Malaysian Journal of Computer Science, Vol. 16 No. 2, December 2003, pp. 21-37

SURVEY OF NETWORK PROCESSORS (NP)

K. Ettikan
School of Computer Science
University Science Malaysia
11700 Penang, Malaysia
Tel: + 604-6579581
Fax: +604-6573335
email : ettikan@cs.usm.my

Rosni Abdullah
School of Computer Science
University Science Malaysia
11700 Penang, Malaysia
Tel: + 604-6579581
Fax: +604-6573335
email : rosni@cs.usm.my

ABSTRACT

Network processing is becoming increasingly challenging to the network architects and engineers in terms of hardware design and application development due to an increase in packet processing complexity and constantly evolving protocol standards. New inventions in the transmission medium such as DWDM, SDH and GigaEthernet increase bandwidth capacity of the network. Meanwhile, more network-oriented applications are becoming popular. All these require faster and programmable packet processing capabilities in the inter-connecting network nodes. Packet processing technology of network equipment is seeing a migration from ASIC solutions to NP. In this paper, we review the latest technology of NP, which has been designed today for next generation networks. NP has to adapt to rapid protocol standards change and perform at wire speed like ASIC solutions, using considerably easier programmable NPs besides maintaining short time-to-market and time-in-market which is essential to meet tomorrow's network demand. This paper discusses the trend in NP architecture, packet processing classification functions and the challenges ahead for the network processors architecture. The authors feel this is the first survey paper on NP.

Keywords: ***Network Processor, Packet Processing, Parallel Processing, Address Lookup and Programming Element***

1.0 INTRODUCTION

Migrating from the previous telecommunication network which was more for voice to the Internet for all types of communications, audio, video and data regardless real time or non-real time is a major turning point in the telecommunication industry. Introduction of various new hardware, software and protocol advancement in the network field such as optical fiber, GigaEthernet, Dense Wavelength Division Multiplexing Dense Wavelength Division Multiplexing (DWDM), Synchronous Digital Hierarchy (SDH), faster microprocessor and integration of voice into packet communication keeps the Internet traffic growth ongoing. This requires enormous bandwidth, faster and more sophisticated packets processing at interconnecting nodes, which decide the destination of each packet on the Internet. In order to support such a complex need, many companies including Intel [5], IBM [7], Broadcom [11], MMC [8] etc. are introducing new programmable network processors. Previously used ASIC (Application Specific Integrated Circuit) is not preferred due to its long and expensive hardwired design and development cycle time which does not have programmability feature. Furthermore, it imposes a higher risk of reversibility for protocol changes. The transition from ASIC to NP solution for network packet processing equipment is deemed possible due to the above reasons. High speed packet processing is becoming almost impossible by a single chip solution.

Table 1: Average packet size processing time requirement

Media Speed		44 bytes packet processing time
155Mbps	OC-3	2.3us
622Mpbs	OC-12	514ns
2.5Gbps	OC-48	128ns
10Gbps	OC-192	32ns
40Gbps	OC-768	8ns

At OC-768 line rate, assuming packet size of 44bytes, packet arrival rate is about 114 x10^6 packets per second that need to be processed. If the line-rate is to be met at any time, each packet should be processed within 8ns. It has been estimated that in average each packet requires 500 instructions such as address look-up, classification, encapsulation and transposition. In conclusion, for OC-768 line rate, the network processor requires 57 GIPS, which at least for the moment cannot be processed by a single processor.

This paper reviews the existing NP technology with state of the art solutions and explores further the challenges ahead for the specific building block of NP. All the major building blocks of a NP is detailed and discussed by comparing different vendors' solutions in terms of their architecture, design and implementation if available. Research articles [4, 20, 23, 24, 25, 26, 35, 36, 37, 38, 39] in this area were explored with industrial NP solutions for comparison purposes. Section 2 generalises today's NP architecture. Section 3 details all major component architecture of a NP. Network packet processing steps are explained in Section 4. Simple implementation of table look-up is discussed in Section 5. NP benchmarking effort is described in Section 6. Section 7 discusses the industry standardisation work. Finally, Section 8 concludes the paper and describes the direction of our future research work.

2.0 TODAY'S NPS GENERAL ARCHITECTURE

There are many NPs [4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 21] with distinguishable architecture designs to suit specific network configuration, types of packets being processed, speed of the network physical layer, packet processing approach such as parallelism or pipelining, multithreading capability, co-processor support, memory types with various access speeds, different types of data flow and queue management techniques either by hardware or software approach. These NPs are composed of various components with different configurations depending on the design specification. However, the common objective is to process network packets for the particular network, maintain wire-speed if possible and give programmable flexibility to adopt rapidly evolving protocol definitions. Common components of a NP are:

i) Configurable or programmable processing elements (PE) with or without multithreading capabilities.
ii) Local cache memories with different access speeds.
iii) External memory interfaces with different access speeds and sizes such as SRAM and SDRAM.
iv) Function specific co-processors or ASICs such as look-up.
v) Packet classifiers and queue managers, high-speed internal bus for component connectivity.
vi) External network connection bus and interfaces.
vii) Switch fabric support.
viii) Connection to external general processor.
ix) Hardware specific instruction set.
x) Compilers.
xi) Debugging and programming tools with some GUI support.

Fig. 1 illustrates the above-mentioned features in a NP. NPs are usually supported by Control Plane Processor (CPP) for control plane packet processing whereas General Purpose Processor (GPP) for Management Plane packet processing is optional. CPP and GPP have their own supporting units that enable them to work independently from other components of NP. Communication interfaces between these processors and data plane processor are present when decoupling of tasks is necessary.

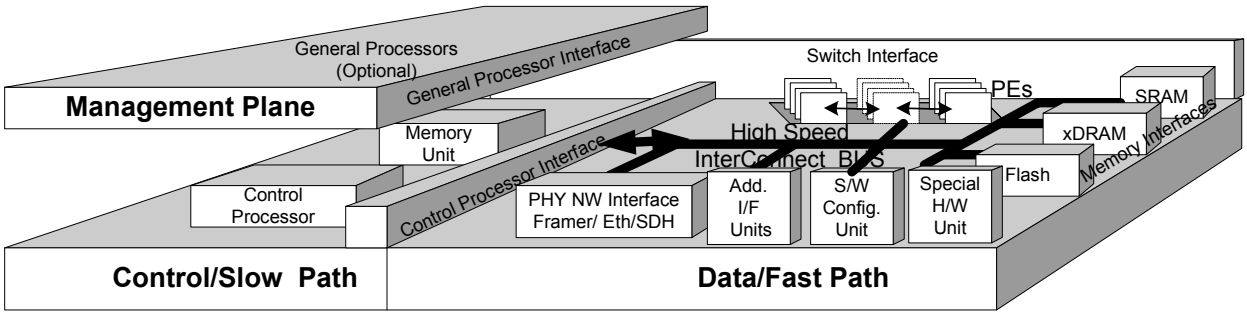

Fig. 1: General Network Processor Architecture

In summary NP can be classified as a RISC processor which has been specifically architectured to process network packets with or without support of other peripheral processors such as GPP or CPP.

2.1 Classes of NP

Network processors reside in four different network sectors, namely Customer Premises or Home Network (CPHN), Access Network (AN), Edge or Metro Network (EMN) and finally Core Network (CN) [25]. Combination of the above common components will vary based on the particular sector that NP resides. CPHN NP does not require high processing power since the required bandwidth is low, ranging from 1Mbps to tens of Mbps but less than OC-3. Such NPs also do not require Management Plane, additional interface units, switch interface or special hardware units. They perform service termination functions and act as routers, layer three to layer seven switches, Storage Attached Network (SAN) Servers and also Wireless LAN packet processing. There are many NPs for this market segment such as MMC nP7020 [8], Intel IXP22x [5] and Vitesse IQ2100 [21].

Some larger organisations may require a much larger bandwidth for switching and routing, up to 1Gbps/OC12 such as Gigabit Ethernet. This can be categorised as *AN* sector where data and voice traffic merge. Service aggregation by small Internet Service Providers (ISPs) happens in this sector. NPs from Intel IXP12x0 [5], Vitesse IQ2102 [21], Broadcom BCM112x [11], and MMC nP7xx0 [8] fit into this market segment. *EMN* involves multiple city connectivity or larger national ISP connections. IP service switching, Quality of Service, bandwidth aggregation, voice or data gateways and Service Level Agreement switching are some of the supported services in this sector. This requires connecting bandwidth, ranging from 1 Gbps to 2.5Gbps/OC-48 and equipment such as DSLAM, 3G wireless Node-B and RNC use NPs to perform their network functions. IBM Power NP4GS3 [7], Motorola C-Port C-5 [14], Agere Payload Plus [10], Vitesse IQ2132 [21], Xelerator X10x [12], and MMC nP7250 [8] and Intel IXP2400 [5] are some examples of NP in this segment.

CN connects multiple regions and countries for the Internet traffic exchange. This sector also includes ISP exchange points. It performs complex functions such as IP service switching and routing, Quality of Service, bandwidth aggregation, voice or data gateways, MPLS and Service Level Agreement switching for the highest bandwidth requirement. 2.5 Gbps to multiple 10Gbps/OC192 bandwidth equipment reside in this category. Some of the available NPs in this market segment are from MMC nP7510 [8], Intel IXP2800 [5], Bay Microsystems Montego [9], Lexra [6], Broadcom BCM1250 [11], Agere Payload Plus 10G [10] and EZ-Chip NP 1[13]. The bandwidth requirement will continuously grow in the future and higher speed NPs such as OC728 will be available very soon. Technology advancement such as SDH and DWDM will pave the way for this realization.

Serving very high bandwidth network or merging network points require immense computing resources. This can be resolved by cascading NPs by processor pooling (PP) [1]. Another solution is channel specific Distributed processor (DP) [2]. Motorola C-5e [14] is one example of the second method. DP decentralises the processor to each channel where the packet processing activities take place. In order to meet wire-speed processing rate, in PP the processors are pooled together to handle the network packets from any channels. As the processing power requirements increase, the number of NPs can be augmented. This first approach allows internal NP scaling without any changes to the attached external interfaces. Processing functions will be distributed among all the NPs by adopting functional or context pipelining with parallel techniques in mind. This is suitable if similar processing techniques are required with less predictability for the packet processing. Fig. 2 illustrates both these architectures.

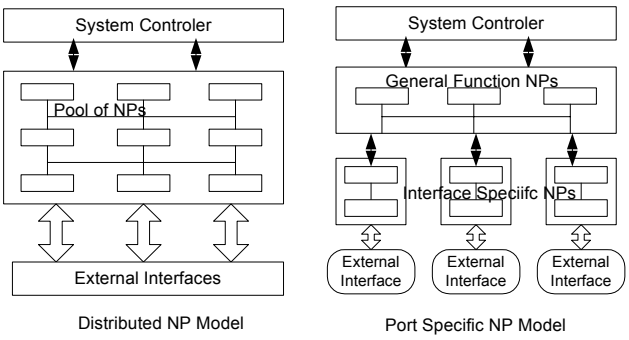

Fig. 2: NP organisation in packet processing units

The second method brings packet processing near to the source, hence allowing channel dedicated packet processing with direct links to the external interfaces. Channels with varying processing needs are suitable for this technique. Packets that were destined to other channels will be forwarded via interconnecting bus to be processed and en-

queued for transmission. The combination of both methods is also possible to provide packet-processing flexibility in turn for higher speed.

Generally NPs use shared bus for inter-component communication. The communication is required for packet movement such as for packet pulling from input queue or packet pushing to output queue, memory reading and writing operations, operation of specific ASIC functions to the packet such as encryption, decryption, cyclic redundancy check (CRC) and etc. Tens of GHz transfer speeds are required to ensure the data movement within the components. Assuming, for any given 44bytes packet size at OC-768 line rate, functions such as classification, look-up, translation, encryption and error checking executed with average 100 memory access per packet, for bus width of 16 bits, then 2500 GHz speed is required. By distributing the load to a larger number of nodes such as 8 or 16, the speed requirement will reduce to tens of GHz, which is feasible to achieve at today's technology. Considering the future limitation, octagon architecture [3] has been proposed compared to current bus architecture. It has been demonstrated by STMicroelectronics [4] that inter-processor or component communication speed requirement can be met by this architecture. However, most of today's industries' NPs are connected via bus architecture.

3.0 NP SPECIFIC FEATURES

The following section gives an in-depth explanation on current trends and design approaches and challenges for the major components of a NP. These components play an important role in NP packet processing function and behavior.

3.1 Memory Requirement

Memory has always been an important factor in any network equipment design due to the cost. Faster memories cost more and in return give better access time. An OC-12 line rate NP will handle 1.77 million packets per second if 44 bytes size of IP packet is adopted. Assuming that the line rate is maintained and packets are being processed at least at arrival rate per channel, approximately 80 MB of memory space is required. If 15% headroom is given to other memory storage, then 92 MB of memory will be used. For four channel support, memory requirement quadruples to 368MB. OC-192 line rate requires more than 5 GB memory space for four channels which is costly. Fig. 3 illustrates the minimum memory required for various line rate per packet size with the above assumptions. Thus, an effective and efficient memory management is necessary to ensure cost effective network products.

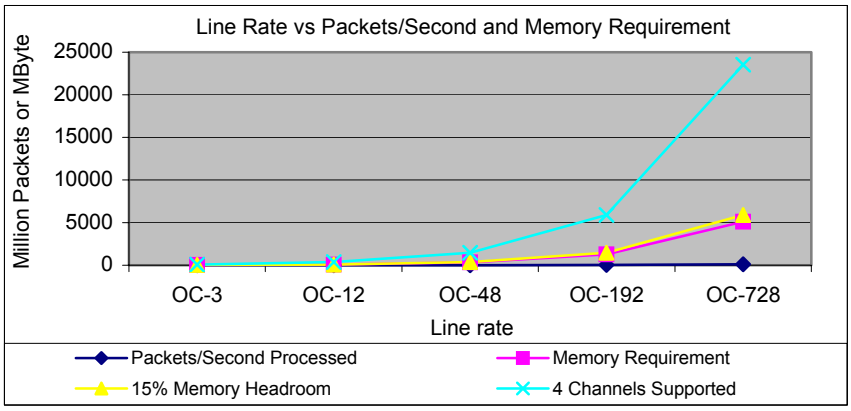

Fig. 3: Memory requirement and packets being processed at various line rates

Large memory requirements are unavoidable due to the complexity of packet management processes within the NP. For example, TCP flow control packets create a ping-pong scenario where they need to be maintained for each flow, to ensure reliable transfer. Metering and time-stamping activities to ensure QoS per flow traffic require packet storage. Advanced queuing algorithms implementation per flow in the buffer to maintain Class of Service (CoS) and/or Quality of Service (QoS) is essential. Flow control is also important for a network node that handles many channels to ensure fairness among all the ports and to ensure that congestion is avoided at all times.

Algorithms such as Deficit Round Robin (DRR), Weighted Random Early Detection (WRED) and Forward Error Detection (FED) have to be implemented to manage large amounts of packet flow to be received, stored, processed

and transmitted out by the router. Services such as ATM, require strict quality of service, while maintaining minimum latency and jitter for guaranteed services. Besides that, the deeper the packet processing is within a packet, such as layer 5 filtering, more memory will be required since multiple layer headers need to be stored before processing takes place at the payload. As such, the above assumption of memory space may be insufficient, if more complex tasks need to be performed by a NP.

NP requires memory to store incoming or outgoing packets for short/long term, executing codes that were developed to run routine functions, temporary data and permanent data. Incoming or outgoing packets are usually stored in buffers before they are moved to long-term memory. Registers are examples of short-term memory and xDRAM is a long term memory. Long term memory is used to store packets that need to be processed further. Data structure such as queues and link-lists are used to manage this complex memory region for best utilisation. SDRAM, DDRAM and QDRAM are used for storage purposes. Developed code that needs to be executed by the PE is normally stored in code store memory. Machine instruction sets which are usually short and simple, are used for data movement. Temporary variable data which are passed from one function to another or from one component to another such as intermediate results of processing functions, are stored in cache, scratch, CAM and TCAM memories.

Assuming that there is enough memory space, memory access rate is still an issue. OC-192 line rate requires data to be accessed and processed at 40 Gbps. Today's commercial DRAM only supports hundreds of Mbps which is far below the required speed. As such, sophisticated memory interleaving techniques that use context pipeline memory access coupled with distributed memory architecture with multiple memory units are necessary to fulfill this challenging requirement.

A small but high speed specialised memory co-processors can be used to store frequently accessed data. This will increase overall performance of the NP besides ensuring short and predictable data access latency. The memory management of co-processors such as allocation, de-allocation, initialization, insertion, deletion and modification are performed by hardware. The most time consuming activity of a processor is related to memory transfer, read or write. PEs can assign these tasks to co-processors and switch to other useful tasks using hardware multi-threading feature. This is generally known as latency hiding.

3.2 Queue Management and Switching

Queue control and management of network equipments such as routers and switches are a continuous challenge. Gigabit and Terabit per second equipments have multiple ports and multistage fabrics which require tens and hundreds of switching chips to manage input and output network packet queues. NPs have multiple ports that increase the number of packets received and processed from the network. Higher layer packets such as TCP require packet order management which is time and resource consuming. TCP also requires re-transmission of lost packets before it can be re-ordered, creating an oscillating condition. Order management requires the packets to be ordered before being processed and transmitted at output queue. Protocols such as ATM, maintains cell sequences making packet processing much simpler compared to IP. This raises two questions. How parallelism can be implemented with this requirement? How queue management can be improved? Generally, queue management is done at input queue, output queue or combination of both. Many studies have been conducted in queue control and management for high-speed equipments [32] which are generally used in NPs.

NPs will have packets arriving at the input interface that need to be classified and queued before being further processed. NP implementations adopt hardware and/or software support for classification and queue management for guaranteed performance. Queue management can be done either by flow or by aggregation. It is necessary to ensure efficient queue management mechanisms are in place for packet processing rate control, fairness control and congestion avoidance [26].

3.3 Packet Processing Functions

Network processing functions can be classified based on the operation types that are being performed on the packets. They can be divided as data plane functions or control plane functions. The data plane functions categorisation are as follows:

i) Classification

This is the first and most common function performed on the packets. Based on certain field or fields of the packet and rules, packet will be classified for the necessary processing action. For example, in IP packet, the source and

destination addresses are examined together with other fields before forwarding is done. Routing table entries within the NP memory is used as a rule to decide the next hop for the packet. In Asynchronous Transfer Mode (ATM), cells are checked for the header field information such as VPI and VCI against the look-up table before deciding which interface the cell is to be switched to. Content inspection is one form of classification function which can be performed from Layer 2 until Layer 7 of Internet Model [23]. However, as the depth of classification increases the overall performance of NP will decrease due to the dramatic workload increase.

ii) Transformation
The PDU content will be either replaced, inserted, deleted or modified before retransmitted. Operations such as protocol translation, segmentation & reassembly and PDU encapsulation & decapsulation are few examples of these.

iii) Stream Processing
This function includes receiving the packets and storing it temporarily for traffic shaping the stream to meet certain criteria. Traffic shaping, marking and metering are done to ensure QoS (Quality of Service).

iv) Queuing and Manipulation
Received packets need to be queued in different queues for further processing. During this processing stage, packet may be dropped in order to satisfy certain rules. For example, packet filtering based on header information requires foreign packets to be dropped.

v) Encryption & Decryption
Awareness in packet level security (IPSec), requires each packet to be encrypted. Complexity of this activity requires hardwired solution for this specific function to ensure wire speed packet processing.

vi) Error Detection & Correction
Errors in the packets need to be identified and corrected. For example, CRC (Cyclic Redundancy Check) has to be performed for each packet if any content in the header is changed.

vii) Compression
Bandwidth conscious protocols such as wireless protocol (PDCP in 3G stacks) require header compression to be performed at the routers to optimise the bandwidth usage.

Control and Management Plane functions can be categorised as follows:
i) Error and Informational Message
Packets that carry information or error messages that need to be processed by the node will be processed by the GPM (For example, ICMP error messages).

ii) Control and Management Message
Control and management packets like OAM cells in ATM will also be processed by the control plane using GPM, since they do not require fast packet processing capability.

iii) Statistics & Monitoring
Packets statistics and monitoring such as dropped, lost, error and processed packet will be maintained by the control plane.

Despite having a tighter timing window to process a single packet as shown in Table 1, NP has to perform the above functions for each packet. For example, secure IPv6 supported application in 3G communications requires almost all the functions to be performed in the underlying router. The additional burden of supporting multi-protocols on the same NP poses a greater challenge to NP in packet processing. Thus, in programmable NP, besides having high frequency processors, packet centric programming technique, optimised compiler, functions partitioning, processing techniques, optimal resource utilisation and optimised algorithms are important to ensure wire-speed packet processing.

3.4 Instruction Set, Programming Language and Compilers

NP requires efficient bit-stream-oriented programming language and instruction sets to analyse, modify and move data within given nanoseconds between many memory regions to meet the line-rate performance [25]. The challenge is to have a language with a short list of instruction sets which is general enough to code any communication protocols, algorithms and problems. A common programmer with a short learning and

implementing cycle time must also easily understand the language. A good compiler will be able to produce a small amount of assembly codes without compromising the execution quality. NP products have a high pressure to have short time-to-market thus reducing development cycle time. At the same time, short time-in-market need be maintained for monetary returns. Usually, the development stage does not exceed more than 6 months for a specific NP product, which has substantial packet processing capabilities and features.

Simplified C and microcode [5] or picocode [7] are famous approaches taken by many NP vendors to ensure their success in market penetration. The programming language and reduced instruction sets are very specific to their product and close to the machine instruction codes such as assembly. This implies that the programmer has to be familiar with the hardware before doing the programming. Some NPs can be programmed with higher-level languages such as C language to reduce the effort in picking-up a machine specific language. This naturally produces more overhead while compiling and results in larger binary codes. Application specific [21] or hardware specific [16, 17] instruction sets are not viable solutions for NPs. This is because NP is a programmable unit. Furthermore, the rapid and constant evolving standards and protocols in the telecommunication field requires a programmable solution. Applications that are being supported by NP will change from one network sector to another while the hardware co-processors will change from one NP family to another.

Instruction set or programming languages development process should consider and exploit the generic packet features such as small in size, bit and bytes in length, vary in size, with two portions of data field header and payload, easily modifiable, comparable and requires simple arithmetic operations. Fig. 4 shows the packet header or payload that does not fit the width of the memory of register. Thus, operation onto the data becomes difficult and requires more cycles if it is not aligned.

Packets which arrive at different rates and sizes need to be moved from the receive buffer to long-term memory. Additional header may be required to mark and move the packet from buffer as described in the beginning of Section 3.0. Different sizes of packets need to be moved regardless of the bus width size to different memory regions. For example, Fig. 4 shows general memory format which can be SRAM, SDRAM, registers or cache with different widths. Different memory types require different access methods, widths and cycles. Moving reassembled data from long-term memory to other functional components such as decryption or encryption units for further packet processing such as look-up, swapping, comparing, transforming and substitution requires different instruction sets. Finally, the processed packets may be transmitted to different outgoing ports or even different protocols that have different packet formats. These demanding tasks need to be performed by the instruction sets.

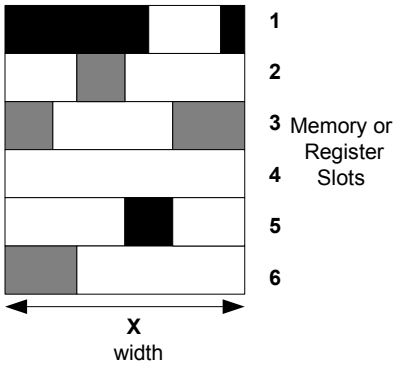

Fig. 4: Variable size packet or data organisation in memories and registers

Having an efficient compiler is as important as having effective NP instruction sets and programming languages. Coded instructions need to be translated to machine understandable codes that are small and non-redundant. Compilers have a front-end portion which does source code analysis, generation of intermediate codes, optimisation and garbage collection, while backend does the intermediate code to actual NP specific assembly code mapping. The backend function is important since it will decide the final code creation, register usage, register associated modes and memory allocation for the code. Many researchers are making an effort [18, 20] to develop good compilers and instruction sets that can produce hand-written quality of assembly code for any given high-level language. This will always be the constant ultimate challenge for NP. Some of researchers have implemented these tools on their NPs and proven its efficiency [4, 6, 15, 19]. Moving away from error prone and time-consuming assembly programming is extremely important for reducing the development cycle.

3.5 Software Architecture and Programming Tools

NP software architecture as illustrated in Fig. 5, is loosely integrated and specific to different types of processors that are being supported. Adopting Fig. 1, NP has Data Plane Processor (DPP) and CPP/GPP with different software architectures. DPP has its own instruction sets, which are machine specific and tightly coupled with it's own system architecture. Compiler, linker and assemblers are provided to construct machine codes as programmed by programmers using the instruction sets and DPP programming languages. These machine codes will be used by the PEs in the NPs for code executions. DPP protocol stack such as AAL2, AAL5, IP etc. for NPs need to be built using DPP programming languages. Various applications can be built using these protocol stacks.

CPP runs on RTOS (Real Time Operating Systems) such as Linux [33] or VxWorks [34]. If there are any GPP present in NP, then they will be using a normal operating system for less critical packet processing functions. CP protocol stacks such as SCCP, SCTP, SIP etc. are developed for a specific network application using higher generation languages such as C and C++. Third party API (Application Programming Interfaces) and NP Services, which are supported by RTOS on CP, can be used to speed-up network applications development. The CP and DP will communicate via API. Simulators and debuggers are important to the developers especially for DP programming which need to be verified for hardware mapping of the codes. Simulators allow developers to develop various network applications with different protocol stacks and test them on the NP. Performance estimation and bottleneck of a specific design can be done using simulator. On the other hand, debuggers allow developers to understand, rectify and visualise errors in network protocol design with good GUI tools. These supporting tools are essential for short software development cycle.

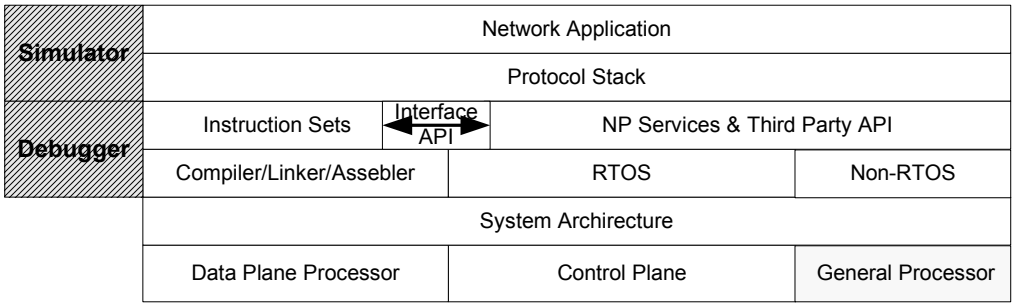

Fig. 5: General NP Software Architecture

FlexWare [4] is an example of industry standard embedded software development architecture which has the explained tools for NP specific application development. Many other NP vendors have similar software architecture in general and provide the above-mentioned tools for easy programming [12].

3.6 Processing Elements (PE)

PEs as shown in Fig. 5, are being used in many commercial products with different name references such as microengine [5], processing element and etc. They are the heart of programmable NPs and are found in variable numbers. Generally, they are specialised RISC with dedicated fast access local memory, interface to external SRAM, DRAM memories, ALU (Arithmetic Logic Unit), executable code store, co-processors and hardware multithreading support.

As packets arrive at incoming memory, they will be stored before being processed by the PE. PE will execute instructions based on the code, which is stored in the Code Store. If required, co-processor support for example encryption/decryption, CRC check functions are locally available to increase the processing efficiency based on the PE design. Local high-speed cache is available to store intermediate data or information. Data from external memory will be accessed via External Memory Interfaces and stored in local cache for further usage. For example, if table look-up is needed and the table is stored in external memory, the data will be fetched and stored in the local cache and replaced using techniques such as LRU according to the requirement. Access to other external co-processors such as a hashing unit is available via External Interfaces at PE. ALU performs the usual fetch, decode and execute of the operation using op-codes from the code store. Finally, the processed packets are stored in outgoing memory to be queued, further processed if necessary and transmitted. Fig. 6 shows an example of PE with the functional units.

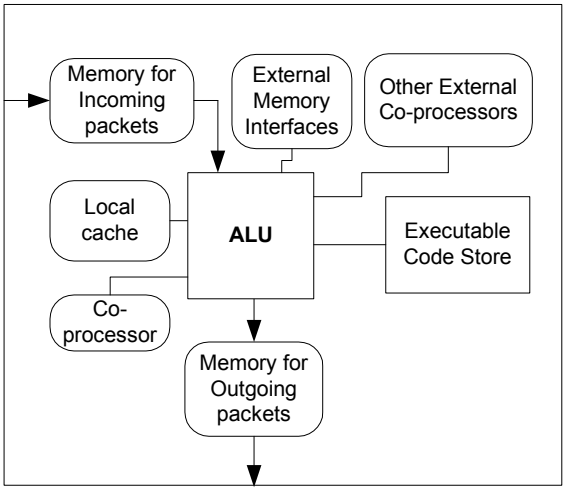

Fig. 6: General PE Architecture

Being the core functional unit in NP, PE comes in different processing speeds, ranging from 166MHz to 1.4GHz at today's NPs [5, 6, 7, 8, 9, 10, 11, 12, 13, 21]. The organisation of PEs in NPs varies from vendor to vendor. Usually, PE organisation is specific to their NP design with one of the following modes independent, clusters or scattered. High-speed data and control bus is necessary to handle the vast data and to control packet transfer within PE, with external component to PE and between PEs. The numbers of PEs in NPs vary from vendor to vendor and range from a single PE up to tens of them [5, 6, 7, 8, 9, 10, 11, 12, 13, 21].

Multithreading is local to a PE and further discussed in Section 4.10. This allows parallelism and pipelining for packet level processing besides latency hiding for memory accesses.

3.7 Parallelisation and Pipelining

Adding multiple NPs is becoming a common trend in high-end network products to meet the ever-increasing demand for high-speed packet processing at wire rate such as OC-48 and OC-192 [26]. This will be a continuing trend unless there is a drastic change in network product design. Parallelisation can take place at micro or macro level at NPs. For a particular NP, having multiple PEs and multiple processing units (for example Lexra [6] and Intel IXP2800 [5] have 16 PE in a single NP) is a macro level parallelisation. While multiple threads within in a PE is a micro level approach (for example there are 8 threads in each Intel IXP2800 and Lexra NPs). Some NPs also allow the packets to be processed by different processors, for example Control Plane Processor for control packets, CISC Processor for statistics data and NP itself for data plane packets.

Network packets are smaller with different processing requirements. The 'n' streams of packet data are easily received, processed, moved, stored and transmitted. On arrival, the classification step possibly can separate packets based on the type (e.g, IP, ATM) then the packets are put in a queue to be processed in parallel fashion. Multiple PE, co-processor and processing units act in either SIMD (Single Instruction Multiple Data) or MIMD (Multiple Instruction Multiple Data) to process the network packets. Generally all IP streams require look-up and forwarding which is straightforward and works in SIMD fashion. Complex processing and time-consuming task sequences such as classification, reassembly, look-up, time-stamping, metering, data transformation, assembly, etc. can be executed in MIMD fashion using multiple PEs. Fig. 7 demonstrates the SIMD and MIMD approach for single or multiple functions on various packets.

Pipelining is another approach to enhance parallelism in NP and is widely used today [12]. '***n***' number of packets can be broken in to '***m***' processing steps and processed. Deep pipelining structure can be executed by utilising multiple PEs. Each pipeline stage will execute the packet and pass them on to the next stage. At any time there will be '***n***' x '***m***' processing activity at the NPs. Pipelining can be done either in SIMD or MIMD fashion. In SIMD, every PE or thread performs the same function on different packets while in MIMD, every PE or thread performs different functions on different packets. In some cases, combination of SIMD and MIMD is possible.

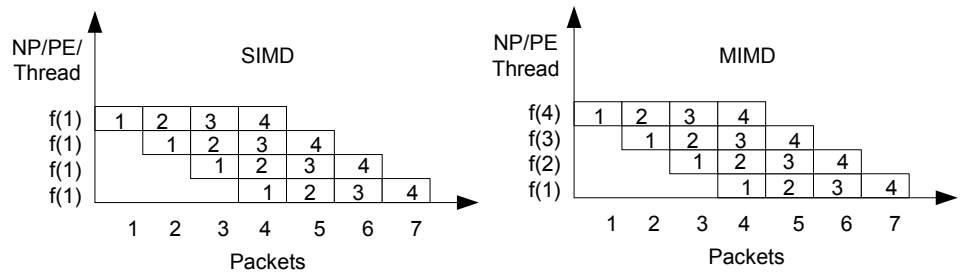

Fig. 7: SIMD and MIMD implementation either at NP, PE or Thread level for functions against packets

Having said that parallelism and pipelining helps to improve processing capabilities, there are hidden costs to its implementation too. If flow dependent packets such TCP and re-assembled RLC packets in 3G network at Node-B are being processed, pipelining will fail and packet dependency will occur. So inter PE or thread communication is important for information exchange. Shared memory is required to process these packets. Memory synchronisation then becomes an important element in packet processing and may consume valuable processing time. Inter-processor communication may also be required if tasks between NPs need to be synchronised. As such, designing and writing software for NPs becomes more difficult than for CISC computers.

The ultimate aim of parallelism is to increase the speed-up as stated by Amdahl's Law. If the number of processing elements increases by a factor **N,** then the time spent to perform the task decreases by a factor **1/N**. However, in practice very few programs achieve this level of scalability in network packet processing. The major causes for this are inter-packet dependency for processing, a need for PE/thread to PE/thread communication and an unbalanced load among PEs/threads.

However, if inter-packet dependency, memory dependency and packet processing synchronisation effort are high, then the performance will reduce thus defeating the purpose of parallelism in NP. The task decomposition must be done carefully with granularity of task in mind. Coarse task may burden a PE while fine-granule task needs more effort in the task synchronisation between multiple PE and threads. PE/thread to PE/thread communication will cause latency that needs to be taken care of. Finally, dynamic or static packet processing load balancing needs to be done among NP or PEs to ensure balanced task distribution. A study [22] shows that workload at NP is highly regular and predictable which allows for scheduling for performance improvement. This will be an interesting area of research within NP in the near future.

3.8 Multi-Threading

Multithreading is another form of task pipelining by adopting hardware methods instead of software context switching. Each PE can have many threads that do hardware context switching for any specific task that they own. For example, Intel IXP1200 [5] has 6 threads, Intel IXP2800 has 16 threads and Lexra NP has 8 threads [6]. Multithreading allows full utilisation of expensive processing elements time by switching between subtasks to hide operation latency. For example, memory access and hardware unit probing takes a longer period of time compared to any arithmetic functions. While waiting for the access response, the processing element can switch its context to another sub-task. PE controls only one thread at any given time. This allows parallel operations to be done at the same time by the independent threads associated with a PE. Context swapping and explicit pipelining allows full leverage of processing elements at all times in round-robin fashion. Some implementations give full freedom to the programmer to intercept any threads operating under the PE on a priority basis.

Context switching allows any thread to give away the control of the PE, save necessary information at that point of time and wait its turn to obtain the control back to the PE, before it can continue its next operation as specified in the code. However, PE can perform memory or hardware unit access at this point of time as explained earlier. The trade-off for this action is necessary as time has to be spent to save current context and perform task switching which is sometimes not worth the effort. Besides that, one has to bear in mind that threading may cause memory contention and can lead to a deadlock situation. So, memory locking, mutual exclusion and semaphore techniques have to be implemented to avoid deadlocks.

3.9 Function Specific or General Purpose Coprocessors

Co-processors can be considered as a type of ASIC since they perform similar functions for all packets that are being served. For example, every IP packet requires classification, table look-up and in some cases encryption/ decryption. These functions can be performed by co-processors which give a better performance. In a NP, co-processors can do these similar functions across all type of network packets. Packet classification or pattern matching, queue management, CRC, memory management, table look-up and encryption/decryption are some of the examples. Many vendors [5, 14, 15] have adopted this method to accelerate the NP performance.

Content Addressable Memory (CAM) and Ternary CAM (TCAM) are used [6, 12] to store look-up table entry, QoS determination information, multi-field forwarding look-up and MPLS related field look-up. These implementation techniques make data search, data matching, and data retrieval for CAMs relatively much faster than normal memories. Usually, carrier network units have about 1 million entries to TCAM. This ensures IP and MPLS packet look-up can be performed in a fraction of a second. Due to the high cost factor, CAMs have limited usage at NPs.

3.10 Caching

Caching is an important feature of all the NPs. Data is being cached either for control store or data store. Control store allows caching of frequently used program codes in the NP, while data store keeps frequently used network data for faster packet processing. For example, an IP address lookup program is frequently used compared to an IP route option-processing program for a core router. In this case IP address lookup program will be cached in memories such as RAM, while IP route option-processing program may be stored in SDRAM. Frequently used IP address and the lookup information will be cached in RAM for faster routing in routers. It is the NP programmers' choice to optimize the caching function for total performance improvement in a NP based equipment. Cache memory presents in most of the NP that were surveyed in this work.

4.0 NP PACKET PROCESSING PROCESS

Network communication by OSI layers conceptually allows packets to be generated by application layer passed to network layer and finally sent to the physical layer to be transported to the destination as shown in Fig. 8. During the transportation process, packets will be analysed at different layers to ensure the right destination of the packet. Electrical signals or waveforms will be converted to a series of bit streams, which we call packets or PDU (Protocol Data Unit) with header and payload information. The overall packet processing tasks can be broken to three main tasks, receiving (RX), processing (PS) and transmitting (TX) as shown in Fig. 9.

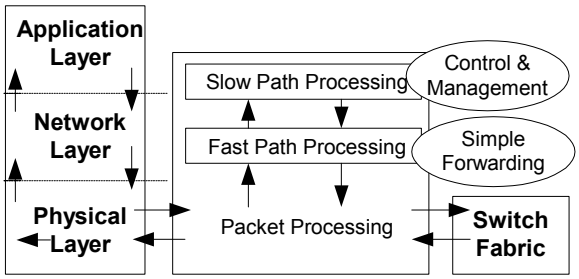

Fig. 8: Packet processing by processors at network layers

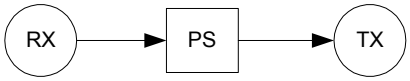

Fig. 9: Major tasks in network packet processing

Packets that arrive at the physical network interfaces can be in any form such as Asynchronous Transfer Mode (ATM), Synchronous Optical Network (SONET) or Ethernet and in any size. Additional NP specific packet header (usually Media Access Control specific to the NP) is tagged to the arrived packet. This allows easy packet identification, packet control and packet movement within the NP. Packet with variable length x (Refer Fig. 9), is fragmented to fixed size packets, usually matching the internal bus width. Usually, the initial packet has start-of-

packet (SOP) index as indicated. The last packet has end-of-packet (EOP) index in the header and if necessary, padding is done to fill the remaining space. The fragmented packet is then moved to the memory with removal of EOP and SOP index header before storage, forming the original packet. This step continues for all incoming packets as shown in Fig. 10.

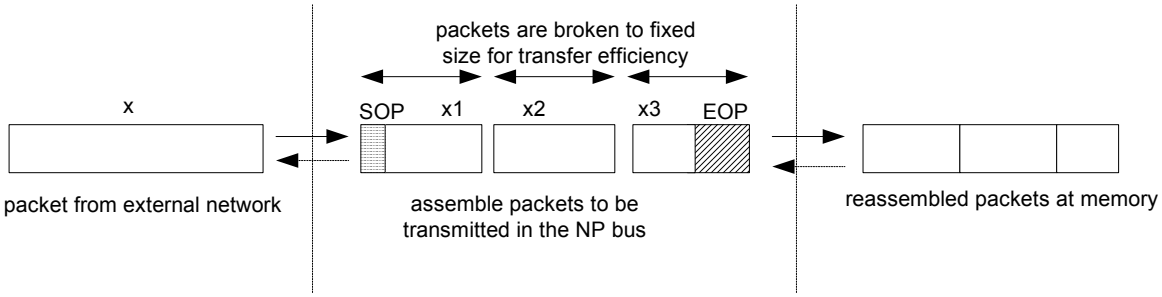

Fig. 10: Packet movement as it arrives at network physical layer, transported to memory and stored in memory

Next, the packet moves to the processing stage. The most important and time critical functions will be taking place at this stage. Firstly, the arrived packets are moved to short-term memory by the PE for buffering via the bus. Then, these packets are quickly moved to long-term memory such as SDRAM or DDRAM for further manipulation. At this point, active packet manipulation activity by the PE begins. Firstly, packet classification and reassembly will be done based on the packet identification. Necessary information or rules such as look-up table information and pattern matching information will be stored in fast access memory like SRAM for the PE to access and perform operation to the packets. Packets will be modified based on the packet type and re-written to the SDRAM memory. Various vendors differ in the implementation technique by having their own co-processor support to speed-up the packet processing at this stage. Some of the processing steps may differ in order, depending on the architecture but all of the previously mentioned steps will be covered.

Most of the NPs [5, 6, 7, 8, 9, 10, 11, 12] in the market use some hardwired co-processor support for fixed functions such as classification engine, encryption-decryption engine, look-up engine, error checking engine, statistic/ metering engine, hashing and etc. Constant functions are suitable for implementation on co-processors, which can speed up the processing. However, functions that are prone to changes, need software implementation at PEs. PEs are also responsible to coordinate all the functions by moving packets to and from the memory via the interconnecting bus. Highly accessible data will be stored in very fast access memories such as flash, local cache, CAM or TCAM.

Finally, the processed packet will be transmitted out. PEs will check for processed data availability in SDRAM space and also transmit queue readiness. If data is available and transmit queue is ready to transmit the packet, the packet will be assembled and moved to TX stage. Normally, data will be held in short term memory before being transmitted to ensure that continuity and wire-speed packet transmission is maintained at all times.

Commercial NPs have similar packet processing functions for RX, PS and TX depending on the packet processing method, PP or DP. Intel IXP1200 [5] processor is an example of DP, for packet classification, and buffering/storing operations being performed before packets are passed to the PE for further processing. Generally, other processors from Lexra [6], IBM [7], MMC Networks [8] and Agere [10] perform similar functions but differ in implementation techniques. For example, Agere's NP performs these operations by using a functional processor called Fast Pattern Processor (FPP) to get the arrived data, perform packet recognition, classification and reassembly. Then the packet is passed to Routing Switch Processor (RSP) which handles queuing, packet modification, traffic shaping, QoS and segmentation. Finally, packets are transmitted out from the outgoing buffer. Agere's System Interface manages the FPP and RSP communication and packet management.

Meanwhile, Motorola C-5e [14] that adopts PP approach has channel specific programmable processors called Channel Processors (CP). These processors are dedicated for receiving, processing and transmitting specific type of packets at that channel. Executive Processor (XP) provides network control and management functions to user applications. It also has a few other programmable processors to perform packet management, control and other supporting functions. High-speed interconnecting buses are used to connect the processors based on the functional requirements.

NPs from different vendors vary in processing method. Generally one or both of the processing paths as shown in Fig. 8 will process network packets in any NP. Fast Path is normally known as Data Plane while Slow Path is called Control Plane. Fast Path is handled by NPs while a CPP handles Slow Path. Fast path has the responsibility to analyze all arriving packets and perform forwarding at line rate for minimal service required packets. It pushes packets that need complex processing to slow path. Fast Path functions include look-up base on the packet header information and table entries in NP, packet classification etc. On the other hand, Slow Path handles small amounts of control and management type of packets that require more complex processing such as network management, error messages and configuration information. If management plane is present, the GPP will perform packet statistics related functions.

5.0 SIMPLE LOOK-UP IMPLEMENTATION

We have implemented a simple table look-up problem in CISC (Intel® Pentium® III) and NP (IXP1200) for comparison of performance using different programming languages for various combinations.

Table 2: Average time indicate the time unit per lookup in US for 1 million look-up iteration with 512 IP address entries on various techniques for PIII and IXP1200. (X- indicates no test value)

Look-up Technique\ Prog. Language	C++ (Pentium III)	Java (Pentium III)	Perl (Pentium III)	Work Bench (IXP1200)
Direct Table	0.029	0.031	X	X
Hash Table	X	0.48	9.0	X
Trie-based Table	1.41	X	X	0.71

Table 2 shows the performance impact of different languages for simple 512 entries IP address look-up function using different methods on different processors. We did not implement all the look-up techniques using different languages for different processors, since our aim is to just high-light that programming language selection for different compilers and processors has a major impact on the overall performance packet/task processing. In this case, IXP1200 needs half of the GP time to perform the similar look-up. NPs are well designed for network related problems with suitable programming language and compilers.

6.0 BENCHMARKING NPS

A benchmark [31] or performance measurement model is a program or technique which is used to measure a performance characteristic of a computer systems/applications. It often measures only one characteristic such as floating-point speed, I/O speed, speedup, latency or throughput. These characteristics will be used as standard data for comparison of different applications. The correct technique to measure this metrics or characteristics also plays an important role to determine the accuracy of the final results. Benchmarking NP performance is a challenging task since NPs differ in their architecture. NPs may have multiprocessors, multiple PEs, different number of threads and various co-processor support.

However, establishing a single set of metrics to measure the NPs performance is important for customers to select an appropriate NP to suit their products. Usually, benchmark and simulation results have a major impact on the product design and vendor selection. Benchmarking tools such as CommBench [28], MediaBench [29], NetBench [30], and other NP/vendor specific performance measurement tools [5, 7, 8, 14] are well studied and available for use.

CommBench [28] characterises the network application using four different packet headers and payload processing applications. On the other hand, NetBench [30] uses various NP type applications, which can be categorised as micro-level, IP-level and application level. These categories of application are executed and compared based on standard metrics for benchmarking purposes. MediaBench [29] was designed for multimedia and communication systems, which are in many ways similar to network processors.

7.0 STANDARDISATION EFFORT

There are dozens of NP products in the market and there will be many more to come. However, interoperability between these NPs is almost impossible, as they do not share the same standards. Due to this difficulty, Common Programming Interface Forum (CPIX) and Common Switch Interface Consortium (CSIX) were formed to coordinate the effort. Later, these organisations were replaced by NPF (Network Processing Forum) [27]. The forum consists of Hardware Working Group, Software Working Group and Benchmarking Working Group whose aim is to introduce standards among NP vendors so that interoperability is possible and to reduce design complexity and shorten the time-to-market. This allows design and development of a NP using multiple components from various vendors.

8.0 CONCLUSION

NP differs from other processors due to its dedicated function of network packet processing. Many components form the building block of the NP. These components play significant roles in influencing the overall performance, protocol implementation techniques, specific features implementation methods, costs and development time. The physical component has a fixed impact on the overall performance and can be predicted and calculated such as the PEs, multi threads, co-processors and memory units. However, the programmable feature of the NP gives enormous room for further research. Queue organisation and management is another large area that requires in-depth study for NP which handles various types of packets. Packet processing functions themselves can be further improved by utilising the parallelisation, pipelining and multithreading feature in a NP.

Programmable NP needs efficient packet centric programming techniques, optimised compilers, even functions partitioning, even workload distribution among thread/micro engines, effective processing techniques, optimal resource utilisation and optimised algorithms implementation. If any of the above features are not designed and implemented cautiously, degradation of the performance occurs. 'Best effort' performance tuning is also important after the software implementation, to achieve line rate or near line rate performance. The introduction of high frequency network processors and more numbers of threads or micro engines will ensure that the packet processing speed is achieved. However, we must remember that the packet processing complexity as well as the need to support multi protocol on the same NP is increasing. This will be a continuing challenge for NP designers and software engineers.

NP allows easy implementation of protocols. It is the key to solve upper layer issues in the programmable way at the network processing nodes, either by adopting the fast path or slow path approach. The purpose of this research is to adopt one of the network layer problems such as look-up and implement it in a NP for better performance and optimised techniques.

It will be an interesting task to further explore usage of NP in areas such as mobile computing, active networks, GRID computing, Bio-Informatics, etc. The enormous computing power provided by the NP can be used as an integrated computing resource by building a GRID network. Besides, the processing capability of these NPs is also possible for exploitation in active packet processing. Bio-Informatics is a new area of research, where large amount of data related to biological structure need to be processed. Large integrated distributed network can be built using the NPs to support the computing resource demanding tasks. Deployment of mobile computing throughout the globe requires huge networked processing resources. NP also can play a vital role in this area.

REFERENCES

[1] K.Ettikan and Rosni Abdullah, "High Speed Packet Processing Challenges for Network Processors", *Proceedings, M2USIC (MMU International Symposium on Information and Communications Technologies)* 2003, 2-3 October, Malaysia, 2003.

[2] T. Wolf, J. S. Turner, "Design Issues for High-Performance Active Routers", *Selected Areas in Communications, IEEE Journal*, Volume: 19 Issue: 3, March 2001, pp. 404-409.

[3] F. Karim, A. Nguyen, S. Dev, R. Rao, "On-Chip Communication Architecture for OC-768 Network Processors". *Design Automation Conference Proceedings*, 2001, pp. 678-683.

[4] *P. G. Paulin, F. Karim, P. Bromley,* "Network Processor: A Perspective on Market Requirements, Processor Architecture and Embedded S/W Tools". *Design, Automation and Test in Europe, 2001, Conference and Exhibition Proceedings*, 2001, pp. 420-427.

[5] Intel, Networking & Communications Building Blocks, Network Processors, *http://developer.intel.com/design/network/products/npfamily/index.htm*, June, 2002.

[6] Paul Alexander, Application of Programmable Multithreaded Network Processor Architecture to OC-192 Packet Forwarding, *http://www.lexra.com*, June 2002.

[7] IBM, Networking Technologies, *http://www-3.ibm.com/chips/products/wired/*, June 2002.

[8] MMC Networks, Revolutionizing the Way Internet Infrastructure is Built, *http://www.mmcnet.com/Solutions/*, June 2002.

[9] Bay Microsystems, Montego, *http://www.baymicrosystems.com/solutions/montego.html*, June 2002.

[10] Agere Systems, The Challenge for Next Generation Network Processors, *http://www.agere.com/metro_regional_transport/network_processors.html*

[11] Broadcom Corporation, Broadband Processor Product Line, *http://www.broadcom.com/broadbandproducts.html*, June 2002.

[12] Xelerated , Xelerator™ Product Family, A Complete Network Processor And Traffic Manager Offering, *http://www.xelerated.com/upload/products/Product%20Family_APR02.pdf*, June 2002.

[13] Ezchip Technologies, NP-1 10-Gigabit 7-Layer Network Processor, *http://www.ezchip.com/html/in_prod.html*, June 2002.

[14] Motorola, C-Port™ Network Processors, *http://e-www.motorola.com/webapp/sps/site/* June, 2002.

[15] Vitesse Semiconductor, IQ2000 and IQ2200 Families of Network Processors, *http://www.vitesse.com/products/*, June 2002.

[16] G. Araujo and S. Malik, "Optimal Code Generation for Embedded Memory Non-Homogeneous Register Architectures", in *Proc. 8th Int. Symp. System Synthesi*s, Sept. 1995, pp. 36-41.

[17] S. Liao, S. Devadas, K. Keutzer, S. Tjiang, and A. Wang, "Code Optimization Techniques for Embedded DSP Microprocessors," in *Proc. 32nd Design Automation Conf*., June 1995, pp. 599-604.

[18] Wagner, J.; Leupers, R., "C Compiler Design for a Network Processor". *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions*, Volume: 20 Issue: 11, Nov. 2001, pp. 1302-1308.

[19] J. Wagner and R. Leupers, C Compiler Design for Industrial Network Processor, *http://www.cis.ohio-state.edu/~gb/cis888.12g/Papers/c-compiler-design-for.pdf*, June 2002.

[20] Xiaoning Nie, Gazsi, L., Engel, F., Fettweis, G., "A New Network Processor Architecture for High-Speed Communmications". *Signal Processing Systems, SiPS 99, IEEE Workshop*, 1999, pp. 548-557.

[21] M. Stephenson, J. Babb, and S. Amarasinghe, "Bitwidth Analysis with Application to Silicon Compilation", in *Proc. ACM SIGPLAN Conf. Program Language Design and Implementatio*n, June 2000, pp. 108-120.

[22] Tilman Wolf and Mark A. Franklin, "Locality-Aware Predictive Scheduling of Network Processors". *Performance Analysis of Systems and Software, ISPASS, IEEE International Symposium*, 2001, pp. 152-159.

[23] Feliks Welfeld, "Network Processing in Content Inspection Application". *System Synthesis, The 14th International Symposium*, 2001, pp. 197-201.

[24] J. M. Rabaey, M. Potkonjak, F. Koushanfar, Suet-Fei Li; T. Tuan, "Challenges and Opportunities in Broadband and Wireless Communication Designs". *Computer Aided Design, ICCAD, IEEE/ACM International Conference*, 2000, pp. 76-82.

[25] Joseph William, "Architectures for Network Processing". *VLSI Technology, Systems, and Applications, 2001. Proceedings of Technical Papers, International Symposium*, 2001, pp. 61-64.

[26] W. Bux, W. E. Denzel, T. Engbersen, A. Herkersdorf, R. P. Luijten, "Technologies and Building Blocks for Fast Packet Forwarding". *IEEE Communications Magazine*, Volume 39 Issue 1, Jan. 2001 pp 70-77.

[27] Network Processing Forum,
http:/www.npforum.org, June 2002.

[28] T. Wolf, M. Franklin, "Commbench – A Telecommunications Benchmark for Network Processors". *Performance Analysis of Systems and Software, ISPASS, IEEE International Symposium*, 2000, pp. 154-162.

[29] C. Lee, et.al, "MediaBench: A Tool for Evaluating and Synthesizing". *Multimedia and Communications Systems, in Proc. of International Symposium on Microarchitecture, IEEE Micro-30*, 1997.

[30] G. Memik, W. H. Mangione-Smith and W. Hu, "NetBench: A Benchmarking Suite for Network Processors". *Computer Aided Design, ICCAD, IEEE/ACM International Conference 2001*, pp. 39-42.

[31] Ted G. Lewis and Hesham El-Rewini with In-Kyu Kim, *Introduction to Parallel Computing*. Prentice Hall International, 1992.

[32] M. Shreedhar and G. Varghese, "Efficient Fait Queuing Using Deficit Round Robin", *Proc. of SIGCOMM '95, Cambridge, MA,* 1995.

[33] Linux, *http://www.linux.org*, June 2002.

[34] VxWorks, *http://www.vxworks.com*, June 2002.

[35] B. Liljeqvist, "Visions and Facts – A Survey of Network Processors". *Master's Thesis, Department of Computer Science, Chalmers University of Technology*, October 2003.

[36] N. Shah, Understanding Network Processors. *Master's Thesis*. Dept. of Electrical Engineering and Computer Science, Univ. of California, Berkeley. 2001.

[37] M. Tsai, C. Kulkarni, C. Sauer, N. Shah, K. Keutzer, "A Benchmarking Methodology for Network Processors". *1st Workshop on Network Processors (NP-1), 8th Int. Symposium on High Performance Computing Architectures (HPCA-8)*, 2002.

[38] N. Shah, K. Keutzer, "Network Processors: Origin of Species", *in Proceedings of ISCIS XVII, The Seventeenth International Symposium on Computer and Information Sciences*, 2002.

[39] N. Shah, W. Plishker, K. Keutzer, "NP-Click: A Programming Model for the Intel IXP1200". 2nd *Workshop on Network Processors (NP-2), 9th Intl Symposium on High Performance Computing Architectures (HPCA-9)*, 2003.

BIOGRAPHY

K. Ettikan with M.Sc (Computer Science) and Bachelor of Computer Science, both from University of Science, Malaysia, is currently working at Network Processing Group of Intel Corporation, Penang, Malaysia as Network Software Engineer. He is responsible for design, development and testing of network protocol software for Intel families of Network Processors. He has significant development and test experience in IXP22x, IXP4xx, IXP1200 and IXP2xxx families of NP. He has carried-out research activities and implementation issues in IP (IPv6)/ATM/Ethernet related subjects with numerous publications. His area of interest/research includes IPv6, Anycast, ATM, High Speed Networks, Network Processing and Service Locating.

Rosni Abdullah has been a lecturer at the School of Computer Science, Universiti Sains Malaysia in Penang Malaysia since 1987. She received an award from the university to pursue her PhD which was completed in April 1997 in the area of Parallel Computing. She was promoted to Associate Professor in 2000. Throughout her academic career, she has held several administrative posts such as First Year Coordinator, Programme Chairman, Deputy Dean for Postgraduate Studies and Research and Acting Dean. She is currently Head of the Parallel and Distributed Processing Research Group and works actively in the area of parallel algorithms for numerical methods as well as bioinformatics.