FACTA UNIVERSITATIS (NIŠ)
SER.: ELEC. ENERG. vol. 17, April 2004, 81-97

Access Latency Reduction in Contemporary DRAM Memories

Vladimir Stanković and Nebojša Milenković

Abstract: Performances of DRAM memories are characterized by memory latency and bandwidth. Contemporary DRAM memories more successfully satisfy demands for higher bandwidth than lower latency. In this paper solutions, which may reduce latency of these memories, are investigated. These solutions are two new controller policies called 'Write-miss Only Close-Page' and 'Write-miss Only Close-Page-Open previous Page' as well as several address remapping schemes. 'Write-miss Only Close-Page' policy is basically a combination of the policies 'Open-Page' and 'Close-Page-Autoprecharge'. For all DRAM reads 'Open-Page' policy is used. Also for all DRAM writes that cause row-buffer hits 'Open-Page' policy is used. For all DRAM writes that cause row-buffer misses 'Close- Page-Autoprecharge' policy is used. 'Write-miss Only Close-Page-Open previous Page' policy is the same as 'Write-miss Only Close-Page', except that after the precharge the previously open row is opened again. Simulations show improvements in using these combined policies.

Permutation-based Page Interleaving scheme is known as an effective address remapping scheme for reducing row-buffer conflicts, which are consequence of conflict cache memory misses. This scheme is based on using xor circuits for changing bank indices of data blocks that fit into the same cache memory line set. We improve this scheme by proposing five similar schemes, with slightly better effectiveness. Three of the proposed schemes have approximately the same performances, but do not use xor circuits at all. Two of the proposed schemes use xor circuits but have slightly better performances.

Keywords: DRAM, memory latency, DRAM controller, controller policy, address remapping, DRAM simulator.

Manuscript received October 22, 2003.

The authors are with the Faculty of Electronic Engineering, University of Niš, Beogradska 14, Serbia and Montenegro (e-mail: vstankovic@elfak.ni.ac.yu).

1 Introduction

Contemporary microprocessors have reached performances that were hard to imagine a few decades ago. Processors with ILP fetch and execute 2-6 instructions in every clock cycle. Clock cycles that are order of magnitude of GHz have been reached. Processors with these posibilities demand memory with proper speed. Contemporary memories are hierarchicaly organized, so their speed mostly is defined by cache memories and main memory. Cache memories are in 2-3 levels, with the highest level that comunicates with the processor, and the lowest level that comunicates with the main memory. Dynamic RAM (DRAM) memories have been a primary choice for implementing main memories for many years.

Main memory performances, when serving a cache miss, are determined by two parameters: memory latency time and memory bandwidth. The memory latency time is mainly determined by the memory core. It decreases in the last years for about 5% per year. The memory bandwidth can be increased by various architectural and implementational solutions of the interface between the memory bus and the DRAM core, in which great progress have been achived in the last years. In this paper we have concentrated on solutions for decreasing memory latency. The reason for that lies in the fact that contemporary DRAM memories more efficiently satisfy the demands for increased bandwidth, than for decreased latency. Contemporary DRAM memories are equiped with some of the following architectural solutions, which enable memory latency decreasing:

- row (page) buffer, associated to the memory cell array, in which the content of the open row is temporaly stored;
- greater number of memory banks, achived by division of the memory cell array into smaller arrays, with separated decoders and row buffers, associated to each bank;
- SRAM memory in a form of cache, associated with the DRAM core, which can be used longer and more efficiently than the row buffer, with the same positive effects.

This paper presents some results that arise from researching the ways of using the specified architectural solutions of DRAM memories, in order to decrease the average memory latency. It is organized as follows. Section 2 contains related works of various authors. Section 3 contains a short description of considered types of DRAM memories. In section 4 experimental environment is defined. Section 5 contains considered DRAM controller polices. Address remapping is considered in section 6. Section 7 contains conclusions.

2 Related Works

Many authors in their papers deal with DRAM memory performance improvement possibilities. Let us review some of those papers.

V. Cuppu, B. Jacob, B. Davis and T. Mudge in [1] study a number of DRAM memory types: FPM DRAM, EDO DRAM, SDRAM, ESDRAM, SLDRAM, CRDRAM and DRDRAM. They were using Simplescalar [2] for obtaining the results. Their main conclusion is that contemporary DRAM memories 'attack' directly the bandwidth problem, but not the latency problem. Some memories, like SDRAM, ESDRAM, SLDRAM and RDRAM, have reduced the stall time due to limited bandwidth by a factor of 3 compared to earlier DRAM architectures, while omitting proper latency improvements. They also conclude that in the future the bus will become a bottleneck, i.e. that by increasing the bus width, the row access time will become significant. V. Cuppu and B. Davis publish the results of further researching in [3]. This paper comprehends studying results for various parameters like bus speed, bus width, number of independent channels, number of banks and burst mode length. The most important conclusions are that the memory organization can influence the total time by a factor of 3, and that the most important parameters are the channel speed, and the data access granularity - each of these two parameters can independently influence the performances by a factor of almost 2.

Z. Zhang, Z. Zhu and X. Zhang in [4] propose an address remapping called 'Permutation- based Page Interleaving Scheme'. This remapping eliminates row-buffer conflicts induced by L2 cache conflict misses. It requires an addition of xor logic circuits to the DRAM controller, whose number is determined by the width of bank field in the DRAM address. They compare the new policy to other three policies: 'Page Interleaving Scheme', 'Cache Line Interleaving Scheme' and 'Swapping Scheme'. The results show improvements over these policies.

In [5] B. Davis, T. Mudge, B. Jacob and V. Cuppu show the significance of address remappings for performances. They conclude that, for most architecture, address remappings can have much higher influence then the DRAM controller policies.

B. T. Davis in its doctoral dissertation [6] gives some results considering various parameter influences, like: processor clock frequency, cache memory interaction, effect of using MSHR-s (Miss information Status Holding Register), DRAM controller polices, address remappings, bus utilization, DRAM access concurrency.

One of the ways for L2 cache misses penalty decreasing is data prefetching. W. Lin, S. Reinhardt and D. Bourger in [7] propose a memory system variant, which uses a hardware controlled L2 cache data prefetching. They use DRDRAM

memory with four channels, and apply the 'Permutation-based Page Interleaving Scheme' from [4] for address remapping and prefetch scheduling, based on information about already open pages.

DRAMs with small amount of cache integrated in the same chip (cached DRAMs) working with ILP processors are investigated in [8]. Z. Zhang, Z. Zhu and X. Zhang find that: cached DRAM consistently improves performance as the ILP degree increases; cached DRAM exploits memory access locality more effectively then other contemporary DRAM types; compared to a highly effective permutation-based DRAM interleaving technique, cached DRAM still substantially improves performance.

3 Types of DRAM Memories

There are many types of DRAM memories: conventional DRAM, FPM DRAM, and EDO DRAM, which are controlled asynchronously by the processor or the memory controller. Latest types of DRAMs are synchronous DRAM (SDRAM), Synchronous Link DRAM (SLDRAM) and Rambus DRAM (RDRAM), as well as cache enhanced SDRAMs as Virtual Channel SDRAM (VC SDRAM) from NEC and Enhanced SDRAM (ESDRAM) from EMS. All these new memories have innovative portion in their interface, or access mechanism, which is synchronous, while the DRAM core remains essentially unchanged. Unfortunately, SLDRAM, whose projected parameters have been very promising, have not yet entered into production. Nevertheless, we have included this memory in our investigation, wishing to compare its performance with other actual DRAM types. Further improvement of SDRAM is Double Data Rate (DDR) mechanism, which doubles the rate of data transfer to/from DRAM. DDR is applied in DDR SDRAM, SLDRAM, and RDRAM. For RDRAM, we have considered two types, Concurrent RDRAM (CRDRAM) which is the second generation, and Direct RDRAM (DRDRAM), which is the third and current generation of this memory.

SDRAM has synchronous interface such that SDRAM latches information to and from the controller based on a clock signal. SDRAM devices typically have a programmable register that holds a bytes-per-request value. SDRAM may therefore return many bytes over several cycles per request. The advantages include elimination of the timing strobes and availability of data from the SDRAM on each clock cycle. The underlying architecture of the SDRAM core is the same as in a conventional DRAM. With improved interface, which enables data transfer in and out of SDRAM on both clock signal edges, Double Data Rate SDRAM (DDR SDRAM) is achieved. The original SDRAM is then named Single Date Rate SDRAM (SDR SDRAM). Examples of these memories are [9] for SDR SDRAM and [10] for DDR

SDRAM.

SLDRAM [11] is an adaptation of Ram-Link standard for DRAM, and is another IEEE standard (P1596.7). The SLDRAM specification is therefore an open standard allowing for use by vendors without licensing fees. SLDRAM uses a packet-based split request/response protocol. Its bus interface is designed to run at clock speeds of 200-600 MHz and has a two-byte-wide datapath. SLDRAM supports multiple concurrent transactions, provided that all transactions reference different internal banks.

CRDRAM [12] uses one-byte-wide multiplexed address/data bus to connect the memory controller to the CRDRAM devices. The bus runs at 300 MHz and transfers data on both clock edges (DDR) to achieve a theoretical peak of 600 Mbytes/s. Transactions occur on the bus using a split request/response protocol. Because the bus is multiplexed between address and data, only one transaction may use the bus during any 4-clock cycle period, referred to as an octcycle. The protocol uses packet transactions; first the address packet is driven, then the data. Different transactions can require different numbers of octcycles, depending on the transaction type, data location within the device, number of devices on the channel, etc.

DRDRAM [13] uses 400 MHz 3-byte-wide channel (2 for data, 1 for addresses/commands). Like previous Rambus parts, Direct Rambus parts transfer data at both clock edges (DDR), implying a maximum bandwidth of 1.6 Gbytes/s. DRDRAMs are divided into 16 banks with 17 half-row buffers. Each half-row buffer is shared between two adjacent banks, which implies that adjacent banks cannot be active simultaneously. A critical difference between CRDRAM and DRDRAM is that DRDRAM partitions the bus into different components, hence three transactions can simultaneously utilize the different portions of the DRDRAM interface.

The Enhanced Synchronous DRAM [14] architecture has a single row cache line for each of the memory banks. If an access is a hit to the cached row, the access time is equivalent to that of accessing a fast SRAM cache. ESDRAM can overlap memory precharging and refreshing operations with cache accessing. The no-write-transfer ability of ESDRAM allows a write to bypass the cache, storing into the DRAM array (via the sense-amps) without affecting the state of the cache. This ability is useful if the controller is able to identify a write-only stream or page. In that case, the pollution of the cache can be avoided. ESDRAM devices have circuitry to guarantee that the cache never contains stale data if this functionality is used. NEC has proposed a cache enhanced SDRAM variant based upon their Virtual Channel (VC) architecture [15]. The intent of the VC architecture is to reduce the average DRAM access time via two mechanisms: (1) to use on-chip SRAM cache to reduce the number of accesses to the DRAM array; and (2) to

organize this SRAM cache in such manner that it allows multiple open channels to the same bank, reducing latency in cases where there are two or more access streams alternating between different rows in the same bank.

4 Experimental Environment

All the illustrated results are attained using the simulator Sim-Outorder from the Simplescalar Tool Set [2]. Sim-Outorder has a very simple DRAM simulator, so it was replaced with DRAM simulators written by the authors. The research presented in this paper has lasted for a longer period of time, during which the situation in the DRAM world has been changing, and so was the choice of implied DRAM memories, too. Thus, some DRAM memories, which have been included in the research at the beginning, have been dropped, and some new ones have been engaged. That is the reason why different DRAM memories appear in different result overviews. Table 1 and Table 2 show the characteristics of all simulated DRAMs. Columns named Row access and Column access of this table, related to VCDRAM, contain no values, because those are not of interest. The only thing that is important considering VCDRAM is the channel read time, which is 20ns. In our simulations we have chosen the VCDRAM version that contains 16 channels.

The rest of the simulated computer configuration is as follows: the processor is a superscalar processor that issues at most 4 instructions on every clock cycle and supports out of order instruction execution. The processor clock frequency is 1 GHz. At the beginning 2 variants of processors were simulated: the first one issues the instructions out of order and the second one does not. Since the results were very similar, we kept the variant that issues instructions out of order. As a branch predictor a two-level branch predictor was used.

There are two levels of cache memories. The first one contains separate instruction cache and data cache. They are both 16KB large; they use direct mapping and have line size of 32B. The second level contains a unified cache, 1MB large, with set-associative mapping - associativity of 4, and line size of 128B. All the cache memories use write-back policy. The second level cache performs writes to main memory through a write buffer. Initially selected values for the write buffer capacity were 4 items and 8 items. Because of result similarity only the value of 4 items was kept. The capacity of one item is equal to the second level cache line (128B).

The simulated memory bus has 128 data lines and operates on 100MHz. Although some memory types (from Table 1 and Table 2) may operate on higher frequencies, that cannot be utilized for the adopted bus frequency of 100 MHz. In order all the simulated memories to be able to use this bus optimally, proper organizations for all of them were simulated. The idea was to be possible to get necessary

number of bits needed for bus transmission (128 bits) with a single read of data that have minimal allowed length. Also it was taken care to obtain memories that are 128MB large in all the cases. So we have simulated these memory organizations:

- 8 SDR SDRAM parallel chips, 16MB each,
- 2 channels of 4 DDR SDRAM parallel chips, 16MB each,
- 2 parallel channels of 8 SLDRAM chips, 8MB each,
- 2 parallel channels of 8 CRDRAM chips, 8MB each,
- 1 channel of 16 DRDRAM chips, 8MB each,
- 2 channels of 8 ESDRAM parallel chips, 8MB each,
- 2 channels of 8 VCDRAM parallel chips, 8MB each.

The term 'parallel' in these memory organizations means that adequate DRAM chips (channels) are simultaneously activated, in order to get 128 bits with a single data access.

Table 1. Simulated DRAM memory characteristics.

DRAM type	Chip capacity	N^o of chips	N^o of banks (per chip)	N^o of rows	Row capacity
SDR SDRAM	16MB	8	4	4096	1KB
DDR SDRAM	16MB	8	4	4096	1KB
SLDRAM	8MB	16	8	1024	1KB
CRDRAM	8MB	16	4	1024	2KB
DRDRAM	8MB	16	16	512	1KB
ESDRAM	8MB	16	4	4096	512B
VCDRAM	8MB	16	2	8192	512B

Table 2. Simulated DRAM memory characteristics.

DRAM type	Data lines	Frequency	DDR	Row access	Column access	Precharge
SDR SDRAM	16	100 MHz	No	20 ns	30 ns	20 ns
DDR SDRAM	16	100 MHz	Yes	20 ns	30 ns	20 ns
SLDRAM	16	200 MHz	Yes	33 ns	37 ns	30 ns
CRDRAM	8	300 MHz	Yes	30 ns	20 ns	27 ns
DRDRAM	16	400 MHz	Yes	27.5 ns	20 ns	25 ns
ESDRAM	16	100 MHz	No	20 ns	10 ns	15 ns
VCDRAM	16	100 MHz	No	-	-	20 ns

For simulation we have used a subset of the benchmark set SPEC95. The benchmarks that were being executed are cc1, compress95, ijpeg, li, m88ksim and perl. The input data were chosen to give the maximum number of executed instructions. Information about these benchmarks is shown in Table 3. The prefixes IL1,

Table 3. Data about used benchmarks.

Benchmark	Cc1	Compress	Ijpeg	Li	88ksim	Perl
Executed ins.	1575 e6	3786973	610.2 e6	1145.6 e6	569.4 e6	2750.8 e6
Committed ins.	1264 e6	3643064	555.4 e6	957.0 e6	490.6 e6	2390.0 e6
IL1 accesses	1770 e6	3855247	632.0 e6	1224.8 e6	613.1 e6	3009.0 e6
IL1 hits	1662 e6	3835043	629.3 e6	1209.3 e6	591.3 e6	2871.1 e6
IL1 misses	108.3 e6	20204	2692807	15523834	21747675	130.0 e6
IL1 replacem.	108.3 e6	19722	2692295	15523322	21747163	138.0 e6
DL1 accesses	550.6 e6	2194268	143.0 e6	471.8 e6	133.6 e6	1067.6 e6
DL1 hits	532.8 e6	1959348	140.0 e6	465.1 e6	131.8 e6	1043.0 e6
DL1 misses	17782454	234920	2995200	6705133	1826805	24569447
DL1 replacem.	17781942	234408	2994688	6704621	1826293	24568935
DL1 write-bck	6197609	230258	1063997	4447416	745627	15649995
UL2 accesses	132.3 e6	485382	6752004	26676383	24320107	178.2 e6
UL2 hits	132.2 e6	481134	6720824	26675125	24314631	177.4 e6
UL2 misses	88773	4248	31180	1258	5476	768972
UL2 replacem.	72393	0	14801	0	0	752588
UL2 write-bck	25944	0	14609	0	0	551470

DL1, and UL2 are related to cache memory and mean Instruction Level 1, Data Level 1, and Unified Level 2, respectively. As can be seen, the benchmarks can be divided into two groups. The first group, which comprises compress95, li, and m88ksim, consists of programs in which, during execution, there are no write-backs from the second level cache into the DRAM, so there are no writes into the DRAM (all DRAM accesses are reads). The second group, on the other hand, comprises cc1, ijpeg, and perl, and has DRAM reads as well as DRAM writes.

5 DRAM Controller Policies

DRAM controller receives data demands from the processor and emits commands to the DRAM core that precharges the proper bank, activates the row (opens the page) and accesses certain columns of the open page. The controller also organizes periodical DRAM refreshment. Two dominant controller policies are Close-Page-Autoprecharge and Open-Page.

With Close-Page-Autoprecharge policy, after every DRAM access, the controller closes the page of the accessed bank issuing a precharge. Every time an access occurs, both row (page) access and column access must be done, as all the pages in the system are closed. That means that for most accesses the latency is the same: tRCD + CL. It can happen the latency to be a little bit longer because of the refreshment effect or because an access occurs to a bank for which autoprecharge has not finished yet. Close-Page-Autoprecharge policy represents a conservative policy, because it is expected that the following access to the same bank will ref-

erence another page. The policy can make good results when successive accesses to same DRAM banks are spread over distinct pages. Close-Page-Autoprecharge policy has deterministic latency and is simple for implementation.

Open-Page policy implies that, after accessing the DRAM memory page, that page remains open, i.e. present in the bank sense amplifiers. The idea is in the assumption that the next access to that bank will be referenced to the same page, thus eliminating the need for a page opening (RAS). For that kind of situation it is said that a row buffer hit or an open page hit occurs. In that case the access would last only as long as the column access is, i.e. the latency would be equal to CL. On the other hand, if the access is referenced to some other page, not the open one, it is said that a row buffer miss or an open page miss occurs. In this case, the latency is equal to tRP+tRCD+CL. This policy is less deterministic (the latency is unsteady), and the performances depend on the number of row buffer hits. When locality of memory references is good (poor), performances are also good (poor).

Open-Page policy demands more complex controller than Close-Page-Autoprecharge policy. For every bank in the system it must be known which page is open, which means that the controller must have a register for each bank. On every access to the bank it must be checked, using this register, whether the access belongs to the open page or not, and a proper command should be sent.

5.1 Write-miss only close-page policy and write-miss only close-page-open previous page policy

Table 4 shows row buffer hit probabilities for the 6 benchmark programs, when applying Open-Page policy. It can be seen that programs with no DRAM writes (compress95, li, m88ksim) have rather good row buffer hit probabilities, better than programs with DRAM writes (cc1, ijpeg, perl). In Table 5, hit probabilities for programs with DRAM writes are given separately for reads and writes. It can be seen that in all the cases row buffer read hit probabilities are much greater than row buffer write hit probabilities. All these results are obviously consequence of two facts: 1. Cache memory uses write-back policy, and 2. DRAM writes are done through a write buffer. That way both read and write hit probabilities decrease. The moment a DRAM write occurs there is a high probability that the row that is open is the wrong one, and thus the low write hit probability. After the write is performed, there is a certain probability that leaving that row open will cause further row buffer misses. This gives an idea of trying a combined policy of opening and closing the row, which we have called the *Write-miss Only Close-Page policy*. Instead of using Open-Page policy all the time, Close-Page- Autoprecharge policy should be used with every write that causes a row buffer miss. This means that, if a row buffer

write miss occurs, then that row is closed after the write is performed, and if a row buffer write hit occurs, then we leave that row open. The write miss means that the row into which the write is to be performed, probably is not the one into which the next access (into that bank) will occur, so it makes sense to close that row. The write hit, on the other hand, means that probably the next access (into that bank) will occur into that row, so we leave that row open. Since reads incur no problems, we use Open-Page policy for reads all the time.

Table 4. Row buffer hit probabilities.

	Cc1	Compress95	Ijpeg	Li	M88ksim	Perl
SDR SDRAM	0.35	0.80	0.35	0.66	0.79	0.12
SLDRAM	0.40	0.85	0.39	0.76	0.84	0.14
CRDRAM	0.44	0.91	0.43	0.84	0.90	0.13
DRDRAM	0.37	0.84	0.33	0.74	0.82	0.12

Table 5. Row buffer read hit and write hit probabilities.

	Cc1 read	Cc1 write	Ijpeg read	Ijpeg write	Perl read	Perl write
SDR SDRAM	0.43	0.09	0.47	0.09	0.15	0.07
SLDRAM	0.49	0.10	0.51	0.11	0.18	0.07
CRDRAM	0.54	0.10	0.56	0.13	0.18	0.06
DRDRAM	0.46	0.07	0.45	0.08	0.17	0.05

Obtained results show improvements in all the cases, which gives encouragement for trying to further improve the Write-miss Only Close-Page policy. After closing the page (on a write miss), an attempt can be made by opening again the previously open page. Namely, since DRAM writes damage the 'natural' data access stream (by both cache write back policy and write buffers) there is a certain probability that, in case of a write miss, the next access will be referenced to the same page that was open the moment the write miss occurred. If this probability is high enough, opening the same page again after performing the write should introduce performance gain. We have named this policy *Write-miss Only Close-Page-Open previous Page*.

The results are shown in Table 6. They show alongside average latency times for Open- Page (OP), Write-miss Only Close-Page (Wm1) and Write-miss Only Close-Page-Open previous Page (Wm2). In our simulations the Open-Page policy approved as better then the Close-Page Autoprecharge policy for all the cases. That is the reason why we have compared Write-miss Only Close-Page and Write-miss Only Close-Page-Open previous Page with only Open-Page. As can be seen, combined policies are better than Open-Page in all the cases. Latency reductions with Write-miss Only Close-Page are in the range from 5.9% (cc1 with SDRAM) to 15.1% (perl with CRDRAM). Latency reductions with Write-miss Only Close-

Page-Open previous Page are in the range from 3.9% (cc1 with SDRAM) to 21.4% (ijpeg with CRDRAM). Write-miss Only Close-Page-Open previous Page, compared to Write-miss Only Close-Page in some cases (8 of 12 cases) gives improvement, and in some cases (4 of 12 cases) gives exacerbation. The exacerbations are in the range from -3.1% (perl with SDRAM) to -0.5% (cc1 with DRDRAM). The improvements are in the range from 1.7% (cc1 with CRDRAM) to 11.5% (ijpeg with CRDRAM).

Table 6. Average latency (in nsec) when using Open-Page (OP), Write-miss Only Close-Page (Wm1), and Write-miss Only Close-Page-Open previous Page (Wm2).

Benchmark	Cc1			Ijpeg			Perl		
Policy	OP	Wm1	Wm2	OP	Wm1	Wm2	P	Wm1	Wm2
SDR SDRAM	56.14	52.80	53.95	56.55	51.93	50.21	65.88	59.32	61.16
SLDRAM	69.07	63.77	64.14	67.80	60.85	56.54	80.24	69.73	68.03
CRDRAM	47.43	43.01	42.27	46.41	41.24	36.48	60.75	51.59	48.79
DRDRAM	51.34	47.17	47.39	53.55	47.72	44.78	64.22	55.58	53.53

Implementation of the suggested policies demands proper changes in the DRAM controller. Let us compare these changes to a controller that uses Open-Page policy. The combined policies would not make the controller much more complex. The controller already remembers which rows are open into which bank, which can be directly used for implementing Write-miss Only Close-Page-Open previous Page. What are the needed changes? For both proposed policies, logic should be added that, in case of a write miss, performs precharge after doing the write. In case of a Write-miss Only Close-Page-Open previous Page, this logic should also instruct an open command for the previously open row (whose index is already stored in the proper register), after the precharge. It is obvious that these changes would increase slightly the controller price, while the latency would remain the same - the added logic will operate in parallel with the write being performed, and the write will obviously last much longer.

6 Address Remappings

During DRAM data placement a decision how the data will be placed, i.e. how the address will be interpreted, must be made. One of the ways could be the following. The first part of the data address, consisting of the most significant bits, determines the group of chips, the following part determines the bank, the next part determines the row, and the last part, consisting of the least significant bits, determines the column. The term ”group” relates to chips (several chips or just a single one) that are always activated simultaneously during DRAM references. Let us name this way ’gbrc’ (Group-Bank-Row-Column). Using this policy means that, as the

address is growing, the first thing that changes is the column, then the row, then the bank, and, at the end the group. This way of data placement is not good, because the data will occupy very little number of banks - it could happen all the data to fit into only one bank. In that case it is logical to assume that the row buffer hit probability will not be high. A little bit better variant would be 'grbc' (Group-Row-Bank-Column). The best one should be 'rgbc' (Row-Group-Bank-Column). In that case all the available banks will be used (if there are that much data, of course). Table 7 shows results for three of the benchmarks, which prove the last statement. In most of the cases, 'rgbc' is the best, 'grbc' is on the second place, and 'gbrc' is the worst.

The 'rgbc' scheme, in which the most significant bits determine the row, the least significant bits determine the column, and the medium ones determine the bank, is also known as a Page Interleaving scheme. Although good, this scheme has a big disadvantage, which is shown in [4]. Any L2 conflicting addresses (having same L2 set index but different L2 tags) are row- buffer conflicting (having same bank index but different row indices). In the last statement L2 is the label for the second cache level, and it is assumed that there are two cache levels in the system. This can be seen from Fig. 1. This figure shows 'rgbc' policy for a DRAM memory, and set-associative mapping for L2 cache memory. The usual lengths of the fields are like in the figure, which means that $r \leq t$, and $t+i \leq r+g+b$. The consequence is that the field 'Index' contains the fields 'Group' and 'Bank'. Hence, addresses with same L2 set index and different L2 tags are row-buffer conflicting, i.e. occupy the same bank but are placed in different rows. In order to resolve this problem, Zhang et al. [4] propose a scheme named Permutation-based Page Interleaving scheme. It consists of the Page Interleaving scheme with changed bank index. The new value of the bank index is a result of a xor (exclusive or) operation over the old bank index and the low order bits of tag. They compare the new scheme with Page Interleaving scheme, Cache Line Interleaving scheme, and Swapping scheme, and show excellence of the proposed Permutation-based Page Interleaving scheme.

Table 7. Row buffer hit probability for 'gbrc', 'grbc' and 'rgbc'.

Benchmark	Compress95			Li			M88ksim		
Policy	gbrc	grbc	rgbc	gbrc	grbc	rgbc	gbrc	grbc	rgbc
SDR SDRAM	0.82	0.80	0.80	0.63	0.65	0.66	0.78	0.79	0.79
SLDRAM	0.82	0.84	0.85	0.63	0.69	0.76	0.78	0.81	0.84
CRDRAM	0.89	0.89	0.91	0.69	0.74	0.84	0.84	0.86	0.90

After modifying the bank index, the L2 conflicting addresses are no longer directed to the same bank, assuming that their tags differ in the lower parts. If the tags do not differ in the lower but in the higher parts, then the row buffer conflict remains. Since it is more likely that tags will differ in the lower, and not the higher

parts, because of data locality, this seems like a good solution. A question arises whether it is possible to obtain a scheme similar to Permutation-based Page Interleaving scheme, but without applying the xor operation. Such scheme would be faster, easier and cheaper to implement. In order to manage such scheme, modifications should be done, such that some of the fields 'Group' or 'Bank' or both of them become included by the field 'Tag', instead of the field 'Index'. Some of the possible solutions are presented in Fig. 2. These solutions are based on simple swapping with shifting of the row bits and group bits ('rgrbc'), the row bits and bank bits ('rbrgc'), or the row bits and both group and bank bits ('rgbrc'). Now different tags, with a proper probability, will mean different banks, i.e. different groups so the influence of conflicting L2 addresses on creating row buffer misses will be much lower. In all the solutions the field 'Group' i.e. 'Bank' contains the lower, and not the higher part of the 'Tag', since higher probability that conflicting addresses will differ in the lower, and not the higher part of the 'Tag'. Let us compare the three solutions among themselves. The first two solutions ('rgrbc' and 'rbrgc') are very similar, and should have similar performances in cases where field lengths of 'Group' and 'Bank' are near equal. The third solution ('rgbrc') fills the part of the tag with both 'Group' and 'Bank' bits, so it is the best in eliminating the row buffer misses that are consequence of L2 conflict cache misses. However, its disadvantage is that it decreases the number of used banks, since it converges to 'gbrc'. Whether this solution will be better or worse than the first two depends on two basic factors, which are also mutually dependent. The first factor is the size of the part of row buffer misses caused by conflict cache misses. If the number of row buffer misses caused by conflict cache misses is much greater than the remainder (number of row buffer misses not caused by conflict cache misses), then this solution should give better performances. If not, the first two solutions should be better. The second factor is the size of the address space accessed by the program. If the address space is large enough, the accessed data will spread over all the available banks in the system. In that case this solution will be better than the first two solutions. On the other hand, if all the accessed data fit into only a single bank or two, that will increase the number of row buffer misses, so the first two solutions will be better. Considering the number of banks in contemporary DRAMs, it is more likely that the first two solutions will be better than the third, but not necessarily. The first two solutions should have similar performances. Larger differences are possible in cases where there are larger differences in the lengths of fields 'Group' and 'Bank'. Besides the three considered solutions, where complete fields 'Group' i.e. 'Bank' are moved into the field 'Tag', one could experiment with variants that only parts of those fields (one bit or two etc.) are moved into the 'Tag' field.

Variants into which not both fields 'Group' and 'Bank' are moved into the field

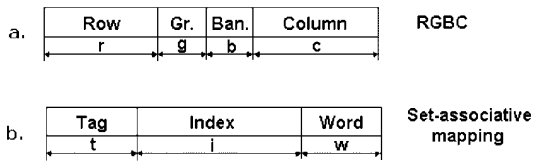

Fig. 1. Address fields for DRAM using 'rgbc' (a.) and address fields for cache memory with set- associative mapping (b.).

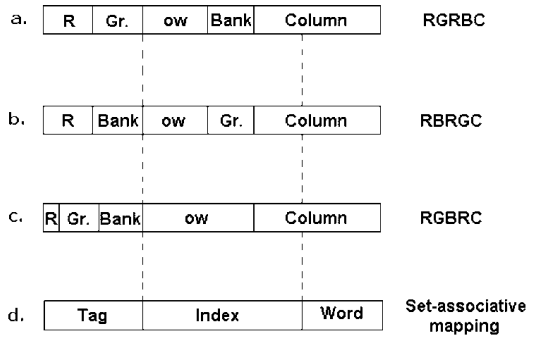

Fig. 2. Comparison of address fields for 'rgrbc' (a.), 'rbrgc' (b.) and 'rgbrc' (c.) with address fields for cache memory with set-associative mapping (d.).

'Tag', but only one of them ('rgrbc' and 'rbrgc') can be additionally improved, but at the price of using xor operation. We will describe this improvement for 'rgrbc' policy. Namely, it could happen that the tags of the L2 conflicting addresses differ in the higher parts, and not the lower ones. In that case 'rgrbc' policy will not give improvement, the row buffer miss will not be resolved. However, we could now apply the xor operation for changing the 'Bank' field. This process is shown in Fig. 3. Now if tags of the missed data block and the data block concurrent differ in their lower bits, then concurrent addresses belong to different groups. If the tags differ in their higher bits, after the xor operation they belong to different banks. We have named this policy 'rgrbcx', since it is derived from 'rgrbc', and 'x' stands for xor. Following the same principle, policy 'rbrgc' gives the policy 'rbrgcx' (this time the field 'Group' is changed not the field 'Bank').

The results are shown in Fig. 4. It presents a comparison of Permutation based Page Interleaving scheme ('pbpi') with the proposed address remappings. The two schemes that use xor operations ('rgrbcx' and 'rbrgcx') are better than or equal to 'pbpi' (except in only one case li with SDRAM). Implementation of these two schemes does not demand any changes in the DRAM controller, when compared

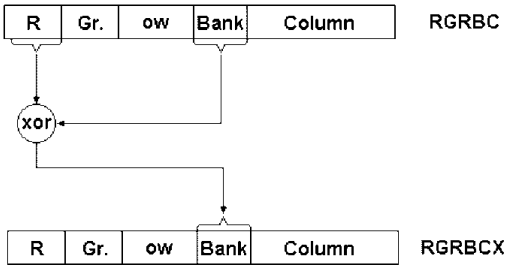

Fig. 3. Derivation of 'rgrbcx' from 'rgrbc'.

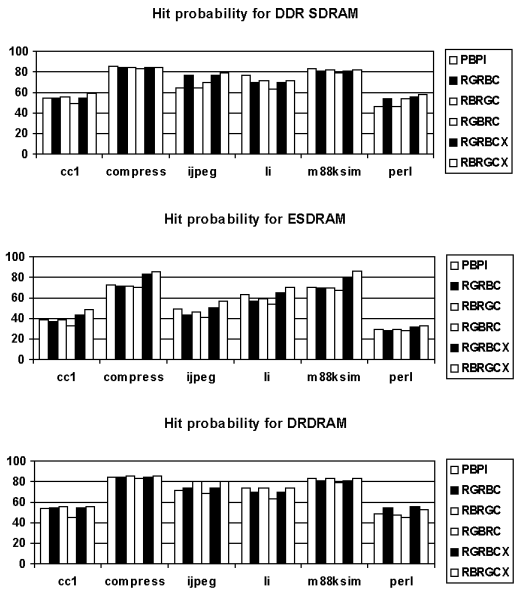

Fig. 4. Comparison of proposed address remappings with Permutation based Page Interleaving scheme.

to 'pbpi' scheme. The price and latency of the xor circuits are the same; hence there is a performance improvement here with same costs. The three schemes that do not use xor operations ('rgrbc', 'rbrgc', 'rgbrc') show approximately same performances as 'pbpi'. In some cases there are improvements, and in some cases there are exacerbations. If we compare those three among themselves, 'rgbrc' is the worse in most the cases, as expected. In case of these three schemes (with no xor operations), there are changes to be done in the controller. The xor circuits should be removed, which will make the controller faster and cheaper. So, there are no improvements here in performances, but there are in price and speed. Es-

sentially, the controller implementation in this case is as complex as in the case of ordinary Page Interleaving scheme.

As one of quality criteria, a preservation of spatial locality of memory references is used [4], which is satisfied for 'pbpi'. In all of our solutions this criterion is also satisfied, namely all addresses in the same page are also in the same page after the address remapping.

7 Conclusion

Performances of DRAM memories are characterized by memory latency and bandwidth. Contemporary DRAM memories more successfully satisfy demands for higher bandwidth than lower latency. In this paper an emphasis on decreasing the memory latency was put. It is achieved by applying techniques on the DRAM controller. First an observation is made considering the difference in row buffer read and write hit probabilities, as a consequence of two factors: cache memory write back policy and using write buffers for DRAM writes. This indicates that write misses damage the data access locality. Two new controller policies, which resolve this drawback, are proposed: Write-miss Only Close-Page and Write-miss Only Close- Page-Open previous Page. Both of them outperform Open-Page and Close-Page-Autoprecharge. Demanded hardware changes in order to implement the proposed policies, compared to Open- Page, are insignificant, and do not decrease the controller speed.

Next, we briefly compare three variants of data placement: 'gbrc', 'grbc' and 'rgbc'. The last one is also known as the Page Interleaving scheme. The results are as expected - 'rgbc' best and 'gbrc' worst.

Finally, we improve the Permutation Based Page Interleaving scheme. This scheme is a form of address remapping, which removes row buffer misses resulted from conflict cache misses and sacrificed data block writebacks. It is derived from Page Interleaving scheme, by changing the bank index using xor operation. The using of xor operation increases the controller latency and price. We propose three new schemes, with similar performances as the Permutation Based Page Interleaving scheme, but with no xor operation. Better insight of the nature of L2 cache misses might help in selecting the optimal among these three schemes. This will be the subject of our future research. Also, we propose two new schemes that use xor operations, and have greater or equal performances as Permutation Based Page Interleaving scheme.

References

[1] V. Cuppu, B. Jacob, B. Davis, and T.Mudge, "A performance comparison of contemporary dram architectures," in *Proc. of the 26th ISCA*, Athlanta GA, May2–4, 1999.

[2] "The simplescalar tool set, version 2.0," University of Wisconsin-Madison, Computer Sciences Department, Tech. Rep. 1342, June 1997.

[3] V. Cuppu and B. Jacob, "Concurrency, latency, or system overhead: Which has the largest impact on uniprocessor dram-system performance?" in *Proceedings of the 28th International Symposium on Computer Architecture*, Göteborg, Sweden, June 30/July 4, 2001.

[4] Z. Zhang, Z. Zhu, and Z. Zhang, "Permutation-based page interleaving scheme to reduce row-buffer conflicts and exploit data locality," in *Proceedings of the 33rd Annual International Symposium on Microarchitecture*, Monterey, California, Dec.10–13, 2000.

[5] B. Davis, T. Mudge, B. Jacob, and V. Cuppu, "Ddr2 and low latency variants," in *In Proc. Memory Wall Workshop, 27th ISCA*, Vancouver BC, Canada, June 2000.

[6] B. T. Davis, "Modern dram architectures," Ph.D. dissertation, University of Michigan, Michigan, 2001.

[7] W. Lin, S. K. Reinhardt, and D. Bourger, "Designing a modern memory hierarchy with hardware prefetching," *IEEE Transactions on Computers*, vol. 50, Nov. 2001.

[8] Z. Zhang, Z. Zhu, and X. Zhang, "Cached dram for ILP processor memory access latency reduction," *IEEE Micro*, pp. 22–32, July/Aug. 2001.

[9] Synchronous dynamic RAM MB81F121642-75/-102/-102L-81F121642.pdf. Fujitsu Limited. [Online]. Available: www.fujitsumicro.com

[10] Double data rate SDRAM MB81PL121647-75/-80/-10 -MB81PL121647.pdf. Fujitsu Limited. [Online]. Available: www.fujitsumicro.com

[11] MT49V8M18C - 4m× 8×2 banks SLDRAM. Micron Technology Inc. [Online]. Available: www.sldram.com

[12] (1998, Apr.) Concurrent RDRAM 16/18Mbit (2M×8/9) & 64/72Mbit (8M×8/9). Rambus Inc. [Online]. Available: www.rambus.com

[13] (1998, Nov.) Direct RDRAM 64/72Mbit (256k×16/18×16d). Rambus Inc. [Online]. Available: www.rambus.com

[14] (2000) 64Mbit - enhanced SDRAM 8M×8, 4M×16 ESDRAM, product brief. Enhanced Memory systems Inc. [Online]. Available: www.edram.com

[15] (1999) MOS integrated circuit PD45125421, 45125821, 45125161 128Mbit virtual channel SDRAM, preliminary data sheet. Nec Corporation.