

A Software-Based Real-Time MPEG-2 Video Encoder

Jeff McVeigh, George K. Chen, Judi Goldstein, Atul Gupta, Mike Keith, and Steve Wood

Abstract—Dedicated hardware previously has been required to perform real-time MPEG-2 video encoding. However, with increases in clock frequency and the introduction of video-specific instruction sets, general-purpose processors can now approximate the function and performance of single-function hardware. In this paper, we describe a software-only MPEG-2 (MP@ML) video encoder implemented on a personal computer using an IntelTM Pentium[®] III processor. This encoder is capable of real-time operation while consuming less than 70% of the processor. The main contribution of this work is a set of algorithmic simplifications that significantly reduces the computational load of the encoding process while only slightly degrading the subjective video quality compared to encoders that are more exhaustive.

Index Terms—MPEG-2, real-time implementation, video encoder.

I. INTRODUCTION

MPEG-2 [1] is used widely in state-of-the-art video systems, including DVD, DBS, and HDTV. Its popularity can be attributed to its efficient compression of both interlaced and progressive content at bit rates ranging from 4 Mbits/s (DVD) to 19 Mbits/s (HDTV). Severe bandwidth constraints and the “encode once, decode many times” nature of these applications dictate that the content should be encoded at the highest possible quality for the desired bit rate. Hence, dedicated high-cost hardware encoders typically are used in broadcast and mastering systems.

Recently, numerous consumer-oriented applications have emerged that require real-time MPEG-2 video encoding. Examples include nonlinear editing and archiving of home movies onto writeable DVDs, and timeshifted viewing of broadcast television. Since these applications require an encoder in each consumer device, the encoder must be easy to use, and it must be low cost. Due to increases in clock frequency, the introduction of video-specific instruction sets [2], and the availability of low-cost storage devices, the personal computer is now well suited to meet the requirements of these emerging applications.

In this paper, we present design details and performance results of a software-only MPEG-2 (MP@ML) video encoder. The main contribution of this work is a set of novel algorithmic simplifications that significantly reduces the computational load of the encoding process. These simplifications, combined with judicious design decisions and a fully optimized implementation, enable real-time operation on a Pentium III processor per-

sonal computer. In spite of these simplifications, we demonstrate that our encoder requires only a moderate increase in bit rate to achieve visual quality similar to that obtained by the publicly available Test Model 5 (TM5) implementation¹ [3], [4].

The outline of the paper is as follows. Section II provides complexity and quality requirements for the encoder. Section III describes the design decisions and algorithmic simplifications developed to achieve these requirements. The encoder complexity and quality results are provided in Section IV. Section V contains a summary and conclusions.

II. REQUIREMENTS

Prior to developing our encoder, we established a set of requirements that must be achieved to satisfy our target applications. The most important requirement was to be able to encode at the maximum MP@ML resolution and frame rate² while consuming less than 70% of the computational resources of a 500-MHz Pentium III processor.³ We selected a maximum CPU usage of 70% to provide headroom for other system tasks, such as audio processing and user-interface control. This requirement translates into having to encode up to 40 500 macroblocks using at most $0.70 \times 500 \times 10^6$ processing cycles per second.

Additionally, since the encoder was targeted at consumer applications, we realized that it did not need to achieve the same quality as professional MPEG-2 encoders for a given bit rate. Instead, we set a goal of subjective visual quality similar to S-VHS while being able to record up to two hours of video in less than 5 GB's of storage. This implies a bit rate of approximately 6 Mbits/s.

To satisfy our requirements, the encoder must be able to compress each macroblock using less than 8642 cycles of the processor. In comparison, we estimate that a fully optimized TM5 implementation would consume over 7.5×10^6 cycles per macroblock.⁴ Fortunately, the MPEG-2 video standard only describes the bit-stream syntax and decoding process; this permits a great degree of leeway in the encoding process.

¹TM5 was developed to provide a reference implementation that provides adequate visual quality. It is well understood that TM5 is not the most efficient or highest quality encoder available. Instead, we use it throughout this paper as a baseline for comparisons.

²The maximum resolution and frame rate is 720×480 at 30 frames/s for NTSC and 720×576 at 25 frames/s for PAL.

³At the time that this encoder was developed, the fastest publicly available Pentium III processor operated at 500 MHz. This encoder will consume a correspondingly lower percentage of faster processors available today.

⁴This estimate assumes an I-frame spacing of 15, an I/P-frame spacing of 3, a motion-estimation search range of ± 63 by ± 31 pixels, and 200 processing cycles per reference macroblock prediction distortion calculation.

Manuscript received July 1999; revised February 2000. This paper was recommended by Guest Editor A. Tabatabai.

The authors are with the Intel Architecture Labs, Intel Corporation, Hillsboro, OR 97124 USA (e-mail: jeffrey.mcveigh@intel.com).

Publisher Item Identifier S 1051-8215(00)08213-6.

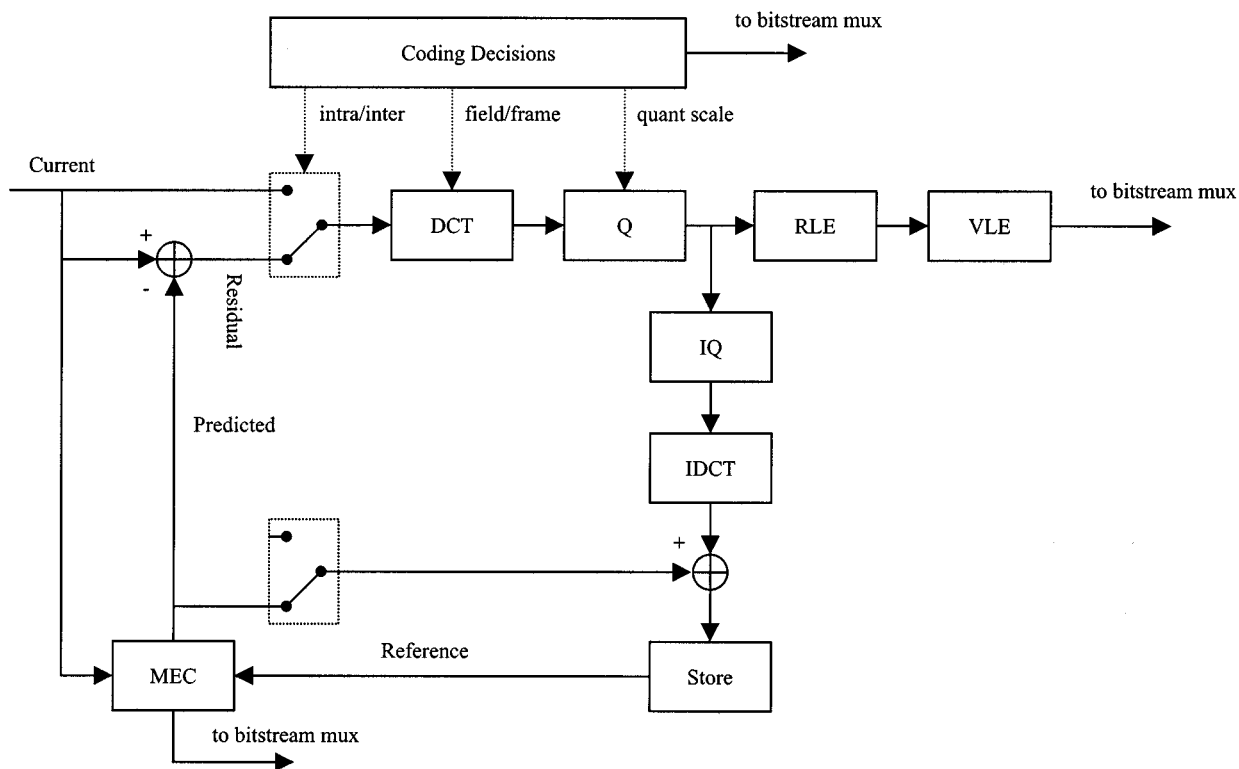


Fig. 1. MPEG-2 video encoder block diagram.

III. DESIGN

The typical MPEG-2 video encoder block diagram is illustrated in Fig. 1. We separated this encoding process into functions that provide little flexibility in their implementation, and those that can be modified and tuned greatly. The relatively inflexible functions include the discrete cosine transform (DCT), quantization (Q), run-length encoding (RLE), and variable length encoding (VLE) modules. We used straightforward implementations of these functions, with the only notable detail that we tightly coupled the DCT and quantization stages based on the technique described in [5]. Based on experimental comparisons, we developed a set of simplifications for the other encoding functions that yielded the greatest speed-up while minimizing quality degradations.⁵

A. GOP Structure and Reference Frame Reconstruction

It is well known that motion estimation often consumes the largest percentage of processing cycles for an MPEG-2 video encoder. This, coupled with our desire to minimize the number of frames that must be reconstructed, led us to our first design decision. Namely, we use an IBB GOP structure ($N = 3$, $M = 3$). While this structure limits coding efficiency, since every third frame is intra-coded, motion estimation is performed on only two-thirds of the frames. This GOP structure also causes all inter-coded frames to be predicted from an I-frame. By keeping these “prediction chains” to one frame, we were able to use

⁵It should be noted that our encoder also supports operating modes that do not use all of these simplifications. These increased complexity modes are useful for providing improved visual quality and/or compression efficiency with corresponding increases in processing resources.

a shortcut that makes the encoder technically nonconformant without perceptually degrading the visual quality.

Since a decoder only has access to previously decoded reference frames, an encoder normally uses the identical reference frames for the prediction of inter-coded frames. Reconstructing frames at the encoder entails inverse DCT (IDCT) and inverse quantization (IQ) stages. To avoid these processing steps, our encoder uses the original I-frame data for the prediction of the B-frames. Therefore, the block diagram of our encoder does not include the IDCT and IQ steps shown in Fig. 1. Instead, the original I-frame data is stored directly for the motion estimation and compensation (MEC) step for subsequent inter-coded pictures.

Although using the original reference frame data in the motion compensation step causes the encoder and decoder to lose synchronization, the error accumulation is minimized due to the high I-frame frequency. In addition, the maximum error accumulation is equal to the quantization error for the reference I-frame, which is relatively small when encoding at high bit rates.

B. Prediction Constraints

For interlaced frame pictures, MPEG-2 permits each macroblock to be encoded using either frame or field motion. If field motion vectors are used, each field of the macroblock may be predicted from either the top or the bottom field of the reference frame. Furthermore, macroblocks within B-frames may be bidirectionally predicted. For B-frames, TM5 performs six complete motion estimation processes to find the motion vector that yields the minimum prediction distortion among these various choices.

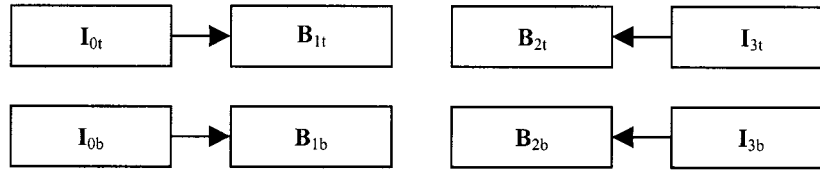


Fig. 2. Unidirectional B-frame and even-parity prediction.

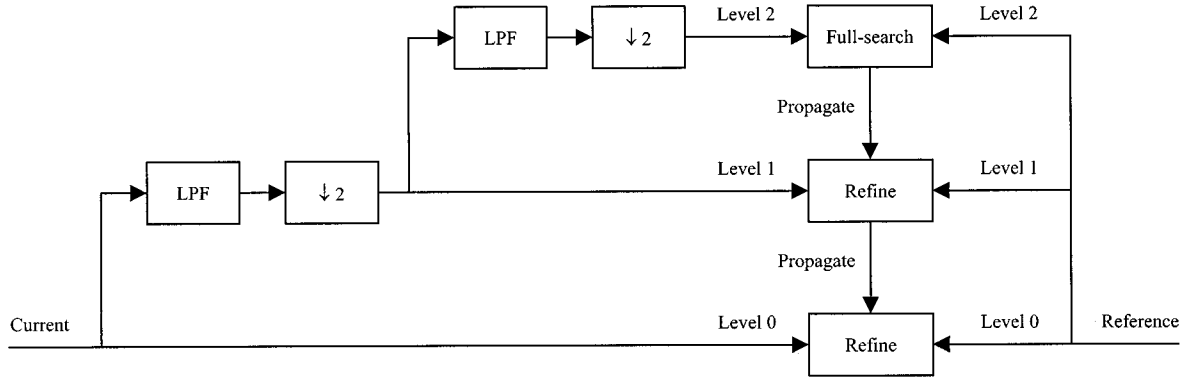


Fig. 3. Hierarchical motion estimation block diagram.

To avoid exhaustively searching for the best motion vector, our encoder only performs “even-parity” field motion estimation. In other words, we perform field-based motion estimation to find the best vector for the top field of the macroblock using only the top reference field, and the vector for the bottom field is estimated from only the bottom reference field. If the resulting vectors are equivalent, they are coded as a frame motion vector, which is useful for stationary regions.

Additionally, for B-frames, we only perform motion estimation using the temporally closest reference frame (i.e., unidirectional prediction). While these constraints can limit the prediction quality, especially at scene changes, only one complete motion estimation process is required for each frame.

These prediction constraints are illustrated in Fig. 2. The top field of the first B-frame (B_{1t}) is predicted from only the top field of the temporally previous I-frame (I_{0t}). Similarly, the bottom field of the second B-frame (B_{2b}) is predicted from only the bottom field of the temporally next I-frame (I_{3t}).

C. Motion Estimation

Early simulation results indicated that a motion estimation search range of ± 63 pixels horizontally and ± 15 pixels vertically adequately captures the motion between successive frames in most broadcast content. If we were to implement a full-search technique such as in TM5, this search range would require approximately 4000 prediction distortion calculations per macroblock. To reduce this complexity, we implemented a three-level hierarchical motion estimation technique that is a simplified version of the method described in [6] (see Fig. 3).

We first perform a two-stage pyramidal decomposition of the reference frame and the current frame to be encoded, which yields three resolutions of luminance data. Each stage of the decomposition entails two-dimensional low-pass filtering (LPF)

and subsampling steps. To simplify implementation, we use a separable 2-D LPF, whose 1-D impulse response is given by

$$h[n] = 0.25\delta[n-1] + 0.5\delta[n] + 0.25\delta[n+1].$$

Each field is decomposed independently to maintain the interlaced content of the frame data. We label the full-resolution data Level 0, the quarter-resolution data Level 1, and the sixteenth-resolution data Level 2.

After decomposing the field data, full-search motion estimation is performed on the Level 2 data. Since this level has a resolution that is 1/4 that of the original data in each dimension, the target horizontal and vertical search ranges are reduced by a factor of four. In addition, since we only operate on field data, the vertical search range is reduced by another factor of two. This results in a search range of ± 15 pixels horizontally and ± 1 pixels vertically (field coordinates). At all levels, we still use the full field macroblock size of 16×8 pixels, so there are only 1/16th the original number of macroblocks in Level 2.

After calculating the motion vectors for each field macroblock in Level 2, these vectors are propagated to the Level 1 resolution by multiplying each component by two. Since a macroblock in Level 2 spans four macroblocks in Level 1, each motion vector becomes the candidate vector for four macroblocks in Level 1. Each candidate motion vector is refined independently using a conjugate search technique, whereby we first search horizontally by ± 1 pixels and then vertically by ± 1 pixels around the horizontal location with the minimum prediction distortion.

Again, the Level 1 motion vectors are propagated to four macroblocks in Level 0. However, prior to refining each candidate motion vector to the full-resolution data, the prediction distortion for the (0, 0) motion vector is compared against a threshold to estimate whether motion exists within the macroblock.

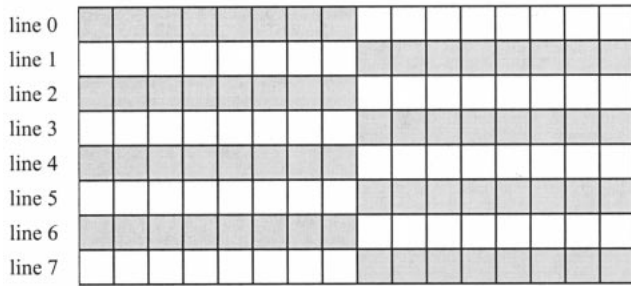


Fig. 4. Pseudo-checkerboard field macroblock MAD. The MAD is calculated only over the 64 shaded pixel locations, as opposed to all pixel locations for a traditional MAD calculation.

threshold is adaptively set to 1/4 the average distortion from the Level 1 refinement stage. If the distortion is less than the threshold, the motion vector is set to (0, 0) and the final refinement stage is skipped for this macroblock. Otherwise, the Level 1 motion vectors are refined using the conjugate refinement process. Finally, half-pixel motion vectors are obtained using another conjugate refinement.

Even with the addition of the decomposition process, the entire hierarchical motion estimation process requires approximately 20 prediction distortion calculations per macroblock, which is 200 times less complex than a full-search technique. We have also found that for the same effective search range the hierarchical technique closely approximates the performance of a full-search for typical broadcast content.

One final simplification on the motion estimation process is the prediction distortion measure used to find the best motion vectors. As is common practice, our distortion measure is based on the mean absolute difference (MAD) between luminance samples within the current and reference macroblocks. However, instead of calculating the MAD over all pixels in the macroblock, we use a “pseudo-checkerboard pattern” MAD. For a 16×8 field macroblock, this checkerboard MAD consists of calculating the MAD over the 8 pixels on the left-half side of the even-numbered lines and the 8 pixels on the right-half side of the odd-numbered lines (see Fig. 4). Therefore, the MAD is calculated only over 64 pixels within the 16×8 field macroblock.

We have found that this subsampling structure adequately represents the content of the macroblock while reducing the complexity of the motion estimation process by another factor of two. It is also more amenable to parallel (SIMD) processing than a standard-checkerboard pattern [7].

D. Coding Decisions

Upon the completion of the motion estimation and compensation stage, an encoder must make numerous decisions on how each macroblock will be encoded. These decisions include intra/inter coding, field/frame DCT and quantization scale selection. The underlying objective of these decisions is to improve the overall rate-distortion performance of the compressed information. We developed numerous simplifications, compared to TM5, that effectively reuse macroblock statistics from the motion estimation stage to make these decisions. Additionally, we developed a technique to detect whether an

inter-coded block can be skipped prior to the DCT stage to further reduce the encoding complexity.

1) *Intra- versus Inter-Coded Macroblock*: The first decision that must be made in inter-coded frames is whether to intra- or inter-code each macroblock. Motion-estimation MAD gives an indication of how well the macroblock will be inter coded; the smaller the MAD, the fewer the bits typically required to represent the residual information. For the intra/inter decision, we compare this MAD to an estimate of how well the macroblock will be intra-coded, which we term the “intra complexity.”

We separate the intra complexity into horizontal and vertical components. These components estimate the relative horizontal and vertical similarity within the macroblock, respectively. For uniform macroblocks, which can be efficiently intra coded, each component will be relatively small.

We define the horizontal complexity component as merely the mean absolute differences between horizontally adjacent pixels. Since either field or frame DCT can be used on each macroblock, the vertical complexity component is equal to the minimum of the field and frame vertical complexities. We define the frame vertical complexity as the MAD between vertically adjacent pixels, while the field vertical complexity as the MAD between pixels that are separated by one vertical line.

These complexity measures are given by the following equations, where $x(i, j)$ is the value of the pixel at the i th row and j th column within the macroblock:

$$\begin{aligned} \text{horizontal complexity} &= \frac{1}{(16)(15)} \sum_{i=0}^{15} \sum_{j=0}^{14} |x(i, j) - x(i, j+1)| \\ \text{frame vertical complexity} &= \frac{1}{(15)(16)} \sum_{i=0}^{14} \sum_{j=0}^{15} |x(i, j) - x(i+1, j)| \\ \text{field vertical complexity} &= \frac{1}{(14)(16)} \sum_{i=0}^{13} \sum_{j=0}^{15} |x(i, j) - x(i+2, j)| \end{aligned}$$

The intra-macroblock complexity is then given by the average of the horizontal and vertical complexities. If the intra activity is less than the motion-estimation MAD, the macroblock is intra coded; otherwise, it is inter coded.

2) *Field versus Frame DCT*: If the macroblock is intra coded, the field versus frame DCT decision is based on the vertical complexity. If the frame vertical complexity is less than the field vertical complexity, it is assumed that the macroblock will compact into fewer DCT coefficients when ordered as a frame macroblock. Therefore, frame-based DCT is performed for this case. When the field complexity is less than the frame complexity, field-based DCT is performed.

For inter-coded macroblocks, we rely solely on the motion type for the DCT decision. If the motion-estimation step yielded two identical field motion vectors for the macroblock, frame motion, and frame DCT are used. Otherwise, field motion and field DCT are used.

3) *Macroblock Quantization Scale*: The complexity calculations used for the previous decisions are used again to select

TABLE I
QUALITY AND CPU USAGE COMPARISON OF DIFFERENT OPERATING-MODES OF REAL-TIME ENCODER AND TM5 SIMULATOR
AT 6 MBITS/S. THE CPU USAGE RESULTS ARE BASED ON A PENTIUM III PROCESSOR OPERATED AT 500 MHz

	Intel A	Intel B	Intel C	Intel D	TM5
GOP (N, M)	(3, 3)	(3, 3)	(15, 3)	(15, 3)	(15, 3)
Use original reference	Yes	No	No	No	No
Empty block detection	Yes	Yes	Yes	No	No
Simplified ME	Yes	Yes	Yes	Yes	No
CPU usage (%)	66.97	77.28	90.22	97.15	60000 (est.)
PSNR (dB)	36.26	36.35	37.02	36.93	37.56

the macroblock quantization scale for rate control. To simplify comparisons, we used a rate-control mechanism similar to that described in TM5.

The quantization scale for each macroblock is calculated taking into consideration the actual versus target compressed frame sizes, the VBV buffer fullness, and the relative macroblock "activity." Since our complexity measures provide an estimate for how well a macroblock can be compressed, these values are used by the rate control mechanism. For intra-coded macroblocks, the activity is set to the intra complexity. For inter-coded macroblocks, the activity is set to the motion-estimation MAD.

4) *Empty Block Detection*: Depending on the motion-estimation performance and the macroblock quantization scale, many inter-coded blocks will contain only zero-valued DCT coefficients after quantization; i.e., they are empty. Simulation results indicated that approximately half of the blocks were empty when encoding at 6 Mbits/s. The run-length and variable-length encoding stages obviously can be skipped for these blocks. To further simplify the encoding process, we developed a technique to detect empty blocks prior to the DCT stage. Our method is similar to other techniques, such as the one described in [8], except ours is based on an adaptive threshold to increase the detection accuracy.

Prior to using the detector, a threshold must be obtained. During initialization, we first calculate the MAD between the original and predicted blocks. Using a common rate-distortion model, we then calculate the ratio between the block MAD and the quantization scale (quant). Each block is then passed through the entire encoding process and the empty/not-empty state of the block is used with the ratio to construct two conditional probability functions. After encoding an entire frame, a threshold is calculated that minimizes the probability of false alarms and misses for this empirical data. This completes the initialization stage, and the detector is enabled for subsequent frames.

The MAD/quantization-scale ratio is then calculated for each block after motion compensation. If the ratio is less than the threshold, it is assumed that the block is empty and the remaining encoding stages are skipped. To ensure that the threshold accurately represents the statistics of the video sequence, the threshold is updated periodically.

Fig. 5 shows the two conditional probability functions and the calculated threshold for a sample video sequence. For this sequence, the detector achieves a probability of detection (say empty when empty) of over 92% and a probability of false alarm (say empty when not-empty) of under 6%.

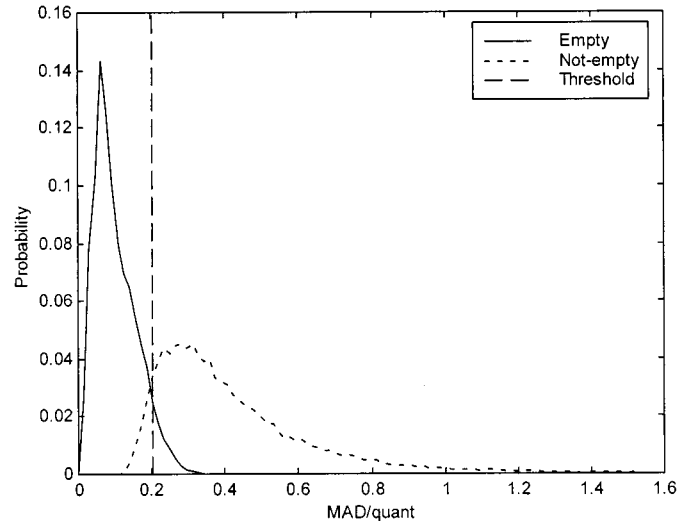


Fig. 5. Sample conditional probability functions for empty block detection.

IV. PERFORMANCE

To analyze the quality and CPU usage of our encoder, we used a 30-second sequence generated through the concatenation of the familiar "stefan," "news," and "coastguard" sequences (720×480 resolution, 29.97 frames/second). This sequence was used to emulate typical broadcast content that contains scene changes and both high and low motion segments.

By using the Pentium cycle counter, we calculated an average of 8268 processing cycles per macroblock for our encoder operating at 6 Mbits/s with all simplifications employed. This translates into roughly 67% of a 500-MHz Pentium III processor.⁶ To illustrate how the various simplifications affect the CPU usage and visual quality, we encoded the same test sequence at four different operating modes of our real-time encoder ("RT1," "RT2," etc.). Each successive operating mode uses fewer encoding simplifications. For example, RT1 uses all described simplifications, while RT2 uses reconstructed reference frames in the motion compensation stage. These results are provided in Table I.

A number of observations can be made from these results. First, in its lowest complexity mode (RT1), our encoder achieves its target CPU utilization of under 70%; however,

⁶The encoder consumes a correspondingly lower percentage of processors operated at higher clock speeds. For example, it consumes approximately 40% of a 933-MHz Pentium III processor. Therefore, the CPU usage of the encoder scales linearly with the processor clock speed, with a scaling factor of

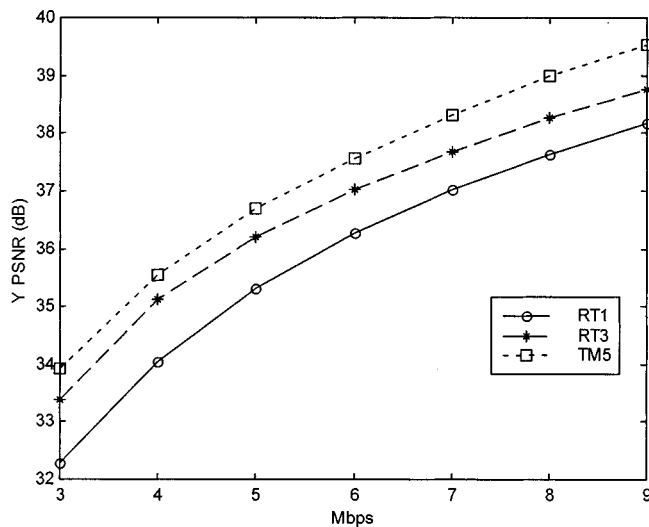


Fig. 6. Rate-distortion comparison between real-time and TM5 encoders.

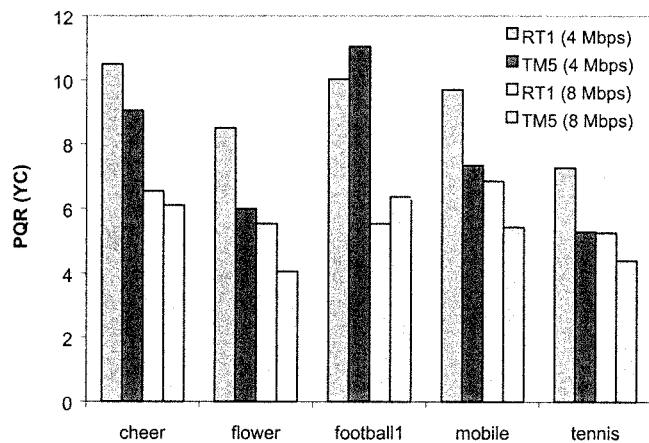


Fig. 7. PQA200 comparison between real-time and TM5 encoders.

the PSNR for the test sequence is significantly lower (1.3 dB) than TM5. Second, using reconstructed I-frames only slightly improves the visual quality, while requiring over 10% more of the processing resources. Third, when the GOP length is increased to 15 frames, the real-time encoder begins to approach the visual quality of TM5 (within 0.5 dB) while still allowing real-time operation. Finally, not using the empty block detector actually reduces the PSNR while at the same time increases the CPU utilization.

We then performed quality comparisons between the TM5 simulator and our encoder operating at its lowest (RT1) and mid-complexity modes (RT3). The TM5 simulator used an I-frame spacing of 15, an I/P frame spacing of 3 and a motion estimation search range of ± 63 pixels horizontally and ± 31 pixels vertically. PSNR versus bit rate results on the concatenated test sequence are shown in Fig. 6. This plot indicates that RT1 achieves approximately the same visual quality operating at 6 Mbps/s as a TM5 implementation does at 4.5 Mbps/s. The RT3 mode achieves the same quality operating at roughly 5 Mbps/s.

Finally, we used a Tektronix PQA200 picture quality analysis system [9] on additional test sequences to compare the perceived video quality. These results are provided in Fig. 7, where lower values indicate fewer perceived distortions or improved visual quality.

V. CONCLUSION

We have presented the design and performance details of a software-only MPEG-2 (MP@ML) video encoder. A set of novel encoding simplifications enables real-time operation on a Pentium III processor personal computer. When operating at its highest complexity real-time mode, the encoder achieves a PSNR that is only 0.5-dB lower than that achieved by an exhaustive TM5 implementation at 6 Mbps/s. The low system cost and processing-power/quality scalability of this encoder make it well suited for consumer video applications.

REFERENCES

- [1] *Information Technology—Generic Coding of Moving Pictures and Associated Audio Information: Video*, ISO/IEC 13818-2:1996(E), 1996.
- [2] *Intel Architecture Software Developer's Manual: Instruction Set Reference*, Intel, Doc. 243 191, 1999.
- [3] *MPEG-2 Video Test Model 5*, ISO/IEC JTC1/SC29/WG11, Doc. N0400, Apr. 1993.
- [4] (1996) MPEG Software Simulation Group, *Test model 5 simulator, version 1.2*. [Online]. Available: <http://www.mpeg.org/MPEG/>
- [5] Y. Arai, T. Agui, and M. Nakajima, "A fast DCT-SQ scheme for images," *Trans. IEICE*, vol. E 71, no. 11, pp. 1095–1097, Nov. 1988.
- [6] M. Accame, F. De Natale, and D. Giusto, "High performance hierarchical block-based motion estimation for real-time video coding," *Real-Time Imaging*, vol. 4, no. 1, pp. 67–79, Feb. 1998.
- [7] C.-K. Cheung and L.-M. Po, "A hierarchical block motion estimation algorithm using partial distortion measure," *Proc. Int. Conf. Image Processing*, vol. 3, pp. 606–609, 1997.
- [8] A. Bist, W. Wu, and A. Hsueh, "Intelligent pre-quantization in motion compensated video coding," *ITU SG16*, Doc. Q15-D-35, Apr. 1998.
- [9] *PQA200 Picture Quality Analysis System: User Manual*, Tektronix, Doc. 071-0091-03, 1998.



Jeff McVeigh received the B.S. degree in electrical engineering in 1992 from Duke University, Durham, NC, and the Ph.D. degree in electrical and computer engineering in 1996 from Carnegie Mellon University, Pittsburgh, PA.

He joined the Intel Architecture Labs of Intel Corporation, Hillsboro, OR, in 1996, where he currently manages the Streaming Video Technology Group. In addition to the MPEG-2 video encoder, he has also developed real-time, software codecs for videoconferencing, and Internet streaming applications, and he represents Intel in the ISO/MPEG standardization effort, where he has co-chaired an *ad hoc* group chartered to improve visual quality on progressive-scan display devices. His current main area of research interest addresses the delivery of high-quality video over heterogeneous networks.

George K. Chen received the B.S. degree in aerospace engineering in 1993, and the M.S. degree in electrical engineering in 1994, both from Georgia Institute of Technology, Atlanta, GA.

He joined the Video Audio Technologies Lab of Intel Architecture Labs, Intel Corporation, Hillsboro, OR, in 1995, primarily working on host CPU-based real-time video encoding algorithms. Past projects include Intel Indeo 4.X Video Real-Time Encoder, Intel Indeo 5.X Video Real-Time Encoder, Intel MPEG-2 Video Real-Time Encoder, and Intel DV codec. His current research interests include high-definition real-time video compression and 3-D/video processing.



Judi Goldstein received the B.S. degree in computer science in 1985 from Beaver College, Glenside, PA.

She joined the David Sarnoff Research Center in 1984, and Intel Corporation, Hillsboro, OR, in 1988. She has worked on a multitude of interactive video projects for the PC, including the first real-time full-screen and full-frame-rate video playback ever demonstrated on a PC, in 1987. In addition to video playback, she has worked on a variety of real-time video capture and compression products. Her current research interests include streaming technologies.



Mike Keith received the B.S.E.E. degree from New Jersey Institute of Technology, Newark, in 1977, and the M.S.E.E. degree from Stanford University, Stanford, CA, in 1978.

He was with the David Sarnoff Research Center, Princeton, NJ, from 1980 to 1990, working on PC multimedia, culminating with work on Digital Video Interactive, a technology subsequently acquired by Intel Corporation. During 1990–1998, he was with Intel Corporation, Hillsboro, OR, working on various projects involving digital video and audio compression, and since 1998 as a Consultant. He is the author of about 50 papers, two books, and 52 U.S. patents. His current fields of interest include software optimization, intellectual property, and recreational linguistics.

Atul Gupta received the B.Tech. degree in electrical engineering from the Indian Institute of Technology, Delhi, India, in 1980, and the M.Sc. degree in computer science from the University of Alberta, Alberta, Canada, in 1982.

He joined Intel Corporation, Hillsboro, OR, in 1983. Past projects have included C and Ada compilers, programming language tools, and video codecs. His current interests include video processing and wireless networks.



Steve Wood received the B.S. degree in mathematics and the B.S. degree in computer science in 1982 from Oregon State University, Corvallis, OR.

He has held various engineering and management positions for Tektronix, Mentor Graphics, and Intel, as well as owning and operating a software consulting firm. He is currently investigating semantic video-processing algorithms at Intel Corporation, Hillsboro, OR.