Czech Technical University in Prague
Faculty of Information Technology
Department of Digital Design

Extendable and Scalable FPGA-based High-speed Packet Processing.

by

Jiří Halák

A thesis submitted to
the Faculty of Information Technology, Czech Technical University in Prague,
in partial fulfilment of the requirements for the degree of Doctor.

PhD programme: Informatics

Prague, February 2013

Thesis Supervisor:
Hana Kubátová
Department of Digital Design
Faculty of Information Technology
Czech Technical University in Prague
Thákurova 9
160 00 Prague 6
Czech Republic

Thesis Co-Supervisor:
Sven Ubik
CESNET z.s.p.o.
Zikova 4
160 00 Prague 6
Czech Republic

Abstract

Many high-speed (10 Gb/s and above) network monitoring and traffic processing applications require hardware acceleration. Different applications require different functions placed in hardware. Software solutions are mostly less reliable and require multiple PCs and additional hardware to work without packet loss. Current packet capture cards include fixed firmware, which is difficult to extend. The extendable and scalable solutions are desirable mainly for research purposes but also for real applications, which require extendability or scalability.

The beyond high-definition video transfers and processing applications over the high-speed Internet are still more demanded application. Remote cooperation all over the world improves the completion time and reduces the resources required. The rapid development of new applications and high network data throughput is the key element for uncompressed transfers in the ultimate quality. The use of platforms and architectures already available can greatly improve the future development of the new applications.

This dissertation thesis consists of selected publications by the author which address the new architectures for network packet processing, especially reconfigurable module based architecture for network packet processing and passive network monitoring, and describes the extension of this architecture for video transfers and processing. I have presented several practical experiments which evaluate the architecture and proposed techniques.

Keywords:

FPGA, Reconfiguration, Network packet processing, High-speed networks, HD-SDI, Beyond-high-definition video, Video processing.

Acknowledgements

First of all, I would like to express my gratitude to my dissertation thesis supervisors, Hana Kubatova and Sven Ubik for their professional guidance.

This work was supported mainly by the research plan MSM6383917201 - "Optical National Research Network and its New Applications".

This work has been partially supported by international project "Multi-Gigabit European Academic Network" (GN2) and Czech Technical Agency project POVROS (TA01010324).

Contents

List of Figures

List of Tables

Abbreviations

Digital Design Terminology

FPGA	Field Programmable Gate Array
DAC	Digital-to-Analog convertor
ADC	Analog-to-Digital converter
DPR	Dynamic Partial Reconfiguration
MTPP	Modular Traffic Processing Platform
IP core	Intellectual Property core

Computer Networks Terminology

XAUI	10Gb Attachment Unit Interface.
NIC	Network Interface Controller.
MAC	Media Access Layer.
BER	Bit Error Rate.
CRC	Cyclic Redundancy Check.
SFD	Start of Frame Delimiter.
SONET	Synchronous Optical Networking.
SDH	Synchronous Digital Hierarchy.
ISO	International Organization for Standardization.
OSI	Open Systems Interconnection.

Video processing Terminology

SMPTE	Society of Motion Picture and Television Engineers
SDI	Serial digital interface is a family of video interfaces standardized by SMPTE.
HD-SDI	High-definition SDI.
HDMI	High Definition Multimedia Interface.
DVI	Digital Visual Interface.
4K	A video format which has approximately 4000 columns. Number of lines is based on aspect ratio.
8K	A video format which has approximately 8000 columns. Number of lines is based on aspect ratio.
SHV	Super-High Vision - a video format which increase the 4K precision to 8K but for only green color component which is most dominant.
CAVE	An immersive virtual reality environment where projectors are directed to three, four, five or six of the walls of a room-sized cube.
HD	High-Definition.
UHD	Ultra High-Definition.
UHDTV	Ultra High-definition Television.

Chapter 1

Introduction

This chapter gives an overview of the problems and presents contributions and structure of the dissertation thesis.

1.1 Motivation

The main goal was to design an architecture for network monitoring purposes for speeds starting at 10Gb/s, scalable to higher speeds and fully extendable by new features and processing capabilities [D.1, D.2]. The available network processing architectures ended at 10Gb/s and even at this speed there were some throughput and processing problems. On the other hand, there were several commercially available products that were fully operational at 10Gb/s, but were very expensive or without any possibility of user configurations.

Pure software solutions depend on the computer configuration and even now are still mostly unable to process data at full 10Gb/s speed. Even with the best computer hardware currently available, a single computer can hardly manage to process all data packets in the 10Gb/s Ethernet. In order to work without packet loss, software solutions require multiple PCs and the additional hardware. Complex processing operations realized in software only are not possible. Pure software solutions are still less reliable and mostly add unnecessary processing latency.

In the later phases, the work started to focus on ultra-high-definition video transfers and processing. Video transfers are an expected driver application area of the future Internet. Picture resolution has been increasing over time. Ultra-high-definition video (such as 4K) is already used in some areas, such as scientific visualizations, the movie industry, medical applications or even CAVE-to-CAVE visualizations. I have discovered that the architecture for network packet processing is also suitable for this area of expertise. This knowledge resulted in additional work allowing us to extend the network packet processing architecture for video data processing [D.3, D.4].

1.2 Problem Statement

The high speed processing of network packets is generally a complex problem. It is hard to process all data without any loses on data speeds of 10-100Gb/s, especially on computer based systems. Hardware accelerated solutions are necessary for this task. They are designed to fit perfectly the target application and to achieve the best performance possible for current technology, mostly using highly parallel processing. On the other hand, development of a hardware platform takes a long time even using FPGA chips and any customization requires repeating a complex designing process. An easily customizable hardware solution is highly demanded especially in research areas, which can greatly speed up all designing processes [41].

1.2.1 Passive network monitoring and packet processing

In passive network monitoring, real user traffic is processed directly as opposed to active monitoring, when injected test traffic is used. Passive monitoring allows us to detect properties of real traffic, such as security attacks, traffic dynamics or real packet loss rate. Packet traces are also a useful resource for networking research. Increasing the speed of current optical networks demands proper monitoring of multigigabit lines.

Proper monitoring requires the processing of all incoming data packets, which means processing the worst possible case that is a continuous stream of the shortest data packets possible. The Ethernet standard defines the shortest possible packet as 64B long, but shorter data chunks have to be noted. Those data chunks can be a sign of some line problems or attack attempt. The most complex part of the Ethernet packet for processing is a packet header. A Packet header contains all necessary information about the transported data; packet body is mostly discarded except for string search or data intrusion and analysis applications.

A simple header analysis of one packet cannot take longer than the real space between two headers. In the worst case dealing with a continuous stream of 64B packets on 10Gb/s line, every header must be processed in less than 67.2 nanoseconds [24].

$$64Bpacket + 8Bpreamble + 12Bgap = 84B = 14,880,952packets/s = 67.2ns/packet$$

Packet processing is even more time demanding because we need to process packet header and make necessary changes, examine or even change packet payload data. The processing speed has to be adjusted according to the worst possible scenario, all necessary analysis and processing must be done within 67.2ns to process all packets without a single packet loss.

1.2.2 Video packetization and processing

Video transfers are a more demanded application in the modern Internet. Streaming in ultra high quality (more than HD or multiple streams, 3D, etc...) or even remote

cooperation (where video is transported in both directions) together with a basic video processing capability is becoming a common request of audio/video professionals. PC based video software using a frame buffer which is packetized then sent to another side and then depacketized to another frame buffer is slow, and the delay in communication between two endpoints is mostly more than noticeable. Higher resolution video streams like 4K or 8K require multiple PCs to be even properly displayed. Dedicated Video processing equipment becomes expensive. Designing a new video processing platform from start takes a long time and modularity and scalability of this solution is not guaranteed. A new approach, where video can be processed like network traffic is an interesting and easily adaptable solution which can greatly improve video communication over a future Internet.

1.3 Contributions of the Thesis

The following contributions are described in Chapter 3.

Scalable embedded architecture for 10-100 Gb/s packet processing for FPGA devices. I have proposed a new scalable and embedded architecture for network packet processing intended for 10-100 Gb/s links. The architecture is designed for processing the network packets at full speed at the worst case without a packet drop, which is possible using the balance between clock rate, data width and pipelining level together with a simple interface between plug-in modules for data processing. The architecture is designed to be distributable between several FPGAs or in a single FPGA, thus allowing scalability even if current FPGA technology cannot support the required resources. The main goals of the architecture are a flexibility of target applications, easy extendability based on plug-in modules and good availability for a high-speed network research community. The 10 Gb implementation of this architecture was implemented by CESNET in a MTPP (Modular Traffic Processing Platform) device.

This contribution was published in papers [D.1] and [D.2] included in this dissertation thesis. Several results of this work were converted into patent [P.2].

Hardware modules for network monitoring on the MTPP platform. The MTPP platform is a dynamically reconfigurable hardware platform for 10 Gb/s network packet processing developed by CESNET, which is based on the architecture described in the first contribution. I have proposed and implemented plug-in modules for the MTPP platform allowing the use of the MTPP for a passive network monitoring. The set of modules consists of standard and burst based statistical modules and modules for bit error rate tests. The resulting monitoring solution is still being used in the CESNET2 network.

This contribution was published in papers [D.1] and [D.2] included in this dissertation thesis.

Packet processing architecture for beyond-high-definition video transfers. I have proposed an extension to the architecture described in the first contribution allowing

to transfer and process beyond-high-definition video or several independent channels of high-definition video. This new approach where video can be processed like network traffic is an interesting and easily adaptable solution, which can greatly improve video communication over a future Internet. This work was adopted by the Czech Technical Agency project POVROS, whose goal is to move the results of this research to the market.

This contribution was published in papers [D.3] and [D.4] included in this dissertation thesis. Several results of this work are included in the utility patent [P.5].

1.4 Structure of the Thesis

My dissertation thesis is organized into 5 chapters as follows:

1. *Introduction*: Sets out the context of this thesis together with the goals. There is also a list of contributions of this dissertation thesis.

2. *Background and State-of-the-Art*: Introduces the reader to the necessary theoretical background and surveys the current state-of-the-art.

3. *Contributions*: Summarizes the contributions of this dissertation thesis.

4. *Conclusions*: Summarizes the results of the research, suggests possible topics for further research, and concludes the thesis.

5. *Publications Included in Thesis*: This section contains publications included in Dissertation thesis.

Chapter 2

State-of-the-Art

This Chapter describes the theoretical background and related work. The theoretical background Section explains the necessary terminology to understand the text. A reader familiar with network packet processing, FPGAs and beyond-high-definition video can skip this section. The Related Work Section introduces comparable or related work.

2.1 Theoretical Background

This Section explains the necessary terminology of network packet processing, FPGAs and beyond-definition video transfers and processing.

2.1.1 Ethernet Packet Processing

Ethernet [24] data is encapsulated in frames as shown in Figure 2.1. The fields in the frame are transmitted from left to right. The bytes within the frame are transmitted from left to right (from least significant bit to most significant bit unless specified otherwise). All frames in the Ethernet networks [24] consist of preamble, SFD(Start of Frame Delimiter), header, data and CRC sections shown in Figure 2.1. Preamble and SFD are the unique set of bits beginning Ethernet packet. In the header, there is information about the packet such as source and destination address, protocol type, packet length, etc. The data section is the body of the packet, which is carrying the data that has to be transported. The CRC section contains 32-bit CRC checksum of the packet to discover corrupted packets caused by errors in communication medium and transceivers.

In packet processing, we are interested in the packet content and information about the packet. The control parts of the packet such as preamble, SFD and CRC should be taken care of automatically. According to the ISO/OSI standard [23], the control parts of the packet are available only on the physical layer, which purpose is to make a general network interface to all types of communication medium. To provide a higher level of abstraction we can use the Media Access Layer(MAC) that removes the preamble, checks the CRC on the receiver side, adds preamble and computes the CRC on the transmitter side. On the

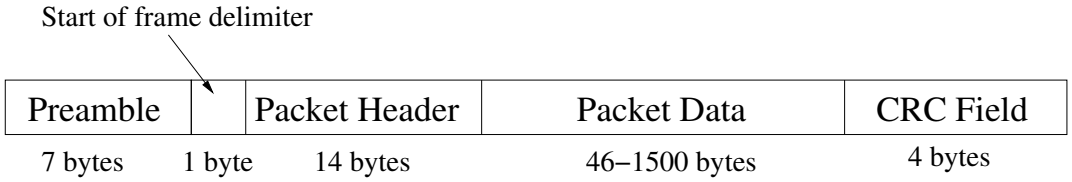

Figure 2.1: Ethernet Packet Structure.

Formats	Standard
HD	SMPTE 274M-2008 [20]
UHDTV1, UHDTV2	SMPTE 2036-1-2009 [22]
2K, 4K, 8K	Multiple standards SMPTE 428-x-20xx [6]

Table 2.1: Video standards

MAC layer, we work only with the packet content. At this point, we do not require more abstract approach, which is provided by upper layers defined in the ISO/OSI standard [23]. The packet content is only the packet header and packet body with the transported data. Therefore, when analyzing contents of the packet stream, we can work on the MAC layer. However to analyze line errors on the network, we should work on the physical layer. When transmitting new packets there is the same choice. When we need to generate only the packet content, we can use the MAC layer. However to generate packets in any condition such as bad CRC, corrupted data or to simulate the network problems, we have to work on the physical layer.

2.1.2 High-definition and beyond-high-definition video

When dealing with high-definition(HD) or ultra-high-definition(UHD) video, we work with image resolution 1920x1080 pixels or better. The standardized image resolutions for HD and UHD videos are described in Figure 2.2. The video standards are produced by the Society of Motion Picture & Television Engineers (SMPTE). The standards are basically divided to television standards and digital cinema standards. Television resolution starts at a common HD resolution and rises with multiplication of HD screens to UHDTV1 and UHDTV2. Digital cinema common HD resolution is 2048x1080 and is marked as 2K, which is the number of video columns, and is further scaled by the multiplication of 2K screens to 4K or 8K. Images can be progressive or interlaced with a variety of frames per second. Video resolutions with corresponding standards are summarized in Table 2.1

There are several standard video interfaces for uncompressed video. The most common are Digital Visual Interface (DVI) [8], High Definition Multimedia Interface (HDMI) [11] and Serial Digital Interface (SDI) [19, 1]. DVI is a video display interface developed by the Digital Display Working Group. The digital interface is used to connect a video source to a display device, such as a computer monitor. HDMI is a compact audio/video

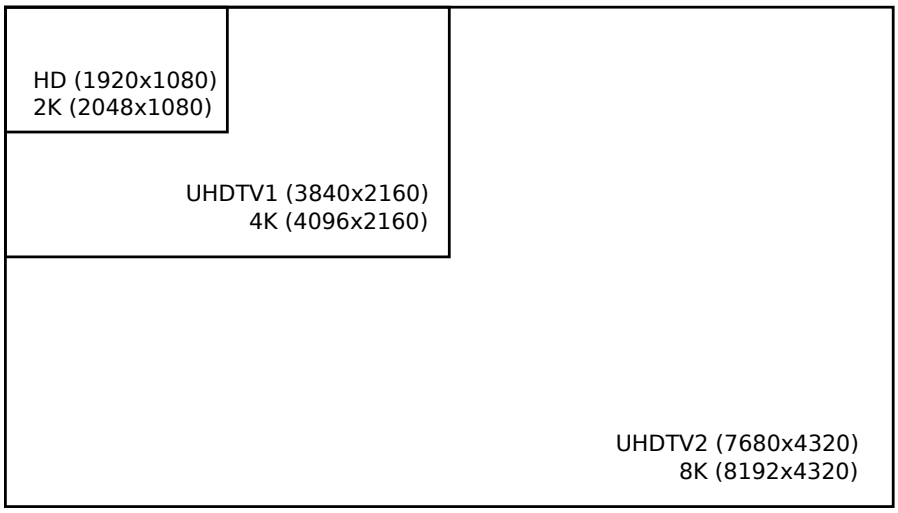

Figure 2.2: High-definition and Ultra-high-definition image formats.

Interface	data rate	best format	Standard
HD-SDI	1.485 Gbit/s	2K YCrCb	SMPTE 292M [1]
Dual link HD-SDI	2.970 Gbit/s	2K RGB	SMPTE 372M [7]
3G-SDI	2.970 Gbit/s	2K RGB	SMPTE 424M [21]

Table 2.2: SDI interface standards

interface for transferring encrypted uncompressed digital audio/video data between the HDMI-compliant devices. SDI is a family of video interfaces standardized by SMPTE.

Although the data is transported in the comparable way on all interfaces, we have chosen the SDI interface. The use of SDI for the transmission of high definition video streams is now a common industry practice. The interface is designed for video and audio transmissions and supports all variety of encryption methods. SDI standards are summarized in the Table 2.2

The emerging interface, commonly known in the industry as Dual Link HD-SDI essentially consists of a pair of HD-SDI links. A more recent interface 3G-SDI consists of a single 2.970 Gbit/s serial link will replace the Dual Link HD-SDI in the future. A single HD-SDI link can transfer HD video only in 10-bit YCrCb format in 4:2:2 color subsampling. Dual Link HD-SDI and 3G-SDI can transfer up to 12-bit RGB in 4:4:4 or two single HD-SDI channels which can be independent or synchronized for 3D video. Higher resolutions are transferred using the multiple HD-SDI links which match the required throughput. For example, 4K video in YCrCb 4:2:2 can be transferred by 4 HD-SDI channels (one for each image quadrant) and 4K in full RGB color precision can be transferred by 8 HD-SDI chan-

nels or 4 Dual Link HD-SDI, 3G-SDI channels. More information about color subsampling and precision can be found in [1].

2.1.3 Programmable Hardware and FPGAs

FPGAs(Field Programmable Gate Arrays) [35] are considered to be the most advanced programmable hardware. Programmable hardware as opposed to dedicated hardware is defined as a hardware part or a chip with some method of user configuration. The dedicated hardware is designed for a specific operation or a set of operations that cannot be changed. On the other hand, the primary target of programmable hardware is general usage and user configuration, which enables the user to define his own behavior. The most obvious programmable is a processor, which is a single hardware device allowing countless user defined functionalities.

The basic concept of programmable hardware reaches far beyond processor cores. Programmable gate arrays are the chips containing a universal set of basic hardware components (gates) that can be arranged by the user to form the dedicated hardware circuit. They consist of set of universal components and general routing matrix. The device configuration is being held in a memory, which is directly connected to the routing matrix and basic components. Using this kind of programmable hardware, the user can form any hardware functionality, even a full featured processor core. Most of the modern hardware devices directly contain one or more processor cores.

FPGAs are the most advanced programmable gate arrays. FPGAs contain programmable logic components called logic blocks and a configurable routing matrix which allows the blocks to be connected together in many different configurations. Logic blocks can be configured to perform complex combinational functions, or merely simple logic gates like AND and XOR. In most FPGAs, the logic blocks also include memory elements, which may be simple flip-flops or more complete blocks of memory. In addition to digital functions, some FPGAs have analog features. The most common analog feature is programmable slew rate and drive strength on each output pin, or differential comparators on input pins designed to be connected to differential signaling channels. Some FPGAs have integrated peripheral Analog-to-Digital Converters (ADCs) and Digital-to-Analog Converters (DACs) with analog signal conditioning blocks, and the most advanced FPGAs can contain the dedicated hardware blocks like digital signal processing blocks, processor cores or physical layers for numerous communication interfaces.

2.1.3.1 Dynamic Partial Reconfiguration

Dynamic partial reconfiguration (DPR) is a process when only part of an FPGA configuration is modified [32, 38]. It is possible to modify part of an FPGA configuration in runtime without shutting down an FPGA operation, provided that the part we are modifying is not used during the upload of the modified part. Otherwise, the design can behave unpredictably during reconfiguration. We have chosen Xilinx FPGAs for their good DPR support and many available development board types.

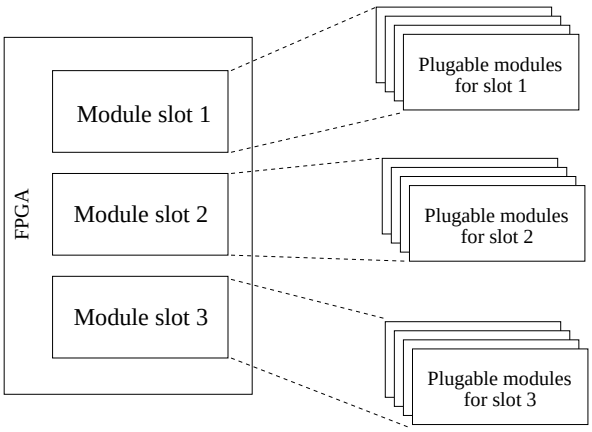

Figure 2.3: Reconfigurable Modules.

The partial reconfiguration can be realized in three different ways: the **differential partial reconfiguration**, the **modular partial reconfiguration** [26, 25] and the **partition based reconfiguration** [27].

In the differential approach, the reconfiguration is done using the differential bitstream. This bitstream contains only the changes between the old and the new configuration. When this bitstream is loaded into the FPGA with the old configuration it changes to the new configuration. The main purpose of this technique is the small change in the FPGA configuration. It cannot be used definitely to change the routing on the chip, and extreme caution has to be used to prevent unpredictable behavior during the application of a bitstream.

In the modular approach, rectangular areas are reserved on the chip. The part of the design that remains unchanged all the time is called **fixed design** and it has to be able to handle the reconfiguration process without collapsing. The part of the design reserved for reconfiguration is called **the reconfigurable area** and it has to be strictly divided from the fixed design. Design flow tools cannot use the reserved place, so it remains empty (except for some global clock resources or dedicated routing, which can mostly cross this area). The only way of fixed and reconfigurable design interconnections are the CLBs. The reserved reconfigurable area is called a module. The restrictions for the modules creation differ for every FPGA vendor and family so special care must be taken to implement the modular design correctly for the desired FPGA architecture. The use of reconfigurable modules is illustrated in Figure 2.3.

For every module the same number of bitstreams are generated as the number of module slots. This way every module can be inserted to every slot.

Partition based reconfiguration is a modern approach but also a special case. Implementing a partially reconfigurable FPGA design is similar to implementing multiple non partially reconfigurable designs that share common logic. Partitions are used to ensure that the common logic between the multiple designs is identical. A partition is a logical

section of the design, defined by the user at a hierarchical boundary, to be considered for design reuse. A partition is either implemented as new or preserved from a previous implementation. A partition that is preserved maintains not only the identical functionality but also the identical implementation. Partition based can be considered a special case of the modular approach. Partitions allow more flexibility in available resources for reconfigurable regions but the need of full bitstream for every configuration is very limiting for large reconfigurable architectures with many reconfigurable modules.

2.2 Related Work

This Section introduces comparable or related work, and the advantages of the presented architecture.

2.2.1 Network packet processing

This Subsection introduces the related work in network packet processing. The main difference of the presented architecture between all the related work is that data processing in the presented architecture is only hardware based, and the main processing core is designed for a maximum data throughput and a processing strength. As such, it was not designed to send the data for further analysis to the PC (even if this modification is possible), but was designed for the main network nodes or high speed backbones networks.

2.2.1.1 NetCOPE platform

The NetCOPE platform [4], which is a part of the Liberouter [3] project is a realization of a general platform for the rapid development of network applications [44] on the family of COMBO cards. The proposed platform includes network interface blocks (1G/10G Ethernet), high-speed programmable bus-master connection to the software layer via PCI-X or PCI Express bus and a generic interface to a potential hardware accelerator for network processing. The generic data transfer protocol between the network and co-processor interfaces allows for an easy integration into the target application. The platform further offers a set of IP cores usable as basic building blocks for a wide range of network applications, including cores for packet analysis, classification, packet modification, precise timestamps, pattern matching, statistics etc.

NetCOPE is not a physical platform but a set of IP cores for easy assembling of new applications based on a COMBO card. This solution offers high universality and flexibility but requires a PC and dedicated software. The COMBO cards were tested under the CESNET Performance Monitoring Project [5].

The main differences of the presented architecture are:

1. The presented architecture is primary designed to be extremely simple and allow the high data throughput which is only limited by the actual network bandwidth. The NetCOPE platform may be more universal allowing better hardware/software

co-design but has some limitations in strictly performance demanding applications, which was tested under CESNET Performance monitoring project [5].

2. The presented architecture is modular based with a highly customizable modular processing unit, which allows easy functional extendability and future development. The DPR also allows much faster development because the fixed part of the design can be supplied as a binary bitstream.

3. The presented architecture is a standalone solution which does not require a PC, this also includes a focus on the different applications than a PC based card.

2.2.1.2 FPX platform

The platform is described in [36]. It is an open platform for development of network processing modules in reprogrammable hardware. The platform is based on the FPGA chip with dynamic partial reconfiguration. The platform includes an interface for 1Gb Ethernet and is intended for easy development of new network processing applications at a speed of 1Gb/s.

The FPX platform was used for the implementation of many applications such as packet analysis, cryptography, packet modification, etc [34, 40, 36, 30]. This architecture was designed for lower speeds so is not optimized for multigigabit data flows. My approach, described in 3.1, is directly focused on the high data throughput and pipelined data processing in multiple independent modules.

2.2.1.3 NetFPGA platform

The platform is described in [37]. It is a network platform for the academic community, which allows faster development of applications for multigigabit networks. The basic selection is similar to NetCOPE platform. The NetFPGA platform is also a set of cores which can be used for the development of network applications. The platform also has a hardware part, which is a PCI-E board dedicated for this purpose. The differences between this approach and my approach are the same as of the NetCOPE platform, although it is well designed for the computer aided data analysis, my approach is dedicated for high performance applications. The NetFPGA platform focuses more at multiple 1Gb/s links. The example application running at the NetFPGA platform is a SwitchBlade [28].

2.2.1.4 Scalable Multi-FPGA platform

Scalable Multi-FPGA platform [39] is similar to the presented architecture. It is focused on high compute power to perform complex real-time processing of network packets. The architecture is designed as a scalable multi-device system which support any number of FPGAs connected to the ring. The current prototype is distributed in four FPGAs connected to a ring. This platform focuses on processing of the higher network level protocols.

There are defined several plug-in modules per FPGA which can implement a processing function.

This solution is very similar to the presented architecture except that it is focused more on higher network protocol levels. However this architecture is younger than the presented architecture and the presented architecture is oriented for more applications, which is proved by the third contribution, which adapted this architecture for ultra-high-definition video processing.

2.2.1.5 Endace Products

Endace [9] produces network traffic monitoring technology. It provides network monitoring, latency measurement and application solutions to capture, inspect and report on data packets on speeds of up to 10Gb/s. Endace uses combined software and hardware solutions.

In most solutions, PCI Express cards with FPGA chips are used to capture packets, or optionally make a simple filtering or classification, and store them to the computer memory. All other processing is done in software. This solution depends on the performance of the host computer. An accelerator is used to transport data to the computer memory with the lowest possible CPU usage. Therefore the host computer has to process incoming packets at full speed.

This solution can be used as accelerated network card with filtering and classification options. With the combination of software tools wide range of applications can be created. The applications performance depends on the performance of selected software tools. The hardware cards and software tools were tested under the CESNET Performance Monitoring Project [5]. The hardware accelerated cards can store packets from 10Gb Ethernet at full speed to the computer memory but currently there is no way to process all packets in software without any losses.

2.2.1.6 Network analyzers

Network analyzers are commercial equipment capable of the massive processing power required to process the data at the full speed on the optical networks at the speeds of up to 100Gb/s [13, 10]. Their major disadvantage however is almost zero extendability and no means of user own configuration. The implementation of any additional functionality is also not possible. The analyzers are mostly larger than a common computer and multiple time more expensive than the presented solution.

2.2.2 Beyond-high-definition video transfer and processing

This Subsection introduces related work in beyond-high-definition video transfer and processing. The presented architecture is a new approach and as such there are only a few comparable solutions. The main advantage over all existing solutions is that the presented architecture is based on the network packet processing architecture and was not developed

as video processing architecture from the start. This proves the universality and extendability of the network packet processing architecture described in the first contribution and shows a new and simpler way designing a video transport and processing equipment which operates over the standard Ethernet Network. The target application of this approach is not a single HD or UHD transfer or a compressed video transfer (even if it is also capable to do so) but a massive multichannel UHD or HD transfer in ultimate uncompressed quality, and processing at the maximum possible network speed which may be 10Gb/s or above.

2.2.2.1 UltraGrid

UltraGrid [33] from the Laboratory of Advanced Networking Technologies (ANTLab) is a software implementation of high-quality low-latency video and audio transmissions using commodity PC and Mac hardware. This software is beeing developed with the cooperation of CESNET and Masaryk University in Brno. UltraGrid uses uncompressed or very low compression-ratio streams to achieve up to 4K resolution with as low as 100ms end-to-end latency. UltraGrid is used, amongst others in areas like collaborative environments, medical cinematography, broadcasting applications and various educational activities. Ultragrid software currently supports many advanced functionalities, such as 3D support or SAGE support. This solution is fully software based and requires a dedicated PC with specialized hardware.

The main differences between Ultragrid and the presented solution is:

1. UltraGrid is a software solution and as such is greatly dependent on the computers performance. UltraGrid is capable of uncompressed 4K transfers and is currently being tested for 8K or SHV. The software however is greatly dependent on dedicated PC hardware and the complex setup requires several PCs with dedicated hardware on each side.

2. The presented architecture is a pure hardware solution which allows multigigabit speeds of 10Gb/s and above in one small size device.

3. The presented hardware based architecture can process video with the lowest latency, which will be always several times lower than for PC based applications.

2.2.2.2 Professional industrial equipment

Net Insight's [17] Nimbra 600 series switch can transport 8x HD-SDI or 3G SDI channels over an SONET/SDH network. There are several commercially available solutions for transport of compressed 4K video over the Internet, for example NTT Electronics [18] ES8000/DS8000 4K MPEG-2 encoder/decoder complemented with NA5000 IP interface unit and intoPIX's [12] system of PRISTINE PCI-E FPGA boards and JPEG 2000 IP cores.

NTT Electronics have also presented several articles [42, 43] about their technology and several new products which are capable of transmitting the 4K video over the IP network even from multiple locations.

The main advantages of the presented architecture over the available equipment:

1. The presented architecture is designed for high data rates which allows the uncompressed transfer of multiple channels. The bandwidth is only limited by actual network bandwidth up to a multigigabit speed of 10Gb/s and above.

2. The presented architecture is modular based which allows easy functional extendability and future development.

3. The hardware based uncompressed transfer adds only a minimum processing latency.

4. The original concept of modular network processing architecture extends the possible functionality base.

5. The presented architecture is an easy adaptable solution for the research community.

2.3 Summary

The listed network packet processing architectures are currently very similar and offer rapid development of networking applications, high versatility and performance. When the project started, there was no available architecture with a guaranteed wire-speed throughput for 10Gb and above although they were more versatile. The current situation is much more balanced, and all architectures are almost equal in performance, extendability and rapid development of user applications. The proposed architecture was designed primarily for the wire-speed throughput of beyond 10Gb/s and plug-in based hardware modules for rapid development of user applications which was not possible with the available network processing solutions. Even now the proposed architecture offers comparable quality to other solutions and excels in throughput, which was the original goal.

The situation in video transfers and processing technology where the presence of similar architectures is missing, allowed the extension of the network processing architecture for video transfer and processing. The video processing applications do not offer the same portfolio of platforms and architectures which would allow the easy development of user applications and provide a stable base for further research. The resulting architecture inherits all the advantages of the network processing architecture without the need of developing a dedicated video processing solution. Moreover, the current solutions are mostly limited by the network capabilities and data throughput for the uncompressed video, which can be easy covered by the proposed architecture.

Chapter 3

Contributions of the Thesis

This Chapter gives the overview of contributions presented in this dissertation thesis. Every Section in this Chapter gives an overview of one contribution of this Dissertation thesis. The published results are attached in Chapter 5.

3.1 Scalable embedded architecture for 10-100 Gb/s packet processing for FPGA devices.

I have proposed a new scalable and embedded architecture for network packet processing intended for 10-100 Gb/s links. The architecture is designed for processing the network packets at full speed at the worst case without a packet drop, which is possible using the balance between clock rate, data width and pipelining level together with a simple interface between plug-in modules for data processing. The architecture is designed to be distributable between several FPGAs or in a single FPGA, thus allowing scalability even if current FPGA technology cannot support the required resources. The main goals of the architecture are a flexibility of target applications, easy extendability based on plug-in modules and good availability for the high-speed network research community. The 10 Gb implementation of this architecture was implemented by CESNET in a MTPP (Modular Traffic Processing Platform) device.

This contribution was published in papers [D.1] and [D.2] included in this dissertation thesis. Several results of this work were converted into patent [P.2].

3.1.1 Background

We required an architecture mainly focused on the throughput, which was not the common cause when this architecture was proposed. The resulting architecture must be simple, well scalable and extendable. I have chosen the series-parallel partial order for the composition of basic processing modules. In order-theoretic mathematics, a series-parallel partial order is a partially ordered set built up from smaller series-parallel partial orders by two simple

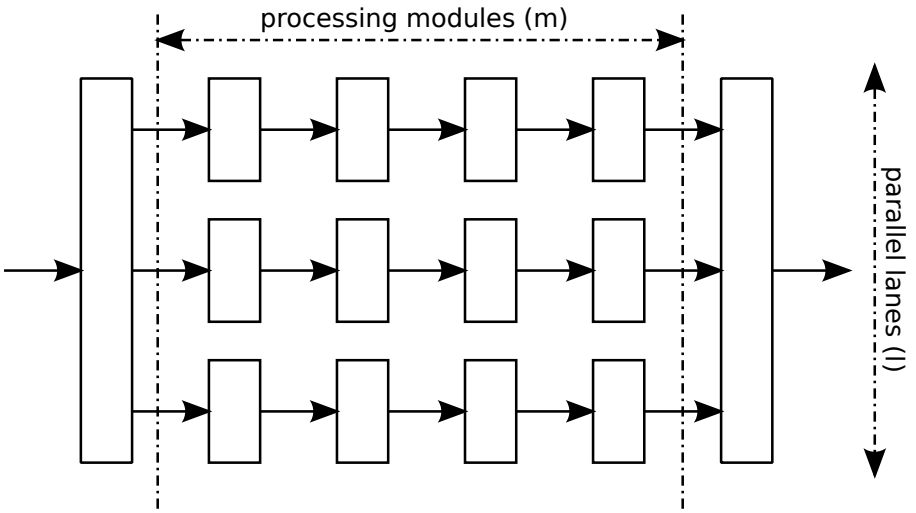

Figure 3.1: Modular processing core.

composition operations [29]. This technique has been already used in other areas for throughput maximization such as dataflow programming [31].

This component composition is perfectly scalable and extendable, also when using identical components, they can be changed freely for each other, which can greatly improve the configurability of the whole composition. I have adapted this model for the main processing core of the proposed architecture fulfilling the primary requirements for the resulting architecture.

3.1.2 The Architecture Overview

The architecture is divided to the two main sections. The static section and the modular processing core.

The modular processing core is the scalable processing core of the architecture. It is composed of the basic modules, each with the same unified and well defined interface and specific function.

The static part of the architecture consists of the static elements such as the processor unit, data interfaces or communication interfaces. The static elements are marked as static because they are not scalable or reconfigurable in any way and are static for all configurations of the architecture. The embedded processor system handles the device configuration and the user communication interfaces.

The main modular processing core is designed for maximum throughput and processing strength. The overview of this architecture is described in Figure 3.1. The modules are assembled to a number of parallel processing lanes, each lane containing several processing modules. The scalability and processing strength depends on the organization and the number of hardware modules. The number of modules (m) is the number of modules in

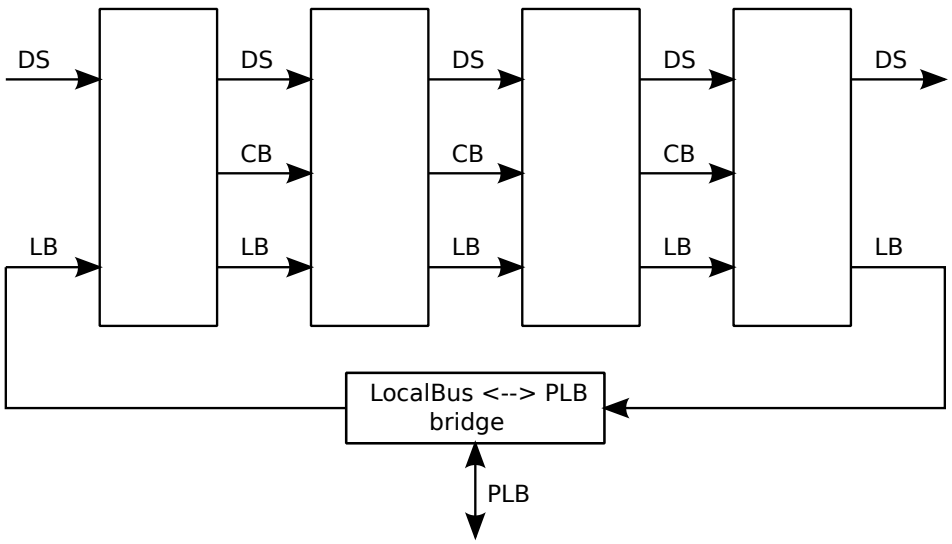

Figure 3.2: Buses for module interconnection.

every processing lane. The larger m increases the number of modules processing one data stream. One module can be implemented as a standalone function; thus more processing functions can be applied to the data stream. The number of parallel processing lanes (l) increase the data width which can be processed, increasing the throughput of the processing core. The scalability of the processing core is in two ways: Throughput (by increasing the parallelism) and the number of functions which can be applied to the data stream (by increasing the number of modules in a single lane).

The modules in every lane are connected with three buses; every bus has its specific function. The overview of the module interconnection is described in Figure 3.2. The most important is the fact that all connections can be made one-way, which simplifies the processing core scalability and allows mode pipeline depths.

Data Stream (DS) is a main processing bus which transports the data that needs to be processed which are the network packets in our case. The hardware modules are inserted directly into the data stream. Data can be processed, analyzed or remain unchanged. The only limitation of this approach is that precautions must be taken during the development of new modules to ensure that the data is properly passed to the next module.

Command Bus (CB) is the communication one way interface, which allows the sending of some additional information to the next module. At the fastest possible processing speed, there is no place in the data stream to insert any additional control data or commands for the next modules. This bus allows every module to attach any message or command which should be passed to the next modules. The following modules can read those messages and behave accordingly or ignore them. The example of the usability of command bus is the header analyzer which analyzes and qualifies the data packets. The packets cannot be altered; thus the gathered informations, which will be used by the next modules in a row, will be sent on the command bus.

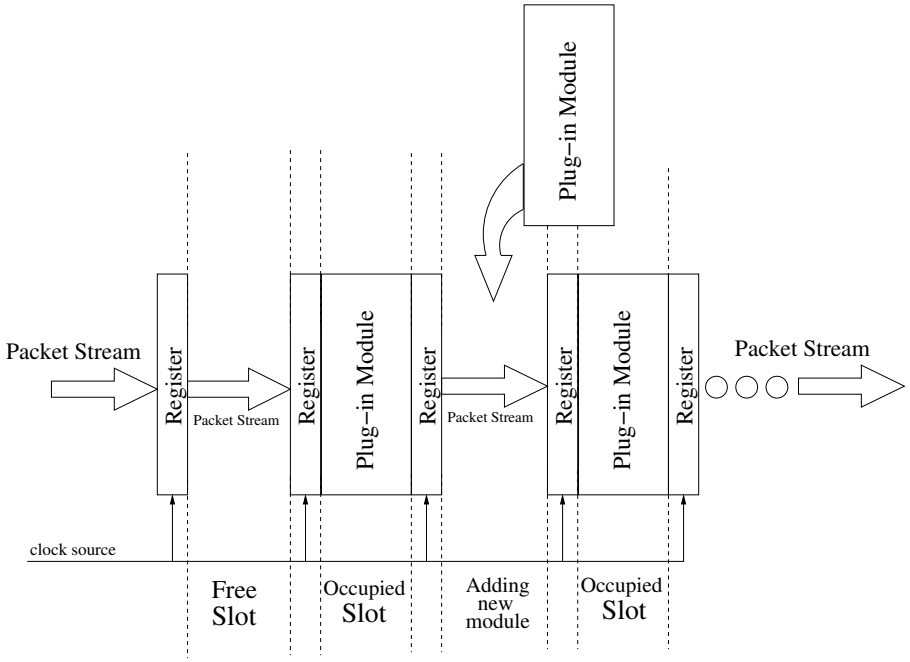

Figure 3.3: Plug-in Modules in Packet Stream.

Local Bus (LB) is the processor communication interface. It is accessed through the PLB to the local bus bridge, which translates the PLB transactions to the one-way communication through the modules. The advantage of this interconnection is that the number of modules connected to the local bus is not limited. The only limitation is that the latency of every PLB transaction is equal to the number of modules (m), which has almost no impact on the PLB performance considering the processor type and speed which is used in this application.

Custom hardware acceleration modules are based on Dynamic Partial Reconfiguration of FPGAs described in Section 2.1.3.1. It allows us to use many different hardware modules with a wide area of functionalities that can be freely combined and assembled building a final desired functionality. The whole situation is described in Figure 3.3. The modules have to forward the packet stream to the next module. The registers have to be inserted between the plug-in modules to make a pipeline in the stream. Those registers have an important function. They create slots for the plug-in modules which is synchronized by the packet stream clock frequency. Therefore, max combinational path cannot be longer then the longest path in the module and the packet stream timing do not depend on the module timing. It is an advantage of the new plug-in module developement that care must be taken only for the plug-in module timing.

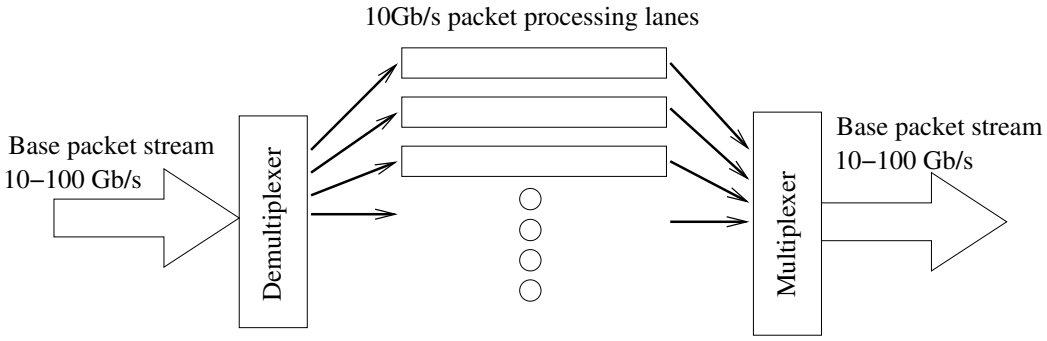

Figure 3.4: Packet stream processing in parallel lanes

3.1.3 Processing in parallel lanes

The tested speed for the data processing for one lane is about 10Gb/s, which can be achieved by clock frequency 156,25Mhz and 64 bit data width. The data processing for speeds above 10Gb/s is possible in the following ways:

- Increase the clock frequency. Can be doubled in modern FPGAs.
- Use wider data width. This depends on available FPGA resources. Although FPGA resources are practically unlimited, reasonably implementable data width in modern FPGAs can be up to 256 bits.
- The last possibility for speed increase is the processing in parallel lanes.

Data processing at speed of 40Gb/s can be achieved by higher clock frequency (max 2x 10Gb clock frequency) or wider data width (up to 4x 10Gb data width). This information was obtain experimentally using the best FPGA chips from Xilinx Virtex-5 series [A.7].

The higher speed of up to 100Gb/s requires additional processing parallelism. Another way of improving the data throughput is processing the data in parallel lanes. The high-speed base packet stream is first demultiplexed into several parallel lanes and at the end multiplexed back together into main high-speed packet stream as shown in Figure 3.4. With an increasing number of parallel lanes the area in the FPGA for every lane decreases. Therefore larger FPGA have to be used to keep the same area available for every lane, optionally using more FPGAs for additional resources.

3.1.4 Resource distribution across the FPGAs

All high-speed paths in the architecture are generally data packet streams which can be redirected to another FPGA through the High-speed serial interface. The number of those redirections depend on available high-speed serial interfaces of current FPGAs. This means that we can use additional FPGAs for several enhancements which cannot be fit to a single chip because of e.g. technology limitations.

The static parts of the architecture must always occupy the single master FPGA. The static part is everything except the processing modules. The main multiplexers and demultiplexers of the processing core which allow the use of more parallel lanes must be also located in the master FPGA. The static part is however very small compared with the processing core and even the smallest available FPGAs can be used for them.

The module(s) relocation can be done in a following ways:

- Reroute the high-speed path between modules. This feature allows us to relocate one or more reconfigurable modules to another FPGA. Complex, resource demanding modules may require this feature to overcome actual technology limitations.

- Reroute one or more parallel processing lanes. When the implementation of additional processing lanes is required and there are no available resources, one or more processing lanes can be implemented in another FPGA.

3.1.5 Summary

The presented architecture has several advantages over other commercial and research solutions described in Chapter 2.2.

The architecture is designed with the main goal of maximum throughput. Multiple applications can be built with the use of a few reconfigerable modules or by simply developing new modules without the need of complex knowledge about FPGAs and multigigabit-speed networks. This architecture can virtually exist over more FPGAs or it can be moved to a single FPGA when technology moves forward. The source code base, including modules, can be relatively huge but still able to adapt to a currently required solution. There can be a subset of functionality loaded to an FPGA and other unused modules and configuration can be stored in external memory and loaded on demand using FPGA Partial Reconfiguration.

All modules have the same, simple and well-defined interface, it is much easier to develop a new module with the unified interface than to extend a monolithic design or a design where modules have different interfaces or depends on other parts of the system.

3.2 Hardware modules for network monitoring on the MTPP platform.

The MTPP [14] platform is a dynamically reconfigurable hardware platform for 10 Gb/s network packet processing developed by CESNET, which is based on the architecture described in the first contribution. I have proposed and implemented plug-in modules for the MTPP platform allowing the use of the MTPP for a passive network monitoring. The set of modules consists of standard and burst based statistical modules and modules for bit error rate tests. The resulting monitoring solution is still being used in CESNET2 network.

This contribution was published in papers [D.1] and [D.2] included in this dissertation thesis.

3.2.1 The MTPP platform and passive monitoring

MTPP platform is a dynamically reconfigurable hardware platform for 10 Gb/s network packet processing developed by CESNET. The device uses FPGAs dynamic reconfiguration and is designed for rapid development of high-speed network packet processing applications. Received packets can be preprocessed in MAC, can have a timestamp attached, or left intact, then they follow to a set of reconfigurable plug-in modules for packet processing. At the end packets can be also post-processed by MAC layer or left intact. I have designed several hardware plug-in modules for MTPP which allow several passive monitoring tasks at full 10Gb/s speed.

3.2.2 Monitoring modules

The hardware modules for passive network monitoring consists of three standalone modules:

Network statistics. This module stores the statistical information about incoming packets. It provides a number of packets in several predefined groups and a number of bytes in the corresponding group. Groups are divided into: short frames (shorter than 64B), standard frames (64B-1518B), jumbo frames (1518B-9018B) and super jumbo frames (longer than 9018B). Standard frames are further divided into more groups. Statistics for every group is stored into two independent memory banks, which allows the data to be extracted atomically. The module can also track bad CRC checksums or network line errors reported by the physical layer.

Traffic Burst Quantization. This module gathers the statistics about packet bursts. Several groups of burst lengths can be configured along with the maximum inter-burst gap. The inter-frame gap longer than the inter-burst gap divides the required bursts. For every defined burst-length category, the number of bursts, the number of packets in all bursts and the number of bytes in all bursts is stored. Statistics for every group is stored into two independent memory banks, which allows the data to be extracted atomically. This module enables a better statistical overview of the monitored traffic because data bursts mostly contains related data.

Bit Error Rate Tester This module is used for bit error rate (BER) testing of selected lines. The advantage over the standard BER testers is that this module can packetize the test patterns. Test pattern is divided into pre-configured frames which can have their own headers and CRC checksums allowing them to pass through switches and routers. The Ethernet header is mostly sufficient, but the module allows fully customized header

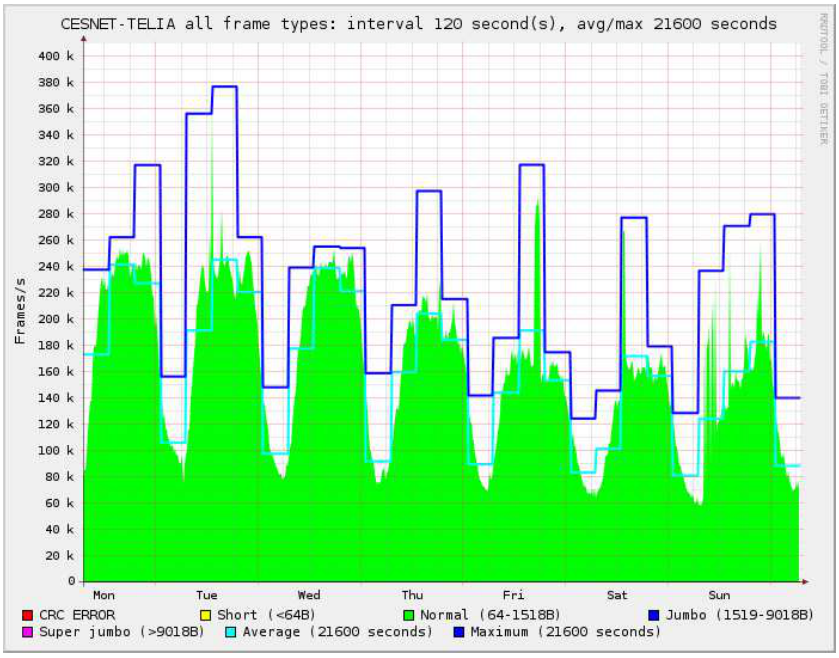

Figure 3.5: Monitoring of traffic dynamics, packet sizes and errors with MTPP support

including higher level layers such as TCP/IP or UDP. This feature enables BER checking of the dedicated lines even with active elements. All active elements in the line must be configured to pass bad CRC frames or recalculate the CRCs or the measurement will be in-accurate caused by the loss of dropped packets.

3.2.3 Summary

I have proposed and implemented hardware plug-in modules for MTPP which enables passive network monitoring on this device. Monitoring modules works at full throughput of MTPP which allows them to process all data at full 10Gb/s at the worst case. The resulting monitoring solution is still being used in the CESNET2 network [16]. The example of monitoring output is shown on Figure 3.5.

3.3 Packet processing architecture for beyond-high-definition video transfers.

I have proposed an extension for the architecture described in the first contribution allowing the transfer and process beyond-high-definition video or several independent channels of high-definition video. This new approach, where video can be processed like network traffic, is an interesting and easily adaptable solution which can greatly improve video

communication over a future Internet. This work was adopted by the Czech Technical Agency project POVROS, whose goal is to move the results of this research to the market.

This contribution was published in papers [D.3] and [D.4] included in this dissertation thesis. Several results of this work are included in the utility patent [P.5].

3.3.1 Background

The architecture for the network packet processing described in the first contribution is based on the series-parallel processing model. The common series-parallel model is also used in video processing applications where maximum throughput and processing strength is desirable. Our goal was to design an architecture for video transfers and processing over the standard Ethernet network. The main goal of the architecture should be good scalability and extendability to higher resolutions and frame-rates along with the maximum throughput and minimum added processing latency in the video channel. Video is preferred in the uncompressed state which offers the ultimate quality and is also commonly used in movie post-production, where the work with compressed video would decrease the quality of the final product.

The series-parallel processing model is already used in video and computer graphics processing. On the other hand, our solution is primarily designed for the Ethernet networks. This presumptions indicate the possibility of using previously proposed architecture, which was intended for network packet processing, for video transfers and processing. If the architecture can be easily adapted for this task, the future development of video processing applications can be greatly reduced using the already designed approach. Plus the use of the network packet processing architecture for video data transfer and processing will open a new way of video applications development focused on the high data volumes with the lowest possible added latency.

3.3.2 The Architecture Overview

The embedded architecture for real-time video transport and processing is based on the scalable architecture for network packet processing described in Section 3.1. This whole architecture operates at network clock domain of attached network interface and can be used for various modular data packet processing. Since a video signal consists of special packets, we can make a simple conversion transporting the video packets to a network clock domain and back. This way we can use a network packet processing architecture for video packet processing.

3.3.3 Video packetization architecture

The HD-SDI interface has a defined structure [1], but not all data needs to be transferred. Video rows include blanking areas (horizontal blanking interval) and a video frame includes blanking video rows (vertical blanking interval). The whole situation is illustrated in Figure 3.6. Blanking areas can contain some secondary information such as audio, encryption

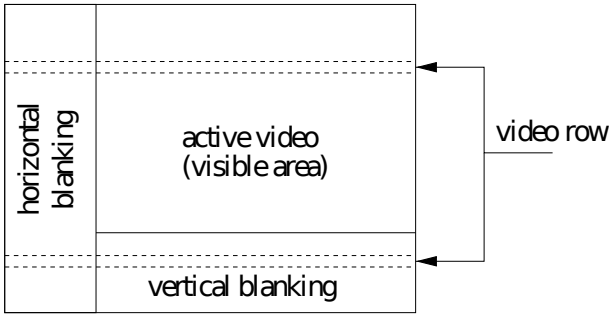

Figure 3.6: Video frame structure transported through HD-SDI channel

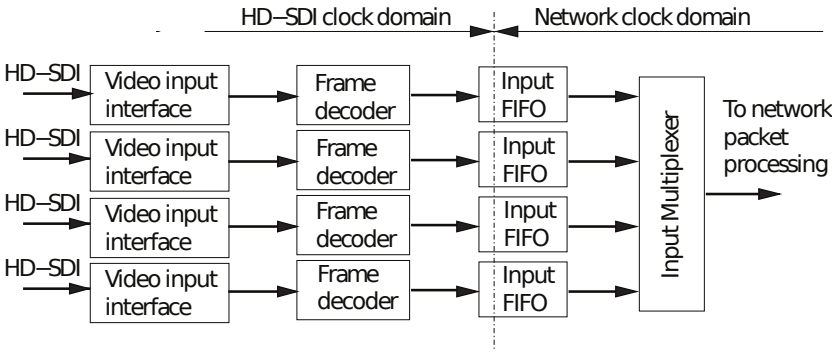

Figure 3.7: Input video packetization and connection to packet processing for 4 channels.

or video format specification, which we can choose to transport or not. The video rows, including all additional data, start with a specified header and end with a CRC checksum; thus we can process them as if they are separated data packets. Those packets can be properly extracted and converted to the Ethernet packets. We can only focus on HD-SDI interface because Dual Link HD-SDI and 3G-SDI links consists of two separate HD-SDI links. We can process only HD-SDI data and optionally bind them together to Dual Link HD-SDI or 3G-SDI interface. One HD-SDI channel can transport only one 2K or HD channel, but for higher resolutions, several identical channels are used.

Video packetization is a conversion between video and network packets, allowing video data to be processed in the network packet processing modules. There is input packetization and output video de-packetization, shown in Figures 3.7 and 3.8.

The input packetization consists of the video input interface and the frame decoder. The video input interface implements low-layer communication with the HD-SDI. The frame decoder extracts video packets, converts them to network packets and attaches headers with video format parameters. The output de-packetization consists of the video frame generator and the video output interface. The frame generator receives network packets and generates valid image to the video output interface based on information contained in network packet headers. Clock domain boundaries are crossed using a dual-port memory configured as a

packet FIFO. The video (de)packetization is located in the HD-SDI clock domain and the network packet processing modules are located in the network clock domain. The example configuration in Figures 3.7 and 3.8 includes four HD-SDI channels. The channels are independent and can be added freely just with a simple modification of the channel multiplexer. This operation can be completely parameterized.

3.3.4 Video Data Packetization

The HD-SDI channel has a bit rate of 1.485 Gb/s. Only six full single HD-SDI channels can be processed with 10 Gb/s architecture. When we need to focus on 4K video, we need to process up to 8 channels for RGB 4K or 3D YCrCb 4K video. When we strip video packets of blanking intervals, we get a bit rate between 1 Gb/s and 1.3 Gb/s depending on a picture resolution and frame rate. This means that the 10 Gbit Ethernet network can transfer up to eight HD-SDI video channels and with some video formats even including additional data, such as audio or encryption information.

The example bitrates of eight channels of selected video formats stripped of blanking

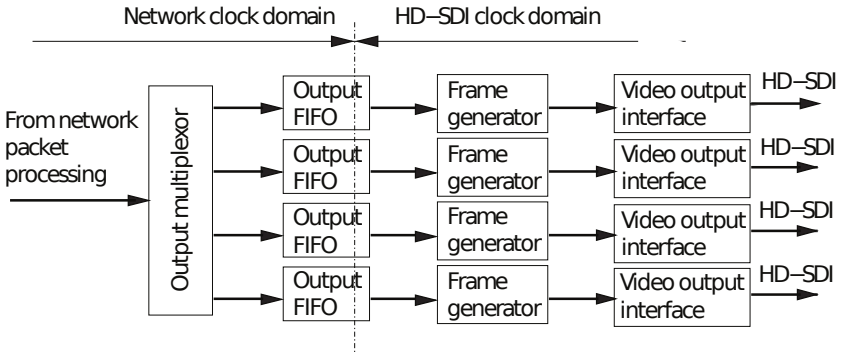

Figure 3.8: Output video packetization and connection to packet processing for 4 channels.

Table 3.1: Video formats bitrates

Format	Bitrate eight channels (Gb/s)	Bitrate one channel (Gb/s)
2K/24	8,7	1,08
1080/24	8,2	1,025
1080/25	8,5	1,06
1080/30	10,4!	1,3
720/50	7,6	0,95
720/60	9,1	1,14

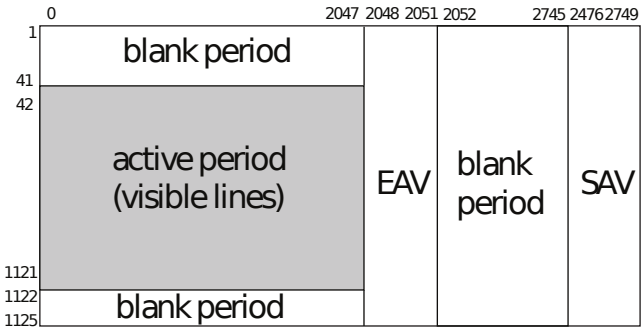

Figure 3.9: Format of the 2K frame.

intervals are summarized in Table 3.1. The 30fps HD formats can be still transferred, but image crop must be applied.

When we need to process additional data such as audio or encryption data, there are several possible ways of mapping the data into network packets. Those techniques can be used only for channels containing any additional data except visible video. Three options are described below. The resulting bit rate for 4K (four quadrants send over 4 or 8 HD-SDI channels) at the physical layer is summarized in 3.2. These rates also include embedded audio and the packet header overhead. We assume 24 frames per second as per the D-Cinema format.

Complete HD-SDI data. One solution is to transfer all HD-SDI data. One complete line of the 2K frame (one quadrant) consists of 2750 samples. One complete frame consists of 1125 lines as described in Figure 3.9. Along with the active video, blanking areas are transported. Blanking areas can consist of additional data like audio channels, additional data for various encoding and encryption methods, etc. [2].

The advantage of this solution is that embedded audio and embedded data (eg. for encryption) are included with no additional effort. The disadvantage is a high resulting bit rate if used on all channels.

Just image & audio bits. An alternative solution is to extract and transfer just the image and audio data. One active line of 2K video consists of 2048 image samples. One active 2K video frame includes 1080 active lines. The advantage of this solution is a lower bit rate, which allows transmission of all subsampling and color depth options, including RGB at 12-bits per color. The disadvantage is more complex data transformation at both the sender and receiver. The embedded audio and embedded data also need to be extracted and transmitted by an additional mechanism.

Active area samples. Another solution is to transfer SDI data in its original format, but just for columns that include image samples, embedded audio or embedded data. Using this solution, we can transfer eight HD-SDI channels over a 10 Gb/s network with simpler

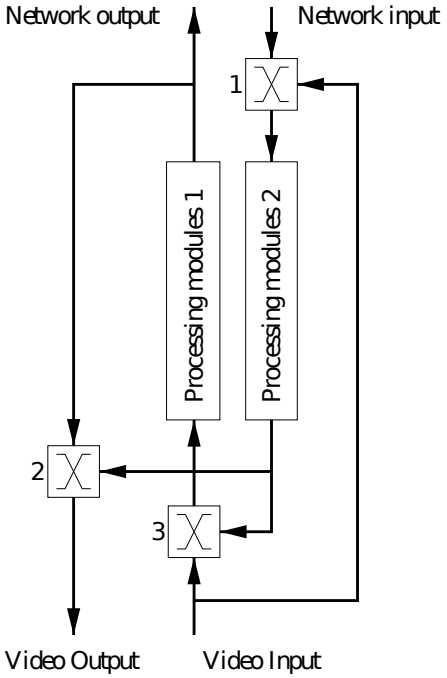

Figure 3.10: Schematic of interconnections in the processing core.

data transformations on the sender and receiver. Still, all subsampling and color depth options are possible.

	10-bit 4:2:2	10-bit RGB	12-bit 4:2:2	12-bit RGB
Complete HD-SDI data	6.00 Gb/s	12.00 Gb/s	12.00 Gb/s	12.00 Gb/s
Just image & audio bits	4.58 Gb/s	6.50 Gb/s	5.22 Gb/s	7.77 Gb/s
Active area samples	4.59 Gb/s	9.18 Gb/s	9.18 Gb/s	9.18 Gb/s

Table 3.2: Bit rates at the physical layer

3.3.5 Network Packet Processing Core

The main processing core, which consists of the reconfigurable processing modules, was divided into two sets of processing modules. A set of switches can be arranged to allow a packet flow between the network and video domains in several ways. A schematic of this interconnection is shown in Figure 3.10. Both module sets are organized according the original proposal described on Figure 3.3. There are several ways of data flow configuration through the processing modules:

- From network input to network output through switch 1, processing modules 2, switch 3 and processing modules 1. All processing modules are dedicated for network to network packet processing.
- From network to video, full-duplex, one set of processing modules for each direction. From network input through switch 1 and processing modules 2 to video output. From video input through switch 3 and processing modules 1 to network output.
- From video input to video output through switch 3, processing modules 2, switch 3, processing modules 1 and switch 2. All processing modules are dedicated for video packet processing.

3.3.6 Summary

I have proposed the embedded architecture for real-time video transport and processing based on the scalable architecture for network packet processing described in Section 3.1. This approach applies the advantages of modular network packet processing to video transport and processing. In the future Internet, more beyond-high-definition video transfers are demanded. and this approach enables easy development and scalability of video transfer and processing solutions.

The resulting architecture was adopted by the Czech Technical Agency project POVROS, whose goal is to move the results of this research to the market. The architecture was implemented and extended by additional techniques, such as low-latency video display, by CESNET to a device MVTP-4K [15].

Chapter 4

Conclusions

This chapter summarizes the contributions of this dissertation thesis and briefly suggests the possible future work.

4.1 Summary

I have proposed the universal scalable network packet processing architecture which focuses on maximum throughput and processing strength. The network packet processing architecture was also enhanced for video transport and processing purposes which also showed its universality.

In a first contribution, I proposed a new scalable and embedded architecture for network packet processing intended for 10-100 Gb/s links. The architecture is designed for processing the network packets at full speed at the worst case without a packet drop, which is possible using the balance between clock rate, data width and pipelining level together with a simple interface between plug-in modules for data processing. The architecture is designed to be distributable between several FPGAs or in a single FPGA, thus allowing the scalability even if current FPGA technology cannot support required resources. The main goals of the architecture are flexibility of target applications, easy extendability based on plug-in modules and good availability for high-speed network research community. The 10 Gb implementation of this architecture was implemented by CESNET in a MTPP (Modular Traffic Processing Platform) device.

The second contribution validates the use of this architecture in passive network monitoring. I have proposed and implemented plug-in modules for the MTPP platform allowing the use of the MTPP for a passive network monitoring. The set of modules consists of standard and burst based statistical modules and modules for bit error rate tests. The resulting monitoring solution is still being used in CESNET2 network.

The third contribution presented the extension for the architecture described in the first contribution allowing the transfer and process beyond-high-definition video or several independent channels of high-definition video. This new approach, where video can be

processed like network traffic, is an interesting and easily adaptable solution which can greatly improve video communication over a future Internet.

The latest work was adopted by the Czech Technical Agency project POVROS, whose goal is to move the results of this research to the market. The proposed architecture was successfully implemented as a device MVTP-4K [15] by CESNET.

4.2 Future Work

The author of the dissertation thesis suggests exploring the following:

- The architecture can be further improved or optimized for more applications which can be demanded in the future.
- The architecture and the implemented platforms MTPP and MVTP-4K are module based, which enables the possibility of enhancing their functionalities. More applications can be designed and implemented to increase the potential of the presented architecture.
- The presented solution can be used as a starting point for the other research projects which require massive parallel data processing. The extendability and scalability of the presented solution can also speedup any future research based on this architecture.

Chapter 5

Publications Included in Thesis

This section contains key publications included in this dissertation thesis. Every section of this Chapter presents one publication and starts with related information about the publication. Formata used in this dissertation thesismay result in visual differences from the published originals

Those documents include grey text that signify that it belongs to other authors and is not included in this dissertation thesis.

5.1 MTPP - Modular Traffic Processing Platform

This paper was published on 12th IEEE Symposium on Design and Diagnostics of Electronic Systems in 2009 [D.1]

MTPP - Modular Traffic Processing Platform

Jiří Halák
CESNET
Email: halak@cesnet.cz

Sven Ubik
CESNET
Email: ubik@cesnet.cz

***Abstract*—High-speed (10 Gb/s and above) network monitoring and traffic processing requires hardware acceleration. Different applications require different functions to be placed in hardware. Current packet capture cards include fixed firmware, which is difficult to extend.**

In this paper we propose an architecture for Modular Traffic Processing Platform (MTPP), which enables end users to easily modify hardware processing without any FPGA development. On the other hand, developers can create new processing modules with much reduced effort thanks to simple module interfaces and isolation of module time constraints.

I. Introduction

In passive network monitoring we directly process real network traffic, as opposed to active monitoring, which uses injected test packets. Passive monitoring is attractive in that it allows to detect properties inherent to real traffic, such as security attacks, traffic dynamics or real packet loss rate. Passive monitoring of high-speed links (10 Gb/s and above) requires hardware acceleration. Different applications can benefit from hardware offloading of different functions.

We are proposing a new architecture for processing network packets at speed of 10Gb/s and above running on FPGA with partial dynamic reconfiguration. The primary benefit of the proposed architecture is that it enables end users to easily modify packet processing in hardware for their applications, without any VHDL programming or FPGA development software. Users can create lots of different firmware variants simply by loading binary firmware modules into available slots according to specification in a text file. Development of new modules is also greatly simplified by separating timing of modules from one another and from the rest of firmware.

A 10 Gb/s version has been implemented and is used in production monitoring. A 40 Gb/s version is in development.

II. The problem and requirements

Current packet capture cards for passive monitoring [1], [2] are supplied by manufacturers with simple firmware that includes packet header filtering or classification. The firmware is fixed and cannot be changed by users to add more functions.

Development kits for custom firmware development are usually expensive and development of new modules is difficult. The programmer needs to study lots of details of existing design, where the new module should be integrated. And development of firmware for high-speed packet processing is particularly difficult, because timing constraints need to be resolved for all components together — the whole design is routed and placed as one piece.

Many passive monitoring applications are implemented using DiMAPI [3] middleware, which enables programmers to work on higher level of abstraction and allows to port applications across multiple packet capture cards. Since little monitoring functionality can be offloaded to current packet capture cards, most packet processing is done in software at limited speed.

We set forth the following requirements to be fullfiled by our architecture:

- R1 - Easy modification of hardware processing by end users without programming
- R2 - Easy development of new firmware modules
- R3 - Full line rate operation at 10 Gb/s with scalability to higher speeds
- R4 - Standalone device for field operation without external PC

In the following chapter we indicate how our architecture maps to the requirements R1 - R4.

III. Architecture

The proposed MTPP (Modular Traffic Processing Platform) architecture addresses the problem and requirements identified

above. *Traffic* in MTPP means network packets, but also a stream of bits, such as in BER (Bit Error Rate) applications. Packets (or stream of bits) can be captured and analyzed, modified or inserted into the network by different modules.

A. Hardware

MTPP can be ported to various hardware. We have tested it on the following hardware:

- Two alternative FPGA boards - Xilinx Virtex-II Pro FF1152 Development Kit and Xilinx Virtex-5 LXT PCI Express Development Kit
- Two alternative optical transceiver boards - AEL1002 board for XFP transceiver and our own board for Xenpak transceiver

An FPGA board and an optical transceiver board are connected by Rocket I/O channels carried over SMA or Infiniband cables. The boards are commercially available except for the Xenpak transceiver board, which was developed ourselves. The Xenpak board is relatively simple, because Xenpak transceiver includes its own 4-channel 10 Gb/s mux/demux and can be connected directly to Rocket I/O channels. MTPP is a standalone device and does not require a PC.

B. Firmware runtime modification (R1)

To allow end users to easily modify hardware functionality without programming, we designed the firmware architecture illustrated in Fig. 2. Packets pass through a sequence of slots connected by registers. Each slot can be filled-in by one packet processing module. A set of modules for basic monitoring functions have been implemented and are available. A user specifies, using a text file, which modules should be loaded to which slots upon the platform startup. Each module can be loaded to any slot, possibly multiple times. Modules can be replaced at any time using partial dynamic reconfiguration. Modules and the configuration file are placed on Compact-Flash memory. In this way a user can configure many different variants of monitoring functionality in firmware and change it at runtime according to application requirements. This also saves space in FPGA, since only functionality required by running applications needs to be loaded into FPGA.

An input MAC block can be enabled before and an output MAC block after the sequence of modules. The MAC block implementes standard functions of the link layer, including checking and generating frame CRC. Additionally, it also computes various frame, byte and error statistics. Two register banks are available for all statistics. One bank is always active and counting. The other bank can be used to read results obtained previously. The two banks can be switched atomically. If the MAC blocks are enables, the processing modules work above the link layer, that is they are processing complete Ethernet frames. If the MAC blocks are disabled, the processing modules work at the XGMII level.

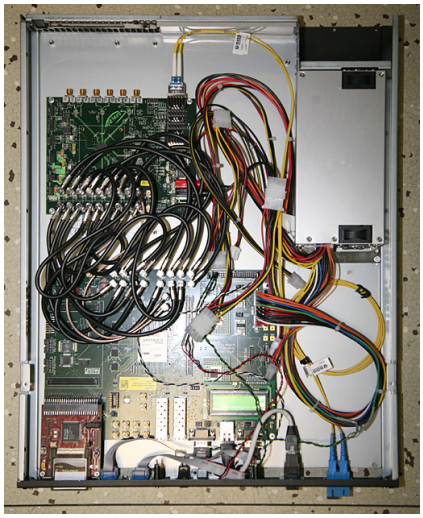

Fig. 1. MTPP in version with Virtex-II Pro FPGA board and XFP transceiver board

The input and output MAC blocks are part of the backplane, which also includes a block to assign packet timestamps (TS) and a block for communication via MDIO interface. This interface can be used, for example, to initialize the Xenpak transceiver before packet reception or transmission.

The following modules are currently available to users:

- INIT-MOD - empty module that passes data to the next slot
- PKT_CNT - packet counter, when placed in multiple slots, it can show the number of packets between phases of processing
- PKT_LOS - emulates specified fixed or pseudorandom packet loss for testing of communication protocols and

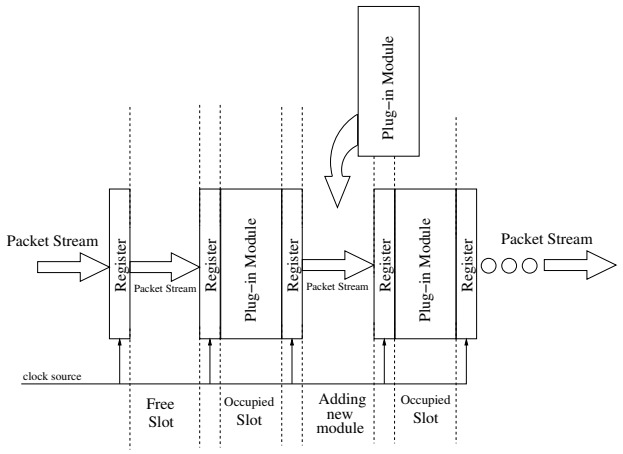

Fig. 2. Firmware architecture

network behaviour

- BERT - Bit Error Rate tester, it can use various predefined or user-defined test patterns
- BURST - quantifies traffic burstiness by classification of traffic burst sizes and measuring the number of frames and bytes in bursts, where inter-burst gap can be freely specified
- PKT_HAL (in development) - IPv4, IPv6, UDP and TCP header analysis to check errors and count statistics

The above modules can be loaded by the user into slots arbitrarily — in any number and in any order. From the user perspective, the backplane with MAC, TS and MDIO blocks appears as a module PLATFORM loaded into special slot 0.

C. Module development (R2)

To simplify development of new modules, the proposed framework includes two features. First, all modules are connected by unified interface, which includes Data Bus (DB) to carry the packet stream, Control Bus (CB) to carry commands between modules and Local Bus (LB) to read and write module registers. Module developers only need to understand this common interface.

Second, the timing constraints of each module are completely separated from other modules and from the backplane that connects the modules. The backplane is already routed and placed inside FPGA and never changes when new modules are developed and inserted into slots. A new module can be synthesized, routed and placed separately from other modules and from the backplane. Slots are connected by registers, which are synchronized with the passing packet stream. The registers are part of the backplane and are placed in fixed positions inside the FPGA. Therefore, the time-critical path is limited to the one module, which is being added. This greatly alleviates timing-related problems when developing firmware for 10 Gb/s and faster processing.

Each module receives 8 bytes of data every clock cycle. The module needs to accept this new data every clock cycle, but the internal latency can be multiple clock cycles, as needed. We currently use six slots available for modules. The number or slots in the backplane can be changed and depends on the size of FPGA and on the space that we wish to be available for one module.

D. Line-rate operation and scalability (R3)

The firmware framework is designed to operate at sustained full line rate of 10 Gb/s at all frame sizes. This data rate could in principle be achieved by various products of data width and clock frequency. A compromise needs to be find since both large data width (many wires) and high clock frequency make the design difficult to synthesize. We use the 64-bit data width and 156,25 MHz clock frequency as the most suitable combination for processing of 10 Gb/s packet streams.

On Virtex-5 family FPGAs, reconfigurable regions can occupy any rectangular areas and miltiple regions can share the same column. This feature can be used to create multiple parallel lanes of module slots (Fig. 3). The number of possible parallel lanes depends on FPGA size, slot size and slot number in one lane. The parallel lanes can be used for several purposes possibly at the same time:

- Input traffic can be split into multiple lanes running the same modules for higher performance. 40 Gb/s processing can be achieved by four 10 Gb/s lanes. We are currently developing a prototype 40 Gb/s platform.
- Lanes can be divided into active, which process packets and inactive, which can be modified by dynamic reconfiguration. Switching between active and inactive lanes allows runtime firmware modification without loss of any packets.
- Traffic can be pre-classified and distributed into multiple lanes running different modules for customized processing dependent on packet class.

E. Reconfiguration control and application integration (R4)

The partial dynamic reconfiguration needs to be controlled by some unit external to the loaded modules. Also, the module registers needs to be written for configuration and read to obtain monitoring results. To minimize the cost, we eliminated the need of an external PC or another FPGA normally required for these tasks. We created a Linux distribution with 2.6 kernel to run on an embedded processor for both purposes (control of reconfiguration and module register access). It can run on PowerPC processor in the FPGAs where this processor is available or on MicroBlaze soft core processor in other FPGAs. Both alternatives have been tested.

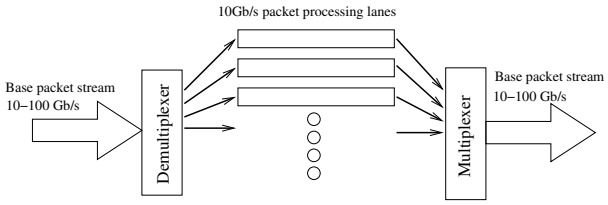

Fig. 3. Packet stream processing in parallel lanes

A user can login to the embedded Linux over SSH connection or from the RS-232 console. The embedded Linux also enables easy implementation of communication by HTTP and SNMP. We have developed a modular software utility called `mtpp` to communicate with the packet processing modules. `mtpp` can list modules currently loaded into slots, list registers available in each module and read or write specified registers, which are used for module configuration and monitoring results storage. All register addresses can be specified by numerical values or by names. Numerical values can be used for unknown types of modules, such as when a new module is being developed. For a known module type, register names can be easily added to `mtpp`. An example of using `mttp` is shown in Fig. 4. Here the occupancy of slots is first listed, all reconfigurable slots include INIT-MOD module, which is an empty module, which passes packets to the next slot. The PLATFORM shown in slot 0 represents the address space of the backplane and can be used to communicate with registers of the input and output MAC. In the second command, registers in slot 0 whose name includes CRC are listed. In the final command, the counter of CRC errors in the first bank of the input MAC is read.

Scripts can be developed to call the `mtpp` utility remotely

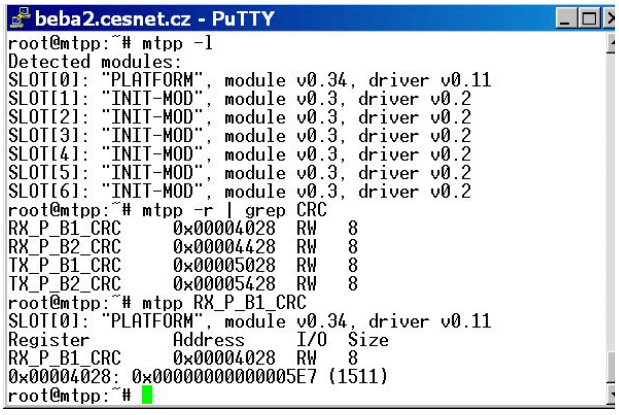

Fig. 4. Example of using the `mtpp` utility

on distributed monitoring boxes from a central server. We use such a script to create graphs of network traffic dynamics, its distribution into packet sizes and number of CRC errors, see Fig. 5. Two prototypes are currently used for production monitoring.

IV. Performance

The prototype hardware with backplane firmware and PKT_CNT modules have been tested by the Ixia 1600 hardware packet generator at sustained 10 Gb/s rate sent in 64-byte Ethernet frames (the shortest legal frames including Ethernet header and CRC). 10^{11} frames have been passed with zero loss and zero CRC error. This is better than 10^{-13} bit error rate, which is comparable to current high-quality optical circuits.

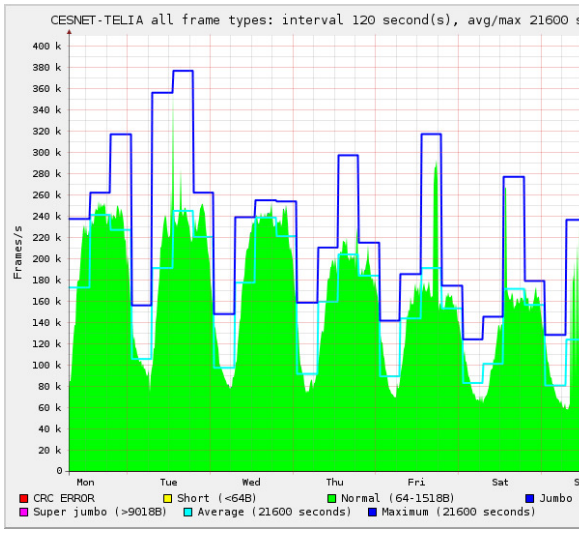

Fig. 5. Monitoring of traffic dynamics, packet sizes and errors with MTPP support

Performance of all developed modules have also been tested at sustained 10 Gb/s rate sent in 64-byte Ethernet frames, although with lower number of frames to limit the test time, since various module configuration need to be tested. All currently available modules provide 100% line-rate performance at 10 Gb/s.

Utilisation of main FPGA resources is shown in Table I. Note that slices in Virtex 5 are approximately twice as large as in Virtex II Pro. The BRAM modules are also different between the two FPGAs. Therefore the type of FPGA should be considered with utilisation figures. *Total* is the size of FPGA, *Backplane* is resources used by the platform backplane, *Slot max.* is theoretical volume of resources for one module slot in case of division into 6 slots and *Slot real* is what we really allocated for each slot. In case of Virtex 5, approximately half

Virtex II Pro (XC2VP50)			
Slices			
Total	Backplane	Slot max.	Slot real
23616	9961	2275	2000
RAMB16s			
Total	Backplane	Slot max.	Slot real
232	51	30	
Virtex 5 (XC5VLXT110T)			
Slices			
Total	Backplane	Slot max.	Slot real
17280	6819	1743	1000
BlockRAM / FIFO			
Total	Backplane	Slot max.	Slot real
148	32	19	

TABLE I
UTILISATION OF FPGA RESOURCES

of slices used for backplane belong to MicroBlaze processor implementation. BRAMs can be used by slots unequally — different number of BRAMs by different slots.

V. RELATED WORK

Our work is based on FPGA hardware and partial reconfiguration of FPGA devices. The FPGA circuits, design flows and partial reconfiguration techniques have been decribed in [4]. The partial reconfiguration is well mastered by Xilinx design tools.

FPX platform [5], [6] has been designed for development and operation of packet processing modules in reprogrammable network hardware, such as routers and switches. FPX uses second FPGA to control dynamic reconfiguration of the main FPGA. It has been implemented to support OC-48 (2.5 Gb/s) ports.

NetCOPE platform [7] is designed for rapid development of packet processing firmware for COMBO [8] cards. NetCOPE is a set of IP cores that provide building blocks for FPGA designs such as I/O blocks, memory controllers or PCI Express support.

Force10 P-series [9] are boxes for 1 Gb/s and 10 Gb/s hardware-accelerated firewalls. The box includes embedded FPGA device that processes a set of static or configurable Snort-like rules.

VI. CONCLUSIONS

The proposed architecture for high-speed network traffic processing in programmable hardware enables end users to easily modify hardware processing without programming. Development of new firmware modules has been greatly simplified by simple module interface and by separating

5.2 Multigigabit network traffic processing

This paper was published on The International Conference on Field Programmable Logic and Applications in 2009 [D.2]

This paper includes a grey text part which is not included in this dissertation thesis. The described technique was not proposed by the author but was only used. This technique was proposed by Petr Žejdl, who is mentioned in Acknowledgments of this paper, and will be part of his dissertation thesis.

MULTIGIGABIT NETWORK TRAFFIC PROCESSING

Jiří Halák

CESNET
email: halak@cesnet.cz

ABSTRACT

High-speed (10 Gb/s and above) network monitoring and traffic processing requires hardware acceleration. Different applications require different functions to be placed in hardware.

This paper presents the architecture and platform for processing network packets at speed of 10 Gb/s. The platform can process the packets at full speed without a packet drop and is easily extendable by plug-in modules that can be changed without the need of shutting down the entire platform.

1. INTRODUCTION

We have developed a new architecture for multigigabit network traffic processing in programmable hardware. The architecture is designed to process network packets at the full speed of 10 Gigabit Ethernet at all packet sizes. The platform is intended for network monitoring, packet generation or packet processing purposes, for instance, real-time video applications.

The platform has been implemented inside a single FPGA chip and tested on commercially available hardware equipment. The idea is to implement universal plug-in modules with basic functionality and the framework to place the modules together and make the desired global functionality. Implementing the whole platform in a single FPGA greatly reduces requirements for a customized hardware.

2. ARCHITECTURE

The architecture is divided into the framework, which is a static part of the design and contains all components that do not change during the runtime, and the reconfigurable plug-in modules that make the main processing and can be changed in runtime. The framework architecture is shown in Fig. 1. The framework with modules works similar as dataflow processing except that it works directly with whole network packets.

The plug-in modules are inserted directly into a packet stream. Each module can make its own operations on a packet stream, compute statistics from the stream, modify packets or generate new packets. The plug-in modules are implemented using the partial dynamic reconfiguration. It allows us to change the functionality without the need of shutting down the whole platform.

The framework includes blocks that are necessary to support the reconfiguration, input and output MAC blocks, a block to assign packet timestamps and a block for communication via MDIO interface. It also includes an embedded processor system with a customized Linux distribution. A software utility running in the embedded Linux can be used to access and reconfigure the plug-in modules.

The current prototypes use six plug-in modules. This number has been selected as compromise. Higher number of plug-in modules is possible, but at the same time larger part of the FPGA would be used by interconnecting logic, leaving less space for modules themselves.

2.1. MAC

The MAC layer is the optional component. It has not the full scale of functions as the standard Ethernet MAC, but it implements most of standard functions and fits perfectly for the modular framework.

The MAC is designed to fit to the Packet stream in the reconfigurable framework. It connects to the original XG-

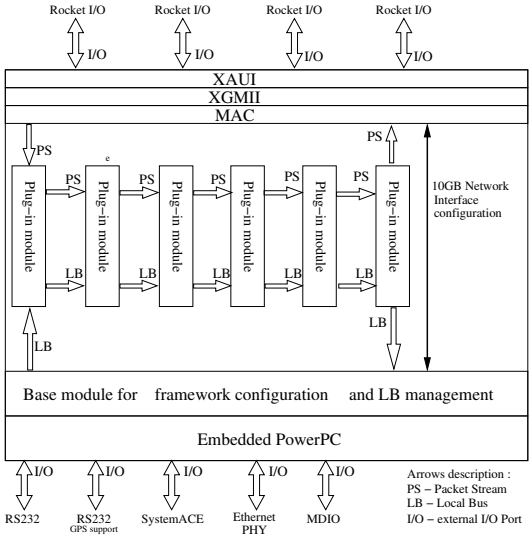

Figure 1: The framework and plug-in modules architecture overview

MII data so it uses the same set of signals as the XGMII interface. The MAC structure is shown in Fig. 2 The main functions of the MAC are:

- XGMII control signal management
- Adding/removing preamble
- Data alignment (first packet byte is always on the first position of a 64-bit data chunk)
- Loopback feature
- CRC checking and generation
- Optional bad packet drop
- Statistical unit for gathering packet statistics

The customized MAC does not have any handshakes to tell the design about processing packets as the regular MAC. It is designed to process the worst case of all 64-byte packets at full speed.

2.2. Timestamps

The timestamp unit assigns timestamps to incoming packets. A timestamp is assigned at the time when the first bit

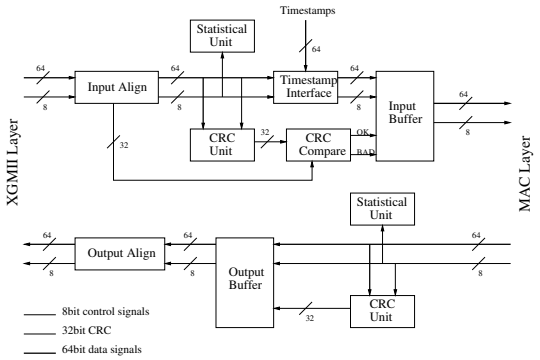

Figure 2: Customized MAC layer

of the Ethernet header is received. The timestamp is 64 bits wide, where the upper 32 bits are used to store seconds and the lower 32 bits to store a fraction of a second. Some applications need only precise time differences between incoming packets and do not need precise absolute values of timestamps. For those applications that need precise absolute packet timestamps or real time information, the timestamp unit can be synchronized to an external PPS (Pulse per second) signal, such as from a GPS receiver. The real time can be set from the embedded Linux, for instance, to the current operating system time.

The simple schematic of the timestamp unit and its realtime evaluation is shown in Fig. 3. The numbers along lines connecting all blocks indicate how many parallel wires are used for each connection. The wires are divided into two groups. The first group are wires counting seconds and the second group are wires for the precision of one second. The main time calculation is done in RealTime Register (RT_REG) that is increased by the value stored in the Incremental Register (INC_REG) every clock cycle. The value of INC_REG is precalculated according to the clock frequency. When no GPS is connected, PPS signal never occurs and timestamps are generated unsynchronized. When GPS is connected, on active PPS signal every second the precision bits are copied to the PPS Register (PPS_REG). Software utility running as a daemon in the embedded Linux is checking the PPS_REG and balancing the value in the INC_REG. The utility tries to reach 0 in PPS_REG. The real time obtained from the GPS can be set in the RT_REG when we need the absolute timestamp generation.

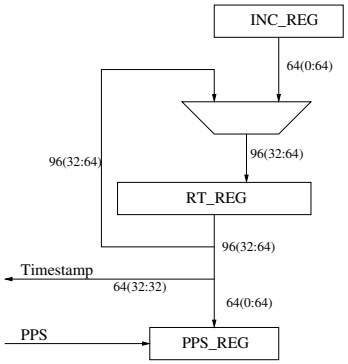

Figure 3: Timestamp unit

3. PLUG-IN MODULES

To allow end users to easily modify hardware functionality without programming, we designed the following plug-in modules architecture (see Fig. 4. Packets pass through a sequence of slots connected by registers. The modules forward the packet stream to the next module. The registers are inserted between the plug-in modules. The registers have the important functions:

- The registers are synchronized by the packet stream clock frequency, so every slot is synchronized with the packet stream and every module receives packet data every clock cycle.
- The number of the modules is limited only by the size of the FPGA device. Adding a new module only increases latency by additional module latency and one clock cycle for additional module slot.
- The critical path of the module cannot be longer then the longest combinational path in the module so the module timing is independent on the global design timing.

A set of modules for basic monitoring functions have been implemented and are available. The modules can be loaded by the user into slots arbitrarily — in any number and in any order.

3.1. Module reconfiguration

FPGA is configured by downloading a bitstream through the JTAG configuration port. The configuration process must be

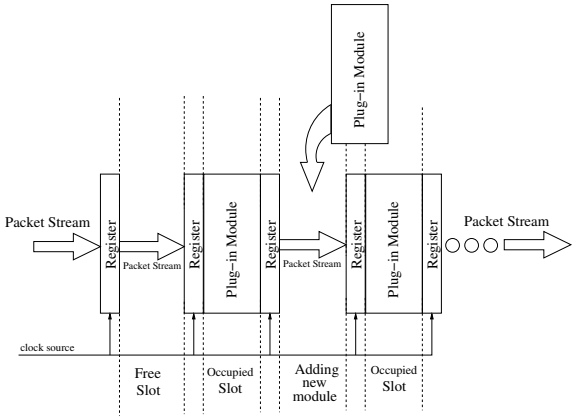

Figure 4: Plug-in modules architecture

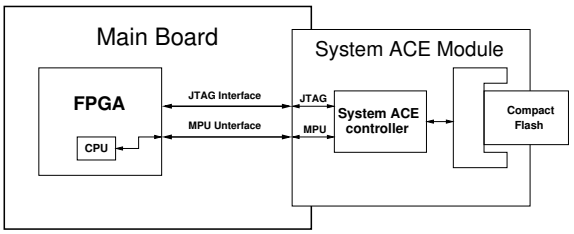

Figure 5: FPGA Configuration using System ACE controller

repeated each time the device is powered up, because FPGA is volatile. To simplify this step XILINX developed a System ACE controller [8] that can configure FPGA using ace files stored in a removable CompactFlash card that connects to the controller. The ace file is a package compiled from original bistream(s) and additional information used to initialize embedded RAMs and CPU.

The controller is connected to FPGA through MPU interface, which is accessible from the embedded Linux. The configuration process is controlled by our modified Linux kernel module xilinx_sysace. The original module supports only access to CompactFlash. The configuration process is depicted in Figure 5.

To manage device reconfiguration without the need of shutting down the whole platform, the partial dynamic reconfiguration process is used. The plug-in modules are downloaded into the FPGA on demand, the rest of the system remains intact.

The system ACE chip allows to specify several partial configurations. Each configuration has an unique address. Reconfigurable modules are stored in a form of ace files and assigned to the particular addresses.

3.2. Module development

To simplify development of new modules, the proposed framework includes two features. First, all modules are connected by a unified interface, which includes Data Bus (DB) to carry the packet stream, Control Bus (CB) to carry commands between modules and Local Bus (LB) to read and write module registers. Module developers only need to understand this common interface.

Second, the timing constraints of each module are completely separated from other modules and from the framework backplane that connects the modules. The framework is already routed and placed inside FPGA and never changes when new modules are developed and inserted into slots. When slot registers are placed to the specific locations on the border or inside the FPGA areas dedicated for the specific plug-in module, the framework design without the modules can be placed and routed into the FPGA without the need of modules implementation. A new module can be synthesized, routed and placed separately from other modules and from the framework. The plug-in modules can be implemented independently and do not need to be placed, routed and tested together with the rest of the design. The module can be loaded to the device or changed inside the device without the need to reprogram all the design inside the FPGA device. This greatly alleviates timing-related problems when developing firmware for 10 Gb/s and faster processing.

4. REMOTE ACCESS

The platform was designed to work as a fully standalone device. It does not a require a PC. It offers many ways of remote access. The embedded Linux supports terminal access, SSH, HTTP server and SNMP. HTTP server hosts web base configuration and access environment. SNMP enables all modules registers to be accessed via MIB (Management Information Base) objects.

The platform has its own branch in the *enterprises* subtree of MIB (Management Information Base):

```
iso.org.dod.internet.private.enterprises.cesnet.mtpp
```

Since multiple instances of the same module type can be loaded to multiple slots, the modules are represented by an array of objects in MIB. The object index counts instances of a given module type, rather than slots. For example, the object *pktCntr.2* is the second module of type PKT_CNTR. This module can be loaded to any slot after the first module of type PKT_CNTR, for example, to slot 5.

5. PROTOTYPES

The platform can be ported to various hardware. We have tested it on the following hardware:

- Two alternative FPGA boards - Xilinx Virtex-II Pro FF1152 Development Kit and Xilinx Virtex-5 LXT PCI Express Development Kit
- Two alternative optical transceiver boards - AEL1002 board for XFP transceiver and our own board for Xenpak transceiver

An FPGA board and an optical transceiver board are connected by Rocket I/O channels carried over SMA or Infiniband cables. The boards are commercially available except for the Xenpak transceiver board, which was developed ourselves. The Xenpak board is relatively simple, because Xenpak transceiver includes its own 4-channel 10 Gb/s mux / demux and can be connected directly to Rocket I/O channels. The prototypes are standalone devices and do not require a PC.

Several reconfigurable modules have been developed including PKT_CNTR (packet counter, can be used in multiple slots to count packets between various processing), PKT_GEN (packet generator), PKT_LOSS (loss emulator), BURST (traffic burst quantification) and BERT (bit error rate test). All modules have been tested by a 10 Gb/s hardware packet generator and analyzer at full speed for at least 10^{11} of 64-byte packets.

6. RELATED WORK

The FPGA circuits, design flows and partial reconfiguration techniques have been described in [3]. The partial reconfiguration is well mastered by Xilinx design tools.

FPX platform [4] has been designed for development and operation of packet processing modules in reprogrammable network hardware, such as routers and switches. FPX uses second FPGA to control dynamic reconfiguration of the main FPGA. It has been implemented to support OC-48 (2.5 Gb/s) ports.

NetCOPE platform [5] is designed for rapid development of packet processing firmware for COMBO [6] cards. NetCOPE is a set of IP cores that provide building blocks for FPGA designs such as I/O blocks, memory controllers or PCI Express support.

Force10 P-series [7] are boxes for 1 Gb/s and 10 Gb/s hardware-accelerated firewalls. The box includes embedded FPGA device that processes a set of static or configurable Snort-like rules.

7. CONCLUSIONS

We have developed a new architecture for multigigabit network traffic processing in programmable hardware. The architecture is designed to process network packets at full speed at the worst case without a packet drop. The architecture consists of hardware accelerator based on plug-in modules and an embedded processor system. Development of new hardware plug-in modules has been greatly simplified by simple module interface and by separating time constraints of modules from one another and from the backplane.

The implemented prototypes have been tested for 10 Gb/s line-rate operation and are used in production monitoring. Parallel module lanes enable scalability to higher speeds. A 40 Gb/s prototype is currently in development.

Acknowledgement

The author would like to thank to Petr Žejdl for significant software contributions and useful advice when developing VHDL code of the MTPP prototype.

8. REFERENCES

[1] DAG cards, Endace Corporation, `www.endace.com`.

[2] Napatech cards, `www.napatech.com`.

[3] L. House, J. Hill, and C. Maxfield, *The Design Warrior's Guide to FPGAs*, Mentor Graphics Corporation and Xilinx, Inc., 2004. ISBN: 0-7506-7604-3.

[4] E. L. Horta, J. W. Lockwood, D. E. Taylor, and D. Parlour, *Dynamic Hardware Plugins in an FPGA With Partial Runtime Reconfiguration*, Design Automation Conference (DAC), June 2002, New Orleans, Louisiana, USA, pp. 343-348, ISBN ISSN: 0738-100X, 1-58113-461-4.

[5] T. Martinek, M. Kosek, *NetCOPE: Platform for Rapid Development of Network Applications*, 11th IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems, 2008, DDECS 2008.

[6] COMBO cards, Invea-Tech company, `www.invea.cz`.

[7] Force10 P-Series, `www.force10networks.com/products/pseries.asp`.

[8] Xilinx Inc., System ACE: The drop-in configuration solution for FPGA configuration needs. `http://www.xilinx.com/`

[9] Xilinx Inc., HWICAP core, `http://www.xilinx.com/products/ipcenter/opb_hwicap.htm`.

5.3 Real-time long-distance transfer of uncompressed 4K video for remote collaboration

This paper was published in Future Generation Computer Systems Journal in 2010 [D.3]
Author list:

- Jiří Halák
- Petr Žejdl
- Sven Ubik
- Michal Krsek
- Felix Nevřela

Real-time long-distance transfers of uncompressed 4K video for remote collaboration

Jiří Halák[a], Michal Krsek[a], Sven Ubik[a], Petr Žejdl[a], Felix Nevřela[b]

[a] *CESNET, Zikova 4, Prague 6, Czech Republic*
[b] *CINEPOST, Kříženeckého Nám, 5/322, Prague 5, Czech Republic*

Abstract

Better than high-definition resolution video content (such as 4K) is already used in some areas, such as scientific visualization or film post-production. Effective collaboration in these areas requires real-time transfers of such video content. Two main technical issues are high-data volume and time synchronization when transferring over an asynchronous network such as the current Internet.

In this article we discuss design options for a real-time long-distance uncompressed 4K video transfer system. We present our practical experience with such transfers and show how they can be used to increase productivity in film post-production, as an application example.

Keywords: 4K video, uncompressed video, network video transfers, remote collaboration, color grading

1. Introduction

Video transfers are an expected driver application area of the future Internet. The picture resolution has been increasing over the time. Better than high-definition resolution video (such as 4K) is already used in some areas, such as scientific visualization or film industry. For ultimate quality, required for instance in a color grading process in film post-production, work with a signal that has not been compressed is preferable. Productivity of a

Email addresses: halak@cesnet.cz (Jiří Halák), michalk@cesnet.cz (Michal Krsek), ubik@cesnet.cz (Sven Ubik), zejdlp@cesnet.cz (Petr Žejdl), felix.nevrela@cinepost.cz (Felix Nevřela)

distributed team can be significantly increased when the video signal can be transferred over the network in real time, to discuss and perform processing of video content. Two main technical issues are high-data volume and time synchronization when transferring over an asynchronous network such as the current Internet.

The data volume ranges from 4.2 Gb/s for 4:2:2 subsampling [1], 10-bit color depth and 24 frames per second to over 9.6 Gb/s for RGB (no subsampling), 12-bit color depth and 30 frames per second. Overhead in packet headers needs to be added.

Real-time video streaming requires that the speed of rendering on the receiver side matches the rate of video source on the sender side. When the sender and receiver are connected over an asynchronous network, such as Ethernet, the receiver cannot directly synchronize its clock with the sender.

Therefore, the receiver needs to implement a technique that adjusts the speed of data arrival and maintains a continuous stream of video and audio data to the rendering device, preferably with minimal added latency for remote interactive applications.

We implemented the proposed architecture in a device called MVTP-4K (Modular Video Transfer Platform).

The structure of this paper is as follows. In chapter 2 we summarize the main requirements on the system for real-time high-definition video transfers. The proposed system architecture is described in chapter 3. Our practical experience is described in chapter 4. Related work is referred in chapter 5 and our conclusions and thought about future directions are provided in chapter 6.

2. Requirements and design constraints

In order to satisfy the needs of targeted applications, we set the following set of requirements to be fulfilled by the proposed solution:

- Real-time transfer of uncompressed 4K video content over a long-distance asynchronous network with no observable visual impairments
- Support of at least 24 frames per second, 12-bit color depth and no subsampling (RGB)
- Video input and output in multiple HD-SDI channels

- 10 Gigabit Ethernet network interface
- Pixel correct synchronization between picture quadrants
- Audio synchronized with video
- Small added latency (see below)

The rationale behind these requirements, in addition to those already mentioned in chapter 1, is as follows.

The use of HD-SDI channels for transmission of high definition video signal is now a common industry practice. Three variants are currently in use — HD-SDI [2], dual-link HD-SDI [3] and 3G-SDI [4]. Mapping of digital video signals into these channels is specified for HD (1920x1080) [5, 3], 2K (2048x1080, D-Cinema operational level 2 and 3) [3] and for other lower resolution formats. The 4K format (4096x2160, D-Cinema operational level 1 or 3840x2160, UHDTV1 [6]) is typically transferred in four quadrants, each in 2K or HD format carried over a separate HD-SDI channel.

Asynchronous Ethernet technology is currently more frequently deployed in 10 Gb/s networks than synchronous SONET/SDH, due to its simplicity and therefore lower cost. Ethernet will likely play even more important role in future 40 Gb/s and 100 Gb/s networks, although often enveloped in a synchronous Optical Transport Network (OTN) [7].

The rule of thumb is that the maximum acceptable one-way latency for remote interactive work is around 150 ms. This latency can easily be caused just by network propagation delay. Therefore, the video transfer system should add minimal further latency. Buffering of one frame at 24 frames per second adds 41 milliseconds.

3. Architecture

3.1. Hardware

Real-time processing of multi-gigabit data rates is difficult on PC-based platforms with standard operating systems not designed for real-time operation. We were looking for a real-time design that is scalable to higher data rates (such as for 8K or UHDTV2 format), higher network speeds (such as 40 and 100 Gb/s) and that can be integrated with commonly requested video processing functions, such as encryption, transcoding or compression. This implies highly parallel and trully real-time data paths. DSP (Digital Signal

Processor) and FPGA (Field Programmable Gate Arrays) are the standard technologies in this area. We selected FPGA, due to high data bandwidth and no requirements on floating-point operations.

The selected FPGA circuit needs to have a sufficient number of fast channels for input and output of the HD-SDI data. The sustained speed needs to be 1.485 Gb/s for HD-SDI and 2.97 Gb/s for 3G-SDI. For the 4K format we need four or eight HD-SDI interfaces depending on the exact format and interface speed. Xilinx, Altera and Lattice all have FPGA circuits that satisfy these requirements. We selected Xilinx Virtex 5 LXT series that allowed us to reuse some design blocks that we have developed for network monitoring devices.

The hardware architecture is shown in Fig. 1. The HD-SDI board converts electrical levels and timing between input and output HD-SDI channels on one side and Virtex RocketIO channels on the other side. The FPGA board processes the video signal and connects to an optical transceiver, which converts electrical and optical signals for network transmission.

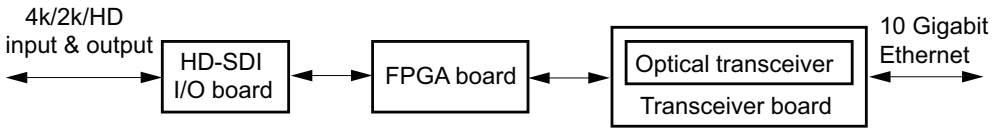

Figure 1: Hardware architecture

This architecture allows operation in several modes illustrated in Fig. 2. Two devices can be used to transfer video content over a network or a single device can be used as a video processor or a network processor.

3.2. Packetization

There are several possible ways of mapping the HD-SDI data into network packets. Three options are described below. The resulting bit rate for 4K (all four quadrants) at the physical layer is summarized in Table 2. These rates include also embedded audio and the packet header overhead. We assume 24 frames per second as per the D-Cinema format.

Complete HD-SDI data. One solution is to transfer all HD-SDI data. One complete line of the 2K frame (one quadrant) consists of 2750 samples. One complete frame consists of 1125 lines (Fig. 3). The 1.485 Gb/s data stream is divided into 10-bit words. At 24 frames per second, there are two 10-bit words

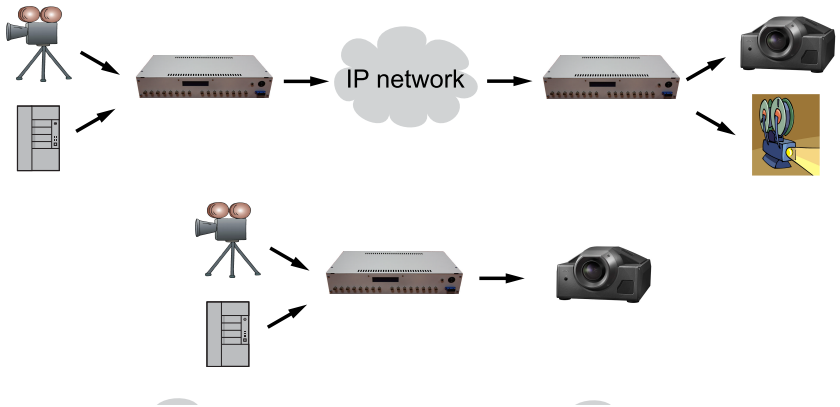

Figure 2: Operation modes — network transfer (above), video processor (center), network processor (bottom)

available for one sample. With 10-bit color depth and 4:2:2 subsampling, one word can carry the luma substream (Y') and the other word can carry the color substream ($C'_R C'_B$) [2]. Other color depth and subsampling options require the use of dual HD-SDI or 3G-SDI, both requiring twice the network bandwidth.

The advantage of this solution is that embedded audio and embedded data are included with no additional effort. The disadvantage is high resulting bit rate. We can only transmit four HD-SDI channels over a 10 Gb/s network. Therefore, only 4:2:2 subsampling and 10-bit color depth is possible.

Just image & audio bits. An alternative solution is to extract and transfer just the image and audio bits. One active line consists of 2048 image samples. One frame includes 1080 active lines. The number of bits per active sample depends on subsampling and color depth. When dual-link HD-SDI is used, one sample must by extracted from both channels, where it is distributed.

The advantage of this solution is a lower bit rate, which allows transmission of all subsampling and color depth options, including RGB at 12-bits per color. The disadvantage is more complex data transformation on both the sender and receiver. The embedded audio and embedded data also need to be extracted and transmitted by an additional mechanism.

Active area samples. Another solution is to transfer HD-SDI data in their original format, but just for columns that include image samples, embedded audio or embedded data. Using this solution, we can transfer eight HD-SDI channels over a 10 Gb/s network with simpler data transformations on the sender and receiver. Still, all subsampling and color depth options are possible. This is the solution used in the current version of our platform.

Embedded audio can be located in up to 268 words of 10 bits per one line, including ancillary (blank period) lines [8], increasing the number of bytes to transfer per line by 335. In the ancillary lines, just the embedded audio needs to be transferred.

In order to support different formats, such as HD and 2K (per quadrant), each line also includes 8 bytes of Active Video Stream Parameters described in table 1.

Parameter	Bit size	Description
hblank	12	Pixels of horizontal blank period
avideo	13	Pixels of horizontal active period
alines_min	12	Number of first active period line
alines_max	12	Number of first blank period line
total_lines	12	Total number of lines
"00"	2	Alignment bits
first_row	1	1 for the first line of the frame, 0 otherwise

Table 1: Active Video Stream Parameters

The required number of bytes per line approximately corresponds with the maximum payload size of one Ethernet jumbo frame. Therefore we chose to pack one line of one quadrant per one Ethernet frame.

Each frame adds an overhead of at least 46 bytes (8 bytes for UDP header, 20 bytes for IP header, 14 bytes for Ethernet header and 4 bytes for Ethernet CRC). More bytes may be required for Ethernet VLANs, MPLS, IP options or higher layer protocol headers. In order to calculate bit rate at the physical layer, additional 20 byte intervals per packet need to be included for the Ethernet preamble and the Inter Frame Gap (IFG).

If we send lines of four quadrants in a round-robin fashion, the quadrant synchronization is guaranteed. However, we need to care for the sender - receiver synchronization and resolve lost or out of time packets. We discuss these issues in the following sections.

	10-bit 4:2:2	10-bit RGB	12-bit 4:2:2	12-bit RGB
Complete HD-SDI data	6.00 Gb/s	12.00 Gb/s	12.00 Gb/s	12.00 Gb/s
Just image & audio bits	4.58 Gb/s	6.50 Gb/s	5.22 Gb/s	7.77 Gb/s
Active area samples	4.59 Gb/s	9.18 Gb/s	9.18 Gb/s	9.18 Gb/s

Table 2: Bit rates at the physical layer

3.3. Rendering rate adaptation

There are two sources of rate difference between the data arriving to the receiver from the network and the data to be sent to the rendering device. First, the internal clock of the sender can be different from the internal clock of the receiver, within a tolerance permitted by the transmission protocol. The HD-SDI clock rate is specified as 1.485 Gb/s, 1.485/1.001 Gb/s, 2.97 Gb/s or 2.97/1.001 Gb/s. Most devices indicate acceptable tolerance of 100 ppm. Second, network delay variation [9] (jitter) can be introduced due to network traffic conditions. The jitter needs to be accommodated for by the receiver FIFO memory.

Several alternative techniques can be used to adjust the sending and rendering rate over an asynchronous network: receiver feedback, frame buffer, blank period adjustments and rendering clock adjustments.

Receiver feedback. The receiver can send feedback to the sender requesting sending rate adjustments (flow control). This technique is used in window-based transport layer protocols, such as TCP or in some link layer protocols, such as PAUSE frames in Ethernet.

The adjusted data rate appears at the receiver after round-trip time (RTT), which is typically hundreds of milliseconds on the long-distance Internet connections. In high-definition video transfers, this technique would require a large receiver FIFO memory, which would not fit in FPGA resources and which would introduce high added latency. Therefore we decided to use another technique.

Frame buffer. This technique requires the receiver to have a FIFO memory large enough to accommodate at least three complete frames. Then the rendering device can be driven by a fixed clock source in the receiver. The

top of the FIFO memory holds the frame that is being sent to the rendering device. When sending of this frame completes, the next FIFO position should include the next frame to be sent. The third (and possibly further) FIFO positions can hold more data depending on network jitter.

After a long period when the skew between the sender and receiver clock rates causes that the frame in the second FIFO position is not completely available in time, the frame in the first FIFO position is rendered again. Similarly, if the FIFO cannot accommodate more incoming data, one frame is dropped. Of course, this solution is not suitable when audio signals are transferred along with video signals.

In the worst case when both the sender and receiver clocks are shifted by 100 ppm in the opposite directions, the error can expand to the whole frame in $1/200 * 10^{-6} = 5000$ frames. At 24 frames per second, it equals to approx. 3.5 minutes. A large FIFO memory can accommodate a shift by several frames, at the expense of latency.

Unfortunately, we could not use this solution due to limited memory resources in the FPGA circuit.

Blank period adjustments. Most digital video transfer formats include a blank period in addition to an active period (visible lines) in each frame. The blank period is used for embedded audio, ancillary data and synchronization. The structure of the 2K frame [10] (or one quadrant of the 4K frame) as transmitted using an HD-SDI channel is shown in Fig. 3. Each frame consists of 1125 lines, which include 1080 active lines. Each line consists of 2750 samples, which include 2048 active samples. The active period is shown in gray color. The SAV (start of active video) and EAV (end of active video) are synchronizing sequences present in each line.

One simple technique to adjust the rendering speed to the rate of incoming data is to add or remove some samples or lines in the blank period. While this technique does not comply with SMPTE standards, we tested that it works with some devices. Since we required compliance with standards, we also decided not to use this technique in our final solution.

Rendering clock adjustments. Adjusting the clock of HD-SDI channels between the receiver and the rendering device within the permitted tolerance gives the receiver some level of adaptation to the rate of incoming data. This solution requires a tunable oscillator and a closed-loop controller in the receiver. It is a solution that we implemented in our device.

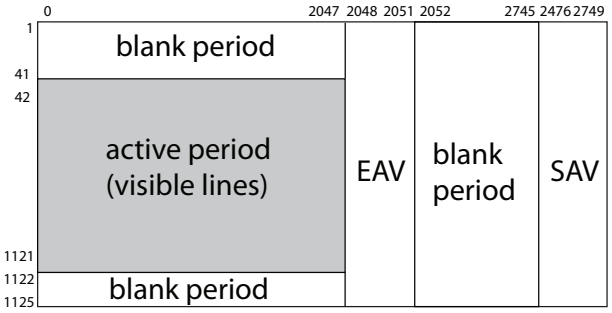

Figure 3: Format of the 2K frame

3.4. Receiver controller

In order to adjust the rendering clock to the data source rate, we used a common PI controller [11]. The complete receiver control structure is shown in Fig. 4. The FIFO input clock is driven by data arriving from the network. The FIFO output clock is driven by a clock generator, which is tuned by the PI controller.

The bare adjusting of the rendering clock would stabilize the rendered picture, but it does not guarantee the frame alignment. The top of the picture could appear anywhere on the screen.

Therefore, we used as the feedback variable to drive the controller the delay between the time when the first active line enters the receiver FIFO memory and the time when the receiver starts to send the first active line to the rendering device measured in clock cycles.

This delay is written to a register for each frame and sampled by the regulator every 200 ms. The controller then uses a weighted moving average of eight last samples. The purpose of this average is to smooth out fluctuations in FIFO occupancy due to network jitter.

One may argue why we do not use the time when the first active line leaves the FIFO instead. This would actually introduce instability, which we confirmed in our tests. If the FIFO occupancy increases due to the receiver clock being slower than the sender clock, packet loss at the FIFO input will increase as a result of network jitter. This will increase probability that the first active line (or any other line used by the regulator) is also lost. And this will cause the regulator slower to react thereby allowing further increase in the FIFO occupancy and higher packet loss.

The feedback variable is compared with the desired value, which is the

number of cycles it takes for data to pass through the receiver FIFO at a desired average occupation added to the latency of the output video processor. This value should be optimized empirically for maximum picture stability, see section 4. The PI controller then produces adjustments to the clock generator frequency.

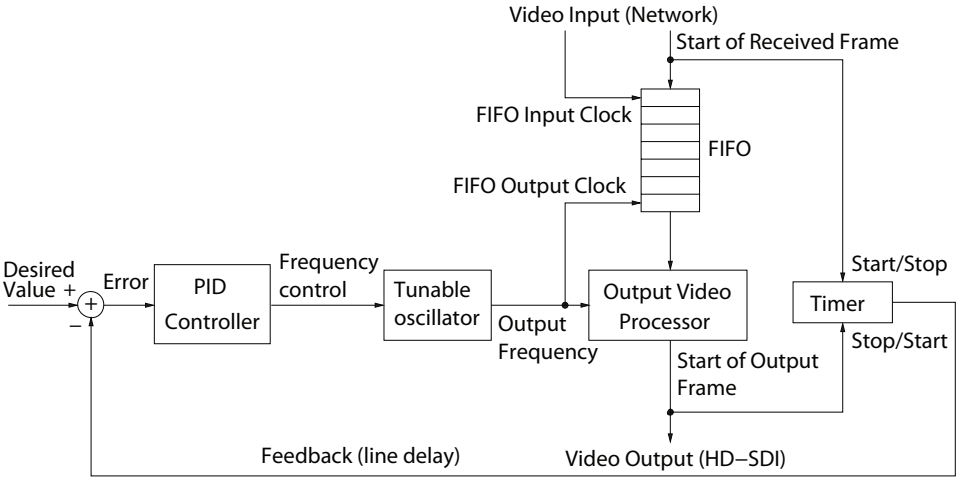

Figure 4: Receiver control structure

3.4.1. Out-of-range adjustments

The receiver FIFO can accommodate some fluctuations in data arrivals on its input end. But when a line is not available at the output end when it should be rendered, the receiver sends another line to the rendering device, because the signal on the synchronous HD-SDI channels cannot be stopped. This is either some line, which happens to be at the FIFO output end or a copy of the previous line if FIFO is empty. When the newly arriving line does not fit into the receiver FIFO, it is dropped. These events causes the whole picture to roll up or down. The controller will see a large value of the error variable and try to compensate quickly.

4. Practical experience

We first tested our device in a laboratory by transferring patterns from the Gefen 2K generator over a short distance. The signal from the generator

was passively distributed to all four quadrants. On the receiver side, an HD-SDI to HDMI convertor and scaler with an HD monitor were used to check the signal.

As the next step, we transferred the 4K signal over a long-distance loop from Prague to Chicago and back to Prague.

Finally, we demonstrated a real use of the technology for real-time remote color grading of uncompressed 4K video content between continents on the Cinegrid 2009 workshop.

4.1. Laboratory tests

We implemented the proposed architecture on the Xilinx Virtex 5 FPGA device. The software part runs on an embedded Linux, which runs inside the FPGA using the softcore Microblaze [12] processor. The Linux environment provides access to firmware variables and runs the PI controller process, which communicates with a tunable oscillator. The prototype version used an external laboratory clock generator with an RS-232 interface. The current version uses a built-in SiLabs tunable oscillator with digital interface. It is more convenient to generate commands for the tunable oscillator in software, rather than hardware.

Without a controller, the picture started to roll down or up after about 3-4 seconds even when communicating over a single 10 Gigabit Ethernet segment. This was due to the difference and wander of the sender and receiver clocks. When the controller was activated, the picture was perfectly stable. The controller reads the feedback variable, computes the error variable and generates commands for the clock generator once per 200 ms. This frequency is well sufficient to compensate the drift between the sender and receiver clocks if they are within 100 ppm tolerance. The highest picture stability was provided when the desired value of the PI regulator was set to approximately 900. Since one clock corresponds to one video sample on the line, it is approximately 1/3 line. Most fluctuations in data arrivals were due to network jitter, which was handled by the FIFO memory.

A few lines of the PI controller output are shown in Fig. 5. Each lines indicates one 200 ms adjustment. The numbers show from left to right: feedback variable in clocks, moving average of feedback variable in clocks, change of this average between two samples, resulting frequency to be set on the clock generator in Hz and change of this frequency in Hz.

```
raster=   779 avgf=  1034 speed\_avgf= -45.20 freq=148496064 df=  21.26
raster=  1778 avgf=  1220 speed\_avgf=  12.63 freq=148496048 df=  -9.51
raster=   750 avgf=  1102 speed\_avgf= -19.89 freq=148496048 df=   7.92
```

Figure 5: Example of PI controller output

4.2. GLIF loopback test

We tested our device over the GLIF (Global Lambda Interchange Facility) network in an L3 loop Prague - Chicago - Prague. The air distance was approx. 9072 miles or 14602 kilometers. The setup is illustrated in Fig. 6. In this configuration, network jitter was much higher and frequently exceeded the FIFO capacity. By duplicating or removing single rows complemented with rendering clock adjustments by the controller, the impairments were not subjectively observable.

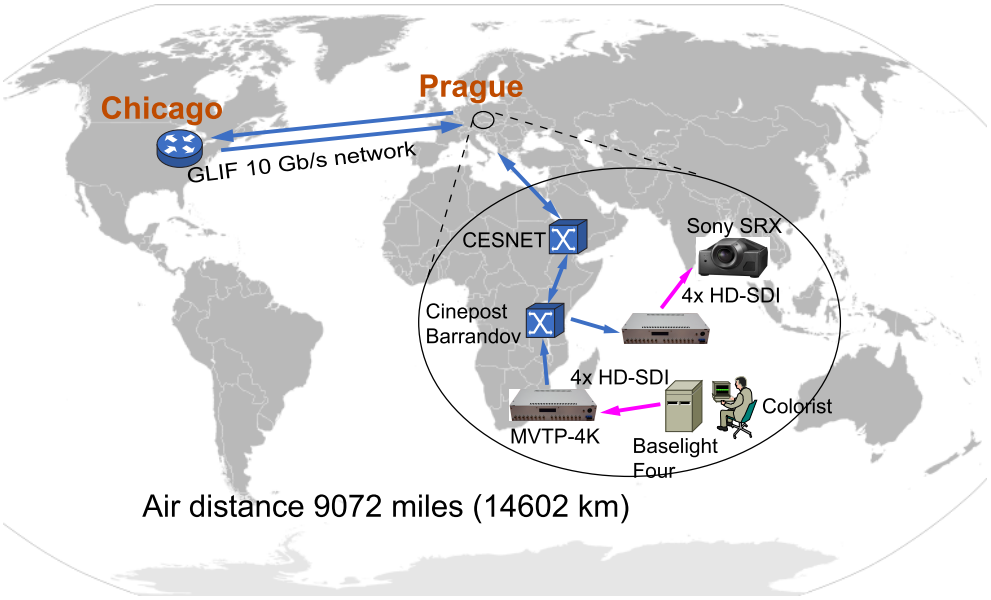

Figure 6: Remote loopback test (Prague-Chicago-Prague loop, 14602 kilometers)

4.3. CineGrid demonstration - Remote color grading

On the CineGrid 2009 workshop we demonstrated the use of the described technology for real-time remote color grading of uncompressed 4K video between continents over a distance of more than 6200 miles (10000 km). Cine-

Grid is a not-for-profit membership organization whose aim is to build a multidisciplinary community that promotes research, development and adoption of technologies for exchange of high-quality digital media over high-speed networks.

The aim of the demonstration was presentation of remote collaboration during the color grading process, where a grading system and its operator (the colorist) were in Prague, while the Director of Photography (DoP), who instructed the colorist what to do and checked the results, was in San Diego. The current state of the art in the movie industry is to have all persons on the same place. This leads to a non-effective allocation of resources during the post-production phase, where the key people are often highly distributed across continents and spend a lot of non-productive time while traveling.

The demonstration setup is illustrated in Fig. 7. The 4K content was streamed from the Baselight Four in the Cinepost corporation at Barrandov Studios in Prague. This content was transferred using two MVTP-4K devices over the GLIF network from Prague over Chicago to the University of California in San Diego (UCSD), where the Cinegrid workshop took place.

The connection over the GLIF network consisted of a series of 10 Gb/s circuits inter-connected by an L3 router in Chicago and several L2 switches along the route. The used VLAN was not completely dedicated for the demonstration and there was a small volume of other background traffic.

Additionally, there was also a bidirectional LifeSize videoconference connection between Cinepost and the Cinegrid venue. The director of photography (DoP) at the Cinegrid workshop used this videoconference to discuss color grading of the 4K content with the colorist person in Cinepost, who performed these corrections in real-time on the 4K content streamed to the Cinegrid venue. The overall appearance in the UCSD lecture hall is shown in Fig. 8.

5. Related work

Net Insight's [13] Nimbra 600 series switch can transport 8x HD-SDI or 3G-SDI channels over a synchronous SONET/SDH network. There are several commercially available solutions for transport of compressed 4K video over the Internet, for example NTT Electronics [14] ES8000/DS8000 4K MPEG-2 encoder/decoder complemented with NA5000 IP interface unit or intoPIX's [15] system of PRISTINE PCI-E FPGA boards and JPEG 2000 IP cores.

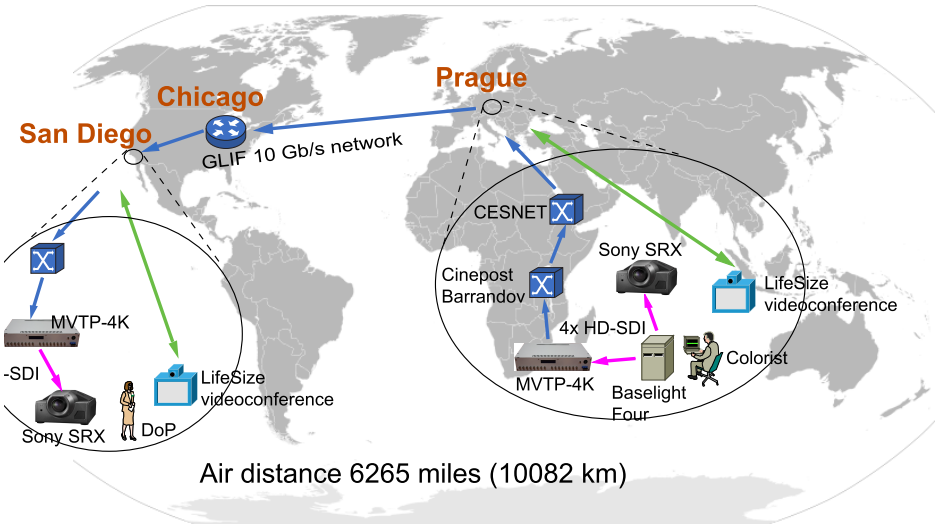

Figure 7: Schematic diagram of the CineGrid 2009 demonstration

The system described in this article differs in that it transports uncompressed 4K video streams over an asynchronous network.

6. Conclusion and future directions

We have demonstrated that it is possible to transfer an uncompressed 4K video content over a long-distance network in real time without observable visual disturbances using FPGA technology for data framing, deframing and rendering speed control.

The technology enables distributed teams to significantly increase their productivity by sharing the uncompressed 4K video content in real time. The application areas include film post-production, scientific visualisation, art performances or high-quality videoconferences. The paradigm (division of participants) can also be used for other parts of the film (post-)production, including editing, special effects or dailies workflow.

In our further work we plan to move the receiver controller to hardware, which should allow to react more quickly to oncoming FIFO overflow or underflow. We will also include a separate controller for each dual-link HD-SDI channel, which will allow transferring multiple lower resolution streams (2K or HD) from different unsynchronized sources, such as individual cameras. Transmision of multiple synchronized 2K or HD streams, such as for 3D ap-

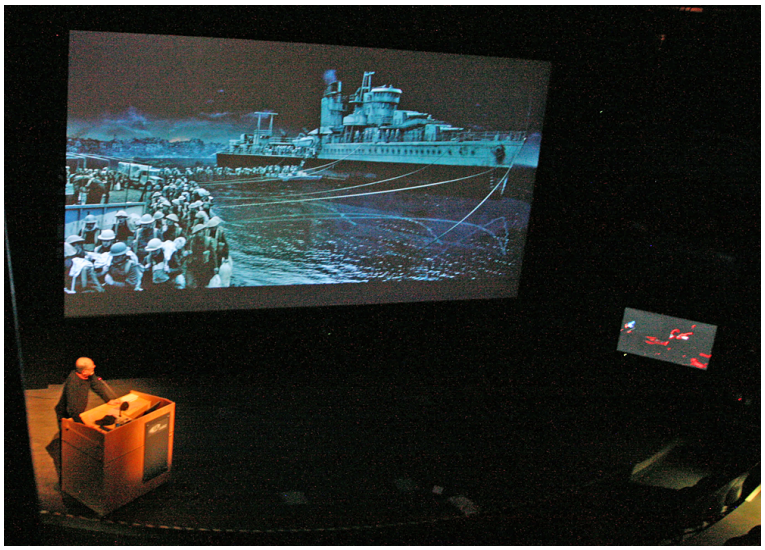

Figure 8: Cinegrid 2009 demonstration appearance

plications is possible now and transmission of 4K 3D streams will be possible with the future firmware version.

The firmware can also be extended by procesing functions, such as encryption or transcoding. An 8K uncompressed version over a 40 Gb/s network is technically possible if there is demand.

References

[1] Charles Poynton, "Chroma subsampling notation", chapter in "Digital Video and HDTV: Algorithms and Interfaces", Morgan Kaufmann, 2002.

[2] "1.5 Gb/s Signal/Data Serial Interface", SMPTE 292-2008.

[3] "Dual Link 1.5 Gb/s Digital Interface for 1920 x 1080 and 2048 x 1080 Picture Formats", SMPTE 372-2009.

[4] "Television - 3 Gb/s Signal/Data Serial Interface", SMPTE 424M-2006.

[5] "Television - 1920 x 1080 Image Sample Structure, Digital Representation and Digital Timing Reference Sequences for Multiple Picture Rates", SMPTE 274M-2008.

[6] "Ultra High Definition Television - Image Parameter Values for Program Production", SMPTE 2036-1-2009.

[7] "Interfaces for the optical transport network (OTN)", ITU-T Recommendation G.709.

[8] "24-Bit Digital Audio Format for SMPTE 292 Bit-Serial Interface", SMPTE 299-2009.

[9] C. Demichelis, P. Chimento, "IP Packet Delay Variation Metric for IP Performance Metrics (IPPM)", RFC 3393, IETF, 2002.

[10] *D-Cinema Distribution Master - Image Pixel Structure Level 3 - Serial Digital Interface Signal Formatting*, SMPTE 428-9-2008.

[11] Bela Liptak. *Instrument Engineers' Handbook, Third Edition: Process Control*, Butterworth-Heinemann, ISBN 0801982421.

[12] MicroBlaze Soft Processor Core, `http://www.xilinx.com/tools/microblaze.htm`.

[13] Net Insight AB, `http://www.netinsight.se`.

[14] NTT Electronics, `http://www.ntt-electronics.com`.

[15] intoPIX, `http://www.intopix.com`.

6.1. Biography

Sven Ubik received his MSc. and Dr. in Computer Science from the Czech Technical University. He is currently with the Research department of CESNET. His research interests include network monitoring, high-definition video, programmable hardware and optical networks.

Jiří Halák and Petr Žejdl received their MSc. at the Czech Technical University and are working towards their PhD in Computer Science. They are also working as researchers in CESNET, particularly in the field of programmable hardware.

Michal Krsek received his BSc. at the University of West Bohemia. He is currently with CESNET and his research interests include high-definition video and its applications.

Felix Nevřela is a Managing Director of the Cinepost corporation.

6.2. Acknowledgements

This work has been supported by the Ministry of Education, Youth and Sports of the Czech Republic under the research intent MSM6383917201 "Optical Network of National Research and Its New Applications".

The tests and Cinegrid demonstration using the Baselight Four and the Sony SRX 4K projector were possible thanks to the generosity of the Cinepost corporation.

5.4 Scalable Embedded Architecture for High-speed Video Transmissions and Processing

This paper was published on The Sixth International Conference on Systems and Networks Communications in 2011 [D.4]. The authors received The Best Paper Award for this paper.

Author list:

- Jiří Halák
- Petr Žejdl
- Sven Ubik

Scalable Embedded Architecture for High-speed Video Transmissions and Processing

Jiří Halák, Sven Ubik, Petr Žejdl
CESNET / CTU Prague
Zikova 4, Prague 6 / Kolejni 550, Prague 6
Czech Republic
email: {halak,ubik,zejdlp}@cesnet.cz

***Abstract*—In this paper, we present a scalable and extendable hardware architecture for processing and transfer of ultra-high-definition video over high-speed 10/40/100 Gbit networks with very low latency. We implemented this architecture in a single FPGA device. Data processing is divided between FPGA resources and an embedded operating system. The FPGA resources can be moved between various processing functions depending on the device mode. The resulting inexpensive and compact device is intended for high quality video transfers and processing with a low latency and to support deployment in education and remote venues.**

Keywords-HD-SDI, video, FPGA, network communication, high-speed

I. Introduction

Video transfers are an expected driver application area of the future Internet. Picture resolution has been increasing over time. Better-than-high-definition-resolution video (such as 4K) is already used in some areas, such as scientific visualization, the film industry or even medical applications.

For the ultimate quality, required for instance in film post-production or live remote surgery transmissions, working with a signal that has not been compressed is preferable. The productivity of a distributed team can be significantly increased when the video signal can be transferred over the network in a real time to enable cooperation that is more effective. Two of the main technical issues are high-data volume and time synchronization when transferring over an asynchronous network such as the current Internet.

Currently available solutions mostly consist of multiple devices (computers, conversion boxes, sync boxes, audio boxes, etc.), which are expensive and harder to setup, increasing the logistics costs. We designed an embedded modular and scalable architecture which fits into a single mid-size FPGA device including all the required functionality and reducing the complexness and costs of this solution. We implemented this architecture and developed a device called MVTP-4K (Modular Video Transfer Platform). We have already used several prototypes in field tests to support applications in film post-production and live medical applications.

This paper is organized as follows: In Section II, we summarize the hardware requirements of our design. In Section III, we present our architecture for video transfers and processing. In Section IV, we present our prototype. In Section V, we summarize our experience with device field tests, In Section VI, we compare our solution with other available devices.

II. Requirements

We have set the following set of requirements for our architecture:

- Video inputs and outputs SDI, HD-SDI or 3G channels
- 10/40/100 Gbit network interface or multiple interfaces
- Very small added latency
- Extendable design for additional processing such as compression or encryption
- Fit into available FPGA devices and fully implementable in one mid-size FPGA device with additional

interfaces

The use of a single- and dual-link HD-SDI channel for the transmission of high definition video streams is now a common industry practice and it is specified in SMPTE 274 [1] and SMPTE 372 [2]. This includes HD (1920x1080) and 2K (2048x1080) formats. The 4K (4096x2160) signals are typically transferred in four quadrants, each in 2K format carried over a separate dual-link HD-SDI channel. 3D transmissions are typically transferred as two independent 2K or 4K channels, some require additional synchronization.

The FPGA circuit was chosen as the processing device due to its versatility that allows us to build a complete embedded solution and to host all required functionality to combine video transmissions with other functions, such as compression, encryption or transcoding.

The architecture must be scalable to allow multiple configurations based on currently available FPGA devices and interfaces, assuming the speed of communication interfaces will increase, and eventually be usable with future 100 Gigabit Ethernet networks and similar high-speed media.

We require an unnoticeable latency added to the network propagation delay for real-time applications. Unnoticeable latency for audio/video applications is below 60 ms for untrained audience and below 30 ms for professional audience.

III. The Architecture

This section describes the proposed architecture.

A. Background

In our previous work we designed and implemented a scalable hardware architecture for network packet processing [3], [4]. This architecture consists of a set of reconfigurable modules for packet processing and a communication interface. The architecture was designed for maximum flexibility and multi-gigabit speeds starting at 10Gb/s. The main processing core was designed to be fully scalable for 10/40/100Gb speeds.

We have designed an interface and developed a prototype for 40Gb/s SONET/SDH networks [5] for basic data processing and testing of 40Gb/s networks and currently we are experimenting with 100Gb Ethernet interface in FPGA devices.

B. Design Overview

Real-time processing of multi-gigabit data rates is difficult on PC-based platforms with standard operating systems not designed for real-time operation. We were looking for a real-time design that is scalable to higher data rates (such as for 8K or UHDTV2 format), higher network speeds (such as 40 and 100 Gb/s) and can be integrated with commonly requested video processing functions, such as encryption, transcoding or compression. Real-time operation means to add a very low latency to a network delay and enable true live experience. This design is fully automated and all embedded in a single FPGA device.

The embedded architecture for real-time video transport and processing is based on our previous work. The core is the scalable architecture for network packet processing [3], [4] designed especially for Ethernet networks. This whole architecture operates at network clock domain of attached network interface and can be used for various modular data packet processing. Since video signal consists of special packets, we can make a simple conversion to transport the video packets to a network clock domain and back. This way we can use a network packet processing architecture for video packet processing.

When we convert data packets from network clock domain to video clock domain certain mechanisms need to be used. Ethernet Network is an asynchronous network, on the other hand, HD-SDI is a synchronous channel thus an advanced techniques for synchronization of data packets crossing from network domain to video domain are required [6].

Address range of all processing modules, routes and hardware configuration registers is mapped to an embedded processor logic bus (PLB) address range using a simple bus bridge of our own design. An embedded processor can be a dedicated Power PC processor or soft-core Microblaze [7] processor. An embedded processor is running a customized Linux distribution and all peripherals are managed by Linux drivers or dedicated software tools. The embedded Linux distribution also provides all means of communication with a device, such as ssh server, web server, display and keyboard controllers and eventually can also handle 10/100/1000 Mbit and multi-gigabit interfaces. The Multi-gigabit Ethernet Net-

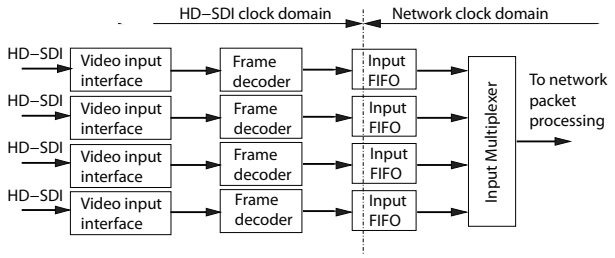

Figure 1. Input video processing and connection to packet processing for 4 channels.

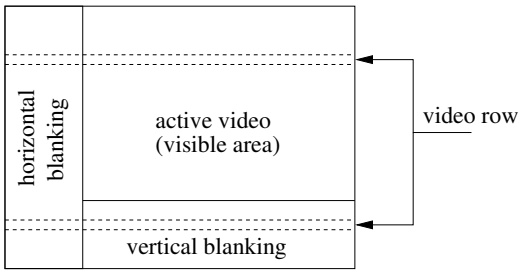

Figure 3. Video frame structure transported through HD-SDI channel

work Interface for an embedded processor is described in a subsection III-F.

C. Video Processing Modules

Video processing modules do a conversion between video and network packets, allowing video data to be processed in the network packet processing core (section III-D). There are two video processing modules, the input and output module, shown in Figures 1 and 2.

The input module consists of the video input interface and the frame decoder. The video input interface implements low-layer communication with the HD-SDI equalizer chips through Rocket IO channels and the frame decoder extracts video packets, converts them to network packets and attaches headers with video format parameters.

The output module consists of the video frame generator and the video output interface. The frame generator receives network packets and generates valid image to the video output interface based on information contained in network packet headers.

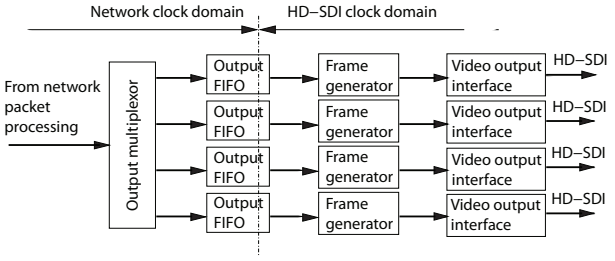

Figure 2. Output video processing and connection to packet processing for 4 channels.

A video packet includes a video pixel row with specified headers and control characters. We use a dual-port memory as a packet FIFO to cross clock domain boundaries. The video processing modules are located in the HD-SDI clock domain and the network packet processing modules are located in the network clock domain. The example configuration in Figures 1 and 2 includes four HD-SDI channels. The channels are independent and can be added freely just with a simple modification of the channel multiplexer. This operation can be completely parameterized.

The HD-SDI interface has a bit rate of 1.485 Gbit/s [8] but not all data needs to be transferred. Video rows include blanking areas (horizontal blanking interval) and a video frame includes blanking video rows (vertical blanking interval). The whole situation is illustrated in Figure 3. Blanking areas can contain some secondary information such as audio, encryption or video format specification, which we can choose to transport or not. When we strip video packets of those blanking intervals, we get a bit rate between 1 Gb/s and 1.3 Gb/s depending on a picture resolution and frame rate. This means that the 10 Gbit Ethernet network can transfer up to eight HD-SDI video channels and with some video formats even including additional data, such as audio or encryption information.

The example bitrates of eight channels of selected video

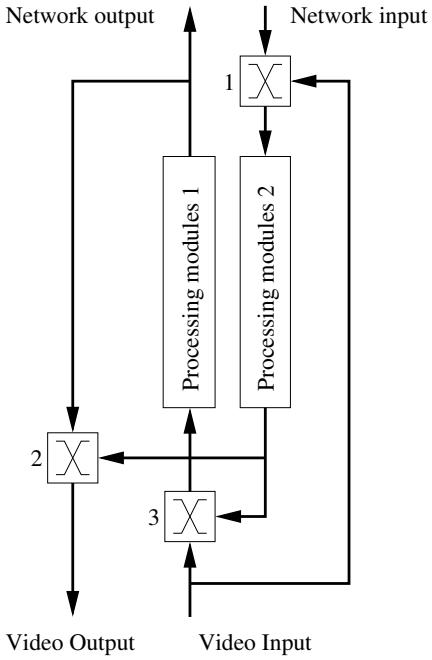

Figure 4. Schematic of interconnections in the processing core.

formats stripped of blanking intervals are summarized in Table I. The 30fps HD formats can be still transferred, but image crop must be applied.

TABLE I
VIDEO FORMATS BITRATES

Format	Bitrate eight channels (Gb/s)	Bitrate one channel (Gb/s)
2K/24	8,7	1,08
1080/24	8,2	1,025
1080/25	8,5	1,06
1080/30	10,4!	1,3
720/50	7,6	0,95
720/60	9,1	1,14

D. Network Packet Processing Core

The main processing core consists of two sets of processing modules. We have extended our original architecture [3], [4] with the video interface and video processing modules described in section III-C. The network packet processing core is divided into two parts. A set of switches can be arranged to allow a packet flow between the network and video domains in several ways. A schematic of this interconnection is shown in Figure 4. The following configurations are possible:

- From network input to network output through switch 1, processing modules 2, switch 3 and processing modules 1. All processing modules are dedicated for network to network packet processing.
- From network to video, full-duplex, one set of processing modules for each direction. From network input through switch 1 and processing modules 2 to video output. From video input through switch 3 and processing modules 1 to network output.
- From video input to video output through switch 3, processing modules 2, switch 3, processing modules 1 and switch 2. All processing modules are dedicated for video packet processing.

Data stream processing modules are inserted directly to the packet stream. Every processing module works as an individual processing unit. The advantage is that modules

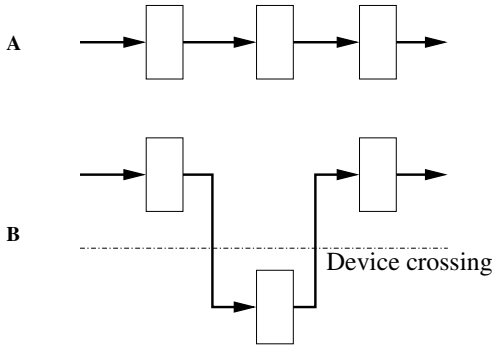

Figure 5. Processing modules

can occupy different FPGA devices. When we need to implement a complex module such as video encoder/decoder, we may find more suitable to use more FPGA devices. For this purpose, the architecture is designed to relocate a packet stream through a high-speed FPGA ports to another device and make a cross-device interconnection of processing modules. Both options are shown in Figure 5. Option A: Modules are connected in a single device. Option B: Module interconnection crosses multiple devices over a high-speed interface.

E. Processing latency

The concept of intra-frame processing of video packets as network packets enables extra low latency of video processing and transmission. This opens a way to truly real-time collaboration support. The processing design itself has a low latency under 1 ms. Video packets are buffered only when synchronizing from the network asynchronous domain to the video synchronous domain. However, low delay variation in the network is required to allow design latency under 1 ms. In lower quality networks the buffering level needs to be obviously increased. The extreme cause is a single frame buffer adding a maximum latency of about 30 ms.

F. Network Interface

High-speed network interface consists of hardware and software parts, which are controlled by an embedded operating system. Incomming packets containing video data are sent to output video processing module, on the other hand network management packets such as ARP or ping are sent to software network driver. Outgoing packets have two different sources, packets containing video data are sent from input video processing module and network management packets are sent from software network driver.

The block diagram of the network interface is shown in Figure 6. Incoming packets are classified in the packet classifier and distributed between video processing modules (VPM) and RX FIFO. Outgoing packets are sent from VPM or from TX FIFO. Because there are two paths producing packets, packet multiplexer is included in the design. It is multiplexing packets in a round-robin fashion. RX and TX FIFO are connected to the CPU through the processor local bus. Therefore, both memories are accessible from software. The packet classifier is also connected to the CPU, but the connection is not shown. The CPU is embedded inside FPGA either.

The packet classifier contains memory for four classification rules. Each rule can be marked as going to the VPM and/or going to RX FIFO. The memory is configured from software. Currently we use three rules. The first rule is marked as going to the VPM and the other two rules are marked as going to RX FIFO. The fourth rule is not used and is reserved for future use.

The rules are as follows:

- Rule for UDP packets containing video data.
- Rule for ARP packets for address resolution.
- Rule for ICMP packets (ping command).

The software part is based on embedded Linux. Network interface is accessible through the Linux TUN/TAP driver [9], which provides packet reception and transmission

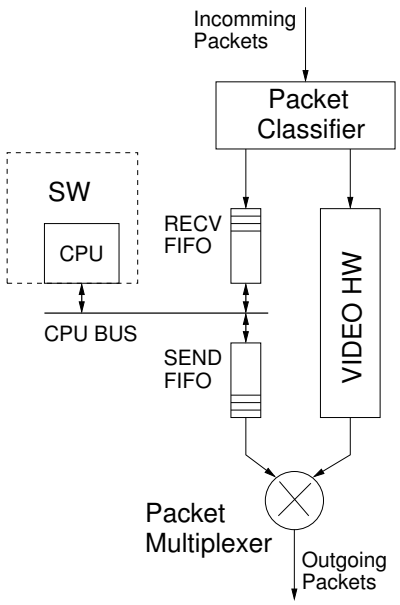

Figure 6. Ethernet Interface Block Diagram

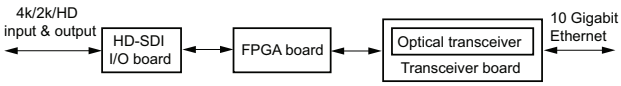

Figure 7. System architecture overview.

for user space programs. The program controlling network hardware is running as a daemon in the user space, and through TUN/TAP driver provides a new network interface. This new interface behaves like an ordinary network interface such as eth. Therefore, all networking services are available through this interface.

IV. MVTP-4K PROTOTYPE

We have designed and build a MVTP-4K (Modular Video Transfer Platform) device which implements proposed architecture and validates it in field tests. The MVTP-4K is a portable device of our own construction for transmission of multiple high-definition video streams including 4k, 2k and HD over a 10 Gigabit Ethernet Network. The device consists of a main FPGA board with 8 HD-SDI video interfaces and one 10Gbit Ethernet interface. Brief structure is shown in Figure 7. The device supports all 4K, 2K and HD resolutions and all corresponding frame rates. 3D transmissions are also supported. Because the data processing is based on data packet processing we can even transport data not fully supported without the need of unpacking them from video signal. This allows us to transport audio data or encryption data embedded in the video stream.

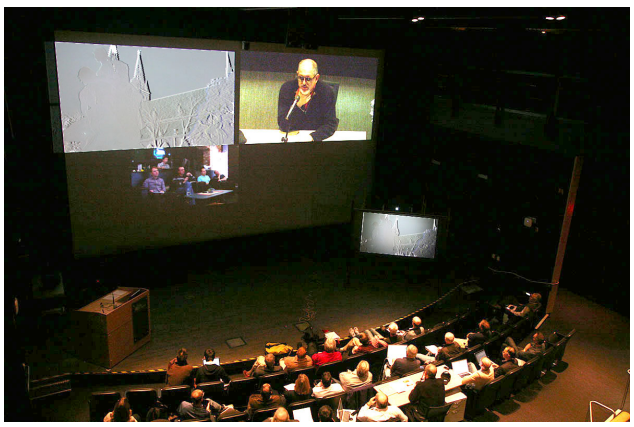

Figure 8. Practical use of the technology at Cinegrid 2010 event

We have chosen a Xilinx Virtex FPGA series because it provides all building blocks and tools required to implement our architecture. The prototype is based on an extended platform for network packet processing called MTPP [3]. Whole architecture fits to a mid-size FPGA device Virtex 5 series XCV5LX110T. We have experimentally confirmed that the device adds a low latency of less than 1 ms.

Mid-size Xilinx FPGAs can be obtained under 3000$ a piece in a small quantities. For advanced hardware functions such as encryption or encoding, a larger FPGA or a second mid-size FPGA is required.

V. PRACTICAL EXPERIENCE

We have demonstrated our system at the Cinegrid 2009 and Cinegrid 2010 workshops. The aim was to demonstrate that such technology can enable real-time remote cooperation of a distributed team and thus increase productivity. In the first event, a stream of uncompressed 4K video was transferred from the Barrandov studios in Prague to the venue in San Diego over a distance of more than 10000 km to perform remote color grading in a real-time. In the second event, a stream of 3D 2K video was transferred from the UPP Company in Prague to the venue in San Diego to perform remote real-time postproduction processing of 3D images. The 3D grading performed at the venue with the signal transferred by our device is illustrated in Fig.8.

In order to evaluate the system suitability for e-Health applications, we transferred several surgical operations from the daVinci Surgical System [10] which produces HD stereoscopic signal in 1080i format. The picture quality was subjectively approved by invited medical experts as suitable for highly illustrative student training or presentations of surgical procedures on symposia.

VI. RELATED WORK

There are several commercial products, which allow transport of SDI, HD-SDI or 3G channels over network.

Net Insight's [11] Nimbra 600 series switch can transport 8x HD-SDI or 3G SDI channels over an SONET/SDH network. There are several commercially available solutions for transport of compressed 4K video over the Internet, for example NTT Electronics [12] ES8000/DS8000 4K MPEG-2 encoder/decoder complemented with NA5000 IP interface unit and intoPIX's [13] system of PRISTINE PCI-E FPGA boards and JPEG 2000 IP cores.

UltraGrid from Laboratory of Advanced Networking Technologies is a software for real-time transmissions of high-defintion video [14]. This solution is a fully software based and requires dedicated PC with specialized hardware.

The architecture and design described in this article differs in that it is a hardware solution fully scalable to higher speeds. The number of video and network interfaces is parameterized and can be easily extended. The FPGA enabled parallelism allows our architecture to process several video channels at once and to transfer every video format contained in SDI, HD-SDI or 3G interface. The architecture is designed to be embedded to a single FPGA device but some larger processing modules can be relocated to other FPGA devices. Our design has a very small added latency around 1 ms that enables a true real-time distributed team cooperation.

VII. Conclusion

We have extended a scalable architecture for network packet processing [3], [4] by video interfaces options. The resulting architecture is designed to process or transport video data over an asynchronous network with very low added latency. The design enables true real-time distributed team cooperation. The real-time team cooperation was demonstrated in several applications in the cinema industry and e-Learning in medicine. The architecture also fulfills the hardware requirements that we set and we successfully implemented this architecture in a single FPGA device and presented its capabilities.

Acknowledgment

This work has been supported by the Ministry of Education, Youth and Sports of the Czech Republic under the research intent MSM6383917201 Optical Network of National Research and Its New Applications.

References

[1] "1920 x 1080 Image Sample Structure, Digital Representation and Digital Timing Reference Sequences for Multiple Picture Rates." Society of Motion Picture and Television Engineers., 2005.

[2] "Dual Link 1.5 Gb/s Digital Interface for 1920 x 1080 and 2048 x 1080 Picture Formats." Society of Motion Picture and Television Engineers., 2009.

[3] J. Halak and S. Ubik, "MTPP - Modular Traffic Processing Platform," in *12th IEEE Symposium on Design and Diagnostics of Electronic Systems, DDECS 2009*, Liberec, Czech Republic, April 2009, pp. 170–173.

[4] J. Halak, "Multigigabit network traffic processing," in *Proc. The International Conference on Field Programmable Logic and Applications, FPL 2009*. Prague, Czech Republic: IEEE Computer Society, August 2010, pp. 521–524.

[5] J. Halak, S. Ubik, and P. Zejdl, "Data stream processing for 40 Gb/s networks," in *Proc. Fifth International Conference on Digital Telecommunications, ICDT 2010*. Athens/Glyfada, Greece: IEEE Computer Society, June 2010, pp. 149–152.

[6] ——, "Receiver synchronization in video streaming with short latency over asynchronous networks." Vienna, Austria: IEEE Computer Society, April 2010, pp. 403–405.

[7] MicroBlaze Soft Processor Core. (Last accessed: July, 2011). [Online]. Available: http://www.xilinx.com/tools/microblaze.htm

[8] "1.5 Gb/s Signal/Data Serial Interface." Society of Motion Picture and Television Engineers., 2008.

[9] Universal TUN/TAP device driver, Linux Kernel Documentation. (Last accessed: July, 2011). [Online]. Available: http://kernel.org/doc/Documentation/networking/tuntap.txt

[10] The da Vinci Surgical System, Intuitive Surgical. (Last accessed: July, 2011). [Online]. Available: http://www.intuitivesurgical.com/products/faq/index.aspx

[11] Net Insight AB. (Last accessed: July, 2011). [Online]. Available: http://www.netinsight.se

[12] NTT Electronics. (Last accessed: July, 2011). [Online]. Available: http://www.ntt-electronics.com

[13] intoPIX. (Last accessed: July, 2011). [Online]. Available: http://www.intopix.com

[14] P. Holub, L. Matyska, M. Liska, L. Hejtmanek, J. Denemark, T. Rebok, A. Hutanu, R. Paruchuri, J. Radil, and E. Hladk, "High-definition multimedia for multiparty low-latency interactive communication," *Future Generation Computer Systems*, vol. "22", no. "8", pp. "856 – 861", October "2006". [Online]. Available: "http://www.sciencedirect.com/science/article/pii/S0167739X06000380"

Bibliography

[1] 1.5 gb/s signal/data serial interface. *Society of Motion Picture & Television Engineers*. SMPTE 292-2008.

[2] Ancillary data packet and space formatting. *Society of Motion Picture & Television Engineers*. Multiple standards SMPTE 291M-2006.

[3] Cesnet, liberouter. `http://www.liberouter.org`.

[4] Cesnet, liberouter, netcope project. `http://www.liberouter.org/netcope/index.php`.

[5] Cesnet, performance monitoring and optimization. `http://www.ces.net/project/qosip`.

[6] D-cinema distribution master. *Society of Motion Picture & Television Engineers*. Multiple standards SMPTE 428-x-20xx.

[7] Dual link 1.5 gb/s digital interface for 1920 x 1080 and 2048 x 1080 picture formats. *Society of Motion Picture & Television Engineers*. SMPTE 372-2009.

[8] Dvi 1.0 specification. *The Digital Display Working Group*. `http://www.ddwg.org/`.

[9] Endace company. `http://www.endace.com`.

[10] Exfo. `http://www.exfo.com/`.

[11] High definition multimedia interface. `http://www.hdmi.org`.

[12] intopix. `http://www.intopix.com`.

[13] Ixia. `http://www.ixiacom.com/products/index.php`.

[14] Modular traffic processing platform. `http://www.ces.net/project/qosip/`.

[15] Modular video transfer platform. `http://www.mvtp4k.com/`.

[16] Mtpp programmable platform under routine operation in the cesnet2 network. *Press Release*. `http://www.ces.net/doc/press/2009/pr090226.html`.

[17] Net insight ab. `http://www.netinsight.se`.

[18] Ntt electronics. `http://www.ntt-electronics.com`.

[19] Sdtv digital signal/data serial digital interface. *Society of Motion Picture & Television Engineers*. SMPTE 259M-2008.

[20] Television - 1920 x 1080 image sample structure, digital representation and digital timing reference sequences for multiple picture rates. *Society of Motion Picture & Television Engineers*. SMPTE 274M-2008.

[21] Television - 3 gb/s signal/data serial interface. *Society of Motion Picture & Television Engineers*. SMPTE 424M-2006.

[22] Ultra high definition television - image parameter values for program production. *Society of Motion Picture & Television Engineers*. SMPTE 2036-1-2009.

[23] International standard iso/iec 7498-1. *ISO - International Organization for Standardization*, 1996. `http://standards.iso.org/ittf/PubliclyAvailableStandards/s020269_ISO_IEC_7498-1_19`

[24] Ieee 802.3: Ethernet. *IEEE Standards Association*, 2000. `http://standards.ieee.org/about/get/802/802.3.html`.

[25] Xilinx inc., two flows for partial reconfiguration: Module based or difference based. 2004. Technical report, XAPP290.

[26] Xilinx inc., early access partial reconfiguration user guide, ug208. 2005. `http://www.xilinx.com/support/prealounge/protected/index.htm`.

[27] Xilinx partial reconfiguration user guide. 2011. User Guide, UG702.

[28] Muhammad Bilal Anwer, Murtaza Motiwala, Mukarram bin Tariq, and Nick Feamster. SwitchBlade: a platform for rapid deployment of network protocols on programmable hardware. *Proceedings of the ACM SIGCOMM 2010 conference*, pages 183–194, 2010.

[29] Denis Bechet, Philippe de Groote, Christian Retore, and Projet Calligramme. A Complete Axiomatisation for the Inclusion of Series-Parallel Partial Orders. *Proceedings of 8th International Conference, RTA-97 Sitges*, pages 230–240, 1997.

[30] F. Braun, J. W. Lockwood, and M. Waldvogel. Layered protocol wrappers for internet packet processing in reconfigurable hardware. *IEEE Micro, Volume 22*, pages 66–74, februar 2002.

[31] A.N. Choudhary, B. Narahari, D.M. Nicol, and R. Simha. Optimal processor assignment for a class of pipelined computations. *Parallel and Distributed Systems*, pages 439–445, 1994.

[32] M. Dyer, C. Plessl, and M. Platzner. Partially reconfigurable cores for xilinx virtex. *2002 Proceedings of Field Programmable Logic and Application (FPL) Conference*, 2002.

[33] Petr Holub, Ludek Matyska, Milos Liska, Lukas Hejtmanek, Jiri Denemark, Tomas Rebok, Andrei Hutanu, Ravi Paruchuri, Jan Radil, and Eva Hladk. High-definition multimedia for multiparty low-latency interactive communication. *Future Generation Computer Systems*, pages 856–861, October 2006.

[34] Edson L. Horta, John W. Lockwood, David E. Taylor, and David Parlour. Dynamic hardware plugins in an fpga with partial runtime reconfiguration. *Design Automation Conference (DAC)*, june 2002.

[35] L. House, J. Hill, and C Maxfield. The design warrior's guide to fpgas. 2004.

[36] J. W. Lockwood. An open platform for development of network processing modules in reprogrammable hardware. *IEC Design-Con01, (Santa Clara, CA)*, pages WB–19, januar 2001.

[37] John W. Lockwood, Nick McKeown, Greg Watson, Glen Gibb, Paul Hartke, Jad Naous, Ramanan Raghuraman, and Jianying Luo. NetFPGA–An Open Platform for Gigabit-Rate Network Switching and Routing. *MSE '07 Proceedings of the 2007 IEEE International Conference on Microelectronic Systems Education*, pages 160–161, 2007.

[38] P. Lysaght, B. Blodget, J. Mason, J. Young, and B. Bridgford. Invited paper: Enhanced architectures, design methodologies and cad tools for dynamic reconfiguration of xilinx fpgas. *Field Programmable Logic and Application (FPL) Conference*, 2006.

[39] Sascha Muhlbach and Andreas Koch. A Scalable Multi-FPGA Platform for Complex Networking Applications. *IEEE 19th Annual International Symposium on Field-Programmable Custom Computing Machines*, pages 81–84, 2011.

[40] J. Resano, D. Verkest, D. Mozos, S. Vernalde, and F. Catthoor. Application of task concurrency management on dynamically reconfigurable hardware platforms. *IEEE Symposium on Field-Programmable Custom Computing Machines*, pages 278–279, 2003.

[41] David V. Schuehler and John W. Lockwood. A modular system for fpga-based tcp flow processing in high-speed networks. pages 301–310, 2004.

[42] Daisuke Shirai, Masahiko Kitamura, Tatsuya Fujii, Atsushi Takahara, Kunitake Kaneko, and Naohisa Ohta. Multi-point 4K/2K layered video streaming for remote collaboration. *Future Generation Computer Systems - Volume 27 Issue 7, July, 2011*, pages 986–990, 2011.

[43] Daisuke Shirai, Masahiko Kitamura, Tatsuya Fujii, Atsushi Takahara, Kunitake Kaneko, Naohisa Ohta, and 18 others. International real-time streaming of 4K digital cinema. *Future Generation Computer Systems - Volume 22 Issue 8, October 2006*, pages 929–939, 2006.

[44] M. Kosek T. Martinek. Netcope: Platform for rapid development of network applications. *11th IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems*, 2008.

List of papers included in the Thesis

[D.1] J. Halak and S. Ubik. *MTPP - Modular Traffic Processing Platform.* 12th IEEE Symposium on Design and Diagnostics of Electronic Systems, pp. 170-173, Liberec, Czech Republic, 2009

[D.2] J. Halak *Multigigabit network traffic processing.* Proc. The International Conference on Field Programmable Logic and Applications, IEEE Computer Society, pp. 521-524, Prague, 2010, ISBN:978-1-4244-3892-1

The paper has been cited in:

- C. E. Neely, G. J. Brebner, W. Shang, *Flexible and Modular Support for Timing Functions in High Performance Networking Acceleration*, FPL, 2010

[D.3] J. Halak, M. Krsek, S. Ubik, P. Zejdl and F. Nevrela. *Real-time long-distance transfer of uncompressed 4K video for remote collaboration.* Future Generation Computer Systems, ISSN:0167-739X, 2010.

[D.4] J. Halak, S. Ubik and P. Zejdl. *Scalable Embedded Architecture for High-speed Video Transmissions and Processing.* The Sixth International Conference on Systems and Networks Communications, pp. 161-166, Barcelona, 2011, ISBN: 978-1-61208-166-3

Patents held by the Author

[P.1] J. Halak, S. Ubik and P. Zejdl. *A device for receiving of high-definition video signal with low-latency transmission over an asynchronous packet network*. World Intellectual Property Office (WIPO) PCT Patent 2011/116735.

[P.2] J. Halak, S. Ubik and P. Zejdl. *Modular Programmable Platform for High-Speed Hardware Packet Processing*. Patent no. 2007-850.(Patent paper in Czech language).

[P.3] J. Halak, S. Ubik and P. Zejdl. *Apparatus for receiving video signal of high resolution transmitted with a small delay through asynchronous packet computer network*. Patent no. 2010-226.(Patent paper in Czech language).

[P.4] J. Halak, S. Ubik and P. Zejdl. *A device for reception of high-resolution video signals transferred with small delay over an asynchronous computer network*. Utility patent no. 2010-22484.(Utility patent paper in Czech language).

[P.5] J. Halak, S. Ubik and P. Zejdl. *A device for processing and transmission high-resolution video signals over an computer network*. Utility patent no. 2009-22001.(Utility patent paper in Czech language).

Other Related Publications by the Author

[A.1] S. Ubik, J. Halak, P. Zejdl and M. Krsek. *Low-latency transmissions for remote collaboration in post-production.* SMPTE Annual Technical Conference & Exhibition 2012 Hollywood, California, 2012

[A.2] S. Ubik, Z. Travnicek, P. Zejdl and J. Halak. *Remote Access to 3D Models for Research, Engineering, and Art.* IEEE MultiMedia 2012 pp. 12-19, 2012, ISSN: 1070-986X

[A.3] J. Navratil, S. Ubik, P. Peciva, J. Halak, P. Zejdl and J. Schraml *Live video transmission as tool in teaching students and exchanging the experiences in medical fields.* Proceedings of The 1st Internet & Business Conference 2012. Rovinj, Croatia, 2012, ISSN:1848-5278

[A.4] J. Navratil, M. Sarek, S. Ubik, J. Halak, P. Zejdl, P. Peciva and J. Schraml *Real-Time Stereoscopic Streaming of Robotic Surgeries.* 13th IEEE International Conference on e-Health Networking, Application and Services, pp. 40-45, Columbia, USA, 2011, ISBN: 978-1-61284-695-8

[A.5] J. Halak, S. Ubik and P. Zejdl. *Receiver synchronization in video streaming with short latency over asynchronous networks.* 13th IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems, pp. 403-405, Vienna, 2010, ISBN:978-1-4244-6612-2

[A.6] J. Halak, J. Navratil, P. Peciva, M. Sarek, S. Ubik and P. Zejdl. *Live long-distance stereoscopic transmissions of surgical procedures.* 8th Int. Conference on Emerging eLearning Technologies and Applications, The High Tatras, Slovakia, 2010, ISBN:978-80-8086-166-7

[A.7] J. Halak, S. Ubik and P. Zejdl. *Data Stream Processing for 40 Gb/s Networks.* Proc. Fifth International Conference on Digital Telecommunications, IEEE Computer Society, pp. 149-152, Athens/Glyfada, 2010

Other Publications by the Author

[A.8] S. Ubik, P. Zejdl and J. Halak *Real-time anonymization in passive network monitoring.* Proc. Third International Conference on Networking and Services, IEEE Computer Society, 2007, ISBN:0-7695-2858-9

The paper has been cited in:

- A. Blake, R. Nelson, *Scalable Architecture for Prefix Preserving Anonymization of IP Addresses*, Proceedings of the 8th international workshop on Embedded Computer Systems: Architectures, Modeling, and Simulation, July 21-24, 2008, Samos, Greece
- A. Jain, K. Prasad, G. Kutty, M. John and P. Balan, *Analyzing and Recommending DPI Countermeasure Tools for Protecting User Privacy*, Capstone Papers, 2009.

[A.9] P. Zejdl and J. Halak *FPGA-based acceleration of Packet Header Anonymization.* POSTER 2007, Dept. of CS&E, Czech Technical University, Prague, 2007.

[A.10] S. Ubik, P. Žejdl and J. Halák. *FPGA-based packet header anonymization.* Technical Report, Praha: CESNET, 2006. 16/2006. 17 pages. ISBN 978-80-239-9285-4 `http://www.cesnet.cz/doc/techzpravy/2006/anon/`

[A.11] J. Halák. *Using the Dynamic Reconfiguration of FPGA circuits in the Fault Tolerant Systems.* Počítačové architektúry a diagnostika. Bratislava: Ústav informatiky SAV, 2006, pp. 83-88. ISBN 80-969202-2-7