# PERFORMANCE ANALYSIS OF HARDWARE ORIENTED ALGORITHM MODIFICATIONS IN H.264

*Tu-Chih Wang, Yu-Wen Huang, Hung-Chi Fang and Liang-Gee Chen*

DSP/IC Design Lab,
Graduate Institute of Electronic Engineering,
Department of Electrical Engineering,
National Taiwan University

## ABSTRACT

H.264 [1] is initiated by ITU-T as H.26L and will become a joint standard of ITU-T and MPEG. The coding complexity of H.264 is much higher than MPEG-4 simple profile and advance simple profile algorithms. In order to achieve real-time encoding, hardware implementation is required.

The original test model of H.264 (JM) [2] is designed to achieve high coding performance. Some algorithms of the test model require lots of operations with little coding efficiency improvement. And some algorithms create data dependencies that prevent parallel hardware accelerations. This paper presents analysis of H.264 video coding algorithm in a hardware-oriented viewpoint. Intra prediction, hadamard transform and motion estimation algorithms are reviewed and modified to a hardware friendly configuration. The rate distortion penalties of these modifications are simulated and shown in this paper.

## 1. INTRODUCTION

H.264 is the next generation video coding standard that provides ultra high coding efficiency and network friendly functionalities. It has been a hot candidate for future's video streaming and communications. Although the coding performance of H.264 is good, more than four times of the algorithm complexity prevents its practical real-time implementation.

Several previous papers and documents[3-5] have addressed the coding complexity of this new state of art video coding algorithm. All of these papers use software-profiling techniques to retrieve the complexity of the H.264 algorithm. Although the profiling result is accurate for software implementation, the potential of parallel processing in hardware design can't be easily captured by this result.

In this paper, we reviewed the H.264 algorithm and look for the crucial points for hardware implementation. We modified the algorithm to enable efficient hardware implementation. Our algorithm modifications are all made in the prediction part, which occupies more than 90% of total computations. In addition, the hardware-oriented algorithm modifications are simulated through various video sequences to clarify the effects of the coding performance.

## 2. H.264 ALGORITHM

The block diagram of H.264 algorithm is shown in Fig 1. Video frames are captured into intra prediction and inter prediction parts. If the frame type is intra, the inter prediction part will be disabled. Multiple reference frames and variable block size motion estimation is used for inter prediction. The best mode among these prediction modes is chosen in the mode selection block. The input frame is then subtracted from the prediction and forms the residual blocks. The residual blocks are transformed by 4x4 integer DCT for luminance and 2x2 transform for chrominance DC coefficients. Scan and quantization procedures are then applied to the coefficients. The entropy coder receives these quantized coefficients and generates output codewords. The mode information is also transformed by the mode tables and fed into the entropy coder. The reconstruction loop includes the dequantization, inverse transform and deblocking filter. Finally, the reconstruct frame is written to the frame buffer for motion estimation.

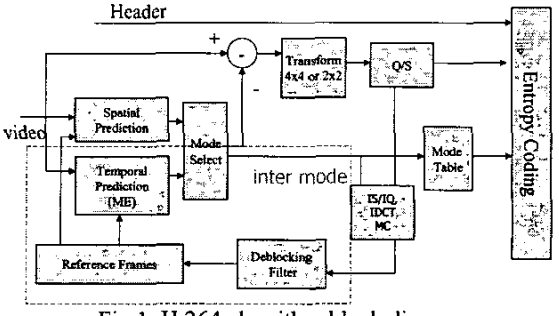


Fig 1. H.264 algorithm block diagram

0-7803-7663-3/03/$17.00 ©2003 IEEE



This paper was originally published in the Proceedings of the 2003 IEEE International Conference on Acoustics, Speech, & Signal Processing, April 6-10, 2003, Hong Kong (cancelled). Reprinted with permission.

The algorithm modifications applied to JM4.0d are mainly on the prediction parts (spatial prediction and temporal prediction) since the computations of these two prediction parts consume almost all the computation power. In the experiences of HW/SW partition, these two parts should be allocated as a hardware implementation and thus need to be modified to a hardware-oriented manner.

Our hardware model is a MB (macroblock) processing engine, which encodes MB by MB in a sequential manner. There may be MB pipeline in the encoding system, so the reconstructed process may not complete when the next MB's prediction process begins. Many commercial video encoders that uses MB engine can be fitted into this hardware model.

## 3. INTRA PREDICTION

The algorithm block diagram in Fig.1 is similar to previous video coding standards like H.261, H.263 and MPEG-4. The main difference of H.264 algorithm to other video coding standards is the intra prediction part, which consists nine 4×4 and four 16×16 intra prediction modes. Intra prediction requires reconstructed image pixels for prediction. In a typical MB engine, reconstructed data only can be obtained after coding. This data dependency results in difficulties for hardware implementation.

### 3.1. 4×4 intra prediction

The data dependency of 4×4 intra prediction mode is shown in Fig. 2. The pixels from a to p is predicted from A-N and Q. Pixels labeled in upper case are reconstructed pixels. Because there are 16 4×4 blocks in a MB, predictor can't get the reconstructed pixels when previous blocks are not coded. JM uses two-pass algorithm to code these blocks. It requires all the blocks passing the transform, quantization, dequantization and inverse transformation loop to do a 4×4 intra prediction, which is too complex for the hardware implementation.

We modified this algorithm as follows: we replace the entire reconstructed pixel by the original input pixels. With this modification, 4×4 intra prediction and transform can be processed in a pipelined manner without bubbles.

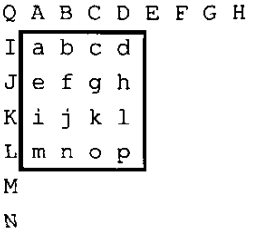


Fig.2 4×4 intra prediction

### 3.2. 16×16 intra prediction

Fig. 3 shows the data dependency of 16×16 intra prediction. The current MB is predicted by the 17 pixels from upper MBs and 16 pixels from the left MB. Since the reconstructed pixel of the left MB may not ready when the current MB's prediction process begins. The pixels of the left MB (light gray part of Fig. 3) are replaced by original pixels.

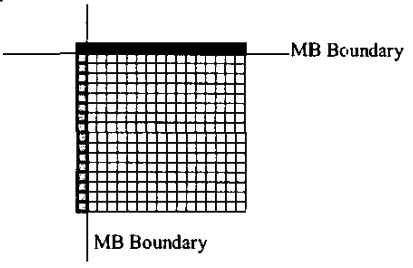


Fig.3 Data dependency of 16×16 intra prediction

### 3.3. Mode decision problem

The simple replacement of reconstructed pixels to original pixels may cause mode decision problem. Fig.4 show a R-D curve of the intra prediction modification. The simulation sequence is "Claire" at 10fps. The curve marked as "org_intra" is the coding performance of the intra prediction modification. The great degradation of PSNR is caused by errors from the mode decision. The original pixels are much correlated than reconstructed pixels since they belong to the same frame. The prediction error in the modified intra prediction will be much less than original version. In order to reduce the error rate of the mode decision, we modified the error cost function by adding an error term.

The error term represents the mismatch between original pixels and reconstructed pixels. Its value is related to QP because QP affects the mismatch between original pixels and reconstructed pixels. The quantization effect in H.264 increases exponentially while QP increases linearly. In order to match the quantization effect in H.264,we choose the error term in the form of $a/b^{(51-QP)}$, where a and b are parameters to be determined.

Theoretically, the parameter b should be set to 1.12 to match the increasing of quantization effect in H.264 because H.264 increases 12% quantization effect in each QP step. Because the cost function is calculated in the hadamard transform domain, each coefficient is scaled and not equal weighted. And the probability distribution of each transformed coefficient varies. In contrast with using the theoretical value, we use experiments to determine this parameter.

By experiments, we set 80 to a and 1.07 to b for 4×4 intra prediction. For 16×16 intra prediction, a is set to 400 and b is set to 1.07. This parameter set is test throughout various sequences and obtained good result in every

sequence. The R-D curve of this modification is shown as "mod_intra" curve in Fig.4. The mode decision errors are eliminated and the PSNR performance is almost the same as original JM.

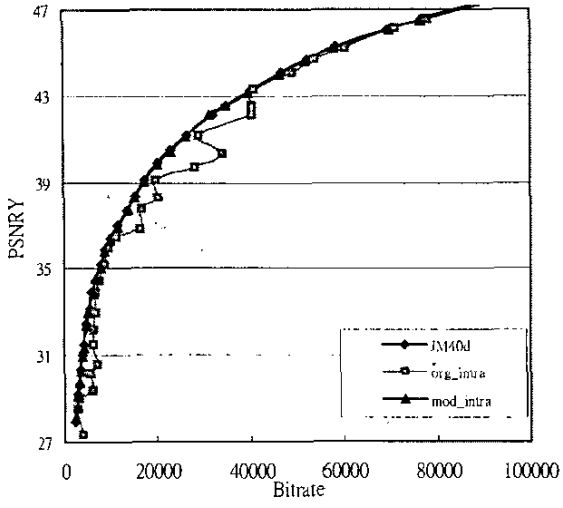


Fig. 4 R-D Curve of "Claire" sequence

## 4. MOTION ESTIMATION

H.264 uses variable block size, quarter pixel precision and multiple reference frames motion estimation. JM uses full search scheme with original searching point adjusted to motion predictor. At the integer precision stage, distortion is calculated using SAD. A compensated term for motion vector data will be added to the distortion to get better performance. Full search motion estimation is supported by various hardware architectures. But the choices of search range and motion predictor in JM are not practical for hardware design.

### 4.1. Search Range

Hardware motion estimator usually uses internal memory to reduce the requirement of external memory bandwidth. A typical search range data reuse scheme is shown in Fig.5. The search range is –16 - +15. The left 3×3 blocks in Fig.5 represent the search range of current MB's motion estimation process, while the right 3×3 blocks represent search range of the next MB's motion estimation process. The overlapped area is the data reused in this scheme.

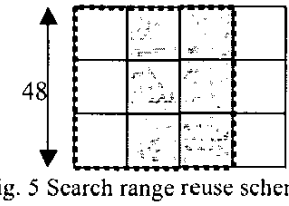


Fig. 5 Search range reuse scheme

In order to fit JM to this data reuse model, the search range original should be located at (0,0). This restriction results video quality degradation only if the true motion vector is out of the search range.

### 4.2. Motion Predictor

JM uses motion predictor to determine bits for the motion vector data and calculate coding penalty of the motion vector data. The penalty will be considered at all motion estimation stage to get better rate distortion performance.

Fig.6 shows the dependency of the motion predictor. P1 to P4 are previous coded MBs. The motion vectors from P1 to P4 are used for motion predictor calculation. The motion predictor calculation problem rises if there is a MB pipeline in the MB processing engine. The motion vector of P1 will be unavailable when motion estimation performs on current MB ( C in Fig.6).

In order to break the dependency in the motion predictor calculation, Only P2 to P4 are used in our modification algorithm. The final motion vector will be coded using predictor calculated from P1 to P4. The modification only affects motion estimation penalty calculation. Thus, our modification algorithm is still compatible to standard.

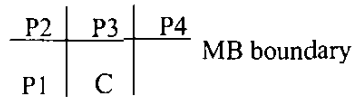


Fig. 6 Motion predictor data dependency

### 4.3. Quarter Pixel Precision Motion Estimation

In H.264, half pixel used for motion estimation is generated through a 2D 6-tap interpolation filter. 2D filter requires line buffer to perform transpose operations. The line buffer is large in hardware implementation. Motion compensation scheme requires this interpolation filter because the motion compensation is in the coding loop. But direct interpolation method can be used in motion compensation hardware because the motion vector is assigned.

One option to reduce hardware cost is using simpler method to generate quarter pixel precision data. The data used for motion estimation is not required to be the same with motion compensation, but some mismatch will occur and reduce the coding performance. We used bilinear interpolation for half pixel interpolation in this paper to evaluation this option.

### 4.4. Hadamard Transform

Hadamard transform performs simple transform to estimate bit generated after transform. It is used to replace SAD of prediction part. This additional option can be turned off if we want to design a low cost hardware.

## 5. SIMULATION RESULT

The software simulation is performed on the "Foreman", "grandma", "salesman", and "carphone" sequences. The frame rate is 10 frames per second. The reference frame number is set to one. And R-D optimization mode is turned off for hardware consideration. Because JM has no rate control method in version 4.0d, the rate distortion curve is generated by QP sweeping. The rate-distortion curve is shown in Fig7-8 and Table 1-2.

From the simulation results, we found that the PSNR degradation is small in our modified intra prediction. The PSNR reduces little in integer motion estimation modification with slow moving sequences. The QME modification results in 0.4 to 0.6 db PSNR degradation on average. This modification should be considered only in low cost system. From Table 1, the PSNR of each mode at 64kbps will decrease no more than 0.58 db.

Table 1. PSNR(Y) of algorithm modifications at 64kbps

| | Foreman | Grandma | Salesman | Carphone |
|---|---|---|---|---|
| JM4.0d | 34.68 | 41.14 | 39.61 | 35.81 |
| Intra_mod | 34.64 | 41.14 | 39.58 | 35.65 |
| IME_mod | 34.36 | 41.11 | 39.45 | 35.71 |
| IME+QME | 34.10 | 40.78 | 39.24 | 35.40 |
| Hadamard off | 34.36 | 40.95 | 39.28 | 35.55 |

Table 2. PSNR(Y) of algorithm modifications at 32kbps

| | Foreman | Grandma | Salesman | Carphone |
|---|---|---|---|---|
| JM4.0d | 31.28 | 37.91 | 35.55 | 32.40 |
| Intra_mod | 31.20 | 37.91 | 35.50 | 32.20 |
| IME_mod | 30.95 | 37.83 | 35.47 | 32.25 |
| QME_mod | 30.65 | 37.57 | 35.18 | 32.05 |
| Hadamard off | 30.94 | 37.71 | 35.22 | 32.10 |

## 6. CONCLUTION

In this paper, hardware oriented H.264 algorithms modifications are presented. Based on these modifications, parallel processing could be realized in MB based processing hardware. The impacts of these algorithm modifications are clarified through software simulations. Results show the video quality degradations of intra prediction and integer precision motion estimation modifications are nearly unnoticed. QME modification and turning off hadamard transform may be a solution for low cost encoder.


[1] Committee Draft of Joint Video Specification ( ITU-T Rec. H.264| ISO/IEC 14496-10 AVC) , July. 2002.

[2] Joint Video Team (JVT) software JM4.0d, August, 2002.

[3] Tu-Chih Wang, Hung-Chi Fang, Wei-Min Chao, Hong-Hui Chen and Liang-Gee Chen, "An UVLC encoder architecture for H.26L ", Proceeding of ISCAS, May, 2002.

[4] Minhua Zhou, "Benchmark Analysis of H.26L Decoder Functional Blocks", ITU-T, VCEG-N23, Sept., 2001.

[5] "Main Results of the AVC Complexity Analysis", ISO/IEC JTC1/SC29/WG11 N4964, July, 2002.


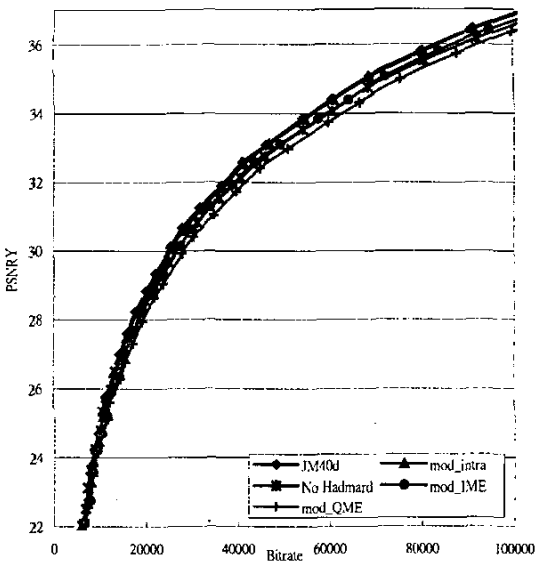


Fig. 7 Foreman QCIF 10fps

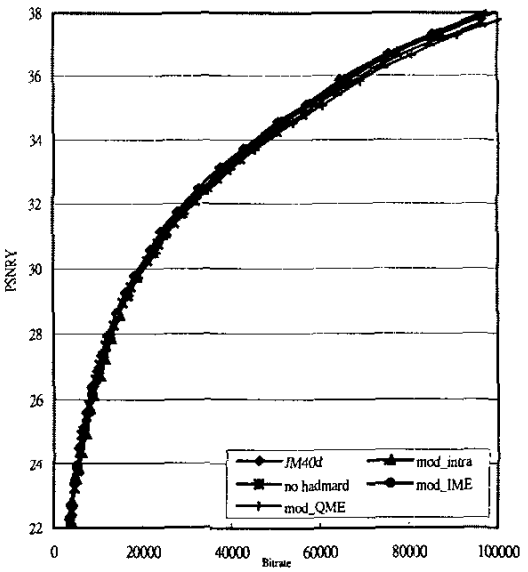


Fig. 8 Carphone QCIF 10fps