FPGA Implementation of IP Packet Header Parsing Hardware

Danijela Efnusheva, Aristotel Tentov, Ana Cholakoska and Marija Kalendar
Computer Science and Engineering Department, Faculty of Electrical Engineering and Information Technologies, Ss. Cyril and Methodius University, Skopje, Macedonia
{danijela, toto, acholak, marijaka}@feit.ukim.edu.mk

Keywords: FPGA, Header Parser, IP Packet Processing, Multi-gigabit Networks, Network Processor.

Abstract: The rapid expansion of Internet has caused enormous increase in number of users, servers, connections and demands for new applications, services, and protocols in the modern multi-gigabit computer networks. The technology advances have resulted with significant increase of network connection links capacities, especially with the support for fiber-optic communications, while on the other hand the networking router's hardware and software have experienced many difficulties to timely satisfy the novel imposed requirements for high throughput, bandwidth and speed, and low delays. Considering that most network processors spend a significant part of processor cycles to provide IP packet header field access by means of general-purpose processing, in this paper we propose a specialized IP header parsing hardware that is intended to provide much faster IP packet processing, by allowing direct access to non byte- or word-aligned fields found in IPv4/IPv6 packet headers. The proposed IP packet header parser is designed as a specialized hardware logic that is added to the memory where the IP packet headers are placed; and is described in VHDL and then implemented in Virtex7 VC709 Field Programmable Gate Array (FPGA) board. The simulation timing diagrams and FPGA synthesis (implementation) reports are discussed and analyzed in this paper.

1 INTRODUCTION

Internet as the most popular and most widely used network is constantly growing with an extremely large pace, (Ahmadi, 2006). This is due to the ever increasing number of users, servers, connections and new applications. In parallel, the speed of the networking links grows constantly, especially with the great expansion of the fiber-optic technology. As a result of the increased network traffic, the networking hardware remains as the bottleneck for constructing high speed networks. Network processors (NPs) have become the most popular solution to this problem, (Wheeler, 2013). In general they are defined as chip-programmable devices, which are specially tailored to perform several network processing operations, including: header parsing, bit-field manipulation, pattern matching, table look-ups, and data movement, (Lekkas, 2013).

NPs are usually implemented as application specific instruction processors (ASIPs) that mainly include many processing engines (PE), dedicated hardware accelerators, network interfaces, adjusted memory architectures, interconnection mechanisms and provide support for various parallelization techniques, (Shorfuzzaman, Eskicioglu, Graham, 2004). NPs might be used in different types of network equipment such as routers, switches, IDS or firewalls, (Giladi, 2008). Over the last few years many vendors have developed their own NPs, which resuled with many NP architectures existing on the market. Moreover, many novel approaches, such as the NetFPGA architecture, (Naous, Gibb, Bolouki, McKeown, 2008), or software routers, (Petracca, Birkea, Bianco, 2008), are constantly emerging.

The most popular NPs, which are used today, include one or many parallel homo- or heterogeneous processing cores. For instance, Intel's IXP2800 processor, (Intel, 2005), includes 16 identical multi-threaded general-purpose RISC processors organized as a pool of parallel homogenous processing cores that can be easily programmed with great flexibility towards ever-changing services and protocols. Furthermore, EZChip has introduced the first NP with 100 ARM cache-coherent programmable processor cores, (Doud, 2015), that is by far the largest 64-bit ARM processor yet announced.

The discussed NPs confirm that most of the operations in NPs are performed by general-purpose RISC-based processing cores as a cheaper but slower solution, combined with custom-tailored hardware that is more expensive but also more energy-efficient and faster. If network packet processing is analyzed on general-purpose processing cores then it can be easily concluded that a significant part of processor cycles will be spent on packet header parsing, especially when the packet header fields are non byte- or word-aligned. In such case, some bit-wise logical and arithmetical operations are needed in order to extract the value of the appropriate field from the packet header.

Network processing usually begins by copying the packets into a shared memory buffer that is available for further processing by the processor. This buffer may be upgraded with specialized hardware to perform the field extraction operations directly on its output, before forwarding them to the processor. The basic idea of this approach is to replace the bit-wise logical and arithmetic operations by a special parsing logic that will extract the header fields from the on-chip memory and provide them to the processor. The result of using this header parsing logic should be a single-cycle memory access to these non byte- or word- aligned header fields.

The header parsing logic is simple to design, provided that it will be specially adapted to work with IPv4/IPv6 header formats. Actually, the proposed header parsing hardware will be used for reading a single IPv4/IPv6 header field from the memory, or writing to a single IPv4/IPv6 header field into the memory. If this logic is manufactured as an ASIC it cannot be reused for other header formats, so in this paper we investigate the possibilities to utilize a reconfigurable hardware platform like Virtex7 VC709 FPGA, (Xilinx, 2016). In fact, FPGA technology is very suitable for use, providing a compromise between performance, price and re-progrmability, (Cardoso, Hubner, 2011).

The rest of this paper is organized as follows: Section II gives an overview of different networking hardware and software solutions intended to speed up network processing and also discusses several aproaches used for simplifing packet header parsing. Sections III describes the proposed IP header parsing logic and explains its ability to allow single-cycle memory access to non byte- or word- aligned packet header fields. Section IV presents simulations and synthesis results from the FPGA implementation of the IP header parsing hardware model in VHDL. Section V concludes the paper, outlining the benefits of the proposed IP header parsing module.

2 STATE OF THE ART

Each network device that exists in the computer networks examines fields in the packet headers in order to decide what to do with each packet. As a result, the process of identifying and extracting fields in a packet header is subject to a vast amount of research, (Gibb, Varghese, Horowitz, McKeown, 2013). With the ever increasing speed of network links, the research is mostly focused on hardware acceleration for achieving suitable processing speeds, (Kořenek, 2013). This is mainly done by combining application-specific coprocessors with general-purpose multiprocessor systems, or reconfigurable FPGA platforms.

The basic function of each network device is to process the ingress data flow accepted by the physical interface, and then forward the packets to an outbound port, after the processing is finished. In order to achieve this, network devices are usually designed as a composition of four functional blocks: physical interface, data plane, control plane and switching interface, (Lekkas, 2013). Generally NPs are used to perform fast packet processing in the data plane. On the other hand, the slow packet processing in the control plane (configuration and management, execution of routing protocols) is mostly handed by general purpose processor.

NP operation begins with the receipt of an input stream of data packets. After that, usually the IP header of the received packets is being processed, by analyzing, parsing and modifying its content, (Giladi, 2008). NPs might include some specialized hardware units to perform classification of packets, lookup and pattern matching, queue management and traffic control. After the completion of all the required operations, the network processing is finished and the packet is sent out through the switching fabric to the appropriate outbound port.

According to (Hauger, Wild, Mutter, 2009) simpler packet processing and higher speeds can be achieved if the most time–consuming network processing operations are simplified, and some appropriate choices of the routing protocol functionalities are made. As a result, many different approaches have been proposed, including label concept and several other algorithms for faster table lookup given by (Gupta, Lin, McKeown, 1998) and (Eatherton, Varghese, Dittia, 2004).

In general, NP software is getting closer to the NP hardware, such as in (Kekely, Puš, Kořenek, 2014) where part of the packet processing tasks such as classification or security are offloaded to application-specific coprocessors that are used and controlled by the software. In this way, the coprocessor hardware handles the heavy part of the packet processing, at the same time leaving more specific network traffic analyses to the general-purpose processor. As follows, a flexible network processing system with high throughput is built. Some researchers also try to unify the view on the various network hardware systems, as well as their offloading coprocessors, by developing a common abstraction layer for network software development, (Bolla, Bruschi, Lombardo, Podda, 2014).

Other proposals make big use of FPGA technology for packet parsing, as it is very suitable for implementation of pipeline architectures and thus ideal for achieving high-speed network stream processing, (Puš, Kekely, Kořenek, 2014). Actually, the reconfigurable FPGA boards can be used to design flexible multiprocessing systems that adjust themselves to the current packet traffic protocols and characteristics. This approach is given by (Attig, Brebner, 2011), who propose use of PP as a simple high-level language for describing packet parsing algorithms in an implementation-independent manner. Similarly, in (Brebner, Jiang, 2014), a special descriptive language PX is used to describe the kind of network processing that is needed in a system, and then a special tool generates the whole multiprocessor system as an RTL description.

3 DESIGN OF IP PACKET HEADER PARSING UNIT

The general idea of this paper is to propose an IP packet header parsing hardware module that will allow single cycle access (read or write) to various IP header fields. As a result, the proposed IP header parsing unit would speed up packet processing, allowing same access time for a packet header field as the access to any random memory word, even when it is not byte- or word- aligned. This approach would have huge impact on network processing hardware and would provide increased overall network throughput in computer networks at all.

In order to achieve single-cycle access, the proposed IP packet header parsing unit will use part of the memory address space to directly address various IP packet header fields. This technique is known as memory aliasing, and allows each IP header field to be accessed with a separate memory address value. When such address is input in the IP header parsing module it selects the corresponding word from memory, and afterwards depending on the field, the word is processed in order to extract it. This may include shifting the word and/or modification of its bits. A scheme of the proposed logic, used to read out a single IP header filed, is presented in Fig.1.

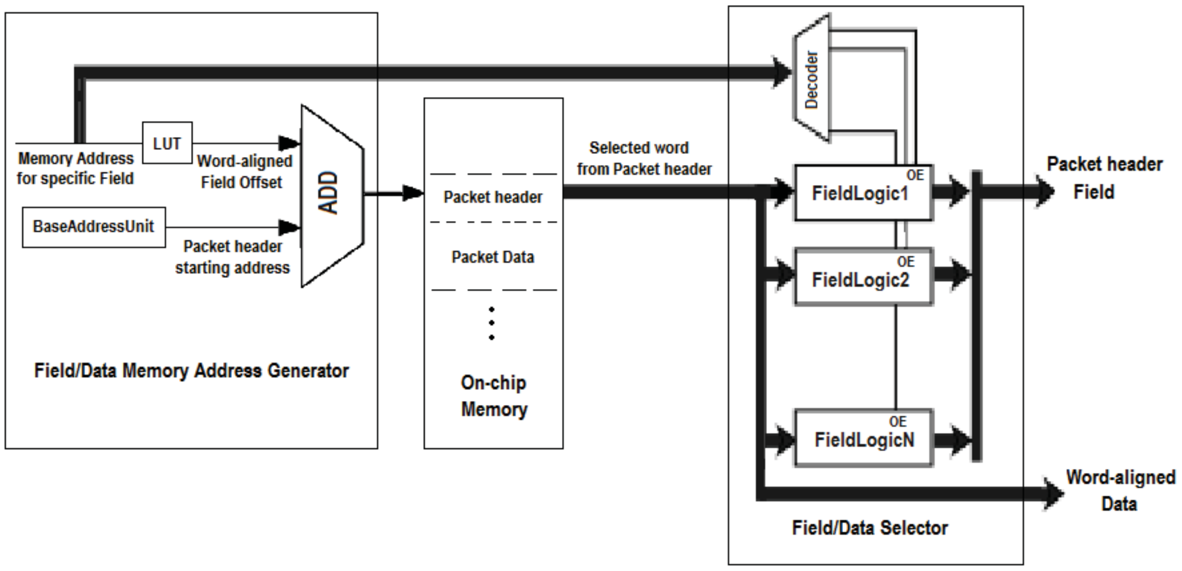
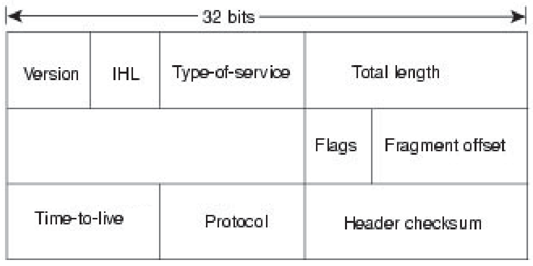


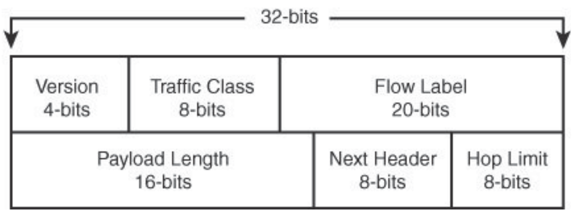
Figure 1: Reading a single IPv4/IPv6 header field with the IP header parsing hardware unit.

IPv4 Header



IPv4 Header @0000100000001000
version 4
headerLength 4
typeofService 8
firstwordfirstHalf 16 //used for IP checksum
totalLength 16
identifier 16
flags 3
fragmentOffset 13
secondwordsecondHalf 16 // used for IP checksum
timetoLive 8
protocol 8
thirdwordfirstHalf 16 //used for IP checksum
headerChecksum 16

IPv6 Header



IPv6 Header @0000110000000000
version 4
trafficClass 8
flowLabel 20
payloadLength 16
nextHeader 8
hopLimit 8

Figure 2: Description of IPv4 and IPv6 headers.

The IP header parsing logic is designed so that it assumes that a packet with IPv4 or IPv6 header format is located in a fixed area of the memory. The description of the format of IPv4 or IPv6 packet headers is shown in Fig. 2. In the given IP header descriptions the first line defines the name of the IP header and its location in memory, while each following line contains the definition of a single field. For each IP header field, the name and its size in bits are specified. The IP header fields are defined in the order that they appear in the IP header.

Table 1: Look up table in IP header parsing logic.

MemoryAddress for IP header field	**Word-aligned IP header Field Offset**
0000h (IPv4 version)	0000h (first word)
0001h (IPv4 headerLength)	0000h (first word)
0002h (IPv4 typeofService)	0000h (first word)
0003h (IPv4 firstwordfirstHalf)	0000h (first word)
0004h (IPv4 totalLength)	0000h (first word)
0005h (IPv4 identifier)	0001h (second word)
0006h (IPv4 flags)	0001h (second word)
0007h (IPv4 fragmentOffset)	0001h (second word)
0008h (IPv4 secondwordsecondHalf)	0001h (second word)
0009h (IPv4 timetoLive)	0002h (third word)
000Ah (IPv4 protocol)	0002h (third word)
000Bh (IPv4 thirdwordfirstHalf)	0002h (third word)
000Ch (IPv4 headerChecksum)	0002h (third word)
000Dh (IPv6 version)	0000h (first word)
000Eh (IPv6 trafficClass)	0000h (first word)
000Fh (IPv6 flowLabel)	0000h (first word)
0010h (IPv6 payloadLength)	0001h (second word)
0011h (IPv6 nextHeader)	0001h (second word)
0012h (IPv6 hopLimit)	0001h (second word)

The IP packet header starting address, which is specified in the IP header description, is placed in a specific base address unit that is part of the IP header parsing logic. Besides that, the input memory address for the specific IP header field is translated into a field offset by the lookup table (LUT), as given in Table 1. The field offset represents a word-aligned offset to the starting IP header packet address, which points to the location where the given IP packet header field is placed. This means that if the length of a specific field is smaller than the memory word length, then the closest word-aligned offset is selected and put in the LUT table.

The address of the memory word that holds the required IP packet header field is calculated by adding the field offset to the IP packet header starting address. Once the word is selected, it is read

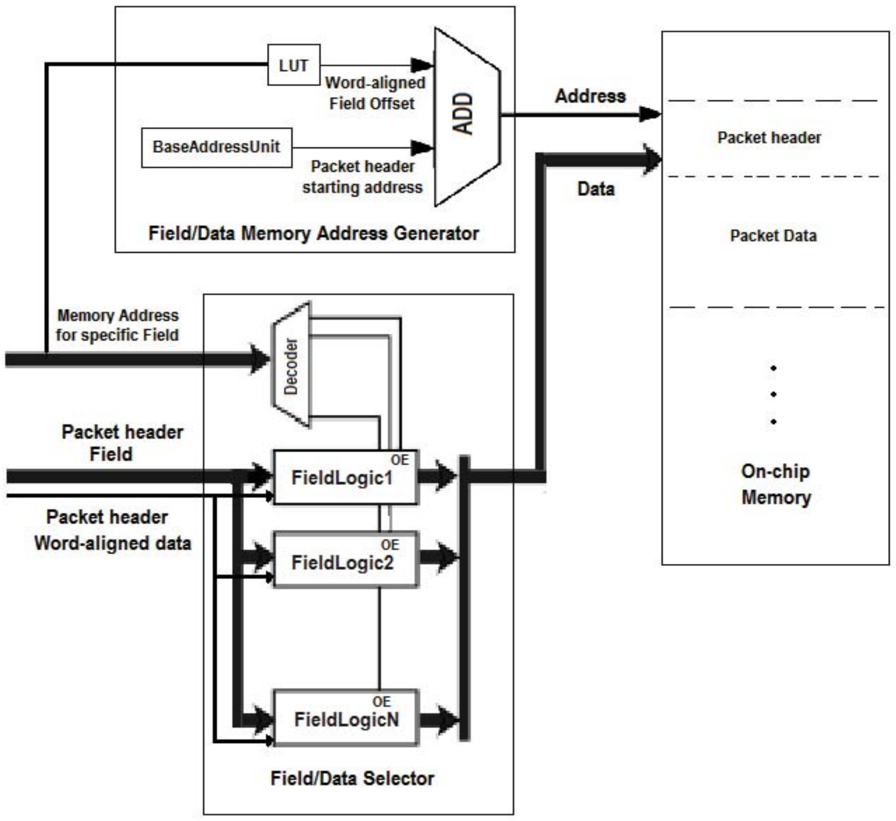

LUT
Word-aligned Field Offset
ADD
BaseAddressUnit
Packet header starting address
Field/Data Memory Address Generator
Address
Data
Packet header
Packet Data
On-chip Memory
Memory Address for specific Field
Decoder
Packet header Field
Packet header Word-aligned data
OE
FieldLogic1
FieldLogic2
FieldLogicN
Field/Data Selector

Figure 3: Writing to a single IPv4/IPv6 header field with the IP header parsing hardware unit.

from the memory and then forwarded to the field processing units. Each field processing is separated into a field logic (FL) block that is activated by the output enable (OE) signal connected to a decoder output. The decoder is also driven by part of the input memory address, causing only one of the FL units to be selected at a given moment. Each FL block is responsible to perform some bit-wise and/or shifting operations in order to extract and then zero-extend the appropriate IP header field. In the case when an IP header field is word-aligned, then its FL block is empty and the word is directly forwarded from memory to the module output.

The presented IP header parsing module form Fig. 1 shows the hardware that is needed to read out a single IP header field from memory. The same concept is used for writing directly to the IP header field in memory, as can be seen in Fig. 3. The both modules select the address of the memory word that holds the required IP packet header field in the same way. The only difference between them is that the packet header word-aligned data read from memory and the IP packet header field that should be written to the memory are applied to each field logic block, when writing is performed. In this way, the decoder that is driven by part of the input memory address activates only one of the FL units and then the selected FL block sets the input IP packet header field to the appropriate position in the input packet header word-aligned data. After that the whole word, including the appropriate IP header field is written to the generated address into the memory.

The given approach of direct access to IP header fields obviously brings much faster packet processing in comparison with the bare general-purpose processing, used by nearly all network processors. For example, a comparison between RISC-based general-purpose MIPS processor, (Patterson, Hennessy, 2014) with and without IP header parsing logic has shown that the number of instructions needed to load all fields from IPv4/IPv6 header is decreased by 40%/45% when IP header parsing unit is used.

4 FPGA IMPLEMENTATION OF IP PACKET HEADER PARSING UNIT

The proposed IP header parsing logic was described in VHDL, by means of Xilinx VIVADO Design Suite tool. This software environment includes a simulator for performing functional analysis of VHDL models, and several other tools for hardware synthesis and FPGA implementation. The FPGA technology is utterly suitable for research purposes, due to its advantage in terms of speed, cost, flexibility and ease of re-programmability, (Cardoso, Hubner, 2011). Therefore, for the FPGA implementation of the proposed IP header parsing logic, we make use of Virtex7 VC709 evaluation platform, (Xilinx, 2016).

The VHDL model of the proposed IP header parsing logic used for reading IP header fields is a module that includes three sub blocks: Field/Data address memory generator, on-chip memory and Field/Data Selector. This top module receives a memory address for specific IP header field and an IP packet header starting address as an input, and produces an IP packet header field or a word-aligned data as an output. This unit is optimized only to extract fields from IPv4 and IPv6 headers, but it can be easily extended and reconfigured to work with other packet header formats. This extension would introduce some modifications into the look up table and would require definition of novel field logic blocks in the IP header parsing logic.

The schematic of the IP header parsing logic used for direct access to IP header fields that has been generated in Xilinx VIVADO Design Suite is shown in Fig. 4. In addition to that, Fig. 5 presents the schematic of IP header parsing logic that is used for writing to IP header fields. This schematic has been generated in Xilinx VIVADO Design Suite and as shown in Fig. 5 is composed of RAM memory and a ShiftBackComputeDataAndAddress module that consists of memory address generator and data field selector, which are used to generate the write address and the data that should be written into the RAM memory.

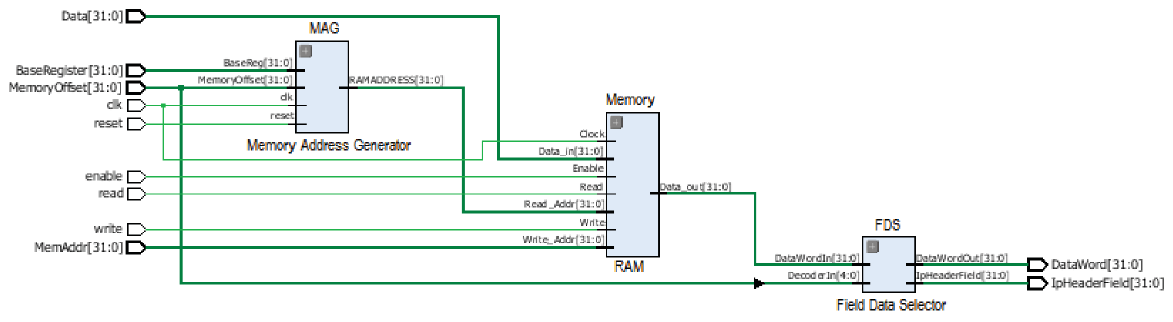


Figure 4: Schematic of IP header parsing logic used for direct access to IP header fields. The module is described in VHDL and then generated in Xilinx VIVADO Design Suite.

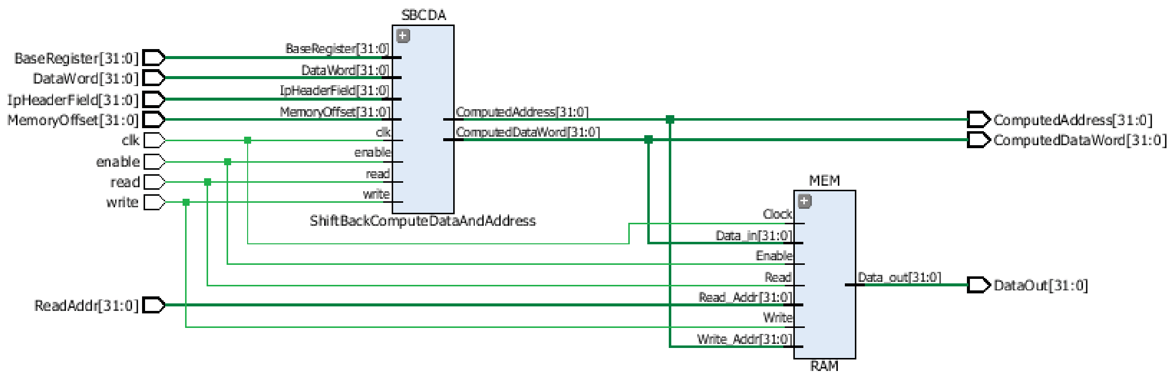
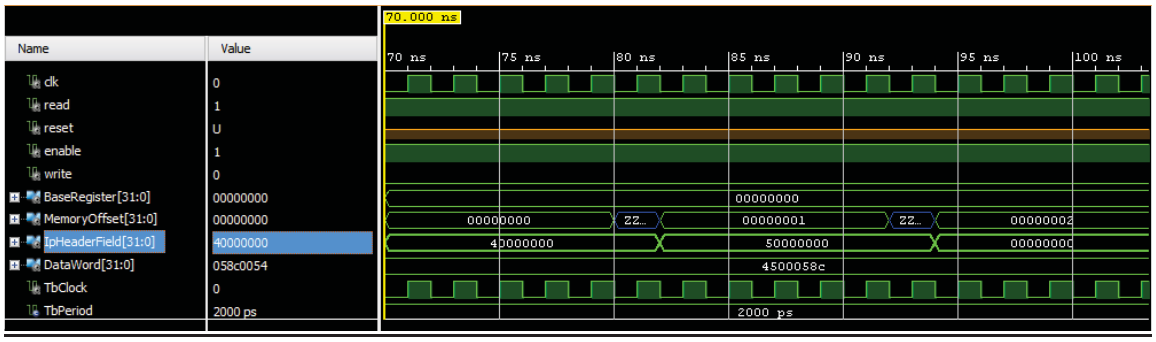
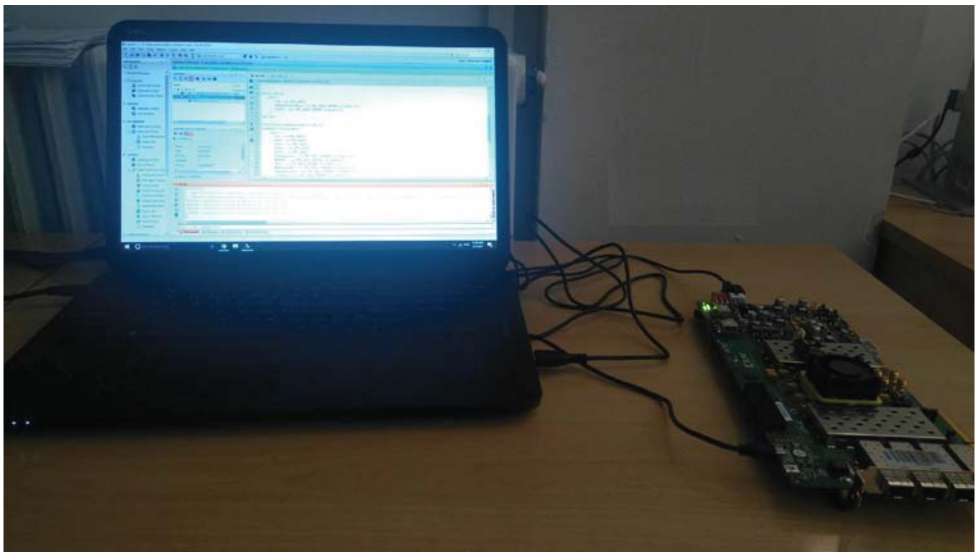


Figure 5: Schematic of IP header parsing logic used for writing to IP header fields. The module is described in VHDL and then generated in Xilinx VIVADO Design Suite.



Name	Value
clk	0
read	1
reset	U
enable	1
write	0
BaseRegister[31:0]	00000000
MemoryOffset[31:0]	00000000
IpHeaderField[31:0]	40000000
DataWord[31:0]	058c0054
TbClock	0
TbPeriod	2000 ps

Figure 6: Simulation of direct access to IP header fields in VIVADO simulator.



a)

b)

Figure 7(a and b): Implementation of IP header parsing logic in Virtex 7 VC709 FPGA board.

Fig. 6 presents simulation results of the IP header parsing top module (which includes modules for read or write to IP header fields), while performing extraction of several fields (version, header length and type of service) from an IPv4 packet header. For the given simulation scenario, it is considered that the memory is already filled with several IP packets, whose IP headers are later parsed and inspected. The waveform signal given in Fig. 6 verifies that the proposed IP header parsing logic works properly.

Once the functional simulation is finished, FPGA synthesis and implementation of the proposed IP header parsing module are performed. The synthesis results show that the IP header parsing logic can be implemented in Virtex7 VC709 evaluation platform, by utilizing 0.01% of the slice registers and 0.35% of the slice LUT resources, which is less than 1% of the occupied FPGA slice resources. As a result of the low FPGA resource's utilization, the initial IP header parsing logic design can be further extended (for other packet header formats) and then implemented in the same Virtex 7 VC 709 FPGA board. According to that, the use of FPGA technology makes the proposed IP header parsing hardware very flexible and also cheap for implementation.

Fig. 7 presents the FPGA implementation of the proposed IP header parsing logic. For that purpose we have created a constraint file which makes use of the input Switch Pins and output LEDs of the Virtex 7 VC 709 FPGA board. Therefore, we have used the Switch Pins to set the specific memory address of the IP header field that should be parsed. Once the IP header field has been selected, the output LEDs light were showing which FL block was activated, during the appropriate IP header field extraction. In this way we were able to test the proposed IP header parsing module in real hardware (FPGA prototype).

5 CONCLUSIONS

This paper proposes an IP header parsing hardware module that allows single- cycle memory access to non byte- or word- aligned fields in IPv4 and IPv6 packet header formats. This approach accelerates the packet processing in both general-purpose and application-specific processor architectures, as IP header field access is a very frequent operation in network processing. Actually, it was shown that a MIPS processor that is extended with IP header parsing logic achieves 40/45% faster header parsing of IPv4/IPv6 packets, in comparison with a bare MIPS processor.

The main focus of this paper is the FPGA implementation of the proposed IP header parsing logic. Considering that the implemented IP header parsing logic utilizes less than 1% of the occupied FPGA slice resources, future work would include comparison of hardware complexities for various header formats and justification of the additional hardware over the performance improvement. It is obvious that these modifications would require extensions of the look up table and definition of novel field logic blocks in the existing IP header parsing logic. Having this possibility to generate parsing modules for specific packet headers, and reconfigure the system to start using them, whenever there is a need for a new networking protocol, is very attractive. This approach makes use of FPGA re-configurability, which has proven to be an ideal solution for achieving reasonable speed at low price.

REFERENCES

Ahmadi, M., Wong, S., 2006. Network processors: challenges and trends. In *17th Annual Workshop on Circuits, Systems and Signal Processing.*

Wheeler, B., 2013. *A new era of network processing*. LinleyGroup Bob Wheeler's White paper.

Lekkas, P. C., 2013. *Network Processors: Architectures, Protocols and Platforms*, McGraw-Hill Professional.

Shorfuzzaman, M., Eskicioglu, R., Graham, P., 2004. *Architectures for network processors: key features, evaluation, and trends,* Communications in Computing, pp.141-146.

Giladi, R., 2008. *Network Processors - Architecture, Programming and Implementation*, Ben-Gurion University of the Negev and EZchip Technologies Ltd.

Naous, J., Gibb, G., Bolouki, S., McKeown, N., 2008. NetFPGA: reusable router architecture for experimental research, in *Sigcomm Presto Workshop*.

Petracca, M., Birkea, R., Bianco, A., 2008. HERO: High speed enhanced routing operation in software routers NICs. in *IEEE Telecommunication Networking Workshop on QoS in Multiservice IP Networks*.

Intel, 2005. *Intel® IXP2800 and IXP2850 network processors*, Product Brief.

Doud, B., 2015. Accelerating the data plane with the Tile-mx manycore processor, in *Linley Data Center Conference*.

Xilinx, 2016. *VC709 Evaluation Board for the Virtex-7 FPGA*. User guide.

Cardoso, J. M. P., Hubner, M., 2011. *Reconfigurable Computing: From FPGAs to Hardware/Software Codesign*, Springer-Verlag.

Gibb, G., Varghese, G., Horowitz, M., McKeown, N., 2013. Design principles for packet parsers. In

ACM/IEEE Symposium on Architectures for Networking and Communications Systems, pp. 13–24.

Kořenek, J., 2013. Hardware acceleration in computer networks. In *16th International Symposium on Design and Diagnostics of Electronic Circuits Systems*.

Hauger, S., Wild, T., Mutter, A., 2009. Packet processing at 100 Gbps and beyond—challenges and perspectives. In *15th International Conference on High Performance Switching and Routing*.

Gupta, P., Lin, S., McKeown, N., 1998. Routing lookups in hardware at memory access speeds. In *IEEE Infocom'98*, pp. 1240–1247.

Eatherton, W., Varghese, G., Dittia, Z., 2004. Tree bitmap: hardware/software IP lookups with incremental updates. In *Sigcomm Computer Communication Review*, vol. 34, no. 2.

Kekely, L., Puš, V., Kořenek, J., 2014. Software Defined Monitoring of application protocols. In *IEEE Conference on Computer Communications*, pp. 1725–1733.

Bolla, R., Bruschi, R., Lombardo, C., Podda, F., 2014. OpenFlow in the Small: A Flexible and Efficient Network Acceleration Framework for Multi-Core System. In *IEEE Transactions on Network and Service Management*, pp. 390-404.

Puš, V., Kekely, L., Kořenek, J., 2014. Design methodology of configurable high performance packet parser for FPGA. In *17th International Symposium on Design and Diagnostics of Electronic Circuits Systems*, pp. 189–194.

Attig, M., Brebner, G., 2011. 400 Gb/s Programmable Packet Parsing on a Single FPGA. In *Seventh ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, pp. 12-23.

Brebner, G., Jiang, W., 2014. High-Speed Packet Processing using Reconfigurable Computing. In *IEEE Micro*, vol. 34, no. 1, pp. 8– 18.

Patterson, D., A., Hennessy, J., L., 2014. *Computer organization and design: the hardware/software interface*, Elsevier. 5th ed.