A DRAM/SRAM Memory Scheme for Fast Packet Buffers

Jorge García-Vidal, *Member*, *IEEE*, Maribel March, Llorenç Cerdà, *Member*, *IEEE*, Jesús Corbal, and Mateo Valero, *Fellow*, *IEEE*

Abstract—We address the design of high-speed packet buffers for Internet routers. We use a general DRAM/SRAM architecture for which previous proposals can be seen as particular cases. For this architecture, large SRAMs are needed to sustain high line rates and a large number of interfaces. A novel algorithm for DRAM bank allocation is presented that reduces the SRAM size requirements of previously proposed schemes by almost an order of magnitude, without having memory fragmentation problems. A technological evaluation shows that our design can support thousands of queues for line rates up to 160 Gbps.

Index Terms—Router architecture, packet buffers, high-performance memory systems, storage schemes.

1 Introduction

A router is a network node connecting several transmission lines, whose basic function is to forward *Internetworking Protocol* (IP) packets across the lines depending on the packet's destination IP address and the information stored in its routing table. The main functional units of a router are:

1. *line interfaces*, which connect the router to each transmission line,
2. *packet processors*, which process the packet headers, look up routing tables, classify packets, and perform related tasks,
3. *packet buffers*, which store the packets waiting to be forwarded,
4. *switch fabric*, which interconnects the router's packet processing units, and
5. *system processor*, which performs the control functions, such as routing table maintenance and configuration tasks.

A basic measure of the performance of a router is its switching capacity, measured as the product of the line-rates of the transmission lines times the number of line interfaces. We can give this measurement in bits/sec or in packets per second (for example, assuming a packet-size of 40 Bytes). Currently, the evolution of high-speed routers is determined by the advances in optical transmission technologies such as DWDM (Dense Wavelength-Division Multiplexing), which makes it possible to exploit the huge potential bandwidth of optical fibers [36]. Data carried by optical fibers continues to double every 8-12 months, with a single fiber capacity exceeding 10 Tbps in the near future [39], [26], [3].

Moreover, this optical transmission technology can already achieve 80-120 wavelengths per fiber for commercial systems, while, in experimental settings, thousands of wavelengths have been multiplexed into fiber [44]. Consequently, the required switching capacity of a router increases due to the increases in the line rates and to the increase in the number of line interfaces (if we are using DWDM, each wavelength is equivalent to a line interface).

A switch built with purely electronic technology cannot deal with the rapid increases in the required switching capacity, so it seems reasonable to consider introducing other types of device, such as optical devices in units with all-optical technology or hybrid electro-optical switches. However, optical technology presents a major limitation since nothing analogous to electronic RAM exists in optics [12]. Although, there are some alternatives that use optical fiber-delay lines with other components such as optical gate switches, optical couplers, optical amplifiers, and wavelength converters (see [15], [21], [32], [42], [51], [25], [24]), they are not commercially feasible. Since the recent introduction of the load-balanced switch described in [5], some hybrid electro-optical switches have been proposed, but they also have several problems that need to be solved to make them practical (see [35], [33], [34]). Therefore, we consider that it is interesting to explore the extent to which one can scale the speed of high-speed electronic packet buffers.

Packet buffers are essential components in the design of a packet switch. Their main function is to absorb surplus traffic directed toward a given interface of the switch. The size and other characteristics of a buffer packet have a direct impact on the performance of the switch and the dynamics of the congestion control mechanisms used in the network.

For a packet buffer, the required bandwidth is at least twice the line interface transmission rates. Furthermore, high-speed routers not only need high-speed buffers, but they also may need large buffer storage. Usually, to calculate the size of the packet buffer required in a packet

• *J. García-Vidal, M. March, L. Cerdà, and M. Valero are with the Computer Architecture Department, Technical University of Catalonia, c/Jordi Girona 1-3, 08034 Barcelona, Spain. M. Valero is also with the BSC, Barcelona Supercomputing Center.*
E-mail: {jorge, mmarch, llorenc, mateo}@ac.upc.edu.
• *J. Corbal is with BSSAD, ILB, Intel, c/Jordi Girona 1-3, 08034 Barcelona, Spain. E-mail: jesus.corbal@intel.com.*

Manuscript received 29 July 2005; revised 14 Dec. 2005; accepted 3 Jan. 2006; published online 22 Mar. 2006.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-0251-0705.

switch, manufacturers use the well-known rule-of-thumb $buffer\ size = RTT \times C$, in which RTT is the round-trip time of the Internet and C is the line rate of the line interface, so a packet switch that uses line interfaces with $C = 40$ Gbps in a network with $RTT = 200$ ms requires packet buffers able to read/write 80 Gbps and a size of 1 GB.

The validity of the previous dimensioning rule in very high capacity links has recently been questioned. In [2], [48], it is argued that, in backbone routers that switch thousands of TCP flows, the phenomenon of flow synchronization does not occur when there is congestion ([19], [9]), so we can drastically reduce the packet buffer size, using, as a dimensioning rule, $buffer\ size = \frac{RTT \times C}{\sqrt{N}}$, where N is the number of active TCP flows in the link. On the other hand, in [11], it is argued that this dimensioning rule, which focuses on maintaining a high link utilization and low delay, can lead to high loss rate and poor performance for many applications. Another dimensioning rule, which, in fact, can give buffer size values even larger than the ones obtained with the bandwidth-RTT rule, is derived. Given the lack of accurate models for Internet traffic and the closed-loop nature of TCP, analytical or simulation models cannot give the final answer to the buffer dimensioning question and, thus, real measurements in congested Internet links are needed. None of the alternative dimensioning rules have been extensively tested to date in real scenarios and buffer sizing of Internet routers remains an open question. Today, routers manufacturers seem to favor the use of large buffers. For instance, the Cisco CRS-1 modular service card with a 40 Gbps line rate incorporates a 2 GB packet buffer memory per line card and side, ingress, and egress (see [7]).

The problem of packet storage is not only related to the absorption of traffic peaks. As an interesting example, we might mention the case of the packet buffers used in OBS (Optical Burst Switching) edge routers. Optical Burst Switching (OBS) is assumed to be the most practical solution in the near future [6], [45], [38], [50], [31] for efficiently using the huge potential bandwidth of optical fibers that the advances in DWDM technology have made possible. In OBS, packets from various sources are aggregated into bursts at the ingress edge of an OBS network and disassembled at the egress edge router. A control packet is sent first to set up a connection (by reserving the appropriate amount of bandwidth and configuring the switch fabric along a path), followed by the burst of data. This signaling process implies that packets from a burst should be stored at the edge routers during timescales of milliseconds, which means that edge OBS routers may require buffer sizes in the order of Giga bytes of data.

Traditionally, fast packet buffers were built using low-latency SRAM. However, with the increasing capacity requirements, high density DRAMs have become the preferred choice. DRAM-based packet buffers can easily provide a bandwidth of up to around 1 Gbps, but, if we increase the required bandwidth to several Gbps, the design becomes difficult. For instance, [20] addresses the design of a packet buffer using a single-chip 16 Mb SDRAM with a 16 bit data interface and a 100 MHz clock. Even though the peak bandwidth is 1.6 Gbps, the guaranteed bandwidth drops to 1.2 Gbps due to the activate and precharge overhead. A multiple chip design would increase the buffer bandwidth, but the increase in bandwidth would not be proportional to the total number of chips. Using, for instance, the same SDRAM parameters, an 8-chip configuration with an 8x wider bus would provide a guaranteed bandwidth of only 5.12 Gbps. Increasing the number of chips and widening the data bus therefore yields diminishing returns, while creating problems [13] such as higher memory granularity, more memory components in the line card, and wider data paths.

The low efficiency of multichip DRAM buffers can be improved by using some special techniques aimed at reducing bank conflicts in a DRAM buffer such as pipelining and out-of-order access techniques [47], [46], [37] or exploiting row locality whenever possible in order to enhance average-case DRAM bandwidth [8], [22]. Using faster DRAM components (e.g., RLDRAM [28], FCDRAM [14], etc.) would also lead to faster buffers. However, from the previous discussion, it is clear that, to support a line-rate as high as OC-3072, alternatives to DRAM-only buffers should not be considered.

Taking these facts into account, the fastest packet buffers with worst-case bandwidth guarantees that can be found in the literature are hybrid SRAM/DRAM designs, first described in [29]. In these, Virtual Output Queuing and a combined SRAM/DRAM packet buffer architecture are used. SRAM only stores the tail and head of queues in order to ensure the line rate and DRAM stores the rest of them in order to ensure the large storage that is needed. To our knowledge, the hybrid design proposals made in this field are [29], [17], and [18].

Our novel proposal presented in this paper maintains the hybrid SRAM/DRAM design of [29], but introduces the following changes: 1) We redesign the functional blocks that govern SRAM/DRAM memory transfers to obtain a *general* hybrid SRAM/DRAM design, for which the schemes of [29] and [17] can be seen as particular cases. 2) We propose a new algorithm that reduces the SRAM size of the scheme proposed in [29] almost by an order of magnitude. Furthermore, this new algorithm avoids the memory fragmentation problem of the scheme proposed in [17]. To the best of our knowledge, the design proposed in this paper is the fastest that has been published to date for large packet buffers.

A technological evaluation presented in this paper shows that our design can support thousands of queues for line rates of 160 Gbps using commodity DRAM.

The rest of the paper is organized as follows: In Section 2, we explain the system assumptions for the paper. In Sections 3 and 4, we describe our proposal and we argue that the previous ones are particular cases of our design. We then discuss some implementation issues. Section 9 is devoted to previous work on VOQ buffer design and Section 10 gives some concluding remarks.

2 System Assumptions

During the next few years, aggregate router throughput will probably grow by increasing the number of interfaces rather than increasing line rates [36]. Although line rates have increased rapidly over the past years (up to OC-192 or

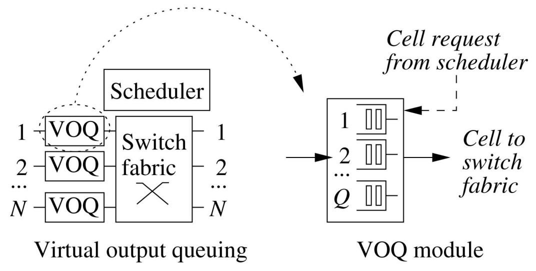


Fig. 1. Input-queuing router architecture using VOQ buffer.

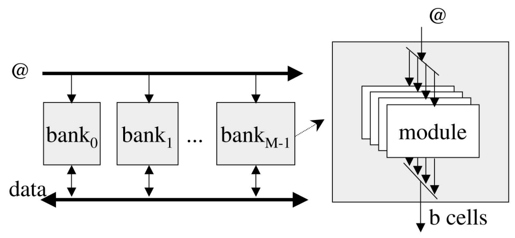


Fig. 2. Organization of a DRAM in banks: configuration of a DRDRAM-style memory and internal structure of a memory bank.

OC-768 [30], [7]), it seems that this increase is close to its electronic limits: around OC-3072 [36].

The use of Dense Wavelength-Division Multiplexing (DWDM), however, increases the number of channels available on a single fiber (without increasing the individual line rates), leading to a number of interfaces on the order of several hundred.

Our target design is to support line rates as high as OC-3072, a number of line interfaces in the order of thousands, with the capacity of grouping cells into a number of internal logical queues (e.g., using Virtual Output Queuing, as we explain below). We can therefore set several parameters that are of utmost importance in the packet buffer design: required bandwidth, buffer size, basic time-slot, and number of internal data structures internal to the buffer.

Required bandwidth. For input-queuing architecture, the required packet buffer bandwidth is twice the line rate as every packet must be both written and read from memory before being forwarded. In a shared memory buffering, packets continue to recirculate through the switch fabric in a common buffer pool, with each output removing one packet from the group each time slot, so the required total buffer is twice the line rate per number of input interfaces. In the numerical examples that we use in this paper, we do not consider any further speeding up. Note, however, that, in practice, a speed-up of 1.5-2 is commonly used to compensate for the allocation and scheduling conflicts.

Buffer size. As we discussed in the previous section, router manufacturers usually employ packet buffers of a size equal to an estimate of a typical packet round-trip time over the Internet times the line rate [13]. Taking a typical round-trip time of 0.2 sec, the required buffer size for a line rate of 160 Gbps is 4 GB.

Basic time-slot. We assume that packets in the router are internally fragmented into units that we will call cells. We will take a fixed-length of 64 bytes [4]. Cells are handled as independent units, although they are reassembled at the output port before packet transmission. The system operates synchronously into fixed time-slots, which correspond to the transmission time of a cell at the line rate. For example, for a line rate of 160 Gbps, the basic time-slot is of 3.2 ns.

Number of internal data structures. As is well known, in order to achieve full link utilization, input-buffered routers require the use of *Virtual Output Queuing* (VOQ) [43]. In VOQ, (see Fig. 1), the input buffer maintains Q separate logical FIFO queues. Each logical queue corresponds to an output line interface and a class of service. When a cell reaches the input line interface, it is placed at the tail of the queue corresponding to its outgoing interface. When an input port receives a request for a cell addressed to a given output, the cell is taken from the head of the corresponding queue in the VOQ buffer. We will assume that our packet buffer incorporates this mechanism. We assume that the number of Virtual Output Queues to be supported is around 1,000.

DRAM bank interleaving. In response to the growing gap between processor and memory speed, DRAM manufacturers have created several new architectures that address the problem of latency, bandwidth and cycle time (e.g., DDR-SDRAM [23]or RAMBUS DRDRAM [10]). All of these commercial DRAM solutions implement a number of memory banks—as many as 512—that can be addressed independently. Banking in DRAM allows an access to one bank to begin while the other is still busy. Thus, by performing several on-the-fly requests to different banks, we can reduce the "random" access time of a DRAM memory system that implies a reduction in the SRAM size needed by a hybrid DRAM/SRAM architecture, as is shown in [16].

Fig. 2 illustrates the concept of a memory bank and 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*. In the following, we shall refer to the data granularity used as b. Furthermore, in a conventional DRAM memory system, the data is interleaved across all memory banks using a specific policy and the memory controller is simply in charge of broadcasting the addresses to them. Each memory bank has a special logic that determines whether or not the address identifies a data item that the bank contains. We will assume that our packet buffer incorporates DRAM bank interleaving.

3 General Hybrid SRAM/DRAM Architecture (GHDS)

Our proposal is a *general* hybrid SRAM/DRAM design of which the schemes of [29] and [17] are particular cases, as we assess in the following section.

Fig. 3 shows the GHDS architecture. The system consists of 1) two fast but costly SRAM memory modules (t-SRAM and h-SRAM), 2) a slow but low-cost DRAM memory, and 3) the functional blocks that govern the transfers between

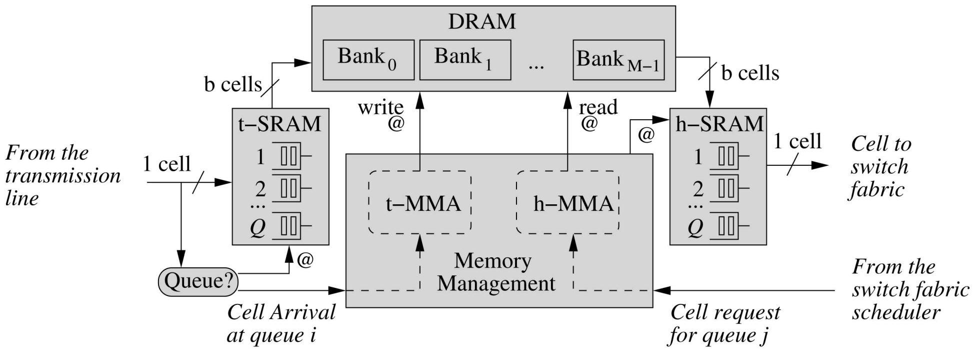
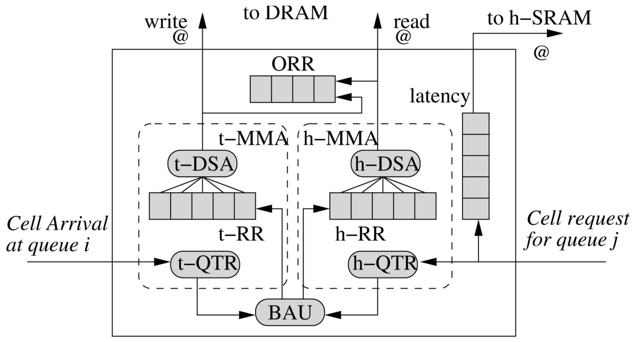


Fig. 3. General hybrid DRAM/SRAM memory architecture (GHDS).



Legend:
RR: Requests Register
ORR: Ongoing Requests Register
BAU: Bank Allocation Unit
QTR: Queue Transfer Requester
DSA: DRAM Scheduler Algorithm
MMA: Memory Management Algorithm

Fig. 4. Memory management of GHDS.

DRAM/SRAM memory modules (indicated as Memory Management in Fig. 3).

In the scheme shown in Fig. 3, we maintain the hybrid SRAM/DRAM memory organization of [29] with the addition of units for DRAM bank allocation and DRAM scheduling. These units are key to obtaining a system which can fully exploit the capabilities of interleaved memory access.

The DRAM memory is organized in M banks and data are interleaved among them. However, note that there are two fundamental limits to using bank interleaving. The first is the bus address speed, that is, the cycle time required to rebroadcast an address to all memory banks. The second is the problem of bank collisions. In order to fully exploit the potential bandwidth of an interleaved memory system, we need to guarantee that the same bank is not accessed twice within its random access time (T). A *bank conflict* occurs when this constraint cannot be fulfilled. The implementation of conflict-free mechanisms is especially relevant in the context of fast packet buffering because a collision would result in the loss of a packet. In proposals [29] and [17], no bank collision ever takes place. We present in this paper an alternative system for which collisions only occur with a very small probability, independently of the traffic patterns.

The t-SRAM and h-SRAM, respectively, cache the *tail* and *head* of each VOQ logical queue. The rest is stored in DRAM. Cells that come into the buffer are placed in the t-SRAM, whereas cells that will leave the system in the near future are placed in the h-SRAM. Since the SRAM memory bandwidth must fit the line rate, the SRAM access time must be less than or equal to the transmission time of a cell (that is the *time slot*). The availability of room in the t-SRAM and the availability of cells to be served in the h-SRAM is controlled using two Memory Management Algorithms: the *tail Memory Management Algorithm* (t-MMA) and the *head Memory Management Algorithm* (h-MMA), respectively, which must *guarantee* that there is always room in the t-SRAM for an incoming packet and that any packet to be output is always present in the h-SRAM before the outputting needs to be done (i.e., the cache never misses).

Therefore, the accesses to DRAM are managed by the t-MMA and the h-MMA. When the occupancy of the t-SRAM reaches a given threshold, a transfer from t-SRAM to DRAM of a group of cells addressed to the same output interface is ordered by the t-MMA. Conversely, when the h-SRAM needs to serve a cell that currently resides in DRAM, the h-MMA orders a group transfer from DRAM to h-SRAM. In order to match DRAM/SRAM access times, transfers between DRAM and SRAM occur in batches of b cells. Note that these transfers have a size in cells that should be set to the ratio of DRAM random access time to the transmission time of a cell. In the following two subsections, we describe these algorithms in depth.

3.1 Tail Memory Management Algorithm

Every b time-slots, the *tail Memory Management Algorithm* (t-MMA) selects a queue and a memory bank from which b cells must be transferred from t-SRAM to DRAM. These transfers should guarantee that the t-SRAM does not fill up before DRAM. Otherwise, losses would occur before the DRAM is full.

The t-MMA module consists of (see Fig. 4): a *Queue Transfer Requester* module (t-QTR), a *Request Register* (t-RR), and a *DRAM Scheduler Algorithm* module (t-DSA). Two additional modules, a *Bank Allocation Unit* (BAU) and the *Ongoing Request Register* (ORR), described later in this section, are shared by both t-MMA and h-MMA.

The functional blocks of the t-MMA work as follows: When a cell for queue i arrives to be stored in the t-SRAM, the t-QTR decides whether a transfer from the t-SRAM to DRAM has to be scheduled for this queue. Since the t-SRAM has to be emptied as soon as possible, the t-QTR schedules a transfer whenever it is able to do so, i.e., when b cells of queue i are standing in the t-SRAM. Equivalently, let C_i^t be a counter of the number of cells arriving at queue i (C_i^t is initialized to 0). Each time a cell arrives for queue i, C_i^t is increased and the t-QTR issues a transfer request for queue i if $(C_i^t \bmod b) = 0$.

The request issued by the t-QTR is processed by a *Bank Allocation Unit* (BAU), which in turn chooses the bank in which cells should be allocated (the algorithm will be discussed in Section 5). The request issued by the BAU contains the queue from which b cells must be transferred and the bank in which these cells will be placed.

In order to avoid DRAM bank conflicts, a *tail DRAM Scheduler Algorithm* (t-DSA) is used. It takes into account two registers: the *Tail Requests Register* (t-RR) and the *Ongoing Requests Register* (ORR). The t-RR is a shift register that stores the t-SRAM requests processed by the BAU that have not yet been fulfilled. Every b slots, the t-DSA selects one of the transfer requests pending on the t-RR, which can be located at any position of the register, and issues a write transfer to DRAM. To choose it, the t-DSA may take into account the information stored in the *Ongoing Requests Register* (ORR). The ORR is a shift register that stores the identifiers of the banks that are currently being accessed. If a new request were issued to any of these banks, a bank conflict would arise. Hence, the banks with identifiers stored in the ORR are locked. Therefore, the t-DSA chooses the *oldest request in the t-RR addressed to a bank which is not locked*, starting a new transfer of b cells and placing the memory bank identifiers at the tail of the ORR.

3.2 Head Memory Management Algorithm

The transfers between the h-SRAM and DRAM are managed by the *head Memory Management Algorithm* (h-MMA). Now, the h-MMA has to guarantee that cells transferred between DRAM and h-SRAM can accommodate the sequence of cells requested, for example, by the *switch fabric scheduler* in an input-queuing switch. Otherwise, the cell requested may not be present in the h-SRAM because it may not have been transferred from the DRAM yet. We shall refer to this condition as a *miss*.

Again, the h-MMA algorithm is simple: Schedule a transfer for queue i whenever the number of requests for cells belonging to queue i exceeds the number of cells from this queue present in the h-SRAM. Equivalently, let C_i^h be a counter of the number of cells requested from queue i (C_i^h is initialized to 0). Each time a cell from queue i is requested, C_i^h is increased and the *head Queue Transfer Requester* module (h-QTR) issues a transfer request for queue i if $(C_i^h \bmod b) = 1$. We shall refer to the delay as the *h-MMA response time* since the h-QTR schedules a transfer until the corresponding download of b cells from DRAM to h-SRAM is finished. Analogously, we define the *t-MMA response time* as the delay from when the t-QTR schedules a transfer until the corresponding upload of b cells from t-SRAM to DRAM is finished.

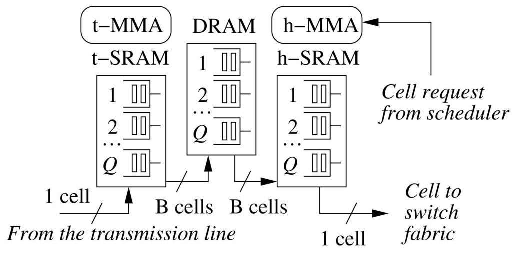


Fig. 5. Hybrid memory architecture.

The rest of the functional blocks of the h-MMA works analogously to those of the t-MMA. Nevertheless, we need an additional *latency* register (see Fig. 4). This register introduces a delay since a request is issued until the h-SRAM is accessed to grant the corresponding cell. This delay is necessary to cope with the response time of the h-MMA. Furthermore, note that, in order to have a zero miss probability, the delay introduced by the latency register should be equal to the *maximum* response time of the h-MMA.

4 Extension of the General Model to Previous DRAM/SRAM Schemes

In this section, we show that previously proposed DRAM/SRAM schemes, such as [29] and [17], are particular cases of the general model introduced in Section 3.

4.1 Random Access DRAM System (RADS)

In this paper, we shall refer to the hybrid DRAM/SRAM scheme proposed in [29] as the *Random Access DRAM System* (RADS). This scheme is shown in Fig. 5. Since DRAM bank interleaving is not exploited, this memory system cannot take advantage of banks. Let us define B as the minimum granularity that can be used if we require a random transfer from any SRAM and any DRAM memory bank. In this case, B is limited by the random access time of the DRAM (T), e.g., if the link rate is R bps and the cell size is C bits, we would have: $B \geq 2T R/C$ (we use $2T$ since, each B time slots, we have to do a read and a write transfer to DRAM). Therefore, to analyze this proposal, we are forced to rely on the worst-case scenario, so the data granularity is given by the DRAM random access of a single bank ($b = B$). In this case, no specific BAU is needed (i.e., consecutive accesses to DRAM can be done to any bank). The DSA can be seen as a FIFO scheduler, which alternatively chooses the oldest write and the oldest read stored into the t-RR and the h-RR, respectively. This scheme is equivalent to the so-called *Early Critical Queue First* (ECQF) proposed in [29]. It can be shown, using a worst-case pattern argument, that the required h-SRAM, t-SRAM, and latency register size in cells is $Q(B-1)+1$. Shorter sizes would lead to miss probabilities that could be large for some specific traffic patterns.

Note that, in this hybrid architecture scheme, the transfers between SRAM and DRAM have a size in cells that should be set to the ratio of DRAM random access time to the transmission time of a cell. As this factor directly influences the SRAM size, large SRAMs are needed to sustain high line rates and a large number of interfaces.

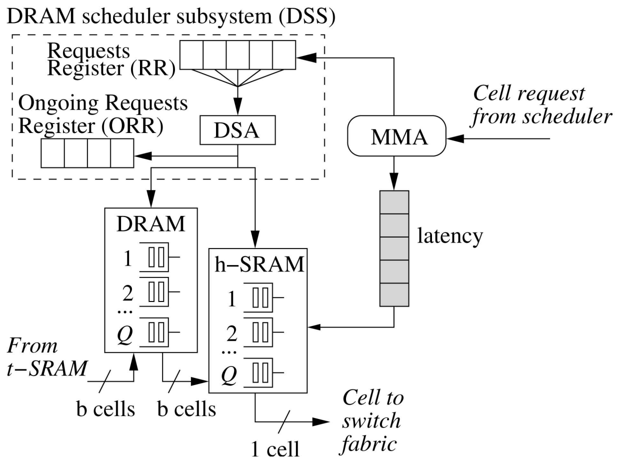


Fig. 6. CFDS memory architecture of the packet buffer.

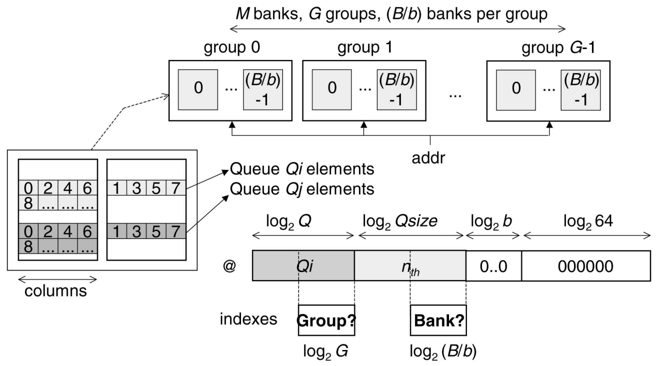


Fig. 7. Proposed memory bank interleaving. The example on the left illustrates the block-cyclic interleaving using $B/b = 2$ banks per group.

This, in turn, limits what access times are attainable. This buffer design would support line rates of up to OC-3072, but only for a reduced number of interfaces. Thus, although the scheme proposed in [29] ensures zero loss probability for cells coming to a nonfull buffer, the required SRAM size for a large number of interfaces becomes too large.

4.2 Conflict-Free DRAM System (CFDS)

In [17], we described a scheme which aims at reducing the SRAM size of [29] while supporting a larger number of interfaces. The scheme we proposed in [17] is based on the observation that the effective DRAM access time can be reduced by overlapping multiple accesses to different banks, that is, by exploiting the potential DRAM bank interleaving. This allows us to reduce the granularity of the accesses, thereby also reducing the SRAM size.

Fig. 6 summarizes the Conflict-Free DRAM System (CFDS) memory architecture. In this proposal, we maintain the same hybrid SRAM/DRAM structure and MMA subsystem as [29], but we completely redesign the DRAM system. We propose a DRAM storage scheme and its associated access method that achieves a conflict-free access memory organization with a reduction in the granularity of DRAM accesses.

Note that if we use a DRAM 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 (as we can perform sequential accesses at an M times faster rate). However, remember from Section 2 that we need to guarantee that the same bank is not accessed twice within its random access time. Otherwise, a bank conflict occurs.

The DRAM memory organization is as follows: Let M be the number of DRAM banks. We organize these banks into $G = M/(B/b)$ groups of B/b banks per group (see Fig. 7). Each group stores cells of Q/G queues. Banks are accessed by transferring b cells from the same queue. In order to avoid bank conflicts, the cells in each queue are stored in blocks of b cells following a round-robin configuration among all the banks belonging to the same group in which the queue was assigned (block-cyclic interleaving).

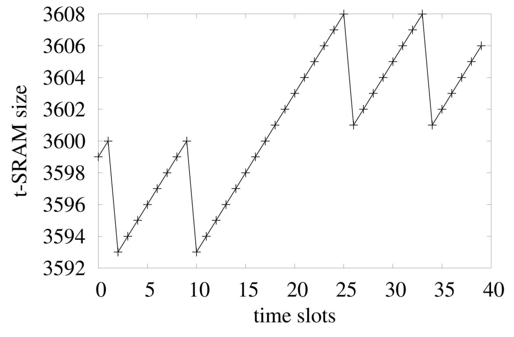
The transfers between the DRAM and SRAM are managed by the *DRAM Scheduler Subsystem* (DSS) shown in Fig. 6. It hides the DRAM bank organization from the former MMA Subsystem, which operates under the illusion that the DRAM access time is b time-slots, even though the actual DRAM access time is B time-slots. This is the main difference between [29] and [17]: $b < B$ cells are transferred between DRAM and SRAM every DRAM memory access time. However, in reality, the DRAM access time remains B time slots. It is this illusion that reduces the SRAM size.

The DSS uses a *DRAM Scheduler Algorithm* (DSA) to avoid bank conflicts, using two registers: the *Requests Register* (RR) and the *Ongoing Requests Register* (ORR). The behavior of the DSA is analogous to that explained in Section 3 for the GHDS model. The DSA must thereby choose the oldest eligible request in the RR and, then, issue the write or read transfer to or from DRAM.

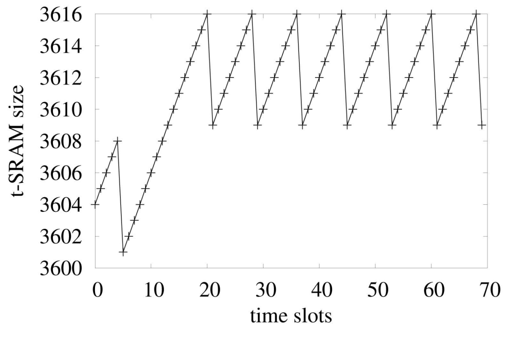
This implies that the DRAM subsystem may deliver cells out of order. Reordering these cells implies an additional cost in terms of latency and SRAM size. The additional delay, equal to the maximum delay that a replenish request can suffer due to the DSA reordering, is introduced by the *latency* shift register shown in Fig. 6. However, analysis in [16] shows that this reordering is bounded and that a zero miss condition can be guaranteed. Moreover, the benefits of decreasing the granularity outweigh the additional cost introduced by the reordering process.

This memory scheme has the drawback of DRAM—*memory fragmentation*, i.e., certain traffic patterns would lead to a situation in which only a fraction of DRAM memory can be used, depending on the assignment of queues to memory groups.

In [17], we alleviated this by using a *renaming of queues* mechanism that reduces the probability of DRAM memory fragmentation. It consists of associating each logical queue name Q^l, used internally for identifying queues assigned to a certain group, to more than one physical queue name (Q^p). Initially, when no cells from Q^l_i are stored in DRAM, a free Q^p_j identifier from the group with the fewest cells is assigned to it. If cells reaching this queue find that the DRAM assigned to the group is full, a new Q^w_k will be chosen from another group that could offer free DRAM space. By doing this, cells from a given logical queue can reside in more than one memory group and can potentially occupy the whole DRAM system. Renaming of queues



(a)



(b)

Fig. 8. Example of the h-SRAM occupancy with the RBAU scheme when (a) a first miss and (b) a second miss occurs and no speed-up is introduced (for $M = 32$, $Q = 512$ and $B = 32$ and $b = 8$).

makes it much more difficult for memory fragmentation to arise. However, for some traffic patterns, fragmentation cannot be avoided.

5 Random BAU Scheme (RBAU)

In this section, we describe a scheme, first presented in [18], that exploits DRAM bank organization as in [17] (see Section 4). It allows a data granularity of $b < B$ and, thus, reduces the SRAM size. Moreover, the scheme described in this section does not have the memory fragmentation problem of [17].

The BAU we propose randomly chooses a DRAM memory bank for every transfer request issued by the QTR. This random selection is done as follows: Let r_n^i be the $n\mathrm{th}$ request for the $i\mathrm{th}$ queue issued by the t-QTR. Then, the DRAM memory bank allocated to r_n^i is randomly chosen among all the banks, provided that requests $r^i_{n-\frac{B}{b}+1}, \ldots, r^i_{n-1}, r^i_n$ are always addressed to different banks (i.e., different banks are chosen for any B/b consecutive requests for the same queue). As the queues are FIFO, consecutive h-QTR requests for the same queue will also correspond to different bank accesses. In this way, we avoid bank conflicts in transfers of cells from the same queue (as we see in Section 3, we can only access the same bank every B time slots and we access DRAM every b time slots).

The associated DSA chooses the oldest eligible request in the RR, i.e., the oldest request that can be issued to DRAM without suffering bank conflict. This RBAU scheme could be easily implemented as a Linear Feedback Shift Register (LFSR) or Tauseworth generator (e.g., [49], [1]).

6 SRAM Dimensioning

In this section, we describe some dimensioning guidelines for the SRAM modules used in our GHDS Architecture for the RBAU mechanism explained in Section 5.

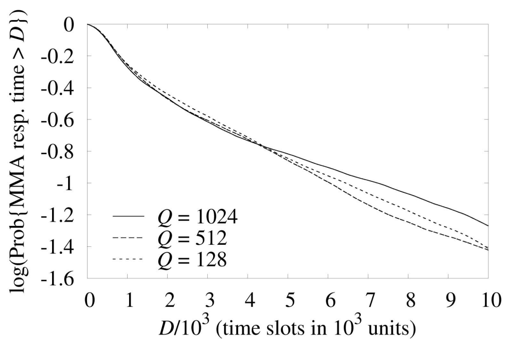
Let us first assume $b = B$. As we explained in Section 4, there are never bank conflicts and the scheme is equivalent to the ECQF mechanism proposed in [29], thus requiring sizes for both SRAM and the latency register of $Q\,(B-1)+1$.

Now, let us consider a scenario with a granularity $b < B$, in which bank conflicts may occur. A miss occurs when all the transfer requests stored in the t-RR or in the h-RR are addressed to banks that are busy at that moment, so the t-DSA or the h-DSA, respectively lose the chance of issuing a read or write request to DRAM. In the following, we outline the dimensioning of both t-SRAM and h-SRAM.

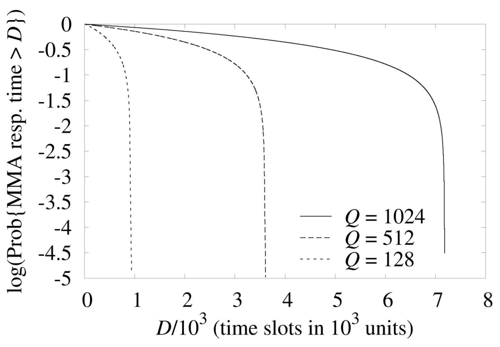
For the t-SRAM, a miss on the t-DSA implies that no write request will be served for the next B slots. Therefore, the t-SRAM will increase its maximum size in b cells. Furthermore, as the t-RR can receive requests every slot and the t-DSA only runs every b slots, the system cannot recover from this loss. Fig. 8 shows this behavior.

In order to overcome this problem, we should introduce a small speed-up in the t-DSA. This speed-up consists of adding extra read cycles in the t-DSA. During these extra cycles, the t-DSA can serve requests accumulated when misses occur, thus making the system recovery feasible from this situation. In the next section, we show that, for practical purposes, the t-SRAM size can be given by $Q(b-1)+1$.

Let us now consider the h-SRAM case. If the h-DSA cannot issue any read request to DRAM because of a miss, there will not be any read transfer between DRAM and h-SRAM after B slots. For this reason, when its associated cell reaches the head of the latency register, the h-SRAM will not be able to serve it. Hence, the maximum response time could be as high as $Q\,(B-1)+1$. This response time would occur if the scheduler issued Q consecutive requests addressed to different queues and all the requested cells were stored in the same DRAM memory bank. If we want to guarantee a zero miss probability, we would need an h-SRAM, t-SRAM, and Latency Register of size $Q\,(B-1)+1$. This maximum value would be needed in the event that all requests are addressed to cells stored in the same bank. However, given the random bank assignment policy used by the RBAU scheme, we expect the probability of the former event to be extremely low (on the order of $\frac{1}{M^Q}$), independently of the traffic pattern. In other words, it is plausible to assume that the event leading to the maximum response time $Q\,(B-1)+1$ using the RBAU scheme is very unlikely to happen. In fact, in the next section, we show that, for practical purposes and realistic values of M, the system can be



(a)



(b)

Fig. 9. Complementary CDF of the MMA response time for different values of Q, $M = 32$, and $B = 32$. In (a), $b = 1$, whereas, in (b), $b = 8$.

dimensioned as if no bank conflicts occur, i.e., assuming a MMA maximum response time of $Q(b-1)+1$.

7 Numerical Results

In this section, we analyze the RBAU Scheme described in Section 5. For the results shown here, we use the following scenario: The t-MMA and h-MMA, respectively, receive a sequence of cell arrivals and scheduler requests in a round-robin for queues $1, 2, \ldots, Q$. In response to this pattern, the t-QTR and the h-QTR generate periodic bursts of transfer requests for queues $1, 2, \ldots, Q$. Note, however, that the random bank assignment performed by the BAU makes the exact sequence of cell arrivals and scheduler requests irrelevant.

For h-SRAM dimensioning purposes, the key parameter to study is the h-MMA maximum response time (see Section 6). Remember from Section 6 that it would be the same for the t-MMA. Fig. 9a shows the Complementary Cumulative Distribution Function (Complementary CDF) of the MMA response time under the load conditions previously described for $M = 32$, $B = 32$, $b = 1$, and different values of Q. The three lines are almost coincident, indicating that, in this case, the delay is almost independent of Q. When $b > 1$, the response time is very much dependent of Q, as is clearly shown in the results in Fig. 9b for $b = 8$ and different values of Q. It is easy to check that the complementary CDF of the delay has a sharp decreasing for a delay of $Q(b-1)+1$ time-slots.

Fig. 10 shows the results obtained for a fixed value of Q and different values of b. We can see again that, for values of b higher than 2, the curves have a sharp decrease at point $Q(b-1)+1$.

The influence of M, the number of memory banks, is easily assessed in Fig. 11. The curves were obtained for $Q = 1,024$, $B = 32$, and $b = 8$. As we can see, the lines associated with values between 256 and 8 are almost coincident, indicating that the delay is essentially independent of the number of banks used when we have more than four banks. Similar conclusions are drawn from Fig. 11b for $b = 2$. In this case, the number of banks required for achieving this insensitivity is $M = 16$.

The previous numerical results show that $Q(b-1)+1$ is a plausible dimensioning rule for the t-SRAM, h-SRAM, and the latency register in the Random BAU Scheme for realistic values of M and b. As a consequence, provided that we can build a fast enough MMA unit, the SRAM size can be almost an order of magnitude lower than the one that would be required using the design given in [29]. Furthermore, the RBAU Scheme does not have the DRAM fragmentation problem of the design we proposed in [17]. It is clear that increasing the value M will lead to a lower collision probability when the previous dimensioning rule is used. Although we do not have available analytical expressions or bounds for this collision probability, we can expect that large values of M (e.g., $M = 128$) will lead to tiny values of collision probabilities, even for low values of b.

Additional numerical results can be found in [18].

8 Evaluation of the GHDS Memory Architecture

In this section, we will extend the scope of the study of the RADS, CFDS, and RBAU systems to design and implementation issues, taking into account technological constraints.

We call RADS to the hybrid DRAM/SRAM scheme proposed in [29] based on determinist worst case and access granularity matching the DRAM random access time (see Section 4.1). We call CFDS to the variation proposed in [17], where we combined a specific bank interleaving scheme with a memory reordering system that allowed us to leverage zero packet lossed with a granularity lower than the DRAM random access time (see Section 4.2). We finally, as shown in

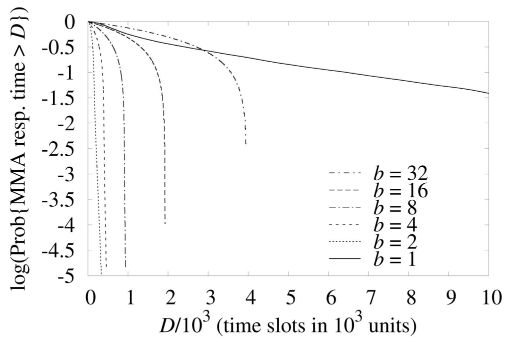


Fig. 10. Complementary CDF of the MMA response time for $Q = 128$, $M = 32$, and different values of b.

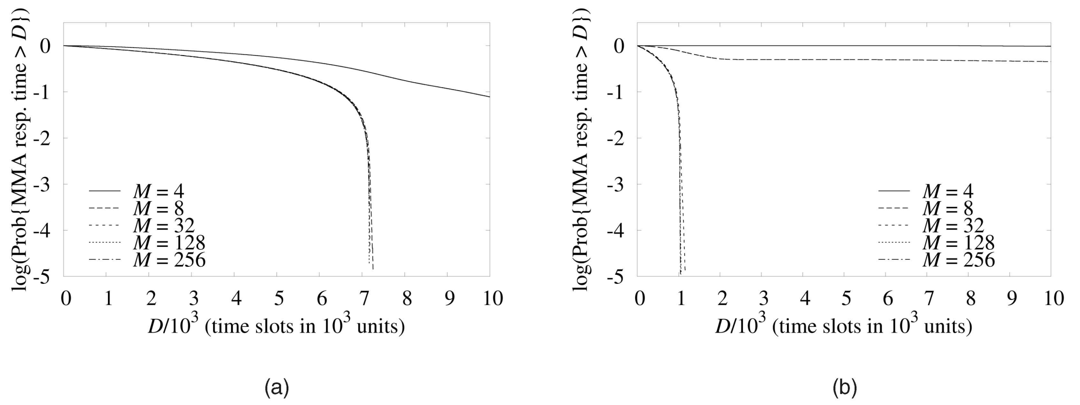


Fig. 11. Complementary CDF of the MMA response time for different values of M and (a) $b = 8$ and (b) $b = 2$.

Section 5, call RBAU to our general GHDS hybrid SRAM/DRAM architecture with Random BAU scheme.

We pose the problem of implementing an SRAM structure able to handle several queues and we propose two design alternatives: one targeted at low cost (area and power) and one targeted at high performance. We finally estimate the area and access time and determine the viability of the proposed SRAM design. We show how RBAU helps to alleviate the limits of technology by requiring smaller SRAMs (to an even greater extent than previous systems such as CFDS), thus allowing simpler designs to accomplish the area and time restrictions.

Throughout this section, we shall assume OC768 and OC3072 links, each with 128 and 512 queues, respectively.

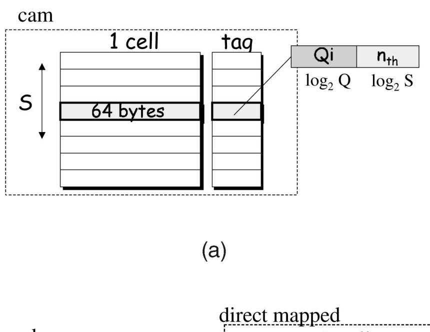
8.1 Design of SRAM Buffers

The main focus of this section is to establish the possible technological barriers in the near future for the RADS/CFDS/RBAU SRAM buffer schemes. We have used CACTI 3.0 [41] to estimate the access time (in ns) and the area (in cm^2) of different implementations of the t-SRAM and h-SRAM buffers using a 130 nm technological process as a baseline. We have also evaluated capability limits for near-future technological processes: 65 and 45 nm. CACTI is an integrated cache access time, cycle time, area, aspect ratio, and power analytical model. The main advantage of CACTI is that, as all these models are integrated together, trade-offs between the different parameters are all based on the same assumptions and, hence, are mutually consistent.

We have assumed that the t-SRAM and h-SRAM are shared by all the queues. The design of a unified (shared) SRAM buffer is not as trivial as the design of a distributed (isolated) SRAM buffer, in which each queue has its own partition of the available memory. The second kind of SRAM buffer could be easily implemented as a set of circular queues implemented with simple direct-mapped SRAM structures. On the other hand, in the shared SRAM buffer, we need a mechanism to identify where exactly the $n\text{th}$ element of a given queue q_i is placed. Intuitively, this is similar to the design of Q linked lists in which the next cell to access by the arbiter is located at the head of the correspondent list and the next cell to store coming from the DRAM is placed at the tail of the correspondent list.

Fig. 12 shows two different alternatives for implementing a unified SRAM buffer, one of them targeted at achieving a very short access time (and, hence, suited to high-performance implementations) and the other targeted at achieving the lowest impact on area and power consumption (and, hence, suited to cost-effective implementations).

The *global CAM* design consists of a full content-addressable memory. In such memory, packety cells are stored out of order and can be indexed using a tag. Each cell's tag identifies the queue where that cell belongs and the relative order inside the list of cells of that same queue. When the address (queue identifier and order) of the cell is set, the CAM searches across all entries for the related cell. Note that we assume that the refreshes from the DRAM are serialized along B time slots at a rate of one cell per slot. This implementation requires one CAM port (to look for a given cell entry) and one write port (to allocate new entries). Additionally, the system requires a method to perform allocation of new cells in available entries. A very simple way of handling this is to implement a free-list as a direct-



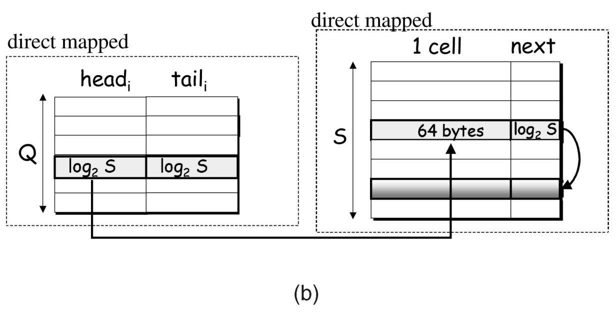


Fig. 12. SRAM design alternatives: (a) global CAM and (b) unified linked-list.

mapped circular buffer that holds indexes to unused entries of the *global CAM*. Such a structure would require one read port and one write port, and one head and tail pointer.

The functional timing behavior of the *global CAM* is as follows: Reading a new item implies CAMing the memory array for the specified item. This gives us the index of the entry just read so that we can write this index in the free-list as a new available entry. Writing a new item involves accessing the free-list to obtain a valid index to write. Once this index is obtained, we can do a direct-mapped write onto our CAM array.

Note that we could conceivably reduce the critical path of the access by overlapping the accesses to the CAM array and the free-list. In order to enable this, we would need a lookahead of one when obtaining a valid entry in the free-list and latch it. The drawback of this technique is that we may produce bubbles when the SRAM is full, which is actually a very unwanted artifact. Therefore, we will consider that the access to both structures needs to be serialized.

Finally, the *unified linked-list* proposal is a straightforward implementation of Q linked lists onto a direct-mapped memory structure. Each entry of that direct-mapped SRAM contains one cell and a pointer to the next cell (another entry of the same structure). In order to be able to identify the head and the tail of each linked list, we have another direct-mapped structure that stores the head and tail pointers for each of the queues. Of course, we also need a free-list to determine available entries in the SRAM array whenever a new item is written. The advantage of this implementation is that it does not require a complex CAM implementation. However, it requires one additional write operation to store the position of the new tail onto the pointer field of the old tail. As a result, the structure needs one read port and two write ports per structure (or having to perform three accesses with one single read/write port for a time-multiplexed configuration).

Since this design is targeted at low power and area impact, the access is performed time-multiplexed. That is, instead of having several read-write ports that can act in parallel, we have a single read/write port for each structure so that the set of accesses can be serialized. The lower the number of ports, the lower the area and power consumption of a SRAM array. The obvious downside of this alternative is that we considerably increase the overall time required to perform all the actions.

The functional timing behavior of the *unified linked-list* is as follows: Reading a new item involves first reading the head-tail array to determine the location of the head of a given queue. Then, we read this item from the SRAM array, which, at the same time, gives us the pointer of the new head, which is required to update the head-tail entry in the first array. At the same time, while the main SRAM array is accessed, since we know that the old head will become a new available entry, we can update the free-list (hence, this access is out of the critical path). Writing a new item involves, first, in parallel, reading from the head-tail array the current tail of the queue while reading a new available entry from the free-list. After that, two consecutive writing accesses to the main SRAM array are needed: one to write on, which was the current tail, the index of the new tail (which is given by the free-list), and one to write the actual new cells onto what has already become the new tail of the queue, i.e., the available entry obtained from the free-list.

8.1.1 SRAM Buffer Implications in GHDS Systems

For any GHDS configuration, we must observe two main issues. First, the cells coming from the DRAM memory system of a given queue may come out-of-order. Second, for CFDS, the SRAM must contain additional entries to be able to hold elements before they are scheduled by the MMA. The first problem can be easily overcome by implementing some basic changes to our proposed SRAM structures to allow them to insert cells from a queue out of its natural order:

- *global CAM*: Implementation of writing operations out-of-order is trivial in this configuration as only more bits are required in the ordinal tag field.
- *unified linked-list*: Out-of-order writing operations are complex inside a linked-list. However, an easy solution is to implement $Q \times M$ linked-lists instead of Q as M is the number of banks and two operations on the same bank are always performed in strict order. The selection of the subqueue can be easily performed with a round-robin mechanism per queue.

8.2 SRAM Buffer Performance

8.2.1 RADS SRAM Buffer Limitations

Fig. 13 shows the access time and the area of the different SRAM implementations for OC768 and OC3072 as a function of the number of slots of the lookahead. The required SRAM size depends on the size of the lookahead: It is maximum for $lookahead = 0$ and minimum for $lookahead = Q(B-1)+1$ (see [29]).

Note that, in the numbers of Fig. 13, we account for the effect of both t-SRAM and h-SRAM (the area is the combination of both, while the time is the most restrictive one).

For the area graphs, we also show the size of the SRAM structures in MB. The OC768 system SRAM size ranges between 600 kB (for the minimum lookahead) and 128 kB (for the maximum lookahead). The OC3072 system SRAM size ranges between 10.5 MB (for the minimum lookahead) and 4.0 MB (for the maximum).

For an OC768 system, we need to access a new cell every 12.8 ns (assuming 64 bytes wide cells). We can observe from Fig. 13 that the access times of all SRAM alternatives are far below that number, even for the shortest lookahead. Therefore, as access time is not a concern, we shall focus on those implementations with a minimum area. For instance, the *global CAM* implementation has an overall area of more than 0.4 cm^2 for lookaheads shorter than 100-200 slots, which could represent a significant fraction of the overall transistor budget of a medium cost system. On the other hand, the *time-mux unified linked list* exhibits an area smaller than 0.2 cm^2 with very small lookaheads and as low as 0.1 cm^2. For this reason, in the rest of the paper, we will target the *time-mux unified linked list* SRAM for all OC768 systems.

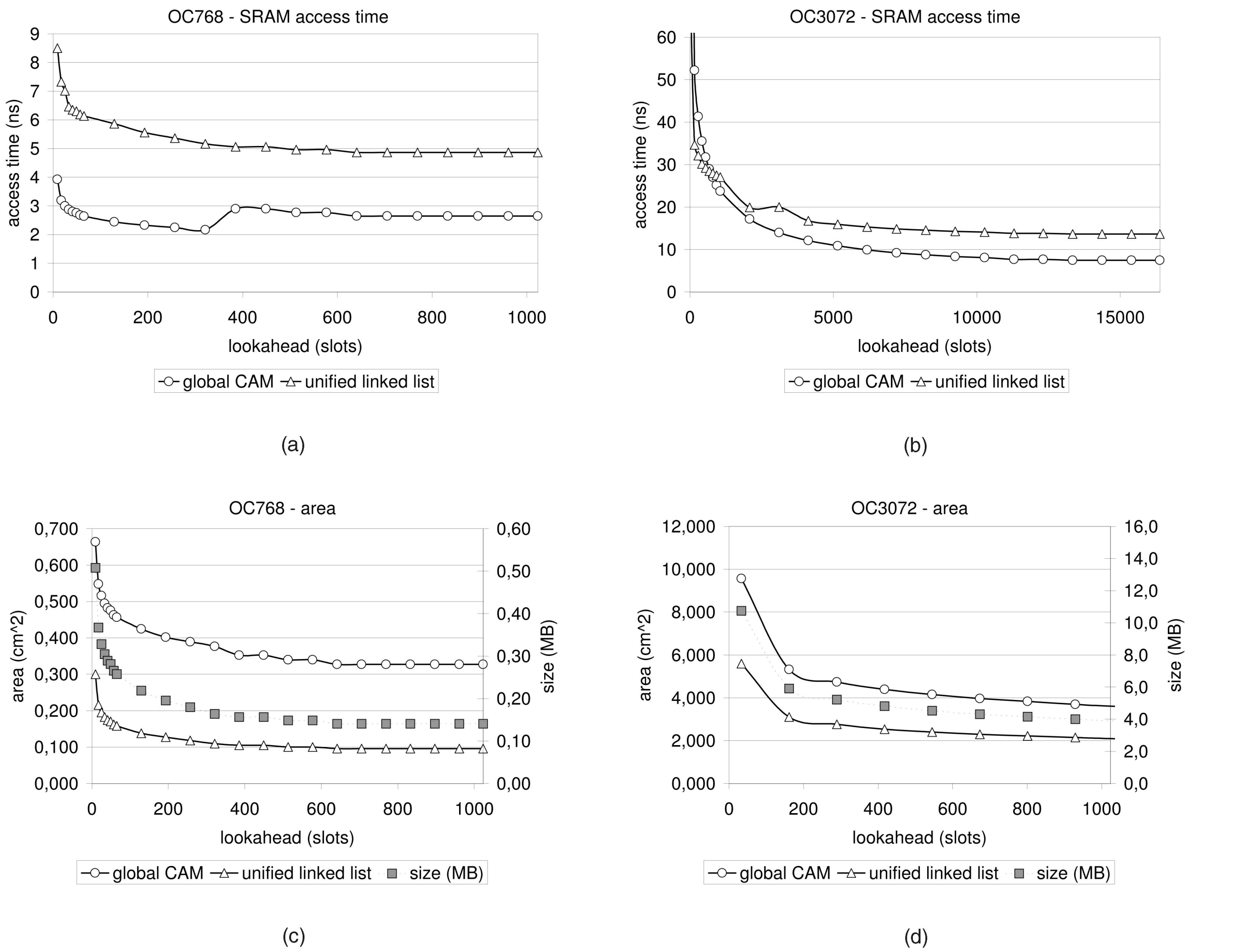


Fig. 13. h-SRAM area and access time as a function of the lookahead for the RAIDS scheme (Q = 128, B = 8 for OC768 and Q = 512, B = 32 for OC3072).

For an OC3072 system, we need to access a new cell every 3.2 ns, which is a significantly harder constraint to meet, taking into account that the SRAM buffers are now larger. Indeed, if we look at Fig. 13, we can clearly appreciate the fact that no SRAM implementation is able to comply with the 3.2 ns target (even for the longest lookaheads). The fastest implementation is the *global CAM*, which has an access time slightly higher than 7 ns, and that for extremely long lookahead delays. Furthermore, if we look at the area results for the different alternatives, we can observe that all the configurations exhibit areas larger than 2 cm^2, which could be another limiting factor of design even for the budget of high-end systems. As the access time is clearly the constraint for OC3072, for the rest of the paper we will use the *global CAM* implementation (the fastest) for evaluation purposes.

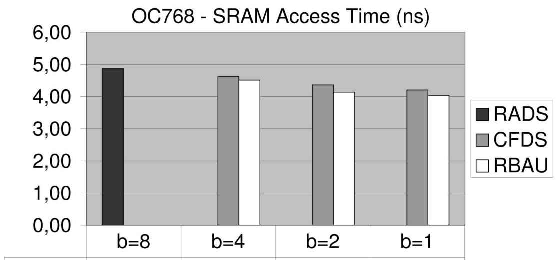
8.2.2 RBAU Performance Improvements

Fig. 14 allows us to demonstrate the performance benefits of using RBAU instead of RADS and CFDS. It shows the area (of both h-SRAM and t-SRAM) and most restricting access time for OC768 and OC3072 for the maximum lookahead according to the data granularity. Again, the number of queues Q is 128 for the OC768 system and 512 for the OC3072 system. We assume the number of banks M to be 256.

We should remember that RBAU should always leverage smaller areas and shorter latencies than CFDS for a given configuration since RBAU does not need the extra SRAM entries needed by CFDS, used to accommodate the bounded (yet, still significative) level or reordering of memory accesses. As shown previously, this improvement comes at the cost of a very small percentage of packet losses (when CFDS was targeted at granting zero packet losses).

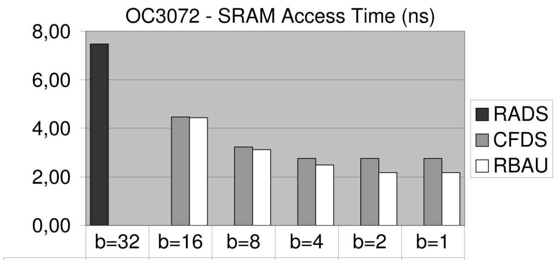
There are two main conclusions that can be inferred from the results in Fig. 14. First, one can see the evident advantages of either CFDS and RBAU relative to RADS. For an OC768 (in which area is the main factor of merit), a RBAU system with $b = 2$ achieves an area 2.3X smaller than RADS and 10 percent smaller than CFDS. For $b = 1$, the differences are even greater (4.1X compared to RADS, 1.4X compared to CFDS). Note that it is not always possible to keep decreasing the main memory data granularity as we may be ultimately limited by the main bus cycle time. However, for OC768, a DRAM cycle time of 12.8 ns would be perfectly affordable.

However, the especially significant gains are for the OC3072 system. An RBAU system with $b = 8$ is compliant



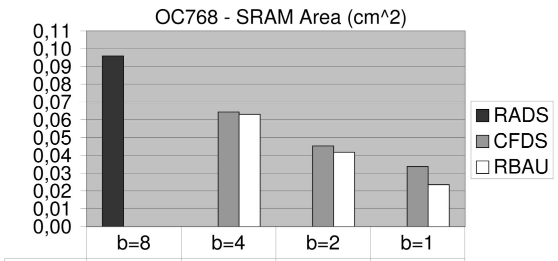
	b=8	b=4	b=2	b=1
■ RADS	4,87			
▨ CFDS		4,62	4,36	4,21
□ RBAU		4,51	4,13	4,04

(a)



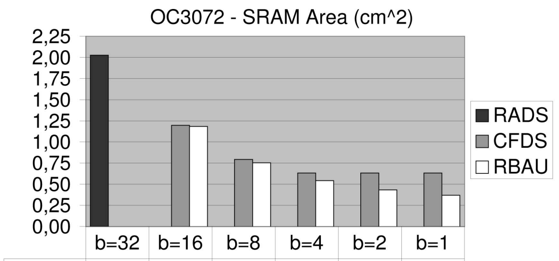
	b=32	b=16	b=8	b=4	b=2	b=1
■ RADS	7,47					
▨ CFDS		4,47	3,23	2,75	2,75	2,75
□ RBAU		4,44	3,12	2,49	2,17	2,17

(b)



	b=8	b=4	b=2	b=1
■ RADS	0,10			
▨ CFDS		0,06	0,05	0,03
□ RBAU		0,06	0,04	0,02

(c)



	b=32	b=16	b=8	b=4	b=2	b=1
■ RADS	2,03					
▨ CFDS		1,20	0,79	0,63	0,63	0,63
□ RBAU		1,18	0,75	0,54	0,43	0,37

(d)

Fig. 14. SRAM area (both h-SRMA and t-SRMA) and access time as a function of the granularity, for RADS, CFDS, and RBAU configurations.

with the requirements of buffering packets at 160 Gbps as *the access time is lower than 3.2 ns*. For instance, the CFDS system requires a lower granularity ($b = 4$) to match the same requirements, putting more pressure on the DRAM memory system performance (as it requires a shorter main bus cycle time).

Moreover, this is accomplished with an affordable area (less than 0.75 cm^2 overall and as low as 0.37 cm^2 if we keep decreasing the granularity to the minimum value). This contrasts strongly with its RADS counterpart, which is not able to access data in 7 ns (hence, not being compliant for an OC3072 design), even with a delay of more than 50 μs and the nonirrelevant area of 2 cm^2 (almost the size of the Pentium IV or a 90 nm version of IBM's cell chip, which has an die size of 214 mm^2).

The second important conclusion is that there is an optimal value of b for a given CFDS implementation. The reason for this is the trade-off between the SRAM size required to tolerate the unpredictability of arrivals from the arbiter—which is proportional to b—and the SRAM size required to absorb the level of reordering of the accesses from the DRAM—which is proportional to $1/b$—(see [17]).

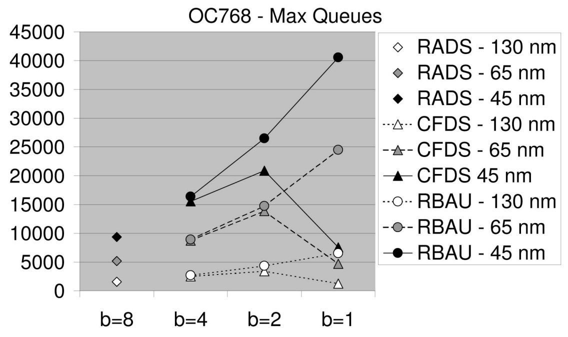
RBAU does not suffer from this shortcoming as it presents a trade-off between packet loss percentage and reordering size. At the cost of a very small probability of packet losses, we can still keep reducing the data granularity to match the timing and bandwidth specifications.

8.2.3 Scalability for Future Implementations

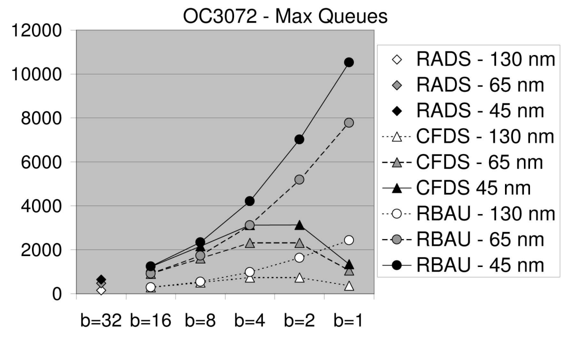
A very interesting experiment is to determine the scalability of our proposed system for a near future scenario. There are two parameters that determine whether a given GHDS implementation is compliant with the timing targets of a packet buffer. The first one is the characteristics of the link rate (the packed bandwidth), while the other is the technological process used to implement the SRAM buffers.

In order to extrapolate the future scalability, we measured the maximum number of queues tolerated by GHDS, varying three different parameters: the granularity b, the link rate, and the technological process. For the link rates, we selected, in addition to OC768 and OC3072, a near future value of OC12288 (640 Gbps). For the technological processes, in addition to 130 nm, we selected the close to current state-of-the-art 60 nm process (to be featured next year) and the following one (45 nm). Note that we did not consider 90 nm, which is the current state-of-the-art process, as we are interested in future projections.

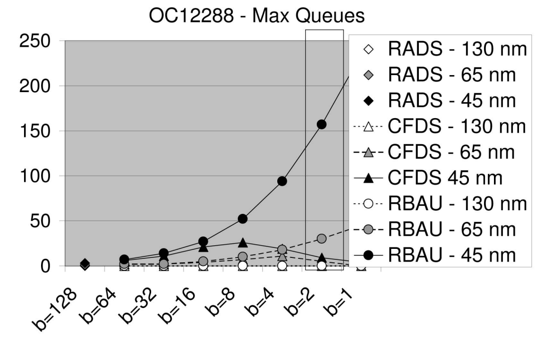
Fig. 15 shows the maximum number of queues that the different SRAM buffer approaches can afford, taking into account the access time constraints (12.8 ns of maximum delay for OC768, 3.2 ns for OC3072, and 0.8 ns for OC12288). The graphs also show the effect of using different technological processes (130, 60, and 45 nm). It is expected that the higher the link rate, the lower the number of queues. At the same time, the more advanced the technological process, the higher the number of queues. On the other hand, for CFDS, we should expect a trade-off



(a)



(b)



(c)

Fig. 15. Maximum number of queues for OC768, OC3072, and OC12288.

in the granularity b as too low a granularity may produce too high a level of disorder between transactions that override the granularity reduction benefits. Such a trade-off does not exist for the more general GHDS since we are no longer looking for a zero packet loss rate. However, we should take into account that some values of b cannot be feasible for certain link rates as the limiting factor ends up being the DRAM main memory system cycle time. For instance, assuming a DDR3 memory system at 1 GHz, we would not be able to implement an OC12288 system with $b = 1$ as the required cycle time would be 0.8 ns.

As shown in the figure, RBAU (compared to RADS) allows around four more queues for OC768, around 16 times more queues for OC3072, and around 50 times more queues for OC12288. The differences remain quite constant for the different technology processes. Additionally, RBAU, compared to CFDS, allows around two times more queues for OC768, around three times more queues for OC3072 and, finally, around six times more queues for OC12288 (with a realistic granularity of $b = 2$).

Another very interesting consequence that we may observe from the graphics is that, if we have a target of around 2,000 queues, 130 nm is good enough for OC768, while 65 nm is good enough for OC3072. On the other hand, we may observe that there is a dramatic change in behavior for OC12288. We may observe that, even for the best technological process (45 nm), the number of maximal queues we can handle is low (less than 32 for CFDS and less than 200 for RBAU and a reasonable granularity). This last observation poses the interesting challenge of designing a new system that is able to handle hundreds of queues for OC12288 as it is clear that RBAU benefits reach their limits due to the advance of link rate technology and the CMOS scalability problems. Ultimately, it becomes evident that the performance of high-performance routers may become limited by the packet buffer performance in a span of five years.

9 Related Work

Virtual Output Queuing was proposed for first time in [43] (with the name of "dynamically-allocated multi-queue buffers"). The amount of buffering and the line rates considered in this seminal paper were far lower than those required for our target application: high-speed backbone routers. For OC192 (10 Gbps) line rates, a time slot is lower than the random access time of DRAM. Nikologiannis and Katevnis [37] propose a design using DRAM only for a VOQ buffer architecture working at this line rate. The proposed design uses out-of-order memory access in order to reduce the number of bank conflicts, although it does not guarantee zero miss loses. Hasan et al. [22] propose techniques that exploit row locality whenever possible in order to enhance the average-case DRAM bandwidth. However, this scheme may have a significant miss probability for special traffic patterns.

For faster line rates, a hybrid SRAM-DRAM implementation of a VOQ buffer using ECQF for the h-MMA is discussed in [29]. This is the starting point of our work.

There are many proposals that exploit the bank organization of DRAM memory [40], [46], [27]. This is especially true in the vector processor domain. The novelty of our proposal resides in the application of this technique to the context of fast packet buffering. For our RBAU system, we assumed a theoretically ideal random hashing mechanism for distributing cells across memory banks. In a practical implementation, we could use many of the proposed hashing mechanisms found in the literature.

10 Conclusions

In this paper, we have proposed a general design for hybrid SRAM/DRAM packet buffers. We have shown that two previously proposed hybrid SRAM/DRAM packet buffer designs ([29] and [17]) can be seen as particular cases of our general scheme.

Based on this general scheme, we have proposed a Random BAU (RBAU) Scheme that randomly chooses a DRAM memory bank for every transfer between SRAM and DRAM. The numerical results show that this scheme would require an SRAM size almost an order of magnitude lower than the scheme given in [29], without the memory fragmentation problem of the scheme proposed in [17].

Although the RBAU Scheme proposed in this paper does not have a zero miss probability, the results support the conclusion that the randomization process among memory banks allows an extremely low miss probability to be guaranteed *for any traffic pattern*. We think that our design may be useful for building very large and fast future packet switches.

We are planning to use Cacti 4.0 (since a near release has been recently announced) to add even better tech process projections and include also power consumption numbers.

Acknowledgments

This work was supported by the Ministry of Education of Spain under grants TEC2004-06437-C05-05 and TIN2004-07739-C02-01, by the European NoE EuroNGI, and by a grant from IBM.

References

[1] Altera, "Linear Feedback Shift Register Megafunction," Dec. 2001, http://www.altera.com/literature/sb/sb11_01.pdf.

[2] G. Appenzeller, I. Keslassy, and N. McKeown, "Sizing Router Buffers," *ACM SIGCOMM,* Aug. 2004.

[3] S.B. Murkherjee, D. Danerjee, and A. Murkherjee, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Multiple-Node CAS," *IEEE/ACM Trans. Networking,* vol. 4, pp. 684-696, 1996.

[4] V. Bollapragada, R. White, and C. Murphy, *Inside Cisco IOS Software Architecture.* Cisco Press, July 2000.

[5] C. Chang, D. Lee, and Y. Jou, "Load Balanced Birkhoff-von Neumann Switches, Part I: One-Stage Buffering," *Computer Comm.,* vol. 25, pp. 661-622, 2002.

[6] Y. Chen and J. Turner, "WDM Burst Switching For Petabit Capacity Routers," *Proc. Military Comm. Conf. (MILCOM),* pp. 968-973 1999.

[7] Cisco, "Cisco Carrier Router System," http://www.cisco.com/en/US/products/ps5763/index.html, 2005.

[8] J. Corbal, R. Espasa, and M. Valero, "Command-Vector Memory System," *Proc. IEEE Parallel Architectures and Compilation Techniques (PACT '98),* Nov. 1998.

[9] C. Crisp, "Provisioning Internet Backbone Networks to Support Latency Sensitive Applications," PhD dissertation, Stanford Univ., June 2002.

[10] R. Crisp, "Direct Rambus Technology: The New Main Memory Standard," *IEEE Micro,* vol. 7, pp. 18-28, Nov./Dec. 1997.

[11] A. Dhamdhere, H. Jiang, and C. Dovrolis, "Buffer Sizing for Congested Internet Links," *Proc. IEEE Infocomm,* Mar. 2005.

[12] M.D.K. Hunter and I. Andonovic, "Buffering in Optical Packet Switches," *J. Lightwave Technology,* vol. 16, pp. 2081-2094, Dec. 1998.

[13] W. Eatherton, "Router/Switch Architecture with Networking Specific Memories," *Proc. Memory, Storage, and Serial Interface Technology Conf. (MemCon 2002),* Oct. 2002.

[14] Fujitsu, "256M bit Double Data Rate FCRAM, MB81N26847B/261647B-50/-55/-60 data sheet," http://www.fujitsu.com, 2004.

[15] J. Gabriagues and J. Jacob, "OASIS: A High-Speed Photonic ATM Switch—Results and Perspectives," *Proc. 15th Int'l Switching Symp.,* pp. 457-461, Apr. 1995.

[16] J. García, L. Cerdà, and J. Corbal, "A Conflict-Free Memory Banking Architecture for Fast Packet Buffers," Technical Report UPC-DAC-2002-50, Politechnic Univ. of Catalonia, July 2002, http://www.ac.upc.es/recerca/reports/DAC/2002.

[17] J. García, J. Corbal, L. Cerdà, and M. Valero, "Design and Implementation of High-Performance Memory Systems for Future Packet Buffers," *Proc. MICRO '03,* Dec. 2003.

[18] J. García, M. March, L. Cerdà, J. Corbal, and M. Valero, "On the Design of Hybrid DRAM/SRAM Memory Schemes for Fast Packet Buffers," *Proc. IEEE High Performance Switching and Routing (HPSR),* pp. 15-19, Apr. 2004.

[19] G. Iannaccone, M. May, and C. Diot, "Aggregate Traffic Performance with Active Queue Management and Drop from Tail," *SIGCOMM,* vol. 37, pp. 277-306, 2001.

[20] G. Glykopoulos, "Design and Implementation of a 1.2 Gbit/s ATM Cell Buffer Using a Synchronous DRAM Chip," Technical Report 221, ICS-FORTH, July 1998, http://www.ii.uib.no/~markatos/avg/publications.html.

[21] C. Guillermot, "Transparent Optical Packet Switching: The European ACTS KEOPS Project Approach," *J. Lightwave Technology,* vol. 16, no. 12, pp. 2117-2133, Dec. 1998.

[22] J. Hasan, S. Chandra, and T. Vijaykumar, "Efficient Use of Memory Bandwidth to Improve Network Processor Throughput," *Proc. 30th Ann. Int'l Symp. Computer Architecture (ISCA),* June 2003.

[23] Hitachi, "Hitachi 166 Mhz SDRAM," *Hitachi HM5257XXb series,* 2000, http://semiconductor.hitachi.com/dram/dram_modules.htm.

[24] D. Hunter, "WASPNET: A Wavelength Switched Packet Network," *IEEE Comm. Magazine,* pp. 120-129, Mar. 1999.

[25] D. Hunter, W. Cornwall, T. Gilfedder, A. Franzen, and I. Andonovic, "SLOB: A Switch with Large Optical Buffers for Packet Switching," *IEEE/OSA J. Lightwave Technology,* vol. 16, no. 10, Oct. 1998.

[26] A.F.I. Chlamtac and T. Zang, "Lightpah Routing in Large WDM Networks," *IEEE J. Selected Areas in Comm.,* vol. 14, pp. 909-913, June 1996.

[27] D. Harper and J. Jump, "Performance Evaluation of Vector Accesses in Parallel Memories Using a Skewed Storage Scheme," *Proc. 13th Int'l Symp. Computer Architecture (ISCA),* pp. 324-328, 1986.

[28] Infineon Technologies, "RLDRAM. High Density, High-Bandwidth Memory for Networking Applications,"http://www.infineon.com, 2004.

[29] S. Iyer, R. Kompella, and N. McKeown, "Designing Buffers for Router Line Cards," Technical Report TR02-HPNG-031001, Stanford Univ., Nov. 2002, http://klamath.stanford.edu/~sundaes/publications.html.

[30] Juniper, "Juniper T640," http://www.juniper.net/products/tseries, 2005.

[31] J.S.K. Dolzer, C. Gauger, and S. Bodamer, "Evaluation of Reservation Mechanisms for Optical Burst Switching," *AEU Int'l J. Electronics and Comm.,* vol. 55, no. 1, 2001.

[32] M. Karol, "Shared-Memory Optical Packet (ATM) Switch," *Multigigabit Fiber Comm. Systems,* vol. 2024, pp. 212-222, 1993.

[33] I. Keslassy, S. Chuang, and N. Mckeown, "Architectures and Algorithms for a Load-Balanced Switch," Technical Report TR03-HPNG-061501, Stanford Univ., June 2003.

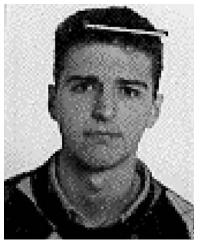
[34] I. Keslassy, S. Chuang, and N. McKeown, "A Load-Balanced Switch with an Arbitrary Number of Linecards," *Proc. IEEE Infocom,* Mar. 2004.

[35] I. Keslassy, S. Chuang, K. Yuu, D. Miller, M. Horowitz, O. Solgaard, and N. McKeown, "Scaling Internet Routers Using Optics," *ACM SIGCOMM,* Aug. 2003.
[36] C. Minkenberg, R. Luijten, W. Denzel, and M. Gusat, "Current Issues in Packet Switch Design," *Proc. HotNets-I,* Oct. 2002.
[37] A. Nikologiannis and M. Katevenis, "Efficient Per-Flow Queueing in DRAM at OC-192 Line Rate Using Out of Order Execution Techniques," *Proc. IEEE Int'l Conf. Comm.,* June 2001.
[38] C. Quiao and M. Yoo, "Optical Burst Switching (OBS): A New Paradigm for an Optical Internet," *J. High Speed Networks,* vol. 8, no. 1, pp. 69-84, 1999.
[39] R. Ramaswami and K.N. Sirvajan, *Optical Networks.* San Mateo, Calif.: Morgan Kaufman, 1990.
[40] B. Rau, M. Schlansker, and D. Yen, "The Cydra 5 Stride-Insensitive Memory System," *Proc. Int'l Conf Parallel Processing,* pp. 242-246, 1989.
[41] P. Shivakumar and N. Jouppi, "Cacti 3.0: An Integrated Cache Timing, Power and Area Model," technical report, Compaq Computer Corp., Aug. 2001, http://research.compaq.com/wrl/people/jouppi/CACTI.html.
[42] R. Spanke, "Architectures for Large Nonblocking Optical Space Switches," *IEEE J. Quantum Electronics,* vol. 22, no. 6, pp. 964-967, June 1986.
[43] Y. Tamir and G. Frazier, "High-Performance Multi-Queue Buffers for VLSI Communication Switches," *Proc. 15th Int'l Symp. Computer Architecture (ISCA),* pp. 343-354, May 1988.
[44] S. Tariq, M. Dhodhi, J. Palais, and R. Ahmed, "Next Generation DWDM Networks: Demands, Capabilities and Limitations," *Proc. Canadian Conf. Electrical and Computer Eng.,* pp. 1003-1007, 2000.
[45] J. Turner, "Terabit Burst Switching," *J. High Speed Networks,* vol. 8, pp. 3-6, 1999.
[46] M. Valero, T. Lang, J. LLaberia, M. Peiron, E. Ayguade, and J. Navarro, "Increasing the Number of Strides for Conflict-Free Vector Access," *Proc. 19th Int'l Symp. Computer Architecture (ISCA),* pp. 372-381, May 1992.
[47] M. Valero, T. Lang, M. Peiron, and E. Ayguade, "Increasing the Number of Conflict-Free Vector Access," *IEEE Trans. Computers,* vol. 44, no. 5, pp. 634-646, May 1995.
[48] D. Wischik and N. McKeown, "Part I: Buffer Sizes for Core Routers," *Computer Comm. Rev.,* vol. 35, no. 3, pp. 75-78, 2005.
[49] Xilinx, "Pseudo Random Number Generator," Dec. 2001, http://www.xilinx.com/xcell/xl35/xl35_44.pdf.
[50] C.Q.Y. Chen and X. Yu, "Optical Burst Switching (OBS): A New Area in Optical Networking Research," *IEEE Network Magazine,* vol. 18, no. 4, pp. 16-23, 2004.
[51] W. Zhong and R. Tucker, "Wavelength Routing-Based Photonic Packet Buffers and Their Applications in Photonic Packet Switching Systems," *J. Lightwave Technology,* vol. 16, no. 10, pp. 1737-45, Oct. 1998.

Jorge García-Vidal graduated as a telecomunications engineer in 1988 from the Polytechnic University of Catalonia (UPC), Barcelona, and received the PhD degree in telecommunications (UPC, 1992, Best UPC Telecommunication Thesis Award). During 1992-1993, he was a visiting scientist at the University of Arizona. Since 2003, he has been a full professor in the Computer Architecture Department of UPC. His current research interests are ad hoc and sensor networks, design of networking equipment, and performance evaluation of computer systems. He is a member of the IEEE.

Maribel March received the engineering degree in computer science in 2002 from the Polytechnic University of Catalonia (UPC). The same year, she joined the Computer Architecture Department of UPC as an assistant professor. Her PhD advisor is Jorge García-Vidal and her PhD co-advisor is Mateo Valero. Her thesis is about high-speed switching and routing. Mainly, she is working o the design of fast packet buffers.

Llorenç Cerdà received the engineering degree in telecommunications in 1993 from the Technical University of Catalonia (UPC). He joined the Computer Architecture Department of UPC in 1994, where he received the PhD degree in 2000 and, currently, he is an assistant professor. He has published a substantial number of papers and participated in several industry and EU funded research projects in the fields of ATM, IP micromobility, wireless networks, and fast packet buffers. He is a member of the IEEE.



Jesús Corbal received the MS degree in electrical engineering from the Universitat Politecnica de Catalunya in 1997. He went on to receive the PhD degree in computer science from the Department of Computer Architecture at the same university in 2002. He worked as an assistant teacher for the Department of Computer Architecture in Barcelona for three years, focusing his research on vector architectures, memory system design and multimedia acceleration. He is currently working for BSSAD, VSSAD (DEG) at Intel Labs Barcelona.

Mateo Valero received the PhD degree from the Polytechnic University of Catalonia (UPC) in 1980. He is a professor in the Computer Architecture Department at UPC. His research interests focus on high-performance architectures. He has published approximately 400 papers on these topics. He is the director of the Barcelona Supercomputing Center, the National Center of Supercomputing in Spain. Dr. Valero has been honored with several awards, including the King Jaime I by the Generalitat Valenciana and the Spanish national award "Julio Rey Pastor" for his research on IT technologies. In 2001, he was appointed a fellow of the IEEE, in 2002, an Intel Distinguished research fellow and, in 2003, a fellow of the ACM. In 1994, he became a founding member of the Royal Spanish Academy of Engineering. In 2005, he was elected Correspondant Academic of the Spanish Royal Academy of Sciences, and his native town of Alfamén named their public college after him.