Design and HDL implementation of original 64 – bit network processor core

Danijela Jakimovska, Student Member, IEEE, Goce Dokoski, Marija Kalendar, Member, IEEE, Aristotel Tentov, Member, IEEE

***Abstract* — As digital technology evolves the demand for new standards and protocols, cost constraints and time-to-market requirements also increase. To meet these challenges, a common solution today is to replace the hardwired application specific integrated circuits (ASIC) with application-specific instruction-set processors (ASIP). This trend is especially apparent in the field of network packet processing, more specifically by the use of ASIP network processors in the telecommunication equipment. Their architecture is usually a subject to various trade-offs such as between performance and flexibility. Accordingly, in this paper we discuss the network processor design, and we propose modified 64-bit general purpose RISC architecture. The proposed network processor is implemented in a language for instruction set architectures (LISA), which allows us to verify its functionality and to evaluate its performance capabilities. Additionally, the LISA model was used for hardware description level (HDL) code generation, which can be additionally simulated on FPGA board.**

***Keywords* — FPGA, network processor architecture, next generation networks, RISC, VHDL.**

I. INTRODUCTION

Computer networks and internet grow rapidly and the number of users and services is constantly increasing as well. Network devices must follow this evolution in order to cope with the increased network traffic and the needs for real-time data processing at very high speeds, up to multi Gb/s. Not to mention that data, voice and video networks are converging and that users are looking for on-demand services delivered in any place, at any time, [1] - [3].

Routers are traditionally designed as application specific integrated circuits, adjusted for the tasks of routing and forwarding information, [1], [4]. However, this approach has showed inflexibility when it comes to adding new capabilities. At the same time, the availability of the System on Chip (SoC) technology, Field programmable gate architecture (FPGA), and the Complex programmable logic device (CPLD), has enabled many new possibilities in processor design. This led to the concept of a network processor (NP), characterized as chip-programmable device, customized for network processing application, [1] - [5]. NPs are usually implemented as ASIP processors, with customized instruction set, which may be based on reduced instruction set computing (RISC), complex instruction set computing (CISC) or very long instruction word (VLIW) architecture etc., [2]. NPs have proven themselves as the most appropriate solution for achieving a good balance among price, flexibility and performance required for network packet processing. In fact, they are flexible and programmable like general purpose processor (GPP) and at the same time they can be very specific and may achieve high performance like ASIC.

Nowadays, there are a lot of different NP architectures, and is expected that the NP market will show strong growth in the near future. Since NP vendors target various applications with different requirements, many approaches have been applied and many new ideas are emerging, such as the NetFPGA architecture [6], or software routers, [7]. In this paper we discuss the current trends in NP design in order to propose novel 64-bit RISC-based network processor architecture. As well we analyze the improvements and performance capabilities of the proposed architecture, and also we evaluate the HDL description of the processor, in order to verify its design. Lastly, we can implement the processor core in a real hardware platform such as FPGA board, considering its limitations.

The rest of this paper is organized as follows: Section II outlines the state of the art in NP design and as well gives some ideas for FPGA use in network processing domain. Section III describes the proposed network processor architecture, analyzes its implementation characteristics, and outlines its performance capabilities. The paper concludes in section IV, where the benefits of the proposed solution are pointed out and the intended future work that should be carried out is emphasized.

Danijela Jakimovska, Msc, is teaching and research assistant at the Faculty of Electrical Engineering and Information Technologies, Ss. Cyril and Methodius University, Skopje, R. Macedonia (e-mail: danijela@feit.ukim.edu.mk).

Goce Dokoski, Msc, is teaching and research assistant at the Faculty of Electrical Engineering and Information Technologies, Ss. Cyril and Methodius University, Skopje, R. Macedonia (e-mail: goce.dokoski@feit.ukim.edu.mk).

Marija Kalendar, Msc, is Ph.D. student, and teaching and research assistant at the Faculty of Electrical Engineering and Information Technologies, Ss. Cyril and Methodius University, Skopje, R. Macedonia (e-mail: marijaka@feit.ukim.edu.mk).

Aristotel Tentov, Phd, is professor at the Faculty of Electrical Engineering and Information Technologies, Ss. Cyril and Methodius University, Skopje, R. Macedonia (e-mail: toto@feit.ukim.edu.mk).

II. State of the Art

NPs development started about 20 years ago [2], and since then, a vide variety of NPs have been proposed, each characterized with different organization and concepts. Nowadays, the most well-known NPs utilize multi-core architectures that can operate in parallel, pipeline or hybrid mode, [2], [3]. The architectures can vary, depending on the NP application category which can be: entry-, mid- or core-level. Usually mid-level NPs that process packets at higher layers, implement parallel architectures, while core-level NPs that require highest processing speeds at the lower network layers, implement pipeline architectures, [2], [8].

According to [2], NPs can be classified in pipelines or parallel pools of processors, depending on the organization of their processing cores or hardware accelerators. The NPs may have one or more cores which can be homo- or hetero-geneous. The Intel IXP2800 processor incorporates 16 identical multi-threaded processor cores, organized as a pool of parallel homogeneous processing elements [9], and an additional 32-bit XScale core responsible for control plane processing. Each processing element represents 32-bit RISC core with the same instruction set. This NP organization is very advantageous in the simplicity of programming the elements, and thus the Intel IXP2800 NP can achieve 10 Gb/s processing speed. On the other hand, the Agere's NPs utilize multi-core organization with heterogeneous processing elements. They are implemented as pipeline, where each various core is responsible for one pipeline stage processing, [2]. Furthermore, the EZChip's NP-1-4 network processors are an example of a pipeline of heterogeneous multi-core processors. Their cores are called Traffic Optimized Processors (TOP cores), since they are optimized for specific processing tasks. In fact, the heterogeneousness complicates the programming, but it allows achieving near-ASIC processing speeds. Therefore, the newest NP-4 processor can reach a total throughput of 100 Gb/s, [10].

Considering all this, we can notice that the NP research area is very popular and is constantly advancing with new solutions and improvements. However, the processor design is not so simple, having in mind the price and the time that has to be spend until the chip verification and real hardware implementation. This is where the FPGA technology comes of great significance, as it is very appropriate for hardware testing and simulation of HDL processor description.

An example of a popular usage of FPGA for network hardware design is the NetFPGA platform. The NetFPGA is constructed as a PCI card that contains an FPGA, four 1GigE ports and buffer memory (SDRAM and DRAM). It consists of modules connected as a sequence of stages in a pipeline, which communicate by using a simple packet-based synchronous first-in first-out (FIFO) push interface, [6]. Although its primary intent is research, the NetFPGA emphasizes the idea of combining FPGA with the NPs.

III. Design and Implementation of Novel Network Processor Architecture

In order to meet the increased speed and performance requirements, we propose novel NP architecture, based on RISC and Harvard organization. Initially, we are going to modify general purpose RISC core architecture, and as well enhance its fundamental instruction set. Afterwards we will try to verify the proposed design and evaluate its performance capabilities. This paper presents the design and implementation of simply one 64-bit RISC-based processor core that can be further used as part of some upcoming multi-processor NP organization.

A. Basic components

The proposed NP architecture is based on a standard RISC 64-bit Harvard processor architecture combined with several hardware accelerators and adjusted for IP packet processing. The Harvard organization is used because it is very convenient for simultaneous read/write operations to the data/program memory. Furthermore, the simplicity of RISC architecture, improves the NP organization, by allowing short simple one cycle instructions to be executed in a 5-stage (fetch, decode, execute, memory access, write back) pipeline. Actually, the main idea behind this NP architecture is the possibility of modifying 64-bit general purpose RISC processor architecture and adapting it for network processing application.

The proposed RISC core architecture includes: internal program and data memory (instruction and data cache), 64–bit arithmetical logic unit, two operand and one result register, 128 general purpose registers and 64 packet header registers (packet header buffer). The processor core, as usual, includes program counter and instruction register, responsible for instruction execution control. The NP design is enriched with packet buffer status register, used for storing some important information (IP version, header length etc.) of the packet being processed, and special functional unit responsible for CRC code validation and calculation. The proposed internal architecture of the NP core is presented on Figure 1.

We consider that this NP core does general packet processing for the both IP versions, [11], [12]. Initially, packets received on the MAC interface, are stored in the data memory, and after the processing is finished, they are sent out to the forwarding engine. Usually, the only part of the packet that is being processed is the IP header. As a result, the basic IP header is first loaded and stored in the additional 64 packet header registers. The network processor has enough register resources for storing the options field of IPv4 header and one extension header of IPv6 header, in the packet buffer registers. The main idea behind this is a novel routing protocol that is supported by this network processor architecture, described in [13]. This routing protocol uses the option fields of IPv4 header or one extension header of IPv6 header for storing the whole IP packet path, similar to the source routing option.

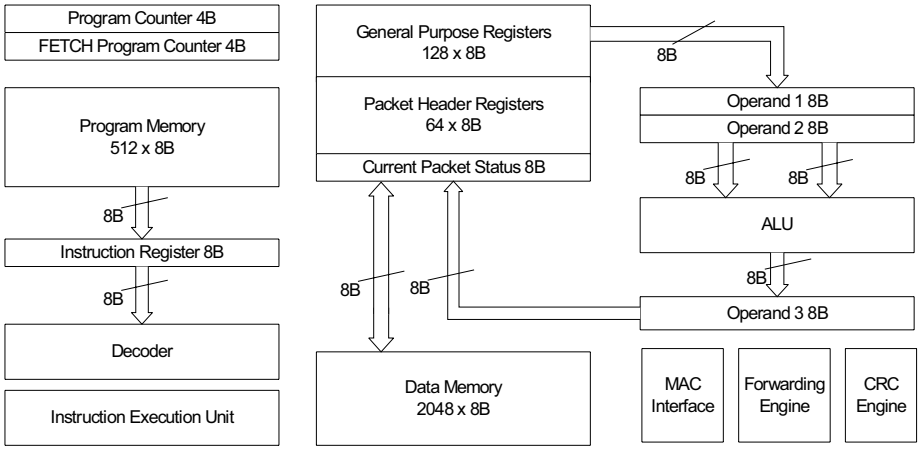

Fig. 1. Proposed network processor architecture

Actually, the NP is composed of 128 general purpose registers, and 64 packet header registers, and all of them can be addressed with completely 8 bits. The codes starting with b00, b01 or b10 are used to denote register indexes, and the remaining 64 codes, starting with b11 are used for addressing the packet header fields (IP version, TTL), placed in the packet header buffer, and named as alias registers. Actually, the last 64 codes are divided, one half for IPv4 fields, and the other for IPv6 fields. Then, for example, the first general purpose register is addressed as b000000000, and the first field of the packet header (IPv4), is addressed as b11000000. All instructions can work with these alias registers as operands specified by the appropriate IP header field name (ex. ip4_ver, ip4_header_length, etc.). This allows for a more flexible (and faster) packet header processing and greater convenience to the programmer. When the compiler is built, this kind of access to the packet header fields will be allowed via system calls.

A. Instruction set

The NP instruction set is based on the simple RISC instruction set, additionally optimized for IP packet processing by employing several instructions for hardware accelerators control, CRC code calculation and validation, and by enabling direct manipulation with the IP header fields. The instructions are 64-bits long and they can be defined in three various formats: register, immediate and control (R, I and C format, accordingly).

The register instructions (sub, add, xor etc.) are three-address instructions, operating with registers. These instructions allow simultaneous shift of the second operand, during the execution of an arithmetical/logical instruction. On the other hand, the immediate instructions (load, store, add, etc.) always include at least one immediate operand and they are used for register-to-memory or memory-to-register transfer, and conditional brunches. However, depending on the operands used, some instructions (ex. comparison) can be implemented as either R- or I-type. The last instruction format, C – type, is used for: unconditional branching, procedure calls, CRC code calculation and validation, and traps.

Finally, the processor should support very simple addressing modes, since it has RISC based architecture, [2]. Thus, it implements some of the simplest addressing modes like register, immediate and index addressing. Finally, all the processor operations are generally related with memory or various register accesses.

B. HDL processor implementation

The proposed NP architecture was modeled using the language for instructions set architectures - LISA. It is a modeling language general enough to model any kind of instruction set driven processors, and yet powerful enough to model specialized instruction set processors, [14]. Therefore we used LISA language to describe the modified RISC architecture, and analyze its functionalities. Accordingly, we defined its memory and bus architecture, a standard 5-stage instruction pipeline and the instruction set specific to the network processing.

The LISA model was used for processor simulation and verification of its functionality. Additionally we wrote some simple assembler programs and analyzed the operation of the NP core. The simulation results confirmed that the processor core is working properly, so its memory and register resources were correctly modified. During the verification process, we could count the number of processor cycles required for the assembler programs execution, and further measure the processor performance.

Later, the LISA model was used for automatic generation of VHDL code description of the NP core. The HDL code allowed us to investigate the processor architecture at lower level. We used ISE design suite tool to synthesize the NP core and evaluate its implementation characteristics. Therefore, the NP core incorporates 13612 flip flops, 16 adders/subtractors, 286 registers, 211 comparators, 208 multiplexers and some additional components. These statistics must be taken in consideration, because the circuit complexity can significantly influence on power consumption, overheating and overall performance capabilities of the designed NP core.

C. Performance evaluation

In order to estimate the performance trade-offs for the proposed NP core architecture, we would calculate the theoretical maximum of instruction cycles allowed for each IP packet processing at the desired speeds of 1/10/100 Gb/s. For that purpose, we made some computations, assuming that the network processor core is working at 250 MHz (frequency that can be achieved on Xilinx Virtex5 FPGA board) and that the average data packet size is 512B. In order to achieve 1/10/100 Gb/s speed, the NP core should finish the processing of one packet within 1024/102/10 processor cycles, accordingly.

Considering the proposed NP architecture, we evaluated its performance by analyzing assembler programs for general IPv4 and IPv6 packet processing. We simulated these programs and estimated the number of processor cycles needed for each IP packet processing. Additionally we considered that the NP core can achieve different results, according to the memory type used (SRAM/DRAM), [15]. In our analyses, we compared the performance of the proposed modified RISC core with general RISC core, and afterwards we measured the possible improvements. The results are shown on figure 2.

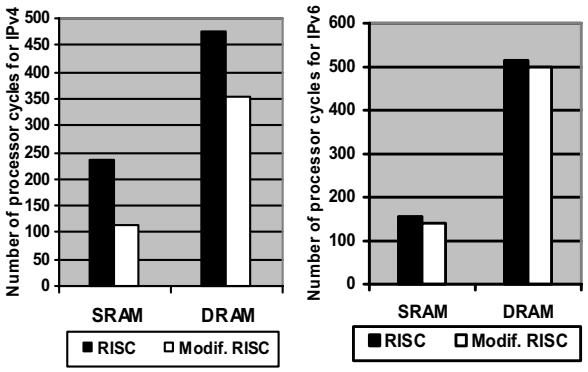

Fig. 2. IP packet processing cycles (IPv4 and IPv6) for RISC and modified RISC network processor core, depending on the memory type (SRAM/DRAM) used

Figure 2 presents that the modified RISC core, using SRAM memory, achieves IP processing for 115, and 140 processor cycles, for IPv4, and IPv6 packets, accordingly. On the other side, if the modified RISC core, utilizes DRAM memory, the IP packet processing cycle's number is 355 and 500, for IPv4, and IPv6, accordingly. In the both cases, the modified RISC NP architecture achieves better results, compared to the general RISC processor core. Actually, if the proposed modified RISC core uses SRAM memory, the packet processing is accelerated by 51% and 10% for Ipv4, and IPv6 packets, accordingly. For the modified RISC processor core using DRAM memory the achieved improvements are: 25% and 3% for Ipv4, and IPv6 packet processing, accordingly. These results show that the proposed NP architecture, implemented on Virtex 5 Xilinx FPGA board can achieve throughput in the theoretical boundaries of 2-9 Gb/s. The initial results are very promising and the authors will continue to investigate new ways for improving the NP core performance.

IV. Conclusion

In this paper, we propose a novel NP architecture that should be able to cope with multi-gigabit networks. Its key architectural aspects are: enhanced instruction set, instruction level parallelism by employing five stage pipelines, one cycle execution of complex instructions, use of packet buffer registers for holding the IP header, and direct manipulation with the IP header fields. The proposed NP core architecture was described in LISA, and afterwards simulated and verified. Additionally we could measure the design characteristics and evaluate its performance capabilities. Obviously, higher throughput can be achieved in real hardware, without the Xilinx VIRTEX 5 FPGA board implementation limitations.

A lot of work still can be done, as for example speeding up the IP routing process and investigating new strategies for memory architectures. Some of these ideas are already described in another paper, [13]. However, all the work that has been done bothers the authors to continue with the research, and achieve better and more realistic results.

References

[1] H. Jonathan Chao, Bin Liu, "High Performance Switches and Routers High speed switches and routers", Wiley-IEEE Press, May 2007

[2] Ran Giladi, "Network Processors - Architecture, Programming and Implementation", Morgan Kaufmann Publisher, Ben-Gurion University of the Negev and EZchip Technologies Ltd., 2008

[3] Mahmood Ahmadi, Stephan Wong, "Network Processors: Challenges and Trends", Proceedings of the 17th Annual Workshop on Circuits, Systems and Signal Processing, ProRisc, pp. 222-232, Veldhoven, The Netherlands, November 2006

[4] Panos C. Lekkas, "Network Processors: Architectures, Protocols and Platforms", McGraw-Hill Professional, 2003

[5] Mohammad Shorfuzzaman, Rasit Eskicioglu, Peter Graham, "Architectures for Network Processors: Key Features, Evaluation, and Trends", Proc. on Communications in Computing, pp.141-146, 2004

[6] Jad Naous, Sara Bolouki, Glen Gibb, Nick McKeown, "NetFPGA: Reusable Router Architecture for Experimental Research", Proceedings of PRESTO, Stanford University, California, USA, 2008

[7] Michele Petraccaa, Robert Birkea, Andrea Bianco, "HERO: High-speed enhanced routing operation in software routers NICs" , Proceedings of the 4th international telecommunication networking workshop on QoS in multiservice IP networks, Politec. di Torino, 2008

[8] Simon Hauger, Thomas Wild, Arthur Mutter, Andreas Kirstädter, Kimon Karras, Rainer Ohlendorf, Frank Feller, and Joachim Scharf, "Packet Processing at 100 Gbps and Beyond—Challenges and Perspectives", in Proceedings of the 10. ITG Symposium on Photonic Networks, May 2009

[9] Intel Corporation, Intel IXP2800 Network Processor® Product Brief, For OC-192/10 Gbps network edge and core applications, 2004

[10] NP-4, 100-Gigabit Network Processor for Carrier Ethernet Applications, Product Brief, 2010

[11] Andreas Moestedt, Peter Sjödin, Torsten Köhler, "Header Processing Requirements and Implementation Complexity for IPv4 Routers", White paper, HP Laboratories Bristol, September, 1998

[12] RFC2460, "Internet Protocol, Version 6 (IPv6) Specification"

[13] Marija Kalendar, Aristotel Tentov, Danijela Jakimovska and Goce Dokoski, "Modified IP Routing Process Supported by a Novel Network processor Architecture", In the proceeding of 54th ETRAN conference, Serbia, 2010

[14] CoWare Processor Designer Product Family, LISA Language Reference Manual, Product Version V2009.1.1, CoWare, Inc., 2009

[15] W. Eatherton, G. Varghese, Z. Dittia, "Tree bitmap: Hardware/Software IP Lookups with Incremental Updates", SIGCOMM Comput. Commun. Rev., vol. 34, no. 2, 2004.