Embedded Gigabit Ethernet Controller's FPGA Prototype Performance Testing

D.I. Slinkin

Scientific Research Institute for System Analysis of the Russian Academy of Sciences, Moscow,

DmitrySlinkin@gmail.com

Abstract — Modern microprocessors have one or more cores. They can also contain embedded controllers of peripheral interfaces. Field-programmable gate arrays (FPGAs) are used for testing designs of microcircuits as their electric prototypes with reduced frequency, which are usually 100-200MHz. The microprocessor with KOMDIV64 architecture includes an embedded Gigabit Ethernet controller. One of the tasks for the testing was to evaluate the performance of the FPGA-prototype of that controller. The CPU loading resulting from the high bandwidth of Gigabit Ethernet can become a "bottle neck" for the entire system. The direct inserting of the Ethernet controller's model into the microprocessor's FPGA prototype hasn't provided good results due to the core's limited frequency. It was proposed to evaluate the performance of the Ethernet controllers' prototype by installing a board with its FPGA prototype into the PCI slot of an IBM-PC compatible computer. The article begins with presenting existing information on factors affecting the performance of the Ethernet, mentioned in the English-language materials on the subject, as well as software and commonly used hardware CPU offloading methods. The goal was to experimentally evaluate the performance of the Gigabit Ethernet Controller's FPGA prototype in order to confirm the absence of internal delays in the project and in addition, to assess the efficiency of hardware CPU offloading features. For brevity, only the results of "netperf TCP_STREAM" test under Linux are presented. The "gnome-system-monitor" program, or the "netperf" built-in function estimated the CPU load. The results were compared with an analog by Marvell and with a microprocessor after its manufacturing. As a result of the testing, the received speed of the CPU embedded Ethernet is almost equal to the FPGA prototype on IBM PC, but the transmission speed is much higher. The efficiency of the hardware CPU offloading features turned out to be moderate.

Keywords — Gigabit Ethernet, performance, testing, microprocessor, FPGA, Linux, netperf.

I. INTRODUCTION

The network controller operates in a system that consists of hardware, operation system, driver, and application programs. Each component has an influence on the overall performance. At a speed of 1Gbps/second up to 1488 thousand packets of 64 bytes in length or 81 thousand packets of 1512 bytes can pass through the network controller. In the traditional model, for each packet there

will be set an interrupt, which will entail a significant load on the system. It is a rule of thumb, that for working with TCP at a speed of 1Gbps a 1 GHz frequency of the processor is required. For microprocessors with a frequency of about 800 MHz, the performance of the peripheral interface controllers will be limited. The 1000Base-T bandwidth is 125 Mbytes/s. For microprocessor-embedded controllers with an AXI, a bus bandwidth of 1Gbps will be quite sufficient. For modules with a 33 MHz PCI bus available 132 Mbytes/s can become a limitation, if several devices are working at the same time. The real speed of the network controller can be lower than the nominal bandwidth of the channel. In the process of developing a project of embedded Gigabit Ethernet controller, the experimental evaluation of its FPGA prototype's performance was conducted.

II. FACTORS THAT AFFECT THE PERFORMANCE

Let's consider the "Receive livelock" situation. When a packet is received, the Ethernet controller sets out hardware interrupt, which has the highest priority among the interface functions of kernel and application programs. The processor stores the context (the values of registers and the pipeline) in the reserved RAM area and transfers control to the interrupt handler. Operations of the context saving and restoring are difficult for optimization. The interrupt handler integrated into the driver sends the received packets to a higher level queue. To reduce overhead expenses, the driver processes all packets with the "open" descriptors.

After clearing the receiving buffer, the interrupt handler calls the kernel scheduler, which is calling a top-level procedure to process received packets. If packages are destined for application processes, the kernel copies the data to RAM and then continues to schedule its tasks. Here the factors affecting efficiency are the costing of data copying into RAM and crossing the core level boundary.

If a new interrupt is set during working of a top-level procedure, which has a normal priority, then the procedure execution is suspended, and control is again transferred to an interrupt handler. If interrupt rate is high, the top-level packet processing procedures and application processes cannot continue to work. The queue will be overfilled and there will be no free buffers left. The interrupt handler will not be able to transfer packets to higher layers. The

statistics of receiving counter will be increasing, but all packets will be dropped, and the system will stop performing its functions.

A real system has limited bandwidth and is capable to handle packets with arrival rate below the determined maximum. The term MLFRR (Maximum Loss-Free Receive Rate), proposed by K.K. Ramakrishnan, is used to describe this problem. The dependence of the system capacity on the input load is presented in Fig. 1.

Fig. 1. Dependence of system capacity on input load, by materials of [1]

The MFLRR is calculated by the formula:

$$MLFRR = \frac{1}{2 * T_{intr} + t_p + t_a^{sw} + t_a},$$

$$T_{intr} = (t_i^{sw} + t_i)$$

Where:

t_i^{sw} — the context switching time for the interrupt handler;

t_i — interrupt processing time;

t_p —time of packet processing by the receive protocol;

t_a^{sw} —time of context switching for application program;

t_a — packet processing time by application program.

The formula does not take into account cache misses. The considered model uses two interrupts per one incoming packet. The network controller sets a hardware interrupt, when the packet is received and placed in the list of descriptors. The second one is an asynchronous software interrupt (softirq) connected with the transfer of data presented in the form of Socket Buffer (SKB) to the Linux kernel for processing. The most significant times in the model are t_p (from 45% to 77%) and t_a (from 14% to 39%) depending on the system load.

For 10Gbps+ speed controllers cache memory failure is of importance. G. Liao, and L. Bhuyan found out that for a server with an Intel Xeon Quad-Core processor running Linux and a 10Gbps network, the main bottlenecks are: driver operations (> 26%), copying data from SKB buffers

to user buffers (up to 34% depending on data size), and SKB buffers recovery after copying (> 20%) [2].

Software polling method or hardware interrupt moderation are used to reduce the number of interrupts. The polling method is used for disabling interrupts and for transferring control over receiving of packets to the OS. The system itself decides, how many packets it can accept and process, and how many resources should be provided to the kernel and application processes. If too many packets arrive, redundant ones are dropped in the network adapter.

The polling method for Linux, called “NAPI” (New API) [3] works in a following way:

- the network adapter is working in a usual mode with interrupts;
- when receiving the first packet, the driver registers the device in the poll list and disables interrupts;
- from time to time OS is checking the poll list and processes packets;
- after all packets have been processed, the device is removed from the poll list and the interrupts re-start.

The hardware method is called “Interrupt moderation”. Some network cards have automatic configuration of this mechanism. For manual setting, there are two timers available for a programmer: absolute and packet.

The absolute timer starts after receiving the first data packet. The following packages do not affect its operation. After the timer expires, an interrupt is generated and a new report begins. The OS processes several packets at the moment of receiving the interrupt.

The disadvantage of absolute timer is as follows, if only one packet arrives, its receiving will be delayed until the timeout. Therefore, due to the high latency, the absolute timer isn't suitable for low network traffic.

The pooling of interrupts with high input traffic is shown in Fig. 2.

Fig. 2. Interrupts pooling with high input traffic, by materials of [4]

The packet timer is starting, when a packet is received or sent. If the controller has time to process another packet before the timer expires, the timer is reset and restarted. The interrupt is generated over period of time. A packet timer minimizes delays, but it is useless for high network traffic — there can be no interrupts, because the timer is reset every time a packet is received or sent. When the

polling mechanism is used, the system can receive no more than $(Q/2 \cdot T)$ packets per second. The MLFRR will be the minimum between the packet receiving rate and the ratio of processor time, free from interrupt processing, to the sum of times of the receive protocol and the application program [1] (adapted):

$$MLFRR = \min \left(\frac{Q}{2 \cdot T}, \frac{1 - \frac{T_{intr}}{T}}{T_p + T_a} \right),$$

$$T_a = (k_p t_a + t_a^{sw}),$$

$$T_p = k_p t_p$$

Where:

Q – maximum number (quota) of packets merged in a single hardware interrupt;

T – period of the timer;

k_p – number of packets received from the descriptors list in a single software interrupt, $k_p \leq Q$.

The timing diagram of the incoming packets processing during a time poll and asynchronous software interrupt (softirq) for Linux is presented in the Fig.3. Processor idle time is marked by: t_i .

Fig. 3. Packet processing diagram, by materials of [5]

It is recommended to use a packet and absolute timers together. In the case of high traffic, an absolute timer will ensure the delivery of packets. And with medium and low traffic, the packet timer will reduce latency. The disadvantage of the pooling method is the difficulty in selecting the parameters Q and T .

The solution can be in choosing one of three interrupt policies: decreasing latency, maximizing processor unloading, providing maximum performance. The right selection of the interrupt control policy is also relevant for 40Gbps and 100Gbps Ethernet controllers [6].

III. CPU OFFLOADING METHODS

To offload the CPU, manufacturers include into their network adapters hardware mechanisms for calculating checksums for TCP, UDP and IP. The "TCP segmentation offload" mechanism breaks up large blocks of data into

smaller packets. The efficiency of hardware unloading of CS is estimated at 10%. The "Zero-Copy" technology involves direct data transfer, using DMA, between the network controller and the application program, bypassing intermediate buffers in the OS kernel space.

When done in this way, data copy operations are reduced, and processor time is saved. The authors of [7] proposed a TOE module (TCP / IP Offload Engine) for Linux using "Zero-Copy". The testing showed a decrease in TCP/IP time by 11% at receiving, and 10-20% at transmission.

The MSG_ZEROCOPY software interface proposed by Willem de Bruijn, which is included in the Linux 4.14 kernel, is designed to work with TCP, UDP, RAW, and packet sockets. The performed experiments have shown that for the "netperf" test, TCP_STREAM, 10Gbps, the MSG_ZEROCOPY allows up to 92% saving of the cycles (processor) and 39% of the total system cycles. Savings in performing of some real-life tasks amounted to 8% of the time and 5% of requests per second [8].

The "flow control" function is used to regulate traffic depending on the bandwidth of the receiving adapter. In a buffer overflow threat, the partner is given a special pause code. "Flow control" works only in full duplex mode. In a half-duplex mode, the "Back pressure" method is used, which involves sending a collision signal to the transmitting port and consequently suspending the transmission. As a result, the processor does not spend resources on re-sending data.

During the operation of high-level protocols and application programs, a "Starvation of transmits" problem can appear and is connected with a higher priority of receiving packets in comparison with transmission. Reducing the transmission priority is explained by desire to decrease packet loss with high input traffic. A situation occurs when interrupts from the transmitter have a lower priority than receiving, if events from the receiver are processed first by the driver, or if the pooling mechanism for the transfer is set to low priority. If the system is overloaded for a long time, the low priority of the transmitter leads to a reduction or even a complete stop of the transmission process - packets can be waiting for sending, but the transmitter interface is free.

IV. THE PERFORMANCE EVALUATION METHOD

The Experimental performance evaluation of the FPGA prototype was part of the Ethernet controller project testing. In parallel, integration testing of the FPGA-prototype was performed via loading Linux and running driver on a computer. The second task was to compare performance with the foreign analogue. The third goal was to assess the effectiveness of the hardware processor unloading features, implemented in the FPGA-prototype. Let's consider possible approaches to performance testing:

1. Installation of the PMC mezzanine with the FPGA prototype into a serially manufactured processor module. In this case, the Ethernet performance will be limited by

the frequency of the microprocessor, which is leveled to hundreds of MHz.

2. Embedding of the Ethernet controller as a functional block into the FPGA prototype of microprocessor. In this case, the FPGA is unsoldered on a “research board”, which is specially designed for developing and debugging a project of a microprocessor, and including all the peripheral interfaces. The advantage is involving the AXI bus and approximation to the integrated network controller’s operating conditions. However, the performance will be limited by the frequency of the FPGA, about 100 MHz, which is clearly not enough.

3. Installation of the PMC mezzanine with the FPGA prototype into an IBM PC compatible computer, Fig. 4. In this case, the microprocessor’s performance, declared by the manufacturer at 2-3 GHz, will not become a bottleneck. The frequency of the FPGA is enough to estimate the performance of the future Ethernet controller, at speeds up to 1Gbps. There is a possibility to compare a prototype Ethernet controller with foreign analogues, when working under Linux because the distribution kit includes the necessary drivers. An additional advantage is the integration testing of the FPGA-prototype with Linux and IBM-PC compatible computers.

Fig. 4. The FPGA prototype of the 1Gbps Ethernet controller installed in an IBM-PC compatible computer

The key results have been obtained, when the FPGA prototype is installed into the IBM-PC. For brevity, the article presents only the results of the test “netperf, TCP_STREAM” under Linux. The maximum packet length MTU 1500 is common for computer networks. The

total processor load was estimated using the “gnome-system-monitor” program for IBM-PC compatible computers, and the “top -b | grep CPU” command for the board with the microprocessor. The built-in function “-c” of the “netperf” test allows to evaluate CPU utilization by a specific process (test), its results are presented in brackets.

The IBM-PC compatible computer was used with Intel Pentium 3.2 GHz processor, SiS 655FX chipset, DIMM DDR-400 RAM, 1GB PCI bus frequency 33MHz. The hyper-threading mechanism was turned off - the processor worked as a single-core one. Since the measurements were taken at different times, the versions of the Linux kernel were different.

V. THE RESULTS OF TESTS

The FPGA prototype of the Ethernet controller was installed in the PCI slot of the PC. Testing has shown the dependence of the results on the chipset. Increasing MTU to 9000 (“Jumbo Frames”) boosted the performance [9]. The effectiveness of the built-in interrupt moderation mechanism was evaluated with the following settings: 4000 interrupts per second, packet timer 40μs the Linux kernel 2.6.32. The results are presented in table 1.

Table 1

The results of interrupt moderation for the FPGA-prototype installed in the IBM-PC

Interrupt Moderation is Disabled		
	Speed, Mbps	CPU Load
Transmitting	457	27%
Receiving	644	50%
Interrupt Moderation is Enabled		
	Speed, Mbps	CPU Load
Transmitting	465	30%
Receiving	591	35%

To evaluate the effectiveness of the of TCP checksum offloading hardware mechanism, an updated driver that involved NAPI was used on Linux kernel 3.2.54. The interrupt moderation mechanism was disabled. The results are presented in table 2.

Table 2

The results of TCP checksums offloading hardware for the FPGA-prototype installed in the IBM-PC

Hardware offloading of TCP checksums is Disabled		
	Speed, Mbps	CPU Load
Transmitting	402	31% (49%)
Receiving	570	54% (98%)
Hardware offloading of TCP checksums is Enabled.		
	Speed, Mbps	CPU Load
Transmitting	405	32% (50%)
Receiving	648	44% (99%)

The TCP checksum offloading on receiving increased the performance by 12%. For data transmit, the checksum offloading did not affect performance. However, no performance increase was registered in comparison with the table 1.

For comparison with a foreign analogue, Marvell 88E8003 network controller was installed in the instrumental IBM-PC at the same slot as the FPGA - prototype. The driver used the NAPI mechanism. The results are presented in table 3.

Table 3

Results for the Marvell controller in the IBM-PC

	Speed, Mbps	CPU Load
Transmitting	551	34% (37%)
Receiving	654	42% (99%)

For foreign network adapters, it is possible to receive the result of TCP_STREAM testing up to 940-960Mbps. But for the configuration of existing stand, the FPGA prototype performance turned out to be comparable with Marvell 88E8003.

After manufacturing of serial microprocessors with the KOMDIV64 architecture operating at 800 MHz frequency, the performance measurements of the integrated network controller were performed. Linux with a kernel 4.14.26 was used. The results are shown in Table 4.

Table 4

The test results of the embedded 1Gbps Ethernet controller at the board with microprocessor

	Speed, Mbps	CPU Load
Transmitting	688	98% (100%)
Receiving	714	98% (100%)

The receiving speed is close to the speed of the FPGA prototype on the IBM-PC, but the transmission speed is much higher. When running the "netperf UDP_STREAM test on a board with the microprocessor, the following result was obtained: 962 Mbps for transmission (of which 944 Mbps was received by the partner) and 541 Mbps for receiving (the result of the transmission of the partner 962 Mbps).

V. CONCLUSION

During the study the method of performance testing of embedded Gigabit Ethernet controllers FPGA-prototypes was tried out. The standard approach is to use the Linux and the "netperf" test. Usually, for testing at the SRISA RAS, are used self-developed prototype boards.

That's why installing a FPGA prototype in an IBM-PC was unusual solution for us. It allowed escaping the problem of low frequency of processor modules, initially available for testing. The author of the article prepared a hardware and software stand for testing the performance of Ethernet controllers.

The task of evaluating of the efficiency of the CPU offload hardware features was formulated in cooperation with the specialist-developer during the testing. The author adapted a driver for working with the Linux 2.6.32 kernel, combining the Linux and performed testing. The speed of the embedded network controller exceeded the expected values. Microprocessors with embedded Gigabit Ethernet controllers are serially manufactured. Production of these modules is being prepared for launching.

ACKNOWLEDGMENTS

Thanks to Alexey Krynitsky - for consultations on the hardware. Thanks to Anton Pavlov, Pyotr Mamonov and Alexey Kuleshov for consultations on drivers and Linux.

REFERENCES

- [1] A. Indiresan, A. Mehra, and K. G. Shin, Receive Livelock Elimination via Intelligent Interface Backoff/ TCL Technical Report/ University of Michigan. 1998. URL: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.3.1.41> (access date: 18.03.2018).
- [2] G. Liao, L. Bhuyan Analyzing performance and power efficiency of network processing over 10 GbE // *Journal of Parallel and Distributed Computing*. URL: <https://doi.org/10.1016/j.jpdc.2012.02.016> (access date: 18.03.2018).
- [3] «NAPI» / The Linux Foundation. URL: <http://www.linuxfoundation.org/collaborate/workgroups/networking/napi> (access date: 18.03.2018)
- [4] Interrupt Moderation Using Intel®GbE Controllers / Intel Corporation. 2007. URL: <http://www.intel.com/content/dam/doc/application-note/gbe-controllers-interrupt-moderation-appl-note.pdf> (access date: 18.03.2018).
- [5] X. Chang, J. K. Muppala, Z. Han, J. Liu Analysis of Interrupt Coalescing Schemes for Receive-Livelock Problem in Gigabit Ethernet Network Hosts / IEEE International Conference on Communications. 2008. p. 1835-1839. URL: <http://repository.ust.hk/ir/bitstream/1783.1-3229/1/045333881.pdf> (access date: 18.03.2018).
- [6] K. Liang, A. Kleen, J. Brandenburg Improve network performance by setting per-queue interrupt moderation in Linux / 01.org (Intel open source). 2017. URL: <https://01.org/linux-interrupt-moderation> (access date: 18.03.2018).
- [7] L. Tianhua, Z. Hongfeng, L. Jie, Z. Chuansheng Design and Implementation of Zero-Copy for Linux / International Journal of Computer Information Systems and Industrial Management Applications (IJCISIM), 2011, p.9-18/URL: http://www.mirlabs.org/ijcism/regular_papers_2011/Paper2.pdf (access date: 18.03.2018).
- [8] W. de Bruijn Sendmsg copy avoidance with MSG_ZEROCOPY. draft. 2017/ URL: <https://netdevconf.org/2.1/papers/netdev.pdf> (access date: 18.03.2018).
- [9] D. Slinkin Testirovanie proizvoditel'nosti prototipa kontrollera Ethernet 1 Gbit/s (Testing the performance of a 1 Gbps Ethernet controller prototype) // *Elektronika, mikro- i nanoelektronika. Sbornik nauchnyh trudov*. Pod red. V.Ja. Stenina. Moscow. 2013. S. 208-210. (rus) Zagurenko A.G., Korotovskikh V.A., Kolesnikov A.A., Timonov A.V., Kardymon D.V. Techno-economic optimization of the design of hydraulic fracturing. *Neftyanoe khozaystvo - Oil Industry*, 2008, no.11, pp. 54-57 (in Russian).