On the Design of Hybrid DRAM/SRAM Memory Schemes for Fast Packet Buffers

Jorge García, Maribel March, Llorenç Cerdà, Jesús Corbal, and Mateo Valero
Polytechnic University of Catalonia - Computer Architecture Dept.
{jorge,mmarch,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 general scheme includes some designs previously proposed as particular cases.

Based on this general scheme we propose a new 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 previously proposed schemes without the problem of memory fragmentation.

I. Introduction

THE fastest available high-speed routers today support up to 16 interfaces at OC-192 (10 Gb/s) or OC-768 (40 Gb/s) line rates. It is devised, however, that next generation high-end systems will support a much more larger number of interfaces (e.g. 624 or even more) at OC-192, OC-768 or even OC-3072 (160 Gb/s) line rates [1].

Packet buffers for the 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 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 the hybrid SRAM/DRAM designs. This memory organization for packet buffers was first proposed by the research group of N. McKeown (see [2]). The scheme proposed in [2] ensures zero loss probability for cells coming to a non full buffer. For a large number of interfaces, however, the required SRAM size becomes too large. In [3] we described an scheme which aims at reducing the SRAM size of [2] while supporting a larger number of interfaces. The scheme we proposed in [3] is based on the observation that the effective DRAM access time can be reduced by overlapping multiple accesses to different banks, allowing us to reduce the granularity of the accesses thereby reducing the SRAM size. However, the memory scheme presented in [3] had the drawback of DRAM memory fragmentation, i.e. certain traffic patterns would lead to a situation where only a fraction of the DRAM memory could be used. In [4] we introduced a *renaming of queues* scheme that would reduce the probability of DRAM memory fragmentation. However, since the traffic patters are unpredictable, it is not possible to assess the probability of the DRAM memory fragmentation.

In the proposal presented in this paper we maintain the hybrid SRAM/DRAM design of [2] and [3], however: (i) We redesign the functional blocs that governs SRAM/DRAM memory transfers. Our proposal is a *general* hybrid SRAM/DRAM design such that [2] and [3] schemes are particular cases of our general scheme. (ii) We propose a new algorithm for the general scheme that can reduce the by almost an order of magnitude the SRAM size of the scheme proposed in [2]. Furthermore, this new algorithm would not have the memory fragmentation problem of the scheme proposed in [4].

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 governs 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 in DRAM. The SRAM memory bandwidth must fit the line rate, which means that the SRAM access time must be less than or equal to the transmission time of a cell (we shall refer to this time as *time slot*).

The DRAM memory is organized in M banks. Figure 2 illustrates the concept of a memory bank. A memory bank is

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 TIC98-0511OC02-01 and by the Dep. of Univ. (Generalitat de Catalunya) under grant CIRIT 2001-SGR-00226 and by a grant from IBM.

0-7803-8375-3/04/$20.00 © 2004 IEEE

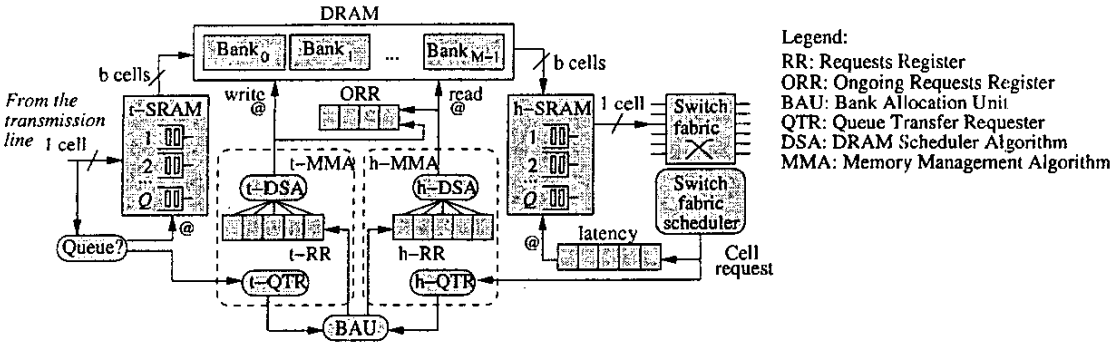

Fig. 1. General Hybrid DRAM/SRAM memory architecture.

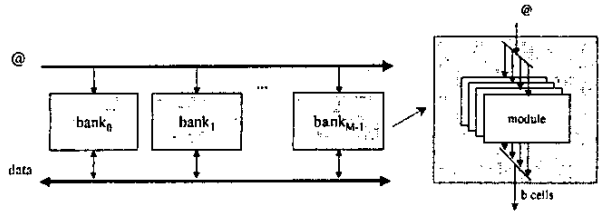

Fig. 2. Organization of a DRAM in banks: configuration of a DRDRAM-style memory and internal structure of a memory bank.

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 measured in cells is the *data granularity*. We shall refer as b to the granularity used in our scheme. 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 could be used if we require a random transfer from 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, we would have: $B \geq 2T/R$ (we use $2T$ since each B time slots we have to do a read and a write transfer to DRAM). Now, 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 an 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 constrain cannot be fulfilled.

In the following we shall first explain how cell transfers between t-SRAM to DRAM are managed, and then we will do the same for cell transfers between h-SRAM and DRAM.

Every b time-slots, the *tail Memory Management Algorithm* (t-MMA) selects a queue and a memory bank from which b cells are to 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 the h-MMA described later in this section.

The functional blocks of t-MMA work as follows: When a cell for queue i arrives from the transmission line, the t-QTR decides whether a transfer from 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 possible, i.e. when b cells of queue i are standing in 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 t-QTR issues a transfer request for queue i if $(C_i^t \mod b) = 0$.

The request issued by t-QTR is processed by the *Bank Allocation Unit* (BAU), which in turn chooses the bank where the cells should be allocated (the algorithm to do so will be discussed in later sections). 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, the *tail DRAM Scheduler Algorithm* t-DSA selects one of the transfer requests pending in t-RR every b slots issuing the transfer from t-SRAM to DRAM. In order to choose one of the pending transfers in 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 t-SRAM and DRAM. In the following we shall focus on the transfers between h-SRAM and DRAM. These transfers are managed by the *head Memory Management Algorithm* (h-MMA). Now 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 as it may not yet have been transferred from the DRAM. We shall refer this condition as a *miss*.

Again, the h-QTR algorithm is simple: Schedule a transfer for queue i whenever the number of request for cells from queue i issued by the scheduler, exceeds the number of cells from this queue present in 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 request a cell from queue i, C_i^h is increased and t-QTR issues a transfer request for queue i if $(C_i^h \mod b) = 1$. We shall refer as the *h-MMA response time* to the delay 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 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 h-MMA work analogously to those of t-MMA. Now, however, we need an additional *latency* register (see Figure 1). This register introduces a delay since the scheduler issues 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 never bank conflicts, and no specific BAU is needed (i.e. consecutive accesses to DRAM can be done to any bank). The DSA can then be seen as a FIFO scheduler, which alternatively chooses the oldest write and the oldest read stored in the RR. This scheme is equivalent to the so called *Early Critical Queue First* (ECQF) proposed in [2]. The required h-SRAM, t-SRAM and latency register sizes are of $Q(B-1)+1$ cells. Shorter sizes would lead to miss probabilities that could be large for some specific traffic patterns.

In [4] 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. Then, store each batch of b cells of the same queue in round-robin fashion among the banks of the group where the queue was assigned. The DSA must choose again the oldest eligible request in the RR. In [4] 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 [2].

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 referred to as *memory fragmentation*. In [4] we alleviated this problem by means of a renaming scheme of the queues. However, even with the renaming scheme, memory fragmentation could still arise for certain traffic patterns.

IV. Random BAU Scheme

In this section we describe a scheme that would exploit DRAM bank organization as in [4] (see section III). Therefore, this scheme allows a data granularity $b < B$, and thus, reducing the SRAM size. However, the scheme described in this section does not have the memory fragmentation problem of [4].

The BAU we propose randomly chooses a DRAM memory bank for every queue 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 the all the banks, provided that requests $r_{n-\frac{B}{b}+1}^i, \ldots, r_{n-1}^i, r_n^i$ are always addressed to different banks (i.e. different banks are chosen for any B/b consecutive requests for the same queue). Since the queues are FIFO, h-QTR consecutive requests for the same queue will also correspond to different bank accesses. Therefore, doing this way we avoid bank conflicts transferring cells from the same queue (remember from section II that 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.

In order to obtain some dimension guidelines for the Random BAU Scheme let assume first that we choose $b = B$. As we explained in section III, in this case there are never bank conflicts and the scheme is equivalent to the ECQF scheme proposed in [2]. We now derive the required SRAM size and the size of the latency register. We shall first focused on the t-SRAM dimensioning. Assume that the t-SRAM is empty and a pattern of cells from each queue in round robin fashion arrive at t-SRAM. The t-QTR would issue the first transfer from t-SRAM to DRAM after $Q(B-1)+1$ time slots. This request would be immediately processed by the t-DSA, starting an upload of B cells from t-SRAM to DRAM (belonging to the first queue having B cells standing in t-SRAM). If we assume that the cells are removed from SRAM when the upload to DRAM is initiated, we would need a t-SRAM size of $Q(B-1)+1$ cells. Consider now the dimensioning of the h-SRAM. Again, assume that the h-SRAM is empty and the scheduler request a pattern of cells from each queue in round robin fashion. The h-QTR would issue a download from DRAM to h-SRAM for every queue during Q consecutive time slots. Therefore, the response time of the h-MMA

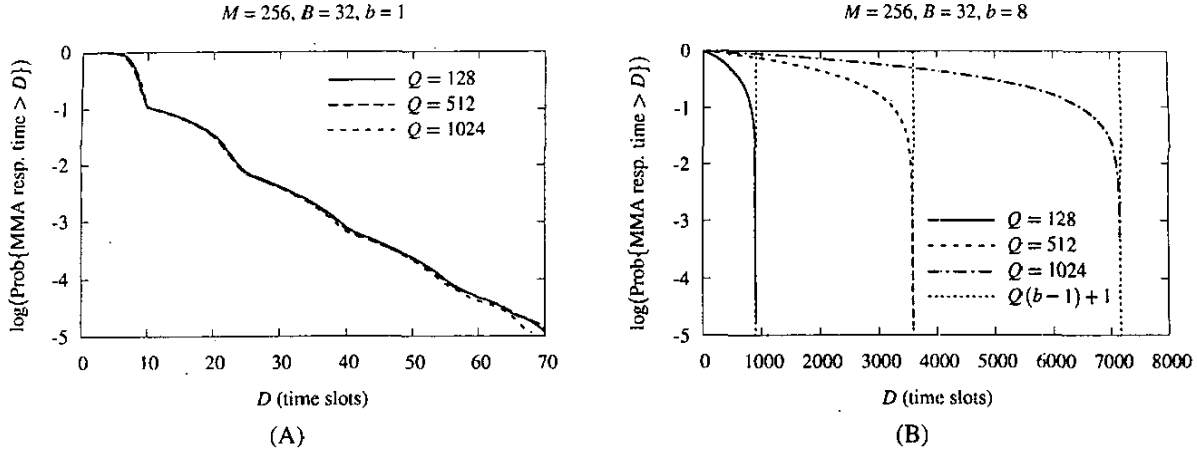

Fig. 3. Complementary distribution function of the MMA response time for different values of Q and $b = 1$ (A), and $b = 8$ (B).

for the n-th download would be of $nB - n + 1, n = 1, \cdots, Q$. Therefore, the maximum h-SRAM response time (and thus, the size of the latency register) would be of $Q(B-1)+1$. Moreover, the size of the h-SRAM should be of $Q(B-1)+1$ cells. For example, if the scheduler stops requesting cells after the first round robin, there would be this amount of cells in the h-SRAM when the Q-th download is completed.

It is interesting to note that the maximum response time of t-MMA is equal to the maximum response time of h-MMA ($Q(B-1)+1$). Analogously to the h-MMA, the maximum t-MMA response time is produced when the t-QTR issues a bursts of transfers requests for the Q queues in Q consecutive time-slots. For instance, if we assume the round robin pattern of arrivals used in the previous discussion, this bursts of of transfers requests would occur at time slots $Q(B-1)+1$, $Q(B-1)+2, \cdots, QB$. In fact, a round robin pattern of arrivals (respectively scheduler requests) would produce the same pattern of transfer of requests will issued by the t-QTR respectively h-QTR. This pattern consist of bursts of Q consecutive transfer requests for all the queues, occurring with a period of QB time slots. The maximum response time would be reached by the last transfer request, and both t-MMA and h-MMA distributions would be the same. Therefore, from now on, we shall not distinguish between the t-MMA and h-MMA response times, and we shall refer to them simply as the *MMA response time*. Furthermore, we shall refer to the round robin pattern of arrivals (respectively scheduler requests) as a *worst case scenario* since they lead to the maximum MMA response time.

Let us now consider a scenario using a granularity $b < B$. Because bank-conflicts may occur, 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 were stored in the same DRAM memory bank. Therefore, 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 ($\frac{1}{M^Q}$) 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 would occur, i.e, assuming an MMA maximum response time of $Q(b-1)+1$.

V. Numerical Results

In this section we analyze the Random BAU Scheme described in section IV. For dimensioning purposes, the key parameter to study is the MMA maximum response time MMA (see section IV).

For the results shown in this section, we use the worst case scenario described in section IV: The t-MMA and h-MMA respectively receive a sequence of cell arrivals and scheduler requests in round robin fashion for queues $1, 2, \ldots, Q$. In response to these patterns, the t-QTR and the h-QTR will generate periodic bursts of transfers requests for queues $1, 2, \ldots, Q$.

Figure 3.(A) shows the Complementary Distribution Function (CDF) of the response time of the MMA under the load conditions previously described when $M = 256, B = 32$, $B = 1$, and $b = 1$ for different values of Q. The three lines are almost coincident, indicating that in this case the delay is independent of the value Q.

When $b > 1$, the response time of the MMA depends on the value of Q. In Figure 3.(B) we show the results for $b = 8$ and different values of Q. Figure 4 shows the results obtained for a fixed value of Q and different values of b. As we

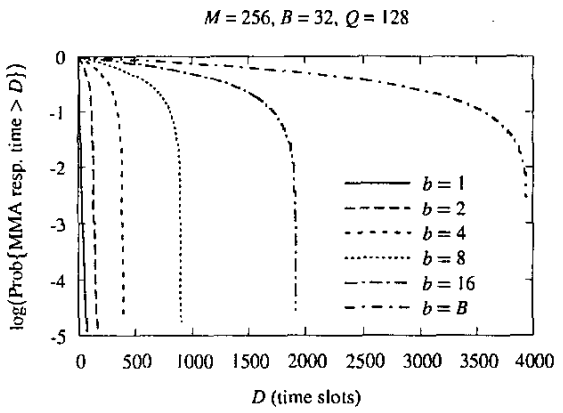

Fig. 4. Complementary distribution function of the MMA response time for $Q = 128$ and different values of b.

see, for values of $b > 2$ the curves have a sharp decrease at point $Q(b-1)+1$. This indicates that it is very unlikely that the response time of the h-MMA is larger than $Q(b-1)+1$.

The previous numerical results show that $Q(b-1)+1$ is plausible dimensioning rule for t-SRAM, h-SRAM and the Latency register of 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 [2]. Furthermore, the Random BAU Scheme does not have the DRAM fragmentation problem of the design we proposed in [4].

VI. Related Work

Virtual Output Queuing was proposed for the first time in [5] (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. [6] proposes 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. [7] proposes techniques that exploit row locality whenever possible in order to enhance 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 [2]. This is the scheme we used as starting point for our work.

There are many proposals exploiting the bank organization of DRAM memory [8], [9], [10]. This is especially true in the vector processor domain. The novelty of our technique resides in the application of this technique to the context of fast packet buffering.

VII. 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 ([2] and [4]) 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 [2], without the memory fragmentation problem of the scheme proposed in [4].

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 implementaion of the scheme proposed in this paper, (ii) an analytical model for system dimensioning.

References

[1] C. Minkenberg, R.P. Luijten, W. Denzel, and M. Gusat, "Current Issues in Packet Switch Design," in *Proc. HotNets-I*, Princeton, NJ, Oct 2002.

[2] S. Iyer, R. Kompella, and N. McKeown, "Designing Buffers for Router Line Cards," Tech. Rep. TR02-HPNG-031001, Stanford University, Nov. 2002.

[3] J. García, Ll. Cerdà, and J. Corbal, "A Conflict-Free Memory Banking Architecture for Fast Packet Buffers," Tech. Rep. UPC-DAC-2002-50, Politechnic University of Catalonia, July 2002.

[4] 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.

[5] Y.J. Tamir and G.L. Frazier, "High-Performance Multi-Queue Buffers for VLSI Communication Switches," in *15th ISCA*, Honolulu, Hawaii, May 1988, pp. 343–354.

[6] A. Nikologiannis 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.

[7] 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.

[8] B.R. Rau, M.S. Schlansker, and D.W.L. Yen, "The Cydra 5 stride-insensitive Memory System," *International Conference on Parallel Processing*, pp. 242–246, 1989.

[9] 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*, pp. 372–381, May 1992.

[10] D.T. Harper III and J.R. Jump, "Performance Evaluation of Vector Accesses in Parallel Memories using a Skewed Storage Scheme," *13th ISCA*, pp. 324–328, 1986.