Memory Systems

Cache, DRAM, Disk

Bruce Jacob
University of Maryland at College Park

Spencer W. Ng
Hitachi Global Storage Technologies

David T. Wang
MetaRAM

With Contributions By

Samuel Rodriguez
Advanced Micro Devices

ELSEVIER

AMSTERDAM • BOSTON • HEIDELBERG LONDON
NEW YORK • OXFORD • PARIS • SAN DIEGO
SAN FRANCISCO • SINGAPORE • SYDNEY • TOKYO
Morgan Kaufmann is an imprint of Elsevier

MORGAN KAUFMANN PUBLISHERS

open-page-optimal address mapping schemes. For example, Figure 13.4 shows that in the single/asymmetric channel mode, the address mapping scheme in the 82955X MCH can be represented as k:l:r:b:n:z, and in the symmetric dual channel mode, the address mapping scheme can be represented as l:r:b:n:k:z. In both cases, the column address fields are mapped to the low address ranges so that spatially adjacent memory address locations can be directed to the same open page. Similarly, in the various address mapping schemes illustrated in Figure 13.4, the 82955X MCH shows that the side effect of granting the end-users the ability to configure the memory system with differently organized memory modules is that rank parallelism to spatially adjacent memory accesses is lost. Although the rank address field is not explicitly illustrated in Figure 13.4, the use of the address boundary registers and per-rank address mapping schemes means that the rank address field is mapped to the high address ranges above the row address field.

Figure 13.4 shows that the 82955X MCH has been cleverly designed so that most of the bit positions are directed to the same address fields regardless of the organization of the memory modules in the memory system. For example, Figure 13.4 shows that physical address bits 16 through 26 are used to denote row addresses 0 through 10 in the single/asymmetric channel mode, regardless of the number and type of memory modules placed in the memory system. In this manner, only a few bit positions will have to be dynamically adjusted depending on the organization of the memory system, and bit positions shown with the grey background in Figure 13.4 are always directed to the same address fields.

Finally, the address mapping scheme in the 82955X MCH means that single threaded streaming applications often cannot take advantage of the parallelism afforded by multiple ranks and the two channels in asymmetric channel mode. Fortunately, multiprocessor and multi-threaded processor systems with concurrently executing contexts can access different regions of memory and may be able to take advantage of the parallelism afforded by the multiple ranks and multiple channels in asymmetric channel mode. However, the amount of achievable parallelism depends on the specific access request sequences and the locations of the data structures accessed by the concurrently executing process contexts.

13.3.6 Bank Address Aliasing (Stride Collision)

One additional issue in the consideration of an address mapping scheme is the problem of bank address aliasing. The problem of bank address aliasing occurs when arrays whose respective sizes are relatively large powers-of-two are accessed concurrently with strided accesses to the same bank. Figure 13.4 shows that in a system that uses 1-GB DDR2 SDRAM memory modules with the 82955X MCH in dual channel mode, the bank address for each access is obtained from physical address bit positions 14 through 16. That is, in this system configuration, all contiguously allocated arrays that are aligned on address boundaries that are integer multiples of 2^{17} bytes from each other would have array elements that map to identical banks for all corresponding array elements.

For example, the task of array summation, where the array elements of arrays A and B are added together and then stored into array C, requires that the corresponding elements of A, B, and C be accessed concurrently. In the case where arrays A, B, and C are contiguously allocated by the system and mapped to integer multiples of 128-kB address boundaries from each other, then array elements A[i], B[i], and C[i], would be mapped to different rows within the same bank for all valid array indices i, resulting in multiple bank conflicts for each step of the array summation process in the system described above.

In general, the bank address aliasing problem can be alleviated by several different methods. One method that can alleviate the bank address aliasing problem is the conscientious application of padding or offsets to large arrays so that bank conflicts are not generated throughout concurrent array accesses to those large arrays.[10] A second method that can alleviate the bank address aliasing problem is the conscientious design

[10] A simple offset insertion increased STREAM Triad bandwidth by 25% in a test system with an Intel i875P system controller.

of a memory management unit that can purposefully allocate large arrays to non-contiguous pages in the physical address space. In this manner, the chance of a bank conflict changes from a guaranteed event that occurs for every single access to the array to a probabilistic event that depends on the number of banks and ranks in the memory system. Finally, improved address mapping schemes have been proposed to alleviate the bank address aliasing problem, and they are described in the following section.

Hardware Solution to the Address Aliasing Problem

The bank address aliasing problem has been investigated by Lin et al. [2001] and Zhang et al. [2000]. The schemes proposed by Lin and Zhang are similar to schemes applied to different memory systems. The basic idea of taking the row address and bitwise XOR'ed with the bank address to generate new bank addresses that are not aligned for concurrent accesses to large arrays is common to both designs. However, the generous rank and bank parallelism in the fully configured Direct RDRAM memory system allowed Lin to create a 1:1 mapping that permutes the available number of banks through the entire address space in the system configuration examined. In contrast, Zhang illustrated a more modest memory system where the page index was larger than the bank index. The mapping scheme described by Zhang is shown in Figure 13.5. Figure 13.5 shows that the problem for the scheme described by Zhang is that there are relatively few banks in contemporary SDRAM and DDRx SDRAM memory systems, and for a DRAM memory system with 2^b banks, there are only 2^b possible permutations in a 1:1 mapping that maps a physical address to the memory address. In the bank address permutation scheme for the conventional SDRAM-type memory system proposed by Zhang, the address aliasing problem is simply shifted to a larger granularity. That is, without the bank permutation scheme illustrated in Figure 13.5, arrays aligned on address boundaries of $2^{(b+p)}$ bytes would suffer a bank conflict on every pair of concurrent array accesses. The implementation of the bank permutation scheme means that arrays aligned on address boundaries of $2^{(b+p)}$ bytes no longer suffer from the same address aliasing problem, but arrays that are aligned on address boundaries of $2^{(b+p+b)}$ bytes continue to suffer a bank conflict on every pair of concurrent array accesses. Essentially, there are not enough banks to rotate through the entire address space in a contemporary memory system to completely avoid the memory address aliasing problem.

13.4 Performance Optimization

The performance characteristic of a modern DRAM memory controller depends on implementation-specific DRAM command and memory transaction ordering policies. A DRAM controller can be designed to minimize complexity without regard to performance or designed to extract the maximum performance from the memory system by implementing aggressive DRAM command and memory transaction ordering policies. DRAM command and transaction

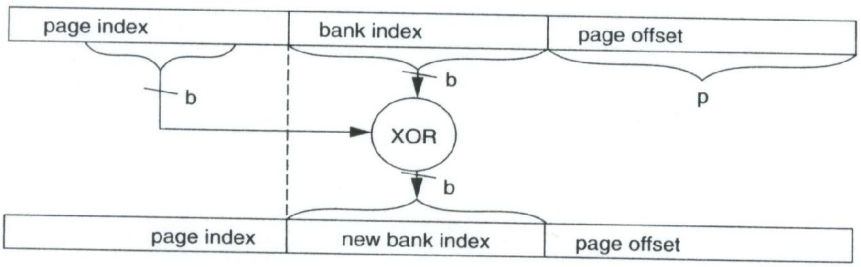

FIGURE 13.5: Address mapping scheme proposed by Zhang et al. [2000].

ordering policies have been studied by Briggs et al. [2002], Cuppu et al. [1999], Hur and Lin [2004], McKee et al. [1996], Lin et al. [2001], and Rixner et al. [2000]. In studies performed by Briggs et al., Cuppu et al., McKee et al., Lin et al., and Rixner et al., various DRAM-centric scheduling schemes are examined. In the study performed by Hur et al., the observation is noted that the ideal DRAM scheduling algorithm depends not only on the optimality of scheduling to the DRAM memory system, but also on the requirement of the application. In particular, the integration of DRAM memory controllers with the processor core onto the same silicon die means that the processor core can interact directly with the memory controller and provide direct feedback to select the optimal DRAM command scheduling algorithm.

The design of a high-performance DRAM memory controller is further complicated by the emergence of modern, high-performance, multi-threaded processors and multi-core processors. While the use of multi-threading has been promoted as a way to hide the effects of memory-access latency in modern computer systems, the net effect of multi-threaded and multi-core processors on a DRAM memory system is that the intermixed memory request stream from the multiple threaded contexts to the DRAM memory system disrupts the row locality of the request pattern and increases bank conflicts [Lin et al. 2001]. As a result, an optimal DRAM controller design not only has to account for the idiosyncrasies of specific DRAM memory systems, application-specific requirements, but also the type and number of processing elements in the system.

The large number of design factors that a design engineer must consider underlines the complexity of a high-performance DRAM memory controller. Fortunately, some basic strategies exist in common for the design of high-performance DRAM memory controllers. Specifically, the strategies of bank-centric organization, write caching, and seniors first are common to many high-performance DRAM controllers, while specific adaptive arbitration algorithms are unique to specific DRAM controllers and systems.

13.4.1 Write Caching

One strategy deployed in many modern DRAM controllers is the strategy of write caching. The basic idea for write caching is that write requests are typically non-critical in terms of latency, but read requests are typically critical in terms of latency. As a result, it is typically desirable to defer write requests and allow read requests to proceed ahead of write requests, as long as the memory ordering model of the system supports this optimization and the functional correctness of programs is not violated. Furthermore, DRAM devices are typically poorly designed to support back-to-back read and write requests. In particular, a column read command that occurs immediately after a column write command typically incurs a large penalty in the data bus turnaround time in conventional DDRx SDRAM devices due to the fact that the column read command must await the availability of the internal datapath of the DRAM device that is shared between read and write commands.

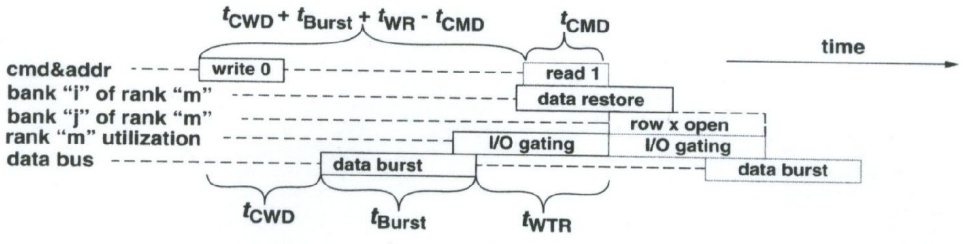

FIGURE 13.6: Write command following read command to open banks.