Introduction to H.264 Advanced Video Coding

Jian-Wen Chen Chao-Yang Kao Youn-Long Lin

Department of Computer Science
National Tsing Hua University
Hsin-Chu, TAIWAN 300
Tel : +886-3-573-1072
Fax : +886-3-572-3694
e-mail : ylin@cs.nthu.edu.tw.

Abstract - We give a tutorial on video coding principles and standards with emphasis on the latest technology called H.264 or MPEG-4 Part 10. We describe a basic method called block-based hybrid coding employed by most video coding standards. We use graphical illustration to show the functionality. This paper is suitable for those who are interested in implementing video codec in embedded software, pure hardwired, or a combination of both.

I. Introduction

Digitized video has played an important role in many consumer electronics applications including VCD, DVD, video phone, portable media player, video conferencing, video recording, e-learning etc. In order to provide solutions of high quality (high frame resolution, high frame rate, and low distortion) or low cost (low bit rate for storage or transmission) or both, video compression is indispensable. Advancement in semiconductor technology makes possible efficient implementation of effective but computationally complicated compression methods.

Because there are a wide range of target applications from low-end to high-end under various constraints such as power consumption and area cost, an application-specific implementation may be pure software, pure hardwired, or something in between. In order to do an optimal implementation, it is essential to fully understand the principles behind and algorithms employed in video coding. Starting with MPEG-1[1] and H.261 [2], video coding techniques/standards have gone through several generations.

The latest standard is called H.264 (also called MPEG-4 AVC, Advanced Video Coding defined in MPEG-4 Part 10) [3][4][5]. Compared with previous standards, H.264 achieves up to 50% improvement in bit-rate efficiency. It has been adopted by many application standards such as HD DVD [6], DVB-H [7], HD-DTV [8], etc. Therefore, its implementation is a very popular research topic to date. In this tutorial paper, we introduce the essential features of H.264.

The rest of this paper is organized as following. In Section II, we give an outline of the block-based hybrid video coding method. In Section III, we describe in more detail each basic coding function. Finally, in Section IV, we draw some conclusions.

II. Block-Based Hybrid Video Coding

A digitized video signal consists of a periodical sequence of images called frame. Each frame consists of a two dimensional array of pixels. Each pixel consists of three color components, R, G and B. Usually, pixel data is converted from RGB to another color space called YUV in which U and V components can be sub-sampled. A block-based coding approach divides a frame into macroblocks each consisting of say 16x16 pixels. In a 4:2:0 format, each MB consists of 16x16 = 256 Y components and 8x8 = 64 U and 64 V components. Each of three components of an MB is processed separately.

Fig. 1 shows a pseudo-code description of how to compress a frame MB by MB. To compress an MB, we use a hybrid of three techniques: prediction, transformation & quantization, and entropy coding. The procedure works on a frame of video. For video sequence level, we need a top level handler, which is not covered in this paper. In the pseudo code, f_t denotes the current frame to be compressed and mode could be I, P, or B.

```
procedure encode_a_frame (f_t, mode)

    for I = 1, N          //** N: #rows of MBs per frame
        for J = 1, M    //** M: #columns of MBs per frame
            Curr_MB = MB(f_t, I, J);
            case (mode)
                I: Pred_MB = Intra_Pred (f'_t, I, J);
                P: Pred_MB = ME (f'_t-1, I, J);
                B: Pred_MB = ME (f'_t-1, f'_t+1, I, J);

            Res_MB = Curr_MB – Pred_MB;
            Res_Coef = Quant(Transform(Res_MB));
            Output(Entropy_code(Res_Coef));

            Reconst_res = ITransform(IQuant(Res_Coef)) ;
            Reconst_MB = Reconst_res + Pred_MB;
            Insert(Reconst_MB, f'_t) ;

end encode_a_frame;
```

Fig. 1. Pseudo Code for Block-Based Hybrid Coding a Video Frame

Prediction tries to find a reference MB that is similar to the current MB under processing so that, instead of the whole current MB, only their (hopefully small) difference needs to be coded. Depending on where the reference MB comes from, prediction is classified into inter-frame prediction and intra-frame prediction. In an inter-predict (P or B) mode, the reference MB is somewhere in a frame before or after the current frame, where the current MB resides. It could also be some weighted function of MBs

0-7803-9451-8/06/$20.00 ©2006 IEEE.

SNAP-1017
IPR2025-01263

from multiple frames. In an intra-predict (I) mode, the reference MB is usually calculated with mathematical functions of neighboring pixels of the current MB.

The difference between the current MB and its prediction is called residual error data (residual). It is transformed from spatial domain to frequency domain by means of discrete cosine transform. Because human visual system is more sensitive to low frequency image and less sensitive to high frequency ones, quantization is applied such that more low frequency information is retained while more high frequency information discarded.

The third and final type of compression is entropy coding. A variable-length coding gives shorter codes to more probable symbols and longer codes to less probable ones such that the total bit count is minimized. After this phase, the output bit stream is ready for transmission or storage.

There is also a decoding path in the encoder. Because in the decoder side only the reconstructed frame instead of the original frame is available, we have to use a reconstructed frame as the reference for prediction. Therefore, in the bottom part of Fig. 1, we obtain the restored residual data by performing inverse quantization and then inverse transformation. Adding the restored residual to the predicted MB, we get the reconstructed MB that is then inserted to the reconstructed frame f'_t. Now, the reconstructed frame can be referred to by either the current I-type compression or future P-type or B-type prediction.

In the next section, we explain in more detail each of the video coding functions invoked in the pseudo code.

III. Basic Video Coding Functions

A. Prediction

Prediction exploits the spatial or the temporal redundancy of a video sequence so that only the difference between actual and predict instead of the whole image data need to be encoded. There are two types of prediction: intra prediction for I-type frame and inter prediction for P-type (Predictive) and B-type (Bidirectional Predictive) frame.

Intra Prediction -- There exists high similarity among neighboring blocks in a video frame. Consequently, a block can be predicted from its neighboring pixels of already coded and reconstructed blocks. The prediction is carried out by means of a set of mathematical functions.

In H.264/AVC, an I-type 16x16 4:2:0 MB has its luminance component (one 16x16) and chrominance components (two 8x8 blocks) separately predicted. There are many ways to predict a macroblock as illustrated in Fig. 2. The luminance component may be intra-predicted as one single INTRA16x16 block or 16 INTRA4x4 blocks. When using the INTRA4×4 case, each 4×4 block utilizes one of nine prediction modes (one DC prediction mode and eight directional prediction modes). When using the INTRA16×16 case, which is well suited for smooth image area, a uniform prediction is performed for the whole luminance component of a macroblock. Four prediction modes are defined. Each chrominance component is predicted as a single 8x8 block using one of four modes.

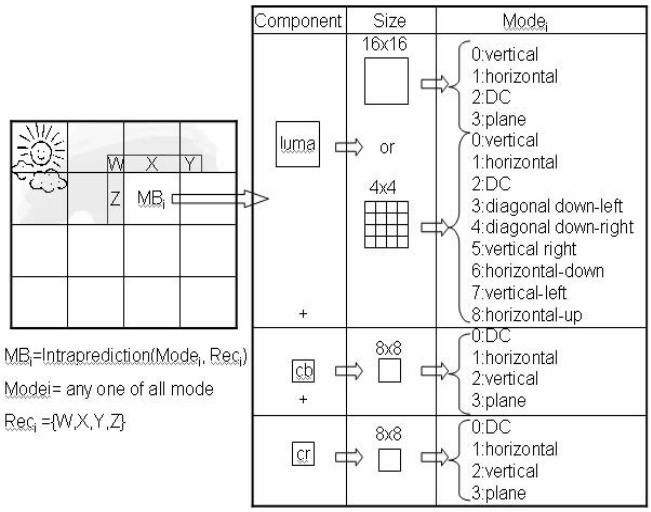
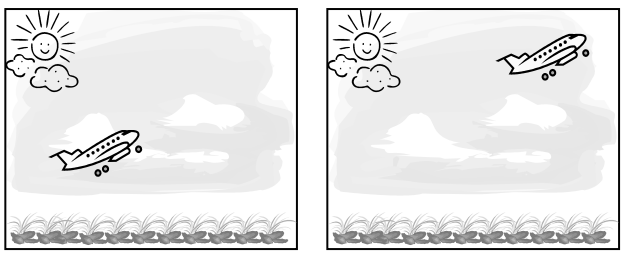


Fig. 2. Overview of H.264 intra prediction modes

Inter Prediction (Motion Estimation) -- High quality video sequences usually have high frame rate at 30 or 60 frames per second (fps). Therefore, two successive frames in a video sequence are very likely to be similar. The goal of inter prediction is to utilize this temporal redundancy to reduce data need to be encoded. In Fig. 3, for example, when encoding frame t, we only need to encode the difference between frame t-1 and frame t (i.e., the airplane) instead of the whole frame t. This is called motion estimated inter-frame prediction.



t-1 t

Fig. 3. Successive video frames

In most video coding standards, the block-based motion estimation (BME) [9] is used to estimate for movement of a rectangular block from the current frame. For each M x N-pixel current block in the current frame, BME compares it with some or all of possible M x N candidate blocks in the search area in the reference frame for the best match, as shown in Fig. 4. The reference frame may be a previous frame or a next frame in P-type coding, or both in B-type coding. A popular matching criterion is to measure the residual calculated by subtracting the current block from the candidate block, so that the candidate block that minimizes

the residual is chosen as the best match. The cost function is called sum of absolute difference (SAD), which is the sum of pixel by pixel absolute difference between predicted and actual image.

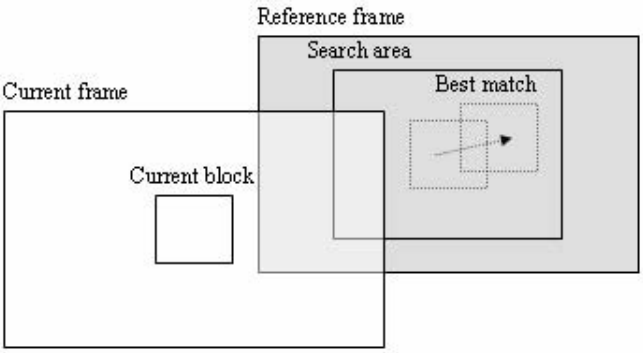
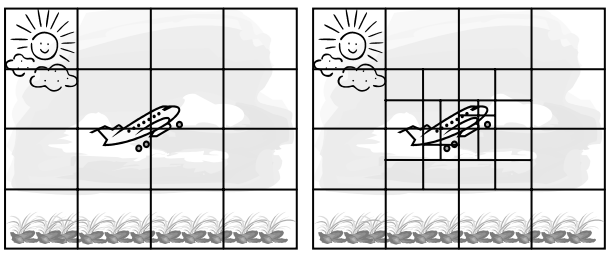


Fig. 4. Block-based motion estimation

There are three new features of motion estimation in H.264: variable block-size, multiple reference frames and quarter-pixel accuracy.

Variable block-size – Block size determines tradeoff between the residual error and the number of motion vectors transmitted. In the previous video coding standards, the block-size of motion estimation is fixed, such as 8 x 8 (MPEG-1, MPEG-2) or 16 x 16 (MPEG-4). Fixed block-size motion estimation (FBSME) spends the same efforts when estimating the motion of moving objects and background (no motion). This method causes low coding efficiency. In H.264, each macroblock (16 x16 pixels) may be split into sub-macroblocks in four ways: one 16 x 16 sub-macroblock, two 16 x 8 sub-macroblocks, two 8 x 16 sub-macroblocks, or four 8 x 8 sub-macroblocks. If the 8 x 8 mode is chosen, each of the four 8 x 8 sub-macroblocks may be split further in four ways: one 8 x 8 partition, two 8 x 4 partitions, two 4 x 8 partitions or four 4 x 4 partitions. Therefore, variable block-size motion estimation (VBSME) uses smaller block size for moving objects and larger block size for background, as shown in Fig. 5, to increase the video quality and the coding efficiency.



FBSME VBSME

Fig. 5. Comparisons between FBSME and VBSME

Multiple reference frames -- In previous video coding standards, there is only one reference frame for motion estimation. In H.264, the number of reference frames increases to 5, as shown in Fig. 6, for P frame and to 10 (5 previous frames and 5 next frames) for B frame [10]. More reference frames result in smaller residual data and, therefore, lower bit rate. Nevertheless, it requires more computation and more memory traffic.

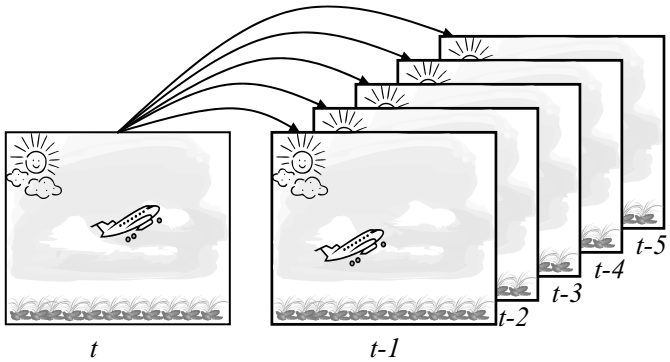


Fig. 6. Multiple reference frames for motion estimation

Quarter-pixel accuracy -- In previous video coding standards, motion vector accuracy is half-pixel at most. In H.264, motion vector accuracy is down to quarter-pixel and results in smaller residual data.

B. Compensation

Corresponding to prediction, there is also two kinds of compensation, intra compensation for I-type frame and inter compensation for P-type and B-type frame.

Intra Compensation – According to the encoding process, intra compensation regenerates the current block pixels by one of 13 modes (9 for Intra4x4 and 4 for Intra16x16) for luminance component and one of 4 modes for chrominance components.

Inter Compensation (Motion Compensation) -- Inter compensation is used in a decoding path to generate the inter-frame motion predicted (estimated) pixels by using motion vectors, reference index and reference pixel from inter prediction, as shown in Fig. 7. In H.264, inter compensation [11] also allows variable block-size, multiple reference frames and quarter-pixel accurate motion vector. Its luminance interpolation uses a 6-tap filter for half-pixel and a 2-tap filter for quarter pixel while the chrominance one uses neighboring four integer pixels to predict pixels up to accuracy of 1/8 pixel. It can refer to forward frames for P macroblocks and both forward and backward frames for B macroblocks. It allows arbitrary weighting factors for bidirectional weighted prediction.

C. Transformation and Quantization

The difference between the actual and predicted data is called residual error data. Discrete Cosine Transform (DCT) is a popular block-based transform for image and video compression. It transforms the residual data from time domain representation to frequency domain representation. Because most image and video are low frequency data,

DCT can centralize the coding information.

The main functionality of quantization is to scale down the transformed coefficients and to reduce the coding information. Because human visual system is less sensitive to high frequency image component, some video and image compression standards may use higher scaling-value (quantization parameter) for high frequency data.

The H.264 standard employs a 4x4 integer DCT [12]. Fig. 8 illustrates transformation and quantization in H.264 with an example.

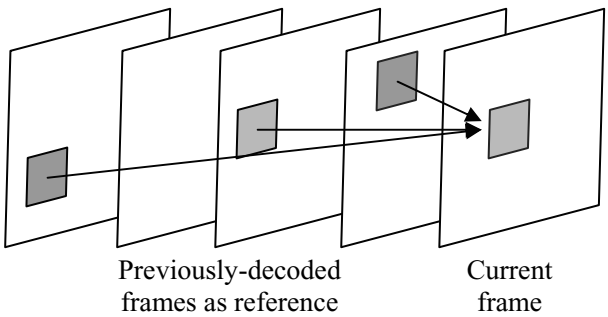


Fig. 7. Inter Compensation

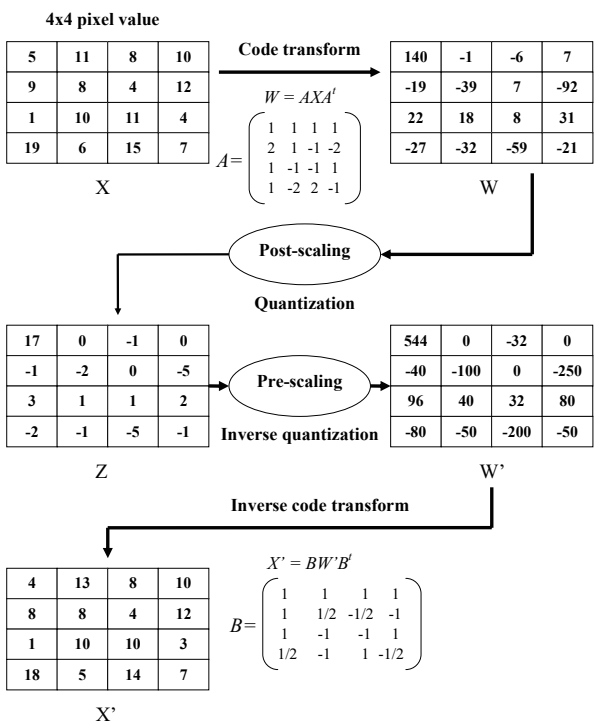


Fig. 8. Illustrating Transformation and Quantization

In Fig. 8, X is a 4x4 block of residual data. After integer DCT, we get W, a 4x4 coefficient matrix. Its upper left portion represents lower frequency components of X while its lower right portion gives higher frequency components. Z is the quantized version of W. We can see that the amount of data is much smaller than that of X, the original residual data. Z is the information to be entropy-coded and passed to the decoder part. W' is the scale-up (inversely quantized) version of Z. After applying inverse integer DCT (IDCT) on W', we get X', which is the decoded residual. Note that X' is not exactly identical to X. That is, this process is lossy due to the irreversibility of quantization.

D. In-loop filter

One of the disadvantages of block-based video coding is that discontinuity is likely to appear at the block edge. In order to reduce this effect, the H.264 standard employs the deblocking filter [13] to eliminate blocking artifact and thus generate a smooth picture.

In the encoder side, deblocking filter can reduce the difference between the reconstructed block and the original block. According to some experiments, it can not only improve PSNR, but also achieve up to 9% bit-rate saving. Fig. 9 depicts the input and output of deblocking filter.

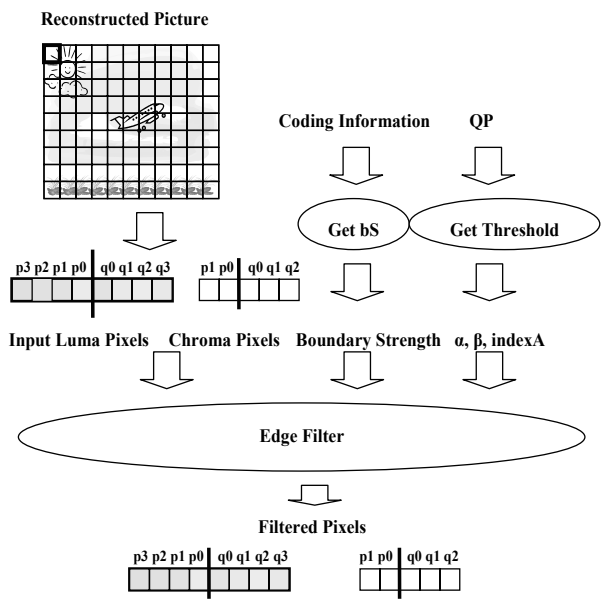


Fig. 9. Deblocking Filter Illustration

The deblocking filter works on one 16x16 MB at a time. It filters every boundary defined by 4x4 blocks within the MB. The deblocking filter consists of a horizontal filtering across all vertical edges and a vertical filtering across all horizontal edges. Therefore, for the luma component, it goes through 4 vertical boundaries and 4 horizontal boundaries with each boundary requiring 16 filtering operations. For both chroma components, it goes through 2 vertical boundaries and 2 horizontal boundaries with each boundary consisting of 8 filtering operations. As depicted in Fig. 9, inputs to a filtering operation includes eight luma pixels (p3,

p2, p1, p0, q0, q1, q2, q3) or five chroma pixels (p0, q0, q1, q2, q3), boundary strength, and threshold variables. At most six luma pixels (p2, p1, p0, q0, q1, q2) or two chroma pixels (q0, q1) will be modified by the filter. After the whole reconstructed frame is filtered, it is ready for display as well as being a reference picture.

The boundary strength (bS) is used to set the filter strength. As the boundary strength increasing, it eliminate more blocking artifact. The threshold variables are used to distinguish the true edge from the false edge.

E. Entropy Coding

The entropy encoder is responsible of converting the syntax elements (quantized coefficients and other information such as motion vectors, prediction modes, etc) to bit stream and then the entropy decoder can recover syntax elements from bit stream. Its function is similar to that of WinZip, which is commonly used for compressing files in the Windows Operating System.

There are two popular entropy coding methods, variable length coding and arithmetic coding. The former encodes symbol by looking up a Huffman table. Therefore, it must represent a symbol with one or more integer number of bits. On the other hand, arithmetic coding encodes a symbol by its appearance probability. So, it can represent a symbol with fractional number of bits and, thus, achieve higher compression efficiency than variable length coding does.

The H.264 standard defines two entropy coding methods: context adaptive variable length coding (CAVLC) and context based adaptive arithmetic coding (CABAC) [14]. For baseline profile, only CAVLC is employed. For main profile, both CAVLC and CABAC must be supported. According to our experiments, CABAC can achieve up to 7% bit rate saving at the expense of more computation complexity in comparison with CAVLC. Fig. 10 shows the coding flow of CABAC.

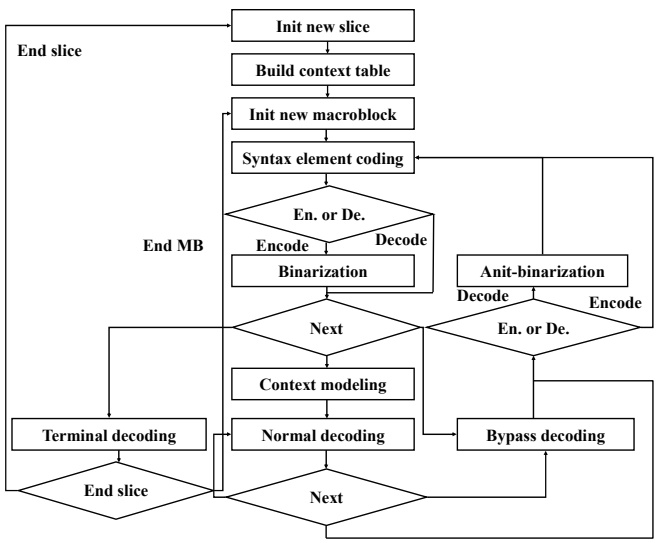
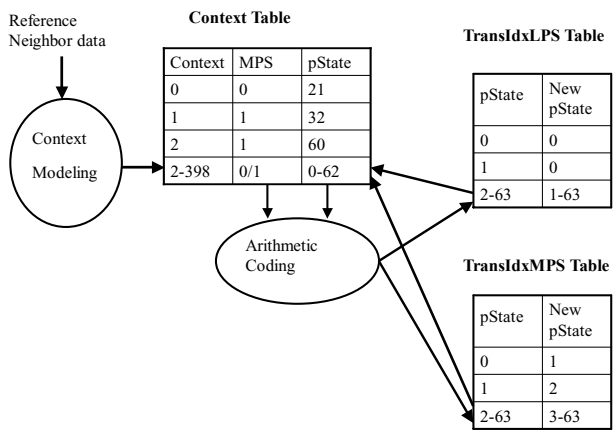


Fig. 9. The CABAC decoding flow

When the CABAC circuit processes a new slice, it first builds the context table before processing the first macroblock of the current slice. The basic information unit is called syntax element. For encoding, it will binarize these syntax elements before calculating the context value and then goes on to arithmetic coding. CABAC defines three arithmetic coding methods: normal decoding, bypass decoding and terminal decoding. After arithmetic coding, it proceeds to decode the next syntax element. For CABAC decoding, we have to convert the decoding result back to real syntax element value.

Most syntax elements go through the normal decoding process as shown in Fig. 11. Before decoding we get the context value through context modeling. Then, the decoder can look up the context table and get MPS value and pState. With these variable value, it goes to arithmetic coding. After arithmetic coding, it will update the context table by looking up TransIdxLPS table or TransIdxMPS table depending on whether the decoding result is equivalent to the MPS value. After entropy encoding, the bit stream is ready for output to a storage media or transmission medium.



Context	MPS	pState
0	0	21
1	1	32
2	1	60
2-398	0/1	0-62

pState	New pState
0	0
1	0
2-63	1-63

pState	New pState
0	1
1	2
2-63	3-63

Fig. 11. Normal Decoding Process in CABAC

IV. Summary and Conclusions

We have given a brief introduction to video coding with emphasis on H.264, the latest international video coding standards. Starting from MPEG-1 and H.261, most video coding standards follow the block-based hybrid coding approach. Great impact has been achieved with early generations of standards such as MPEG-1 for VCD and MPEG-2 for DVD and Digital TV.

Requirement for high quality applications drives continue development of the next generation standards. As shown in Table 1, compared with MPEG-4, H.264 calls for more sophisticated implementation of every part of the coding process. Fortunately, advancement in semiconductor manufacturing technology has made their low cost implementation possible.

Because H.264 was defined targeted towards a wide range of applications from low bit-rate low-resolution such mobile video conferencing to high-rate high-definition such as Ultra HDTV, there will not be a single implementation method that fits all. One can implement a baseline version for low end application with software running on an embedded microprocessor or a DSP core with possible video-specific instruction extension. For very high end application, hardwired acceleration of critical functions such as motion estimation/compensation and deblocking filter, or even the whole system, might be necessary. There is yet another approach called application-specific instruction set processor (ASIP) that defines custom instructions based on the function of video coding.

No matter which method is employed, there is yet another tradeoff between implementation completeness and coding efficiency. Quite often we see an implementation that trade quality for simplification of implementation. For example, a deblocking filter may treat all variable-size blocks as composition of 4x4 blocks or a simple bilinear interpolation is substituted for the standard 6-tap filter during sub-pixel motion estimation. This is not encouraged because it will lose the original spirit of H.264. That is, achieving big gain by accumulating small gain in every part of the coding process.

Acknowledgements

This work is supported in part by the National Science Council of Taiwan under grants no. NSC94-2215-E-007-029, NSC94-2220-E-007-007, NSC94-2220-E-007-009 and NSC94-2220-E-007-017, the Ministry of Economic Affairs of Taiwan under grant no. 94-EC-17-A-01-S1-038, and Taiwan Semiconductor Manufacturing Company under grant no. NTHU-0416. The authors would like to thank their colleagues, C. R. Chang, S. Y. Shih, H. C. Tzeng, C. L. Chiu, and Y. H. Chang, for collaborating in the NTHU video decoder project.

References

[1] MPEG (Moving Pictures Expert Group), Final Text for ISO/IEC11172, “Information technology − Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s” , ISO/IEC, 1993.

[2] CCITT Recommendation H.261, International Telecommunication Union, “Video Codec for Audiovisual Services at px64 kbit/s”, 1993

[3] “Draft ITU-T recommendation and final draft international standard of joint video specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC), “JVT G050, 2003.

[4] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, “Overview of the H.264/AVC video coding standard”, *IEEE Transactions on Circuit and System for Video Technology,* pp. 560-576, Jul. 2003.

[5] Iain E. G. Richardson, *H.264 and MPEG-4 Video Compression: Video Coding for Next-generation Multimedia,* John Wiley & Sons, 2003.

[6] Kobota, T.; “HD DVD-overview of next generation optical disc format”, IT to HD: Visions of Broadcasting in the 21st Century, The IEE 2-Day Seminar on (Ref. No. 2004/10760), pp. 213-224, 2004

[7] European Telecommunications Standards Institute (ETSI) ETS 300 744v1.1.2 (1997-08) : Digital Video Broadcasting (DVB) ; Framing Structure, Channel Coding and Modulation for Digital Terrestrial Television,1997

[8] Code of Federal Regulations. Title 47, Telecommunications, Parts 70-79 Revised as of Oct.1, 2003

[9] M. Flierl, T. Wiegand, and B. Girod, "A Locally Optimal Design Algorithm for Block-Based Multi-Hypothesis Motion-Compensated Prediction", *in Data Compression Conference,* pp: 239-248, 1998.

[10] T. Wiegand and B. Girod, “Multi-frame Motion-Compensated Prediction for Video Transmission”, Kluwer Academic Publishers, Sept. 2001.

[11] T. Wedi, "Motion Compensation in H.264/AVC", *in IEEE Transactions on Circuits and Systems for Video Technolog*y.

[12] H. Malvar, A. Hallapuro, M. Karczewicz, and L. Kerofsky, “Low-Complexity Transform and Quantization in H.264/AVC”, *in IEEE Transactions on Circuits and Systems for Video Technology*, pp: 598-603, Jul. 2003.

[13] D. Marpe, H. Schwarz, T. Wiegand, “Context-based adaptive binary arithmetic coding in the H.264/AVC video compression standard”, *IEEE Transactions on Circuits and Systems for Video Technology,* pp: 620-636, Jul. 2003.

[14] P. List, A. Joch, J. Lainema, G. Bjntegaard, and M. Karczewicz, “Adaptive deblocking filter,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, pp. 614-619, 2003.

Table 1. Comparison between MPEG-4 and H.264

Standard	MPEG-4	H.264
Block size	16*16 or 8*8	16*16 to 4*4
Transform	8*8 DCT	4*4 integer DCT
Entropy coding	VLC	VLC,CAVLC, CABAC
Ref frame	1 frame	Multiple (5) frames
Picture type	I, P, B	I, P, B, SI, SP
Coding efficiency	1	2
Decoder complexity	1	2.6
Target applications	Mobile devices	DTV, HD-DVD, Mobile devices