Exploring Reusable Frame Buffer Data for MPEG-4 Video Decoding

Wei-Cheng Lin and Chung-Ho Chen
Department of Electrical Engineering and
Institute of Computer and Communication Engineering, National Cheng-Kung University
No.1 University Road, Tainan, Taiwan 70101, R.O.C.
kevin@casmail.ee.ncku.edu.tw chchen@mail.ncku.edu.tw

***Abstract*—This paper presents a novel approach that avoids unnecessary frame buffer accesses by exploring reusable frame buffer data for MPEG-4 simple profile video decoders. In an MPEG decoder, each decoded macroblock is stored in the frame buffer for future reference and video display. If a newly decoded macroblock overwrites the predecessor that is exactly the same, then the old data stored in the frame buffer are reusable. Frame buffer accesses of this kind become unnecessary. We propose a design that greatly reduces the number of such unnecessary memory accesses through the detection and management of the reusable data in the frame buffers. The experimental results show that the proposed scheme achieves as much as 53% (28% on the average) saving in frame buffer accesses, without any sacrifice of the image quality and with only a negligible hardware overhead.**

I. INTRODUCTION

In an MPEG decoder, frame buffers are used to store the reference frames for the prediction of consecutive pictures. The frame buffers are often too large to fit into the on-chip memory and are typically stored in the external memory. It is reported that the frame buffer accesses may occupy over 90% of the off-chip memory accesses [1]. Consequently, eliminating unnecessary frame buffer accesses is an effective way to save memory energy usage.

In general, a frame buffer is usually re-written by a newly decoded frame since the number of frame buffer, NFB for short, is limited. A decoded macroblock always overwrites the one belonged to a previous frame. When these two macroblocks are exactly the same, the one stored in the frame buffer is reusable. This means that reading of the referenced data from the frame buffer and writing of the decoded data to the frame buffer are both redundant.

In this work, we propose a mechanism, which can detect the reusable macroblocks in the frame buffers, to avoid unnecessary frame buffer accesses. The experimental results show that this mechanism reduces the number of frame buffer accesses as much as 53% (28% on the average) without affecting the video quality and the hardware overhead is minimal.

The rest of this paper is organized as follows. Section II discusses the related work. Section III explains the concept of stationary and reusable macroblocks. Section IV presents the reusable data block detector and the implementation. The simulation system and results are discussed in Section V and Section VI, respectively. Finally, Section VII gives the conclusion of this paper.

II. RELATED WORK

For data-dominated video applications, a number of recent researches have targeted at reducing the external memory accesses by exploiting hierarchical memory organization which can efficiently utilize the available temporal locality in the data accesses [2]. Several proposals attempt to explore reusable data at the architecture level, including data flow optimizations [3]. Compiler-based optimization techniques for reducing external memory accesses through special scratch-pad memory have been reported in [4, 5]. The research in [4] focuses on increasing the reusable data in a multiprocessor based embedded system. In [5], it employs a software-managed scratch-pad memory to minimize the number of data transfers for array-based applications.

Shih and et. al. have proposed a mechanism to reduce memory accesses and the size of memory requirements for MPEG-2 video decoder [6]. This is accomplished by using a variable-length coder to compress reconstructed frame data before storing in the memory, and a variable-length decoder to de-compress the frame data that are read from the memory for motion compensation or display. In [7], a high-level memory management scheme has been presented to decrease the number of external memory accesses and reduce the on-chip memory size for H.263 video decoder. This scheme does not consider the display unit that affects the number of frame buffers used. In [8], Moshnyaga and et al. have proposed an architecture that reduces computations of motion estimation and frame buffer writes for MPEG video encoding by exploiting the stationary macroblocks which are determined by a given threshold.

III. STATIONARY AND REUSABLE MACROBLOCK

A. *Stationary Macroblock*

The MPEG-4 simple profile video streams are comprised of two types of frames: intra (I) frames and predictive (P) frames. An I-frame only contains the spatial information of a video frame and can be decoded independently without referencing to any other frame. A P-frame is decoded from the previously decoded P- or I-frame, which is called a reference frame. Each coded macroblock of the P-frame is either the Inter or Intra

mode. Decoding an Inter macroblock implies adding a reference macroblock that is obtained from the reference frame according to the motion vector(s) with a texture decoded residual macroblock.

More specifically, a macroblock is organized as four Y data blocks (8 × 8 pixels), one U data block (8 × 8 pixels), and one V data block (8 × 8 pixels). The four Y data blocks are combined to form a luminance data block (16 × 16 pixels). U and V data blocks are also combined to form a chrominance data block (16 × 8 pixels). We call the luminance, chrominance, U and V data block in the residual macroblock as L-, C-, U- and V- residual respectively. Furthermore, when an Inter macroblock is decoded, the bitstream gives either one motion vector (for luminance data block) or four motion vectors (for four Y data blocks). We refer to such motion vectors as either 16 × 16 or 8 × 8 motion vector. The chrominance motion vector is not coded but originated from the luminance motion vectors. If the L-residual, C-residual, and the 16 × 16 motion vector are all zero, the decoded macroblock is exactly the same as its reference macroblock. In this case, the decoded macroblock is referred to as stationary.

The occurrence of stationary macroblocks strongly depends on not only the inherent feature of the video sequence, but also the encoder-defined quantization parameter (QP) that ranges from 1 to 31. Typically, a video sequence that is encoded by using a larger QP results in lower bit-rate, lower image quality, and more zero residuals.

B. Reusable Macroblock

To see the advantage of reusing macroblocks in the frame buffers, consider the sequence of three frames (I_0, P_1 and P_2) that only comprise one moving object (a bird) as shown in Fig. 1. We assume that each frame contains only 12 macroblocks and the number of frame buffers used is two (NFB =2). This implies that the decoding rate at least equals to the display rate [9]. Suppose that two frame buffers are empty at the beginning of decoding the I-frame (I_0). In the time period T_0, the I-frame (I_0) is decoded and stored in frame buffer FB_0. During the next period T_1, frame buffer FB_0 is read twice for serving two purposes, one for video display and the other for decoding the P-frame (P_1) which is to be stored in frame buffer FB_1.

Also, in the time period T_2, the decoded P-frame (P_2) is scheduled to be written into frame buffer FB_0 that has stored the previous decoded frame I_0. However, the decoded P-frames (P_1 and P_2) have the ten and nine stationary (shaded) macroblocks respectively and there are nine stationary macroblocks overlapped. This means that the frame buffer FB_0 has nine reusable (thick line) macroblocks which are entirely equal to those in the decoded frame P_2. If these reusable macroblocks can be identified, then we can skip the reading of the nine reference macroblocks from FB_1 and writing of the nine reconstructed macroblocks to FB_0. Thus, the total frame buffer accesses can be reduced by 50% ((3+3+12)/(12+12+12)×100%) during the time period of T_2.

Typically, the display rate is constant and the decoding rate varies since the bitstream length varies with the content of the video stream. Therefore, if the decoding rate can not always be greater than the display rate, it is necessary to increase the number of frame buffer (NFB) to avoid buffer underflow.

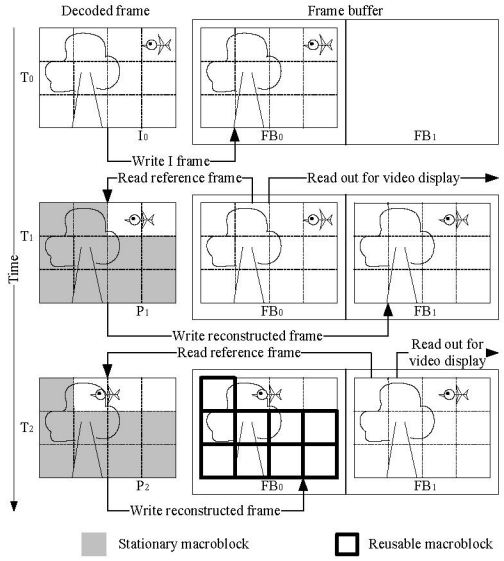

Fig. 1. Reusable macroblocks in frame buffers (NFB = 2).

IV. Reusable Macroblock (Data Block) Detector

A. Methodology

To identify the reusable macroblocks, we explore the correlation between two macroblocks of which one is being decoded and the other is to be overwritten in the frame buffer. For example, as shown in Fig. 2(a), suppose that NFB is equal to n. We assume that n P-frames (P_0, P_1, …, P_{n-1}) have been decoded and stored in the frame buffers, and frame buffer FB_0 is recycled (writable) since the frame P_0 has been displayed. In this case, macroblock M_n in frame P_n is being decoded and the decoded data are going to overwrite macroblock M_0 stored in frame buffer FB_0.

The correlation between macroblock M_0 and M_n can be generated through macroblock M_1, M_2, … and M_{n-1}. We refer to these macroblocks (M_1, M_2, …, M_{n-1}) and frames (P_1, P_2, …, P_{n-1}) as intermediate macroblocks and frames. If macroblock M_n and each corresponding intermediate macroblock are all stationary, then macroblock M_0 in frame buffer FB_0 is reusable because it is completely equal to macroblock M_n.

To detect the reusable macroblocks, we incorporate a macroblock bitmap (MBB) for each frame buffer. The MBB has as many cells as the number of macroblocks in a frame. Each cell records whether the motion vector and residual of a corresponding macroblock are equal to zero or not. In our experiment, we found that the probability of that an 8 × 8 motion vector is equal to zero is very low. Thus, we do not

actually record four 8 × 8 motion vectors, but just focus on the 16 × 16 motion vector. We use one bit, named luminance (L) bit, to indicate whether a luminance data block is stationary. If the 16 × 16 motion vector and the L-residual of a macroblock are both zero, the corresponding L bit is set to 1. On the other hand, the U- and V- residual of both being zero is highly correlated. Thus, we use one bit, named chrominance (C) bit, to record chrominance data block. Similarly, the C bit is set to 1 if a 16 × 16 motion vector and the C-residual are both zero. The information for zero motion vectors and zero residuals can be easily obtained by looking up the macroblock header [10].

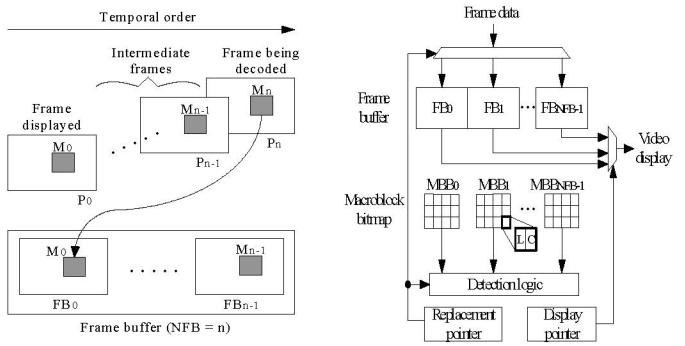

Fig. 2. (a) Intermediate macroblocks and frames, (b) Logical block diagram of the reusable data block detector.

Fig. 2(b) outlines the logical block diagram of the reusable data block detector with frame buffers. Each MBB's cell contains the L bit and C bit. To decide which frame buffer should be displayed or replaced, we employ two sequential counters, a replacement pointer and a display pointer that count from 0 to NFB–1 and repeat since the bitstream decoding and video display are all in order. When a new frame is decoded, the replacement pointer indicates a frame buffer (called victim frame buffer) and its corresponding MBB (called victim MBB) to store the incoming frame data and record the L and C status. When a new macroblock is decoded, we examine its luminance (chrominance) data block. If it is stationary, then the corresponding L (C) bit in the victim MBB is set to 1, otherwise it is set to 0. At the same time, the detection logic determines if the L (C) bits corresponding to the intermediate macroblocks and the newly decoded macroblock are all equal to 1. If this is true, then the luminance (chrominance) data block to be overwritten in the victim frame buffer is reusable.

B. Hardware Implementation

To minimize the hardware overhead, the entire MBBs can be stored in the external memory. For the reusable data block detector, we use a small MBB buffer, for each frame buffer, to cache portion of the MBBs data as depicted in Fig. 3. The size of an MBB buffer shown here has a total of 16 entries (32 bits). The MBB buffer can be implemented as a shift register that simply shifts as a macroblock is decoded. A 2-bit register (L and C) is used to record whether the newly decoded macroblock is stationary. Before this register is updated, its data are stored to the MBB buffer corresponding to the victim MBB through the de-multiplexer. In the detection logic, the comparator and OR-gates are used to don't care the L and C values which come from the MBB buffer corresponding to the victim MBB. The AND-gates are used to decide whether the data blocks in the victim frame buffer are reusable.

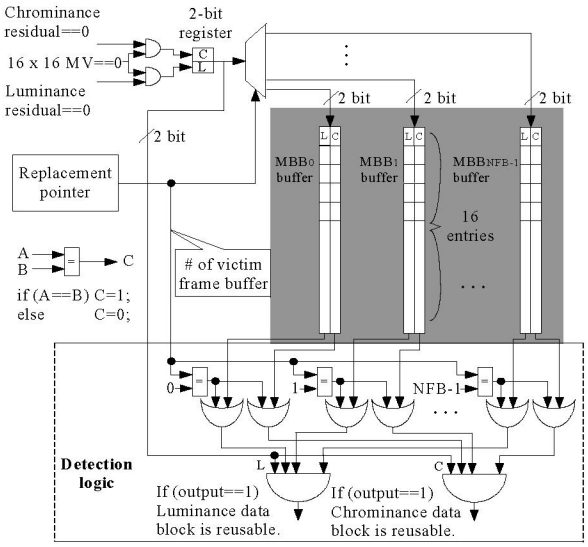

Fig. 3. The circuit of reusable data block detector.

We implemented the reusable data block detector for the MPEG-4 simple profile decoder with 0.25μm CMOS standard cell technology. The cost of the design is 754 (1068) equivalent gates when NFB= 2 (4). This hardware cost is negligible compared to the cost of MPEG-4 simple profile decoder.

V. Simulation System

To evaluate the proposed scheme, we implement a video codec for MPEG-4 Simple Profile at Level 3 (SP @ L3) which supports CIF (352 × 288) resolution up to 30 frames per second. In the encoder, the ME search range is ±13 and the Intra-period (IP) is 30 frames. The encoder generates the bitstreams running on twelve video sequences from QP = 1 to QP = 31. The decoder decodes each bitstream with different number of frame buffers used (NFB = 2 to 4) and counts the number of reusable data blocks.

The experiments were run on twelve video sequences which were classified into three classes [10] according to spatial frequency and the amount of movement as shown in Table I. The frame size of *Football* is 352 × 240 pixels while the others are 352 × 288 pixels. Performance comparisons are done on the first 150 frames of *Bus*, 120 frames of *Football* and 300 frames of the others.

Table I Video benchmarks.

Class	Video sequence	Spatial frequency	Amount of movement
I	Mother & daughter, Akiyo, Hall monitor, Container ship	Low	Low
II	Foreman, News, Silent voice, Coastguard	Medium	Low
III	Bus, Stefan, Mobil & calendar, Football	High	Medium

VI. Simulation Results

Fig. 4 shows the percentage of reduced frame buffer access obtained and the effect of QP value on it when NFB = 2. The reduction ranges from around 2% for video sequence *Bus* to as much as 53% for video sequence *Akiyo* as QP = 16. Up to 35%, 23% and 3% of access reduction can be achieved for video sequences of class I, II and III when QP = 8. The reason for class III video sequences that has the least amount of reduction in memory access is that these video sequences have much higher spatial frequency and movement which lead to very small amount of zero residuals and zero motion vectors. In addition, the experiment also shows that the percentage of the reduced memory access is increased as the QP value is raised. This increment occurs because a higher QP value results in more zero residuals.

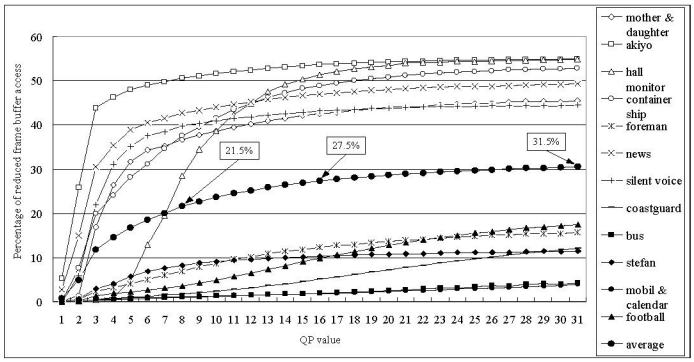

Fig. 4. The QP value versus the percentage of reduced frame buffer access when NFB = 2.

Fig. 5 presents the percentage of the reduced memory access for QP = 16 as we vary the number of frame buffer used. The experiments show that the percentage of the reduced memory access is decreased when the number of frame buffer is increased. This can be explained as follows:

1. When the decoder decodes an I-frame, each cell of the MBB related to this I-frame is set to zero. After the NFB–1 P-frames following the I-frame are decoded, there are no reusable data blocks in the frame buffers because this zero-MBB of the I-frame is referred.
2. When a macroblock is decoded, NFB–1 cells are checked to see if they are all equal to 1. Consequently, for a larger NFB, the probability of that the macroblock in the frame buffer is reusable is decreased.

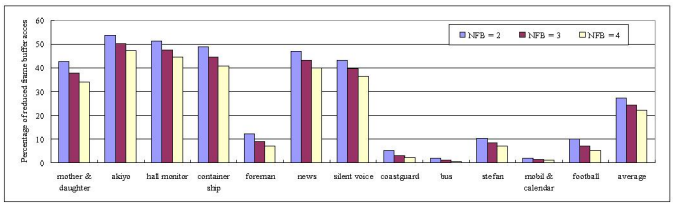

Fig. 5. The number of frame buffer versus the percentage of frame buffer access reduction.

VII. Conclusion

In this paper, we have proposed a novel scheme for reducing frame buffer accesses in an MPEG-4 simple profiler decoder. In particular, the scheme finds a reusable data block which takes place when a data block overwrites an exactly the same copy in the frame buffer. Simulations of the proposed method have shown that the approach can eliminate as much as 53% (28% on the average) of the frame buffer accesses, without affecting the video quality. In addition, the hardware overhead incurred is minimal. Our current work focuses on extending the approach to MPEG-4 advance simple profile decoder that includes bidirectional prediction frames (B frame).

Acknowledgement

The work in this paper is in part supported by the National Science Council, Taiwan ROC, under NSC 94-2220-E-006-004.

References

[1] J.-H. Li and Nam Ling, "Architecture and Bus-Arbitration Schemes for. MPEG-2 Video Decoder," *IEEE Transactions on Circuits and Systems for VideoTtechnology*, vol. 9, no. 5, pp.727-736, August 1999.

[2] Z. Xu , S. Sohoni , R. Min, and Y. Hu, "An Analysis of Cache Performance of Multimedia Applications," *IEEE Transactions on Computers,* vol. 53, no.1, pp.20-38, January 2004.

[3] S. Goel, M. Shaaban, T. Darwish, H. Mahmoud, and M. Bayoumi, "Memory Accesses Reduction for MIME Algorithm," *International Conference on Multimedia and Expo ICME,* pp. II-805-II-808, July 2003.

[4] M. Kandemir, J. Ramanujam, and A. Choudhary, "Exploiting shared scratch pad memory space in embedded multiprocessor systems," *Proc. 39th Design Automation Conf.,* pp. 219-224, New Orleans, U.S.A., June 2002.

[5] M. Kandemir, J. Ramanujam, M. Irwin, V. Narayanan, I. Kadayif, and A. Parikh, "A Compiler Based Approach for Dynamically Managing Scratch-Pad Memories in Embedded Systems," *IEEE Transactions on Computer-Aided Design,* vol. 23, no. 2, pp. 243-260, February 2004.

[6] C.-W. J. Shih, N. Ling, and T. Ogunfunmi, "Memory Reduction by Haar Wavelet Transform for MPEG Decoder," *IEEE Transactions on Consumer Electronics*, vol. 45, no. 3, pp. 867-873, August 1999.

[7] L. Nachtergaele, F. Catthoor, B. Kapoor, S. Janssens, and D. Moolenaar, "Low power data transfer and storage exploration for H.263 video decoder system," *IEEE Journal on Selected Areas in Communications,* vol. 16, no. 1, pp. 120-129, January 1998.

[8] V. G. Moshnyaga, K. Masunaga, and N. Kajiwara "A Data Reusing Architecture for MPEG Video Coding," *Proceedings of the 2004 IEEE International Symposium on Circuits and Systems (ISCAS 2004)*, Vancouver, CANADA, pp. III-797-III-800, May 23-26, 2004.

[9] D. Isovic, G. Fohler, and L. Steffens, "Real-time issues of MPEG-2 playout in resource constrained systems," *Journal of Embedded Computing*, issue 3, pp.1-13, June 2004.

[10] ISO/IEC JTC1/SC29/WG11, "MPEG4 Video Verification Model Version 18.0," January 2001.