Grzegorz DULNIK, **Adam GRZELKA**, Adam ŁUCZAK
CHAIR OF MULTIMEDIA TELECOMMUNICATIONS AND MICROELECTRONICS
Poznań University of Technology
Polanka 3, 60-965 Poznań, POLAND

DOI 10.1000/182

Gigabit Ethernet interface with embedded lossless data encoder on FPGA

Abstract

This paper proposes high speed data encoder, which is dedicated to encode data sent by Gigabit Ethernet interface. The main aim of this encoder is not to achieve high bitrate reduction, but to create the coder, which can be efficiently implemented on FPGA and easily modified according to different content. This encoder can be added to existing systems and greatly reduce the network load by a small change in FPGA project design.

Keywords: FPGA, Encoder, Gigabit Ethernet

1. Introduction

Some applications, especially in the area of advanced video or image processing need to capture data from multimedia devices without any change. Lossy encoded data are useless for testing new encoders or advanced processing like depth estimation. Losslessly encoded data is often captured by dedicated devices - they can collect data from dedicated high speed interfaces (e.g. SDI), or be implemented in device (e.g. read data from image converter). In such solutions, uncompressed data should be stored on hard disc or sent to external data storage.

The quality of multimedia services in digital telecommunication network is increasing. Consumers want to have access to data of high resolution without any distortion like e.g. blocking effect. To ensure this, telecommunication network and digital device must be able to process high throughput. To reduce increasing data size, effective compression methods are commonly in use. Compression increases data throughput however encoders are usually complex and often constructed as dedicated hardware. Different data content requires different coding methods – encoders are not flexible. Dedicated coders have the highest coding efficiency for specific data content and are the best solution when the data content is constant. Almost always coding content are unchanged but some applications, especially in research purposes, need another approach.

Lossy encoders are commonly used to deliver multimedia data. Differences between original and decoded data are insignificant for consumer, and encoder can greatly improve compression efficiency by skipping this insignificant information. Some special applications need lossless encoders. These encoders are commonly known and used, but their coding efficiency is lower and the computational complexity is still high.

This paper contains an idea of Gigabit Ethernet interface with lossless encoder implementation as was shown in Fig. 1. The main aim of this work is not to reduce bitrate significantly, but to create flexible encoder for different content, which can be efficiently implemented on FPGA. The created coder can work with different interfaces but adaptation of coding method applies only interfaces which divide input stream into packages. The coder can be modified by adding or removing coding algorithms – can be adjusted to data included in the data stream. The paper contains testing results of the proposed system.

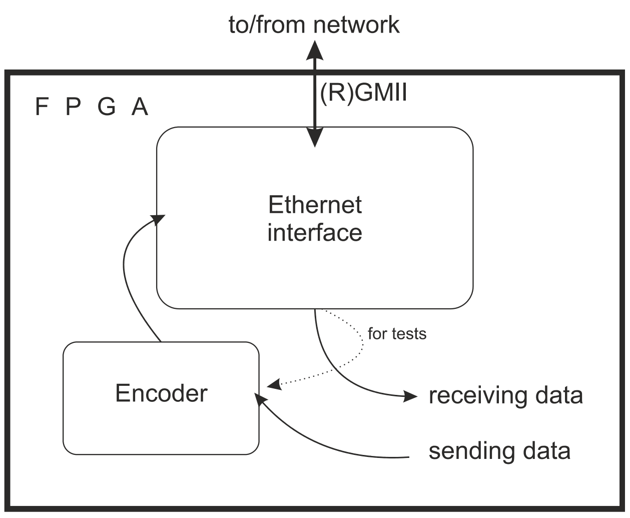

Fig. 1. Sending data encoder in FPGA

2. Ethernet interface

Ethernet interface is not designed for a specific data content. It can transmit different data in the same time and can connect different devices into one network. For these reasons this solution is so popular and flexible for different purposes. Ethernet is commonly used in FPGA devices because it can replace some dedicated interfaces[1]. Many applications can use the same interface, what can reduce time needed to make a new project. High throughput and content transparency make it perfect for testing the described encoder.

The package transmitted by the network can use various protocols which provide different features. Ethernet interface, which provides all possible options in FPGA device, would be too complex and in large part unused.

We propose to implement Ethernet interface as simple as possible. Our interface uses only one protocol – UDP[2] and can send and receive package only from one device in the network (from one IP[3] and one MAC address[1]). It has on advantage – every received and sent packet can have different length. Architecture of the proposed Ethernet interface was shown in Fig. 2.

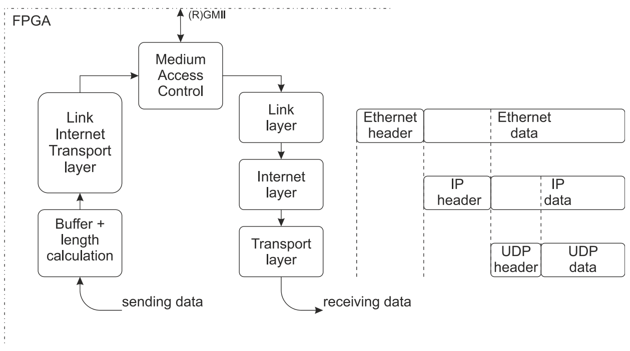

Fig. 2. Ethernet interface – block diagram

The data is coming to FPGA device from network through special interface which is independent from the transmission medium – GMII or RGMII ([Reduced] Gigabit Media-Independent Interface[4]). First part of Ethernet Interface splits the stream into two independent parts – received and currently being sent. The received part is divided into smaller parts. Each module verifies consequent protocol. Some important fields (e.g. MAC address, higher layer protocol) are checked and if the value is the same as the expected value, the header is removed and data is sent to the next part. The data leaving Ethernet interface is UDP data. The sending module uses a different approach – the headers of all layers are created by one common module. To fill all headers one additional information is needed – the data length. This information is known once the last byte of sending packet is received. For these reason first sending module holds the data in buffer and calculates size of the packet. When the packet is completed, two pieces of information (length and data) are available for the next module. The next module adds the headers and completed packet is sent to the network by (R)GMII.

3. Coding tree

In order to provide encoder flexibility, the encoder was implemented as a coding tree which is shown in Fig. 3. Each packet sent in the network is coded independently and can use a different coding algorithm. Coding tree uses the solutions which can be efficiently implemented in FPGA – as the encoder should be a module without large complex parts. If the coded packet being coded contains data unable to compress using available encoders, it is sent without compression. This way the coding tree never increases the data stream even if it has limited number of encoders to choose. The proposed encoder works as a module which codes only the package whose size can be decreased by used coding algorithms. Information about the chosen path was added to the stream as a first byte.

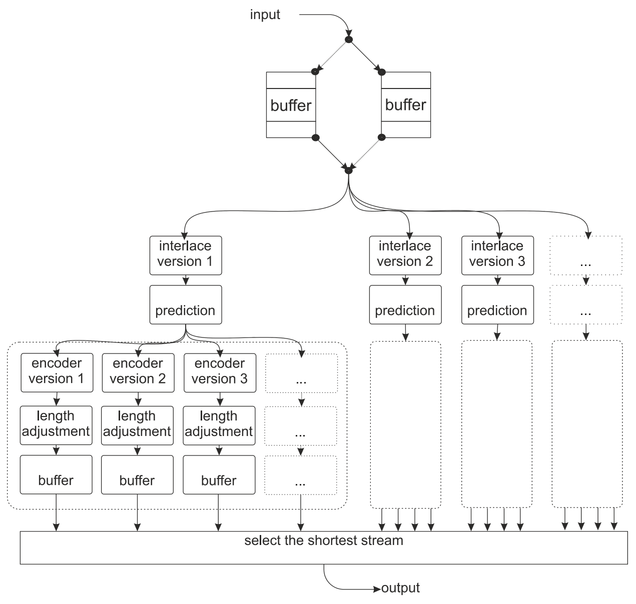

Fig. 3. Encoder coding tree block diagram

Encoders can work more efficiently when the input stream is be changed by a simple operation like prediction or interlacing. The first operation is commonly known and often used[5]. Proper interlacing can be really important for some types of input data. This can be true when data represents multichannel audio or comes from image converter (represents bayer matrix). These two operations are used in some coding paths and can increase coding efficiency.

4. Coding tree implementation

Encoding tree can be implemented like shown in Fig. 3, but this architecture is not optimal. Each coded path needs to be written into a buffer and to wait for the choice of the best path. Only the best path data is used and sent into the network. This step can be performed without using many buffers. This solution is shown in Fig. 4. First, data is saved into input buffer. Next, packets are coded through all available paths. There is no need to remember all coded data because only the size of data is important. FPGA synthesers can remove parts of encoders because output data ports are used only to count the returned data size. When all encoders finish encoding, the best path is selected. This information is used in the next step when the same packet is coded by a chosen path and sent into the network.

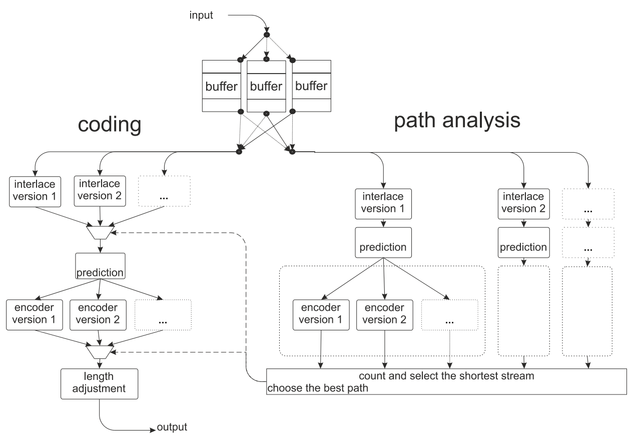

Fig. 4. Encoder implementation block diagram

Input buffer can work independently with three packets: the first is written into memory, the second is read by part of the coding tree which chooses the best coding path and the third is coded and sent into the network. Delays made by input buffers are lower when the size of input packets is constant. In different cases the delay is as long as time of processing of the largest packet.

5. Tested encoder

The tested coding tree implements coding paths able to encode wide spectrum of input data types. The tested system was shown in Fig. 1 using dotted line. Data incoming from network was coded and sent back to the sender. Therefore, data can be sent to encoder to estimate coding efficiency. The testing device was sending data by network, receiving and decoding them to check if the encoder works properly.

In the presented tests, the coding tree used 4 versions of interlacing (without, every second, third or fourth byte), prediction (previous byte subtraction) and five encoders (3 versions of RLC encoders, exponential Golomb encoder and static Huffman encoder[5][6]).

In Table 1 the synthesis results of basic network components are shown. As one may see, important modules of our encoder use only small part of resources. The biggest part (e.g. length adjustment) is not an important part of the encoder and can be implemented in a more efficient way to reduce significantly the overall size of the encoder.

Tab. 1. Synthesis results for Virtex5 xc5vlx50-2 device

			Slices	**Slice-ce-Reg**	**LUTs**	**LUT RAM**	**BRAM/ FIFO**
Ethernet			891	976	2104	879	0
•	Medium Access Control		64	88	140	9	0
•	Receiver		94	209	183	0	0
•	•	Link Layer	46	83	97	0	0
•	•	Internet Layer	16	41	16	0	0
•	•	Transport Layer	19	43	28	0	0
•	Transmitter		733	679	1781	870	0
•	•	Link Internet Layer	220	525	407	0	0
•	•	Transport Layer	513	154	1374	870	0

	Slics	Slice-Reg	LUTs	LUT RAM	BRAM/ FIFO
Encoder	2091	3098	5732	2	9
• Input Buffers	131	36	292	0	8
• Path Analysis	1227	1370	4123	1	0
• Coding Tree	695	1626	1226	1	1
• • Interlace v0	11	13	25	0	0
• • Interlace v1	15	16	44	0	0
• • Interlace v2	20	18	56	0	0
• • Interlace v3	23	20	57	0	0
• • Prediction	19	37	42	1	0
• • RLC v1	19	30	31	0	0
• • RLC v2	8	13	14	0	0
• • RLC v3	8	17	20	0	0
• • Exponential Golomb encoder	11	20	7	0	0
• • Huffman encoder	0	0	0	0	1
• • Length adjustment	410	1179	630	0	0

6. Results

The first part of tests contains only uncompressed high resolution video sequences represented by luminance and two chrominances. Every test sequence was represented by 100,000 packets, 1kB each. For testing, 21 sequences were used. During the tests, the encoder had coded two gigabytes of data. The results were shown in Fig. 5

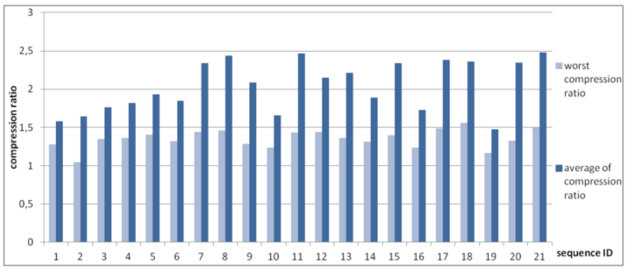

Fig. 5. Test results for uncompressed video sequences

As shown on the bar graph, almost all test sequences have average compression ratio not lower than 1.5. Eleven sequences have compression ratio higher than two – data size was reduced by half.

The second part of the tests contains only packets with uncompressed audio. Each coded audio stream was divided into 10,000 packets, 1kB each. The test results were presented in Fig. 6.

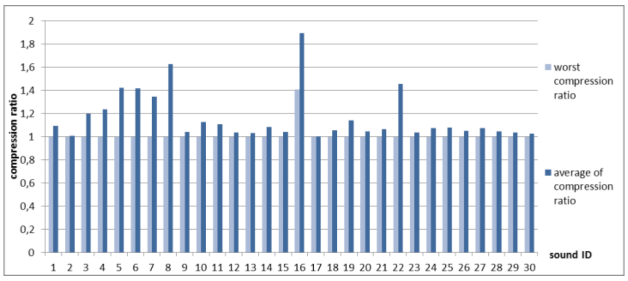

Fig. 6. Test results for uncompressed audio

As was presented, uncompressed audio was more difficult to be compressed efficiently. Many tested files were sent hardly any size reduction – the encoders in the coding tree reduced the size of the coded packets but the reduction was not significant, or data was sent without any compression.

7. Conclusions

The proposed Gigabit Ethernet FPGA interface with lossless data encoder can be a useful part of every FPGA device with high-speed interfaces. The presented results show significant throughput reduction with the use of simple encoders on FPGA and stream packetization. The path without encoders ensures that incompressible data will not be larger than originally. In more than half tested sequences, the compression ratio was higher than 2, so compressed data from stereo-camera would be smaller than stream from the standard (monoscopic) camera. Despite small size of the encoder, it can provide significant bit stream reduction.

8. Acknowledgements

The presented work has been funded by the Polish Ministry of Science and Higher Education within the status activity task “Signal processing and antenna optimization for acquisition, processing, analysis and presentation in 3D television systems” in 2015.

9. References

[1] N. Alachiotis, S. A. Berger, A. Stamatakis, Efficient PC-FPGA Communication over Gigabit Ethernet, CIT, Munich, 2010
[2] J. Postel, User Datagram Protocol, RFC 768 (Standard), Internet Engineering Task Force, August 1980
[3] J. Postel, Internet Protocol, RFC 791 Stand-ard, Internet Engineering Task Force, Sep-tember 1981.
[4] IEEE Std 802.3-2012, IEEE Standard for Ethernet, New York, December 2012
[5] M. Domański, Obraz cyfrowy, WKiŁ, Warsaw, 2010
[6] C. Thiele, B. Vizzotto, A. Martins, V. Rosa, S. Bampi, A low-cost and high efficiency entropy encoder architecture for H.264/AVC, VLSI-SoC, Santa Cruz, 2012

Received: 00.00.2014 *Paper reviewed* *Accepted: 05.01.2015*

At the end of the monthly MAM issue there will be biographical notes of all authors of papers. They should be prepared in accordance with the example.

Ph.D. Adam ŁUCZAK

Received his M.Sc. and Ph.D. degrees from Poznan University of Technology in 1997 and 2001, respectively. In 1997 he joined the image processing team at Poznan University of Technology. Member of of Polish Society Theoretical and Applied Electrical Engineering (PTETiS). His research activities include video coders control, MPEG-4/H.264 systems and hardware implementations of digital signal processing algorithms. He received Annual Fellowship for Young Scientist from the Foundation for Polish Science (FNP) and also a Group Award of the Minister of the National Education for research in the National Education for research in image compression. Currently he is involved in some project on video coding and video delivery.

e-mail: aluczak@multimedia.edu.pl

M.Sc. Adam GRZELKA

Received M.Sc. degree from Poznan University of Technology in 2014. He is a Ph.D. student at the Chair of Multimedia Telecommunications and Microelectronics. The main area of his professional activities are image processing, FTV (Free Viewpoint Television) and FPGA – especially implementation of compression algorithms and communication interfaces.

e-mail: agrzelka@multimedia.edu.pl

Eng. Grzegorz DULNIK

He is a M.Sc. student at the Chair of Multimedia Telecommunications and Microelectronics. The main area of his interests are FPGA devices and microcontroller programming.

e-mail: dulnik.g@wp.pl