CCITT SG XV
Working Party XV/1
Experts group on ATM Video Coding

Document, AVC-323
July 1992

INTERNATIONAL ORGANISATION FOR STANDARDISATION

ORGANISATION INTERNATIONALE DE NORMALISATION

ISO-IEC/JTC1/SC29/WG11

CODED REPRESENTATION OF PICTURE AND AUDIO INFORMATION

ISO-IEC/JTC1/SC29/WG11
MPEG 92/N0245

Title: **Test Model 2**

Status: **Draft**

Source: **Test Model Editing Committee**

1 INTRODUCTION

This document gives a comprehensive description of the MPEG-2 Test Model (TM). This model is used in the course of the research for comparison purposes.

In order to obtain results for comparison this document describes some techniques that are not a matter of standardisation. Some of these techniques are of debatable value but are included to provide a common basis for comparisons. In order to have comparable simulation results the methods described in this document are therefore mandatory.

Items currently under discussion in the Test Model Editing Committee are:

- Motion vector syntax in the Macroblock Layer
- Inclusion of CD 11172 terms
- Cleanup of the compatibility and spatial scalabilty section
- The inclusion of some frequency scalability figures
- and many other issues

This may result in an update of this document at a certain time.

It should be noted that some core experiments are not listed in this document.

The readers are asked to give comments and corrections to remove ambiguous parts. Please send them to:

Arian Koster
PTT Research
Tel: +31 70 332 5664
Fax: +31 70 332 6477
Email: a.koster@research.ptt.nl

2 GENERAL CODEC OUTLINE

The generic structure of the test model is based on the following main issues:

- input / output format CCIR 601
- Pre- and post processing, as described in section 3.
- Random access of coded pictures, which requires the definition of Group of pictures, as described in Section 4 and 6.
- Motion Vector search in forward and/or backward direction, as described in Section 5.
- Prediction modes, forward, backward and bi-directional motion compensated, field or frame motion compensation, as described in section 6.
- DCT, on frames or fields as described in section 7
- Entropy coding, as described in section 8.
- 4 Mbps and 9 Mbps target rate, including multiplexing and regulation, as described in section 9 an 10 respectively.
- Scalable bit streams as described in Appendix D.
- Experiments for cell loss are given in Appendix F.
- Experiments for compatibility are given in Appendix G.

2.1 Arithmetic Precision

In order to reduce discrepancies between implementations of this TM, the following rules for arithmetic operations are specified.

(a) Where arithmetic precision is not specified, such as in the calculation of DCT transform coefficients, the precision should be sufficient so that significant errors do not occur in the final integer values

(b) The operation / specifies integer division with truncation towards zero. For example, 7/4 is truncated to 1, and -7/4 is truncated to -1.

(c) The operation // specifies integer division with rounding to the nearest integer. Half-integer values are rounded away from zero unless otherwise specified. For example, 3//2 is rounded to 2 and -3//2 is rounded to -2.

(d) Where ranges of values are given by two dots, the end points are included if a bracket is present, and excluded if the 'less then' (<) and 'greater then' (>) characters are used. For example, [a .. b> means from a to b, including a but excluding b.

3 SOURCE FORMATS

This section gives a description of the Source Formats and their conversion from and to CCIR 601. For the purposes of the simulation work, only the particular formats explained in this section will be used, except for CCIR 601 which is well defined in the CCIR documentation.

3.1 Input Formats

The Input Formats consists of component coded video Y, Cb and Cr. The simulated algorithm uses two source input formats for moving pictures. The differences are the number of lines, the frame rate and the pixel aspect ratio. One is for 525 lines per frame and 60Hz, the other one is for 625 lines per frame and 50Hz.

Input Formats derived from CCIR 601 and defined in this chapter are:

- 4:2:2-50 The actual CCIR 601 signal at 50Hz
- 4:2:2-60 The actual CCIR 601 signal at 60Hz
- 4:2:0 CCIR 601 with half chrominance resolution at 50Hz
- 4:2:0 CCIR 601 with half chrominance resolution at 60Hz
- SIF-525 Progressive format, with 1/8 of the 4:2:0-60 resolution
- SIF-625 Progressive format, with 1/8 of the 4:2:0-50 resolution
- CIF Progressive format, as SIF but fixed paramets, being the maximum of SIF-525 and SIF-625
- SIF-odd SIF taken from CCIR 601 odd fields
- SIF-even SIF taken from CCIR 601 even fields
- HHR Interlaced format with Half horizontal resolution of 4:2:0
- SIF-I Interlaced format, with 1/8 of the 4:2:0 resolution

The parameters for the so called active 4:2:0-525-format and active 4:2:0-625-format frames are:

	4:2:0-625	4:2:2-625	4:2:0-525	4:2:2-525
Number of active lines				
Luminance (Y)	576	576	480	480 / 488 ?
Chrominance (Cb,Cr)	288	576	240	480 / 488 ?
Number of active pixels per line				
Luminance (Y)	704	704	720	720
Chrominance (Cb,Cr)	352	352	360	360
Frame rate (Hz)	25	25	30	30
Frame aspect ratio (hor:ver)	4:3	4:3	4:3	4:3

Table 3.1: Active 4:2:0 and 4:2:2 Formats

For compatibility with MPEG1 or scalability a second set of formats is defined, the MPEG1 SIF. The term SIF is used to indicate this format defined in table 3.2. The parameters for the so called active SIF-525 and active SIF-625 frames are:

	SIF525	SIF625
Number of active lines		
Luminance (Y)	240	288
Chrominance (Cb,Cr)	120	144
Number of active pixels per line		
Luminance (Y)	352	352
Chrominance (Cb,Cr)	176	176
Frame rate (Hz)	30	25
Frame aspect ratio (hor:ver)	4:3	4:3

Table 3.2: Active SIF Format

For compatibility with H.261 a third format is defined, the Common Intermediate Format (CIF). The parameters for the so called active CIF are:

	CIF
Number of active lines	
Luminance (Y)	288
Chrominance (Cb,Cr)	144
Number of active pixels per line	
Luminance (Y)	352
Chrominance (Cb,Cr)	176
Frame rate (Hz)	30
Frame aspect ratio (hor:ver)	4:3

Table 3.3: Active CIF Format

When scalable extensions are used, a hierarchy of formats can exist, with the highest resolution equal to the CCIR 601 Active 4:2:0 format, and with lower resolutions having either 1/2, 1/4, or 1/8, the number of pixels in each row and column.

3.2 Definition of fields and frames

The CCIR 601 and the active 4:2:0 formats are both interlaced. A frame in these formats consists of two fields. The two fields are merged in one frame. The odd lines are within one field the even lines in the other field. There is a sampling time difference between the two fields. Let us define FIELD1 as the field preceding FIELD2.

1. Video data is 50 Hz or 60 Hz fields per second. The first field is the odd field, and is numbered field 1. The second field is the even field and is numbered field 2 and so on. So odd numbered fields are odd fields and even numbered fields are even fields.

2. 50 Hz fields have 288 lines each, and 60 Hz fields have 240 lines each. The fields are considered to be interlaced, and the first line of the first (odd) field is above the first line of the second (even) field for both 50 and 60 Hz.

3. The field lines are numbered as if they are combined into a frame, and the numbering starts at one. So, the first line of the frame, which is the first line of the first (odd) field is line 1. The second line of the frame which is the first line of the second (even) field is line 2. And so on, so odd numbered lines are in odd fields, and even numbered lines are in even fields.

4. For display of 50 Hz material, the 288 lines are the active 288 lines of that format. This will display correctly since in that format, the first line of the first field is above the first line of the second field.

5. For display of 60 Hz material, the 240 lines are placed in a specific set of the 243 active lines of that format. The first (odd) field is displayed on lines 21, 23, ..., 499, and the second (even) field is displayed on lines 22, 24, ..., 500. The active lines 19, 501, and 503 of the odd fields and the active lines 18, 20, and 502 of the even fields are displayed as black.

3.3 Conversion of CCIR 601 to the Input formats

For conversions to several formats a number of filters will be defined. At the edges of the image data is it recommended to repeat the pixel value at edge.

3.3.1 Conversion of CCIR 601 to the 4:2:0 format

Pre processing is applied to convert the CCIR 601 format to the 4:2:0 format. This is described in the following.

First the signal is cropped from 720 luminance pels per line to 704 pels per line by removing 8 pels from the left and 8 pels from the right. Similarly the 360 chrominance pels per line are cropped to 352 pels per line by removing 4 pels from the left and 4 pels from the right.

Luminance: two fields are merged in their geometrical order to form a frame.
Remark: Some processing in the running of the coding scheme is however field based (DCT coding, prediction); thus it is still needed to know for each line of pixels which field it originates from.

Chrominance: The following 7 tap vertical filter is used to pre-filter the FIELD1
[-29, 0, 88, 138, 88, 0, -29] /256
Then, vertical sub sampling by 2 is performed.

The following 4-tap vertical filter is used to decimate the FIELD2:
[1, 7, 7, 1]/16
Then, vertical sub sampling by 2 is performed.

The two sub sampled chrominance fields are merged to form a frame. This is shown in figure 3.1.

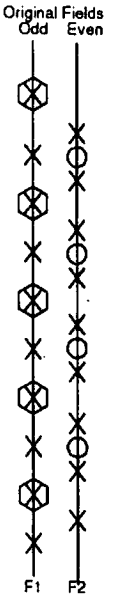


Figure 3.1: 4:2:0 Chrominance sub sampling in the fields

Figure 3.2 : 4:2:0 Chrominance sub sampling in a frame

NOTE: The horizontal positions of the chrominance sample is not rigth in between 4 luminance pixels.

In figure 3.1 and 3.2 the following symbols are used:
X the vertical position of the original lines
○ the vertical position of lines of the sub sampled odd field
⊗ the vertical position of lines of the sub sampled even field

3.3.2 Conversion of CCIR 601 to SIF

The CCIR 601 formats are converted into their corresponding SIFs by sampling odd fields and using the decimation filter of table 3.4.

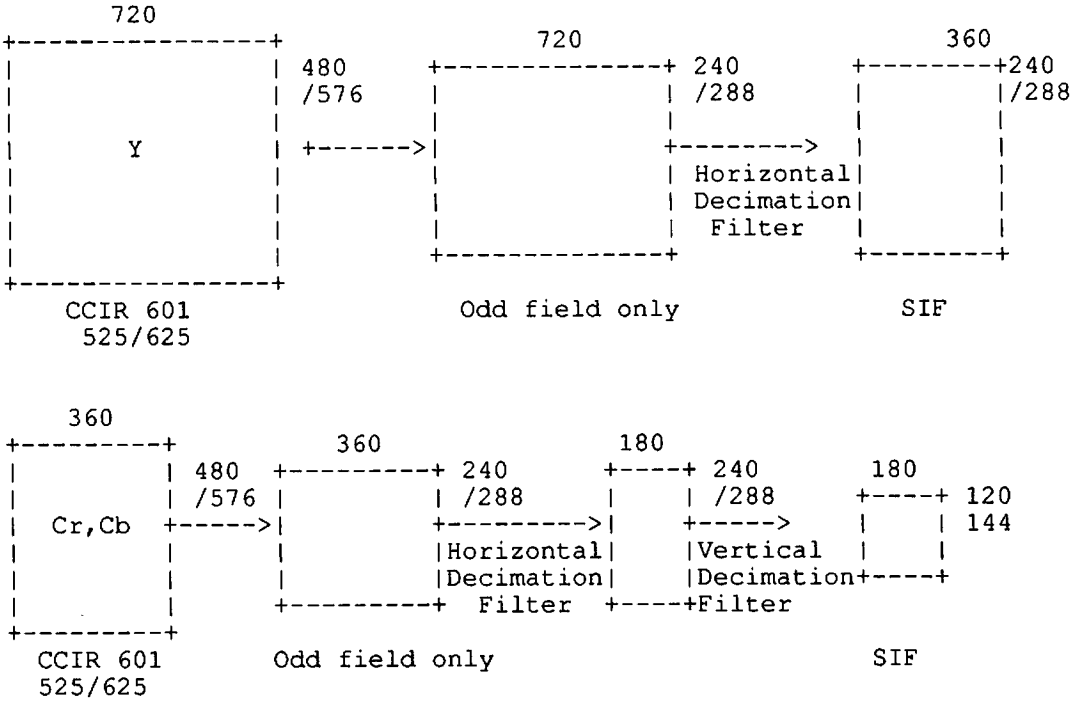


Figure 3.3 Conversion from CCIR 601 into SIF

The filter coefficients are depicted in table 3.4.

-29	0	88	138	88	0	-29	//256

Table 3.4 Decimation filter

Note: the odd fields contain the top most full line

3.3.3. Conversion of CCIR 601 to SIF Odd and SIF Even

The CCIR 601 formats are converted into their corresponding SIF Odd by sampling odd fields and SIF Even by sampling even fields and then applying horizontal decimation files of Table 3.4 in each case.

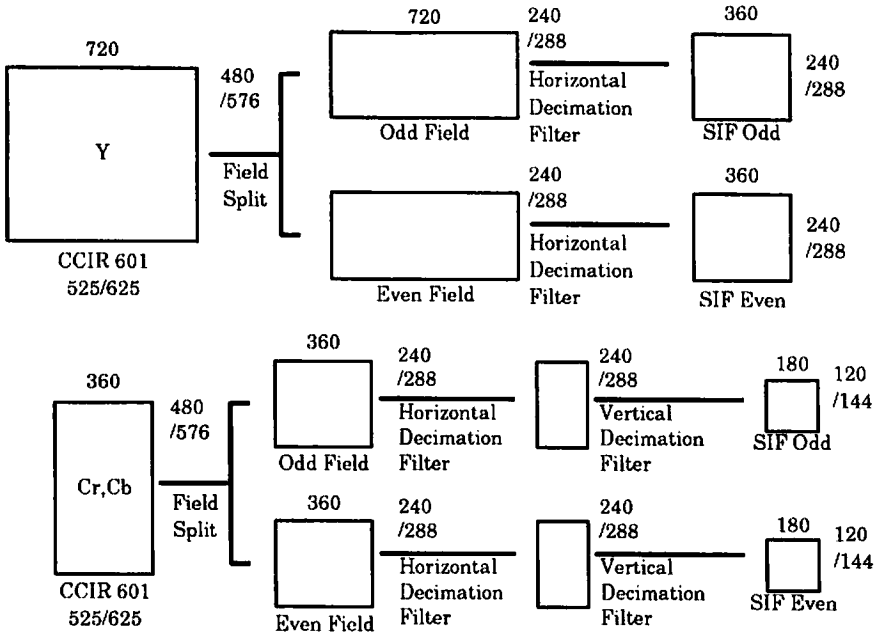


Fig 3.4: Conversion from CCIR 601 into SIF Odd and SIF Even

3.3.4 Conversion of CCIR 601 to HHR

The CCIR 601 formats are converted into their corresponding HHR's by first decimating to SIF Odd and SIF Even as in previous section, and then creating interlaced frames by alternating between lines of SIF Odd's and SIF Even's.

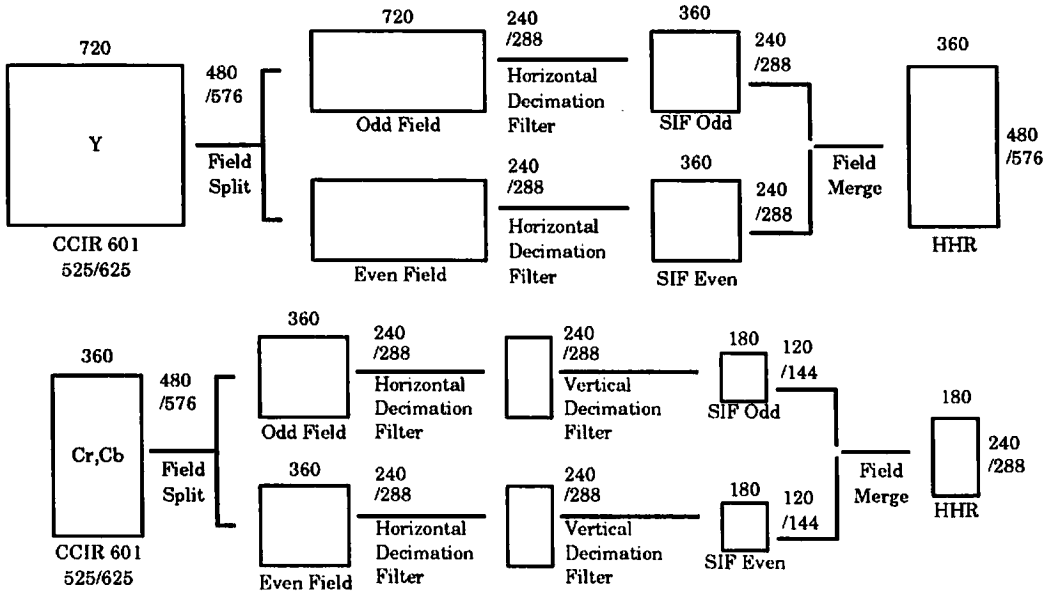


Fig. 3.5 Conversion from CCIR 601 into HHR

3.3.5 Conversion of CCIR 601 to SIF Interlaced (SIF-I)

The CCIR formats are converted into their corresponding SIF interlaced by following the sequence of decimation operations show in Fig. 3.6. The horizontal filter used for decimation is from Table 3.4. The filter used for vertical decimation of odd fields is also from Table 3.4; for decimation of even fields a new filter is specified below.

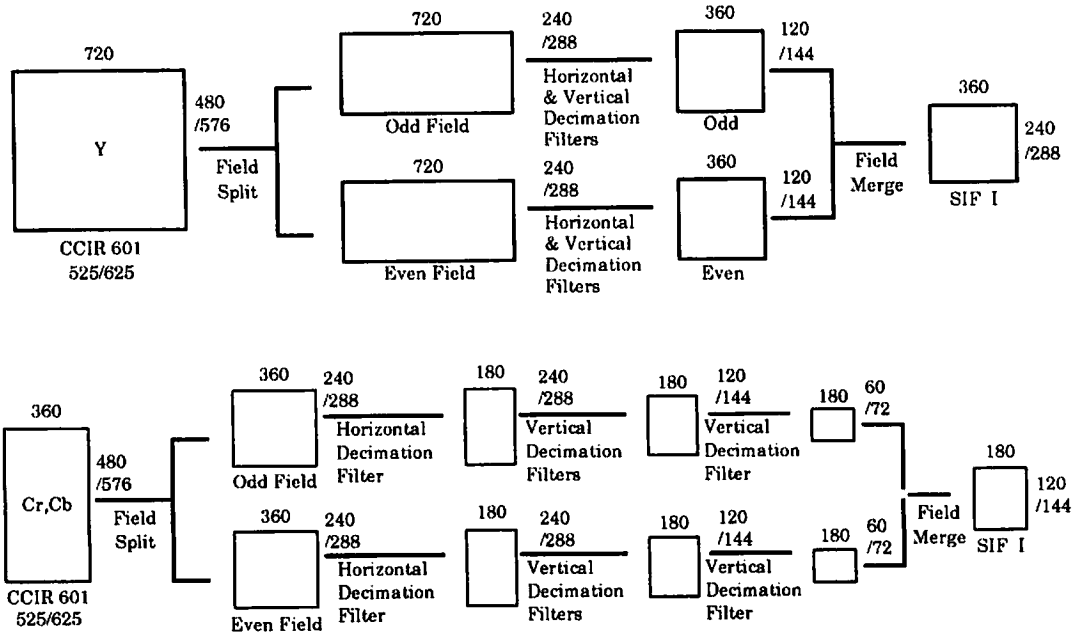


Fig. 3.6: Conversion from CCIR 601 into SIF Interlaced (SIF-I)

Horizontal Filter: Table 3.4

Vertical Filter:

Odd Field	Table 3.4
Even Field	-4, 23, 109, 109, 23, -4

Note: The SIF-I interlaced images generated seem devoid of jerkiness but appear blurry. Better choice of decimation filters needs further investigation.

3.4 Conversion of the Input Formats to CCIR 601

3.4.1 Conversion of the 4:2:0 Format to CCIR 601

Luminance samples of each 4:2:0 field are copied to the corresponding CCIR 601 field.

Chrominance samples are not horizontally resampled.

Vertical resampling of the chrominance is done differently on field 1 and field 2 because of the different locations of the chrominance samples.

In field 1, the chrominance samples in the CCIR 601 field are obtained by interpolating the chrominance samples in field 1 only of the 4:2:0 format. Referring to line numbers defined in the 4:2:2 frame, samples on lines 1, 5, 9 etc. are copied from the corresponding lines in the 4:2:0 field. Samples on lines 3, 7, 11 etc. are interpolated by the even tap filter [1, 1]//2 from the corresponding adjacent lines in the 4:2:0 field.

In field 2, the chrominance samples in the CCIR 601 field are obtained by interpolating the chrominance samples in field 2 only of the 4:2:0 format. Referring to line numbers defined in the 4:2:2 frame, samples on lines 2, 6, 10 etc. are interpolated from the corresponding adjacent lines in the 4:2:0 field using a [1, 3]//4 filter. Samples on lines 4, 8, 12 etc. are interpolated by a [3, 1]//4 filter from the corresponding adjacent lines in the 4:2:0 field.

3.4.2 Conversion of SIF to CCIR 601

A SIF is converted to its corresponding CCIR 601 format by using the interpolation filter of table 3.5.

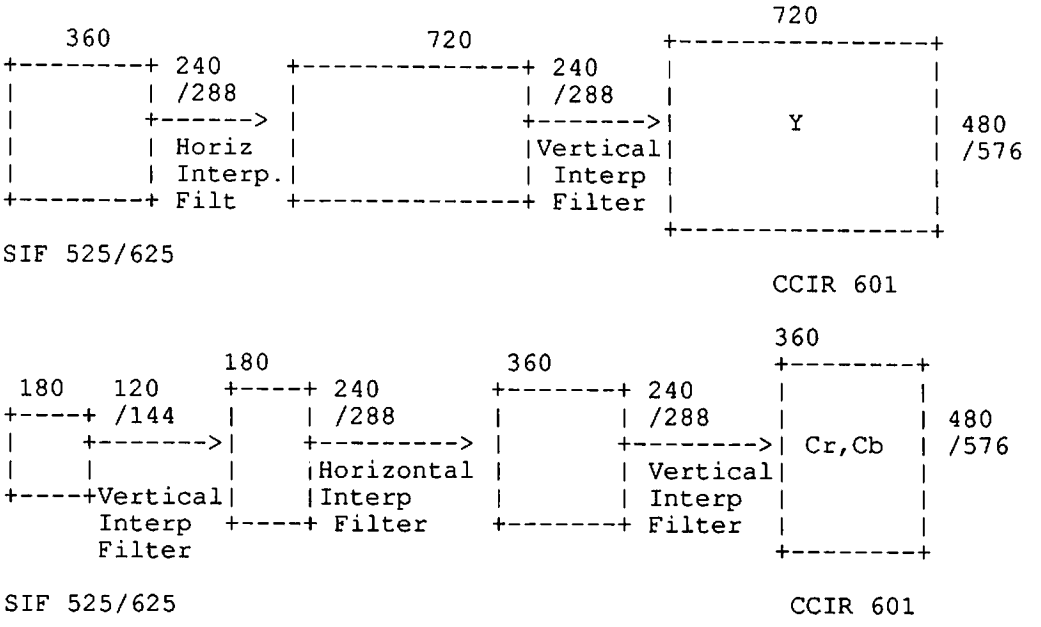


Figure 3.7: Conversion of SIFs to CCIR 601 formats

The filter coefficients are shown in table 3.6.

-12	0	140	256	140	0	-12	//256

Table 3.5 Interpolation filter

Note: the active pel area should be obtained from the significant pel area by padding a black level around the border of the significant pel area.

3.4.3 Conversion of SIF Odd and SIF Even to CCIR 601

SIF Odd and SIF Even are interpolated using interpolation filters of Table 3.5 and interlaced CCIR 601 is created by merging of fields by alternating between lines of upsampled odd and even fields.

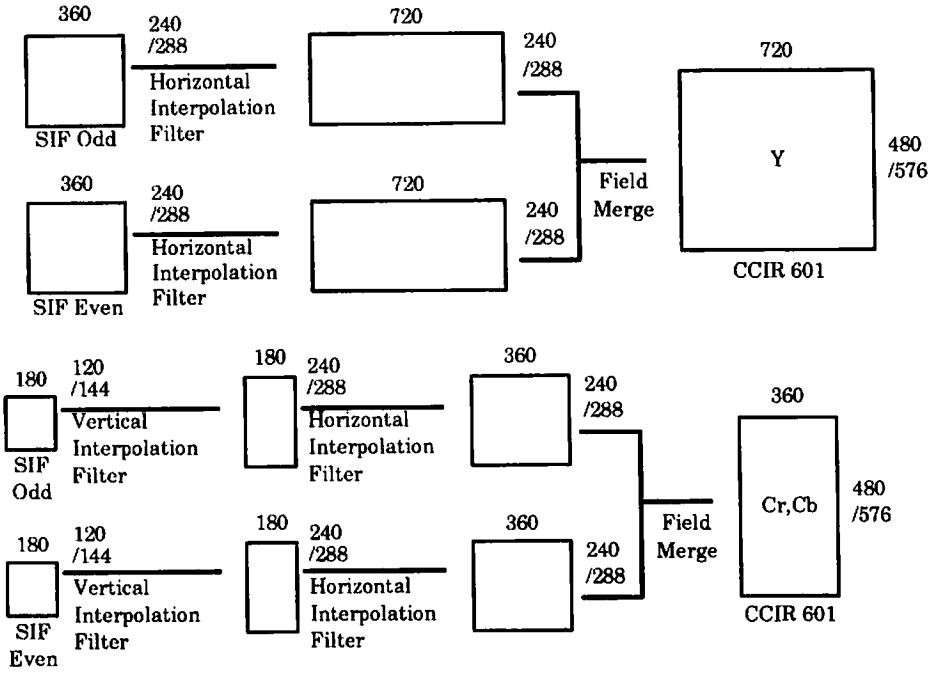


Figure 3.8: Conversion of SIF Odd and Even to CCIR 601

3.4.4 Conversion of HHR to CCIR 601

HHR is split into SIF Odd and SIF Even, each of which are interpolated using interpolation filter of Table 3.5 and interlaced CCIR 601 is created by merging of fields consisting of alternating between lines of upsampled odd and even fields.

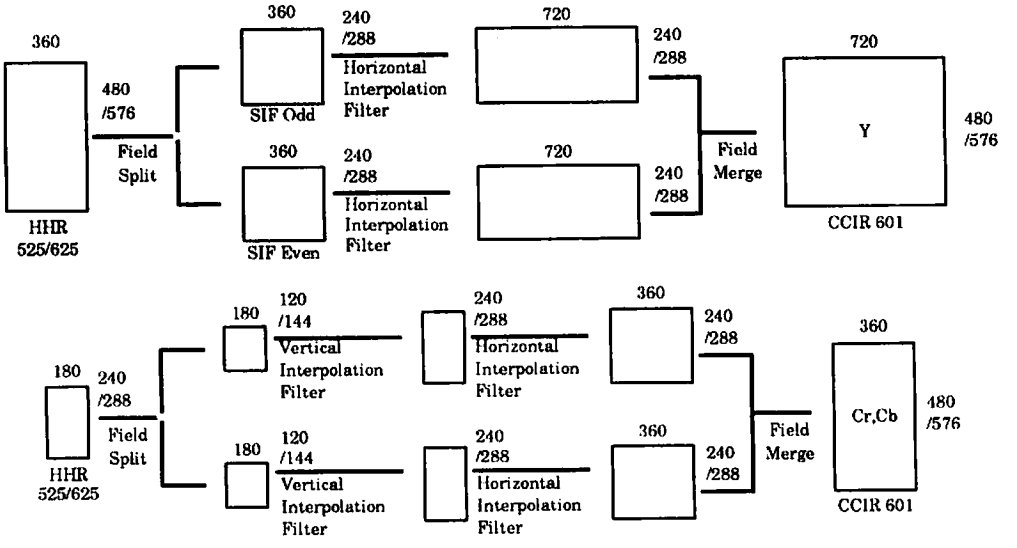


Figure 3.9: Conversion of HHR to CCIR 601

3.4.5 Conversion of SIF interlaced to CCIR 601

SIF interlaced format is interpolated to CCIR 601 by following the sequence of operations shown in Fig. 3.9. The horizontal filter used for interpolation is that of Table 3.5. The filter used for vertical interpolation of odd fields is also that of Table 3.5; for vertical interpolation of even fields a new filter is specified below.

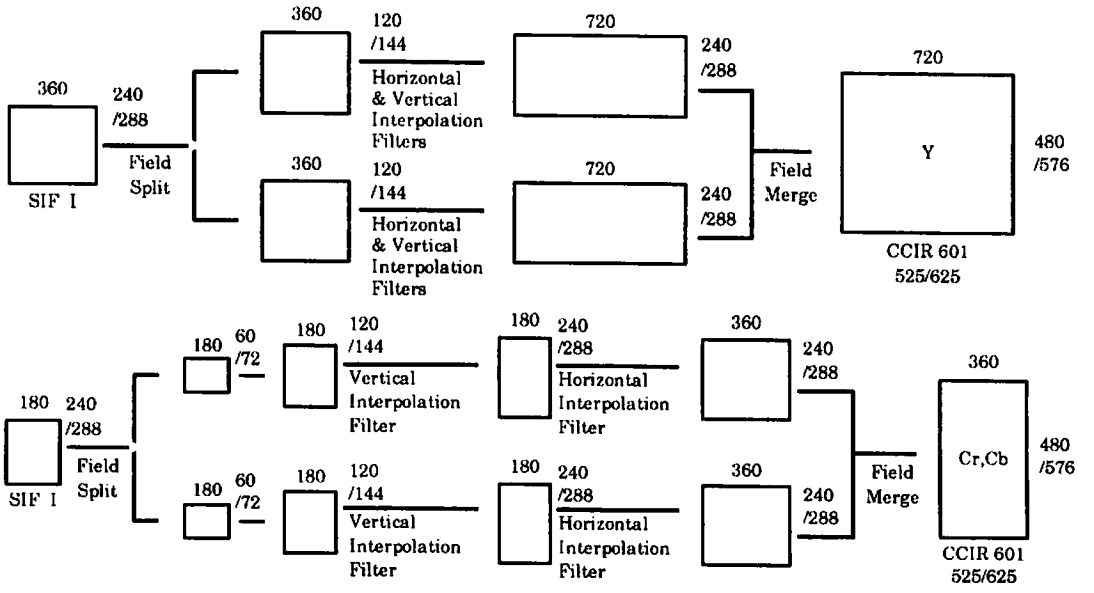


Figure 3.10: Conversion of SIF interlaced (SIF-I) to CCIR 601

Horizontal Filter: Table 3.5
Vertical Filter:
Odd field: Table 3.5
Even field: -4, 40, 220, 220, 40, -4

Note: The upsampled CCIR 601 images seem devoid of jerkiness but appear quite blurry. Better choice of interpolation filters needs further investigation.

4 LAYERED STRUCTURE OF VIDEO DATA

4.1 Sequence

A sequence consists of one or more concatenated Groups of Pictures.

4.2 Group of pictures

A Group of Pictures consists of one or more consecutive pictures. The order in which pictures are displayed differs from the order in which the coded versions appear in the bit stream. In the bit stream, the first frame in a Group of Pictures is always an intra picture. In display order, the last picture in a Group of Pictures is always an intra or predicted picture, and the first is either an intra picture or the first bi-directional picture of the consecutive series of bi-directional pictures which immediately precedes the first intra picture.

It should be noted that the first Group of pictures will start with an Intra Picture, and as consequence this Group of pictures will have less Bi-directional pictures then the other Groups of pictures.

4.3 Picture

Each Picture can be Frame-Structure or Field-Structure.

4.3.1 Field-Structure Picture

The terms *Field-Picture* and *Frame-Picture* is used instead of *Field-Structure Picture*. and *Frame-Structure Picture*.

A Field-Picture is formed of two fields transmitted in a successive order. The transmission order of the fields is fixed and same as display order (odd field first, even field second).in case of the simplified syntax.

In the syntax for the core experiments the transmission order of the fields is flexible in case of Field-Structure P- or I-Pictures, but not flexible in case of Field-Structure B-Pictures (field 1 followed by field 2),

In Field-Structure P- and B-Pictures, both fields must be P- or B-. The fields of those Field-Pictures are called *P-Field* and *B-Field*..

However, when an Intra Field-Picture, It is possible (but not required) to use Predictive-coding-type to transmit the second field.

A Picture Header is transmitted *before each field* of the Field-Picture. Therefore, if a transmission error causes the decoder to desynchronize while decoding field 1 of a Field-Structure B-Picture, a resync can be done at the Picture Header of field 2, and field 2 can be decoded correctly.

In case of Field-Pictures, Field 1 can be used as prediction for field 2 except in the case of B-Fields.

4.3.2. Frame Pictures

Pictures can be intra, predicted, or interpolated pictures (known as I-pictures, P-pictures, and B-pictures - see section 6.1). The arrangement of pictures, in display order, in a Group of Pictures of this TM for the frame-picture coding mode is shown in Figure 4.1. In the figure frames -1 to 13 are part of a Group of Picture.

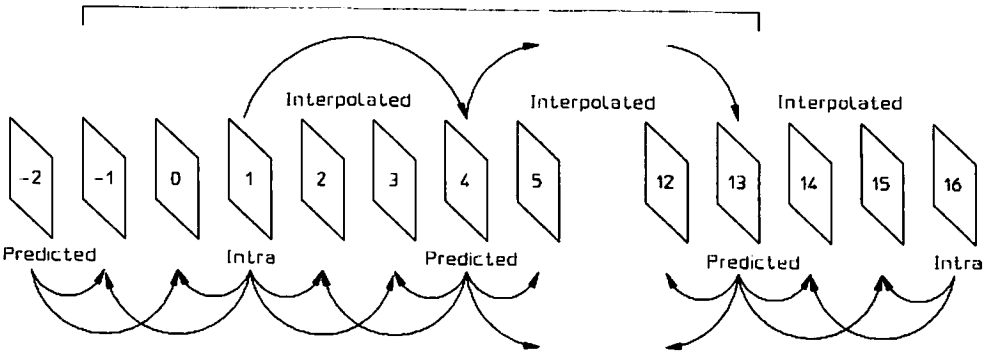


Figure 4.1 Structure of a Group of Pictures in Frame Picture Coding mode in display order

4.4 Macro block Slice

A frame is devided into a number of contiguous macroblock slices, for this TM a fixed structure is used and given in figure 4.2.

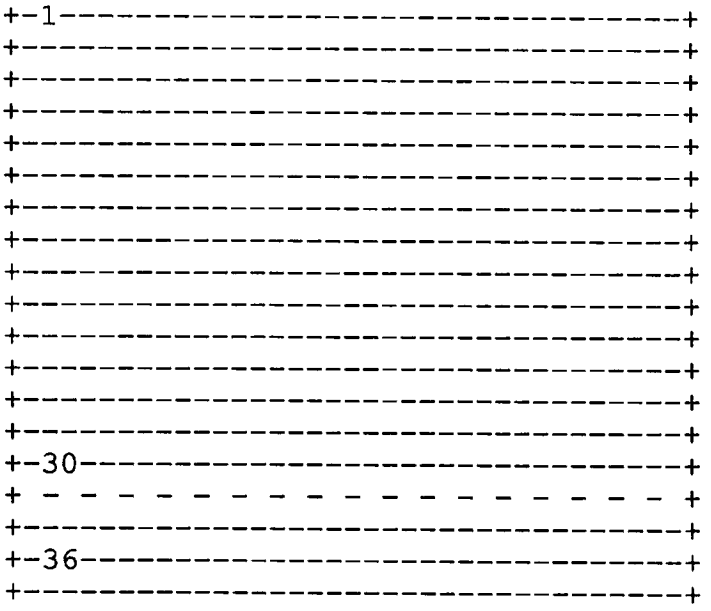


Figure 4.2 Arrangement of Slices in a Picture in Frame Coding mode

For the purposes of simulation, each frame consists of 30 or 36 Macro block Slices (MBS, see section 4.4). 4:2:0-525 has 30 MBSs and 4:2:0-625 has 36. These MBSs cover the significant pel area. The arrangement of these MBSs in a frame is shown in figure 4.2.

A Macroblock Slice consists of a variable number of macroblocks. A Macroblock Slice can start at any MB and finish at any other MB in the same frame. In this Test Model, a Macroblock Slice consists of a single row of 44 Macroblocks, beginning at the left edge of the picture, and ending at the right edge.

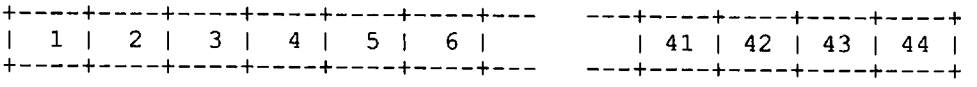


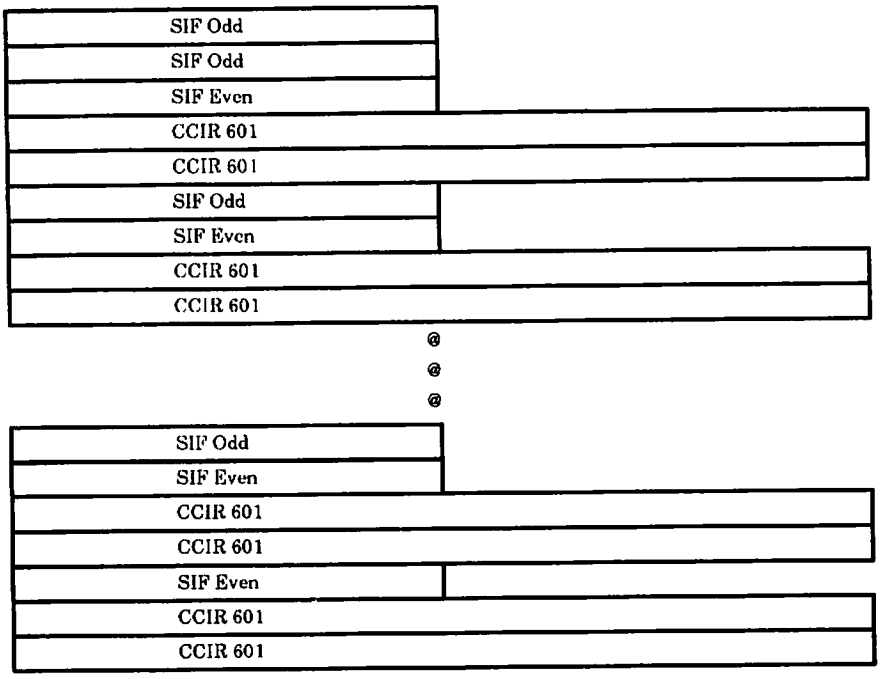
Figure 4.3: Test model Macroblock Slice Structure

When scalable extensions are used (Annex D), the Slice layer may contain Macroblocks of resolution lower than 16x16.

4.4.1 Slave Slice (Frequency scalable extension)

Slave slices are layers of Slave macroblocks which are spatially co-located with the Macroblocks in the Slice layer.

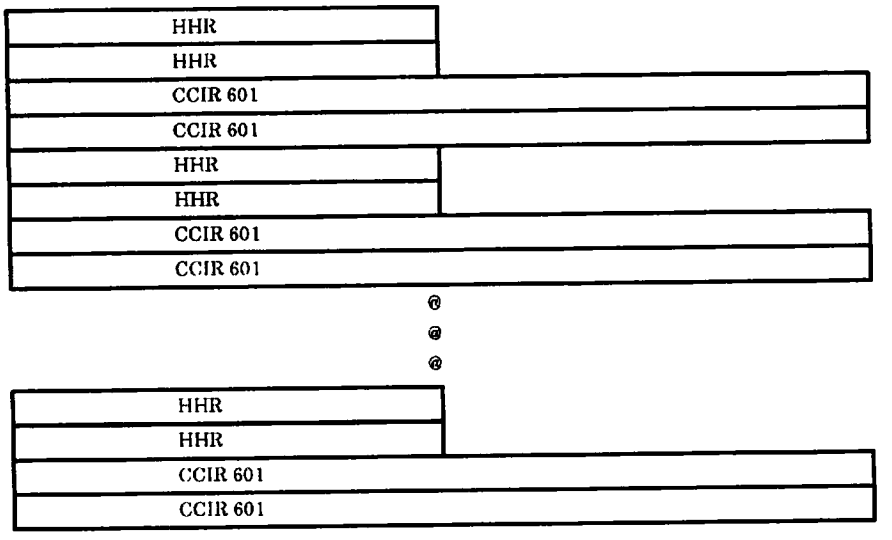
4.4.2 Slices in a Picture - Compatibility Experiment 2 (Appendix G.2)



[Editorial note: This figure has to be changed according to Figure G.2(i) and G.2(ii) in document MPEG92/354]

SIF Odd is MPEG-1 coded to produce MPEG-1 compatible constrained bit stream.
Decoded SIF Odd and SIF Even are used as compatible prediction of CCIR 601.

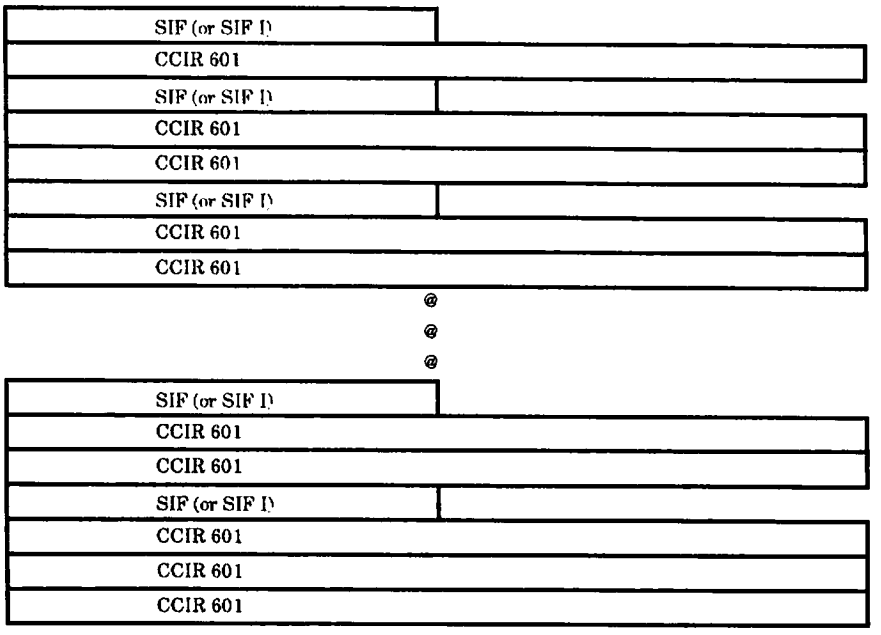
4.4.3 Slices in a Picture - Compatibility Experiment 3 (Appendix G.3)



[Editorial note: This figure has to be changed according to Figure G.4 in document MPEG92/354]

HHR is MPEG-1 coded to produce MPEG-1 compatible unconstrained bit stream.
Decoded HHR layer is used as compatible prediction of CCIR 601.

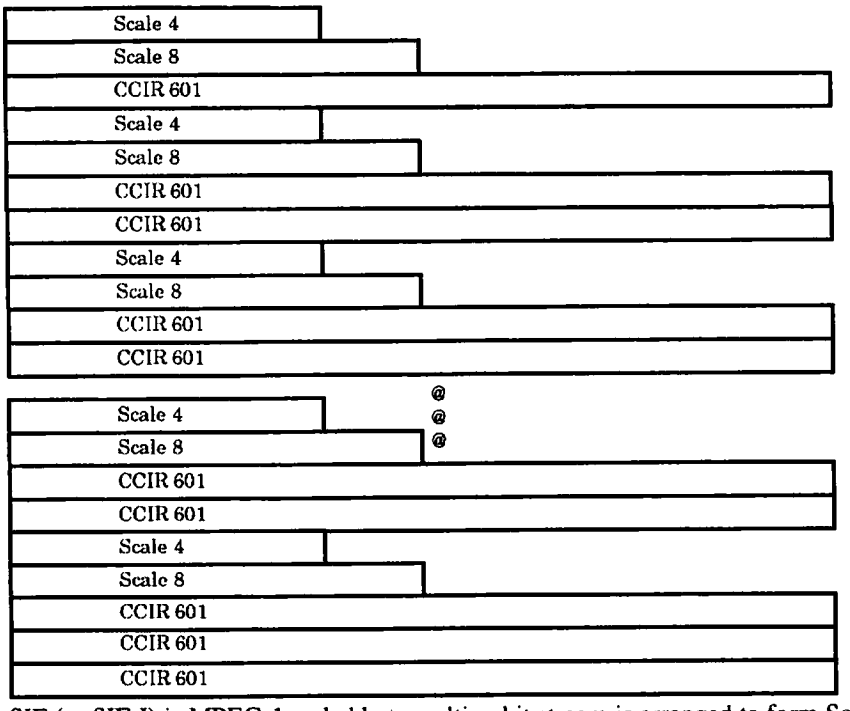
4.4.4 Slices in a Picture - Compatibility Experiment 4 (Appendix G.4)



[Editorial note: This figure has to be changed according to Figure G.4 in document MPEG92/354]

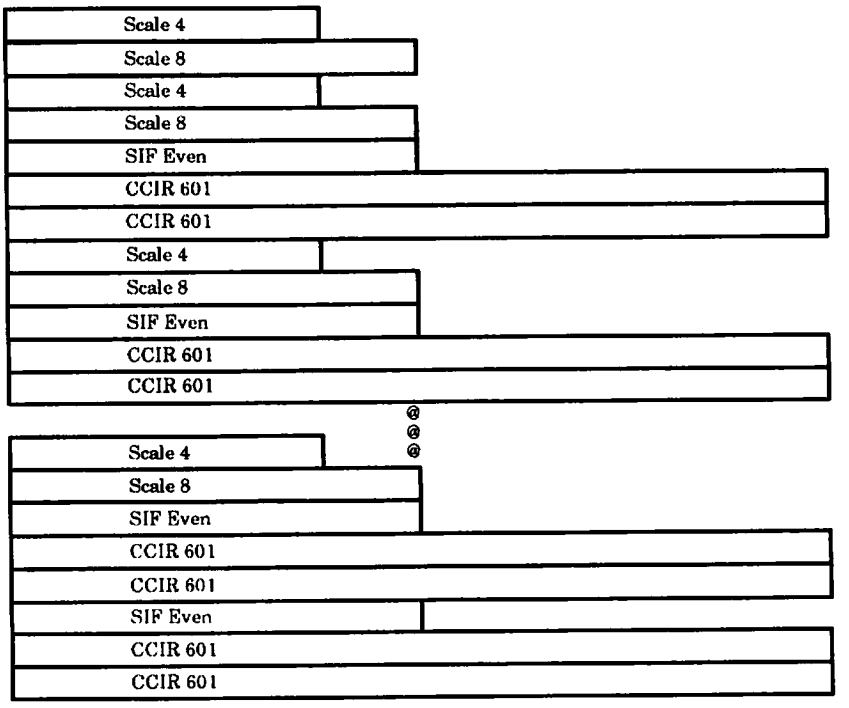
SIF (of SIF-I) is MPEG-1 coded to produce MPEG-1 compatible constrained bit stream.
Decoded SIF (or SIF-I) is used as compatible prediction of CCIR 601.

4.4.5 Slices in a Picture - Hybrid Experiment 1(a) (Appendix I.3)



SIF (or SIF-I) is MPEG-1 coded but resulting bit stream is arranged to form Scale 4 and Scale 8 sub streams. Decoded SIF (or SIF-I) layer is used as spatial prediction of CCIR 601.

4.4.6 Slices in a Picture - Hybrid Experiment 1(b) (Appendix I.3)



SIF Odd is MPEG-1 coded but resulting bit stream is arranged to form Scale 4 and Scale 8 sub streams. Decoded SIF Odd layer is used to predict SIF Even layer. SIF Odd and SIF Even are used as spatial prediction for CCIR 601.

Note: The given slice multiplexing structures may not work for all picture (frame field) structures that can be selected in compatibility and hybrid experiments. These multiplexing structures should however be used as a guide if particular variation of an experiment requires a new structure.

4.5 Macroblock

A 4:2:0 Macroblock consists of 6 blocks. This structure holds 4 Y, 1 Cb and 1 Cr Blocks and is depicted in figure 4.4a.

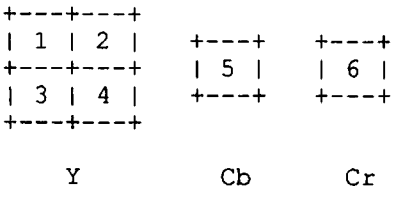


Figure 4.4a: General 4:2:0 Macroblock structure

When the picture format is 4:2:2, a Macroblock consists of 8 blocks.

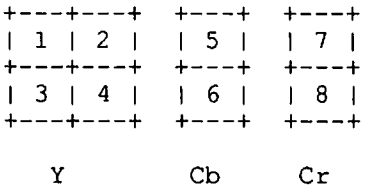


Figure 4.4b: 4:2:2 Macroblock structure

The internal organisation within the Macroblock is different for Frame and Field DCT coding, and is depicted for the luminance blocks in figure 4.5 and 4.6. The chrominance block is in frame order for both DCT coding macroblock types.

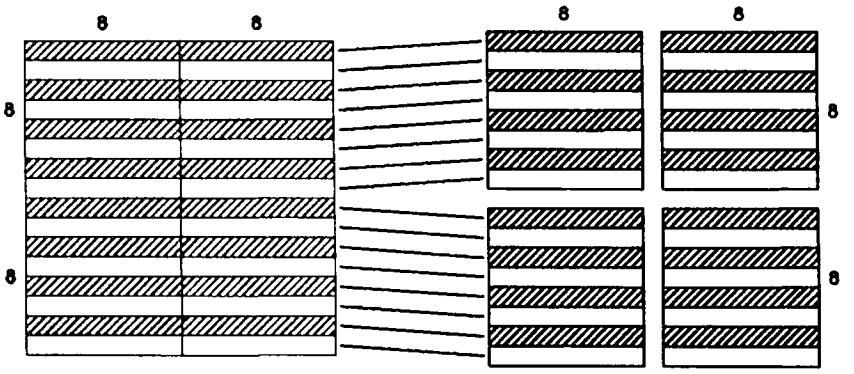


Figure 4.5 : Luminance Macroblock Structure in Frame DCT Coding

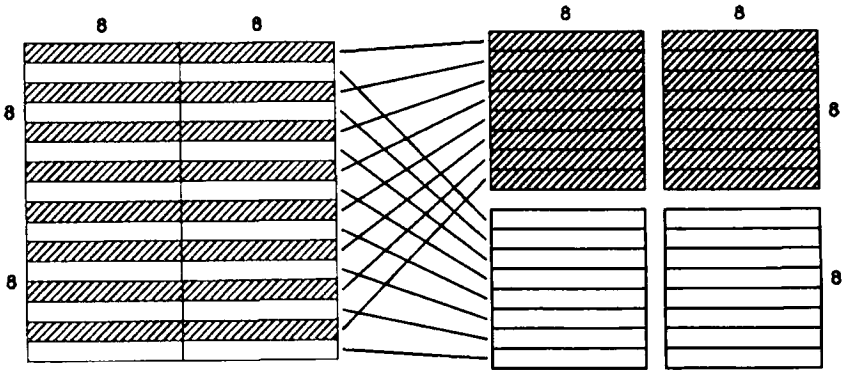


Figure 4.6 : Luminance Macroblock Structure in Field DCT Coding

When scalable extensions are used (Annex D), Macroblocks may contain scaled_blocks of resolution lower than 8x8.

4.5.1 Slave_macroblock (Frequency scalable extension)

Slave_macroblocks are layers of slave_blocks which are spatially co-located with the scaled_blocks in the Macroblock layer.

4.6 Block

A Block consists of an array of 8x8 coefficients. Figure 4.7 shows coefficients in the block in zigzag scanned order.

```
+--+--+--+--+--+--+--+--+
| 1| 2| 6| 7|15|16|28|29|    +------> increasing cycles per
+--+--+--+--+--+--+--+--+    |        picture width
| 3| 5| 8|14|17|27|30|43|    |
+--+--+--+--+--+--+--+--+    |
| 4| 9|13|18|26|31|42|44|    |
+--+--+--+--+--+--+--+--+    V
|10|12|19|25|32|41|45|54|
+--+--+--+--+--+--+--+--+    increasing cycles per
|11|20|24|33|40|46|53|55|    picture height
+--+--+--+--+--+--+--+--+
|21|23|34|39|47|52|56|61|
+--+--+--+--+--+--+--+--+
|22|35|38|48|51|57|60|62|
+--+--+--+--+--+--+--+--+
|36|37|49|50|58|59|63|64|
+--+--+--+--+--+--+--+--+
```

Figure 4.7: Block structure

4.6.1 Scaled_block (Frequency scalable extension)

When frequency scalable extensions are used (Annex D), a scaled_block is used instead of a block. A scaled_block may consist of an array of N x N coefficients, where N is 1, 2, 4, or 8".

4.6.2 Slave_block (Frequency scalable extension)

Slave_blocks are arrays of coefficients, which are used to enhance the spatial or amplitude resolution of the coefficients in the corresponding Scaled_block layer. Figure 4.8 shows the Scaled_block and Slave_block structures that are possible in a frequency scalable bit stream.

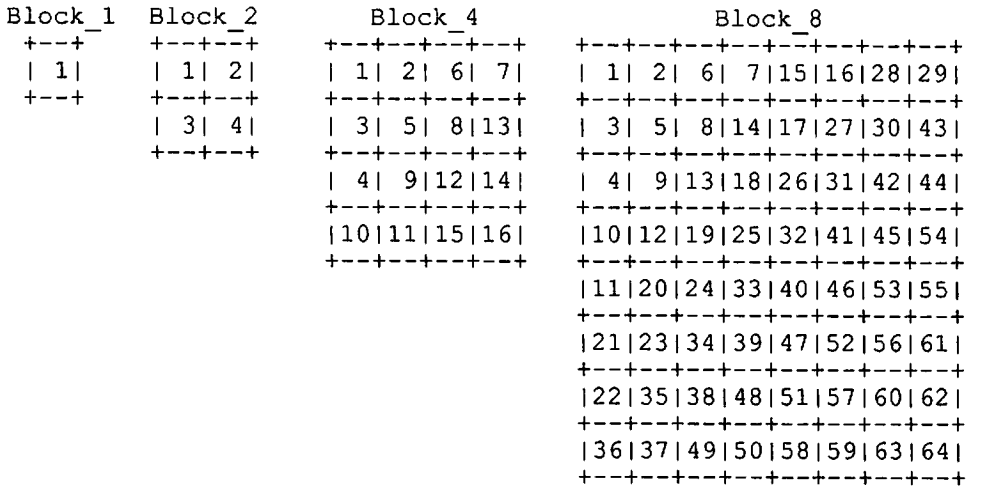


Figure 4.8: Block structures for scalable bit streams

5 MOTION ESTIMATION AND COMPENSATION

To exploit temporal redundancy, motion estimation and compensation are used for prediction.

Prediction is called forward if reference is made to a frame in the past (in display order) and called backward if reference is made to a frame in the future. It is called interpolative if reference is made to both future and past.

For this TM the search range should be appropriate for each sequence, and therefore a vector search range per sequence is listed below:

Table Tennis:	±15 pels/frame
Flower Garden	±15 pels/frame
Calendar	±15 pels/frame
Popple	±15 pels/frame
Football	±31 pels/frame
PRL CAR	±63 pels/frame

A positive value of the horizontal or vertical component of the motion vector signifies that the prediction is formed from pixels in the referenced frame, which are spatially to the right or below the pixels being predicted.

5.1 Motion Vector Estimation

For the P and B-frames, two types of motion vectors, Frame Motion Vectors and Field Motion Vectors, will be estimated for each macroblock. In the case of Frame Motion Vectors, one motion vector will be generated in each direction per macroblock, which corresponds to a 16x16 pels luminance area. For the case of Field Motion Vectors, two motion vectors per macroblock will be generated for each direction, one for each of the fields. Each vector corresponds to a 16x8 pels luminance area.

The algorithm uses two steps. First a full search algorithm is applied on original pictures with full pel accuracy. Second a half pel refinement is used, using the local decoded picture.

5.1.1 Full Search

A simplified Frame and Field Motion Estimation routine is listed below. In this routine the following relation is used:

(AE of Frame) = (AE of FIELD1) + (AE of FIELD2)

where AE represents a sum of absolute errors.

With this routine three vectors are calculated, MV_FIELD1, MV_FIELD2 and MV_FRAME.

```
Min_FIELD1 = MAXINT;
Min_FIELD2 = MAXINT;
for (y = -YRange; y < YRange; y++) {
    for (x = -XRange; x < XRange; x++) {
        AE_FIELD1 = AE_Macroblock(prediction_mb(x,y),
                                  lines_of_FIELD1_of_current_mb);
        AE_FIELD2 = AE_Macroblock(prediction_mb(x,y),
                                  lines_of_FIELD2_of_current_mb);
        AE_FRAME = AE_FIELD1 + AE_FIELD2;
        if (AE_FIELD1 < Min_FIELD1) {
            MV_FIELD1 = (x,y);
            Min_FIELD1 = AE_FIELD1;
        }
        if (AE_FIELD2 < Min_FIELD2) {
            MV_FIELD2 = (x,y);
            Min_FIELD2 = AE_FIELD2;
        }
        if (AE_FRAME < Min_FRAME) {
            MV_FRAME = (x,y);
            Min_FRAME = AE_FRAME;
        }
    }
}
```

The search is constrained to take place within the boundaries of the significant pel area. Motion vectors which refer to pixels outside the significant pel area are excluded.

5.1.2 Half pel search

The half pel refinement uses the eight neighbouring half-pel positions in the referenced corresponding local decoded field or frame which are evaluated in the following order:

```
1 2 3
4 0 5
6 7 8
```

where 0 represents the previously evaluated integer-pel position. The value of the spatially interpolated pels are calculated as follows:

S(x+0.5,y) = (S(x,y)+S(x+1,y))//2,
S(x ,y+0.5) = (S(x,y)+S(x,y+1))//2,
S(x+0.5,y+0.5) = (S(x,y)+S(x+1,y)+S(x,y+1)+S(x+1,y+1))//4.

where x, y are the integer-pel horizontal and vertical coordinates, and S is the pel value. If two or more positions have the same total absolute difference, the first is used for motion estimation.

NOTE: In field searches, the refence system is the correspondig field. In a field the line distance is 1.

5.2 Motion Compensation

Motion compensation is performed differently for field coding and for frame coding. General formulas for frame and field coding are listed below.

Forward motion compensation is performed as follows:

$$S(x, y) = S_1(x + FMV_x(x, y), y + FMV_y(x, y))$$

Backward motion compensation is performed as follows:

$$S(x, y) = S_{M+1}(x + BMV_x(x,y), y + BMV_y(x,y))$$

Temporal interpolation is performed by averaging.

$$S(x,y) = (S_1(x + FMV_x(x,y) , y + FMV_y(x,y)) + S_{M+1}(x + BMV_x(x,y), y + BMV_y(x,y)))//2$$

where FMV is the forward motion compensated macroblock, thus making reference to a 'previous picture', and BMV is the backward motion compensated macroblock, making reference to a 'future picture'.

A displacement vector for the chrominance is derived by halving the component values of the corresponding MB vector, using the formula from CD 11172, section .....:

```
right_for = (recon_right_for / 2) >> 1;
down_for = (recon_down_for / 2) >> 1;
right_half_for = recon_right_for/2 - 2*right_for;
down_half_for = recon_down_for/2 - 2*down_for;
```

5.2.1 Frame Motion Compensation

In frame prediction macroblocks there is one vector per macroblock. Vectors measure displacements on a frame sampling grid. Therefore an odd-valued vertical displacement causes a prediction from the fields of opposite parity. Vertical half pixel values are interpolated between samples from fields of opposite parity. Chrominance vectors are obtained directly by using the formulae above. The vertical motion compensation is illustrated in figure 5.1.

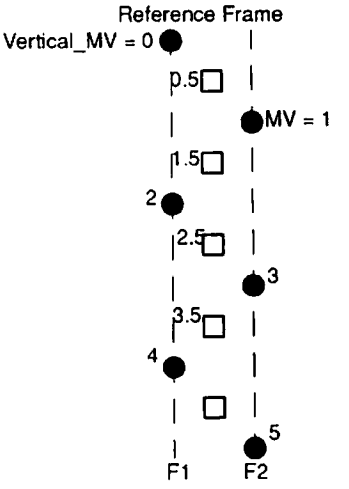
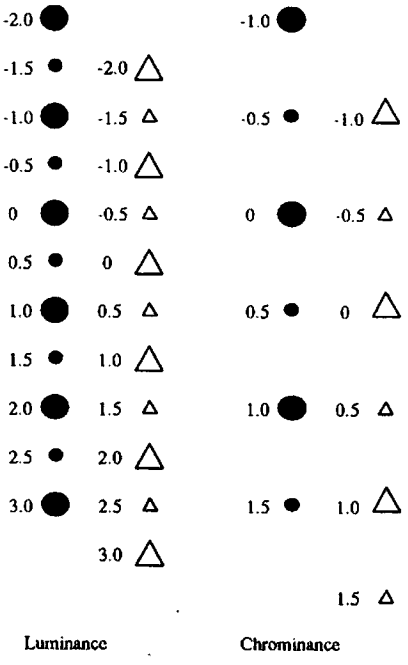


Figure 5.1: Frame Motion Compensation

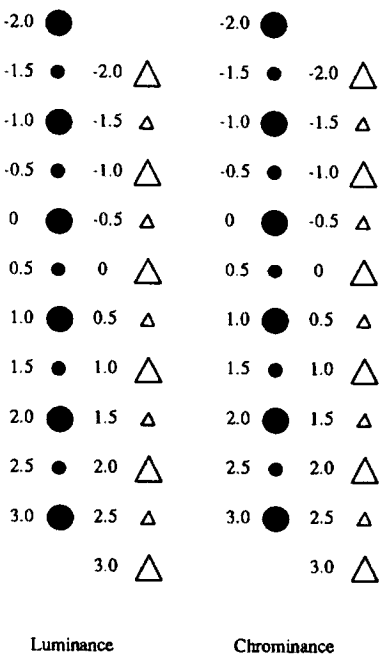
5.2.2 Field Motion Compensation

Field-based MV is expressed in the very same way as frame-based vectors would be if the source (reference) field and the destination field were considered as "frames" (see Figure).

Considering that in each field, lines are numbers 1.0, 2.0, 3.0, ... (1 is the top line of the field), if the pel located at line "n" of the destination field is predicted from line "m" of the reference field, the vertical coordinate of the field vector is "n-m".
Note: "m" and "n" are expressed in units of one vertical half-pel in the field.
When necessary, **motion_vertical_field_select** (one bit) will be transmitted to identify the selected field.



4:2:0 format



4:2:2 format and 4:4:4 format

5.2.2.1. Chrominance Field-based MV

In 4:2:0 sequences :

- The vertical coordinate of the chroma Field-based MV is derived by dividing by 2 the vertical coordinate of the luma Field-based MV, as done in MPEG-1.
- The horizontal coordinate of the chroma MV (Field-based or Frame-based) is derived by dividing by 2 the horizontal coordinate of the luma MV, as done in MPEG-1.

In 4:2:2 sequences :

- The vertical coordinate of the Field-based MV for chroma is equal to the vertical coordinate of the luma Field-based MV.
- The horizontal coordinate of the chroma MV (Field-based or Frame-based) is derived by dividing by 2 the horizontal coordinate of the luma MV, as done in MPEG-1.

In 4:4:4 sequences :

- The horizontal (resp. vertical) coordinate of MV for chroma is equal to the horizontal (resp. vertical) coordinate of the luma MV.

6 MODES AND MODE SELECTION

In section 6.1, a coding structure with different frame modes is introduced. Within each frame, macroblocks may be coded in several ways, thus aiming at high coding efficiency. The MB modes for intra, predicted and interpolated frames are shown in 6.2 to 6.4.

6.1 Picture types

Pictures are coded in several modes as a trade-off between coding efficiency and random accessibility. There are basically three picture coding modes, or picture types:

- I-pictures: intra coded pictures.
- P-pictures: forward motion-compensated prediction pictures.
- B-pictures: motion compensated interpolation pictures.

Although, in principle, freedom could be allowed for choosing one of these methods for a certain picture, for the Test model a fixed, periodic structure is used depending on the respective picture.

Every N-th frame of a sequence starting with the first frame is coded as intra frame i.e. frames 1, N+1, etc. (see Fig. 5.1). Following every M-th frame in between (within a Group of Pictures) is a predicted frame coded relative to the previous predicted or intra frame. The interpolated frames are coded with reference to the two closest previous and next predicted or intra frames. In this TM, M=3 and N=15 for 29.97 Hz and M=3 and N=12 for 25 Hz.

The following parameters are currently to be used for most of the core- experiments:

Frame rate	25 Hz	29.97 Hz
N	12	15
M	3	3

Coding modes available for predicted and interpolated frames are described in detail in the following paragraphs.

6.2 Macroblock types in an intra picture

In an I-picture the following macroblock types are provided:

- Intra
- Intra with modified quantizer

See also table B.2a

Independent of the macroblock type a compatible prediction and field/frame DCT coding indications are given in the bit stream, see also chapter 9, the macroblock layer section.

The macroblock type selection is done in the following order:

- Compatible prediction
- Field/frame DCT coding
- Modified quantizer

6.3 Macroblock types in a predicted picture

In predicted frames the following macroblocks types can be distinguished:
- Motion compensation coded
- No motion compensation coded
- Motion compensation not coded
- Intra
- Motion compensation coded with modified quantizer
- No Motion compensation coded with modified quantizer
- Intra with modified quantizer

See also table B.2b

Independent of the macroblock type a compatible prediction, field/frame DCT coding and field/frame motion vector prediction indications are given in the bit stream, see also chapter 9, the macroblock layer section.

Macroblock type selection is done in the following order:
- MC/no MC - Field/Frame prediction
- Intra/Inter
- Compatible prediction
- Modified quantizer
- Field/frame DCT coding
- Coded/not Coded

6.4 Macroblock types in an interpolated picture

In interpolated frames the following macroblock types are provided:
- Interpolate, not Coded
- Interpolate, Coded
- Backwards, not Coded
- Backwards, Coded
- Forwards, not Coded
- Forwards, Coded
- Intra
- Interpolate with modified quantizer
- Backwards with modified quantizer
- Forwards with modified quantizer
- Intra with modified quantizer

Independent of the macroblock type a compatible prediction, field/frame DCT coding and field/frame motion vector prediction indications are given in the bit stream, see also chapter 9, the macroblock layer section.

Macroblock type selection is done in the following order:
- Interpolative/Forwards/Backwards - Frame/Field prediction
- Intra/Inter
- Compatible prediction
- Modified quantizer
- Field/frame DCT coding
- Coded/not Coded

6.5 Selection criteria

The following rules apply to interlace and compatible bit streams. A subset of them apply to scalable bit streams as well. For details refer to Annex D.

6.5.1 Motion Compensation/No Motion Compensation - Frame/Field

For P-frames, the decision of selecting the Frame Motion Vector or the Field Motion Vector is by SE comparison of each error signal: If (SE of Frame) <= (SE of FIELD1 + SE of FIELD2) then the Frame Motion Vector is chosen.

The decision of MC/no MC will be SE based. If (SE of MC) < (SE of No MC) then MC mode.

6.5.2 Forward/Backward/Interpolative - Field/Frame prediction

Each B-frame Macroblock has possible Forward/Backward/Interpolated modes, and each mode has further Frame/Field prediction mode, so there totally 6 possible modes. All SE of the error signals of each mode will be calculated, and the mode with the least SE is chosen. In the case of two modes having the same SE, the mode with Frame prediction only will have higher priority, and also forward prediction will have higher priority than backward with interpolated mode the least priority.

6.5.3 Compatible prediction

When the experiment is not intended for compatible coding, this mode is not selected. When the experiment is intended for compatible coding the following criterion is used.

See appendix G.

6.5.4 Intra/Inter coding

The implementation of the intra/non-intra decision is based on the comparison of VAR and VAROR as computed in the following algorithm:

```
for (i = 1; i <= 16; i++) {
        for (j = 1; j <= 16; j++) {
                OR  = O(i,j);
                Dif  = OR - S(i,j);
                VAR  = VAR + Dif*Dif;
                VAROR=VAROR + OR*OR;
                MWOR =MWOR + OR;
        }
}
VAR = VAR/256;
VAROR=VAROR/256 - MWOR/256*MWOR/256;
```

Where: O(i,j) denotes the pixels in the original macroblock. S(i,j) denotes the pixels of the reconstructed macroblock, in the frame referred to by the motion vector. Full arithmetic precision is used. The characteristics of the decision are described in Fig. 6.1. (Non-intra decision includes the solid line in Fig. 6.1.)

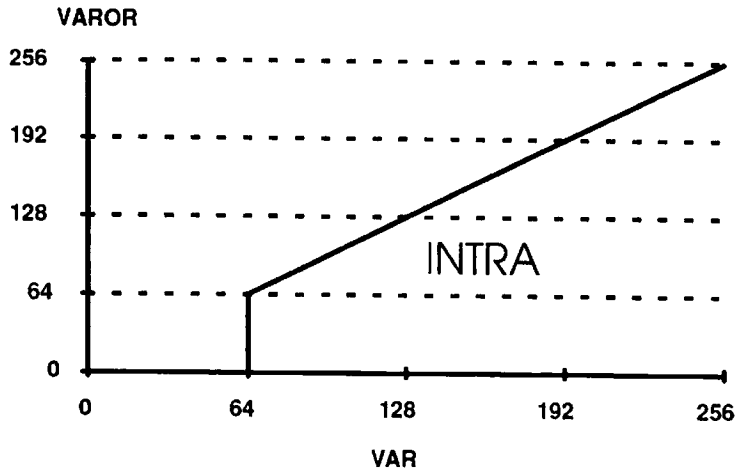


Figure 6.1: Characteristic Intra/Inter

6.5.5 Modified Quantizer

In chapter 10 "Rate control and quantization control" the algorithm for the calculation of the quantizer is given. If the quantizer given by this algorithm is not equal to the previous quantizer the modified quantizer indication is used.

6.5.6 Field/Frame DCT coding decisions

Field based DCT coding rather than frame based DCT coding is used if the following equation holds:
if (var_1 <= var_2 + offset)
Frame based DCT coding
else
Field based DCT coding

offset = 4096 for intra MB
offset = 0 for inter MB
Where var_1 and Var_2 are calculated with the following lines:

```
var_1 = 0;
var_2 = 0;
for (Pix = 0; Pix < 16; Pix++) {
    for (Line = 0; Line < 16; Line += 2) {
        Sum = O(Pix, Line) - O(Pix, Line+1);
        var_1 += (sum * sum);
    }
    for (Line = 0; Line < 16; Line += 4) {
        Sum_1 = O(Pix, Line) - O(Pix, Line+2);
        Sum_2 = O(Pix, Line+1) - O(Pix, Line+3);
        var_2 += (sum_1 * sum_1) + (sum_2 * sum_2)
    }
}
```

where O(Pix, Line) denotes a pel of the 16 x 16 macroblock to be transformed.

6.5.7 Coded/Not Coded

The choice of coded or not coded is a result of quantization. When all coefficients are zero then a block is not coded. A Macroblock is not coded if no block in it is coded, else it is coded.

7 TRANSFORMATION AND QUANTIZATION

While mode selection and local motion compensation are based on the macroblock structure, the transformation and quantization is based on 8*8 blocks.

Blocks are transformed with a 2-dimensional DCT as explained in Appendix A. Each block of 8*8 pixels thus results in a block of 8*8 transform coefficients. The DCT coefficients are quantized as described in sections 7.1 and 7.2.

7.1 Quantization of Intra Macroblocks

Intra frame DCT coefficients are quantized with a uniform quantizer without a dead-zone.

7.1.1 DC Coefficients

The quantizer step-size for the DC coefficient of the luminance and chrominance components is always 8. Thus, the quantized DC value, QDC, is calculated as:

$$QDC = dc \,//\, 8$$

where "dc" is the 11-bit unquantized mean value of a block.

7.1.2 AC Coefficients

AC coefficients ac(i,j) are first quantised by individual quantisation factors,

$$ac\sim(i,j) = (16 * ac(i,j)) \,//\, q(i,j)$$

where q(i,j) is the (i,j)th element of the Intra quantizer matrix given in figure 7.1. ac~(i,j) is limited to the range [-2048, 2047].

8	16	19	22	26	27	29	34
16	16	22	24	27	29	34	37
19	22	26	27	29	34	34	38
22	22	26	27	29	34	37	40
22	26	27	29	32	35	40	48
26	27	29	32	35	40	48	58
26	27	29	34	38	46	56	69
27	29	35	38	46	56	69	83

Figure 7.1 - Intra quantizer matrix

The step-size for quantizing the scaled DCT coefficients, ac~(i,j), is derived from the quantization parameter, mquant (also called quantiser_scale see section 9). Mquant is calculated for each macroblock by the algorithm defined in Section 10 and is stored in the bit stream in the slice header and, optionally, in any macroblock (see Section 9 for the syntax of the bit-stream and Section 10 for the calculation of mquant in the encoder).

The quantized level QAC(i,j) is given by:

$$QAC(i,j) = ac\sim(i,j) + sign(ac\sim(i,j))*(p * mquant \,//\, q) \,/\, (2*mquant)$$

and QAC(i,j) is limited to the range [-255..255].

PERHAPS WOULD NEED TO EXTEND TO [-2047, +2047] TO MEET REQUIREMENTS FOR CCIR 601 QUALITY, SEE ALSO APPENDIX Q
For this TM p=3, and q = 4.

7.2 Quantization Non Intra Macroblocks

Non-intra macroblocks in Predicted and Interpolated frames are quantized with a uniform quantizer that has a dead-zone about zero. A non-intra quantizer matrix, given in figure 7.2, is used.

16	17	18	19	20	21	22	23
17	18	19	20	21	22	23	24
18	19	20	21	22	23	24	25
19	20	21	22	23	24	26	27
20	21	22	23	25	26	27	28
21	22	23	24	26	27	28	30
22	23	24	26	27	28	30	31
23	24	25	27	28	30	31	33

Figure 7.2 - Non-intra quantizer matrix

The step-size for quantizing both the scaled DC and AC coefficients is derived from the quantization parameter, mquant. Mquant is calculated for each macroblock by the algorithm defined in Section 10. The following formulae describe the quantization process. Note that INTRA type macroblocks in predicted and interpolated frames are quantized in exactly the same manner as macroblocks in Intra-pictures (section 7.1) and not as described in this section.

$$ac\sim(i,j) = (16 * ac(i,j)) \,//\, q(i,j)$$

where:

q(i,j) is the non-intra quantizer matrix given in figure 7.2

$$\begin{aligned} QAC(i,j) &= ac\sim(i,j) \;/\; (2*mquant) && \text{IF mquant == odd} \\ &= (ac\sim(i,j)+1) \;/\; (2*mquant) && \text{IF mquant == even AND } ac\sim(i,j)<0 \\ &= (ac\sim(i,j)-1) \;/\; (2*mquant) && \text{IF mquant == even AND } ac\sim(i,j)>0 \end{aligned}$$

QAC (i,j) is limited to the range [-255..255].

PERHAPS WOULD NEED TO EXTEND TO [-2047, +2047] TO MEET REQUIREMENTS FOR CCIR 601 QUALITY, SEE ALSO APPENDIX Q.

7.3 Inverse Quantization

7.3.1 Intra-coded macroblocks

This section applies to all macroblocks in Intra-Frames and Intra macroblocks in Predicted and Interpolated Frames. Reconstruction levels, rec(i,j), are derived from the following formulae.

```
rec(i,j) = mquant * 2 * QAC(i,j) * wI(i,j) / 16

if (rec (i,j)  is an EVEN number && rec(i,j) > 0)
        rec (i,j) = rec(i,j) - 1

if (rec (i,j)  is an then number && rec(i,j) < 0)
        rec (i,j) = rec(i,j) + 1

if (QAC (i,j) == 0)
        rec (i,j) = 0

The DC term is special case
        rec (1,1) = 8 * QDC
```

Where:

mquant is the quantization parameter stored in the bit stream and calculated according to the algorithm in section 10.

rec(i,j) is limited to the range [-2048..2047].

7.3.2 Non-Intra-coded macroblocks

This section applies to all non-Intra macroblocks in Predicted and Interpolated Frames. Reconstruction levels, rec(i,j), are derived from the following formulae.

```
if (QAC(i,j) > 0)
        rec(i,j) =  (2 * QAC(i,j) + 1) * mquant * wN(i,j) / 16

if (QAC(i,j) < 0)
        rec(i,j) =  (2 * QAC(i,j) - 1) * mquant * wN(i,j) / 16

if (rec (i,j)  is an EVEN number && rec(i,) > 0)
        rec (i,j) = rec(i,j) - 1

if (rec (i,j)  is an EVEN number && rec(i,j) <0)
        rec (i,j) = rec(i,j) + 1

if (QAC (i,j) == 0)
        rec (i,j) = 0
```

rec(i,j) is limited to the range [-2048..2047].

8 CODING

This section describes the coding methods used to code the attributes and data in each macroblock. The overall syntax of the video coding is described in the following section, section 9.

The spatial position of each macroblock is encoded by a variable length code, the macroblock address (MBA). The use of macroblock addressing is described in section 8.1.

Macroblocks may take on one of a number of different modes. The modes available depend on the picture type. Section 6 describes the procedures used by the encoder to decide on which mode to use. The mode selected is identified in the bit stream by a variable length code known as MTYPE. The use of MTYPE is described in section 8.2.

The coding of motion vectors is addressed in section 8.3.

Some blocks do not contain any DCT coefficient data. To transmit which blocks of a macroblock are coded and which are non-coded, the coded block pattern (CBP) variable length code is used (see section 8.4).

The coefficients in a block are coded with VLC tables as described in section 8.5, 8.6, and 8.7.

For additional information about frequency and spatially scalable bit streams, refer to Annex D, G and I.

8.1 Macroblock Addressing

Relative addressing is used to code the position of all macroblocks in all frames. Macroblocks for which no data is stored are run-length encoded using the MBA; these macroblocks are called *skipped* macroblocks.

In Intra frames there are no skipped macroblocks. In predicted frames a macroblock is skipped if its motion vector is zero, all the quantized DCT coefficients are zero, and it is not the first or last macroblock in the slice. In interpolated frames, a macroblock is skipped if it has the same MTYPE as the prior macroblock, its motion vectors are the same as the corresponding motion vectors in the prior macroblock, all its quantized DCT coefficients are zero, and it is not the first or last macroblock in the slice.

A macroblock address (MBA) is a variable length code word indicating the position of a macroblock within a MB-Slice. The order of macroblocks is top-left to bottom-right in raster-scan order and is shown in Figure 4.2. For the first non-skipped macroblock in a macroblock slice, MBA is the macroblock count from the left side of the frame. For the Test Model this corresponds to the absolute address in figure 4.3. For subsequent macroblocks, MBA is the difference between the absolute addresses of the macroblock and the last non-skipped macroblock. The code table for MBA is given in Table B.1.

The macro_block_escape is a fixed bit-string "0000 0001 000" which is used when the difference macroblock_address_increment is greater then 33. It causes the value of macroblock_address_increment to be 33 greater than the value that will be decoded by subsequenct macroblock_escapes and the macroblock_address_increment codewords.

For example, if there are two macroblock_escape codewords preceding the macroblock_address_increment, then 66 is added to the value indicated by macroblock_address_increment.

An extra code word is available in the table for bit stuffing immediately after a macroblock slice header or a coded macroblock (MBA Stuffing). This code word should be discarded by decoders.

8.2 Macroblock Type

Each frame has one of the three modes:

1	Intra	(I-frames)
2	Predicted	(P-frames)
3	Interpolated	(B-frames)

For these three frame types different VLC tables for the Macroblock types are used. See table B.2a for Intra, table B.2b for predictive-coded pictures and table B.2c for bidirectionally predictive-coded pictures.

Methods for mode decisions are described in section 6. In macroblocks that modify the quantizer control parameter the MTYPE code word is followed by a 5-bit number giving the new value of the quantization parameter, mquant, in the range [1..31].

8.2.1 Compatible Prediction Flag

A one or two-bit codeword, **compatible_type,** immediately follows the MBTYPE VLC, if the bit stream is indicated as being compatible in the sequence header.

The definition of this codeword is in the definition of the Macroblock layer.

8.2.2 Field/Frame Coding Flag

A one-bit flag, **interlaced_macroblock_type**, immediately follows the MBTYPE VLC. If its value is 1 it indicates that the macroblock coefficient data is in field order as described in chapter 6. If its value is 0 it indicates that the macroblock coefficient data is in frame order.

8.2.3 Field/Frame Motion Compensation Flag

A one-bit flag, **interlaced_motion_type**, immediately follows the **interlaced_macroblock_type** flag. If its value is 1 it indicates that field-based motion prediction is used as described in section 5.2.2. If its value is 0 it indicates that frame-based motion prediction is used as described in section 5.2.1. If field-based prediction is used twice as many motion vectors are stored as are needed in the case of frame-based prediction.

8.3 Motion Vectors

Motion vectors for predicted and interpolated frames are coded differentially within a macroblock slice, obeying the following rules:

- Every forward or backward motion vector is coded relative to the last vector of the same type. Each component of the vector is coded independently, the horizontal component first and then the vertical component.

- The prediction motion vector is set to zero in the macroblocks at the start of a macroblock slice, or if the last macroblock was coded in the intra mode. (**Note**: that in predictive frames a No MC decision corresponds to a reset to zero of the prediction motion vector.)

- In interpolative frames, only vectors that are used for the selected prediction mode (MB type) are coded. Only vectors that have been coded are used as prediction motion vectors.

The VLC used to encode the differential motion vector data depends upon the range of the vectors. The maximum range that can be represented is determined by the **forward_f_code** and **backward_f_code** encoded in the picture header. (**Note**: in this Test Model the **full_pel_flag** is never set - all vectors have half-pel accuracy).

The differential motion vector component is calculated. Its range is compared with the values given in table 8.1 and is reduced to fall in the correct range by the following algorithm:

```
if (diff_vector < -range)
        diff_vector = diff_vector + 2*range;
else if (diff_vector > range-1)
        diff_vector = diff_vector - 2*range;
```

forward_f_code or backward_f_code	Range
1	16
2	32
3	64
4	128
5	256
6	512
7	1024

Table 8.1 Range for motion vectors

This value is scaled and coded in two parts by concatenating a VLC found from table B.4 and a fixed length part according to the following algorithm:

Let f_code be either the **forward_f_code** or **backward_f_code** as appropriate, and diff_vector be the differential motion vector reduced to the correct range.

```
if (diff_vector == 0) {
        residual = 0;
        vlc_code_magnitude = 0;
}
else {
        scale_factor = 1 << (f_code - 1);
        residual = (abs(diff_vector) - 1) % scale_factor;
        vlc_code_magnitude = (abs (diff_vector) - residual) / scale_factor;
        if (scale_factor != 1)
                vlc_code_magnitude += 1;
}
```

vlc_code_magnitude and the sign of diff_vector are encoded according to table B.4. The residual is encoded as a fixed length code using (f_code-1) bits.

For example to encode the following string of vector components (measured in half pel units)

3 10 30 30 -14 -16 27 24

The range is such that an f value of 2 can be used. The initial prediction is zero, so the differential values are:

3 7 20 0 -44 -2 43 -3

The differential values are reduced to the range -32 to +31 by adding or subtracting the modulus 64 corresponding to the forward_f_code of 2:

3 7 20 0 20 -2 -21 -3

These values are then scaled and coded in two parts (the table gives the pair of values to be encoded (vlc, residual)):

(2, 0) (4,0) (10, 1) (0, 0) (10, 1) (-1, 1) (-11, 0) (-2, 0)

The order in a slice is in raster scan order, except for Macroblocks coded in Field prediction mode, where the upper two luminance blocks vector is predicted from the preceding Macroblock and the two lower luminance blocks vector is predicted from

Prediction is changed according to Appendix J. $$$$ MERGE the text

8.4 Coded Block Pattern

There are three types for the coded block pattern, one for the 4:2:0, one the 4:2:2 coding modes and one for 4:4:4.

8.4.1 4:2:0

If MTYPE shows that the macroblock is not INTRA coded and all the coefficients of a block are zero after quantization, the block is declared to be not coded. If all six blocks in a macroblock are not coded, the macroblock is declared to be not coded. In all other cases the macroblock is declared to be coded.
If the MTYPE shows that the macroblock is INTRA all blocks are declared to be coded and the CBP code word is not used.
A pattern number defines which blocks within the MB are coded;
Pattern number = $32*P_1 + 16*P_2 + 8*P_3 + 4*P_4 + 2*P_5 + P_6$
where P_n is 1 if any coefficient is present for block n, else 0. Block numbering is given in Figure 4.4.
The pattern number is coded using table B.3 Coded Block pattern

8.4.2 4:2:2

When the picture format is 4:2:2, the pattern number is coded with an 8 bit FLC.

8.4.3 4:4:4

When the picture format is 4:4:4, use 12 bit FLC.

8.5 Intra frame Coefficient Coding

8.5.1 DC Prediction

PERHAPS WOULD NEED TO USE 9 BITS DC PRECISION TO MEET REQUIREMENTS FOR CCIR 601 QUALITY, SEE ALSO APPENDIX Q.

After the DC coefficient of a block has been quantized to 8 bits according to section 7.1.1, it is coded loss less by a DPCM technique. Coding of the luminance blocks within a macroblock follows the normal scan of figure 4.4. Thus the DC value of block 4 becomes the DC predictor for block 1 of the following macroblock. Three independent predictors are used, one each for Y, Cr and Cb.

At the left edge of a macroblock slice, the DC predictor is set to 128 (for the first block (luminance) and the chrominance blocks). At the rest of a macroblock slice, the DC predictor is simply the previously coded DC value of the same type (Y, Cr, or Cb).

At the decoder the original quantized DC values are exactly recovered by following the inverse procedure.

The differential DC values thus generated are categorised according to their "size" as shown in the table below.

DIFFERENTIAL DC (absolute value)	SIZE
0	0
1	1
2 to 3	2
4 to 7	3
8 to 15	4
16 to 31	5
32 to 63	6
64 to 127	7
128 to 255	8

Table 8.2 Differential DC size and VLC

The size value is VLC coded according to table B.5a (luminance) and B.5b (chrominance).

For each category enough additional bits are appended to the SIZE code to uniquely identify which difference in that category actually occurred (table 8.3). The additional bits thus define the signed amplitude of the difference data. The number of additional bits (sign included) is equal to the SIZE value.

DIFFERENTIAL DC	SIZE	ADDITIONAL CODE
-255 to -128	8	00000000 to 01111111
-127 to -64	7	0000000 to 0111111
-63 to -32	6	000000 to 011111
-31 to -16	5	00000 to 01111
-15 to -8	4	0000 to 0111
-7 to -4	3	000 to 011
3 to -2	2	00 to 01
-1	1	0
0	0	
1	1	1
2 to 3	2	10 to 11
4 to 7	3	100 to 111
8 to 15	4	1000 to 1111
16 to 31	5	10000 to 11111
32 to 63	6	100000 to 111111
64 to 127	7	1000000 to 1111111
128 to 255	8	10000000 to 11111111

Table 8.3. Differential DC additional codes

8.5.2 AC Coefficients

AC coefficients are coded as described in section 8.7.

8.6 Non-Intraframe Coefficient Coding

8.6.1 Intra blocks

Intra blocks in non-intra frames are coded as in intra frames. At the start of the macroblock, the DC predictors for luminance and chrominance are reset to 128, unless the previous block was also intra; in this case, the predictors are obtained from the previous block as in intra frames (section 8.5.1).

AC coefficients are coded as described in section 8.7.Transform coefficient data is always present for all 6 blocks in a macroblock when MTYPE indicates INTRA.

8.6.2 Non intra blocks

In other cases MTYPE and CBP signal which blocks have coefficient data transmitted for them. The quantized transform coefficients are sequentially transmitted according to the zig-zag sequence given in Figure 4.5.

8.6.3 Frequency scalable blocks

Coding of intra and non intra blocks in a frequency scalable bit stream is described in Annex D.

8.7 Coding of Transform Coefficients

The combinations of zero-run and the following value are encoded with variable length codes as listed in table B.5c to B.5f. End of Block (EOB) is in this set. Because CBP indicates those blocks with no coefficient data, EOB cannot occur as the first coefficient. Hence EOB does not appear in the VLC table for the first coefficient. Note that EOB is stored for all coded blocks.

The last bit 's' denotes the sign of the level, '0' for positive '1' for negative.

The most commonly occurring combinations of successive zeros (RUN) and the following value (LEVEL) are encoded with variable length codes. Other combinations of (RUN, LEVEL) are encoded with a 20-bit or 28-bit word consisting of 6 bits ESCAPE, 6 bits RUN and 8 or 16 bits LEVEL. For the variable length encoding there are two code tables, one being used for the first transmitted LEVEL in INTER and INTER + MC blocks, the second for all other LEVELs except the first one in INTRA blocks, which is encode as described in section 8.6.1. See table B.5g

DOCUMENT 408 PROPOSES MAXIMUM 24 BITS WORD FOR RUN/LEVEL ESCAPE AND EXTENDED RANGE

9 VIDEO MULTIPLEX CODER

In this section the video multiplex is explained. Unless specified otherwise the most significant bit occurs first. This is Bit 1 and is the left most bit in the code tables in this document.

9.1 Method of Describing Bit Stream Syntax

Each data item in the bit stream is in bold type. It is described by its name, its length in bits, and a mnemonic for its type and order of transmission.

The action caused by a decoded data element in a bit stream depends on the value of that data element and on data elements previously decoded. The following constructs are used to express the conditions when data elements are present, and are in normal type:

while (condition) { If the condition is true, then the group of data elements occurs next
data_element in the data stream. This repeats until the condition is not true.
}

do {
data_element The data element always occurs at least once.
} while (condition) The data element is repeated until the condition is not true.

if (condition) { If the condition is true, then the first group of data elements occurs
data_element next in the data stream.
}
else { If the condition is not true, then the second group of data elements
data_element occurs next in the data stream.
}

for (i = 0; i < n; i++) { The group of data elements occurs n times. Conditional constructs
data_element within the group of data elements may depend on the value of the
... loop control variable i, which is set to zero for the first occurrence,
} incremented to one for the second occurrence, and so forth.

As noted, the group of data elements may contain nested conditional constructs. For compactness, the { } are omitted when only one data element follows.

data_element [n] data_element [n] is the n+1th element of an array of data.

data_element [m..n] is the inclusive range of bits between bit m and bit n in the data_element.

While the syntax is expressed in procedural terms, it should not be assumed that this section implements a satisfactory decoding procedure. In particular, it defines a correct and error-free input bit stream. Actual decoders must include a means to look for start codes in order to begin decoding correctly, and to identify errors, erasures or insertions while decoding. The methods to identify these situations, and the actions to be taken, are not standardised.

Definition of bytealigned function

The function bytealigned () returns 1 if the current position is on a byte boundary, that is the next bit in the bit stream is the first bit in a byte.

Definition of nextbits function

The function nextbits () permits comparison of a bit string with the next bits to be decoded in the bit stream.

Definition of next_start_code function

The next_start_code function removes any zero bit and zero byte stuffing and locates the next start code.

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

9.2 Mnemonics

The following mnemonics are defined to describe the different data types used in the coded bit-stream.

bslbf Bit string, left bit first, where "left" is the order in which bit strings are written in the standard. Bit strings are written as a string of 1s and 0s within single quote marks, e.g. '1000 0001'. Blanks within a bit string are for ease of reading and have no significance.

uimsbf Unsigned integer, most significant bit first.

vlclbf Variable length code, left bit first, where "left" refers to the order in which the VLC codes are written in Annex B.

The byte order of multi-byte words is most significant byte first.

9.3 Specification of the Coded Video Bit stream Syntax

9.3.1 Start Codes

Start codes are reserved bit patterns that do not otherwise occur in the video stream. All start codes are byte aligned.

name	hexadecimal value
picture_start_code	00000100
slice_start_codes (including slice_vertical_position:	00000101 through 000001AF
reserved	000001B0
reserved	000001B1
user_data_start_code	000001B2
sequence_header_code	000001B3
sequence_error_code	000001B4
extension_start_code	000001B5
reserved	000001B6
sequence_end_code	000001B7
group_start_code	000001B8
system start codes (see note)	000001B9 through 000001FF
NOTE - system start codes are defined in Part 1 CD 11172	

The use of the start codes is defined in the following syntax description with the exception of the sequence_error_code. The sequence_error_code has been allocated for use by the digital storage media interface to indicate where uncorrectable errors have been detected.

9.3.1.1 Slice Start Codes - For Scalability and Compatibility

name	hexadecimal value
picture_start_code	00000100
slice_start_codes (including slice_vertical_position:	00000101 through 000001AF
slice_start_codes (note)	000001B0
sif_even_start_code	000001B1
user_data_start_code	000001B2
sequence_header_code	000001B3
sequence_error_code	000001B4
extension_start_code	000001B5
ccir_slice_start_code	000001B6
sequence_end_code	000001B7
group_start_code	000001B8
system start codes (see note)	000001B9 through 000001FF
NOTE - system start codes are defined in Part 1 CD 11172	

note: In the above table, slice_start_codes identify lowest spatial or frequency scal in the bitstream. Thus in a compatible coding scheme (G.2..G4.) they are used to identify sif_odd_slice or hhr_slice, whereas in frequency scalable schems or in spatial/frequency hybrid schemes they identify the lowest frequency scale present.

[Editors note: proper names need to be defined for sif_even_start_code and ccir_slice_start_code in order to reflect the generic coding approach of MPEG, this table needs to be mixed with the table in 9.3.1]

9.3.2 Video Sequence Layer

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**	**bslbf**
}		

9.3.3 Sequence Header

sequence_header() {		
sequence_header_code	**32**	**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_quantizer_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**
compatible	**3**	**uimsbf**
sscalable	**1**	**uimsbf**
fscalable	**1**	**uimsbf**
chroma_format	**2**	**uimsbf**
reserved	**1**	**uimsbf**
if (fscalable) {		
do {		
fscale_code	**8**	**uimsbf**
} while (nextbits != '00000111')		
end_of_fscales_code	**8**	**'00000111'**
}		
if (sscalable) {		
do {		
sscale_code	**8**	**uimsbf**
} while (nextbits != '00001111')		
end_of_sscales_code	**8**	**'00001111'**
}		
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()		
}		
}		

compatible -- This is a three-bit integer defined in the following table, which indicates if a prediction from another codec is available and the type of coding used in the other codec.

binary value	Standard
000	not compatible
001	MPEG1 compatible
010	H.261 compatible
011	MPEG2 compatible
100	reserved
...	reserved
111	reserved

Default value: 000

For scalable bit streams this integer is set to zero.

NOTE: **interlaced** IS CHANGED TO EXTENDED_SYNTAX

extended_syntax - This is NOT a syntax element, **extended_syntax** is set to 1 when the extension_start_code is found in the sequence_header, otherwise it is set to 0.

fscalable - This is a one-bit integer defined in the following table.

0	not fscalable
1	fscalable

Default value: 0

sscalable - This is a one-bit integer defined in the following table:

0	not sscalable
1	sscalable

Default value: 0

chroma_format - This is a 2 bit integer defined in the following table:

00	reserved
01	4:2:0
10	4:2:2
11	4:4:4

Default value: 01

fscale_code - This is an 8-bit integer that defines the DCT size for the scalable layers, i.e. DCT_size = 1<<scale_code. The values of this integer are:

fscale_code	
0	scale 1 layer
1	scale 2 layer
2	scale 4 layer
3	scale 8 layer
4	reserved
5	reserved
6	reserved
7	**end_of_fscales_code**
8	reserved
256	reserved

Example: The bit stream with fscale_codes 1, 2, 3, 3, 7 would be interpreted to mean that there are four layers, a scale-2, followed by a scale-4, followed by two scale-8 layers.

sscale_code - This is an 8-bit integer that defines the coding standard and compatibility if any for each spatial resolutioni layer. The DCT size for all spatial scales is 8. The values of this integer are:

sscale code	Feature of Spatial Scale
0	not compatible
1	MPEG-1 compatible sif odd
2	H@261 compatible cif
3	MPEG-1 compatible sif i
4	MPEG-1 compatible hhr
5	sif even coded with MPEG-1 compatible sif odd
6	ccir 601 coded with MPEG-1 compatible sif odd
7	ccir 601 coded with MPEG-1 compatible sif i
8	ccir 601 coded with sif even and MPEG-1 compatible sif odd
9	ccir 601 coded with MPEG-1 compatible hhr
10	reserved
11	reserved
@	@
@	@
@	@
15	end of sscale code
@	@
@	@
@	@
255	reserved

Example: A bit stream with sscale-codes 1,5,8,15 would be interpreted to mean that there are three spatial layers, MPEG-1 compatible SIF Odd, followed by SIF Even coded with respect to SIF Odd, followed by CCIR 601 coded with respect to both SIF Odd and SIF Even.

9.3.4 Group of Pictures Layer

THIS LAYER COULD BE REMOVED, AND USE PICTURE LAYER (OF INTRA PICTURES) TO SIMPLIFY SYNTAX, IT REMAINS FOR MPEG-1 FORWARD COMPATIBITY

group_of_pictures() {		
group_start_code	**32**	**bslbf**
time_code	**25**	
closed_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)		
}		

9.3.5 Picture Layer

```
picture() {
  picture_start_code                                          32          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
    picture_structure                                         2           uimsbf
    forward_reference_fields                                  2           uimsbf
    backward_reference_fields                                 2           uimsbf
    if ( chroma_format == "01" ) {    /* 4:2:0 */
      chroma_postprocessing_type                              1           uimsbf
    } else {
      reserved                                                1           uimsbf
    }
    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()
    if ( fscalable || sscalable ) {
      while ( nextbits() == slave_slice_start_code ) {
        slave_slice()
      }
    }
  } while ( nextbits() == slice_start_code )
}
```

picture_structure - This is a 2-bit integer defined in the table below.

11	Frame-Picture
01	Field 1 of a Field-Picture
10	Field 2 of a Field-Picture
00	reserved

Default value: 11

forward_reference_fields - This is a 2-bit integer defined in the table below.

11	Forward prediction from Field 1 and Field 2
01	Forward prediction only from Field 1
10	Forward prediction only from Field 2
00	No forward prediction in this Picture

Default value: ??, what is the meaning of this field for non-interlaced pictures

backward_reference_fields - This is a 2-bit integer defined in the table below.

11	Backward prediction from Field 1 and Field 2
01	Backward prediction only from Field 1
10	Backward prediction only from Field 2
00	No backward prediction in this Picture

Default value: ??, what is the meaning of this field for non-interlaced pictures

chroma_postprocessing_type - This is a 1bit integer that indicate how the chroma samples of a 4:2:0 Picture must be postprocessed for display. It is defined in the table below.

0	SIF interlaced (for interlaced Pictures)
1	SIF (for progressive Pictures)

Default value: 1

9.3.6 Slice Layer

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

9.3.6.1 Slave Slice Layer

slave_slice() {		
slave_slice_start_code	**32**	**bslbf**
quantizer_delta_magnitude	**5**	**uimsbf**
quantizer_delta_sign	**1**	**uimsbf**
dct_size	**8**	**uimsbf**
for (s=0; s<slice_size; s++)		
slave_macroblock(dct_size)		
}		

For frequency scalable bit streams, the following definitions apply: dct_size - 1, 2, 4, or 8.
quantizer_delta: This integer is added to all "quantizer_scale" values in the slice and macroblock layers, to derive the corresponding quantizer_scale values (mquant) in the slave_slice and slave_macroblock layers. For the Test Model this delta is zero.
quantizer_delta_magnitude: specifies the magnitude of "quantizer_delta".
quantizer_delta_sign: specifies the sign of "quantizer_delta".
slice_size: the total number of macroblocks in the slice layer (44).

9.3.7 Macroblock Layer

motion_vector () {		
motion_horizontal_code	**1-11**	**vlclbf**
if ((horizontal_f != 1) && (motion_horizontal_code != 0))		
motion_horizontal_r	**1-6**	**uimsbf**
motion_vertical_code	**1-11**	**vlclbf**
if ((vertical_f != 1) && (motion_vertical_code != 0))		
motion_vertical_r	**1-6**	**uimsbf**
}		

forward_field_motion_vector () {		
if (forward_reference_field == "11")		
motion_vertical_field_select	**1**	**uimsbf**
motion_vector()	...	...
}		

backward_field_motion_vector () {		
if (backward_reference_field == "11")		
motion_vertical_field_select	**1**	**uimsbf**
motion_vector()	...	...
}		

COMMENT : IN CASE OF DUAL-FIELD PREDICTION AND "DUAL-PRIME" CORE EXPERIMENT,, THE SYNTAX ALLOWS ARBITRARY REFERENCE FIELD SELECTION FOR EACH VECTOR. THIS FLEXIBILITY MAY BE NOT NECESSARY

motion_vertical_field_select - This a 1 bit defined as follows :

0	prediction comes from Reference Field 1
1	prediction comes from Reference Field 2

forward_motion_vectors () {		
if (motion_vector_count == 1) {		
if (mv_format == frame) {		
forward_motion_vector()	...	...
} else {		
forward_field_motion_vector()	...	...
}	...	...
} else {		
forward_field_motion_vector_1()	...	...
forward_field_motion_vector_2()	...	...
}		
}		

backward_motion_vectors () {		
if (motion_vector_count == 1) {		
if (mv_format == frame) {		
backward_motion_vector()	...	...
} else {		
backward_field_motion_vector()	...	...
}	...	...
} else {		
backward_field_motion_vector_1()	...	...
backward_field_motion_vector_2()	...	...
}		
}		

coded_block_pattern () {		
if (chroma_format == "01") /* 4:2:0 */		
coded_block_pattern_420	**3-9**	**vlclbf**
else if (chroma_format == "10") /* 4:2:2 */		
coded_block_pattern_422	**8**	**uimsbf**
else /* 4:4:4 */		
coded_block_pattern_444	**12**	**uimsbf**
}		

macroblock() {		
while (nextbits() == '0000 0001 111')		
macroblock_stuffing	**11**	**vlclbf**
while (nextbits() == '0000 0001 000')		
macroblock_escape	**11**	**vlclbf**
macroblock_address_increment	**1-11**	**vlclbf**
macroblock_type	**1-6**	**vlclbf**
if (compatible) {		
if (picture_structure == "11") { /* Frame-picture */		
compatible_type	**2**	**uimsbf**
} else {		
compatible_type	**1**	**uimsbf**
}		
}		
if (extended_syntax) { /* not MPEG-1 syntax */		
if (macroblock_motion_forward ||		
macroblock_motion_backward)		
if (picture_structure == "11") { /* Frame-Picture */		
frame_motion_type	**2**	**uimsbf**
} else {		
field_motion_type	**2**	**uimsbf**
}		
}		
if (picture_structure == "11") /* Frame-Picture */		
&& ((macroblock_intra || macroblock_pattern))		
dct_type	**1**	**uimsbf**
}		
if (macroblock_quant)		
quantizer_scale	**5**	**uimsbf**
if (macroblock_motion_forward)		
forward_motion_vectors()	**...**	**...**
if (macroblock_motion_backward)		
backward_motion_vectors()	**...**	**...**
if (macroblock_pattern)		
coded_block_pattern()	**...**	**...**
for (i=0; i<block_count; i++)		
if (scalable) {		
scaled_block(i)		
}		
else {		
block(i)		
}		
}		
if (picture_coding_type == 4)		
end_of_macroblock	**1**	**"1"**
}		

compatible_type - This is a one or two-bit integer defined in the following tables.

binary value	Prediction	compatible_type[0,1,2,3,4,5]
00	Non Compatible	0,0,0,0,0,0
01	FIELD1 Compatible prediction	1,1,0,0,1,1
10	FIELD2 Compatible prediction	0,0,1,1,1,1
11	Field1&2 Compatible prediction	1,1,1,1,1,1

For picture_structure == frame_structure

binary value	Prediction	compatible_type[0,1,2,3,4,5]
0	Non Compatible	0,0,0,0,0,0
1	Compatible prediction	1,1,1,1,1,1

For picture_structure == field_structure

compatible_type[i] - This flag is used to indicate the compatible prediction per block, see also the block layer.

dct_type - This is a one-bit integer indicating whether the macroblock is frame DCT coded or field DCT coded. If this is set to "1", the macroblock is field DCT coded.

field_motion_type - This is a 2-bit code indicating the macroblock motion prediction, defined in the following table:

code	prediction type	**motion_vector_count**	**mv_format**
00	Field-based prediction	1	field
01	Dual-field prediction	2	field
10	Simplified FAMC	1	frame
11	CORE EXPERIMENTS	...	...

frame_motion_type - This is a 2-bit code indicating the macroblock motion prediction, defined in the following table:

code	prediction type	**motion_vector_count**	**mv_format**
00	Field-based prediction	2	field
01	Frame-based prediction	1	frame
10	Simplified FAMC	1	frame
11	CORE EXPERIMENTS	...	...

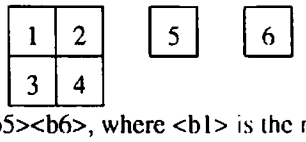
motion_vector_count - This is NOT a syntax element. **motion_vector_count** is derived from **field_motion_type** or **frame_motion_type** as indicated in the tables.

mv_format- This is NOT a syntax element. **mv_format** is derived from **field_motion_type** or **frame_motion_type** as indicated in the tables. **mv_format** indicates if the motion vector is a field-motion vector or a frame-motion vector. **mv_format** is used in the syntax of the motion vectors.and in the process of motion vector prediction.

block_count - This is *not* a syntax element. **block_count** is derived from chroma_format as follows :

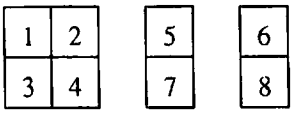
chroma_format	**block_count**
4:2:0	6
4:2:2	8
4:4:4	12

The 6 blocks in a 4:2:0 MB are numbered in the following order :



In this case, ***cbp*** is a 6-bit integer represented as : <b1><b2><b3><b4><b5><b6>, where <b1> is the most significant bit. <b*n*> is set to 1 when block *n* is coded.

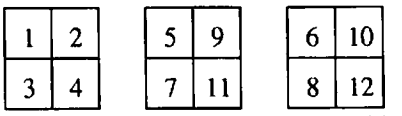
The 8 blocks in a 4:2:2 MB are numbered in the following order :



In this case, ***cbp*** is a 8-bit integer represented as : <b1><b2><b3><b4><b5><b6><b7><b8>, where <b1> is the most significant bit. <b*n*> is set to 1 when block *n* is coded.

NOTE THAT ORDER HAS BEEN CHANGED FOR 4:2:2 FOR EASIER HARDWARE IMPLEMENTATION.

The 12 blocks in a 4:4:4 MB are in the following order :



In this case, ***cbp*** is a 12-bit integer represented as : <b1>...<b12>, where <b1> is the most significant bit. <b*n*> is set to 1 when block *n* is coded.

9.3.7.1 Slave Macroblock Layer

```
slave_macroblock(dct_size) {
  for (i=0; i<6; i++) {
    slave_block[i, dct_size]
  }
}
```

9.3.8 Block Layer

NOTE : 9-BIT DC PRECISION AND EXTENDED RANGE OF QAC COEFFICIENT WOULD IMPLY CHANGES IN THE SYNTAX OF BLOCK LAYER

block(i) {		
if (pattern_code[i]) {		
if (macroblock_intra) {		
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 {		
dct_coeff_first	**2-28**	**vlclbf**
}		
if (picture_coding_type != 4) {		
while (nextbits() != '10')		
dct_coeff_next	**3-28**	**vlclbf**
end_of_block	**2**	**"10"**
}		
}		
}		

9.3.8.1 Scaled Block Layer

For scalable bit streams the following syntax extensions apply:

```
scaled_block(i) {
  if (pattern_code[i]) {
    if (macroblock_intra) {
      if (i<4) {
        dct_dc_size_luminance                    2-7    vlclbf
        if (dct_dc_size_luminance != 0)
          dct_dc_differential                    1-8    vlclbf
      }
      else {
        dct_dc_size_chrominance                  2-8    vlclbf
        if (dct_dc_size_chrominance != 0)
          dct_dc_differential                    1-8    vlclbf
      }
    }
    if (dct_size > 1) {
      while ((nextbits() != eob_code) && more_coefs)
        next_dct_coef(dct_size)                  2-16   vlclbf
      if (more_coefs)
        end_of_block                             2-16   vlclbf
      }
    }
  }
}

slave_block [i,dct_size] {
  if (pattern_code[i]) {
    while ((nextbits() != eob_code) && more_coefs) {
      next_dct_coef(dct_size)                    2-16   vlclbf
    }
    if (more_coefs) {
      end_of_block                               2-16   vlclbf
    }
  }
}
```

pattern_code(i) - For slave_blocks, this code is the same as that of the correlated scaled_block in the slice layer.

more_coefs - more_coefs is true if we have not already decoded the last coefficient in the block of DCT coefficients except that, for the 8x8 slave_slice, more_coefs is always true (this is to retain compatibility with MPEG-1 style of coding 8x8 blocks, which always includes an end_of_block code).

eob_code - An end_of_block Huffman code specified in the appropriate resolution scale VLC table.

next_dct_coef - DCT coefficient coded by run/amplitude or run/size VLCs. The VLC table used depends on "dct_size", as explained in Annex D.

10 RATE CONTROL AND QUANTIZATION CONTROL

This section describes the procedure for controlling the bit-rate of the Test Model by adapting the macroblock quantization parameter. The algorithm works in three-steps:

1 **Target bit allocation**: this step estimates the number of bits available to code the next frame. It is performed before coding the frame.

2 **Rate control**: by means of a "virtual buffer", this step sets the reference value of the quantization parameter for each macroblock.

3 **Adaptive quantization**: this step modulates the reference value of the quantization parameter according to the spatial activity in the macroblock to derive the value of the quantization parameter, mquant, that is used to quantize the macroblock.

Step 1 - Bit Allocation

Complexity estimation

After a frame of a certain type (I, P, or B) is encoded, the respective "global complexity measure" (X_i, X_p, or X_b) is updated as:

$$X_i = S_i\ Q_i\ , \qquad X_p = S_p\ Q_p, \qquad X_b = S_b\ Q_b$$

where S_i, S_p, S_b are the number of bits generated by encoding this frame and Q_i, Q_p and Q_b are the average quantization parameter computed by averaging the actual quantization values used during the encoding of the all the macroblocks, including the skipped macroblocks.

Initial values

$X_i = 160 * \mathbf{bit\_rate} / 115$
$X_p = 60 * \mathbf{bit\_rate} / 115$
$X_b = 42 * \mathbf{bit\_rate} / 115$

bit_rate is measured in bits/s.

Picture Target Setting

The target number of bits for the next frame in the Group of pictures (T_i, T_p, or T_b) is computed as:

$$T_i = \max\left\{ \frac{R}{1 + \frac{N_p X_p}{X_i K_p} + \frac{N_b X_b}{X_i K_b}}\ ,\ \mathbf{bit\_rate} / (8*\mathbf{picture\_rate}) \right\}$$

$$T_p = \max\left\{ \frac{R}{N_p + \frac{N_b K_p X_b}{K_b X_p}}\ ,\ \mathbf{bit\_rate} / (8*\mathbf{picture\_rate}) \right\}$$

$$T_b = \max\left\{ \frac{R}{N_b + \frac{N_p K_b X_p}{K_p X_b}}\ ,\ \mathbf{bit\_rate} / (8*\mathbf{picture\_rate}) \right\}$$

Where:

K_p and K_b are "universal" constants dependent on the quantization matrices. For the matrices specified in sections 7.1 and 7.2 $K_p = 1.0$ and $K_b = 1.4$.

R is the remaining number of bits assigned to the GROUP OF PICTURES. R is updated as follows:

After encoding a frame , $R = R - S_{i,p,b}$

Where is $S_{i,p,b}$ is the number of bits generated in the picture just encoded (picture type is I, P or B).

Before encoding the first frame in a GROUP OF PICTURES (an I-frame):

R = G + R
G = **bit_rate** * N / **picture_rate**
N is the number of frames in the GROUP OF PICTURES.

At the start of the sequence R = 0.

N_p and N_b are the number of P-frames and B-frames remaining in the current GROUP OF PICTURES in the encoding order.

I	B	B	P	B	B	P	B	B	P	B	B
						R-bits					
						$N_p = 2$					
						$N_b = 4$					

Figure 10.1 - GROUP OF PICTURES structure

Step 2 - Rate Control

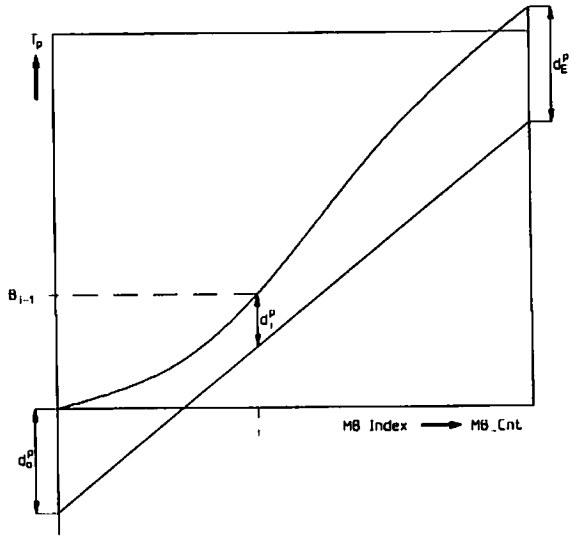


Figure 10.2 : Rate Control for P-frames

Before encoding macroblock j (j >= 1), compute the fullness of the appropriate virtual buffer:

$$d_j^i = d0^i + B_{j-1} - \frac{T_i(j-1)}{MB\_cnt}$$

or

$$T_p(j-1)$$

$$d_j^p = d_0^p + B_{j-1} - \frac{}{MB\_cnt}$$

or

$$d_j^b = d_0^b + B_{j-1} - \frac{T_b(j-1)}{MB\_cnt}$$

depending on the picture type.

where

d_0^i, d_0^p, d_0^b are initial fullnesses of virtual buffers - one for each frame type.

B_j is the number of bits generated by encoding all macroblocks in the frame up to and including j.

MB_cnt is the number of macroblocks in the frame.

d_j^i, d_j^p, d_j^b are the fullnesses of virtual buffers at macroblock j- one for each frame type.

The final fullness of the virtual buffer (d_j^i , d_j^p, d_j^b: j = MB_cnt) is used as d_0^i, d_0^p, d_0^b for encoding the next frame of the same type.

Next compute the reference quantization parameter Q_j for macroblock j as follows:

$$Q_j = \frac{d_j * 31}{r}$$

where the "reaction parameter" r is given by

$$r = 2 * \textbf{bit\_rate} / \textbf{picture\_rate}$$

and d_j is the fullness of the appropriate virtual buffer.

The initial value for the virtual buffer fullness is:

$$d_0^i = 10 * r/31$$
$$d_0^p = K_p\, d_0^i$$
$$d_0^b = K_b\, d_0^i$$

Step 3 - Adaptive Quantization

Compute a spatial activity measure for the macroblock j from the four luminance sub-blocks using the intra (i.e. original) pixels values:

$$act_j = 1 + \min_{sblk=1,4}(var\_sblk)$$

where

$$var\_sblk = \frac{1}{64} \sum_{k=1}^{64} (P_k - P\_mean)^2$$

$$P\_mean = \frac{1}{64} \sum_{k=1}^{64} P_k$$

and P_k are the pixel values in the original 8*8 block.

Normalise act_j:

$$N\_act_j = \frac{2 * act_j + avg\_act}{act_j + 2 * avg\_act}$$

avg_act is the average value of act_j the last frame to be encoded. On the first frame, avg_act = 400.

Obtain $mquant_j$ as:

$$mquant_j = Q_j * N\_act_j$$

where Q_j is the reference quantization parameter obtained in step 2. The final value of $mquant_j$ is clipped to the range [1..31] and is used and coded as described in sections 7, 8 and 9 in either the slice or macroblock layer.

Known Limitations

- Step 1 does not handle scene changes efficiently.

- Step 3 does not work well on highly interlaced material, since the entire method uses frame macroblocks.

- A wrong value of avg_act is used in step 3 after a scene change.

- VBV compliance is not guaranteed.

APPENDIX A: Discrete Cosine Transform (DCT)

The 2-dimensional DCT is defined as:

$$F(u,v) = (1/4)\,C(u)\,C(v) \sum_{x=0}^{7}\sum_{y=0}^{7} f(x,y)\cos\left\{\frac{(2x+1)u\pi}{16}\right\}\cos\left\{\frac{(2y+1)v\pi}{16}\right\},$$

with u, v, x, y = 0, 1, 2, ... 7

where x, y = spatial coordinates in the pel domain
u, v = coordinates in the transform domain

$$C(u), C(v) = \begin{cases} 1/\mathrm{SQRT}(2) & \text{for } u, v = 0 \\ 1 & \text{otherwise} \end{cases}$$

The inverse DCT (IDCT) is defined as:

$$f(x,y) = (1/4) \sum_{u=0}^{7}\sum_{v=0}^{7} C(u)\,C(v)\,F(u,v)\cos\left\{\frac{(2x+1)u\pi}{16}\right\}\cos\left\{\frac{(2y+1)v\pi}{16}\right\},$$

The input to the forward transform and output from the inverse transform is represented with 9 bits. The coefficients are represented in 12 bits. The dynamic range of the DCT coefficients is (-2048, ..., 2047).

Accuracy Specification

The 8 by 8 inverse discrete transform shall conform to IEEE Draft Standard Specification for the Implementations of 8 by 8 Inverse Discrete Cosine Transform, P1180/D2, July 18, 1990. Note that Section 2.3 P1180/D2 "Considerations of Specifying IDCT Mismatch Errors" requires the specification of periodic intra-picture coding in order to control the accumulation of mismatch errors. The maximum refresh period requirement for this standard shall be 132 pictures, the same as indicated in P1180/D2 for visual telephony according to CCITT Recommendation H.261 (see Bibliography).

APPENDIX B: VARIABLE LENGTH CODE TABLES

Introduction

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

B.1 Macroblock Addressing

Table B.1. 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
0000 0101 11	16	0000 0011 001	32
		0000 0011 000	33
		0000 0001 111	macroblock_stuffing
		0000 0001 000	macroblock_escape

B.2 Macroblock Type

Table B.2a. Variable length codes for macroblock_type in intra-coded pictures (I-pictures).

VLC code	macroblock_ quant	macroblock_ motion_ forward	macroblock motion_ backward	macroblock_ pattern	macroblock_ intra
1	0	0	0	0	1
01	1	0	0	0	1
001	0	0	0	0	0

The last code-word applies only to compatible coders.

Table B.2b. Variable length codes for macroblock_type in predictive-coded pictures (P-pictures).

VLC code	macroblock_ quant	macroblock_ motion_ forward	macroblock motion_ backward	macroblock_ pattern	macroblock_ intra
1	0	1	0	1	0
01	0	0	0	1	0
001	0	1	0	0	0
00011	0	0	0	0	1
00010	1	1	0	1	0
00001	1	0	0	1	0
000001	1	0	0	0	1
0000001	0	0	0	0	0

The last code-word applies only to compatible coders.

Table B.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
0000001	0	0	0	0	0

The last code-word applies only to compatible coders.

Table B.2d. Variable length codes for macroblock_type in DC intra-coded pictures (D-pictures).

VLC code	macroblock_ quant	macroblock_ motion_ forward	macroblock_ motion_ backward	macroblock_ pattern	macroblock_ intra
1	0	0	0	0	1

B.3 Macroblock Pattern

NOTE THAT DOCUMENTS 375 AND 332 DESCRIBE TWO POSSIBLE CHANGES FOR ENCODING OF CBP

Table B.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		

B.4 Motion Vectors

Table B.4. Variable length codes for motion_horizontal_forward_code, motion_vertical_forward_code, motion_horizontal_backward_code, and motion_vertical_backward_code.

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

B.5 DCT Coefficients

TABLE B5A AND B5B CAN BE EXTENDED ONE LINE FOR 9-BIT INTRA DC (DOCUMENT 408), SEE APPENDIX Q.

TABLE B5G CAN BE SIMPLIFIED AND EXTENDED TO COVER WIDER RANGE (DOCUMENT 408), SEE APPENDIX Q.

IMPLEMENTATION GROUP WOULD WELCOME AN ALTERNATIVE TO THE "TRICK" FOR THE FIRST COEFFICIENT

Table B.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 B.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

Table B.5c. Variable length codes for dct_coeff_first and dct_coeff_next.

dct_coeff_first and dct_coeff_nex variable length code (NOTE1)	run	level
10	end_of_block	
1 s (NOTE2)	0	1
11 s (NOTE3)	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

NOTE1 - The last bit 's' denotes the sign of the level, '0' for positive '1' for negative.
NOTE2 - This code shall be used for dct_coeff_first.
NOTE3 - This code shall be used for dct_coeff_next.

Table B.5d. Variable length codes for dct_coeff_first and dct_coeff_next (continued).

dct_coeff_first and dct_coeff_nex variable length code (NOTE)	run	level
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
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

NOTE - The last bit 's' denotes the sign of the level, '0' for positive, '1' for negative.

Table B.5e. Variable length codes for dct_coeff_first and dct_coeff_next (continued).

dct_coeff_first and dct_coeff_nex variable length code (NOTE)	run	level
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
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

NOTE - The last bit 's' denotes the sign of the level, '0' for positive, '1' for negative.

Table B.5f. Variable length codes for dct_coeff_first and dct_coeff_next (continued).

dct_coeff_first and dct_coeff_nex variable length code (NOTE)	run	level
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
NOTE - The last bit 's' denotes the sign of the level, '0' for positive, '1' for negative.		

Table B.5g. Encoding of run and level following escape code as a 20-bit fixed length code (-127 <= level <= 127) or as a 28-bit fixed length code (-255 <= level <= -128, 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

APPENDIX C : Video Buffering Verifier

Constant rate coded video bit streams shall meet constraints imposed through a Video Buffering Verifier (VBV) defined in Section C.1.

The VBV is a hypothetical decoder which is conceptually connected to the output of an encoder. Coded data is placed in the buffer at the constant bit rate that is being used. Coded data is removed from the buffer as defined in Section C.1.4, below. It is a requirement of the encoder (or editor) that the bit stream it produces will not cause the VBV to either overflow or underflow.

C.1 Video Buffering Verifier

1. The VBV and the video encoder have the same clock frequency as well as the same picture rate, and are operated synchronously.

2. The VBV has a receiving buffer of size B, where B is given in the vbv_buffer_size field in the sequence header.

3. The VBV is initially empty. It is filled from the bit stream for the time specified by the vbv_delay field in the video bit stream.

4. All of the data for the picture which has been in the buffer longest is instantaneously removed. Then after each subsequent picture interval all of the data for the picture which (at that time) has been in the buffer longest is instantaneously removed. Sequence header and group of picture layer data elements which immediately precede a picture are removed at the same time as that picture. The VBV is examined immediately before removing any data (sequence header data, group of picture layer or picture) and immediately after each picture is removed. Each time the VBV is examined its occupancy shall lie between zero bits and B bits where B is the size of the VBV buffer indicated by vbv_buffer_size in the sequence header.

This is a requirement on the video bit stream including coded picture data, user data and all stuffing.

To meet these requirements the number of bits for the (n+1)'th coded picture d_{n+1} (including any preceding sequence header and group of picture layer data elements) must satisfy:

$$d_{n+1} > B_n + 2R/P - B$$

$$d_{n+1} <= B_n + R/P$$

where:

$n >= 0$
B_n is the buffer occupancy just after time t_n
R = bit rate
P = number of pictures per second
t_n is the time when the n'th coded picture is removed from the VBV buffer

$$$$ insert figure here

Figure C.1 VBV Buffer Occupancy

APPENDIX D: Scalability Syntax, Encoder, and Decoder Description

D.1 INTRODUCTION

The syntax for frequency domain scalable bit streams has been detailed in Chapter 9. This appendix describes the operation of the Test Model encoder and decoder for frequency domain scalability experiments, in the spirit of a delta with respect to the non-scalable Test Model. Where core experiments depart from these descriptions, the departures are documented in the descriptions of the core experiments themselves.

The frequency domain scalability syntax extensions enable the implementation of hierarchical pyramid and sub-band schemes in the frequency (DCT) domain. Although the syntax is flexible, the Test Model corresponds to a three layer case with the following resolutions:

1. CCIR 601	704x480(576)	(Scale-8)
2. SIF	352x240(288)	(Scale-4)
3. QSIF	176x120(144)	(Scale-2)

Figures D.1 and D.2 are block diagrams of the Test Model encoder and decoder for the frequency domain scalability experiments.

figure to be inserted

Figure D.1: Frequency domain scalable encoder.

figure to be inserted

Figure D.2: Frequency domain scalable decoder.

One fundamental characteristic of a frequency domain scalable bit-stream is that all side information, including sequence, picture, and slice headers; and macroblock attributes, such as macroblock address increment and coded block pattern are placed in the lowest layer. Virtually no side information appears in any of the other layers (the exception is that each slice in the other layers starts with a small header.

In the non-scalable Test Model, a **macroblock** is subdivided into six 8x8 **blocks** of luminance and chrominance information (assuming a 4:2:0 format), and each **block** is coded using the 8x8 Discrete Cosine Transform (DCT). The frequency domain scalability extensions allow coding of multiple video resolutions by implementing a hierarchy of layers corresponding to subsets of the 8x8 DCT coefficients. In decoding to a target resolution, an inverse DCT of a size that matches the resolution of the target layer is used. Thus, for the SIF layer, a 4x4 IDCT is used. Because of the use of 4x4 coefficients and DCTs, this layer is also referred to as scale-4. For the QCIF layer, a 2x2 IDCT is used (scale-2). In addition, motion vectors used at a particular resolution are derived from the high resolution vectors by scaling according to the size of the DCT at that layer, relative to that at the highest resolution layer.

It is possible to implement sub-band and pyramidal encoders that operate in the frequency domain, using the same syntax. In the case of a pyramidal encoder, DCT data for coefficients in a higher layer are coded differentially with respect to the corresponding

DCT coefficients from the next lower layer. In the sub-band case, each frequency coefficient appears in one and only one layer.

The following sections describe those parts of the non-scalable Test Model which require modification or clarification to in order to implement the basic scalable encoder and decoder. The organization of this appendix follows that of the main body of the Test Model, to the extent possible.

D.2 LAYERED STRUCTURE OF VIDEO DATA AND MULTIPLEXING OF FREQUENCY SCALES

The video data in the scalable Test Model is layered in exactly the same fashion as that of the non-scalable Test Model, with the following additions. Data for various scales is multiplexed at the level of the **slice layer**, according to the syntax in Chapter 9. In a scalable bit-stream, the **slice layer** contains data for the lowest layer; each slice is followed by a variable number of **slave_slices** that contain data for the other layers. The syntax for the scalable **slice layer** is compatible with that of the non-scaled **slice layer**. The size of the DCT to be used at the lowest scale is given as **extra_information_slice** in each **slice** header; for the other scales, it is given in the corresponding **slave_slice** header.

The **macroblock** syntax for a scalable bit-stream is also compatible with the corresponding syntax for a non-scalable bit-stream. To preserve the MB structure, all MB attributes are coded with the lowest scale, using the standard **macroblock** syntax (except that "**scaled_blocks**" are coded in the lowest resolution layer instead of "**blocks**"). Thus MB addresses, types, motion vectors, and coded block patterns, are coded together with the information for the lowest scale "**scaled_blocks**". The low resolution "**scaled_blocks**" are coded, however, using a modification of the usual "**block**" syntax, as shown in Chapter 9.

The **slave_slices** contain **slave_macroblocks** which, in turn, contain additional DCT coefficient data. These data increase the spatial and/or amplitude resolution of the **scaled_blocks** in the MBs. Because the MB attributes of the slice layer are inherited by the **slave_macroblocks**, DCT data is included in **slave_macroblocks** only for blocks marked by the corresponding **Coded Block Pattern (CBP)**.

In summary, each slice of the non-scalable Test Model is replaced in the scalable Test Model by a slice containing 2x2 DCT data for QCIF resolution pictures and all the MB side information for all layers; and two slave_slices, the first containing 4x4 DCT data for SIF resolution pictures, and the second, 8x8 DCT data for CCIR 601 resolution pictures.

D.3 MOTION ESTIMATION AND COMPENSATION

Motion vectors for the highest resolution layer are estimated using the non-scalable Test Model full-search technique. For lower resolution motion compensation, the high resolution motion vectors are scaled down in magnitude, but the full precision available at high resolution is retained. Thus, scale-4 luminance motion vectors have 1/4-pixel precision, and scale-2 vectors have 1/8-pixel precision. A low resolution vector is computed by multiplying the corresponding high resolution vector by the ratio of the block sizes in the target resolution and the high resolution layers. Bi-linear interpolation for motion-compensated prediction is performed according to the following method, which represents a generalization of the method used in TM1 to do motion compensation of non-scalable video (these formulas apply to scale-8).

Consider the definitions:

Rs(x,y) = pixel x,y of reference image at scale-s.
Ps(x,y) = pixel x,y of motion-compensated prediction image at scale-s.

For scale-s (s = 1, 2, 4, 8), the following luminance vector computations are done, given (n,m), the integers representing the half-pixel motion at scale-8.

(D.1) shift = 1 + log2(8/s)
(D.2) factor = 16/s

(D.3) xs = n >> shift
(D.4) fxs = n - factor * xs
(D.5) ys = m >> shift
(D.6) fys = m - factor * ys

Then, the prediction image at pixel (x,y) of scale-s Ps(x,y) is given by

Ps(x,y) = ((factor-fys)/factor) * [p1] + (fys/factor) * [p2],

where

p1 = ((factor-fxs)/factor) * Rs(x+xs, y+ys) + (fxs/factor) * Rs(x+xs+1, y+ys),

and

p2 = ((factor-fxs)/factor) * Rs(x+xs, y+ys+1) + (fxs/factor) * Rs(x+xs+1, y+ys+1).

Overall, there is division by factor**2, which can be implemented as a logical shift right by 2* log2(factor) = 2*(4 - log2(scale)) bits.

Chrominance motion-compensation is the same, except that at scale-s, equations (D.1-6) are applied using s/2 instead of s.

D.4 MODES AND MODE SELECTION

The same picture types and coding modes used in the non-scalable Test Model are used in the frequency domain scalable extension. However, the selection criteria are not completely identical. The **motion compensation/no motion compensation, forward/backward/interpolated** and **intra/inter coding** decisions are made by examining the appropriate high resolution signals, in the same manner as for the non-scalable case. However, the determination whether a block is **coded** or **not coded** is made by examining the quantized DCT coefficients of **all** layers. Furthermore, only the frame modes of DCT coding and motion compensation are used, so no frame/field tests are required. The modified quantizer scheme of Test Model 1, Chapter 10 is sufficient as is, since the same rate control strategy is used in the frequency domain scalable extension.

D.5 UPWARD DCT COEFFICIENT PREDICTION

In a pyramidal frequency domain scalable codec, DCT coefficients in a low resolution layer are used to predict the corresponding coefficients in the next (higher) resolution layer. Thus, the 2x2 DCT coefficients of the Test Model **slice** layer are used to predict the upper-left 2x2 coefficients of the corresponding coefficients in the 4x4 **slave_slice** layer. Similarly, the 4x4 coefficients of the first **slave_slice** layer are used to predict the upper-left 4x4 coefficients of the corresponding coefficients in the 8x8 **slave_slice** layer. After computing the prediction differences, these differences together with the new (not predicted) coefficient data are coded using a zig-zag scan pattern as shown in Figure 4.8 of Test Model 1.

D.6 TRANSFORMATION AND QUANTIZATION

D.6.2 Transformation

In order to facilitate upward DCT coefficient prediction, DCT and IDCT formulas with non-standard normalization are used, as defined in Table D.1.

Table D.1: DCT definitions for frequency domain scalability.

Scale	Forward DCT	Inverse DCT
2	4*DCT(2x2)	IDCT(2x2)/4
4	2*DCT(4x4)	IDCT(4x4)/2

8	DCT(8x8)	IDCT(8x8)

Here, DCT(NxN) and IDCT(NxN) are the standard 2-dimensional definitions for the transforms of size NxN.

D.6.2 Upward Coefficient Prediction and Quantization

In the scalable Test Model, the same quantization matrix is used for all resolution scales. For the 2x2 layer, the upper left 2x2 elements of the 8x8 quantizer matrix are used. Similarly, for the 4x4 layer, the upper left 4x4 elements of the 8x8 matrix are used. Prior to quantization of a particular scale, the quantized and rebuilt coefficients from the next lower scale are used as a prediction of the corresponding coefficients in the current scale. Because of these features and the fact that, in the scalable Test Model extension, the same MQUANT value is applied to the coefficients of all layers, the quantized DCT coefficients in the 2x2 and 4x4 layers could be obtained by simply extracting the appropriate coefficients from the corresponding set of quantized 8x8 coefficients.

(We emphasize that, the generality exists to use at least different MQUANT values in different layers, as is done in some of the Core Experiments. See section D.6.3. It may also be desirable to use different quantization matrices in different scales, in some applications. However, this is not supported in the current Test Model.)

Aside from the above considerations, quantization in the scalable extensions is identical to that in the non-scalable Test Model. In general, to rebuild the DCT coefficients for a target scale, the following steps are needed:

1. inverse quantization of the DCT coefficients of the target scale and all lower scales using the appropriate quantizer scale factor "MQUANT_s" (MQUANT_2 for scale 2, MQUANT_4 for scale 4, and MQUANT_8 for scale 8), and quantization matrix, and

2. for appropriate coefficients, summation of the corresponding coefficient resulting from the previous step (upward DCT coefficient prediction).

Note that this method differs from that in the scalable extension of Test Model 1.

D.6.3 BANDWIDTH CONTROL OF RESOLUTION LAYERS

The Slave_slice layer specification includes a **quantizer_delta** parameter. For each MB in a **slave_slice** layer, this delta is specified with reference to the corresponding quantizer scale factor used in the Slice layer, as explained in section 9.3.6. This parameter is used to derive the "MQUANT" values used in the slave layers. For the basic encoder, **quantizer_delta** is always zero.

D.7 CODING

D.7.1 DCT COEFFICIENT CODING

Coding of DCT coefficients is done using a combination of two-dimensional run/amplitude VLC tables and FLC escape tables. The FLC coding is exactly as in Test Model 1. However, there is one VLC table for each scale. The VLC tables are specified below.

Table I: Scale 2 run/amplitude variable length codes

Run	Amplitude	length	Codeword
EOB		1	1

0	1	2	01s
0	2	4	0011s
1	1	4	0010s
0	3	5	00011s
2	1	6	000101s
0	4	6	000100s
1	2	7	0000111s
0	5	7	0000110s
0	6	7	0000101s
1	3	8	00001001s
3	1	8	00001000s
0	7	8	00000111s
ESCAPE		8	00000110
0	8	8	00000101s
2	2	8	00000100s
0	9	9	000000111s
1	4	9	000000110s
0	10	9	000000101s
0	11	9	000000100s
1	5	10	0000000111s
0	12	10	0000000110s
0	13	10	0000000101s
1	6	10	0000000100s
2	3	10	0000000011s
0	14	11	00000000101s
0	15	11	00000000100s
1	7	11	00000000011s
0	16	11	00000000010s
0	17	12	000000000011s
1	8	12	000000000010s
0	18	12	000000000001s

Table II: Scale 4 run/amplitude variable length codes

Run	Amp	Len	Codeword	Run	Amp	Len	Codeword
0	1	2	11s	1	5	10	0000001110s
EOB		2	10	14	1	10	0000001101s
1	1	3	011s	13	1	10	0000001100s
0	2	4	0101s	0	11	10	0000001011s
2	1	4	0100s	3	3	10	0000001010s
3	1	5	00111s	6	2	10	0000001001s
0	3	5	00110s	0	12	10	0000001000s
4	1	5	00101s	1	6	10	0000000111s
1	2	6	001001s	15	1	11	00000001101s
5	1	6	001000s	2	4	11	00000001100s
0	4	6	000111s	0	13	11	00000001011s
6	1	7	0001101s	0	14	11	00000001010s
0	5	7	0001100s	4	3	11	00000001001s
2	2	7	0001011s	1	7	11	00000001000s
9	1	7	0001010s	5	3	11	00000000111s
7	1	7	0001001s	0	15	11	00000000110s
1	3	7	0001000s	9	2	12	000000001011s
8	1	7	0000111s	3	4	12	000000001010s
0	6	7	0000110s	0	16	12	000000001001s
3	2	8	00001011s	1	8	12	000000001000s

0	7	8	00001010s		0	17	12	000000000111s
10	1	8	00001001s		7	2	12	000000000110s
0	8	8	00001000s		8	2	12	000000000101s
1	4	9	000001111s		2	5	13	0000000001001s
4	2	9	000001110s		0	18	13	0000000001000s
11	1	9	000001101s		1	9	13	0000000000111s
5	2	9	000001100s		0	19	13	0000000000110s
0	9	9	000001011s		4	4	13	0000000000101s
2	3	9	000001010s		1	10	13	0000000000100s
ESCAPE		9	000001001		5	4	13	0000000000011s
12	1	9	000001000s		0	20	13	0000000000010s
0	10	10	0000001111s		0	21	13	0000000000001s

Table III: Scale 8 run/amplitude variable length code table

Run	Amp	LEN	CODEWORD	Run	Amp	LEN	CODEWORD
EOB		2	11	31	1	10	0000010001s
0	1	3	101s	29	1	11	00000100001s
1	1	3	100s	2	3	11	00000100000s
2	1	4	0111s	19	2	11	00000011111s
10	1	5	01101s	10	5	11	00000011110s
3	1	5	01100s	8	3	11	00000011101s
4	1	5	01011s	13	4	11	00000011100s
9	1	5	01010s	0	7	11	00000011011s
5	1	5	01001s	3	3	11	00000011010s
0	2	5	01000s	1	4	11	00000011001s
8	1	6	001111s	28	1	11	00000011000s
6	1	6	001110s	21	2	11	00000010111s
7	1	6	001101s	14	5	11	00000010110s
11	1	6	001100s	20	3	11	00000010101s
14	1	6	001011s	9	4	11	00000010100s
12	1	6	001010s	0	8	11	00000010011s
20	1	6	001001s	12	3	11	00000010010s
13	1	7	0010001s	13	5	12	000000100011s
10	2	7	0010000s	15	2	12	000000100010s
0	3	7	0001111s	48	1	12	000000100001s
15	1	7	0001110s	37	1	12	000000100000s
35	1	7	0001101s	4	3	12	000000011111s
19	1	7	0001100s	10	6	12	000000011110s
21	1	7	0001011s	14	6	12	000000011101s
1	2	8	00010101s	56	1	12	000000011100s
ESCAPE		8	00010100	5	3	12	000000011011s
16	1	8	00010011s	1	5	12	000000011010s
0	4	8	00010010s	0	9	12	000000011001s
9	2	8	00010001s	11	3	12	000000011000s
10	3	9	000100001s	2	4	12	000000010111s
2	2	9	000100000s	13	6	12	000000010110s
14	2	9	000011111s	3	4	12	000000010101s
22	1	9	000011110s	35	3	12	000000010100s
8	2	9	000011101s	40	1	12	000000010011s
13	2	9	000011100s	16	2	12	000000010010s
3	2	9	000011011s	12	4	12	000000010001s
17	1	9	000011010s	9	5	12	000000010000s
23	1	9	000011001s	14	7	12	000000001111s

18	1	9	000011000s	24	2	12	000000001110s
0	5	9	000010111s	8	4	12	000000001101s
24	1	9	000010110s	6	3	12	000000001100s
20	2	9	000010101s	10	7	13	0000000010111s
25	1	9	000010100s	7	3	13	0000000010110s
34	1	9	000010011s	38	1	13	0000000010101s
1	3	10	0000100101s	0	10	13	0000000010100s
14	3	10	0000100100s	39	1	13	0000000010011s
4	2	10	0000100011s	50	1	13	0000000010010s
36	1	10	0000100010s	17	2	13	0000000010001s
26	1	10	0000100001s	20	4	13	0000000010000s
11	2	10	0000100000s	13	7	13	0000000001111s
10	4	10	0000011111s	0	11	13	0000000001110s
35	2	10	0000011110s	41	1	13	0000000001101s
12	2	10	0000011101s	14	8	13	0000000001100s
5	2	10	0000011100s	46	1	13	0000000001011s
33	1	10	0000011011s	49	1	13	0000000001010s
13	3	10	0000011010s	1	6	13	0000000001001s
9	3	10	0000011001s	12	5	13	0000000001000s
27	1	10	0000011000s	25	2	13	0000000000111s
0	6	10	0000010111s	10	8	13	0000000000110s
32	1	10	0000010110s	4	4	13	0000000000101s
6	2	10	0000010101s	51	1	13	0000000000100s
7	2	10	0000010100s	2	5	13	0000000000011s
14	4	10	0000010011s	13	8	13	0000000000010s
30	1	10	0000010010s	19	3	13	0000000000001s

D.8 VIDEO MULTIPLEX CODER

See Test Model 2, Chapter 9.

D.9 RATE CONTROL AND QUANTIZATION CONTROL

Core Experiments that require methods other than those described in Test Model 1 contain descriptions.

Appendix F: Cell loss experiments

F.1 Cell loss

Cell loss can occur unpredictably in ATM networks. This document proposes a method of simulating cell loss. A specification for a packetized bit stream has been defined. A model of bursty cell loss is defined and analysed in order to allow the simulation of bursty cell loss. The proposed specification and model are simplified; no attempt is made to model actual ATM networks; the main objective of the model is to allow consistent simulation of the effects of cell loss on video coding.

F.1.1 Bit stream specification

The coded bit stream is packetized into 48 byte cells consisting of a four bit sequence number (SN), a four bit sequence number protection field (SNP) and 47 bytes of coded data. In the stored file each cell is preceded by a Cell Identification byte (CI). The syntax is as follows:

< CI ><SN><SNP>< 47 bytes of data >

The CI byte consists of the bit string '1011010' followed by the priority bit. The priority bit is set to '1' for low priority cells and '0' for high priority cells. The cell loss ratio for low priority cells may be different to that for high priority cells. SN is incremented by one after every cell. The sequence number protection is set to zero.

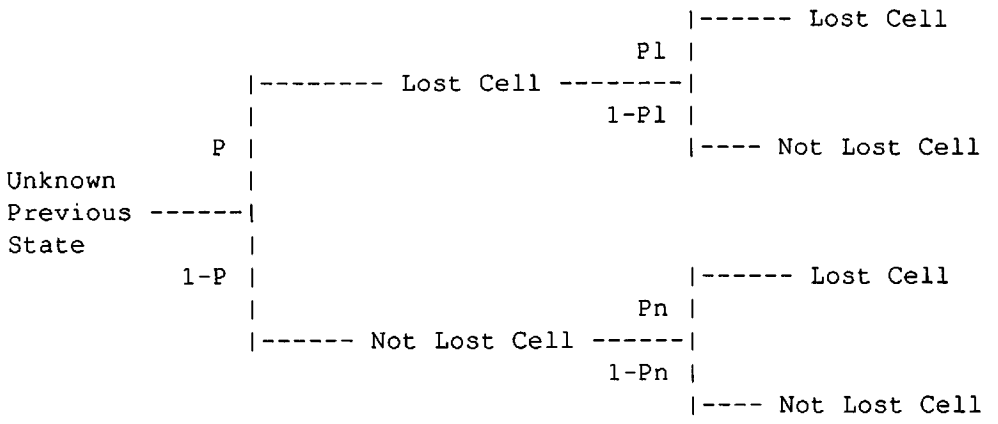
For a lost cell the cell is discarded.

F.1.2 Calculation of cell loss probabilities

This section outlines a method for determining whether any cell in a bit stream should be marked as lost. Cell loss is assumed to be random, with the probability of cell loss depending only on whether the previous cell of the same priority was lost.

Firstly the mean cell loss rate and the mean burst of consecutive cells lost is calculated from the probabilities of cell loss. These equations are then rearranged in order to express the cell loss probabilities in terms of the mean cell loss rate and the mean burst of consecutive cells lost.

The following notation is used. The probability that any cell is lost is given by P, the probability that a cell is lost given that the previous one was not lost is given by P_n and the probability that a cell is lost given that the previous one was lost is given by P_l. These probabilities are illustrated in the tree diagram below.



F.1.3 Calculation of mean cell loss rate

The mean cell loss probability is given by P. In this section a relationship between P, P_n and P_l is derived, as follows, by finding two equivalent expressions for the probability of a given cell being lost. A lost cell can occur in two ways: immediately after a cell has been lost and after a cell has been received. The probability that a cell is lost, P, is the sum of the probability that the cell is lost given the previous cell was lost multiplied by the probability that the previous cell was lost, $P * P_l$, and the probability that the cell is lost given the previous cell was not lost multiplied by the probability that the previous cell was not lost, $(1 - P) * P_n$. So,

$$P = P * P_l + (1 - P) * P_n$$

So

$$P = P_n / (1 - P_l + P_n) \quad (1)$$

F.1.4 Calculation of mean burst of consecutive cells lost

A burst of lost cells is defined as a sequence of consecutive cells all of which are marked as lost. It is preceded by and followed by one or more cells that are marked as not lost. The length of the burst of lost cells is defined as the number of cells in a burst that are marked as lost. The mean burst of consecutive cells lost is defined as the mean burst length. This number must always be greater than or equal to one.

A burst starts when a cell is lost after one or more cells have not been lost. The probability that this is a burst of length one is equal to the probability that the next cell is not lost, that is, $1 - P_l$. The probability that this is a burst of length two is equal to the probability that the next cell is lost and the one after that is not lost, that is, $P_l * (1 - P_l)$. The probability of a burst of length n is $P_l^{(n-1)} * (1 - P_l)$. The mean burst length, B, is therefore given by:

$$B = (1 - P_l) + 2 * P_l * (1 - P_l) + 3 * P_l^2 * (1 - P_l) + \ldots$$

Summing this series leads to the result:

$$B = 1 / (1 - P_l) \quad (2)$$

F.1.5 Calculation of cell loss probabilities

Rearranging equation (2) gives:

$$P_l = 1 - 1/B \quad (3)$$

Rearranging equation (1) gives:

$$P_n = P * (1 - P_l) / (1 - P)$$

Using equation (3) gives:

$$P_n = P / (B * (1 - P)) \quad (4)$$

F.1.6 Simulation of cell loss

Equations (3) and (4) allow the probabilities of cell loss to be calculated from the average cell loss rate and the mean length of bursts of lost cells. Cell loss can easily be simulated using these probabilities: assume that the first cell is received, then the probability that the next will be lost is given by P_n. The probability that a cell is lost is always P_n, unless the previous cell was also lost in which case the relevant probability is P_l.

A simulation of cell loss only needs a random number generator, the values of P_n and P_l and the knowledge of whether the previous cell of the same priority was lost or not. Pseudo-Pascal code to perform cell loss is given below. Random is a function that returns a random number between zero and one; its implementation is given below.

```
PreviousCellLost := FALSE;
Write('Enter mean cell loss rate and burst length');
Readln(P,B);
PL := 1 - 1/B
PN := P / (B * (1-P) )

For CellCount := 1 To NumberOfCells DO
  BEGIN
    CASE PreviousCellLost OF
      TRUE  :  IF Random < PL THEN CellLost := TRUE
                             ELSE CellLost := FALSE;
      FALSE :  IF Random < PN THEN CellLost := TRUE
                             ELSE CellLost := FALSE;
      END;
    Write(CellLost);
    PreviousCellLost := CellLost;
    END;
END.
```

If the priority bit is used then the cell loss generator must be implemented separately for each of the priorities.

F.1.7 Random number generation

To ensure the consistent simulation of cell loss, it is necessary to ensure that the same sequence of random numbers is generated by all simulations regardless of the machine or programming language used. This section describes a method for the generation of such random numbers.

Random numbers are generated by use of a 31 bit shift register which cycles pseudo-randomly through (2^31 - 1) states (the value of zero is never achieved). The shift operation is defined by the pseudo-Pascal code below.

```
DO 31 times
Begin
  Bit30  := (ShiftRegister & 2^30) DIV 2^30
  Bit25  := (ShiftRegister & 2^25) DIV 2^25
  ShiftRegister := (2*ShiftRegister MOD 2^31) + (Bit30 XOR Bit25);
End
```

To generate a random number, the shift register is first shifted as above and then divided by (2^31 - 1). It may be easier to use it as it is, and multiply the probabilities in the program above by (2^31 - 1).

A separate random number generator is used for low and high priority cell loss. For each, the shift register is initialised to a value of 1 and is then shifted 100 times. If this is not done, the first few random numbers will be small, leading to the loss of the first cells in the bit stream.

F.2 Parameters

This section suggests specific values of the parameters to allow consistent simulation of the effects of cell loss on video coding.

The cell loss experiment will use a mean cell loss rate of 1 in 1000 and a mean burst length of 2. Only low priority cells are lost. The following formula gives the value of P to use for low priority cells.

$$P = 10^{-3} \times \frac{\text{Total Bit rate}}{\text{Total bit rate - Bit rate for high priority cells}}$$

For example:

Total bit rate 4Mbits/s

High priority bit rate 2Mbits/s (50% of Total)

then the mean cell loss rate figure for the cell loss simulation program is 2×10^{-3}.

Other cell loss experiments at different cell loss rates can also be shown, cell loss rates such as 1 in 100.

For all experiments the following table should be completed.

		High priority bit rate	Low priority bit rate
1-layer			
2-layer	base		
	enhance		

Appendix G: Compatibility and spatial scalability

Scalability is achieved by spatial reduction in the pel and temporal domain. Compatibility is a specific implementation of the spatial scalabilty. The following section will describe several core experiments. An ad-hoc group will be formed, which will discuss results and will work on improvements.

A. EXPERIMENTS LIST

Expt. No.	Description	Organizations	Doc. No.	Picture structure
1(a)	MPEG-1 based Prediction of Prediction Error versus Prediction of Input	BT, Toshiba	257, 291	Frame Structure
1(b)	MPEG-1 based Prediction of Prediction Error versus Prediction of Input	PTT	293	Field Structure
1(c)	H.261 based Prediction of Prediction Error versus Prediction of Input	France Telecom	399	Frame Structure
2(a)	Field Structure CCIR-601 coding with SIF Fields Prediction and 3 layers	AT&T, Bellcore, Columbia Univ	354	Field Structure
2(b)	Field Structure CCIR-601 coding with SIF Fields Prediction and 2 layers	AT&T, Bellcore, Columbia Univ	354	Field Structure
3	Frame Structure CCIR-601coding with SIF Fields Prediction and 3 layers	AT&T, Bellcore, Columbia Univ	354	Frame Structure
4	Compatible Coding for Higher Picture Formats	BT	_	Frame Structure

B. GUIDELINES

Bitrates

Expt. No.	Number of Layers	Bitrate Layer 1 Mbit/s	Bitrate Layer 2 Mbit/s	Bitrate Layer 3 Mbit/s
1(a)	2	1.15	2.85	-
1(b)	2	1.15	2.85	-
1(c)	2	1.15	2.85	-
2(a)	3	1.15	0.85	2.0
2(b)	2	1.15	2.85	-
3	3	1.15	0.85	2.0
4	2	2.0	6.0	-

M, N

Compatible Standard	M,N for 30 Hz	M,N for 25 Hz
MPEG-1	M=3, N=15	M=3, N=12
H.261	M=1, N=∞	M=1, N=∞

MC Prediction Modes

Frame Structure	Field structure
Frame MC	Field MC
Field MC	Field submacro MC
FAMC	Dual Field MC

C. EXPERIMENT DETAILS

Expt 1(a) MPEG-1 SIF based Prediction of Prediction Error versus Prediction of Input in Frame Structure Pictures

This experiment compares two techniques, one that uses the "Prediction of Prediction Error" with that which uses the "Prediction of Input". The lower layer is SIF odd fields which are coded using MPEG-1. The upper layer is CCIR-601 frames coded using MPEG-2 Frame Structure coding and with optional compatible prediction from the lower layer. The Encoder/Decoder block diagrams are shown in MPEG Docs. 92/291 and 92/292. The syntax required for this experiment is shown in MPEG Doc 92/293.

The SIF layer (decoded error or decoded picture) is upsampled for compatible prediction of odd fields and/or even fields of CCIR-601 frames. On a macroblock basis, two-bits specify the compatible type employed.

Expt 1(b) MPEG-1 SIF based Prediction of Prediction Error versus Prediction of Input in Field Structure Pictures

This experiment compares two techniques, one that uses the "Prediction of Prediction Error" with that which uses the "Prediction of Input". The lower layer is SIF odd fields which are coded using MPEG-1. The upper layer is CCIR-601 fields coded using MPEG-2 Field Structure coding and with optional compatible prediction from the lower layer.

The Encoder/Decoder block diagrams are shown in MPEG Docs. 92/291 and 92/292. The syntax required for this experiment is shown in MPEG Doc 92/293.

The SIF layer (decoded error or decoded picture) is upsampled for compatible prediction of CCIR odd fields only. On a macroblock basis, one-bit specifies the compatible type employed.

Expt 1(c) H.261 SIF based Prediction of Prediction Error versus Prediction of Input in Frame Structure Pictures

This experiment compares two techniques , one that uses the "Prediction of Prediction Error" with that which uses the "Prediction of Input". The lower layer is SIF odd fields which are coded using H.261. The upper layer is CCIR-601 frames coded using MPEG-2 (H.26X) Frame Structure coding and with optional compatible prediction from the lower layer.

The Encoder/Decoder block diagrams are similar to the one's in MPEG Docs.92/291 and 92/292. The syntax required for this experiment is shown in MPEG Doc 92/293.

The SIF layer (decoded error or decoded picture) is upsampled for compatible prediction of CCIR odd and/or even fields. On a macroblock basis, two-bits specify the compatible type employed.

Expt 2(a) Field Structure CCIR-601 Coding with SIF Fields Prediction and 3 layers

This experiment allows three layers: SIF odd fields, SIF even fields and CCIR-601 fields. The SIF odd fields layer is coded using the MPEG-1 syntax. The SIF even fields layer is coded using the MPEG-2 field structure syntax and employ adaptive field/dual field/field submacroblock MC prediction. The CCIR-601 odd and even fields layer is coded using the MPEG-2 field structure syntax and employs adpative field/dual field/field submacroblock MC prediction as well as optional compatible prediction obtained from the corresponding reconstructed SIF odd or SIF even field (lower) layers.

The Encoder/Decoder block diagrams, syntax and slice multiplexing structure are specified in MPEG Doc 92/354 (Experiment G.2(i)).

Expt 2(b) Field Structure CCIR-601 Coding with SIF Fields Prediction and 2 Layers

This experiment allows two layers: SIF odd fields and CCIR-601 fields. The SIF even fields layer is not coded. The SIF odd fields layer is coded using the MPEG-1 syntax.

The CCIR-601 odd fields layer is coded using the MPEG-2 field structure syntax and employs adaptive field/dual field/field submacroblock MC prediction as well as optional compatible prediction obtained from the reconstructed SIF odd fields (lower) layer. The CCIR-601 even fields are also coded using the MPEG-2 field structure syntax but do not use any compatible prediction.

The Encoder/Decoder block diagrams, syntax and slice multiplexing structure are specified in MPEG Doc 92/354 (Experiment G.2(ii)).

Expt 3 Frame Structure CCIR-601 Coding with SIF Fields Prediction and 3 Layers

This experiment allows three layers: SIF odd fields, SIF even fields and CCIR-601 frames. The SIF odd fields layer is coded using the MPEG-1 syntax. The SIF even fields layer is coded using the MPEG-2 field structure syntax and employs adaptive field/dual field/field submacroblock MC prediction.

The CCIR-601 frames layer is coded using the MPEG-2 frame-structure syntax and employs adaptive frame/field/field adjusted MC prediction as well as optional compatible prediction obtained from combining reconstructed SIF odd and SIF evenfields layers.

The Encoder/Decoder block diagrams, syntax and slice multiplexing structure are specified in MPEG Doc 92/354 (Experiment G.4).

D. OTHER CONDITIONS

- Experiments 1, 2 and 3 employ interpolation filter with weights (1/2, 1, 1/2).
- When compatible prediction is selected for an Intar macroblock, the DC value is not differentially coded and all DC predictors are reset (as in Doc. 92/293).
- Non-intra Weighting matrix is employed for quantizing all compatible predicted macroblocks (as in Doc. 92/293).
- Expt. 1 requires syntax specified in MPEG Doc 92/293, while Expt. 2 and 3 requires syntax specified in MPEG Doc 92/354.
- For Expts. 2 and 3, the macroblock type table is extended to include compatible macroblock types as follows.

Additional MB types in I- Pictures

VLC code	macro_quant	cbp	compatible
001	0	1	1
0001	1	1	1
00001	0	0	1

Additional MB types in P- Pictures

VLC code	macro_quant	cbp	compatible
0000001	0	1	1
00000001	1	1	1
000000001	0	0	1

Additional MB types in B- Pictures

VLC code	macro_quant	cbp	compatible
0000001	0	1	1
00000001	1	1	1
000000001	0	0	1

E. EXTENDED RESOLUTION CODING

Expt 4 Compatible Coding for Higher Resolution Picture Formats

The purpose of this experiment is to start investigations into compatible coding of higher picture formats using spatial scalability approach. The essential feature to be investigated is coding performance when using interlaced format in both layers.

The initial experiments focus on two resolution layers: 1/4 of CCIR-601 interlaced (SIF_I) and CCIR-601 interlaced. However, the proposed scheme should be extendable to three resolution layers.

The Encoder/Decoder block diagrams and the syntax is described in the Appendix G of the test model as well as in MPEG Docs 92/291,92/293 and 92/354. A specific example of the encoder that shows compatible prediction derived from locally decoded lower layer picture is shown in MPEG Doc 92/291.

Conversion of CCIR-601 to SIF_I

For initial experiments the simple down conversion defined in Sec. 3.3.5 of MPEG Doc 92/160 (TM1) is used. It is recognized that a better choice of decimation filters is desirable; experimenters are suggested to propose alternatives.

Coding of SIF_I Layer

This is the lower layer and uses the MPEG-2 frame-structure syntax with compatible prediction switched off.

The coding parameters employed are N=12/15 (for 25/30 Hz), M=3 and bitrate is 2 Mbit/s.

Coding of CCIR-601 Layer

The compatible prediction is selected on a macroblock basis and is made from the corresponding 8x8 luminance block and each 4x4 chrominance block of the lower layer. This can be done either on locally decoded picture or the prediction error, the choice of these is the subject of another core experiment.

The 8x8 luminance block and the each of the 4x4 chrominance blocks is interpolated in the following way:

- upsample horizontally by 1:2 by (1/2,1,1/2) filter.
- vertical interpolate the lines of block belonging to field 1 by using (1/2,1,1/2) filter, whereas lines belonging to field 2 are interpolated by (3/4,1/4) and (1/4,3/4) filters.

The coding parameters employed are N=12/15 (for 25/30 Hz), M=3 and bitrate is 6 Mbit/s.

To display the lower resolution layer the conversion described in section 3.4.5 of MPEG Doc 92/160 can be used. Experimenters are encouraged to investigate other suitable alternatives.

If compatible coding with higher resolution pictures, for example, 4xCCIR-601 interlaced format is investigated the bit rates for the two layers should be increased to 5 Mbit/s for the CCIR-601 layer, and 15 Mbit/s for 4xCCIR-601 layer.

Compatible Block Header Syntax

```
block( i ) {
    if ( pattern_code[i] ) {
      if ( macroblock_intra ) {
        if (!compatible_type[i]) {
          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 {
        dct_coeff_first                              2-28       vlclbf
      }
      if ( picture_coding_type != 4 ) {
        while ( nextbits() != '10')
          dct_coeff_next                             3-28       vlclbf
        end_of_block                                 2          "10"
      }
    }
  }
```

By making this change in the block layer it is no longer necessary to transmit the DC coefficient with the differential DC code if a compatible prediction is made for this block.

When a compatible prediction is made the DC coefficient is likely to be coded efficiently in the compatible layer, which will give the prediction to the higher resolution layer.

Weighting matrix

Whenever a macroblock is coded with a compatible prediction only (this is called INTRA compatible) then it is more appropriate to code the block as a non intra block, and therefor the non-intra weighting matrix should be used for quantization.

DC predictors

DC predictors have to be reset for the INTRA compatible macroblocks.

G.1 Experiment 1

For this the conversion of section 3.3.2 is used, which will give SIF images decimated from the FIELD1 part of a sequence.

G.1.1 General Parameters

Global parameters for coding are:
M=3, N=12, for 25Hz
M=3, N=15 for 30Hz
Bit rate:
MPEG1 layer to be coded at 1.5Mbit/s
Total bit rate 4Mbit/s.

G.1.2 Coding of the SIF

The coding is performed entirely in line with MPEG1. For the encoder the TM is used except all non MPEG1 non compliant additions (frame based like SM3).

G.1.3 Coding

The coding is performed entirely in line with the TM, with the following additions:

G.1.3.1Compatible prediction method

On a macroblock basis a prediction of the error signal can be made after the subtraction. This prediction is generated from the corresponding coded 8*8 prediction error block of the lower layer (See figure G.1 and G.2). This 8*8 block is then upsampled to a 16*8 block by a [1/2, 1, 1/2] filter. A similar operation is perform on the chrominance 4*4 subblock to give a 8*4 prediction block.

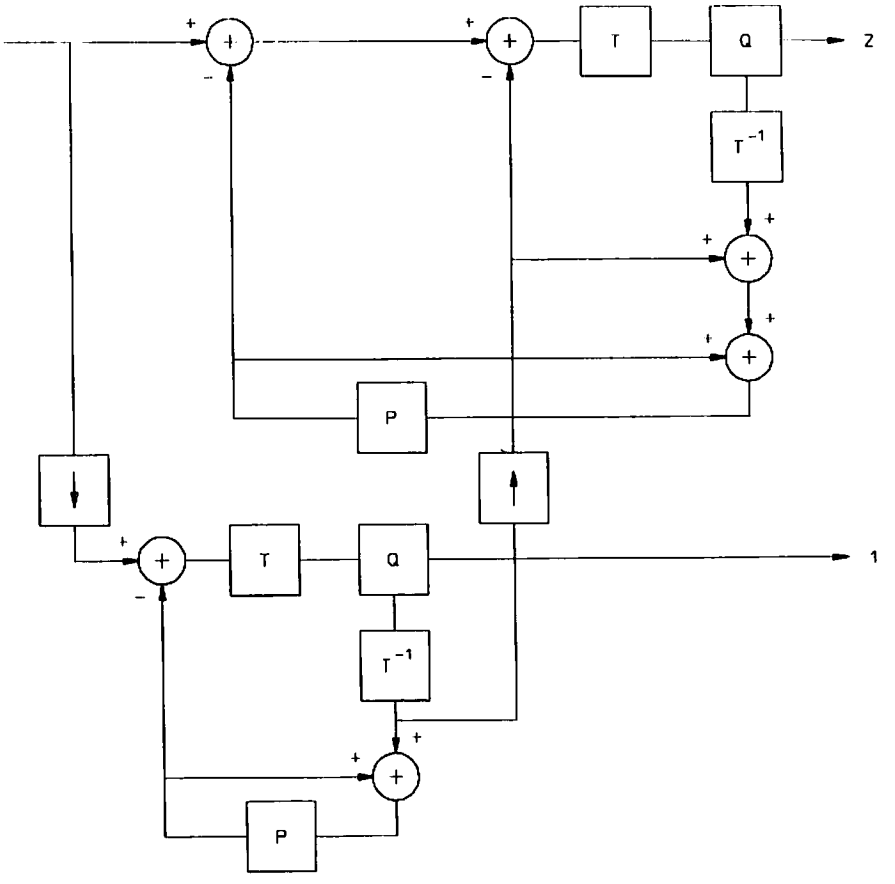


Figure G.1: Compatible encoder structure

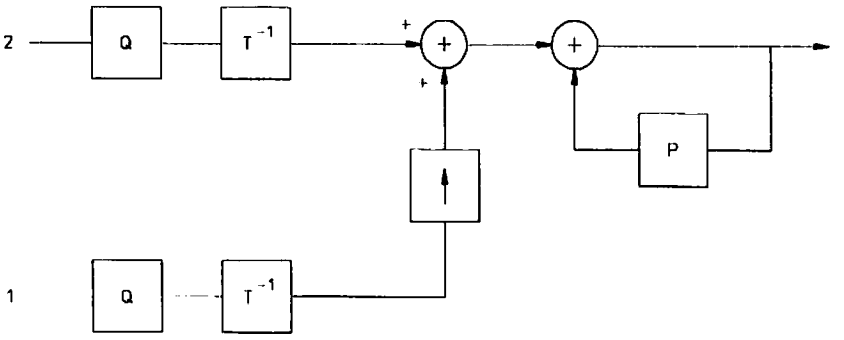


Figure G.2: Compatible decoder structure

G.1.3.2 Selection method

There are two cases:
1 - The high resolution signal is coded intra
2- The high resolution signal is coded inter

Case 1: The selection decision is on the luminance. The implementation of the compatible decision is based on the comparison of VARPE and VARPPE on a field by field basis as computed in the following algorithm:

```
for (i = 1; i <= 16; i++) {
        for (j = 1; j <= 8; j++) {
                OR = PE(i,j)
                DIF = OR - PPE(i,j)
                VARPPE += DIF * DIF
                VARPE += OR * OR
                MWOR += OR
        }
}
VARPPE /= 256;
VARPE = VARPE/256-MWOR/256*MWOR/256
```

Where: PE(i,j) denotes the pixels in the prediction error macroblock of the corresponding field. PPE(i,j) denotes the pixels of the predicted prediction error macroblock from the lower layer.

The characteristics of the decision are described in figure 6.1 (Non-Intra decision includes the solid line in figure 6.1)

Case 2: The selection decision is on the luminance. The implementation of the compatible decision is based on the comparison of VARPE and VARPPE on a field by field basis as computed in the following algorithm:

```
for (i = 1; i <= 16; i++) {
        for (j = 1; j <= 8; j++) {
                VARPE += PE(i,j) * PE(i,j);
                VARPPE += PPE(i,j) * PPE(i,j);
        }
}
```

If (VARPE < VARPPE) then non-compatible prediction for that field is used
else compatible prediction for that field is used.

Where: PE(i,j) denotes the pixels in the prediction error macroblock of the corresponding field. PPE(i,j) denotes the pixels of the predicted prediction error macroblock from the lower layer.

G.1.3.3 Quantization

Quantization is in line with chapter 7. There is one exception, which is intra quantization in only used on intra macroblock type and no compatible prediction (compatible_type = 00).

G.1.3.4 Coefficient coding

In all picture types, macroblocks in which a compatible prediction has been made (i.e. when compatible_type is not 00) the quantized coefficients are sequentially transmitted according to the zig-zag sequence given in figure 4.5.

The combinations of zero-run and the following value are encoded with variable length codes as listed in table B.5c to B.5f. End of Block (EOB) is in this set. Because CBP indicates those blocks with no coefficient data, EOB cannot occur as the first coefficient. Hence EOB does not appear in the VLC table for the first coefficient. Note that EOB is stored for all coded blocks.

The last bit 's' denotes the sign of the level, '0' for positive '1' for negative.

The remaining combinations of (RUN, LEVEL) are encoded with a 20-bit or 28-bit word consisting of 6 bits ESCAPE, 6 bits RUN and 8-bits or 16-bits LEVEL. RUN is a 6 bit fixed length code. LEVEL is an 8-bit or 16-bit fixed length code. See table B.5g

G.1.3.5 Fixed Macroblock type

If the compatible prediction results in no transmitted coefficients for entire macroblock e.g. CBP = 0, and the macroblock is inter with no motion compensation then the macroblock MUST still be transmitted. This requires a new macroblock type per picture mode (see table B.2)

G.2 Experiment 2

G.2.1 Syntax extensions

Below extensions to the syntax are given necessary to perform spatial scalable experiments

9.3.3 Sequence Header

sequence_header ( ) {

if (nextbits ( ) == extension_start_code) {			
extension_start_code	32	uimsbf	
sscalable	1		uimsbf
if (sscalable) {			
do {			
sscale_code	8		uimsbf
} while (nextbits ! = '00001111')			
end_of_sscales_code	8		'00001111'

```
        }
        interlaced                                  1           uimsbf
        fscalable                                   1           uimsbf
        reserved                                    3           uimsbf
        if (fscalable) {
                do {
                        fscale_code                 8           uimsbf
                } while (nextbits ! = '000000111')
                end_of_fscales_code                 8           '00000111'
        }
        while (nextbits ( ) ! = '0000 0000 0000 0000 0001'){
                sequence_extension_data                     8
        }
        next-start-code ( )
}
```

}

fscalable - This is a one-bit integer defined in the following table.

binary value	feature
0	not frequency scalable
1	frequency scalable

fscale_code - This is an 8-bit integer that defines DCT size for the scalable layers; i.e. DCT_size = 1<<scale_code. The values of this integer are:

fscale_code	Feature of Frequency Scale
0	fscale 1 layer
1	fscale 2 layer
2	fscale 4 layer
3	fscale 8 layer
4	reserved
5	reserved
6	reserved
7	end_of_fscale_code
8	reserved
@	@
@	@
@	@
255	reserved

Example: A bit stream with fscale-codes 1,2,3,3,7 would be interpreted to mean that there are four frequency layers a scale-2, followed by a scale-4, followed by two scale-8 layers.

9.3.5 Picture Layer

```
picture ( ) {

do {
        if (tmp_start_code == slave_slice_start_code) {
                slave_slice ( )
        }
        else {
                slice ( )
        }

} while (tmp_start_code = nextbits ( )) ==(slice_start_code || sif_even_start_code ||
                                        ccir_601_slice_start_code || slave_slice_start_code ))

}
```

9.3.6 Slice Layer

```
slice ( ) {
        slice_start_code || sif_even_slice_start_code                    32      bslbf
                                || ccir_601_slice_start_code


        if (fscalable) {
                extra_bit_slice                                          1       "1"
                dct_size                                                 8       uimsbf
        }




}
```

9.3.7 Macroblock Layer

```
macroblock ( ) {



        if (sscalable)
                compatible_type                                          1       uimsbf


for (i = 0; i < 6; i ++ ) {
        if (fscalable && sscalable) {
                if (slice_start_code) {
                        scaled_block (i, dct_size)
                }
                else {
                        block (i)
                }
        }
        else if (fscalable) {
                scaled_block (i, dct_size)
        }
        else {
                block (i);
        }
}


}
```

The text corresponding to the following sections G.2, G.3 and G.4 will be modified before the next meeting, and the reflects experiments before the Rio Meeting.

G.2 Compatibility Experiment 2: MPEG-1 Field Coding in a Three Layer Structure

In this experiment, three spatial resolution layers are allowed. Layer 1 consists of MPEG-1 coded SIF odd fields and producers a MPEG-1 compatible constrained bit stream. Layer 2 consists of SIF even fields coded

using MPEG-2 field/dual field adaptive macroblock motion compensation allowed in field structure pictures, within a sequence of frames. Layer 1 and Layer 2 combined also form equivalent HHR resolution. Layer 3 is CCIR 601 and uses adaptive choice of temporal prediction from previous coded and spatial reduction (obtained by upsampling decoded Layer 1 and Layer 2) corresponding to current temporal reference. For this layer, all macroblock adaptive motion compensation modes in either frame-picture or field-pictures can be employed and are subject to experimentation. This compatible and spatially scalable coding scheme is illustrated by functional diagram of Fig. G.3.

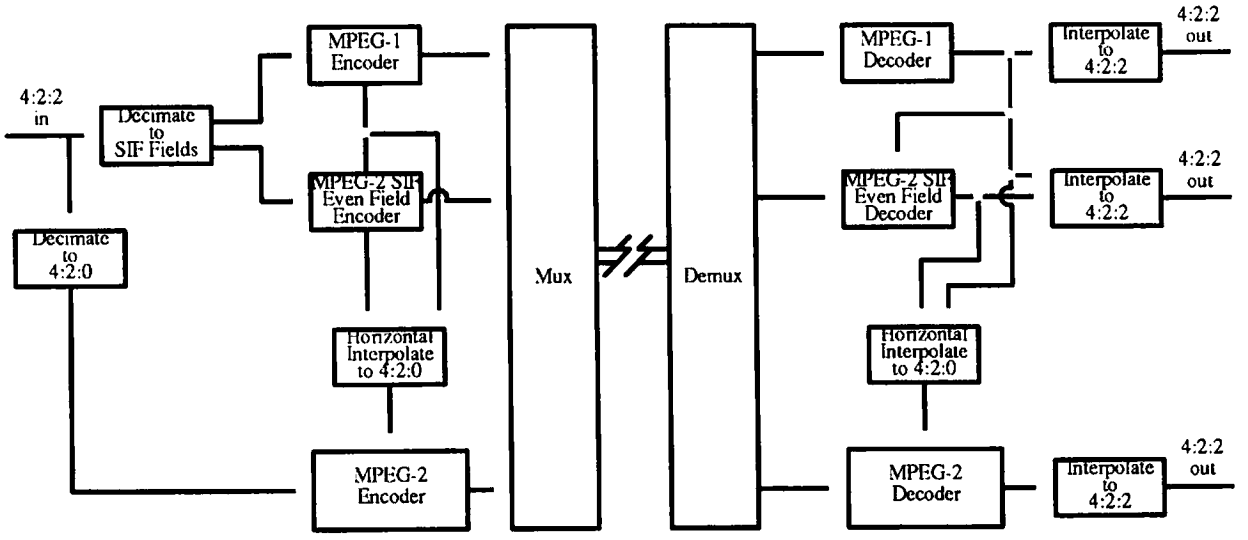


Fig. G.3 Functional Diagram of a 3 Layer Spatially Scalable and Constrained Compatible Scheme

Filters used in decimation and for interpolation are described in Sections 3.3.3 and 3.4.3 respectively.

G.2.1 Data Rates

Layer 1: SIF odd MPEG-1 coded at 1.15 Mbit/s
Layer 2: SIF Even MPEG-2 Field/Dual Field coded with MPEG-1 prediction at 0.85 Mbit/s
Layer 3: CCIR 601 MPEG-2 coded (with spatial predictions from Layer 1 and Layer 2) at 2.0 Mbit/s

The total data rate for Layers 1, 2, and 3 is 4 Mbit/s. Experimentation can performed with alternate data rate assignments of 1.5, 0.9, and 1.6 Mbit/s corresponding to Layers 1, 2, and 3.

G.2.2 MPEG-1 Encoding

For simulations, SM3 with following modifications is employed.
- No variable thresholding
- Intra matrix also employed for Non-Intra macroblocks
- Motion estimation range horizontally and vertically of

using MPEG-2 field/dual field adaptive macroblock motion compensation allowed in field structure pictures, within a sequence of frames. Layer 1 and Layer 2 combined also form equivalent HHR resolution. Layer 3 is CCIR 601 and uses adaptive choice of temporal prediction from previous coded and spatial reduction (obtained by upsampling decoded Layer 1 and Layer 2) corresponding to current temporal reference. For this layer, all macroblock adaptive motion compensation modes in either frame-picture or field-pictures can be employed and are subject to experimentation. This compatible and spatially scalable coding scheme is illustrated by functional diagram of Fig. G.3.

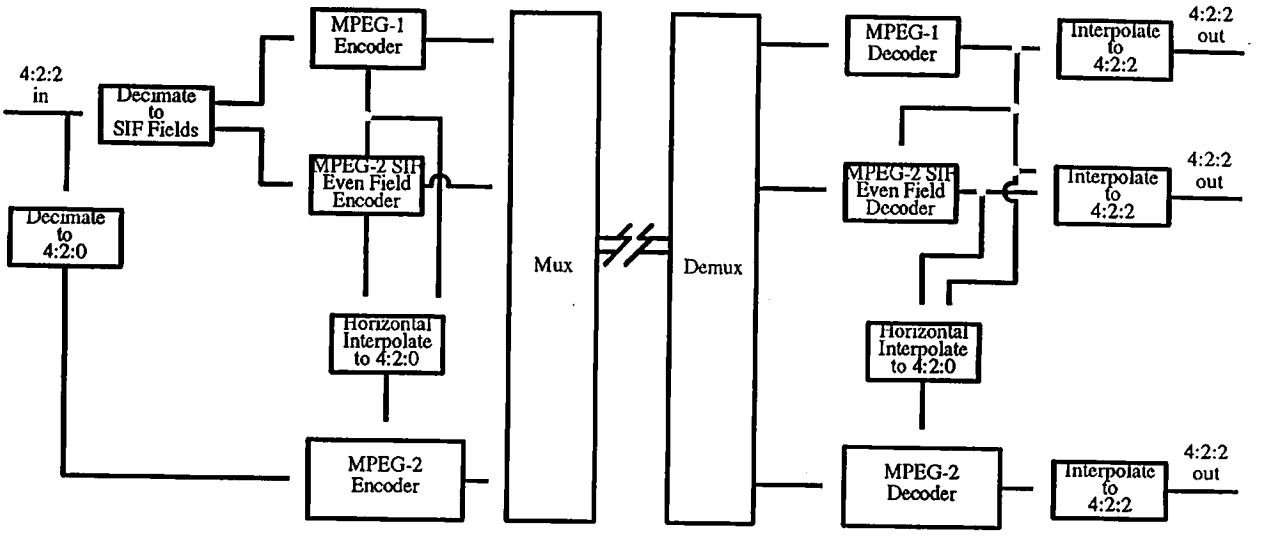


Fig. G.3 Functional Diagram of a 3 Layer Spatially Scalable and Constrained Compatible Scheme

Filters used in decimation and for interpolation are described in Sections 3.3.3 and 3.4.3 respectively.

G.2.1 Data Rates

Layer 1: SIF odd MPEG-1 coded at 1.15 Mbit/s
Layer 2: SIF Even MPEG-2 Field/Dual Field coded with MPEG-1 prediction at 0.85 Mbit/s
Layer 3: CCIR 601 MPEG-2 coded (with spatial predictions from Layer 1 and Layer 2) at 2.0 Mbit/s

The total data rate for Layers 1, 2, and 3 is 4 Mbit/s. Experimentation can performed with alternate data rate assignments of 1.5, 0.9, and 1.6 Mbit/s corresponding to Layers 1, 2, and 3.

G.2.2 MPEG-1 Encoding

For simulations, SM3 with following modifications is employed.
- No variable thresholding
- Intra matrix also employed for Non-Intra macroblocks
- Motion estimation range horizontally and vertically of ®10.5 pels between consecutive pictures.
- Mquant and rate-control of MPEG-2 PWD0.

Coding is carried out with M=3, N=15 (30 Hz)/N = 12 (25 Hz) at 1.15 Mbit/s.

G.2.3. Field/Dual Field MPEG-1 Predictive Encoding

The choice of Field/Dual Field motion compensation is made on a macroblock basis when encoding macroblocks of even fields using MPEG-2 field-picture option. Odd fields are coded exactly as MPEG-1 and are used in prediction of even fields. The motion compensation (MC) prediction macroblock modes are illustrated in Fig. G.4(a) and Fig. G.4(b). Flied-structure pictures are employed, odd fields (O) and even fields (E) are shown along with examples of MC prediction modes in SIF even fields both for P- and B- pictures.

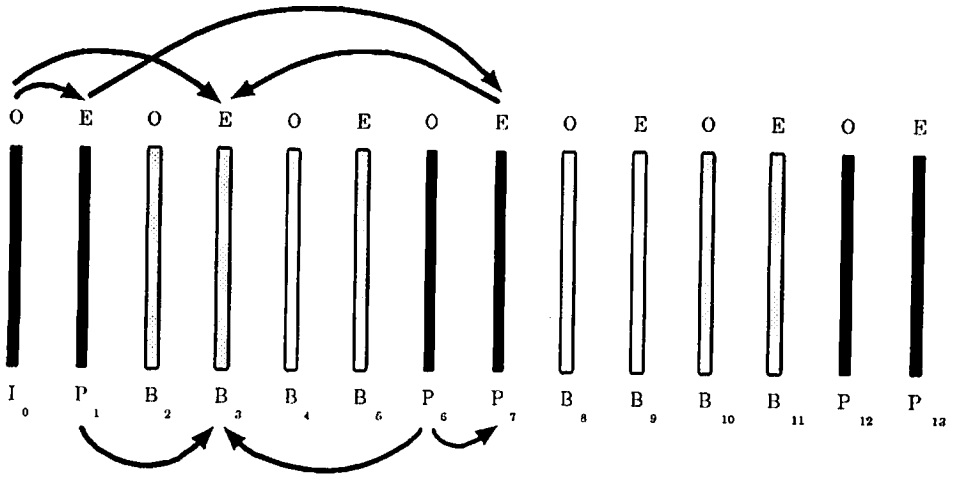


Fig G.4(a) MC Prediction for Field based Macroblocks in Even Fields.

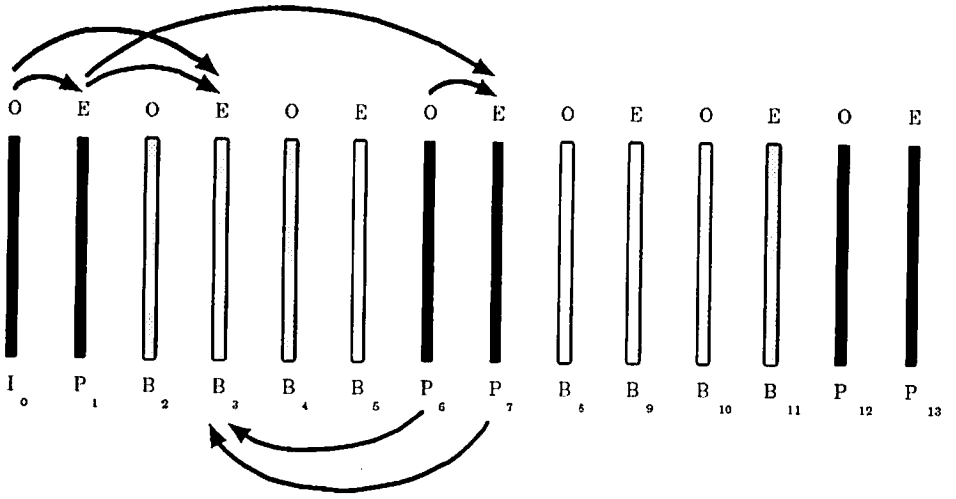


Fig G.4(b) MC Prediction for Dual Field Based Macroblocks in Even Fields

G.2.4 CCIR 601 Encoding with Compatible Prediction

Either frame-pictures of field-pictures can be assumed for coding of this layer. If frame-pictures are employed, macroblocks can adaptively choose between frame, field, or dual field based MC from previous coded pictures as well as spatial prediction for current (combined layer 1 and layer 2) lower resolution picture. In field structure pictures, on a macroblock basis, field or dual field based MC modes can be selected. The "interpolate to 4:2:0" in Fig. G.3 forms the spatial prediction and is performed on macroblock basis.

G.2.5 Syntax and Multiplexing

At sequence layer **sscalable** is set to '1'; **fscalable** is set to '0'; and **interlace** is set to '1'.
The **sscale_code** sequence for this experiment is 1,5,8,15, as discussed in example in Sec. 9.3.3. A slice based multiplexing scheme of Sec. 4.3.1 is used with start codes from Sec. 9.3.1.2.

G.3 Compatibility Experiment 3: MPEG-1 Frame Coding in a Two Layer Scheme

In this experiment, two spatial resolution layers are allowed. Layer 1 consists of MPEG-1 coded HHR frames and produces MPEG-1 compatible unconstrained bit stream. Layer 2 is CCIR 601 and uses adaptive choice of temporal prediction from previous coded and spatial prediction (obtained by upsampling decoded HHR) corresponding to current temporal reference. For this layer, on a macroblock basis, best MC mode is

adaptively chosen from among various options available in frame (or field) structure pictures and is subject to experimentation. This compatible and spatially scalable coding scheme is illustrated functional diagram of Fig. G.5.

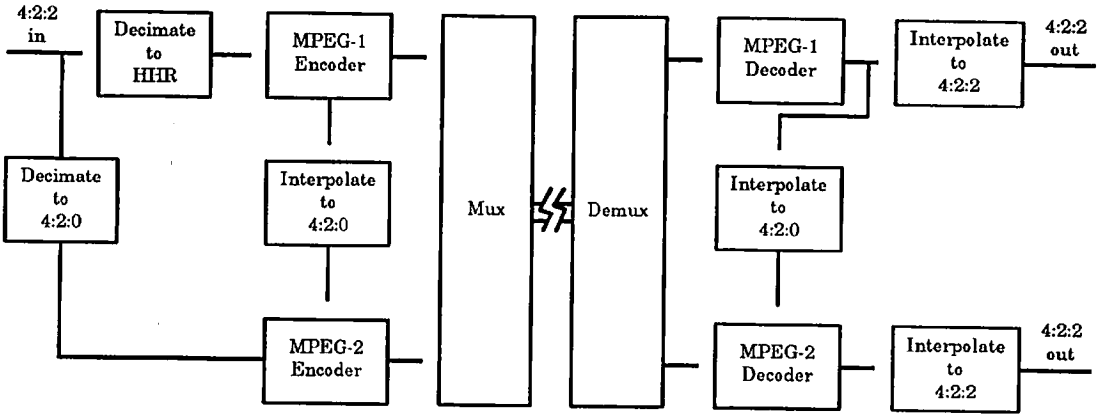


Fig. G.5 Functional Diagram of a 2 Layer Spatially Scalable Code and Unconstrained Compatible Scheme.

Filters used for decimation and for interpolation are described in Sections 3.3.4 and 3.4.4 respectively.

G.3.1 Data Rates

Layer 1: HHR MPEG-1 coded at 2.0 Mbit/s
Layer 2: CCIR 601 MPEG-2 coded (with spatial prediction from Layer 1) at 2.0 Mbit/s

The total data rate for Layers 1 and 2 is 4 Mbit/s. Experiments can be performed with alternate data rate assignments of 2.4 and 1.6 Mbit/s corresponding to Layers 1 and 2.

G.3.2 MPEG-1 Encoding

For simulation, SM3 with following modifications is employed.
- No variable thresholding
- Intra-motion also employed for Non Intra macroblocks.
- Motion estimation range horizontally 10.5 pels and vertically 15.5 pels between consecutive pictures.
- Mquant and rate control of MPEG-2 PWD0.

Coding is carried out with M=3, N=15 (30 Hz)/N = 12 (25 Hz) at 2.0 Mbit/s.

G.3.3 CCIR 601 Encoding with Compatible Prediction

Either frame structure or field structure pictures can be assumed for coding of this layer. If frame-pictures are employed, macroblocks can adaptively choose between frame, field, or dual field based MC from previous coded pictures as well as spatial prediction from current (Layer 1) lower resolution pictures. In field-pictures, on a macroblock basis, field or dual-field based MC modes can be selected. The "interpolate to 4:2:0" in Fig. G.5 forms the spatial prediction and is performed on a macroblock basis.

G.3.4 Syntax and Multiplexing

At sequence layer **sscalable** is set to '1', **fscalable** is set to '0', and **interlace** is set to '1'. The **sscale_code** sequence for this experiment is 4,9,15, the first two identify coding standards for each resolution and the last is **end_of_sscale_code** (as discussed in Sec. 9.3.3). A slice based multiplexing scheme of Sec. 4.3.2 is used with start codes from Sec. 9.3.1.3.

G.4 Compatibility Experiment 4: MPEG-1 Field Coding in a Two Layer Scheme

In this experiment, two spatial resolution layers are allowed. Layer 1 consists of MPEG-1 coded either SIF odd or SIF interlaced fields and produces a MPEG-1 compatible constrained bit stream. Layer 2 is CCIR 601 and uses adaptive choice of temporal prediction from previous coded and spatial prediction (obtained by upsampling decoded SIF odd or SIF interlaced) corresponding to current temporal reference. If SIF odd is used in Layer 1, Layer 2 could be field structure so that on a macroblock basis, field or dual field MC as well as horizontally interpolated SIF odd field prediction can be employed. If SIF interlaced is used in Layer 1, Layer 2 could be frame structure and on a macroblock basis, best prediction mode could be adaptively chosen from among various options available in frame structured pictures. This compatible and spatially scalable coding scheme is illustrated by functional diagram of Fig. G.6.

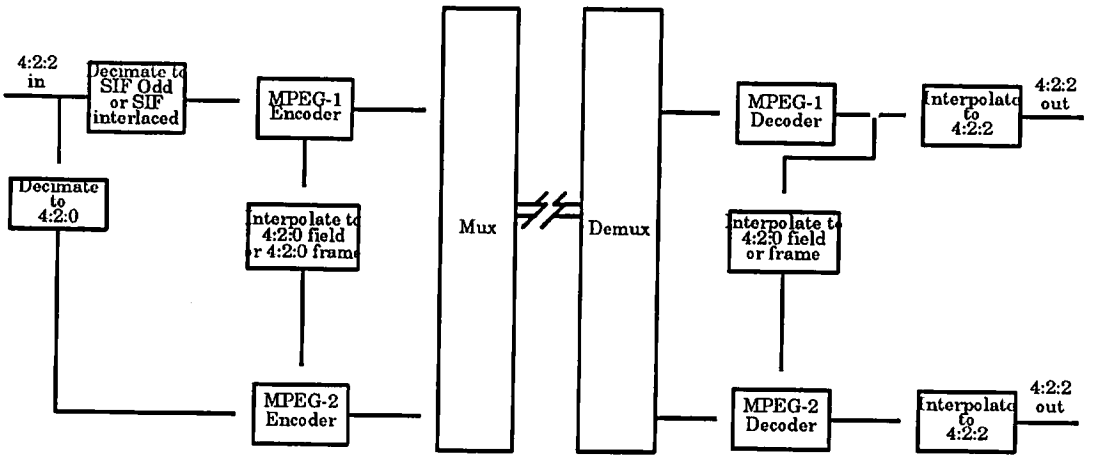


Fig. G.6 Functional Diagram of a 2 Layer MPEG-1 Constrained Compatible Code

Filters used for decimation and for interpolation for SIF odd are described in Sec. 3.3.2 and 3.4.2, filters for SIF interlaced are specified in Sec. 3.3.5 and 3.4.5.

G.4.1 Data Rates

Layer 1: SIF odd or SIF interlaced MPEG-1 coded at 1.15 Mbit/s
Layer 2: CCIR 601 MPEG-2 coded (with spatial prediction for Layer 1) at 2.85 Mbit/s

The total data rate for Layers 1 and 2 is 4 Mbit/s. Experiments can be performed with alternate data rate assignments of 1.5 and 2.5 Mbit/s corresponding to Layers 1 and 2.

G.4.2. MPEG-1 Encoding

For simulation, SM3 with following modifications is employed:

- No variable thresholding
- Intramatrix also employed for Non Intra macroblock
- Motion estimation range horizontally and vertically is ®10.5 pels between consecutive pictures
- Mquant and rate control of MPEG-2 PWD0.

Coding is carried out with M = 3, N = 15 (30 Hz)/N = 12 (25 Hz) at 1.15 Mbit/s

G.4.3 CCIR 601 Encoding with Compatible Prediction

If SIF odd is employed in Layer 1, field structure coding is assumed for CCIR601 layer. If SIF interlaced is employed in Layer 1, frame structure coding is assumed for CCIR601 layer. For field-pictures, on a macroblock basis, field or dual field MC mode can be selected. The "interpolate to 4:2:0 field or 4:2:0 frame" in Fig. G.6 forms the spatial prediction and is performed on a macroblock basis.

G.4.4. Syntax and Multiplexing

At sequence layer **sscalable** is set to '1', **fscalable** is set to '0', and **interlace** is set to '1'. The **sscalable_code** sequence when this experiment is performed with SIF odd is 1,6,15; when SIF interlaced is used the sequence 3,7,15. The **end_of_sscale_code** is 15. A slice based multiplexing scheme of Sec. 4.3.3 with start codes from Sec. 9.3.1.4 are used. It is worth noting that if SIF odd and field structure coding are employed, slice based multiplexing scheme of Sec. 4.3.3 may be used for CCIR 601 MPEG-2 coded odd field only, even fields may contain only CCIR 601 slices.

Appendix H: Low delay coding.

H.1 Simulation guidelines for low delay profile.

H.1.1 Coding structure.

In the low delay profile it is assumed that the total coding/decoding delay shall be kept below 150 ms. However, it is realized that coding/decoding delay considerably below that limit could be desirable for many applications (e.g. two way communication over satellite links).

In the following the delay will be measured in "field periods" - fp. For 50 and 60 Hz pictures the 150 ms correspond to:

- 50 Hz -> 150 ms = 7.5 fp.
- 60 Hz -> 150 ms = 9.0 fp.

The coding/decoding delay is considered to consist mainly of two parts:
- Buffer delay. A typical value is 5 fp. This is mainly due to the regular INTRA picture update. With a different update mechanism it is assumed that the buffer delay may be reduced to 2 fp.
- Delay resulting from frame/field reordering. This will be called basic delay.

The sum of these two contribution should be below 150 ms - or as low as possible. The possible coding modes fulfilling this criteria are listed in the table below.

Predictor performance:
Coding performance is important also for low delay coding. Performance depends to a large extent on the quality of the predictor. The preferred predictors for the different modes are:

- Field coding and M>1: The predictors in TM1 performs well.
- Frame coding and M=1: FAMC - or similar - are the preferred modes.
- Field coding and M=1: A prediction mode similar to FAMC have shown very promising performance and would be preferable. The syntax is the same as for field coding with one vector pr. MB as defined in TM1.

The table below list:
- Possible coding modes that fulfills the delay criteria.
- Preferred prediction modes (for M=1).
- Different delays.
- Preliminary performance ranking. This is far from complete due to lacking results from core experiments. (For "field, M=1" there is one simulation but no direct comparison).

Coding mode	M	Buffer delay	Basic delay	Total delay	Performance ranking (low="good")
Frame FAMC - or similar	1	5 fp	2 fp	7 fp	2
Field FAMC - or similar	1	5 fp	0 fp	5 fp	-
Frame FAMC - or similar	1	2 fp	2 fp	4 fp	2
Field FAMC - or similar	1	2 fp	0 fp	2 fp	-
Field structure	2	2 fp	4 fp	6 fp	1

H.1.2 Handling of scene change to maintain low delay.

A major contribution to the buffer delay is the complete INTRA frames/fields coding. To get around this delay it is necessary for the low delay profile to have the possibility of picture skipping. The encoder may decide to skip frames after a "large" picture due to scene cut.

VBV consideration.

Refer to appendix C in the TM.

The following procedure for removing data from the buffer is used for the low delay profile.

1:
After each subsequent picture interval, data is removed instantaneously from the buffer. Two situations may occur:

A:
A whole picture is removed from the buffer. This is the normal situation defined in TM, Appendix C.;
B:
The content of the buffer is only parts of a picture. In this case the whole picture may not be reconstructed. The following actions shall then be taken:
- Continue displaying the last decoded field.
- At the next picture interval, continue reading data from the buffer and decode the picture.

Repeat the procedure until all the bits of a picture have been removed and a complete picture is reconstructed. The picture may then be displayed.

2:
Temporal reference is counting the pictures to be displayed only. An example is given below. Assume that the source pictures are A,B,C,D,E,F etc.

Temporal reference without picture skipping:

```
      A  B  C  D  E  F
TR:   1  2  3  4  5  6
```

Temporal reference with picture E skipping:

```
      A  B  C  D  F
TR:   1  2  3  4  5
```

The procedure described here gives room for the scene change handling defined in document MPEG92/248 and shown in Figure 1 part 3): "Frame dropping is allowed" in that document.

H.1.3 How to handle the first picture.

In the low delay profile it is of interest to study the "steady state" performance of buffer delay. It is therefore desirable to pretend that we are in the steady state situation from the beginning of the sequence. The proposed way to obtain this is the following:

- Code the first picture INTRA with QP=16 (for 4 Mb/s).
- After the first picture the buffer is set to a value corresponding to 1/60th of a second (66667 bits for 4 Mb/s).
- The number of bits for the first picture - used for buffer regulation - is set to the average number of bits for the sequence.
- In evaluating the delay only buffer filling after the first picture is considered.

H.1.4 Definition of intra slice/column coding.

To reduce buffer delay the updating is made with forced INTRA slices rather than forced INTRA pictures. The procedure for doing the forced INTRA slice coding is shown in the figure below. All the modes given in the table above are covered. The updating for the different modes are arranged so that the time for total update is the same for all modes.

Instead of intra slice updating, intra column updating may be used. In this case one column of MBs is treated as an "update slice". The updating procedure for columns can be the same as for slices indicated in the figure below.

- o Frame mode, M=1: Two slices pr. frame.
- o Field mode, M=1: One slice every other field.
- o Field mode, M=2: One slice every field for P-fields.

To guarantee that errors do not propagate there is restriction on predictions in region 1. Motion vectors in region 1 may not refer to areas in region 2 (refer to the figure).

With this method the time for total refresh is 500 ms for 60 Hz sequences and 720 ms for 50 Hz sequences. It should be noted that the maximum entry times for channel hopping are twice as large as the mentioned values.

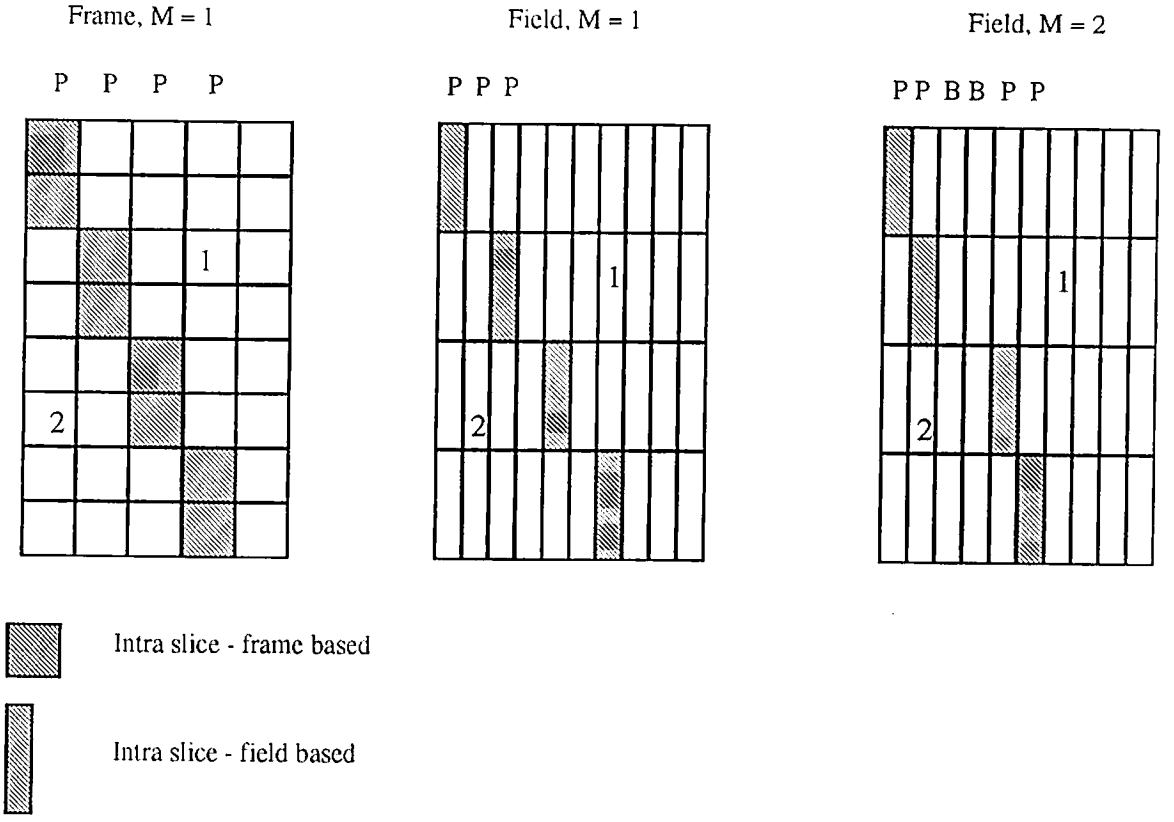


Figure H.1 Low Delay Coding stategy

H.1.5 Rate control.

INTRA SLICES

For intra slice updating, a modified buffer regulation is introduced.

TM1 rate control assumes that the generated bit amount B_j increases constantly. Therefore buffer occupancy d_j^p is calculated as follows.

$$d_j^p = d_0^p + B_{j-1} - \tilde{B}_{j-1} \qquad (1)$$

$$\tilde{B}_j = T_p * \frac{j}{MB\_cnt} \qquad (2)$$

Here $\tilde{B}_j$ is a prediction value of Bj. It increases linearly as shown in figure 2. When using intra slices, $\tilde{B}_j$ is modified as in figure 3.

Target bit amount is also modified to $T_{p_p} + T_{p_i}$. The target bit amounts are calculated as :

$$T_{p_p} = \frac{Number\_of\_P\_slice}{MB\_cnt} * T_p \qquad (3)$$

$$T_{p_i} = \frac{Number\_of\_I\_slice}{MB\_cnt} * T_I \qquad (4)$$

The rate control may then be operated as in equation (1).

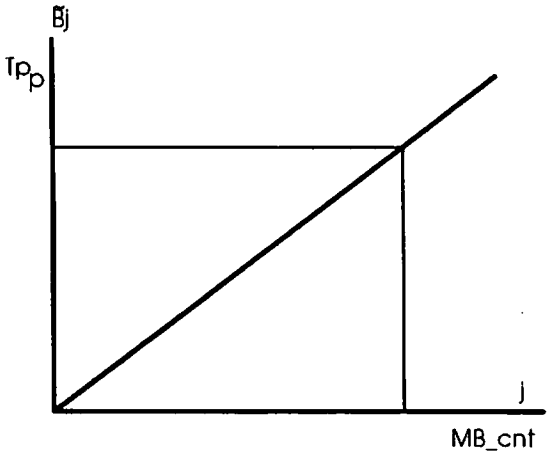


Figure H.2

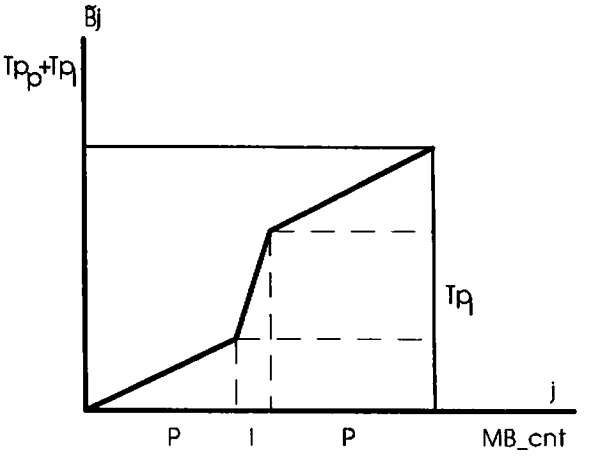


Figure H.3

SKIPPED PICTURES

For picture skipping the following buffer regulation is used:

- For the first picture after a scene change QP is fixed to the same value as for the first picture.
- Rate control is activated again from the second picture after the scene change.

H.1.6 Influence of leaky prediction on low delay coding.

If leaky prediction is needed for other purposes, it could also be useful for the low delay profile. Forced INTRA slices could be replaced by leaky prediction. Possible advantages could be reduced buffer delay and no visible INTRA updating.

H.2 Experiments

The three coding structures were picked up for low delay at Angra meeting.

- Frame structure M=1
- Field structure M=1
- Field structure M=2

The core experiments for low delay should be carried out on the above structures. The rate control with forced intra picture and with picture skipping can be found in Appendix H "Guide line of simulation with low delay profile". In other cases rate control in as in TM1.

H.2.1 Comparison of coding efficiency among predictions at low delay.

Every coding structure has several prediction modes. In order to seek the suitable prediction for low delay, the performance of predictions will be compared within the following table.

prediction / structure	Fr/Fi	Fr/Fi/FAMC	SVMC	Fr/Fi/Dual'/Dual
Fr M=1				
Fi M=1				
Fi M=2				

Experiments should be performed so that proper comparisons cab be made concerning;

- Prediction
- Coding structure.

H.2.2 Comparison of coding efficiency and delay between using intra picture and forced intra slice

Forced intra slice can reduce the buffering delay because of smoothing the number of bits per frame. To check the impact of intra slice , the coding efficiency and delay will be compared between using intra picture and forced intra slice as the same table of experiment 1.

H.2.3 Comparison of coding efficiency and delay between with and without picture skipping

At scene change, picture skipping is useful technique to keep the delay low. The coding efficiency and delay with/without picture skipping at scene change will be compared on the same table of experiment 1. To get scene change, two sequences are combined (e.g. Flower Garden / Mobile Calendar).

APPENDIX I : Frequency Domain Scalability Core Experiments

Core Experiment I.1: Interlace-in-Interlace Extraction

I.1.1: Background

It has become apparent that the case of a two-layer scalable video hierarchy with both layers interlaced is very important. It is further noted that the quality of each layer is critical in some applications. It is therefore important to investigate encoder solutions that address this issue.

To produce a lower resolution layer in frequency scalable video, a subset of the DCT coefficients of the 8x8 blocks are extracted. In the case of non-interlaced video, the upper left 4x4 coefficients are chosen. When this technique is applied to interlaced source material, the temporal resolution is reduced by a factor of two. This is unacceptable for high quality applications.

A first solution to this problem was to adaptively use frame and field coding and motion compensation modes. However, field-based extraction of the lower layer produces samples on non-equally spaced lines, none of which are spatially co-located with those produced by frame-based extraction as shown in Figure I.1.

figure to be inserted

Figure I.1 Raster mis-match between frame and field based 4x4 extraction.

This effect results in very poor motion-compensated prediction when the current macroblock and the prediction macroblock in the reference frame are coded in different modes. Moreover even at high bit rates, a high quality lower layer signal is not available.

In an attempt to avoid the complexity of correcting the raster position of the field-based lower layer pixels, some alternative solutions, involving the use of non-lowpass subsets of the DCT coefficients in scale-4, have been proposed and will be evaluated during this core experiment.

It is well-understood that the interlaced-in-interlaced mode of operation could well be used with complex encoder structures designed to satisfy quality requirements of specific applications. However, in order to isolate the effect of the options under test from other aspects of frequency domain scalable systems, we propose to conduct this core experiment using a simple encoder structure which produces a single layer bit-stream.

Core experiment I.1 addresses the low layer encoder, which produces SIF resolution images. This decoder, given in Figure I.2, is identical in structure to the upper layer decoder with the exception of the quarter-pixel resolution for the motion-compensation as described in Appendix D. The decoder will decode the entire bit-stream, extracting one of the subsets of the scale-8 DCT coefficients given in Figure I.3. Images produced with each of these subsets will be compared to determine which subset gives the subjectively best tradeoff of motion rendition, vertical resolution, and aliasing.

figure to be inserted

Figure I.2: Scale-4 decoder for core experiment I.1

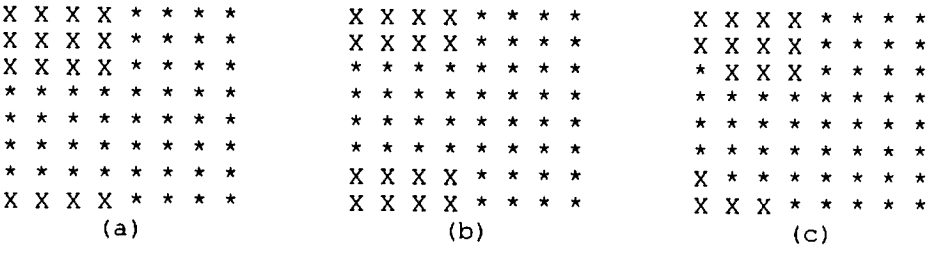


Figure I.3. Three possible DCT coefficient subsets.

Note: This core experiment uses a single-layer bit-stream. Bit-stream scalability requires additional base/slave layer multiplexing, which in turn requires the scanning order to be defined appropriately. The best scanning path depends on the selected set of DCT coefficients, and is still under investigation.

I.1.2: Description

The encoder and upper layer decoder are identical to their Test Model 1 non-scalable equivalents under the following conditions:

- Chroma sub-sampling 4:2:0
- Bit rate 4.0 Mb/s
- N=15 (12), M=3
- Frame coding and motion compensation modes only
- Other aspects as in TM1 and/or as in TM2 Appendix D

The lower layer decoder has the following features which differ from the scalable Test Model 1 scale-4 decoder:

- scale-4 DCT coefficients extracted from single layer bit-stream
- 1/4 pixel motion vector resolution in scale 4

Test sequences: Bicycle, Cheer Leaders

I.1.3: Syntax Changes

None required for this core experiment.

Core Experiment I.2: Pyramid Encoder Improvements

This core experiment addresses the issue of retaining the high quality high resolution layer of frequency domain scalable encoders, while improving the quality of the lower layer(s). The fundamental nature of the experiment should make the results useful across a range of application profiles.

I.2.1: Background

A single-loop two-layer encoder using frame coding and motion-compensation modes only can be made efficient at high resolution, but suffers from drift problems at lower resolutions. A significant improvement in lower resolution quality is realized when the full precision of the motion vector information as it exists in the bit-stream is used to perform the lower resolution motion compensation as described in Appendix D. However, it is felt that more improvements can be made to the basic framework of a multi-layer encoder to further improve the quality.

One investigation along this line has been the use of motion-compensation loops in more than the highest resolution layer. When the multiple loops are uncoupled except for the upward coefficient prediction, as shown in MPEG 91/212 Figures 5a and 6, the drift problem is solved, but at the expense of reducing the upward prediction efficiency. The result is a tilt in the balance of quality away from the high resolution layer. Overall, the coding noise is increased. In an attempt to control this quality balance more carefully, coupled motion compensation prediction loops have been studied, in which the coupling takes the follow form. At each layer, the prediction error signal is formed by a weighted average of the appropriate DCT coefficients from more than one scale, as in MPEG 92/288 Figure 1. These weights allow one to realize encoders which vary smoothly from the uncoupled case, which represents the best drift correction we currently know of, to the simple case in which the low resolution coefficients are extracted from the high resolution layer. It was hoped

that some in-between setting of the weights would produce a better quality balance. However, the basic problems of low layer drift and upward prediction efficiency are not simultaneously resolved.

A solution to the drift problem was presented in MPEG 92/288 on the basis of a subband scheme using multiple prediction loops in encoder and decoder. This scheme will be addressed in the next Core Experiment trying to identify the maximum quality scheme with less hardware complexity constraints.

In an attempt to find simple solutions to the fundamental problems, we propose a Core Experiment in which the value of the lower layer coefficients as a predictor for those in the higher layer is studied. In particular, we envision two classes of variations. The first would use this upward DCT coefficient prediction in all MBs of certain frame types but not in those of others. The second class would allow adaptive use of the upward prediction on a macroblock basis. The purpose of the core experiment is to compare the picture quality resulting from the various approaches.

The first variation to be tested is one in which the upward prediction is unconditionally disabled for P- and B-frames, to determine whether global prediciton of the prediction error (which is precisely what the upward prediction is doing in P- and B-pictures) is effective. This encoder is known as Version A. The second variation (Versions B and C) will employ adaptive upward coefficient prediction. This adaptivity will be possible at the MB level.

In the conditional upward prediction encoder Version B, upward prediction is done if the mean square prediction error is less than the energy of the predicted signal. As a benchmark, an encoder that uses the mode requiring fewer bits will also be implemented (Version C). The results in terms of SNR and picture quality produced by these two encoders will be compared with that of the basic encoder (with upward prediction), as well as with Version A.

In either of these experiments, improvements in the upward prediction will result in quality improvement at both resolutions, since the quantization factors used in each scale are identical. It is also of interest to consider the effect of conditional upward coefficient prediction in cases where the lower layer is bit-rate-constrained through more coarse quantization, aa this is likely to worsen the prediction efficiency. It should be noted that even in the worst case in which upward prediction never helps, the frequency pyramid has advantages over simulcast in not requiring redundant coding of side information and in efficiently coding I-frame data. Although no core experiment along these lines is proposed at this time, independent investigation is encouraged. Furthermore, we feel that important improvements in frequency domain scalable encoder performance can be made in areas other than conditional upward prediction and motion vector resolution, and urge our MPEG colleagues to consider this problem.

These encoders all fall within the framework shown in Figure I.4.

figure to be inserted

Figure I.4: Two-layer scalable encoder with conditional upward prediction.

The use of upward prediction is summarized in Table I.1.

Table I.1. Use of upward prediction in different picture types

Encoder Version	Upward Prediction I	P	B
A	y	n	n
B	y	c	c
C	y	c	c

y: upward prediction used

n: upward prediction not used
c: conditional

I.2.2: Description

The base encoder for this experiment has the following specifications:

- two resolution layers (scale 8 and scale 4), pyramidal structure
- two motion-compensation loops with down-sampling filter equivalant to an 8x8 forward DCT performed on each block followed by a 4x4 IDCT on the upper left 4x4 subblock.
- unconditional upward DCT coefficient prediction
- equal MQUANT in all layers
- other aspects as in TM1 and/or TM2 Appendix D

Test sequence: Mobile and Calendar

I.2.3: Syntax Changes

To indicate the use of Version A in the Test Model, 3 single-bit codes, **up_I**, **up_P**, and **up_B**, are inserted in the sequence layer after **end_of_fscales_code**. The interpretation of these bits are as follows:

up_I = 0 -> upward prediction used in I-pictures
up_I = 1 -> upward prediction not used in I-pictures

up_P = 0 -> upward prediction used in P-pictures
up_P = 1 -> upward prediction not used in P-pictures

up_B = 0 -> upward prediction used in B-pictures
up_B = 1 -> upward prediction not used in B-pictures

For Versions B and C, we use the **up_IPB** flags at the sequence layer, but with the following interpretation:

up_I = 0 -> adaptive upward prediction used in I-pictures
up_I = 1 -> upward prediction always used in I-pictures

up_P = 0 -> adaptive upward prediction used in P-pictures
up_P = 1 -> upward prediction always used in P-pictures

up_B = 0 -> adaptive upward prediction used in B-pictures
up_B = 1 -> upward prediction always used in B-pictures

In order to indicate whether upward prediction is to be used in a given MB, we introduce a MB layer codeword **up_pred**, used in a fashion similar to the currently existing MB syntax for **compatible_type**. The interpretation of the codeword is simply:

up_pred = 1 -> perform upward prediction in the current MB
up_pred = 0 -> no upward prediction in the current MB

Syntax modification at the Macroblock layer (directly after the **compatible_type** test) is as follows:

```
if (frame_type == 1)
    if (up_I == 1)
        up_pred                          1      uimsbf
else if (frame_type == 2)
    if (up_P == 1)
        up_pred                          1      uimsbf
else if (frame_type == 3)
```

```
if (up_B == 3)
   up_pred                              1      uimsbf
```

To emphasize, this codeword exists in the bit-stream only for Versions B and C of Core Experiment I.2, and only for frametypes using adaptive upward prediction.

Core Experiment I.3: Maximum Quality Encoder

The purpose is to find a two-layer coding scheme with best possible subjective quality, which should also be balanced between the layers. The performance of subband and pyramid solutions to frequency scalability will be compared.

Theoretically a critically sampled (subband) scheme for layered coding should yield a superior coding performance over a pyramid scheme. It will then be suited to applications that are not satisfied with the quality from the pyramid approaches developed in the scalability group as well as in the compatibility group.

However this involves optimization of the underlying algorithm, i.e. taking away as many constraints as possible. Specifically no rate control will be used on the individual layers. Moreover one of the schemes proposes to use a multiple loop decoder for the top layer(s).

The following algorithms are proposed for this experiment:

a) the best pyramid scheme that can be identified from Core Experiment I.2
b) the subband scheme in Fig. 1 of MPEG 92/288 that was developed after the Haifa meeting.
c) the new subband scheme proposed in Fig. 3 of MPEG 92/288

Common parameters:

- Chrominance subsampling: 4:2:0
- Bit rate: 4 MBit/s
- N=12 (15), M=3
- two resolution layers (scale 8 and scale 4)
- Frame prediction and coding only
- 1/4 pixel motion vector resolution in scale 4
- equal QP in all layers
- other aspects as in TM1

The quality reference, for subjective comparisons as well as SNR calculations, is the SIF version of the original signal.

The most appropriate scheme for a) and its parameters are subject to the results of core experiment 2. Initially the encoder in MPEG 92/288 Fig 1. might be used with weighting factors of 0.5:0.5 for the two prediction error signals. The downsampling filter is the same as in Core Experiment I.2.

The weighting factors for b) shall be selected (iteratively) such that the SNR of the upper layer matches the one of a). The SNR and - more important - the subjective quality of the lower layer will be compared with a).

Proposal c) has no variables, so that a straight forward comparison of SNRs and picture quality with the other two candidates can be done.

The two subband schemes need a change in DCT scanning order to allow for bit stream scalability. The following scan order is proposed:

```
1  2  5 10 17 26 37 50
```

```
 3  4  7 12 19 28 39 52
 6  8  9 14 21 30 41 54
11 13 15 16 23 32 43 56
18 20 22 24 25 34 45 58
27 29 31 33 35 36 47 60
38 40 42 44 46 48 49 62
51 53 55 57 59 61 63 64
```

To signal the use of this scanning order for the subband schemes, a flag **subband** is introduced in the sequence header right after the **end_of_fscale_code** flag:

subband = 1 -> alternate scan order

Note: The maximum quality issue is relatively independent from the-interlace-in interlace problem as long as the results of Core Experiment I.1 are applicable here. Therefore the above scanning can be used which is suitable for non-interlaced material. Mobile & Calendar (as a mildly interlaced sequence) is selected as test sequence for this experiment.

Core Experiment I.4: Scalable Side Information

I.4.1. Application

Any application which uses frequency scalability techniques and requires one or more layers with a sufficiently low bandwidth that all overhead information can not be transmitted within or below that layer. For example, multichannel transmission with a coding layer below 1 mbit/s.

I.4.2. Experiment details

I.4.2.1 Multichannel scalability, each channel with a fixed bandwidth.

Resolution scales: 1/16, 1/4, 1 (QSIF, SIF and 601)
Bitrates: 0.75, 1.5 and 4.0.
All layers must be at full temporal resolution.

I.4.2.2 Multichannel scalability for entertainment (each channel has fixed bandwidth)

Resolution scales: 1/4, 1, 1 (SIF, 601 and 601)
Bitrates: 1.5, 3.0, 4.0.
All layers must be at full temporal resolution.

I.4.3. Syntax extensions

The primary extension to the syntax is to support sending side information (macroblock type, address increment, coded block pattern and motion vector information) in the higher layers of the encoding. We introduce a new type of slave slice, called an incremental_slice, which contains incremental_macroblocks. Also we change the method of interpreting motion vector information. Rather than send full size motion vectors in the lowest layer and scaling them in the decoder for use at each layer , the motion vectors are sent at the scale appropriate to the dct_size of the slice within which they are transmitted. For use in higher quality slices they are scaled up. The higher layer slice may optionally contain refinement information for the motion vector.

I.4.3.1 Sequence header:

Add a one bit code **incremental_side_information** which is set to one to indicate that the slave_slice code now signals a incremental_slice and the use of incremental motion vectors.

```
incremental_slice()
{
```

```
      slave_slice_code                                          32    bslbf
      quantizer_scale                                           5     uimsbf
      dct_size                                                        8
      uimsbf
      do {
       incremental_macroblock(dct_size)
      } while (nextbits() != 000 0000 0000 0000 0000 0000)
      next_start_code
}

incremental_macroblock(dct_size)
{
      while ( nextbits() == 0000 0001 111)
         macroblock_stuffing                                          11
      vlclbf
      while ( nextbits() == 0000 0001 000)
         macroblock_escape                                            11
      vlclbf
      macroblock_address_increment                                    1-11
      vlclbf
      if (new_macroblock)
         macroblock_type                                              1-6
      vlclbf
      else
         slave_macroblock_type                                        1-3
      vlclbf
      if (macroblock_quant)
         quantizer_scale                                              5
      uimsbf
      if (new_macroblock)
         if (compatible)
            compatible_type                                           2
      uimsbf
         if (interlaced) {
            if (picture_structure == 0) {
               if (macroblock_intra || macroblock_pattern)
                  interlaced_macroblock_type                          1
      uimsbf
               if (macroblock_motion_forward || macroblock_motion_backward)
                  interlace_motion_type                               2
      uimsbf
            } else {
               if (macroblock_motion_forward XOR
macroblock_motion_backward)
                  field_interlace_type                                1
      uimsbf
            }
         }
      }
      if (motion_refinement) {
         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 (interlace_motion_type) {
               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 (interlace_motion_type) {
             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 (chroma_format == 0) {
      if (macroblock_pattern)
         coded_block_pattern                                     3-9
      vlclbf
      for (i = 0; i < 6; i++)
         if (new_macroblock)
            scaled_block(i)
         else
            slave_block(i, dct_size)
   } else {
      if (macroblock_pattern)
         coded_block_pattern                                     8
      uimsbf
      for (i = 0; i < 8; i++)
         if (new_macroblock)
            scaled_block(i)
         else
            slave_block(i, dct_size)
      }
   }
   if (picture_coding_type == 4)
      end_of_macroblock                                          1      1
}
```

new_macroblock is true if no previous information has been sent about the macroblock at the current macroblock address.

I.4.4. Coding

I.4.4.1 Slave Macroblock Addressing

Relative slave macroblock address are coded as described in section 8.1.

I.4.4.2 Slave Macroblock Type

Each macroblock has one of the three modes:

1	Intra	(I-blocks)
2	Predicted	(P-blocks)
3	Interpolated	(B-blocks)

For these block types different VLC tables for the Slave Macroblock types are used. See table a for Intra, table b for predictive-coded blocks and table c for bidirectionally predictive-coded blocks. I blocks are all blocks within and I picture and intra coded blocks within P and B pictures. P blocks are predicted blocks within a P picture and forward or backward (but not both) predicted blocks within a B picture. B blocks are those blocks within a B picture which are bi-directionally predicted. The block type is determined by the first macroblock_type transmitted for the macroblock.
Methods for mode decisions are described in section 6. MTYPE is based on the highest resolution.

I.4.4.3 Macroblock_quant

Macroblock_quant is 1 when a new **quantizer_scale** is transmitted as part of the current incremental_macroblock. When **macroblock_quant** is zero, **quantizer_scale** is inherited from the previous incremental_macroblock within the current incremental_slice. For the first incremental_macroblock within an incremental_slice, a **macroblock_quant** of zero indicates use of the **quantizer_scale** transmitted at the incremental_slice level.

I.4.4.4 Motion Vectors

Motion vectors are determined as described in Annex D. The scaled motion vector is transmitted in the base layer. When a block is not transmitted then the motion vector is zero. Coding of this scaled motion vector is as described in section 8. When **Δ_mv_present** is set, slave motion vectors are transmitted. Slave motion vectors are coded as an increment to the appropriately scaled up motion vector of the closest layer below. for which a motion vector exists. For example, if a motion vector was originally transmitted in a layer with a dct_size of 2 and is now scaled up for use in a layer with dct_size of 4, it is multiplied by two along each axis and the slave motion vector is added. This new vector forms the basis for the scaled up vector of the next layer. The incremental motion vector is coded using the same variable length code table B.4. The first value transmitted for a particular motion vector, base_vector, is calculated as described in section 8.3 except the appropriate f_code is reduced by 1 for each scale reduction factor of 2. All refinement motion vectors are calculated with an f_code of 1.

I.4.4.5 Slave Coded Block Pattern

When the coded block pattern is not sent as part of an incremental_macroblock, it is assumed to be zero -- no DCT coefficients are decoded. The coded_block_pattern is not inherited between layers unlike in normal frequeny scalability. Coded block pattern are determined by the quantized coefficients of the current layer.

I.4.4.6 Coefficient Coding

Coding of intra and non-intra blocks ina a scalable bitstream is described in Annex D.

I.4.4.7 Multi-layer Rate Control

A seperate mquant value is determined for each incremental macroblock in the bitstream. The quantizer scale is only transmitted when needed as described above. The method for determining the mquant values is described in Section I.6, steps 1, 2a, and 3a.

I.4.4.8 Code / No Code decisions

Incremental macroblocks are not coded when there is no new information to be transmitted. When there are DCT or motion vector refinements the incremental macroblock must be coded.

I.4.4.9 Motion Estimation

A multiple level motion estimation routine is necessary for optimum refinement of the motion vectors at each layer. A top down hierarchical search is used. A full search is done at the highest resolution from original to original with a half pel refinement performed between original and reconstruction. The original image is decimated using a forward DCT kernel at high resolution, followed by a smaller inverse DCT of the low frequency components. The motion search in the lower scale is done from the decimated original to the lower scale reconstruction *by a refinement search around the scaled motion vector form the layer above. A search window of +/-1 pixel is used followed by a half pixel search.

I.4.5. Slave Macroblock Type

Table a. Variable length codes for slave_macroblock_type for blocks that are intra-coded (I-pictures, P-pictures, B-pictures).

	macroblock_quant	macroblock_pattern
0	0	1
1	1	1

Table b. Variable length codes for slave_macroblock_type for blocks that are predictive-coded (P-pictures, B-pictures).

VLC code	macroblock_quant	macroblock_pattern
11	0	1
0	0	0
10	1	1

Table c. Variable length codes for slave_macroblock_type for blocks that are bidirectionally predictive-coded (B-pictures).

VLC code	macroblock_quant	Δ_mv_present	macroblock_pattern
00	0	0	1
01	0	1	0
11	0	1	1
100	1	0	1
101	1	1	1

Core Experiment I.5: Scalable VLC coding

Two suggestions have been presented aimed at improving the efficiency of VLC coding in scalable systems (MPEG 92/361 & MPEG 92/356). MPEG 92/356 suggests a way to reduce the number of EOB's that need to be sent by extending the last run in each layer into the next one, and using that "run over" to implicity specify

an EOB for the layer. MPEG 92/361 suggests that different Run/Length VLC table be used at each layer. Below, the 2 experiments are described. It is suggested that both experiments be combined.

I.5.1: Experiment 1

I.5.1.1: Description

The basic idea is to send implied EOB's at lower layers in order to reduce the total number of EOB's transmitted, thereby improving the efficiency of the algorithm. There are two variations, one for subband and one for pyramid scalable algorithms.

Subband Case (as suggested in MPEG 92/356)

Layer	Coefficients always decoded	Coefficients carried over to next layer
fscale_2	1-5	4
fscale_4	6-14	11
fscale_8	15-64	

At fscale_2, the first 5 coeffients in the standard zigzag scan are coded. Coefficient 4 is transmitted to the decoder, but not used to decode the small image, and simply stored for use at the next level. If coefficient 5 is a zero, the encoder can choose to terminate the block at this layer with an EOB, or it can code it as a zero in a run/length pair that runs over into the information for the next layer - implicitly specifying the EOB. Any extra coefficients are thereby also transmitted to the decoder, but not used to produce the lowest layer image - they are stored for use at the next levels.

At fscale_4, decoding of VLC's continues at the position beyond the last coefficient transmitted from the previous level, and then proceeds as it did for the previous level.
Fscale_8 continues similiarly, and terminates with an EOB or a coefficient in position 64.

Pyramid Case

Layer	Coefficients always decoded	Coefficients carried over to next layer
fscale_2	1-5	4
fscale_4	6-25	11
fscale_8	26-64	

At fscale_2, coefficients for the first 5 coefficients are sent. If these 5 coefficients end in one or more zeros, the encoder can choose to send it as an EOB, or code the next VLC, resulting in an implied EOB as above in the sub-band case.

At fscale_4, differential values are coded for each coefficient that was transmitted at fscale_2, and quantized values are transmitted for the remaining values in the block up to coefficient 14.

At fscale_8, values transmitted for coefficients that were transmitted at the previous layer are differential, while values for the others are the quantized coefficients.Syntax Changes from MPEG 92/356.

I.5.1.2: Syntax Changes (from MPEG 92/356)

Sequence Layer:

```
do {
```

```
            :
      fscale_num_coef
      fscale_max_run
            :
} while(next_bits != '10000000')
```

Scaled Block Layer:

```
fscale_coef_cnt = coef_block(fscale_num_coef, fscale_max_run)
fscale_extra_prev = fscale_coef_cnt - fscale_num_coef
```

Slave Block Layers:

```
fscale_rem_coef = fscale_num_coef - fscale_extra_prev
if(fscale_rem_coef < 0) fscale_num_coef = 0
fscale_coef_cnt=coef_block(fscale_rem_coef, fscale_max_run)
fscale_etra_prev = fscale_coef_cnt - fscale_rem_coef

int coef_block(fscale_num_coef, fscale_max_run)
{
      fscale_coef_cnt = 0
      run = 0
      while( fscale_coef_cnt < fscale_num_coef)
            next_dct_coef(dct_size)
            fscale_coef_cnt++
            if(dct_coef_zero())run++
            else run = 0
      }
      if (run != 0){
            do{
                  next_dct_coef(dct_size)
                  fscale_coef_cnt++
                  if(dct_coef_zero()){
                        run++
                        if(run== fscale_max_run + 1){
                              end_of_block
                              break
                        }
                  }
            } while(dct_coef_zero())
      }
      return(fscale_coef)
}
```

I.5.2: Experiment I.5.2

I.5.2.1: Background and Description

This experiment is to compare the result of using the standard VLC run/length table to using different tables at each layer. Also, a different coefficient scan pattern derived from Subband coding can be used in both pyramid coding and sub-band coding with the new set of Huffman tables.

Alternate Scan:

```
 1  2  5 10 17 26 37 50
 3  4  7 12 19 28 39 52
 6  8  9 14 21 30 41 54
11 13 15 16 23 32 43 56
18 20 22 24 25 34 45 58
27 29 31 33 35 36 47 60
38 40 42 44 46 48 49 62
51 53 55 57 59 61 63 64
```

I.5.2.2: Syntax Changes and VLC's (from MPEG 92/361)

```
scaled_block(i) {
    if (pattern_code[i]) {
        if (macroblock_intra) {
            if (i<4) {
                dct_dc_size_luminance                    2-7   vlclbf
                if (dct_dc_size_luminance != 0)
                    dct_dc_differential                  1-8   vlclbf
            } else {
                dct_dc_size_chrominance                  2-8   vlclbf
                if (dct_dc_size_chrominance != 0)
                    dct_dc_differential                  1-8   vlclbf
            }
        }

        if (dct_size > 1) {
            while (nextbits() != eob_code)
                next_dct_coef(dct_size)                  2-16  vlclbf
            end_of_block (dct_size)                      2-16  vlclbf
        }
    }
}
```

Slave block Layer:

```
slave_block [i,dct_size] {
    if (pattern_code[i]) {
        while ((nextbits() != eob_code) && more_coefs) {
            next_dct_coef(dct_size)                      2-16  vlclbf
        }
        end_of_block (dct_size)                          2-16  vlclbf
    }
}
```

Core Experiment I.6: Slice vs. MB Rate Control

I.6.1: Background and Description

This core experiment will compare the effect of performing rate control at a slice or MB granularity. The outcome of this experiment is to determine whether higher resolution layer rate control at slice granularity performs as well as at macroblock granularity.

In both approaches, rate control is carried out at macroblock granularity for the low resolution layer . Also, for both approaches a multiplication factor is sent in the slice header for the higher resolution layers. In the slice granularity approach, for the higher resolution layers, each macroblock Mquant is the product of the multiplication factor and Mquant from the low resolution layer. For the macroblock granularity approach, the encoder can signal the use of :

- a new Mquant
- the slice multiplication factor
- the previous macroblock's Mquant

for each MB.

Both approaches are based on the TM1 rate control algorithm. Only the equations that differ from TM1 are given below. The picture target setting equations are common to the two approaches:

Step 1 - Bit allocation

Complexity estimation

$$X_{ipb}(l,m,h) = S_{ipb}(l,m,h) * Q_{ipb}(l,m,h)$$

where:

l refers to the lower resolution layer
m refers to the medium resolution layer
h refers to the higher resolution layer

Initial conditions:

$$X_i(l,m,h) = 160 * br\%(l,m,h) * bit\_rate/115$$
$$X_p(l,m,h) = 60 * br\%(l,m,h) * bit\_rate/115$$
$$X_b(l,m,h) = 42 * br\%(l,m,h) * bit\_rate/115$$

Picture target setting

$$T_i(l,m,h) = \max \{ \frac{R(l,m,h)}{1+\frac{N_p X_p(l,m,h)}{X_i(l,m,h) K_p} + \frac{N_p X_b(l,m,h)}{X_i(l,m,h)\, Kb}}, br\%(l,m,h)*bit\_rate / (8*picture\_rate) \}$$

$$T_p(l,m,h) = \max \{ \frac{R(l,m,h)}{N_p + \frac{N_b X_b(l,m,h) K_p}{X_p(l,m,h) K_b}}), br\%(l,m,h)*bit\_rate / (8*picture\_rate) \}$$

$$T_b(l,m,h) = \max \{ \frac{R(l,m,h)}{N_b + \frac{N_p X_p(l,m,h) K_b}{X_b(l,m,h) K_p}}, br\%(l,m,h)*bit\_rate / (8*picture\_rate) \}$$

After coding a frame, $R(l,m,h) = R(l,m,h) - S_{ipb}(l,m,h)$. Before encoding the first frame in a GOP:

$$R(l,m,h) = br\%(l,m,h)*G + R(l,m,h).$$

Step 2 - Rate Control

The two rate control aproaches differ at this step.

a) MB granularity approach

$$d_j^{ipb}(l,m,h) = d_{j0}^{ipb}(l,m,h) + B_{j-1}(l,m,h) - \frac{T_{ipb}(l,m,h)*(j-1)}{MB\_cnt}$$

$$Q_j(l,m,h) = \frac{d_j(l,m,h) * 31}{br\%(l,m,h)*r}$$

Initial conditions:

$$d_0^i(l,m,h) = \frac{10*br\%(l,m,h)*r}{31}$$
$$d_0^p(l,m,h) = K_p * d_0^i(l,m,h)$$
$$d_0^b(l,m,h) = K_b * d_0^i(l,m,h)$$

b) Slice granularity approach

$$d_s^{ipb}(l,m,h) = d_{s0}^{ipb}(l,m,h) + B_{s-1}(l,m,h) - \frac{T_{ipb}(l,m,h)*(s-1)}{Slc\_cnt}$$

$$Q_s(l,m,h) = \frac{d_s(l,m,h) * 31}{br\%(l,m,h)*r}$$

where Slc_cnt is the number of slices in a frame and s is the current slice index.

Initial conditions:

$$d_0^i(l,m,h) = \frac{10*br\%(l,m,h)*r}{31}$$
$$d_0^p(l,m,h) = K_p * d_0^i(l,m,h)$$
$$d_0^b(l,m,h) = K_b * d_0^i(l,m,h)$$

Step 3 - Adaptive quantization
This step also differs for the two approaches.

a) MB granularity approach

$$mquant_j(l,m,h) = Q_j(l,m,h) * N\_act_j$$

where N_act_j is as defined in the TM1

b) Slice granularity approach

For the lower resolution layer:

$$mquant_j(l) = Q_j(l) * N\_act_j;$$

For the higher resolution layers:

$$M\_ratio_s(m,h) = \frac{Q_s(m,h)}{Q_s(l)}$$

A 9 bit pattern (slave_slice_quantizer_ratio)

16 8 4 2 1 0.5 0.25 0.125 0.0625

is used to represent M_ratio_s, e.g a slave_slice_quantizer_ratio of 0.75 would be represented as 000001100. At the decoder the MB mquant is

$$mquant_j(m,h) = M\_ratio_s(m,h)*mquant_j(l).$$

I.6.2: Syntax extensions

a) MB granularity approach

A different mquant value may be required for each layer within the same macro-block. The syntax was extended to support transmission of the extra quantizer information. The new slave macroblock definition is:

```
slave_macroblock(dct_size) {
      slave_macroblock_quant                    1      uimsbf
      if (slave_macroblock_quant)
            slave_macroblock_quantizer_scale    5      uimsbf
      if (slave_macroblock_quant != 1)
            slave_macroblock_quantizer_source   1      uimsbf
```

```
        for (i = 0; i < 6; i++) {
                slave_block(i, dct_size)
        }
}
```

When **slave_macroblock_quant** is 1, the slave macroblock contains a new 5 bit quantizer for use in decoding the slave DCT coefficients. Otherwise a one bit field, **slave_macroblock_quantizer_source** is transmitted to indicate whether the slave_blocks should be dequantized using the same **mquant** as the previous slave macroblock (**slave_macroblock_quantizer_source** = 1), or using the product of the lower resolution layer macroblock's quantizer and the slice quantizer multiplier - $\mathbf{M\_ratio_S}$ (**slave_macroblock_quantizer_source** = 0)

b) Slice granularity approach

Slave slice layer:

```
slave_slice() {
        slave_slice_start_code                  32      bslbf
        slave_slice_quantizer_ratio              9      bslbf
        dct_size 8          uimsbf
        for(s=0; s<slice_size; s++) {
                slave_macroblock(dct_size)
        }
}
```

Core Experiment I.7: Encoder with Drift Correction Layer and Improved Motion Rendition

The purpose of this experiment is to investigate the effect of field/frame adaptive MC and DCT on a low resolution layer of a frequency scalable bitstream. Another purpose is to investigate improvements to the motion rendition of this layer.

The encoders we investigate generate a bitstream with three f-scalable layers: two 4x4 layers, and one 8x8 layer.

The data format is 4:2:2 to avoid the mismatch of the DCT mode. In the 4:2:0 format, the chrominance blocks are coded with frame DCT even if the luminance blocks are coded with field DCTs.

The frame/field adaptive MC and DCT are as described in TM1. The decision algorithm is the same as TM1 adapted to the simulation conditions below. Mode and other attributes of the slave layers are inherited from the master layer, as with the basic codec.

I.7.1 SIMULATION CONDITIONS

- Frame MC mode + Frame DCT mode
- Field MC mode + Frame DCT mode
- Frame MC mode + Field DCT mode
- Field MC mode + Field DCT mode
- Field/Frame adaptive mode, where DCT mode follows MC mode.

The actual experiments will use conditions 1 or 5 only.

I.7.2 SYNTAX OF THE PROPOSED EXPERIMENT

Most of the basic syntax is preserved in this experiment. Some minor modifications are explained throughout this text. One such modification is an extension of the basic multiplexing, to allow for the existence of two

layers at scale 4, only one of which (the base layer) is used in the reconstruction of the high resolution (scale 8) pictures. The other scale 4 layer (slave 2 below) is used as an SNR enhancement for the scale 4 base layer only.

4x4 base bitstream	master
8x8 bitstream	slave 1
4x4 difference bitstream	slave 2

To indicate this, the "fscalable" flag in the sequence header is set, followed by the following **fscales**:

bit stream	**scale code**
4x4	2
8x8	3
4x4 difference	2
end_of_scale	7

In decoding at the 8x8 resolution, the last 4x4 layer should be ignored.

I.7.3 ENCODER DESCRIPTION

The block diagram of the drift correction encoder is specified in Figure I.6. This encoder generates three hierarchical layers. Two of the hierarchical layers are generated in the same manner as the basic encoder, i.e. by extracting the 4x4 DCT coefficients from the full resolution 8x8 blocks. These two layers are the only ones required to reconstruct the full resolution video. The third layer is a difference layer generated by subtracting the output of a second encoder (operating at the 4x4 resolution) from the extracted 4x4 coefficient layer. Therefore this layer contains data that can be used to correct the drift error that would be generated by decoding with the extracted 4x4 layer alone.

figure to be inserted

Figure I.6: Drift correcting encoder.

I.7.3.1 Rate Control of the difference bitstream

Basically this drift correction method is able to be applied to every picture type: I, P, and B. However, the I-picture does not need to be compensated, because it does not have a "drift error". B-pictures also may not need to be compensated, because drift error in these pictures do not propagate. Therefore compensation may only be needed for P-pictures. If this is the case, we propose to keep the 4x4 difference slave_slices in the bitstream, but only with some flag to indicate that it is empty.

The rate control of P-picture is based on TM1. The modification is only the initial values:

Xi = bitrate/ 115
Xp = 260*bitrate/ 115
Xb = bitrate/ 115

There are two sets of target bitrates for this experiment:

1)	difference bitstream:	0.5 Mbit/s
	4x4 + 8x8 bitstream:	3.5 Mbit/s
2)	difference bitstream:	1.0 Mbit/s
	4x4 + 8x8 bitstream:	3.0 Mbit/s

I.7.4 DECODER DESCRIPTION

The corresponding decoder is shown in Figure I.7. The decoder for the 8x8 resolution is the same as a basic encoder with two layers. It therefore has no drift error. To decode 4x4 resolution video without drift error, the two 4x4 layers should be dequantized and added before applying the inverse DCT.

figure to be inserted

Figure I.7: Drift coreecting decoder.

I.7.5 EXPERIMENTS

The results from the following experiments should be compared.

I.7.5.1 ADAPTIVE DCT/MC CODING

In this experiment Simulation codition 5 is used.

I.7.5.2 SCAN PATTERNS FOR INTERLACE-IN-INTERLACE DECODING

For this experiment Simulation condition 1 is used. In addition, instead of the upper left 4x4 coefficients, the upper left and the lower left 4x2 coefficients are extracted and supplied to the 4x4 IDCT. The zig-zag scan needs to be modified for this experiment:

1	2	6	7	23	24	35	36
3	5	8	13	25	34	37	46
17	18	22	26	33	38	45	47
19	21	27	32	39	44	48	54
20	28	31	40	43	49	53	55
29	30	41	42	50	52	56	61
4	9	12	14	51	57	60	62
10	11	15	16	58	59	63	64

Core Experiment I.8: Frequency Scanning

Text for this experiment is to be supplied by G. Schamel of Heinrich Hertz Institute.

Core Experiment I.9: Efficient Frequency Scalability Core Experiment

I.9.1: Background and Description

Reference: MPEG 92/356

Goals:

1. Study impact on picture quality of encofrcing DCT coefficients of lower frequency scales to be subset of coefficients along the zigzag scan of 8x8 DCT coefficient block.

2. Measure saving in overhead bits (in sub-band approach) of sending EOB code conditionally, assuming lower layer frequency scale docoders with limited run/length capability.

Experiment:

This experiment consists of studying the impact on picture quality of using for lower frequency scales, coefficients that fall along the zigzag scan of the 8x8 DCT coefficient block, compared to choosing coefficients for lower frequency scales corresponding to a square block. A two layer approach with the following parameters is used:

Frequency Scale	# of coefficients belonging to sub-band	# of coefficients selected in zigzag scan
fscale 4	16	14
fscale 8	48 120	50

The second part of this experiment measures the savings in EOB overhead bits because of including implicit EOB (marked by the first coefficient of the next higher frequency scale). A three layer sub-band splitting of 8x8 DCT coefficients is employed and EOBs are sent only when absolutely necessary, depending on the decoding capabilities of the corresponding decoder. The parameters used are:

Frequency Scale	# of coefficients belonging to sub-band	# of coefficients selected in zigzag scan	Maximum run allowed in VLC
fscale 2	4	3	3
fscale 4	12	11	15
fscale 8	48	50	63

I.9.2 Syntax Specification

See Section I.5.1.2 for this syntax.

Core Experiment I.10: Spatial/Frequency Hybrid Scalability

I.10.1: Background and Description

Reference: MPEG 92/355, MPEG 92/356

Goal:

Investigate the effectiveness of spatial/frequency hybrid to derive well-balanced multi-resolution scales.

Experiment:

This is a revised specification of experiment I.3.1 of Appendix I in TM1. In this experiment, one spatial layer and two frequency layers are generated. The SIF odd fields are encoded using MPEG-1, but the resulting bit-stream is partitioned in "fscale4" and "fscale8" sub-streams representing frequency scales derived from SIF resolution. In generating "fscale4" and fscale8", use of efficient frequency scaling (Doc MPEG 92/356) is made to keep complexity of generating frequency scales low and to minimize the overhead. The total bit-rate for "fscale4" and "fscale8" is 1.15 Mbit/s.

The higher layer is CCIR-601 field-structure encoded using MPEG-2 at 2.85 Mbit/s. The motion compensation prediction options in field structure coding, such as adaptive field/dual field are utilized. For CCIR-601 Odd fields, an optional prediction is generated by interpolating the reconstructed SIF image.

The block diagram and syntax are described in Doc MPEG 92/355, syntax for more efficient frequency scalability is given in Doc MPEG 92/356.

I.11 Hybrid (Spatial and Frequency) Scalability

I.11.1 Hybrid Experiment 1(a): A 3 Layer Hybrid Scalable Scheme

In this experiment, an MPEG-1 encoder is used to code SIF odd or SIF interlaced (SIF-I) fields, but the encoded data is organised into two frequency scales ("fscale 8" and "fscale4"). The "fscale4" is base-layer and carries (4 x 4 block) DCT coefficients, motion vectors, coded block pattern, and macroblock type overhead for all macroblocks in each SIF field. The "fscale8" is slave layer and carries only the remaining DCT coefficients coded in MPEG-1 encoder. The base-layer, slave-layer relationship is exactly the same as that in frequency scalable experiments. For base-layer, zig-zag scan and VLC's are the same as for frequency scalable experiments, whereas slave-layer uses MPEG-1 zig-zag scan and VLC's. The decoded "fscale8"

(including "fscale4") macroblocks are upsampled and allow spatial prediction in addition to other temporal prediction (MC mode) choice available when coding CCIR 601 resolution and MPEG-2 encoder.

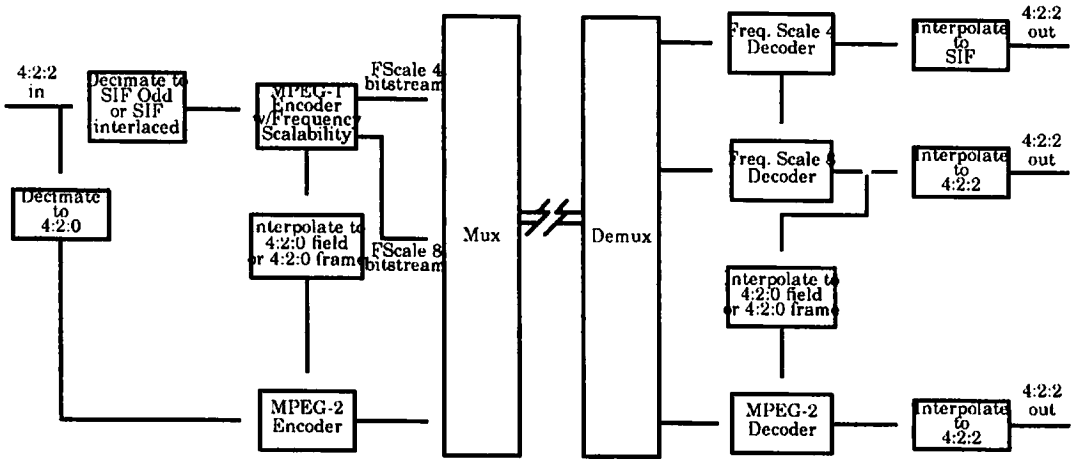


Fig. I.1 Functional Diagram of a 3 Layer (spatial and frequency) Scalable Hybrid Scheme

Filters used for decimation and interpolation are described in Sec. 3.3.2, 3.3.5, 3.4.2, and 3.4.5

For simulations, data rates are assigned to various layers according to Appendix G.4.1, MPEG-1 encoding parameters used are that from Appendix G.4.3, and CCIR 601 MPEG-2 encoding follows Appendix G.4.3.

At sequence layer both **sscalable** and **fscalable** are set to '1'. The **sscalable_code** and **fscale_code** can be derived from tables in Sec. 9.3.3.

The slice based multiplexing scheme used is that of Sec. 4.3.4.

I.11.2 Hybrid Experiment 1(b): A 4 layer Hybrid Scalable Scheme

In this experiment, an MPEG-1 encoder issued to code SIF odd fields, and the encoded data is organised into two frequency scales ("fscale4" and "fscale8") just like in Hybrid Experiment 1(a). The "fscale4" is base-layer and carries DCT coefficients, motion vectors, coded block pattern, and macroblock type overhead for all 330 macroblocks in each SIF field. The "fscale8" is slave layer and carries only the remaining DCT coefficients coded in MPEG-1 encoder. The base-layer, slave-layer relationship is exactly the same as that in frequency scalable experiments. For base-layer zig-zag scan and VLC's are the same as for frequency scalable experiments, whereas slave-layer uses MPEG-1 zig-zag scan and VLC's. The decoded "fscale8" (included "fscale4") macroblocks allow the choice of MC prediction from Odd fields in addition to MC prediction from previously decoded Even fields when coding SIF Even fields with MPEG-2 encoder. Decoded odd and even field (slices) are upsampled on a macroblock basis and allow spatial prediction in addition to other temporal predictions (MC mode) choices available when coding CCIR resolution and MPEG-2 encoder.

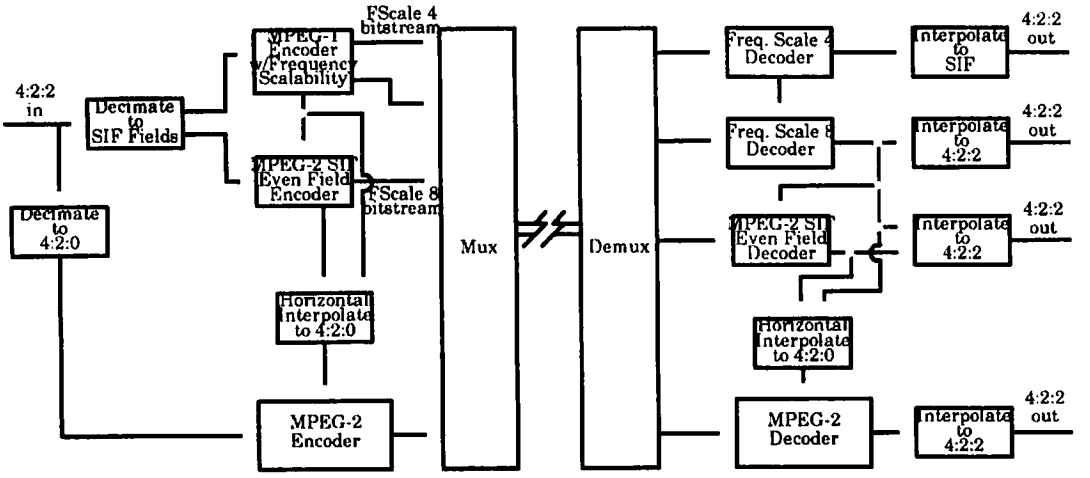


Fig. I.2 Function Diagram of a 4 Layer (spatial and frequency) Scalable Hybrid Scheme

Filters used for decimation and interpolation are described in Sec. 3.3.3 and 3.4.3.

For simulations, data rates are assigned to various layers according to G.2.1. MPEG-1 encoding parameters used are that from Appendix G.2.2, and CCIR 601 MPEG-2 encoding follows Appendix G.2.3.

At sequence layer both **sscalable** and **fscalable** are set to '1'. The **scale_code** and **fscale_code** can be derived from tables in Sec. 9.3.3. The slice based multiplexing scheme used is that of Sec. 4.3.5.

Appendix J: Delta on Frame/field prediction

In MBs that are field DCT coded, chrominance block structure is as follows :
o When the picture format is 4:2:2 and 4:4:4, the chrominance blocks structure is analogous to that of the luminance.
o When the picture format is 4:2:0, the chrominance blocks is structure is equal to that used for frame coded MBs. In other words, chroma is always frame coded.

It was agreed that when frame-based prediction is used in non-progressive pictures, the reference field for chroma prediction may not be the correct one.

There are four prediction motion vectors : PMV1, PMV2, PMV3 and PMV4. They are reset to zero at the start of a slice and at intra-coded MBs.

The prediction MVs (PMV1 to PMV4) *are always expressed in Frame MV coordinates.*
For the prediction of Field-based MVs (mv_type == "field"), the following rules are used:

- **On the decoder side :**
  When a Field-based MV is derived, the vertical coordinate of the PMV is shifted right by 1 bit (with sign extension) before adding the vertical differential.
  Then the Field-based MV is stored in the appropriate PMV(s) after shifting left by 1 bit its vertical coordinate.
- **On the encoder side :**
  When a Field-based MV is encoded, the vertical coordinate of the PMV is shifted right by 1 bit (with sign extension) before it is subtracted from the field MV vertical coordinate.
  Then the Field-based MV is stored in the appropriate PMV(s) after shifting left by 1 bit its vertical coordinate.

1. mv_type == "frame" :

In P-Frames or P-Fields, PMV1 is used. PMV2, PMV3 and PMV4 are reset to PMV1

In B-Frames or B-Fields, PMV1 is used for forward motion vector prediction, and PMV3 is used for backward motion vector prediction. PMV2 is reset to PMV1, and PMV4 is reset to PMV3.

2. mv_type == "field" :

In P-Frame or P-Field, PMV1 is used for motion vector prediction from field 1 , PMV3 is used for motion vector prediction from field 2. PMV2 is reset to PMV1, and PMV4 is reset to PMV3.

In B-Frame, PMV1 and PMV2 are used for forward motion vector prediction from field 1 and 2, and PMV3 and PMV4 are used for backward motion vector prediction from fields 1 and 2.

COMMENT : PMV2 AND PMV4 ARE IN FACT NEVER USED FOR MV PREDICTION IN P-PICTURES

A sequence can contain at the same time Field-Pictures and Frame-Pictures. However, in most experiments, the Field/Frame nature of the Pictures is predefined according to the picture-coding-type, and according to the hardware complexity that would be required to decode such sequences.

As guideline, the following levels are proposed for experiments (cf document 334):

Level 5 : (Equivalent to the Frame-Sequence of TM-1)
All frames are transmitted as Frame-picture
Level 4 :.
I- and P-frames are transmitted as Frame-picture.
B-Pictures are transmitted as Field-pictures only.
Level 4 (alternate) :.

P-Pictures are transmitted as Frame-picture.
I-Pictures are Field-pictures, field 2 being a P-field.
B-Pictures must be transmitted as Field-picture only.

Level 3 :

Each I-, P- and B-Picture is transmitted in Field-picture only.
B-Field are predicted using only 3 reference fields

Level 2 : (Equivalent to the Field-Sequence of TM-1 with 2 reference fields)

Each I-, P- and B-Picture is transmitted in Field-picture only.
B-Field are predicted using only 2 reference fields

Level 1 : (for low delay experiments)

Each Picture is transmitted in Field-picture.
No B-Picture is allowed.
P-Fields are predicted using 2 reference fields

Level 0 : (cheapest hardware)

Each Picture is transmitted in Field-picture.
No B-Picture is allowed.
P-Fields are predicted using only 1 reference field

Field 1 can be used as prediction for field 2 except in the case of B-Fields.

The number of fields that can be used for prediction is flexible when Field-Picture are used. For forward prediction, the minimum number of reference fields is 1, and the maximum is 2.

Reference fields for forward prediction are the latest two fields NOT part of a B-picture. This means in particular that when a Field-structure P-Picture is decoded, field 2 will use field 1 as one reference.
As a side effect, unless M=1, FAMC cannot be used to predict the field 2 of a Field-structure P-Picture.

In a field-structure B-picture, the closest two fields NOT part of a B-picture are used for prediction (forward and backward).
In a field-picture, the syntax of the size of a MB is 16x16 in both picture structures. Core experiments may be done for 16*8 sub-Macroblocks.

Simplified FAMC Prediction :
<<< described in separate document >>>

Dual Field Prediction (for Field-Pictures only)

P-Picture :
Two forward motion vectors to predict the pels of the field.
The prediction is obtained by averaging the two predictions obtained with the two motion vectors.

B-Picture :
Two forward motion vectors or two backward motion vectors are used to predict the pels of the field. In an interpolated MB, dual field prediction is not allowed.
The prediction is obtained by averaging the two predictions obtained with the two motion vectors.

Decision rules for : The mode giving the smallest MSE is selected.

Appendix K: Motion Compensation for Simplified FAMC

Basically, the motion compensation for FAMC is interpolation from four pixels in reference frame. The address calculations of these 4 points are defined in the next page, and it is illustrated in Fig. a.1. The horizontal (X) axis interpolation is rounded in half pixel precision because of hardware simplification, and the vertical interpolation is arithmetic precision interpolation.

```
   P=ref_frame(x_1stFi1,y_1stFi)  Q=ref_frame(x_1stFi2,y_1stFi)
1stFi line  - - - o - - - - - - @ - - - - - - o - - - - - - -
                                ^ ref_1stFi = (P + Q)
                              b :
                                v
                                * FAMC_MB(xl,yl) = (a*ref_1stFi +
b*ref_2ndFi)//16
                                ^
                              a :
                                v ref_2ndFi = (R + S)
2ndFi line    - - - o - - - - - @ - - - - - o - - - - - - -
   R=ref_frame(x_2ndFi1,y_2ndFi)    S=ref_frame(x_2ndFi2,y_2ndFi)
```

Fig. a.1 FAMC prediction for 1stFi line

```
Get_Simplified_FAMC_MB_for_Forward (Frame_Distance, Origin, FAMC_MV, FAMC_MB){
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
LeftPel = 0,  RightPel = 703,  Top1stFiLine = 0,  Bottom1stFiLine = 478,
Top2ndFiLine = 1,  Bottom2ndFiLine = 479
M = 16 // (2*Frame_Distance))       /* M = N * 16; N = 1/(2*Frame_Distance) */
                                    /*   = 8    When Frame_Distance=1  */
                                    /*   = 4    When Frame_Distance=2  */
                                    /*   = 3    When Frame_Distance=3  */

/* For first Field */
xorigin_1stFi1 = xorigin + (2 * FAMC_MVx)/2
xorigin_1stFi2 = xorigin + (2 * FAMC_MVx)//2
xorigin_2ndFi1 = xorigin + (2*FAMC_MVx - (M*FAMC_MVx)//8)/2
xorigin_2ndFi2 = xorigin + (2*FAMC_MVx - (M*FAMC_MVx)//8)//2
yorigin_1stFi  = yorigin + Adjacent_1stFi_Line_for_1st_Field_for_Forward
                           (Frame_Distance, FAMC_MVy)
yorigin_2ndFi   = yorigin + Adjacent_2ndFi_Line_for_1st_Field_for_Forward
                           (Frame_Distance, FAMC_MVy)

for (xl=0; xl<16; ++xl) {
   x_1stFi1 = xl + xorigin_1stFi1          /* Addressing X of pixel P      */
   x_1stFi2 = xl + xorigin_1stFi2          /* Addressing X of pixel Q      */
   x_2ndFi1 = xl + xorigin_2ndFi1          /* Addressing X of pixel R      */
   x_2ndFi2 = xl + xorigin_2ndFi2          /* Addressing X of pixel S      */

   /* In case that the required pixel is out of frame   */
   if (x_1stFi1 < LeftPel)  x_1stFi1 = LeftPel
   if (x_1stFi1 > RightPel) x_1stFi1 = RightPel
   if (x_2ndFi1 < LeftPel)  x_2ndFi1 = LeftPel
   if (x_2ndFi1 > RightPel) x_2ndFi1 = RightPel

   for (yl=0; yl<16; yl=yl+2) {
      y_1stFi = yl + yorigin_1stFi         /* Addressing Y of P & Q pixels */
      y_2ndFi = yl + yorigin_2ndFi         /* Addressing Y of R & S pixels */

      /* In case that the required pixel is out of frame   */
      if (y_1stFi < Top1stFiLine) y_1stFi = Top1stFiLine
      if (y_1stFi > Bottom1stFiLine) y_1stFi = Bottom1stFiLine
      if (y_2ndFi < Top2ndFiLine) y_2ndFi = Top2ndFiLine
      if (y_2ndFi > Bottom2ndFiLine) y_2ndFi = Bottom2ndFiLine

         /* interpolation     */
         ref_1stFi = ref_frame(x_1stFi1,y_1stFi) + ref_frame(x_1stFi2,y_1stFi)
         ref_2ndFi = ref_frame(x_2ndFi1,y_2ndFi) + ref_frame(x_2ndFi2,y_2ndFi)
         FAMC_MB(xl,yl) = (a*ref_1stFi + b*ref_2ndFi)/716
```

```
      }
   }
/* For Second Field */
xorigin_2ndFi1 = xorigin +(2 * FAMC_MVx)/2
xorigin_2ndFi2 = xorigin +(2 * FAMC_MVx)//2
xorigin_1stFi1 = xorigin +(2 * FAMC_MVx + (M*FAMC_MVx)//8)/2
xorigin_1stFi2 = xorigin +(2 * FAMC_MVx + (M*FAMC_MVx)//8)//2
yorigin_2ndFi  = yorigin + Adjacent_2ndFi_Line_for_2nd_Field_for_Forward
                           (Frame_Distance, FAMC_MVy)
yorigin_1stFi  = yorigin + Adjacent_1stFi_Line_for_2nd_Field_for_Forward
                           (Frame_Distance, FAMC_MVy)

for (xl=0; xl<16; ++xl) {
   x_2ndFi1 = xl + xorigin_2ndFi1          /* Addressing X of pixel R      */
   x_2ndFi2 = xl + xorigin_2ndFi2          /* Addressing X of pixel S      */
   x_1stFi1 = xl + xorigin_1stFi1          /* Addressing X of pixel P      */
   x_1stFi2 = xl + xorigin_1stFi2          /* Addressing X of pixel Q      */

   /* In case that the required pixel is out of frame   */
   if (x_2ndFi1 < LeftPel)  x_2ndFi1 = LeftPel
   if (x_2ndFi1 > RightPel) x_2ndFi1 = RightPel
   if (x_1stFi1 < LeftPel)  x_1stFi1 = LeftPel
   if (x_1stFi1 > RightPel) x_1stFi1 = RightPel

   for (yl=1; yl<16; yl=yl+2) {
      y_2ndFi = yl + yorigin_2ndFi         /* Addressing Y of R & S pixels */
      y_1stFi = yl + yorigin_1stFi         /* Addressing Y of P & Q pixels */

      /* In case that the required pixel is out of frame   */
      if (y_1stFi < Top1stFiLine) y_1stFi = Top1stFiLine
      if (y_1stFi > Bottom1stFiLine) y_1stFi = Bottom1stFiLine
      if (y_2ndFi < Top2ndFiLine) y_2ndFi = Top2ndFiLine
      if (y_2ndFi > Bottom2ndFiLine) y_2ndFi = Bottom2ndFiLine

         /* interpolation     */
         ref_2ndFi = ref_frame(x_2ndFi1,y_2ndFi) + ref_frame(x_2ndFi2,y_2ndFi)
         ref_1stFi = ref_frame(x_1stFi1,y_1stFi) + ref_frame(x_1stFi2,y_1stFi)
         FAMC_MB(xl,yl) = (a*ref_2ndFi + b*ref_1stFi)//16
      }
   }
}
```

Backward Motion Compensation is a little different from forward one because the relative position of both field of reference and predicted frame is inversed. That is, in forward prediction, 2nd field of reference is near to predicted frame, but in backward prediction, 1st field of reference is near to predicted one in time domain. Different points from forward are indicated by '^'.

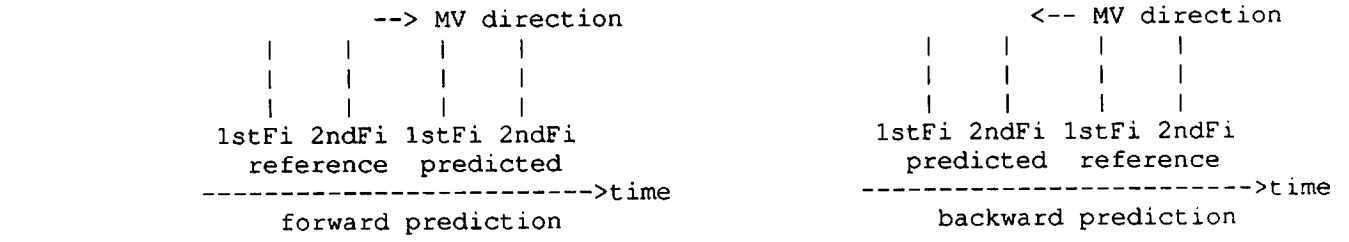


Fig a.2 difference of forward and backward prediction

```
Get_Simplified_FAMC_MB_for_Backward (Frame_Distance, Origin, FAMC_MV, FAMC_MB){
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
LeftPel = 0,  RightPel = 703,  Top1stFiLine = 0,  Bottom1stFiLine = 478,
Top2ndFiLine = 1,  Bottom2ndFiLine = 479
M = 16 // (2*Frame_Distance))       /* M = N * 16; N = 1/(2*Frame_Distance)   */
                                    /*   = 8    When Frame_Distance=1   */
                                    /*   = 4    When Frame_Distance=2   */
                                    /*   = 3    When Frame_Distance=3   */

/* For first Field */
```

```
xorigin_1stFi1 = xorigin + (2 * FAMC_MVx)/2
xorigin_1stFi2 = xorigin + (2 * FAMC_MVx)//2
xorigin_2ndFi1 = xorigin + (2*FAMC_MVx + (N*FAMC_MVx)//8)/2
                                ^^^
xorigin_2ndFi2 = xorigin + (2*FAMC_MVx + (N*FAMC_MVx)//8)//2
                                ^^^
yorigin_1stFi  = yorigin + Adjacent_1stFi_Line_for_1st_Field_for_Backward
                           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
                           (Frame_Distance, FAMC_MVy)
yorigin_2ndFi  = yorigin + Adjacent_2ndFi_Line_for_1st_Field_for_Backward
                           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
                           (Frame_Distance, FAMC_MVy)

for (xl=0; xl<16; ++xl) {
   x_1stFi1 = xl + xorigin_1stFi1          /* Addressing X of pixel P      */
   x_1stFi2 = xl + xorigin_1stFi2          /* Addressing X of pixel Q      */
   x_2ndFi1 = xl + xorigin_2ndFi1          /* Addressing X of pixel R      */
   x_2ndFi2 = xl + xorigin_2ndFi2          /* Addressing X of pixel S      */

   /* In case that the required pixel is out of frame   */
   if (x_1stFi1 < LeftPel)  x_1stFi1 = LeftPel
   if (x_1stFi1 > RightPel) x_1stFi1 = RightPel
   if (x_2ndFi1 < LeftPel)  x_2ndFi1 = LeftPel
   if (x_2ndFi1 > RightPel) x_2ndFi1 = RightPel

   for (yl=0; yl<16; yl=yl+2) {
      y_1stFi = yl + yorigin_1stFi         /* Addressing Y of P & Q pixels */
      y_2ndFi = yl + yorigin_2ndFi         /* Addressing Y of R & S pixels */

      /* In case that the required pixel is out of frame   */
      if (y_1stFi < Top1stFiLine) y_1stFi = Top1stFiLine
      if (y_1stFi > Bottom1stFiLine) y_1stFi = Bottom1stFiLine
      if (y_2ndFi < Top2ndFiLine) y_2ndFi = Top2ndFiLine
      if (y_2ndFi > Bottom2ndFiLine) y_2ndFi = Bottom2ndFiLine

         /* interpolation     */
         ref_1stFi = ref_frame(x_1stFi1,y_1stFi) + ref_frame(x_1stFi2,y_1stFi)
         ref_2ndFi = ref_frame(x_2ndFi1,y_2ndFi) + ref_frame(x_2ndFi2,y_2ndFi)
         FAMC_MB(xl,yl) = (a*ref_1stFi + b*ref_2ndFi)//16
      }
   }
/* For Second Field */
xorigin_2ndFi1 = xorigin +(2 * FAMC_MVx)/2
xorigin_2ndFi2 = xorigin +(2 * FAMC_MVx)//2
xorigin_1stFi1 = xorigin +(2 * FAMC_MVx - (N*FAMC_MVx)//8)/2
                                ^^^
xorigin_1stFi2 = xorigin +(2 * FAMC_MVx - (N*FAMC_MVx)//8)//2
                                ^^^
yorigin_2ndFi  = yorigin + Adjacent_2ndFi_Line_for_2nd_Field_for_Backward
                           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
                           (Frame_Distance, FAMC_MVy)
yorigin_1stFi  = yorigin + Adjacent_1stFi_Line_for_2nd_Field_for_Backward
                           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
                           (Frame_Distance, FAMC_MVy)

for (xl=0; xl<16; ++xl) {
   x_2ndFi1 = xl + xorigin_2ndFi1          /* Addressing X of pixel R      */
   x_2ndFi2 = xl + xorigin_2ndFi2          /* Addressing X of pixel S      */
   x_1stFi1 = xl + xorigin_1stFi1          /* Addressing X of pixel P      */
   x_1stFi2 = xl + xorigin_1stFi2          /* Addressing X of pixel Q      */

   /* In case that the required pixel is out of frame   */
   if (x_2ndFi1 < LeftPel)  x_2ndFi1 = LeftPel
   if (x_2ndFi1 > RightPel) x_2ndFi1 = RightPel
   if (x_1stFi1 < LeftPel)  x_1stFi1 = LeftPel
   if (x_1stFi1 > RightPel) x_1stFi1 = RightPel

   for (yl=1; yl<16; yl=yl+2) {
      y_2ndFi = yl + yorigin_2ndFi         /* Addressing Y of R & S pixels */
      y_1stFi = yl + yorigin_1stFi         /* Addressing Y of P & Q pixels */

      /* In case that the required pixel is out of frame   */
```

```
      if (y_1stFi < Top1stFiLine) y_1stFi = Top1stFiLine
      if (y_1stFi > Bottom1stFiLine) y_1stFi = Bottom1stFiLine
      if (y_2ndFi < Top2ndFiLine) y_2ndFi = Top2ndFiLine
      if (y_2ndFi > Bottom2ndFiLine) y_2ndFi = Bottom2ndFiLine

         /* interpolation     */
         ref_2ndFi = ref_frame(x_2ndFi1,y_2ndFi) + ref_frame(x_2ndFi2,y_2ndFi)
         ref_1stFi = ref_frame(x_1stFi1,y_1stFi) + ref_frame(x_1stFi2,y_1stFi)
         FAMC_MB(x1,y1) = (a*ref_2ndFi + b*ref_1stFi)//16
      }
   }
}
```

(Adjacent_XXX_Line_for_XXX_Field_for_XXX) functions are shown in Table 1.1 to 1.4, and the vertical interpolation coefficients (a,b) are shown in Table 2.1 to 2.2.

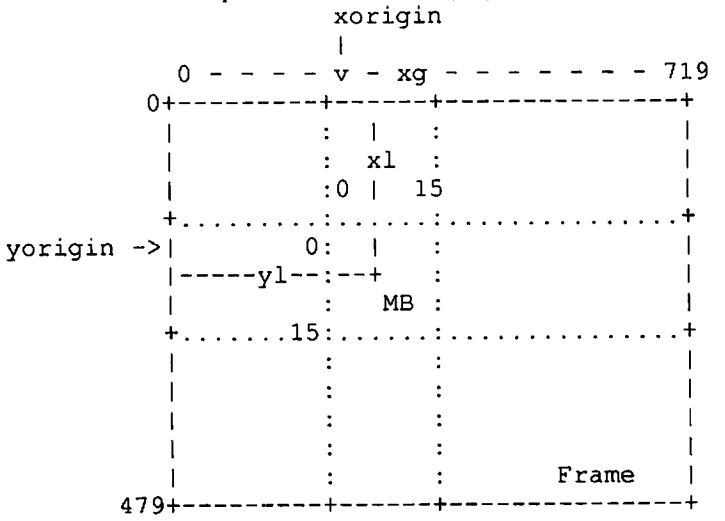


Fig. a.3

Prediction for Chrominance is calculated in the same manner of luminance with replacing the FAMC_MV to the vector which is derived by halving component value of the corresponding MB vector, using the formula from DC 11172;

```
right_for = (recon_right_for/2)>>1;
down_for = (recon_down_for/2)>>1;
right_half_for = recon_right_for/2 - 2*right_for;
down_half_for = recon_down_for/2 - 2*down_for;
```

An example of FAMC for luminance and chrominance are depicted in Fig. 1.

2. Motion Vector Estimation for FAMC

For ME simplification, 2 Step MV search algorithm is used.

1) Step 1 ; Integer pel, 2 line accuracy
In step 1, frame-base ME is performed with 2(V) x 1(H) accuracy.

```
Min_AE = MAXINT
for (j=(-YRange); j<(YRange+1); j+=2) {
   for (i=(-XRange); i<(XRange+1); ++i) {
      Get_Prediction_MB_by_Frame_Prediction (i, j, prediction_mb)
      AE_mb = AE_macroblock (current_mb, prediction_mb)
      if (AE_mb < Min_AE) {
         Min_AE = AE_mb
         FAMC_MV = (i,j)
      }
   }
}
```

2) Step 2 ; Half pel accuracy

In step 2, simplified FAMC based-ME is performed on the twenty neighboring positions which are evaluated the following order;

```
 1   2   3
 4   5   6
 7   8   9
10   0  11
12  13  14
15  16  17
18  19  20
```

where 0 represents the evaluated position in step 1.
Min_AE as a result of in Step 1 is used as an initial value in Step 2.

```
for (j=-3; j<4; ++j) {
   for (i=-1; i<2; ++i) {
      Get_Simplified_FAMC_MB_for_xxxx (Frame_Distance, Origin,
                  (FAMC_MVx_2int+0.5*i, FAMC_MVy_2int+0.5*j), FAMC_MB)
      AE_famc = AE_macroblock (current_mb, FAMC_MB)
      if (AE_famc < Min_AE) {
         Min_AE = AE_famc
         FAMC_MV = (FAMC_MVx_2int+0.5*i, FAMC_MVy_2int+0.5*j)
      }
   }
}
```

where (FAMC_MV_2int) represents the motion vector which is detected in step 1 motion estimation stage.

Table 1.1
Adj._1stFi_L._for_1st_F._for_Forward
Adj._2ndFi_L._for_2nd_F._for_Backward

FAMC_MVy	Frame_Distance 1	2	3
0.0	0	0	0
0.5	0	0	0
1.0	0	0	0
1.5	0 + 0	0 + 2	0 + 2
2.0	2	2	2
2.5	4	2	2
3.0	4	2	2
3.5	4	2	2
4.0	0	4	4
4.5	0	6	4
5.0	0	6	4
5.5	4 + 0	6	4
6.0	2	6	6
6.5	4	6	8
7.0	4	8	8
7.5	4	8	8
8.0	0	0	8
8.5	0	0	10
9.0	0	0	10
9.5	8 + 0	8 + 2	10
10.0	2	2	10
10.5	4	2	10
11.0	4	2	12
11.5	4	2	12
12.0	0	4	
12.5	0	6	
13.0	0	6	
13.5	12+ 0	6	12+
14.0	2	6	
14.5	4	6	
15.0	4	8	
15.5	4	8	
16.0	Repeated		
16.5			
17.0			

Table 1.2
Adj._2ndFi_L._for_1st_F._for_Forward
Adj._1stFi_L._for_2nd_F._for_Backward

FAMC_MVy	Frame_Distance 1	2	3
0.0	1	1	1
0.5	1	1	1
1.0	1	1	1
1.5	0 + 1	0 + 1	0 + 1
2.0	1	1	1
2.5	1	3	3
3.0	1	3	3
3.5	1	3	3
4.0	1	3	3
4.5	1	3	5
5.0	1	3	5
5.5	2 + 1	3	5
6.0	1	5	5
6.5	1	5	5
7.0	1	5	5
7.5	1	5	5
8.0	1	1	7
8.5	1	1	7
9.0	1	1	7
9.5	4 + 1	6 + 1	7
10.0	1	1	9
10.5	1	3	9
11.0	1	3	9
11.5	1	3	9
12.0	1	3	
12.5	1	3	
13.0	1	3	
13.5	6 + 1	3	10+
14.0	1	5	
14.5	1	5	
15.0	1	5	
15.5	1	5	
16.0	Repeated		
16.5			
17.0			

Table 1.3
Adj._1stFi_L._for_2nd_F._for_Forward
Adj._2ndFi_L._for_1st_F._for_Backward

FAMC_MVy	Frame_Distance 1	2	3
0.0	1	1	1
0.5	1	1	1
1.0	1	1	1
1.5	0 + 1	0 + 1	0 + 1
2.0	3	3	3
2.5	5	3	3
3.0	5	3	3
3.5	5	3	3
4.0	1	5	5
4.5	1	7	5
5.0	1	7	5
5.5	6 + 1	7	5
6.0	3	7	7
6.5	5	9	9
7.0	5	9	9
7.5	5	9	9
8.0	1	1	9
8.5	1	1	11
9.0	1	1	11
9.5	12+ 1	10+ 1	11
10.0	3	3	11
10.5	5	3	13
11.0	5	3	13

Table 1.4
Adj._2ndFi_L._for_2nd_F._for_Forward
Adj._1stFi_L._for_1st_F._for_Backward

FAMC_MVy	Frame_Distance 1	2	3
0.0	0	0	0
0.5	0	0	0
1.0	2	2	2
1.5	0 + 2	0 + 2	0 + 2
2.0	2	2	2
2.5	2	4	2
3.0	2	4	4
3.5	4	4	4
4.0	0	4	4
4.5	0	4	6
5.0	2	4	6
5.5	4 + 2	4	6
6.0	2	6	6
6.5	2	6	6
7.0	2	6	6
7.5	4	8	6
8.0	0	0	8
8.5	0	0	8
9.0	2	2	8
9.5	8 + 2	8 + 2	10
10.0	2	2	10
10.5	2	4	10
11.0	2	4	10

11.5	5	3	13
12.0	1	5	
12.5	1	7	
13.0	1	7	
13.5	18+ 1	7	14+
14.0	3	7	
14.5	5	9	
15.0	5	9	
15.5	5	9	
16.0	Repeated		
16.5			
17.0			

11.5	4	4	12
12.0	0	4	
12.5	0	4	
13.0	2	4	
13.5	12+ 2	4	12+
14.0	2	6	
14.5	2	6	
15.0	2	6	
15.5	4	8	
16.0	Repeated		
16.5			
17.0			

*In case of FAMC_MVy < 0, the sign of all figures in Table 1.1 - 1.4 should be changed to minus.

Table 2.1 (a,b)
(yl == 1stFi) in forward prediction
(yl == 2ndFi) in backward prediction

FAMC_MVy	Frame_Distance 1 a, b	2 a, b	3 a, b
-1.0	3, 5	2, 6	1, 7
-0.5	5, 3	4, 4	4, 4
0.0	8, 0	8, 0	8, 0
0.5	5, 3	4, 4	4, 4
1.0	3, 5	2, 6	1, 7
1.5	1, 7	2, 6	3, 5
2.0	8, 0	8, 0	8, 0
2.5	1, 7	6, 2	5, 3
3.0	3, 5	3, 5	3, 5
3.5	5, 3	2, 6	0, 8
4.0	8, 0	8, 0	8, 0
4.5	5, 3	2, 6	6, 2
5.0	3, 5	3, 5	4, 4
5.5	1, 7	6, 2	2, 6
6.0	8, 0	8, 0	8, 0
6.5	1, 7	2, 6	2, 6
7.0	3, 5	2, 6	4, 4
7.5	5, 3	4, 4	6, 2
8.0	8, 0	8, 0	8, 0
8.5	5, 3	4, 4	0, 8
9.0	3, 5	2, 6	3, 5
9.5	1, 7	2, 6	5, 3
10.0	8, 0	8, 0	8, 0
10.5	1, 7	6, 2	3, 5
11.0	3, 5	3, 5	1, 7
11.5	5, 3	2, 6	4, 4
12.0	8, 0	8, 0	
12.5	5, 3	2, 6	
13.0	3, 5	3, 5	
13.5	1, 7	6, 2	
14.0	8, 0	8, 0	
14.5	1, 7	2, 6	
15.0	3, 5	2, 6	
15.5	5, 3	4, 4	
16.0	Repeated		
16.5			
17.0			

Table 2.2 (a,b)
(yl == 2ndFi) in forward prediction
(yl == 1stFi) in backward prediction

FAMC_MVy	Frame_Distance 1 a, b	2 a, b	3 a, b
-1.0	3, 5	2, 6	1, 7
-0.5	3, 5	3, 5	4, 4
0.0	8, 0	8, 0	8, 0
0.5	3, 5	3, 5	4, 4
1.0	3, 5	2, 6	1, 7
1.5	6, 2	5, 3	5, 3
2.0	8, 0	8, 0	8, 0
2.5	6, 2	1, 7	1, 7
3.0	3, 5	3, 5	3, 5
3.5	3, 5	6, 2	5, 3
4.0	8, 0	8, 0	8, 0
4.5	3, 5	6, 2	1, 7
5.0	3, 5	3, 5	4, 4
5.5	6, 2	1, 7	6, 2
6.0	8, 0	8, 0	8, 0
6.5	6, 2	5, 3	6, 2
7.0	3, 5	2, 6	4, 4
7.5	3, 5	3, 5	1, 7
8.0	8, 0	8, 0	8, 0
8.5	3, 5	3, 5	5, 3
9.0	3, 5	2, 6	3, 5
9.5	6, 2	5, 3	1, 7
10.0	8, 0	8, 0	8, 0
10.5	6, 2	1, 7	5, 3
11.0	3, 5	3, 5	1, 7
11.5	3, 5	6, 2	4, 4
12.0	8, 0	8, 0	
12.5	3, 5	6, 2	
13.0	3, 5	3, 5	
13.5	6, 2	1, 7	
14.0	8, 0	8, 0	
14.5	6, 2	5, 3	
15.0	3, 5	2, 6	
15.5	3, 5	3, 5	
16.0	Repeated		
16.5			
17.0			

*In case of FAMC_MVy < 0, all coefficients (a,b) are cyclically repeated in Table 2.1 and 2.2.

Appendix L: Core experiments on prediction modes

Nine core experiments on prediction modes are listed in this appendix:

1. Simplified FAMC (Matsushita, Philips)
2. SVMC (KDD, TCE)
3. DUAL-PRIME (Dual') (Toshiba, GCT, Mitsubishi)
4. Global Motion Compensation (Matsushita, KDD)
5. Leaky Prediction 1 (JVC, AT&T)
6. Leaky Prediction 2 (CCITT, CCET)
7. Reverse Order Prediction (Columbia Univ., AT&T, Bellcore)
8. Simplification of Test Model (SONY, GCT, Mitsubishi)
9. 6x8 Macroblock (JVC, Columbia Univ., AT&T, Bellcore)

Core Experiment No.1 Simplified FAMC

The specification of simplified FAMC can be found in ANNEX B of MPEG 92/249.

1) The coefficient for interpolation is truncated to 3 bits, namely the multiplication of 1/8.

2) Simplified FAMC is not allowed in averaged MBs in B-pictures.

Simplified FAMC is applied to field structure same as frame structure. Attension should be paid in "field FAMC" with M=1, because the relative parity of the two reference fields are inversed for the first and second field, both in the forward and backward prediction direction.

Core Experiment No.2 SVMC

SVMC motion estimation is described in **A.1**. The sample program is shown in **A.2**. The corresponding syntax for SVMC is described in **A.3**. The reference and performance of SVMC can be found in MPEG92/246. Since SVMC already includes frame type prediction as well as field type prediction, such modes as frame and field prediction can be replaced or represented by SVMC which uses only one MV per MB. Although SVMC has four modes in it, three modes except simplified FAMC use the same fractional pel picture, and only the selection of prediction points differs as shown in A.1. As for simplified FAMC, interpolation mode in B-picture is excluded and fast search method is used, where first stage of this search method is common for all the four modes.

SVMC is also applicable to field picture by considering only one field(instead of frame) to predict for each time. Although the most part is the same, the complete description of SVMC for field picture will be specified at a later time.

L.2.1 SVMC motion estimation for frame picture

SVMC motion estimation is performed in two steps.

Step 1 Frame vector search
In this case, frame block search using integer pel accuracy is performed. For vertical direction, block matching is carried out for every other line(..-4,-2,0,+2,+4,..) which enables motion estimation of same parity field. By this line hopping, the amount of calculation is half of that of normal frame block matching in integer pel accuracy.

Step 2 SVMC search for 21 points for each mode
By using the motion vector obtained in the above as an offset, SVMC search is performed for limited search area. The area consists of +/-0.5 for horizontal direction and +/- 1.5 for vertical direction which makes total of 21[=3 (H) x 7 (V)] search points for each mode.

There are four prediction modes in SVMC.
a) Simplified FAMC
b) Same parity field MC
c) Near field MC
d) Modified dual field MC

The first one, simplified FAMC(Figure A1-(1)) is used where ME and MC are simplified according to MPEG92/249 and an interpolation mode in B-picture is excluded in order to reduce the memory band width (same condition as Core Experiment No.1 Simplified FAMC).

The second one, same parity field MC is performed by searching motion vector using the same parity field data as shown in Figure A1-(2). In this case, each point in each field is predicted using a point in the same field with 1/2 pel accuracy to horizontal and 1/4 pel to vertical direction.

The third one, near field MC is performed by searching motion vector using a near field in the time domain to the input picture as shown in Figure A1-(3). In this case, each point in both fields is predicted using a point in the nearest field in terms of time axis with 1/2 pel accuracy to horizontal and 1/4 pel to vertical direction.

The fourth one, modified dual field MC is performed by using the average data of two fields as shown in Figure A1-(4). In this case, each point in each field is predicted using two points in two fields with 1/2 pel accuracy to horizontal and 1/4 pel to vertical direction. In this fourth mode, an interpolation mode in B-picture is excluded.

In each mode a) - d), 21 points are searched using above prediction data and best position is searched in terms of MAD. Then mode decision is made by finding smallest MSE among four modes. When MSE is the same, the priority is given to b),c),a),d) order in the above modes.

Unlike FAMC, modes b) and c) do not require complicated weighting calculation by multiplier using two points,since these two schemes only accesses fixed points no matter how frame-distance value and motion vector may change on condition that pixel data of 1/2 pel accuracy in the horizontal and 1/4 pel in the vertical direction is obtained. Although mode d) requires access of two pixels for simple averaging, there is no need of multiplier.

As for chrominance, half value of motion vector is used in the same way as TM1.

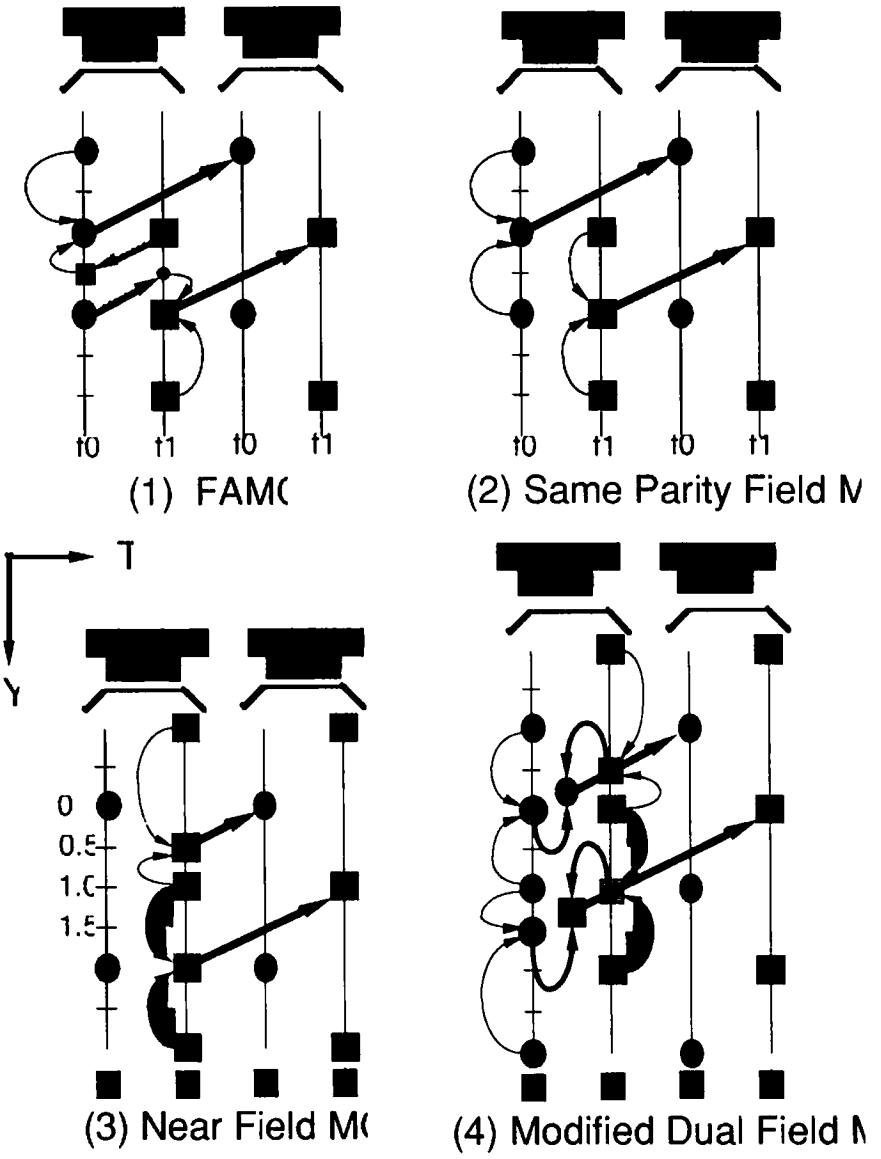


Figure L1 SVMC(Single Vector Motion Compensation) for frame picture

L.2.2 SVMC Program (Except FAMC) for frame structure

```
Get_SVMC_MB (mvx,mvy,XSRT,YSRT)
/*      mvx ... -1.5,-1.0,-0.5,0,0.5,1.0,1.5 ...
        mvy ... -1.5,-1.0,-0.5,0,0.5,1.0,1.5 ...
        XSRT ... MB horizontal position
        YSRT ... MB vertical position       */
{
/*      Same_Field_MB ... Same Parity Field Prediction MB
        Diff_Field_MB ... Different Parity Field Prediction MB
        Near_Field_MB ... Near Field Prediction MB
        Dual_Field_MB ... Field Interpolation MB
        EtoE ... Field Distance from Reference Field Even to Predicted Field Even
        EtoO ... Field Distance from Reference Field Odd to Predicted Field Even
        OtoO ... Field Distance from Reference Field Odd to Predicted Field Odd
        OtoE ... Field Distance from Reference Field Even to Predicted Field Odd*/

        if (Forward Prediction) {
                EtoE = Frame_Distance*2 ;
```

```
                EtoO = Frame_Distance*2-1 ;
                OtoO = Frame_Distance*2 ;
                OtoE = Frame_Distance*2+1 ;
        }
        else {
                EtoE = Frame_Distance*2 ;
                EtoO = Frame_Distance*2+1 ;
                OtoO = Frame_Distance*2 ;
                OtoE = Frame_Distance*2-1 ;
        }

/* Positioning (Predicted Field = Even) */
        /* Halfpel horizontal position to Even Field */
        xofs_even1 = (mvx*2)/2 ;
        xofs_even2 = (mvx*2)//2 ;

        /* Quaterpel vertical position to Even Field */
        yofs_even1 = ((mvy*2)/4)*2 ;
        if (mvy  < 0)  yofs_even2 = (((mvy*2)-3)/4)*2;
        else           yofs_even2 = (((mvy*2)+3)/4)*2;

        /* Halfpel horizontal position to Odd Field */
        xofs_odd1 = ((mvx*2*EtoO)//EtoE)/2 ;
        xofs_odd2 = ((mvx*2*EtoO)//EtoE)//2 ;

        /* Quaterpel vertical position to Odd Field */
        if (mvy < 0) {
                yofs_odd1 = (((mvy*2*EtoO)//EtoE - 2)/4)*2 + 1 ;
                yofs_odd2 = (((mvy*2*EtoO)//EtoE - 5)/4)*2 + 1 ;
        }
        else {
                yofs_odd1 = (((mvy*2*EtoO)//EtoE + 2)/4)*2 - 1 ;
                yofs_odd2 = (((mvy*2*EtoO)//EtoE + 5)/4)*2 - 1 ;
        }

        /* Weighting Parameter for Same Parity Field Prediction */
        if (yofs_even1 == yofs_even2) {
                same_wt1 = 4 ;
                same_wt2 = 0 ;
        }
        else {
                same_wt1 = absolute (yofs_even2*2-(mvy*2)) ;
                same_wt2 = absolute (yofs_even1*2-(mvy*2)) ;
        }

        /* Weighting Parameter for Difference Parity Field Prediction */
        if (yofs_odd1 == yofs_odd2) {
                diff_wt1 = 4 ;
                diff_wt2 = 0 ;
        }
        else {
                diff_wt1 = absolute (yofs_odd2*2 - (mvy*2*EtoO)//EtoE) ;
                diff_wt2 = absolute (yofs_odd1*2 - (mvy*2*EtoO)//EtoE) ;
        }

/* Prediction main (Predicted Field = Even) */
        for (y = 0;y < 16;y += 2)
                for (x = 0;x < 16;x ++) {
```

```
                    /* Even Field to Even Field Prediction */
                    x_even1 = XSRT + x + xofs_even1 ;
                    x_even2 = XSRT + x + xofs_even2 ;
                    y_even1 = YSRT + y + yofs_even1 ;
                    y_even2 = YSRT + y + yofs_even2 ;
                    ref_evenx1 = ref_frame[y_even1][x_even1] + ref_frame[y_even1][x_even2] ;
                    ref_evenx2 = ref_frame[y_even2][x_even1] + ref_frame[y_even2][x_even2] ;
                    Same_Field_MB[y][x] = (same_wt1*ref_evenx1 + same_wt2*ref_evenx2)//8 ;

                    /* Odd Field to Even Field Prediction */
                    x_odd1 = XSRT + x + xofs_odd1 ;
                    x_odd2 = XSRT + x + xofs_odd2 ;
                    y_odd1 = YSRT + y + yofs_odd1 ;
                    y_odd2 = YSRT + y + yofs_odd2 ;
                    ref_oddx1 = ref_frame[y_odd1][x_odd1] + ref_frame[y_odd1][x_odd2] ;
                    ref_oddx2 = ref_frame[y_odd2][x_odd1] + ref_frame[y_odd2][x_odd2] ;
                    Diff_Field_MB[y][x] = (diff_wt1*ref_oddx1 + diff_wt2*ref_oddx2)//8 ;

                    if (Forward Prediction)    Near_Field_MB[y][x] = Diff_Field_MB[y][x] ;
                    else                       Near_Field_MB[y][x] = Same_Field_MB[y][x] ;
                }

/* Positioning (Predicted Field = Odd) */
        /* Halfpel horizontal position to Odd Field */
        xofs_odd1 = (mvx*2)/2 ;
        xofs_odd2 = (mvx*2)//2 ;

        /* Quaterpel vertical position to Odd Field */
        yofs_odd1 = ((mvy*2)/4)*2 ;
        if (mvy  < 0)  yofs_odd2 = (((mvy*2)-3)/4)*2;
        else        yofs_odd2 = (((mvy*2)+3)/4)*2;

        /* Halfpel horizontal position to Even Field */
        xofs_even1 = ((mvx*2*OtoE)//OtoO)/2 ;
        xofs_even2 = ((mvx*2*OtoE)//OtoO)//2 ;

        /* Quaterpel vertical position to Even Field */
        if (mvy < 0) {
                yofs_even1 = (((mvy*2*OtoE)//OtoO - 2)/4)*2 + 1 ;
                yofs_even2 = (((mvy*2*OtoE)//OtoO - 5)/4)*2 + 1 ;
        }
        else {
                yofs_even1 = (((mvy*2*OtoE)//OtoO + 2)/4)*2 - 1 ;
                yofs_even2 = (((mvy*2*OtoE)//OtoO + 5)/4)*2 - 1 ;
        }

        /* Weighting Parameter for Same Parity Field Prediction */
        if (yofs_odd1 == yofs_odd2) {
                same_wt1 = 4 ;
                same_wt2 = 0 ;
        }
        else {
                same_wt1 = absolute (yofs_odd2*2-(mvy*2)) ;
                same_wt2 = absolute (yofs_odd1*2-(mvy*2)) ;
        }

        /* Weighting Parameter for Difference Parity Field Prediction */
        if (yofs_even1 == yofs_even2) {
```

```
                diff_wt1 = 4 ;
                diff_wt2 = 0 ;
        }
        else {
                diff_wt1 = absolute (yofs_even2*2 - (mvy*2*OtoE)//OtoO) ;
                diff_wt2 = absolute (yofs_even1*2 - (mvy*2*OtoE)//OtoO) ;
        }

/* Prediction main (Predicted Field = Odd)*/
        for (y = 1;y < 16;y += 2)
                for (x = 0;x < 16;x ++) {
                        /* Odd Field to Odd Field Prediction */
                        x_odd1 = XSRT + x + xofs_odd1 ;
                        x_odd2 = XSRT + x + xofs_odd2 ;
                        y_odd1 = YSRT + y + yofs_odd1 ;
                        y_odd2 = YSRT + y + yofs_odd2 ;
                        ref_oddx1 = ref_frame[y_odd1][x_odd1] + ref_frame[y_odd1][x_odd2] ;
                        ref_oddx2 = ref_frame[y_odd2][x_odd1] + ref_frame[y_odd2][x_odd2] ;
                        Same_Field_MB[y][x] = (same_wt1*ref_oddx1 + same_wt2*ref_oddx2)//8 ;

                        /* Even Field to Odd Field Prediction */
                        x_even1 = XSRT + x + xofs_even1 ;
                        x_even2 = XSRT + x + xofs_even2 ;
                        y_even1 = YSRT + y + yofs_even1 ;
                        y_even2 = YSRT + y + yofs_even2 ;
                        ref_evenx1 = ref_frame[y_even1][x_even1] + ref_frame[y_even1][x_even2] ;
                        ref_evenx2 = ref_frame[y_even2][x_even1] + ref_frame[y_even2][x_even2] ;
                        Diff_Field_MB[y][x] = (diff_wt1*ref_evenx1 + diff_wt2*ref_evenx2)//8 ;

                        if (Forward Prediction)    Near_Field_MB[y][x] = Same_Field_MB[y][x] ;
                        else                       Near_Field_MB[y][x] = Diff_Field_MB[y][x] ;
                }

/* Dual Field Prediction */
        for (y = 0;y < 16;y ++)
                for (x = 0;x < 16;x ++)
                        Dual_Field_MB[y][x] = (Same_Field_MB[y][x]
                                                + Diff_Field_MB[y][x])//2 ;
}
```

L.2.3 Syntax change corresponding to SVMC

Macroblock Layer

```
macroblock() {
            .
            .
            .
    if (interlaced) {   /* not MPEG-1 syntax */
      if ( macroblock_motion_forward ||
                  macroblock_motion_backward ) {
        if ( picture_structure == "11" ) {  /* Frame-Picture */
         frame_motion_type                          2          uimsbf
          if ( frame_motion_type == "11" )
          SVMC_type                                 2          uimsb
        } else {
         field_motion_type                          2          uimsbf
          if ( field_motion_type == "11" )
          SVMC_type                                 2          uimsbf
        }
      }
      if ( ( picture_structure == "11" ) /* Frame-Picture */
            && ( macroblock_intra || macroblock_pattern ) )
        dct_type                                    1          uimsbf
    }
            .
            .
            .
  }
```

SVMC_type - this is two-bit integer indicating the macroblock motion prediction,defined in the following table:

binary value	Motion Prediction
00	Same Parity Field Prediction
01	Near Field Prediction(frame picture) Diff.Parity Field Prediction(field picture)
10	Simplified FAMC
11	Modified Dual Field Prediction

field_motion_type - This is a 2-bit code indicating the macroblock motion prediction, defined in the following table:

code	prediction type	motion_vector_count	mv_format
11	CORE EXPERIMENTS(SVMC)	1	frame

frame_motion_type - This is a 2-bit code indicating the macroblock motion prediction, defined in the following table:

code	prediction type	motion_vector_count	mv_format
11	CORE EXPERIMENTS(SVMC)	1	frame

If you have any questions or request on MV, please contact to nakajima@spg.elb.kddlabs.co.jp
or Y.Nakajima, KDD R&D Labs., tel +81(492)66-7891, fax +81(492)66-7510

Core Experiment No. 3 DUAL-PRIME (Dual')

DUAL-PRIME prediction is an improved version of dual field prediction defined in TM1, reducing overhead for transmitting motion vectors. It needs only one field motion vector and one additional very small differential vector per macroblock. Its operation is considered to be both improvement of vertical resolution and adaptive spatiotemporal loop filtering. No pel by pel basis multiplication is needed and its decoder memory bandwidth is one-half that for original FAMC.

All the information for DUAL-PRIME is described in MPEG92/259, according to the syntax and the field motion vector transmission method of TM1. This document is the revised version of the syntax, decoding process and motion vector estimation method for DUAL-PRIME, according to the syntax and the field motion vector transmission method of TM2 with extension to the field-picture.

1. Syntax Extension for Dual' in TM2

```
-------------------------------------------------------
| forward_motion_vectors() {                          |
|     if (motion_vector_count == 1) {                 |
|         if (mv_format == frame) {                   |
|             forward_motion_vector()        ...  ... |
|         } else {                                    |
|             forward_field_motion_vector()  ...  ... |
I             if (frame_motion_type == '11'           I
I              || field_motion_type == '11')          I
I               dmv()                          ...  ... I
|         }                                           |
|     } else {                                        |
|         forward_field_motion_vector_1()    ...  ... |
|         forward_field_motion_vector_2()    ...  ... |
|     }                                               |
| }                                                   |
-------------------------------------------------------
```

```
-------------------------------------------------------
| backward_motion_vectors() {                         |
|     if (motion_vector_count == 1) {                 |
|         if (mv_format == frame) {                   |
|             backward_motion_vector()       ...  ... |
|         } else {                                    |
|             backward_field_motion_vector() ...  ... |
I             if (frame_motion_type == '11'           I
I              || field_motion_type == '11')          I
I                dmv()                          ...  ... I
|         }                                           |
|     } else {                                        |
|         backward_field_motion_vector_1()   ...  ... |
|         backward_field_motion_vector_2()   ...  ... |
|     }                                               |
| }                                                   |
-------------------------------------------------------
```

```
-------------------------------------------------------
I dmv() {                                             I
I     dmv_horizontal                    1-2     vlclbf I
I     dmv_vertical                      1-2     vlclbf I
I }                                                   I
-------------------------------------------------------
```

```
-------------------------------------------------------------
| forward_field_motion_vector() {                           |
I     if (forward_reference_fields == '11'                  I
I      && !(frame_motion_type == '11' ||                    I
I           field_motion_type == '11'))                     I
|        motion_vertical_field_select       1       uimsbf |
|     }                                                     |
|     motion_vector()                       ...     ...    |
| }                                                         |
-------------------------------------------------------------


-------------------------------------------------------------
| backward_field_motion_vector() {                          |
I     if (backward_reference_fields == '11'                 I
I      && !(frame_motion_type == '11' ||                    I
I           field_motion_type == '11'))                     I
|        motion_vertical_field_select       1       uimsbf |
|     }                                                     |
|     motion_vector()                       ...     ...    |
| }                                                         |
-------------------------------------------------------------
```

field_motion_type -- This is a 2-bit code indicating the macroblock motion prediction, defined in the following table:

code	prediction type	motion_vector_count	mv_format
00	Field-based prediction	1	field
01	Dual-field prediction	2	field
10	Simplified FAMC	1	frame
11	Core Experiments (Dual')	1	field

frame_motion_type -- This is a 2-bit code indicating the macroblock motion prediction, defined in the following table:

code	prediction type	motion_vector_count	mv_format
00	Field-based prediction	2	field
01	Frame-based prediction	1	frame
10	Simplified FAMC	1	frame
11	Core Experiments (Dual')	1	field

VLC table for dmv_horizontal/vertical

dmv_horizontal/vertical	value
0	0
10	1
11	-1

2. Decoding process of motion vectors

In dual' field based forward prediction, the following procedure is used to reconstruct the motion vectors. In backward prediction, it is also valid except for replacing the term 'for' to 'back'. If field_motion_type is dual' ('11') or frame_motion_type is dual' ('11'), foward_field_motion_vector() and dmv() are transmitted.

Let recon_{right/down}_for means the horizontal/vertical component of MV_for, which is reconstructed from foward_field_motion_vector(). Let dmv_{horizontal/vertical}_for means the horizontal/vertical component of DMV_for, which is reconstructed from dmv().

Let recon_{right/down}_for_OtoO, recon_{right/down}_for_EtoO, recon_{right/down}_for_EtoE and recon_{right/down}_for_OtoE mean the horizontal/vertical component of MV_for_OtoO, MV_for_EtoO, MV_for_EtoE and MV_for_OtoE.

Let dist_OtoO, dist_EtoO, dist_EtoE and dist_OtoE be the field distance between the {odd(#1)/even(#2)} reference field and the {odd(#1)/even(#2)} field to be decoded. Note that dist_OtoO and dist_EtoE is equal to the value 2*Frame_distance in FAMC description.

```
/* Substitute the decoded motion vector for the motion
   vector of same parity */
recon_right_for_OtoO = recon_right_for_EtoE = recon_right_for;
recon_down_for_OtoO = recon_down_for_EtoE = recon_down_for;

/* Derive the cross point of MV_for_OtoO in the even reference field,
    truncate it to the nearest one-half pixel unit point toward zero,
    add DMV_for to get MV_for_EtoO
    and express it in the opposite parity field coordinate */
recon_right_for_EtoO = ((recon_right_for_OtoO * dist_EtoO) / dist_OtoO)
                     + dmv_horizontal_forward;
recon_down_for_EtoO = ((recon_down_for_OtoO * dist_EtoO) / dist_OtoO)
                    + dmv_vertical_forward - 1;

/* Derive the cross point of MV_for_EtoE in the odd reference field,
    truncate it to the nearest one-half pixel unit point toward zero,
    add DMV_for to get MV_for_OtoE
    and express it in the opposite parity field coordinate */
recon_right_for_OtoE = ((recon_right_for_EtoE * dist_OtoE) / dist_EtoE)
                     + dmv_horizontal_forward;
recon_down_for_OtoE = ((recon_down_for_EtoE * dist_OtoE) / dist_EtoE)
                    + dmv_vertical_forward + 1;
```

For a Frame-Picture (picture_structure == '11'), all of the procedure mentioned above is executed to reconstruct the motion vectors. For a Field-Picture (picture_structure == '10' or picture_structure == '01'), part of the procedure mentioned above is executed to reconstruct the motion vectors. That is, if the parity of the field to be decoded is odd (picture_structure == '10'), the procedure related with *_OtoO and *_EtoO should be executed, and if the parity of the field to be decoded is even (picture_structure == '01'), the procedure related with *_EtoE and *OtoE should be executed.

3. Motion vector estimation process

At the first step, org_MV_for_{O/E}to{O/E}, which is the field motion vector from the odd/even field to be referenced, to the odd/even field to be coded, is derived, according to the field motion vector estimation process of this TM with half pel search using original pictures. The second step is as follows:

```
min_AE = MAXINT;
foreach i (OtoO EtoO EtoE OtoE) {
     /* $i may be OtoO, EtoO, EtoE or OtoE (4 candidates) */
     if ($i == 'OtoO' || $i == 'EtoE') {
          /* org_MV_$i is the motion vector for the field of same parity */
          recon_right_for = recon_right_for_OtoO
                          = recon_right_for_EtoE = org_recon_right_for_$i;
          recon_down_for = recon_down_for_OtoO
                         = recon_down_for_EtoE = org_recon_down_for_$i;
     }
```

```
else {
     /* org_MV_$i is the motion vector for the field
        of opposite parity */
     recon_right_for = recon_right_for_0to0 = recon_right_for_EtoE
                     = (org_recon_right_for_$i * dist_0to0) / dist_$i;

     /* If ($i == 'Eto0'), the vertical component of the motion vector
        should be incremented by one to be expressed in the same
        parity field coordinate.
        If ($i == '0toE'), the vertical component of the motion vector
        should be decremented by one to be expressed in the same
        parity field coordinate. */
     recon_down_for = recon_down_for_0to0 = recon_down_for_EtoE
                    = ((org_recon_down_for_$i + ($i == 'Eto0') * 1
                    + ($i == '0toE') * (-1)) * dist_0to0) / dist_$i;
}

/* Cut the local decoded reference fields of the same parity
   according to the motion vectors. */
cut_ref(recon_right_for_0to0, recon_down_for_0to0, 0dd_reference,
        lines_of_Odd_field_of_MB1, MBadrs);
cut_ref(recon_right_for_EtoE, recon_down_for_EtoE, Even_reference,
        lines_of_Even_field_of_MB1, MBadrs);

for (dmv_vertical_forward = -1;dmv_vertical_forward<=1;
                            dmv_vertical_forward++) {
     for (dmv_horizontal_forward = -1;dmv_horizontal_forward<=1;
                                 dmv_horizontal_forward++) {
          /* DMV's are prepared (9 candidates) */

          /* Derive MV_for_Eto0 from MV_for_0to0.
             The vertical component should be decremented by one,
             according to the TM2 motion vector definition. */
          recon_right_for_Eto0 = ((recon_right_for_0to0 * dist_Eto0)
                               / dist_0to0) + dmv_horizontal_forward;
          recon_down_for_Eto0 = ((recon_down_for_0to0 * dist_Eto0)
                              / dist_0to0)  + dmv_vertical_forward - 1;

          /* Derive MV_for_0toE from MV_for_EtoE.
             The vertical component should be incremented by one,
             according to the TM2 motion vector definition. */
          recon_right_for_0toE = ((recon_right_for_EtoE * dist_0toE)
                               / dist_EtoE) + dmv_horizontal_forward;
          recon_down_for_0toE = ((recon_down_for_EtoE * dist_0toE)
                              / dist_EtoE) + dmv_vertical_forward + 1;

          /* Cut the local decoded reference fields of the opposite
             parity according to the motion vectors. */
          cut_ref(recon_right_for_Eto0, recon_down_for_Eto0,
                  Even_reference, lines_of_Odd_field_of_MB2, MBadrs);
          cut_ref(recon_right_for_0toE, recon_down_for_0toE,
                  Odd_reference, lines_of_Even_field_of_MB2, MBadrs);

          /* Interpole the fields of both parity */
          interpole(lines_of_Odd_field_of_MB1,
                    lines_of_Odd_field_of_MB2,
                    lines_of_Odd_field_of_prediction_MB);
          interpole(lines_of_Even_field_of_MB1,
                    lines_of_Even_field_of_MB2,
                    lines_of_Even_field_of_prediction_MB);

          /* Calculate the absolute error */
          AE_Odd = AE_Macroblock(lines_of_Odd_field_of_prediction_MB,
                                 lines_of_Odd_field_of_current_MB);
          AE_Even = AE_Macroblock(lines_of_Even_field_of_prediction_MB,
                                  lines_of_Even_field_of_current_MB);
          AE = AE_Odd + AE_Even;
          if (AE < min_AE) {
               min_AE = AE;
               MV_for = (recon_right_for, recon_down_for);
               DMV_for = (dmv_horizontal_forward, dmv_vertical_forward);
          }
```

```
            }
        }
}
```

cut_ref(recon_right, recon_down, Reference, Buf, MBadrs): The reference signals pointed by recon_{right/down}_for and MBadrs in field buffer "Reference" are copied to macroblock buffer "Buf".
interpole(Buf1, Buf2, Buf): Average signals of buffer "Buf1" and "Buf2" are stored in buffer "Buf".

For a Frame-Picture, all of the procedure mentioned above is executed to reconstruct the motion vectors (number of candidates are 4*9 = 36). Note that MB1, MB2, prediction_MB and current_MB are frame macroblocks, and odd/even lines in them are in the odd/even field.

For a Field-Picture, part of the procedure mentioned above is executed to reconstruct the motion vectors. If the parity of the field to be coded is odd, the procedure related with *_OtoO and *_EtoO should be executed (number of candidates are 2*9 = 18). Note that MB1, MB2, prediction_MB and current_MB are field macroblocks, and all lines in them are in the odd field. If the parity of the field to be coded is even, the procedure related with *_EtoE and *OtoE should be executed (number of candidates are 2*9 = 18).
Note that MB1, MB2, prediction_MB and current_MB are field macroblocks, and all lines in them are in the even field.

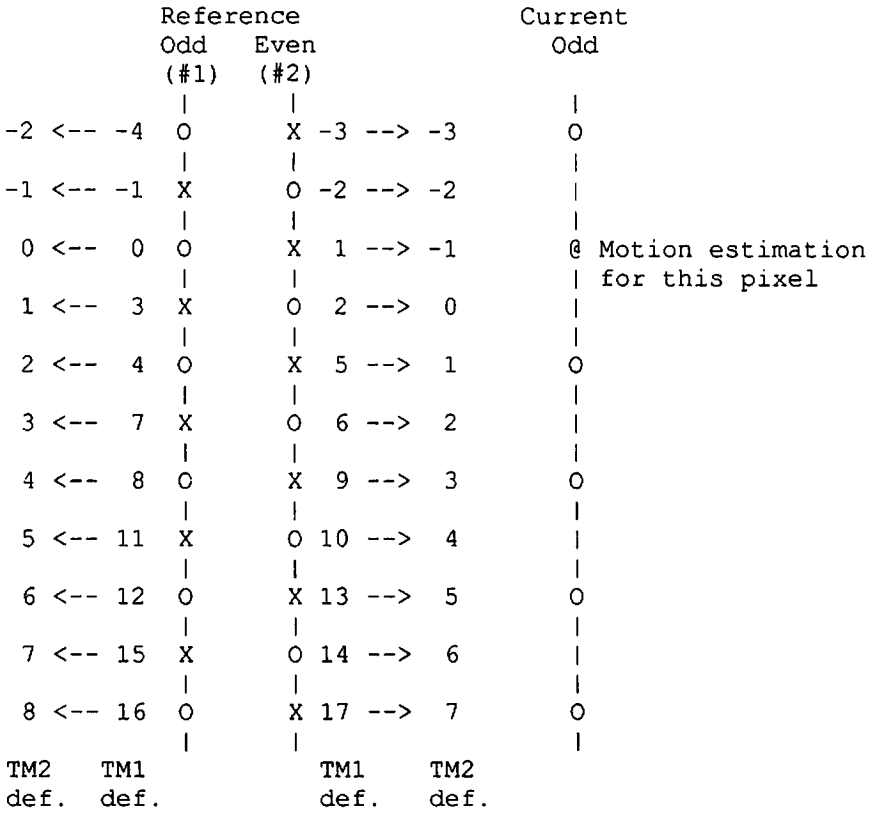
Note:
1) In motion vector estimation process, if the motion vector, obtained by the extension of the vector and summation of dmv, points out of the picture, the candidate is not used.
2) In both motion vector estimation process and decoding process of motion vectors, if the value of the half pixel point located in the edge of the picture must be used, the value of the integer pixel point next to it is used.

If you have some questions, please contact to: odaka@eel.rdc.toshiba.co.jp
or T.Odaka, Toshiba R&D center, fax +81(44)549-2276

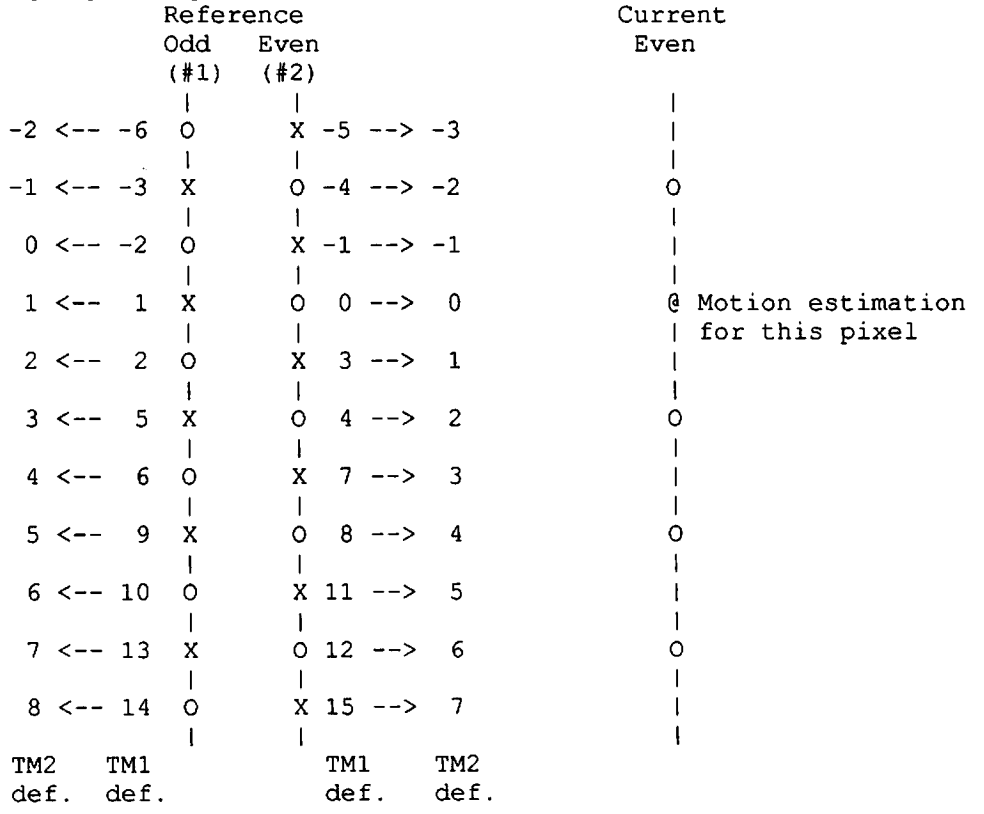
4. Supplimentary explanation for Figure A-1 in MPEG92/259.

The notation '(TM1 def.) --> (TM2 def.)' means the replacement of motion vector expressed in TM1 definition with motion vector expressed in this TM definition in half-pel unit.

[Lower left figure of Figure A-1 in MPEG92/259]



[Lower right figure of Figure A-1 in MPEG92/259]



Core Experiment No.4 Global Motion Compensation

Global motion compensation is applied to I-pictures coded in frame mode. It is a kind of pre-processing that shifts the relative position between the two fields in an I-picture based on a pair of global MV. After reconstruction, the relative position between the two fields has to be shifted back to the original position for display (at decoder side) or as the reference picture (at encoder side).

1. Global Motion Estimation

1) Global MV is determined from the luminance pixels only.
2) Fields 1 and 2 of an I-picture are splitted. Field 2 is fixed while field 1 becomes the reference field.
3) The motions between fields 1 and 2 of an I-picture is estimated in full pel precision horizontally and half pel precision vertically.Thus, the reference field is composed of field 1 plus its vertical interpolation.
4) A pair of motion vector is determined for each macroblock by full search block matching method under the minimum absolute error criterion. The block size in each field is 16(H)x8(V). The difference is always between fields 2 and 1, or between field 2 and the interpolated lines of the reference field.
5) The search area is set to half of the search area for inter-frame motion vectors as the global motion between two fields cannot be larger than one half of the motions between two frames. Thus, for an inter-frame motion search range of +- 15 pels, the inter-field motion search range is +- 7 pels.
6) With the MV's obtained above, a pair of global MV is determined by taking the average of the components which occupy more than a certain percentage (majority) of all the MV's. This is calculated separately for the x and y components.
7) For the y component, GlobalMV_y is modified as follows to avoid breaking down the interlaced structure of the original picture.:

```
if (GlobalMV_y == negative odd number)
        GlobalMV_y = GlobalMV_y + 1;
if (GlobalMV_y == positive odd number)
        GlobalMV_y = GlobalMV_y - 1;
if (GlobalMV_y == even number)
        GlobalMV_y = GlobalMV_y;
```

2 Global Motion Compensation

1) Global motion compensation is applied to the luminance of I-pictures only.
2) Based on the pair of global MV obtained, field 1 of an I-picture is shifted in full pel precision both horizontally and vertically.
3) If meaningful pixels cover the whole 720 pels of the lines and the horizontal global motion compensation is performed before the cropping. If meaningful pixels cover exactly the 704 pel area, every other lines at one side of the borders will be left with no meaningful pixels after the compensation. One can fill up the vacancies by repeating the nearest pixel in each line.
4) Every other lines at one side of the horizontal borders will be left with no meaningful lines after the vertical global motion compensation.The values of the nearest line can be repeated to fill up the vacancies.
5) The global-motion-compensated I-picture is then encoded as usual.
6) The pair of global MV is included in the bit stream as describe in the next section.
7) At the decoder side, the global MV is used to shift the field back to its original position after decoding and reconstruction. The vertical and horizontal borders are dealt with one of the methods described in 3) and 4), respectively.

3. Modification of the Bit Stream Syntax

The global motion vector (GlobalMV_x, GlobalMV_y) is appended to the Picture Layer. These parameters are integers, range from -7 through +7 or -15 through +15. They are offset by 15 to produce unsigned integers. In the modified syntax, the global_motion_flag is a one bit integer. It is set to 1 when there is global motion, and to 0 if (GlobalMV_x, GlobalMV_y) = (0, 0).

```
if (nextbits() == extension_start_code ) {
    extension_start_code                    32          bslbf
    picture_structure                       2           uimsbf
    forward_reference_fields                2           uimsbf
    backward_reference_fields               2           uimsbf
  global_motion_flag                        1           uimsbf
    if (global_motion_flag==1){
        GlobalMV_x                          5           uimsbf
        GlobalMV_y                          5           uimsbf
    }
     .....
}
```

If you have some questions, please contact to :
toshi@drl.mei.co.jp
or Toshiya Takahashi, Matsushita,
FAX +81-6-949-2218.

Core Experiment No. 5 Leaky Prediction 1

Purpose: To verify the coding efficiency , improvement of visual quality and accessibility
Definition ;
Leaky Prediction 1 (Leaked MC filter)
1. Rational leak prediction for high spatial frequency (stable loop filter) is used.
2. Apply P-prediction and B-prediction with out FAMC
3. Modify the coefficients of half-pel filter in MC

type 1 (non-DC leak)
b' = (a +14b + c) / 16
e = (- a +9b +9c - d) / 16

a x b x e + c x d x
pel allocation (vertically or horizontally)

type 2 (DC leak)
b' = (a +12b + c) / 16
i = (- a +8b +8c - d) / 16

a , b c , d = reference pels
b' , e = predictive pels

Core Experiments
type 1 (non-DC leak) / type 2 (DC-leak)
Change Horizontal only / Horizontal and Vertical
Syntax
In the sequence layer, one bit flag may be defined to indicate as follow.
"0" : normal prediction
"1" : leaked prediction

Reference
MPEG 92 / 261
If any question , contact ;

Kenji Sugiyama Digital Technologies Research Dept. Central R&D Center JVC
58-7, Shinmei-cho , Yokosuka, Kanagawa 239, JAPAN
e-mail : k-sgym@krhm.jvc-victor.co.jp Tel : +81-468-36-9275 Fax : +81-468-36-8540

Core Experiment No.6 - Leaky Prediction 2

The syntax used for transmitting the leak_factor is described precisely in MPEG 92/340.

A 3-bit codeword is sent in the header of each P-picture. This codeword ranges from "001" to "111" ("000" is prohibited). This codeword is mapped to an integer n by natural binary mapping (uimsbf) except for "111" where n is infinite.

The leak_factor LF is obtained as 1 - 1 / 2 ** n. It is equal to 1 for infinite n (no leaky prediction).

The leaky prediction is obtained as LF * (pred_pixel - 128) + 128
where pred_pixel is the prediction pixel obtained by any prediction method.

As suggested by the CCITT Experts Group, the multiplication in the above equation is implemented in ***floating point*** followed by ***truncation towards 128 *** in 8 bits.

An algorithm for computing the leak_factor LF is given in MPEG 92/340.
This method computes the correlation between the the current picture and the (non-leaky) motion-compensated prediction from the previous I/P picture. This correlation is quantized to the nearest allowed leak_factor.

EXPERIMENTS:

1. Compare the performance of M=3, N=12 with fixed LF=0.875 or variable LF , M=3, and no I-pictures (after the first).

2. Simulate channel errors or cell losses and demonstrate resiliency.

3..Check the capability of channel tune-in for M=1 and M=3.

In the above experiments, the quantizer matrices are the default matrices in the TM.
If the matrices are downloaded in the picture layer, this should be stated

Core Experiment No.7 Reverse Order Prediction

(Option to use reverse field order in field structure coding)

In field-structure syntax, each field is preceded by a picture header. The order of field processing may be reversed using the two bits of picture-structure:

11 frame picture
10 field 1 picture
01 field 2 picture
00 reserved

Since "B" frame s are not affected, the improvement of reversing field order is more evident in low-delay mode (M=1).

This experiment compares (a) mormal field order coding and (b) reverse field order coding. CCIR-601 fields are coded using MPEG2 field structure coding at 4 Mbit/s, using adaptive single/dual-field predictions with M=1, N=15 (30Hz) or N=12 (25Hz). In (b), the I and P-picrures are reversed in field processing using the above table, resulting in improved performance in sequences with uncovered background.

(a) Io -> Po -> Po ->
\ / \ / \ NOTE ; UPWARD AND DOWNWARD ARROWS
Pe -> Pe -> Pe -> SHOULD BE WRITTEN !!!!

(b) Po -> Po -> Po ->
\ \ \ NOTE ; UPWARD&&BACKWARD AND
Ie -> Pe -> Pe -> DOWNWARD&&LOWSLOPE ARROWS
SHOULD BE WRITTEN !!!!

---------->
display time

Core Experiment No.8 Simplification of Test Model

Introduction

Test Model 1 (TM1) : Frame/Field Adaptive Coding has two problems. One is algorithm complexity, because TM1 has adaptations of Frame MC/Field MC and Frame DCT/Field DCT. The other is the amount of memories in hardware implementation. TM1 coding method needs more memories, e.g. compared with Frame Base Field Structure Coding.To solve these problems, this document proposes the simplification of Test Model 1 : Frame/Field Adaptive Coding.

Proposed Method

Proposed Method 1
Test Model 1 : Frame/Field Adaptive Coding has 4 modes. They are as follows :

1. Frame MC - Frame DCT
2. Frame MC - Field DCT
3. Field MC - Frame DCT
4. Field MC - Field DCT

In mode 2 and 3, macroblock reordering is needed between Motion Compensation and DCT, and this increases hardware complexity of Frame/Field Adaptive Coding Method. To solve this problem, the simplified method that inhibits mode 2 and 3, i.e. restricts mode 1 and 4 only is investigated. This simplified method decides DCT mode by means of MC mode in non-intra MB. For example, if MC mode is decided to Frame Mode by comparing prediction errors, DCT mode is decided Frame Mode automatically. This simplified method saves not only Macroblock reordering after Motion Compensation but also the process of DCT mode decision in the encoder. Moreover TM1 has two bits(flags) to inform MC and DCT modes, but this simplified method needs only one bit(flag) to inform MC/DCT mode. In intra MB the DCT mode is decided by the same method as TM1.

Another Reason for Proposed Method 1

The combination of Field MC and Frame DCT mode shows poor performance at core experiments of Scalability. Therefore the combination of Field MC and Frame DCT mode is not recommended by the requirement of Scalability.

Proposed Method 2

The second proposed method fixes Picture Structure of B-Pictures to Frame Base Field Structure. By this method, frame memory for the B-Pictures becomes needless because B-Pictures can be displayed as soon as it is decoded.

The problem of this method is that the matching between the time the
decoder needs to decode 1 slice and the time the decoder needs to display

1 slice becomes critical. However, This problem is also occurred in the TM1 if TM1 has only 1 field memory for B-Pictures.

Proposed Method 3

This method is a combination of Proposed Method 1 and Proposed Method 2. In P-Pictures, Proposed Method 1 is used. In B-Pictures, Proposed Method 2 is used. This method doesn't need Macroblock reordering, nor does it need frame memory for the B-Pictures.

Simulation Conditions

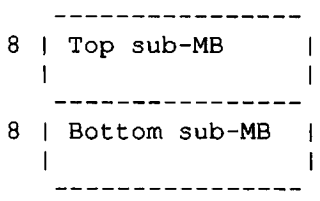
Simulation conditions are as follows, and other conditions are the same as Test Model 1.

GOP & Prediction : M=3, N=12
Sequence : Flower Garden, Mobile & Calendar, CheerLeaders, Bicycle, Football, Ftball, TableTennis
Bitrate : 4Mbps
Rate Control : Test Model 2

Core Experiment No. 9 16x8 Sub-Macroblock MC

1. Purpose: To evaluate the coding efficiency and improvement of visual quality

2. Definition of sub-macroblock
   In case of field coding, 16x16 macroblock may not be correct to apply the same MC operation, then 16x16 macroblock may be coded as two 16x8 sub-macrobloks with different MC operation.



3. Description of sub-MB motion compensation
(1) Different motion vector(s) for each 16x8 sub-MB while same MB-type
- decision-criteria-1: whether one 16x16 or two 16x8

```
If(MV_of_top_sub-MB == MV_of_bottom_sub_MB)
  do  (16x16 MC)
  else
     do  (two 16x8 MC)
```

- decision-criteria-2: whether one 16x16 or two 16x8

```
If((16x16 MC_MSE) < (16x8 MC_MSE_sum x1.125))
  do  (16x16 MC)
  else
     do  (two 16x8 MC)
```

where "16x8_MC_MSE_sum" refers to sum of 16x8_MC_MSE of "Top" and "Bottom" sub MB's.

(2) Different MC operation for each 16x8 sub-MB while MVs limited by 2 per this MB
- MVs is limited by two then in case of two 16x8 sub-MB, it can be MC operated as follows.
  - The meaning of "forward-MC-type" will be changed to

"forward-MC-type for top-sub-MB" and
"backward-MC-type for bottom-sub-MC"
- The meaning of "backward-MC-type" will be changed to
"backward-MC-type for top-sub-MB" and
"forward-MC-type for bottom-sub-MC"
- The meaning of "interpolated-MC-type" will be changed to
"both forward-MC-type" with different MV for each sub-MB.
- The decision criteria is the MSE criteria.

4. Syntax change
- In the picture layer, one bit flag will be added to indicate whether 3.(1) or 3.(2) sub-MB scheme will be applied when indicated by "sub-MB-flag in MB layer.
- In the MB layer one bit flag of "sub-MB-flag" will be added just after MB-type field to know whether normal MB or two sub-MBs.

If any question, contact;
Kenji Sugiyama / JVC
or
Atul Puri / AT&T

APPENDIX Q: QUANTISATION EXPERIMENTS

Source: Ad-hoc group on quantisation Angra dos Reis ,July 1992

Q.1 SCANNING

Q.1.1 DEPENDENT SCANNING

This experiment is to determine the effectiveness of adapting the scan pattern according to the DCT mode decision (Frame/Field) with the following scans:

Scan Patterns: Frame DCT Zigzag as in Test Model
Field DCT Scan a,b,c or d

with: scan a : As given in CCIR Rec. 723
scan b: As given for field blocks in MPEG 92/261
scan c: As described in MPEG 92/144
scan d: An optimum scan to be determined and communicated within the ad hoc group.

Test Conditions: Bit rate 4 and 9 Mbit/s
Prediction modes must be clearly stated as these may effect the efficiency of adaptive scanning

Criteria for assessment:

i) SNR averaged over sequence and given for I,P and B frames
ii) Bits/frame for the first I,P and B frames in sequence
iii) Bits/frame using MQUANT equal to 5,10 and 15 for the first 3 I, P and B frames and the average over the sequence for I, P and B frames.

Q.1.2 INDEPENDENT SCANNING

This involves adaptation of the scanning pattern independant of the mode selected for the DCT.

Scan 1: Zigzag
Scan 2: Vertical (as in MPEG 92/241)

Decision Criteria: Either "a posteriori" or "a priori" . An a posteriori selection choses the scan pattern which gives the lowest number of bits after the VLC. An a priori decision is made before variable-length coding. A suitable a priori criterion will be circulated within the ad-hoc group.

Syntax for independent scanning: Insert the following after interlaced_macroblock_type

```
if (macroblock_intra || macroblock_pattern) {
    vertical_zigzag_scanning                    1 uimsbf
}
```

Test Conditions: 4 and 9 Mbit/s
Prediction modes must be clearly stated

Criteria for assessment:

1. SNR averaged over sequence (figures given separately for I, P and B frames)
2. Bits/frame for the first I, P and B frames of the sequence.

3. Bits/frame using MQUANT equal to 5 and 10 and 15 for the first I and P and B frame and also the average over the sequence separately for I, P and B frames.

Additional statistics: Percentage usage of each scan in separately for I, P and B frames.

Q.2 EXPERIMENTS ON MATRIX ADAPTATION

Q.2.1 MATRIX ADAPTATION WITH MACROBLOCK TYPE

It is proposed to download 4 matrices at the picture layer. The syntax for this as follows:

Before "picture_extension-data" insert the following:

```
load_quantiser_matrices                 1    uimsbf
if (load_quantiser_matrices) {
    quantiser_matrix_one                8*64 uimsbf
    quantiser_matrix_two                8*64 uimsbf
    quantiser_matrix_three              8*64 uimsbf
    quantiser_matrix_four               8*64 uimsbf
    matrix_assignment_table              16  uimsbf
}
```

where the matrix assignment table, which assigns one of the four matrices to the different macroblock types, is constructed as follows:

Macroblock type			Matrix Selection (bits)
Intra	Frame	Y	2
Intra	Frame	C	2
Intra	Field	Y	2
Intra	Field	C	2
Non_intra	Frame	Y	2
Non_intra	Frame	C	2
Non_intra	Field	Y	2
Non_intra	Field	C	2

SPECIFICATION OF MATRICES: To be optimised and distributed within ad-hoc group for both 4:2:0 and 4:2:2 cases

Initial comparison: For macroblock types Intra/*/* and Non_intra/*/* use matrices as defined in TM together with those defined in MPEG 92/261 for the 4:2:0 case.

SPECIFICATION OF ASSIGNMENT TABLE: To be optimised and distributed within ad-hoc group

ASSESSMENT CRITERIA: Picture quality !

Q.2.2 EXPERIMENT OF ADAPTIVE CONTROL OF MATRIX SELECTION AT MB LEVEL

Four matrices are transmitted at the picture layer as described in previous experiment. Transmission of the assignment table is not necessary.

At the MB layer, after "coded_block_pattern()", insert:

```
mb_quantiser_matrix_select          2    uimsbf
```

This variable identifies which of the four matrices is to be used for coding all the blocks in this MB.

CRITERIA FOR MATRIX DEFINITION AND SELECTION: To be determined and distributed in quantisation ad-hoc group.

An initial experiment would be to adapt the weighting matrix according to the criticality of the macroblock; i.e. a macroblock can have one of four possible values of criticality which selects one of the 4 matrices. The meaning of criticality is specified in the following table:

Criticality	Weighting Matrix Adjustment
0	Lower precision of all but lower frequency terms
1	No change
2	Increase precision of frequency terms representing interlace
3	Increase precision of low frequency terms

ASSESSMENT CRITERIA: i) Picture Quality

Q.2.3 Quantizer Weighting Matrices

At the sequence layer four matrices wdelta0[64] , wdelta1[64], wdelta2[64] and wdelta3[64] containing 8-bit signed integers are transmitted, these are relative weighting matrices which when added to Intra or NonIntra matrices yield actual weighting matrices used for quantization.

Chrominance Weighting

At the sequence layer, appropriate wdelta matrix to be used for determining the chrominance matrix (by summing it to corresponding Intra or NonIntra matrix) is identified. Thus this matrix is global for all chrominance weighting in the sequence.

Local Weighting Adaptation

The weighting matrix used for a macroblock depends on the "criticality" of the macroblock. A macroblock can have one of the four possible values of criticality and each criticality value corresponds to a specific wdelta matrix. The following table clarifies the meaning of macroblock criticality.

Criticality	Weighting Matrix Adjustment
0	Lower precision of all but low frequency terms
1	no change in any term
2	increase precision of frequency terms representing interlace
3	increase precision of low frequency terms

A fixed 2-bit code per macroblock (this can be made VLC later) identifies the criticality.

The exact wdelta [] matrices will be specified with in the ad hoc group.

Syntax

```
sequence_header() {

    load_delta_matrices                 1
    if (load_delta_matrices) {
        wdelta0[64]                     8*64
        wdelta1[64]                     8*64
        wdelta2[64]                     8*64
        wdelta3[64]                     8*64
```

```
        }
        chroma_matrix_enable        1
        wdelta_select                       2



}

macroblock() {



        wdelta_select                       2



}
```

Q.3 EXPERIMENTS ON QUANTISER RANGE, PRECISION, AND CONTROL

Q.3.1 EXPERIMENT EXTENDING RANGE OF TRANSMITTED COEFFICIENT

This experiment is to test the effectiveness and necessity of increasing the maximum range of the transmitted coefficients from +/- 255 to +/- 2047

One bit is added at the Picture Layer, to indicate that a single 12-bit level *tcoef* escape is used, rather that the mechanism of *escape/double-escape*, as in MPEG-1.

tcoef_escape_format	
0	Compatible MPEG-1
1	12-bit Escape

When *12-bit Escape* is selected, the syntax of the escape changes. The *escape code* "000001" is followed by the *run* (6-bit VLC), followed by the *level* (12-bit VLC).
The table for the escaped level becomes :

FLC	level
100000000001	-2047
100000000010	-2046
...	...
111111111111	-1
forbidden	
000000000001	+1
...	...
011111111111	+2047

The mechanism of double-escape (28 bits) is not used is this case, making the maximum length of an encoded run-level equal to 24 bits.
As a side effect, this proposal may simplify the implementation of encoders and decoders, as it is no more necessary to check whether MQUANT need to be changed to avoid coefficient clamping at 255, and the maximum number of bits per transmitted coefficient is reduced from 28 to 24 bits

ASSESSMENT CRITERION: Picture quality on pictures which generate large coefficients with low MQuant values (e.g. still pictures)

Q.3.2 EXPERIMENT TO INCREASE PRECISION OF INTRA DC COEFFICIENTS

One bit is added at the Picture Layer, to indicate that Intra DC coefficients are transmitted with a 9-bit precision rather than 8-bit.

intra_dc_format	
0	Compatible MPEG-1 (8-bit Precision)
1	9-bit Precision

When *9-bit Precision* is selected, the Intra DC coefficients are quantized (resp. dequantized) with no-dead-zone and a step-size equal to 4, rather than 8, and transmitted with 9-bit precision rather than 8-bits.
When *9-bit Precision* is selected, the value 256 is used, rather than 128, to initialize or reset the Intra DC predictors.
The tables B5a and B5b used to decoding *bit Precision* Intra DC coefficients are extended in a natural way :

Luminance	Size
100	0
00	1
01	2
101	3
110	4
1110	5
11110	6
111110	7
1111110	8
11111110	9

Chrominance	Size
00	0
01	1
10	2
110	3
1110	4
11110	5
111110	6
1111110	7
11111110	8
111111110	9

ASSESSMENT CRITERION: Picture quality on appropriate pictures

Q.3.3 EXPERIMENT TO EXTEND RANGE OF MQUANT

A) INREASING MQUANT TO 6-BIT

The MQUANT range of 1 to 31 is insufficient for very high resolution video and also the precision of MQUANT is not sufficient at low values for high quality coding.

In addition to default table of values for MQUANT from 1 to 31, an mquant_extend table is proposed. This table covers a range of 1 to 63 and is nonlinear in nature, providing higher precision for smaller values of MQUANT and coarser precision for finer values. An example of the MQUANT values allowed for this mquant_extend table are the following:

```
1     1.5   2     2.5   3   .....   14    14.5  15    15.5

16    17    18    ..................    29    30    31

33    35    37    ..................    59    61    63
```

The mquant_extend table is downloaded at the sequence layer with the following syntax in the Sequence Header:

```
sequence_header(){
        .
        .
        .
    mquant_extend              1
    if(mquant_extend)
        mquant_extend[64]      8*64     uimsbf
            .
```

.
.

}

If necessary, this table could be loaded at the picture layer

It is recognised that many ways of enabling a larger range of MQUANT are possible and optimisations in spacing of MQUANT values is dependent on bit rate and image complexity. It is therefore important that any experiments are performed on material containing high resolution details as well as at higher bit rates (above 10 Mbit/s)

B) 5-BIT MQUANT WITH ADAPTIVE ASSIGNMENT OF VALUE

In this experiment the syntax and objectives are similar to the last experiment except that the number of levels assigned to MQUANT remains at 32 and an mquant_assign table (8*32 bits) is downloaded at the picture layer.

Q.3.4 EXPERIMENT ON BLOCK ADAPTIVE QUANTISATION

To improve picture quality locally in a scene, refinement of MQUANT may be necessary on a block basis. A 1-bit code decides whether such a refinement is necessary; if so, 2-bits are sent for every coded luminance block within that macrblock. The mechanism for refinement is via use of BQSCALE, which can have one of four possible values such as {0.5, 0.75, 1.0, 1.5}. The MQUANT value for the macroblock is multiplied by the chosen BQSCALE and used as the block quantiser value. For macroblocks not requiring refinement, no BQSCALE bits are necessary.

The choice of BQSCALE for a block depends on the following:

- The variance of a block
- The variances of its three causal neighbours. These are used to detect the presence of an edge. If such an edge is detected, BQSCALE of 0.5 or 0.75 is selected.

It is noted that the decision criteria, concerning whether or not a macroblock needs to be refined with respect to MQUANT, depends on the bit rate available and picture quality desirable.

The syntax is:

Macroblock Layer

```
macroblock(){
        .
        .
    if(macroblock_quant){
        quantiser_scale                    5 (/6)
}
quantizer_scale_refine                     1
        .
        .
}
```

Block Layer

```
block(i){
    if(pattern_code[i]){
        if(i<4){
            if(quantiser_scale_refine){
              bqscale                                    2
            }
        }
    }
}
```

Q.4 Alternate DCT (DCT/NTC)

Q.4.1 INTRODUCTION

In this proposal of core experiment, algorithm enhancement to the current TM to improve coding efficiency of edge information which based on the concept of adaptive DCT/Non-DCT is to be examined. The basic coding process additions to the current TM are Edge Block Detection Process, Prediction Process, and Quantization Process. They are described in detail in the following sections.

Q.4.2 CORE EXPERIMENT REFERENCE MODEL

Reference Model : TM2 Frame structure, with Frame/Field Adaptive Modes only
Data Rate : 4 Mbits/s
Test Sequences : Mobile & Calendar, or similar
Algorithms : NTC1 (Section 3), and NTC2 (Section4)

Bit Stream Syntax :

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 NTC (method 1 or 2) coded blocks in the Macroblock. (eg. first received bit is "1", the first block in the Macroblock is non-DCT coded, and so on.)

Macroblock Layer :

```
    ....
if ((core_experiment==NTC1) || (core_experiment==NTC2)) {
    adaptive_coded_type                                  1
    if ( adaptive_coded_type == 1 )
        adaptive_coded_block_pattern                     6
        }

if ( macroblock_quant )
    quantizer_scale                                      5       umsbf

    ...
```

Block Layer :

data_scan_path_flag--- This is a 2bits FLC to indicate which data scan path is selected . Table are shown in the document MPEG92/275.

differential_flag--- This is a 1bit to indicate differential on or off. If it is 1, the differential is on, if it is 0, the differential is off.

block_quantizer_scale --- An unsigned integer in the range 1 to 31(5bits) used to scale the reconstruction level of the retreived NTC quantized levels..

In NTC1 coding, the base value is treated the same as DCT dc coefficients. So the DCT dc coefficients and NTC base are coded by the Z-differential technique same as TM0.

block (i) {		
if (pattern_code[i]) {		
if ((core_experiment==NTC1)&&(adaptive_coded_block_pattern[i]==1)) {		
data_scan_path_type	2	
differential_flag	1	
block_quantizer_scale	5	
}		
if (macroblock_intra) {		
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 {		
dct_coeff_first	2-28	vlclbf
}		
if (picture_coding_type != 4) {		
while (nextbits() != '10')		
coded_coeff_next	3-28	vlclbf
end_of_block	2	
}		
}		
}		

Q.4.3 DESCRIPTION OF NTC1 METHOD

1) METHOD OVERVIEW

NTC1 is a predictive coding method in the block layer. Examples of NTC1 are shown in Figure 1 and 2 in the document MPEG92/275. The following 4 techniques are applied in NTC.

1.1) Prediction of block representative value

In NTC1 the block representative value is called "base". NTC1 module predicts the base value and encodes the differential values between the base and each pixel in the block.

1.2) Adaptive Quantization

The Quantization is applied to the differential values between the base and each pixel in the block. The values are quantized with a flat weighting and with a deadzone equal to the stepsize. So the quantizer is the same as for non intra macroblocks in the TM0.

1.3) Adaptive Data Scan

For each block the NTC1 module selects one scan path which gives the fewest number of run/level events. Then for each block the NTC1 module selects either "differential on" or "differential off" depending on the number of VLC events. If the differential is on, 1 dimensional differential coding is applied for the scanned data..

2) Algorithm for NTC1

The algorithm is described in the document MPEG92/275. Please refer to the document.

Q.4.4 DESCRIPTION OF NTC2 METHOD

1) METHOD OVERVIEW

A simplified block diagram of the TM0 is described in figure 1 with extensions to Adaptive DCT/DPCM method as shaded boxes. The Macroblock Type Decision Module in figure 1 performs the decision of Motion Prediction Mode (Intra, Forward, Backward, Frame, Field, etc), the Motion Compensate Process, and the Frame/Field DCT Decision.

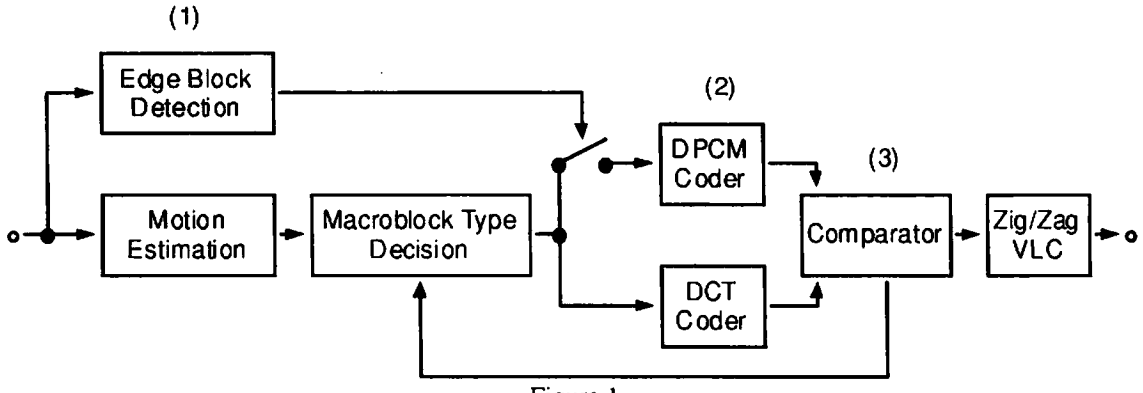


Figure 1

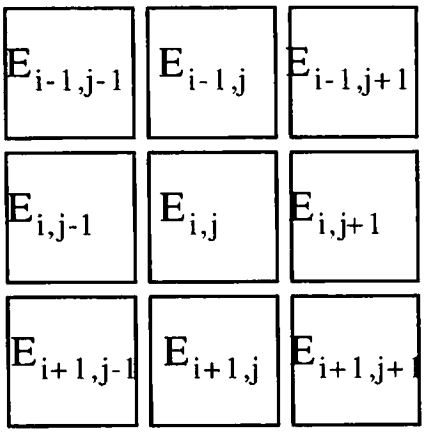
The Edge Block Detection (1) process is performed on original input blocks. Decision on Edge/Non-Edge Block is done based on Block signal energy and the neighboring Block signal energy. If the Block is detected as Edge Block, it will also be coded with the DPCM Coder (2). The reconstructed blocks (after coding and decoding) from the DCT Coder and the DPCM Coder are then compared based on Mean-Square Error (MSE) with reference to the original input Block at the Comparator (3) module, and the coder with best result (lower MSE) will be selected, and output coded Block to the Zig/Zag Scanner and VLC. This module may be omitted for the core experiment to simplify implementation, and selection is then based on Edge Detection result only.

2) 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 (they may be optimized for special sequences - encoder issue).



Activity :
$E_{i,j} = (Block_{x,y} - Block_{mean})^2$

Luma Edge Block Condition :
if $(E_{i,j} > T1)$ & (any $E_{surrounding} < T2$)

Chroma Edge Block Condition :
if (any Luma Edge in Macroblock) & $(E_{i,j} > T3)$

Figure 2

Due to the fact that all surrounding blocks will be examined, an input block will not be considered for edge condition if it is at the picture boundary (first row, last row, first column,, last column). In case of a Field DCT Macroblock in an adaptive Frame/Field system, luma block 1 & 3 (similarly block 2 & 4) of the macroblock must satisfy the edge block condition together in order to be considered as edge blocks since the edge conditions are checked on frame based block only; hence, corresponding to the Frame/Field DCT, Frame/Field DPCM is applied to the edge blocks.

3) DPCM CODER

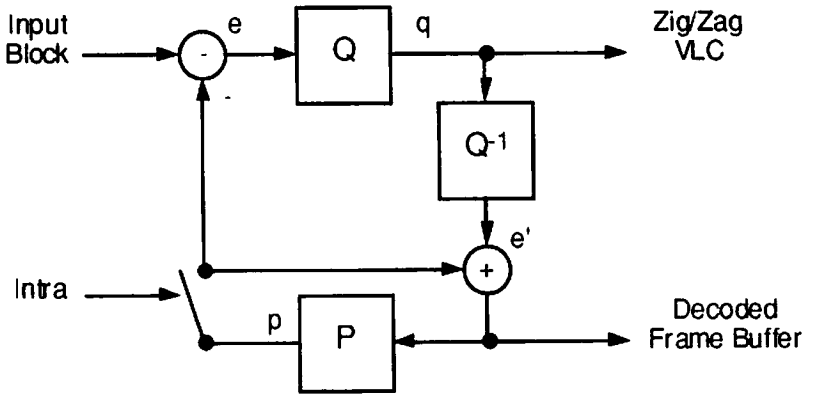


Figure 3

The DPCM Coder is illustrated in figure 3. A pel is scanned left-to-right and top-to-bottom from the input edge block to be coded. For an Intra Coded Block, the input block is coded as following :

1) The first pel (top-left corner of the block) is not coded in the DPCM coder. It is sent to the VLC for transmission as "DC" of the block.
2) The quantization parameter *b* is calculated from the reference quantization parameter *Q* (with no adaptive quantization applied for macroblock containing at least one edge block) of the macroblock (j) as :
$b = Q_j \times 0.1875 + 0.625$
3) For each of all other pels, a predicted value *p* is made based on the average of passed coded top and left pels (if available), and the prediction error *e* is quantized as :

$q = (int)(\frac{\sqrt{|e|}}{b}) \times Sign\ (e)$

The Inverse quantization is :

$e' = (int)((q^2+|q|) \times b^2) \times Sign\ (q)$

4) All quantized coefficients are output to the TM0 Zig/Zag and VLC for transmission as "AC" of the block.

For a non-intra Coded Block, the input block is coded as following :

1) The quantization parameter b is calculated from the reference quantization parameter Q (with no adaptive quantization applied for macroblock containing at least one edge block) of the macroblock (j) as :
$b = Q_j \times 0.1875 + 1.125$

2) Each pel e in the block is quantized as :

$q = (int)(\frac{\sqrt{4|e|+b^2}}{2b} - 0.5) \times Sign\ (e)$

The Inverse quantization is :

$e' = (int)((q^2+|q|+\frac{|q|+k}{2}) \times b^2) \times Sign\ (q)$

where k = **Error!**)

3) All quantized coefficients are output to the TM0 Zig/Zag and VLC for transmission as "AC" of the block.

An example "C" implementation of the DPCM Coder is as follow :

```
DpcmBlock (block, qBlock, Q, mbType)

    int     block[8][8];
    int     qBlock[8][8];
    int     Q;
    int     mbType;{

    int     x, y;
    int     pel, p, e, dec;
    int     q, k;
    double b;

    if (mbType==Intra) b = (double)Q * 0.1875 + 0.625;
    else b = (double)Q * 0.1875 + 1.125;
    if (b<1.7) b = 1.7;

    for (y=0; y<8; y++) {
        for (x=0; x<8; x++) {

            pel = block[y][x];

            if (mbType==Intra) {
                if ((x==0)&&(y==0)) {    // Transmit not Quantized
                    q = block[0][0];
                    dec = pel;}
                else {
                    if (x==0) p = block[y-1][0];
                    else if (y==0) p = block[0][x-1];
                    else p = (block[y][x-1] + block[y-1][x]) / 2;

                    e = pel - p;
                    q = (int)(sqrt((double)(abs(e)))/b) * Sign(e);

                    dec = p + (int)((q*q+abs(q))*b*b) * Sign(q);
                    }
                }
            else {
                e = pel;
                q = (int)(sqrt(4.0*abs(e)+b*b)/(2.0*b) - 0.5) * Sign(e);

                if (q==0) k = 0; else k = 1;
                dec = (int)((q*q+abs(q)+(abs(q)+k)/2)*b*b) * Sign(q);
                }
            qBlock[y][x] = q;
```

```
            block[y][x] = dec;     //     Local Decoded Value
            }
        }
    for (y=0; y<8; y++)
        for (x=0; x<8; x++) block[y][x] = ClampInteger (&block[y][x], 0, 255);
    }
```

For real-time implementation consideration, the Quantization and inverse Quantization process can be implemented by table-look-up method.

4) COMPARATOR

The local decoded block from the DCT Coder and the DPCM Coder can be compared to the original (uncoded) block in this module based on the Mean-square-Error measure MSE (other measuring criteria is possible). Final selection of output quantized coefficients of the block from one of the two coders is made based on lowest MSE. This module only maximizes signal/noise ratio of the decoded pictures. Again, this is done only for SNR optimization (not necessary picture quality improvement).

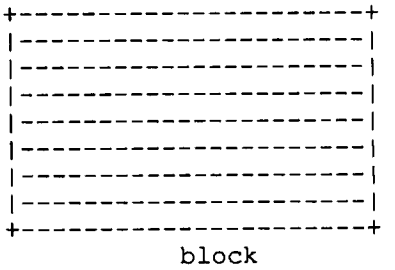
5) REFERENCE

All figures in section 4 (figure 1,2,..) can be refered to in document MPEG 92/368. For further questions, please feel free to contact :

Lucas Hui
AV/Information Research Center
Asia Matsushita Electric (S) Ptd Ltd
Email : lucas@avirc.ams.mei.co.jp
Tel : +65 381-5460
Fax : +65 285-7237

Q.5 NON-8 x 8 DCT

Purpose : To belify the coding efficiency and improvement of visual quality
Difinition ;
8 x 1 DCT
1. Horizontally one dimensional 8 DCT that described in MPEG1 CD is used.
2. Block size is same as normal 8 x 8 DCT , then 8 sets of horizontal DCT coeff. exist in each block independently.



3. 8 x 1 DCT is applied to Non-Intra MB only
4. Each coefficient is multiplied by 2 instead of 4 that is in case of 8 x 8 DCT.
5. VLC(AC coeff.) is same as 8 x 8 DCT (TM)
6. Q-Matrix and Scanning for 8 x1 DCT block are basically one dimensional as below;

0	8	16	24	32	40	48	56
1	9	17	25	33	41	49	57
2	10	18	26	34	42	50	58
3	11	19	27	35	43	51	59
4	12	20	28	36	44	52	60
5	13	21	29	37	45	53	61
6	14	22	30	38	46	54	62
7	15	23	31	39	47	55	63

Scan order for 8 x 1 DCT

22	23	24	26	27	28	30	31
22	23	24	26	27	28	30	31
22	23	24	26	27	28	30	31
22	23	24	26	27	28	30	31
22	23	24	26	27	28	30	31
22	23	24	26	27	28	30	31
22	23	24	26	27	28	30	31
22	23	24	26	27	28	30	31

Q-Mat. for 8 x 1 DCT

8 x 4 DCT

1. Use 8 x 4 DCT intead of 8 x 8 DCT.
2. Block size is same as normal 8 x 8 DCT , then 2 sets of DCT coeff. exist in each block independently.

block

3. 8 x 4 DCT is applied to only non-intra MB
4. Each coefficient is multiplied by suare root 8 instead of 4 that is in case of 8 x 8 DCT.
5. VLC(AC coeff.) is same as 8 x 8 DCT (TM)
6. Q-Matrix and Scanning for 8 x4 DCT block are as below;

0	2	10	12	26	28	42	44
4	8	14	24	30	40	46	56
6	16	22	32	38	48	54	58
18	20	34	36	50	52	60	62
1	3	11	13	27	29	43	45
5	9	15	25	31	41	47	57
7	17	23	32	39	49	55	59
19	21	36	37	51	53	61	63

Scan order for 8 x 4 DCT

18	19	20	21	22	23	24	25
19	20	21	22	23	24	25	26
20	21	22	23	24	25	26	27
21	22	23	24	25	26	27	28
18	19	20	21	22	23	24	25
19	20	21	22	23	24	25	26
20	21	22	23	24	25	26	27
21	22	23	24	25	26	27	28

Q-Mat. for 8 x 4 DCT

Adaptation of 8 x 8 / 8 x 1 or 8 x 8 / 8 x 4 DCT

Switching unit : Macro Block

Decision 1 (post)

1. Calculate amount of data in each MB
2. Select smaller one

Decision 2 (pri)

pri-decision method will be added by JVC

Core experiments

1. 8 x 1 DCT / 8 x 4 DCT
2. All non-intra are 8 x 1 (8 x 4) / adaptive
3. Field-structure / Frame-structure
4. Decision 1./ Decision 2

Syntax

if (macroblock_pattern)		
coded_block_pattern	**3 -- 9**	**vlclbf**

non_dct will be signalled in the sequence_layer

in macroblock layer will be changed

if (macroblock_pattern) {		
coded_block_pattern	**3 -- 9**	**vlclbf**
if (non_dct)		
non_8x8dct_flag	**1**	**uimsbf**

Reference
MPEG 92 / 093, MPEG 92 / 261

If any question, contact ;

Kenji Sugiyama Digital Technologies Research Dept. Central R&D Center JVC
58-7, Shinmei-cho , Yokosuka, Kanagawa 239, JAPAN
e-mail : k-sgym@krhm.jvc-victor.co.jp Tel : +81-468-36-9275 Fax : +81-468-36-8540

CONTENTS