LETTER

A Low-Power Packet Memory Architecture with a Latency-Aware Packet Mapping Method

Hyuk-Jun LEE[†], ***Member***, **Seung-Chul KIM**[†], ***Nonmember***, ***and*** **Eui-Young CHUNG**[††a)], ***Member***

SUMMARY A packet memory stores packets in internet routers and it requires typically $RTT \times C$ for the buffer space, e.g. several *GBytes*, where RTT is an average round-trip time of a TCP flow and C is the bandwidth of the router's output link. It is implemented with DRAM parts which are accessed in parallel to achieve required bandwidth. They consume significant power in a router whose scalability is heavily limited by power and heat problems. Previous work shows the packet memory size can be reduced to $\frac{RTT \times C}{\sqrt{N}}$, where N is the number of long-lived TCP flows. In this paper, we propose a novel packet memory architecture which splits the packet memory into on-chip and off-chip packet memories. We also propose a low-power packet mapping method for this architecture by estimating the latency of packets and mapping packets with small latencies to the on-chip memory. The experimental results show that our proposed architecture and mapping method reduce the dynamic power consumption of the off-chip memory by as much as 94.1% with only 50% of the packet buffer size suggested by the previous work in realistic scenarios.

key words: *router, packet memory, low power, TCP*

1. Introduction

In routers, scalability is a critical issue and usually limited by power. Packet memory accounts for a significant portion of the power consumption of a router [1]. The off-chip DRAM packet memory builds output queues where incoming packets are stored. Since the occupancy of a queue is time-varying, its storage space is dynamically allocated using a linked list data structure [2]. One linked list is built per queue. Figure 1 (A) shows a conventional packet buffer where a linked list for a queue contains two packets: $P0$ and $P1$. Whenever a packet is added to or removed from a linked list, off-chip DRAM memories are accessed and consume tens of Watts.

Previous work [3] suggests that we only need a small packet buffer as the number of TCP flows increases in core routers, which motivates us to propose a memory architecture which splits a memory into on-chip and off-chip memories for power saving. In addition, we propose a novel packet mapping method that maps packets on on-chip and off-chip memories by estimating the latency of packets through a router and utilizes the on-chip memory efficiently, which can reduce power consumption significantly.

Manuscript received August 20, 2012.
Manuscript revised November 12, 2012.

[†]The authors are with Computer Science and Engineering, Sogang University, Korea.
[††]The author is with Electrical Engineering, Yonsei University, Korea.
a) E-mail: eychung@yonsei.ac.kr

DOI: 10.1587/transinf.E96.D.963

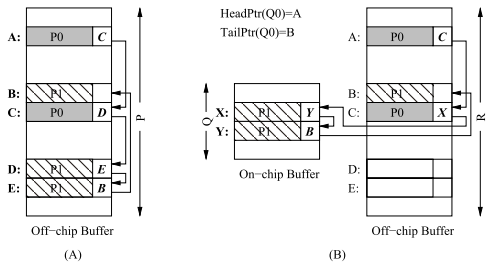

Fig. 1 Packet mapping in packet memory. (A) is a head pointer of Q0 and (B) is a tail pointer of Q0.

2. Low Power Packet Memory Architecture

Embedded DRAM makes it possible to integrate a relatively large memory on a die. Embedded DRAM takes far less power. Our scheme proposes that embedded DRAM implements the on-chip memory and both on-chip and off-chip memories are segmented in the same way as shown in Fig. 1 (B). When a packet comes into the router, memory segments for the packet are allocated from the on-chip memory if the space is available in the on-chip memory. If the occupancy of the on-chip memory becomes larger than a predefined threshold, new segments are allocated on the off-chip memory via a proposed mapping method. Figure 1 (B) shows a case where the linked list for a queue is straddled over both on-chip and off-chip memories.

The rationale behind this mapping is based on the finding in [3]. The finding was that the output link bandwidth utilization can reach 99% when the packet memory size is $\frac{RTT \times C}{\sqrt{N}}$ where N is the number of TCP flows provided that most traffic are desynchronized long-lived TCP flows. Considering modern routers support tens and hundreds of thousand queues [3], [4], the scaling by $\sqrt{N}$ could be substantial. This implies that we can reduce off-chip memory accesses significantly with a relatively small on-chip memory and save significant power consumption. In the next section, we show how to allocate space for incoming packets in either on-chip or off-chip memory using the proposed method under different scheduling schemes to maximize the power saving.

3. Latency-Aware Packet Mapping Algorithm

Although the router tries to capture most packets in the on-chip memory in the proposed method, an off-chip memory is needed to handle following overflow cases. First, the num-

ber of queues supported in routers may not be fixed due to reprogramming in the field or time-varying active queues. For a small number of queues, the buffering requirement gets bigger. Second, routers use random early detection (RED) to desynchronize TCP flows and they need a larger buffer than required to make RED work and avoid TCP synchronization due to tail drops [5]. Finally, the work [3] assumes a single output queue and simple FIFO scheduling. Real routers, however, shape the bandwidth of queues non-uniformly and employ various scheduling algorithms, e.g. priority scheduling or deficit round robin scheduling (DRR) [6]. Thus, it is very difficult to estimate the optimal on-chip memory size and misprediction could result in overflow into an external memory.

To handle the case when the on-chip memory is filled up, we propose a packet mapping algorithm in Fig. 2. There are three threshold values used in the mapping algorithm: overflow error (global), overflow warning (global), and congested (per-queue) threshold. The overflow error threshold is set to the size of on-chip memory. The overflow warning threshold is programmed to 5∼10% less than the overflow error threshold so that directing packets for the congested queues to the off-chip memory does not kick in too early. When the occupancy of the on-chip memory is between the overflow warning and error threshold, the memory space for packets arriving to *congested* queues is allocated in the off-chip memory to avoid thrashing the on-chip memory because those packets are likely to stay in the router for long time due to the congestion. The key idea in our mapping algorithm is to identify the packets that potentially have large latency through the router and place them on the off-chip memory when on-chip memory is about to being fully occupied. The latency estimation is done by comparing the depth of individual queues against their per-queue congested threshold values. The congested threshold in Eq. (2) is derived from Little's law shown in Eq. (1).

Algorithm I. Packet Mapping Algorithm
input packet i of size s for queue q
current queue depth d_q for queue q
old on-chip packet memory occupancy p_{t-1}
output new on-chip packet memory occupancy p_t

```
begin
  if p_{t-1} < overflow warning threshold
    allocate packet i on on-chip packet memory;
    p_t = p_{t-1} + s;
  else
    if p_{t-1} < overflow error threshold
      if d_q > congested threshold of queue q
        allocate packet i on off-chip packet memory;
      else
        allocate packet i on on-chip packet memory;
        p_t = p_{t-1} + s;
      endif
    else//p_{t-1} >= overflow error threshold
      allocate packet i on off-chip packet memory;
    endif
  endif
end
```

Fig. 2 Mapping packets on on-chip and off-chip packet memory.

$$L_i = \lambda_i \times W_i \tag{1}$$

$$congested\ threshold_i = \alpha_i \times \lambda_i \times target\ W_i \tag{2}$$

In Eq. (1), L_i, λ_i, W_i are average queue length, average effective arrival rate, and average queue latency for queue i respectively. The average effective arrival rate is typically given for TCP flows because the arrival rate is shaped to the allocated bandwidth. Thus, by measuring the depth of a queue, we can estimate the latency of packets. We choose *target* W_i such that it is greater than average latency of low latency queues but smaller than the high latency queues so that only packets for the low latency queues are allocated on on-chip memory dynamically. This *target* W_i can be statically determined from average queue latencies which are derived from queue bandwidth allocation and input traffic pattern. Through simulations, we measure the average queue latency and queue depth for different queues. Based on these average queue latencies, *target* W_i is chosen statically. α_i is a multiplication factor that is determined empirically from experiments so that *congested threshold*$_i$ is set higher than the average queue depth of low latency queues but lower than that of high latency queues.

4. Experimental Results

We modify ns-2 [7] to model routers employing two different scheduling policies: priority scheduling and deficit round robin. The first implements priority scheduling for low latency packets and the second implements deficit round robin for ensuring fairness. The purpose of this experiment is to validate the effectiveness of our method in two most popular scheduling policies. We simulate hundreds of different realistic scenarios using the configuration in Fig. 3. N TCP sources send traffic to a router and N sources are mapped to the output queues of the router. Scheduler implements either priority scheduling or DRR.

4.1 Priority Scheduling

In priority scheduling, queues are prioritized and scheduled accordingly. Packets belonging to the high priority queues (HPQs) are scheduled early and have very small latency through the router. The purpose of this experiment is that the proposed mapping method identify these high-priority low-latency packets and map them on the on-chip memory

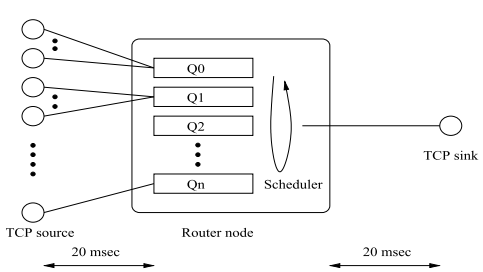

Fig. 3 Simulation configuration with RTT = 80 msec and *output link bandwidth* = 100 Mbps.

to increase the utilization of on-chip memory.

In this experiment, we use 9 queues from queue 0 to queue 8. The 9 queues represent 9 different priority levels, i.e. queue 0 represents the highest priority and queue 8 represents the lowest. Queue 0 through 7 are referred as high priority queues and queue 8 is referred as a low priority queue. We map 100 TCP flows to 9 queues and each flow is bandwidth-shaped to 2 Mbps. Since our work is the first work to propose a packet mapping method for on-chip and off-chip memory, we implement a latency-unaware packet mapping method to compare with our proposed latency-aware packet mapping method. In the latency-unaware method, packets are allocated on the off-chip memory whenever the on-chip memory is full.

We evaluate the performance of our proposed method with respect to the amount of low latency packets. For this experiment, the error threshold is set to 50 KB and the warning threshold is set to 45 KB. The error threshold is set to the half of suggested optimal packet memory size ($\frac{80\,\text{msec}\times 100\,\text{Mbps}}{\sqrt{100}} = 100\,\text{KB}$) in [3] so that (1) many packets are overflown to the off-chip memory to emulate the overflow case and (2) we show the smaller memory than [3] is sufficient thanks to the high utilization of on-chip memory by the proposed mapping method. Figure 4 shows the results of the case where we increase the number of flows per HPQ to increase low latency packets. The number of flows per HPQ is swept from two (4 Mbps per queue) to eight (16 Mbps per queue). For queue 0 through 7 (high priority queues), *target* W_i is 1 msec and α_is are 2.58, 3.44, 5.16, and 10.33 for 8, 6, 4, and 2 flows per HPQ. For queue 8 (low priority queue), *target* W_i is 1 msec and α_is are 36.57, 6.37, 1.08, and 0.60 for 8, 6, 4, and 2 flows per HPQ respectively. One interesting point is that more packets are mapped on the off-chip memory in the latency-unaware method as we increase number of flows for HPQs. This is because frequently arriving low-latency high-priority packets are pushed away to off-chip more often by low-priority packets as we increase high priority packets. When the number of flows is 8 for HPQs, 95% of total packets are allocated on the off-chip memory in the latency-unaware method (Fig. 4 (a)) whereas only 5.9% of packets are so in the proposed method (Fig. 4 (b)). This indicates that we can reduce 94.1% of power consumption for the external memory access with the proposed scheme with only 50% of the suggested packet memory size in [3].

As the bandwidth of latency sensitive traffic increases, the gain from the proposed scheme is significant. Considering more and more internet traffic become latency sensitive due to the nature of increasing real-time applications, this result is very promising.

4.2 Deficit Round Robin Scheduling

DRR is a modified weighted round robin scheduling algorithm that guarantees fairness among different queues by allocating weights proportional to the bandwidths of queues. A quantum, expressed in number of bytes, represents the scheduling weight and is assigned to each queue. In DRR scheduling, a large quantum is assigned to a queue to allocate large bandwidth or reduce the latency of packets. The purpose of this experiment is that the proposed mapping method identifies the low-latency packets belonging to queues with a large quantum and maps them on the on-chip memory to increase the utilization of on-chip memory.

In the first experiment using DRR scheduling, 100 TCP flows are mapped to 10 different queues. The quantum for the queue 0 through 3 is set to ten times larger value than rest of queues so that packets for those queues have small latencies. In this test case, we verify the performance of the proposed method under highly skewed bandwidth allocation. That is, packets through high bandwidth queues will have low latency due to large bandwidth allocation and we try to verify if our proposed method efficiently maps low latency packets on on-chip memory. We wanted to have as many high bandwidth queues as possible. Four high bandwidth queues can have 20 Mbps each and the remaining 20 Mbps is shared among low bandwidth queues. *target* W_i is 0.01 sec. α_is for queue 0 through 3 (high bandwidth queues) are 2.2

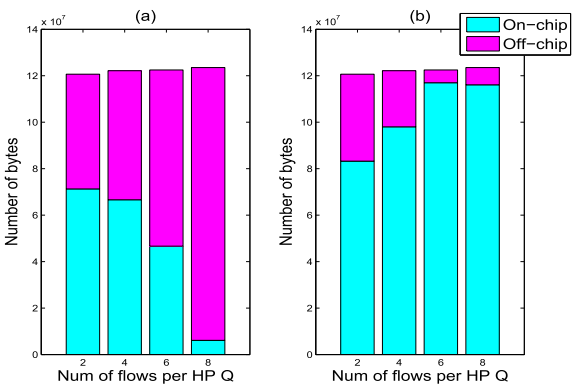

Fig. 4 Sweeping the number of flows per HPQ from two to eight for (a) the latency-unaware packet mapping method and (b) the proposed method. (*Error_Thr* = 50 KB, *Warning_Thr* = 45 KB, *Congested_Thr* = 5 KB)

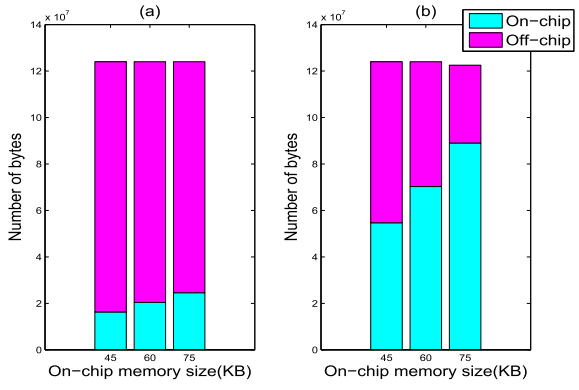

Fig. 5 Sweeping the on-chip memory size in deficit round robin scheduling for (a) the latency-unaware packet mapping method and (b) the proposed method. Four queues have 10 times larger quantum than others. (*Congested_Thr* = 45 KB)

whereas α_is for queue 4 through 9 are 7.8. The result for the first experiment is shown in Fig. 5. The proposed method in Fig. 5 (b) maps 3.35∼3.62 times more bytes on the on-chip memory compared with the latency-unaware method in Fig. 5 (a). With the proposed mapping method, 75 KB of on-chip memory captures 72.7% of total bytes and reduces the power consumption proportionally. Compared with the results from the priority scheduling shown in Sect. 4.1, on-chip memory captures more bytes as we increase the on-chip memory size from 45 KB to 75 KB. In priority scheduling, 50 KB of on-chip memory already captures significant portion of packets. For this reason, we show the results for different on-chip memory sizes in priority and DRR scheduling.

In the second experiment using DRR scheduling, we choose a test case where the bandwidth for queues are more smoothly distributed (monotonically decreasing). That is, bandwidth for queue 0 is ten times that for queue 9 and bandwidth for queue 1 is nine times that for queue 9 and so on. *target* W_i is 0.05 sec. α_is for queue 0 through 9 are 0.45∼3.49 whereas $Error\_Thr$, $Warning\_Thr$, and $Congested\_Thr$ are set to 75 KB, 70 KB, and 45 KB respectively. With the latency-unaware method, 10.5% of packets are mapped on on-chip memory whereas 22.6% of packets are mapped on on-chip memory with the latency-aware method. The latency-aware method shows 2.15 times better mapping efficiency.

5. Conclusion

The proposed method significantly reduces the power consumption caused by the off-chip packet memory accesses via a small on-chip memory implemented with embedded DRAM and a novel packet mapping scheme. Our packet mapping method estimates the latency of packets based on the queue depth and maps low latency packets to the on-chip memory to increase the utilization of the on-chip memory.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (No.2011-0023798, No. 2010-0025423, and 2010-0026822) and the Sogang University Research Grant of 2011 (No.201110026).

References

[1] J. Chabarek, J. Sommers, P. Barford, C. Estan, D. Tsiang, and S. Wright, "Power awareness in network design and routing," Proc. INFOCOM, pp.457–465, Phoenix, Arizona, USA, April 2008.
[2] H.-J. Lee and E.-Y. Chung, "Scalable QoS-aware memory controller for high-bandwidth packet memory," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.16, no.3, pp.289–301, 2008.
[3] G. Appenzeller, I. Keslassy, and N. McKeown, "Sizing router buffers," Proc. SIGCOMM, pp.281–292, Portland, Oregon, USA, Aug. 2004,
[4] The Cisco QuantumFlow Processor: Cisco's Next Generation Network Processor, http://www.cisco.com/en/US/prod/collateral/routers/ps9343/solution_overview_c22-448936.pdf
[5] R. Srikant, The Mathematics of Internet Congestion Control, Birkhauser, Boston, MA, 2004.
[6] M. Shreedhar and G. Varghese, "Efficient fair queueing using deficit round robin," IEEE/ACM Trans. Netw., vol.4, no.3, pp.375–385, 1996.
[7] The Network Simulator - NS-2. http://www.isi.edu/nsnam/ns/