Power Efficient Packet Buffering in a Protocol Processor

Ulf Nordqvist
Division of Computer Engineering
Department of Electrical Engineering
Linköping University, SE-581 83 Linköping, Sweden
Phone: +46-13-282903,
E-mail: ulfnor@isy.liu.se

ABSTRACT

In the emerging research area of protocol processors (PP) there exist many hardware platform proposals. One example of such a platform solution has been proposed by the author in a series of papers. The papers have mainly been focusing on datapath organization and optimization. This paper briefly introduce the proposed platform and the system integration perspective. Further this paper introduce the optimization process of the three levels of packet buffers. This optimization process is very complex and the optimal architectural solution is strongly dependent on a large number of parameters, e.g. network type and traffic, host system and physical implementation process. Using simulation of energy consumption characteristics a number of architectural conclusions have been made.

1. INTRODUCTION

Both computer and human communication networks use protocols with ever increasing demands on speed, cost, and flexibility. For network node components such as routers, switches and bridges, the performance needs have been fulfilled using Application Specific Integrated Circuits (ASIC) or Application Specific Standard Products (ASSP) since these applications traditionally have had quite moderate demands on programmability. These traditional approaches will probably continue to co-exist with more programmable solutions such as network processors (NP) in the future, due to their relatively cost-insensitive and performance demanding consumers. Having said this, it is clear that the networking industry is requesting moore programmable devices in tomorrows network.

In order to let the end-users 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 situated on the network interface card processing the lower layers in the OSI-Reference Model and a CPU-RISC based SW implementation of the upper layers. Usage of standard, general purpose CPU:s, is expensive in terms of cost, space and power due to their lack of dedicated hardware. Today it is easy to find Network Interface Card (NIC) supporting multi-gigabit networks but such bandwidth can not be utilized by the host since it requires the host to be fully loaded processing layer 3 and 4 protocols, leaving nothing for the application and system processing. The research focus has mainly been on router and switching applications so far, but in the future the terminals will also require offloading using programmable high-speed solutions.

A PP architecture intended to be used as a offloading device in a network terminal was proposed by the author in [1]. As most PP, 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 2. The platform has so far only been specified for packet reception. The remaining parts of this paper will focus on considerations and optimization of the memory architecture in the proposed PP platform. Especially the buffering of incoming packets using registers, SRAM FIFO and SRAM memories will be discussed in chapter 3 and 4.

2. PROTOCOL PROCESSOR

The proposed PP is intended for offloading of packet processing in a network terminal. The PP can handle up to layer 4 type of protocols. An overview of the system integration is illustrated by figure: 1. The PP consist of three major components. The first one is a general purpose micro controller responsible for the control intensive processing of the slow path. This type of processing tasks is common in the higher protocol layers. The second component is the programmable protocol processor (PPP) which is responsible for the high-performance acceleration of the data-intensive processing tasks. The last type of components are memories. There are three types of memories, a program memory for the control unit in the PPP, a control memory where inter-packet control information such as connection variables is stored. There is also a SRAM based packet buffer (PB), situated between the offloading PP-device and the host. The packet buffer is used for storage of incoming packets until they have finally been accepted as correct packets which should be provided to the host. If a packet buffer not is possible to accommodate on-chip the received packet data has to be stored in the hosts main memory.

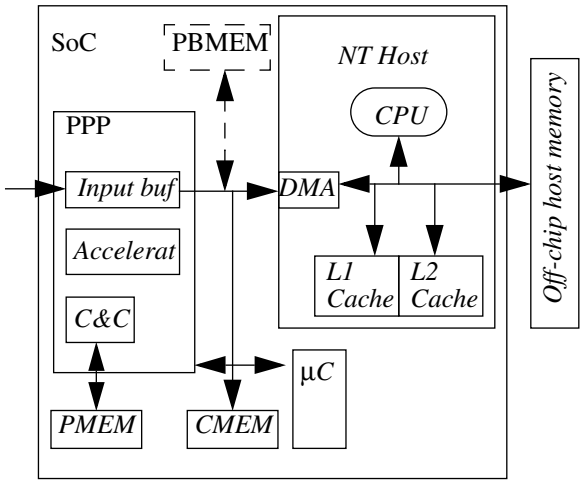

Figure 1. System and memory organization overview.

2.1. Streaming data

The PPP operates on streaming data. This means that the operations have to be exactly synchronized with the incoming 32 bit wide data stream. The main task of the protocol processor during packet reception is to decide if a packet should be discarded or accepted for further processing.

3. PACKET BUFFERING

According to figure: 1 received packets will stream through the fast path (e.i. PPP) and if the packet is accepted, the payload data will be stored in a memory. Then the application running on the host CPU can access the data for further processing. The memory where the packets are stored must be at least 1 MB in order for a TCP payload to be accommodated. There exist many different ways of organizing the memory architecture in the terminal. Some of the design choices are illustrated by figure: 2. The optimal solution depends mainly on the available chip area, e.i. on-chip memories will always be desirable if they are possible to accommodate. Minimal area will be used if the protocol processor and the host can use shared memories (figure: 2 a and b). The problem with shared memories is that the host processing (e.i. the application) will be interrupted leading to performance degradation. Remember that the purpose of the protocol processor is to offload and accelerate the application processing running on the host. In order to reduce this host OS disturbance a special on-chip packet buffer (> 1 MB) can be used for intermediate storage of the incoming packets. This is illustrated by figure: 2 c. The host memory organization is not a part of this research project and is very hard to predict, but as illustrated in figure: 2 the proposed PP platform can be integrated with a wide variety of host architectures.

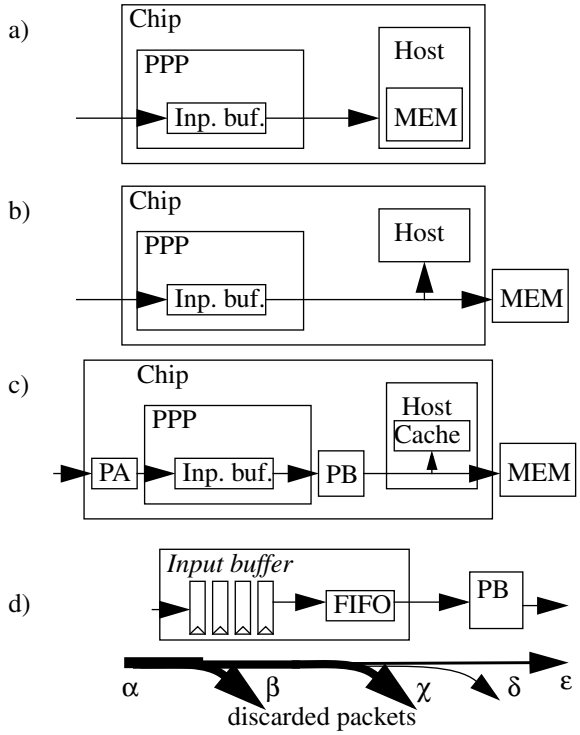

Figure 2. Input buffering design choices. a) Data is transferred directly to the host on chip memory. b) Data stored in the host-off-chip memory. c) A Phy ASIC deals with layer2 reception. A Packet Buffer SRAM stores received packets before the host access the data. d) Some packets are discarded while streaming through the 3 levels of receiving packet buffers.

3.1. Optimization

As illustrated by figure: 2 d, the received packets streaming through the input buffer into the PB SRAM will be discarded if the protocol processor detects any errors in the packet. Packets will be discarded if the address, port numbers, checksums, CRC, etc. is errornous. This is the reason why it make sense to use three different packet buffering components in the architecture.

Example: *According to figure: 2 d),* α *packets are received and streams through the registers in the input buffer. Then* β *packets will be discarded leaving* $\alpha-\beta$ *packets to be buffered in the SRAM FIFO. While streaming through the FIFO another* χ *packets will be discarded. According to this finally* $\varepsilon=\alpha-\beta-\chi-\delta$ *packets will be accepted.*

In order to reduce fan-out in the PPP and thereby enabling high speed operation, at least five 32-bit wide registers have to be used.

3.2. Optimization parameters

What are the input parameters determining the sizes of the three input buffer stages?

- ***Network traffic.*** The length, protocol type, and transmission error rate.
- ***Discard decision latency.*** Depends on type of packet, type of error, and the fast path implementation.
- ***Buffer stage (dynamic) energy cost.*** Depends on size, clock frequency, activity, implementation method and process.

4. SIMULATIONS

In order to optimize the average energy consumption in the 3 input buffer stages a number of assumptions have to be made. First of all I assume that the packets are non-fragmented and that the length and type of packets is distributed according to table 1 and 2. The decision latency for different protocols and errors is specified by the program controlling the PPP operation.

Table 1: Packet size distribution.

1-10 B	11-490 B	491-510 B	511-1500B
40%	30%	20%	10%

Table 2: Packet type distribution and discard decision latency.

Type	# of packets	Decision latency Address (Checksum)
Ethernet	104	4 (Length)
ARP	2	5 (length)
RARP	2	5 (Length)
TCP	65	10 (Length)
UDP	32	10 (Length)
IPv4	97	6 (8)
ICMP	2	6 (10)
IGMMP	1	6 (10)
Total	104	

Using MatLab a number of different simulations have been made. During the simulations the error-rates, buffer costs and buffer sizes has been used as input parameters to find the input buffer configuration that gives minimum average energy consumed per packet received.

4.1. Simulation results

As illustrated by figure: 3 the discarded packets will consume more energy if the number of register and FIFO stages are low. The reason for this is that most packets can be discarded before the data has streamed into the big packet buffer memory (1 MB SRAM). In figure: 4 we can see that the accepted packets will consume less energy per packet if the number of register and FIFO stages are reduced. The reason is that all accepted packets except the control oriented (e.g. IGMP, ICMP, ARP, RARP) will be stored in the packet buffer memory anyway. In figure: 5 the two different types of packets (discarded and accepted) has been combined and we can identify the architecture that gives a minimal energy consumption. Using assumptions on the error rates in a normal network extracted from a network statistics [2] and information on energy consumption of different FIFO, registers and memories [3-8], the resulting optimal solution with minimal energy consumed per packet is displayed in table 3 and 4. The tables state that in order to optimize the packet buffer architecture from a power perspective, we have to choose a small register based buffer, a medium sized FIFO buffer and a large PB SRAM. The energy consumed in the three buffer stages is about 40000 times the switching energy in a standard cell based flip-flop..

Table 3: Dependency on error rates in the network. (Error factor 1 corresponds to CRC error rate = 10^{-5} and TCP lost packets = 0.01 as examples)

Error factor	Register stages	FIFO Stages	Energy/ Packet [10^5FF]
10	**5**	**122**	4.03
5	**5**	**122**	4.02
1	**5**	**122**	4.02
10 (Including PHYASIC)	**5**	**126**	3.98

Table 4: Dependency on buffer implementation. (L=FIFO Length)

Register stage write cost	FIFO write cost	PB Access cost	#Reg	#FIFO
4.5	15+L	700	**5**	**122**
4.5	15+2L	700	**5**	**97**
4.5	15+2L	1500	**5**	**122**
4.5	15+L	1500	**5**	**125**

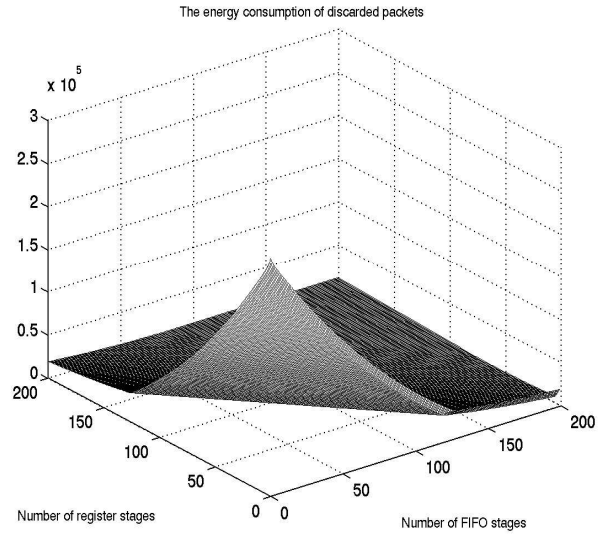

Figure 3. Average energy consumption of discarded packets.

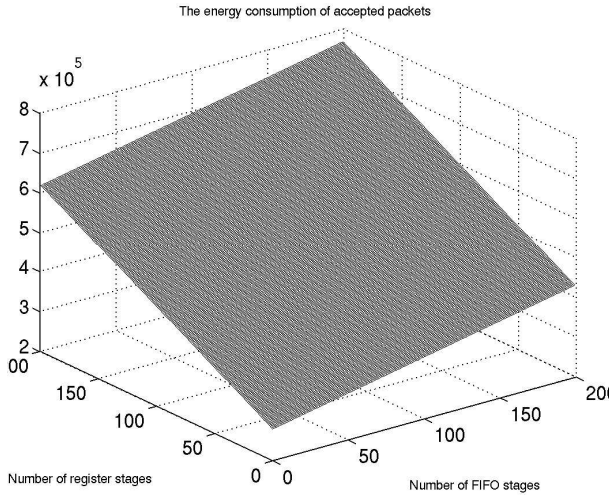

Figure 4. Average energy consumption of accepted packets.

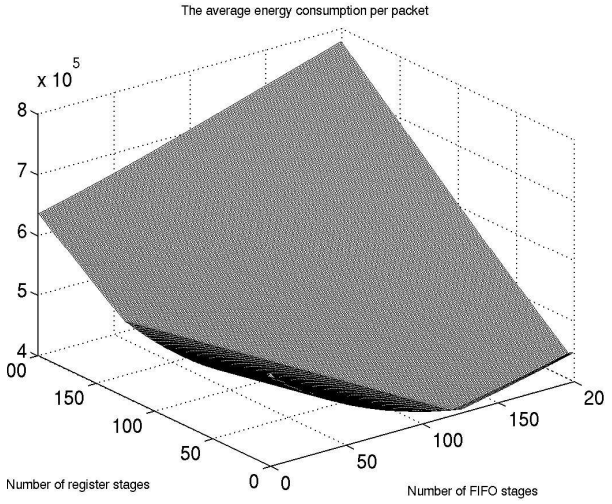

Figure 5. Average per-packet energy consumption in the three input buffers. Minima in [5, 122].

5. CONCLUSIONS AND FURTHER WORK

To conclude the different simulation results obtained we can note that the optimal packet buffer organization is to use a minimal number of registers (5), a relatively large RAM based FIFO (97-125 X 32bit). Further we can notice that the packet error rate does not affect these figures very much (< 1%). This means that the very rough assumptions made on different error rates in a typical network still are acceptable. The same thing apply to the assumptions made on energy cost.

In order to get a better understanding on the real energy cost of the two SRAM, a memory compiler and detailed power estimations or simulations are needed.

So far we have assumed that the packets are non-fragmented. If fragmented packets are allowed we can expect the optimal number of register stages to grove since the decision latency is much higher (10-40 clock cycles).

Finally the optimal solution depends on the host system memory architecture. If the PB MEM is replaced by the host memory the input buffer (registers and FIFO) should be larger.

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] *IPTraf* IP Network Monitoring Software*, on the www*, http://iptraf.seul.org/

[3] S. Barbagallo, M.Lobetti Bodoni, D.Medina, G.De Blasio, M.Ferloni and D.Sciuto, "A Parametric Design of Bui-in Self-Test FIFO Embedded Memory," *IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, pp221-229, 1996.

[4] A.R. Feldman and T Van Duzer, "Hybrid Josephson-CMOS FIFO," *IEEE Transaction on Applied Superconductivity*, Vol.5, No.2, pp2648-2651, 1995.

[5] G.N.Pham and K.C.Schmitt, "A High Throughput, Asynchronous, Dual Port FIFO Memory Implemented in ASIC Technology," *Second Annual IEEE ASIC Seminar and Exhibit*, pp, 1989.

[6] M. Hashimoto, etc., "A 20ns 256K*4 FIFO Memory," *IEEE J. of Solid State Circuits, Vol.23*, No.2, pp490-499, 1988.

[7] Austria Micro Systems, "0.35 m CMOS Libraries (C35)", *on the www*, http://asic.austriamicrosystems.com/data books/index_c35.html

[8] NEC, "Data-sheet µPD431000A", *on the www*, http://www.nec.com