Power Optimized Packet Buffering in a Protocol Processor

Ulf Nordqvist and Dake Liu
Department of Electrical Engineering
Linköping University, SE-581 83 Linköping, Sweden
Phone: +46-13-28-{2903, 1256}, Email: {ulfnor, dake}@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, mainly focusing on datapath organization and optimization. The proposed platform is unique since the fast path process incoming packets before storage in the input buffer. This paper proposes that a FIFO buffer should be added to the input buffer to lower the power consumption. The optimization process and the optimal input buffer architecture are dependent on a large number of parameters, e.g. network type and traffic, host system and physical implementation process. Simulating energy consumption characteristics, a number of architectural conclusions have been made. Especially an input packet buffer configuration is proposed which can be used in a wide variety of network applications and host systems.

1. INTRODUCTION

Both computer and human communication networks use protocols with ever increasing demands on speed, cost, and flexibility. 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, area 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 authors in [1] and [12]. 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 proposed architecture has a unique strategy for storage and wirespeed processing of incoming packet data. Instead of storing the data in a input buffer before it is processed as traditional network processing hardware, the proposed architecture manage some of the fast path processing before the packet is either discarded or stored for further processing. The protocol processor is further discussed in chapter 2. The remaining part 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 on- or off-chip SRAM memories will be discussed in chapter 3 and 4.

2. PROTOCOL PROCESSOR

An overview of the PP architecture 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. This fast path process the data on wirespeed as it streams through a chain of flip-flop based registers. 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 (=parameter) 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 (by the slow path) as correct packets which should be provided to the host. If a packet buffer can not be accommodated off- or on-chip, the received packet data has to be stored in the hosts main memory.

2.1. Streaming data

The PPP operates on streaming data. This means that the operations have to be exactly synchronized with the

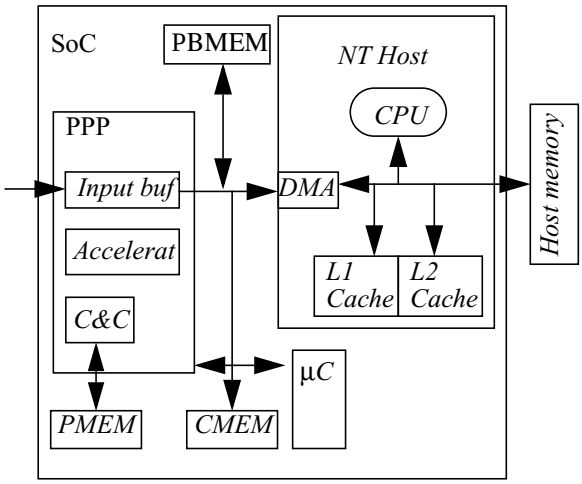

Figure 1. System and memory organization overview.

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. The fast path requires a few clock cycles in order to decide if the data should be stored or discarded, i.e. the decision latency.

3. PACKET BUFFERING

According to figure 1 received packets will stream through the fast path (i.e. PPP) and if the packet is accepted, the payload data will be stored in the hosts (main) memory, where 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 datagram to be accommodated.

There exist many different ways of organizing the memory architecture in the terminal. The main memory organization alternatives are illustrated by figure 2.

- Case a): Off chip packet buffer memory on a NIC.
- Case b): On chip packet buffer memory on a NIC.
- Case c): Large shared off-chip memory on motherboard.
- Case d): On-chip packet buffer on the motherboard CPU chip.

The optimal solution depends mainly on the available chip area, i.e. 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. The problem with shared memories is that the host processing (i.e. 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 should be used for intermediate storage of the incoming packets. The packet buffer memory (PBMEM) should be able to store at least 100 ns of traffic, i.e. 1 MB in a 10 Gb/s network. The host memory organization is not a part of this research project but

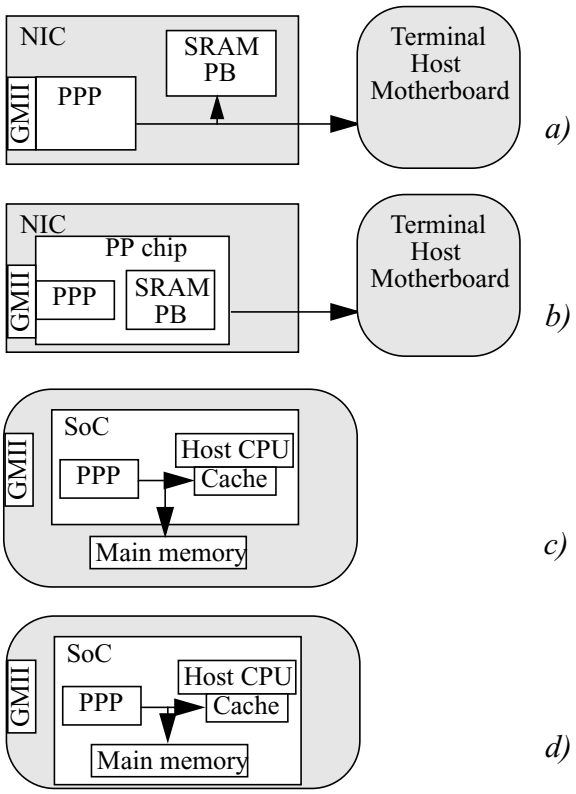

Figure 2. Packet buffering memory organization alternatives. The memory access energy cost is dependent on the selected memory organization.

as illustrated in figure 2 the proposed PP platform can be integrated with a wide variety of host architectures.

3.1. Optimization strategy

As depicted in figure 3, 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 add an extra FIFO buffer in the fast path input buffer (and use three different packet buffering components in the architecture) in order to lower the power consumption.

Example: *According to figure 3 b),* 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. This means that only* ε *packets will be stored in the large memory. Hence, to add the FIFO buffer actually lowers the energy cost.*

3.2. Optimization parameters

Since the energy cost for the memory access will be dependent on which memory organization is chosen it is natural to think that the optimal input buffer architecture

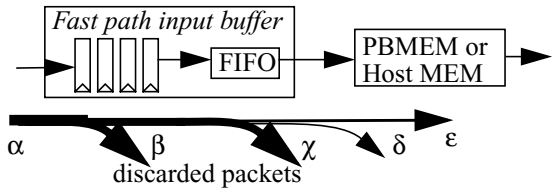

Figure 3. Packet flow during reception. Some packets are discarded while streaming through the three proposed packet buffers. These are the register chain and the FIFO in the fast path and the memory buffer. Accepted packets streams through all buffers and into the host main memory.

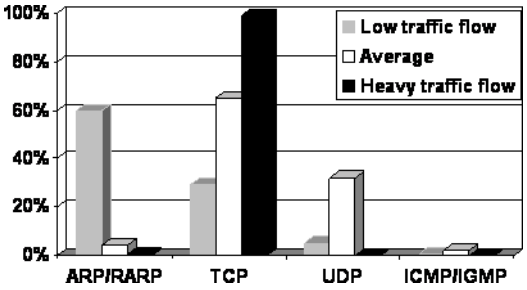

Figure 4. Packet type distribution in various traffic flows. At low traffic ARP/RARP dominates while TCP dominates during heavy traffic flows. Note that no traffic from real time applications has been modeled.

is dependent on the selected memory organization. Other parameters determining the sizes of the three input buffer stages for a certain memory system environment are:

- ***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 (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 three input buffer stages, all the parameters listed in section 3.2 must be measured as accurately as possible. The first parameter to investigate is the network traffic. We have used a network analyzing tool [11] to do this. Further we haven chosen to investigate three cases of network traffic flows. These are low traffic (no transmission, only reception), heavy traffic (file transfer application) and the average traffic flow (router traffic downscaled). The protocol type distribution for these three different traffic situations are illustrated by figure 4.

Further the length is distributed according to table 1 in the average traffic flow. The decision latency for different protocols and errors is specified by the program controlling the PPP operation. In the simulated architecture the decision latency is given by table 2.

Table 1: Packet size distribution in the average traffic flow.

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

Table 2: Discard decision latency for different protocol processing tasks.

Protocol type	Decision latency Address Check # Clock cycles	Decision latency Checksum # Clock cycles
Ethernet	4	packet length in B/4
ARP	5	packet length in B/4
RARP	5	packet length in B/4
TCP	10	packet length in B/4
UDP	10	packet length in B/4
IPv4	6	8
ICMP	6	10
IGMMP	6	10

The energy cost for access of the register chain, the SRAM based FIFO and the packet buffer SRAM memory has been modeled using cost functions mainly from [9]. In order to estimate the energy cost for access of a memory system (eq. (1)) accurately, it is essential to carefully model the effective capacitance according to eq. (2).

$$E_{MEM} = \frac{1}{2} Vdd^2 \cdot C_{eff} \quad (1)$$

$$C_{eff} = C_{onchipSRAM} + C_{offchipSRAM} + C_{interconnect} \quad (2)$$

We have assumed a 0.35 µm standard cell process for the flip-flops in the register chain and the use of a memory library optimized for low power. 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. Using MatLab a number of different simulations have been made. During the simulations the error-rates, packet buffer energy costs and traffic flows 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

Using information on energy consumption of different FIFO, registers and memories [3-10], the resulting optimal solution with minimal energy consumed per packet is displayed in table 3. Our simulations shows that the discarded packets will consume more energy if the num-

ber of register and FIFO stages are low. The reason is that most packets can be discarded before the data has streamed into the big packet buffer memory (1 MB SRAM). Meanwhile 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, which are transferred to the control memory after the FIFO buffer) will be stored in the packet buffer memory anyway. The simulations states that in order to optimize the packet buffer power consumption, we have to choose a small register based buffer, a medium sized FIFO buffer and a large PB SRAM. The average per packet energy consumed in the three buffer stages is in the magnitude of a few µJ (minimal 1.3). Further the table shows that the memory organization which decides the energy cost for a memory access has little effect on the result. Instead it is mainly the packet size and discard latency that determines the optimal buffer configuration.

Table 3: Optimal packet buffer organization for different traffic flows and memory organizations. Optimal number of register chain stages is 5.

Traffic flow type	Memory organization	# of FIFO stages	Energy savings
Low	a)	**125**	**12%**
	b)	**125**	**13%**
	c)	**125**	**13%**
	d)	**125**	**14%**
Heavy	a)	**375**	**15%**
	b)	**375**	**15%**
	c)	**375**	**15%**
	d)	**375**	**16%**
Average	a)	**256**	**24%**
	b)	**256**	**27%**
	c)	**276**	**24%**
	d)	**264**	**29%**

Table 3 also shows that the average energy consumption in the 3 buffer stages is significantly reduced compared to if no FIFO buffer would have been used. In a traditional type of input buffer the energy cost would increase instead. Simulations also indicates that the network error-rate has a low impact on the resulting buffer organization although it has a large impact on the energy savings.

From a general system power consumption perspective, it is natural to reduce the average power consumption instead of the peak power consumption. The average traffic flow in a network terminal is however hard to estimate since it is strongly depending on the applications running on the host CPU. 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).

5. CONCLUSIONS

This paper proposes that an extra FIFO is added to the input buffer in the proposed fast path architecture. This reduces the power consumption because some packets can be discarded before they are stored in a large energy consuming memory. Our simulation results indicates that the optimal input packet buffer organization is to use a minimal number of registers (5) together with a relatively large RAM based FIFO (264 X 32bit).

The very rough assumptions made on different error rates in a typical network are acceptable since they do not strongly effect the optimal buffer configuration. The same thing apply to the estimations of energy costs for access of the various buffer components.

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

[9] F. Catthoor et. al., “Custom memory management methodology”, Kluwer Academic Publishers, 1998

[10] A. Berkeman, “ASIC Implementation of a Delayless Acoustic Echo Canceller”, Ph.D. Thesis, Lund University, Sweden, Dec. 2002

[11] Ethereal Network Analyzer, *on the www*, http://www.ethereal.com/

[12] T. Henrikson, U. Nordqvist, and D. Liu, “Specification of a Configurable General-Purpose Protocol-Processor”, *Proceedings of the CSNDSP,* 2000, Bournemouth