A Memory Reduction Approach for MPEG Video Coding

Kajiwara, Naoki
Department of Electronics Engineering and Computer Science, Fukuoka University

Moshnyaga, Vasily G.
Department of Electronics Engineering and Computer Science, Fukuoka University

https://doi.org/10.15017/7659

出版情報：International Symposium on Intelligent Signal Processing and Communication Systems || || p399-402, pp.399-402, 2003-12
バージョン：
権利関係：

A MEMORY REDUCTION APPROACH FOR MPEG VIDEO CODING

Naoki Kajiwara and Vasily G. Moshnyaga

Department of Electronics Engineering and Computer Science, Fukuoka University
8-19-1 Nanakuma, Jonan-ku, Fukuoka 814-0180, JAPAN

ABSTRACT

This paper presents an architectural enhancement for reducing memory requirements of MPEG video coding based on incremental memory sharing between the reconstructed picture frames. The method exploits the temporal locality of block-based hybrid coding by dynamically replacing the processed macroblocks of the previously reconstructed picture frame with macroblocks of a newly reconstructed picture frame. Simulation results show that using this method we can reduce the total memory size by a 13%-32% for bidirectional prediction and almost half for unidirectional prediction schemes without any impact on image quality and throughput.

1. INTRODUCTION

1.1. Motivation

The growing demand for multimedia wireless video systems, such as portable video phones, hand-held DVD devices, mobile video conferencing units, etc. motivates the need for small and low power video encoders. In modern MPEG video encoder, such as [1, 2], memory is the biggest contributor to area and power consumption: it accounts over 30% of the total encoding energy and occupies almost half of the whole system area.

Typically there are two main memories in MPEG encoder: work memory and frame memory, as shown in Figure 1. The frame memory stores a reconstructed image of the reference frame to be used in motion estimation to eliminate the temporal redundancy of current image frame. To support bidirectional prediction mode of the MPEG, the frame memory keeps two reference frames (which can be I or P frames). The work memory is used to reorder the input video stream into a pre-defined coding sequence. If the input order is as shown in Figure 2, the work memory stores two B-pictures in order to ensure the following encoding order: I(1) P(4) B(2) B(3) P(7) B(5) B(6) P(10) B(8) B(9). To satisfy MPEG-4 level 3 processing requirements (CIF format, 30f/s), these two memories require 16Mbit of DRAM [1, 2]. Since a single 16-bit access to a 4Mbit external memory takes three orders of magnitude more power than a 16-bit adder [3], design solutions capable of minimizing the size and capacitance of the these memories are extremely important.

In this paper we present a novel architectural technique,

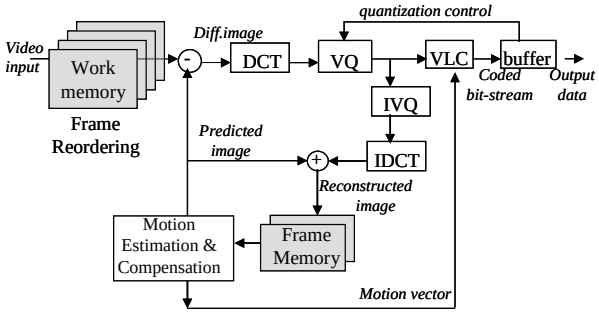

Figure 1: Block diagram of the MPEG video coding

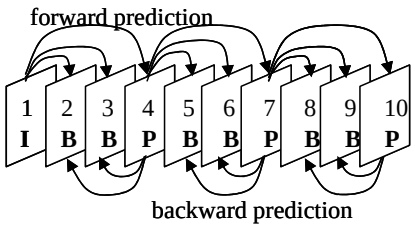

Figure 2: MPEG prediction mode

which significantly reduces MPEG memory requirements based on dynamic memory sharing between the reconstructed picture frames.

1.2. Related Research

Recently there have been reported a number of approaches to memory capacitance/size reduction. Many of them have concentrated on technology and circuit optimizations which can decrease bit-line energy (e.g. cell structure modification, charge recycling, limited bit-line swing, data retention circuitry, etc. [5],[6] At the algorithmic level, the focus has been put on code transformations, capable of reducing the number of memory accesses. Program transformations to eliminate redundant memory accesses have been proposed in [7], [8]. A systematic methodology to memory exploration in multimedia oriented systems and embedded cores has been presented in [9]-[11]. These works show that application-specific data-flow transformations, such as data width optimization, modification of computation order, loop manipulations,

shifting of delay lines through the algorithm, data reuse, etc. can shrink the memory size significantly. Work [12] studied the memory architecture design tradeoffs for video processors taking into account the bottom-up circuit parameters as well as top-down performance requirements. These approaches, however, have considered only a simplified relation between the picture frames and are not applicable to bidirectional prediction of advanced encoding schemes.

1.3. Contribution

In this paper we present a new approach to MPEG memory optimization based on incremental memory sharing between the reference and the reconstructed frames in bi-directional encoding. The method exploits the temporal locality of macroblock-based video picture motion estimation by dynamically allocating the picture elements or pixels of a currently reconstructed picture in place of corresponding pixels of the reference frame. It works equally well for unidirectional and bi-directional prediction schemes. Simulation results show that using this method we can reduce the total memory size by almost half for the unidirectional schemes and 25% for the bi-directional prediction schemes without any impact on performance and picture quality.

The paper is organized as follows. The next section presents the approach and outlines its implementation. Section 3 analyzes the performance. Conclusions are drawn in section 4.

2. THE PROPOSED APPROACH

2.1 Main idea

The key idea of our approach is to gradually renew the frame memory content by overwriting the previous reconstructed frame pixels with pixels of the newly reconstructed frame. The approach is based on observation that the frame memory data are utilized gradually block-by-block imposed by block-matching motion estimation, the only user of the memory. In the block-matching motion estimation, the current frame, F_c, is divided into non-overlapping rectangular macroblocks of $N*N$ pixels. The motion vector for each macroblock $X \in F_c$ is estimated by searching for such a macroblock Y within a search window surrounding by p pixels the position of X in the reference frame F_r, and minimizes the matching criterion (such as accumulated sum of absolute differences).

During motion estimation, the frame memory is accessed for reading $(N+2p)*(N+2p)$ pixels of the search window and writing N^2 pixels of the reconstructed macroblock. Due to sequential processing of macroblocks in the frame F_c, the reference frame data gradually become useless and can be replaced by data of a newly reconstructed frame under a restriction that a macroblock $X_{ij} \in F_c$ overwrites macroblock $Y_{ij} \in F_r$ with identical coordinates. Thus, while one region of the frame memory stores the previous frame data, another region of the memory is replaced with pixels of a newly reconstructed frame.

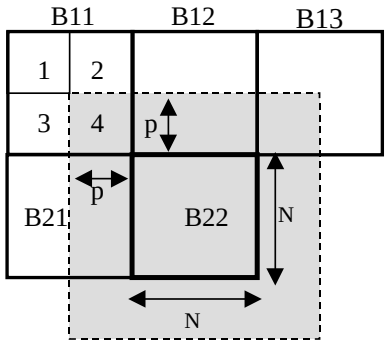

Figure 3: An illustration of data sharing approach

Figure 3 illustrates the concept on example of four $N*N$ macroblocks (depicted by thick solid lines), which belong to two adjacent macroblock slices of the frame. The shaded pattern in the figure exemplifies the search window of block B22. Due to displacement, p, of the search windows for these blocks, the region 1 of the previous frame becomes vacant after processing block B11 of the current frame, region 2 after processing block B12, region 3 after processing B12, and region 4 after processing B22. Consequently, after processing the block B22, the frame memory allocated for storing the block B11 of the previous frame can be efficiently used for storing the block B11 of the current frame. As result of such picture overlapping (or memory sharing), the reconstructed P-picture can replace dynamically the previously saved I-picture or P-picture within the same memory unit without affecting the coding algorithm. What we need is a FIFO-type buffer on the memory input to postpone writing of the current block pixels into the memory until all pixels of the previous block are processed. For p=N/2, the size of the buffer can be expressed as follows:

$$Buffer\ size = (2*W/N+2)*(N/2)^{2.} \quad (1)$$

where, W denotes the picture width.

2.2 Memory architecture

Figure 4 shows the proposed memory architecture. The shaded rectangles in the figure depict memory units, the curved blocks represent computation units and trapezoids define multiplexers. Two features distinguish our architecture from the conventional designs: namely, the

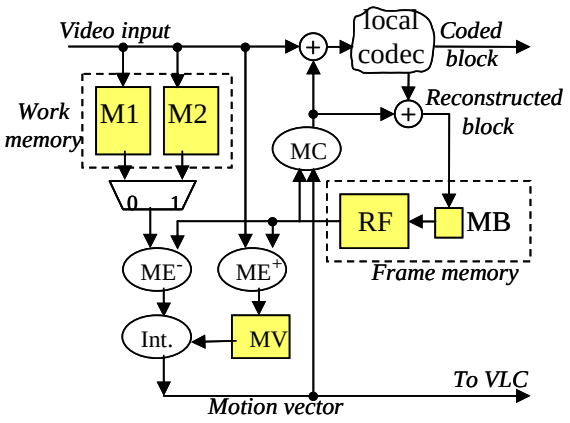

Figure 4: The proposed memory architecture

Cycle	Input frame	Memory read	Motion estimation		Memory write	Channel
			ME+	ME-		
1	I	-	-	-	MB, RF	I
2	B1	RF	(I,B1)	-	M1, MV	-
3	B2	RF	(I,B2)	-	M2, MV	-
4	P1	RF	(I,P1)	-	MB,RF	P1
5	B3	RF,M1,MV	(P1,B3)	(P,B1)	M1,MV	B1
6	B4	RF,M2,MV	(P1,B4)	(P,B2)	M2,MV	B2
7	P2	RF	(P1,P2)	-	MB,RF	P2
8	B5	RF,M1,MV	(P2,B5)	(P,B3)	M1,MV	B3
9	B6	RF,M2,MV	(P1,B6)	(P,B4)	M2,MV	B4
10	P3	RF	(I,P1)	-	MB,RF	P3
…	…	…	…	…	…	…

Figure 5: The basic data flow in the proposed architecture

frame memory configuration and the motion vector memory (MV). The frame memory is composed of reconstructed data storage (RF) and a memory buffer (MB), placed on its input. In contrast to related memory architectures [1, 2], the RF storage is reduced to only one frame, which can be dynamically shared by reconstructed pictures, as discussed above. The motion vector memory (MV) temporally keeps the results of forward motion estimation for B-frames till the bidirectional prediction becomes possible. The M1 and M2 are two partitions of the work memory which store two adjacent B-frames of the input stream (even and odd), respectively, ME^+ and ME^- are motion estimators of the forward- and the backward motion vectors, respectively; *Int*. is the interpolation unit, MC is the compensation unit, *local codec* implements the image coding/reconstruction sequence involving discrete cosine transform (DCT), quantization (Q), inverse quantization (IQ), and inverse DCT (IDCT), as shown in Fig.1.

The architecture works as follows. We start with encoding of the I-frame, while saving its reconstructed image in the MR memory. Then the next picture (which is the first B-frame of the input stream) is broadcasted to the M1 memory and the ME+ unit. The former stores it for the backward prediction, while the later computes the forward motion vectors using the I-frame, as a reference, and outputs the vectors to the MV memory. Similarly, we treat the next (even) B-picture of input sequence, detecting its forward motion vectors, and storing the picture with its motion vectors in ME and MV memories, respectively. When P-enters the system, we predict its motion in regards to the I-frame while writing the reconstructed macroblocks of the P-frame into the MB buffer and eventually into the RF storage, replacing the I-frame data dynamically blocks by blocks. The predicted error image and the motion vectors of the P-frame are sent to the channel. Thus, at the end of this iteration, we have P-picture written to the RF storage, two B-pictures in the work memory with their forward motion vectors computed and saved in the MV memory. When the next picture (the third in the sequence) arrives, we read RF and M1 in order to estimate its forward motion estimation at ME+ in parallel with the backward motion detection for the first frame B(1) at ME-. In both cases, the P-frame is used as a reference. The forward motion vectors are stored in MV memory simultaneously replacing the corresponding motion vectors of the frame B(1), which are read from MV. Based on these forward and backward motion vectors we interpolate the motion of the frame B(1) and send the encoded error picture to the channel. Notice that current picture B(3) replaces the previous (odd) picture B(1) in the M1 memory on the block-basis. Similarly we proceed with the next B picture of the stream, with only difference that now the picture replacement occurs in M2, not M1.

Figure 5 outlines the basic data flow in the system for the input sequence shown in Fig.2. Here, *Cycle* denotes a time interval of a single frame encoding; *Input frame* shows the type of the current source frame and its number in the stream, *memory read* and *memory write* define memories which are read (write) in the corresponding cycle, respectively, *Channel* represents the type of the frame transmitted to the channel. As we see the proposed architecture maintains the actual transmission order of the MPEG.

3. EXPERIMENTAL RESULTS

To evaluate the proposed approach we modified the MPEG-4 simulation program correspondingly and tested it

on unidirectional and bidirectional coding patterns, using the SIF color picture format (288x352 pixels, 30frames per second), assuming that N=16, p=8. Experiments with various picture benchmarks confirmed an absolute compliance of the proposed coding approach with the conventional MPEG coding on performance as well as on PSNR. At the same time, an experimental evaluation of the overall size of the work memory and frame memory proved that our approach is very economic. Figure 6 depicts the results obtained for five picture formats: Sub-QCIF (96x128 pixels), QCIF (176x144 pixels), SIF (288x352 pixels), 4CIF (704x576pixels), 16CIF (1152x1408 pixels) and CCIR601 (720x486 pixels) on different encoding patterns in comparison to the conventional architecture [1]. We observe that the memory saving depends on the encoding pattern significantly, and is almost flatten to the picture format. Incorporation of all three picture types with five B-pictures in between the I- and P-frames reduces the overall saving to a moderate (13%) level. The less B frames in the sequence the better savings. For the most practical case of two B-pictures, we can save almost 25% of the frame memory, while for the regular I-picture update and no B-pictures (i.e. I P P P P I P P P..), our approach achieves the best results reducing the overall size of the work memory and the frame memory almost by half.

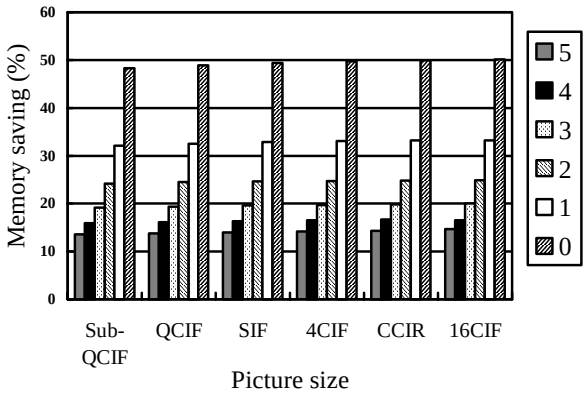

Figure 6: Memory savings achieved by the proposed approach as a function of picture format and the number of B-pictures in the coding sequence

4. SUMMARY

A new algorithmic and architectural enhancement has been presented for reducing memory requirements of the MPEG video coding. In particular, we targeted the minimization of frame memory based on dynamic macroblock replacement. Simulations have shown that the efficacy of our approach does not vary with the picture format, but depends strongly on the coding pattern. For the most frequently used encoding sequence of 9 still frames with two B frames in between the I(P) frames it can save as much as a quarter of total memory required for the work memory and frame memory in comparison to the existing design without affecting the picture quality and performance. Currently we are evaluating the impact of the proposed memory reduction on energy saving.

References

[1] T. Hashimoto et al, "A 90mW MPEG4 video codec LSI with capability for core profile", *ISSCC Digest of technical Papers*, pp.140-141, Feb. 2001.

[2] Nishikawa, et al., "A 60MHz 240mW MPEG-4 video phone LSI with 16Mbit embedded DRAM," *ISSCC Digest of technical Papers*, pp.230-231, Feb. 2000.

[3] ISO/IEC JTC 1/SC 29/WG11, *Generic coding of moving pictures and associate audio information, Part 1: Systems, Part2: Video, Part 3: Audio*, CD 13818, May 1995.

[4] B.Gordon, E.Tsern, T.Meng, ``Design of low-power video decompression chip set for portable applications", *J. VLSI Signal Processing Systems*, Vol.13, pp.125-142, 1996.

[5] B.Amrutur, "Techniques to reduce power in fast wide memories", *Proc. 1994 Int. Symp. on Low Power Electronic*s, pp.92-93, 1994.

[6] K.Itoh, "Reviews and prospects of low power memory circuits", in *Low Power CMOS Design*, A. Chandrakasan, R. Brodersen, (eds), pp.313-317, 1997.

[7] K. Itoh, "Low power memory design", in *Low Power Design Methodologies*, J.Rabaey, M.Pedram (eds), pp.201-251, 1996.

[8] J. Davidson and S. Jinturkar, "Memory access coalescing: A technique for eliminating redundant memory accesses", *PLDI*, Vol.29, No.6, June 1994.

[9] D. Kolson, A. Nicolau, N. Dutt, "Integrating program transformations in the memory based synthesis of image and video applications", *Proc. IEEE ICCAD*, pp.27-30, 1994.

[10] F. Cathoor, S. Wuytack, E. De Greef, F. Balasa, L. Nachtergaele, and A. Vandecapelle "*Custom memory management methodology, exploration of memory organization for embedded multimedia system design*", Kluwer Academic Publishers., 1998.

[11] F. Catthoor, "Power efficient data storage and transfer methodologies: current solutions and remaining problems," *Proc. IEEE Int.Workshop on VLSI*, pp.29-34, 1999.

[12] T. Van Achteren, R. Lauwereins, F. Catthoor, "Systematic data reuse exploration methodology for irregular access patterns", *Proc. 13th ACM/IEEE Int. Symp. on System-Level Synthesis*, pp.115-121, Sep. 2000.

[13] S. Dutta, W. Wolf, and A.Wolfe, "A Methodology to evaluate memory architecture tradeoffs for video signal processors", *IEEE Trans.Circuits and Systems for Video Technology*, Vol.8, No.1, pp.36-53, Feb. 1998.