Lukasz LOPACINSKI [1], Joerg NOLTE [1], Steffen BUECHNER [1],
Marcin BRZOZOWSKI [2], Rolf KRAEMER [2]
[1] BRANDENBURG UNIVERSITY OF TECHNOLOGY, Cottbus, Germany
[2] IHP, Frankfurt Oder, Germany

100 Gbps wireless – data link layer VHDL implementation

Abstract

In this paper, we describe implementation and hardware used for a wireless 100 Gbps data link layer demonstrator. So fast stream processing requires a highly parallelized approach. The timing requirements of the 100 Gbps networks are so demanding that there is no chance to deal with this task as a single stream in a field programmable gate array (FPGA). Due to this reason, we introduce and validate one of possible architectures that can solve the task. The 100 Gbps implementation is explained in detail, and the most important parameters of the FPGA design are mentioned.

Keywords: data link layer, wireless, 100 Gbps, FPGA.

1. Introduction

In the next few years, we can expect radio transceivers operating at 100 Gbps. So fast data streams require not only an ultra-fast physical layer (PHY), but also other parts of the system have to be ready to process so fast streams. The scientists from the University of Stuttgart have already made some experiments with 96 Gbps transmission at 240 GHz [1]. We are also collaborating with scientists from the Real100G project at University Cottbus, where some work about the mixed analog-digital baseband is in progress [2]. We would like to prepare a data link layer processor, which can be connected to the previously mentioned work.

Firstly, we have to find a hardware which can process a 100 Gbps stream. In this publication, we focus on the hardware and implementation issues.

Most of the algorithms used in the project have to be supported by hardware accelerators, or have to be implemented fully in a hardware. We decided to use a traditional FPGA technology.

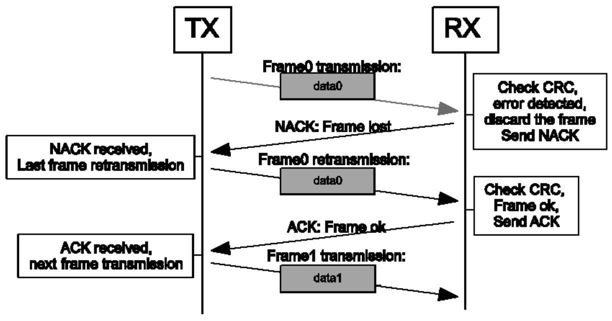

TX
RX
Frame0 transmission:
data0
Check CRC, error detected, discard the frame Send NACK
NACK: Frame lost
NACK received, Last frame retransmission
Frame0 retransmission:
data0
Check CRC, Frame ok, Send ACK
ACK: Frame ok
ACK received, next frame transmission
Frame1 transmission:
data1

Fig. 1. Example of the ARQ scheme

2. Data link layer techniques

Automatic repeat request (ARQ) [3, 4] is responsible for reliable data transmission. A transmitter sends frames and a receiver confirms all successful transmissions. If any frame was not delivered due to any reason, then it is retransmitted (Fig. 1).

A HARQ (hybrid automatic repeat request) is a combined technique of forward error correction (FEC) and ARQ. The FEC algorithms correct errors in the received frames. The FEC requires some pre-calculated redundancy data added to the payload. The redundancy information is used to locate the bit errors. If there is no possibility to correct all errors due to the limited redundancy data, then the frame is retransmitted by the ARQ.

3. Related work

In our End2End100 project, we concentrate on robustness of the link. The main tasks are the ARQ and the FEC. We have published about difficulties and possible hardware architectures for these tasks [5]. The error correction results of a selected FEC are discussed in [6].

A similar system to ours is described in [7]. The authors present the results and hardware issues of the 60 GHz OFDM transceiver. They describe how the dedicated hardware of the Virtex5 FPGA can be used to implement the demonstrator.

We use a Virtex7 FPGA and our VHDL implementation can process 97 Gbps of continuous user data (a single FPGA chip). In this paper, we would like to discuss some hardware aspects from our point of view.

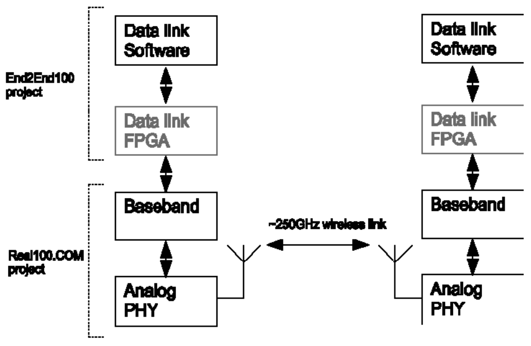


Fig. 2. The demonstrator architecture for 100 Gbps wireless transmission. The data link layer FPGA is marked in red

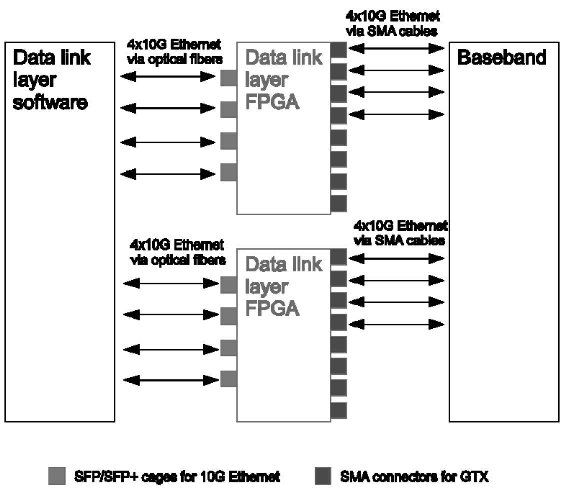

Data link layer software
4x10G Ethernet via optical fibers
Data link layer FPGA
4x10G Ethernet via SMA cables
Baseband
4x10G Ethernet via optical fibers
Data link layer FPGA
4x10G Ethernet via SMA cables
SFP/SFP+ cages for 10G Ethernet
SMA connectors for GTX

Fig. 3. The architecture supporting 80 Gbps. The limit of 80 Gbps is defined by the software processor – only eight Ethernet ports are available

4. Interfaces

Our End2End-group is responsible for preparation of the data link layer FPGA (marked in red in Fig. 2). The analog and baseband processing is a domain of the Real100.COM project.

The data link layer FPGA has to use compatible interfaces to the baseband and to the software processors.

The chosen VC709 development kit has very limited connectivity. Only four 10GBASE-R Ethernet ports are available. Additionally, the kit has the possibility to use FMC-HPC extension cards. On the card, four additional 10GBASE-R Ethernets or eight GTH serial transceivers with SMA connectors can be installed. We decided to use the embedded Ethernet ports to connect to the MAC software processor. The SMA ports, located on the FMC daughter card, will be used to connect to the baseband (Fig. 3). We can read and write up to 40 Gbps from/to the FPGA. This is limited by the interfaces. To transfer 100 Gbps, we have to use at least three FPGA kits and the data link VHDL implementation has to be split.

Unfortunately, the data link software processor (Tilera TILEncore-Gx72) supports only eight 10G Ethernet ports, thus we are constrained to 80 Gbps (Fig. 3). We will try to resolve this problem later.

Generally, the Virtex7 XC7VX690T FPGA chip supports up to 56 GTH channels [8], but only a few of them are provided on the VC709 board [9]. We cannot develop a PCB dedicated for us, due to the cost.

5. Lane approach

Our implementation uses parallel lanes to deal with the 100 Gbps stream. The FPGA cannot handle the stream at once. Thus, we split the data to parallel lanes. A similar lane approach for a PHY layer is used in 40 and 100 Gbps Ethernet [10].

A single lane architecture is shown in Fig 4. We use two clock domains for the single lane (six independent clock domains in the whole FPGA). The line rate of the Ethernet is 10.3125 Gbps, and requires a 156.25 MHz reference clock. All signals going from and to the core are synchronized with this clock. The FEC parts (RS coders) use a 200 MHz clock. The clock difference compensates overhead cycles of the FEC data processing. By using a faster clock on the coder and decoder part, we support a continuous transfer of 10 Gbps on every lane. In such a situation, the RS decoders achieve the peak throughput of 12.8 Gbps per lane. This is not a constant value, because we use padding at the end of the Reed-Solomon blocks. During the padding, the user data is not coded, and these cycles are wasted.

The design cannot be clocked faster than 220 MHz. With this value, we get setup violations in the registers responsible for the RS decoding state machine.

A single RS coder supports the maximum throughput of ~1.6 Gbps in our case. Due to this reason, we mix the data between eight instances of the RS (8 × 1.6 Gbps = 12.8 Gbps). Additionally, this solution improves the burst error correction because the longest burst errors are split to different decoders.

Currently, the RS coders are based on an eight-bit symbol and support the following coding formats: 237, 239, 241, 243, 245, 247, 249, 251, and 253. It is flexible implementation, and allows us to implement an adaptive hybrid automatic repeat request (an adaptive HARQ). More details can be found in [3], [5].

6. FEC decoding throughput and area usage

We researched some available FEC implementations (Table 1). The clock speed and the throughput are estimated when a single instance is implemented in the FPGA. We use the XC7VX690T FPGA and this device supports 433k LUTs. It can be observed that the RS coding consumes fewer resources, and achieves a higher throughput than the convolutional coding. This is the main reason to use the RS algorithm in our implementation. We compare the error correcting capabilities of the RS and Viterbi decoder in [6].

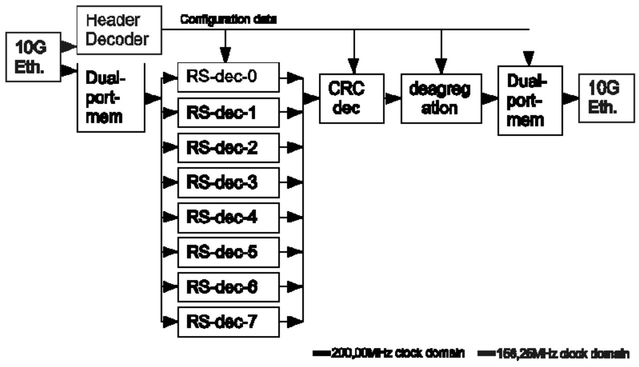


Fig. 4. The single FPGA decoder lane architecture

Tab. 1. Comparison of FEC implementations

Implementation	*Ref.*	*Throughput [Gbps]*	*LUT area*	*FPGA / speed grade*
Xilinx Viterbi	[11]	0.28	2.5 k	Virtex7/-1
Xilinx Viterbi	[11]	0.4	2.5 k	Virtex7/-3
Creonic Viterbi	[12]	0.25	3.0 k	Virtex6/-1
Creonic Viterbi	[12]	0.14	3.0 k	Spartan6/-2
IHP Viterbi	[13]	0.17	12 k	Virtex7/-2
Xilinx RS Decoder	[14]	2.2	0.8 k	Virtex7/-1
Xilinx RS Decoder	[14]	3.0	0.8 k	Virtex7/-3
IHP RS Decoder	[13]	2.0	2.7 k	Virtex7/-2
IHP RS Encoder	[13]	3.2	0.2 k	Virtex7/-2
Xilinx RS Encoder	[15]	2.9	0.26 k	Virtex7/-1
Xilinx RS Encoder	[15]	4.4	0.26 k	Virtex7/-3

7. RS decoding delay

The Xilinx RS decoder based on the 8-bit symbol introduces an extensive delay for the data processing (Fig. 5). The delay depends on the number of redundancy symbols. If more than 18 symbols are required for the redundancy, then the decoder cannot support a continuous stream. It requires some idle cycles for the calculation, and no data can be provided to the core. During this time, the throughput is radically decreased. Fig. 6 presents the overview of the Xilinx RS decoder throughput running at 200 MHz.

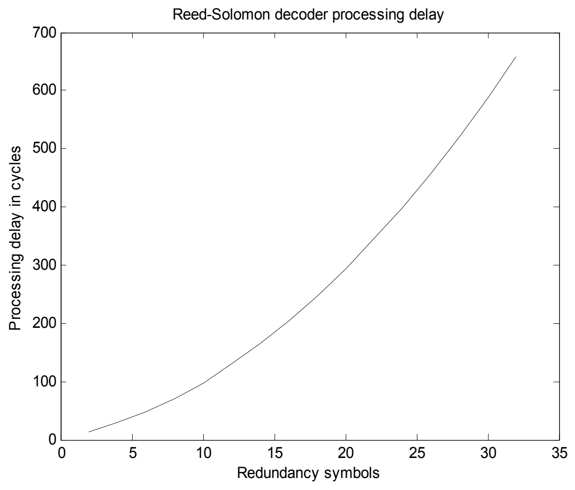


Fig. 5. Latency of the Xilinx RS decoder

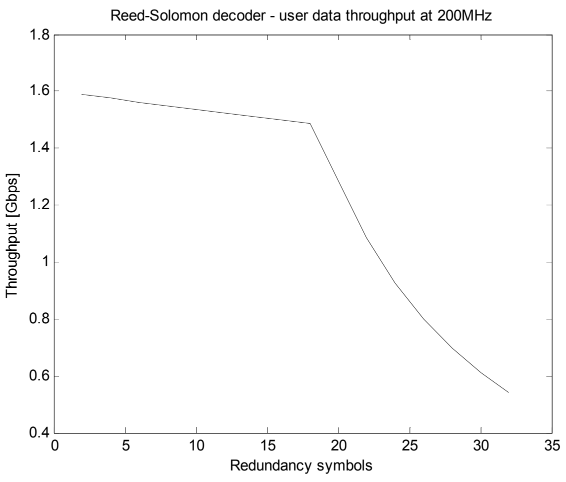


Fig. 6. User data throughput of the Xilinx RS decoder

8. LDPC coding

The IHP institute owns the implementation of LDPC codes and the IHP publishes the results of this solution [16]. It uses soft coding and generally consumes much more resources than the RS. The LDPC implementation synthesized in 40 nm Infineon technology consumes almost 2.5 times more silicon area and achieves the 5 times lower data throughput than the IHP-RS(255,223) decoder [16]. There is available the paper [17] in which a new approach for the LDPC coding is presented (loops unrolling). The solution realized in a 65 nm ASIC (12 mm2) achieves up to 160 Gbps.

9. Triple modular redundancy

The header decoding performance is very important for our architecture. First of all, it synchronizes the transmit (TX) and receive (RX) FPGAs. The state machines on both sides use some dedicated bits in the header. When these bits are lost during the transmission, then the FPGAs are desynchronized. Both sides wait for a timeout to restart the transmission. That leads to efficiency degradation. Due to this reason, we use more robust coding for the headers. Our first idea was to use the RS(255,223) with 32 redundancy symbols (255-223=32). Unfortunately, the RS decoding introduces delays. The Xilinx decoding implementation requires up to 700 cycles to calculate a single RS(255,223) block. This adds idle cycles to the decoding pipeline. Such a situation dramatically reduces the efficiency and the throughput. To resolve this problem, we introduced a triple redundancy module. In our implementation, it decodes the header in a single clock cycle. Thus, there are no idle cycles in the pipeline. Additionally, the error correction performance is on a satisfied level. We published the measured results in [6]. In this paper, we will introduce the hardware structure of it (Fig. 7). The transmitter sends the header three times. The receiver tries to recalculate the CRC of every copy. Simultaneously, the output of a majority voting circuit is tested. If any of four branches confirms that the data is correct, then the header is successfully decoded.

10. FPGA demonstrator

We measured the throughput achieved on two FPGA boards connected back-to-back with 10 lanes (4 × Optical Fiber Ethernets + 6 × SMA cables; the SMA cables are not supported by the Tilera software processor, and internal FPGA frames generators were used). Such a demonstrator achieves 97 Gbps (continuous transfer). In Fig. 8, we present the experimental setup. In such a configuration, we use 279429 six-input-LUTs and 234038 FFs. It is respectively 65% and 27% of the total resources available in the Virtex7 xc7vx690t device. The slices occupation is equal to 79%.

The error correction results of the experimental demonstrator are shown in Fig. 9. We plotted our adaptive solution denoted as RS-IHP-BTU with RS(255,253) and RS(255,237). The adaptive solution finds the compromise between the overhead and bit error rate (BER). That maximizes the user data throughput and takes an advantage of all supported RS codes. Theoretically, it should apply an optimal RS setting for all tested BER values. The implementation reduces the coding on “good” links and increases on “bad” links (Fig. 9). The FPGA changes the coding from RS(255,253) to RS(255,237) and uses intermediate codes in between. The algorithm is described in [18].

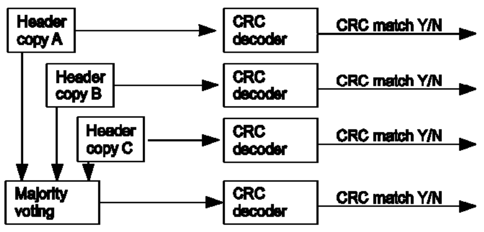


Fig. 7. The header decoder (triple modular redundancy)

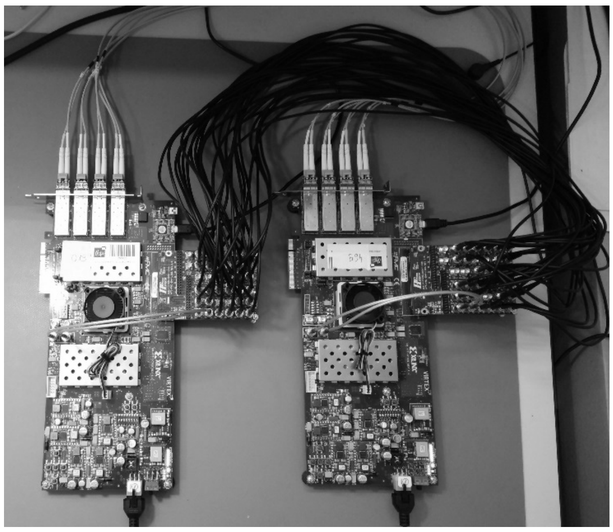


Fig. 8. Experimental setup of the 97 Gbps demonstrator

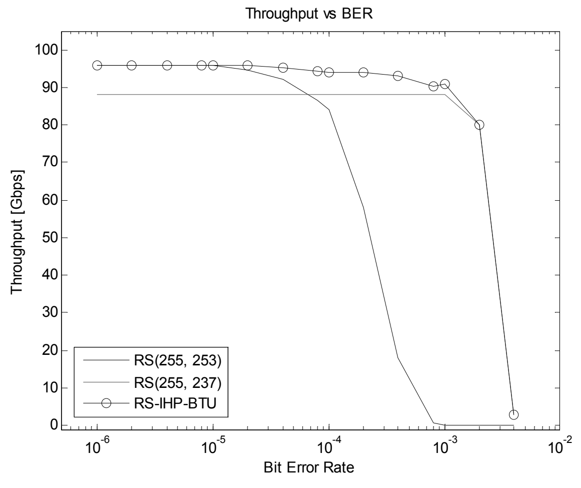


Fig. 9. Results of the experimental setup

This paper is related to End2End100 project and cooperates with other proposed projects of the DFG SPP1655. This group of projects will investigate a complete wireless 100 Gbps system at ultra-high frequencies (240 GHz).

11. References

[1] Boes, Messinger, Antes, Meier, Tessmann, Inam and Kallfass: Ultra-Broadband MMIC-Based Wireless Link at 240 GHz Enabled by 64 GS/s DAC.

[2] Krishne Gowda, Wolf, Kraemer, Scheytt and Kallfass: Wireless 100 Gb/s: PHY layer overview and challenges in the THz freqency band. In Wireless and Microwave Technology Conference (WAMICON), 2014 IEEE 15th Annual, 2014.

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

[4] Minn, Zeng and Bhargava: On ARQ scheme with adaptive error control. Vehicular Technology, IEEE Transactions on, vol. 50, no. 6, pp. 1426-1436, 2001.

[5] Lopacinski, Brzozowski, Kraemer and Nolte: 100 Gbps Wireless - Challenges to the data link layer. In ICTF 2014, 2014.

[6] Lopacinski, Brzozowski and Kraemer: A 100 Gbps data link layer with a frame segmentation and hybrid automatic repeat request. Science and Information Conference 2015, 2015.

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

[8] 7 Series FPGAs Overview - DS180.

[9] VC709 Evaluation Board for the Virtex-7 FPGA.

[10] 802.3ba media access control parameter, physical layer, and management parameters for 40 Gb/s and 100 Gb/s operation.

[11] LogiCORE IP ViterbiDecoder v8.0.

[12] Viterbi Decoder User Guide V 1.0.0, Jan. 16, 2012.

[13] Marinkovic, Piz, Choi, Panic, Ehrig and Grass: Performance evaluation of channel coding for Gbps 60-GHz OFDM-based wireless communications. In Personal Indoor and Mobile Radio Communications (PIMRC), 2010 IEEE 21st International Symposium, 2010.

[14] LogiCORE IP Reed-Solomon Decoder v8.0.

[15] LogiCORE IP Reed-Solomon Encoder v8.0.

[16] Marinkovic, Krstic, Grass and Piz: Performance and complexity analysis of channel coding schemes for multi-Gbps wireless communications. In Personal Indoor and Mobile Radio Communications (PIMRC), 2012 IEEE 23rd International Symposium, 2012.

[17] Schlaefer and When: A New Dimension of Parallelism in Ultra High Throughput LDPC Decoding. 2013 IEEE Workshop on Signal Processing Systems, 2013.

[18] Lopacinski, Nolte, Buechner, Brzozowski and Kraemer: Design and implementation of an adaptive algorithm for hybrid automatic repeat request. DDECS, 2015.

Received: 16.04.2015 Paper reviewed Accepted: 02.06.2015

Lukasz LOPACINSKI, MSc

He received his MSc degree in computer science from West Pomeranian University of Technology, Szczecin, Poland, in 2009. Since 2007, he worked in industrial companies in field of embedded systems and wireless communication. Currently he is working in BTU Cottbus (Germany).

e-mail: lukasz.lopacinski@b-tu.de

Prof. Jörg NOLTE, PhD, eng.

He is professor for distributed systems and operating systems at the Brandenburg University of Technology in Cottbus (Germany). He received his MS (Dipl. Inform., 1988) and PhD (Dr. Ing., 1994) in computer science from the Technical University of Berlin. He was a principal member and finally the vice-head of the PEACE group at GMD FIRST (Berlin). His major research interests are operating systems, middleware and programming languages for parallel, distributed and embedded systems.

e-mail: Joerg.Nolte@b-tu.de

Steffen BÜCHNER

He received his diploma in computer science from the BTU Cottbus in the year 2011. After that, he participated at several research projects at the Distributed Systems / Operating Systems Group of the BTU Cottbus in the fields wireless sensor networks and embedded systems. Currently he is working on a parallel event stream processing concept for utilizing the processing power of embedded many cores for ultra-high data rate wireless communication protocol handling.

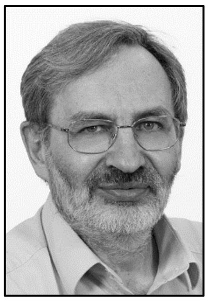
e-mail: Steffen.Buechner@b-tu.de

Marcin BRZOZOWSKI, PhD, eng.

He received his diploma and PhD degree in computer science from BTU Cottbus, Germany, in 2006 and 2012 respectively. Currently he is working with networking and embedded systems in IHP Germany. His research interests include computer networks and operating systems.

e-mail: brzozowski@ihp-microelectronics.com

Prof. Rolf KRAEMER, PhD, eng.



He received his diploma and PhD from RWTH Aachen in electrical engineering and computer-science in 1979 and 1985. He joined the Philips research laboratories in 1985 where he worked in different positions and responsibilities. In 1998 he became professor at the technical university of Cottbus with the joined appointment of the department head of wireless systems at the IHP in Frankfurt (Oder). In the IHP he leads a research department with approximately 70 researchers in topics of high speed wireless communication, context aware middleware, sensor networks.

e-mail: kraemer@ihp-microelectronics.com