

Efficient Field Processing Cores in an Innovative Protocol Processor System-on-Chip

G. Lykakis¹, N. Mouratidis¹, K. Vlachos², N. Nikolaou², S. Perissakis³, G. Sourdis⁴,
G. Konstantoulakis¹, D. Pnevmatikatos⁴, D. Reisis⁵

¹InAccess Networks, 230, Sigrou Av, GR-17672, Athens, Greece

²Bell Laboratories AT EMEA, Larenseweg 50, 1200BD Hilversum, The Netherlands

³Ellemedia Technologies, 223, Sigrou Av, GR-17121, Athens, Greece

⁴Department of Electrical and Computer Engineering, Technical University of Crete, Chania, Greece

⁵Department of Electrical and Computer Engineering, National Technical University of Athens, Greece

Abstract

We present an innovative protocol processor component that combines wire-speed processing for low-level, and best effort processing for higher-level protocols. The component is a System-on-Chip that integrates variable size packet buffering, specialised cores for header and field processing, generic RISC cores and scheduling blocks. We focus on the main innovation, the reprogrammable pipeline module, and discuss its internal architecture, optimised to perform field processing on byte streams, as well as protocol processing on complex data structures. Furthermore, we present how modern and new tools were used in system dimensioning, design, and verification phases. The chip is able to handle up to 512K flows organised in individual queues. It embeds 5 custom cores optimised for field processing, 3 typical RISC cores for packet processing and 11 generic and application specific hardware blocks. It's been prototyped in UMC 0.18uCMOS technology in a 1096-pin BGA package and operates at 200MHz for 2.5Gbps links.

1. Introduction

The evolution of networks, along with the ever-increasing users' demand for networking services, has imposed the development of high-capacity telecom systems. Modern systems and services must combine various features so as to convince for their affordability, applicability and profitability. One of the main breakthroughs is the development of technology and solutions to support efficiently and effectively the entire service offering chain, focusing on the provision of acceptable QoS [8]. In this end-to-end path, a number of active nodes are involved, each being able to process high bandwidth data, ranging from a few megabits (edge network) to several gigabits (core network) [7]. A critical issue in such systems is the processing of complex communication protocols and functions not only in the OSI Physical and Net-

work Layers, but usually going up to Layer 4 or higher, depending on the type and the characteristics of the network node.

With time-to-market becoming the dominant force in the networking world, many companies have turned to RISC technology with optimised architectures for their systems. Another option is a hybrid approach with System-on-Chip (SoC) solutions combining both processor technologies, using one or more RISC cores and special hardware blocks to perform specific tasks. These components called Network Processors have exhibited an enormous advance in the turn of the millennium. For higher layer protocol functions that are not performed at wire speed, more than one high performance processing units are employed. Such functions include routing protocols, statistical compiling and reporting, error processing, connection admission control, as well as network and transport layers protocol processing (e.g. ATM/AAL, TCP/IP, SSCOP). In these applications, the processing units are inadequate to support the protocol processing requirements for the entire set of active sessions. This constitutes a major system resources bottleneck [8], because the complexity of the protocol algorithms requires higher computational power than that offered by today's processor technology. Furthermore, for such complex SoC implementations, enhanced CAD tools are needed along the entire development cycle.

In this paper we present a highly integrated and complex component designed as a SoC, able to perform towards two directions. On one hand, it processes at wire speed the low level IP or ATM based protocol functions for speeds up to 2.5Gbps, providing standard per flow queuing and scheduling. On the other hand, it integrates innovative processing elements with specialised field processors and typical RISC cores in order to process higher layer protocols. This component called the Programmable Protocol Processor (PRO3) offers a unified processing platform for both wire speed and best effort processing while very interesting trade-offs and configurations can be realised depending on the application.

PRO3 achieves to involve efficient processing elements in the data paths of the system for all or part of the 2.5Gbps traffic. Section 2 presents the PRO3 architecture, while Section 3 presents the innovative field and packet processing cores of PRO3. Section 4 discusses typical applications of PRO3 while Section 5 presents a performance evaluation of these processing blocks. Finally Section 6 concludes the paper.

2. PRO3 architecture

The PRO3 architecture follows a different approach than typical network processors in the area of high speed protocol processing. PRO3 combines two levels of functionality: (i) it processes at wire speed lower layer streams (e.g. IP or ATM layers) and (ii) it aims at accelerating execution of higher layer protocols by integrating specially enhanced high-performance RISC cores for bit and byte processing. Thus the PRO3 system offers three end-to-end processing paths: (i) hardware based packet reception, storage and forwarding; (ii) wire speed packet processing involving specialised CPU cores, and (iii) best effort processing involving typical RISC CPUs. CPU demanding and (hard) real-time protocol functions are handled by the programmable hardware, while the remaining functions as well as higher layer protocols are handled by the on-chip or the off-chip RISC CPUs in an integrated way. The concept of the PRO3 architecture is to provide the required processing power through a novel design, incorporating parallelism and pipelining and by integrating generic micro-programmed cores with components optimised for specific protocol processing tasks. Furthermore, efficient scheduling components are integrated so as to facilitate internal packet processing on a fair, balanced manner as well as to control data streams generated by the chip.

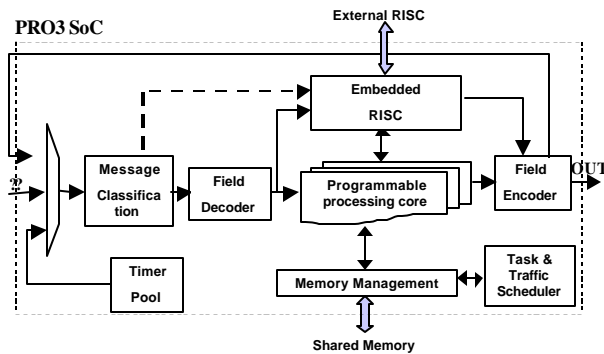


Figure 1. PRO3 SoC functional architecture

In general, the following sequence of operations is applied to each incoming packet: reception, verification, classification, storage, processing, and transmission. Each generic operation consists of a set of lower level functions, while all operations can be understood as pipeline

stages. In case of exception (error) the entire task and its data is redirected to the internal or the external RISC CPU for error handling. The latter is the cornerstone of PRO3 concept. Normally, a protocol consists of a complex FSM that is mostly devoted for error handling and connection set-up and tear down. In error free operation (today network links report very low bit error rate) only a very small part of the FSM is used. PRO3 targets to accelerate this part of functionality for higher layer protocols.

Figure 1 depicts the functional architecture of the protocol processor that is similar to the way a protocol is executed in software. After being classified, each packet (or connection) is assigned a unique flow id. Then a field decoder (header processor) extracts the necessary fields and forwards to the processing unit structured information. Finally a field or packet encoder (header processor) composes the outgoing packet (byte stream).

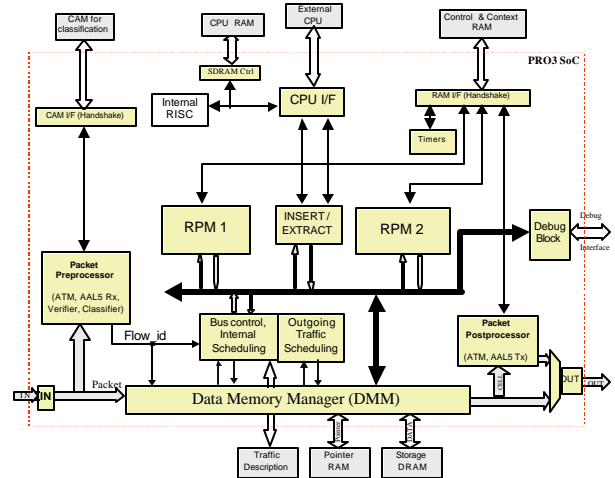


Figure 2. PRO3 SoC physical architecture

The networking applications bottlenecks analysed in [1] and [2], presented that a certain subset of the protocol tasks are highly resource consuming either in terms of computational complexity or memory bandwidth. These critical functions can be accelerated with fixed or programmable hardware, leading to significant system performance enhancement. To optimise performance, the Reprogrammable Pipeline Module (RPM) processes the most frequently and time-consuming protocol FSM segments in error free conditions according to the application. Usually the result of packet processing is a new or modified packet and an update of the protocol FSM context. RPM receives the entire packet or only its headers and processes the respective protocol message at wire speed. This module consists of a bit/field parser (Field Extractor), a RISC CPU with parallel I/O and processing operations, and a packet constructor (Field Modifier). Implementation of timers, memory management, per flow queuing, data and protocol context buffering are also an integral part of this design and potentially a bottleneck in

generic architectures, that can be offloaded to dedicated hardware units.

Since programmability was set as a major requirement, RISC based micro-engines were selected for the implementation of the protocol processing functions. The physical architecture of PRO3 is depicted in Figure 2. PRO3 has been designed as a SoC integrating modules for pre-processing and post-processing, two RPMs, a header processor for programmable classification, and a typical RISC core for control and supervision. We integrated two RPM modules operating in parallel, in order to facilitate load balancing as well as execution of protocols with different incoming and outgoing data flows.

An important component is the Data Memory Manager (DMM) that implements per flow queuing for up to 512K queues. Packets are stored in the external DDR DRAM in queues implemented as linked list data structures [3] supporting both cell and variable packet size queuing. The DMM retrieves (parts of) packets in a FIFO manner per queue, in response to specific commands, and delivers them to the RPMs, to the internal RISC, to the host CPU (via the insert/extract interface) or directly to the output interface. Along all the processing data paths, special feedback signals allow extension of the processing time-slot of each packet thus giving more flexibility to the system. The received packets are stored, and usually only the first 64 or 128 bytes, which contain control information, are forwarded to the RPMs for processing. This efficient way of processing offers clear advantage of the architecture, that differentiates it from other network processor designs.

3. PRO3 processing components

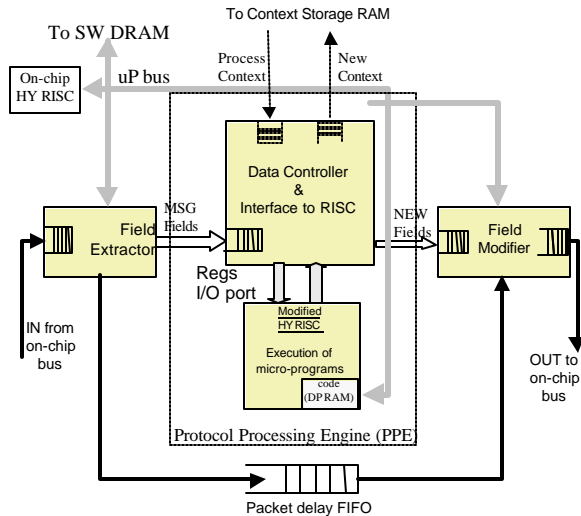


Figure 3. RPM block diagram

The main processing element of PRO3 is the Reprogrammable Pipeline Module (RPM) as depicted in Figure

3. RPM is an innovative module optimised to perform packet (involving both byte and structured) processing. Each RPM consists of a modified Hyperstone RISC core [4] surrounded by a Field Extraction (FEX) engine, which directly loads the required protocol data to the RISC for processing, and a Field Modification engine (FMO) for packet construction and header modification. All form a powerful 3-stage pipeline module capable of providing the mixed hardware and software processing heart of the system. The RPM RISC was enhanced so as to optimise the data I/O operation, by performing in parallel (pipeline) packet processing of two consecutive time-slots. A feedback signal from the RISC to the entire RPM can extend the processing cycle of a certain packet. This design offers an extreme advantage on tasks with high functional diversity. In this way, the use and efficiency of a typical RISC processor core is enhanced yielding a clear cost/performance advantage.

An important feature of the RPM architecture is the Packet Delay FIFO. The received data (entire or part of packet) are temporarily stored in the Delay FIFO waiting for packet processing results. The Field Modifier receives the delayed data along with the results from the RISC. Depending on these results and the application, it performs the required modifications on the delayed packet and sends it back to the DMM.

3.1. The Field Extraction engine

The Field Extraction engine (FEX) is a small RISC with a three-stage pipeline architecture. It is fully programmable and operates on a protocol or application specific firmware. Only specific fields are extracted from the received (part of the) packet. Payload does not need to be entirely sent from DMM to RPM for processing. FEX acts as a typical software structure to alleviate the RISC from packet verification, bit and byte processing, and leave only generic software execution for it. This results in a constant ratio of cycle budget to packet length and optimal total processing time.

Field extraction operation can start either from the packet header or the trailer(s). The component is able to process data with a maximum throughput of 6.4Gbps at 200MHz. The average throughput may be less, since some instructions need more cycles or since one 32-bit word may contain several fields extracted separately. The block diagram of the module is depicted in Figure 4. FEX implements a three-stage generic pipeline for enhanced performance. The I/O operations are also pipelined using memory shadowing.

FEX has been designed as a custom RISC for bit and byte processing. It executes 9 basic instructions and 4 optional commands. Each instruction can be combined and executed in parallel with any or all of the commands. The instructions are used to parse the data, while the commands are used to control the internal registers. FEX

uses 4 generic registers, a Program Counter and a Data Pointer as depicted in Figure 4. The instructions are: NOP; EXTRACT (n, b) that extracts a field of $n + 1$ bits with rightmost being bit b ; MOV A to DP; MOV B to DP; ADD A to DP; ADD B to DP; JMP C, a, addr; JMP D, a, addr (if C or D is equal to a then jump to addr); STR (restart); and FLGS type, flags (sent flags to PPE). The commands are: DEC DP; INC DP; DEC A; and DEC B. FEX executes firmware up to 2K-instruction that is sufficient enough for such applications. A specific compiler tool has been implemented to aid software development.

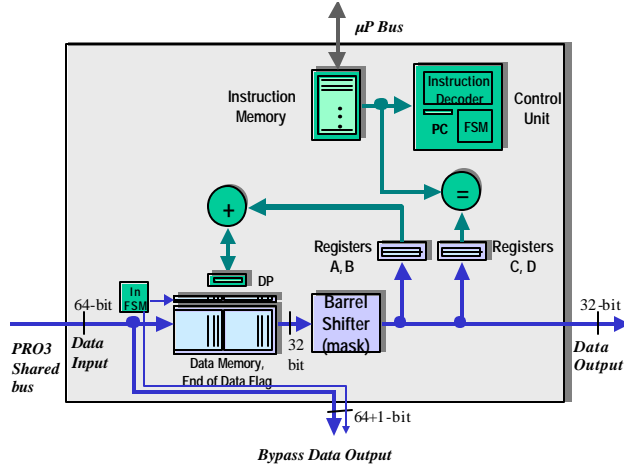


Figure 4. FEX block diagram

A similar FEX engine is also used in the pre-processing block of the header processing, for the construction of a classification key (e.g. 128 bit long for IPv4) up to 144 bits, that triggers the CAM-search. This increases PRO3 applicability to other networking protocols including MPLS, IPv6 and gigabit Ethernet.

3.2. The Protocol Processing Engine (PPE)

PPE is the generic processing element in the RPM and the entire PRO3 system. PPE consists of logic for control and data transfers as well as an enhanced RISC CPU optimised to pipeline I/O operations with instruction execution. This modified RISC is a derivative of the standard Hyperstone E1-32XS microprocessor core [4]. The modified Hyperstone RISC uses 32 global and 64 local registers of 32 bits each, 16 global and 16 local registers directly addressable. Two sets of 14 global registers and 64 local registers are accessible from the PPE controller via a special port. Core accesses are switchable between the two sets of 14 global registers and between the two parts of a 32+32 register partitioning of the 64 local registers. In this way, the PPE control logic can put state and packet information into the register file, and through the added read port, updated FSM context and packet information is read out. The RISC integrates 16 KByte dual-ported on-chip memory with the second port accessible outside the

core, used for initialisation. Pipelined memory accesses allow overlapping with instruction execution.

Within PPE, the Data Controller interfaces and transfers data between the Field Extractor, the Field Modifier, the Modified Hyperstone RISC and the Read/Write Control RAM (where the context is stored). The controller transfers the packet fields from the Field Extractor to the local portion of the RISC registers, and initiates packet processing when the RISC has completed processing of the previous packet. Furthermore, it reads and updates flow state information (context) from/to the Read/Write Control RAM and forward it to the RISC core. A feedback signal from the RISC to the PPE can extend the processing cycle of a certain packet. This may stall the operation of RPM in order to be able to accommodate extended packet-processing requirements. It is worth noting that the RISC core does not have direct access to the external memory. Instead, the application firmware is resident on its on-chip cache memory.

3.3. The Field Modification engine

The Field Modification (FMO) engine is similar to FEX but with dual operation. It also adopts a three-stage pipeline architecture with totally programmable operation. Its target is to compose the protocol message (byte stream) according to the protocol specifications (firmware controlled), taking as input the results of processing from PPE (fields) and the data from the Delay FIFO. FMO performs one of the following: (i) reject the packet(s) in case of errors, (ii) modify the original packet with the fields received from PPE, and (iii) compose a new packet. The new or modified packet is sent to the destination through the on-chip bus. This results in a constant ratio of cycle budget to packet length and optimal total processing time.

The component is able to process data with a maximum throughput of 6.4Gbps. This is accomplished by using a 32-bit wide data path and operating clock frequency of 200MHz (System Clock). The average throughput may be less, since some instructions need more clock cycles or since one 32-bit word may be composed of several fields received separately. The block diagram of the module is depicted in Figure 5. FMO operation is controlled by microcode stored in an internal SRAM (up to 2K instructions). The internal or external CPU can download the firmware dynamically at run-time, through the microprocessor interface. As in the entire PRO3, for FMO the time is distinguished into Processing Slots. Each Processing Slot can be variable in time and is able to accommodate and process a (part of) packet up to 7 64-byte segments.

FMO has been designed as a custom RISC for field processing. It executes 16 basic instructions and 6 optional commands. Each instruction can be combined with one or more of the commands. The instruction and the commands are executed in parallel. The instructions are

fields, whereas the payload is temporarily stored directly in Delay FIFO, small improvement is anticipated. It is worth noting that the total number of instructions (and processing time) for two segments (128-byte) is smaller than that for one segment (64-byte). This is due to the fact that the firmware easily identifies the case of two segments and based on the IP header lengths (resides in first segment) *jumps* directly to the fields to be extracted, which reside in the second segment.

The cycle budget of FEX is close to 4 Mpackets/sec (Mpps), assuming average packets. This rate can be doubled when the traffic is balanced between the two RPM modules and in this way 8Mpps traffic can be sustained. This throughput exceeds the OC-48 rate assuming worst case TCP/IP traffic of 40-byte packet length.

On the contrary, FMO optimisations were of absolute importance. The total processing time in FMO depends on the packet length, thus the total processing time depends both on the header length and on the packet length. To this end, optimisation was possible yielding significant improve in its performance. For example the average cycle-to-instruction ratio was 2.2 and after optimisation was decreased down to 1.7. That was attainable after detecting, which firmware routines are most often called and which are the most cycle-consuming. Thus a significant decrease in the number of executed instructions was achieved, almost 60%, resulting in shorter processing times and in an improved instruction-to-clock ratio.

Concerning the PPE block and mainly the Modified RISC, its throughput depends heavily on the custom running application and it is estimated, that for complex applications, like TCP state updating, less than 200 instructions are needed and this of course has an impact in the overall throughput. However for complex scenarios this is a trade off that any network processor faces. Based on our analysis, by using two RPM modules and balancing the load between these two (supported by the Internal Scheduler design) 4Mpps can be sustained at worst case, with only TCP traffic. For the average IP packet this rate exceeds the OC-48 rate of 2,5Gbps.

6. Conclusions

In this paper the Programmable Protocol Processor architecture was presented with emphasis in the acceleration of packet processing using an innovative concept of a 3-stage pipeline processing operation and certain scheduling and buffering in order to balance workload and resolve internal contention. The design is suitable both for cell and packet based network-processing applications. PRO3 chip can be understood in core or high-speed network systems including switches, interworking units, broadband terminals and servers, broadband gateways and in general in systems that require the processing of protocols higher than just the physical and network layers. Typical applications of PRO3 based systems include

stateful inspection, firewalling, network address translation, massive processing of protocol instances, control nodes of large switches and interworking functions.

The component architecture yields efficient VLSI implementation, with low memory requirements and flexibility to support multiple service disciplines in a programmable way, supporting thousands of flows concurrently. The chip has been fabricated in UMC 0.18 μ m CMOS process occupying about 52mm² of area and packaged in a 1096 BGA package. Samples are under testing.



Figure 6. PRO3 die

References

- [1] G. Konstantoulakis, et al. "A Novel Architecture for Efficient Protocol Processing in High Speed Communication Environments", in *proc. of ECUMN'2000*, Colmar, France, October 2000.
- [2] N. Nikolaou, et al "Application Decomposition for High-Speed Network Processing Platforms", in *proc. of 2nd ECUMN'2002* April 2000, Colmar France.
- [3] A. Nikologiannis, M. Katevenis, "Efficient Per-Flow Queueing in DRAM at OC-192 Line Rate using Out-of-Order Execution Techniques", in *proc. of ICC2001*, Helsinki, Finland, June 2001
- [4] Hyperstone AG, E1-32X <http://www.hyperstone.com>.
- [5] D. Verkest, et al, "Matisse: a system-on-chip design methodology emphasizing dynamic memory management", VLSI '98. Proc. of IEEE Computer Society Workshop, 1998 Pages: 110–115.
- [6] EDN Embedded Microprocessor Benchmark Consortium "Network Processing Platform Benchmarking Methodology Framework", Draft 1.0 RFC, July 2000.
- [7] "Technologies and Building Blocks for Fast Packet Forwarding", Werner Bux et al IEEE Comm. Mag., Jan 2001
- [8] "A network processor architecture for flexible QoS control in very high-speed line interfaces", H. Shimonishi, T. Murase, IEEE workshop on High Performance Switching and Routing, 2001 p 402-406.