CESNET Technical Report 2/2014

HANIC 100G: Hardware accelerator for 100 Gbps network traffic monitoring

Viktor Puš, Lukáš Kekely, Martin Špinler, Václav Hummel, Jan Palička

Received 3. 10. 2014

Abstract

This technical report describes design, implementation and evaluation of hardware accelerator for the monitoring of 100 Gbps networks. We call the system HANIC - Hardware Accelerated Network Interface Card. HANIC uses custom FPGA-based card, Linux OS device drivers, libraries and tools to create the system for high-speed packet capture and processing. We provide design and implementation details together with the achieved results to show that the system is capable of full 100 Gbps traffic processing.

Keywords: 100GE, Ethernet, FPGA, monitoring, security, HANIC

1 Introduction

The task of network traffic monitoring is one of the key concepts in modern network engineering and security. Among the desired properties of many monitoring applications is the ability to process unsampled network traffic in order to obtain the maximal possible amount of usable information and to detect even single-packet attacks.

Current commodity hardware for high-speed network traffic flow monitoring usually consists of a PC with one or two network interface cards (NICs) with a 10 G Ethernet interface. The role of the hardware is merely to capture the data from the network. The hardware setup is supplemented by open- or closed-source software performing the subsequent steps of the flow monitoring process, i.e. packet header parsing, packet aggregation to flow records, and flow record export.

This model is, however, no longer feasible in the light of recent advances in network speeds, most notably with the rising adoption of 100 Gbps Ethernet standard IEEE 802.3ba. The main reason is the unavailability of commodity 100GE NICs. Intel has recently announced the XL710 network adapters [1] with the support for 40 GE interfaces, so that we can expect 100 GE coming soon.

But there is yet another issue which has to be dealt with: insufficient single-threaded performance of current CPUs. 100 Gbps Ethernet transmits up to 150 million packets per second in each direction, which is only 18 cycles of a 3 GHz CPU. Given the fact that each new network packet is an inevitable CPU cache miss, it is clear that a single thread is unable to do any meaningful task over 100 Gbps traffic.

Our work addresses these issues by designing the traffic processing accelerator using the custom FPGA-based card COMBO-100G [2] developed by CESNET and INVEA-TECH [3] and shown in Figure 1. The card is equipped with the CFP2

© CESNET, 2015

cage which accepts variety of optical modules as described in [2]. The most notable 100GE optical PMDs (also supported by our firmware) are 100GBASE-LR4 and 100GBASE-SR10. The main component of the card is the powerful Xilinx Virtex-7 H580T FPGA, which connects to the CFP2 module as well as to the card's PCI-Express 16-lane gen3 interface.

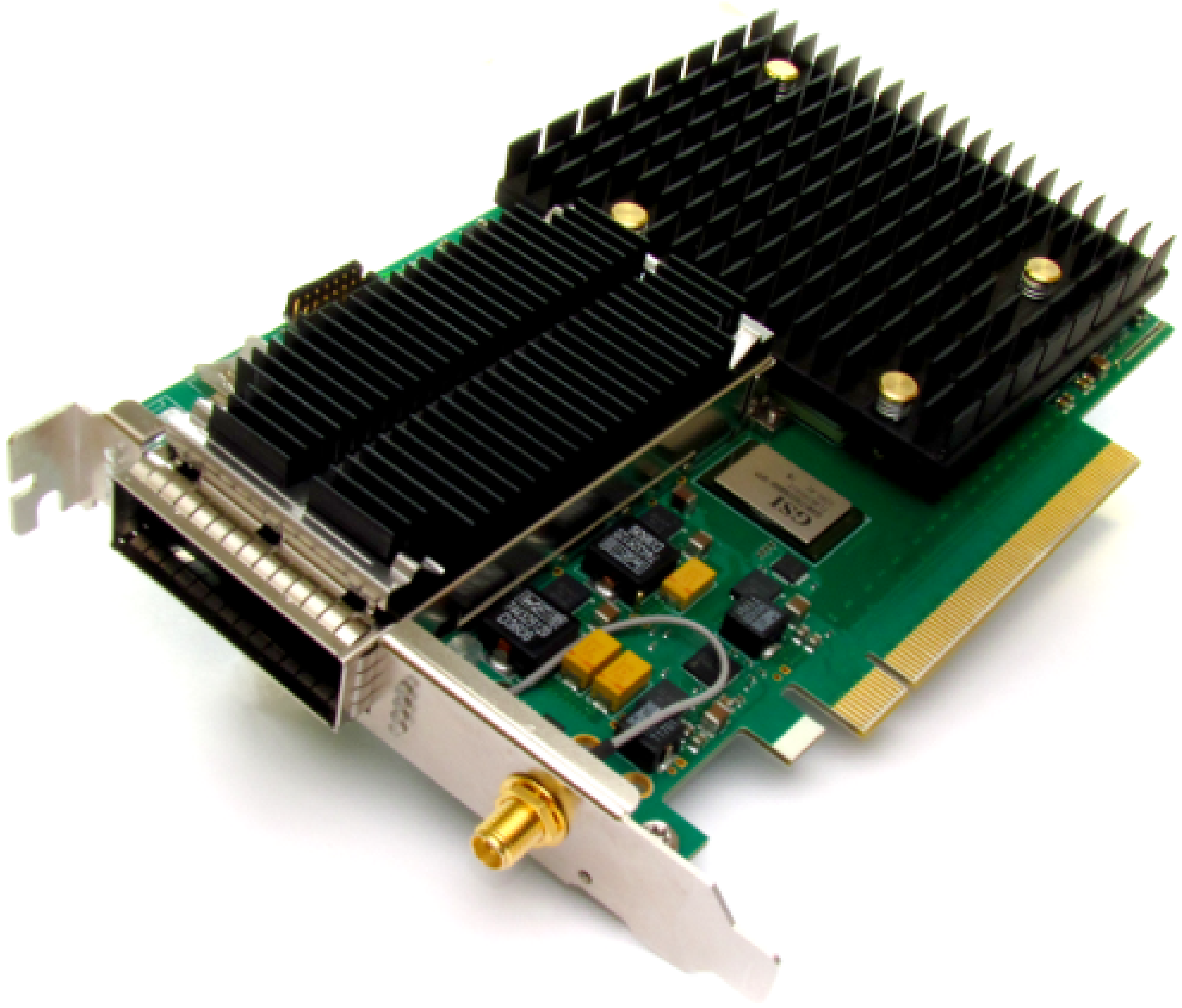

Figure 1. COMBO-100G card

In the remaining parts of the report, we describe and evaluate the FPGA firmware designed to achieve the maximal possible overall system throughput when receiving and processing network packets.

2 System architecture

2.1 NetCOPE Development Platform

The basic firmware functionality such as Ethernet PMA, PCS and MAC sublayers and PCI-Express communication are part of the NetCOPE development platform described in [4]. Here we present only the basic concepts of NetCOPE and continue with the description of the HANIC firmware application.

The greater part of Physical Medium Attachment (PMA) Ethernet sublayer is implemented by four of the the FPGA's embedded GTZ 25 GHz transceivers, which provide the basic functionality of data serialization, deserialization and clock data recovery. There is the option to switch these transceivers to 10 GHz speed and use additional six GTH transceivers to implement SR10 version of 100 G Ethernet standard. Other Ethernet functionality is implemented directly in the FPGA logic. That includes data scrambling and descrambling, Frame Check Sum (CRC) checking at RX and generating at TX and other functions. The MAC layer supports jumbo frames up to 16 352 Bytes long.

NetCOPE is also responsible for the PCI-Express transfers. With the use of GTH transceivers and two FPGA's embedded x8 PCIe blocks, NetCOPE implements Physical, Data Link and Transactions layers of the PCI-Express protocol. The implementation of 100 Gbps transfers over the PCI-Express bus in our system is rather novel and non-traditional. Current FPGAs support PCIe hard macros up to gen3 x8. Given the data rate of 8 Gbps per PCIe gen3 lane, the theoretical throughput of a single PCIe interface supported by any current FPGA is 64 Gbps - not enough for 100 Gbps applications. Real throughput is even lower due to the PCIe protocol overhead. The question here is: Can we get to 100 Gbps with current FPGAs? There is an option to include a PCIe switch chip in the x8+x8=x16 configuration on the card. The switch chip will, however, have a power consumption somewhere around 6 Watts, not to speak about the complicated PCB and increased cost. There is a much more elegant way to 100 Gbps: PCIe bifurcation.

This concept was introduced by Intel's Core CPUs a couple of years ago, but the necessary support from the motherboard vendors was often neglected. With this situation slowly changing, it is now possible to make use of bifurcation to build a 100 Gbps system with a single FPGA and without the need for PCIe switch chip on-board. With bifurcation, single physical PCIe x16 slot can be configured at boot time to function as two electrical and logical PCIe x8 interfaces. The operating system running on the CPU sees two logical PCIe devices, but this can be easily disguised by the device driver so that the user applications see a single (yet very fast) application layer interface. Figure 2 shows the conceptual scheme of bifurcation, while Figure 3 illustrates the measured throughput of our solution. The throughput was measured by using internal synthetic traffic generator, which was able to generate even more than 100~Gbps of traffic.

2.2 Hardware Accelerated Network Interface Card

The HANIC firmware design stems from a conventional NIC, but has several important features that set it apart from a typical commodity devices and help to improve general throughput when receiving packets. The card automatically assigns 64-bit timestamp to each packet at the moment of reception. The timestamp has resolution of one nanosecond, which is better than what can be achieved by assigning it later by the software application. Also the processor load associated with the timestamp generation is removed.

The HANIC firmware has also the option for traffic filtering. The filter supports filtering by the following fields:

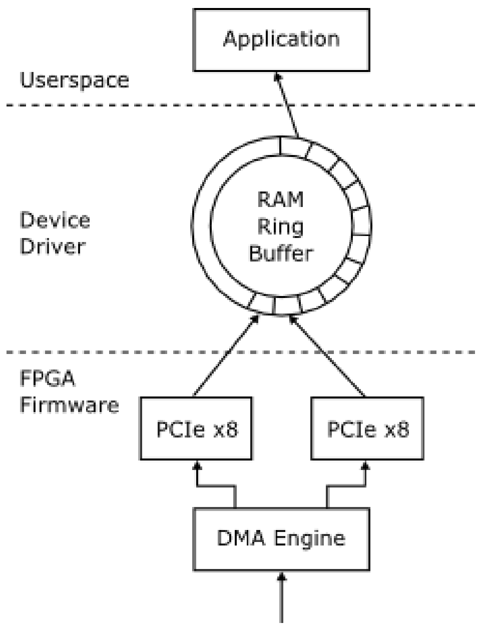

Figure 2. Conceptual scheme of 100 Gbps PCI-Express transfers directly from FPGA

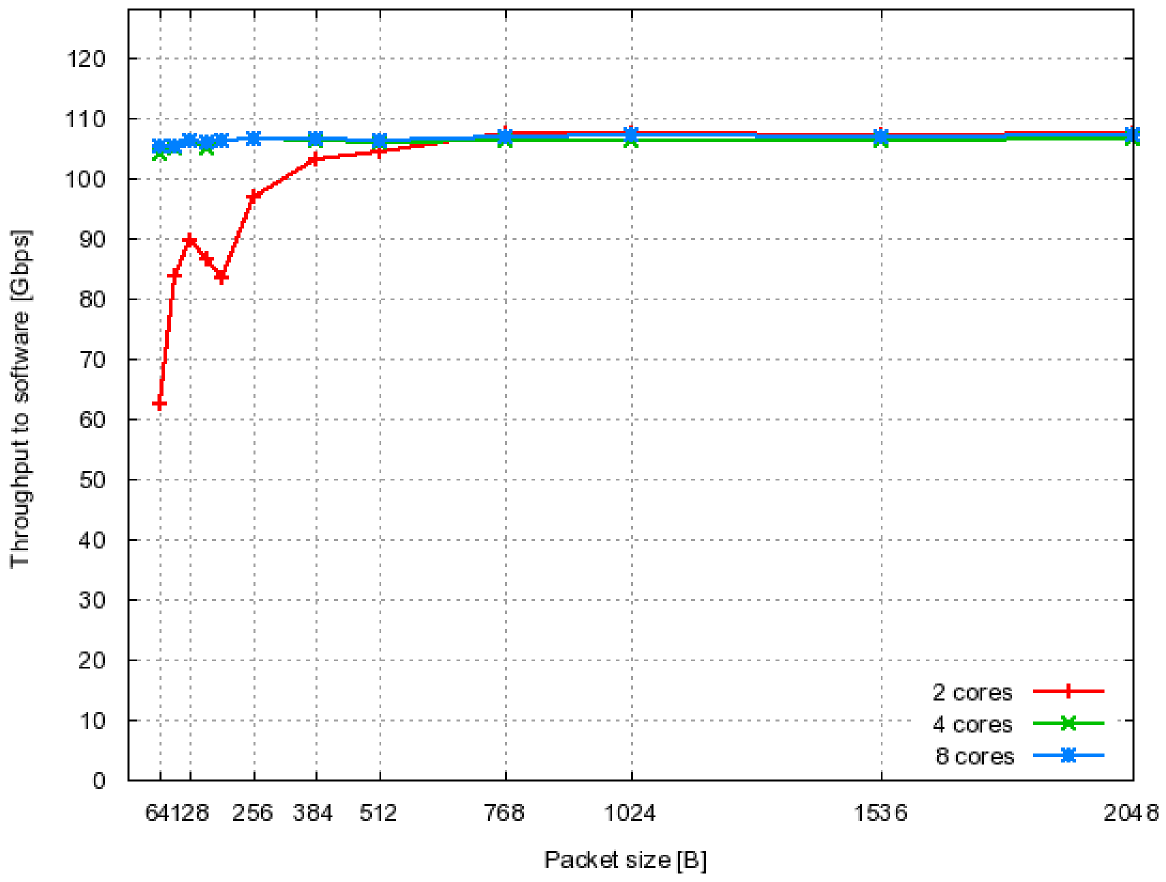

Figure 3. Measured throughput of bifurcated PCI-Express interface

— 32 source IP addresses (IPv4 or IPv6, including prefixes)
— 32 destination IP addresses (IPv4 or IPv6, including prefixes)

— Unlimited number of L4 protocols
— Unlimited number of source and destination TCP/UDP port numbers

The firmware can be set either to receive only the packets that match the filter and drop the rest, or vice versa: to receive everything except for the matched packets. The filtering rules may also define the target CPU core for the matching packet (see later).

The firmware is capable of packet sampling with the ratios 1:1, 1:2, 1:4, 1:8, 1:16, which can be enabled when the software processing cannot keep up with the incoming data rate.

Another feature is configurable packet trimming. The card can be set to trim the packets to a predefined length, thus saving the PCI-Express and memory subsystem bandwidth, most notably for long packets. This feature is clearly intended for the purpose of flow monitoring, since the relevant information (packet header) is typically at the beginning of each packet.

Further extension of this feature leads to packet parsing directly by the card and transferring only the parsed packet header fields to the host RAM. We use high-speed modular packet header parser described in [4]. In addition to the bandwidth saving, the processor load is also reduced, because it no longer needs to perform the packet parsing operation. The parsed fields are sent in the so-called Unified Header (UH) format which has fixed structure and thus removes the need for complicated parsing conditions in the corresponding processor code.

Our packet parser is able to deal with the following protocol stack:

— Ethernet
— Up to four MPLS headers
— Up to four IEEE 802.1Q tags (VLANs)
— IPv4 or IPv6 (with up to two extension headers)
— TCP, UDP or ICMP

To better utilize current multicore CPUs, the firmware features the option to distribute the packets into eight independent DMA channels. Target channel number of each packet is a hash value computed from several fields of the packet header, such as IP addresses, protocol number, port numbers. This ensures consistent mapping of network flows to DMA channels, so that the software threads always see complete flows. In the common traffic with large number of flows, the traffic is evenly distributed among the DMA channels.

The DMA transfers themselves are simplified and optimized to bypass the OS kernel network stack. Large ring buffers are allocated in RAM during the OS driver initialization, and the packets are then uploaded to these buffers by the card almost autonomously. The only communication between the driver and the card is the exchange of buffer start and end pointers (which mark empty and free space in the buffer), and configurable interrupts generated by the card. The OS driver never touches or even copies the packets, it only maps portions of the buffers to userspace applications. This way the overhead of software processing is minimized and maximum CPU time is left for the application itself. Directly mapping the ring buffers memory to userspace also ensures that the data are copied only once, from the NIC to the RAM. This decreases the load of the memory subsystem, which helps to increase the overall performance.

Figure 4 shows the conceptual scheme of the HANIC firmware, which roughly corresponds to the actual pipelined implementation for the FPGA. Note that the gray blocks represent the NetCOPE platform functionality, while the white block are the HANIC-specific functions.

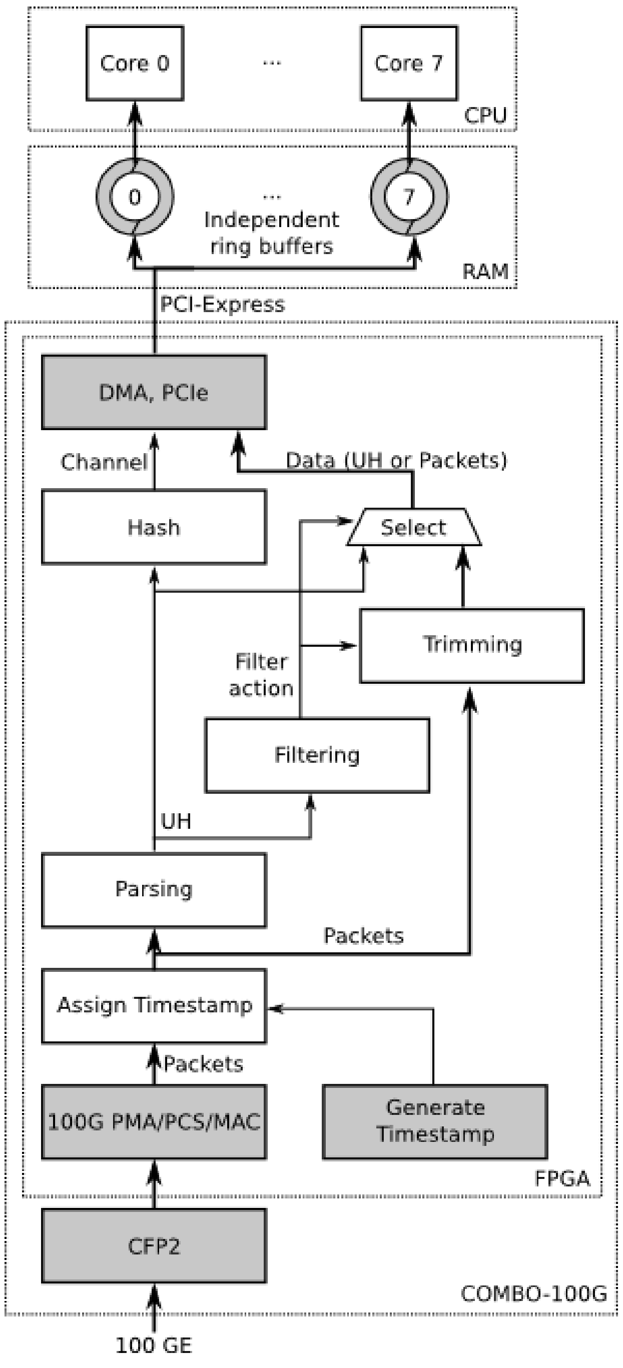

Figure 4. HANIC functional scheme

3 Results

3.1 FPGA Implementation

The firmware for the FPGA is implemented in the VHDL language and compiled with the Xilinx Vivado 2014.2 tool. To achieve the desired throughput, the core of the firmware runs at 204 MHz using a 512 bits wide data pipeline. The PCI-Express subsystem, however, uses dual 256 bits wide paths at 250 MHz. Table 1 summarizes the FPGA resources used by the HANIC firmware. It can be seen that the FPGA chip area is not fully utilized, so that the firmware can be further extended by additional features.

Table 1. FPGA resources of the 100G HANIC firmware

Resource	**LUT**	**FF**	**BRAM**	**GTZ**	**GTH**
Utilization	129643	98623	207	4	16
Available	362800	725600	940	8	48
% Utilized	35.73	13.59	22.02	50	33.33

3.2 System Throughput

We measure the throughput using only medium performance CPU in order to show that the distribution to CPU cores is effective. The CPU used is Intel Xeon E5-2670 (2.6 GHz, turbo 3.3 GHz, 20 MB cache), together with eight modules of DDR3 memories (8 GB, 1600 MHz each).

We use Spirent TestCenter hardware generator to generate input traffic to HANIC. The generator is set to generate traffic at the maximum rate allowed by the 100G Ethernet standard (which is slightly below 100 Gbps when measured at L2 due to protocol overhead). The generator is set to generate standard IPv4 packets with random IP addresses to achieve balanced traffic distribution among CPU cores. The software used to measure the throughput is a simple multithreaded packet counter in order to demonstrate primarily the firmware speed and features.

Figure 5 shows the measured throughput.

It can be seen that the throughput is slightly below 100 % in some cases (please note that the vertical axis has scale from 99.5 % to 100 %). That comes as a surprise, given the fact that the measured throughput of the sole PCI-Express interface is around 107 Gbps regardless of the packet length (see Figure 3).

The cause of the slight performance drop is rather complex. We have observed extended time periods (up to 8 microsecond) of PCI-Express bus inactivity, which was forced by the receiving side of communication (the CPU-integrated PCI-Express endpoint). This leads to buffering of packets in the firmware, and in rare cases even to packet drops due to buffer overflow. This is more evident towards longer packets, because then the Ethernet protocol overhead is lower, and therefore more data is effectively received. It is worth noting that packet drops only occur when receiving packets from fully loaded 100 Gbps Ethernet line - a situation that is extremely rare in practical deployments. We are, however, currently searching for the cause of these states of inactivity and how to avoid them.

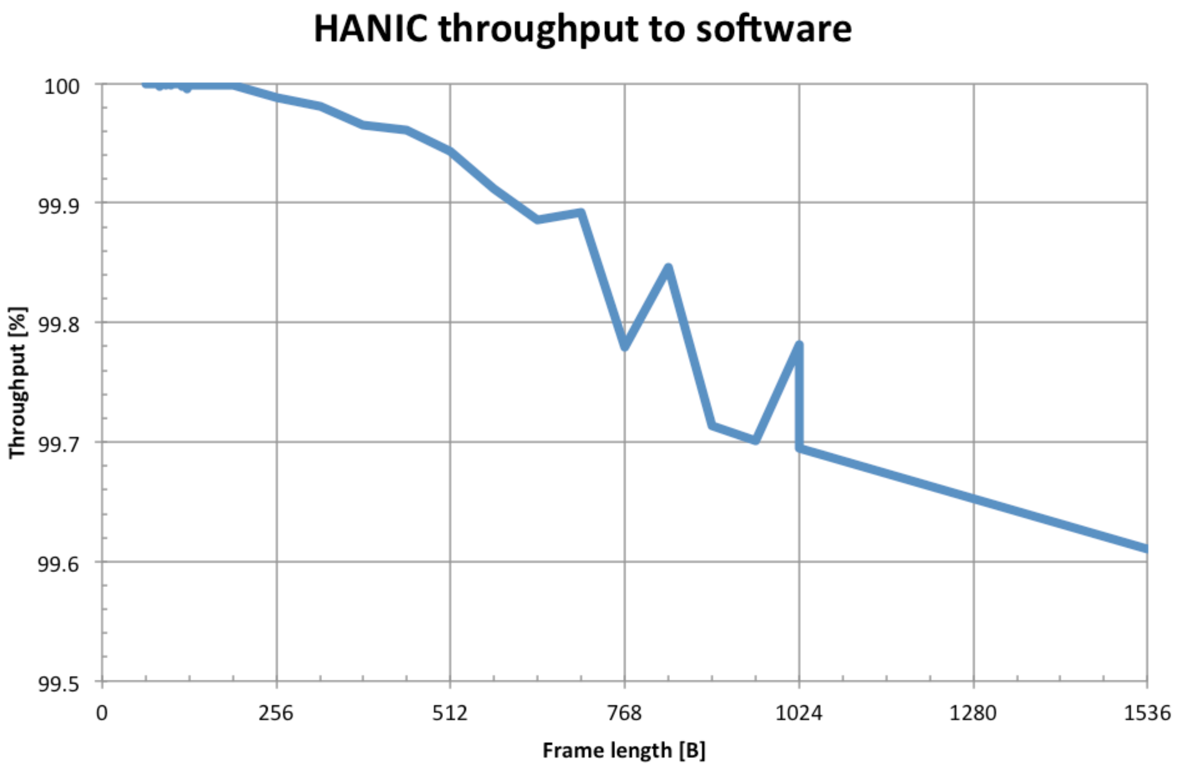

Figure 5. HANIC receive throughput

In addition to throughput, we have also measured the CPU load of the multithreaded packet counter. Figure 6 shows the measured throughput. CPU load in the graph is aggregated from all eight CPU cores, therefore the theoretical maximum is 800 %. It can be seen that the CPU load is directly proportional to the packet rate (rather than the raw throughput). Given the fact that the observed average packet length in the CESNET2 network is roughly around 900 Bytes, it is clear that the CPU has spare computational power to do additional processing of the network traffic.

4 Conclusion

We have presented results of our ongoing research and development of high-speed network monitoring hardware. Our COMBO-100G card and its HANIC firmware offer unique performance parameters. The report demonstrated that our system is able to convey full 100 Gbps Ethernet traffic to the host CPU for further processing. Minor packet drops are possible only in synthetic laboratory conditions and are not likely in any real life deployments.

Our future work will focus on further improvements of the firmware functionality. We have already started experiments with 16 independent DMA channels in order to utilize modern multi-core CPUs. Also the filtering capabilities of HANIC will be significantly improved in the future versions.

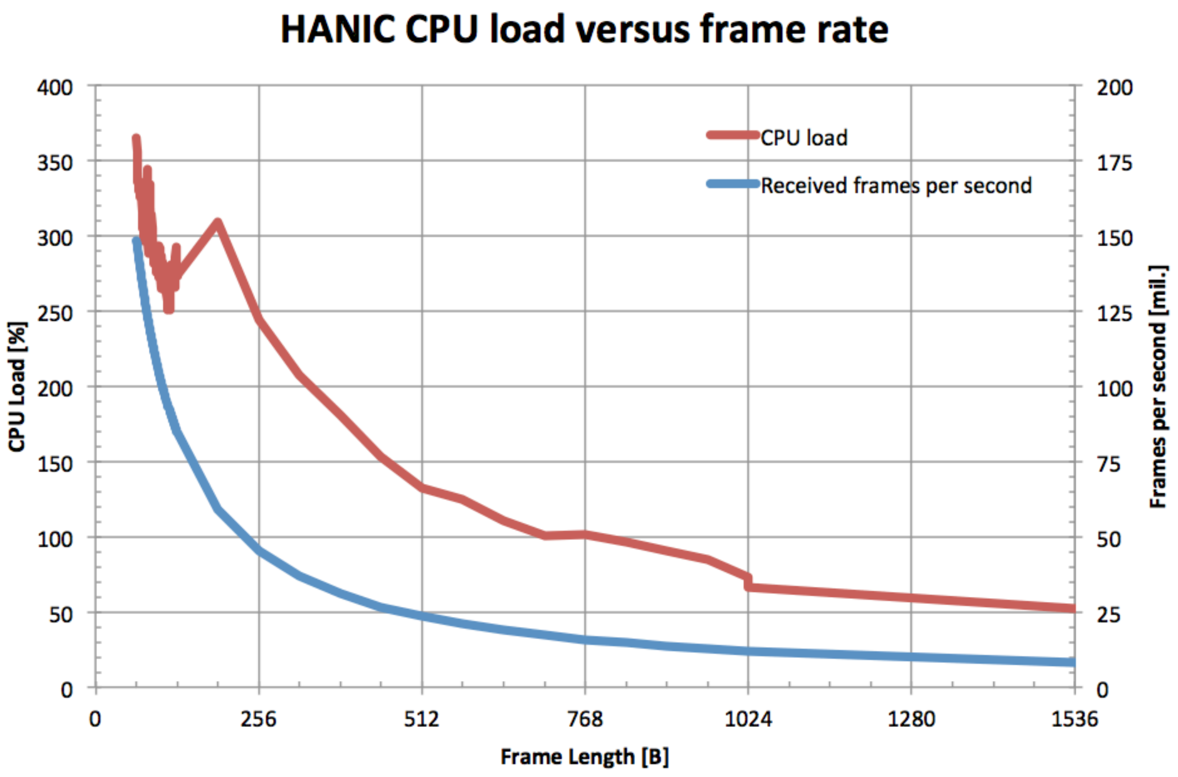

Figure 6. Software load depending on frame rate

References

[1] Intel corp.: *Intel® Ethernet Converged Network Adapters XL710 10/40 GbE.* Available online[1]

[2] Štěpán Friedl, Viktor Puš, Jiří Matoušek, Martin Špinler: *Designing a Card for 100 Gb/s Network Monitoring.*
CESNET technical report 7/2013 Available online[2]

[3] INVEA-TECH.
Available online[3]

[4] Viktor Puš, Lukáš Kekely, Jan Kořenek: *Design Methodology of Configurable High Performance Packet Parser for FPGA.*
In: Proceedings of the 17th IEEE Symposium on Design and Diagnostics of Electronic Circuits and Systems, Warsaw, Poland, 2014

1 `http://www.intel.com/content/www/us/en/network-adapters/converged-network-adapters/ethernet-xl710.html`

2 `http://www.cesnet.cz/wp-content/uploads/2014/02/card.pdf`

3 `https://www.invea.com/en/go/fpga`