Journal of Zhejiang University SCIENCE
ISSN 1009-3095
http://www.zju.edu.cn/jzus
E-mail: jzus@zju.edu.cn

A complexity-scalable software-based MPEG-2 video encoder

CHEN Guo-bin (陈国斌)[†1], LU Xin-ning (陆新宁)[2], WANG Xing-guo (王兴国)[1], LIU Ji-lin (刘济林)[1]

([1]*Institute of Information & Communication Engineering, Zhejiang University, Hangzhou 310027, China*)

([2]*Division of Science & Technology, Department of Public Security, Zhejiang Province, Hangzhou 310027, China*)

[†]E-mail: chenguobin@zju.edu.cn

Received Apr. 16, 2003; revision accepted Spet. 21, 2003

Abstract: With the development of general-purpose processors (GPP) and video signal processing algorithms, it is possible to implement a software-based real-time video encoder on GPP, and its low cost and easy upgrade attract developers' interests to transfer video encoding from specialized hardware to more flexible software. In this paper, the encoding structure is set up first to support complexity scalability; then a lot of high performance algorithms are used on the key time-consuming modules in coding process; finally, at programming level, processor characteristics are considered to improve data access efficiency and processing parallelism. Other programming methods such as lookup table are adopted to reduce the computational complexity. Simulation results showed that these ideas could not only improve the global performance of video coding, but also provide great flexibility in complexity regulation.

Key words: Video coding, Complexity scalability, Motion estimation, SIMD
Document code: A **CLC number:** TN919.81

INTRODUCTION

Digital video compression techniques have played an important role in the field of telecommunication and multimedia systems where bandwidth and storage resources are limited. The prime important mission of video coding is reducing the coding bitrate without losing quality. MPEG-2 (ISO/IEC, 1995) released by experts of ITU-T and ISO/IEC is a remarkable achievement of video compression technology and widely used in digital storage media, television and communications systems.

The compression methods employed in MPEG-2 are hybrid DPCM/DCT algorithms. Due to its high computational demands, video encoding was mainly carried out on specialized hardware. But professional equipments are too expensive for the consumer market and lacked flexibility. However, with recent progress in general-purpose processors and digital signal processors (DSP), video encoding on these processors without hardware assistance may be an alternative with advantages of short development cycles, easy upgrade and low cost. Video coding consumes much computation time, and so, is a problem for real-time applications, such as on-line transmission of live video, videoconference, etc. Fortunately, as relevant standards only specify the syntax for bitstream and decoding processing, this permits much leeway in the encoding process (ISO/IEC, 1995). Our objective is to set up an encoding system with a scalable complexity so that the computing power can match with the operation requirements.

The rest of this paper is organized as follows. In Section II MPEG-2 encoding architecture is generally summarized. In Section III Coding complexity measurement is analyzed and the hotspots in the encoding process are highlighted. The optimization and simplification strategies for complexity-scalable video codecs are presented in Section IV. Simulation results and analyses are given in Section V, and conclusions are drawn in the last section.

MPEG-2 CODING PRINCIPLES

DPCM/DCT based video encoding involves motion estimation and compensation, DCT and IDCT, quantization and inverse quantization, and variable length coding. A block diagram of a typical MPEG-2 encoder is shown in Fig.1, where each block specifies a particular function being performed. An input frame F_n is presented for encoding. The encoder first makes a decision on the current frame to be coded as inter-frame or intra-frame at frame level. Then the frame is segmented in units of a macroblock (MB). In an inter-frame mode, the encoder first does motion estimation (ME) to find the motion vectors (MV) for each block and determines whether to use temporal prediction or not. If temporal prediction is employed, the corresponding mode is called an inter mode. After prediction, DCT is applied on prediction errors to remove spatial correlation. Quantization (Q) is the only loss function to remove psycho-visual redundancy in the coding process. Then quantized DCT coefficients are sent to the multiplex coder to form final bitstream using entropy coding. After all MBs of the current frame have been processed, inverse quantization together with IDCT is done to reconstruct the previous coded frame for the future reference. In an intra-frame, all MBs should be coded in intra mode where no temporal prediction is used. Segmented blocks are delivered directly to DCT block, and the following process is similar to that of inter-frame.

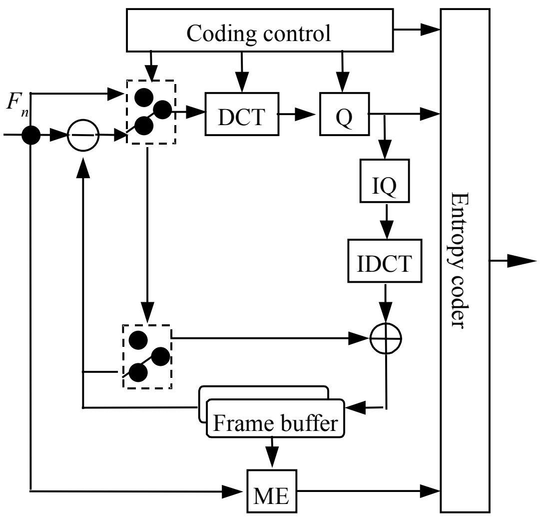

Fig.1 Block diagram of a typical MPEG-2 video encoder
Q: Quantization; ME: Motion estimation; IQ: Inverse quantization

COMPLEXITY ANALYSIS

In fast video coding, coding performance is often measured by the coding frame rate r_{frm} with certain reconstructed video quality. Here, we use r_{frm} as the measurement of the relative coding complexity (RCC) of each frame. RCC can be expressed as

$$c_{\mathrm{r}} = f(r_{\mathrm{frm}}) \quad (1)$$

Therefore, our target is to achieve a near constant complexity; that is, to maintain as much as possible a certain coding frame rate for a desired performance. A feedback mechanism is used to regulate complexity and achieve tradeoff between the quality and coding speed.

Before analyzing our optimization strategies for complexity regulation, we should get the hotspots of compression algorithms first. We use an MPEG-2 encoder (MSSG, 1996) from MPEG software simulation group (MSSG) as the benchmark. Table 1 shows the relative weights of the key coding modules. Obviously, ME is the biggest bottleneck. Intensive analysis showed that all other modules feature more or less parallelism except for entropy coding and some of them can be accelerated

Table 1 Relative weights of the key modules in an MPEG-2 encoder

Sequence	ME	DCT	Q	IDCT	IQ	VLC
Stefan	89.84	4.51	1.04	0.75	0.37	0.62
Coastguard	91.28	4.09	0.94	0.66	0.33	0.61
Foreman	73.48	10.23	4.03	3.25	0.82	0.99

at algorithm level. The flexibility of standard also provides chances for implementing various compliant video encoders. Fig.2 shows a complexity-scalable MPEG-2 video encoder.

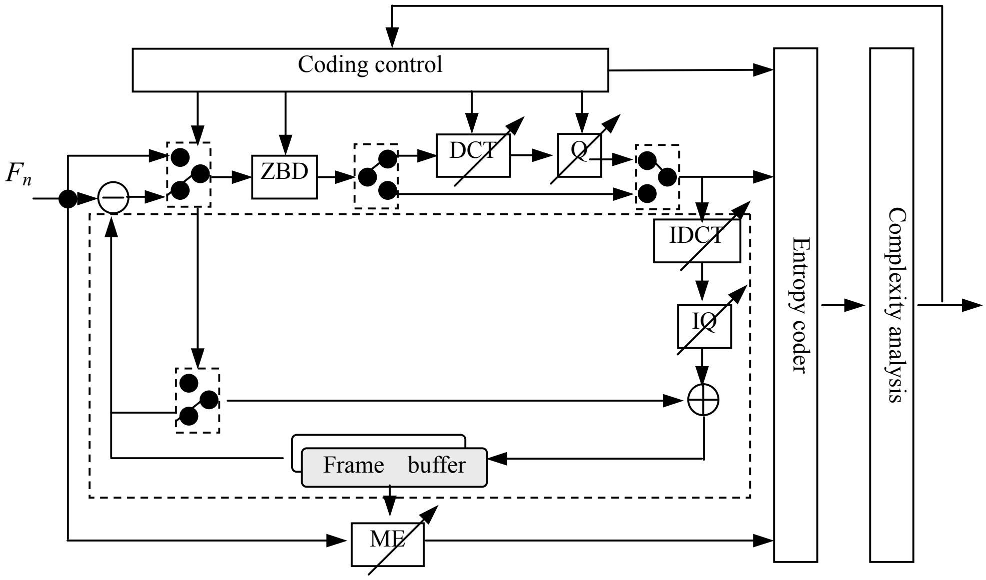

Fig.2 A complexity-scalable MPEG-2 video encoder
ZBD: Zero block detection

OPTIMIZATION STRATEGIES

Many experiments were conducted to derive some strategies for optimizing the video encoder to meet various processor powers.

Coding control strategies

1. Mode selection

Compared with MPEG-1, MPEG-2 has added extra tools for more efficient compression of interlaced video signals. For the interlaced sequence, encoder permits each inter MB to be compensated by either frame mode or field mode. However, the computation complexities of two compensation modes are different, with that of the latter being almost doubled. For instance, when field motion estimation is used in motion estimation, there exist five search modes, from top field to top field, bottom field to bottom field, top field to bottom field, bottom field to top field and frame to frame, while progressive video only requires search from frame to frame.

There are two methods for reducing interlaced video coding complexity. The first is to limit mode decision, e.g. as aforementioned, only search from top field to top field and bottom field to bottom field are adopt for motion estimation and compensation (McVeigh *et al.*, 2000). Another method is to convert interlaced signals into progressive sequence before compression, which results in more computation reduction than the first idea.

2. Zero block detection

In low bit rate applications, all the DCT coefficients of many of the blocks are zero after quantization. This means that if we can detect a priori which blocks are being zeroed out after quantization, we can forego DCT, quantization, inverse quantization and IDCT for those blocks. For example, if the sum of the absolute error (SAE) of block coefficients is less than a certain threshold, the block can be classified as a zero block (Bist *et al.*, 1998; Lin, 2002).

$$SAE = \sum_{i=1}^{8}\sum_{j=1}^{8}\left|e(i,j)\right| \le Thres \times q\_scale \qquad (2)$$

Where $e(i, j)$ is the pel value at (i, j)-th location, q_scale is quantiser scale.

3. No reconstructing frame process

When the bitrate is high, there are no obvious quality differences between the original frames and reconstructed frames (McVeigh *et al.*, 2000). We were able to devise a shortcut that skips the reconstruction process (indicate by the dash-line in Fig.2). Based on this idea the encoding operation could be reduced dramatically, and it only results in small degradation in perceptual image quality.

4. Pseudo skipping frame

In real-time coding scenario or live transmission, it is common that either microprocessors power may be inefficient to complete compression in a given time, or the target bitrate is too low. When this condition appears, the video coding speed and quality will deteriorated importantly. A way to solve this problem is pseudo dropping frame or skipping current frame and forcing encoder re-encode previous frame to keep the decoder synchronized. The merit of dropping frames is their very low computational complexity because the previous coded frame information can be re-used, and perceptual image quality is better; the other benefit is that more bits can be saved for later frame use.

Individual function optimization

1. Motion estimation

In a hybrid DPCM/DCT encoder, motion estimation is the most computationally intensive function, consumes up to 70% coding operation when brute force search is used. It determines both coding speed and reconstructed picture quality. Several algorithm use the MV field spatio-temporal correlations to improve search speed; the most famous of such algorithms are MVFAST and PMVFAST (ISO/IEC, 2000). Compared with full search algorithm, they reduce computational complexity significantly with a little drop in objective picture quality. When ME is only performed on reference frames, greater computation decrease and higher reconstructed frames quality can be obtained than that obtained by first performing ME on original frames and refining the reference frames later. The use of simpler matching cost function can lead to further computation reduction.

2. DCT, IDCT

DCT is a close approximate to the KL transform for a large class of images, and the fact that DCT is a fast transform makes it widely used in video coding. But its computation demand still cannot meet the real-time video-processing requirement. Typical encoders usually calculate all 8×8 DCT coefficients regardless of the quantization. However, in DCT-based image coding, quantized transform coefficients are often zero (Chen *et al.*, 2002; Pao and Sun, 1998), especially in inter mode. This means we can use an approximation of calculations for DCT with acceptable quality degradation. Various DCT coefficient masks can be applied to DCT blocks to reduce the amount of DCT computation. Fig.3 illustrates a set of such masks in descending order of complexity.

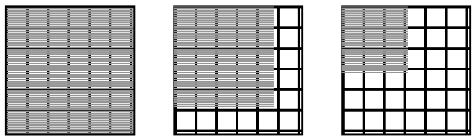

Fig.3 Examples of DCT masks

For IDCT, when an all-zero block is detected, frame-reconstruction will yield substantial speedup by eliminating the regular inverse DCT. When the block only has a DC coefficient, complex calculations can be replaced with a simple assignment.

$$f(x,y)=\frac{1}{8}F(0,0) \qquad 0\le x,y<8 \tag{3}$$

Where $f(x, y)$ is the IDCT coefficient value at (x, y)-th location, $F(0,0)$ is the DC coefficient.

Processor-specific methods

In this section, lots of optimization concepts related to processor architectures are presented. Some issues require using assembly language while others only need a small adjustment on high-level language (Gerber, 2002).

1. SIMD

Both coarse and fine-grained parallelism exists in DPCM/DCT encoding process. Many microprocessor vendors have taken advantage of single instruction multi-data (SIMD) technologies to extend the parallelism of data processing and provide a set of instructions for these multimedia applications, such as Intel's MMX, SSE and SSE2, Motorola's Altivec, SUN's VIS or Mip's MDMX (Conte *et al.*, 1997). In particular, developers can use application-orient instructions to optimize video decoding and encoding. In MPEG-*x* and H.26*x* standards, pavg, psad in SSE and SSE2, pdist in VIS, etc, could be used to speedup motion compensation (MC) and motion estimation (ME) algorithms. Except for Huffman (variable length) coding, all other operations in video coding can use SIMD technologies.

2. Prefetching

The gap between processor speed and memory access causes a significant time consumption in the memory system. Data prefetching is a very efficient method for improving memory access performance. Intel Pentium IV has two mechanisms for prefetching. One is a software-controlled prefetch, the other is an automatic hardware prefetching. It is reported that proper prefetching can result in up to two times speed improvement in memory access (Daniel *et al.*, 2000). To maximize the benefits from these cache control instructions, careful attention should be paid to issues such as identifying datasets worth prefetching.

3. Branching

One of the most basic operations of software is the conditional branch. Unfortunately, conditional branches are also one of the most difficult instructions for the processor to execute efficiently, because it breaks the in-order flow of instructions. Improving the predictability of branches or eliminating and reducing the number of branches can increase the operation speed significantly. Intel offers some instructions, such as cmov and cset, to eliminate branches. Loop unrolling can also decrease branching overhead, since it eliminates some of branches (Gerber, 2002).

Programming optimization methods

There are several methods for optimizing code at programming level.

(1) Use handcraft assembly language to explicitly optimize code.

(2) Use short byte data type and integer type.

(3) Use lookup tables

A lookup table which stores pre-calculated results can be used to avoid executing slow instructions. Because memory access speeds are faster than arithmetical operation speed, processing time could be reduced greatly. In DPCM/DCT coding, there are many multiply/divide operations in quantization and inverse quantization; the lookup table may be an attractive alternative for reducing computation complexity. If lookup tables are used, we must organize the table to maximize the cache hit and keep the table as small as possible (Gerber, 2002).

SIMULATION RESULTS

We conducted extensive simulation to test the performance of our algorithms on Pentium IV processors (1.6 G). The results on two sequences are listed in stefan and coast-guard (CIF, 120 frames). The former is compressed at a bitrate of 1.5 Mbps; the latter is at 1.2 Mbps. Both sequences are encoded at 30 fps, and the length of GOP is 15, the length between two anchor frames is 1. We evaluated the performance of the following algorithms: reference algorithm from MSSG (RA), optimization algorithm with fast motion estimation and processor-specific method (OA1), OA1 combined with zero block detection (OA2) and smart skip frame algorithm (SSFA). Full search is used for RA in motion estimation stage, while other algorithms adopt PMVFAST. The PSNR and coding complexity comparison is shown in Fig.4 and Fig.5 respectively, and average performance comparison is listed in Table 2.

A number of conclusions could be made from our experiments. Compared with RA, other algorithms always have a speedup of over 40 times with a comparable quality. We only regulated the complexity in a low range. To get faster coding speed, we could enlarge the threshold of zero block detection Thres. But Thres was greater than 15 for stefan and 7 for coastguard, led to great degradation

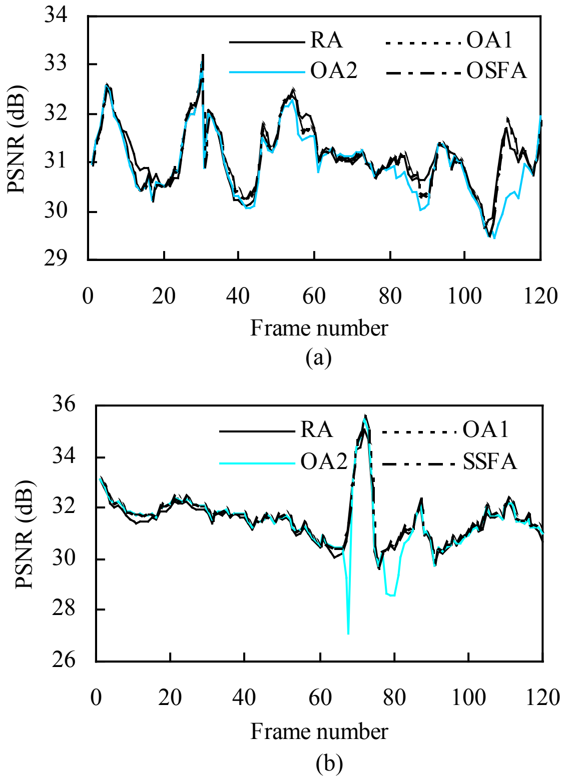

Fig.4 PSNR comparison of each algorithm
(a) Stefan; (b) coastguard

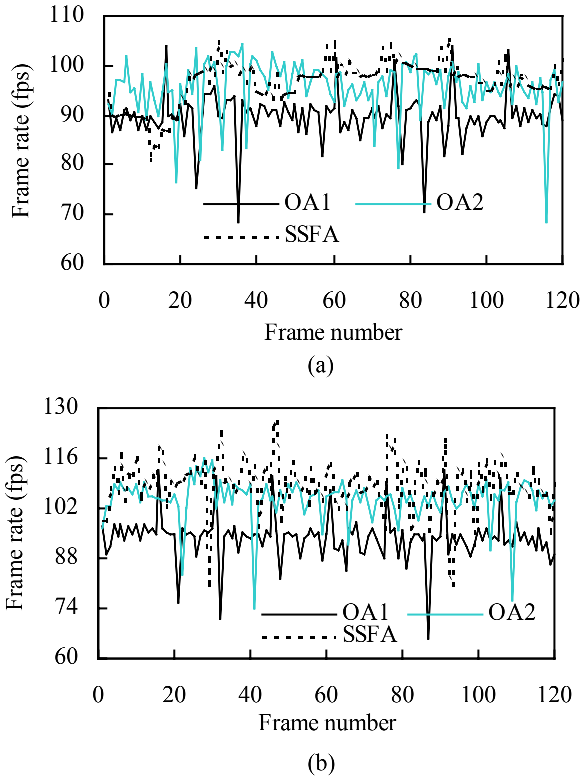

Fig. 5 Coding speed comparison of each optimized algorithm (a) Stefan; (b) Coastguard

Table 2 Average performance comparison of difference algorithm

Sequence	Algorithm	PSNR	Frame rate	Speedup
Stefan	RA	31.152	2.187	1
	OA1	31.140	89.875	41.095
	OA2	30.99	95.9	43.85
	SSFA	31.136	96.81	44.266
Coastguard	RA	31.467	2.023	1
	OA1	31.615	90.861	44.914
	OA2	31.446	98.541	48.71
	SSFA	31.614	97.542	48.217

in some frames due to mis-evaluation of zero block. For SSFA, we set the desired frame rate to a high one; the complexity regulation feedback mechanism would drop frames according previous coding speed. However, if the desired coding frame rate was too high beyond the processors' capacity, the coder had to drop too many frames. In this case, the reconstructed videos yielded annoying "jitter" for the reason of irregular frame dropping, and their perceptual quality deteriorated obviously.

CONCLUSION

This paper proposed several strategies for realizing a complexity-scalable video encoder. Most of them were considered from the viewpoint of video coding algorithms and structure; others were related to the processors' architecture and programming. Different combinations of these methods could be selected to yield various complexity encoders to meet special requirements. MPEG-2 encoding was used here to illustrate the efficiency of these methods, but most of them could be applied on other DPCM/DCT based video encoder. It should be emphasized that all these ideas involve tradeoff between implementation complexity and target video quality. If the coding speed is the most required but computing power is insufficient, image quality deterioration is inevitable.

Reference

Bist, A., Wu, W., Hsueh, A., 1998. Intelligent Pre-Quantization in Motion Compensation Video Coding. Proc: ITU-T Study Group 16 Video Experts Group, Doc. Q15-D-35, Tampere, Finland.

Chen, Y., Zhong, Z., Lan, T., Peng, S., van Zon, K., 2002, Regulated complexity scalable MPEG-2 video decoding for media processors. *IEEE Trans Circuits Syst. Video Technol.*, **12**(8):678-687.

Conte, T., Dubey, P., Jennings, M., Lee, R., Peleg, A., Rathnam, S., Schlansker, M., Song, P., Wolfe, A., 1997. Challenges to combining general-purpose and multimedia processors. *Computer*, **30**(12):33-37.

Daniel, F., Lee, R., Flynn, M., 2000. Hardware and software cache prefetching techniques for MPEG benchmarks. *IEEE Trans. Circuits Syst. Video Technol.*, **10**(5):782-796.

Gerber, R., 2002. The Software Optimization Cookbook. Intel Press, Hillsboro.

ISO/IEC, 1995. Information Technology, Generic Coding of Moving Pictures and Associated Audio Information. Recommendation ITU-T H.262, Geneva, Switzerland.

ISO/IEC, 2000. Optimization Model-Version 1.0. ISO/IEC/JTC1/SC29/WG11 N3324, Noordwijkerout, Finland.

Lin, W., 2002. On fast firmware/software-based video coding. *IEEE Trans. Consumer Electronics*, **48**(2):209-219.

McVeigh, J., Chen, G., Goldstein, J., Gupta, A., Keith, M., Wood, S., 2000. A software-based real-time MPEG-2 video encoder. *IEEE Trans. Circuits and Syst. Video Technology*, **10**(7):1178-1184.

MSSG, 1996. MPEG-2 Encoder/Decoder Ver.1.2. url: http://www.mpeg.org/MSSG.

Pao, I., Sun, M., 1998. Approximation of calculations for forward discrete cosine transform. *IEEE Trans. Circuits and Syst. Video Technology*, **8**(3):264-268.