An Embedded Networking SoC for purely Ethernet MANs/WANs

Th. Orphanoudakis[1], G. Kornaros[2], I. Mavroidis[2] , A. Nikologiannis[2], I. Papaefstathiou[2]

1. *University of Peloponnese, Karaiskaki str., 22100, Tripoli, Greece, fanis@uop.gr*

2. *Technical University of Crete, Electronics & Computer Engineering, Kounoupidiana, Chania, Crete, Greece {kornaros,maurog,anikol, ygp}@mhl.tuc.gr*

Abstract

Ethernet technology is lo longer used only in Local Area Networks (LANs); it is continuously gaining momentum in the Metropolitan Area Networks(MANs) and Wide Area Networks(WANs). This paper presents a multi-service access concentrator core that has been designed specifically for multi-service, purely Ethernet, access nodes. In particular the presented system is optimised for Ethernet traffic aggregation over MPLS-based optical backbone networks. Moreover, we also demonstrate the bottlenecks that have been identified when such networking applications are executed in a general-purpose network processing device, and the techniques we used in order to bypass them. Our experiments results, executed in a state-of-the-art FPGA-based platform, strongly support that the combination of general purpose processing units with powerful specialized hardware modules, is the most cost-effective approach for designing systems that (i) can support today's and future network speeds and applications, in purely Ethernet networks, (ii) provide the end-user with the required programmability.

Keywords: *Network processor, FPGA, multi-service access concentrator, switching, Ethernet, Multi-processor.*

1 INTRODUCTION

The growth in the number of network nodes, data traffic and link rates, continuing at a steady pace during the last decade, has imposed the development of high-capacity telecommunication systems. Moreover, in the network access and edge domains of such telecommunication frameworks, even though the bandwidth is very high, the equipment is shared by a small number of users and therefore low investment and operational costs are critical factors. Furthermore, a very important current trend in the networking world is to migrate functions from their current position, in the middle of a network, to the edge of it and thus closer to the subscriber.

The device which can provide the requested processing power while being close to the network edge is the access concentrator. Such a system is located at a MAN or LAN and provides access to a high-speed backbone WAN. Since Ethernet technology is gaining momentum in the MAN/WAN area, while transparent LAN bridging is becoming a very successful service [1], a very important networking application is the Ethernet traffic aggregation over an optical WAN backbone; in such a framework, the Multi-Protocol Label Switching (MPLS) scheme is most commonly used.

In this paper we present a programmable Network Processing Unit (NPU), prototyped in a high-end FPGA platform, which is tailored to the newly introduced, purely Ethernet, high-end network-access environment. This specialised device is much less expensive, in terms of hardware, than the existing, general-purpose solutions, while it provides the same level of performance. In this work, we also discuss the use of this NPU as the basic component for building high-performance multi-service access concentrators and the performance it achieves when a real-world networking application is executed on it. Furthermore, we present, in detail, the specific characteristics of a certain class of networking applications, highlighting their bottlenecks, when executed by general-purpose CPU cores; this analysis can be used as a guideline for anyone wishing to design network concentrating systems.

2 RELATED WORK

A long list of companies have developed high-end NPUs, based on several proprietary architectures with different programming models. The majority of the commercial, high-end NPUs fall mainly into two categories: the ones that use a large number of simple RISC CPUs, and those that use a small number of high-end, special purpose CPUs optimized for processing network streams. The first category

involves solutions from big semiconductor players, such as Intel's IXP family that has from 9 to 17 very simple microprocessors [3], Motorola's C-5 [7] with 16 simple, general-purpose CPUs and 5 very simple special-purpose co-processors, and CISCO's Toaster family [7] with 16 simple microcontrollers. The aforementioned NPUs claim to handle data rates from 2.5 to 10Gb/sec, but as it is shown in [7] the actual bandwidth they can service depends heavily on the application. Thus, for complex applications the performance degrades dramatically. Moreover, their performance is limited by the intercommunication overhead, and the limited off-chip memory bandwidth. The second category includes Broadcom/Sibyte's Mercurian [9] with four 2-way, superscalar, ultra high-speed CPUs that incorporate also special-purpose processing units, Clearwater's CNP810SP [10], with an 8 thread, 10-issue CPU that can simultaneously process 8 packets and includes special-purpose processing units as well as EZChip's NP1 [7] with a number of processors, specialized for ultra-high speed memory lookups.

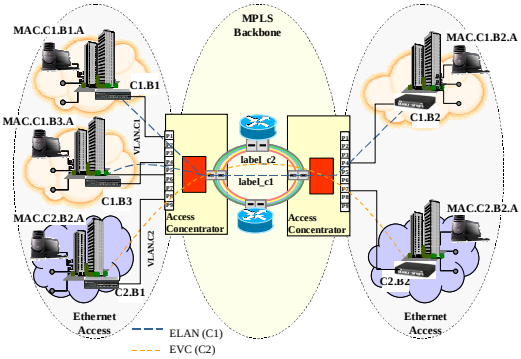

Figure 1 : An Ethernet MAN/WAN

Although, these NPUs, provide higher processing power for some complex network protocols, they lack the parallelism of the first category. Therefore, their performance, in terms of bandwidth serviced, is lower than the one of the first category, for processing large numbers of independent flows.

Moreover, a number of NPUs implemented in reconfigurable technologies have recently been proposed. Those include : (i) a special platform that use hardware plug-ins in order to accelerate the performance of a programmable router and in which the specialised hardware is placed in different chips than the general purpose processing units [11], (ii) a fully reconfigurable processor that is adapted according to the application needs [12], and (iii) a highly complicated 7-layer processing engine [8].

Although all the above devices can execute all the existing networking applications, the presented system is the only one, to the best of our knowledge, that is fine-tuned especially for the networking concentrator, purely-Ethernet, environment. The device has actually been implemented in a state-of-the-art FPGA which provides both the general purpose processing power by two built-in high performance CPUs and a number of soft-core simple CPUs and the significant power of the specialized programmable hardware blocks. As the performance section clearly demonstrates, the presented system can meet the performance of the state-of-the-art devices, while being about an order of magnitude less expensive than them, in terms of hardware-cost.

3 REQUIREMENTS OF MULTI-SERVICE ACCESS NODES

The reference access concentrator, in which the presented system will be placed, is located at a LAN/MAN edge and provides access to a high-speed backbone WAN, using the Ethernet Technology; in such an environment the service providers can offer both Virtual Private Networking (VPN) services and Internet access from a single packet switched infrastructure. One of the most interesting VPN services is a multipoint Ethernet VPN, commonly referred to as virtual private LAN service (VPLS). VPLS is a VPN class that allows the connection of multiple sites in a single bridged domain over an IP/MPLS network (Figure 1). All customer sites in a VPLS instance appear to be on the same LAN, regardless of their location. VPLS uses an Ethernet interface as the customer handoff, simplifying the LAN/WAN boundary and allowing for rapid and flexible service provisioning.

The VPLS architecture specifies the use of a Provider Edge (PE) router that is capable of learning, bridging and switching on a per-VPLS basis. The PE routers are connected together by a full mesh of MPLS Label Switched Path (LSP) tunnels. Multiple VPLS services can be offered over the same set of LSP tunnels. Signaling is used to negotiate a set of ingress and egress Virtual Connection (VC) labels on a per-service basis. The VC labels are used by the PE routers for de-multiplexing traffic arriving from different VPLS services over the same set of LSP tunnels. PE routers learn the source MAC addresses of the traffic arriving on their network ports. Each PE router maintains a Forwarding Information Base (FIB) for each VPLS service instance and learned MAC addresses are placed in the corresponding FIB table. All traffic is switched based on MAC addresses and forwarded between all participating PE routers using the LSP tunnels. Unknown packets (i.e., the destination MAC address has not been learned) are forwarded to all corresponding PE routers until the target station responds and the MAC address is learned by the PE routers associated with that service.

PE routers must comply with highly demanding requirements in order to support the VPLS service at a network rate of several Gigabits per second.

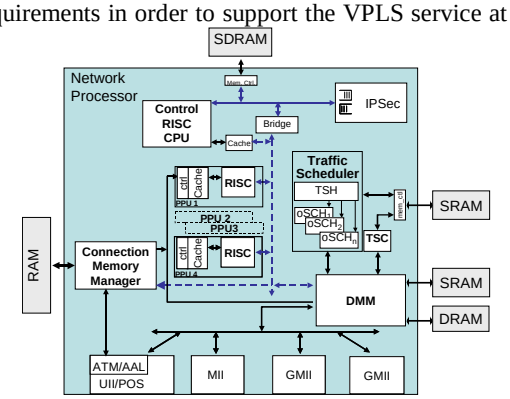

Figure 2 GigaFlow block diagram

The required functionality of a PE includes:

- Port Aggregation
- MAC frame forwarding, multicasting/broadcasting, address learning/aging
- Traffic Management (policing, scheduling, buffer management)
- Protocol encapsulation and Ethernet/Metro bridging
- Control Protocol and Management Support

3.1 Analysis of the Application

A very important issue when designing, our NPU called GigaFlow, was the identification of the performance bottlenecks when the specified class of applications is executed on the general purpose NPUs of Section 2. As this section clearly demonstrates all the dedicated hardware modules of the GigaFlow, as well as its overall architecture, have been designed so as to bypass the bottlenecks faced by the existing devices. In general, Transparent Ethernet bridging over an MPLS-based backbone includes intensive protocol processing as well as high-speed packet switching. More specifically, the complete packet processing flow includes the following functions:

- Packet header extraction and protocol identification
- Source MAC learning (based on input port, source MAC address and VLAN-VID)
- Destination port lookup (based on destination MAC address and VLAN)
- Aging of forwarding table
- Flooding of packets towards all configured ports of a customer's Ethernet VLAN (broadcasting)
- Protocol encapsulation
- Traffic metering, shaping and scheduling.

A first observation is the need for efficient search functions that are performed either on short fields or, in the worst-case, combination of long fields (e.g. MAC and VID). Those searches result is significant bottlenecks when they are executed in General-Purpose processors. Therefore, in the GigaFlow a special scheme has been utilized based on the Connection Memory partitioning which is shown in Figure 3. Long searches, including MAC-address based ones, are performed by a specialized hardware module which returns the memory base-address where configuration/context information resides for each VLAN of the system. The additional requirement for learning leads to double memory accesses whereas aging of those entries is a task performed off-line by control plane software executed in the main CPU. After packet classification and forwarding table updates have been performed, the main processing intensive task is that of header modifications and packet encapsulation. Header construction is performed by simple CPUs whereas the very costly data copying task is executed by a specialized Data Memory Manager(DMM) module; the DMM performs such operations extremely faster than any of the general purpose CPUs utilized in the devices described in the Related Work Section. An additional requirement in Ethernet WAN access is also that of packet flooding within each VLANs broadcast domain. This is a HW assisted procedure only for physically copying data to output ports. In case different header encapsulations/addresses are required for each port (e.g. different labels) this can only be accomplished by SW "loops". Another very expensive, in terms of CPU power, task is the Fair Bandwidth Allocation one. In the Gigaflow this task is also handled by the dedicated hardware units and does not consume any processing resources. Moreover, a complicated Traffic policing is also supported by specialized hardware as the next section demonstrates, whereas packet dropping, in case of excessive load, can be implemented in the management plane.

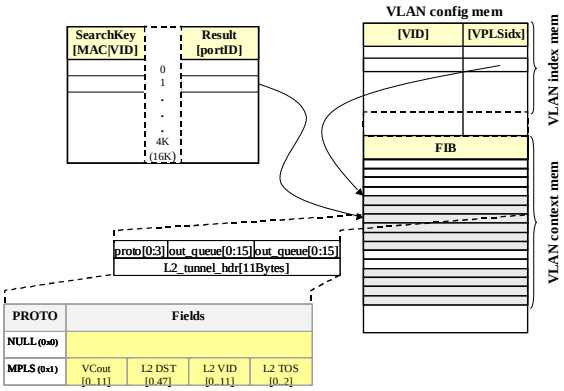

Figure 3 : Connection Memory partitioning

4 NPU ARCHITECTURE

The GigaFlow, demonstrated in Figure 2, implements a hierarchical architecture, allowing several levels of programmable data processing. It can be connected directly to a number of different networks through its high-speed parallel interfaces and perform bridging, protocol processing and packet forwarding between them. Packet Processing Units (PPUs), which are based on simple embedded, low-cost RISC processors, process the network data in the data (fast) path, while the complicated general-purpose CPUs are responsible for the complex control-plane tasks. Moreover, the network packets or cells can be switched between the different network ports without the intervention of any processing unit as long as the hardware queuing and scheduling engines are appropriate configured. GigaFlow features a number of innovations not found in other NPUs in its category. For example, network cells or packets are directly handled by a specialized memory management hardware module supporting a powerful set of commands performing various manipulations on its data structures.

Efficient internal communication is achieved through an optimized mix of high-speed communication buses and shared memory schemes under the control of a specialized task scheduler allowing for prioritised and weighted internal task scheduling and load balancing. Finally a security acceleration engine is available so as to further reduce the required software processing power in security applications.

4.1 *Network Interfaces*

GigaFlow supports 3 types of network interfaces: Fast Ethernet MAC (MII), Gigabit Ethernet MAC (GMII) and AAL5/UTOPIA Level 2 network interfaces; all of them are directly connected to the DMM.

4.2 *Data Memory Management (DMM) engine*

The DMM module implements per-flow queuing for up to 32K independent flows enabling efficient queue handling, variable-length packet storage and access to specific packet segments by the processing engines. The main functions of the DMM are (i) to store the incoming traffic in certain individual queues, (ii) to retrieve parts of the stored packets and forward them to the CPUs for protocol processing. The DMM can also modify the stored packets in certain ways and can forward the stored traffic to the output. The DMM allows stored packets to receive different levels of processing without keeping multiple copies of them in data memory. This significantly minimizes the external-memory throughput requirements and simplifies the software processing in the PPUs. The DMM operates both at fixed and variable length data items. It uses DRAM for data storage and SRAM for the internal data structures. Thus, all the manipulations of those data structures occur in parallel with data transfers, keeping DRAM accesses and overall latency to a minimum. The DMM provides a large instruction set in order to support the diverse protocol processing requirements of any Ethernet device handling queues. Beyond the primitive commands of “enqueue” and “dequeue”, the DMM features a large set of 12 commands to perform various manipulations on its data structures; all those commands are very frequently utilized when the GigaFlow executes the class of applications described in the last section.

4.3 *Connection Memory Management (CMM) Module*

This module controls the accesses to an external SRAM memory, in which the different connection parameters and a number of dynamically-managed lookup tables are stored. A classification engine is employed at Ethernet’s MAC Layer, which uses an innovative hashing scheme and internal replacement of MAC Vendor IDs; the Hash Based Classification Engine (HBCE) compacts the MAC address tables and supports extremely high-speed decisions, while its memory needs are significantly lower than the existing solutions[7]. HBCE is designed to support tens-of-thousands of MAC-address rules and a couple of thousands of VLAN-based and port-based ones. The support of MAC-based classification is another factor that differentiates the GigaFlow from all the devices presented in Section 2, and it significantly increases its performance when it is employed in a purely Ethernet environment.

4.4 *Traffic Management Support*

The Traffic Shaper (TSH) module regulates the outgoing traffic in order to comply to certain bandwidth limitations and it is implementing a Leaky Bucket algorithm[10]. An Output Scheduler (OSC) block forwards outgoing packets to the corresponding network interface depending on flow priorities and assigned flow rates. Its purpose is to regulate the network traffic so as to share the link bandwidth according to the pre-defined weights of the different network flows. The packet selection policy used is based on a work-conserving algorithm which is very similar to Worst-case-Fair Queueing (WF2Q) [6] and which uses a priority queue mechanism based on virtual times.

4.5 *Internal Scheduling and Communication.*

The Task Scheduler (TSC) is responsible for scheduling the processing requests coming from all the active input flows, taking into account their priorities and the availability of the PPUs. After the DMM receives a complete packet and enqueues it to the corresponding input flow, it informs the TSC for the packet arrival. This signals the start of packet processing. As soon as a pending request for processing is eligible for service, the TSC asks the DMM to retrieve and forward the corresponding packet header to a specified PPU.

The DMM communicates with all the processing units through both the IBM Coreconnect OPB bus [7] and the built-in dual port RAMs (DP-RAMs) of the FPGA. The reason for using the latter approach is that the OPB bus does not support burst transfers, therefore its effective data transfer is limited; however it has the advantage of extremely simple interconnection interfaces. In order to efficiently transfer the data between the DMM and the processing units several built-in DP-RAMs are used (i.e. one per processing unit). The DMM is connected to one port of these DP-RAMs, while each processing unit is connected to the second port. The processing units (and the TSC) use the OPB bus to send commands to the DMM, while they use the DP-RAMs for data transfer. A custom communication protocol based on the exchange of memory buffers through this DP-RAM parallelizes the I/O operations with the actual packet processing, completely hiding the latency of the I/O operations. As it has been shown in [8], this latency is a significant part of the total packet processing time, in all the existing architectures, and therefore by completely hiding it, we achieve a substantial acceleration of the overall network processing task.

When the packet-processing phase has been completed, the PPU informs the DMM which in turn it enqueues it at a certain output flow depending on the specified output network interface, the output flow rate and its priority. It is very important that an actual packet payload is stored to the external DRAM and read out of it only once, no matter in how many queues this packet is placed in while being processed.

5 PERFORMANCE EVALUATION AND PROFILING RESULTS

The GigaFlow architecture has been implemented in a Xilinx Virtex II Pro Platform FPGA [8] and tested on a development board equipped with a Virtex II Pro 2vp-70 chip, the required DRAM and SRAM memories and a quad optical Gigabit Ethernet transceiver. The current version with four PPUs and without the IPSec acceleration unit occupies 88.9% of the FPGA area, or in other words 66,184 Logic Cells, and achieves a clock speed of 100MHz. The development environment provided by Xilinx provides all the required tools for compiling the source code for the processors we used, booting the PowerPC CPU, initializing, configuring and simulating the design. We developed the application mostly in C-language (very few time-critical parts have been coded in Assembly) and simulated the design as well as experimented in a real laboratory environment. By not only experimenting but also simulating the placed & routed model of the Gigaflow we had visibility both on the output throughput as well as the processing latency in all the internal blocks and could identify the real bottlenecks of the design.

We measured and simulated different scenarios with a limited number of VLANs and MAC addresses (since the simulation time of the complete model of the chip is prohibitively long) and for a *worst-case* arrival pattern of 64-Byte Ethernet packets with 8 Bytes inter-frame gap. Due to the limited number of addresses all of them could be accommodated on the CMM cache. We have covered all possible packet flows (Ethernet-to-MPLS, MPLS-to-Ethernet, Ethernet-to-Ethernet) using either a single PPU or all four of them.

Table 1 indicates the processing delay per packet for all possible packet flows; it illustrates how the processing delay is analyzed based on the various processing subtasks (header extraction (HX), header modification (HM), forwarding (FW) and MAC learning). As shown in Figure 2 the current FPGA prototype, with 4 PPUs, operating at 100MHz can sustain a throughput of 300-450 Kilo-packets-per-second or in other words 155-255 Mega-bits-per-second (Mbps) even in the worst case of minimum-size 64-byte packets (and 8-byte inter frame gap). Those numbers, wherever possible, have also been verified by the experiments performed in the real device. Scaling the design and assuming a mid-size ASIC and a mature 0.18μm CMOS technology, clock speeds higher than 200MHz were achieved whereas up to 8 PPUs were easily integrated keeping the overall chip size within acceptable limits (less than 5mm x 5mm). Hence the total forwarding capacity was scaled by 4 times (i.e. having twice the clock speed and twice the number of PPUs) to rates up to 1 Gbps. The total complexity in that case was 3.43 million transistors, as reported by the ASIC placement & routing tools, when processing this design.

On the other hand, one of the most efficient NPUs proposed Intel's IXP2400 [3] supports a networking speed of up to 2.5 Gb/sec when executing such a network application, and it is an ASIC implemented in a 0.13nm CMOS technology utilizing 60 million transistors.

Therefore, we claim that the GigaFlow device is the ideal processing heart of a future, purely Ethernet, MAN or WAN since a very successful state-of-the-art Network Processor needs 16 times more silicon that GigaFlow while its performance is only 2.5 times higher than that of our system, eventhough GigaFlow is targeted to a much older 0.18μm CMOS technology; this is due to the fact that GigaFlow has been designed specifically for such an environment, whereas Intel's NPU, as well as all the others presented in the related work section, are more general purpose devices efficiently supporting a large range of different networking applications.

6 CONCLUSIONS

Multi-service, purely Ethernet, Access Concentrators are very demanding systems in terms of protocol processing and data forwarding capacity. Special Purpose systems are required in order to implement such nodes and Network Processors are becoming very popular in such environments due to the fact that they provide both high performance and software programmability. The GigaFlow system presented in this paper is a very cost-efficient alternative that can support both the state-of-the-art networking speeds and today's complicated networking applications. One of the most important such applications is the Ethernet/MPLS traffic aggregation which has been analyzed in detail and its bottlenecks, when executed in standard processing units, highlighted. The performance measurements of the GigaFlow prototype show that the presented design can support up to gigabit/sec rates, at an extremely lower cost (in terms of silicon area) than the existing network processing high-end solutions, since the whole design is tailored to a multi-service, purely Ethernet, environment .

Table 1. Processing delay per packet

	Average Processing delay (μs/packet)					
	1PPU	4PPU	HX	HM	FW	Learn
Eth-Eth	4.2	1.5	48%	0	12%	24%
Eth-MPLS	5.8	2	36%	40%	10%	18%
MPLS-Eth	6.00	1.7	40%	17%	10%	20%

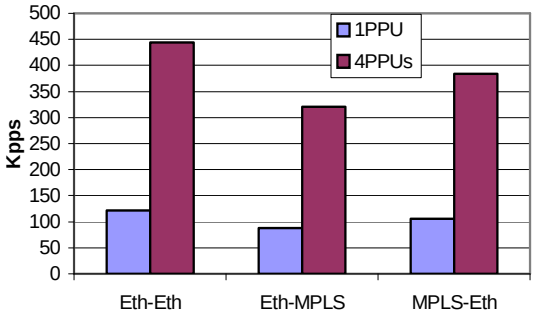

Figure 4 : System data forwarding rate for 1 and 4 PPUs

REFERENCES

[1] IEEE 802.1q Standard, "Virtual Bridged Local Area Networks".

[2] J. Allen at al. "PowerNP Network Processor Hardware, Software and Applications", IBM Systems Journal, Dec 2001.

[3] Sridhar Lakshmanamurthy, Kin-Yip Liu, Yim Pun, Larry Huston, Uday Naik, "Network Processor Performance Analysis Methodology", Intel Technology Journal, Aug 2002.

[4] J. Rexford, F. Bonomi, A. Greenberg, and A. Wong, "Scalable architectures for integrated traffic shaping and link scheduling in high-speed ATM switches", IEEE J. Sel. Areas Commun., 15 (3), (1997) 938-950.

[5] G. Kornaros, et. Al., "A Fully-Programmable Memory Management System Supporting Queue Handling at Multi Gigabit rates", in Proc. of the IEEE, ACM, 40th Design Automation Conference (DAC), California, U.S.A., June 2-6, 2003

[6] Virtex II-Pro, Data Sheet, Xilinx Sep. 2002.

[7] Douglas E. Comer, "Network Systems Design using Network Processors", Prentice Hall, Jan 2003

[8] G. Memik, S. Memik and W. H. Mangione-Smith, "Design and Analysis of a Layer Seven Network Processor Accelerator Using Reconfigurable Logic", 10th IEEE FCCM 2002, Napa, CA, April 23-24 2002.

[9] Broadcom's SiByte NP, http://sibyte.broadcom.com/public/

[10] Clearwater Networks CNP810SP Simultaneous Multithreading (SMT) Core

[11] D. Taylor, J. Turner, J. Lockwood, "Dynamic Hardware Plugins (DHP): Exploiting Reconfigurable Hardware for High-Performance Programmable Routers", *Computer Networks*, vol. 38, no. 3, pp. 295- 310, Feb. 2002

[12] C. Cachris, S. Vassiliadis, "Analysis of a Reconfigurable Network Processor", RAW 2006