Building network Packet Buffers in High Bandwidth Switches and Routers

V.Laxman, Ms.P.Subhadra

1 M.Tech (C.S.E), VCE, Hyderabad.
2 Associate Professor in CSE Dept, VCE, Hyderabad.

Abstract:
All packet switches contain packet buffers to hold packets during times of congestion. The capacity of a high performance router is often dictated by the speed of its packet buffers. High-speed routers rely on well-designed packet buffers that support multiple queues, provide large capacity and short response times. Some researchers 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. And so as link rates increase memory bandwidth requirements grow. With today’s DRAM technology and for an OC192c (10Gb/s) link, it is barely possible to write packets to (read packets from) memory at the rate at which they arrive (depart). As link rates increase, the problem will get harder. There are several techniques for building faster packet buffers, based on ideas from computer architecture such as memory interleaving and banking. While not directly applicable to packet switches, they form the basis of several techniques in use today. In this paper we consider one particular packet buffer architecture consisting of large, slow, low cost, DRAMs coupled with a small, fast SRAM “buffer”. We describe and analyze a memory management algorithm (ECQF-MMA) for replenishing the cache and find a bound on the size of the SRAM.

Keywords: 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 switch 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 (and because each successive generation of interface requires more memory), packet buffers are made from more cost-effective,
lower power, but slower, DRAM whenever possible.

Third, packet buffers are arranged as one or more first-in first-out (FIFO) queues. For example, a shared memory switch maintains at least one FIFO for each output line; and an input-buffered packet switch usually maintains virtual output queues. If the packet switch performs per-flow or per-class queuing, the number of FIFO queues can become very large.

Furthermore, 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.

Fourth, the sequence in which packets are read from the buffer is determined by a scheduling algorithm. For example, in a shared memory switch, the time at which each packet is read from memory is determined by an output-link scheduler (e.g. a WFQ scheduler). Similarly, in a packet switch with input buffers, the time at which each packet is read from memory is determined by an arbiter that determines the configuration of a switch fabric. In both types of switch we can think of there being an arbiter that requests a particular packet be retrieved from memory and delivered within a fixed time. Although deterministic, the sequence of requests is not known to the buffer manager at the time packets are written into memory. As far as the buffer manager is concerned, the sequence of requests is unpredictable, yet it is required to retrieve the packet within a fixed time. This is quite different behaviour from how data is read from main memory in a computer, it is considered acceptable if the time to retrieve data is variable.

In this paper, we will describe and analyze a mechanism in which memory bandwidth is increased by reading (writing) multiple cells from (to) memory in parallel (i.e. there is a single address bus that controls all memory devices in parallel). In other words, when the unpredictable arbiter requests a cell multiple cells are read from DRAM at the same time. The additional cells are stored temporarily in an SRAM until they are required. We can think of this memory hierarchy as a large DRAM containing a set of FIFOs; the head and tail of each FIFO is cached in a (possibly on-chip) SRAM. The SRAM is sized so that whenever the arbiter requests a cell, it is always delivered within a bounded delay, regardless of the sequence of requests.

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. 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.

2.2 Packet Buffer Architectures:

Bridging the speed gap between the SRAM and the DRAM becomes a major challenge.
This speed mismatch does not refer to the bandwidth but concomitant access granularities. 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 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 product of both its bandwidth and the current access time.

There is no guarantee that the speed mismatch is improved. Second, the
definition of $b$ encounters some troubles in modelling 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.

2.2.1 Hybrid SRAM/DRAM Architecture:

First introduced the basic hybrid SRAM/DRAM architecture with one
DRAM sandwiched between two smaller SRAM 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 architecture, also known as Nemo, has been adopted by Cisco.

2.2.2 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 TRC. Assume the bandwidth of a DRAM is BW, b = \( \frac{1}{4} \) BW TRC 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 now capable of accommodating cells 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-sequence problem.

2.2.3 Parallel DRAM Architecture:
Wang et al. proposed a parallel hybrid SRAM/DRAM architecture with k DRAMs named PHSD. Compared with the interleaved architectures [9], [12], [16], [17], 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, \( O(k)b_1\ln Q \) size requirement of SRAM was derived. However, the size requirement of SRAM still accounts for \( O(k)b_1 Q \) in the worst case.

3. AN MMA THAT MINIMIZES THE SIZE OF THE SRAM BUFFER:
We now describe an MMA that minimizes the size of the SRAM buffer while bounding the latency. We call it Earliest Critical Queue First MMA (ECQF-MMA).

A. ECQF-MMA

ECQF-MMA uses a lookahead buffer to hold the unpredictable stream of requests from the arbiter.

Algorithm Description: Every cell time, ECQF-MMA has the opportunity to read from DRAM and decide which, if any, FIFO queue in the egress SRAM buffer to replenish.

Example of ECQF-MMA: shows an example of ECQF-MMA with and. MMA at time computes that queues will become critical at time and respectively. Since is the
earliest critical queue, it is chosen for service from the DRAM. Cells from queues leave the egress SRAM at times.

In Figure 2b the ECQF-MMA determines that is the earliest critical queue and it is chosen for service from the DRAM. Cells from queues leave the egress SRAM at times.

We now derive the size of the egress SRAM required for ECQF-MMA. In the following section we shall make three simplifying assumptions. We shall see later how these assumptions can be relaxed.

The buffer behaviour of SRAM/DRAM architectures and algorithms, especially in the Nemo and the PHSD architectures.

The 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.

Assumption1. (Queues Initially Full) At time the egress SRAM is full and has cells in each queue i.e. the occupancy counter for each queue is b – 1 and hence the total occupancy of the SRAM is Q * b – 1.

Assumption2. (Queues Never Empty) Whenever the MMA decides to refresh a queue, there are always cells present in the DRAM for that queue.

Assumption3. (Request Every Time Slot) The arbiter issues a new request every time slot.

B. Is there always an earliest critical queue to be read?

ECQF-MMA requires that there is always an earliest critical queue to read from.

B. What affects the size of the SRAM buffer?

If the packet buffer maintained just one FIFO queue, the operation would be simple: Each time cells arrived at the ingress SRAM, they would be written into DRAM. This would require cells of storage in the ingress buffer so as to store the cells which arrived before the cell. Similarly, the egress buffers would require cells of storage. When the egress SRAM buffer receives a request from the arbiter, the MMA reads cells from DRAM, grants one cell to the arbiter and stores the remaining cells in SRAM. Every time new requests arrive from the arbiter, new cells are read from DRAM.

When there are more FIFOs the system is more complicated. For example, with Q FIFOs let’s consider what happens as cells depart from the egress SRAM buffer. When a cell departs it may trigger the need to replenish the FIFO to prevent under-run of the FIFO in the future (i.e. a request arrives to find that the cell is in DRAM). If consecutively departing cells causedifferent FIFOs to need replenishment, then a queue of read requests will form waiting for cells to be retrieved from DRAM. If a read request is queued too long, a FIFO in the egress SRAM buffer might not be replenished before it under-runs. Put another
way, the egress SRAM buffer might have to contain a very large number of cells for each FIFO so as to hold sufficient reserves to cover the worst case sequence of departures.

Fortunately, a MMA need not issue read requests to DRAM in the same order that FIFOs become depleted. If the MMA has some knowledge of how quickly a FIFO needs to be replenished, it can give priority to FIFOs with a more urgent need for replenishment. How does the MMA know how urgently a FIFO needs to be replenished? In the scheme considered here, the MMA uses a lookahead buffer of requests from the arbiter. The MMA uses the lookahead buffer to peek at which FIFOs are receiving the most requests. The lookahead buffer increases the latency from when a request is issued, until the cell is delivered. We believe that if the latency is small and bounded, it will be acceptable in most applications. In what follows, it will help if we clearly define the lookahead buffer and the latency in this context.

4. CONCLUSIONS:

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 propose (ECQF-MMA) Algorithm Memory hierarchy of packet buffer, showing large DRAM memory with heads and tails of each FIFO maintained in a smaller SRAM cache.

REFERENCES: