Proceedings of the 9th International Conference on Applied Informatics
Eger, Hungary, January 29–February 1, 2014. Vol. 1. pp. 259–268
doi: 10.14794/ICAI.9.2014.1.259

Line-rate packet processing in hardware: the evolution towards 400 Gbit/s*

Tamás Tóthfalusi, Péter Orosz

University of Debrecen
tothfalusi.tamas@gmail.com
oroszp@unideb.hu

Abstract

Network packet parsing and packet forwarding are general tasks for all routing devices. However, the requirement for line-rate packet processing, independently from the transmission technology, is a common demand against core network equipments. In this paper, we investigate programmable hardware architectures (i.e., Field Programmable Gate Array - FPGA) as central building blocks of the data plane for state-of-the-art 100 Gbit/s network devices. We reveal the benefits and drawbacks of the available hardware architectures (such as Network Processors, Application-Specific Integrated Circuits (ASIC) and FPGAs, respectively). After showing the general packet processing steps on programmable hardware, we describe the problem space of line-rate packet processing in relation to the evolution of transmission technologies, i.e., 1, 10, 100 Gbit/s Ethernet and beyond. Moreover, we present design trade-offs, such as operational frequency, data path width and resource requirement, covering the 1 to 400 Gbit/s throughput range and we propose best practices for their hardware designs.

Keywords: Packet Processing, Data plane, FPGA, 100 Gbit/s Ethernet, 802.3ba

1. Introduction

Data transmission technologies are continuously progressing and are focusing on accelerating speed and managing bandwidth growth. Beyond the widespread 1 Gbit/s and 10 Gbit/s Ethernet links, 100 Gbit/s Ethernet is currently the available state-of-the-art technology, and enhancing Ethernet bandwidth to 400 Gbit/s will be the next step in the evolution. High performance routing and switching

*The research was supported by the European Union and the State of Hungary, co-financed by the European Social Fund in the framework of TÁMOP 4.2.4.A/2-11-1-2012- 0001 'National Excellence Program'.

devices and the related evolving techologies typically appear in the core network first, where packet processing tasks should be always performed at line-rate, in a lossless way. Where are the boundaries of software-based packet processing? Why hardware acceleration comes into the picture to assist networking tasks? This paper aims to answer these questions by introducing typical software bottlenecks, and investigating why hardware acceleration is inevitable for a software based packet processing system when it goes beyond 1 Gbit/s. We demonstrate benefits and drawbacks of the available hardware devices that are able to accelerate a packet processing systems. Moreover, we investigates the evolution of 802.3 media access control (MAC) through design trade-offs and best practices for a line-rate processing and forwarding system.

2. Software bottlenecks

In a x86-based computer, the arrival of a packet triggers an interrupt request. In the legacy case, one interrupt is generated per packet, which results in excessive CPU load at high packet rates. In addition, a network interface card (NIC) has one incoming queue, which can be associated to one CPU core only. However, in server category NICs multi-queueing is supported at packet reception. This design distributes the packets between a number of incoming queues based on some protocol fields (typically IP Source and Destination Addresses). However, each queue is still assigned to only one CPU core. The resources are shared, and the CPU time available for packet reception is controlled by the process scheduler. In this design packet loss depends on the saturation level of the queues and the length of the burst that can be buffered within the system. That is why the efficiency of software-based processing depends on the load of the shared resources.

In a fully saturated 10 Gbit/s Ethernet link, 1.5 x 10^7 packets should be transmitted in each second in the worst case scenario, which a purely software-based packet processing subsystem cannot cope with. Accordingly, hardware assistance is inevitable for line-rate operation beyond 10 Gbit/s

2.1. How hardware can support software-based packet processing?

A limitation in a software can be an advantage in hardware context and vica-versa. In a clever design, software and hardware functions can complement each other. Notably, the main software bottlenecks are the CPU intensive processing tasks such as packet reception, parsing and classification, which can be effectively offloaded to hardware. Nevertheless, there are some hardware drawbacks, because of the physical limitations. Table 1 summarizes the general HW/SW trade-offs. The general benefit of a hardware solution is the real-time processing capability without the need for storing incoming data. However, the architecture is not easily scalable for new protocols and for dynamic protocol depth because of the limited

number of logic cells. Packet classification trade-offs are throughput (in software), which is a hardware strength, and the number of filtering rules (in hardware).

Processing task	HW Bottleneck	SW Bottleneck
Reception	Type of I/O pins	Lossless operation
Parsing	Protocol encapsulation depth Appearing new protocols and fields	Real-time operation
Classification	Number of filter rules	Throughput

Table 1: HW/SW Bottlenecks

3. Hardware acceleration trade-offs

For high performance network equipments. there are three common hardware devices, i.e., network processor (NP), Application Specific Integrated Circuit (ASIC) and Field Programmable Gate Array (FPGA), which are able to assist low-level networking tasks and cooperate with the control plane. NP is a software programmable device, designed for general packet processing tasks at line-rate (e.g., pattern matching, data manipulation, field decrementing, CRC re-calculation, packet segmentation and reassembly). The main resource of the NP is the programmable processing engine (PPE), which is not reconfigurable at hardware level. However, PPEs can work in parallel and/or in pipeline architecture, depending on the task and the internal hardware structure.

ASIC is actually a custom IC, designed for specific tasks. The product is constructed through mask layers, which are used to define the IC's architecture and interconnection. The planning phase is a very time-consuming process, and the extremely high cost of prototyping can also be a disadvantage. Engineers can design it for general or for custom tasks, since the mask layers can be customized. The full system is integrated to one chip, which can guarantee high throughput and enery efficiency. Its logic is not reprogrammable in field.

The resources of an FPGA are logic elements in a regular array layout. The interconnection between the logic cells is reprogrammable, which is generated through software. The FPGA chip can work on general and custom tasks. Its an enabling technology that accelerates the full development cycle since the planning phase is significantly shorter, and the internal architecture of the prototype can be modified without reproducing the chip.

The main difference of these devices is that FPGA is reconfigurable at hardware level, which is a limitation of the other two chips. The connections within the FPGA chip is fully or partially reconfigurable, even through a remote connection, or even

through internal control states, which makes it practical for remote management functions. Another advantage is that innovations and state-of-the-art technologies evolve faster in FPGA devices, which means that a 100 Gbit/s-capable design can be implemented today even on medium category devices appeared on the market 5 years ago. In contrast, 100 Gbit/s-capable NPs, which are announced in 2013, belong to the high-end category. In summary, the FPGA technology provides features for state-of-the-art network tehnologies, and are more flexible for custom packet processing tasks.

4. Lossless packet processing in FPGA: hardware design trade-offs and best practices

An FPGA device can very effectively assist a software solution in the critical packet processing tasks (e.g., reception, parsing, classification), but it also have limitations, which are traceable to the physical resource restriction. A general design goal of a network packet processing system is the high throughput, which means, that the hardware modules within the FPGA device must operate at a relatively high frequency. This minimum frequency is a constraint (from design perspective) for the connection path between the logic elements. During the place and route process the designer software tries to meet all constraints, but the achievable operations per clock cycle can be often a design limitation factor. In addition, the maximum operational frequency is an FPGA bottleneck, because it lies between 20 MHz and 400 MHz, even in the state-of-the-art category devices. The available frequency for the data path also depends on the logical complexity. In order to get a high operational frequency, as a general rule, simple operations should be used in each clock cycle, In this case the data path is shorter and therefore signals can propagate at a higher frequency. Another problem can be the different clock domains, which is a typical network packet processing problem. Notably, the receive modules operating on different clock source from the transmit modules. The synchronization has to be solved. Finally, a basic hardware bottleneck is the limited physical resources. During the planning phase, the main goals should be summarized, to choose an appropriate FPGA chip family, because in later implementation phases the size of the chip, the number and types of I/O PINs and the number of logic elements is not extendable. After the general hardware design trade-offs this paper will describe the line-rate packet processing steps in relation to the evolution of the transmission technologies.

4.1. 1 Gbit/s

The operational frequency of the generic 1 Gbit/s Ethernet MAC (Media Access Control) hardware module is 125 MHz, and its data path is 8-bit wide. The physical size of the MAC module inside the FPGA is relatively small, typically, not more than 5% of a low-end FPGA. The clock frequency is not a challange even in a low-end FPGA, such as Spartan-3A or Spartan-6 from Xilinx inc., which are good

choices for basic packet processing tasks on 1 Gbit/s Ethernet links, i.e., 5-tuple packet parsing or packet timestamping. Because of the 8-bit data path, metadata for packet parsing or packet classification has to be collected byte-by-byte. For example, destination MAC address needs 6 clock cycles to be stored before a parsing step [1]. At 1 Gbit/s bandwidth the processing system has enough clock cycles to perform the main packet processing tasks, according to the worst case, which is 64 clock cycles for a 64-byte Ethernet frame. In a best practice design the entire user data path is 64-bit wide within the chip. In that case, the processing modules have 8 clock cycles to parse and classify each incoming packet, which period is long enough to enable using finite state machines.

4.2. 10 Gbit/s

Stepping up to 10 Gbit/s, the operational frequency of the Ethernet MAC module is increased to 156.25 MHz, and its data path is 64-bit wide. The size of the on-the-market available MAC IP cores varies between 5% and 25% of the logic elements, depending on the type of the FPGA chip. The 64-bit data path enables more protocol fields to arrive in one 64-bit word. Therefore more parsing information can arrive in one clock cycle. Well prepared finite state machines can handle the processing phases effectively, but each module has to store or pass the incoming data to another module for further processing within 8 clock cycles (see the worst case, which is 8 clock cycles). The higher operational frequency (156.25 MHz) requires a mid-class hardware device. However, stepping up to this bandwidth does not imply new internal design in terms of data path width and number of clock cycles between consecutive packets. Handling multiple data lanes increases the size of the 10 Gbit/s MAC module to even 25% of the chip, depending on the target FPGA family.

4.3. 100 Gbit/s

Operating at 100 Gbit/s implies significantly higher core frequency as well as extremely large, 512-bit data path. In this case, every incoming 512-bit word can contain a new, minimum-sized Ethernet frame. This property raises many processing issues. First of all, a processing module cannot store the incoming packets, otherwise it should store 1.5 x 108 packets per second, in worst case. Another challenge is parsing unknown protocol structures, since there is only one clock cycle to decode the protocol fields and classify the packet based on the result. Meanwhile, a new packet can arrive at the next rising edge of the clock signal. In addition, design planning phase is also more complex: the Media Access Control sublayer may be implemented with two significantly different data path handling methods: segmented and non-segmented. Selecting the appropriate mode enables us to optimize the trade-off between frequency and data path width. The only constraint the design should meet is the line-rate processing. The bitrate rules the values of the mentioned two parameters (Fig. 1).

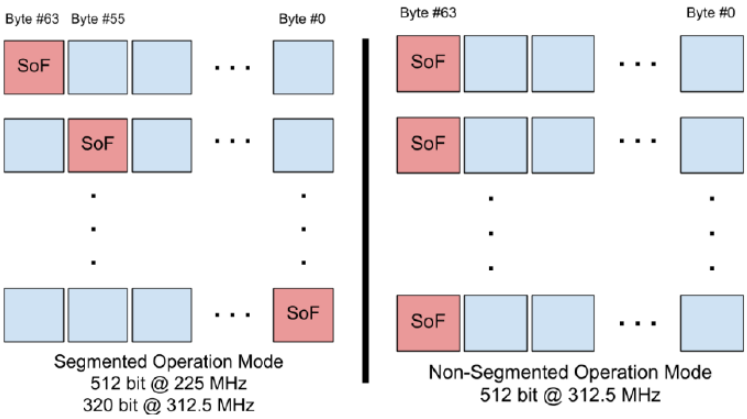

Figure 1: Design trade-offs (Segmented / Non-segmented)

Non-segmented mode is the easier to work with, because the first byte of the packet is always placed at the beginning of a 512-bit word. For example, if the packet ends in the current word, the new incoming packet should start always in the successive word, and the first byte of the packet is always in the byte position #63. Due to the Start of Frame (SOF) synchronization process, the frequency is higher than in segmented mode for the same data path width. The general operational frequency of the MAC in non-segmented mode is 312.5 MHz. Operating a complex logic, i.e., 100 Gbit/s MAC at such a frequency is a real design and implementation challange in FPGAs, since higher frequency implies smaller latency within the architecture.

Considering the internal design of the MAC, segmented operation mode is more straightforward to implement. Nevertheless, due to the additional logics, it results on larger application modules. The main difference is the positioning of the first byte of a new package, which, in this case, can be placed at every 8-byte boundary of the 512-bit word. Since the „Start of Frame" signal is not synchronized to the next word's first byte, frequency is lower than in non-segmented mode, The common operational frequency can be 225 or 312.5 MHz, depending on the data path width.

There are design trade-offs between segmented and non-segmented modes, and there is no generic solution. The architecture is depends on the purpose. To maintain the line-rate processing, the segmented MAC enables a lower operational frequency, and thus the designer can use more complex operations, or alternatively, the data width can be narrower, while keeping up the same frequency. In the latter case, there is higher probability for a protocol field to span over two or more consecutive 320-bit words. For example, a 60-byte IPv4 header may span over even 3 x 320-bit words (see Fig. 2), while using 512-bit data path, the 60-byte IPv4 header can be delivered in 2 x 512 bit words, as with non-segmented processing. In segmented mode the number of resources occuppied by the MAC module is smaller

since there is no synchronization logic. However, offset and index calculations need extra logic to listen to the first byte of the packets. In non-segmented mode the parsing module does not need this extra resource at the cost of a larger physical size.

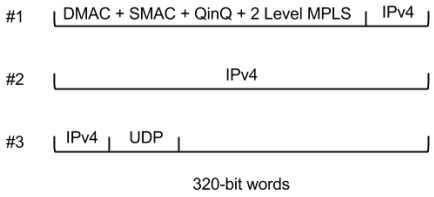

Figure 2: IPv4 header overflow

In a parsing engine, the internal modules have to ensure the decoded protocol fields, so called metadata, to be handled in-sync with the packet they belong to. In the classification phase the destination of the packet depends on these metadata. In non-segmented mode, due to the static position of the first byte, the path of metadata can be fixed. In segmented mode, the position of the first byte is variable and therefore in-sync handling should be implemented.

In a 100 Gbit/s throughput capable subsystem the type of the FPGA family is limited by the type of the I/O transceivers. High throughput transceivers (e.g. GTH and GTZ in Xilinx devices) are available only in high-end devices. Moreover, multi-lane architectures (i.e. ten 64-bit lanes) with multiple transceivers occupy a dominant part of the chip. A 100 Gbit/s MAC module can own even the half of the hardware device.

The sequential operation is a current implementation in the previous 1 Gbit/s and 10 Gbit/s link speed, which is not a viable option in a 100 Gbit/s engine. The minimum Ethernet frame size and the data path width together gives a design limitation for the 100 Gbit/s processing modules, where, in the worst case, a minimum sized packet can arrive in every clock cycle. Because of these property the sequential operation with temporary data storing cannot guarantee loss-less operation. The part of a computation should pass to another processing step, where each processing stage (i.e. pipeline stage) operates at high frequency with simple operations. In conclusion, the best practice is the pipeline architecture, where each pipeline stage has its well defined task.Collecting main design goals, selecting an appropriate FPGA family and determining the operation mode of the 100 Gbit/s MAC are very critical parts of the planning stage, since changing the operation mode is impractical at a later stage.

4.4. 400 Gbit/s

While there is no 400 Gbit/s-ready MAC IP core on the market, we estimates its operational parameters. As we showed previously, the operational frequency of an FPGA device is limited. Maximum frequency in high-end devices is not higher

than 400 MHz. Due to this constraint, the 400 Gbit/s MAC should operate in the 300-400 MHz range. For a backward compatibility with 100 Gbit/s, 312.5 MHz can be a reasonable choice, at the cost of a significantly wider data path (e.g., 2048 bit).

Since the smallest Ethernet frame size remains 64 bytes, one 2048-bit word may include multiple consecutive packets. Accordingly, a 400 Gbit/s-capable packet processing hardware should work with parallel processing engines to sustain the lossless operation. In this arrangement, each processing engine operates at 312.5 MHz with 512-bit internal data path, enabling to process one 64-byte packet per clock cycle. This layout gives the best practice, where the structure implies an appropriate packet distribution system, i.e., a scheduler, which handles the incoming packets between engine instances.

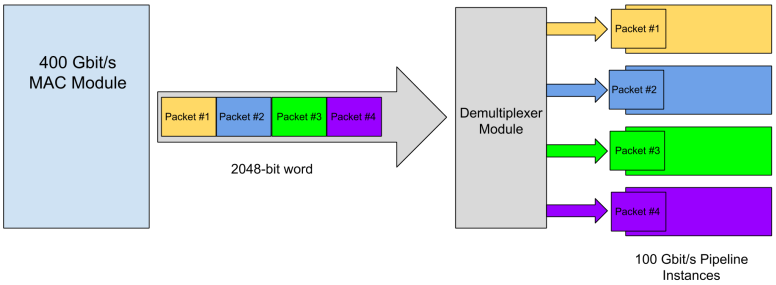

Figure 3: 400 Gbit/s data path

Bandwidth (Gbit/s)	Internal Data Path (bit)	Frequency (MHz)	MAC size (% of the chip)
1	64	125.00	< 5
10	64	156.25	5-25
100	512	312.50	25-50
400	~2048	~312.50	~50

Table 2: Comparison table

Bandwidth (Gbit/s)	Processing time of minimum- sized packets (in clock cycles)	Best Practice architecture
1/10	8	Finite State Machine
100	1	Pipeline architecture
400	1	Multiple Instances of 100G Pipelines

Table 3: Comparison table

5. Related works

Muhammad Bilal Anwer et al. proposed a packet processing system in hardware, which design is based on a NetFPGA-1G platform [1][2]. The architecture is a pipeline-based design, where the modules are able to forwarding at line-rate parallel with OpenFlow switch operation, and IPv4/IPv6 decoding.

Petr Kobiersky et al. proposed a packet header analysis architecture based on a Virtex-5 FPGA device [3]. The implemented modules are able to operate on 20 Gbps network links, where the parser module is generated from XML protocol scheme.

Another NetFPGA-1G based packet parser design is the Kangaroo system [4], which is proposed by Christos Kozantis et al. This architecture is able to operate at 40 Gbps throughput, where the optimal walk of the parser tree is calculated through an offline algorithm.

Weirong Jiang et al. proposed a decision-tree-based hardware design, where the architecture is a dual-pipeline, multi-field packet classification system, which is able to operate at 40 Gbps throughput [5]. [6] introduces a TCAM (Ternary Content Addressable Memory) based classification algorithm, implemented on FPGA. The decoded header informations are split to subfields, where very fast AND operations are used to calculate the result. The design uses 5-tuple rule sets, where the maximum achievable throughput can be even 100 Gbps. [7] presents a 12-tuple based packet classification system, where the architecture is a multi-pipeline, decision tree based design. The maximum achievable throughput is 80 Gbps, in a case of a minimum sized packets.

Thilan Ganegedara et al. proposed a bit vector based lookup scheme, which is implemented on an FPGA device [8]. The parallel hardware architecture is able to operate at 400G+ throughput for minimum sized packets.

6. Conclusion

After showing the software design limitations in high-speed networking we presented how hardware devices can assist to eliminate the bottlenecks of the software based systems. Layer-2 and Layer-3 packet parsing and packet classification can be very effectively performed in hardware. An FPGA device can be a good alternative for a packet preprocessing system, because of the parallel architecture and its support of high-end interface technology. After the hardware acceleration trade-offs, we showed the general hardware based Media Access Control (MAC) operation modes from 1 Gbit/s to 400 Gbit/s. 100 Gbit/s is a big step from the previous standards (1 and 10 Gbps), not only in frequency, but in data path width and operation modes too. We discussed the benefits and drawbacks of the Segmented and Non-Segmented 100 Gbps MAC operation modes, which can be a design bottleneck in a 100 Gbit/s system, according to the type of the FPGA chip. In addition, we showed a novel approach for the 400 Gbit/s network links, where the data path width of the MAC module will be four times the data path width of the 100 Gbps

MAC module. Finally, we summarized the best practices in 1 Gbps to 400 Gbps throughput capable hardware designs, where the general solution is the pipeline, or multi-pipeline architecture with simple operations per clock cycle.

References

[1] M. B. ANWER, M. MOTIWALA, M. BIN TARIQ, N. FEAMSTER, SwitchBlade: A Platform for Rapid Deployment of Network Protocols on Programmable Hardware, ACM SIGCOMM (2010), 183-194.

[2] http://www.netfpga.org

[3] P. KOBIERSKY, J. KORENEK, L. POLACK., Packet Header Analysis and Field Extraction for Multigigabit Networks, 12th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (2009), 96-101.

[4] C. KOZANITIS, J. HUBER, S. SINGH, G. VARGHESE, Leaping multiple headers in a single bound: wire-speed parsing using the Kangaroo system, 29th IEEE Conference on Computer Communications (2010), 830-838.

[5] W. JIANG, V. K. PRASANNA, Large-Scale Wired-Speed Packet Classification on FPGAs, ACM/SIGDA International Symposium on High Performance Interconnects (2004), 23-24.

[6] W. JIANG, V. K. PRASANNA, Field-Split Parallel Architecture for High Performance Multi-Match Packet Classification Using FPGAs, 21th Annual Symposium on Parallelism in Algorithms and Architectures (2009), 188-196.

[7] W. JIANG, V. K. PRASANNA, Scalable Packet Classification on FPGA, IEEE Transactions on Very Large Scale Integration Systems (2012), 1668-1680.

[8] T. GANEGEDARA, V. K. PRASANNA, StrideBV: Single Chip 400G+ Packet Classification, IEEE 13th International Conference on High Performance Switching and Routing (2012), 1-6.