FAST 2D IDCT IMPLEMENTATION WITH MULTIMEDIA INSTRUCTIONS FOR A SOFTWARE MPEG2 DECODER

Eri Murata, Masao Ikekawa and Ichiro Kuroda

C&C Media Research Laboratories, NEC Corporation
4-1-1, Miyazaki, Miyamae-ku, Kawasaki 216, Japan
e-mail: murata@ccm.cl.nec.co.jp

ABSTRACT

This paper presents an implementation of a fast two-dimensional inverse Discrete Cosine Transform (IDCT) with multimedia instructions for a software MPEG2 decoder. IDCT algorithms for sparse blocks which eliminate the calculation for zero coefficients are realized by using multimedia instructions. To reduce the cycle count for IDCT, an adaptive control method for these IDCT algorithms, based on the bit rate and picture type, is proposed and its performance is described. In the implementation of a software MPEG2 decoder, the execution time for IDCT is reduced to 10% by using MMX instructions from original C program. Moreover, using proposed adaptive control, it can further be reduced to 76%.

1. INTRODUCTION

The compression of digital data, including video, audio, and other forms of information, is going to be important for the realization of a variety of multimedia applications on desktop computers. Several international standards which specify the syntax of the compressed bitstream and the method of decoding have been developed.

The software MPEG2 decoder is coming to be a common function for multimedia PCs. However, it requires a lot of computation power, and has usually been implemented with special boards or chips. Recently software MPEG2 decoder is realized which gives us a significant cost effective solution[1][2]. This is made possible by multimedia instructions which calculate multiple pixels in one instruction[3][4].

The MPEG and several other compression standards adopt the DCT/IDCT to encode and decode video data. For the decoder, IDCT is a prime candidate for increasing decoding speed with multimedia instructions, because of parallelism in the computation. An 8×8 2D IDCT designed by row-column decomposition can be accelerated by operating four 1D IDCTs in parallel[5].

As many of the input coefficients of IDCT are zero due to quantization, fast IDCT algorithms for sparse blocks which eliminate the calculation for zero coefficients have been proposed[6][7][8]. These IDCTs are effective in software decoders, because they are designed based on an average case of performance, though hardware decoders must be designed based on the worst case.

This paper describes techniques to reduce the execution time for IDCT using MMX instructions on Pentium processor family. First, fast IDCT algorithms for sparse blocks which eliminate the calculation for zero coefficients realized by using MMX instructions are described. Second, the distribution of zero coefficients for each bit rate and picture type is analyzed. Third, an adaptive control method to select the IDCT algorithms for sparse blocks based on the distribution of zero coefficients is proposed. Finally, an implementation of this technique for a software MPEG2 decoder and its performance are described.

2. IDCT WITH MULTIMEDIA INSTRUCTIONS

2.1. Normal IDCT

An 8×8 point two-dimensional IDCT can be realized by the 8 point one-dimensional IDCTs for eight rows and eight columns. By using MMX instructions, four 1D IDCTs can be performed in parallel as shown in Figure 1. Many algorithms have been proposed for the efficient calculation for the 1D IDCT[9][10]. We adopt the LLM algorithm[11] which requires 11 multiplications and 29 additions, considering speed and accuracy. By using MMX instructions, the cycle count for IDCT is reduced to 10% of the C program implementation of the same algorithm.

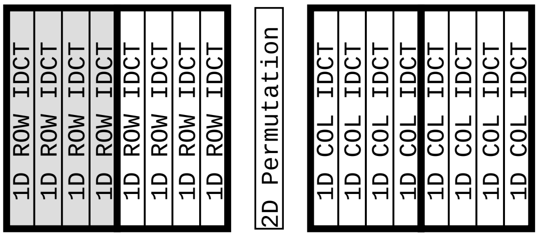


Figure 1: 8×8 2D IDCT with MMX instructions

2.2. IDCT for sparse blocks

Many of the input coefficients of IDCT in MPEG decoder are zero due to quantization. This can be utilized to reduce cycle counts in the IDCT by eliminating the calculation for zero coefficients. For example, a forward mapped IDCT(FMIDCT) which is efficient for IDCT calculation when the input nonzero coefficient is very sparse has been proposed[8]. On the other hand, we can eliminate the calculation by selecting an IDCT among IDCTs for sparse blocks for specific distributions of nonzero coefficients. Here, we realize three types of IDCT by using MMX instruction. Table 1 shows the relative speed of these IDCTs implemented on a Pentium processor with MMX instructions.

(1)IDCT_DC
This is an algorithm for a block which has only one nonzero coefficient in the DC term. The IDCT of that block can be calculated by simply scaling and replacing this DC component over the entire 8×8 block.
(2)IDCT_AC
This is an algorithm for a block which has only one nonzero coefficient in AC terms. We make a look-up table which has the results of one nonzero coefficient IDCT for each position in 8×8 block in advance. IDCT_AC can be realized by accessing the look-up table according to the address of the nonzero component, and multiplying these results by the nonzero component.
(3)IDCT_4×4
This is an algorithm for a block which has only 4×4 low-frequency components. By using MMX instructions which operate four pixels in parallel, an 8×8 2D IDCT can be realized by four 1D IDCTs. The IDCT_4×4 eliminates the calculation for one IDCT which has no nonzero coefficient. Furthermore, the remaining three IDCTs eliminate some operations because half of the input coefficients are zero.

Table 1: Relative speed of IDCTs.
(without penalties for cache misses)

IDCT type	normal	DC	AC	4×4
ratio of execution time	100	11	28	58

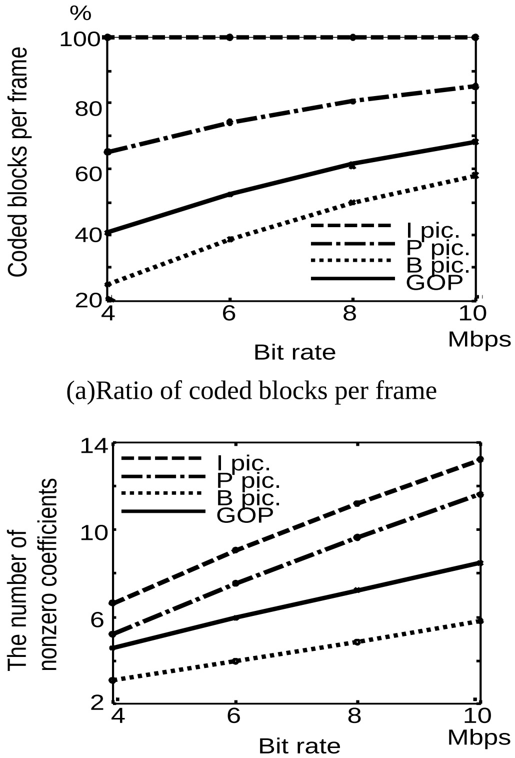
2.3. Distribution of nonzero coefficients

The distribution of nonzero coefficients has been analyzed for the MPEG2 bitstreams at 4, 6, 8, and 10 Mbps. Each frame has 8,100 8×8 blocks and each group of pictures (GOP) includes 1 I-, 4 P-, 10 B-frames. The distribution varies for different video streams, however, in most of the bitstreams analyzed, there are common characteristics for any bit rates and picture types.

Figure 2(a) shows the ratio of coded blocks per frame and Figure 2(b) shows the number of nonzero coefficients per coded block. It can be seen that the percentage of coded blocks is 20 to 50% in B-frame, 65 to 80% in P-frame, 100% in I-frame and many of the input coefficients are zero.

Figure 3 shows the probability of using each IDCT algorithm in overall coded blocks per frame. We select an IDCT algorithm according to the address of the end of block (EOB) code and the number of nonzero coefficients. The address of EOB code represents the position in a block where the remainder of coefficient are zero. We select an IDCT among four algorithms: (1) IDCT_DC when the EOB address is 0, (2) IDCT_AC when the number of nonzero coefficients is 1 and the EOB address is not 0, (3) IDCT_4×4 when the EOB address is less than 10, (4) and normal IDCT otherwise. In I-frame, many blocks use IDCT_DC and IDCT_4×4, while there are few blocks which use IDCT_AC, since the nonzero coefficients are distributed toward the low-frequency area. In B-frame, many blocks use the IDCTs for sparse blocks especially IDCT_AC, since most of the input coefficients are zero. P-picture shows the average characteristics between I- and B-picture. The ratio of the blocks using IDCTs for sparse blocks becomes lower for any picture type when the bit rate is higher.

The execution time for the IDCT per frame is estimated by using the distributions and execution times of each IDCT algorithm. It is reduced as shown in Figure 4. The execution time is reduced to 67 to 85% in I-frame, 73 to 88% in P-frame, 60 to 80% in B-frame when we use the IDCTs for sparse blocks. However, these results don't include the overheads for selecting the IDCT among four IDCTs for sparse blocks nor the penalties for cache misses.



(a)Ratio of coded blocks per frame

(b)The number of nonzero coefficients
(per coded block)

Figure 2: Distribution of nonzero coefficients.

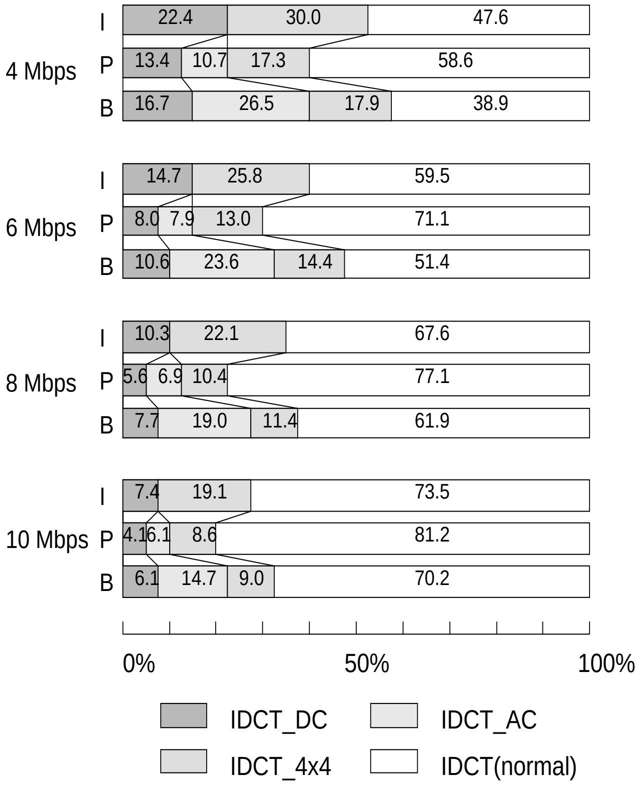


Figure 3: Probability of each IDCT algorithm.

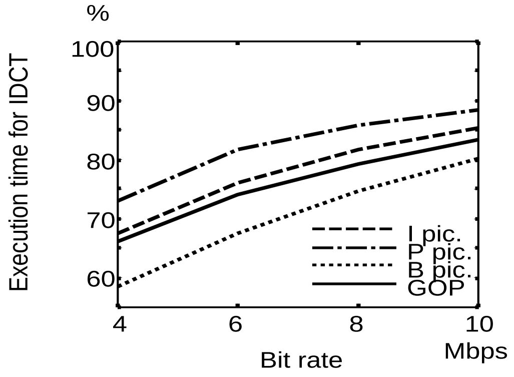


Figure 4: Performance improvement
of execution time for IDCT per frame.
(without overheads and penalties)

3. IMPLEMENTATION IN A SOFTWARE MPEG2 DECODER

3.1. Adaptive control

In this section a fast implementation technique for IDCT in a software MPEG2 decoder is presented. To reduce the execution time for any bit rate and picture type, it is necessary to consider the overhead for selecting IDCT and penalties for cache misses. The overhead increases as the number of IDCT algorithms to prepare is increased. When we use IDCT algorithms which is selected rarely, the overhead of instruction cache miss increases.

Figure 5 shows the block diagram of the adaptive control technique. First, candidate IDCT algorithms are selected in advance based on the bit rate and the picture type. Next, the most efficient IDCT algorithm is selected from the preselected IDCTs according to the EOB address for each block. By using this techniques, the overheads of selecting IDCTs and penalties for cache misses can be reduced. For example, in I- and P-frame, we select an algorithm among the IDCT_DC, the IDCT_4×4, and the normal IDCT when the bit rate is less than 6 Mbps (A), otherwise we choose between the IDCT_4×4 and the normal IDCT (B). IDCT_AC is not used for above cases because it requires a lot of memory accesses, which results in the many penalties for data cache misses. Also, it requires the overhead for counting the number of nonzero coefficients. In B-frame, we select from all of the algorithms for any bit rate (C).

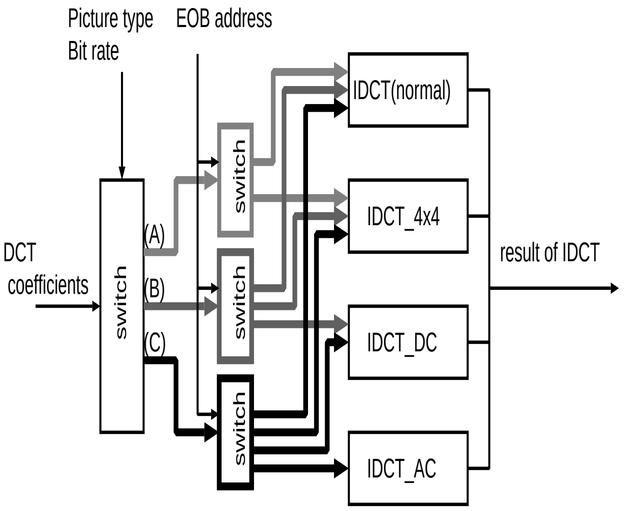


Figure 5: Adaptive control technique.

3.2. Performance Evaluation

The performance improvement of IDCT implemented in a software MPEG2 decoder with MMX instructions is shown in Figure 6. By using the IDCTs for sparse blocks without adaptive control using bit rate and picture type, the execution time including overheads or penalties is reduced to 79 to 96% (Method 1). Furthermore, when we adopt the adaptive control for IDCTs for sparse blocks algorithms, it can be reduced to 76 to 90% (Method 2).

Software decoders skip frames when the CPU power isn't sufficient to achieve real-time decoding. Using the IDCT_4×4 instead of the normal IDCT in B-frame is sometimes effective. In this case, the execution time for IDCT in B-frame is reduced to 60% in B-frame, and 70 to 75% in total, although the image quality is decreased by eliminating the calculation for high-frequency data. However, if we display the decoding image using a low resolution monitor, motion smoothness is more important than the image quality in B-frame (Method 3).

Besides, we have realized the FMIDCT by MMX instructions, which is efficient for IDCT calculation when the input nonzero coefficient is very sparse. The execution time for a block increases with the number of nonzero coefficients. By our experiment, the FMIDCT is effective only when the block has two nonzero coefficients and can't use the IDCT_4×4. This case is only 5% of coded blocks in I- and P-frame, 10% in B-frame. Considering overheads and penalties for the implementation, there is no advantage on the software decoder by using MMX instructions.

By implementing the Method2 for IDCT, and optimizing all over the MPEG2 decoding steps including variable length decoding (VLD)[12], inverse quantization (IQ), and motion compensation (MC), we can realize real-time software decoder of MPEG2-video and Dolby AC-3 audio at 4 Mbps on a 266MHz Pentium II (Pentium Pro processor with MMX technology). The percentage of total time spent in decoding for MPEG2-video is shown in Figure 7.

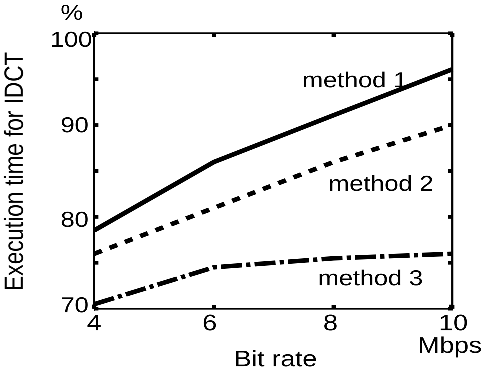


Figure 6: Performance improvement of execution time for IDCT.

Method 1: without adaptive control
Method 2: adaptive control of IDCTs for sparse blocks
Method 3: adaptive control of IDCTs for sparse blocks without normal IDCT in B-frame

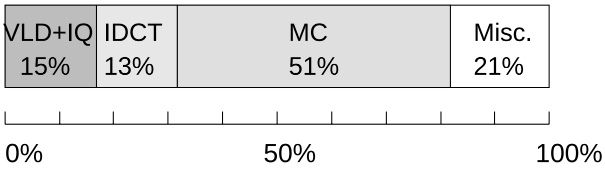


Figure 7: Percentage of time spent. (MPEG2 Video)

4. CONCLUSION

A fast software algorithms for IDCT implementation using adaptive technique as well as MMX instructions is proposed. The execution time for IDCT is reduced to 10% by using MMX instructions from original C program. When an adaptive control of these IDCTs considering the distribution of nonzero coefficients for each bit rate and picture type is used, it can further be reduced to 76%. By implementing the proposed method for IDCT, and optimizing some other steps in a software MPEG2 decoder, the bitstream of MPEG2 video and Dolby AC-3 audio at 4 Mbps can be decoded in real-time on a 266MHz Pentium II.

5. REFERENCES

[1] C. Zhou, et al., "MPEG Video Decoding with the UltraSPARC Visual Instruction Set," COMPCON'95 Digest of Papers, IEEE, PP. 470-475, Mar. 1995.

[2] M. Ikekawa, et al.,"A Real-time Software MPEG-2 Decoder for Multimedia PCs," ICCE 97, WAM1.1, Jun 1997.

[3] B. Lee, "Accelerating Multimedia with Enhanced Microprocessor," IEEE Micro, pp. 22-32, April 1995.

[4] O. Lempel, et al.,"Intel's MMX Technology - A New Instruction Set Extension," COMPCON'97, pp. 255-259, 1997.

[5] Intel, "Using MMX Instructions in a Fast iDCT Algorithm for MPEG Decoding," AP-528, March 1996.

[6] C. Hung, et al., "A Fast Statistical Inverse Discrete cosine Transform for Image Compression," SPIE/IS & Teletronic Imaging, 2187, pp. 196-205, 1994.

[7] B. Lee, et al., "Algorithmic and Architectural Enhancements for Real-time MPEG-1 Decoding on a General Purpose RISC Workstation," IEEE Trans. Circuits and systems for video technology, VOL.5, No.5, Oct. 1995.

[8] L. McMillan, et al., "A forward-mapping realization of the inverse discrete cosine transform," Data Compression Conference 1992, pp. 219-228.

[9] Y. Arai, et al., "A Fast IDCT-SQ Scheme for Images," Trans. IEICE, pp. 1095-1097, 1988.

[10] W. Chen, et al., "A Fast computational Algorithm for the Discrete Cosine Transform," IEEE Trans. COM-25, pp. 1004-1011, 1977.

[11] C. Loeffler, et al., "Practical Fast 1-D DCT Algorithm with Eleven Multiplications,"Proc.ICASSP 1989, pp. 988-991.

[12] D. Ishii, et al., "Parallel Variable Length Decoding with Inverse Quantization for a Software MPEG-2 Decoder", Proc. of the IEEE Workshop on Signal Processing Systems, 1997.