MOTION COMPENSATION MEMORY ACCESS OPTIMIZATION STRATEGIES FOR H.264/AVC DECODER

RongGang Wang[1,2], JinTao Li[1], Chao Huang[1]

[1]Digital Technology Lab, Institute of Computing Technology, Chinese Academy of Sciences
P. O. Box 2704, Beijing, 100080, P. R. China
[2]Graduate School of the Chinese Academy of Sciences
Email: grwang@ict.ac.cn

ABSTRACT

H.264/AVC is the latest standard for video coding drafted jointly by the ISO/IEC Moving Picture Experts Group and the ITU-T Video Coding Experts Group. H.264/AVC provides up to 50% gains in compression efficiency over a wide range of bit rates and video resolutions compared to previous standards. On the other hand, the decoder complexity is about four times that of MPEG-2 and two times that of MPEG-4 Visual Simple Profile. In VLSI implement of H.264/AVC decoder, off-chip memory access is the main time and power consuming operation, and motion compensation module is the main memory access bottleneck. This paper proposes four optimization strategies to reduce memory access data cycles and improve memory data bus utilization. Experiment results show that about 60% data cycles can be reduced and more than 20% memory data bus utilization can be improved by these strategies over typical test sequences.

1. INTRODUCTION

H.264/AVC adopts a series of new techniques to improve coding efficiency, such as flexible block size and quarter-pixel motion compensation, spatial intra prediction, in loop de-blocking filters and context-based adaptive binary arithmetic coding etc. As a result, H.264/AVC gains up to 50% in compression efficiency over a wide range of bit rates and video resolutions compared to previous standards. On the other hand, the decoder complexity is about four times that of MPEG-2 and two times that of MPEG-4 Visual Simple Profile. In VLSI implement of H.264/AVC decoder, off-chip memory access is the main time and power consuming operation.

There are four main modules require off-chip memory access in H.264/AVC decoder, which are reference picture store, de-blocking, display feeder and motion compensation. In addition, there some other off-chip memory access requirements such as stream store and motion vector store, as they are much less than the four main modules, they are ignored from consideration. Table 1 lists the memory access ratio of each module in H.264/AVC decoder in the worst case. We can see that motion compensation module is the main memory access bottleneck of H.264/AVC decoder. Therefore, minimization of memory access operations is a key consideration in H.264/AVC decoder VLSI design.

This paper utilizes the memory access behavior characteristics of motion compensation and proposes four optimization tragedies to reduce memory access data cycles and improve memory data bus utilization. Experiment results show that about 60% data cycles can be reduced and more than 20% memory data bus utilization can be improved by these strategies over typical test sequences.

In section 2 the motion compensation memory access behavior of H.264/AVC decoder is analyzed in detail. Section 3 provides four optimization tragedies based on the motion compensation memory access behavior characteristics. Experiment results are presented in section 4 and in section 5 a conclusion is given.

Table 1. Memory access ratio of each H.264/AVC decoder module (W: frame width, H: frame height)

Module name	Max memory access bytes	Ratio(≈)
Reference picture store	W*H + 2*(W/2)*(H/2)	10%
De-blocking	(W/16)*(H/16-1)*16*4*2*2	5%
Display feeder	W*H + 2*(W/2)*(H/2)	10%
Motion compensation	(W/16)*(H/16)*16*(9*9+2*3*3)*2	75%
Total	≈16*W*H	

2. H.264/AVC DECODER MOTION COMPENSATION MEMORY ACCESS BEHAVIOR ANALYSIS

H.264/AVC CODEC adopts block-based motion compensation, the same principle used by previous major coding standards since H.261. There are two main differences from other standards, one is the support for

variable block size and the other is fine sub-pixel motion vectors [2].

2.1. Tree structured motion compensation

H.264/AVC supports variable motion compensation block sizes ranging from 16x16 to 4x4 luminance samples with many options between the two. The luma component of each macroblock may be split up in 4 ways: 16x16, 16x8, 8x16 or 8x8. Each of the sub-divided regions is a macroblock partition. If the 8x8 mode is chosen, each of the four 8x8 macroblock partitions within the macroblock may be split in a further 4 ways: 8x8, 8x4, 4x8 or 4x4. These partitions and sub-partitions give rise to a large number of possible combinations within each macroblock. This method of partitioning macroblocks into motion compensated sub-blocks of varying size is known as tree structured motion compensation.

The resolution of each chroma component in a macroblock (Cb and Cr) is half that of the luma component. Each chroma block is partitioned in the same way as the luma component, except that the partition sizes have exactly half the horizontal and vertical resolution.

2.2. Sub-pixel motion vectors

Each partition in an inter-coded macroblock is predicted from an area of the same size in a reference picture. The motion vector between the two areas has sub-pixel resolution. The luma and chroma samples at sub-pixel positions do not exist in the reference picture and so it is necessary to create them using interpolation from nearby image samples.

The interpolated sub-pixel samples are generated as follows. In the luma component of the reference picture, first the half-pixel samples are generated (Fig. 1). Each half-pixel sample that is adjacent to two full-pixel samples is interpolated from full-pixel samples using a 6 tap Finite Impulse Response (FIR) filter(1/32, -5/32, 20/32, 20/32, -5/32, 1/32). Once all of the samples adjacent to full-pixel samples have been calculated, the remaining half-pixel positions are calculated by interpolating between 6 horizontal or vertical half-pixel samples from the first set of operations. Once all the half-pixel samples are available, the quarter-pixel positions are produced by linear interpolation. In order to interpolate an M*N luma partition (M is the width and N is the height of current partition), an (M+5)*(N+5) reference data block is required to be read from off-chip memory.

In 4:2:0 sampling format, quarter-pixel resolution motion vectors in the luma component will require eighth-pixel resolution vectors in the chroma components. Interpolated samples are generated at eighth-pixel intervals between full-pixel samples in each chroma component. In this case, two dimensions linear interpolation is used to produce every eighth-pixel chroma sample. In order to interpolate an M*N chroma partition, an (M+1)*(N+1) reference data block is required to be read from off-chip memory.

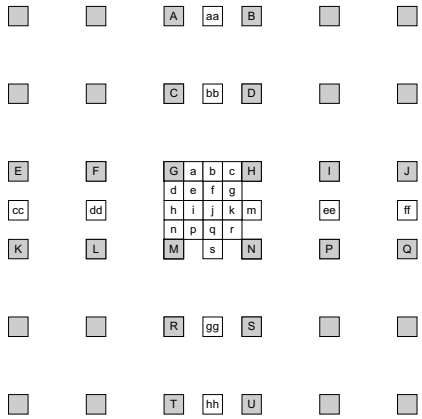

Fig.1. Full-pixel samples (shaded blocks with upper-case letters) and sub-pixel sample positions (un-shaded blocks with lower-case letters) for quarter-pixel sample luma interpolation.

3. MOTION COMPENSATION MEMORY ACCESS OPTIMIZATION STRATEGIES

From the analysis in section 2, we can see that since H.264/AVC support small block size motion compensation and quarter-pixel motion vector (luma component), more reference data is required to be read for motion compensation compared to previous standards and it is very significant to optimize motion compensation memory access in H.264/AVC decoder. In this Section four optimization tragedies of motion compensation memory access are proposed. To facilitate the evaluation of the optimization strategies we define several terms:

Valid reference data: the necessary reference data during motion compensation.

Data bus bandwidth: bit width of memory access data bus.

Data cycle: the clock cycle during which reference data is read from off-chip memory.

Data bus utilization: the ratio between valid reference data and reference data read in data cycles (the ratio may not be 100%, as valid reference data may not be aligned with memory data bus bandwidth).

Data cycles per MB: the average data cycles needed by motion compensating an inter macroblock.

3.1.Strategy 1: variable block size reference data reading

H.264/AVC adopted tree structure variable block size motion compensation, JM73 unified the variable block size into the smallest block size (luma in 4x4), then only the smallest partition interpolation module is competent in motion compensation. While the problem is that the

smallest partition motion compensation has to read more redundant reference data compared to variable block size motion compensation. Provided to predict a single direction luma component 8x8 partition, in the worst case valid reference data are 13*13=169 bytes, But when it is unified into four 4x4 blocks, to predict each 4x4 partition subsequently valid reference data is 9*9*4=324 bytes. As in Fig. 2, the gray part is redundant reference data of the up left 4x4 partition, the total redundancies (155 bytes) are nearly equal to the whole valid reference data of current 8x8 partition. It is expected that reading reference data from off-chip memory according to variable block size will save quite a lot of data cycles.

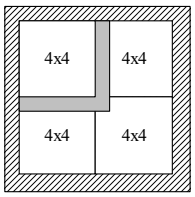

Fig.2. The up left 4x4 block redundant reference data induced by divided 8x8 block into four 4x4 blocks (indicated by grey part).

3.2. Strategy 2: direct interpolation scheme reference data read

During sub-pixel motion compensation, two interpolation schemes can be selected [3]. One is subsequent interpolation scheme, where every pixel on every sub-pixel position is interpolated; the other is direct interpolation scheme which only interpolates the sub-pixel positions that are used in the motion compensated prediction. In decoder, only one pixel in the sixteen positions is needed for motion compensation prediction, so direct interpolation scheme is adopted. Table 2 summaries the interpolation filters and the valid reference data size for one M*N partition of each interpolation position as in Fig.1 (M is the width and N is the height of current partition). Data cycles can be further reduced by this strategy.

Table 2. The interpolation filters and reference data size of one M*N luma partition

Interpolation position	Interpolation filters	Valid reference data size
One full-pixel position: G	No interpolation necessary	M*N
Positions a, b, c	6 tap horizontal filter	(M+5)*N
Positions d, h, n	6 tap vertical filter	M*(N+5)
Positions e, f, g, i, j, k, p, q, r	6*6 tap filter	(M+5)*(N+5)

3.3. Strategy 3: Cb component and Cr component combined reference data reading

Cb and Cr component of one partition share the same motion vector and their reference data shifting is identical in reference frame. Therefore, if Cb and Cr reference data can be combined in a interlaced mode as in Fig.3, valid reference data of Cb component and Cr component can be read from off-chip memory together by doubling shifting samples in horizontal direction. By this way the data bus utilization can be improved greatly and data cycles can be saved as well.

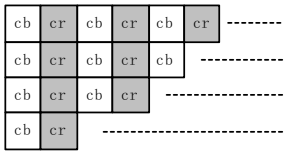

Fig.3.Cb component and Cr component reference data combined in a interlaced mode

3.4.strategy 4: luma component and chroma component reference data reading though separately memory access channels

As H.264/AVC adopted tree structure motion compensation, luma component partition can be small to 4x4 and chroma component partition can be small to 2x2. During motion compensation, random access for small blocks of data is more frequent than previous video coding standards, in this case caches do not necessarily help for saving data cycles. one effective solution is to adopt two memory channels, one is for luma component and the other is for chroma component, by this way the two channels can be accessed in parallel, as a result the clocks of reading reference data can be reduced. Moreover, the two memory channel can have different bandwidth, as chroma reference data is much less than its luma counterpart, the chroma component memory access channel adopts shorter bandwidth than luma component memory access channel, which can improve the data bus utilization.

4. EXPERIMENT RESULTS

We performed four subsequent experiments based on JM73. Each targeted a particular memory access optimization strategy presented in section 3. The first experiment evaluated the effect of variable block size reference data reading, second experiment further evaluated the effect of direct interpolation mode reference data reading based on first experiment, third experiment targeted on the effect of Cb component and Cr component combined reference data reading based on the pervious two experiments, at last one 64 bit bandwidth memory access channel was replaced by two memory access channels, one is luma component memory channel with 48 bit bandwidth, the other is chroma component memory

channel with 16 bit bandwidth to testify the effect of double memory channels based on the three previous experiments. All coders used only one I-picture at the beginning of a sequence, and two B-pictures were inserted between each two successive P-pictures, the QP of I-picture and P-picture was set to 28, the QP of B-picture was set to 30, temporal direct mode was selected. Full search motion estimation with a range of ±16 integer pixel was used by encoders along with the Lagrangian Coder Control [4].

Table 3 and Table 4 summarized the memory access optimization effect of strategy i (i=1, 2, 3, 4) over the combination of strategy 0…i-1, strategy 0 means that there is no optimization.

Experiment results show that each strategy saved data cycles or improved data bus utilization upon previous ones. on the other hand, the optimization effect is somewhat depended on sequence characteristic. The optimization effect of still and low spatial detail sequences is more notable than sequences with complex motion and high spatial detail. On the average, about 60% data cycles can be reduced and more than 20% memory data bus utilization can be improved by combination of the four strategies. The first three strategies reduced quite a lot of data cycles and the last two strategies improved the data bus utilization evidently.

5. CONCLUSION

This paper proposed four motion compensation memory access optimization strategies for H.264/AVC decoder. Experiment results testified their effective. On the average, about 60% data cycles can be reduced and more than 20% memory data bus utilization can be improved by these strategies. In practice, the four strategies can be separately or jointly adopted to optimize the memory access performance of motion compensation in H.264/AVC decoder. We have adopted these strategies in the design of H.264/AVC HD decoder chip named as "LIFEVIEW-1".

6. REFERENCES

[1] Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, "Draft ITU-T recommendation and final draft international standard of joint video specification (ITU-T Rec. H.264/ISO/IEC 14 496-10 AVC," JVTG050, 2003..

[2] Iain E. G. Richardson, "H.264 and MPEG-4 Video Compression", John Wiley & Sons, ISBN 0-470-84837-5 , September 2003, 196-201(2003).

[3] T. Wedi, "Results on complexity and coding performance investigations: Displacement vector resolution and interpolation filter tap size", VCEG-M46, Mar. 2001.

[4] T. Wiegand, H. Schwarz, A. Joch, F. Kossentini, and G. J. Sullivan,"Rate-constrained coder control and comparison of video coding standards," IEEE Trans. Circuits Syst. Video Technol., vol. 13, pp. 688–703, July 2003.

Table 3. Data cycles saved by each optimization strategy

Sequence name	Data cycles per MB without optimization	Strategy 1		Strategy 2		Strategy 3		Strategy 4		Totally saved
		Data cycles per MB	Saved	Data cycles per MB	Saved	Data cycles per MB	Saved	Data cycles per MB	Saved	
Akiyo	657	574	14%	308	46%	243	26%	235	3%	64%
Paris	621	532	14%	316	40%	256	19%	223	13%	64%
News	636	542	14%	300	44%	238	21%	224	6%	64%
Mobile	576	406	29%	340	16%	298	12%	261	12%	55%
Tempete	578	455	21%	380	16%	330	13%	296	10%	49%
Coastguard	521	308	40%	233	24%	199	14%	171	14%	67%

Table4. Memory data bus utilization improved by each optimization strategy

Sequence name	Data bus utilization without optimization	Strategy 1		Strategy 2		Strategy 3		Strategy 4		Totally Improved
		Data bus utilization	Improved	Data bus utilization	Improved	Data bus utilization	Improved	Data bus utilization	Improved	
Akiyo	48.53%	51.88%	3.35%	45.49%	-6.39%	57.71%	10.22%	72.40%	14.69%	23.87%
Paris	48.53%	52.42%	3.89%	46.97%	-5.45%	57.96%	10.99%	72.36%	14.40%	23.83%
News	48.52%	52.13%	3.61%	46.00%	-6.13%	57.89%	11.89%	72.37%	14.48%	23.85%
Mobile	48.53%	56.00%	7.47%	53.13%	-2.87%	60.69%	7.56%	73.60%	12.91%	25.07%
Tempete	48.52%	54.35%	5.83%	51.89%	-2.46%	59.61%	7.72%	72.06%	12.45%	23.54%
Coastguard	48.50%	55.97%	7.47%	53.64%	-2.33%	62.46%	8.82%	75.31%	12.85%	26.81%