Hacking NetCOPE to run on NetFPGA-10G

Pavol Korcek, Vlastimil Kosar,
Martin Zadnik, Karel Koranda
Brno University of Technology, Czech Republic
{ikorcek, ikosar, izadnik}@fit.vutbr.cz
xkoran01@stud.fit.vutbr.cz

Petr Kastovsky
INVEA-TECH, Czech Republic
kastovsky@invea.cz

Categories and Subject Descriptors

C.4 [**Performance of systems**]: Design studies

General Terms

Design, Performance

Keywords

FPGA, NetCOPE, NetFPGA-10G

1. INTRODUCTION

In these days, it is quite common that researchers have an access to a high-speed network infrastructure with 10 Gbps throughput or higher. To fully explore capabilities of such a network, there is a strong initiative to develop hardware-accelerated network cards which are capable of processing network traffic at line rate without a packet loss. The NetFPGA card developed at Stanford University is a straightforward example.

In this abstract, we report on our exercise to port our development framework, called NetCOPE, on NetFPGA-10G card [1] which comes with no software or gateware support so far. The NetCOPE [2] is a configurable gateware and software environment which enables rapid development of network applications on the FPGA acceleration boards. Originally, it was developed by CESNET for COMBO cards [3] but since both cards share much in common porting it should be relatively easy. NetFPGA with NetCOPE can be immediately used as a high-speed packet capture NIC and extended with already available applications' cores such as packet classification [4], network monitoring center [5], flow metering [6], packet generator [7] and others.

2. NETCOPE CONCEPT

The primary goal of NetCOPE is to create a vendor-independent environment that would abstract designer from hardware specifics of various cards. To this end, it provides unified gateware API to various controllers of peripherals such as network interfaces, external memories and PCI-Express bus. It also contains software drivers, libraries and tools to manage operations with the card and data transfers between the card and a host machine. The secondary goal is to provide efficient resource utilization to achieve high-processing performance at the hardware as well as at the software side.

The NetCOPE gateware is composed of three main modules: Generic Interconnection System (GICS) [8], Network module and DMA module. A specific user application core communicates with all three modules as shown in Figure 1. Network module allows the application core to attach arbitrary information (such as timestamp) to an incoming Ethernet frame prior to entering its buffers. Then, it streams out the received frames when the application core is ready. The application core processes the frame and uses DMA module to transfer the data stream to the host memory. DMA module supports up to 32 DMA channels among which the incoming data may be distributed (e.g., flow-wise). GICS comprises a variable-width bus and generic modules which together build an interconnection infrastructure which is connected to PCI-Express module. GICS infrastructure provides slow register interface or faster memory interface for the application core. GICS connects DMA module as well and may also provide additional DMA interface to application core. Except these substantial parts of the NetCOPE platform, there are also some extra modules providing specific functionality – Timestamp module capable of generating 64-bit timestamps synchronized by GPS pulses, modules to work with stream of packets or NetCOPE clock component which may be parametrized to generate user-defined frequency of clocks for the application core. The application core is asynchronously separated from all NetCOPE modules.

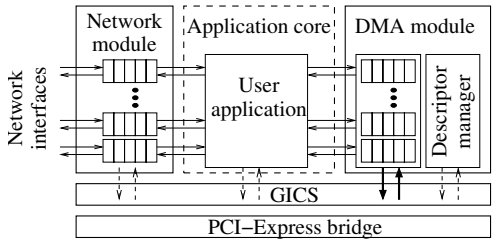

Figure 1: NetCOPE architecture

At the software side, the NetCOPE drivers cover basic input/output operations with hardware card, card initialization, after-boot process, loading gateware, as well as fast DMA transfers using ring buffers. Libraries sitting on top of these drivers provide higher-level API (e.g., libpcap) to access these functions. The most interesting is probably zero-copy data transfer which is achieved by mapping the driver's ring buffers into the contiguous application address space. Moreover, the libraries allow to access same data by several applications concurrently. User application can also read from multiple DMA channels but, based on our experi-

978-0-7695-4521-9/11 $26.00 © 2011 IEEE
DOI 10.1109/ANCS.2011.40

Resource	Used	Available	Percentage
BlockRAM/FIFO	93	324	28%
Slice LUTs	41,549	149,760	27%
Slice Registers	46,920	149,760	31%

Table 1: Resource consumption

ence, the highest throughput is achieved if each application is locked to a given CPU core and assigned single DMA channel. In the basic NetCOPE version, there is separate DMA channel per each network interface.

3. EVALUATION

While porting NetCOPE to NetFPGA we had to face some issues. First of all, NetFPGA is a single card directly equipped with network interfaces while COMBO is a bundle of two cards, mother card and an add-on interface card. As a consequence, it was necessary to move network modules previously placed in FPGA on add-on card to the gateware of the main card. Further, we had to enlarge network module to support four interfaces and propagate this information into an address space. Due to four interfaces we also instantiated DMA module with support of four DMA channels. In order to propagate the information about the modified address space and about the number of DMA channels into the driver we use our own identification component in addition to PCI-Express identification. This allows the driver to transparently work with both cards. Both cards also differ in bijouterie connected to their FPGAs but for our preliminary experiments these are irrelevant.

There were two software modifications necessary. The first was to extend PCI identification table of the driver so it can correctly recognize the card and the second was a minor modification of phyter configuration routine due to different phyter vendor.

The hacked NetCOPE was synthesized into gateware for FPGA on NetFPGA card. Xilinx ISE reports approx 30% of chip utilization (details are in Table 1) after successful place&route with clock frequency of 200 MHz. The basic NetCOPE uses a simple application core which wires together Network and DMA module and operates at the same frequency as NetCOPE itself.

The packet capture results (tested with Spirent Test Center) are not surprising as these are similar to those that we have obtained with COMBO cards and are close to the limit of PCI-Express x8 bus when all overhead of PCI-Express is accounted for, such as 8/10 encoding, overhead of PCI-Express packet headers, flow control packets, link layer packets, etc. [9]. Both cards are compatible with PCI-Express version 2 slot but since both cards use Virtex 5 FPGAs they can only reach version 1 performance due to limited throughput of their IOs. Figure 2 displays the achieved throughput and CPU load when simultaneously receiving packets on two network interfaces without any packet losses according to RFC 2544. The incoming frames were delivered to the simple software application which counted the number of received frames. The application was running as two processes each dedicated to a particular DMA channel and both channels were equally utilized. The configuration of the host is Intel Xeon CPU E5335 @ 2.00 GHz, Supermicro X7DB8 motherboard, 10 GiB DDR2 memory, OS Linux 2.6.18-164.6.1.el5.

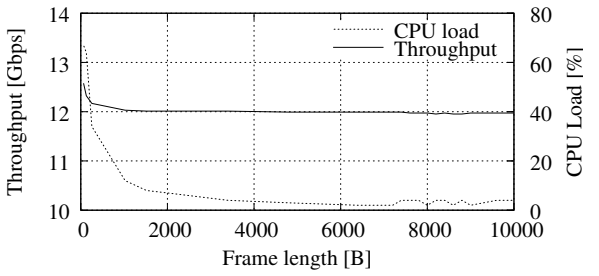

Figure 2: Achieved throughput and CPU utilization for various length of packets

The increased throughput at the shortest frames is caused by smaller overhead of our DMA transfer. The Ethernet standard requires a gap between subsequent packets but this may be omitted if these packets are grouped together and transferred as a payload of a large PCI-Express packet.

4. CONCLUSION

In summary, porting NetCOPE to NetFPGA-10G was successful. It allows NetFPGA to capture and transfer packets in the host at high speed. Our future work will focus on the on-the-fly reprogramming of FPGA (without rebooting the host) which is supported on COMBO but has not been ported to NetFPGA yet.

Acknowledgements

This work has been partially supported by the Research Plan MSM 0021630528, IT4Innovations Centre of Excellence project CZ.1.05/1.1.00/02.0070, the grant BUT FIT-S-11-1 and Sec6net project VG20102015022.

5. REFERENCES

[1] Netfpga web page, www.netfpga.org - netfpga-10g.

[2] T. Martinek and M. Kosek. NetCOPE: Platform for rapid development of network applications. In *Proc. of 2008 IEEE Desing and Diagnostics of Electronic Circuits and Systems Workshop*, pages 219–224, 2008.

[3] Combov2 fpga boards. `http://www.invea-tech.com/products-and-services/combo-fpga-boards`.

[4] T. Dedek and et al. Hardware Packet Filter with IPv6 Support. In *Networking Studies IV, CESNET*, pages 155–166, 2011.

[5] P. Celeda and et al. Hamoc - hardware-accelerated monitoring center. In *Networking Studies IV, CESNET*, pages 107–133, 2011.

[6] M. Zadnik and et al. FlowMon for Network Monitoring. In *Networking Studies IV, CESNET*, pages 45–56, 2011.

[7] J. Matousek and P. Korcek. Precise IPv4/IPv6 Packet Generator Based on NetCOPE Platform. In *Proc. of 2011 IEEE Design and Diagnostics of Electronic Circuits and Systems*, pages 319–324, 2011.

[8] T. Malek, T. Martinek, and J. Korenek. Gics: Generic interconnection system. In *Proc. of 2008 International Conference on Field Programmable Logic and Applications*, pages 263–268, 2008.

[9] A. Goldhammer and J. A. Jr. Understanding performance of pci express systems, 2008.