A Conflict-Free Memory Banking Architecture for Fast VOQ Packet Buffers.

Jorge García, Llorenç Cerdà, Jesús Corbal, and Mateo Valero
Technical University of Catalonia (UPC) - Computer Architecture Dept.
E-mail: {jorge,llorenc,jcorbal,mateo}@ac.upc.es

***Abstract*— In order to support the enormous growth of the Internet, innovative research in every router's subsystem is needed. In this paper we focus our attention on packet buffer design for routers supporting high-speed line rates. More specifically, we address the design of packet buffers using Virtual Output Queuing (VOQ) discipline, which are used in most modern router architectures. The design is based on a previously proposed scheme that uses a combination of SRAM and DRAM modules. We propose a storage scheme that achieves a conflict-free memory bank organization. This leads to a reduction of the granularity of DRAM accesses, resulting in a decrease of storage capacity needed by the SRAM. In the DRAM/SRAM scheme, SRAM memory bandwidth needs to fit the line rate. Since memory bandwidth is limited by its size, searching for memory schemes having a small SRAM size arises as an essential issue for high speed line rates (e.g. OC768, 40 Gbps and OC3072, 160 Gbps).**

I. Introduction

One problem that comes out when designing high speed routers or switches is how to build high bandwidth multi-queue packet buffers. One important example of this are *Virtual Output Queuing* (VOQ) buffers [1], used as input buffers in most of the recently proposed router architectures (e.g: [2], [3], [4]). An VOQ input buffer maintains Q separate logical queues, one for each output port and class of service. When a cell arrives to the input port, it is placed in the queue corresponding to its outgoing port and class of service. When the input port receives a request for a cell addressed to a given output port and class of service, the cell is taken from the head of the corresponding queue in the VOQ buffer.

For line rates below OC192 (10 Gbps) the main complexity of VOQ relays not on the buffer itself, but on the switch fabric scheduler, as for an N port switch it has to control $N \times Q$ queues. For OC192 and beyond, the packet buffer design becomes an interesting and challenging problem, as the required packet buffer bandwidth exceeds the capacity of commercial DRAMs.

In [5], [6] and [7] a VOQ buffer design which uses slow but low cost DRAM coupled with fast but costly SRAM is discussed (see Figure 1). The system consists of two SRAM modules (t-SRAM and h-SRAM), a DRAM system, and two Memory Management Algorithm modules (t-MMA and h-MMA). The tail of each VOQ logical queue is stored in the t-SRAM, the head is stored in the h-SRAM, and the rest is stored in DRAM. In this scheme the SRAM memory bandwidth needs to fit the line rate. This means that the SRAM access time must be less than or equal to the transmission time of a cell (we shall refer to this time as a *slot*). In order to match DRAM/SRAM access times, transfers between DRAM and SRAM occurs in batches of B cells every B slots. B is known as the *data granularity* of the memory scheme.

Thus, every B slots the t-MMA selects a queue from which B cells are to be transferred from t-SRAM to DRAM. This algorithm should guarantee that the t-SRAM is not filled up before DRAM. Otherwise losses would occur before the DRAM is full. A t-MMA that could avoid these losses is simple: transfer to DRAM B cells from any queue with an occupancy counter higher than or equal to B. In this case, the required tail SRAM size would be $Q\,(B-1)+1$ cells.

The h-SRAM is a more complex system. This algorithm has to guarantee that cells transferred between DRAM and h-SRAM can accommodate the sequence of cells requested by the fabric scheduler. Otherwise it may happen that the cell requested by the scheduler is not present in the h-SRAM because it has not been already transferred from the DRAM. We shall refer this condition as a *miss*. We will dedicate the rest of the paper to analyze the h-MMA and h-SRAM, and we shall refer to them simply as MMA and SRAM.

The analysis that can be found in literature related to this DRAM/SRAM design always consider that transfers between SRAM and DRAM are done every random access time of DRAM. We shall refer to this system as *Random Access DRAM System* (RADS). As it is shown in [5], [6] and [7], t-SRAM and h-SRAM sizes are roughly proportional to $Q \times B$ cells (see section II). Decreasing the value B would lead to smaller and hence faster SRAMs, leading to a VOQ design suitable for faster input line rates. Unfortunately, DRAM random access times decrease at a relatively low pace (around 10% every 18 months), and for decreasing the value B we cannot relay on purely technological improvements.

In this paper we propose a new design for the DRAM subsystem that uses DRAM memories with a high number of interleaving memory modules. We propose a storage scheme [8] and associated out of order access to these memory modules that guarantee conflict free access. This feature allows our design to reduce the granularity of the DRAM accesses, and consequently to reduce the size of the SRAM memory and increase the line rate. We shall refer to our memory architecture as *Conflict Free DRAM System* (CFDS).

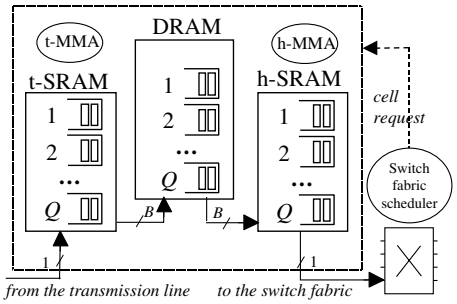

Fig. 1. RADS memory architecture of the packet buffer.

II. Random Access DRAM System (RADS)

In [5], [6] and [7] different proposals for MMAs are studied. In this section we summarize the main dimensioning results obtained in these references.

Figure 2 shows the general MMA scheme: An arbiter (e.g. the switch fabric scheduler) issues a cell request every slot. This request is stored in the tail of a *lookahead shift register* of l positions. At every slot, one cell from the queue demanded by the head of the lookahead is read from the SRAM and granted to the arbiter. In order to guarantee that the requested cell is always in SRAM, every B slots a queue is selected by the MMA and a group of B cells of this queue are transfered from DRAM to SRAM.

The decisions of the MMA take into account (i) the SRAM occupancy counters (i.e. the number of cells of every queue present in the SRAM), and (ii) the arbiter requests stored in the lookahead. The lookahead allows the MMA to select the queue to be replenished knowing the l requests that are going to be issued in the future. Armed with this information, the MMA can take better decisions and hence, the SRAM size can be reduced and still guarantee zero miss probability.

For example, suppose that the parameters of the system shown in Figure 2 are: $Q = 4$, $B = 3$, $l = 5$. Suppose also that the MMA is called with the SRAM occupancy counters and the lookahead values shown in the figure. The MMA should select the queue 1. This queue would be replenished with 3 cells after 3 slots, and would remain with 2 cells after 5 slots. If the MMA would have selected the queue 3, a miss would occur for queue 1 after 5 slots.

The MMA that allows the smallest SRAM is the so called *Earliest Critical Queue First* (ECQF-MMA). The algorithm works as follows: read the lookahead from the head (slot 1) to the tail (slot l). For every request read from the lookahead, decrease the occupancy counter of the corresponding queue. If this *modified* occupancy counter is less than zero, then the queue is said to be critical. The first queue found to be critical is the queue selected by the ECQF-MMA. The minimum SRAM size necessary to have zero miss probability is $\text{SRAM}_{min} = Q\,(B-1)$ and the required lookahead is $l = Q\,(B-1)+1$. Note that this lookahead value guarantees that there is always at least one critical queue.

If we want to reduce the lookahead value ($0 < l < Q(B-1)+1$) the *Most Deficit Queue First with Pipeline Delay* is proposed. The MDQFP-MMA choses first the earliest critical queue, if there is any. If there are not critical queues, it choses the queue with lowest occupancy counter at the end of the lookahead. The SRAM size bound for guaranteeing zero misses is: $\text{SRAM}_{max} \leq Q\,B\,(1+\ln\frac{QB}{l})$

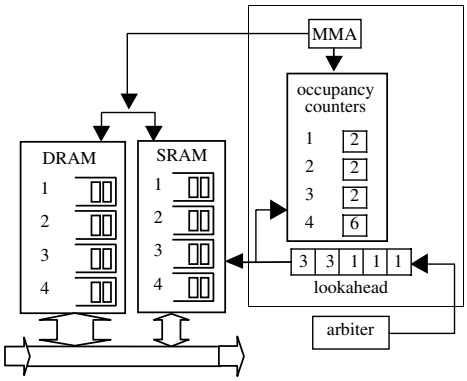

Fig. 2. RADS Memory Architecture.

In the special case of $l = 0$ the MMA consists of selecting the queue having the lower occupancy counter (*Most Deficit Queue First*), and the maximum SRAM size necessary to have zero miss probability is bounded by: $\text{SRAM}_{max} \leq Q\,B\,(2+\ln Q)$.

In [9] we have derived the algorithm in C code shown in Figure 3. This algorithm computes the exact value of the required SRAM size (in cells) for all values of the lookahead.

```
int rads_sram_size(int L, int Q, int B)
{
    int nextround, maxdeficit, slot ;

    if(L > Q * (B-1)) {
        return Q * (B - 1) ;
    }
    if(L > 0) {
        maxdeficit = int((L - 1)/ Q) + 1 ;
        slot = maxdeficit * Q ;
    } else {
        maxdeficit = 0 ;
        slot = 0 ;
    }

    while(1) {
        nextround = int((slot - L) / B) + 1 ;
        if(nextround >= Q-1) {
            maxdeficit = maxdeficit + Q * B - 1 - slot ;
            break ;
        }
        slot += Q - nextround ;
        if(slot >= Q * B) {
            break ;
        }
        ++maxdeficit ;
    }
    return Q * maxdeficit ;
}
```

Fig. 3. Algorithm to compute the SRAM size. The parameters of the function are: lookahead size (L), number of queues (Q) and DRAM granularity (B).

III. Potential of bank interleaving

In response to the growing gap between processor and memory, DRAM manufacturers have created several new architec-

tures that address the problem of latency, bandwidth and cycle time (e.g. DDR-SDRAM [10] or RAMBUS DRDRAM [11]). All these commercial DRAM solutions implement a number of memory banks –as high as 512– that can be addressed independently. If a conflict-free mechanism can be implemented to avoid memory bank contention, the cycle time of a 'random' DRAM access can be significantly reduced. These features are exploited in the memory architecture proposed in section IV. In this section the concept of *memory banking* is introduced.

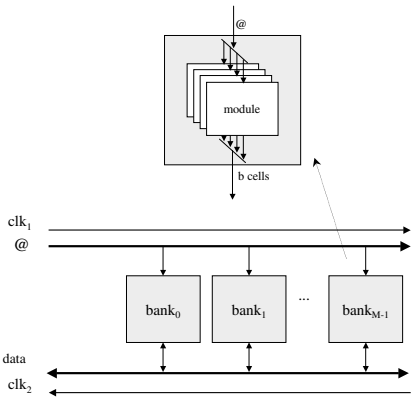

Fig. 4. Organization of a DRAM main memory in banks: (a) internal structure of a memory bank, (b) configuration of a DRDRAM-style memory.

Figure 4.a illustrates the concept of memory bank and the concept of an interleaved memory system. A memory bank is a set of memory modules that are always accessed in parallel with the same memory address. The number of memory modules grouped in parallel is dictated by the size of the data element we want to address. This size in cells is the *data granularity*.

Figure 4.b shows a possible memory bank configuration (similar to that of a DRDRAM-like memory system [11]). In a conventional DRAM memory system, the data is interleaved across all memory banks using a certain policy, and the memory controller is simply responsible of broadcasting the addresses to all of them. Each memory bank has a special logic that determines whether the address identifies a data item that the bank contains or not.

In the RADS scheme described in section II the data granularity (B) was given by the DRAM random access. Now, given an array of M memory banks and a random cycle time of T seconds per bank, it is theoretically possible to initiate a new memory access every T/M seconds. Therefore, the data granularity can be potentially reduced by a factor of M.

There are two fundamental limits to the bank interleaving technique. First, the bus address speed: that is, the cycle time required to broadcast an address to all memory banks. Second, the problem of bank collisions. In order to fully exploit the potential bandwidth of an interleaved memory system, we need to grant that the same bank is not accessed twice within its random access time (T). The implementation of conflict-free mechanisms is specially sensible in the context of fast packet buffering, as we need to enforce that no bank collision is ever produced, otherwise, a packet would be lost as a result.

IV. Conflict Free DRAM System (CFDS)

In this section we describe a novel DRAM memory system that grants conflict-free access with affordable cost. The system is based on a special memory bank organization coupled to a reordering mechanism that schedules the different MMA request guaranteeing that no bank-conflicts are present. Figure 5 summarizes the CFDS memory architecture. Four items stand out in CFDS:

- CFDS exploits the DRAM bank organization.
- The *Virtual SRAM Subsystem* works exactly as the SRAM subsystem of the RADS memory architecture described in section II. It uses, however, a granularity of $b < B$ for cells transfers between DRAM and SRAM.
- The *DRAM Scheduler Subsystem* hides the DRAM bank organization to the former *Virtual SRAM Subsystem*.
- The DRAM subsystem can transfer cells to SRAM with a different order of the requested by the *Virtual SRAM Subsystem*. Reordering these cells implies an additional cost in terms of latency and SRAM size. It can be shown, however, that introducing an additional delay and storage, an exact delivery of cells to the arbiter can be guaranteed. Moreover, the benefits of decreasing the granularity outfit the additional cost introduced by the reordering process.

These items are discussed in the following subsections.

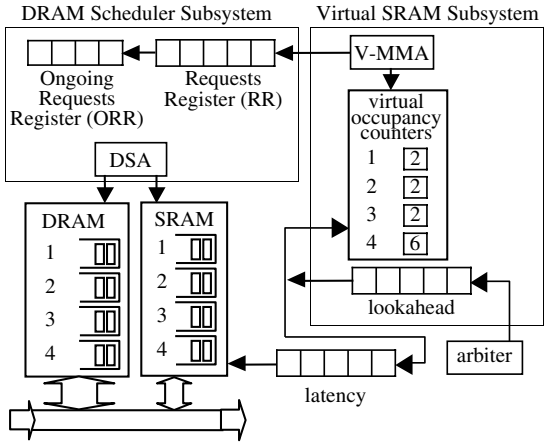

Fig. 5. CFDS Memory Architecture of the packet buffer.

A. DRAM Bank Organization

Let M be the number of DRAM banks. We organize these banks in $G = M/(B/b)$ groups of B/b banks per group (see Figure 6). Each group stores cells of Q/G queues. Banks are accessed transferring b cells of the same queue. Furthermore, in order to avoid bank conflicts, the cells of every queue are stored in a round robin across all the banks of the group (low-order interleaving). Doing this way, we can perform B/b consecutive access to the same queue (transferring B cells overall) without bank conflicts. The distribution of the queues among the maximum number of groups maximizes

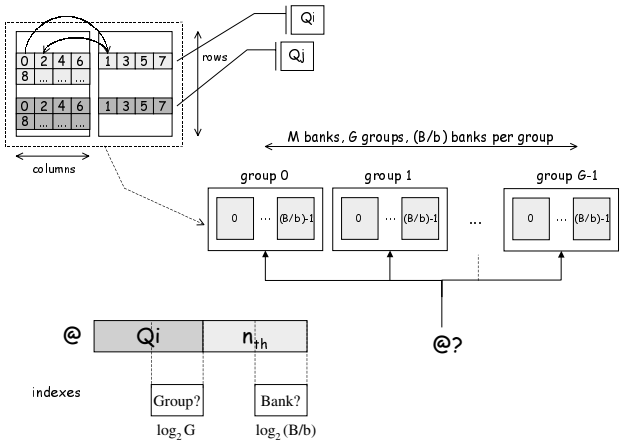

Fig. 6. Proposed memory bank interleaving.

the likelihood of finding independent accesses, reducing the hardware requirements to provide conflict-free access.

In Figure 6 we show the mapping function used to obtain the bank and group indexes. A given request address contains two bit fields, one determining the queue and one determining the relative order. The group index is obtained using the low-order bits of the queue field while the bank inside that group is obtained using the low order-bits of the ordinal field. The rest of the bits are used to determine the row and column addresses of the specified DRAM bank.

B. Virtual SRAM Subsystem

The *Virtual SRAM Subsystem* shown in Figure 5 works in the same way as the RADS memory architecture described in section II, but assuming that $b < B$ cells are transfered every DRAM memory access: Every b slots the *Virtual MMA* (V-MMA), using any of the previously described algorithms, decides the queue to be replenished, issues the request to the *DRAM Scheduler Subsystem*, and updates accordingly the *virtual occupancy counters*.

C. DRAM Scheduler Subsystem

The *DRAM Scheduler Subsystem* (DSS) shown in Figure 5 manages the transfers between the DRAM and SRAM to fulfill the requests issued by the V-MMA. The DSS uses a *DRAM Scheduler Algorithm* (DSA) to avoid bank conflicts, making use of two registers: the *Requests Register* (RR) and the *Ongoing Requests Register* (ORR).

The RR is a shift register that stores the requests made by the V-MMA and that have not being fulfilled yet. Every b slots, the DSA choses a request of the RR, which can be placed at any position of the register. Once a request has been chosen, it is removed from the RR and the requests from this position to the tail of the RR are shifted ahead, making room for the new request that will be issued by the V-MMA b slots later. The ORR is a shift register that stores the banks that currently are being accessed. In case a new request would be issued to any of these banks, a bank conflict would arise. Hence, the banks stored in the ORR are locked and the DSA would never initiate a new transfer of cells residing in these locked banks. Taking into account that a bank is locked during B slots, we need to consider the latest $B/b-1$ ongoing requests. The size of the ORR is hence $B/b-1$.

The DSA choses the *oldest request in the RR addressed to a bank which is not locked*, starting a new transfer of b cells and placing the memory bank where these cells reside at the tail of the ORR. In [9] we demonstrate that if the RR has a size of:

$$L = (2\,Q/G - 1)\,(B/b - 1) + 1, \tag{1}$$

the DSA can always find a non locked request[1].

The factor 2 in the previous expression is due to the fact that we have Q incoming streams from the t-SRAM and Q outcoming streams for the h-SRAM. Note that in case the DSA always choose the head of the RR, the requests delivered by the V-MMA would suffer a delay of $(L-1)\,b$ slots until they are passed to the ORR. If the request at the head of the RR is locked, there will be requests having a delay higher and lower than $(L-1)\,b$ slots. In [9] we demonstrate that the maximum number of times a request can be delayed is:

$$R_{max} = (2\,Q/G - 1)\,(B/b - 1). \tag{2}$$

D. The Latency Register

In the proposed conflict-free access mechanism, the DRAM subsystem may deliver the cells out of order (even those from the same queue that are not located in the same bank). Therefore, we have to introduce a reordering mechanism. The reordering mechanism introduces an additional delay and increments somewhat the SRAM size:

Firstly, an additional delay equal to the maximum delay that a replenish request can suffer due to the DSA reordering has to be added to the lookahead of the V-MMA. This is introduced by the *latency* shift register shown in Figure 5. From the previous results we get that the size of this register must be equal to:

$$\begin{aligned} \text{latency (in slots)} &= b\,((L-1) + R_{max}) \\ &= 2\,b\,(2\,Q/G - 1)\,(B/b - 1). \end{aligned} \tag{3}$$

Finally, note that the replenish requests of the SRAM queues are delivered to DRAM when they leave the RR register, but cells are removed from SRAM when they leave the *latency* shift register. The mismatch between these two events requires increasing the SRAM size (in order to store the cells downloaded to the SRAM before they are granted to the arbiter).

From that, we conclude that two factors contribute to the SRAM size dimensioning: (i) the size required by the *Virtual SRAM Subsystem* given by the algorithm of Figure 3 (using b instead of B), and (ii) the additional SRAM size to cope with the mismatch described above. Summing both terms we have:

$$\text{SRAM size (cells)} = \text{rads\_sram\_size}(L,\,Q,\,b) + b\,R_{max}. \tag{4}$$

[1]Empty requests are considered as requests to a special queue.

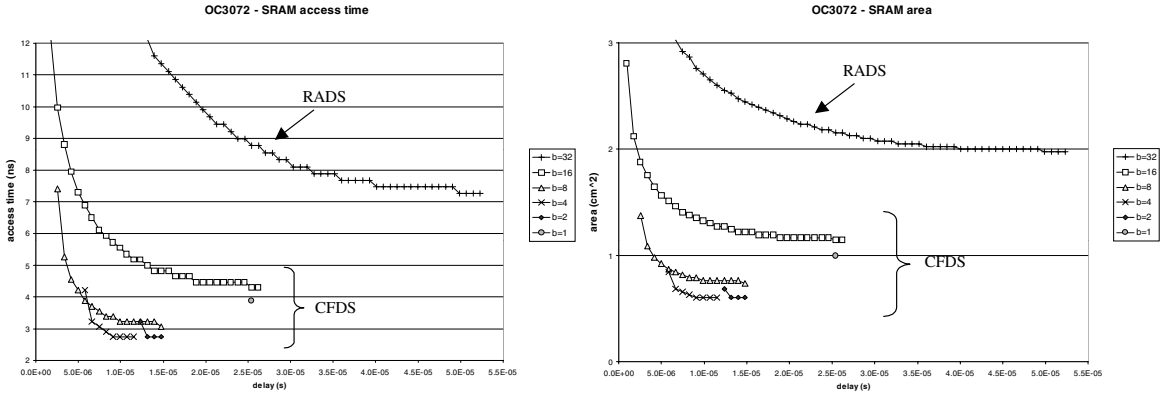

Fig. 7. SRAM area (both h-SRAM and t-SRAM) and access time, as a function of the delay, for the RADS scheme and several CFDS configurations. Q=512, M=256. Delay for RADS = lookahead delay. Delay for CFDS = lookahead delay + latency.

V. Numerical Results

In this section we study the SRAM size reduction that could be achieved using the proposed memory architecture (CFDS) over RADS. Note that a smaller SRAM occupies less silicon area and can operate at faster rates. As in [5] we use $B = 2 \times R \times T/C$, where R is the line rate, T is the random access time of DRAM and C is the the cell size in bits. For OC3072 (R = 160 Gbps), T = 51,2 ns and cells of 64 bytes, we obtain B=32.

Figure 7 allows us to demonstrate the performance benefits of using CFDS instead of a basic RADS approach. The figure shows the area (of both h-SRAM and t-SRAM) and most restricting access time for OC3072 in function of the lookahead delay (measured in μ-seconds). The number of queues Q is 512. The curves with a data granularity of $b = 32$ correspond to the RADS implementation. The rest of the curves correspond to different CDFS configurations varying the value of b. We assume the number of banks M to be 256.

It can be seen the evident advantages of CFDS relative to RADS. A CFDS system with $b = 4$ is compliant with the requirements of buffering packets at 160Gb/s, as *the access time is lower than 3.2 ns*. Moreover, this is accomplished with a modest lookahead delay (10 μs) and an affordable area (0.6 cm^2 overall). This contrasts heavily with its RADS counterpart, which is hardly unable to access data in 7 ns, even with a delay of more than 50 μs and the non-irrelevant area of 2 cm^2.

Another important conclusion is that there is an optimal value of b for a given CFDS implementation. The reason is the trade-off between the SRAM size required to tolerate the unpredictability of arrivals from the arbiter and the SRAM size required to absorb the level of reordering of the accesses from the DRAM. In this case the optimal value is $b = 4$, as it gives minimum SRAM size and access times for a minimum lookahead delay.

VI. Conclusions

In this paper we have proposed a novel architecture targeted at fast packet buffering. In order to overcome the bandwidth problems of current commodity DRAM memory systems, the use of SRAM coupled to DRAM have been proposed in the past. Those SRAM memories act as ingress and egress buffers to allow wide transfers between the buffering system and its main DRAM memory system. The main drawback of this organization is that the data granularity of the DRAM accesses have to be enlarged to sidestep the high cycle times of a single DRAM bank. As a result, the SRAM memories could be too large and slow for very high link rates.

In our proposal, the memory bank organization is exploited in order to achieve conflict-free access to DRAM. This leads to a reduction of the granularity of the DRAM accesses, resulting in a decrease of storage capacity needed by the SRAM.

Acknowledgments

This work was supported by the Ministry of Education of Spain under grant TIC-2001-0956-C04-01 and TIC98-05110C02-01.

References

[1] T. Y.J. and G. L. Frazier, "High-performance multi-queue buffers for vlsi communication switches," in *15th Annual International Symposium on Computer Architecture*, Honolulu, Hawaii, 1988, pp. 343–354.

[2] N. McKeown, "Fast switched backplane for a gigabit switched router," *Cisco Systems white paper*, http://www.cisco.com/warp/public/cc/cisco/mkt/core/1200/tech/fasts_wp.pdf.

[3] C. Minkenberg and T. Engsbersen, "A combined input and output queued packet-switched system based on prizma switch.on-a-chip technology," *IEEE Communications Magazine*, pp. 70–77, December 2000.

[4] F. Abel, C. Minkenberg, R. P. Luitjen, M. Gusta, and I. Iliadis, "A four-terabit single-stage packet switch withlarge round-trip time support," in *10th IEEE Symposium on High Performance Interconnects*, 2002.

[5] S. Iyer, R. Kompella, and N. McKeown, "Analysis of a memory architecture for fast packet buffers," in *IEEE Workshop on High Performance Switching and Routing2001*, Dallas, Texas., 2001.

[6] ——, "Techniques for fast packet buffers," in *Gigabit Networking Workshop'2001*, Anchorage, Alaska, USA, 2001.

[7] ——, *"Designing Packet Buffers for Router Line Cards (Draft version)"*, 2002, http://www.stanford.edu/class/ee384y/papers/bufferdraft.pdf.

[8] D. T. H. III and J. R. Jump, "Performance evaluation of vector accesses in parallel memories using a skewed storage scheme," *International Symposium on Computer Architecture, ISCA*, pp. 324–328, 1986.

[9] J. García, L. Cerdà, and J. Corbal, "A conflict-free memory banking architecture for fast packet buffers," Politechnic University of Catalonia, Tech. Rep. UPC-DAC-2002-50, July 2002, http://www.ac.upc.es/recerca/reports/DAC/2002.

[10] Hitachi, "Hitachi 166 mhz sdram," *Hitachi HM5257XXb series*, 2000, http://semiconductor.hitachi.com/dram/dram_modules.htm.

[11] R. Crisp, "Direct rambus technology: The new main memory standard," *IEEE Micro*, vol. 7, pp. 18–28, November/December 1997.