Designing Traffic-Aware Approach Using Packet Buffer Architecture For High-Bandwidth Routers

Rayipudi Bharat Kumar, B. Tirupathi Reddy

ABSTRACT: All packet switches contain packet buffers to hold packets during times of congestion. High-speed routers rely on well-designed packet buffers that support multiple queues, provide large capacity and short response times and suggested combined SRAM/DRAM hierarchical buffer architectures to meet these challenges. This is particularly true for a shared memory switch where the memory needs to operate at times the line rate, where is the number of ports in the system. Even input queued switches must be able to buffer packets at the rate at which they arrive. We address these issues by first designing an efficient compact buffer that reduces the SRAM size requirement by (k - 1)/k. Then, we introduce a feasible way of coordinating multiple subsystems with a load balancing algorithm that maximizes the overall system performance. Both theoretical analysis and experimental results demonstrate that our load balancing algorithm and the distributed packet buffer architecture can easily scale to meet the buffering needs of high bandwidth links and satisfy the requirements of scale and support for multiple queues.

Index Terms: Router memory, SRAM/DRAM, packet scheduling, memory buffers.

---

1. INTRODUCTION

There are certain characteristics common to packet buffers in almost all types of packet switch. First, as the line rate increases the memory bandwidth of the packet buffers must increase. For example, if a packet switches with ports buffers packets in a single shared memory, then it requires a memory bandwidth. If on the other hand, the packet switch maintains a separate packet buffer for each input, it requires a memory bandwidth. In both cases, the memory bandwidth scales linearly with the line rate. Second, interfaces with faster line rates require larger buffers. As a rule-of-thumb, packet switches employ buffers of size approximately (where is the round trip time for flows passing through the packet switch) for those occasions when the packet switch is the bottleneck for the (TCP) flows passing through it. With an Internet of approximately 0.25 seconds today, a 10Gb/s interface requires 2.5Gbits of memory. Because today this is bigger than a reasonably priced SRAM buffer packet buffers are made from more cost- effective, lower power, but slower, DRAM whenever possible. a packet buffer should be capable of sustaining continuous data streams for both ingress and egress. With the ever-increasing line rate, current available memory technologies, namely SRAM or DRAM alone cannot simultaneously satisfy these three requirements. This prompted researchers to suggest hybrid SRAM/ DRAM (HSD) architecture with a single DRAM, interleaved DRAMs or parallel DRAMs sandwiched between SRAMs.

- The author 1 are with the Department of Computer Science and Engineering. MTech at Lakireddy Bali Reddy College of Engineering. Jawaharlal Nehru Technological University Kakinada, Andhrapradesh, India. E-mail: bharatrayipudi@gmail.com
- The author 2 are with the Department of Computer Science and Engineering, Assistent Professior at Lakireddy Bali Reddy College of Engineering. Jawaharlal Nehru Technological University Kakinada, Andhrapradesh, India.
  E-mail: tirupathireddyb@gmail.com

2. SRAM and DRAM Technology

Current SRAM and DRAM cannot individually meet the access time and capacity requirements of router buffers. While SRAM is fast enough with an access time of around 2.5 ns, its largest size is limited by current technologies to only a few MB. On the other hand, a DRAM can be built with large capacity, but the typical memory access time (i.e., T is too large, around 40 ns. Over the last decade, the DRAM memory access time decreases by only 10 percent every 18 months. In contrast, as the line-rate increases by 100 percent every 18 months, DRAM will fall further behind in satisfying the requirements of high-speed buffers. Given a DRAM family, in order to keep the DRAM modules busy, we need to transfer a minimum size chunk of data into it to effectively utilize the bandwidth provided by the DRAM module. Large memory access time of DRAM requires the system to read/write data from/to any memory address for at least T RC time units. According to our investigation, the current chunk size of DRAMs could range from 64 to 320 Bytes.2 However, given much higher price and smaller capacity of low latency DRAM products, nowadays, high dominates the market making the typical chunk size become 320 Bytes. access data at the granularity of a cell. This requirement however is not applicable to the DRAM. In a cell-based packet buffer where a chunk is much larger than a cell, the payload efficiency and the effective throughput of the entire system are dramatically reduced. This prompted researchers to suggest hybrid SRAM/DRAM (HSD) architecture. To conduct a quantitative analysis, a parameter called was introduced the ratio of access time between the DRAM and the SRAM. Accordingly, the access granularity of DRAM is **b** times that of the SRAM, i.e., **b** cells. This description is simple and straightforward. However, it becomes inadequate under some circumstances. First, the access time alone cannot determine the access granularity. The minimal access granularity has to be related to the other factors, such as the bandwidth, and the frequency. For example, given equals to 10, the access time of the DRAM is 10 times that of the SRAM. According to the definition of **b**, the access granularity of the DRAM should be 10 cells. However, if the bandwidth of the SRAM is twice that of the DRAM, the access granularity of the DRAM becomes five cells. The odd situation also happens when a DRAM with shorter access time and higher bandwidth is

introduced, where the access granularity of new DRAM depends on the

2.1. Packet Buffer Architectures

Bridging the speed gap between the SRAM product of both its bandwidth and the current access time. There is no guarantee that the speed mismatch is improved. Second, the definition of **b**and the DRAM becomes a major challenge. This speed mismatch does not refer to the bandwidth but concomitant access granularity. Due to the variable packet sizes that the IP protocol allows, it is common for packet processors to segment packets into fixed size cells, to make them easier to manage and switch. A common choice for the cell size is 64 Bytes because it is the first power of two larger than the size of a minimum packet (i.e., 40 Bytes). Thus, a packet buffer should be able to encounters some troubles in modeling a complicated architecture that consists of multiple SRAM and DRAM devices. It becomes meaningless to compare the access time of individual memory devices. Third, the definition of b becomes inapplicable when the allowed minimal access granularity of SRAM is less than a cell. Given the same bandwidth, as long as the SRAM still adopts the cell-based access, the access granularity of DRAM has to be less than **b** cells, even if the access time of DRAM is indeed **b** times that of the SRAM.

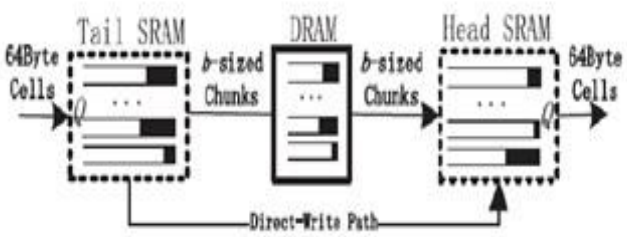

Fig. 1. Nemo Architecture

2.2. Hybrid SRAM/DRAM Architecture

First introduced the basic hybrid M/DRAM architecture with one DRAM architecture, also known as Nemo, has been adopted by Cisco.

2.3. Interleaved DRAM Architectures

Given a certain family of DRAM, the chunk size of **b** is determined by two factors. They are the bandwidth of a DRAM and its T_{RC}. Assume the bandwidth of a DRAM is BW, **b**= BW*T_{RC}. RC and cell-size, **b** can be reduced by replacing a fast DRAM with multiple slower DRAMs, i.e., smaller value of BW for each. In this way, **b** could match the size of cell, thus the batch load is no longer needed. Each DRAM is sandwiched between two smaller SRAM now Capable of accommodating cells memories, where the two SRAMs hold heads and tails of all the queues and the DRAM maintains the middle part of the queues. Shuffling packets between the SRAM and the DRAM is under the control of a memory management algorithm. The principal idea behind their MMA is to temporarily hold amount of data for each queue in both the ingress and egress SRAM, so as to change the scattered DRAM accesses into a continuous one. Since the batch loads are strictly limited within each queue, the size requirement of SRAM for the HSD architecture where Q is the number of FIFO queues. Whenever a FIFO queue accumulates b amount of data, it is transferred to the DRAM through a single write. A SRAM queue size of 2b guarantees against queue overflow. In [27], the authors further suggested that the size of the tail SRAM can be further reduced by introducing a pipeline design. They also introduced a so-called the Earliest Critical Queue First (ECQF)-MMA for the egress, which reduces the size of head SRAM. By introducing an extra delay, the ECQF- MMA now predicts the most critical queue (the one that goes empty or bears the biggest deficit first) and fetches the corresponding b-size chunk of data from the DRAM in advance. This independently. Accordingly, the MMA now needs to coordinate the data transfers between Q queues and **k** DRAMs, which can be formalized as a bipartite graph for maximum matching problem. Given the original b in a packet buffer with only one DRAM is **b** , **k** should be not less than (b1 =64 Bytes), in order to provide an equivalent overall bandwidth. Shrimali and McKeown proposed a memory architecture with (b1=64 Bytes) interleaved DRAMs. The MMA is based on a randomized algorithm, where each cell is written/ read into/from the interleaved DRAM memories randomly, thus it seriously suffers from an out-of-seq1uence problem.

2.4. Parallel DRAM Architecture

Wang et al proposed a parallel hybrid SRAM/DRAM architecture with k (k > b1/64 Bytes) DRAMs named PHSD. Compared with the interleaved architectures, the PHSD reduces the time complexity of MMA to O(k) by introducing k arbiters working in parallel. By further setting a limitation on the burst size of incoming traffic, O(kb1lnQ) size requirement of SRAM was derived. However, the size requirement of SRAM still accounts for O(kb1 Q) in the worst case.

3. DISTRIBUTED PACKET BUFFER ARCHITECTURE

All packet buffering techniques so far have adopted a traffic-agnostic approach while designing the packet buffering algorithms. We must clarify that even though existing approaches do use Q queues, each queue is treated the same by the buffer management algorithms. No effort is made to exploit the inherent characteristics of the corresponding traffic patterns like the arrival rate, burst sizes, transit time requirements through the router, etc. However, a traffic-aware approach to the problem, we believe, will yield new possibilities for conquering the scalability problem. In this paper, we investigate a new dimension to the problem, viz. how to extend the packet buffer architectures by using independent packet buffer subsystems. The overall packet buffer now takes the form of a distributed system composed of several compact packet buffers. We profess that the only real requirement for a packet buffer is that it should be able to absorb incoming traffic at a given rate, and maintain the outgoing traffic at the same rate, while still supporting the requirements for the different data streams transiting through the buffer.

4. COMPACT PACKET BUFFER DESIGN

Addressing the drawbacks of the previous algorithms, we propose a new memory architecture that reduces the system overhead in term of SRAM size while relying on a nonspecific traffic pattern.

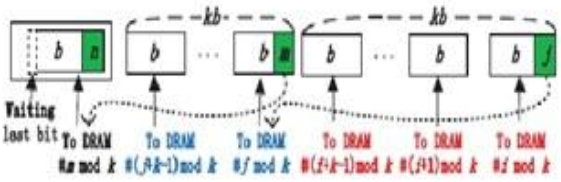

Fig. 2. Buffer behavior in an extended Nemo architecture

In above figure instead of waiting for kb- size of data before doing a batch load, per-queue RRR adopts a fast batch load scheme that dispatches b-sized chunks of data whenever it is accumulated in each queue. These b-sized chunks are dispatched to multiple DRAMs on a per-queue Round Robin basis.

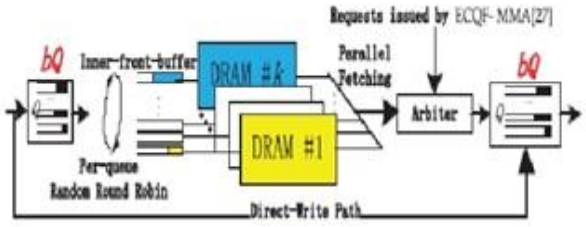

Fig. 3. The 5-stage architecture of a compact buffer.

This shows the architecture of a packet buffer where k independent address bus based DRAMs serve as the main storage sandwiched between two SRAMs. In the ingress, a ***b***Q-size SRAM is maintained as tail cache. The Random Round Robin algorithm manages to uniformly dispatch ***b***-sized chunks among the multiple DRAMs. Given the possibility that multiple chunks could be allocated to the same DRAM in a short period of time, intermediate FIFOs (i.e., inner-front-buffer) are introduced to hold these chunks temporarily so as to prevent unnecessary drops. At the egress we maintain a **b**Q-size SRAM serving as the head cache which is 1=k that of in Nemo. Whenever the scheduler requests data from a queue, its corresponding **b**-sized data (i.e., the first chunk of this queue in the order of FIFO) are fetched from the corresponding DRAM arbitrary. When multiple chunks need to be fetched out from the same DRAM (i.e., the heads of current active queues locate at the same DRAM chip), conflict happens. An arbiter is introduced to solve this problem.

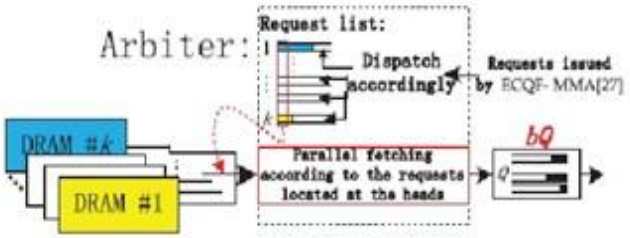

Fig.4. The Architecture of an Arbitar

The internal structure of the arbiter, for each DRAM, the arbiter maintains a separated request list and k request lists in total. Whenever a new request arrives at the arbiter, it will first be identified which specific queue it belongs to. Then, its current RRR sequence will be derived according to the log2k bits information attached to the head of every kb-sized chunk. Referring to its round robin counter, the location of its head chunk and the corresponding request list can be determined. For every round, only the oldest request of each request list can be issued to the DRAMs. Since the queuing length of the request lists could be different, the delay that a request is satisfied varies. In the next section, we will prove that such kind of request queuing delay and the queuing delay in the inner-front-buffers can be upper bounded, given satisfying certain conditions.

5. BUFFER BEHAVIOURS

When we carefully examine the hierarchical packet buffer architectures by using the aforementioned methodology, whether the HSD architecture interleaved DRAMs or parallel DRAM they all rely on three parameters, **k, b**, and Q. The required size of SRAM is always O(kbQ). To understand this phenomenon for its further study, we first examine the buffer behavior of previous hybrid SRAM/DRAM architectures and algorithms, especially in the Nemo and the PHSD architectures. DRAM structure in this extend version of Nemo is implemented as a composition of k DRAMs that simply provides a data bus of width k times that of a single DRAM data bus. Given a fixed chunk size of b for a single DRAM, Nemo increases the scale of batch load by k times, which requires each of the Q queues to maintain kb-size of data. Whenever kb-size of data is accumulated in a queue, it will be written into k DRAMs through a mutual data bus. In this way, the size gap between cell and chunk is compensated. One major drawback of Nemo is that the first (kb-1) size of data cannot depart from the queue until the last bit arrives. This increases the buffering requirement.

1. Given that only the first k(b-64) size of data arrives to a queue, the data could stay in the SRAMs infinitely. So the maximum SRAM size in PHSD is roughly equal to that of Nemo, when b is much larger than 64 Bytes.

2. The departure time of each b-sized chunk in PHSD relies on the corresponding traffic pattern. Given

that only the first k(b-64)+64 recall that the typical values of b for currently available DRAMs are around 64 to 320 Bytes) size of data arrives in each of the multiple queues, it creates an unbalanced traffic allocation among the DRAMs, i.e., only the first DRAM receives b size of data.

3. Although the long-term output is balanced by the per-queue Round Robin scheme, short-term biased output still exists, e.g., the heads of currently active queues in the output may all be allocated to a particular DRAM. Accordingly, this DRAM becomes the bottleneck of entire system in egress.

4. Per-queue Round Robin increases the scale of queuing and maximum matching problem by **k** times making the corresponding MMA less efficient.

5. The same Round Robin sequence adopted by multiple queues may create synchronization effect that overwhelms the first DRAM in the initial stage.

According to the aforementioned analysis, it seems that the per-queue Round-Robin does more harm than good, and both Nemo and PHSD fail to exploit the advantage of having parallel DRAMs. The requirement of SRAM size in the worst case is always O(kbQ).

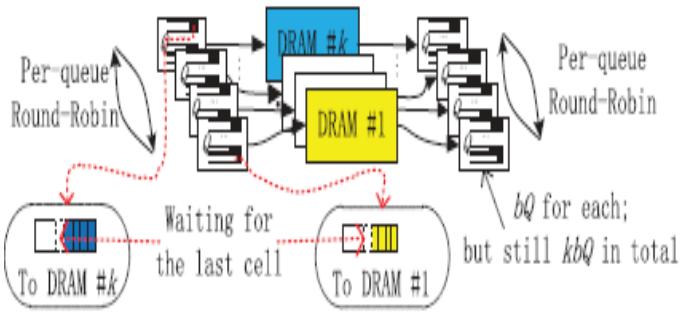

Fig. 5. Buffer behavior in PHSD

5. CONCLUSION

Building packet buffers based on a hybrid SRAM/DRAM architecture while introducing minimum overhead is the major issue discussed in this paper. To distinctly increase the throughput and storage capacity of a packet buffer, a parallel mechanism using multiple DRAM chips should be deployed. Our analysis shows that previous algorithms make very little effects in exploring the advantage of parallel DRAMs leading to the requirement of large size SRAM and high time complexity in memory management. In this paper, we present novel packet buffer architecture by using both fast batch load scheme and a hierarchal distributed structure. It reduces the requirement of SRAM size greatly. Both mathematical analysis and simulation results indicate that the proposed architecture provides guaranteed performance in terms of the low time complexity, short access delay, and upper bounded drop rate, when a small speedup is provided.

6. REFERENCES

[1]. N. Beheshti, E. Burmeister, Y. Ganjali, J. Bowers, D. Blumenthal, and N. McKeown, —Optical Packet Buffersfor Backbone Internet Routers,‖ IEEE/ACM Trans. Networking, vol. 18, no. 5, pp. 1599- 1609, Oct. 2010.

[2]. D. Lin, M. Hamdi, and J. Muppala, —Designing Packet Buffers Using Random Round Robin,‖ Proc. IEEE GlobeCom '10, pp. 1-5,Dec. 2010.

[3]. G. Shrimali and N. McKeown, —Building Packet Buffers with Interleaved Memories,‖ Proc. Workshop High Performance Switching and Routing (HPSR '05), pp. 1-5, May 2005.

[4]. D. Lin and M. Hamdi, —Two-Stage Fair Queuing Using Budget Round-Robin,‖ Proc. IEEE Int'l Conf. Comm. (ICC '10), pp. 1-5, May 2010.

[5]. M. Kabra, S. Saha, and B. Lin, —Fast Buffer Memory with Deterministic Packet Departures,‖ Proc. 14th IEEE Symp. High Performance Interconnects '06, pp. 67-72, 20.