Using Row-major Mapping with Look-ahead Arbiter to Reduce DRAM Energy and Bandwidth Usage in MPEG-4 Decoding

Wei-Cheng Lin and Chung-Ho Chen
Department of Electrical Engineering, National Cheng-Kung University
No.1 University Road, Tainan, Taiwan 70101, R.O.C
Tel: 886-6-2757575-62400-722 Tel: 886-6-2757575-62394
Kevin@casmail.ee.ncku.edu.tw chchen@mail.ncku.edu.tw

Abstract
In this paper, we introduce a novel bus arbiter architecture, called look-ahead arbiter (LAA), to improve bus bandwidth and energy consumption of using DRAM memory in MPEG-4 decoding. A key innovation of the LAA is that it decides which master gets to use the bus next according to the request status and drives out the next request address to the memory controller in advance. Consequently, the memory controller can decide whether to precharge the row currently accessed to avoid unnecessary row-activation and precharge operation. A second key contribution of this work is that we use row-major mapping for video data instead of block-based mapping to save energy and bandwidth through the LAA. Experiment with the MPEG-4 SP@L3 decoder shows that combining row-major mapping along with LAA, the memory controller provides the best performance both in terms of energy reduction and memory bandwidth improvement, up to 32% and 33% respectively.

Keywords:
Arbiter, bandwidth, DRAM, energy consumption, memory controller, MPEG-4 decoding

1. Introduction
Energy consumption and memory bandwidth demand have become the major concern in the design of embedded video processing system. Fig.1 shows a typical block diagram of a present video processor based on SoC platform. The major building blocks include the general purpose CPU, DSP, application–specific coprocessors, video I/O, memory controller and bus arbiter dedicated to providing the coordination for data transferring between on-chip devices and external memory.

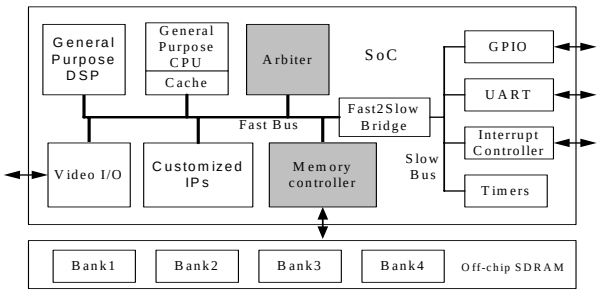


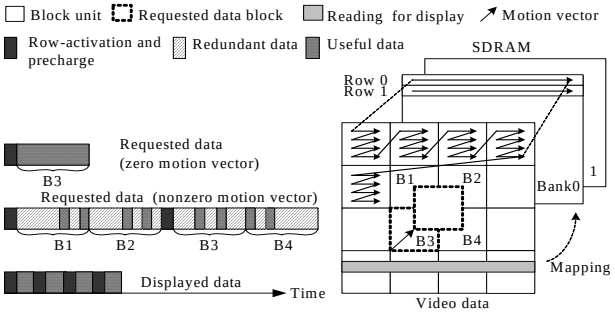
Fig. 1. Video processor based on SoC platform

In this study, we focus on the application of MPEG-4 decoding. Typically, video data that are too large to fit into the on-chip buffers are stored to the off-chip memory, for instance, SDRAM memory. In such a system, high memory bandwidth is required due to the huge amount of video data transferring between video processor and the external SDRAM. However, the SDRAM power dissipation occupies a significant fraction of the overall system power consumption. Thus, it is a good candidate for power and bandwidth minimization.

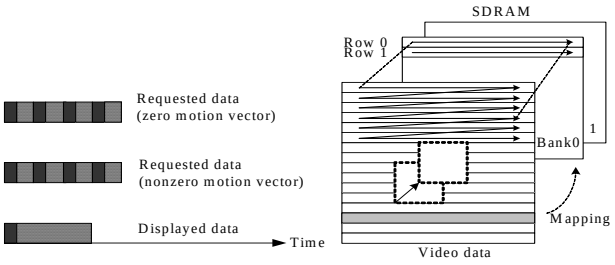
Memory mapping of video data is an important problem in video processing because it directly affects the amount of energy and bandwidth usage of SDRAM. Mapping policies using block-based layout for the video data have been proposed in [4, 5]. Fig. 2 (a) illustrates the block-based mapping policy that can reduce the number of row-activation and precharge operation. In this mapping, the requested data block overlapped with block unit B3 can be accessed in a longer burst mode when the motion vector is zero. However, if the motion vector is not zero, four block units B1, B2, B3, and B4 are transferred because the requested data block overlaps with these four block units. In this example, three-fourths of pixels are redundant in this transferring. Overheads associated with these redundant pixels offset the benefits due to less row-activation and precharge operation in block-based mapping policies. Furthermore, when the video data are read line-wise for display, using block-based mapping is inefficient since each fragment that composes a video line needs a row-activation and a precharge command to specify the address of the fragment.

On the other hand, row-major mapping is inherently suitable for reading video data to display, as shown in Fig. 2 (b). It is observed that the pixel transfer time in row-major mapping has negligible impact from the motion vector value. This is because the address generator employed by a motion compensation module can adjust the starting address of each row in the requested data block according to the motion vector value when the motion vector is nonzero. However, with row-major mapping, it has a negative effect, that is, a row change for the requested data needs a row-activation and precharge operation in order to access the next row in the requested data block.

If the percentage of nonzero motion vector that occurs in a P-VOP is very high, a lot amount of energy and bandwidth are wasted in the transferring of the redundant pixels when a block-based mapping policy is used. In our experiment, the nonzero motion vectors are up to 70% in some P-VOP. Thus, there is an advantage to store the video data in the row-major order in memory.



(a) Block-based mapping



(b) Row-major mapping

Fig. 2. Video data mapping policies and schedule of data transferring

In this paper, we introduce an integrated design of a look-ahead arbiter (LAA) and memory controller to alleviate the negative effect caused by row-major mapping. In this design, the LAA drives out the request address and the next request address to the memory controller according to the request status. Thus, the memory controller can make appropriate decision about whether to precharge an active row based on the access dependence. Consequently, it can avoid unnecessary row-activation and precharge operation to improve bus bandwidth and power consumption when using SDRAM.

The rest of this paper is organized as follows. Section 2 presents the backgrounds and related work. The video data arrangement strategy and proposed architecture are discussed in Section 3 and Section 4, respectively. Section 5 describes the simulation system. Section 6 shows the results of simulations. Finally, Section 7 summarizes the conclusion of this paper.

2. Backgrounds and Related Work

In SDRAM [3], the address that accesses the memory is partitioned into bank, row, and column. Each bank contains an array of memory cells that can be accessed for an entire row at a time. After a row in a bank is activated (i.e. open), which is done through the ACTIVE command, the accessed row is transferred to the bank's row buffer or sense amplifier where READ and WRITE command can take place by supplying the column addresses. Traditionally, the SDRAM either operates in the open page mode or the close page mode. For the open page policy, the selected memory row is stored in the row buffer, which can serve any number of reads or writes (column accesses) until a forced PRECHARGE command is given, or a row miss occurs, or refresh occurs. This precharge operation writes the data in the row buffer back to the memory array (i.e. close the bank), which prepares the bank for subsequent row activation. However, the energy consumed by row-activation and precharge operations occupies a significant fraction of the whole SDRAM energy consumption [8]. Thus, reducing these operations can save significant energy.

Most previous work on DRAM energy and bandwidth issue comes from single processor environments. Such a system does not present the same complexity as today's SoC environment including multi-master, as shown in Fig. 1.

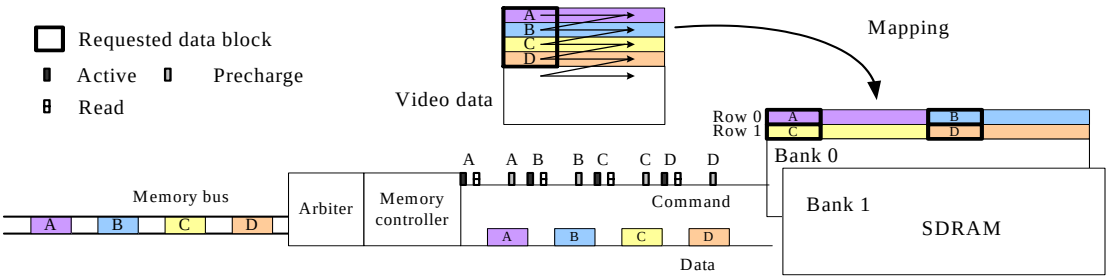
Works in DRAM power management have been previously proposed in [1, 2, 6]. Miura and et al. [1] use open-page policy or close-page policy alternately to reduce the energy consumption and latency when using SDRAM. This is accomplished by using a predictor to predict the access dependence (hit or miss). Approach to reduce DRAM energy with cache-based memory architecture is presented by Fan and et al. [2]. They exploit the fact that the DRAM should directly put into a low-energy mode while there are no access requests. In [6], it is proposed to control power modes of RDRAM according to idle periods predicted by three hardware mechanisms (ATP, CTP and HBP). The idle periods can be explored and integrated by compiler-assisted approach in advance.

In [7], Rixner and et al. propose a mechanism to reduce memory bandwidth, by reordering DRAM operations according to pending memory references. This work may maximize the available memory bandwidth, but it perhaps can not meet the timing requirement specified by a request without considering the priority for each bus master.

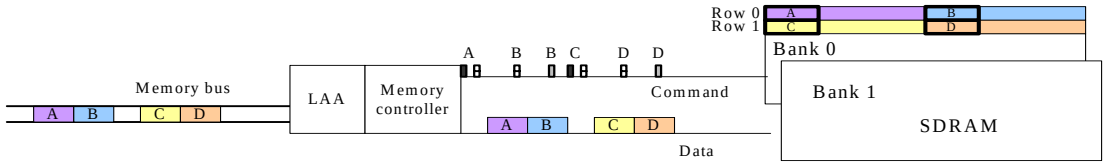
To improve the efficiency of memory access for video processing, Kim and Park [4] proposed an address translation technique that can effectively reduce the number of row-activation and precharge operation. However, this work only examines the MPEG-2 video decoding algorithm without considering the overhead caused by reading video data for display. Approaches to find out a suitable block unit to mapping into SDRAM with minimal memory bandwidth are presented by Jaspers and de With [5].

3. Video Data Arrangement Strategy

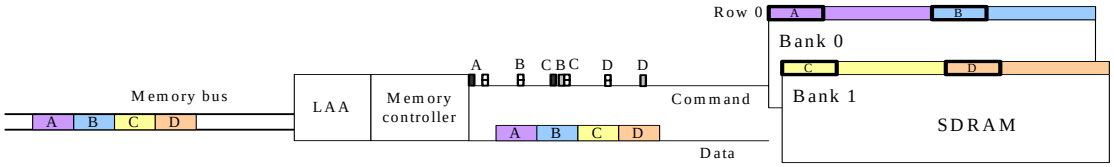
In this section, we explore various data arrangement strategies and implementation method based on row-major mapping of video data. As shown in Fig. 3(a), the video data are stored in row-major order in the external SDRAM. Without the loss of generality, we assume that the row width of the SDRAM is two times of the video width. When 16 pixels in a 4 × 4 block are read from the SDRAM, the data transfer scheme is inefficient due to the fact that the memory controller cannot recognize two consecutive accesses located in the same row of the same bank. Hence, the unnecessary ACTIVE and PRECHARGE command between data A and data B is generated. To overcome this problem, we employ the look-ahead arbiter instead of a conventional arbiter. Fig. 3(b) depicts a reading of requested data block with the LAA that drives out the request address of data A and the next request address of data B to the memory controller simultaneously. Since the request address and the next request address have the same bank field and row field, the seamless data stream of data A and B can be transferred by careful timing of the READ commands (including bank address and column address) for data B.



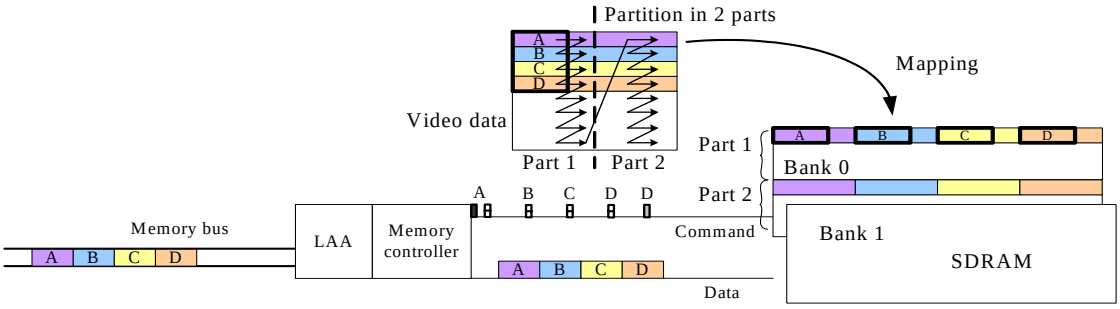
(a) Reading a requested data block with a conventional arbiter



(b) Reading a requested data block with the LAA



(c) Reading a requested data block with the LAA and row interleaving



(d) Reading a requested data block with the LAA and video partition

Fig. 3. Various data arrangement strategies based on row-major mapping and schedule of access

In order to overlap the SDRAM command overhead, the video data can be mapped into two banks of the SDRAM by row interleaving. Fig. 3(c) shows the requested data located in the different banks. If the video data are accessed according to the sequence of A, B, C then D in such interleaved mapping, the ACTIVE and READ commands for data C can be issued in advance to reduce latency of memory access when data B are transferred. The row interleaving reduces the memory-bus bandwidth by taking advantage of two facts. First, in a multiple-bank SDRAM, when one bank performs the ACTIVE or PRECHARGE command, the other bank can be accessed. Second, two successive accesses located in different bank can be made known to the memory controller with the LAA mechanism. Consequently, the command time can be overlapped if each bank is accessed alternately.

To further eliminate row-activation and precharge operations, we place the requested data block into the same row by partitioning the video frame into 2 parts and storing them in row-major order respectively, as shown in Fig. 3(d). Thus, a data block request just requires an ACTIVE and a PRECHARGE command.

The idea behind video partition is to put together the requested data block that has originally been placed into different rows but accessed consecutively. However, applying partition has two drawbacks: decreasing the burst length of reading of video data for display and increasing the access overhead when the requested data block crosses the partition boundary. This impact on energy consumption and bandwidth usage depends on the number of partition. Thus, we can explore the number of partition until the performance declines.

4. The Proposed Architecture

In this section, we present our hardware-assisted approach to improve energy and bandwidth usage of SDRAM. We start with

the look-ahead arbiter and memory controller, and then present the row-interleaving mechanism.

4.1 Look-ahead Arbiter and Memory Controller

In a multi-master system, the bus arbiter and memory controller are of the most important system units that impact system performance and energy usage of memory. However, conventional arbitration schemes have only focused on the use and allocation of the memory bus to the requesting masters. It does not take into consideration of the access dependence which can be used to avoid unnecessary memory command. In this section, we have proposed an effective design that integrates the bus arbiter and memory controller to explore the access dependence.

Fig. 4 shows a conventional data transfer architecture for a multi-master system using the SDRAM. In this architecture, the masters assert the request signal to the arbiter indicating the transfer they wish to perform. After the arbiter grants the bus to a master based on its arbitration policy, the master drives out the address routed to the memory controller through the central multiplexer. The memory controller generates the SDRAM command to access the data in the memory.

The SDRAM is accessed by the memory controller that usually incorporates conventional bank management policy, i.e., open-page or close-page policy [1, 2], to decide whether to precharge the row currently accessed at the end of the access. In the closed page policy, the controller precharges the active row immediately after the access is completed. However, since the memory controller receives consecutive accesses locating in the same row of the SDRAM, the unnecessary row-activation and precharge operations between the accesses increase the energy consumption and access latency. In the open page policy, the active row is opened as long as possible in order to save latency due to precharge and row-activation. In this way, an open bank consumes more energy than a closed bank while there is no further request for the same bank. Obviously, it is not easy to determine a suitable bank management policy when the memory controller does not have enough access dependence information.

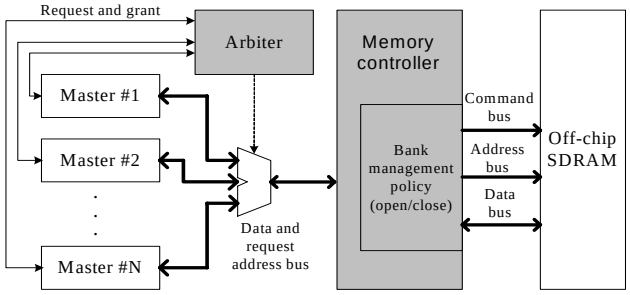


Fig. 4. Conventional data transfer architecture for multi-master system using the SDRAM

To minimize access overhead, we introduce the look-ahead arbiter which is integrated with the memory controller using an access dependence based policy (ADBP) for multi-master architecture. Fig. 5 shows the basic idea of the proposed architecture. In this design, the LAA drives out the request address and the next request address (if any) to the memory controller according to the request status. The memory controller generates the SDRAM command based on three strategies that the ADBP uses. First, if the request address and the next request address have the same bank field and row field, the active row is kept open to reduce operating current and latency. In this way, row-activation and precharge operation is eliminated. Second, if the two addresses have the different bank field, two banks are opened simultaneously to mitigate communication command overheads. Third, if there is no next request address, meaning that the request status is empty, auto-precharge is performed, which deactivates the open row automatically upon the completion of the access burst in conjunction with a specific READ or WRITE command. Furthermore, since the master also drives out its request address and the next request address (if any) to the LAA, thus, we can explore access dependence of consecutive accesses issued by the same master. To eliminate row-activation and precharge operations, the LAA promotes the master that acquires the ownership of the bus to the highest priority when the request address and the next request address from the master have the same bank address and row address.

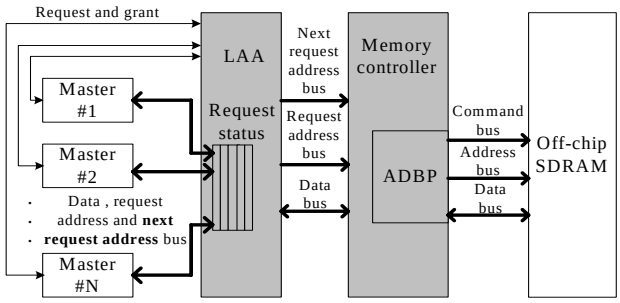


Fig. 5. Novel data transfer architecture for multi-master system using the SDRAM

4.2 Row Interleaving

When consecutive accesses are issued to the SDRAM, the current row and the next row that is referenced (if any) are usually close together in the same bank due to spatial locality. If that occurs, the SDRAM command can not be overlapped, as shown in Fig. 3 (c). Thus, we place the adjacent rows into the different banks, which is referred to row interleaving, to minimize command overhead.

We use an address bus rotation mechanism implemented between the master and arbiter, as shown in Fig. 6, to perform row interleaving. This transparent mechanism is accomplished by routing the address bus wires and not adds any overhead in terms of area or delay.

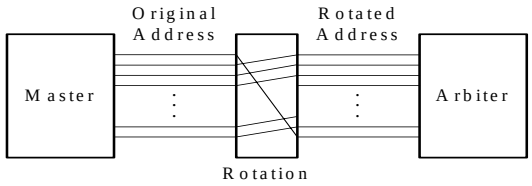
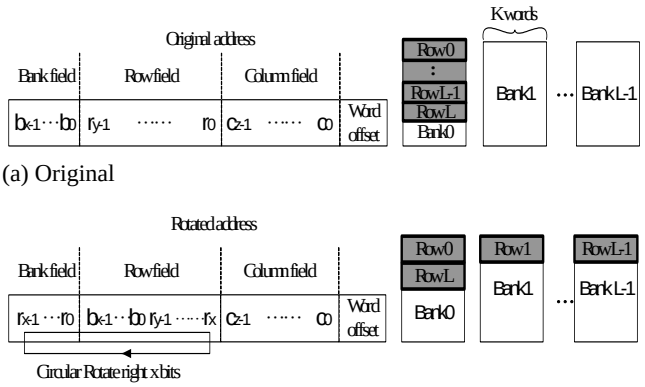


Fig. 6. Address bus rotation scheme

Fig. 7 depicts the basic address rotation policy and memory allocation. The address is divided into four contiguous fields: bank, row, column and word offset field. The bank field determines which bank is made active. The row field is used to index one of the rows of the selected bank while the column field is used to index one of the columns of the row. The word offset field selects the desired byte from the word. The bank, row, and column field are assumed to have x, y and z bits respectively. Consequently, the memory includes L (2^x) banks organized with 2^y rows and K (2^z) words. Fig. 7 (a) shows the original address bus definition and memory allocation before rotation. To perform row interleaving, all bits of the bank field and the row field are rotated right x bits. Fig. 7 (b) illustrates the address bus definition and memory allocation after rotation.



(a) Original

(b) After rotation

Fig. 7. Address bus definition and memory allocation

5. Simulation System

To evaluate the proposed design, we implement a video decoder for MPEG-4 Simple Profile at Level 3 (SP @ L3) which supports CIF (352 × 288) resolution up to 30 frames per second with the AMBA platform [9]. Fig. 8 shows the block diagram of the MPEG-4 decoder system in which various hardware co-processors are used to accelerate the required operations. The main data transferring paths among the processing units are labeled in Fig. 8. The coded bit stream input unit (CI) writes bit stream to the SDRAM. These compressed data are read out again by the coded bit stream output unit (CO), and then the bit stream is processed by both or either of the texture decoding or motion decoding depending on the macroblock type. The reconstructed data that is ready for display and reference is written to the SDRAM by the reconstruction unit (VOPR). The LAA schedules the requests according to a bus-arbitration policy. In our design, video output unit (VO) has the highest priority, followed by CO, then motion compensation unit (MC), VOPR and finally CI.

This system works at 108MHz with 32-bit data bus and the 64Mb SDRAM organized as 4 banks of 2048 rows and 256 columns by 32 bits [3]. Since the clock period is 9.26 ns, the CAS delay is 3 clock cycles and RAS-to-CAS delay is 2 clock cycles. We generate a refresh command every 1684 cycles to meet the refresh requirement. In our experiment, we use an aggressive policy to save more energy. That is, when no request is pending, the memory enters the power-down state immediately.

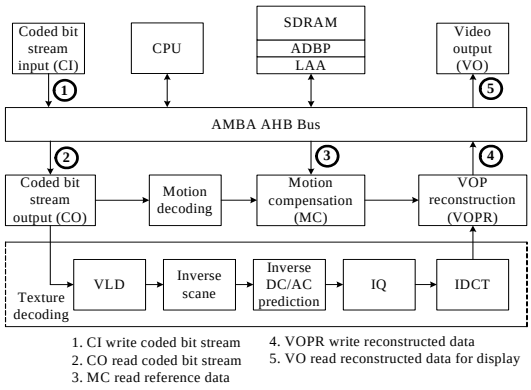


Fig. 8. Data access flow of the simulated system

6. Result of Simulation

To evaluate the energy and bandwidth performance efficiency of the LAA and the ADBP, we carry out two set of experiments. The first experiment implements the row-major mapping with the scheme we propose. For comparison purposes, we also implement a block-based mapping scheme with the block unit of 16 × 16 pixels for luminance and 8 × 8 pixels for chrominance. Table 1 shows the schemes evaluated including the block-based mapping with close-page policy (B_B_C), the block-based mapping with open-page policy (B_B_O), the row-major mapping with close-page policy (R_M_C), the row-major mapping with open-page policy (R_M_O), the row-major mapping with integrated design of the LAA and the memory controller incorporated the ADBP (R_M_L) and the R_M_L with video partitioned into “n” parts (R_M_Ln).

Table 1. The schemes evaluated

	Block based mapping	Row major mapping	Close page policy	Open page policy	LAA & ADBP	Partition into “n” parts
B_B_C	✓		✓			
B_B_O	✓			✓		
R_M_C		✓	✓			
R_M_O		✓		✓		
R_M_L		✓			✓	
R_M_Ln		✓			✓	✓

Fig. 9 shows the comparisons of energy consumption among the schemes investigated for various percentages of nonzero motion vector in a P-VOP. The energy consumed by R_M_C is used as a base. Obviously, the percentage of the nonzero motion vector has a strong impact on the energy consumption of the block-based mapping (B_B_C and B_B_O) because of reading redundant pixels. However, this only has a negligible impact on the energy consumption of row major mapping, with an average overhead of less than 2%. Using R_M_L4 significantly reduces the number of energy consumption around 32% compared to R_M_C. This saving comes from the reduction in the row-activation and precharge operations. However, R_M_L8 consumes more energy than R_M_L4 because the overhead caused by video partition increases.

Fig. 10 compares the bandwidth needed for transfer among the standard scheme (R_M_C) and the other schemes. Using R_M_L4 also significantly reduces the bandwidth by 33% compared to

R_M_C. B_B_O consumes less bandwidth when the percentage of nonzero motion vector is low, but its energy consumption is very high. We have also evaluated row interleaving scheme with low percentage of nonzero motion vector. The row interleaving is more helpful in reducing bandwidth, and therefore achieves 4.5% higher performance than R_M_L4. However, we cannot reduce energy consumption by using row interleaving only.

The I-VOP has similar results as the P-VOP. The results are shown in Fig. 11 and 12. We can observe that using R_M_L2 can achieve the best performance of bandwidth and energy consumption. Since the P-VOP is the majority of all VOPs, we propose to use R_M_L4 to achieve the best energy-delay saving in this system.

7. Conclusion

In MPEG-4 decoding system, memory mapping of video data directly affects the energy consumption and bandwidth usage of SDRAM. Using block-based mapping is inefficient, when the video data are read line-wise for display and the percentage of nonzero motion vector is high. However, when a macroblock is accessed in row-major mapping, unnecessary row-activation and precharge operations increase.

In this paper, we propose an effective design that integrates the bus arbiter and the memory controller. This design can explore the access dependence to reduce unnecessary command. We also present video data partition and row interleaving strategies that provide opportunity for eliminating or overlapping row-activation and precharge operation. Experiment results show that the row-major mapping strategy incorporated the proposed schemes can provide the best performance both in terms of energy reduction and memory bandwidth improvement, up to 32% and 33% respectively. The proposed schemes can be easily extended in other MPEG decoding system.

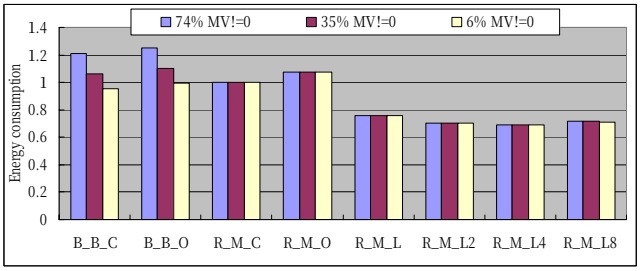


Fig. 9. Comparison of energy consumption for P-VOP

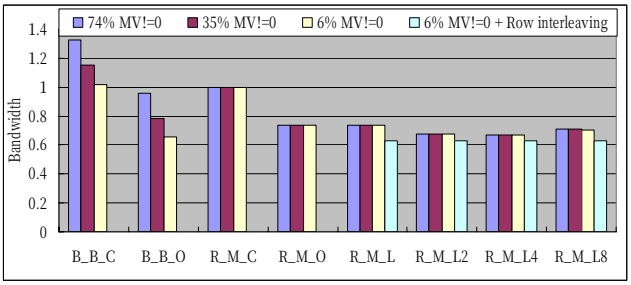


Fig. 10. Comparison of bandwidth for P-VOP

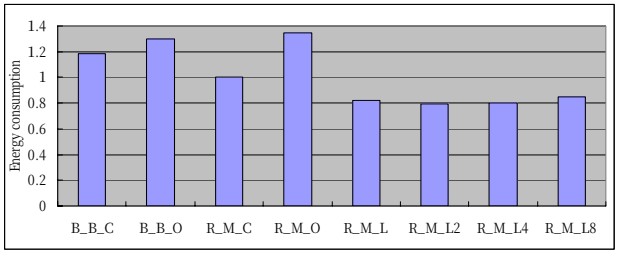


Fig. 11. Comparison of energy consumption for I-VOP

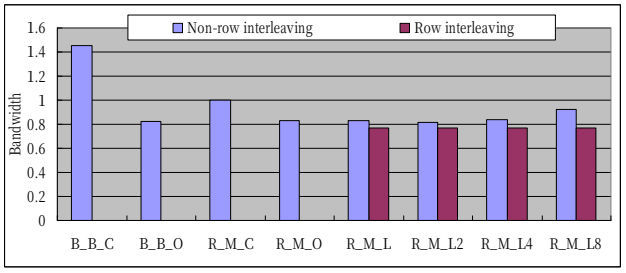


Fig. 12. Comparison of bandwidth for I-VOP

References

[1] S. Miura, K. Ayukawa, and T. Watanabe, "A Dynamic-SDRAM-Mode-Control Scheme for Low-Power Systems with a 32-bit RISC CPU," *in the Proceeding of the International Symposium on Low Power Electronics and Designs*, 2001.

[2] X. Fan, C. S. Ellis and A. R. Lebeck, "Memory Controller Policies for DRAM Power Management," *in the Proceeding of the International Symposium on Low Power Electronics and Designs*, 2001.

[3] Samsung SDRAM 64Mb, H-die (x32) data sheet, 2004.

[4] H. Kim and I.-C. Park, "High-Performance and Low-Power Memory-Interface Architecture for Video Processing Applications," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 11, No.11, pp. 1160-170, November 2001.

[5] E. G. T. Jaspers and P. H. N. de with, "Bandwidth reduction for video processing in consumer systems," *IEEE Transactions on Consumer Electronics*, Vol. 47, No.4, pp. 885-894, November 2001.

[6] V. Delaluz, M. Kandemir, M. N. Vijaykrishnan, and et al., " Hardware and Software Techniques for Controlling DRAM Power Modes," *IEEE Transactions on Computers*, Vol. 50, No. 11, pp. 1154-1173, November 2001.

[7] S. Rixner, W. Dally, and et al., " Memory Access Scheduling," in the *Proceedings of the 27th International Symposium on Computer Architecture*, 2000.

[8] Micron Technical Note TN-46-03 "Calculating Memory System Power For DDR," 2001.

[9] AMBA Specification Rev 2.0, 2000.