Packet Processing on FPGA SoC with DPDK

Jan Viktorin
Faculty of Information Technology
Brno University of Technology
Brno, Czech Republic
Email: iviktorin@fit.vutbr.cz

Jan Korenek
Faculty of Information Technology
Brno University of Technology
Brno, Czech Republic
Email: korenek@fit.vutbr.cz

***Abstract*—One of the most important topics of today is a packet processing in data centers with respect to the power consumption and efficient utilization of computational resources. The ARM architecture has proved to be an energy efficient computational system. Together with an integrated FPGA on a single die, it offers potentially a high performance with respect to the power consumption. DPDK – a set of libraries and drivers intended primarily for fast packet processing – is becoming to be a standard approach for packet processing, especially in data centers. In this paper, we exploit the potential of packet processing based on DPDK and FPGA SoC architectures. Especially, we aim at the potential of utilizing the ARM Cortex-A9 and Cortex-A53 CPUs.**

I. Introduction

DPDK[1] is a set of libraries and drivers for fast packet processing. It runs in the userspace and gets advantage of bypassing the operating system kernel's (Linux or FreeBSD) networking stack to reach high throughput in network applications. The DPDK infrastructure provides a core for memory management (utilizing the hugepages extension of certain MMU architectures), lockless buffer rings, zero-copy communication with PCI devices (network cards) and a support for multicore processing. With this approach, it is possible to achieve a significantly higher throughput in comparison to the standard solutions based on the Linux Kernel. DPDK is becoming to be de facto a standard in the fast network data processing (utilized by e. g. FD.io [1], Open vSwitch [2]).

During the last year, DPDK has been extended to support the ARM [3] and ARM64 [4] architectures. This brings a possibility to reduce the overhead of packet processing (and therefore, to reduce the power consumption) by applications running on ARM-based systems. An interesting target architecture for DPDK is the Xilinx Zynq (ARM Cortex-A9 dual-core at 666 Mhz) as it provides a low-power processing capabilities (on the ARM dual-core processing system) and achieves a high throughput by utilizing the tightly integrated FPGA (an FPGA SoC). The next generation of this chip (Xilinx ZynqMP [5]) is based on the ARM Cortex-A53 quad-core processor (64-bit ARM) with improved on-chip communication channels. DPDK can help to split the processing to multiple processes (and CPU cores) and provides capabilities for hardware acceleration of network algorithms in the FPGA.

[1]Data Plane Development Kit: http://www.dpdk.org

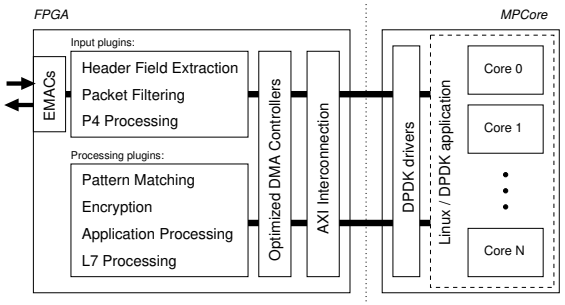

Fig. 1. Packet processing system for DPDK on FPGA SoCs.

This paper shows the potential of utilizing FPGA SoCs running DPDK-based applications intended for data centers and for processing of the network traffic.

II. Architecture

The figure 1 shows a network packet processing system using DPDK on FPGA SoCs It consists of a multicore processor (MPCore), a set of Input Plugins, a set of Processing Plugins and the network interfaces (EMACs). Operations can be offloaded by utilizing the plugins on the RX direction (TX would work in the opposite way):

- Input Plugins – precomputation before reaching any CPU,
- Processing Plugins – performed by CPU on demand after a packet is received.

The Input Plugins can be utilized for packet filtering, hash computations, LPM, early packet aggregation of uninteresting traffic, etc. The Processing Plugins enable to speed up pattern matching, encryption tasks, L7 processing and other application-specific tasks.

Plugins can be loaded dynamically into the FPGA or can be a part of a fixed design depending on the target application. The plugins can be selected by parsing the P4 language [6] specifications, especially the match-action tables part.

A. DPDK and SoCs

The greatest advantage of utilizing DPDK on ARM is a reduction of the underlying operating system's overhead. This comprises a significantly lower number of context switches, bypassing the operating system network subsystem (this can be a disadvantage as it already implements various useful network

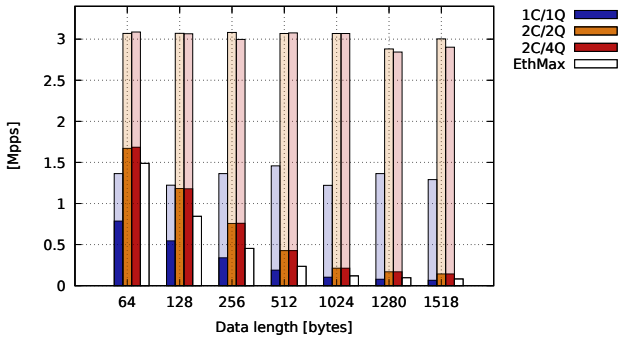

Fig. 2. Packet-rate of loopback on the Xilinx Zynq.

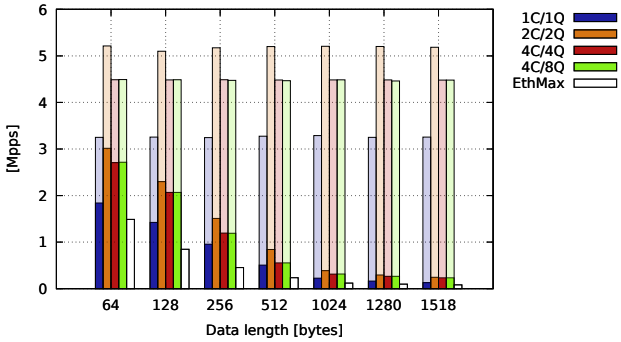

Fig. 3. Packet-rate of loopback on the Pine64.

algorithms), zero-copy approach and an included control of migration of tasks among CPU cores.

The DPDK has been designed to support only PCI-specific devices. There is no standard way how to integrate a SoC-specific (non-PCI) peripherals into DPDK. As a part of this work, we have extended DPDK to support the ARMv7 architecture [3] and recently, a support for SoC-specific peripherals is being discussed on the DPDK's mailing list [7] (for DPDK 16.07, to be released in July 2016).

III. Results

The goal of our measurements has been to show the limits of DPDK-based packet processing on the Xilinx Zynq and Xilinx ZynqMP. As the ZynqMP was not widely available for testing at the time of writing this paper, another SoC was chosen: the Allwinner A64 (Pine64). This SoC is based on the quad-core ARM Cortex-A53, contains a 512 kB L2 cache and operating frequency at 1.2 GHz (ZynqMP would have 1 MB L2 cache and CPU at 1.5 GHz).

To imagine the processing capabilities of DPDK running on the selected platforms, we have measured the SW-only performance of the customized DPDK *null driver* that works as a software loopback. The figures 2 and 3 show the results. The higher packet rate (light colors) was measured by a zero-copy approach. In this case, only the necessary packet handling is performed by the *null driver*. These values show the limits of DPDK itself. The lower packet rate (darker colors) was measured with an added overhead of copying whole packets inside the *null driver*.

The nC/mQ notation denotes an utilization of n cores with m hardware packet queues. The configurations where $m > n$ demonstrates that the performance would not grow without adding more CPU cores. The real throughput of the target system would lay between the throughput of the single-copy and zero-copy approaches.

A. Limits of DPDK-based processing on ARM

The figure 2 shows the two measured cases for the Xilinx Zynq. The real processing capabilities lay between 0.7 Mpps / 528 Mbps and 1.4 Mpps / 916 Mbps per core for the shortest Ethernet frames (64 B).

The results on figure 3 show that the throughput of the Cortex-A53 architecture lays between 1.8 Mpps / 1.25 Gbps and 3.2 Mpps / 2.2 Gbps for the shortest Ethernet frames. It provides slightly higher performace per MHz when compared to the Xilinx Zynq (approx. from 0.9 Mpps to 1.6 Mpps at 600 MHz). However, the Allwinner A64 does not scale when utilizing all four cores. Further investigations suggest that this issue is related to the underestimated L2 cache size.

IV. Conclusion

This paper has exploited the limits of DPDK-based packet processing on the ARM-based systems aiming at the Xilinx Zynq and the Xilinx ZynqMP. We have measured that the Cortex-A53 slightly improves the throughput of the system. However, the performance per Watt should be generally higher for the newer CPU generation. A single Cortex-A53 core at 1.2 GHz can process up to 3.2 Mpps per core on the shortest frames. The processing capabilities can achieve 14 Gbps per core (i. e. 10G Ethernet) on 512 B long frames.

We have proposed a system architecture for the following reseach that will aim at mapping the P4 language to this type of platforms.

Acknowledgment

This research has been supported by the grant MVCR VI20152019001, BUT project FIT-S-14-2297 and the IT4Innovations excellence in science LQ1602.

References

[1] FD.io. (2016) How does fd.io relate to dpdk? [Online]. Available: https://fd.io/news/faqs/how-does-fdio-relate-dpdk

[2] Intel Open Source. (2013) Why intel dpdk vswitch? [Online]. Available: https://01.org/packet-processing/blogs/jdigiglio/2013/why-intel%C2%AE-dpdk-vswitch

[3] J. Viktorin and V. Kosar. (2015) Support armv7 architecture. [Online]. Available: http://thread.gmane.org/gmane.comp.networking.dpdk.devel/27273

[4] J. Jacob. (2015) Dpdk armv8-a support. [Online]. Available: http://thread.gmane.org/gmane.comp.networking.dpdk.devel/27757

[5] Xilinx. (2016) Ultrascale architecture and product overview. [Online]. Available: http://www.xilinx.com/support/documentation/data_sheets/ds890-ultrascale-overview.pdf

[6] P. Bosshart et al, "P4: Programming protocol-independent packet processors," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 87–95, Jul. 2014.

[7] J. Viktorin. (2016) Support non-pci devices. [Online]. Available: http://thread.gmane.org/gmane.comp.networking.dpdk.devel/38486