and the number of corrected errors in the fragments. When the goodput is degraded by losses of data, then the algorithm increases the FEC coding. It is important to define thresholds, when the FEC modes should be changed. One of possible solutions is setting the thresholds to the code rates of the employed codes. For example, when RS (255, 239) and RS (255, 223) are used, then the thresholds can be set to $239/255 \approx 93.7\%$ and to $223/255 \approx 87.5\%$. If the percentage of successfully

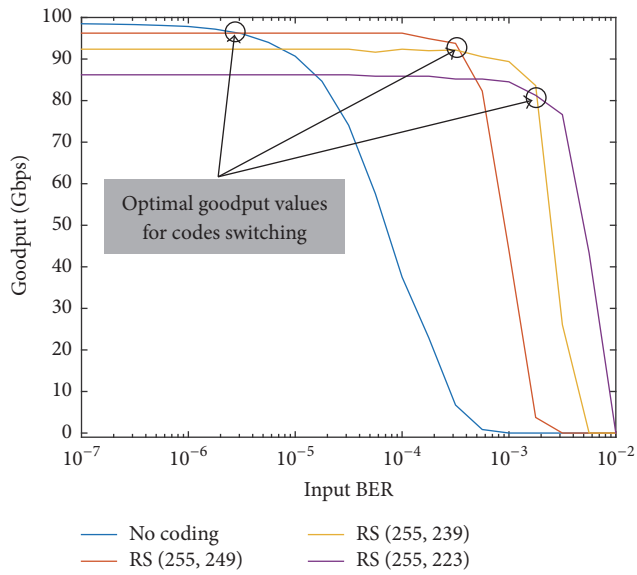


Figure 12: Theory of operation of the link adaptation algorithm. Intersection points of the curves define goodput values when a code switch should be done.

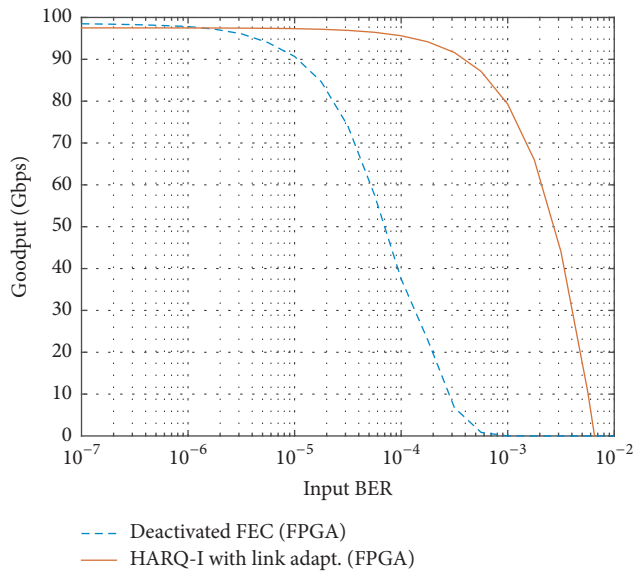


Figure 13: Goodput obtained by employing HARQ-I with link adaptation (denoted as red). The coding is adjusted as a function of estimated BER.

delivered data fragments is below the given values, then a code with a lower code rate is applied. In short, the approach finds a compromise between RS overhead and the rate of the lost data fragments. The thresholds correspond to the code rates and define the upper boundaries of achievable goodput for the codes. Figure 12 explains the theory of operation of the algorithm. Figure 13 shows the results of the proposed approach (the FPGA adjusts the redundancy in a range of 2–18 symbols per RS block). More details about the implemented link adaptation can be found in our articles in [21, 32].

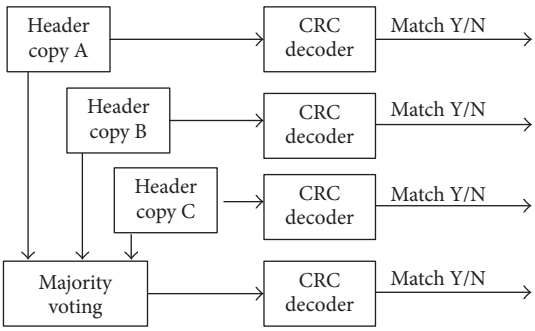


Figure 14: Triple-modular redundancy decoding circuit (source: [19]).

*3.7. FEC Decoder for Frame Headers.* When fragmentation and aggregation are in use, the frame header is the most important part of the frame. Any single error that occurs in the fragmented payload reduces the goodput by $1/n$, where "$n$" is the number of aggregated fragments. The situation deteriorates greatly when a bit error occurs in the frame header. The header contains necessary information for frame decoding, for example, the length and FEC encoding method. Without this information, the frame cannot be decoded and all "$n$" payload fragments are lost. Additionally, complexity of the frame-decoding pipeline is lower when the header is decoded immediately (without a delay). For this purpose, an independent triple-modular redundancy decoder is proposed. A hardware implementation of the method decodes the header in time shorter than 5 ns. It means that information from the header can be used immediately and frame buffering is avoided. This is not possible when the header is encoded with a RS, BCH, or convolutional code.

The triple-modular redundancy encoder sends the header three times, and the decoder checks the CRCs of the copies. If all copies have bit errors, then majority voting is performed on the copies, and the output of the voting is additionally checked. If at least one variant of four is correct, then the header is decoded successfully (Figure 14).

The code rate of the presented method is equal to $R = 1/3$. In the considered case, the header is 20 bytes long and the encoded header uses 40 bytes of redundancy. The assumed frame length is not shorter than 64 kB, so the header overhead is lower than 0.61‰ of the total frame length. Therefore, the coding overhead can be neglected.

The introduced triple-modular routines are uncomplicated but error correction performance is poor. In Figure 15, the algorithm is compared to the other previously mentioned FEC schemes (BCH, HD-LDPC, RS, and HD-Convolutional). Convolutional coding with hard symbol representation achieves approx. 8 dB better results at the same code rate. When the gain is compared to RS (255, 223), then the results are more optimistic. If the header is encoded together with the frame data, then the triple redundancy is at least 2 dB better than the RS (denoted as "*RS (255, 223)*" in Figure 15). Moreover, hardware implementation of the modular redundancy requires only 1 FPGA clock cycle (<5 ns) to provide decoding results. The RS needs ~2000 ns

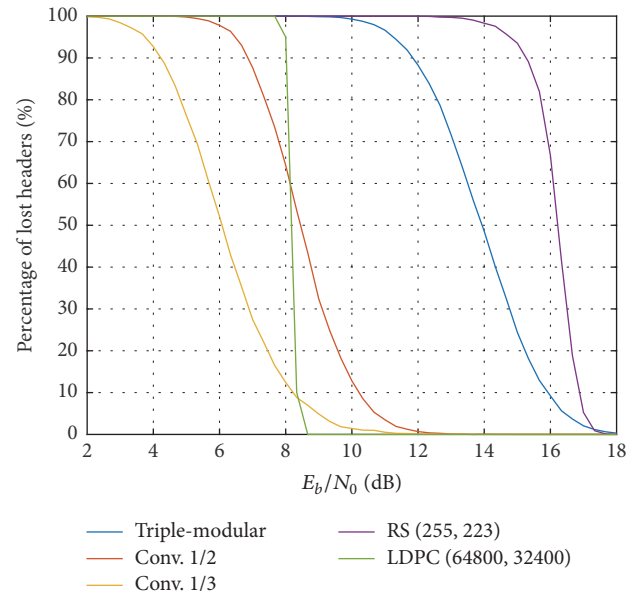


Figure 15: Comparison of error correction performance for different header coding schemes.

in the considered IHP implementation [20] and ~3700 ns in the Xilinx version of the RS IPcore [24]. The Viterbi decoder provided by Xilinx requires at least ~400 ns to perform the same task [8].

*3.8. ACK Frame Length and ACK Encoding.* If the transmitter and receiver do not exchange information about the lost data, then some fragments might be never delivered, and user payload may lack consistency. To avoid such situations, ACK frames are sent after an arbitrary defined number of data frames. When the ACK frame is lost, then the transmitter sends the ACK-request frame after a predefined timeout. Additionally, the ACK transmission requires switching the RF frontends two times (from RX to TX and from TX to RX). At this time and during the timeout duration, the user data is not exchanged. Thus, it has a serious impact on the goodput. To avoid the looseness of the ACKs and to reduce the number of timeouts, the ACKs are encoded with the strongest FEC available in the system (RS 255, 237 in the FPGA implementation). That significantly improves ACK robustness. The overhead induced by the coding is relatively low, because ACK frame is short (~1 kB) and is sent infrequently (not often than every 4 MB of user payload). The ACK frame length depends on the number of successfully received fragments in a single ARQ session (positive acknowledgment). When the fragmentation threshold is lowered, then many small parts have to be sent and acknowledged. Thus, compression methods can be used to reduce the length of the ACK frames. Figure 16 compares three approaches: a basic bit map coding and two versions of sequence coding. A single value and a bit map are sent in the state-of-the-art bit map scheme. The first value defines the first acknowledged fragment number, and the bit map determines all next values. The bit position specifies an offset, and the bit value

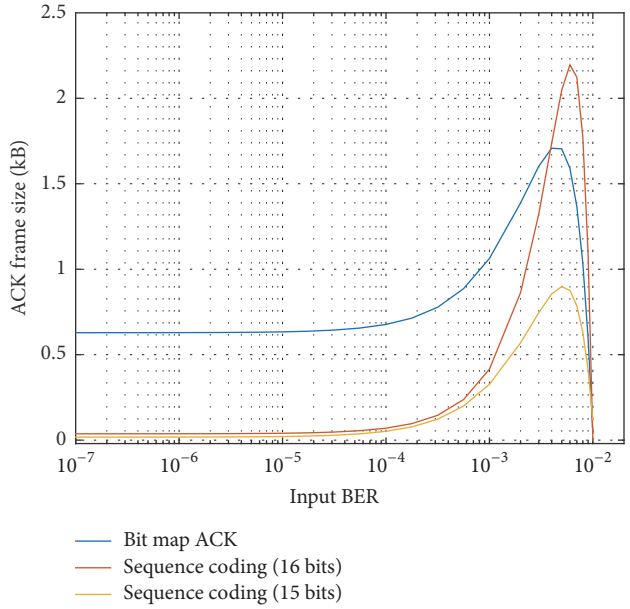


Figure 16: Maximal ACK frame length as a function of the channel BER.

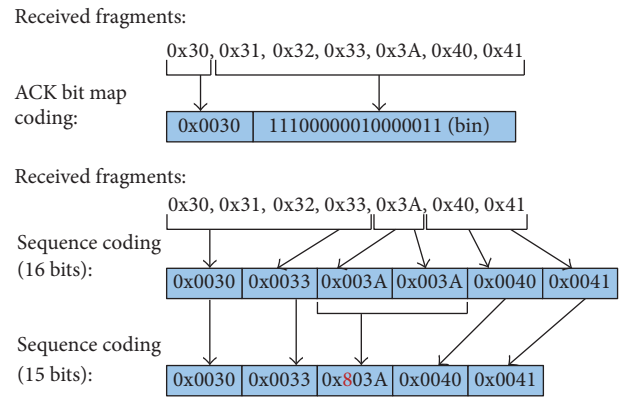


Figure 17: Alternative compressing techniques proposed for an ACK frame.

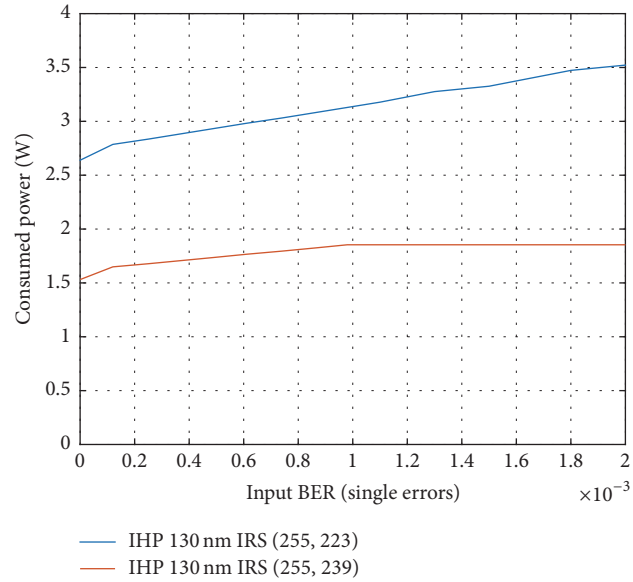


Figure 18: Power consumption as a function of the channel BER.

defines whether the fragment is acknowledged or not. The second and third methods send only a sequence of addresses of successfully received fragments. All three methods are implemented in FPGA/ASIC, and the ACK frame can be generated and compressed on the fly by FPGA/ASIC logic. Such approach reduces ACK processing latency. Figure 16 compares the maximal ACK frame length obtained by the proposed techniques as a function of the channel BER. The proposed method with 15-bit coding significantly reduces the ACK size (<1 kB) under all BER conditions. This additionally improves robustness of the return channel.

The sequence coding with 16 bits writes to memory two uint16 values that define the first and the last sequence number of successfully received fragments in a consecutive subsequence. Such coding is very effective if all or almost all frame-fragments are successfully decoded (transmission at BER $< 1e-4$ according to Figure 16). On the other hand, encoding of single fragments in a long sequence of faulty fragments costs 32 bits: two uint16 values (at BER $> 1e-3$ according to Figure 16). To overcome this problem, a modified coding is proposed. The modified solution uses 15 lower bits for a sequence number, and the most significant bit is reserved to indicate single values. All three methods are compared in Figure 17.

## 4. IHP 130 nm ASIC Synthesis

After successful FPGA validation, the design has been synthesized into IHP 130 nm CMOS technology. The FPGA implemented FEC engine supports RS (255, 237) and currently we cannot support more robust coding in the FPGA. Although changing the technology to 130 nm ASIC allows enabling RS (255, 223) FEC in the ASIC design, this significantly improves error correction performance and the engine can work with BER $\approx 1e-2$ (burst errors). The RS coders consume approx. 88% of the chip resources, and the complete design requires silicon area of 30.64 mm$^2$. The computation effort depends on the number of defected symbols in a data block. Therefore, the consumed power and dissipated energy of the processor are a function of the channel BER (Figure 18). The synthesis tool reports the maximal operational clock up to 210 MHz.

The synthesis and ASIC power profiling tools allow estimating energy per bit for intermediate codes selected by the link adaptation algorithm (Figure 19). Frame processing features (deaggregation, CRC, FSMs, ACK processing, etc.) consume relatively small amount of energy (1.39 pJ/bit) in comparison to the FEC decoding (up to ~41 pJ/bit).

*4.1. $E_b/N_0$ at Physical Layer and Energy Required for Forward Error Correction.* The targeted baseband [4] uses parallel spread spectrum sequences (PSSS) with PAM-16 modulation (4 bits/Hz) and channel deconvolution. PAM-16 modulation

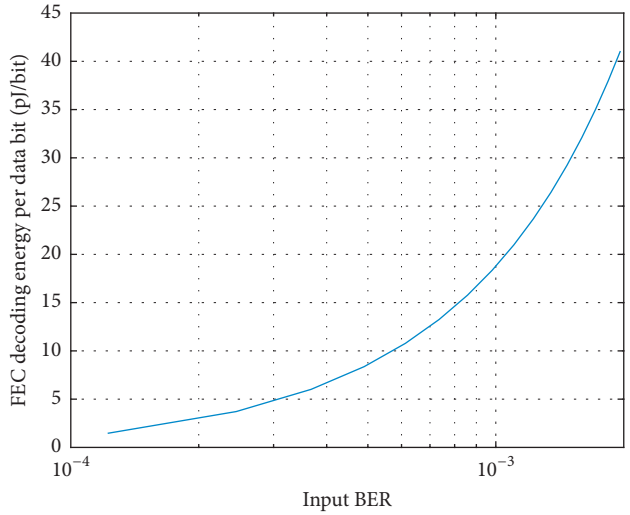


FIGURE 19: Required energy per processed data bit for RS decoding with enabled link adaptation.

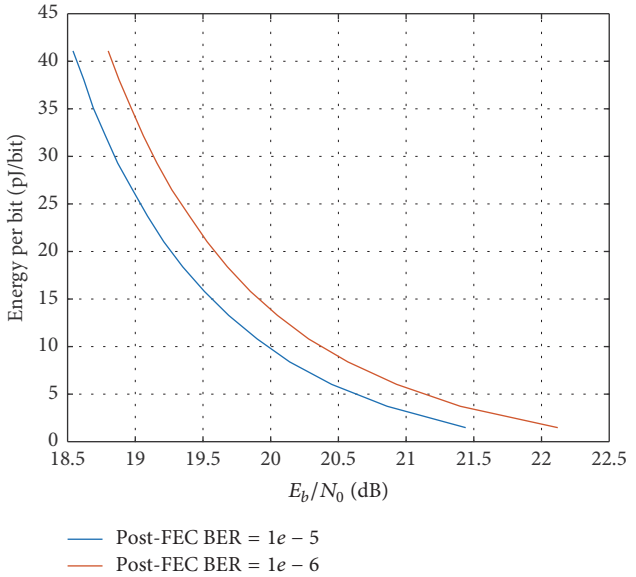


FIGURE 20: Energy per data bit required by the FEC processor to support the required post-FEC BER: $1e-5$ denoted in blue; $1e-6$ denoted in red.

is selected due to relatively uncomplicated hardware required for the PSSS processing in an analog baseband [4, 6, 33]. The PSSS and channel deconvolution provide some processing gain. This gain additionally improves transmission quality. Investigation of the PSSS and channel deconvolution is beyond the scope of this work. Thus, in this paragraph a simplified baseband and RF-frontend without the PSSS and channel deconvolution but with PAM-16 modulation are considered. As a result, required $E_b/N_0$ for the presented datalink layer processor is estimated. Due to the aggregation and fragmentation schemes, the DLL operates with deactivated FEC at BER $\approx 1e-6$ with insignificant goodput degradation (goodput of 97.84% according to Figure 13). Improving the link quality above this value using FEC or increasing transmission power does not improve the goodput significantly but may lead to energy waste. Thus, from an energy efficiency point of view, the operational BER $\approx 1e-6$ is recommended to obtain reasonable goodput. At BER $\approx 1e-5$, the accelerator still operates and achieves goodput of 90.68%. However, BER values higher than $\sim 2e-6$ are not recommended, and in such case, the FEC effort should be increased to reduce the number of retransmitted frames. The required $E_b/N_0$ values for post-FEC BER $= 1e-6$ and post-FEC BER $= 1e-5$ are presented in Figure 20. The measurements shown in Figure 20 correspond to the goodput characteristics shown in Figure 21.

The goodput (Figure 21) decreases with the decrease of the $E_b/N_0$, even if the post-FEC BER values are constant ($1e-6$ and $1e-5$). This is caused by the link adaptation approach but not by retransmissions (ARQ). The processor increases the number of redundancy bits with the decrease of $E_b/N_0$. Thus, the average amount of user data in a frame decreases, so the goodput decreases as well. The presented $E_b/N_0$ values correspond to the worst case without the processing gain of the PSSS and channel deconvolution.

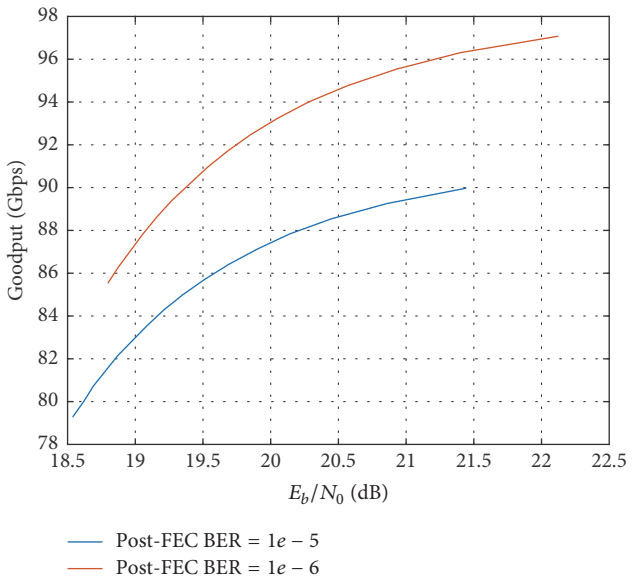


FIGURE 21: Estimated goodput for the selected post-FEC BER values: $1e-5$ denoted in blue; $1e-6$ denoted in red.

## 5. Conclusion

In our article, we define some challenges for 100 Gbps data link layer processor. Later, some algorithms were considered and required simplifications for FPGA/ASIC processing are introduced. This allows implementing and validating the proposed system on a standard FPGA platform. The basic HARQ type-I method with link adaptation can be used instead of more complex HARQ schemes. Efficiency degradation is relatively insignificant, and the type-I method does not require cache memory. This simplifies the design and removes one of the hardware boundaries. Additionally,

we demonstrate FEC engine that works with ~117 Gbps on a single Virtex7 FPGA. Fragmentation, aggregation, and ACK compression additionally improve resistance against bit errors. The triple-modular redundancy decoder simplifies the processing pipeline and reduces header decoding time to few nanoseconds. At the end, consumed energy per bit for the design synthesized into 130 nm IHP CMOS technology is presented.

## Competing Interests

The authors declare that they have no competing interests.

## References


[1] E. Perahia and M. X. Gong, "Gigabit wireless LANs," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 15, no. 3, p. 23, 2011.

[2] "DFG SPP1655," 2016, http://www.wireless100gb.de/.

[3] "Real100G.COM," http://www.wireless100gb.de/project_9_en.html.

[4] T. Messinger, K. KrishneGowda, F. Boes et al., "Multi-level 20 Gbit/s PSSS transmission using a linearity-limited 240 GHz wireless frontend," in *Proceedings of the IEEE International Conference on Microwaves, Communications, Antennas and Electronic Systems (COMCAS '15)*, pp. 1–3, Tel-Aviv, Israel, November 2015.

[5] A. R. Javed, J. C. Scheytt, K. KrishneGowda, and R. Kraemer, "System design considerations for a PSSS transceiver for 100Gbps wireless communication with emphasis on mixed signal implementation," in *Proceedings of the 16th IEEE Annual Wireless and Microwave Technology Conference (WAMICON '15)*, Cocoa Beach, Fla, USA, April 2015.

[6] K. KrishneGowda, A. Wolf, R. Kraemer, J. C. Scheytt, and I. Kallfass, "Wireless 100 Gb/s: PHY layer overview and challenges in the THz freqency band," in *Proceedings of the IEEE 15th Annual Wireless and Microwave Technology Conference (WAMICON '14)*, Tampa, Fla, USA, June 2014.

[7] N. Akkari, J. M. Jornet, P. Wang et al., "Joint physical and link layer error control analysis for nanonetworks in the Terahertz band," *Wireless Networks*, vol. 22, no. 4, pp. 1221–1233, 2016.

[8] Xilinx Corp, "LogiCORE IP Viterbi Decoder v9.0—Product Guide," *Xilinx Datasheet*, 2014.

[9] C. Hermsmeyer, H. Song, R. Schlenk, R. Gemelli, and S. Bunse, "Towards 100G packet processing: challenges and technologies," *Bell Labs Technical Journal*, vol. 14, no. 2, pp. 57–80, 2009.

[10] G. Ducournau, P. Szriftgiser, A. Beck et al., "Ultrawide-bandwidth single-channel 0.4-THz wireless link combining broadband quasi-optic photomixer and coherent detection," *IEEE Transactions on Terahertz Science and Technology*, vol. 4, no. 3, pp. 328–337, 2014.

[11] A. Hirata, T. Kosugi, H. Takahashi et al., "120-GHz-band millimeter-wave photonic wireless link for 10-Gb/s data transmission," *IEEE Transactions on Microwave Theory and Techniques*, vol. 54, no. 5, pp. 1937–1942, 2006.

[12] X. Pang, A. Caballero, A. Dogadaev et al., "100 Gbit/s hybrid optical fiber-wireless link in the W-band (75-110 GHz)," *Optics Express*, vol. 19, no. 25, pp. 24944–24949, 2011.

[13] J. Antes, S. Konig, A. Leuther et al., "220 GHz wireless data transmission experiments up to 30 Gbit/s," in *Proceedings of the IEEE MTT-S International Microwave Symposium (IMS '12)*, pp. 1–3, June 2012.

[14] T. Nagatsuma, S. Horiguchi, Y. Minamikata et al., "Terahertz wireless communications based on photonics technologies," *Optics Express*, vol. 21, no. 20, pp. 23736–23747, 2013.

[15] S. Koenig, D. Lopez-Diaz, J. Antes et al., "Wireless sub-THz communication system with high data rate," *Nature Photonics*, vol. 7, no. 12, pp. 977–981, 2013.

[16] X. Li, J. Yu, J. Zhang, Z. Dong, F. Li, and N. Chi, "A 400G optical wireless integration delivery system," *Optics Express*, vol. 21, no. 16, pp. 18812–18819, 2013.

[17] Altera, *40- and 100-Gbps Ethernet MAC and PHY MegaCore Function User Guide*, Altera, San Jose, Calif, USA, 2014.

[18] Y. Tian, K. Xu, and N. Ansari, "TCP in wireless environments: problems and solutions," *IEEE Communications Magazine*, vol. 43, no. 3, pp. S27–S32, 2005.

[19] L. Lopacinski, M. Brzozowski, R. Kraemer, S. Buechner, and J. Nolte, "100 Gbps wireless—data link layer VHDL implementation," in *Proceedings of the 18th Conference on Reconfigurable Ubiquitous Computing (RUC '15)*, Szczecin, Poland, September 2015.

[20] M. Marinkovic, M. Piz, C.-S. Choi, G. Panic, M. Ehrig, and E. Grass, "Performance evaluation of channel coding for Gbps 60-GHz OFDM-based wireless communications," in *Proceedings of the IEEE 21st International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC '10)*, pp. 994–998, September 2010.

[21] L. Lopacinski, M. Brzozowski, and R. Kraemer, "A 100Gbps data link layer with an adaptive algorithm for forward error correction," in *Proceedings of the IEICE Information and Communication Technology Forum (ICTF '15)*, 2015.

[22] E. Modiano, "An adaptive algorithm for optimizing the packet size used in wireless ARQ protocols," *Wireless Networks*, vol. 5, no. 4, pp. 279–286, 1999.

[23] M. Ehrig and M. Petri, "60 GHz broadband MAC system design for cable replacement in machine vision applications," *AEU—International Journal of Electronics and Communications*, vol. 67, no. 12, pp. 1118–1128, 2013.

[24] Xilinx, *LogiCORE™ IP Reed-Solomon Encoder Product Guide*, Xilinx, 2015.

[25] H. M. Ji, "An optimized processor for fast Reed-Solomon encoding and decoding," in *Proceedings of the IEEE International Conference on Acoustic, Speech, and Signal Processing*, pp. III-3097–III-3100, May 2002.

[26] R. S. Lim, "A decoding procedure for the Reed-Solomon codes," NASA Technical Paper 1286, 1978.

[27] D. V. Sarwate and N. R. Shanbhag, "High-speed architectures for Reed-Solomon decoders," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 9, no. 5, pp. 641–655, 2001.

[28] Creonic, "Viterbi Decoder User Guide," pp. 1–26, 2012, https://www.creonic.com/images/pdf/UG_creonic_viterbi_decoder.pdf.

[29] L. Lopacinski, S. Buechner, and R. Kraemer, "Parallel RS error correction structures dedicated for 100 Gbps wireless data link layer," in *Proceedings of the IEEE International Conference on Ubiquitous Wireless Broadband (ICUWB '15)*, Montreal, Canada, October 2015.

[30] Xilinx Corp, "10G Ethernet PCS/PMA v6.0," *Xilinx Datasheet*, 2015.

[31] Xilinx Corporation, "7 Series FPGAs GTX/GTH Transceivers," Xilinx Datasheet. 2012.

[32] L. Lopacinski, J. Nolte, S. Buechner, M. Brzozowski, and R. Kraemer, "Design and implementation of an adaptive algorithm for hybrid automatic repeat request," in *Proceedings of the 18th IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS '15)*, pp. 263–266, April 2015.

[33] K. KrishneGowda, A. C. Wolf, R. Kraemer, T. Messinger, and I. Kallfass, "Simulation of 100 Gbps using Parallel Sequence Spread Spectrum modulation (PSSS) with 240 GHz radio," in *Proceedings of the 1st URSI Atlantic Radio Science Conference (URSI AT-RASC '15)*, Las Palmas, Spain, May 2015.