

Analysis of a High Performance DRAM/SRAM Memory Scheme for Fast Packet Buffers

Maribel March, Jorge García, Llorenç Cerdà, Jesús Corbal, and Mateo Valero
Polytechnic University of Catalonia - Computer Architecture Dept.

{mmarch,jorge,llorenc,jcorbal,mateo}@ac.upc.es

Abstract—In this paper we address the design of a packet buffer for future high-speed routers that support line rates as high as OC-3072 (160 Gb/s), and a high number of ports and service classes.

We describe a general design for hybrid DRAM/SRAM packet buffers that exploits bank organization of DRAM. This novel General Hybrid DRAM/SRAM Architecture includes some designs previously proposed as particular cases.

Furthermore, we propose a new algorithm that randomly chooses a DRAM memory bank for every transfer between SRAM and DRAM. The numerical results show that not only the required SRAM size should be almost an order of magnitude lower than previously proposed schemes, but also DRAM does not have the problem of fragmentation.

I. INTRODUCTION

Nowadays, the fastest available high-speed routers support up to 16 interfaces at OC-192 (10 Gb/s) or OC-768 (40 Gb/s) line rates. However, it is devised that next generation high-end systems will support a much larger number of interfaces (e.g. 624 or more) at OC-192, OC-768 or even OC-3072 (160 Gb/s) line rates [6].

Packet buffers for next generation routers will require a storage capacity for several Gb (giga bits) of data and a bandwidth of several hundreds of Gb/s. Usually these packet buffers will support Virtual Output Queueing (VOQ), which means that they must manage internal data structures of almost one thousand queues. Moreover, the design must be able to handle any input pattern, and not only traffic patterns that can be present in average. This restrictive condition is usual in networking equipment and leads to design choices that optimize the worst case and not only the most common case. Currently proposed packet buffer architectures do not meet these strict requirements.

To our knowledge, the fastest packet buffers with worst-case bandwidth guarantees that can be found in the literature are hybrid SRAM/DRAM designs. This memory organization was first proposed by the research group of N. McKeown (see [5]). Although the scheme proposed in [5] ensures zero loss probability for cells coming to a non full buffer, the required SRAM size for a large number of

Jesús Corbal is currently working at BSSAD, ILB, INTEL.

This work was supported by the Ministry of Education of Spain under grants TIC-2001-0956-C04-01 and TIC-2001-995-C02-01 and by the Dep. of Univ. (Generalitat de Catalunya) under grant CIRIT 2001-SGR-00226 and by a grant from IBM.

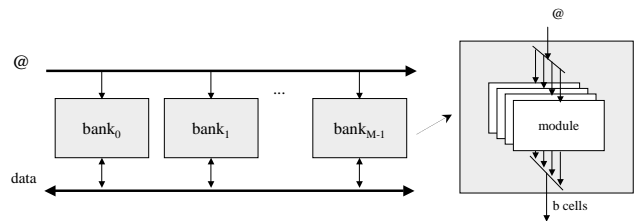


Fig. 2. Organization of a DRAM in banks: configuration of a DRDRAM-style memory and internal structure of a memory bank.

interfaces becomes too large. In [1] we described an scheme which aims for reducing the SRAM size of [5] while supporting a larger number of interfaces. The scheme we proposed in [1] is based on the observation that the effective DRAM access time can be reduced overlapping multiple accesses to different banks. This fact allows us to reduce the granularity of the accesses and thereby also reducing the SRAM size. Nevertheless, the memory scheme presented in [1] had the drawback of DRAM memory fragmentation, i.e. certain traffic patterns would lead to a situation where only a fraction of DRAM memory can be used. In [2] we introduced a *renaming of queues* scheme that reduces the probability of DRAM memory fragmentation. However, since traffic patterns are unpredictable, it is not possible to assess the probability of DRAM memory fragmentation.

Our novel proposal presented in this paper maintains the hybrid SRAM/DRAM design of [5] and [1], but introduces the following changes: (i) We redesign the functional blocks that govern SRAM/DRAM memory transfers. Our proposal is a *general* hybrid SRAM/DRAM design such that [5] and [1] schemes are particular cases. (ii) We propose a new algorithm that reduces the SRAM size of the scheme proposed in [5] almost by an order of magnitude. Furthermore, this new algorithm avoid the memory fragmentation problem of the scheme proposed in [2].

II. GENERAL HYBRID DRAM/SRAM ARCHITECTURE

Figure 1 shows a general hybrid DRAM/SRAM architecture. The system consists of (i) two fast but costly SRAM memory modules (t-SRAM and h-SRAM) (ii) a slow but low cost DRAM memory and (iii) the functional blocks that govern the transfers between DRAM/SRAM memory modules.

The t-SRAM and h-SRAM respectively cache the *tail* and *head* of each VOQ logical queue. The rest is stored

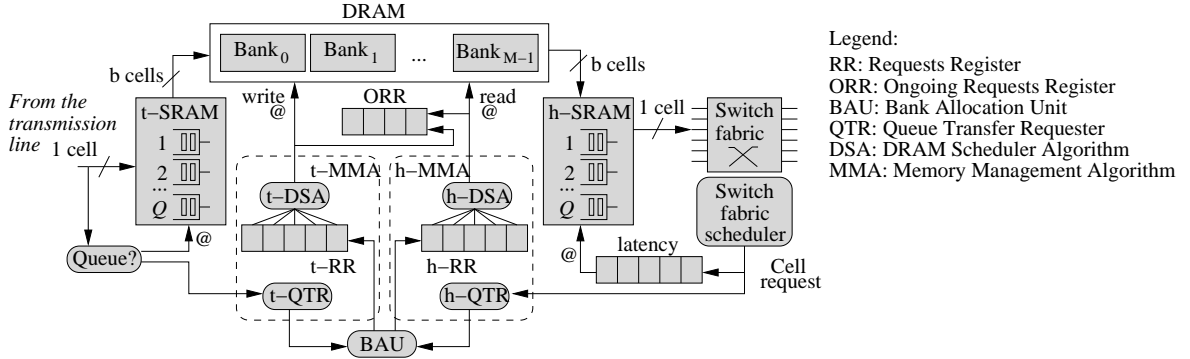


Fig. 1. General Hybrid DRAM/SRAM memory architecture.

in DRAM. 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 (we shall refer to this time as *time slot*).

The DRAM memory is organized in M banks. Figure 2 illustrates the concept of a memory bank. 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 elements we want to address. This size measured in cells is the *data granularity*. We shall refer to the granularity used in our scheme as b . In order to match DRAM/SRAM access times, transfers between DRAM and SRAM occur in batches of b cells.

The bank organization of DRAM memory can be exploited as follows. Let us define B as the minimum granularity that can be used if we require a random transfer from anyone of both SRAM and any DRAM memory bank. In this case, B is limited by the DRAM random access time (T). E.g. if the link rate is R bps and the cell size is C bits, we would have: $B \geq 2TR/C$ (we use $2T$ because each B time slots we have to do one read and one write transfer to DRAM). Then, given an array of M memory banks, 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 M times faster rate). However, we need to guarantee that the same bank is not accessed twice within its random access time. A *bank conflict* occurs when this constraint cannot be fulfilled.

In the following we shall explain how accesses to DRAM are managed. First, we focus on cell transfers from the t-SRAM to DRAM and after that, from DRAM to the h-SRAM.

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 Figure 1): 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), are shared by both t-MMA and h-MMA described later in this section.

The functional blocks of the t-MMA work as follows: When a cell for queue i arrives from the transmission line, 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 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 where cells should be allocated (the algorithm will be discussed in section IV). The request issued by the BAU contains the queue from which b cells must be transferred, and the bank where these cells will be placed. The request is stored in the *tail Request Register* (t-RR). Finally, every b time slots the *tail DRAM Scheduler Algorithm* t-DSA selects one of the transfer requests pending on the t-RR and issues it to DRAM. As a result, the transfer from the t-SRAM to DRAM is initiated. In order to choose one of the pending transfers in the t-RR, the t-DSA may take into account the banks that are being accessed (in order to avoid bank conflicts). The identifiers of these banks are stored in the *Ongoing Requests Register* (ORR).

In the explanation above we have described the transfers between the t-SRAM and DRAM. In the following we shall focus on the transfers between the h-SRAM and DRAM. These transfers 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 by the *switch fabric scheduler* (we shall refer to it simply as the *scheduler*). Otherwise, the cell requested by the scheduler 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-QTR algorithm is simple: Schedule a transfer for queue i whenever the number of requests for cells belonging to queue i , issued by the scheduler, exceeds the number of cells from this queue presents in the h-SRAM. Equivalently, let C_i^h be a counter of the number of cells requested by the scheduler from queue i (C_i^h is initialized to 0). Each time the scheduler requests a cell from queue i , C_i^h is increased and the t-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 since the t-QTR schedules a transfer until the corresponding upload of b cells from t-SRAM to DRAM is finished.

The rest of functional blocks of the h-MMA works analogously to those of the t-MMA. Nevertheless, we need an additional *latency* register (see Figure 1). This register introduces a delay since the scheduler issues a request 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 zero miss probability, the delay introduced by the latency register should be equal to the *maximum* response time of the h-MMA.

III. EXTENSION OF THE GENERAL MODEL TO PREVIOUS DRAM/SRAM SCHEMES

In this section we show that previously proposed DRAM/SRAM schemes are particular cases of the general model introduced in section II.

The simplest BAU scheme appears when we choose $b = B$. In this case there are not bank conflicts, and no specific BAU is needed (i.e. consecutive accesses to DRAM can be done to any bank). In this case, the DSA can be seen as a FIFO scheduler, which alternatively chooses the oldest write and the oldest read stored into the RR. This scheme is equivalent to the so called *Early Critical Queue First* (ECQF) proposed in [5]. 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.

In [2] we proposed the following scheme: Organize the M DRAM banks in $G = M/(B/b)$ groups of B/b banks. Assign Q/G queues to each group of banks. After that, store each batch of b cells belonging to the same queue in a round-robin fashion among banks belonging to the group where the queue was assigned. The DSA must choose again the oldest eligible request in the RR. In [2] we explain how to dimension the scheme in order to have zero miss probability. Furthermore, we show that using a granularity $b < B$ leads to smaller SRAM sizes than the scheme proposed in [5].

A drawback of the scheme previously described is that the assignment of the queues to the memory groups may prevent the full usage of the DRAM. This problem is called *memory fragmentation*. In [2] we alleviated it using a renaming of queues mechanism. In spite of that, memory fragmentation could still arise for certain traffic patterns.

IV. RANDOM BAU SCHEME

In this section we describe a scheme that exploits DRAM bank organization as in [2] (see section III). It allows a data granularity $b < B$, and thus, it reduces the SRAM size. Moreover, the scheme described in this section does not have the memory fragmentation problem of [2].

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 -th request for the i -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 $r_{n-\frac{B}{b}+1}^i, \dots, r_{n-1}^i, r_n^i$ 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. By this way, we avoid bank conflicts in transfers of cells from the same queue (as we see in section II, 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.

V. SRAM DIMENSIONING

In this section we describe some dimensioning guidelines for the SRAM modules used in our General Hybrid SRAM/DRAM Architecture for the Random BAU mechanism explained in IV.

Let us first assume $b = B$. As we explained in section III, there are never bank conflicts and the scheme is equivalent to the ECQF mechanism proposed in [5]. The required sizes for both SRAM and the latency register are $Q(B-1)+1$. The proof of this result can be found in [5].

Now let us consider an scenario with a granularity $b < B$, where bank conflicts may occur. Independently on the BAU scheme used, the system must be able to handle potential misses on any DSA. 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 in that moment, and thus, 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 receives requests every slot and the t-DSA only runs every b slots, the system cannot recover from this loss. Figure 3 is an example that shows this behaviour. Note that the probability of having a new miss is lower, because there is one more element to choose in the t-RR.

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

Let us consider now 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 issues Q consecutive requests addressed to different queues and all the requested cells are stored in the same DRAM memory bank. If we want to guarantee zero miss probability, we would need an h-SRAM, t-SRAM and Latency Register of size $Q(B-1) + 1$. However, given the random bank assignment policy used by the random BAU scheme, the probability of the former event can be extremely low ($\frac{1}{M^Q}$) even for the worst case traffic pattern. In order words, it is plausible to assume that the event leading to the maximum response time ($Q(B-1) + 1$) using the random BAU scheme is very unlikely to happen. In fact, in the next section we show that, for practical purposes, the system can be dimensioned as no bank conflicts occur, i.e, assuming a MMA maximum response time of $Q(b-1) + 1$.

VI. NUMERICAL RESULTS

In this section we analyze the Random BAU Scheme described in section IV. For the results shown here, we use the worst case scenario described in section V: the t-MMA and h-MMA respectively receive a sequence of cell arrivals and scheduler requests in a round robin for queues $1, 2, \dots, Q$. In response to this pattern, the t-QTR and the h-QTR generate periodic bursts of transfer requests for queues $1, 2, \dots, Q$.

For h-SRAM dimensioning purposes, the key parameter to study is the h-MMA maximum response time (see section V. Remember from section V that it would be the same for the t-MMA. Figure 4.(A) shows the Complementary Distribution Function (CDF) of the MMA response time under the load conditions previously described for $M = 256$, $B = 32$, $b = 1$ and different values of Q . The three lines are almost coincident, indicating that, in this case, the delay

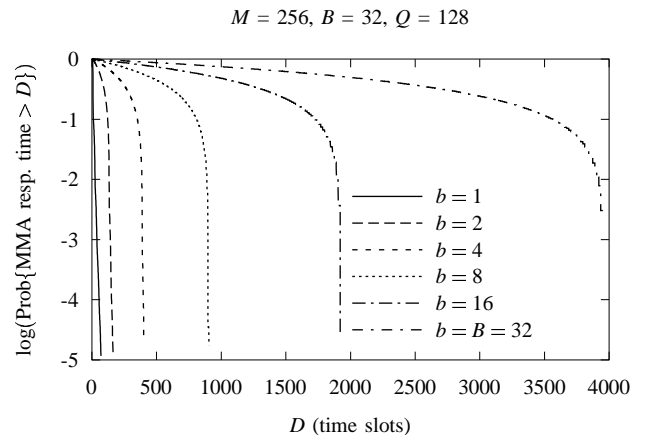


Fig. 5. Complementary distribution function of the MMA response time for $Q = 128$ and different values of b .

is independent of Q . In Figure 4.(B) we show the results for $b = 8$ and different values of Q . Note that when $b > 1$, the response time of the MMA depends on the number of queues.

Figure 5 shows the results obtained for a fixed value of Q and different values of b . As we see, for values of b upper than 2, the curves have a decreasing sharp at point $Q(b-1) + 1$. This indicates that it is very unlikely that the response time of the h-MMA is larger than $Q(b-1) + 1$.

Figure 6.(A) shows the behavior for $Q = 1024$, $B = 32$, $b = 8$ and different values of M . As we see, the lines associated to values between 256 and 8 are almost coincident, indicating that the delay is independent of the number of banks used when it is upper than 4. Figure 6.(B) shows the same for $b = 2$. In this case, the low bound of number of banks is 16. Note that when we reduce the value of b , this limit increase. These results show that increasing the number of banks on DRAM from a certain value does not influence the DRAM performance in terms of delay.

The previous numerical results show that $Q(b-1) + B$ is a plausible dimensioning rule for the t-SRAM and $Q(b-1) + 1$ for the h-SRAM and the latency register in the Random BAU Scheme. Provided that we can build a fast enough MMA unit, this size can be almost an order of magnitude lower than the one that would be required using the design given in [5]. Furthermore, the Random BAU Scheme does not have the DRAM fragmentation problem of the design we proposed in [2].

VII. RELATED WORK

Virtual Output Queuing was proposed for first time in [9] (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 Gb/s) line rates, a time slot is lower than the random access time of DRAM. [7] proposes

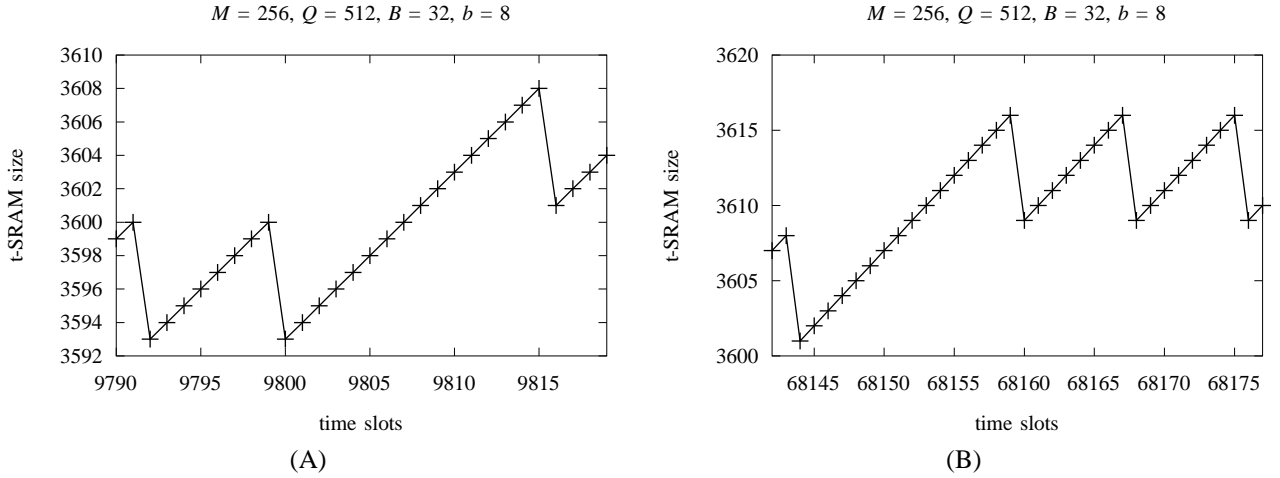


Fig. 3. Example of the h-SRAM occupancy when a first miss (A) and a second miss (B) occurs and no speed-up is introduced (for $M = 256$, $Q = 512$ and $B = 32$ and $b = 8$).

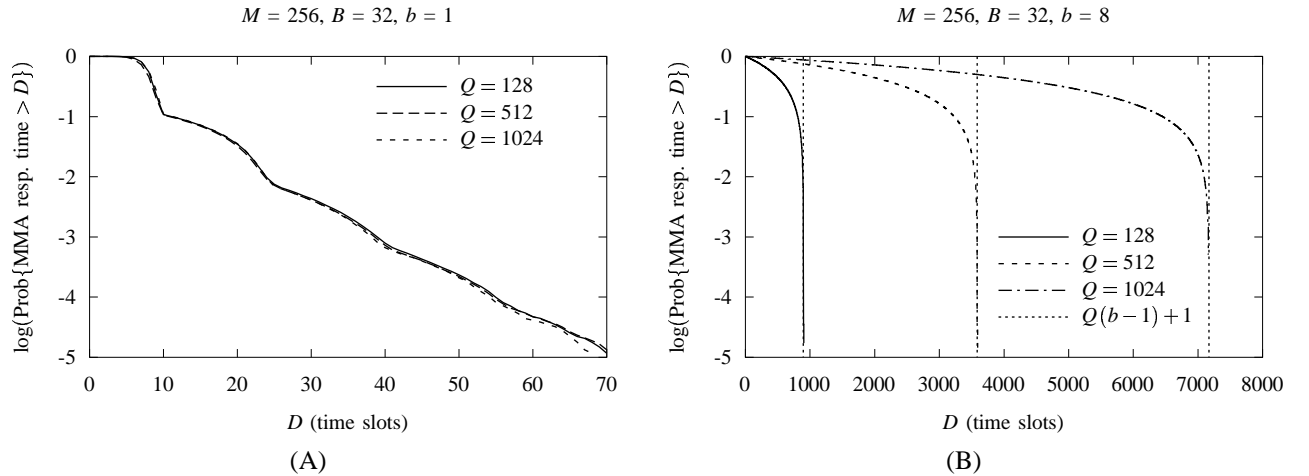


Fig. 4. Complementary distribution function of the MMA response time for different values of Q and $b = 1$ (A), and $b = 8$ (B).

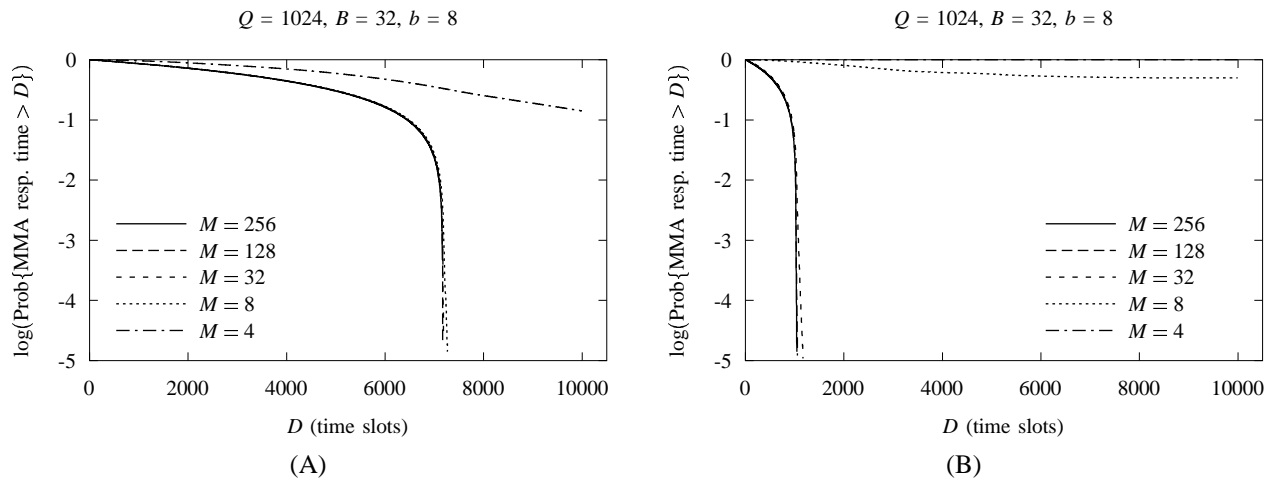


Fig. 6. Complementary distribution function of the MMA response time for different values of M and $b = 8$ (A), and $b = 2$ (B).

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 losses. [3] proposes techniques that exploit row locality whenever is possible, in order to enhance the average-case DRAM

bandwidth. However, this scheme may have significant miss probability for special traffic patterns.

For faster line rates, an hybrid SRAM-DRAM implementation of a VOQ buffer using ECQF for the h-MMA, is discussed in [5]. This is the starting point of our work.

There are many proposals exploiting the bank organization of DRAM memory [8], [10], [4]. 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.

VIII. CONCLUSIONS AND FURTHER WORK

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 ([5] and [2]) can be seen as particular cases of our general scheme.

Based on this general scheme we have proposed a Random BAU 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 [5], without the memory fragmentation problem of the scheme proposed in [2].

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

Further Work: Now we are working on (i) technological aspects of the implementation of the scheme proposed in this paper, (ii) an analytical model for dimensioning our system.

REFERENCES

- [1] J. García, Ll. Cerdà, and J. Corbal. A Conflict-Free Memory Banking Architecture for Fast Packet Buffers. Technical Report UPC-DAC-2002-50, Politechnic University of Catalonia, July 2002.
- [2] J. García, J. Corbal, Ll. Cerdà, and M. Valero. Design and Implementation of High-Performance Memory Systems for Future Packet Buffers. In *Proc. of MICRO'03*, San Diego, CA, December 2003.
- [3] J. Hasan, S. Chandra, and T.N. Vijaykumar. Efficient Use of Memory Bandwidth to Improve Network Processor Throughput. In *30th ISCA*, San Diego, California, USA, June 2003.
- [4] D.T. Harper III and J.R. Jump. Performance Evaluation of Vector Accesses in Parallel Memories using a Skewed Storage Scheme. *13th ISCA*, pages 324–328, 1986.
- [5] S. Iyer, R. Kompella, and N. McKeown. Designing Buffers for Router Line Cards. Technical Report TR02-HPNG-031001, Stanford University, November 2002.
- [6] C. Minkenberg, R.P. Luijten, W. Denzel, and M. Gusat. Current Issues in Packet Switch Design. In *Proc. HotNets-I*, Princeton, NJ, Oct 2002.
- [7] A. Nikolgiannis and M. Katevenis. Efficient Per-Flow Queueing in DRAM at OC-192 Line Rate using Out of Order Execution Techniques. In *IEEE International Conference on Communications*, Helsinki, Finland, June 2001.
- [8] B.R. Rau, M.S. Schlansker, and D.W.L. Yen. The Cydra 5 stride-insensitive Memory System. *International Conference on Parallel Processing*, pages 242–246, 1989.
- [9] Y.J. Tamir and G.L. Frazier. High-Performance Multi-Queue Buffers for VLSI Communication Switches. In *15th ISCA*, pages 343–354, Honolulu, Hawaii, May 1988.
- [10] M. Valero, T. Lang, J.M. LLaberia, M. Peiron, E. Ayguade, and J.J. Navarro. Increasing the Number of Strides for Conflict-Free Vector Access. *19th ISCA*, pages 372–381, May 1992.