AVC-154

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION
ORGANIZATION INTERNATIONALE DE NORMALISATION
ISO/IEC JTC1/SC2/WG11
CODING OF MOVING PICTURES AND ASSOCIATED AUDIO

ISO/IEC JTC1/SC2/WG11
MPEG 91/217
November, 1991

Source : Takuyo Kogure (MATSUSHITA, Registration No.22)
Takanori Seno
Takeshi Yukitake
Akiyoshi Tanaka
Lucas Hui

Title : Matsushita Proposal Description for MPEG-II

Purpose : Proposal Package for MPEG-II

This document describes the complete Matsushita video compression algorithm targeting at coding of full motion color video sequences at up to about 10 Mbits/s data rate for the fulfillment of the ISO/WG11 MPEG Phase II Proposal Package Description MPEG 91/100.

This proposal is submitted by Matsushita with MPEG registration number 22. It is collaboration efforts of AV / Information Research Center of Asia Matsushita Electric (S) Pte. Ltd., Image Technology Research Lab. of Matsushita Electric Industrial Co., Ltd., Matsushita Research Institute Tokyo, Inc., and the Advanced Development Lab. of Matsushita Communication Industrial Co., Ltd.

CONTENTS

1. INTRODUCTION 1

2. DESCRIPTION OF THE PROPOSED ALGORITHM 1
- 2.1 Algorithm Overview 1
- 2.2 Encoder and Decoder 3
- 2.3 Input Picture Format 6
- 2.4 Macroblock Sampling 7
- 2.5 Motion Vector Detection 7
- 2.6 Motion Compensation 7
- 2.7 Interlace Detection 8
- 2.8 Luminance Block Sampling & Chrominance Block Subsampling 9
- 2.9 Edge Detection 11
- 2.10 DCT Coder 11
- 2.11 DPCM Coder 12
- 2.12 Run-Length Encoder (Zig-Zag Scan) 13
- 2.13 Variable Length Coder (VLC) 13
- 2.14 Rate Controller 14

3. DESCRIPTION OF THE BIT STREAM SYNTAX 15
- 3.1 Syntax Explanations 15
- 3.2 Video Sequence Layer 17
- 3.3 Group of Pictures Layer 18
- 3.4 Picture Layer 19
- 3.5 Slice Layer 20
- 3.6 Macroblock Layer 20
- 3.7 Block Layer 21

4. VARIABLE LENGTH CODE TABLES 22
- 4.1 Macroblock Addressing 22
- 4.2 Macroblock Type 24
- 4.3 Macroblock Pattern 25
- 4.4 Motion Vectors 26
- 4.5 Coded Coefficients 27

5. IMPLEMENTATION 39
- 5.1 Overview 39
- 5.2 System Block Diagram 39
- 5.3 Functional description of the encoder 39
- 5.4 Functional description of the decoder 42
- 5.5 Implementation Calculation for each module 43

6. CHARACTERISTICS & FEATURES 49
- 6.1 Random Access 49
- 6.2 Coding and Decoding Delay 49
- 6.3 Compatibility 51
- 6.4 Cell Loss Resilience 52
- 6.5 Fast Forward (FF) and Fast Reverse (FR) Play back 55

7. SIMULATION RESULTS 56
- 7.1 Signal-to-Noise Ratio and Bits for Each Frame 57
- 7.2 Cumulative Bit Count 64
- 7.3 Average Bit Count 66
- 7.4 Bit stream File Size 69
- 7.5 Comparison of Changes in Average SNR for different components 70

8. REFERENCES 72

A. APPENDIX A Motion Compensation Method for Interlaced Pictures 73

1. INTRODUCTION

With terms of reference for the ISO/MPEG phase II set at generic coding for moving picture at up to about 10 Mbits/s, MPEG phase II proposals have attracted many new applications which include digital storage and transmission of high quality video on various media (eg. digital VTR, ATM network, Satellite, Terrestrial Broadcast, Cable network, etc), and interactive communication systems (eg. Video conference, Videophone, Optical Disk System, etc). This document consists of an algorithm proposal by Matsushita for generic video coding targeting at the above mentioned bit-rate and applications.

The Matsushita proposal is basically a superset of the Simulation Model Three (SM3) [1] of MPEG Phase I, and the algorithm proposed is enhanced from SM3 with several objectives :

a) Superior picture quality suitable for broadcasting application with small increase in algorithm complexity;

b) Supporting of both interlace scanned and progressively scanned input / output video format with a single system;

c) Forward Compatibility with MPEG Phase I systems allowing no modification of hardware to decode MPEG-I bit stream;

d) Providing special switchable mode for supporting cell loss resilience requirement for ATM network.

The detailed design, characteristics, and implementation complexity of the Matsushita proposed algorithm are described. Some design considerations have been made in this document to handle channel errors or cell loss in ATM network.

2. DESCRIPTION OF THE PROPOSED ALGORITHM

2.1 Algorithm Overview

The basic of the Matsushita proposal is developed from the SM3 Algorithm which consists of the motion detection. motion compensation, DCT, quantization with rate-control, zig-zag scanning, and VLC modules. On top of the basic, the Matsushita proposal has four new areas of development : Interlace Coding Algorithm, Edge Enhancement Method, Bit Allocation Optimization, and Statistical Improvement.

Interlace Coding Algorithm

In SM3, the input picture sequence is pre-processed where each picture consists of only progressively scanned lines; however, since one of the requirements of MPEG phase II algorithm is be able to handle both interlace and progressive scanned input / output video, it is proposed that all input sequences (interlace or progressive) are to be coded frame-by-frame (frame coding). Present proposal consists of Interlace Coding algorithm to solve problems associated to interlaced frames coding. The interlace coding algorithm has two parts : Interlaced Macroblock Coding, and Interlaced Motion Compensation.

In Interlaced Macroblock Coding, the input frames are sampled into Macroblocks to be sequentially coded (refers to section 2.4). With objects contained in the Macroblock remained stationary, correlation between adjacent lines is high such that the Macroblock can be efficiently coded by a frame coding process. On the other hand, if the said objects are moving, the correlation between the two fields in the Macroblock will be low, and field coding process will be more efficient in this case. The proposed Interlace Macroblock

Coding method examines the correlation between the two fields in each macroblock, and subjects the macroblock to frame coding or field coding according to the examination result by sampling the luminance blocks and subsampling the chrominance blocks in the macroblock (section 2.7 & 2.8).

Interlace Motion Compensation method was designed to handle the time / motion differences between the two fields in a frame, or between fields of two frames. As for the Intra-coded frame (I-Frame), global horizontal motion compensation is applied to the second field reference to the first field in the frame to compensate any global panning motion which causes edge jerkiness in the frame and improve overall frame coding efficiency (section 2.5 & 2.6). For the nonintra-coded frames (P & B-Frames), two set of motion vectors are detected for each macroblock in each prediction direction. The first set consists of a motion vector estimating motion of the macroblock luminance pels, and the second set consists of two motion vectors each estimating motion of a luminance field (even/odd) in the macroblock. Comparison is made between the two sets of motion vector, and selection of one is done based on efficiency and estimation accuracy (section 2.5 & 2.6).

Edge Enhancement Method

Given the fact that edges in the sequence are more important visual information, and generally more difficult to code, a special coder, Adaptive DCT/DPCM Coder, has been designed for coding of such information to improve overall quality of the reconstructed sequence.

Basically, DCT converts statistically dependent pels into independent coefficients. The results are usually energy concentrated into only a few of the coefficients containing the main part of the picture information which can be quantized and run-length encoded. This concept is, however, difficult to apply when the transform pixel block is small in size and contains an edge boundary of an object (Edge Block). Transformation of such edge block will not lead to efficient compaction of signal energy, and ringing effects (corona effects) around the edges are usual to transform coding method.

The Adaptive DCT/DPCM Coder consists of an Edge Detector (section 2.9) which classifies each block to Edge/Non-Edge block type. The non-edge block is subjected to the DCT coder similar the SM3 coder (section 2.10), and the edge block is subjected to a DPCM coder (section 2.11) which performs better decorrelation and localization of the signal in time domain.

Bit Allocation Optimization

SM3 consists of a data compression rate controller that tries to limit the output bit-rate to a certain targeted number, example 1.15 Mbits/s. This rate controller allocates bits at the Macroblock Slice level based on the content of the system buffer and the type of frame it is coding. It however does not consider the distortion generated when too few bits are allocated or the redundancy when there are too many allocated bits. The Rate Controller (section 2.14) has been modified to provide better error distribution within every frame, and to have control of bits distribution among the I, P, and B-Frames.

Bit allocation is also controlled at the Macroblock level by the Quantizer (section 2.10) in the DCT coder. A method of calculating the Macroblock Quantizer step size (MQuant) is designed for the quantization of each Intra-coded Macroblock. As for the Nonintra-coded Macroblock, Conditional Replenishment (CR) method is used for better utilization of the bits.

Statistical Improvement

The run-level codes output from the Run-Length encoder (Zig-Zag Scanner) possess statistical characteristics distinguishable between the Intra-coded Macroblocks and the Nonintra-coded Macroblocks; therefore, it is more efficient to have separate Variable Length Coder (VLC) for encoding of the run-level codes from the Intra-coded and Nonintra-coded Macroblocks instead of a common VLC table as in SM3. In the present proposal, the VLC tables (section 2.13) for the Intra-coded and the Nonintra-coded Macroblocks were each optimized from statistics gathered from the two types of Macroblock.

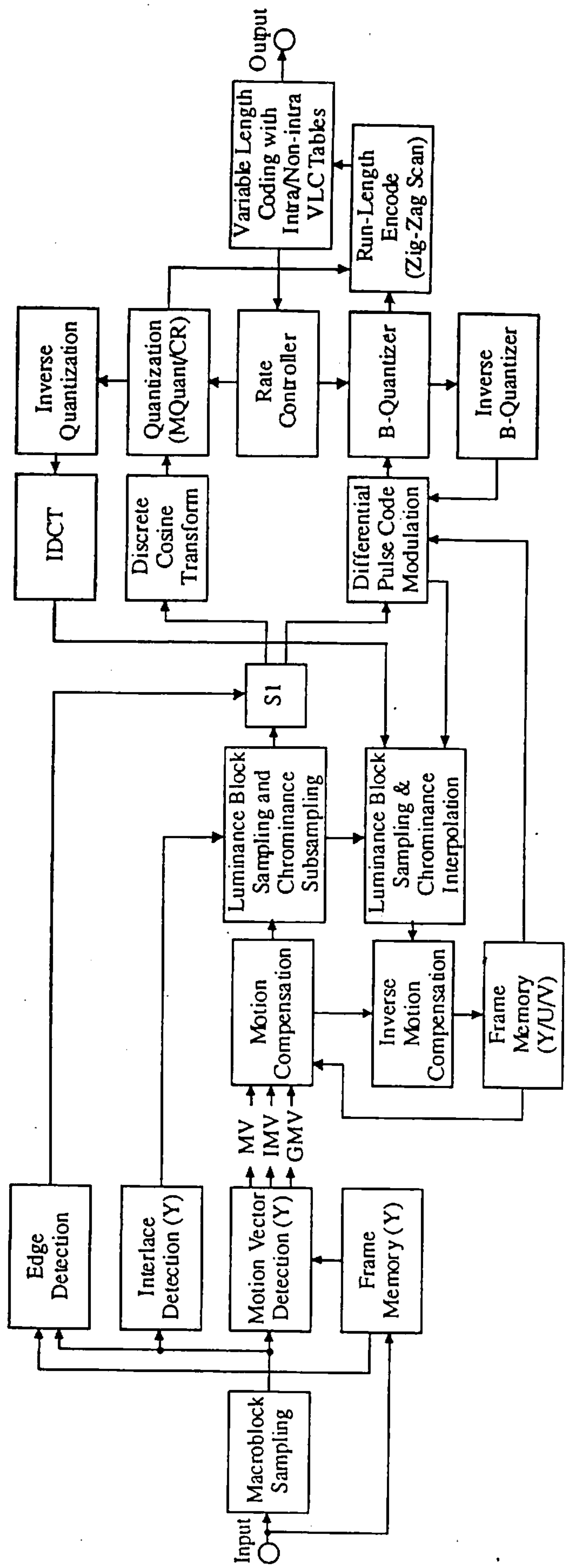
Summary

Items	Basic	Proposals
Picture Format	704x480x30 Y 352x480x30 C	Adaptive Chrominance Subsampling
Frame/Field	Frame Coding	Global MC on I-Frame
Frame Type Ratio	M=3, GOP=12	
Macroblock (MB)	4:1:1	Interlaced MB Coding
Motion Estimation	±15.5 Pel	Interlaced MV Coding
Encoder	MC + DCT	Adaptive DCT/DPCM Coder
Quantizer	SM3	MQuant for Intra MB, Conditional Replenishment for Non-intra MB
VLC	SM3	Intra/Non-intra Run Level VLC
Rate Control	SM3	Adaptive Controller

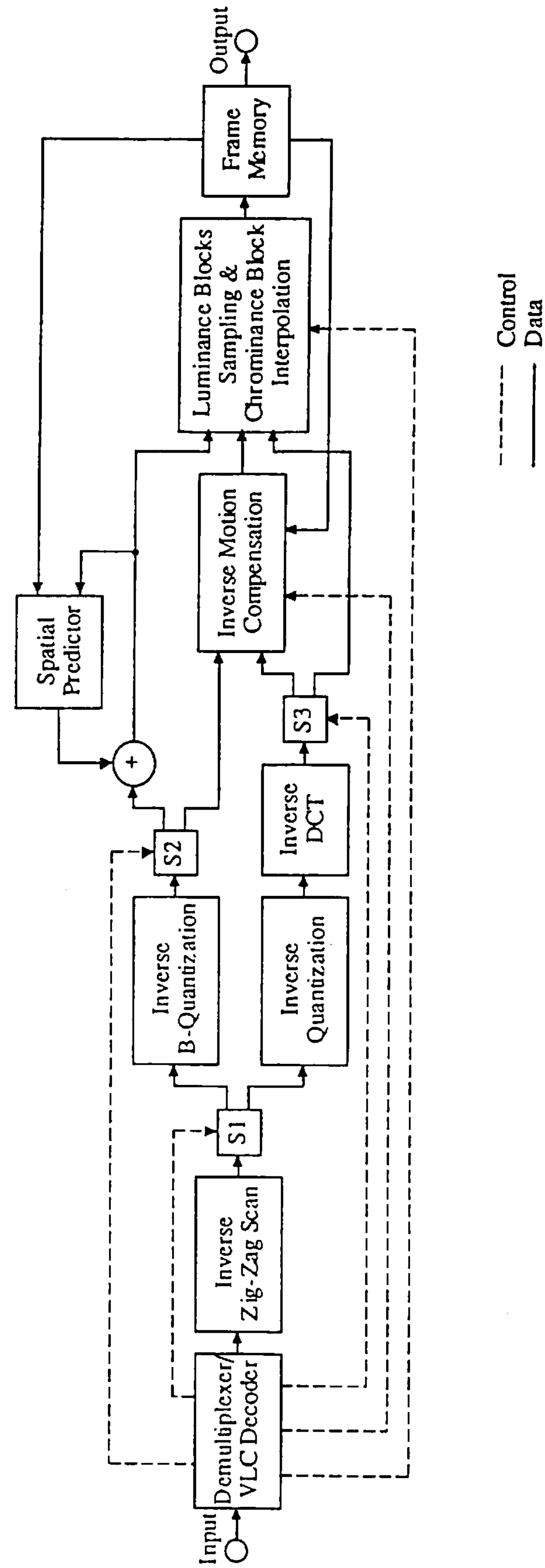
Table 2.1.

2.2 Encoder and Decoder

The overall designs of the Matsushita proposed Encoder and Decoder are illustrated in figure 2.2a and figure 2.2b respectively for demonstration of how each components are integrated together and also the flow of the processed data. Basically, the Encoder contains a local Decoder that performs the reconstruction of the encoded frames for future prediction purpose. The exact functions of each module in the block diagrams are described in the following sections (section 2.3 - 2.14).



Encoder Data Flow Diagram
Figure 2.2a.



Decoder Data Flow Diagram
Figure 2.2b.

2.3 Input Picture Format

The input picture format for simulation is CCIR Rec. 601 [2] with picture rate of 30 frames/second. The two fields (interlaced) of each frame are merged together to form the input frame sequence to the encoder. The coded area of each frame is 704x480 luminance pel area as shown in figure 2.3a.

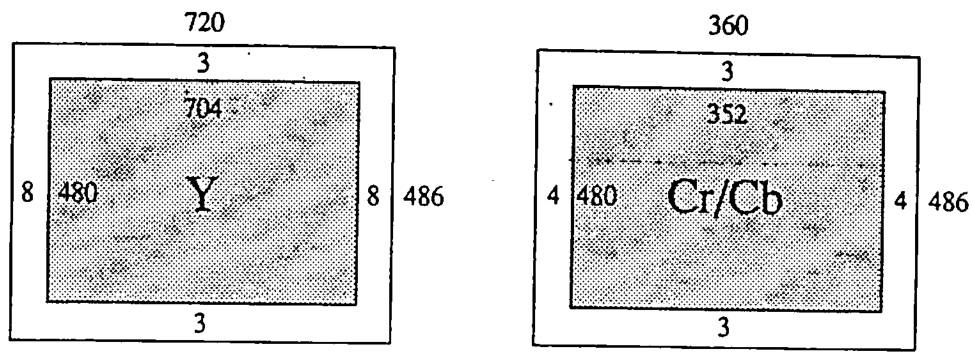


Figure 2.3a. Coded Pel Area

Input frame sequence is divided into groups of pictures (frames), or GOP, containing 12 consecutive frames with the first frame (I-Frame) of each GOP coded using intra coding method for random accessibility. Each third successive frame (P-Frame) starting from the first frame of the GOP is forward-predictive coded, and the rest of the frames (B-Frames) are bi-directional interpolative coded. Figure 2.3b is illustrating this GOP structure (GOP=12, M=3 as in SM3) and the direction of prediction.

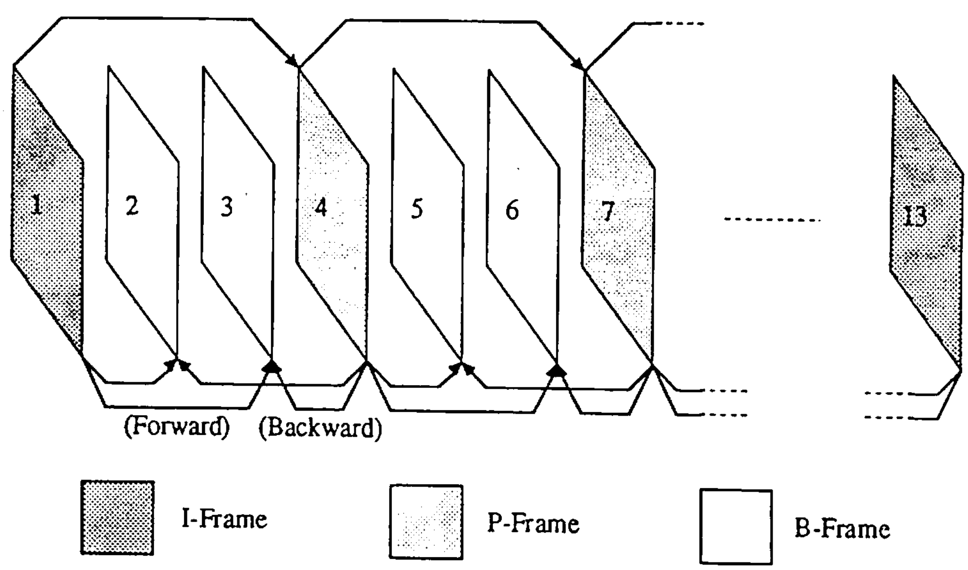


Figure 2.3b. Group of Pictures

2.4 Macroblock Sampling

Each frame in the input picture sequence is partitioned top-to bottom into Slices of Macroblocks where each Macroblock is coded sequentially left-to-right into bit-stream. The structure of the Macroblock is similar to SM3 structure with size of 16x16 luminance pel area and the co-sited 16x8 chrominance pels; hence, there are 30 Slices per frame and 44 Macroblocks per Slice. The sampling of the the Macroblock into luminance Blocks (8x8 pels) and also the vertical subsampling of the chrominance blocks are performed later in the encoder according to the results of the Interlace Detection Module.

2.5 Motion Vector Detection

Different types of Motion Vectors are detected for each of the frame type (I/P/B) in the Motion Vector Detection Module. For the I-Frame, a Global Motion Vector (GMV) which represents the global motion shifting of the second field in the I-Frame with reference to the first field in the same frame is to be detected. The GMV is determined by the following way :

Step 1. The motions between the second and the first field (reference) within the frame are estimated Macroblock by Macroblock in full pel precision in the horizontal direction and half pel precision in the vertical direction. In other words, the second field is fixed, while the reference frame is composed of the first field and its interpolation (in place of the second field). The motion vector of each Macroblock (second field only) is determined by the minimum absolute error for a search range of ±7 pel. The method of calculating the errors is always between the second and the first field, or between the second field and the interpolated (reference) second field.

Step 2. From the motion vectors obtained in step 1, the GMV is determined by taking the averages of the motion displacements which occupy more than a certain percentage (majority set at 40%) of all the motion vectors. This is done separately for the horizontal and vertical motion.

For the P-Frame, 3 motion vectors are estimated for each Macroblock using the same reference frame which is either previously coded I or P-frame: one for the 16x16 pel area (MV), and one each for the odd lines and the even lines of the 16x16 pel area (interlaced motion vectors or IMV). The motion search range is ±15.5 pels horizontally and vertically per frame distance, which is twice the field distance of ±7 pels. Telescopic search technique described in SM3 is used for reference frame distance of more than 1 frame. The search method is mean absolute error performed with full search at integer pel resolution followed by surround 1/2 pel resolution search using only luminance signals only. The interlaced motion vectors (IMV) are detected for better motion compensation in the case where fast motion (acceleration / deceleration) occurs in the sequence.

As for the B-Frame, the same 3 motion vectors are estimated for each Macroblock per each forward reference frame and backward reference frame (hence, 6 motion vectors estimated). The same search range and search method are used for P-Frames and B-Frames.

2.6 Motion Compensation

The objectives of this module is to decide the Macroblock motion compensation mode (Macroblock type), and perform the necessary motion compensation. For the I-Frame, the determined GMV from the Motion Vector Detection Module is used to shift the second field of the frame globally in the respective horizontal and vertical direction. Each Macroblock is re-sampled again from the shifted frame and coded intra.

In the case of Non-intra coded (P & B) frames, the first operation is to select between using one motion vector (MV) or two motion vectors (even and odd lines, IMV) for displacement estimation; this selection process is to be repeated for both prediction directions in the case of B-Frame. The selection process is based on a square error performance comparison method. For P-Frame, the second operation is to select between two modes : intra-coded, or predictive coded. This selection is base on the following process :

Step 1. obtain motion displaced Macroblock from reference frame using the selected MV or IMV;

Step 2. calculate the square error difference (Luminance only) between this obtained motion displaced Macroblock and the current Macroblock to be coded;

Step 3. calculate the Luminance AC energy (summation the square differences of each sample from the Macroblock mean sample value) of the current Macroblock to be coded;

Step 4. if the AC energy calculated in step3 is lower than the square error calculate in step 2, the current Macroblock will be coded using Intra-mode; else, the current Macroblock will be coded using predictive mode and the difference of the current Macroblock and the obtained motion displaced Macroblock will be coded and transmitted.

For B-frame, the second operation is to select one of the coding modes : forward predictive, backward predictive, bi-directional interpolative, or intra-coded. The selection process and coding process is basically the same as the above P-Frame process except that the square error differences will be calculated and compared for : forward motion displaced Macroblock, backward motion displaced Macroblock, and the forward/backward motion interpolated Macroblock. This coding mode selection process is the same as the one described in SM3 for B-frame.

2.7 Interlace Detection

The objective of this module is to detect fast motion occurred in the Macroblock causing the data in the two fields in the Macroblock to be highly decorrelated. The detected result will be used for Luminance block sampling and Chrominance block subsampling. Detection method used basically calculates the square error between the even and odd line pairs of the Macroblock luminance area, and compares the calculated result to the square error calculated between the consecutive odd line pairs and the consecutive even line pairs of the same area. The C function to do this is given as :

```
Interlace_Detection (Macroblock)

    int         Macroblock[BlockHeight][BlockWidth];{

    int         i, j;
    int         dif1, dif2, err1, err2, ratio;

    err1 = 0;
    for (i=0; i<BlockHeight; i+=2)
        for (j=0; j<BlockWidth; j++) {
            /*   Inter-Field Difference  */
            dif1 = Macroblock[i][j] - Macroblock[i+1][j];
            err1 += dif1*dif1;
            }
```

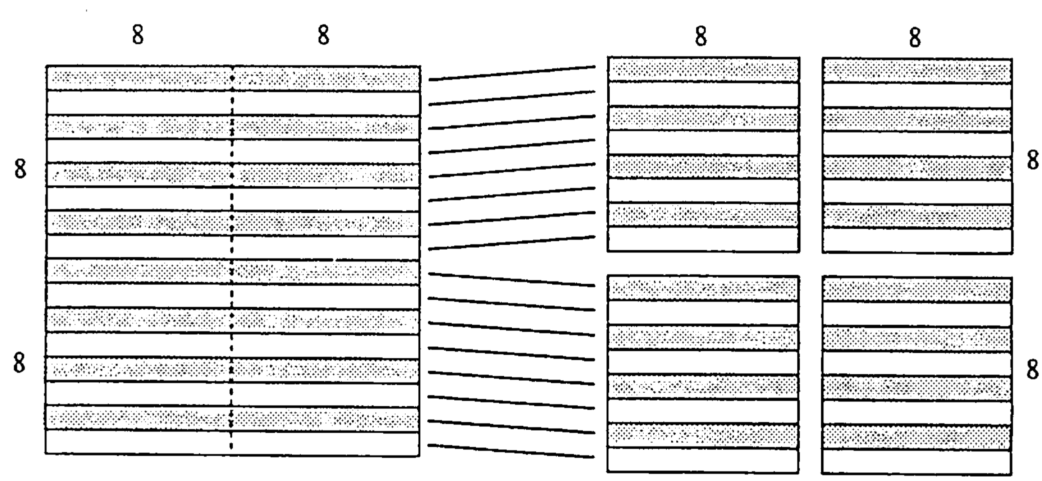
```
err2 = 0;
for (i=0; i<BlockHeight; i+=4)
     for (j=0; j<BlockWidth; j++) {
          /*    Intra-Field Difference */
          dif1 = Macroblock[ i ][j] - Macroblock[i+2][j];
          dif2 = Macroblock[i+1][j] - Macroblock[i+3][j];
           err2 += (dif1*dif1) + (dif2*dif2);
           }

ratio = (err1*10)/err2;
if((err1>100000)&&(ratio>12)) return (True);
else if((err1>8000)&&(ratio>18)) return (True);
else if (ratio>30) return (True);
else return (False);
}
```

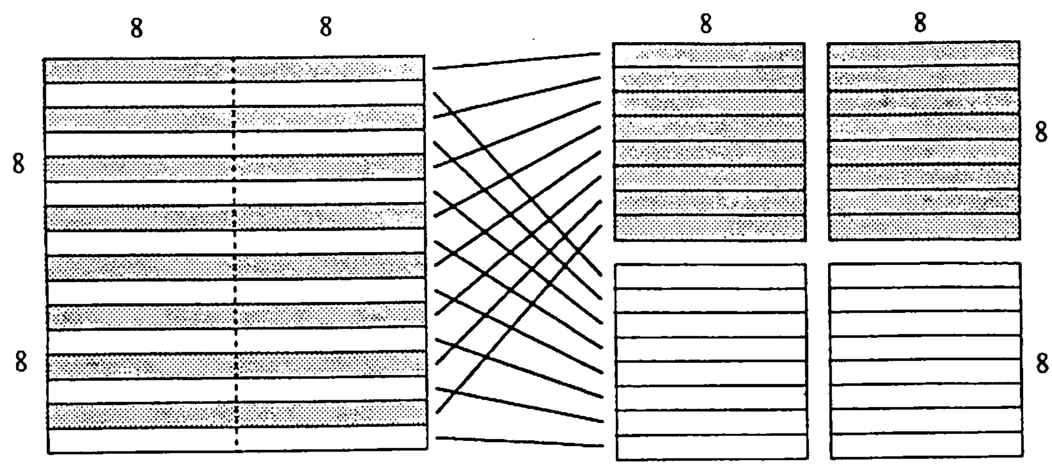
2.8 Luminance Block Sampling & Chrominance Block Subsampling

In the case where fast (horizontal) motion is detected in the Interlaced Detection process, the correlation between two fields in the Macroblock is low; hence, the two fields in the Macroblock are better coded independent of each other (Field Coding). This can be achieved by sampling of the Macroblock into four 8x8 blocks with the two blocks containing the even field samples and two other containing the odd field samples (see figure 2.8). Each set of blocks are coded independent of each other. On the other hand, if no motion is detected by the Interlace Detection process, the conventional way of partitioning the Macroblock into four smaller blocks (SM3) is used to take advantage of higher correlation between the lines in the two fields (Frame Coding).

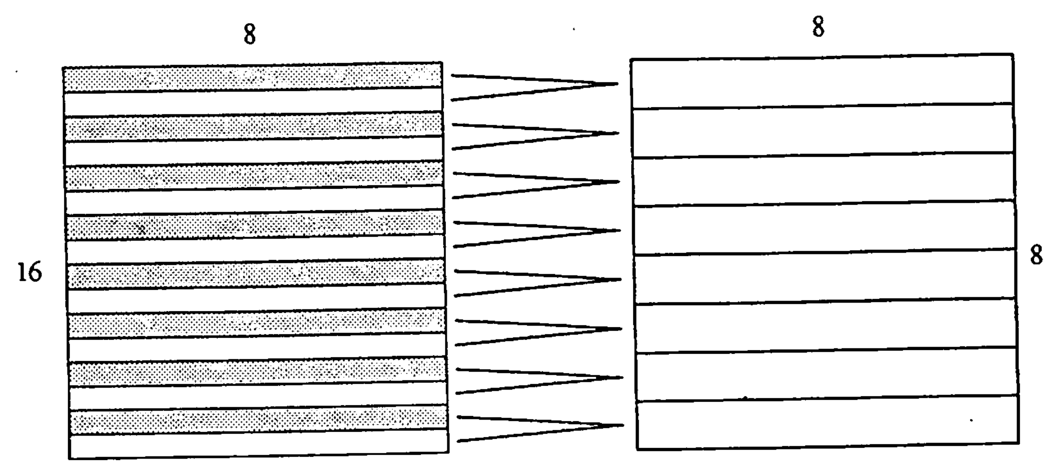
Vertical 2:1 subsampling of the Chrominance data is performed by averaging based on the result of the Interlace Detection process. Basically if inter-field correlation is high (no interlaced motion detected), intra-frame vertical subsampling of chrominance data is performed (Frame Coding); if inter-field correlation is low, intra-field vertical subsampling is performed (Field Coding). Following are diagrams illustrating the mentioned processes :



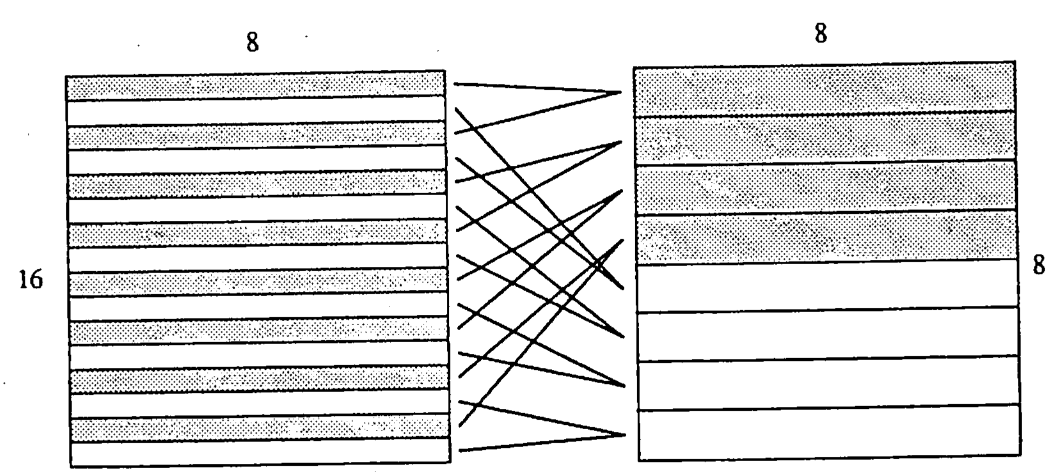
(a) Luminance Block Sampling - Frame Coding



(b) Luminance Block Sampling - Field Coding



(c) Chrominance Block Subsampling - Frame Coding



(d) Chrominance Block Subsampling - Field Coding

Figure 2.8.

2.9 Edge Detection

In the Edge Detection process, the amount of activity in each input Luminance 8x8 block is calculated and compared to a pre-defined threshold value *T1*. The sum of the square deviation of the block pixel intensity values from the mean block intensity value is used as representation of the block activity. If the calculated activity is greater than *T1*, then the activities of the surrounding blocks are compared to another threshold value *T2*. If any one of the activities of the surrounding block is lower than *T2*, the input block is considered as a boundary block between an object in the video sequence and a smooth background or region. In such a case, the input block is classified as an Edge Block.

For the Chrominance block, if any one of the Luminance block in the same Macroblock is detected as an Edge Block, and the calculated activity of the chrominance block is greater then a third threshold value *T3*, the chrominance block is considered as an Edge Block. The threshold values *T1*, *T2* and *T3* are set at 5300, 1000, and 8500 respectively for the simulation.

If the block is determined to be an Edge block, it is subjected (switch S1 in figure 2.2a) to the DPCM coding process which consists of the Spatial Predictor, the B-Quantizer, the Run-Length Encoder (Zig-Zag Scanner) and the Variable Length Coder. The block is subjected to a DCT coding process, if it is not an Edge block, which consists of the Discrete Cosine Transformation process, the Quantizer, the Zig-Zag Scanner, and the Variable Length Coder.

2.10 DCT Coder

In the DCT coder, the Discrete Cosine Transformation process and the Quantization process are defined exactly the same as SM3 Algorithm both in the case of Intra-coded Block, and Nonintra-coded Block.

The quantization stepsize is determined for every intra-coded Macroblock *(MQuant)*. In the MQuant calculation process, the average of the non-zero AC coefficients *(acAverage)* is obtained and compared with the quantization step size *(qSlice)* determined by the Rate-Controller for each Slice of Macroblocks :

```
if(acAverage<qSlice) {
  MQuant = (acAverage*3+qSlice+2) / 4;
  if((MQuant<3)&&(qSlice>=3)) MQuant = 3;
  }
else if(acAverage>qSlice) {
  MQuant = (acAverage*Weight1+qSlice*(10-Weight1)+5) / 10;
  if(MQuant>31) MQuant = 31;
  }
```

where *Weight1* equals to 5 for 4Mbps coding, and value 2 for 9 Mbps coding.

For every nonintra-coded Macroblock, Conditional Replenishment (CR) method is used for better utilization of bits. Basically, this method examines all the DCT coefficients of the transformed block before quantizing the block. If all coefficients are smaller then 1.5 times of the quantization step size used, all coefficients are quantized to zero; otherwise, the coefficients are quantized normally.

The Inverse Quantization process and the Inverse DCT process are also included in the DCT coder to perform local decoding of the Block to be placed into the reference Frame Memory for the case of I-Frame and P-Frame.

2.11 DPCM Coder

The DPCM Coder which processes Edge Blocks consists of the Spatial Predictor (in the DPCM process) and the B-Quantizer. An Adaptive Spatial Predictor is employed for the Intra-coded Blocks, and the standard predictor used by the motion compensation process is employed for the Nonintra-coded Block. The process of prediction and quantization are performed pixel by pixel sequentially. The Adaptive Spatial Predictor for the Intra-coded Block is given as :

```
for (i=0; i<BlockHeight; i++)
  for (j=0; j<BlockWidth; j++) {

    pattern = 0;
    Slope = ReconstructedBlock[i][j-1] - ReconstructedBlock[i][j-2];
    if (Slope>Threshold1) pattern += 8;
    Slope = ReconstructedBlock[i-1][j] - ReconstructedBlock[i-1][j-1];
    if (Slope>Threshold1) pattern += 4;
    Slope = ReconstructedBlock[i-1][j] - ReconstructedBlock[i-2][j];
    if (Slope>Threshold1) pattern += 2;
    Slope = ReconstructedBlock[i][j-1] - ReconstructedBlock[i-1][j-1];
    if (Slope>Threshold1) pattern += 1;

    xWeight = PredictionPattern[pattern][0];
    yWeight = PredictionPattern[pattern][1];

    prediction =
      (ReconstructedBlock[i][j-1]*xWeight+ReconstructedBlock[i-1][j]*yWeight) /
      (xWeight+yWeight);

    QuantizedBlock[i][j] = BQuantize (Block[i][j]-prediction);
    ReconstructedBlock[i][j] = InverseBQuantize (QuantizedBlock[i][j]);
    }
```

Note that reference to previous reconstructed neighboring Block is necessary for the prediction process. The value of *Threshold1* is set at 35 for Luminance Block, and at 15 for Chrominance Block. The value of PredictionPattern is given as :

```
int PredictionPattern[16][2] = {
   (1,1), (1,0), (1,1), (1,0), (1,1), (1,1), (1,1), (1,1),
   (1,1), (1,1), (1,1), (1,1), (0,1), (1,1), (1,1), (1,1)
  };
```

The Spatial Predictor for the Nonintra-coded Luminance and Chrominance Block is much simpler :

prediction = MotionDisplacedBlock[i][j];

where the *MotionDisplacedBlock* is either the forward predicted, backward predicted, or interpolated block obtained from reference frame(s).

As for the B-Quantizer, the equation for quantizing prediction error *(E)* of Intra-coded Block is given by :

$$quantizedValue = (int)\left(\frac{\sqrt{|E|}}{B} - 0.01\right) \times Sign\ (E); \qquad (1)$$

$$\text{where } Sign\,(E) = \begin{cases} 1 & E>0 \\ 0 & E=0 \\ -1 & E<0 \end{cases}$$

and for Nonintra-codedBlock is given by :

$$quantizedValue = (int)\left(\frac{\sqrt{4|E|+B^2}}{2B} - 0.5\right) \times Sign\,(E); \tag{2}$$

The value of *B* is determined by the calculated quantization step size for the slice *(qSlice)*, the Block type, and the *MQuant*. It is calculated by following steps :

qStep = qSlice;
if (MQuant<qStep) qStep = MQuant; / for Intra-coded Block Only */*
*B = qStep * (0.1875) + Boffset;*
if (B<1.7) B = 1.7;

The value of *Boffset* is set at 0.625 for Intra-coded Luminance and Chrominance Block, and 1.125 for Nonintra-coded Luminance and Chrominance Block.

The steps for performing the Inverse B-Quantization for Intra-coded Block are :

$$n = |quantizedValue|;$$
$$inverseQuantized = (int)(\,(n^2+n) \times B^2\,) \times Sign\,(quantizedValue); \tag{3}$$

and for Nonintra-coded Block are :

$$n = |quantizedValue|;$$
$$\text{if } (n==0)\ k = 0;\ \text{else } k = 1;$$
$$inverseQuantized = (int)(\,(n^2+n+\frac{n+k}{2}) * B^2\,) \times Sign\,(quantizedValue);. \tag{4}$$

2.12 Run-Length Encoder (Zig-Zag Scan)

The quantized block from either the DCT coder or the DPCM coder basically consists of many zero coefficients and few small non-zero coefficients which can be run-length encoded efficiently. The run-length encoding method and the Zig-Zag Scan Pattern used in SM3 method have been selected.

2.13 Variable Length Coder (VLC)

The Variable Length Coder does the job of huffman coding of the run-length encoded coefficients and all side information. All variable length codes followed the ISO/WG11 MPEG Paris Meeting (May 1991) version of the MPEG CD Syntax; however, in order to support the new Macroblock Type described in this Proposal, several new data elements were created and inserted into the bit stream syntax. Furthermore, to improve the coding efficiency of the run-level codes from the Run-Length Encoder, two set of VLC tables are used for the Intra-coded and the Nonintra-coded Macroblocks. The two set of VLC tables were optimized from statistics gathered from the two types of Macroblocks, and each contains unique ESC code and EOB code. The bit stream specification which includes the exact VLC/FLC tables used is described in section 3 "Description of the Bit Stream Syntax".

2.14 Rate Controller

The basic design of the Rate Controller which determines the quantization step size for the DCT and DPCM Coder is very similar to the SM3 Rate Controller. There are however some modifications. The objectives of the modifications are : (1) to provide better error distribution within every frame, and (2) to have control of bits distribution among the I, P, and B frames. The detailed operations in the Rate Controller can be described as :

```
/*
        F = Frame Type (I, P, or B);
        B = B1; (for First B-Frame)
        B = B2; (for Second B-Frame)

        Ratio_Control[I] = 10;
        Ratio_Control[P] = 8;
        if (Target_Rate==4 000 000) Ratio_Control[B] = 12;
        if (Target_Rate==9 000 000) Ratio_Control[B] = 10;

        Slice = (Total Number of Slices in a Frame);
        qMin = (Minimum Quantization Step Size for the Frame);
        Slice_Count = (Total Number of Bits Used to Code i-th Slice);
        qSlice = (Quantization Step Size Calculated for i-th Slice);

        Initial Guess For :  4Mbits/s    9Mbits/s
        Last_Count[I] =      462 000     690 000
        Last_Count[P] =      230 000     490 000
        Last_Count[B] =      56 000      180 000
*/

if ((F==I)&&(qMin[I]>qMin[P])) qMin[I] = qMin[P];
if (F==B) qMin[B] = qMin[P] * Ratio_Control[B] / Ratio_Control[P];
qmin = qMin[F];
qmax = 31;

Estimated_Rate = (Last_Count[I]*5 + Last_Count[P]*15 + Last_Count[B]*40)/2;
Slice_Estimate = Last_Count[F] * (Target_Rate/Estimated_Rate) / Slice;

for (qMean=0, i=0; i<Slice; i++) {    /*    For Each Slice    */

        qSlice = (Buffer_Count)//(1024*8) + 1;
        qSlice = qSlice * Ratio_Control[F] / 10;

        if (qSlice>qmax) qSlice = qmax;
        if (qSlice<qmin) qSlice = qmin;

        CODING_ONE_SLICE ();

        qMean += qSlice;
        Buffer_Count += Slice_Count - Slice_Estimate;
        if (Buffer_Count<0) Buffer_Count = 0;

        if ((Buffer_Count==0)&&(qmin>3)) qmin--;
        else if ((Buffer_Count>0)&&(qmin<qMin[F])) qmin++;
        }
qmin = (qMean+C)/Slice - 1;  /*    where C = (Slice-10)  */
if ((F==B2)&&(qmin<qMin[B2])) {
        qMin[P]--;
        if (qMin[P]<3) qMin[P] = 3;
        }
qMin[F] = qmin;
```

3. DESCRIPTION OF THE BIT STREAM SYNTAX

3.1 Syntax Explanations

The bit stream syntax of the present proposal is very similar to the ISO/WG11 MPEG Paris Meeting (May 1991) and Santa Clara Meeting (August 1991) version of the MPEG Video CD Syntax [3]; however, some modifications were made at the Macroblock Layer and the Block Layer to support the present proposed coding algorithm architecture. The method of describing the Bit Stream Syntax is the same method used in the MPEG Video CD, and most of the code tables (start codes, pel_aspect_ratio,, picture_rate, time_code, picture_coding_type, etc) remained unchanged. The syntax diagrams are included in this document (Section 3.2 ~ 3.7) for reference.

The several modifications to the bit stream syntax are :

Group of Pictures Layer :

two_vlc_tables -- Derived from group_extension_data (if any of the received "group_extension_data" byte == "0000 0001"). This parameter is used to indicate to the decoder that the two proposed VLC tables are to be used each to decode the Intra-coded Macroblock run-level coefficients and the Nonintra-coded Macroblock run-level coefficients. If this parameter is not set for the group of pictures, the default (SM3) VLC tables for the run-level coefficients are used.

MPEG2_extension -- Derived from the group_extension_data (if any of the received "group_extension_data" byte == "0000 0010"). This parameter is used to indicate to the decoder that the extended Macroblock Type for MPEG2 bit stream is used.

cell_loss_resilience_mode -- Derived from the group_extension_data (if any of the received "group_extension_data" byte == "0000 0011"). This parameter is used to indicate to the decoder that pictures are coded in cell loss resilience mode. In cell loss resilience mode, the proposed new variable length code table (table 4.1b) for absolute and relative Macroblock address (MBA) is to be used to decode the bit stream. If this parameter is not set, the pictures are coded in normal "non-cell-loss-resilience mode", and the default (SM3) MBA table (table 4.1a) is used.

Picture Layer :

global_motion_horizontal -- Derived from first received extra_information_picture byte (4 msb). This parameter is a integer value ranges ±7 (offset decoded value by -7) indicating the global horizontal motion of the second field in the Intra-coded Frame relative to the reference Intra Frame (first field + interpolated second field from the first). It is used to inverse shifting the decoded second field of the Intra-frame horizontally. This parameter is only valid for I-Frames.

global_motion_vertical -- Derived from the first received extra_information_picture byte (4 lsb). This parameter is a integer value ranges ±7 (offset decoded value by -7) indicating the global vertical motion of the second field in the Intra-coded Frame relative to the reference Intra Frame (first field + interpolated second field from the first). It is used to inverse shifting the decoded second field of the Intra-frame vertically. This parameter is only valid for I-Frames.

Macroblock Layer (refer to syntax diagram) :

interlaced_macroblock_type -- This is a one-bit flag following the macroblock_type code indicating the Luminance Block sampling and the Chrominance Block subsampling type. If it is set to "1", the field coding method is used for the Macroblock.

interlaced_motion_forward -- If macroblock_motion_forward is detected from the macroblock_type, this one-bit flag is present after the interlace_macroblock_type code to indicate whether or not the Interlaced Motion Vector (IMV) is used for forward motion prediction. If this flag is set to "1", two forward motion vectors (one for each field) will be received for the Macroblock. On the other hand, if this flag is "0", only one forward motion vector (VLC as in MPEG-I CD) is used to code the Macroblock.

interlaced_motion_backward -- If macroblock_motion_backward is detected from the macroblock_type, this one-bit flag is present to indicate whether or not the IMV is used for the backward motion prediction. If this flag is set to "1", two backward motion vectors (one for each field) will be received for the Macroblock.

adaptive_coded_type -- This is a 1-bit flag to indicate whether or not the Macroblock is coded adaptively by the DCT/DPCM coder. If this is set to "0", all blocks in the Macroblock are DCT coded.

adaptive_coded_block_pattern -- If the adaptive_coded_type code is "1", the 6-bits adaptive_coded_block_pattern is followed with each bit indicating the block position of the DPCM coded blocks in the Macroblock. (eg. first received bit is "1", the first block in the Macroblock is DPCM coded, and so on.)

motion_horizontal_forward_code_2, motion_horizontal_forward_r_2, motion_vertical_forward_code_2, & motion_vertical_forward_r_2 -- these codes are present if the interlaced_motion_forward is "1" to represent forward motion vector (2) for the second field. The specifications of these codes are same as motion_horizontal_forward_code, motion_horizontal_forward_r, motion_vertical_forward_code, and motion_vertical_forward_r respectively which are in this case represent the forward motion vector (1) of the first field. The forward motion vector 1 is delta coded with reference to the forward motion vector or forward motion vector 1 of the previous Macroblock (reset if previous Macroblock is skipped); and the forward motion vector 2 is delta coded with reference to the forward motion vector 1 of the same Macroblock.

motion_horizontal_backward_code_2, motion_horizontal_backward_r_2, motion_vertical_backward_code_2, & motion_vertical_backward_r_2 -- these codes are present if the interlaced_motion_backward is "1" to represent backward motion vector (2) for the second field. The specifications of these codes are same as motion_horizontal_backward_code, motion_horizontal_backward_r, motion_vertical_backward_code, and motion_vertical_backward_r respectively which are in this case represent the backward motion vector (1) of the first field. The backward motion vector 1 is delta coded with reference to the backward motion vector or backward motion vector 1 of the previous Macroblock (reset if previous Macroblock is skipped); and the backward motion vector 2 is delta coded with reference to the backward motion vector 1 of the same Macroblock.

Block Layer :

coded_coeff_first & coded_coeff_next -- These are variable length codes according to Table 4.5c-e in section 4.5. They are the same vlc obtained from MPEG-I CD dct_coeff_first and dct_coeff_next except that they are used to variable length code the run-level codes output from the DCT coder or the DPCM coder.

intra_coded_coeff -- A variable length code according to Table 4.5f-h in section 4.5. If two_vlc_tables code is set to "1" at the group of pictures layer, this code is used to represent the run-level output of the DCT/DPCM coder for the Intra-coded Macroblock.

nonintra_coded_coeff -- A variable length code according to Table 4.5i-l in section 4.5. If two_vlc_tables code is set to "1" at the group of pictures layer, this code is used to represent the run-level output of the DCT/DPCM coder for the Nonintra-coded Macroblock.

end_of_block -- In stead of the fixed code of "10", if two_vlc_tables code is set to "1" at the group of pictures layer, end_of_block code is defined differently for the Intra-coded and Nonintra-coded Macroblock as according to Table 4.5h and Table 4.5l. (Not that the escape code is modified for Intra-coded and Nonintra-coded Macroblock.)

3.2 Video Sequence Layer

START CODES

Name	Hexadecimal Value
picture_start_code	0000 0100
slice_start_codes (including slice_vertical_positions)	0000 0101 through 0000 01AF
reserved	0000 01B0
reserved	0000 01B1
user_data_start_code	0000 01B2
sequence_header_code	0000 01B3
sequence_error_code	0000 01B4
extension_start_code	0000 01B5
reserved	0000 01B6
sequence_end_code	0000 01B7
group_start_code	0000 01B8
system start codes	0000 01B9 through 0000 01FF

FUNCTION next_start_code

```
next_start_code () {
    while ( !bytealigned() )
        zero_bit                                            1       "0"
    while ( nextbit() != '0000 0000 0000 0000 0000 0001' )
        zero_byte                                           8
    "00000000"
    }
```

```
video_sequence() {
    next_start_code ()
    do {
        sequence_header ()
        do {
            group_of_pictures ()
            } while ( nextbits() == group_start_code )
        } while ( nextbits() == sequence_header_code )
    sequence_end_code                                       32 bits  bslbf
    }
```

Sequence Header

```
sequence_header () {
    sequence_header_code                                32 bits  bslbf
    horizontal_size                                     12       uimsbf
    vertical_size                                       12       uimsbf
    pel_aspect_ratio                                    4        uimsbf
    picture_rate                                        4        uimsbf
    bit_rate                                            18       uimsbf
    marker_bit                                          1        "1"
    vbv_buffer_size                                     10       uimsbf
    constrained_parameter_flag                          1
    load_intra_quantizer_matrix                         1
    if ( load_intra_quantizer_matrix )
        intra_quantizer_matrix[64]                      8*64     uimsbf
    load_non_intra_quatizer_matrix                      1
    if ( load_non_intra_quantizer_matrix )
        non_intra_quantizer_matrix[64]                  8*64     uimsbf
    next_start_code()
    if ( nextbits() == extension_start_code ) {
        extension_start_code                            32       bslbf
        while ( nextbits() != '0000 0000 0000 0000 0000 0001' ) {
            sequence_extension_data                     8
            }
        next_start_code ()
        }
    if ( nextbits() == user_data_start_code ) {
        user_data_start_code                            32       bslbf
        while ( nextbits() != '0000 0000 0000 0000 0000 0001' ) {
            user_data                                   8
            }
        next_start_code ()
        }
    }
```

3.3 Group of Pictures Layer

```
group_of_picture () {
    group_start_code                                    32 bits  bslbf
    time_code                                           25
    close_gop                                           1
    broken_link                                         1
    next_start_code ()
    if ( nextbits() == extension_start_code ) {
        extension_start_code                            32       bslbf
        while ( nextbits() != '0000 0000 0000 0000 0000 0001' ) {
            group_extension_data                        8
            }
        next_start_code ()
        }
    if ( nextbits() == user_data_start_code ) {
        user_data_start_code                            32       bslbf
        while ( nextbits() != '0000 0000 0000 0000 0000 0001' ) {
            user_data                                   8
            }
        next_start_code ()
        }
    do {
        picture ()
        } while ( nextbits() == picture_start_code )
    }
```

3.4 Picture Layer

```
picture () {
    picture_start_code                                          32 bits  bslbf
    temporal_reference                                          10       uimsbf
    picture_coding_type                                         3        uimsbf
    vbv_delay                                                   16       uimsbf
    if ( picture_coding_type==2 || picture_coding_type==3 ) {
        full_pel_forward_vector                                 1
        forward_f_code                                          3        uimsbf
    }
    if ( picture_coding_type == 3 ) {
        full_pel_backward_vector                                1
        backward_f_code                                         3        uimsbf
    }
    while ( nextbits() == '1' ) {
        extra_bit_picture                                       1        "1"
        extra_information_picture                               8
    }
    extra_bit_picture                                           1        "0"
    next_start_code ()

    if ( nextbits() == extension_start_code ) {
        extension_start_code                                    32       bslbf
        while ( nextbits() != '0000 0000 0000 0000 0000 0001' ) {
            picture_extension_data                              8
        }
        next_start_code ()
    }
    if ( nextbits() == user_data_start_code ) {
        user_data_start_code                                    32       bslbf
        while ( nextbits() != '0000 0000 0000 0000 0000 0001') {
            user_data                                           8
        }
        next_start_code ()
    }
    do {
        slice ()
    } while ( nextbits() == slice_start_code )
}
```


3.5 Slice Layer

```
slice () {
    slice_start_code                                            32 bits  bslbf
    quantizer_scale                                             5        uimsbf
    while ( nextbits() == '1' ) {
        extra_bit_slice                                         1        "1"
        extra_information_slice                                 8
    }
    extra_bit_slice                                             1        "0"
    do {
        macroblock ()
    } while ( nextbits() != '0000 0000 0000 0000 0000 000' )
    next_start_code ()
}
```

3.6 Macroblock Layer

```
macroblock () {
    while ( nextbits() == '0000 0001 111' )
        macroblock_stuffing                                     11 bits  vlclbf
    while ( nextbits() == '0000 0001 000' )
        macroblock_escape                                       11       vlclbf

    if (cell_loss_resilience_mode)
        macroblock_address                                      2-12     vlclbf
    else
        macroblock_address_increment                            1-11     vlclbf

    macroblock_type                                             1-6      vlclbf

    if ( MPEG2_extension ) {
        interlaced_macroblock_type                              1
        if ( macroblock_motion_forward )
            interlaced_motion_forward                           1
        if ( macroblock_motion_backward )
            interlaced_motion_backward                          1
        adaptive_coded_type                                     1
        if ( adaptive_coded_type == 1 )
            adaptive_coded_block_pattern                        6
    }

    if ( macroblock_quant )
        quantizer_scale                                         5        uimsbf

    if ( macroblock_motion_forward ) {
        motion_horizontal_forward_code                          1-11     vlclbf
        if (forward_f!=1 && motion_horizontal_forward_code!=0)
            motion_horizontal_forward_r                         1-6      uimsbf
        motion_vertical_forward_code                            1-11     vlclbf
        if (forward_f!=1 && motion_vertical_forward_code!=0)
            motion_vertical_forward_r                           1-6      uimsbf
        if ( interlaced_motion_forward ) {
            motion_horizontal_forward_code_2                    1-11     vlclbf
            if (forward_f!=1 && motion_horizontal_forward_code_2!=0)
                motion_horizontal_forward_r_2                   1-6      uimsbf
            motion_vertical_forward_code_2                      1-11     vlclbf
            if (forward_f!=1 && motion_vertical_forward_code_2!=0)
                motion_vertical_forward_r_2                     1-6      uimsbf
        }
    }
```

```
if ( macroblock_motion_backward ) {
    motion_horizontal_backward_code                              1-11   vlclbf
    if (backward_f!=1 && motion_horizontal_backward_code!=0)
        motion_horizontal_backward_r                             1-6    uimsbf
    motion_vertical_backward_code                                1-11   vlclbf
    if (backward_f!=1 && motion_vertical_backward_code!=0)
        motion_vertical_backward_r                               1-6    uimsbf
    if ( interlaced_motion_backward ) {
        motion_horizontal_backward_code_2                        1-11   vlclbf
        if (backward_f!=1 && motion_horizontal_backward_code_2!=0)
            motion_horizontal_backward_r_2                       1-6    uimsbf
        motion_vertical_backward_code_2                          1-11   vlclbf
        if (backward_f!=1 && motion_vertical_backward_code_2!=0)
            motion_vertical_backward_r_2                         1-6    uimsbf
    }
}
if ( macroblock_pattern )
    coded_block_pattern                                          3-9    vlclbf
for (i=0; i<6; i++)
    block (i)
if ( picture_coding_type==4 )
    end_of_macroblock                                            1      "1"
}
```

3.7 Block Layer

```
block (i) {
    if ( pattern_code[i] ) {
        if ( macroblock_intra && adaptive_coded_block_pattern[i]==0 ) {
            if ( i<4 ) {
                dct_dc_size_luminance                            2-7    vlclbf
                if (dct_dc_size_luminance!=0)
                        dct_dc_differential                      1-8    uimsbf
            }
            else {
                dct_dc_size_chrominance                          2-8    vlclbf
                if (dct_dc_size_chrominance!=0)
                        dct_dc_differential                      1-8    uimsbf
            }
        }
        else {
            if ( two_vlc_tables )
                nonintra_coded_coeff                             3-33   vlclbf
            else
                coded_coeff_first                                2-28   vlclbf
        }

        if ( picture_coding_type != 4 ) {
            while ( nextbits() != end_of_block ) {
                if ( macroblock_intra && two_vlc_tables )
                    intra_coded_coeff                            3-34   vlclbf
                else if ( two_vlc_tables )
                    nonintra_coded_coeff                         3-33   vlclbf
                else
                    coded_coeff_next                             3-28   vlclbf
            }
            end_of_block                                         2-4    vlclbf
        }
    }
}
```


4. VARIABLE LENGTH CODE TABLES

This chapter contains the variable length code tables for macroblock addressing, macroblock type, macroblock pattern, motion vectors, and DCT coefficients.

4.1 Macroblock Addressing

Two Macroblock Address (MBA) tables are defined : Table 4.1a is relative addressing table for non-cell-loss-resilience mode, and the other (Table 4.1b) is absolute and relative addressing table for cell-loss-resilience mode.

Table 4.1a. Variable length codes for macroblock_address_increment.

macroblock_address_increment VLC code	increment value	macroblock_address_increment VLC code	increment value
1	1	0000 0101 10	17
011	2	0000 0101 01	18
010	3	0000 0101 00	19
0011	4	0000 0100 11	20
0010	5	0000 0100 10	21
0001 1	6	0000 0100 011	22
0001 0	7	0000 0100 010	23
0000 111	8	0000 0100 001	24
0000 110	9	0000 0100 000	25
0000 1011	10	0000 0011 111	26
0000 1010	11	0000 0011 110	27
0000 1001	12	0000 0011 101	28
0000 1000	13	0000 0011 100	29
0000 0111	14	0000 0011 011	30
0000 0110	15	0000 0011 010	31
000 0101 11	16	0000 0011 001	32
		0000 0011 000	33
		0000 0001 111	macroblock_stuffing
		0000 0001 000	macroblock_escape

Table 4.1b Variable length codes for absolute and relative addressing

macroblock_address VLC code (relative)	increment value	macroblock_address VLC code (relative)	increment value
01	1	0000 0010 110	17
0011	2	0000 0010 101	18
0010	3	0000 0010 100	19
0001 1	4	0000 0010 011	20
0001 0	5	0000 0010 010	21
0000 11	6	0000 0010 0011	22
0000 10	7	0000 0010 0010	23
0000 0111	8	0000 0010 0001	24
0000 0110	9	0000 0010 0000	25
0000 0101 1	10	0000 0001 1111	26
0000 0101 0	11	0000 0001 1110	27
0000 0100 1	12	0000 0001 1101	28
0000 0100 0	13	0000 0001 1100	29
0000 0011 1	14	0000 0001 1011	30
0000 0011 0	15	0000 0001 1010	31
0000 0010 111	16	0000 0001 1001	32
		0000 0001 1000	33
		0000 0000 1111	macroblock_stuffing
		0000 0000 1000	macroblock_escape
macroblock_address VLC code(absolute)	**absolute address**	**macroblock_address VLC code(absolute)**	**absolute address**
1000 001	1	1010 001	17
1000 010	2	1010 010	18
1000 011	3	1010 011	19
1000 100	4	1010 100	20
1000 101	5	1010 101	21
1000 110	6	1010 110	22
1000 111	7	1010 111	23
1001 000	8	1011 000	24
1001 001	9	1011 001	25
1001 010	10	1011 010	26
1001 011	11	1011 011	27
1001 100	12	1011 100	28
1001 101	13	1011 101	29
1001 110	14	1011 110	30
1001 111	15	1011 111	31
1010 000	16	1100 000	32
		1100 001	33

4.2 Macroblock Type

Table 4.2a. Variable length codes for macroblock_type in intra-coded pictures (I-pictures).

VLC code	macroblock_quant
1	0
01	1

NOTE - macroblock_intra = 1, macroblock_pattern = 0, macroblock_motion_forward = 0, macroblock_motion_backward = 0.

Table 4.2b. Variable length codes for macroblock_type in predictive-coded pictures (P-pictures).

VLC code	macroblock_quant	macroblock_motion_forward	macroblock_pattern	macroblock_intra
1	0	1	1	0
01	0	0	1	0
001	0	1	0	0
00011	0	0	0	1
00010	1	1	1	0
00001	1	0	1	0
000001	1	0	0	1

NOTE - macroblock_motion_backward = 0.

Table 4.2c. Variable length codes for macroblock_type in bidirectionally predictive-coded pictures (B-pictures).

VLC code	macroblock_quant	macroblock_motion_forward	macroblock_motion_backward	macroblock_pattern	macroblock_intra
10	0	1	1	0	0
11	0	1	1	1	0
010	0	0	1	0	0
011	0	0	1	1	0
0010	0	1	0	0	0
0011	0	1	0	1	0
00011	0	0	0	0	1
00010	1	1	1	1	0
000011	1	1	0	1	0
000010	1	0	1	1	0
000001	1	0	0	0	1

4.3 Macroblock Pattern

Table 4.3. Variable length codes for coded_block_pattern

coded_block_pattern VLC code	cbp	coded_block_pattern VLC code	cbp
111	60	0001 1100	35
1101	4	0001 1011	13
1100	8	0001 1010	49
1011	16	0001 1001	21
1010	32	0001 1000	41
1001 1	12	0001 0111	14
1001 0	48	0001 0110	50
1000 1	20	0001 0101	22
1000 0	40	0001 0100	42
0111 1	28	0001 0011	15
0111 0	44	0001 0010	51
0110 1	52	0001 0001	23
0110 0	56	0001 0000	43
0101 1	1	0000 1111	25
0101 0	61	0000 1110	37
0100 1	2	0000 1101	26
0100 0	62	0000 1100	38
0011 11	24	0000 1011	29
0011 10	36	0000 1010	45
0011 01	3	0000 1001	53
0011 00	63	0000 1000	57
0010 111	5	0000 0111	30
0010 110	9	0000 0110	46
0010 101	17	0000 0101	54
0010 100	33	0000 0100	58
0010 011	6	0000 0011 1	31
0010 010	10	0000 0011 0	47
0010 001	18	0000 0010 1	55
0010 000	34	0000 0010 0	59
0001 1111	7	0000 0001 1	27
0001 1110	11	0000 0001 0	39
0001 1101	19		

4.4 Motion Vectors

Table 4.4. Variable length codes for motion_horizontal_forward_code, motion_vertical_forward_code, motion_horizontal_backward_code, and motion_vertical_backward_code. Same variable length codes are used for motion_horizontal_forward_code_2, motion_vertical_forward_code_2, motion_horizontal_backward_code_2, motion_vertical_backward_code_2.
Each motion vector is coded by this VLC and a FLC which length is based on the frame's forward_f_code or backward_f_code. The coding method for the motion vector is given by the MPEG-I CD [3].

motion VLC code	code
0000 0011 001	-16
0000 0011 011	-15
0000 0011 101	-14
0000 0011 111	-13
0000 0100 001	-12
0000 0100 011	-11
0000 0100 11	-10
0000 0101 01	-9
0000 0101 11	-8
0000 0111	-7
0000 1001	-6
0000 1011	-5
0000 111	-4
0001 1	-3
0011	-2
011	-1
1	0
010	1
0010	2
0001 0	3
0000 110	4
0000 1010	5
0000 1000	6
0000 0110	7
0000 0101 10	8
0000 0101 00	9
0000 0100 10	10
0000 0100 010	11
0000 0100 000	12
0000 0011 110	13
0000 0011 100	14
0000 0011 010	15
0000 0011 000	16

4.5 Coded Coefficients

Table 4.5a. Variable length codes for dct_dc_size_luminance.

VLC code	dct_dc_size_luminance
100	0
00	1
01	2
101	3
110	4
1110	5
11110	6
111110	7
1111110	8

Table 4.5b. Variable length codes for dct_dc_size_chrominance

VLC code	dct_dc_size_chrominance
00	0
01	1
10	2
110	3
1110	4
11110	5
111110	6
1111110	7
11111110	8

NOTE - In the following tables, the last bit 's' of the variable length code denotes the sign of the level : '0' for positive, '1' for negative.

Table 4.5c. SM3 variable length codes for coded_coeff_first and coded_coeff_next.

coded_coeff_first and coded_coeff_next VLC code	run	level
10	end_of_block	
1 s (NOTE1)	0	1
11 s (NOTE2)	0	1
011 s	1	1
0100 s	0	2
0101 s	2	1
0010 1 s	0	3
0011 1 s	3	1
0011 0 s	4	1
0001 10 s	1	2
0001 11 s	5	1
0001 01 s	6	1
0001 00 s	7	1
0000 110 s	0	4
0000 100 s	2	2
0000 111 s	8	1
0000 101 s	9	1
0000 01	escape	
0010 0110 s	0	5
0010 0001 s	0	6
0010 0101 s	1	3
0010 0100 s	3	2
0010 0111 s	10	1
0010 0011 s	11	1
0010 0010 s	12	1
0010 0000 s	13	1
0000 0010 10 s	0	7
0000 0011 00 s	1	4
0000 0010 11 s	2	3
0000 0011 11 s	4	2
0000 0010 01 s	5	2
0000 0011 10 s	14	1
0000 0011 01 s	15	1
0000 0010 00 s	16	1
0000 0001 1101 s	0	8
0000 0001 1000 s	0	9
0000 0001 0011 s	0	10
0000 0001 0000 s	0	11
0000 0001 1011 s	1	5
0000 0001 0100 s	2	4
0000 0001 1100 s	3	3
0000 0001 0010 s	4	3
0000 0001 1110 s	6	2
0000 0001 0101 s	7	2
0000 0001 0001 s	8	2

NOTE1 - This code shall be used for coded_coeff_first.
NOTE2 - This code shall be used for coded_coeff_next.

Table 4.5d. SM3 variable length codes for coded_coeff_first and coded_coeff_next (continued).

coded_coeff_first and coded_coeff_next VLC code	run	level
0000 0001 1111 s	17	1
0000 0001 1010 s	18	1
0000 0001 1001 s	19	1
0000 0001 0111 s	20	1
0000 0001 0110 s	21	1
0000 0000 1101 0 s	0	12
0000 0000 1100 1 s	0	13
0000 0000 1100 0 s	0	14
0000 0000 1011 1 s	0	15
0000 0000 1011 0 s	1	6
0000 0000 1010 1 s	1	7
0000 0000 1010 0 s	2	5
0000 0000 1001 1 s	3	4
0000 0000 1001 0 s	5	3
0000 0000 1000 1 s	9	2
0000 0000 1000 0 s	10	2
0000 0000 1111 1 s	22	1
0000 0000 1111 0 s	23	1
0000 0000 1110 1 s	24	1
0000 0000 1110 0 s	25	1
0000 0000 1101 1 s	26	1
0000 0000 0111 11 s	0	16
0000 0000 0111 10 s	0	17
0000 0000 0111 01 s	0	18
0000 0000 0111 00 s	0	19
0000 0000 0110 11 s	0	20
0000 0000 0110 10 s	0	21
0000 0000 0110 01 s	0	22
0000 0000 0110 00 s	0	23
0000 0000 0101 11 s	0	24
0000 0000 0101 10 s	0	25
0000 0000 0101 01 s	0	26
0000 0000 0101 00 s	0	27
0000 0000 0100 11 s	0	28
0000 0000 0100 10 s	0	29
0000 0000 0100 01 s	0	30
0000 0000 0100 00 s	0	31
0000 0000 0011 000 s	0	32
0000 0000 0010 111 s	0	33
0000 0000 0010 110 s	0	34
0000 0000 0010 101 s	0	35
0000 0000 0010 100 s	0	36
0000 0000 0010 011 s	0	37
0000 0000 0010 010 s	0	38
0000 0000 0010 001 s	0	39
0000 0000 0010 000 s	0	40

Table 4.5e. SM3 variable length codes for coded_coeff_first and coded_coeff_next (continued).

coded_coeff_first and coded_coeff_next VLC code	run	level
0000 0000 0011 111 s	1	8
0000 0000 0011 110 s	1	9
0000 0000 0011 101 s	1	10
0000 0000 0011 100 s	1	11
0000 0000 0011 011 s	1	12
0000 0000 0011 010 s	1	13
0000 0000 0011 001 s	1	14
0000 0000 0001 0011 s	1	15
0000 0000 0001 0010 s	1	16
0000 0000 0001 0001 s	1	17
0000 0000 0001 0000 s	1	18
0000 0000 0001 0100 s	6	3
0000 0000 0001 1010 s	11	2
0000 0000 0001 1001 s	12	2
0000 0000 0001 1000 s	13	2
0000 0000 0001 0111 s	14	2
0000 0000 0001 0110 s	15	2
0000 0000 0001 0101 s	16	2
0000 0000 0001 1111 s	27	1
0000 0000 0001 1110 s	28	1
0000 0000 0001 1101 s	29	1
0000 0000 0001 1100 s	30	1
0000 0000 0001 1011 s	31	1

Table 4.5f. New variable length codes for intra_coded_coeff of intra-coded macroblock.

intra_coded_coeff VLC code	run	level
11 s	0	1
011 s	1	1
0101 s	2	1
0001 s	3	1
0010 1 s	4	1
0000 1 s	5	1
0100 01 s	6	1
0010 01 s	7	1
1000 011 s	8	1
0100 001 s	9	1
0000 001 s	10	1
0100 0001 s	11	1
0000 0101 s	12	1
1000 0100 0 s	13	1
0000 0001 1 s	14	1
0010 0001 01 s	15	1
1000 0000 000 s	16	1
0100 0000 001 s	17	1
0010 0000 001 s	18	1
0010 0000 101 s	19	1
1000 1000 001 s	20	1
0010 0001 001 s	21	1
1000 0000 1011 s	22	1
0010 0000 0001 s	23	1
1000 1000 0100 1 s	24	1
0010 0000 0000 s	25	1
1000 1000 1000 1 s	26	1
1000 1000 0001 0 s	27	1
1000 0000 0100 1 s	28	1
0000 0100 0010 1 s	29	1
0000 0100 0001 0 s	30	1
0010 0000 1000 01 s	31	1
0000 0100 0001 1 s	32	1
1000 0000 1001 01 s	33	1
1000 0000 0100 01 s	34	1
0010 0001 0000 01 s	35	1
0000 0100 0100 00 s	36	1
0000 0100 0000 01 s	37	1
0000 0100 0000 0001 s	38	1
1000 0000 0101 001 s	39	1
0000 0100 0010 011 s	40	1
1000 1000 0100 001 s	41	1
0000 0100 0010 0100 s	42	1
0010 0001 0000 0010 s	43	1
1000 1000 1000 0000 s	45	1
0010 0001 0000 0011 s	47	1
0000 0100 0010 0101 s	48	1
1000 0000 0100 0001 s	50	1
0100 0000 0000 0001 s	52	1

Table 4.5g. New variable length codes for intra_coded_coeff of intra-coded macroblock (continued).

intra_coded_coeff VLC code	run	level
101 s	0	2
0100 1 s	1	2
1000 001 s	2	2
1000 0101 s	3	2
1000 1001 0 s	4	2
0100 0000 1 s	5	2
0000 0000 1 s	6	2
0010 0000 11 s	7	2
1000 1000 101 s	8	2
1000 1000 011 s	9	2
0010 0000 011 s	10	2
0010 0001 0001 s	11	2
1000 0000 0101 1 s	12	2
0010 0000 10001 s	13	2
1000 0000 1001 00 s	14	2
1000 1000 1000 001 s	15	2
1000 0000 1000 0001 s	16	2
1000 0000 1000 0000 s	20	2
1000 1000 0000 0110 s	21	2
1000 1000 0000 0111 s	23	2
0010 0000 1000 0011 s	27	2
1000 1000 0000 0101 s	28	2
0011 s	0	3
0010 001 s	1	3
0000 0100 1 s	2	3
0000 0000 1 s	3	3
0010 0000 010 s	4	3
0000 0100 011 s	5	3
0000 0100 0101 s	6	3
1000 1000 0000 1 s	7	3
0010 0000 1001 1 s	8	3
0010 0001 0000 1 s	9	3
1000 0000 0101 01 s	10	3
1000 0000 1000 0010 s	11	3
1000 1000 0000 000 s	12	3
1000 0000 1000 0011 s	13	3
0010 0001 0000 0000 s	14	3
1000 11 s	0	4
1000 0100 1 s	1	4
0000 0001 001 s	2	4
1000 0000 1001 1 s	3	4
1000 1000 0100 01 s	4	4
1000 0000 1010 0 s	5	4
0100 0000 0000 0010 s	6	4
0000 0100 0100 1001 s	7	4
0100 0000 0000 0011 s	8	4
0010 0000 1000 0000 s	9	4

Table 4.5h. New variable length codes for intra_coded_coeff of intra-coded macroblock (continued).

intra_coded_coeff VLC code	run	level
1000 101 s	0	5
1000 0000 11 s	1	5
1000 0000 1010 1 s	2	5
1000 0000 0100 001 s	3	5
1000 0000 0101 000 s	4	5
0000 0100 0100 101 s	5	5
0000 011 s	0	6
1000 0000 001 s	1	6
1000 1000 1000 011 s	2	6
1000 1000 1000 0001 s	3	6
0000 0100 0000 001 s	4	6
1000 1000 0000 0100 s	5	6
1000 0001 s	0	7
0000 0000 001 s	1	7
0000 0100 0010 0000 s	2	7
0000 0100 0100 1000 s	3	7
1000 0000 0100 0000 s	5	7
1000 1001 1 s	0	8
0100 0000 0001 s	1	8
0000 0100 0000 0000 s	3	8
0000 0100 0010 0001 s	5	8
0010 0001 1 s	0	9
0000 0000 0001 s	1	9
1000 1000 11 s	0	10
1000 0000 1000 1 s	1	10
0100 0000 01 s	0	11
0100 0000 0000 1 s	1	11
0000 0001 01 s	0	12
0000 0100 0000 10 s	1	12
1000 0000 011 s	0	13
0000 0100 0000 11 s	1	13
0000 0001 000 s	0	14
0000 0100 0100 01 s	1	14
1000 1000 1001 s	0	15
0100 0000 0000 01 s	1	15
0000 0100 0011 s	0	16
0000 0100 0100 11 s	1	16
1000 1000 0001 1 s	0	17
1000 1000 0100 0000 s	1	17
0010 0000 1001 0 s	0	18
1000 1000 0100 0001 s	1	18
1000 0000 1000 01 s	0	19
0010 0001 0000 0001 s	1	19
1000 1000 1000 010 s	0	20
0010 0000 1000 0001 s	1	20
1000 1000 0000 001 s	0	21
0000 0100 0010 001 s	0	22
0010 0000 1000 0010 s	0	23
1001	end_of_block	
1000 1000 0101	escape	

Table 4.5i. New variable length codes for nonintra_coded_coeff of non-intra-coded macroblock.

nonintra_coded_coeff VLC code	run	level
11 s	0	1
101 s	1	1
0101 s	2	1
1001 s	3	1
0011 0 s	4	1
0001 1 s	5	1
0100 1 s	6	1
0010 11 s	7	1
0010 01 s	8	1
0000 11 s	9	1
0001 01 s	10	1
1000 01 s	11	1
0010 101 s	12	1
0000 101 s	13	1
0000 011 s	14	1
1000 101 s	15	1
0000 1001 s	16	1
0001 0011 s	17	1
0100 0011 s	18	1
0001 0001 s	19	1
0100 0010 s	20	1
1000 0001 s	21	1
0010 0000 1 s	22	1
0010 0011 1 s	23	1
0000 0101 1 s	24	1
0000 0000 1 s	25	1
0000 0100 1 s	26	1
0000 0001 1 s	27	1
1000 0010 1 s	28	1
0010 0010 01 s	29	1
1000 0000 1 s	30	1
0000 1000 00 s	31	1
0010 0010 11 s	32	1
0000 0010 11 s	33	1
0000 1000 11 s	34	1
0000 0010 10 s	35	1
0001 0010 01 s	36	1
0001 0000 11 s	37	1
0010 1000 000 s	38	1
0000 0100 01 s	39	1
0010 0011 000 s	40	1
0100 0000 01 s	41	1
0000 0101 011 s	42	1
0010 1000 001 s	43	1
0001 0000 101 s	44	1
0010 1000 010 s	45	1
0000 1000 101 s	46	1
0010 0010 101 s	47	1
0000 0010 010 s	48	1
0010 0001 001 s	49	1
0010 0010 001 s	50	1

Table 4.5j. New variable length codes for nonintra_coded_coeff of non-intra-coded macroblock (continued).

nonintra_coded_coeff VLC code	run	level
0001 0010 001 s	51	1
0000 0101 000 s	52	1
0100 0000 0000 s	53	1
1000 0010 011 s	54	1
1000 0010 001 s	55	1
0010 0011 001 s	56	1
0010 0011 010 s	57	1
0001 0000 01 s	58	1
0000 0010 001 s	59	1
0010 0001 0001 0 s	60	1
1000 0010 010 s	61	1
0010 0011 011 s	62	1
0100 0000 0001 s	63	1
0011 1 s	0	2
1000 11 s	1	2
1000 100 s	2	2
0000 0011 s	3	2
0010 1000 1 s	4	2
0010 0001 1 s	5	2
0000 0001 0 s	6	2
0100 0000 1 s	7	2
0010 0001 01 s	8	2
0000 0000 01 s	9	2
0001 0000 00 s	10	2
0000 0010 011 s	11	2
0001 0000 100 s	12	2
0010 0010 1001 s	13	2
0000 0000 0001 s	14	2
1000 0000 0001 s	15	2
1000 0000 0000 1 s	16	2
0000 1000 1000 01 s	17	2
0001 0010 0001 11 s	18	2
0000 0101 0100 01 s	19	2
0000 1000 1000 11 s	20	2
1000 0010 0000 01 s	21	2
0001 0010 0000 101 s	22	2
0000 1000 1000 000 s	23	2
0000 0100 0000 001 s	24	2
0010 0001 0000 000 s	25	2
1000 0000 0000 001 s	26	2
0000 1000 1000 001 s	27	2
0000 0010 0000 001 s	28	2
1000 0000 0000 0001 s	30	2
0000 1000 1000 1000 s	32	2
0001 0010 0000 0000 s	33	2
0000 0101 0100 1001 s	34	2
0000 0010 0000 0000 s	35	2
1000 0010 0000 0010 s	36	2
0001 0010 0001 0001 s	43	2
0000 0101 0100 1000 s	61	2

Table 4.5k. New variable length codes for nonintra_coded_coeff of non-intra-coded macroblock (continued).

nonintra_coded_coeff VLC code	run	level
0100 01 s	0	3
0100 0001 s	1	3
0010 0000 00 s	2	3
0010 1000 011 s	3	3
0000 0100 001 s	4	3
1000 0000 001 s	5	3
0000 1000 1001 s	6	3
0000 0100 0001 s	7	3
0000 0010 0001 1 s	8	3
0000 0010 0000 1 s	9	3
1000 0010 0000 1 s	10	3
0001 0010 0000 01 s	11	3
1000 0010 0001 01 s	12	3
0000 1000 1000 101 s	13	3
0000 0010 0000 0100 s	14	3
1000 0010 0001 0000 s	15	3
0010 1001 s	0	4
0010 0000 01 s	1	4
0010 0010 0000 s	2	4
0010 0010 1000 1 s	3	4
1000 0010 0001 1 s	4	4
0010 0010 1000 00 s	5	4
0001 0010 0001 10 s	6	4
1000 0000 0000 01 s	7	4
1000 0010 0001 001 s	8	4
0001 0010 0001 001 s	9	4
1000 0010 0000 0000 s	10	4
0001 0010 0001 0000 s	11	4
1000 0011 s	0	5
0000 0000 001 s	1	5
0000 0010 0001 0 s	2	5
0001 0010 0001 01 s	3	5
0000 0101 0100 001 s	4	5
0000 0100 0000 000 s	5	5
0001 0010 0000 0001 s	6	5
1000 0010 0000 0001 s	7	5

Table 4.5l. New variable length codes for nonintra_coded_coeff of non-intra-coded macroblock (continued).

nonintra_coded_coeff VLC code	run	level
0001 0010 1 s	0	6
0000 0101 0101 1 s	1	6
0000 0100 0000 01 s	2	6
0000 1000 1000 1001 s	3	6
0000 1000 01 s	0	7
0010 0001 0000 1 s	1	7
0000 0101 0100 101 s	2	7
1000 0010 0000 0011 s	3	7
1000 0000 01 s	0	8
0010 0010 1000 01 s	1	8
0000 0010 0000 0001 s	2	8
0000 0101 001 s	0	9
0001 0010 0000 11 s	1	9
0010 0010 0001 s	0	10
0000 0010 0000 011 s	1	10
0000 0101 0101 0 s	0	11
0000 0010 0000 0101 s	1	11
0010 0001 0001 1 s	0	12
0001 0010 0000 1000 s	1	12
0000 0100 0000 1 s	0	13
0010 0001 0000 01 s	0	14
0000 0101 0100 11 s	0	15
0010 0001 0000 001 s	0	16
0000 0101 0100 000 s	0	17
0001 0010 0000 001 s	0	18
0001 0010 0000 1001 s	0	19
1000 0010 0001 0001 s	0	20
011	end_of_block	
0100 0000 001	escape	

Table 4.5m. Encoding of run and level following escape code as a **14-bit fixed** [illegible] 127<=level<=127) or as 22-bit fixed length code (-255<=level<=-128, [illegible] 128<=level<=255).

fixed length code	run
0000 00	0
0000 01	1
0000 10	2
...	...
...	...
...	...
...	...
...	...
...	...
1111 11	63

fixed length code	level
forbidden	-256
1000 0000 0000 0001	-255
1000 0000 0000 0010	-254
...	...
1000 0000 0111 1111	-129
1000 0000 1000 0000	-128
1000 0001	-127
1000 0010	-126
...	...
1111 1110	-2
1111 1111	-1
forbidden	0
0000 0001	1
...	...
0111 1111	127
0000 0000 1000 0000	128
0000 0000 1000 0001	129
...	...
0000 0000 1111 1111	255

5. IMPLEMENTATION

5.1 Overview

The main parts of the encoder and the decoder are the same as MPEG I system, because the Matsushita proposal is basically a superset of MPEG-I: now even some extensions to the main blocks and additional modules are necessary for achieving the proposed algorithm. These extended functions and modules in the encoder can easily be switched from MPEG-I to MPEG-II and vice versa. In the decoder, MPEG-I bitstream can be decoded as well as MPEG-II.

5.2 System Block Diagram

Figure 5.2a and 5.2b show the encoder system block diagram and decoder system block diagram. The encoder consists of an A/D, a Frame memory, a buffer, a Motion detection module and an encoder LSI. The decoder consists of a buffer, a Frame memory, and a D/A. The sizes of the frame memory and coded buffer for the encoder and the decoder are shown in table 5.2.

	Frame memory		Coded data buffer size bit rate	
	number	size	4Mbps	9Mbps
Encoder	6 frame	4.15Mbyte	630Kbit	790Kbit
Decoder	3 frame	2.03Mbyte	630Kbit	790Kbit

Table 5.2 Size of Frame memory and Coded data buffer

The block diagram of the encoder LSI, that is the main part of the encoder, is shown in Figure 5.2c. Comparing with MPEG-I, some modules are extended, these are the quantizer for DCT, VLC and Motion compensation. Besides these, some new modules are added. These are Interlace detection, Edge detection, Block sampling, Block reconstruction and DPCM. The block diagram of these additional modules except Block reconstruction are shown in Figure 5.2d, 5.2e, 5.2f, 5.2g. Block reconstruction is the reversed process of Block sampling. The block diagram of the Decoder LSI is shown in Figure 5.2h. On the decoder, the additional module is only Inverse DPCM.

5.3 Functional description of the encoder

5.3.1 Motion Detection module

This block consists of two portions.

1) Motion vector calculation

Three kinds of motion vectors as follows are calculated using original pictures stored in the frame memory.
- Inter-field Motion vectors for I frame
  Block Size 16x8, full pel resolution, search area of +-7 pixel
- Frame based Inter frame motion vectors
  Block size 16x16, half pel resolution, telescopic search, search area +-15.5 pixel/frame
- Field based Inter frame motion vectors
  Block size 16x8, half pel resolution, telescopic search, search area +-15.5 pixel/frame

2) Global Motion Vector calculation

For Intra frame field motion compensation , Global motion vector is calculated by taking the average of the inter field Motion vectors which occupy more than 40% of all vectors for x and y. The two kinds of inter frame motion vectors and the global motion vectors are transmitted to the Motion compensation module and the VLC module.

5.3.2 Frame memory

Input video data are buffered and converted from raster scan format to progressive scan format. An address generator is included and the starting address and timing is controlled by CPU.

5.3.3 Motion Compensation Module

In this module, Motion compensation is performed and the macroblock type is decided.

1) Calculate the difference between the current block and the motion displaced block.
For motion displaced block in P and B frame, oversampling is done.
For B frame, Interpolated block is generated.
Address controller which generates the starting address offset by motion vector is in this module. The synchronization between input and output is controlled by the CPU.
2) Calculate energy of each macroblock (Luminance only).
3) Decide macro block type using the energy calculated above.
For I frame, macro block type is always Intra.
4) Output residual and the motion displaced block coincide with macro block type.

5.3.4 Interlace Detection Module

This module includes energy calculation units and some hardwares for comparison on purpose of detecting the rapid motion of each macroblock. The input is the current block and the output is the information about which type of coding (field or frame) should be done.

1) Calculate 2 kinds of square error of the macroblock luminance area.
- between the even and the odd line pairs
- between the consecutive odd line pairs and the consecutive even line pairs
2) Compare them with thresholds.
Thresholds are fixed for all sequences and bit-rates.

5.3.5 Edge Detection Module

The block that includes edge is detected. This module consists of AC energy calculation functions, edge judgement units , and a memory to store AC energy of surrounding blocks. CPU manages timing for input data and output results.

For luminance,
1) Calculate AC energy on each input 8x8 block.
2) Compare them with some thresholds and output the results to the block sampling module and the VLC module.
For chrominance,
If any of the luminance block in the same macroblock is detected to contain an edge, then
1) Calculate the AC energy of chrominance block.
2) Compare them with threshold then output the results.
Thresholds are fixed for all sequences and bit-rates.

5.3.6 Block Sampling Module

This module contains 2 block memory, filter for chrominance and selector.
1) Sample the macroblock into four 8x8 blocks using Interlace detection information.
That is achieved by changing the order of data with 2 block memory.
2) For Chrominance, vertical sampling with a linear filter is performed after procedure 1).
3) Transmit blocks that include edge to the DPCM module, and the others without edges to the DCT module.

5.3.7 DCT Module

1) The Discrete Cosine Transform and Inverse DCT are applied on each 8x8 block.
2) The quantization and the inverse quantization are performed to DCT coefficients.
The stepsize is determined for every intra-coded Macroblock from the average of AC coefficients and the quantization parameter generated by Rate Controller.

5.3.8 DPCM Module

This module consists of adders, predictor having fixed table for generating predictive coefficients, memory for storing neighbor pixels, quantizer, and inverse quantizer. The DPCM coding is processed on edge blocks as follows :

For Intra coded block,
1) Calculate difference between current pixel and prediction.
2) Quantize and Inverse quantize.
The stepsize is determined from the quantization parameter generated by Rate Controller. The calculated macroblock quantization parameter (MQuant) is output to VLC module. All parameters used for generating stepsize are constants for all sequences.
3) Reconstruct coded pixels and output to the reconstruction block and to the internal memory.
4) Generate the prediction by referring to the neighboring pixels previously coded and stored in the local memory (2 full lines and 8 lines x 2 pixels).
For the nonintra-coded block,
Skip procedures 1), 3), 4), because the difference between current and the motion displaced block is coded.

5.3.9 Block reconstruction Module

1) Multiplex locally decoded block from DCT and DPCM block.
2) Interpolate the chrominance by using linear interpolating filter.
3) Reorder the pixels with 2 block memory.
4) Reconstruct macroblock by adding the motion displaced macro block and the coded residual.
The components of this module is almost the same as the block sampling module.

5.3.10 VLC Module

1) Multiplex coefficients from DCT and DPCM block.
2) Zig-Zag scan and run-level variable length encode. (coefficients)
There are three run-level tables, for Intra coded block, non-Intra coded block, and for MPEG-I streams.
3) Variable length encode. (side information)
Motion vectors, macro block type, DC coefficients, coded block pattern, macro block address are variable length encoded.
4) Format VLC data to bitstream.
This module has variable length code tables mentioned above and also the code word length tables for formatting data.

5.3.11 Rate Controller & CPU Module

CPU works as Rate controller and besides that it controls the timing between all modules and higher level timing such as Picture layer, GOP layer, Slice layer and so on. Rate control process is as follows.
1) Count the quantity of the buffer fullness.
2) Feedback quantization parameter to the quantizer.
As initial values and the other parameters used are defined for each bit-rates, it is easy for the decoder to set the same value without transmitting these parameters.

5.3.12 Frame memory (coded)

For local decoding, frame memory is necessary for Intra and Predictive frames. Address generators are included and starting address is controlled by CPU.

5.4 Functional description of the decoder

5.4.1 CPU Module

CPU manages the timing of all modules and it regenerates the stepsize of quantizer behaving just the same as the Rate Controller of the encoder.

5.4.2 VLD Module

1) De-format bitstream and variable length decode.
Decoded data except coefficients are output to appropriate modules.
Coefficients are processed as follows.
2) Zig-Zag de-scan and run-length decode.
3) Demultiplex to IDCT or IDPCM module.
This module includes variable length code tables and the code word length tables for de-formatting data.

5.4.3 IDCT Module

Inverse quantization and Inverse Discrete Cosine Transform is processed.
This module is exactly the same as the local decoder.

5.4.4 IDPCM Module

Inverse Quantization and reconstruct pixels.
This module consists of Inverse quantizer, Adder, memory and predictor. The procedures are the same as the local decoder.

5.4.5 Block Reconstruction Module

This module is totally the same as the one in the encoder.

5.4.6 Motion Compensation Module

In this block, the motion displaced block is reconstructed using motion vectors and the macroblock type information transmitted from the encoder.

5.4.7 Frame memory

For reconstructing B frame, two frame memories to store Intra and Predictive frames are necessary and there is one more additional frame memory for display.

5.5 Implementation Calculation for each module

Table 5.5a, and 5.5b show the implementation calculation figures for each module of the encoder and the decoder. These figures are calculated as averages on bit rate of 4Mbps. And it doesn't include DCT and Inverse DCT calculations in DCT and IDCT module.

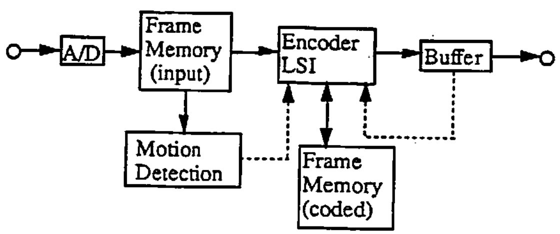


Figure 5.2a. Encoder System Block Diagram

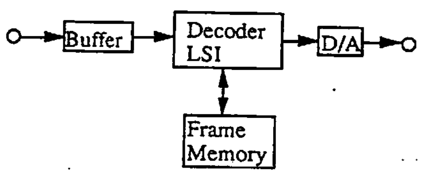


Figure 5.2b. Decoder System Block Diagram

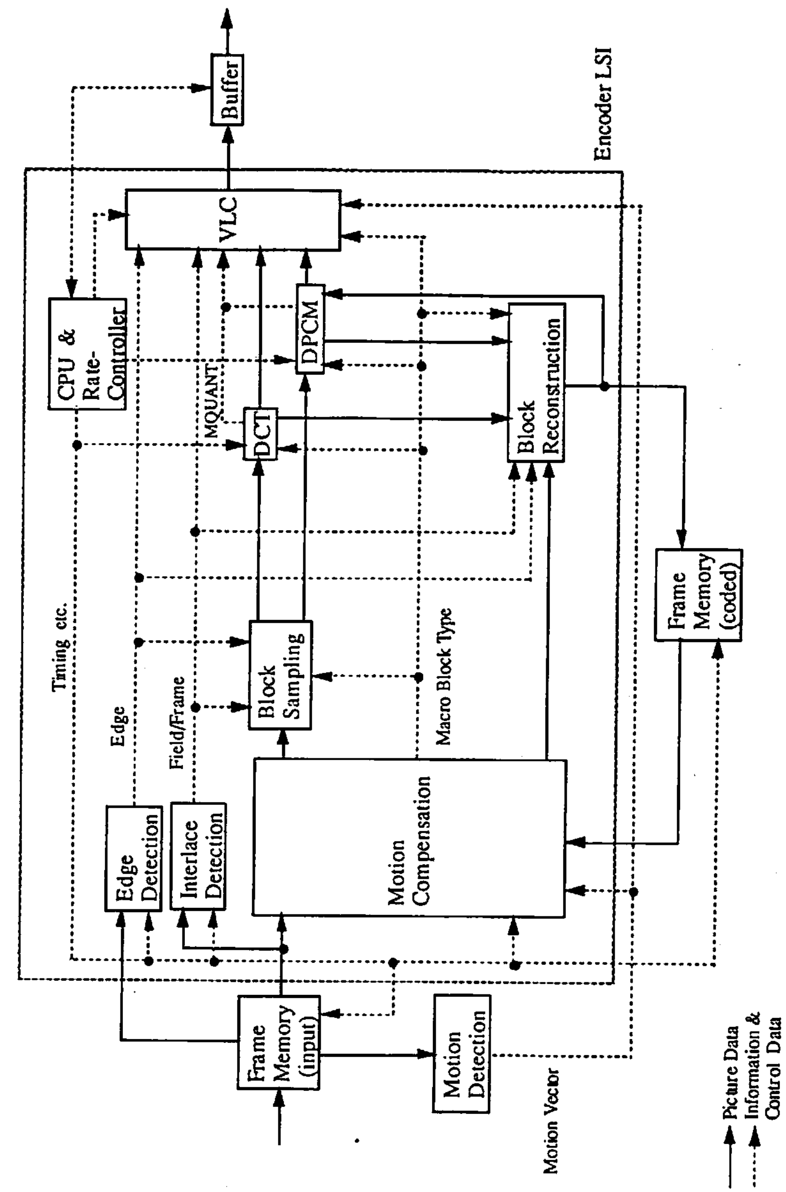


Figure 5.2c. Encoder Block Diagram

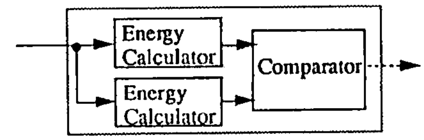


Figure 5.2d. Interlace Detection Module

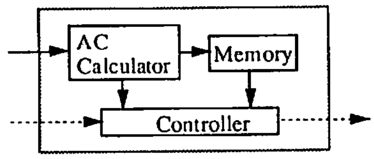


Figure 5.2e. Edge Detection Module

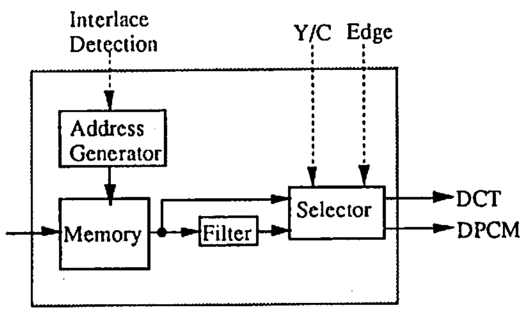


Figure 5.2f. Block Sampling Module

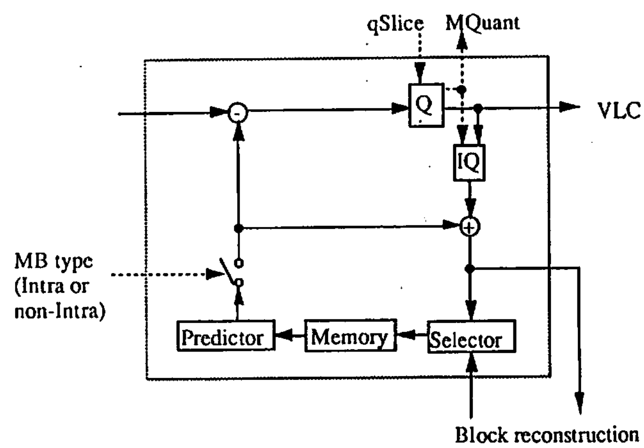


Figure 5.2g. DPCM Module

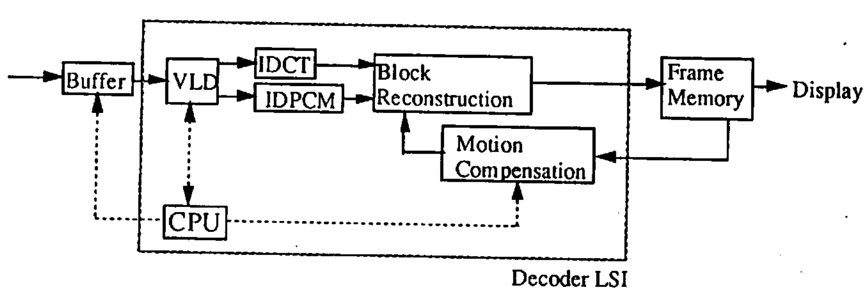


Figure 5.2h. Decoder Block Diagram

Module	Addition width input, bit	Addition width internal, bit	Addition calculation MOPS*	Multiplication width input, bit	Multiplication width internal, bit	Multiplication calculation MOPS*	Memory size bit	Memory width bit	Memory bandwidth MByte/s	Table size bit	Table lookup K/s
Motion Detection	8	9	27319.46								
	9	9	27287.43								
	8	16	54078.57								
	16	17	316.20								
Motion Compensation	8	9	88.70								
	8	16	64.17	8	16	54.98					
	9	9	91.42								
	10	10	0.18								
	16	17	0.29								
Interlace Detection	8	9	10.14	8	16	10.14					
	9	9	10.14								
	8	16	10.14								
	16	17	0.20	16	18	0.08					
Edge Detection	8	16	22.28	8	16	11.31	2.8K	16	0.48		
	14	15	11.14								
	16	17	0.29								
Block Sampling							1.0K	8	20.3		
Block Reconstruction							1.0K	8	20.3		
DCT	12	12	93.08	9	12	15.05				384	913.4
	12	15	14.14	12	14	15.06					
DPCM	8	9	6.09	8	10	4.77	22.7k	8	15.2	48	100.4
	9	10	5.63								
	16	17	2.94	16	18	5.68					
VLC MBA										540	37.1
MTYPE										198	37.1
MV	7	8	0.14							495	69.1
CBP	6	7	0.18							945	36.3
DC-DPCM	9	10	0.02							189	24.7
AC										7.92K	609.3
ZigZag							576	9	15.2		

MOPS* : Mega Operations Per Second

Table 5.5a. Implementation Calculation for Each Module (Encoder)

Module	Addition width input, bit	Addition width internal, bit	Addition calculation MOPS*	Multiplication width input, bit	Multiplication width internal, bit	Multiplication calculation MOPS*	Memory size bit	Memory width bit	Memory bandwidth MByte/s	Table size bit	Table lookup K/s
VLD MBA										540	37.1
MTYPE										198	37.1
MV	8	8	0.14							495	69.1
CBP	7	7	0.18							945	36.3
DC-DPCM	9	9	0.02							189	24.7
AC										7.92K	609.3
ZigZag							576	9	15.2		
Block Reconstruction							1.0K	8	20.3		
IDCT	12	12	67.06	9	12	15.05				384	913.4
IDPCM	8	9	5.79				22.7k	8	15.2	48	100.4
	9	10	1.22								
	16	17	1.52								
				8	10	1.68					
				16	18	2.89					
Motion Compensation	9	9	0.07								
	10	10	0.07								

Table 5.5b. Implementation Calculation for Each Module (Decoder)

6. CHARACTERISTICS & FEATURES

6.1 Random Access

In the present proposal, there is one intra coded frame in every 12 frames for random access entry point. For 30 frames/second system, all intervals between two entry points for random access are therefore about 0.4 second.

6.2 Coding and Decoding Delay

Coding and decoding delays are divided into 3 parts for consideration :

1) encode delay	delay through video input to VLC output
2) transmission buffer delay	delay caused by transmission buffer
3) decode delay	delay through VLD input to video output

And suppositions are that,

1) encoder has a capability to process one frame within 1/30 sec (1 frame time),
2) decoder has a capability to process one frame within 1/30 sec (1 frame time),
3) rate controller exactly controls bits of each frame as follows.

	4 Mbits/s	9 Mbits/s	
I frame	462 000	690 000	bits
P frame	230 000	490 000	bits
B frame	56 000	180 000	bits

All delay times are shown in table 6.2. The details are described in the following sessions.

Table 6.2. Delay time

Item	Delay (msec)
Encode Delay	133.3
Decode Delay	133.3
Encode Decode Delay	166.7
Transmission Buffer Delay	82.2 (4Mbits/s) 43.3 (9Mbits/s)
Total Delay	248.9 (4Mbits/s) 210.0 (9Mbits/s)

6.2.1 Encode Delay

Since the GOP has two B frames between P or I frames, 100 msec (3 frame time) delay is needed at B frame. In addition, 1/30 sec (1 frame time) is needed for Global Motion Vector detection for I frame. So, the total encode delay is about 133 msec (4 frame time) at B frame.

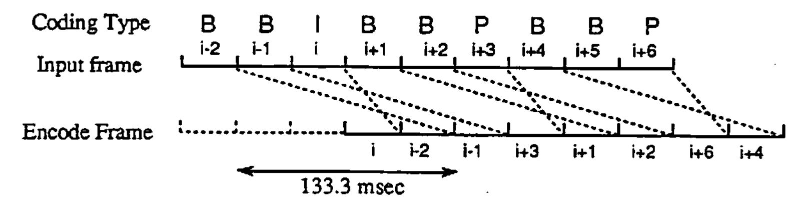


Figure 6.2.1 Encode Delay

6.2.2 Transmission Buffer Delay

Transmission buffer delay depends on maximum bits per frame and transfer bit rate as follows :

Transmission Buffer Delay = Max bits per frame / bit rate - 1/30 (sec)

At 4 Mbits/s, the delay is 426,000 / 4,000,000 - 1/30 = 82.2 msec.
At 9 Mbits/s, the delay is 690,000 / 9,000,000 - 1/30 = 43.3 msec.

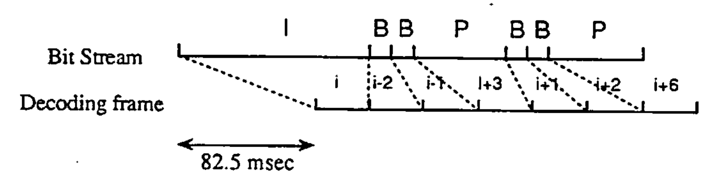


Figure 6.2.2a. Transmission buffer delay (4 Mbits/s)

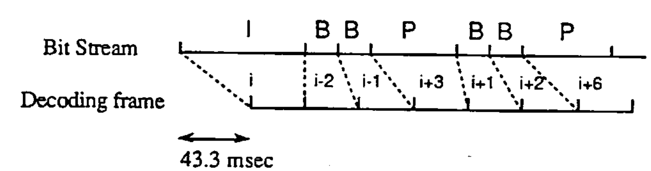


Figure 6.2.2b. Transmission buffer delay (9 Mbits/s)

6.2.3 Decode Delay

The delay caused by GOP structure is 100 msec (3 frame time) at I frame. Furthermore, to display a decoded image onto the monitor, 1/30 sec (1 frame time) delay is needed. The decoder can decode bitstream on MB basis. So, the delay of the decoder processing is 0.025 msec (1/44 macro block x 1/30 slice x 1/30 sec), which is small enough compared with other delays. Hence, the total decode delay at I frame is about 4/30 sec (4 frame time).

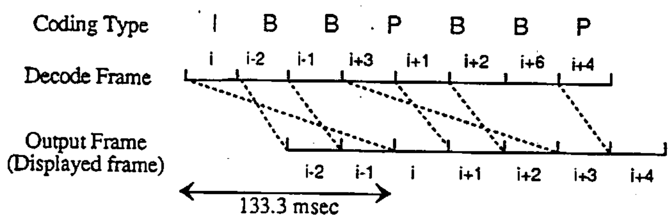


Figure 6.2.3 Decode Delay

6.2.4 Total Delay

The total of encoding and decoding delay except transmission buffer delay is not a summation of both delays because maximum delayed frame type is different between encoder and decoder. This total delay is 5/30 sec (5 frame time) as illustrated in figure 6.2.4.

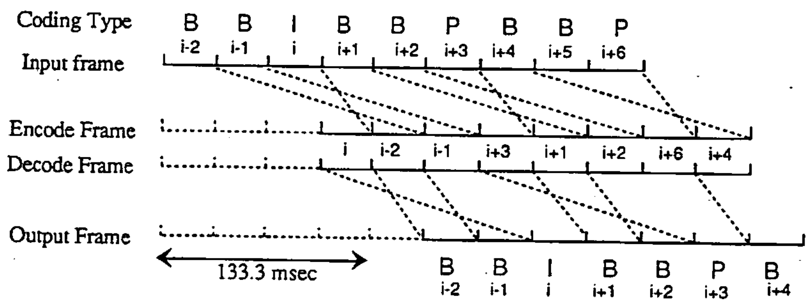


Figure. 6.2.4 Encode and Decode Delay

Transmission buffer delay can simply be added to other delays. So, total delay which includes all delays are about 248.2 msec at 4 Mbits/s and about 210.0 msec at 9 Mbits/s.

6.3 Compatibility

The present proposal supports the forward compatibility with MPEG I system using the Superset Scheme (MPEG-I Syntax + Extension Syntax) as described in the Proposal Package Description Document [4]. MPEG II extension information is inserted as the group_extension_data at GOP layer. The coding of the extension information is done using a "Tag" method which works as followed :

a) Each extension information has a unique "Tag" which is one byte (8 bits) wide;

b) Detailed description (integer number of bytes) of the extension information may follow after its "Tag";

c) All extension information are concatenated together with an "extension_start_code" in front, and terminated with a proper start code. Uniqueness of start code must be maintained in the extension data.

Present extension information proposed (detailed descriptions are in section 3.1) is :

Extension Information	Tag (8 bits)	Further Information
two_vlc_table	0000 0001	None
MPEG2_extension	0000 0010	None
cell_loss_resilience_mode	0000 0011	None

Table 6.3

The Simulcast Scheme [4] may be used in special applications that require backward compatibility. In such cases, the total bandwidth needed is the sum of the bit rate of the MPEG-I and MPEG-II streams. For 4 Mbits/s coding, 1.15 Mbits/s is used for MPEG-I stream and 2.85 Mbits/s is used for MPEG-II stream; Similarly for 9 Mbits/s coding, 1.15 Mbits/s for MPEG-I and 7.85 Mbits/s for MPEG-II.

6.4 Cell Loss Resilience

6.4.1 Cell loss resilience overview

In the present proposed syntax, a new switchable mode, "cell loss resilience mode", for the encoder and decoder is introduced to handle cell loss condition in ATM networks. In this mode, new code table (table 4.1b) and encoding/decoding operations are introduced to localize spatially the degradation due to cell loss problem in an ATM network. These extensions are to :

define new Macroblock Address (MBA) table with absolute and relative address;
reset Motion Vector predictors to "0" in absolute MBA MB;
reset DC predictors to "128" in absolute MBA MB;
and code Quant value as MQuant in absolute MBA MB.

In a cell assembler/disassembler located between a codec and ATM networks, the absolute MBA position information in the bit stream is inserted. In order to protect propagation errors temporally, cells are also prioritized. As for cell loss concealment, the lost MB is displaced using the Motion Vectors of the surrounding MB.

6.4.2 Encoding/Decoding Operations

In SM3, unique words are used in slice, picture,GOP and sequence header; therefore, if cell loss problem occurred, the lost data could only be localized on slice basis. In order to better localize the error, the new MBA table with absolute and relative address is used in cell_loss_resilience mode. Within each slice, the Macroblocks are coded periodically with absolute MBA and relative MBA.

In absolute MBA MB, extra operations are performed at the encoder and decoder :

1) all Motion Vector predictors are reset to "0",
2) all DC predictors are reset to "128" ,
3) the Quant value is coded as MQuant.

6.4.3 Cell assembler/disassembler (CLAD) operation

According to CCITT SGXVIII, one cell is 53 octet length with 44 octet Segmentation And Reassemble sub-layer Protocol Data Unit (SAR-PDU) payload which can be used for video data. This 44 octet payload (352 bits) is divided into 7bit Address Information (AI) and 115 Cell Partitions (CP) with 3 bit length. Each partition has a "0" origin successive Cell Partition Number (CPN).

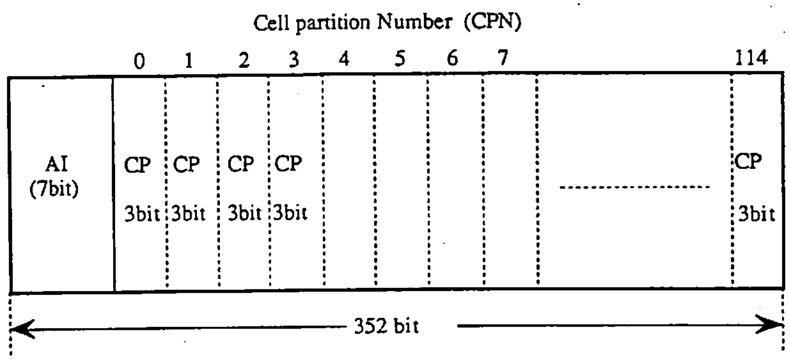


Figure 6.4.3. The Division of SAR-PDU

AI is the binary expression of CPN with whom CP includes the head of an absolute MBA. AI is set to "1111 111" in case of no absolute MBA in the cell.

The absolute MBA head position information in the bit stream is informed from an encoder to a cell assembler. In a cell assembler, if absolute MBA head position is not located the head of CP, one or two bit is stuffed. The stuffed pattern is "1" or "11". In cell disassembler, the absolute MBA head position information is generated using AI. The absolute MBA information and the cell loss occurrence information are informed to a decoder. A cell disassembler does not remove the stuffed bit.

The cells are prioritized in the CLAD, namely the cells including I or P frame data have a high priority and the ones including B frame data have a low priority. AI is inserted in both high and low priority cells.

6.4.4 Decoder Operation

The decoder removes the stuffed bit using the absolute MBA head position information. In case of cell loss, the decoder search the unique word or absolute MBA position and restart decoding from the position. When the decoding is restarted from the absolute MBA position, the decoder judges the vertical position (slice number) of the decoded absolute MBA MB in the following rules.

Case 1 : If the MB number of the absolute MBA MB is grater than the last decoded MB number, the MB is located in the same slice.

Case 2 : If the MB number of the absolute MBA MB is less than or equal to the last decoded MB number, the MB is located in the next slice. If the current slice number is the last one in a frame, the MB is located in the first slice of next frame.

Case 3 : If unexpected vertical slice number is detected after judging as Case 1 and 2, the decoder corrects the decoded MB position as depicted in Figure 6.4.4a. For this correction, the decoder should have 1 slice memory of decoded date.

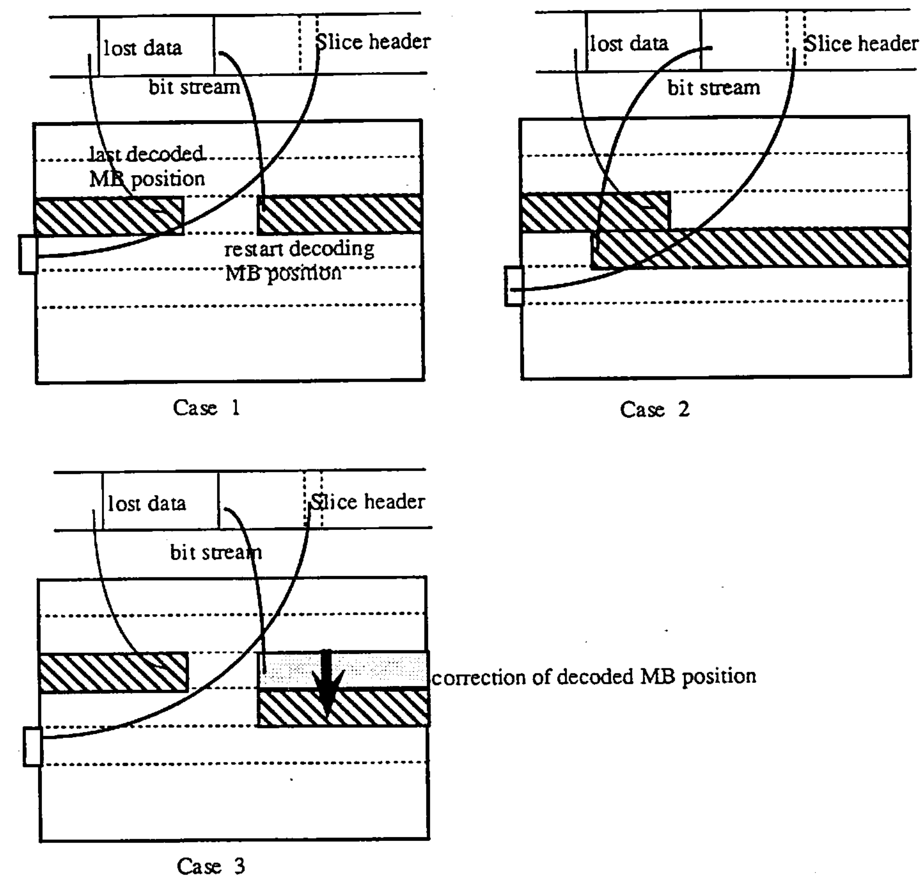


Figure 6.4.4a. Vertical Position Judgement

The intra DPCM MB whose position is right or below of lost MBs is judged the lost MB and the decoded data is discarded by the decoder. After judging lost MBs, the decoder conceals the lost MB data as following steps :

Step1: The decoder checks that there is non-intra and non-lost MB in the surrounding position. The checking order is shown in Figure 6.4.4b. If there is, the lost MB is replaced using the surrounding MB Motion Vectors as not-coded MB.

Step 2: If there is not, this MB is displaced by the same position MB in already decoded and the nearest frame.

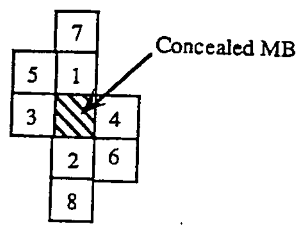


Figure 6.4.4b. Non-Intra, Not-Lost MB Checking Order

6.4.5 Simulation conditions

The computer simulations were carried out under the following conditions.

(1) The cell loss is occurred only low priority cell. It means only B-frame data is lost.
(2) one cell is lost every 0.1 second.
(3) AI is inserted both high and low priority cells.
(4) Absolute MBA is used every four coded MB.

6.5 Fast Forward (FF) and Fast Reverse (FR) Play back

Fast Forward and Fast Reverse playback of the video stream from a digital storage media can be achieved by playing back Intra-coded frames in the present proposal. The speed of these are designed from the number of bits in each Intra coded frames and the bit-rate. For example, when the maximum bits for the Intra coded frames is 520Kbits on the bit rate 4Mbps, it takes 520Kbits / 4Mbps =130msec for reading this frame data from the medium. On the other hand, since the frame rate is 30Hz, that is 33msec, it is necessary to hold on displaying the previous frame for 4 frame time to read this frame data. The speed of FF/FR is calculated as 12/4 =3 as the GOP in Matsushita proposal is 12 frames.

As for the 9Mbps bit-rate system, 4 times speed up is obtained along the same calculation. Figure 6.5a shows the relationship between the speed of FF and the number of iterations for displaying. Figure 6.5b shows the decoder operation of FF on the bit-rate 4Mbps.

4Mbps : 3 times speed

No. of Iteration	Frame Number for Displaying
0	0, 3, 6, 9, 12, 15, 18, 21, 24, 27 ....
4	0, 0, 0, 0, 12, 12, 12, 12, 24, 24 ....

9Mbps : 4 times speed

No. of Iteration	Frame Number for Displaying
0	0, 4, 8, 12, 16, 20, 24, 28, 32, 36 ....
4	0, 0, 0, 12, 12, 12, 24, 24, 24, 36 ....

Figure 6.5a. The Speed of FF and The Number of Iterations for Displaying

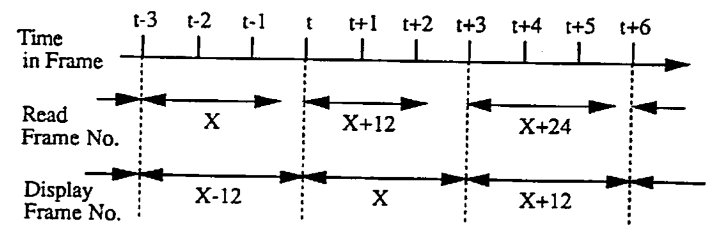


Figure 6.5b. The Decoder Operation of FF on the bit-rate 4Mbps

7. SIMULATION RESULTS

The present proposed algorithm is implemented for simulation of coding and decoding of video sequence at 4 Mbits/s and 9 Mbits/s data rate. The results in terms of bit utilizations and signal-to-noise ratio are included in this section. The tested video sequences are :

Sequence	4 Mbits/s	9 Mbits/s
Flower Garden	X	X
Mobile & Calendar	X	X
Table Tennis	X	X
Popple		X

7.1 Signal-to-Noise Ratio and Bits for Each Frame

Table 7.1a SNR and Bit Count for Flower Garden (4M)

Frame	SNR	Bit Count	Frame	SNR	Bit Count	Frame	SNR	Bit Count
0	32.403	484776	50	31.145	38160	100	30.462	54672
1	31.945	47112	51	32.016	294304	101	30.388	55040
2	31.957	45768	52	31.168	50504	102	31.443	252376
3	32.767	258848	53	31.130	47024	103	30.197	50888
4	31.810	43768	54	32.214	270600	104	29.937	55904
5	31.672	41424	55	31.297	43896	105	31.059	245960
6	32.404	266440	56	31.310	47784	106	30.333	61064
7	31.966	45712	57	32.060	260144	107	30.429	57304
8	31.859	46144	58	31.374	35616	108	30.691	435336
9	32.440	245544	59	31.482	37392	109	30.436	54040
10	31.799	70128	60	30.954	448528	110	30.514	46848
11	31.524	71024	61	30.829	49024	111	31.287	246696
12	31.877	465240	62	30.877	41688	112	30.462	61888
13	31.484	43824	63	31.851	277296	113	30.620	46096
14	31.525	45344	64	31.320	35616	114	31.378	242488
15	32.062	242336	65	31.097	35248	115	30.374	54856
16	31.477	56560	66	31.696	272568	116	30.432	48544
17	31.234	61056	67	31.313	39360	117	31.472	264192
18	32.232	240856	68	31.365	37936	118	31.024	29512
19	31.320	41392	69	32.000	270256	119	31.079	35360
20	31.204	41280	70	31.654	37816	120	30.664	439648
21	31.868	244072	71	31.299	45344	121	30.577	42688
22	31.666	51224	72	31.221	455936	122	30.637	44368
23	31.606	66888	73	30.922	38752	123	31.561	288008
24	31.630	453560	74	31.044	39800	124	31.050	36792
25	31.158	55048	75	31.925	280896	125	30.941	37880
26	31.543	45176	76	31.291	36888	126	31.425	259776
27	32.273	246576	77	31.431	35672	127	30.406	50384
28	31.770	44808	78	31.909	273576	128	30.638	37640
29	31.633	44592	79	31.513	40392	129	31.589	283120
30	32.381	257840	80	31.399	45144	130	31.035	31296
31	31.715	44512	81	32.044	260888	131	31.077	31840
32	31.484	47360	82	31.200	39536	132	30.538	445456
33	32.365	263136	83	31.220	42176	133	30.615	34064
34	31.502	55696	84	30.903	476080	134	30.774	30632
35	31.532	48008	85	31.043	31536	135	31.476	308320
36	31.351	441016	86	30.973	32480	136	31.267	41840
37	31.555	42056	87	31.513	283520	137	31.278	41856
38	31.477	44072	88	31.408	35344	138	31.764	258160
39	32.197	265120	89	31.339	35256	139	31.065	34032
40	31.723	40256	90	31.845	267424	140	30.919	37408
41	31.545	41480	91	31.262	31552	141	31.850	292456
42	32.185	270048	92	30.973	33872	142	31.324	27824
43	31.329	56928	93	31.596	284800	143	31.248	31104
44	31.242	51096	94	31.210	34984	144	30.527	447224
45	32.179	261232	95	30.980	40888	145	30.516	34376
46	31.518	43696	96	30.803	469168	146	30.416	34976
47	31.611	42176	97	30.507	54080	147	31.418	314464
48	31.299	433808	98	30.522	47760	148	31.785	180248
49	31.138	39216	99	31.538	280760	149	31.898	294544

Table 7.1b SNR and Bit Count for Table Tennis (4M)

Frame	SNR	Bit Count	Frame	SNR	Bit Count	Frame	SNR	Bit Count
0	34.953	465848	50	36.810	76224	100	35.446	38808
1	34.580	34296	51	37.430	227800	101	35.607	46872
2	34.633	37856	52	36.573	76288	102	35.724	157680
3	35.262	289872	53	36.488	73584	103	36.261	49832
4	34.921	27752	54	37.082	232528	104	36.265	54120
5	34.649	30536	55	36.498	74872	105	36.472	197200
6	35.112	272032	56	36.456	71872	106	36.775	71768
7	34.380	50824	57	36.884	231216	107	36.613	66544
8	34.210	55848	58	36.327	69576	108	35.594	463336
9	35.442	282624	59	36.405	69560	109	36.492	44640
10	34.691	57096	60	36.851	325240	110	36.549	38744
11	34.545	66720	61	36.436	65232	111	36.945	199064
12	34.339	402368	62	36.414	65760	112	37.142	53224
13	34.207	57704	63	36.940	239088	113	37.072	47632
14	34.259	39760	64	36.209	64280	114	37.613	252808
15	34.837	262552	65	36.078	65272	115	37.201	45040
16	34.294	48688	66	36.820	246152	116	37.132	47744
17	34.233	50216	67	34.583	35392	117	37.671	257816
18	34.976	276664	68	34.749	32296	118	37.209	69552
19	34.392	54008	69	35.149	322824	119	36.800	63512
20	34.395	51872	70	35.074	32656	120	35.353	440080
21	35.036	266288	71	35.042	33496	121	36.084	79880
22	34.165	73232	72	34.955	299488	122	36.204	61248
23	34.049	75144	73	36.090	60456	123	36.897	232992
24	34.737	385752	74	36.100	58216	124	36.359	93336
25	34.486	74632	75	36.331	216856	125	36.460	51496
26	34.833	67504	76	37.056	91168	126	36.795	203024
27	35.488	221504	77	37.018	76680	127	35.930	81880
28	35.081	64480	78	37.419	229592	128	36.153	56064
29	35.132	62104	79	37.120	68824	129	36.833	239912
30	35.725	203624	80	37.109	71056	130	36.672	78008
31	35.498	65920	81	37.656	231400	131	36.340	74368
32	35.702	63352	82	36.796	43000	132	35.451	414272
33	36.477	227248	83	36.930	52504	133	36.366	80944
34	36.407	64592	84	35.495	357984	134	36.080	86752
35	36.691	56160	85	36.819	61168	135	36.638	203496
36	37.539	347320	86	36.813	61064	136	36.210	83968
37	37.207	85872	87	37.243	233352	137	36.259	55120
38	37.291	78352	88	37.155	68352	138	36.487	178200
39	37.803	223280	89	37.220	65976	139	36.240	50592
40	37.272	73728	90	37.839	234616	140	35.986	73464
41	37.211	68008	91	37.267	70592	141	36.665	237576
42	37.943	252448	92	37.294	67904	142	35.973	78296
43	37.238	77456	93	37.959	239000	143	35.951	70080
44	37.208	72832	94	37.168	48904	144	35.194	376264
45	37.910	237568	95	37.241	59000	145	36.068	91960
46	37.539	91648	96	35.625	373520	146	35.750	84544
47	37.367	78856	97	34.054	41032	147	36.491	210272
48	37.817	316032	98	34.324	29200	148	36.450	174592
49	37.006	77232	99	34.795	361816	149	36.407	199432

Table 7.1c SNR and Bit Count for Mobile & Calendar (4M)

Frame	SNR	Bit Count	Frame	SNR	Bit Count	Frame	SNR	Bit Count
0	31.567	507712	50	30.947	48848	100	31.069	50336
1	30.806	51456	51	31.628	216728	101	31.006	47328
2	30.761	52376	52	30.993	56944	102	31.417	227752
3	31.566	239936	53	31.027	53616	103	31.089	47128
4	30.570	56488	54	31.892	234888	104	31.034	51936
5	30.438	53488	55	30.818	57600	105	31.537	238200
6	31.239	226600	56	30.734	57920	106	31.090	54152
7	30.262	56800	57	31.458	232368	107	30.897	56184
8	30.277	54088	58	30.901	56688	108	30.762	485528
9	31.058	220408	59	30.836	53640	109	30.758	61072
10	30.958	64288	60	30.785	476576	110	30.686	57544
11	31.028	67832	61	30.668	57416	111	31.408	221264
12	31.255	481904	62	30.678	55056	112	30.742	60528
13	31.115	59296	63	31.296	230096	113	30.594	60360
14	30.899	59192	64	30.494	52120	114	31.501	215592
15	31.653	207184	65	30.503	53640	115	30.846	50464
16	31.148	55176	66	31.130	229424	116	30.660	60120
17	31.092	53928	67	30.312	56888	117	31.606	202616
18	31.833	211376	68	30.207	53640	118	31.407	59680
19	31.056	58808	69	30.991	230656	119	31.559	55376
20	31.037	57400	70	30.687	59968	120	31.121	496256
21	31.868	226904	71	30.668	57248	121	31.046	61672
22	31.439	54400	72	30.661	474368	122	31.086	52048
23	31.457	56240	73	30.359	54192	123	31.955	223792
24	31.132	477456	74	30.305	52432	124	30.961	69880
25	31.444	45456	75	31.072	241568	125	31.276	56016
26	31.240	51216	76	30.326	70824	126	31.810	212592
27	31.874	222440	77	30.308	60752	127	31.305	60392
28	31.454	51952	78	31.006	212616	128	31.353	53080
29	31.412	50816	79	30.573	63368	129	32.010	213456
30	32.021	229648	80	30.491	61584	130	31.800	55224
31	31.142	85160	81	31.195	207664	131	31.320	63480
32	31.189	65000	82	31.064	56040	132	31.438	506744
33	32.371	212248	83	30.805	61904	133	31.407	54816
34	31.168	58400	84	30.846	483544	134	31.282	60856
35	31.373	59416	85	30.735	58048	135	31.880	202240
36	31.158	488104	86	30.858	56296	136	31.202	54576
37	30.995	64608	87	31.380	205104	137	31.360	56664
38	31.044	62944	88	30.837	61816	138	32.081	205304
39	31.796	209936	89	30.651	61168	139	31.356	49592
40	31.171	56344	90	31.531	216792	140	31.332	53800
41	31.168	55960	91	30.541	58064	141	31.962	217776
42	31.672	204352	92	30.376	58168	142	31.627	51864
43	30.893	51896	93	31.244	218952	143	31.557	55624
44	30.787	52792	94	31.005	56664	144	31.125	512568
45	31.513	226472	95	30.796	58952	145	31.297	45648
46	31.012	62864	96	30.572	480784	146	31.133	48488
47	31.046	56456	97	30.911	54000	147	31.856	216544
48	31.031	484440	98	30.891	50312	148	31.987	221512
49	31.083	52272	99	31.321	213648	149	32.019	229568

Table 7.1d SNR and Bit Count for Flower Garden (9M)

Frame	SNR	Bit Count	Frame	SNR	Bit Count	Frame	SNR	Bit Count
0	34.213	657696	50	33.387	167296	100	32.577	172176
1	34.234	175320	51	34.015	534520	101	32.426	171304
2	34.371	176960	52	33.303	173624	102	34.003	542424
3	35.115	495544	53	33.197	170576	103	32.488	182376
4	34.310	171376	54	34.286	525328	104	32.070	184744
5	34.245	178136	55	33.454	165456	105	33.584	517264
6	34.699	506120	56	33.436	168832	106	32.346	195648
7	34.471	168096	57	34.309	518496	107	32.490	184224
8	34.497	178824	58	33.617	161472	108	32.703	643256
9	34.982	503096	59	33.650	164648	109	32.588	187920
10	33.909	190160	60	32.909	659176	110	32.715	176232
11	33.506	191896	61	32.901	172488	111	33.695	499624
12	33.741	660408	62	33.053	163792	112	32.633	187584
13	33.888	182760	63	34.132	533464	113	32.937	166632
14	34.016	183952	64	33.615	153160	114	33.990	505464
15	34.327	482872	65	33.548	164856	115	32.554	180832
16	33.845	178224	66	33.941	531000	116	32.630	171456
17	33.541	194440	67	33.625	162768	117	34.008	526432
18	34.781	485136	68	33.725	167776	118	33.149	150480
19	33.928	179296	69	34.338	513896	119	33.195	159888
20	33.882	185200	70	33.781	160616	120	32.504	648488
21	34.230	472760	71	33.376	172072	121	32.615	164352
22	34.130	193368	72	33.103	667944	122	32.623	162088
23	33.841	200768	73	32.886	156568	123	33.713	556456
24	33.608	658680	74	32.990	160200	124	33.327	161432
25	33.459	195184	75	33.858	530400	125	33.208	163008
26	33.962	179912	76	33.350	159944	126	33.809	539152
27	34.535	465424	77	33.626	160640	127	32.475	173768
28	34.135	174192	78	34.013	519136	128	32.864	159232
29	34.039	182880	79	33.729	154368	129	33.942	550352
30	34.671	480544	80	33.606	158744	130	33.075	143520
31	34.105	171728	81	34.415	527784	131	33.179	153480
32	33.876	182608	82	33.338	153200	132	32.376	658952
33	34.614	495256	83	33.229	159408	133	32.590	146160
34	33.714	194736	84	32.836	682800	134	32.830	149160
35	33.835	181880	85	33.275	149208	135	33.482	567048
36	33.451	650360	86	33.303	158216	136	33.263	152152
37	33.815	166880	87	33.758	544280	137	33.482	157368
38	33.817	180520	88	33.663	150304	138	34.216	543992
39	34.453	501352	89	33.702	153352	139	33.244	152888
40	34.019	161920	90	34.326	538544	140	33.081	154504
41	33.833	172200	91	33.637	144728	141	34.139	551712
42	34.280	507264	92	33.396	156520	142	33.376	139664
43	33.392	171520	93	33.926	557072	143	33.321	150824
44	33.370	170328	94	33.314	145536	144	32.456	665824
45	34.501	521304	95	33.003	159752	145	32.569	155120
46	33.605	165064	96	32.741	683480	146	32.442	154008
47	33.738	169120	97	32.483	175264	147	33.491	566776
48	33.313	643920	98	32.532	159136	148	34.872	505136
49	33.355	169080	99	33.872	552856	149	33.872	529232

Table 7.1e SNR and Bit Count for Popple (9M)

Frame	SNR	Bit Count	Frame	SNR	Bit Count	Frame	SNR	Bit Count
0	37.627	520584	50	36.537	182656	100	34.221	254912
1	36.611	221272	51	38.593	537376	101	34.114	251784
2	36.441	210688	52	36.368	178464	102	34.802	396552
3	37.976	472120	53	36.390	185264	103	33.663	255752
4	36.366	204736	54	38.554	550040	104	33.550	254032
5	36.488	205552	55	36.433	191360	105	34.577	394992
6	37.991	473656	56	36.343	186768	106	33.663	246568
7	36.410	198024	57	38.354	538880	107	33.471	246400
8	36.489	201320	58	36.510	203584	108	34.804	418392
9	37.988	480680	59	36.426	191552	109	33.714	260264
10	36.459	205160	60	37.362	484824	110	33.639	251432
11	36.404	204720	61	36.260	196080	111	34.989	388680
12	37.204	526184	62	36.246	197376	112	33.760	261448
13	36.305	193040	63	38.395	538984	113	33.749	255472
14	36.366	196920	64	36.396	200200	114	34.843	381488
15	38.120	490312	65	36.361	197944	115	33.852	259744
16	36.358	189816	66	38.251	519368	116	33.816	257248
17	36.373	191424	67	36.446	208368	117	34.714	377400
18	38.284	498704	68	36.415	201536	118	33.569	245736
19	36.317	184152	69	38.245	504960	119	33.522	251512
20	36.413	187224	70	36.456	223344	120	34.628	430160
21	38.370	516864	71	36.176	211856	121	33.282	257736
22	36.643	201096	72	36.995	460928	122	33.365	253632
23	36.490	195752	73	36.345	230976	123	34.357	381072
24	37.476	506824	74	36.103	218256	124	33.165	259944
25	36.443	188552	75	37.448	470096	125	33.005	255272
26	36.372	190400	76	36.193	224032	126	34.238	377672
27	38.500	520256	77	36.194	224688	127	33.285	260624
28	36.557	187432	78	37.240	452432	128	33.141	255704
29	36.536	190968	79	36.248	231840	129	34.118	373808
30	38.523	521984	80	36.234	230984	130	33.034	248224
31	36.482	190048	81	37.449	443976	131	33.142	252592
32	36.525	186456	82	36.479	261664	132	34.416	439488
33	38.507	523400	83	36.466	242896	133	33.116	248480
34	36.640	196552	84	37.353	423944	134	32.958	247856
35	36.546	194688	85	36.587	247576	135	34.119	383304
36	37.432	507424	86	36.353	245056	136	32.861	250960
37	36.457	191160	87	37.055	420992	137	32.787	247392
38	36.494	189240	88	36.009	239104	138	33.920	390112
39	38.297	520584	89	35.816	244824	139	32.743	246536
40	36.482	183984	90	36.928	421472	140	32.878	241760
41	36.523	186744	91	35.706	247736	141	33.947	395976
42	38.423	525536	92	35.600	246176	142	33.084	251048
43	36.562	181904	93	36.556	415440	143	32.876	245392
44	36.499	187264	94	35.139	250256	144	34.208	441648
45	38.516	533336	95	34.948	250896	145	33.108	254784
46	36.543	189816	96	35.781	413944	146	33.050	247048
47	36.527	188600	97	34.883	256544	147	34.165	391568
48	37.345	503576	98	34.787	252016	148	34.185	392880
49	36.424	184248	99	35.529	401728	149	33.943	402104

Table 7.1f SNR and Bit Count for Table Tennis (9M)

Frame	SNR	Bit Count	Frame	SNR	Bit Count	Frame	SNR	Bit Count
0	36.710	672232	50	38.557	241104	100	37.890	210352
1	36.402	187200	51	38.887	409008	101	38.079	203624
2	36.431	173864	52	38.329	238264	102	38.365	480408
3	37.269	526592	53	38.272	242080	103	38.498	214568
4	36.749	143360	54	38.566	408152	104	38.481	214392
5	36.564	196776	55	38.328	238784	105	38.813	472712
6	37.063	507984	56	38.294	241728	106	38.436	211088
7	36.305	209616	57	38.357	406832	107	38.317	205952
8	35.915	207920	58	38.115	242832	108	36.570	497304
9	37.177	472576	59	38.095	237192	109	38.292	212744
10	36.499	223736	60	38.081	467104	110	38.389	209232
11	36.310	227744	61	38.090	223392	111	38.777	476304
12	36.080	606112	62	38.032	227752	112	38.759	223096
13	35.985	222560	63	38.310	424624	113	38.714	220272
14	36.057	204576	64	37.858	219216	114	39.028	463392
15	36.453	439480	65	37.680	222512	115	38.753	229808
16	36.045	209280	66	38.211	434864	116	38.635	223624
17	35.849	211008	67	37.107	164760	117	38.982	448288
18	36.473	443832	68	37.232	174680	118	38.443	215984
19	36.070	215256	69	36.676	453736	119	38.264	222608
20	36.139	212184	70	37.747	178416	120	36.468	497616
21	36.625	439096	71	37.734	180072	121	37.679	223960
22	35.878	253064	72	37.123	513056	122	37.937	222512
23	35.622	235864	73	38.116	204304	123	38.342	441296
24	36.324	565608	74	38.212	207016	124	37.787	227216
25	36.132	246512	75	38.968	465232	125	38.212	222064
26	36.484	222688	76	38.348	213712	126	38.357	439192
27	37.096	399696	77	38.424	210824	127	37.573	224728
28	36.892	229608	78	39.233	470856	128	37.950	225000
29	37.007	223360	79	38.538	221168	129	38.298	436720
30	37.523	398480	80	38.490	215712	130	38.113	224480
31	37.319	228200	81	39.205	460216	131	37.940	223152
32	37.557	227536	82	38.417	226672	132	36.654	500344
33	38.118	400032	83	38.444	218024	133	37.937	231672
34	38.136	223880	84	37.002	502280	134	37.641	231672
35	38.414	213256	85	38.261	209464	135	38.192	428520
36	38.814	499096	86	38.278	213824	136	37.802	239240
37	38.697	223440	87	38.733	455592	137	38.105	235312
38	38.853	221176	88	38.548	236880	138	38.196	412344
39	39.527	437760	89	38.651	232008	139	38.133	232200
40	39.037	233128	90	39.213	413752	140	37.777	234568
41	38.994	227256	91	38.560	221976	141	38.090	413832
42	39.528	432536	92	38.616	227720	142	37.674	231056
43	39.033	236216	93	39.169	422984	143	37.750	233936
44	39.065	232440	94	38.606	233680	144	36.571	499344
45	39.487	422344	95	38.611	231304	145	37.652	237560
46	39.130	247256	96	37.021	495272	146	37.431	237104
47	39.037	241656	97	36.578	165728	147	37.951	407136
48	39.107	467224	98	36.859	163968	148	38.059	393488
49	38.744	244464	99	36.196	479152	149	37.915	408400

Table 7.1g SNR and Bit Count for Mobile & Calendar (9M)

Frame	SNR	Bit Count	Frame	SNR	Bit Count	Frame	SNR	Bit Count
0	33.279	691904	50	33.560	210208	100	33.961	202016
1	33.085	190432	51	33.972	412592	101	33.985	207656
2	32.970	182952	52	33.765	231880	102	34.018	425152
3	34.151	502856	53	33.774	222736	103	34.086	203016
4	33.099	191120	54	34.270	399672	104	34.064	205896
5	33.019	188360	55	33.451	208152	105	34.147	427008
6	34.155	484912	56	33.361	212024	106	33.568	210776
7	33.043	198600	57	33.804	409264	107	33.384	211112
8	33.097	191896	58	33.436	223344	108	32.603	659680
9	34.108	466288	59	33.330	215912	109	33.310	213464
10	33.483	209272	60	32.764	662264	110	33.288	213256
11	33.405	198704	61	33.119	207904	111	33.883	419264
12	33.141	667000	62	33.135	209224	112	33.511	219672
13	33.586	203904	63	33.556	414880	113	33.328	214928
14	33.405	204344	64	33.122	206600	114	34.123	407240
15	34.289	448024	65	33.141	210272	115	33.893	216456
16	34.051	208888	66	33.577	418256	116	33.499	216648
17	33.964	209944	67	32.988	204776	117	34.323	404032
18	34.645	428736	68	32.952	205848	118	33.924	216016
19	33.867	212752	69	33.581	421880	119	34.219	210096
20	33.886	211032	70	33.264	218576	120	32.843	657776
21	34.520	424096	71	33.171	213152	121	33.337	210608
22	33.958	214488	72	32.608	658240	122	33.529	208088
23	34.004	211672	73	32.783	200472	123	34.196	419176
24	33.002	658920	74	32.804	204720	124	33.385	209824
25	34.130	210352	75	33.426	434384	125	33.820	207224
26	33.823	211088	76	33.141	240216	126	34.229	423088
27	34.231	420880	77	33.076	222592	127	34.085	221840
28	34.190	212496	78	33.533	401648	128	34.113	209024
29	34.222	211368	79	33.460	220264	129	34.592	416000
30	34.450	414488	80	33.416	221208	130	34.094	206560
31	33.616	216296	81	33.874	391144	131	33.574	214232
32	33.703	208960	82	33.699	216648	132	33.003	663064
33	35.011	417440	83	33.416	222520	133	33.801	205472
34	33.659	214960	84	32.701	657680	134	33.663	208352
35	33.699	209216	85	33.416	220160	135	34.137	414768
36	32.989	664480	86	33.430	221120	136	33.812	219696
37	33.417	212200	87	33.814	390968	137	33.961	217336
38	33.349	209488	88	33.457	225024	138	34.582	408120
39	34.301	419152	89	33.321	225048	139	34.001	195352
40	33.842	212864	90	33.985	390688	140	34.020	208472
41	33.891	212424	91	33.174	213576	141	34.477	416096
42	34.285	415888	92	33.136	220672	142	34.024	204456
43	33.737	211736	93	33.754	396632	143	33.952	208984
44	33.549	208424	94	33.567	218352	144	32.840	677048
45	34.058	418032	95	33.355	220736	145	33.988	209344
46	33.480	217208	96	32.431	649752	146	33.713	206576
47	33.470	211456	97	33.614	215616	147	34.170	420624
48	32.901	663120	98	33.631	206488	148	34.471	406696
49	33.710	210544	99	33.675	394848	149	34.414	419584

7.2 Cumulative Bit Count

a) Flower Garden (4M) (1 gop approximately = 0.4 second)
Number of bits used for 1 gop is 1666688
Number of bits used for 2 gops is 3266760
Number of bits used for 3 gops is 4873072
Number of bits used for 4 gops is 6472248
Number of bits used for 5 gops is 8070696
Number of bits used for 6 gops is 9661376
Number of bits used for 7 gops is 11251032
Number of bits used for 8 gops is 12838768
Number of bits used for 9 gops is 14523744
Number of bits used for 10 gops is 16089600
Number of bits used for 11 gops is 17673040
Number of bits used for 12 gops is 19256192

b) Table Tennis (4M)
Number of bits used for 1 gop is 1671304
Number of bits used for 2 gops is 3329800
Number of bits used for 3 gops is 4886672
Number of bits used for 4 gops is 6574040
Number of bits used for 5 gops is 8170824
Number of bits used for 6 gops is 9698512
Number of bits used for 7 gops is 11197752
Number of bits used for 8 gops is 12765664
Number of bits used for 9 gops is 14254056
Number of bits used for 10 gops is 15837168
Number of bits used for 11 gops is 17529456
Number of bits used for 12 gops is 19142216

c) Mobile & Calendar (4M)
Number of bits used for 1 gop is 1651472
Number of bits used for 2 gops is 3233280
Number of bits used for 3 gops is 4842488
Number of bits used for 4 gops is 6435216
Number of bits used for 5 gops is 8041168
Number of bits used for 6 gops is 9653896
Number of bits used for 7 gops is 11271208
Number of bits used for 8 gops is 12864776
Number of bits used for 9 gops is 14436536
Number of bits used for 10 gops is 16026680
Number of bits used for 11 gops is 17644568
Number of bits used for 12 gops is 19214424

d) Flower Garden (9M)
Number of bits used for 1 gop is 3593224
Number of bits used for 2 gops is 7192408
Number of bits used for 3 gops is 10755432
Number of bits used for 4 gops is 14293264
Number of bits used for 5 gops is 17856512
Number of bits used for 6 gops is 21411576
Number of bits used for 7 gops is 24919912
Number of bits used for 8 gops is 28460224
Number of bits used for 9 gops is 32181120
Number of bits used for 10 gops is 35736920
Number of bits used for 11 gops is 39312248
Number of bits used for 12 gops is 42836672

e) Popple (9M)
Number of bits used for 1 gop is 3598512
Number of bits used for 2 gops is 7170000
Number of bits used for 3 gops is 10767560
Number of bits used for 4 gops is 14353152
Number of bits used for 5 gops is 17986920
Number of bits used for 6 gops is 21671760
Number of bits used for 7 gops is 25364528
Number of bits used for 8 gops is 29018000
Number of bits used for 9 gops is 32643224
Number of bits used for 10 gops is 36252040
Number of bits used for 11 gops is 39858480
Number of bits used for 12 gops is 43446784

f) Table Tennis (9M)
Number of bits used for 1 gop is 3749600
Number of bits used for 2 gops is 7441912
Number of bits used for 3 gops is 11020768
Number of bits used for 4 gops is 14675072
Number of bits used for 5 gops is 18292736
Number of bits used for 6 gops is 21663864
Number of bits used for 7 gops is 25290656
Number of bits used for 8 gops is 28892120
Number of bits used for 9 gops is 32409336
Number of bits used for 10 gops is 36051992
Number of bits used for 11 gops is 39659928
Number of bits used for 12 gops is 43284624

g) Mobile & Calendar (9M)
Number of bits used for 1 gop is 3697296
Number of bits used for 2 gops is 7342176
Number of bits used for 3 gops is 10948640
Number of bits used for 4 gops is 14561992
Number of bits used for 5 gops is 18181440
Number of bits used for 6 gops is 21775072
Number of bits used for 7 gops is 25409128
Number of bits used for 8 gops is 29009784
Number of bits used for 9 gops is 32569120
Number of bits used for 10 gops is 36179872
Number of bits used for 11 gops is 39783312
Number of bits used for 12 gops is 43353480

7.3 Average Bit Count

TABLE 7.3a : Average Bit Count for Flower Garden Sequence (4M)

	I	P	B
picture header	180	540	1440
slice header	3199	9572	25515
macroblock attributes (SM3)	8985	28233	87517
interlacedMB	3300	9900	23795
interlacedMV	0	8992	39117
adaptive coder	5511	15753	39423
motion vector	0	66779	239603
motion vector2	0	22803	38726
cbp	0	39210	34044
y data	955244	1644454	328784
u data	82836	88392	14882
v data	63672	56772	6344
total data	1135115	1995965	880626
Total = 4011706 bits/sec			

TABLE 7.3b : Average Bit Count for Table Tennis Sequence (4M)

	I	P	B
picture header	180	540	1440
slice header	3173	9546	25459
macroblock attributes (SM3)	8236	33412	86110
interlacedMB	3300	9737	23449
interlacedMV	0	6768	38438
adaptive coder	5310	15584	38700
qstep (mquant)	8183	6433	1481
motion vector	0	47600	224417
motion vector2	0	4099	18054
cbp	0	30300	42881
y data	835978	1523222	695805
u data	39624	45276	15612
v data	52262	63667	23412
total data	956508	1796190	1235263
Total = 3987961 bits/sec			

TABLE 7.3c : Average Bit Count for Mobile & Calendar Sequence (4M)

	I	P	B
picture header	180	540	1440
slice header	3175	9555	25526
macroblock attributes (SM3)	9048	27838	87442
interlacedMB	3300	9899	23973
interlacedMV	0	9058	39224
adaptive coder	13016	39074	100330
qstep (mquant)	12243	4306	1565
motion vector	0	42447	158737
motion vector2	0	7478	18639
cbp	0	43705	46804
y data	952083	1305632	608147
u data	109963	68267	9992
v data	114106	79820	16178
total data	1217378	1647623	1138003
Total = 4003004 bits/sec			

TABLE 7.3d : Average Bit Count for Flower Garden Sequence (9M)

	I	P	B
picture header	180	540	1440
slice header	3175	9562	25504
macroblock attributes (SM3)	8217	26405	93061
interlacedMB	3300	9900	26156
interlacedMV	0	9025	42966
adaptive coder	5511	15750	41855
qstep (mquant)	8088	3816	1717
motion vector	0	66078	242233
motion vector2	0	26746	44523
cbp	0	46903	110177
y data	1376328	3277231	2469350
u data	129006	207206	159410
v data	114923	205701	112049
total data	1648991	3904868	3370446
Total = 8924305 bits/sec			

TABLE 7.3e : Average Bit Count for Popple Sequence (9M)

	I	P	B
picture header	180	540	1440
slice header	3190	9528	25506
macroblock attributes (SM3)	7825	44893	119375
interlacedMB	3300	9900	26352
interlacedMV	0	4510	30263
adaptive coder	6280	18007	46802
qstep (mquant)	6129	9298	21233
motion vector	0	59684	384435
motion vector2	0	8118	53666
cbp	0	29300	86761
y data	863630	2586701	2697768
u data	147319	349686	504481
v data	136108	314460	434470
total data	1174223	3444631	4432558
Total = 9051412 bits/sec			

TABLE 7.3f : Average Bit Count for Table Tennis Sequence (9M)

	I	P	B
picture header	180	540	1440
slice header	3208	9548	25503
macroblock attributes (SM3)	7617	29628	89626
interlacedMB	3300	9896	26017
interlacedMV	0	6856	44502
adaptive coder	5310	15736	41397
qstep (mquant)	5087	3805	1282
motion vector	0	48136	238727
motion vector2	0	4929	21138
cbp	0	37811	111226
y data	1158338	2955827	3581693
u data	55685	87209	88091
v data	70022	110577	117469
total data	1309010	3320503	4388116
Total = 9017629 bits/sec			

TABLE 7.3c : Average Bit Count for Mobile & Calendar Sequence (9M)

	I	P	B
picture header	180	540	1440
slice header	3186	9553	25499
macroblock attributes	8352	25633	91380
interlacedMB	3300	9900	25635
interlacedMV	0	9103	42014
adaptive coder	13016	39076	103330
qstep (mquant)	8761	3459	1591
motion vector	0	42574	161523
motion vector2	0	9006	21715
cbp	0	53812	132842
y data	1304648	2563325	3230130
u data	156551	188791	187469
v data	158802	194721	200843
total data	1657058	3149498	4225418
Total = 9031974 bits/sec			

7.4 Bit stream File Size

```
/usr/research/stchong/mpeg2/stats < 65 > ls -l *.stream.*

-rw-r--r-- 1 stchong   2570270  Oct 9   13:43  flower.stream.4M
-rw-r--r-- 1 stchong   2534927  Oct 9   15:10  tennis.stream.4M
-rw-r--r-- 1 stchong   2561111  Oct 10  14:13  mobile.stream.4M
-rw-r--r-- 1 stchong   5676613  Oct 10  12:06  flower.stream.9M
-rw-r--r-- 1 stchong   5697119  Oct 10  11:48  popple.stream.9M
-rw-r--r-- 1 stchong   5683474  Oct 10  13:48  tennis.stream.9M
-rw-r--r-- 1 stchong   5711686  Oct 10  13:49  mobile.stream.9M
```


7.5 Comparison of Changes in Average SNR for different components

In the tables below, the average SNR for y, u, v, I, P, B and total are tabulated for comparison of the performance of the proposed algorithm (non-cell-loss-resilience mode).

SM3 refers to sequence that was coded with the SM3 algorithm.

Scheme 1 refers to sequence that was coded with SM3 algorithm with extension of proposed Bit Allocation Optimization algorithm that included the MQuant method, the Conditional Replenishment method, and the New rate controller.

Scheme 2 refers to sequence that was coded with the same algorithm as in Scheme 1 plus the proposed Interlace Coding Algorithm which consisted of the Interlaced Motion Compensation method, the Interlaced Macroblock Coding method, and the Global motion Compensation method.

Scheme 3 refers to sequence that was coded with the same algorithm as in Scheme 2 plus the new intra/non-intra run-length vlc tables.

MPEG-II refers to sequence that was coded with the complete proposed Matsushita's MPEG-II algorithm (Scheme 3 plus Edge Enhancement method).

Table 7.5a Average SNR for Flower Garden (4M)

	Average SNR						
sequence	y	u	v	I	P	B	total
SM3	28.86	30.24	32.15	30.71	30.27	30.43	30.41
scheme 1	29.23	30.41	32.10	30.28	30.46	30.67	30.58
scheme 2	30.36	30.67	32.34	31.18	31.04	31.15	31.13
scheme 3	30.52	30.72	32.40	31.22	31.17	31.23	31.21
MPEG-II	30.36	30.67	32.34	31.14	31.05	31.15	31.13

Table 7.5b Average SNR for Table Tennis (4M)

	Average SNR						
sequence	y	u	v	I	P	B	total
SM3	31.86	37.32	37.09	35.32	35.17	35.54	35.42
scheme 1	32.05	37.39	37.22	34.79	35.28	35.76	35.55
scheme 2	32.36	37.62	37.62	35.52	35.59	36.03	35.87
scheme 3	32.49	37.65	37.79	35.66	35.71	36.13	35.98
MPEG-II	32.39	37.62	37.72	35.68	35.62	36.05	35.91

Table 7.5c Average SNR for Mobile & Calendar (4M)

	Average SNR						
sequence	y	u	v	I	P	B	total
SM3	27.31	31.17	31.43	28.59	29.97	30.15	29.97
scheme 1	27.11	31.24	31.54	28.78	29.84	30.17	29.96
scheme 2	29.27	31.61	31.97	30.64	30.83	31.03	30.95
scheme 3	29.46	31.69	32.06	30.74	30.96	31.16	31.07
MPEG-II	29.34	31.56	31.85	31.03	30.79	30.95	30.92

Table 7.5d Average SNR for Flower Garden (9M)

sequence	Average SNR y	u	v	I	P	B	total
SM3	32.68	31.36	33.29	31.92	32.78	32.38	32.44
scheme 1	33.25	31.50	33.34	31.69	32.70	32.83	32.70
scheme 2	34.40	31.77	33.62	32.95	33.28	33.30	33.26
scheme 3	34.60	31.81	33.67	33.07	33.36	33.40	33.36
MPEG-II	34.53	31.80	33.65	33.07	33.32	33.37	33.33

Table 7.5e Average SNR for Popple (9M)

sequence	Average SNR y	u	v	I	P	B	total
SM3	34.60	33.32	34.00	35.60	34.51	33.55	33.97
scheme 1	34.78	33.47	34.17	34.66	34.27	34.02	34.14
scheme 2	35.10	35.65	36.28	36.30	35.92	35.50	35.68
scheme 3	35.09	35.65	36.28	36.29	35.97	35.47	35.67
MPEG-II	34.97	35.50	36.14	36.36	35.77	35.33	35.54

Table 7.5f Average SNR for Table Tennis (9M)

sequence	Average SNR y	u	v	I	P	B	total
SM3	34.80	38.01	38.06	36.79	37.05	36.94	36.96
scheme 1	35.11	38.01	38.07	36.07	6.70	37.33	37.06
scheme 2	35.42	38.33	38.78	37.07	37.14	37.72	37.51
scheme 3	35.65	38.38	38.84	37.21	37.23	37.84	37.62
MPEG-II	35.61	38.36	38.81	37.12	37.22	37.80	37.59

Table 7.5g Average SNR for Mobile & Calendar (9M)

sequence	Average SNR y	u	v	I	P	B	total
SM3	31.27	33.00	33.43	29.81	33.66	32.50	32.57
scheme 1	31.05	32.71	33.22	29.45	32.24	32.74	32.33
scheme 2	33.19	33.17	33.72	32.48	33.24	33.53	33.36
scheme 3	33.41	33.26	33.82	32.59	33.39	33.66	33.50
MPEG-II	33.40	33.18	33.69	32.85	33.24	33.57	33.42

8. REFERENCES

[1] "MPEG Video Simulation Model Three (SM3)", International Organization for Standard, ISO/IEC JTC1/SC2/WG11 MPEG 90/041, July 1990.

[2] CCIR Recommendation 601, "Encoding Parameters of Digital Television for Studies," in CCIR Recommendations and Reports, Vol. XI, ITU, Geneva, Switzerland, 1982.

[3] "Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbits/s", ISO/WG11 MPEG Committee Draft (CD 11172-2), Part 2 Video, August 1991.

[4] "Proposal Package Description for MPEG Phase 2", ISO-IEC JTC1/SC2/WG11 MPEG 91/100 Rev Issue 3, August 23, 1991.

CONTRIBUTIONS

Lucas Hui, Siew Tan Chong, Peck Har Tham
AV / Information Research Center
Asia Matsushita Electric (S) Pte. Ltd.

Toshiya Takahashi, Choong Seng Boon
Image Technology Research Lab.
Matsushita Electric Industrial Co., Ltd.

Wataru Fujikawa, Hiroyuki Uwabo
Matsushita Research Institute Tokyo, Inc.

Takeshi Yukitake, Shuji Inoue, Yutaka Machida
Advanced Development Laboratories
Matsushita Communication Industrial Co., Ltd.

APPENDIX A

A Motion Compensation Method for Interlaced Pictures

A1. INTRODUCTION

In this appendix, we investigated another motion compensation method for interlaced video signal for reference purpose only. This method aimed at providing better motion compensation accuracy without sending complex motion information.

A2. ALGORITHM

A2.1 Overview

To increase the accuracy of prediction for interlace picture without the need to send complex motion information, a method 'Field Adjusted Motion Compensation' was developed. In this method, the 'Even field' is motion compensated from reconstructed odd field which is motion compensated interfields MB by MB and reconstructed even field, and 'Odd field' is also motion compensated by the same way. Motion vector which is used for interfield motion compensation (MVfld) is defined as :

$$MVfld = MV/fd/2 \quad \text{(A1)}$$

MV is a motion vector which is detected between frames, and fd is frame distance.
The advantages for this method are :

1) Since compensate interfield MB by MB, good motion compensation accuracy is obtained,
2) New motion vector detection is not needed for interfield motion compensation,
3) By MVfld force to be 0, this motion compensation corresponds with one of MPEG-I,
4) Since one MV/MB is transmitted, it is easier to make a compatibility with MPEG-I at bit-stream.

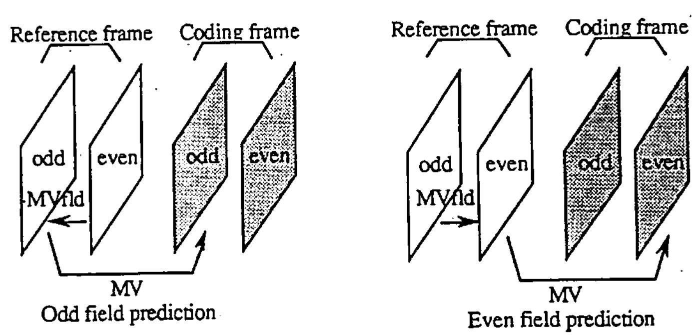


Figure A2.1. Motion compensation scheme

A2.2 Compensation Method

Step 1. Calculate MVfld using equation (A1).
Step 2. Make reference field.
reference odd field:
High resolution field image is made from odd field and motion compensated even field by -MVfld. The pixels of this high resolution field image do not have equal intervals (see fig. A2.2), and the reference odd field which has equal interval pixels is interpolated from nearest pixels.
reference even field:
High resolution field image is made from even field and motion compensated odd field by MVfld. The pixels of this high resolution field image do not have equal intervals (see fig. A2.2), and the reference even field which has equal interval pixels is interpolated from nearest pixels.
Step 3. Motion Compensating using MV field by field.

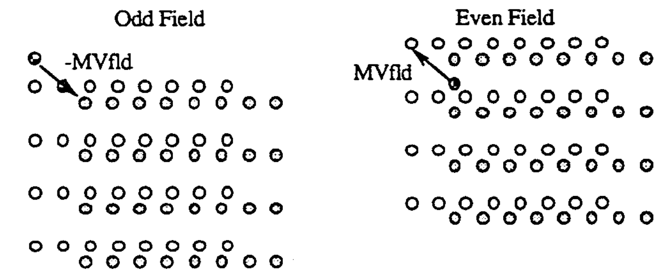


Figure A2.2 High resolution field

A3. SIMULATIONS

A3.1 Simulation model

The simulation mode is based on the Matsushita proposal. Basically, the interlace coding method of the Matsushita proposal is replaced by the Field Adjusted Motion Compensation method for demonstration purposes.

A3.2 Simulation Results

Table A3.2a Flower Garden (4Mbits/s)

	Y	U	V	I	P	B	Total
Fld. Adj. MC	30.44	30.83	32.57	31.13	31.78	31.12	31.28
MPEG-II	30.36	30.67	32.34	31.41	31.05	31.15	31.13

Table A3.2b Mobile & Calendar (4Mbits/s)

	Y	U	V	I	P	B	Total
Fld Adj MC	28.75	31.64	31.93	30.83	31.15	30.63	30.77
MPEG-II	29.34	31.56	31.85	31.03	30.79	30.95	30.92

Table A3.2c Table Tennis (4Mbits/s)

	Y	U	V	I	P	B	Total
Fld. Adj. MC	31.97	37.21	36.03	34.82	35.34	35.01	35.07
MPEG-II	32.39	37.62	37.72	35.68	35.62	36.05	35.91