An Innovative SoC Design for Broadband Residential Applications

D. ECONOMOU, N. MOURATIDIS, G. LYKAKIS, A. TAVOULARIS, A. KOSTOPOULOS, A. MANOUSARIDIS, G. KONSTANTOULAKIS

inAccess Networks, 230, Sigrou Av, GR-17672, Athens, GREECE
email: gkonst@inaccessnetworks.com, http:\\www.inaccessnetworks.com

Abstract: - We present the Convergence Processor, an innovative component that integrates a high performance 32-bit RISC core, a custom IP core optimised for header-processing and other blocks for specific communication interfaces required for the delivery of broadband residential applications. The component is a System-on-Chip supporting the real time processing of packets and protocol data units from various networking interfaces. Its target is to be used as the central processing unit for various multifunctional networking systems including RGs, IADs, STBs, and IP PBXs. We focus on the system architecture and the component reusable IP cores, namely the header processor, the security engine, the data-handling unit and the open source 32-bit RISC LEON CPU. As communication interfaces, the chip integrates 802.3 MAC, ATM, AAL5, HDLC, UART and a generic packet interface for voice and video processing peripherals. It has been prototyped using a large XILINX Virtex FPGA and UMC 0.18u CMOS technology targeting 200MHz operation. The chip supports a total link capacity of 900Mbps and is optimised to deliver an aggregate sustain rate of up to 100Mbps of multimedia traffic at application level with acceptable Quality of Service.

Keywords: - Broadband Processor, Packet Processor, Header Processor, Residential Gateway, Linux, Convergence, System-on-Chip.

1. Introduction

Modern communication networks have already evolved beyond best-effort traffic patterns, towards real-time, delay sensitive traffic mixes, which are used to deliver a variety of services. Devices that were used to simply forward IP traffic are steadily being replaced by more sophisticated devices, called multiservice gateways, that have multiple different and incompatible interfaces. These interfaces cannot be simply bridged at layer 1 or layer 2 but need a higher layer interworking function thus increasing system complexity. Moreover, they have to handle multiple traffic sources in a service-aware manner, so as to guarantee a variety of parameters like bitrate, delay, jitter and burstiness for a mix of traffic patterns ranging from bandwidth-intensive file-transfers to delay-sensitive voice- or video-streams. Depending on applications, all these parameters compose the Quality of Service offered to the user.

Gradually however, specialized access processors have evolved, that integrate these blocks in a system-on-a-chip factor, mainly for reducing overall bill-of-material (BoM) cost, but also for resolving system-level performance bottlenecks caused by poor interconnection logic. Such components range in performance, from low-end processors that simply integrate interfaces for cost-conscious and limited-performance devices such as modems, routers and home gateways for DSL or Cable connectivity of medium bitrate; to high-end processors for access multiplexers, with the ability to multiplex a number of data streams and pass them to upper-level routers or switches for redirection to the core network.

Convergence Processor (CP) is a broadband access processor that has been designed and dimensioned for broadband customer-premises' applications for the small-business and residential sector; it differentiates from existing devices in performance and targets high-end services that require real-time packet video and packet voice traffic. CP is a system-on-chip which combines the interfaces and processing blocks that are required for the realization of gateway systems that combine data, voice and video traffic, while preserving QoS in a service-aware manner.

CP has been designed combining performance and cost optimisations. Compared with other leading implementations of broadband access processors like Philips/Ishoni PTD2210 [1] and Brecis MSP3000 [2], CP extensively employs proprietary programmable hardware logic for packet processing, flow classification, data encryption/decryption, MAC and ATM/AAL. These, along with a 32-channel DMA controller and service-specific prioritisation, allow the on-chip SPARC-compliant CPU to concentrate on system control and execute high-level management tasks, resulting in

a highly scalable and leaner architecture, not limited by the embedded processor speed.

Chapter 2 describes the functional and physical architecture of the system. Chapter 3 focuses on the main and reusable IP cores of CP, namely the CPU, the header processor, the Security engine and the DMA that are of crucial importance for advanced system performance. Chapter 4 presents an analysis and computation of CP performance using reference application scenarios. Finally, Chapter 5 concludes the paper.

2. Architecture

The Convergence Processor (CP) follows hybrid architecture in order to be able to support with certain QoS level a various number of interfaces that are expected to exist in the future broadband CPE environment. The component supports an aggregate of 900Mbps traffic; however we consider that a rather small percentage of this link capacity (about an aggregation of 100Mbps) is used to deliver user services. CP is understood in gateway systems where it has to efficiently support a number of interworking paths among the provided interfaces. From that point of view, the CP component has to be able to accommodate instant peaks up to the aggregate link capacity using on-chip buffers and to support the sustain rate using the system memory for buffering. This bandwidth is not limited by the CP data handling capability but by the software that is running in the system. CP based residential gateway systems support traffic routing at various layers of the OSI stack. Although CP performs lower layer processing in hardware, certain functions are performed in software either at the driver or at a higher software level. CP combines the following operations (i) it processes at wire speed lower layer protocols (e.g. 802.3 MAC, ATM, AAL2, AAL5, HDLC) and (ii) it accelerates the execution of higher layer functions (e.g. IPSec with DES/TDES engine and Firewalling/Routing with the Classification engine). CP target is to provide the required processing power through a novel design, incorporating parallelism and pipelining and by integrating a generic micro-programmed core optimised for header processing. Furthermore, an efficient data handling and packet scheduling component is integrated so as to facilitate all internal data transfers.

The CP SoC supports (i) two 10/100bT Ethernet interfaces and implements IEEE 802.3 MAC processing in hardware, (ii) a UTOPIA Level II/III interface for up to 155Mbps link rate and implements ATM, AAL0, AAL2 and AAL5 protocol processing in hardware for 32 flows, (iii) a configurable streaming interface for voice, video or other peripherals, and (iv) a 2 Mbps interface with HDLC implementation in hardware. Apart from these networking interfaces, CP supports two UART ports, a debug interface that offers a probe in the on-chip bus and a typical micro-processor interface for access to the system memory. Moreover, the CP SoC integrates an efficient Header Processor for programmable field processing and on-chip classification for 256 flows as well as a security engine that implements the DES and TDES algorithms. In addition, an enhanced DMA component performs all data transfers between the external system memory and the chip hardware blocks. Finally, a high performance 32-bit RISC CPU is integrated to run the system OS and the software.

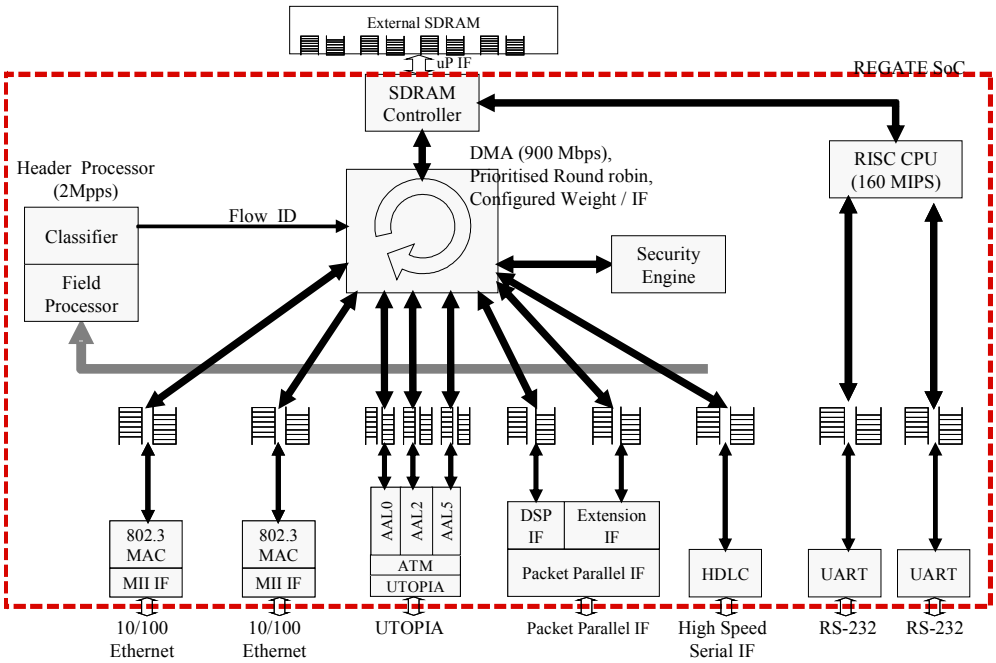

Figure 1: CP SoC functional architecture

Figure 1 depicts the functional architecture of the Convergence Processor SoC. The component supports the following sequence of operations to each incoming packet: reception, low layer protocol processing per port, classification, storage to the main memory and further processing by software. For the outgoing packets, the

sequence of operations involves data transfer to the respective interface block, lower layer processing, and transmission. The transmit direction implements packet level traffic shaping, while on the receive direction certain thresholds and conditions are monitored for each port FIFO. Both traffic-control functions are integral part of the DMA component. In addition to the blocks involved in the transmit and receive data paths, the chip implements a security engine that acts as accelerator and performs DES and TDES. All hardware blocks lead to a significant system performance enhancement, as we will present later. An important feature of the chip is that it integrates the open source LEON 32-b RISC CPU [3]. LEON is a royalty free CPU that has been developed according to the SPARC V8 open architecture.

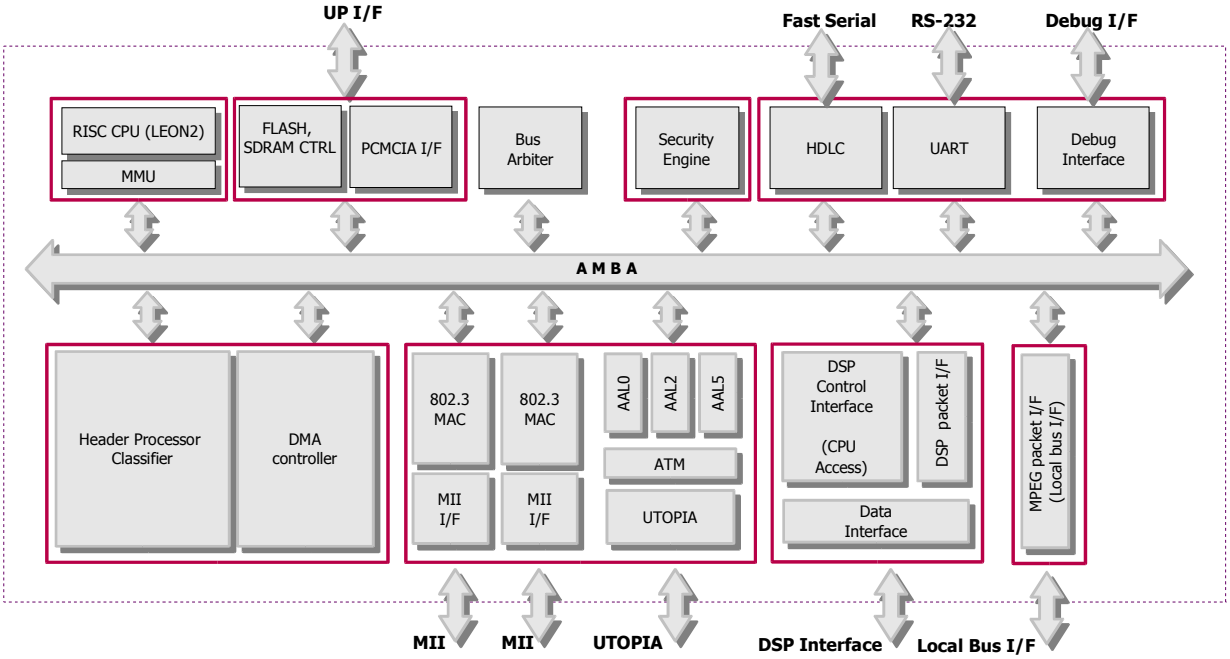

Figure 2: CP SoC block diagram

The component uses the main memory for buffering and the LEON CPU for processing of the data streams. Thus there is no pure end-to-end hardware path at wire speed but for high priority streams all forwarding processing is done at low level software on a packet or a time window basis. The later depends on the application and the traffic profile of the streams. Figure 2 depicts the block diagram of the Convergence Processor SoC. The CP blocks are interconnected with a high speed AMBA (AHB) bus with 3.2 Gbps (32bit @ 100MHz) bandwidth. The entire chip has been designed to run at 100MHz apart from the CPU block that runs at 200MHz. The embedded CPU offers 180 MIPS for software processing.

As depicted in Figure 2, the bus arbiter assigns the on-chip bus either to the CPU or to the DMA. Let assume that the aggregate application bandwidth is 100Mbps and that 20Mbps of this has to be decrypted (sent to the security engine). In this case, the DMA has to transfer 100Mbps total data (e.g. receive 50Mbps and transmit 50Mbps) plus to transfer 20Mbps to the security engine and then back to the memory. Thus the DMA has to handle 140Mbps in total that corresponds to about 5% of the on-chip and of the system bus capacity.

3. Main Processing Blocks

The CP component integrates generic as well as custom IP cores for packet and protocol data unit (PDU) processing. These cores are the LEON CPU, the Header Processor, the Security Engine and the DMA. In the following these blocks are explained in more details.

3.1. LEON CPU

LEON is a 32-bit processor core that conforms to the IEEE-1754 (SPARC V8) specification [3] and includes various peripheral modules, interconnected over a flexible AMBA AHB/APB architecture. It features an Integer unit including all multiply and divide instructions, while the number of register windows is configurable within the limit of the SPARC standard (2-32), with a default setting of 8. LEON supports separate, multi-set instruction and data caches, each configurable with 1-4 sets, 1-64 kbyte/set, 16-32 bytes per line. Sub-blocking is implemented with one valid bit per 32-bit word. The instruction cache uses streaming during line-refill to minimise refill latency. The data cache uses write-through policy and implements a double-word write-buffer. The data cache can also perform bus-snooping on the AHB bus. LEON implements a flexible memory interface that provides a direct interface to PROM, memory mapped I/O devices, SRAM and SDRAM. The memory areas can be programmed to either 8-, 16- or 32-bit data width. LEON

implements an interrupt controller that can manage a total of 15 interrupts, originating from internal and external sources.

3.2. Header Processor

The header processor block consists of the Field Extraction engine (FEX) and the Classifier. In this section we mainly focus on FEX that is considered an important IP block of the chip. FEX is a small RISC that adopts three-stage pipeline architecture. It is fully programmable and operates with protocol or application specific firmware. Payload does not need to be entirely sent from the respective interface to FEX for processing but only the headers, that correspond to the first part of the packet and for certain protocols to the last part of the packet. FEX performs header verification, field processing, and selective field extraction based on user downloaded firmware.

FEX is able to process data with a maximum throughput of 3.2Gbps @ 100MHz. The average throughput is much less, since certain instructions need more than a clock cycle or since one 32-bit word may contain several fields that are separately extracted or checked. The block diagram of the module is depicted in the left-hand side of Figure 3. FEX implements a three-stage generic pipeline for enhanced performance.

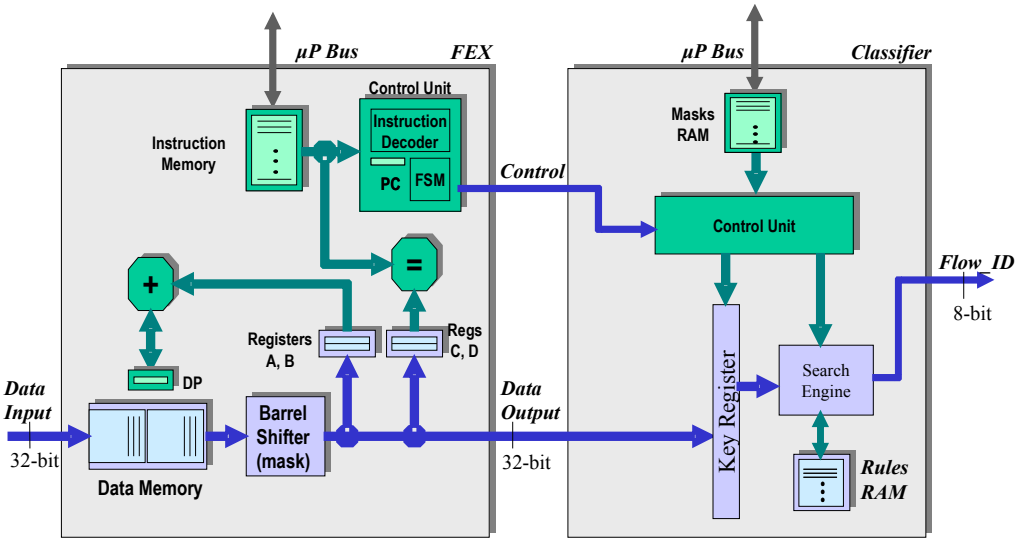

Figure 3: Header Processor block diagram

FEX has been designed as a custom RISC that executes 8 basic instructions and 4 optional commands. Each instruction can be combined and executed in parallel with any or all of the commands thus reducing code size and increasing performance. 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 3. The instructions are: NOP; EXTRACT [n, b] (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); and STR (restart the program). The commands are DEC DP, INC DP, DEC A, and DEC B. FEX executes firmware up to 2K instructions that is sufficient enough for the supported applications. FEX is an innovative IP block that has been reused in PRO3 protocol processor that is a complex Soc for multi gigabit switching systems [4]. In PRO3, FEX architecture is the same and operates at 200MHz.

The Header Processor block includes also the Classifier unit that performs the classification of the specific flow based on FEX results and assigns a Flow_ID to each flow. The Classifier receives the extracted by FEX fields and appends them to the key register so as to construct the classification key. For typical IP traffic the classification key is up to 144 bit wide. The block is able to support up to 256 classification rules, that are distinguished into three sets namely for generic reject (deny), generic accept and per-flow classification. Based on the Classifier result (Flow_ID) the DMA (if Flow_ID is 0 rejects the packet, otherwise it is sent to the CPU) or the CPU decides upon further processing. The Classifier supports 8 masks for all flows, configured by CPU.

3.3. Security Engine

The security engine block acts as accelerator. In this case, the data are not processed on-the-fly while they are transferred to/from the main memory. The block receives the (part of) packet from the system memory (through a dedicated DMA channel) and returns the processed data to the system memory (through another dedicated DMA channel). The block is able to perform either DES or TDES security algorithm at packet level for encryption or decryption of data. The operation to be performed is specified by the software that sends each packet for processing. For specific Flow_IDs pre-configured operations may be performed. Furthermore, the CPU has to define the security key based on which the block will process each flow. The block has a context memory of 16 entries, each being able to hold the security key and the operation to be performed for the respective flow. The security engine has been designed to perform processing of up to 80Mbps data traffic received from the 32-

bit on-chip bus.

3.4. DMA

The DMA is a complex block that supports a number of fast data transfer paths between two buffering points of the system. In our implementation one of the end points is always the system memory (either source or destination of data). The active end point associations can be configured to be up to 32 and depend on the maintenance of the DMA context and traffic profile for each path. Prior to each data transfer operation, the DMA requests and becomes the master of the AMBA bus so as to directly access the system memory. The supported data paths are distinguished in receive (downstream, towards the system memory) and transmit (upstream, from the system memory to a CP block).

The aim of DMA is to perform data transfers without the intervention of the CPU. According to the system aggregate traffic, DMA becomes the master of the on-chip bus for a certain fraction of total bus time. To enhance the traffic multiplexing and optimise the sharing of the on-chip bus for data transfers, the DMA integrates an efficient traffic handling function. It adopts a two level mechanism to define the percentage of time that the block performs the transactions, called traffic descriptor. Apart from the pre-configured traffic descriptors, the DMA can operate on an interrupt basis for a certain interface that is almost overflowed (threshold signal asserted) thus preventing buffer overflow (receiver) or underflow (transmitter). We define as interface timeslot the time assigned to the interface and as DMA timeslot the time that the DMA is master of the bus and serves one or more interfaces.

In order for a traffic profile to be described to the DMA, three types of parameters are used. The first separates overall bus time in two segments, one is used to allocate a cycle budget to the DMA controller, while the other is the guaranteed minimum time interval that the DMA will remain off the system bus. The second parameter defines the segmentation of the DMA cycle budget per data direction, i.e. between transmit and receive. The third parameter allocates DMA time to the served interfaces. This allocation is independent for each direction. Thus, overall transmit and receive times, as well as those related to specific interfaces may be asymmetrical, with no relation to one another. Furthermore, interface timeslots are not necessarily smaller than DMA timeslots. It is, therefore, possible, depending on appropriate configuration, for an interface to be served in one or more DMA timeslots. In this manner it is possible to balance transactions over the bus, and cater as much for low bandwidth or high bandwidth interfaces, that may even carry small or large data packets.

4. Performance Evaluation

To evaluate the architecture and the impact of certain design choices to a CP based networking system we used an FPGA chip prototype. We also used an evaluation board with the FPGA CP prototype, SDRAM, 10/100 Ethernet and ATM. The board integrates also an external StrongARM CPU in order to run experiments with this CPU only and perform comparisons between the two CPUs. For the FPGA prototype we used a large XILINX Virtex FPGA of 800.000 gates.

The LEON CPU core is well optimised for embedded applications. Using 4K+4K caches and a 16x16 multiplier, the dhrystone 2.1 benchmark reports 1,550 iteration/s/MHz using the gcc-2.95.2 compiler (-O2). This translates to 0.9 dhrystone MIPS/MHz using the VAX 11/780 value a reference for one MIPS [3]. Thus in the CP component, LEON offers 180Mbps @ 200MHz. Apart from the required processing power, LEON leads to royalty free SoC implementations, thus reducing the cost that is a major factor for CPE products.

To compute the performance enhancements of integrated CP based networking systems we measured the CPU processing power required for executing these functions in the foreseen broadband access environments. In our analysis we decomposed the processing required in such networking systems using the open source Linux stacks, running on the FPGA evaluation board and we measured the performance parameters for typical voice, video and Internet data traffic. For voice we considered G.711 codec (worst case) that produces 40 bytes of voice data per 5 msec. We considered voice over IP thus for each voice packet, we add 20 bytes the RTP header, 20 bytes the IP header and 14 bytes the Ethernet header. For video we considered a traffic profile that was statistically analysed from real MPEG encoded videos in [5]. For Internet traffic we consider typical statistics that are widely available in the Internet as well as extensive background work performed in the field ([6, 7]).

To evaluate the impact of CP and estimate performance of CP based networking systems, we used the evaluation board to run real application software under Linux. We also used open source or internally developed software blocks, which are used in the target CPE systems. These software modules include the Ethernet driver, the TCP/IP stack, the DES/TDES engine, the VPN software, the routing software and the firewalling software.

In our measurements we analysed the involved software and we computed the CPU MIPS needed per Mbps of average traffic of each profile and for each of the above software modules or functions. Based on these performance parameters we calculated the MIPS that we save with the CP component from a broadband networking system that has to perform the following application scenario:

A. The system passes through two encrypted MPEG-2 video streams (each 5.5Mbps) from the WAN (ATM or Ethernet based) interface to an internal (Ethernet) interface. The streams must be received and forwarded to an IP Set-Top-Box with low jitter (prioritised).
B. 4 voice channels are received from the WAN interface and forwarded to the system DSP through the CP DSP interface with low delay/jitter (e.g. 5msec port-to-port) with minimum processing at driver level.
C. 6 Mbps bi-directional encrypted (Internet) data traffic is

processed where 128 firewall rules are applied.

This scenario may be too complex for typical future residential applications, however it is a demanding example since it involves voice, video and typical Internet data flows. In order to evaluate the performance of the CP based system, we ran typical open source and Linux based software that support such functions including routing, firewalling, and IPSec (DES/TDES). The performance results of this software are considered quite worst case and based on bulk and not optimised software. These figures are compared with measurements taken with optimised software where the packet forwarding function is done in low level using the Header Processor results and bypassing the Linux based software paths. In our measurements, the VPN function was not involved.

In this scenario the total DMA data movements is about 70Mbps since an encrypted stream will pass four times through the bus (receive from the network, sent to and receive from the security engine, transmit to the network). This corresponds to 2.2% of the on-chip bus capacity. Thus only a small percentage of the on-chip bus is assigned to the DMA in a CP based system. On the other hand, in a CPU based system we assume that the CPU performs all functions, including data transfers to the peripherals.

	Linux open source SW		**Linux with Optimized Low level SW**	
Function path	**CP based system**	**CPU based system**	**CP based system**	**CPU based system**
Video forwarding	8,4	647,9	13,2	652,6
Voice processing	23,4	28,4	3,9	11,4
Data processing	85,3	434,2	18,0	367,0
TOTAL MIPS needed	**117,1**	**1.110,5**	**35,0**	**1.031,0**

Table 1: MIPS needed for the existing Linux and for optimised low level software blocks

As depicted in Table 1, CP based system can support the above scenario even with the existing open source software blocks. In addition, when optimised software is used, the CPU offers a small percentage of its power (35 out of 180 MIPS) and can support further user applications without performance or QoS degradation.

Finally, considering worst-case traffic (small IP packets over Ethernet – 64 bytes), the CP based system is able to process and forward 3.1 Mbps of traffic (6,055 pps) using the typical Linux software and 17.4 Mbps of traffic (33,984 pps) using Linux and the optimised low level software for forwarding respectively. Considering large packets (1514 byte long), the above figures are 56Mbps for the open source software and 153 Mbps for the optimised software. These performance figures are by far higher than the Ishoni [1] processor that is able to forward 9.42 Mbps of 64-byte traffic and 98.71 Mbps of 1512-byte traffic [8].

5. Conclusions

In this paper we presented the Convergence Processor system with emphasis on the resuable IP blocks of the design. Furthermore, we analysed the architecture of the design and how innovative and generic for networking systems IP blocks are used to alleviate the host processor from high CPU power consuming tasks.

The component architecture yields efficient VLSI implementation, with low memory requirements and flexibility to support multiple service disciplines in a programmable way. The chip has been prototyped in FPGA and is going to be fabricated in UMC 0.18µm CMOS process occupying about 25mm^2 of area and packaged in a 456 BGA.

Acknowledgement

The described work has been performed within the EU co-funded IST-2000-28429-REGATE R&D project.

References:

[1] www.ishoni.com, see product PTD2210

[2] www.brecis.com, see product MSP3000 (MSP300-Product-Brief.pdf)

[3] LEON-2 users manual, http://www.gaisler.com/doc/leon2-1.0.10.pdf

[4] K. Vlachos et al. *"Processing and Scheduling Components in an Innovative Network Processor Architecture"*, 16th IEEE International conference in VLSI design, New Delhi, India, January 4-8, 2003.

[5] "*Efficient Modelling of VBR MPEG-1 coded Video Sources*", N. Doulamis, A. Doulamis, G. E. Kontantoulakis and G. I. Stassinopoulos, in the proceeding of IEEE Transactions on Circuits and Systems for Video Technology, VOL. 10, N0 1, Feb. 2000.

[6] A.J. McGregor, H-W.Braun and J.A. Brown, "*The NLANR Network Analysis Infrastructure*," IEEE Communications Magazine, Vol. 38 (5): pp. 122-128, May 2000.

[7] "Long-term traffic aspects of the NSFNET", K. Claffy and H.-W. Braun and G. Polyzos, Proceedings of INET'93. http://www.caida.org/Papers/lta.html

[8] Tests Prove Ishoni Platform Sets New Price-Performance http://www.ishoni.com/content/newsstory.asp?article=62