A Comparative Study of Protocol Processors

Ulf Nordqvist, Dake Liu

Department of Electrical Engineering
Linköping University, SE-581 83 Linköping, Sweden
Phone: +46-13-28-{2903, 1256}
E-mail: {ulfnor, dake}@isy.liu.se

ABSTRACT

In the emerging research area of communication network processors, there exist many hardware platform proposals. One example of such a platform solution has been proposed by Linköping University. This paper briefly introduce this platform as well as the design considerations investigated in the research project behind. Further a comparative study of some selected competitive architectural proposals is included.

1. INTRODUCTION

Gigabit Ethernet (GE) today and 10 Gigabit Ethernet (10 GigE) tomorrow, provides the network bandwidth overhead to accommodate the rapid change of growth in organizations today. This growth is accelerated by the fact that there is a strong development towards an increased use of network protocols for applications where other techniques where common earlier, e.g. voice and video. One reason is that packet based network protocols can normally handle a mixture of any kind of traffic.

Analysis of GE workloads indicate that for every Gbit of network traffic a system processes, a GHz of CPU processing power is needed to perform the task. Therefore it is clear that new networking technologies are needed to reserve CPU resources for applications instead of wasting them on processing GE. Using traditional design methods we are already experience the I/O processing gap illustrated by figure: 1.

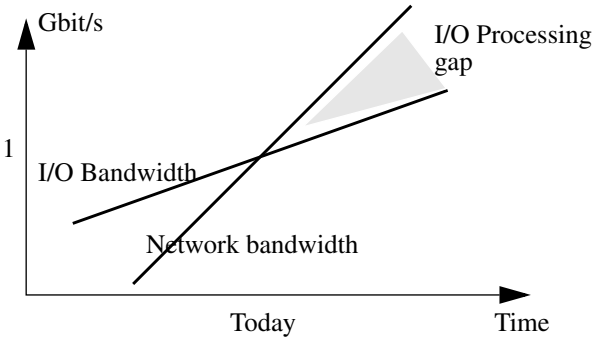

Figure 1. The I/O processing gap has started to become a problem using traditional CPU architectures. The reason is that while the I/O bandwidth approximately follows Moores law (1.5-2X) the Network bandwidth has a 10X improvement for each generation.

For network node components such as routers, switches and bridges, the performance needs can be fulfilled by using Application Specific Integrated Circuits (ASIC) or Application Specific Standard Products (ASSP) since these applications have quite moderate demands on flexibility and these hardware oriented implementation techniques supports tomorrows processing needs. In order to let the end-user take advantage of the bandwidth enhancement in today networks, tomorrows Network Terminal (NT) hardware must support transmission speeds of Gbit/s. Hardware for such NT components is on the other hand sold on a cost-sensitive market share with high demands on flexibility and usability.

Traditionally NT has been implemented using ASIC:s for the lower layers in the OSI-Reference Model with an CPU-RISC based SW implementation of the upper layers including TCP/IP, or completely implemented in software. Usage of standard, general purpose CPU:s, is expensive in terms of cost, space and power due to their lack of dedicated hardware. There is also an upper capacity limit, set by the I/O capacity and the instruction rate of the CPU. By reducing the memory access both memory bottleneck problems and power consumption can be reduced. In order to achieve this, the protocol must be processed at network speed.

To meet these new requirement a new area of communication handling hardware platform solutions have emerged. The solutions available from the academic research community and primarily the industry are extremely diverse depending on their application areas. However this diversity, the communication network platforms can be divided into four main groups according to figure: 2. Depending on application, throughput requirements, power awareness and customer cost sensitivity different platforms selects one of the four different offloading strategies while offloading the host processor by processing packets in layer 2 and up to layer 3 to 7. Depending on applications and public rela-

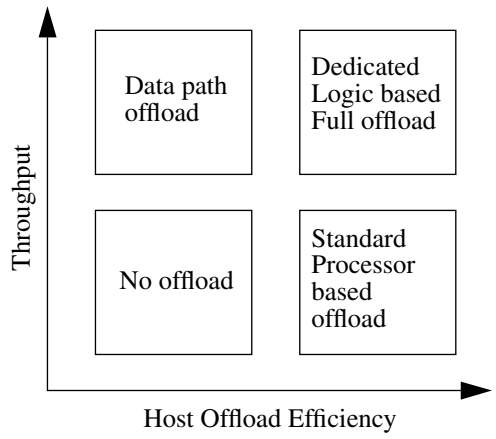

Figure 2. Host offloading strategies

tion criterias these platforms have different names. Common names on various communication network platforms are:

- Network Processors (NP)
- TCP Offload Engines (TOE)
- Protocol Processors (PP)

No naming convention has yet been agreed on, and when the author of this paper selected a name, NP and TOE did not exist. Therefor our name has been PP since 99.

A PP architecture was proposed by the authors in [1]. As most of the protocol processors it consist of more or less programmable devices that can accelerate and offload a host processor, by handling the communication protocol processing. The protocol processor is a domain specific processor that have superior performance over general purpose CPUs but still provides flexibility through programmability within the application domain. The hardware platform of the protocol processor is further discussed in chapter section 3.

In section 2 different application areas and their different needs for hardware acceleration are examined further. In section 4 a survey of commercial and academic architectures can be found. Finally in section 5 further work is discussed and some concluding remarks are being made.

2. DESIGN CONSIDERATIONS

Protocol processors can be, and are being used for many different applications. The only thing these applications have in common is that they use packet based communication network. However versatile these application may seam, there still exist some common requirements that they all put on an communication offloading device. When designing such a device it is of course most important to meet these requirements. On the other hand there also exist a need for detailed analysis of various kinds of requirements each of these application put on a PP, before deciding or designing an appropriate PP architecture. The first issue to consider is how much offloading is required. This strongly depends on the network traffic type and the load the application puts on the host. A fully offloaded dedicated hardware solution becomes much more complex compared to programmable or just datapath offload. Datapath offload means that the data intensive computations which mainly occurs in the lower layers are being accelerated in dedicated hardware. Datapath offloading is today performed in all existing solutions. The question is how big part of the control intensive should be processed using dedicated hardware. Secondly a thorough exploration of the network data types and protocols must be performed in order to find out which kind of accelerating hardware to use in order to obtain an optimal Application Area Domain Specific Processor (A2DSP). Thirdly, both the physical network and the application(s) in the environment where the PP will be used, require power dissipation, throughput and latency parameters to be met.

2.1. Acceleration possibilities

In order to design an optimal PP there is a number of common protocol processing task that one has to consider to process using dedicated hardware accelerators. Some of the most obvious are:

- **Hardware Assisted Timers**. Many protocols and especially TCP uses a number of timers in order to update the connection states.
- **Error Control**. Up to layer 3 CRC and checksums are normally processed in dedicated hardware in most PP but there may exist ULPs including checksums that benefits from hardware assist.
- **Connection State Access**. To determine which connection state variables to access and then provide them to the processing unit, can be hardware assisted which reduces the decision latency.
- **Switching datastreams.** It is important to recognize different data streams in order to decide what to do with them and where to send them for further processing.
- **Encryption/coding.** It is natural to use dedicated embedded encryption/coding hardware if this type of data is common in the network.
- **Memory controller.** The host would be significantly offloaded if the I/O flow to and from the main memory would be controlled outside the host although this might effect the host OS.
- **Embedded memory.** An on-chip internal memory reduces access latency to data and improves the scalability.
- **Program Flow Selection**. Depending on the type of data that comes into a PP, the PP must jump to different program flows. This case-jump operations can be hardware assisted in order to obtain minimal latency. One way of doing this is shown in [4].

2.2. Flexibility requirements

A PP must provide flexibility and adaptability to the changing environment it might operate in. This results in some flexibility requirements, all PP has to meet to some extent:

- **Reconfigurable media adaptation.** In order for PP to be used in different networks and survive over time it must be capable of adaptation for different medias.
- **Programmable connection policy.** A PP must support on-line change of, and control of, the traffic flow.
- **Programmable host interface.** The interface between the PP and the host system must be operating in real time and be highly flexible in order to avoid unnecessary interrupts in the host.
- **Data controlled datapath selection.** The datapath must be configurable or selectable depending on the data header information.

3. PROPOSED ARCITECHTURE

The PP solution presented in this section, as well as the research project behind, only deals with packet reception for NT. Since the transmitting of packets is limited by the applications construction of packets and have lower demands on low latency it is natural to solve the packet reception problem before discussing packet creation acceleration.

In [1] we first proposed the Programmable Protocol Processor (PPP) described in this section.

3.1. System Integration

The protocol processor architectures consists of 2 main units. First we have the PPP responsible for data intensive processing tasks. Secondly we have a micro controller responsible for the control intensive processing task. The system integration of this architecture is illustrated in figure: 3.

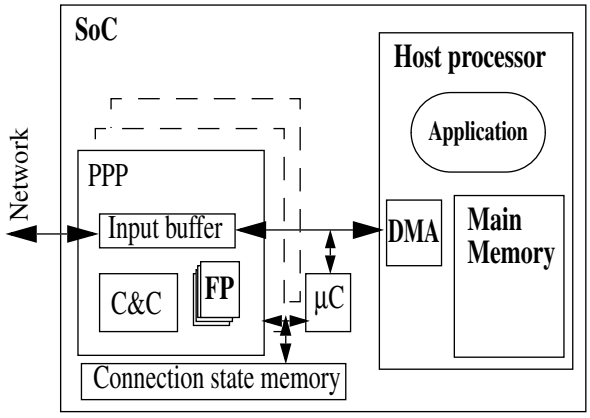

Figure 3. The PPP together with a general purpose micro controller (µC) handles the communication of one network media port. In a system on chip (SoC) many PPP can be used as port-processors in order to provide high bandwidth between the application and the network.

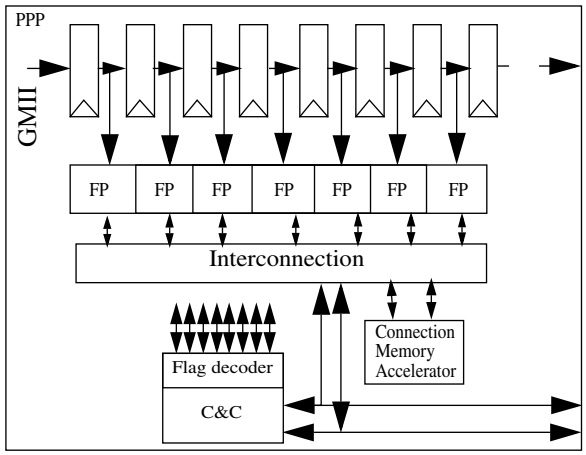

Figure 4. The programmable protocol processor consists of 4 parts: The Counter and Controller, the input buffer chain, accelerating functional pages and a connection memory.

The PPP architecture is illustrated by figure: 4. The PPP operates on a data stream that flows through the pipelined registers. This means that we don not have to store the data before passing it on to the host, which saves power. It also reduces the fan-out from the incoming buffer.

3.2. Functional Pages

The functional pages (FP) are small accelerators for data intensive processing. FPs can normally be configured for different processing tasks using configuration registers. They uses a very limited amount of control signals, in fact flags and configuration vectors are normally enough.

In order to perform protocol reception for TCP/IP/Ethernet including ARP, RARP and UDP, this architecture requires the following FP which have been implemented in RTL code.

- **CRC -** Cyclic Redundancy Check [2]. CRC computations are common in many different protocols. By using a configurable CRC we can adapt to most of them without loosing the high throughput. Such an implementation is illustrated by **figure: 5**
- **XAC -** Extract and compare unit. This unit can extract data from the incoming data stream and also compare data. This is used for protocol recognition.
- **ADD -** Adders. There is a need for several adders in the PP, both 1-complements and 2-complement adders. These are used for counters checksum calculations etc.

This platform also supports for Timer acceleration as well as Decryption, Decoders and SSL functions to be

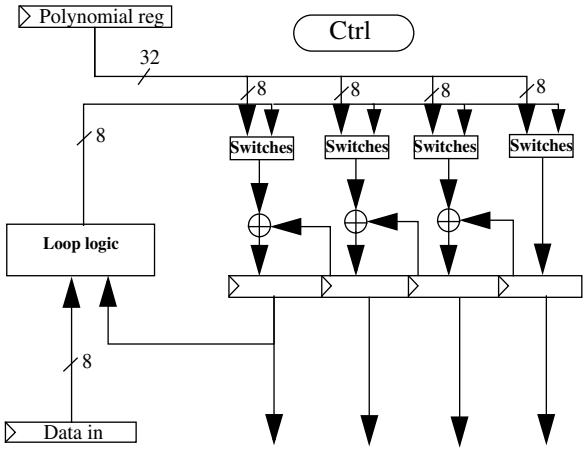

Figure 5. Bytewise calculations enhances the throughput of the configurable CRC. By changing the content of the polynomial register any polynomial up to 32 bit length can be processed.

implemented as FPs, but they remain to be implemented in hardware.

3.3. Control

The Counter and Controller (C&C) unit in the PPP controls the different FP using flags. These flags starts and stops the processing in the FPs. In order to save power the C&C shutdowns the FPs when a packet is discarded or if nothing is received. For XAC FPs the C&C also provides some configuration.

3.4. Micro controller (μC)

Most of the control intensive processing in a protocol processor consists of state machine processing. This type of processing is exactly what general purpose micro controllers are optimized for. The μC also handles the interface to the host OS and DMA. Further the μC initiate and controls the configuration of the data path during setup. The μC uses a embedded memory for connection state variables. When a packet is received the address for the current connection is sent to the μC from the connection memory accelerator in the PP.

3.5. 3-layered flexibility

The PPP architecture has a layered flexibility. First of all the FP can during design face be designed for different functionality and speed. Secondly the configuration registers can be used during run time to change their operation mode. Finally the C&C and interfaces can be programmed using the μC. The processing performed in the μC is of course programmable. This makes the PPP architecture very flexible.

4. COMPARISON WITH EXISTING SOLUTIONS

In order to evaluate the proposed PP architecture a comparison with other research or industrial designs is interesting. This comparison is however hard to do since the academic research area is so immature. Out of the roughly 30 companies today working in this area, half of them are still in what they call "*stealth mode*" so they are also very hard to compare with. Despite this, this section gives an overview of some of the existing applications, architectures and their performance figures.

4.1. iSNAP

The IP Storage Network Access Processor from Silverback [5] terminates and process IP-based storage traffic in a GE with full duplex. It separates the header and data traffic processing. The header processing generates a event which is placed in a queue that communicates via DMA to the host. Meanwhile the packet data is stored in a DRAM until the event is finally created. At the host level the data can then be stored in separate application buffers depending on the upper layer protocol (ULP). This is called PDU awareness. ULP covered are iSCSI, NFS, CIFS and main application areas are servers, storage devices and Network Area Storage (NAS) appliances.

4.2. Trebia SNP

This architecture [6] includes MAC block for mixed medias (wired and fibre-based), a security accelerator, various classification block, a TCP offload engine and a Storage Area Network (SAN) [1] Protocol processor as illustrated by figure: 6. The TCP offload engine can work stand alone terminating TCP connections without involving the host processor. For IP storage applications they claim that their TCP offload engine manage up to

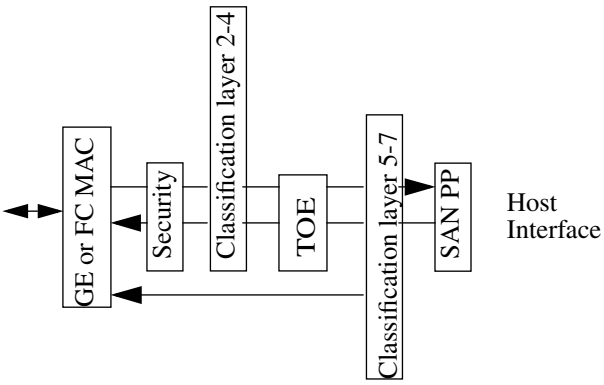

Figure 6. Trebia SNP architecture.

1. Storage Area Networks (SAN) today sees a rapidly increasing use of PP to offload the host. The host is then typically acting as a file server.

10 GigE. The SAN PP is optimized for processing of storage I/O flows and especially iSCSI termination.

4.3. iReady EthernetMAX

The Media Access Xccelerator [8] from iReady is intended for transport offload [7]. It fully terminates TCP/IP at GE speed. The TCP/IP accelerator uses a streaming data architecture similar to the one proposed by this papers author. The data is not stored but instead processed while it is streaming through a 64 bit wide pipeline. The 64 bit wide datapath then process the data using multiple dedicated hardware blocks implementing different state machines. Each state machine block process a specific part of the incoming headers. The processor also uses hardware acceleration of iSCSI and IPSec. Since the complexity of the IPSec processing is 2 to 3 times higher than TCP/IP this architecture is not suitable from a power and cost point-of-view if the use of IPSec packets not is large in the network. The implementation does not use standard programmable devices. Instead dedicated logic for optimal performance is used.

4.4. Alacritech Internet PP

Alacritech [9] provides a Session Layer Interface Card (SLIC) [11] that includes accelerators for GE, network acceleration [10], storage acceleration and dual-purpose server and storage acceleration. Especially their Internet PP (IPP) which offloads TCP/IP and iSCSI processing is interesting. The IPP offers acceleration of non-fragmented TCP connections. This means that data transfers to and from the TCP/IP stack is handled by the IPP while the host system must take care of the connection state processing. Parts of the TCP that IPP does not handle are:

- TCP Connections and breakdowns (SYN segments)
- Fragmented segments
- Retransmission timeout
- Out of order segments
- FIN segments

Despite this down-sized functional coverage in the accelerators, Alacritech claims that 99.9 percent of the TCP/IP traffic is handled by the IPP while the other 0.1 percent is processed by the host processor. Alacritech further stresses the low power and low cost figures of their architecture.

4.5. LayerN UltraLock

The UltraLock [13] illustrated by figure: 7 uses a patented architecture named SIGNET [12]. The UltraLock chip offloads both the Network processing including packet classification and provides acceleration of Secure Socket Layer (SSL). The UltraLock also includes GE MAC accelerators.

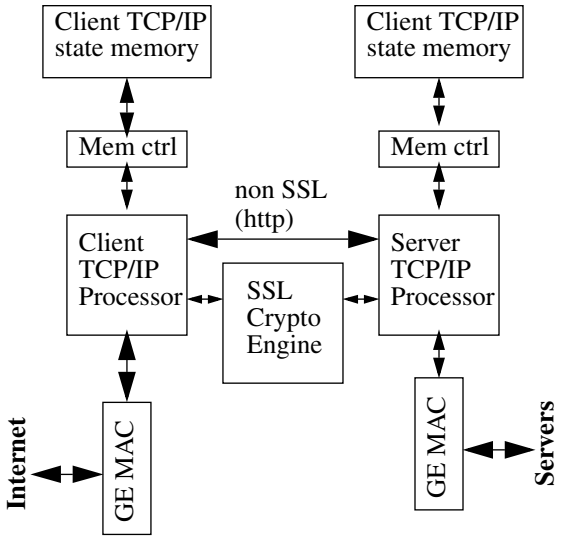

Figure 7. The UltraLock provides acceleration for SSL connections. Ordinary http packets are passed on without any processing in the SSL engine.

In the TCP/IP processor the tasks are distributed among several different dedicated functional blocks in order to improve the throughput. These TCP/IP processors are also pipelined.

4.6. Seaway Streamwise NCP

Seaway Networks [14] offers a streamwise Network Content Processor (NCP) capable of multi-gigabit layer 4 (TCP) termination. The NCP also examine, modifies and replicate data streams based on their content (Layer 5-7). The NCP uses a streamwise switch to send data streams to different content processing devices, e.i. co-processors or general purpose CPUs.

4.7. Emulex LightPulse Fibre HBA

The host bus adapter (HBA) from Emulex [15] includes an ASIC controller, a RISC core and a SAN accelerator. The SAN uses a context cache hardware so that context (PDU information) not must be transported to and from the host and thereby offloading the server PCI bus. The systems have 1 Gbit/s performance and the main feature is the implementation of a strong SAN accelerator for high end servers.

4.8. Intel IXA/IXC/IXS/IOP processors

Intel offers a number of chips to solve different tasks when it comes to what they call Network Infrastructure Processing [17]. First of all they have the Internet eXchange Architecture (IXA) which includes different NP. They uses Xscale instruction set (improved StrongARM) and the peak capacity is today 10 Gbit/s using high end chips while the normal IXP 1200 uses Fast Ethernet. In for example the IXP 1200 the datapath includes 6 different micro engines which supports multithread programmability. This micro engines uses a

application specific instruction set. The IXA type chips is mainly intended for packet processing for switching, protocol conversion, QoS, firewalling and load balancing. Further Intel offers Control Plane Processors in the IXC family. IXC is mostly efficient when they are being used for exception handling and connection states processing. They are normally used in high end systems, e.g. Base Transceiver Stations, Radio Network Controllers and MAN servers. They normally operates together with a IXA type of chip handling the control plane processing. The IXS family contains Media Processors which can be seen as Protocol Processors for acceleration of voice, fax, and data- communication. In a big server a number of these IXS could be used together with one IXA chip. Finally Intel offers I/O processors (IOP) that is a quite general architecture which can be used for SAN acceleration.

4.9. LeWiz Content processor

LeWiz processor [16] process layer 3-7 with hardware acceleration with a line rate capability of Gbit/s. Among other things it performs table lookup for connections, controls a external header data memory, support different types of connections based on URL/source address, and handles XML and URL switching. LeWiz sells both hard and soft cores. The content processor architecture is further described in figure: 8.

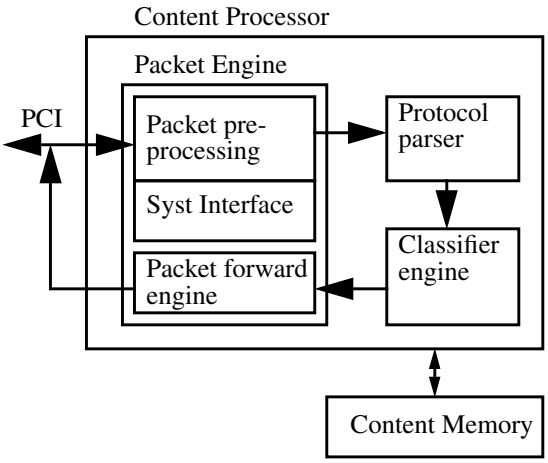

Figure 8. LeWiz content processor. The Packet preprocessor is a TOE. The Protocol parser examines the ULP data (layer 5-7) and based on this it start a search for a classifier using the classifier engine. The classifier then decides priority and is used for redirection of the traffic according to the QoS policy.

4.10. Qlogic SANblade

The SANblade [18] manage 2 Gbit/s line rate using GE or fibre channel medias while performing iSCSI as a HBA. It completely offloads the TCP/IP protocol stack from the host. The SANblade also handles all I/O processing. The SANblade contains internal on-chip memory which they claim to be faster, cooler and moore scalable than using shared memory architectures.

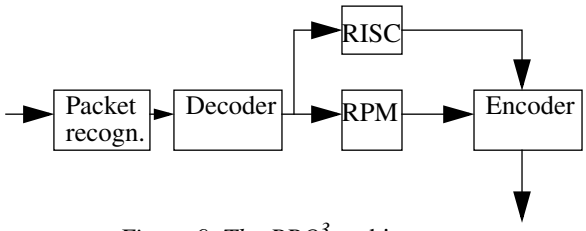

Figure 9. The PRO^3 architecture.

4.11. EU Protocol Processor Project PRO^3

The architecture proposed by PRO^3 [19] consists of 5 parts. Most interesting is the Reconfigurable Pipelined Module which process the data intensive tasks, and the embedded RISC core which takes care of the signaling processing. An illustration of the PRO^3 can be found in figure: 9.

4.12. UCLA Packet decoder

This decoder [20], decodes packets on layer 2-4. The decoder architecture illustrated in figure: 10 consists of one datapath for each layer, e.i. 3 data paths totally. It only uses one control path for the signaling processing. It operates on streaming data using a application specific instruction set and the intended application area is routers.

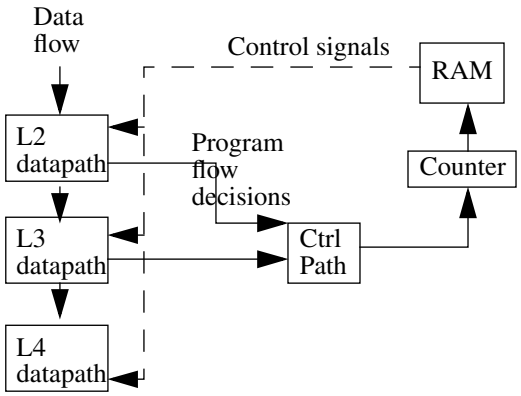

Figure 10. Simplified view of the UCLA processor architecture proposal showing how to accelerate case-jump functions.

5. CONCLUSIONS AND FURTHER WORK

As one can see from the market exploration in section 4, a trend against separation of the protocol processor area into A2DSP optimal for a certain application area emerges. There is a general disagreement on how much functionality to include in the PP and how much should be left for the host. Instead it is clear that TOE, MAC, Encryption accelerators and SAN control accelerators

are being designed and optimized independently. Hopefully this means that we soon can have standardized interfaces between different communication accelerators. In the future we will surely see new protocol processing application areas where A2DSP are worth using. SAN is just the first one becoming commercially interesting. There is no clear trend on the amount of offloading needed in a TOE for NT so here further exploration is needed. One big question that remains unanswered is where the re-ordering of the incoming application data should be done. The question is if the data should be delivered to the main memory unordered or if it should be stored in order in the application buffers. The second alternative demands an embedded data memory to be used. This have off course a big impact on the OS of the host. The comparison clearly shows that there exists solutions to the various new PP specific implementation problems and considerations the research community just have started to examine. Examples are host OS interface, shared memory control etc. Finally this comparison shows that, even if half of the industrial initiative are still working in stealth mode, the PP research project in Linköping University is on the right track.

Continues area, power and critical path analysis is important to perform in order to compare our solution with other alternatives. Implementation studies of timer acceleration is also of interest. Further work must also include a study on the impact the PPP solution have on the host system OS in order to improve the interface.

6. ACKNOWLEDGMENTS

This research was supported by ECSEL graduate research school.

REFERENCES

[1] D. Liu, U. Nordqvist, and C. Svensson, “Configuration-Based Architecture for High Speed and General-Purpose Protocol Processing”, *IEEE Workshop on Signal Processing Systems*, Taipei, Taiwan, 1999, pp. 540-547.

[2] U. Nordqvist, T. Henriksson, and D. Liu, “CRC Generation for Protocol Processing”, *Norchip 2000*, Turku, Finland, pp. 288-293.

[3] Semiconductor Industry Association, *International Technology Roadmap for Semiconductors: 1999 edition*, Austin, TX:International SEMATECH, 1999.

[4] T. Henriksson, U. Nordqvist, D. Liu, “Embedded Protocol Processor for Fast and Efficient Packet Reception", *will appear in*, ECCD 2002

[5] Silverback Systems homepage, *on the www*, http://wwwsilverbacksystems.com

[6] Trebia Networks homepage, *on the www*, http://www.trebia.com

[7] National Semiconductors, “Enabling Next Generation Ethernet”, *on the www*, http://www.trebia.com/EthernetMAXweb.pdf

[8] Minami, et al, “Multiple network protocol encoder/decoder and data processor”, *US patent, no. 6 034 963*

[9] Alacritech Inc. homepage, *on the www*, http://www.alacritech.com

[10] Boucher, et al, “TCP/IP offload network interface device”, US patent, no. 6 434 620

[11] Boucher, et al, “Intelligent network interface system method for protocol processing”, US patent, no. 6 434 620

[12] LayerN Networks, “SIGNET - Secure In-line Networking”, *on the www*, http://www.layern.com/SIGNETWP020419.pdf

[13] Omura, et al, “The Evolution of Modern Digital Security Techniques”, *on the www*, http://www.layern.com/EvolutionWhitePaper.pdf

[14] Seaway Networks homepage, *on the www,* http://www.seawaynetworks.com

[15] Emulex homepage, *on the www,* http://www.emulex.com

[16] LeWiz Communication,Inc. homepage, *on the www*, http://www.lewiz.com

[17] Intel Corp., “Network Infrastructure Processors - Extending Intelligence in the Network”, *white-paper on the www,* http://www.intel.com/design/network/papers/251690001.pdf

[18] QLogic Corp. homepage, *on the www*, http://www.qlogic.com

[19] G. Konstantoulakis, V. Nellas, C. Georgopoulos, T. Orphanoudakis, N. Zervos, M. Steck, D. Verkest, G. Doumenis, D. Resis, N. Nikolaou, J.-A. Sanchez-P., “A Novel Architecture for Efficient Protocol Processing in High Speed Communication Environments”, ECUMN 2000, pp 425-431

[20] M. Attia, I. Verbauwhede, “Programmable Gigabit Ethernet Packet Processor Design Methodology”, ECCTD 2001, vol. III, pp. 177-180