Optimal Data Mapping for Motion Compensation in H.264 Video Decoding

Guo-Shiuan Yu, and Tian Sheuan Chang
Dep. Electronics Engineering,
National Chiao-Tung University,
1001 Ta-Hsueh Rd., Hsinchu, Taiwan
e-mail: {isis, tschang}@twins.ee.nctu.edu.tw

Abstract— **Long initial access cycles of SDRAM are the major performance burden of motion compensation in a video decoder. To minimize its effect while improve overall available memory bandwidth, this paper presents an optimal data mapping scheme for motion compensation in H.264 video coding. This scheme allocates the video data into suitable address and bank according to the access characteristics of SDRAM access and address transition in motion compensation. The resulted allocation can reduce the required bandwidth of motion compensation by 36% when compared to the previous design for 525SD video sequences.**

I. Introduction

Memory access dominates the performance in a video decoder, especially in motion compensation. In a typical video decoder, motion compensation unit will access the required reference data from external SDRAM systems. However, a typical SDARM access consists of a long initial cycle to open a memory row followed by continuous addressed burst access. Thus, if the memory access has discontinuous addresses, it will suffer frequent initial cycles and thus results in performance degradation and larger memory bandwidth. Thus, how to allocate the data to the physical SDRAM is an important task for video decoder.

Targeted to video codec applications, many papers have been proposed to improve SDRAM bandwidth utilization and achieve efficient memory access. Li [1] develops a bus arbitration algorithm optimized with different processing unit to meet the real-time performance. Ling's controller[2] schedules DRAM accesses in pre-determined order to lower the peak bus bandwidth. Kim's memory interface [3] adopts an array-translation technique to reduce power consumption and increase memory bandwidth. Park's history-based memory controller [4] reduces page break to achieve energy and memory latency reduction.

For H.264 application, Kang's AHB based scalable bus architecture and dual memory controller[5] supports 1080 HD under 130MHz. Zhu's SDRAM controller[6] employs the main idea of Kim's memory interface to HDTV application. It focuses on data arrangement and memory mapping to reduce page active overheads so that it not only improve throughput but also provides lower power consumption. However, it doesn't take the memory operation scheduling into consideration. With careful scheduling, extra bandwidth due to page active operation can be reduced.

In this paper, we combine the data mapping and operation scheduling in our design to minimize the SDRAM initial cycles and thus decrease the bandwidth requirement for real-time decoding. To find the optimal mapping, we first use a simple analytical model to find the best data mapping in theory and in practice. Then, we use real video sequences to validate the mapping. Furthermore, we do not only consider the access within a single motion compensation operation for a block (called intra request), which has high probabilities for continuous address, but also consider the access between the blocks (called inter request) by operation scheduling. The resulted design can save 37% of memory bandwidth compared with the previous approaches.

The rest of the paper is organized as follows. First, we brief overview the motion compensation in H.264 video decoding and SDRAM memory access in Section II. Then we present our analytical model for intra request and its simulated results in Section III. Furthermore, we present our operation scheduling for inter request in Section IV. The final simulation results are shown in Section V. Finally, we conclude this paper in Section VI.

II. Overview

A. Basics of SDRAM access

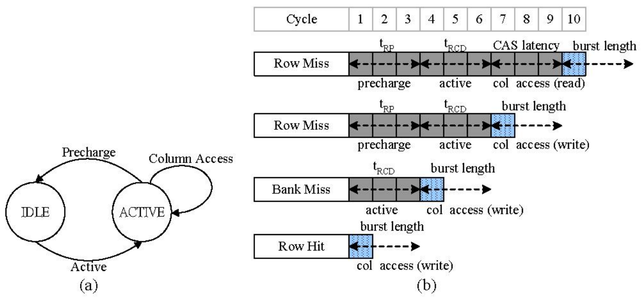
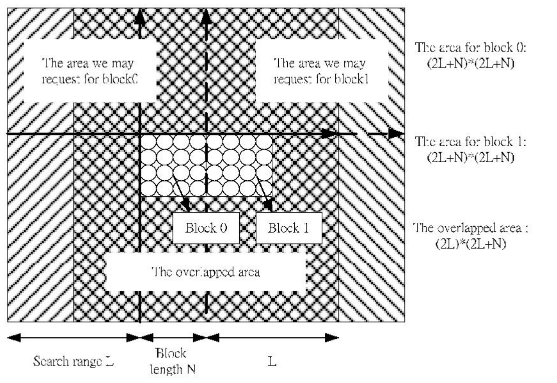


Fig. 1. (a) Simplified bank state diagram, and (b) access latencies of different access statuses.[7].

The cycle for a complete SDRAM access deeply depends on the state of the bank addressed by the SDRAM access. Fig. 1(a) and Fig. 1(b) show a simplified bank state diagram and the access latencies due to different access statuses: bank miss, row miss, and row hit. From a data access viewpoint, low cycle count in the row hit condition is preferred than those in bank miss and row miss. Thus, how to minimize such miss is critical in SDRAM performance. A more completed discussion on various access latencies of a SDRAM access can be found in Lee's paper[7].

B. Memory access in motion compensation

For video applications, the memory request is usually to get a determined size of rectangle image from frame memory like those in motion compensation, intra prediction and deblocking filter process. These data are continuous in spatial domain and the area we may request between two consecutive blocks has high probability to be overlapped. For instance, when we process motion compensation, the required data is bounded by its block size and search range set during encoding. If the search range is L and the block length is N, a 2L by 2L+N rectangle is overlapped. Data in this region has high probability in the opened bank due to previous block access. Thus we can find a method to avoid the row miss and improve bandwidth utilization. Fig. 2 illustrates an example to explain this characteristic.



The area we may request for block0
The area we may request for block1
The area for block 0: (2L+N)*(2L+N)
The area for block 1: (2L+N)*(2L+N)
The overlapped area : (2L)*(2L+N)
Block 0
Block 1
The overlapped area
Search range L
Block length N
L

Fig. 2. Possible required area between adjacent blocks

III. INTRA REQUEST OPTIMIZATION

A. Analytical analysis

According to the characteristics of video data, we can derive the translation between physical location in memory and pixel coordinates in spatial domain to reduce the row miss.

To ease analysis without loss of generality, we degrade this 2-D problem to 1-D domain. Assume that a SDRAM row contains L pixels and N continuous data are requested. The situation of row miss could be as follows. For the case without row misses, the starting point shall lie in the first L-N position of the row. However, if the starting points lie in last N-1 pixels, row miss happened. Assuming the probability of starting point position is uniform distributed, the probability of row miss is

$$p_{row-miss-1D} = \frac{N-1}{L} \tag{1}$$

For constant data length N, larger size of row means fewer row miss and the longer data length leads to higher row-miss probability with fixed row size. Extending above observations to 2-D domain, the total row miss with horizontal row miss and vertical row miss is

$$p_{row-miss-2D} = \frac{N_X - 1}{L_X} + \frac{N_Y - 1}{L_Y} \tag{2}$$

where L_X, and L_Y denote the length of memory window in horizontal and vertical respectively. However, the row size is fixed for a certain type of SDRAM, which implies $L_X * L_Y = \Omega$. Thus, the width and height of the row window are affected each other. The row miss probability should be adjusted as follows:

$$p_{row-miss-2D} = \frac{N_X - 1}{L_X} + \frac{N_Y - 1}{\Omega / L_X} = \frac{\Omega * (N_X - 1) + {L_X}^2 * (N_Y - 1)}{L_X * \Omega} \tag{3}$$

where N_X, N_Y, $L_X > 0$ and Ω denotes the row size. For worst case, $N_X = N_Y$ is equal to the maximum data length. Thus, the row-miss probability has the minimum value when L_X is equal to $\sqrt{\Omega}$.

Above formula is quite simplified. To be practical, we further consider the characteristics of real video sequences. In H.264 the data length we may request for motion compensation is 4, 8, 9, 13, 16 and 21 pixels according to its block modes and sub-pixel motion vectors. Besides, the probability of starting point does not distribute uniformly in many video sequences. The position number that is divisible by 4, which means the 0th, 4th, 8th, 12th ... 4kth ... pixels of row window in vertical or in horizontal, has higher probability to appear. Generally speaking, p_{4k} is 1.5 to 2.5 times larger than others according to our simulation, where p_{4k} denotes the probability of 0th, 4th, 8th, 12th ... 4kth ... positions. This is because the smallest block length is 4 and the effect of zero motion vector. The blocks with zero motion vectors are usually referenced for background image. For larger quantization parameter, this effect becomes more significant. Thus, the row miss probability of H.264 motion compensation is

$$p_{row-miss-MC} = \sum_{N_X=4,8,9,13,16,21} (p_{N_X} * \sum_{n=L_X-N_X+1}^{L_X} p_n) + \sum_{N_Y=4,8,9,13,16,21} (p_{N_Y} * \sum_{m=L_Y-N_Y+1}^{L_Y} p_m)$$

$$= \frac{5*P_{N_X 4} + 11*P_{N_X 9} + 10*P_{N_X 8} + 16*P_{N_X 13} + 20*P_{N_X 16} + 25*P_{N_X 21}}{L_X + L_X / 4} + \frac{5*P_{N_Y 4} + 11*P_{N_Y 9} + 10*P_{N_Y 8} + 16*P_{N_Y 13} + 20*P_{N_Y 16} + 25*P_{N_Y 21}}{L_Y + L_Y / 4} \tag{4}$$

where the P_{NX4} is the probability of data length equal to 4 pixels in horizontal and we assume p_{4k} is twice than others for simplification.

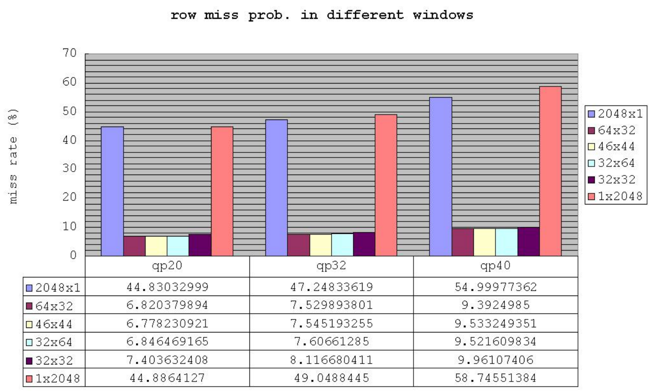
Combining Eq. (4) with simulation statistics, we can find the row miss probability function is

$$\frac{16.866 * \Omega + 16.133 * {L_X}^2}{L_x * \Omega} \tag{5}$$

For $L_X > 0$, this function has a minimum vale when L_X equal to $\sqrt{1.04\Omega}$. A typical Ω, row size of a SDRAM, can be 16384, 8192 or 4096 bits, which is 2048, 1024 or 512 pixels. For our targeted SDRAM, 2048 pixels in a row, the optimized window size should be a 46x44 rectangle. However, it is hard to implement the translation with the 46x44 windows. We adjust the window size to 64x32. Because 32 and 64 are powers of 2, the translation can be easily implemented with bit shift.

B. *Simulation results*

Fig. 3 shows the statistics of row miss in different window size. The test sequences are crew, night, sailormen, and harbour in 525 SD frame size. Comparing with the linear translation like 1x2048 and 2048x1 window size, the 64x32 mapping reduces about 84% of row miss rate. Compared with the optimal 46x44 mapping, the 64x32 mapping has slightly low row miss due to more frequent horizontal motion and 4x4 block size. The rows with large size can decrease the probability of row break, thus the 32x32 window has higher row miss count than 64x32. Due to the video sequences characteristics, the occurrence of horizontal break is more frequent than vertical. Thus, the 64x32 mapping can lead to better performance.



	qp20	qp32	qp40
2048x1	44.83032999	47.24833619	54.99977362
64x32	6.820379894	7.529893801	9.3924985
46x44	6.778230921	7.545193255	9.533249351
32x64	6.846469165	7.60661285	9.521609834
32x32	7.403632408	8.116680411	9.96107406
1x2048	44.8864127	49.0488445	58.74551384

Fig. 3. Miss rate in different row windows

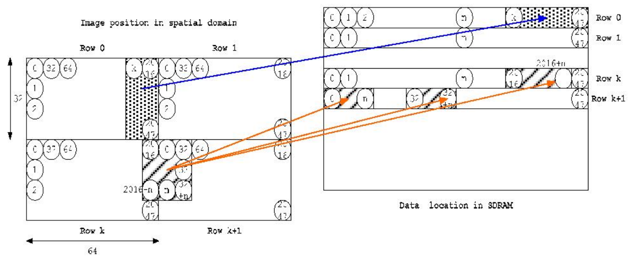


Fig. 4. Translation of physical location and image position

C. *The memory mappings and operations*

Fig. 4 illustrates the mapping between physical location in memory and image position in spatial domain. The latency of single request can be reduced with bank interleaving operation as shown in Fig. 5.

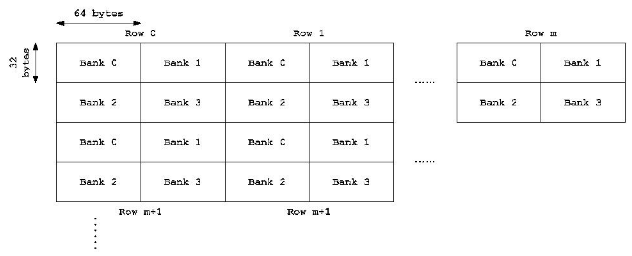


Fig. 5. Bank arrangement with optimization

With the data arrangement mentioned before, the requests can be classified to three kinds as shown in Fig. 6, by assuming open all required rows at the beginning of every request to reduce the control overhead and ease the hardware design.

Case 1: *all data of single access are contained in a row.*

It is clear that this case introduces no row miss, since all the data to be requested are stored in a row. The memory operation contains the row activation, data reading and precharging. Fig. 7 shows the operations under case 1. L+4 cycles are needed to complete this access, where L denotes the number of accessed data.

Case 2: *all data of single access are contained in two rows. The data may be discontinuous in horizontal or in vertical as illustrated in* Fig. 6.

In this case, we suffer two row miss since the accessed data are contained in different rows. However, they are in different banks. We can shorten the latency with bank alternating access. Fig. 7 shows the operations of case 2. We open the rows we may access, read the data in determined order and then precharge the opened row. Total cycle count is L+5.

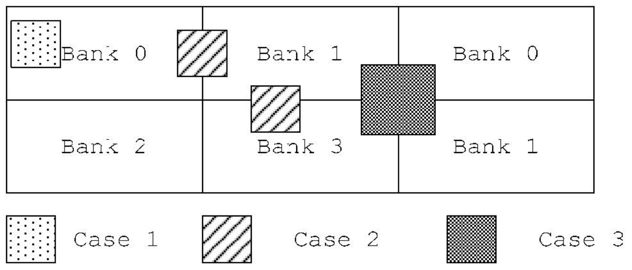


Fig. 6. Request classification

Case 3: *all data of single access are stored in four rows. The data break in horizontal and vertical as illustrated in* Fig. 6.

Four row breaks are encountered in this case. Due to the limitation of SDRAM access cycle, one cycle latency is introduced to meet timing requirement. Fig. 7 illustrates the operations. The number of total cycles is L+7.

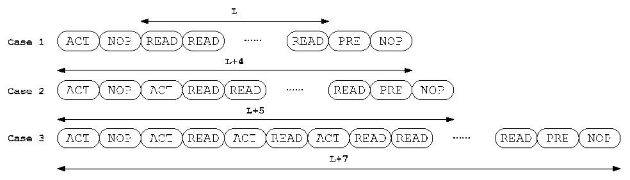
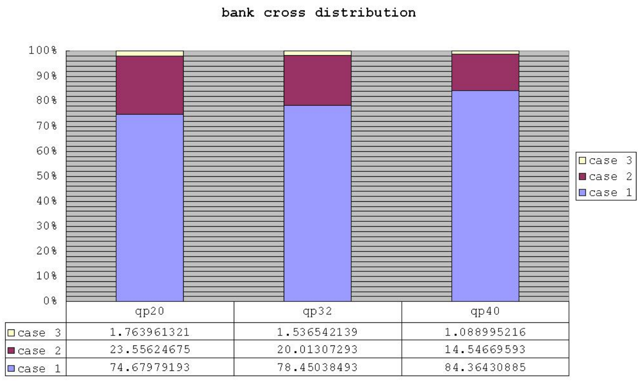


Fig. 7. Request operations in different cases

The probability distribution of these cases is shown in Fig. 8. With the increasing quantization parameters, the cross-bank cases decrease rapidly due to more zero motion vector in high QP. Besides, this result also shows that case 1 occurs most in total accesses. This means we usually only need to open one row in a single request and thus reduce extra bandwidth requirement.



	qp20	qp32	qp40
□ case 3	1.763961321	1.536542139	1.088995216
■ case 2	23.55624675	20.01307293	14.54669593
■ case 1	74.67979193	78.45038493	84.36430885

Fig. 8. Distribution of access types

IV. Inter request optimization

In intra-request optimization, we have determined the optimized data mapping to reduce the row misses. Furthermore, for successive requests, the requested data has high probability to be stored in the same row due to overlapped search range. This access can get the same benefit as the intra request without any row miss. However, there is still a certain amount of data stored in different rows. Thus row miss will occur if closing unused rows by precharging the banks and opening the new rows. To reduce such row misses, we shall consider when and how to close the row by precharing.

There are two types of precharge command, precharge all banks or precharge single bank. Precharing each bank separately is preferred to easily reduce row miss. However, individual precharging has overheads to send more explicit commands to close corresponding rows. In contrast, only one command is needed for all banks precharging. With single bank precharging, we can save one row break from two row breaks to one break, which is relatively small when compared with the one from one break to zero break. The actual gain by simulation is about 0.1% in total memory access cycles. The benefit is so small that we can neglect it. Thus, we choose all banks precharging as our solution considering the hardware control cost and bandwidth performance.

V. Simulation Results

With above intra and inter request optimization, we can efficiently reduce the miss rate from 6.8% (without inter-request optimization) to 1.8% from simulation. Table I shows the comparisons of bandwidth requirement with other designs, while the data of [6] is from our implementation. Our proposed scheme can reduce extra memory access overhead, needs less time to transfer data, and thus save 37% of bandwidth compared to Zhu's design at 525SD video size.

Table I. Comparisons of Bandwidth requirement.

Scheme	Format	Bandwidth (MBps)
proposed	QCIF	2.60
	CIF	12.00
	525SD	46.96
	720HD	135.54
Zhu[6]	525SD	73.85
	720HD	187.25

VI. Conclusion

This paper presents an optimal data mapping for motion compensation used in H.264 video coding. Our scheme can save 37% of memory bandwidth when compared to the previous approach. This scheme can be applied to the memory controller design and can co-work with the selected on-chip-bus. Besides, this scheme can be also applied to other types of memory access in video decoding since these types are subset of that in motion compensation.

Acknolwedgement

This research is sponsored by National Science Council, Taiwan, R.O.C., under grant NSC-922215-E-009-014.

References

[1] J.-H. Li, N. Ling, "Architecture and bus-arbitration schemes for MPEG-2 video decoder," IEEE Transaction on Circuits and Systems for Video Technology, vol. 9, pp.727 – 736, Aug. 1999

[2] N. Ling, N.-T. Wang, D.-J. Ho, "An efficient controller scheme for MPEG-2 video decoder," IEEE Transaction on Consumer Electronics, vol. 44, pp.451 – 458, May 1998

[3] H. Kim, I.-C. Park, "High-performance and low-power memory-interface architecture for video processing applications," IEEE Transaction on Circuits and Systems for Video Technology, vol. 11, pp. 1160 – 1170, Nov. 2001

[4] S.-I. Park, Y. Yi, I.-C. Park, "High performance memory mode control for HDTV decoders," IEEE Transaction on Consumer Electronics, vol. 49, pp.1348 – 1353, Nov. 2003

[5] H.-Y. Kang, K.-A. Jeong, J.-Y. Bae, Y.-S. Lee, S.-H. Lee, "MPEG4 AVC/H.264 decoder with scalable bus architecture and dual memory controller," proc. International Symposium on Circuits and Systems, vol. 2, pp. II - 145-8, May 2004

[6] J. Zhu, L. Hou, W. Wu, R. Wang, C. Huang, J.-T. Li, "High Performance Synchronous DRAMs Controller in H.264 HDTV Decoder", proc. International Conference on Solid-State and Integrated Circuits Technology, vol. 3, pp. 1621 – 1624, Oct. 2004

[7] K.-B. Lee and C.-W. Jen, "Design and verification for configurable memory controller - Memory interface socket soft IP," *Journal of the Chinese Institute of Electrical Engineering*, vol. 8, no. 4, pp.309–323, 2001.