Telecommunication Standardization Sector
Study Group 15
Experts Group for ATM Video Coding
(Rapporteur's Group on Part of Q.2/15)

Document AVC-491b
Version 2
April, 1993

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/ **N0400**

Title: **Test Model 5**

Status: **Draft**

Source: **Test Model Editing Committee**

CONTENTS

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.

Those parts which have been changed from TM4, revision 2 are marked up by a bar in the margin. In this document reference is made to the MPEG-2 Video Working Draft of the Sydney meeting (WD).

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 (reference is made to the WD) and 10 respectively.
- Scalable bitstreams 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.

2.2 Coder block diagram

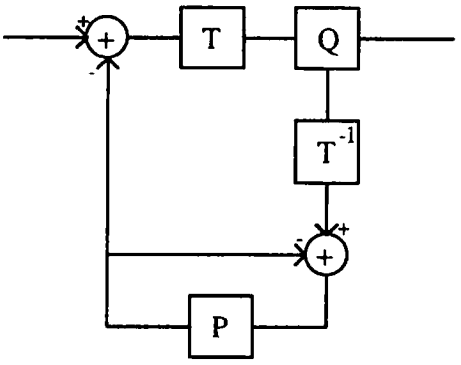


Figure 2.1: Block diagram of basic (MAIN profile) encoder

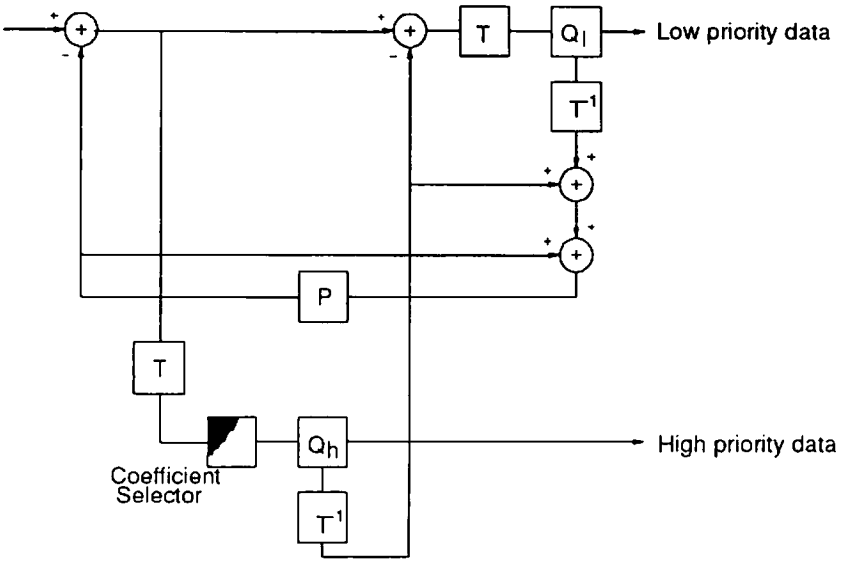


Figure 2.2: Possible logic block diagram of an MPEG2 encoder encapsulating data partinioning/SNR scalability/ frequency scalabitlity

Figure 2.2 encapsulates 3 coding schemes currently on the table:

1. Sarnoff-Thomson's error resilient scheme, Incase Qh and Ql are identical; coefficient coded in the high priority data stream will generate zero's coefficients in the low priority data stream.
2. Philips' error resilient/SNR scalable scheme, In case the coefficient selector selects all coefficients, a coarse quantisation is performed by Qh and a finer re-quantisation is performed by Ql.
3. Australian scalable/error resilient scheme. In case the coefficient selector always selects the 4x4 low frequency coefficien[illegible]d Qh and Ql could be identical: coefficient coded in the high priority data stream will generate [illegible]'s coefficients in the low priority data stream. If Ql performs a coarser quantisation then Qh, the 4x4 low frequency coefficients are refined.

Using the harmonised scalable decoder, the selection of one of the three schemes will be an encoder option. The current macroblock layer syntax for spatial scalability will support all three options, also frequency scalability can provide all these options.

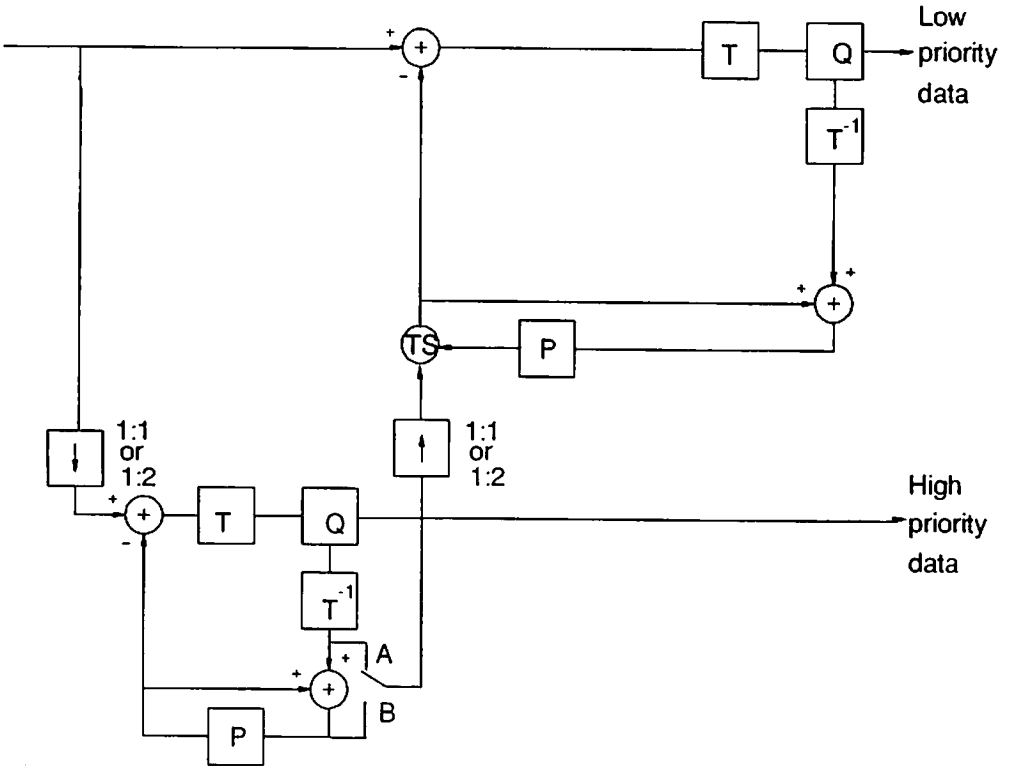


Figure 2.3: Possible implementation of a hybrid harmonised scalable/compatible encoder, able to generate an MPEG1/H.261 high priority data stream

Data paths A and B are not used concurrently. **Which data path is used is signaled in the sequence header.** Data path A is selected for the harmonised decoder (figure 2.5), data path B is selected for full scalable, compatible, higher image quality decoders (figure 2.6).

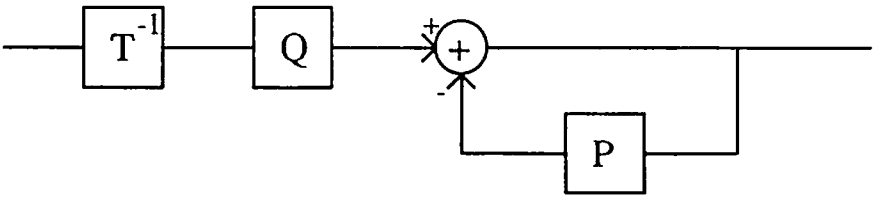
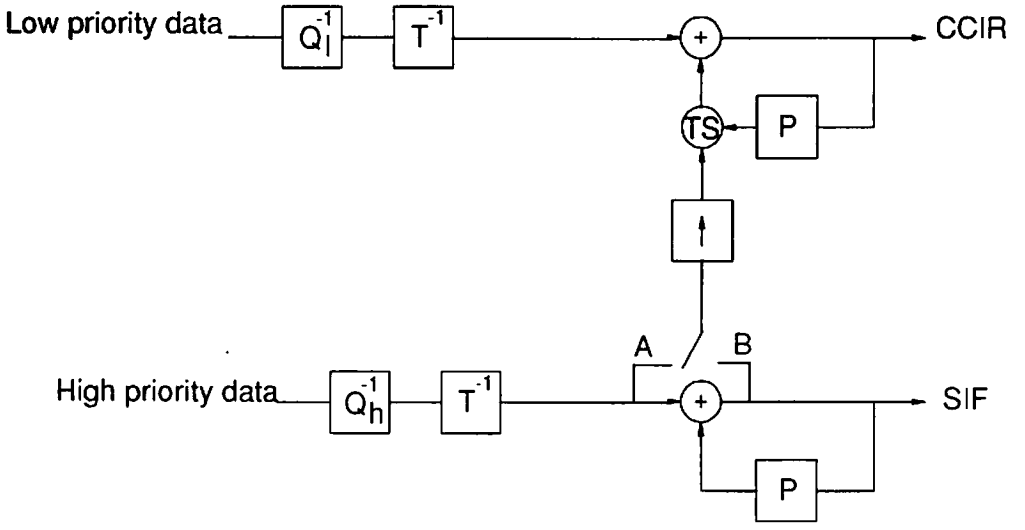


Figure 2.4: Block diagram of basic decoder



TS - Temporal-Spatial weighting switch

Figure 2.5: Full hybrid harmonised scalable/compatible core level decoder structure, allowing
DATA PARTITIONING
SNR SCALABILITY
FREQUENCY SCALABILITY
MPEG1/MPEG2/H.261 COMPATIBILITY

Data paths A and B are not used concurrently. **It is proposed to signal in the sequence header which data path is used.** Data path A is selected for low complexity, error resilient decoders, data path B is selected for full scalable, compatible, higher image quality decoders.

NOTE: The up converters can be switched off (1:1 up conversion). Data path A's up converter for a 1:2 up conversion is a simple linear (1 2 1) filter inside a block for progressive to interlace (the temporal miss aligned field should not be predicted). Data path B's up converter for a 1:2 up conversion is a simple 1 2 1 filter inside a block for progressive to interlace.

2.3 Profiles

In order to allow interworking between coders from different manufactures profiles and levels are defined. It has been defined that if coders are on the same profile and level, they are able to interwork.

More than one profile is defined, and the intention is to define them in such a way that they are allways a superset of subset of each other. This is done to facilitate interworking, this interworking is then possible on the profile and level or profiles and levels they have in common.

2.3.1 MAIN profile

This profile is intended for applications, which do not require extreme error resilience, graceful degradation or scalability, but do require relatively low cost and image quality.

MAIN level

- 720x576x50 or 720x480x59.94 interlaced
- interlaced or progresive procesing format 4:2:0
- upper bitrate 15Mbits/s
- maximum VBV buffer 1 835 008 bits
- non hierachical
- Dual-prime only between two succesive P-frames
- skipped frames are only allowed for M=1.
- can take contraint parameter set MPEG1 bitstreams
- can take MPEG1 bistreams of CCIR 601 resolution (4:2:0) (without macroblock stuffing?)
- No leak is used
- No 8x1 DCT is used
- No sub 16x16 prediction modes are used, however 16x8 is allowed in field pictures
- Two downloadable matrices are used (this is 4:2:0, in 4:2:2 four matrices are used)
- Alternative scan is used and signaled at the picture layer
- A unified Non-linear/MPEG1 quantiser is used
- High priority data of a data partitioning scheme is not used
- 8 bit input is used
- Pan/scan has 1/8 pel accuracy
- Skipped macroblocks are defined compatible to MPEG1

No other levels are defined yet.

2.3.2 Hierarchical profile

For the moment only proposals for two levels are being made, the HDTV level and the TV (CCIR 601) level. Reference is made to the current main profile, on the respective levels, although these have not yet been fully defined by now.

HDTV level

This hierarchical profile at HDTV level is the same as main profile at HDTV level plus the items listed below:

- The harmonised scalable solution. The harmonised scalable solution has not been discussed yet, but would harmonise frequency and spatial scalability into one scheme.
- Error resilience
- 16:9 aspect ratio (1920x1152x25 or 1920x1035x29.97 1440x1152x25 or 1440x1035x29.97)
- Lower level is a CCIR 601 (4:3 aspect ratio, 720x576x25 or 720x480x29.97) being coded with MPEG2 with a 8x8 DCT syntax, the main profile at main level.
- Horizontal and vertical 1:2 up conversion (the CCIR image will expand to 1440x1052 or 1440x960 signal)
- For the complete decoder system: a single loop could be allowed provided enough quality is found.

TV level

This hierarchical profile at TV (MAIN) level is the same as main profile at MAIN level plus the items listed below:

- The harmonised scalable solution. The harmonised solution has not been discussed yet, but would harmonises frequency and spatial scalability into one scheme.
- Error resilience
- 4:2:0 chrominance down converted CCIR 601 (already main level, main profile)
- Lower level is MPEG2 with a 8x8 DCT, or MPEG1 or H.261 syntax.
- Horizontal and vertical 1:2 up conversion (the (Interlaced) SIF or CIF image will expand to a 720x576 or 720x480 signal)
- For the complete decoder system; a single loop could be allowed provided enough quality is found.

2.3.3 Professional profile

This profile has also been refered to as the 'CCIR' or NEXT profile.

HDTV level

This professional profile at HDTV level is the same as hierarchical profile at HDTV level plus the items listed below:

- 4:2:2
- totaly three layers/hierarchical levels
- First lower level is a CCIR 601 (4:3 or 16:9 aspect ratio, 720x576x25 (4:2:2 or 4:2:0) or 720x486x29.97 (4:2:2 or 4:2:0)) or EDTV (16:9 aspect ratio, 960x576x25 or 720x486x29.97)
- Second lower level is a CCIR 601 or SIF level all 4:2:2 or 4:2:0

It will NOT support:

1. 4:4:4
2. 10 bit input

TV level

- This level is a CCIR 601 (4:3 or 16:9 aspect ratio, 720x576x25 (4:2:2 or 4:2:0) or 720x486x29.97 (4:2:2 or 4:2:0)) or EDTV (16:9 aspect ratio, 960x576x25 or 720x486x29.97)

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 with CCIR 601 resolution. 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. Also HDTV inputs are allowed. Two are allowed for 1035 lines per frame, 1920 or 1440 pixels per line for 60Hz, the other two allowed for 1250 lines per frame, 1920 or 1440 pixels per line for 50Hz.

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

- HDTV 4:2:2-50 The actual HDTV signal at 50Hz
- HDTV 4:2:2-60 The actual HDTV signal at 60Hz
- HDTV 4:2:0 HDTV with half chrominance resolution at 50Hz
- HDTV 4:2:0 HDTVwith half chrominance resolution at 60Hz
- 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 1/4 of the 4:2:0-60 resolution
- SIF-625 Progressive format, with 1/4 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:

	4:2:0-1250	4:2:2-1250	4:2:0-1035	4:2:2-1035
Number of active lines				
Luminance (Y)	1152	1152	960	960
Chrominance (Cb,Cr)	576	1152	480	960
Number of active pixels per line				
Luminance (Y)	1920/1440	1920/1440	1920/1440	1920/1440
Chrominance (Cb,Cr)	960/720	960/720	960/720	960/720
Frame rate (Hz)	25	25	30	30
Frame aspect ratio (over)	16:9	16:9	16:9	16:9

Table 3.0: Active 4:2:0 and 4:2:2 HDTV Formats

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
Chrominance (Cb,Cr)	288	576	240	480
Number of active pixels per line				
Luminance (Y)	720	720	720	720
Chrominance (Cb,Cr)	360	360	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 CCIR 601 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

[SEE WD: Chapter 3]

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 picture 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.

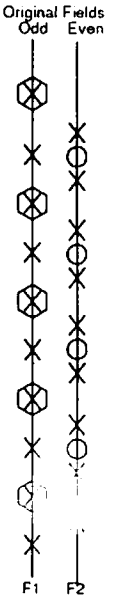


Figure 3.1: 4:2:0 Chrominance sub sampling in the fields

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, see figure 3.3.

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

Table 3.4a Decimation filter

0	29	55	88	55	29	0	//256

Table 3.4b Decimation filter

Note: the FIELD1's 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.

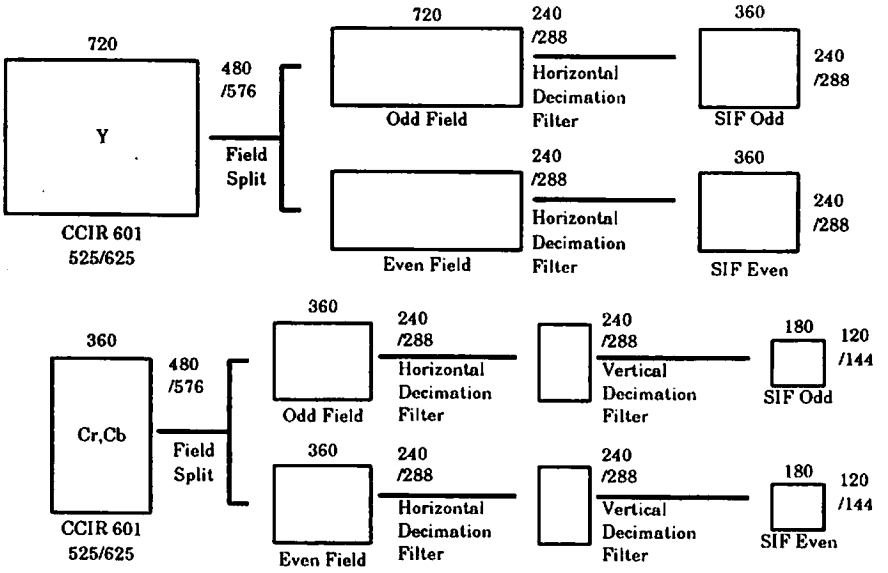


Fig 3.3: Conversion from CCIR 601 into SIF Odd and SIF Even

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.

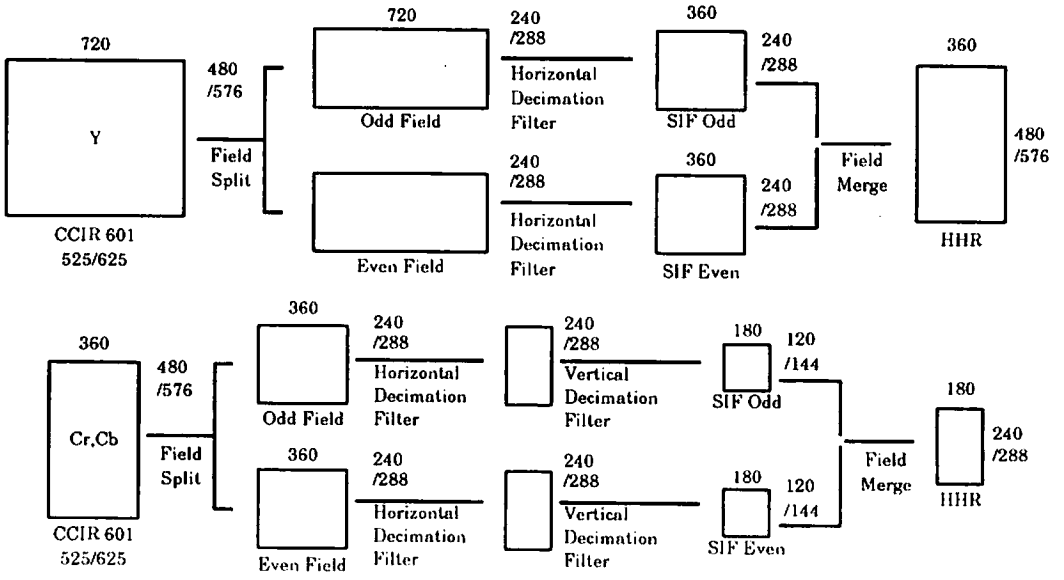


Fig. 3.4 Conversion from CCIR 601 into HHR

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.
See section 3.5 for and alternative methode.

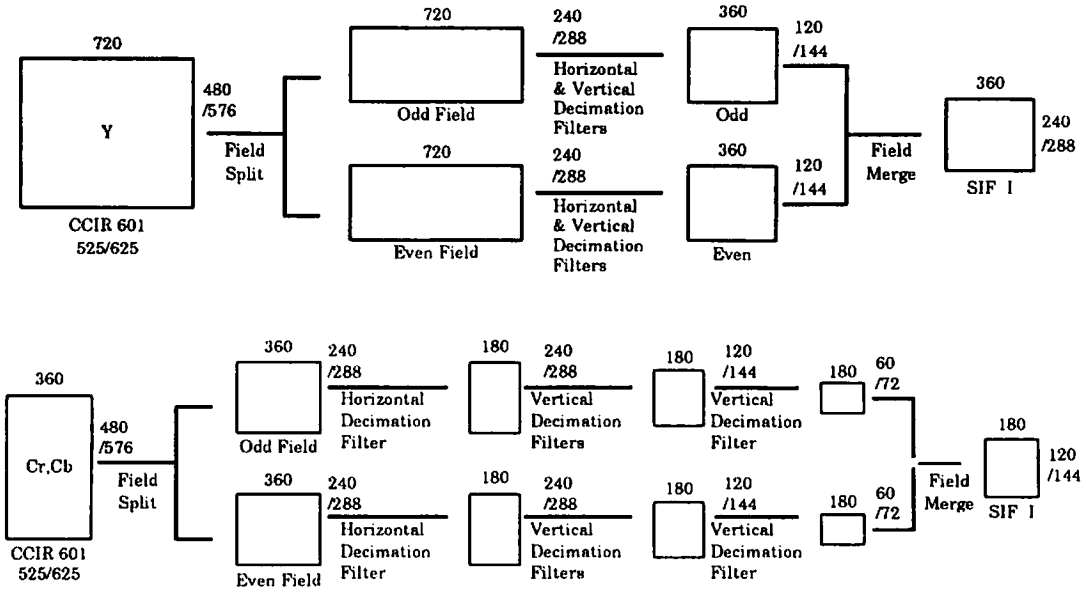


Fig. 3.5: Conversion from CCIR 601 into SIF Interlaced (SIF-I)

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 pictures 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.

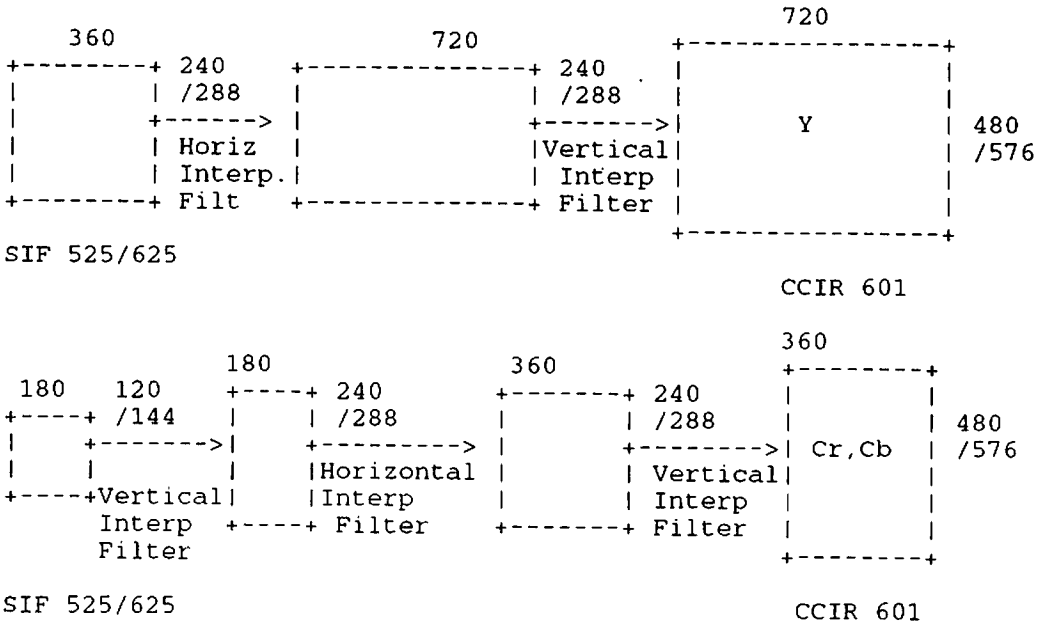


Figure 3.6: Conversion of SIFs to CCIR 601 formats

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.

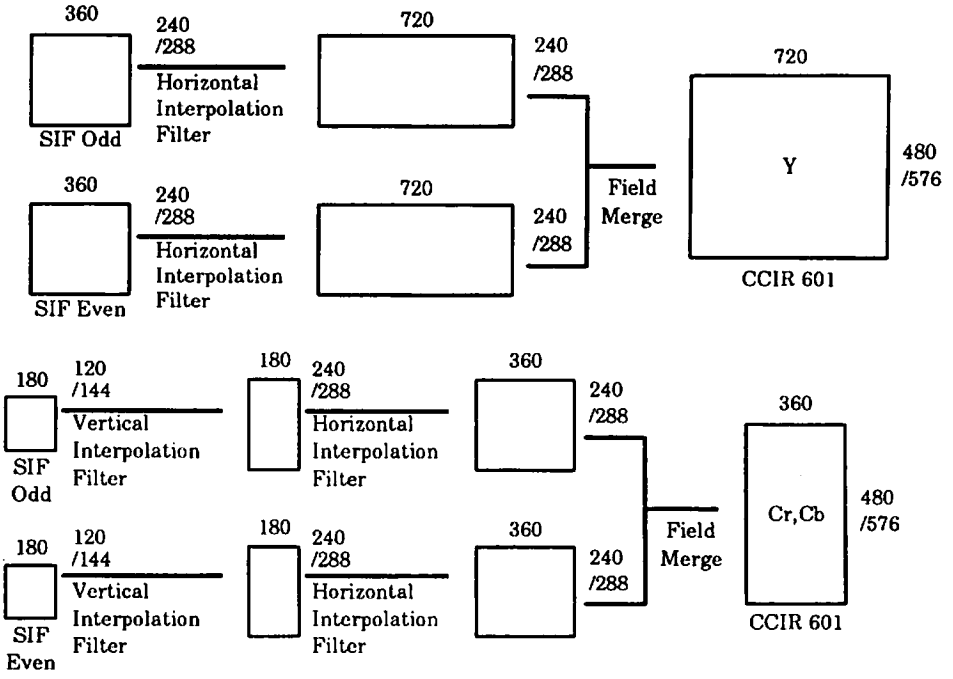


Figure 3.7: Conversion of SIF Odd and Even to CCIR 601

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.

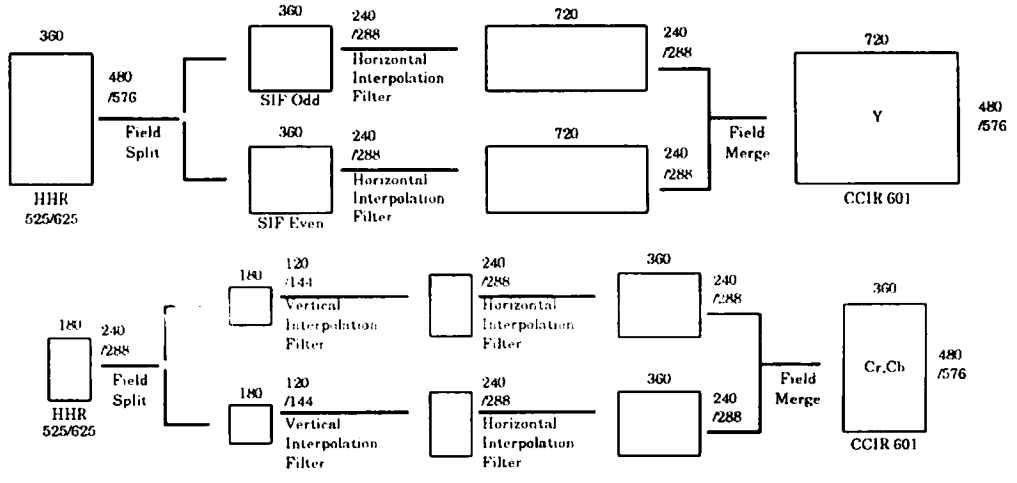


Figure 3.8: Conversion of HHR to CCIR 601

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.

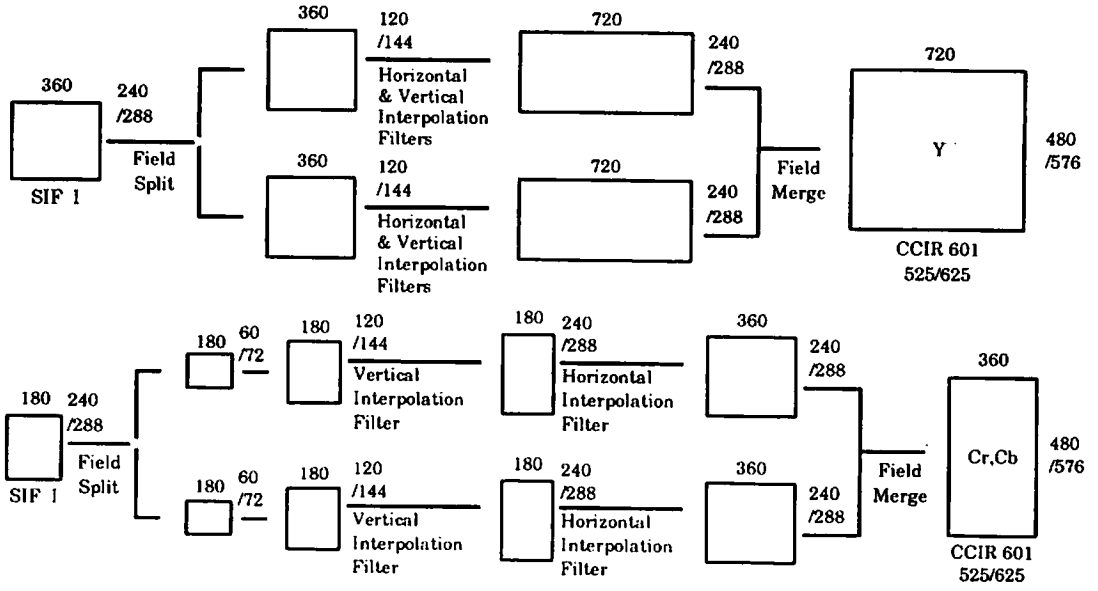


Figure 3.9: Conversion of SIF interlaced (SIF-I) to CCIR 601

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 pictures seem devoid of jerkiness but appear quite blurry. Better choice of interpolation filters needs further investigation.

3.5 Down conversion from interlaced to interlaced

This is not subject to standardisation, but information is provided for simulation purposes. This method has to be used to obtain the lower layer 'original signal', in future experiments related to compatible interlace to interlace coding. (SIF-I can be obtained from CCIR 601, and CCIR 601 can be obtained from interlaced HDTV.)

The following outlines a method for interlace-to-interlace conversion using non-adaptive vertical-temporal filtering apertures. Purely vertical filters compromise vertical resolution to give adequate motion performance and purely temporal filters sacrifice temporal resolution for good vertical frequency response. Vertical-temporal filtering apertures allow increased vertical resolution at low temporal frequencies reducing at higher temporal frequencies.

The derivation of these non-adaptive filter apertures may be understood as a two stage process, the first being an interlace to progressive conversion. This is followed by vertical filtering and sample-rate conversion within a progressive field [Ref. 1]. In the examples given in this paper, only a 2:1 (vertical) conversion is considered. However, once a progressive field has been "produced" (although not necessarily generated) appropriate filtering and vertical sample rate conversion could give any required

ratio.

Applying the filters

The proposed standards conversion will be considered as three separate processes.

1. Interlaced to progressive scan conversion.
2. Vertical filtering and sample rate changing within each progressive field.
3. Progressive to interlaced scan conversion (by appropriate sub-sampling within each progressive field).

In practice, these processes would be combined into a single vertical-temporal filtering and resampling operation.

The horizontal sample rate changing can be considered independently and the filter to used is specified in table 3.4.

Down Conversion

The input interlaced image should be padded with zeros to form a progressive grid, and then filtered, with the 3 field aperture filter given in table 3.6. The result is progressive fields.

The progressive fields are vertical filtered with the filter in table 3.7 prior to 4:1 vertical down sampling, by taking the appropriate lines.

3.6 Upconversion from interlaced to interlaced for display purposes

The low resolution coded image should be padded with zeros to form a progressive grid (e.g 625 interlace to 625 progressive) and then filtered using the three field aperture filter in table 3.6. The result is progressive fields at the higher resolution.

The filter in table 3.8 is a vertical interpolation filter. It is used by first padding with zeros to double the vertical sample rate and then applying this filter to remove unwanted aliases on the vertical frequency axis.

Finally, the picture can be reinterlaced to the output standard by selection of the appropriate lines within each field.

Reference: Devereux, V. G; "Standards conversion between 1250/50 and 625/50 TV systems" IBC 92 paper.

For up conversion, filter defined on a 625 line progressive grid.
For down conversion, filter grid defined on a 1250 line progressive grid.

z	y	coefficient
-1	-2	-0.0625
-1	0	0.125
-1	+2	-0.0625
0	-1	0.5
0	0	1
0	+1	0.5
+1	-2	-0.0625
+1	0	0.125
+1	+2	-0.0625

Table 3.6: Interlace to progressive vertical-temporal filter

Filter is defined on a 1250 line progressive grid.

y	coefficient
-5	-0.02926350
-4	-0.04344845
-3	0.01792812
-2	0.13801003
-1	0.26052380
0	0.31250000
+1	0.26052380
+2	0.13801003
+3	0.01792812
+4	-0.04344845
+5	-0.02926350

Table 3.7: Vertical filter used for 2:1 down sampling within progressive field

Filter defined on a 1250 line progressive grid.

y	coefficient
-5	0.04
-3	-0.16
-1	0.62
0	1
+1	0.62
+3	-0.16
+5	0.04

Table 3.8: Vertical filter used for 2:1 up sampling within progressive field

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 bitstream. In the bitstream, 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 than Bi-directional pictures then the other Groups of pictures.

A sequence can contain at the same time Field-Pictures and Frame-Pictures. Field pictures must occur in pairs.
NOTE: When coding progressive film material (in 60Hz) either TOP FIELD or the BOTTOM FIELD can be first field of a frame picture.

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.

4.3 Picture

Each Picture can be Frame-Structure or Field-Structure this is applicable to interlaced and progressive material. Frame structure, frame prediction may be used for progressive material. However it does not prohibit to use frame field adaptive or field structure.

[SEE WD Chapter 7]

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.

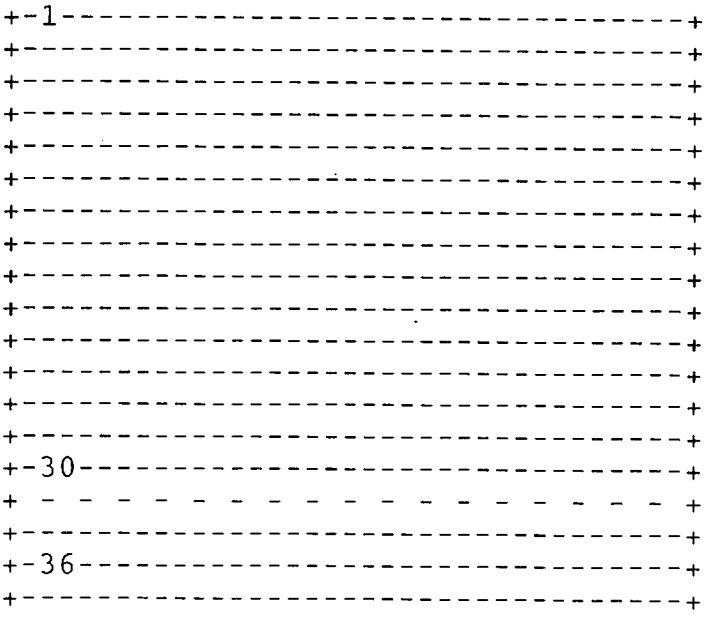


Figure 4.2 Arrangement of Slices in a Picture in Frame Coding mode

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.

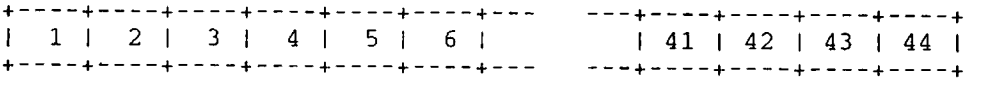


Figure 4.3: Test model Macroblock Slice Structure

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

4.5 Macroblock

[SEE WD Chapter 7]

4.6 Block

[SEE WD Chapter 7]

4.6.2 Block in harmonised scalable solution, for low level decoders

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 bitstream.

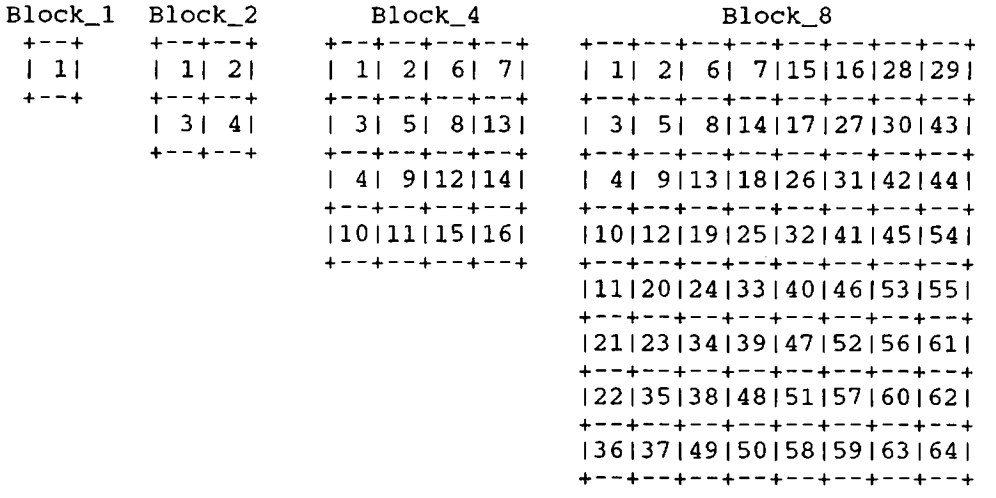


Figure 4.8: Block structures for scalable bitstreams

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	± 7 pels horizontal, ± 3 pels vertical/field
Flower Garden	±15 pels/frame	± 7 pels horizontal, ± 3 pels vertical/field
Calendar	±15 pels/frame	± 7 pels horizontal, ± 3 pels vertical/field
Popple	±15 pels/frame	± 7 pels horizontal, ± 3 pels vertical/field
Footiball	±31 pels/frame	±15 pels horizontal, ± 7 pels vertical/field
PRL CAR	±63 pels/frame	±31 pels horizontal, ±15 pels vertical/field

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_FRAME = MAXINT;
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.

5.1.3 Motion estimation for Special prediction mode

The first step is to obtain four candidate motion vectors as follows :

- First, four field motion vectors with half-pel accuracy from reference field 1 / field 2 to predicted field 1 / field 2 are searched by normal motion vector search defined in the Test Model. Then these vectors are appropriately scaled, if the parity of the predicted field is opposite to that of the predicted field.
- The second step is to evaluate the prediction errors of Dual-prime prediction using possible

combinations of four candidate motion vectors obtained by the first step, and 3Vx3H = 9 candidate differential motion vectors.

The prediction error is computed using the reconstructed pictures. The combination with the smallest MSE is selected.

5.2 Motion Compensation

[SEE WD Chapter 7 and 8]

5.3 Special prediction mode

5.3.1. Overview of *Special* Prediction mode

There is only one *special* prediction mode (Dual-prime) remaining in this Test Model and this is based on Field-based prediction. **THIS IS ONLY USED FOR M=1 CODING (NO B_FRAMES) FOR THE MAIN PROFILE, MAIN LEVEL. FOR OTHER PROFILES AND LEVELS IT HAS NOT BEEN DECIDED.** This mode has been included in particular for low delay applications.

Dual Prime prediction involves the averaging of two forward field based predictions from the last two nearest decoded fields (in time).

In the syntax of the Special prediction mode, for forward prediction, one field motion vector is transmitted, followed by a differential motion vector. Each of the coordinates of the differential motion vector is limited to the values [-1, 0, +1] (half pixel values), and is transmitted with a 1-2 bit code. Combinations of the transmitted field motion vector (possibly scaled according to the field temporal distance) and of the differential motion vector are used for the prediction, as described in the following sections. A separate section defines precisely how field motion vectors are scaled.

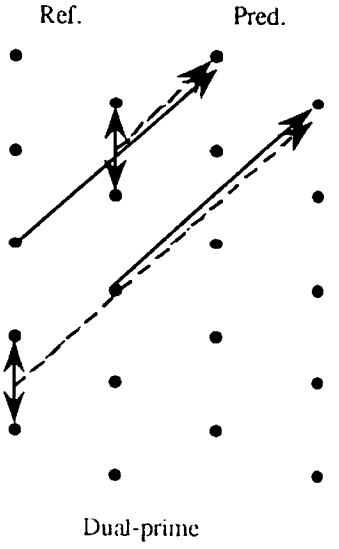


Figure 5.1 : Special prediction mode (frame structure picture coding mode)

Plain arrows represent the transmitted field motion vector. Dashed arrows represent the scaled-up or scaled-down field motion vectors. Vertical arrows represent the transmitted differential motion vector.

5.3.2. Specification of Dual-prime vectors

Motion vectors used for Dual-prime prediction are field motion vectors obtained as follows:

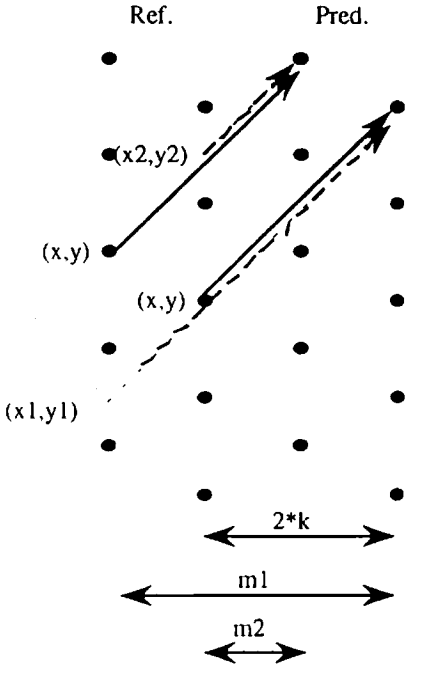
1. If the reference field and the predicted field are same parity, the field motion vector used is equal to the transmitted field motion vector.
2. If the reference field and the predicted field are different parity, the field motion vector used is obtained by adding the differential motion vector to the scaled transmitted motion vector.

NOTE: that the same differential motion vector is used for the scaled-down and the scaled-up field motion vectors.

5.3.3. Temporal Scaling of the Field Motion Vector

The transmitted field motion vector (x, y) corresponds to the temporal distance between two fields of same parity. The horizontal and vertical coordinate are in 1/2-pel units.
The transmitted field motion vector is used for computing two scaled field motion vectors that serve in the Special prediction mode when reference field and predicted fields are opposite parity. One of the scaled field motion vectors is longer ("scaled-up"), the other one is shorter ("scaled-down").
Scaling is done as follows :

If the same parity reference frame is at a distance of 2*k fields from the predicted field, the coordinates (x', y') of the scaled motion vector used for accessing the different-parity field is computed as follows:
x' = (x * K) // 32 (x and x' are integers)
y' = ((y * K) // 32) + e (y and y' are integers)
K = (m * 16) // k (k is integer)
m = field-distance between the predicted field and the different-parity-field. NOTE: FURTHER APPROXIMATION OF SCALING SHALL BE REDEFINED(See MPEG93/227)
The "e" is an adjustment necessary to reflect the vertical shift between the lines of field 1 and field 2. To give an example, line 1 of field 2 is in fact located 1/2 line under line 1 of field 1.
e is defined as follows :
e = -1 if the reference field corresponding to the scaled vector is field 2
e = +1 if the reference field corresponding to the scaled vector is field 1



[NOTE: The formula assumes frame based coding and will be updated]

.3.4. Prediction of Chrominance Blocks

The motion vector used for chrominance is obtained from the luminance Dual-prime motion vector with precisely the same rule as in the case of field-based prediction (for 4:2:0 : divide each coordinate by 2 as described section 5.2.2.1. of TM). The rules of prediction are same as for lumanance.

6 MODES AND MODE SELECTION

In section 6.1, a coding structure with different picture modes is introduced. Within each picture, macroblocks may be coded in several ways, thus aiming at high coding efficiency. The MB modes for intra, predicted and interpolated pictures 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 picture of a sequence starting with the first picture is coded as intra picture i.e. pictures 1, N+1, etc. (see Fig. 5.1). Following every M-th picture in between (within a Group of Pictures) is a predicted picture coded relative to the previous predicted or intra picture. The interpolated pictures are coded with reference to the two closest previous and next predicted or intra pictures. 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:

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

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

6.2 Macroblock type decision

Use the MSE critium to select the best Macroblock mode.

6.2.1 Modification of Decision for Field-based Prediction

In order to take advantage of the Special prediction modes, the decision rule must be modified for Field-based prediction.

It has been noted that quality is improved by choosing Field-based prediction less often, to the benefit of the Special prediction mode, particularly in B-Pictures.

For example, even in cases where Field-based prediction has an MSE slightly better than any of the other prediction modes, it may cost a significant overhead to transmit two field-vectors (four in B-Frames).

Until further improvement, we propose to use the following decision rule in core experiments involving the Special prediction modes :

- Field-based prediction chosen
  - in B-pictures: if (MSE_field + 8 < MSE_best_of_other_modes)
  - in P-Pictures: if (MSE_field < MSE_best_of_other_modes)

where MSE = Mean Square Error PER PEL of predicted MB

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 selected quantizers without a dead-zone.

7.1.1 DC Coefficients

The quantizer step-size for the DC coefficient of the luminance and chrominance components is 8, 4, 2 and 1. Thus, the quantized DC value, QDC, is calculated as:

QDC = dc // 8
QDC(9bit) = dc // 4
QDC(10bit) = dc // 2

where "dc" is the 11-bit unquantized value from the DCT.

7.1.2 AC Coefficients

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

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

where $w_I(i,j)$ 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 bitstream 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:

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

If the tcoef_escape_format flag is set to 0, QAC (i,j) is limited to the range [-255..255].
If the tcoef_escape_format flag is set to 1, QAC (i,j) is limited to the range [-2047 .. 2047].

For this TM p=3, and q = 4.

7.2 Quantization Non Intra Macroblocks

Non-intra macroblocks in Predicted and Interpolated pictures 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 pictures are quantized in exactly the same manner as macroblocks in Intra-pictures (section 7.1) and not as described in this section.

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

where:

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

$$QAC(i,j) = ac\sim(i,j) / (2*mquant)$$

If MPEG1 syntax applies, QAC (i,j) is limited to the range [-255..255].
If MPEG2 syntax applies, no clipping is necessary.

7.3 Dequantization

[SEE WD Chapter 8]

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 bitstream 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 bitstreams, see to Annex D, G and I.

8.1 Macroblock Addressing

[SEE WD Chapter 8]

8.2 Macroblock Type

[SEE WD]

8.2.1 Compatible Prediction

See Appendix G and J.

8.3 Motion Vectors

[SEE WD]

8.4 Coded Block Pattern

[SEE WD Chapter 7]

8.5 Intra picture Coefficient Coding

[SEE WD Chapter 7 and 8]

8.6 Non-Intrapicture Coefficient Coding

[SEE WD Chapter 7 and 8]

8.7 Coding of Transform Coefficients

First of all there are two VLC's, one for non intra macroblocks, and one for intra macroblocks. If MPEG1 only the non intra VLC is used. The two VLC differ in particular in the length of EOB code. The combinations of zero-run and the following value are encoded with variable length codes as listed in table Bx to By in the WD. The last bit 's' denotes the sign of the level, '0' for positive '1' for negative.

Blocks with no coefficient data are indicated by the CBP, and no EOB is required. Therefore EOB cannot occur as the first coefficient, and hence EOB does not appear in the VLC table for the first coefficient.

The most commonly occurring combinations of successive zeros (RUN) and the following value (LEVEL) are encoded with variable length codes listed in the tables. Less common combinations of (RUN, LEVEL) are encoded with a 24-bit word consisting of a 6 bit ESCAPE, a 6 bit RUN and a 12 bit LEVEL.

If MPEG1 then the ESCAPE code is followed by a 6 bit run and 8 or 16 bit level.

9 VIDEO MULTIPLEX CODER

[SEE WD Chapter 7 for most of the MAIN profile syntax]

This chapter will contain the extension to the MAIN profile syntax.

A flow diagram of the MPEG2 video syntax is given in figure 9.1.

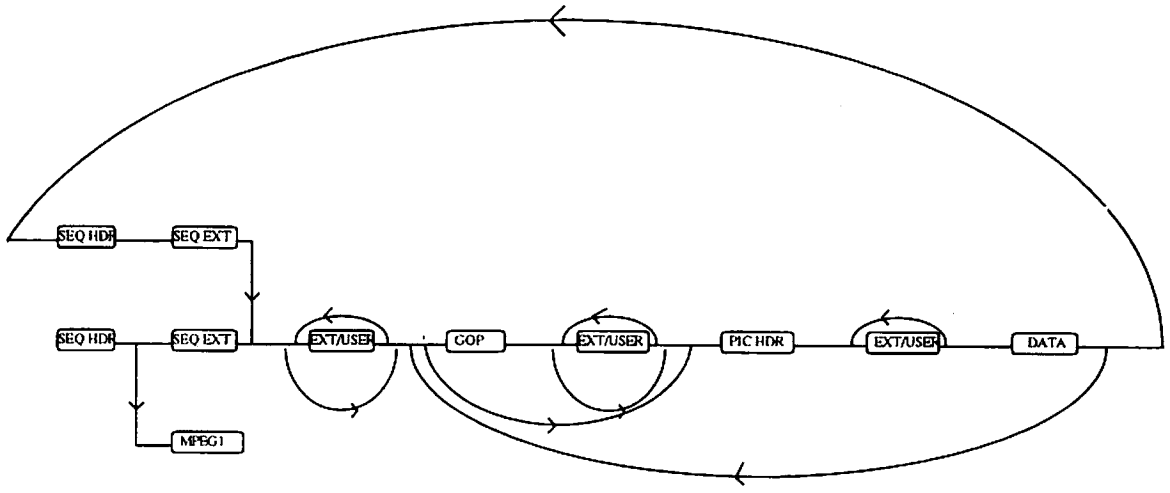


Figure 9.1: Video syntax flow diagram

9.1 Sequence Frequency Extension

sequence_frequency_extension() {		
extension_start_code	32	bslbf
extension_start_code_identifier	4	uimsbf
layer_id	4	uimsbf
fscale_code	8	uimsbf
subband	1	uimsbf
motion_compensation_loop	1	uimsbf
motion_refinement	1	uimsbf
scalable_side_information	1	uimsbf
low_resolution_prediction_horizontal_dimension	15	uimsbf
low_resolution_prediction_vertical_dimension	15	uimsbf
next_start_code()		
}		

layer_id - This is a 4-bit integer defining the scalable layer that the bit stream belong to. The layer_id starts at 0 for the base layer. The layer_id is incremented by 1 for each additional layer.

fscale_code - This is an 8-bit integer that defines the DCT size for the scalable layers. The values of this integer are:

fscale_code	dct_size
0	1
1	2
2	4
3	8
4	reserved
5	reserved
6	reserved
7	reserved
8	reserved
256	reserved

subband - if set to 1, this flag indicates that the lower layer contains the low frequency coefficients of each block, and the next higher layer will contain only the remaining coefficients that make up the block size for that layer. No incremental coefficients can be transmitted for the low frequency coefficients.

motion_compensation_loop - if set to 1, this flag indicates to the decoder that a prediction loop is used in this frequency layer in the encoder. This allows the decoder to take actions to ensure zero drift decoding in the lower scales.

motion_refinement - if set to 1, this flag indicates that incremental motion vector is added to the motion vector from the layer below. When used in conjunction with scalable_side_information, the motion vector of the layer below will have to be scaled by the appropriate factor prior to addition. This flag is set to zero if no refinement of the motion vector is done.

scalable_side_information - if set to 1, the relevent side information is transmitted in the layers where it is required.

low_resolution_prediction_horizontal_dimension - this is a 15 bits integer indicating the horizontal dimension of the low resolution image, using the high resolution as reference, which is used for prediction in the frequency or spatial domain.

low_resolution_prediction_vertical_dimension - this is a 15 bits integer indicating the vertical dimension of the low resolution image, using the high resolution as reference, which is used for prediction in the frequency or spatial domain.

9.2 Sequence spatial extension

sequence_spatial_extension() {	No. of bits	Mnemonic
extension_start_code	32	bslbf
extension_start_code_identifier	4	uimsbf
subsampling_ratio	3	uimsbf
compatible_mtype	2	uimsbf
decoder_memory_configuration	1	uimsbf
low_resolution_prediction_horizontal_dimension	15	uimsbf
low_resolution_prediction_vertical_dimension	15	uimsbf
next_start_code()		
}		

subsampling_ratio - This is an 3-bit integer that defines the subsampling ratio in the lower layer generating the spatial embedded prediction.

subsampling ratio 3 bits	
code	ssr
0	no subsampling
1	1/2 h
2	1/2 v
3	1/2 h, 1/2 v
4	reserved
...	"
7	"

compatible_mtype - This is a two-bit codeword which is used to indicate the set of macroblock_type tables to be used. The table is set out below.

codeword	macroblock_type tables
00	default
01	spatially scalable
10	snr scalable
11	chrominance scalability

If compatible_mtype is set to "00" then tables B.2a, B.2b, B.2c and B.2d.

If compatible_mtype is set to "01" then tables B.2a, B.2b, B.2c and B.2d are are replaced B.2a3, B.2b3, B.2c3 and B.2d3 respectively.

If compatible_mtype is set to "10" then tables B.2a, B.2b, B.2c and B.2d are as follows are replaced B.2a4, B.2b4, B.2c4 and B.2d4 respectively.

decoder_memory_configuration -- this is a 1 bit integer indicating the memory configuration of the compatible decoder. If it is set to 1 25% more memory is required then if set to 0. If the flag is set to 1 the encoder and decoder configuration correspond to the switch set to B in figure J.5 and J.8. If the the flag is set to 0, the switch is set to A in figure J.5 and J.8.

low_resolution_prediction_horizontal_dimension -- this a 17 bits integer indicating the horizontal

dimension of the low resolution image, using the high resolution as reference, which is used for prediction in the frequency or spatial domain.

low_resolution_prediction_vertical_dimension -- this a 17 bits integer indicating the vertical dimension of the low resolution image, using the high resolution as reference, which is used for prediction in the frequency or spatial domain.

9.3 Picture frequency extension

```
picture_frequency_data() {
  if ( picture_coding_type != 5 ) {
    do {
      scaled_slice()
    } while ( nextbits() == slice_start_code )
  }
}
```

9.4 Picture spatial extension

picture_spatial_extension() {	No. of bits	Mnemonic
extension_start_code	32	bslbf
extension_start_code_identifier	4	uimsbf
lower_picture_reference	10	uimsbf
load_prediction_weighting_matrix	1	uimsbf
if (load_prediction_weighting_matrix) {		
prediction_weighting_matrix[8]	4*8	uimsbf
}		
overlap_horizontal_left_upper_offset	15	uimsbf
overlap_horizontal_left_upper_offset	15	uimsbf
if (interlaced &&		
(picture_structure == frame_structure)) {		
overlap_horizontal_left_upper_offset	15	uimsbf
overlap_horizontal_left_upper_offset	15	uimsbf
}		
next_start_code()		
}		

lower_picture_reference -- An unsigned integer value which indicates the lower layer temporal_reference to be used for prediction. If temporal reference is shorter than 10 bit in H.261 case, 5 bits are used for temporal reference), the lower bits are used and 1's are stuffed from MSB. Of course, lower_picture_reference is applied to upper layer only.

load_prediction_weighting_matrix - This is a one-bit flag which is set to "1" if the prediction_weighting_matrix follows. If it is set to "0" then the default values defined below are used until the next occurence of the sequence header. If the pictures are field based

Table 9.3.3.1 Default weights for field based coding.

prediction weight code	Prediction Weightings	
	Same Parity Field	Opposite Parity Field
00	12	4
01	10	4
10	8	4
11	6	4

If the pictures are frame based

Table 9.3.3.2 Default weights for frame based coding.

prediction weight code	Prediction Weightings	
	Same Parity Field	Opposite Parity Field
00	12	4
01	10	6
10	10	8
11	8	8

[NOTE: There is an inconsistency with Appendix G.1.2, experiments for both cases are welcomed]

prediction_weighting_matrix - This is a list of eight 4-bit unsigned integers. The new values replace the default values. The values are sent as five bytes, shortest vlc code first. The most significant 4 bits correspond to the weight for the same parity field. The least significant 4 bits correspond to the weight for the opposite parity field. The allowed values are defined below in table 9.3.3.3. The new values shall be in effect until the next occurence of a sequence header.

Table 9.3.3.3 Allowed weight values.

code	value
0	illegal
1	-0.375
2	-0.25
3	-0.125
4	0
5	0.125
6	0.25
7	0.375
8	0.5
9	0.625
10	0.75
11	0.875
12	1
13	1.125
14	1.25
15	1.375

[NOTE: There is an inconsistency with Appendix G.1.2, experiments for both cases are welcomed]

overlap_horizontal_left_upper_offset -- this is an 15 bits signed integer, indicating the horizontal postion of the left upper corner of a retangular area which can be predicted (in the spatial or frequency domain) from a lower video resolution, this flag applies to the corresponding field, for interlaced, for progressive to the corresponding frame..

overlap_vertical_left_upper_offset -- this is an 15 bits signed integer, indicating the vertical postion of the left upper corner of a retangular area which can be predicted (in the spatial or frequency domain) from a lower video resolution, this flag applies to the corresponding field, for interlaced, for progressive to the corresponding frame.

9.5 Scaled Slice Layer

scaled_slice() {		
slice_start_code	32	bslbf
if (layer_id == 0) quantizer_scale	5	uimsbf
else slave_slice_quantizer_ratio	9	uimsbf
while (next_bits() == '1') {		
extra_bit_slice	1	"1"
extra_information_slice	8	
}		
extra_bit_slice	1	"0"
if (layer_id == 0 || scalable_side_information) {		
do {		
scaled_macroblock()		
} while (next_bits() != 000 0000 0000 0000 0000 0000)		
}		
else {		
do {		
slave_macroblock()		
} while (next_bits() != 000 0000 0000 0000 0000 0000)		
}		
next_start_code()		
}		

slave_slice_quantizer_ratio - A bit pattern used to represent the ratio between the mquant value of the slave layer and base layer. The bit pattern is an integer representation of a real number nnnnn.xxxx, where the MSB corresponds to a value of 16. This allows a range from 0.0625 to beyond 31 for the **slave_slice_quantizer_ratio**.

9.6 Slave Macroblock Layer

```
slave_macroblock() {
  if ( macroblock_motion_forward && motion_refinement)
    forward_motion_vectors()                          ...        ...
  if ( macroblock_motion_backward && motion_refinement)
    backward_motion_vectors()                         ...        ...
  for (i=0; i < block_count; i++) {
    slave_block(i)
  }
}
```

9.7 Scaled Macroblock Layer

scaled_macroblock() {		
if (<sequence extension was not present>)		
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**
if (macroblock_motion_forward ||		
macroblock_motion_backward) {		
if (picture_structure == 'frame') {		
if (frame_pred_frame_dct == 0)		
frame_motion_type	**2**	**uimsbf**
} else {		
field_motion_type	**2**	**uimsbf**
}		
}		
if ((picture_structure == 'frame') &&		
(frame_pred_frame_dct == 0) &&		
(macroblock_intra || macroblock_pattern))		
dct_type	**1**	**uimsbf**
}		
}		
else {		
incremental_macroblock_type	**1-5**	**vlclbf**
}		
if (macroblock_quant)		
quantizer_scale	**5**	**uimsbf**
if (macroblock_motion_forward && motion_refinement)		
forward_motion_vectors()	**...**	**...**
if (macroblock_motion_backward && motion_refinement)		
backward_motion_vectors()	**...**	**...**
if (macroblock_pattern)		
coded_block_pattern()	**...**	**...**
for (i=0; i<block_count; i++) {		
if (new_macroblock) {		
scaled_block(i)		
}		
else {		
if (coded_coefficients) slave_block(i)		
}		
}		
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. Therefore new_macroblock is always true for layer_id == 0.
incremental_macroblock_type - a VLC which enables the selection of additional mquant and cbp values for the current macrobock according to the following table.

VLC Word	macroblock_quant	macroblock_pattern	coded coefficients
1	0	0	1
01	0	1	1
001	1	0	1
0001	1	1	1
00001	0	0	0

coded_coefficients - this indicates if there are any coefficients coded for the layer.

9.8 Spatial Macroblock layer

macroblock() {	No. of bits	Mnemonic
if (<sequence extension was not present>)		
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	1-8	vlclbf
if (macroblock_compatible && picture_coding_type != 1 && compatible_mtype != '10')		
prediction_weight_code	**1-3**	vlclbf
if (macroblock_motion_forward ||		
macroblock_motion_backward) {		
if (picture_structure == 'frame') {		
if (frame_pred_frame_dct == 0)		
frame_motion_type	2	uimsbf
} else {		
field_motion_type	2	uimsbf
}		
}		
if ((picture_structure == 'frame') &&		
(frame_pred_frame_dct == 0) &&		
(macroblock_intra || macroblock_pattern))		
dct_type	1	uimsbf
if (macroblock_quant)		
quantizer_scale	5	uimsbf
if (macroblock_motion_forward ||		
(macroblock_intra && concealment_motion_vectors))		
forward_motion_vectors()	...	...
if (macroblock_motion_backward)		
backward_motion_vectors()	...	...
if (macroblock_intra && concealment_motion_vectors)		
marker_bit	1	

if (macroblock_pattern)		
coded_block_pattern()	...	...
if (!chroma_scalable) {		
block_count_start =		
block_count_end = block_count;		
} else {		
block_count_start = 4;		
if (chroma_format == 4:2:2) {		
block_count_end = 8;		
} else if (chroma_format == 4:4:4) {		
block_count_end = 12;		
}		
}		
for (i=block_count_start; i<block_count_end; i++) {		
if (compatible_mtype == "10" && !macroblock_pattern) {		
snr_block(i)		
} else {		
block(i)		
}		
}		
if (picture_coding_type == 4)		
end_of_macroblock	1	"1"
}		

prediction_weight_code - This is a variable length coded integer coded as per table 9.3.3.1 or 9.3.3.2 which indicates the weighting to be used on the compatible and normal prediction. The weighting is a pair of weights (one for each parity field) each weight has a 4-bit accuracy.

9.8 CBP

coded_block_pattern () {		
if (chroma_scalable) {		
if (chroma_format == 4:2:2) {		
coded_block_pattern_chroma_scalable_422	4	uimsbf
} else {		
if (chroma_format == 4:4:4) {		
coded_block_pattern_chroma_scalable_444	8	uimsbf
}		
}		
} else {		
coded_block_pattern_420	3-9	vlclbf
if (chroma_format == 4:4:4) {		
coded_block_pattern_1	2	uimsbf
coded_block_pattern_2	2	uimsbf
coded_block_pattern_3	2	uimsbf
} else if (chroma_format == 4:2:2) {		
coded_block_pattern_1	2	uimsbf
}		
}		
}		

coded_block_pattern_1: This is a 2-bit FLC made of bits <b7> and <b8> of cbp

coded_block_pattern_2: This is a 2-bit FLC made of bits <b9> and <b10> of cbp

coded_block_pattern_3: This is a 2-bit FLC made of bits <b11> and <b12> of cbp

coded_block_pattern_420 -- This is a VLC derived from bits <b1> to <b6> of cbp, using Table B.3 of MPEG-1, with one extension (see table B.3).

coded_block_pattern_chroma_scalable_422 -- This a 4 bits FLC made of bits <b5> to <b8> of cbp

coded_block_pattern_chroma_scalable_444 -- This a 6 bits FLC made of bits <b5> to <b12> of cbp

9.8 Scaled Block Layer

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**	**uimsbf**
}		
else {		
dct_dc_size_chrominance	**2-8**	**vlclbf**
if(dct_dc_size_chrominance !=0)		
dct_dc_differential	**1-8**	**uimsbf**
}		
}		
else if (scalable_side_information) {		
dct_coeff_first	**2-28**	**vlclbf**
}		
if (picture_coding_type != 4) {		
while (nextbits() != '10' && more_coeffs)		
dct_coeff_next	**3-28**	**vlclbf**
if (more_coeffs) **end_of_block**	**2**	**"10"**
}		
}		
}		

9.9 Slave Block Layer

slave_block(i) {		
if (pattern_code[i]) {		
while (nextbits() != '10' && more_coeffs)		
dct_coeff_next	3-28	vlclbf
if (more_coeffs) **end_of_block**	2	"10"
}		
}		

pattern_code[i] - For slave_blocks, this code is the same as that of the correlated scaled_block in the slice layer unless it is overriden by a slave coded block pattern.

more_coeff - more_coeff 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).

9.10 SNR Block layer

snr_block(i) {		
while (nextbits!=end_of_block) {		
dct_coef_next	3-28	vlclbf
}		
end_of_block	**2**	**"10"**

Comment
- dct_coeff_next refers to table B.5c, B.5d B.5e, B.5f and B.5g that is current vlc coefficient table without the first 'coefficient trick' and therefore an end_of_block codeword is transmitted for every block.

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 picture. It is performed before coding the picture.
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 picture 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 picture 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

$X_i = 160 *$ **bit_rate** $/ 115$
$X_p = 60 *$ **bit_rate** $/ 115$
$X_b = 42 *$ **bit_rate** $/ 115$

bit_rate is measured in bits/s.

Picture Target Setting

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

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

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

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

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 picture , $R = R - S_{i,p,b}$

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 picture in a GROUP OF PICTURES (an I-picture):

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

At the start of the sequence R = 0.

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

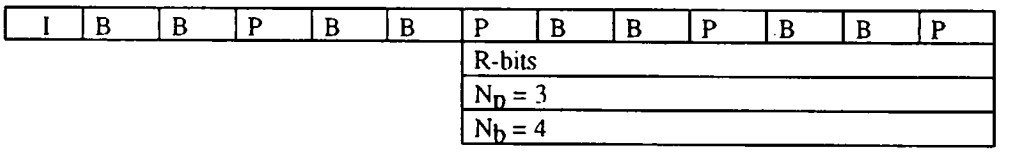


Figure 10.1 - Remaining pictures in GOP

Step 2 - Rate Control

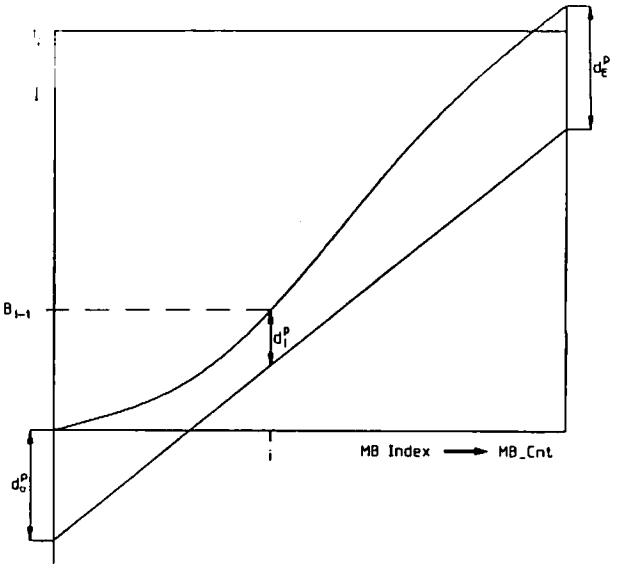


Figure 10.2 : Rate Control for P-pictures

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

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

or

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

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 picture type.
- B_j is the number of bits generated by encoding all macroblocks in the picture up to and including j.
- MB_cnt is the number of macroblocks in the picture.
- d_j^i, d_j^p, d_j^b are the fullnesses of virtual buffers at macroblock j- one for each picture 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 picture 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 frame-organised sub-blocks and the four luminance field-organised sub-blocks using the intra (i.e. original) pixel values:

$$act_j = 1 + \min_{sblk=1,8}(var\_sblk)$$

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 picture to be encoded. On the first picture, 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.
- 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)

[SEE WD Appendix A]

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

[SEE WD Table B.1]

B.2 Macroblock Type and Compatible Macroblock Type

If compatible_mtype is set to "00" then tables B.2a, B.2b, B.2c and B.2d are as follows

[SEE WD Table B.2, B3, B4 and B5]

If compatible_mtype is set to "01" then tables B.2a, B.2b, B.2c and B.2d are as follows

Table B.2a3. 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	macroblock_ compatible
1	0	0	0	1	0	1
01	0	0	0	0	1	0
001	1	0	0	1	0	1
00011	1	0	0	0	1	0
00010	0	0	0	0	0	1

Table B.2b3. 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	macroblock_ compatible
10	0	1	0	1	0	0
11	0	1	0	1	0	1
010	0	0	0	1	0	0
011	0	0	0	1	0	1
0010	0	1	0	0	0	0
0011	0	0	0	0	1	0
000110	0	1	0	0	0	1
000111	1	1	0	1	0	0
000100	1	0	0	1	0	0
000101	1	0	0	0	1	0
0000110	1	1	0	1	0	1
0000111	1	0	0	1	0	1
0000100	0	0	0	0	0	1

Table B.2c3. 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	macroblock_ compatible
10	0	1	1	0	0	0
11	0	1	1	1	0	0
010	0	0	1	0	0	0
011	0	0	1	1	0	0
0010	0	1	0	0	0	0
0011	0	1	0	1	0	0
000110	0	0	1	0	0	1
000111	0	0	1	1	0	1
000100	0	1	0	0	0	1
000101	0	1	0	1	0	1
0000110	0	0	0	0	1	0
0000111	1	1	1	1	0	0
0000100	1	1	0	1	0	0
0000101	1	0	1	1	0	0
00000100	1	0	0	0	1	0
00000101	1	1	0	1	0	1
00000110	1	0	1	1	0	1
00000111	0	0	0	0	0	1

Table B.2c4. 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	macroblock_ compatible
1	0	0	0	0	1	0

Comments
- For the intra picture a compatible macroblock has the shortest code saving 3 bits/macroblock over the previous scheme (1-bit compatible_type flag and 2-bit weight_code).
- In the bidirectionally predicted pictures compatible prediction is not allowed for bidirectionally predicted macroblocks.
- The final entry in tables B.2a, B.2b and B.2c correspond to a macroblock which is compatibly predicted in which there are no coded coefficients and is not of the same macroblock type to the previous coded macroblock (for instance the previous coded macroblock had a different weight_code).

If compatible_mtype is set to "10" then tables B.2a, B.2b, B.2c and B.2d are as follows

Table B.2a4. 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	macroblock_ compatible
1	0	0	0	0	0	1
01	1	0	0	0	0	1
001	0	0	0	1	0	1
0001	1	0	0	1	0	1

Table B.2b4. 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	macroblock_ compatible
1	0		0	0	0	1
01	1	0	0	0	0	1
001	0	0	0	1	0	1
0001	1	0	0	1	0	1

Table B.2c4. 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	macroblock_ compatible
1	0	0	0	0	0	1
01	1	0	0	0	0	1
001	0	0	0	1	0	1
0001	1	0	0	1	0	1

Table B.2d4. 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	macroblock_ compatible
1	0	0	0	0	1	0

B.3 Macroblock Pattern

[SEE WD Table B.6]

B.4 Motion Vectors

[SEE WD Delta Table B.8]

B.5 DCT Coefficients

[SEE WD Table 10, Table 11 and Table 12]

APPENDIX C : VIDEO BUFFER VERIFIER

[SEE WD Annex C]

APPENDIX D: FREQUENCY DOMAIN SCALABILITY EXTENSION

D.1 INTRODUCTION

The syntax for frequency domain scalable bitstreams 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 suband schemes in the frequency (DCT) domain. Although the syntax allows a flexible number of layers, 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.1.1 and D.1.2 are block diagrams outline possible Test Model encoder implementation for the frequency domain scalability experiments for the two highest resolutions. It is thus possible to tailor the complexity of the encoding scheme to the functionality and coding efficiency desired by particular applications.

The drift encoder scheme outlined in Figure D.1.1 can be implemented with low hardware complexity as a simple extension of a MPEG2 Main Profile stand alone coder. It already provides suitable scalable features for many applications when low implementation complexity and high coding efficiency are desired (i.e. layered coding for cell loss resilience, advanced fast forward and fast reverse modes, graceful degradation and basic resolution scalability features). For the reconstruction of the lower scale layer resolutions drift can not be avoided but may not be of concern for some applications. Drift reduction techniques which can be applied at the decoder side and are thus not subject for standardisation have been developed and are outlined in Appendix D.10.

Figure D.1.2 outlines a drift-free implementation which also provides additional flexibility for the video input in the lower scale layers. Any frequency or spatial domain decimation technique can be applied to provide suitable input source for the lower scale layers. This may especially be useful if very good interlace quality is required for the lower resolutions.

The decoder structure for scale_8 and scale_4 is depicted in Figure D.2.1 and Figure D.2.2 and can be used to decode the bit stream generated by either drift or drift-free encoding scheme. The decoder used to reconstruct the highest scale_8 R.601 resolution is a simple extension of a MPEG2 main profile stand-alone implementation with only one decoding loop.

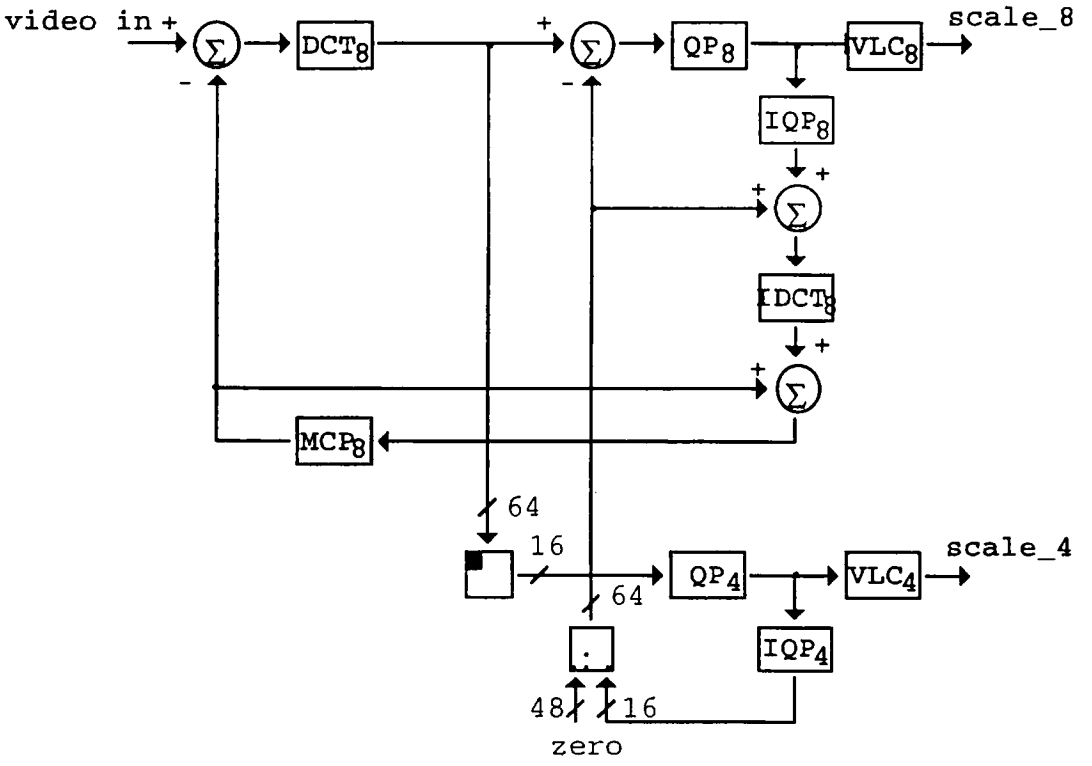


Figure D.1.1: Possible Frequency domain encoder scheme with two layers but only one motion compensation loop (Drift encoder).

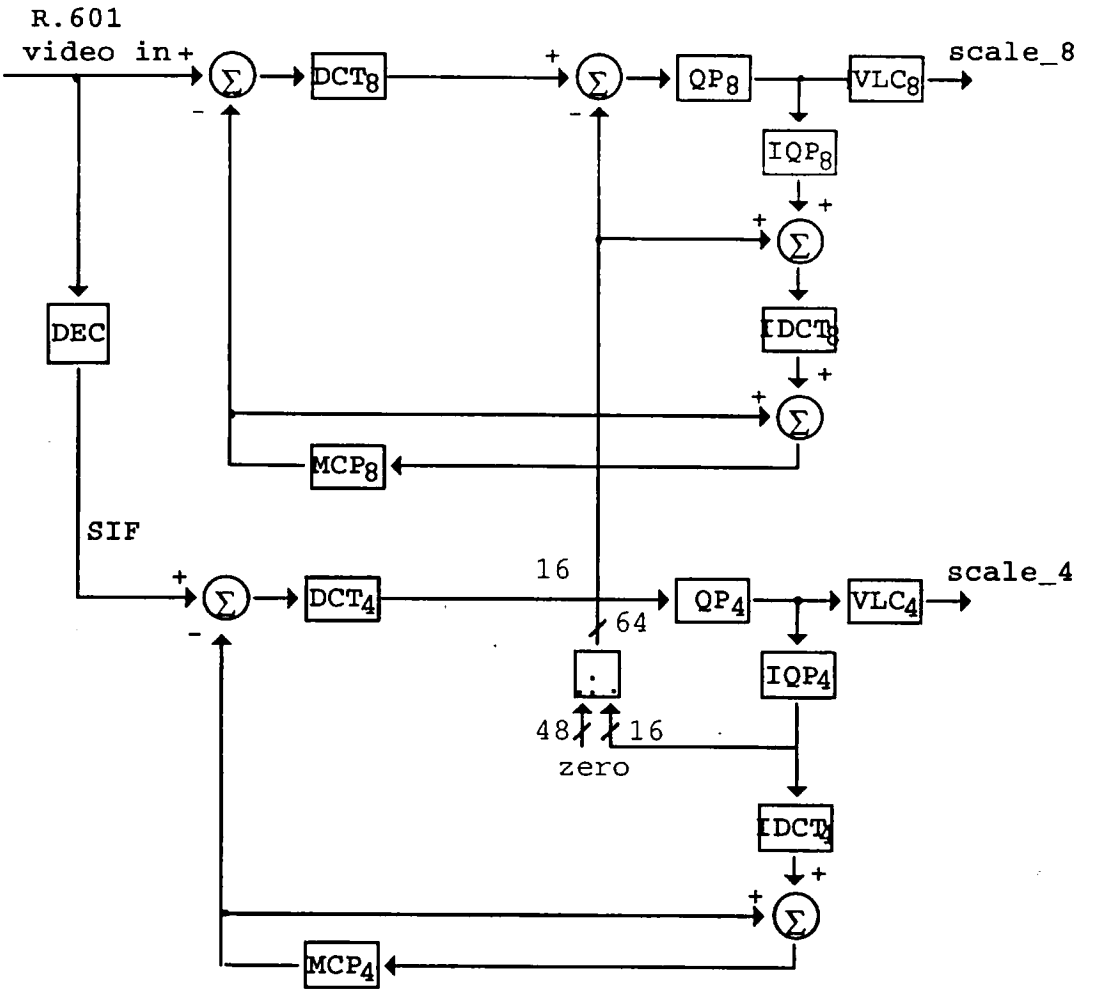


Figure D.1.2: Possible Frequency domain encoder scheme with two layers and two motion compensation loops (Drift-free encoder).

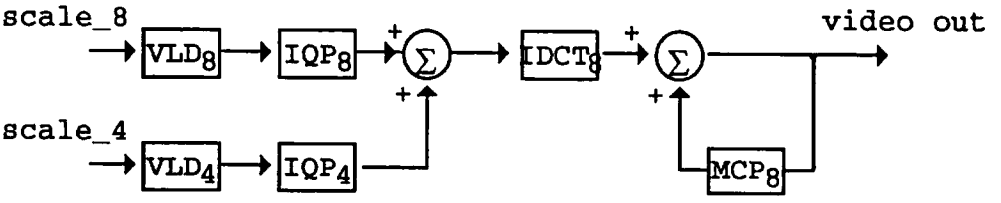


Figure D.2.1: Frequency domain scalable decoder for R.601 resolution

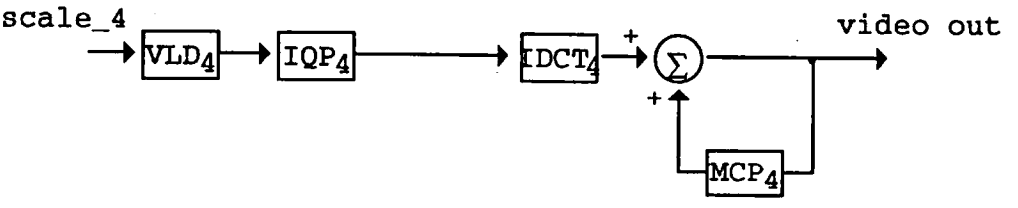


Figure D.2.2: Frequency domain scalable decoder for SIF resolution

One fundamental characteristic of the Test Model 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. One variation under test is the use of scalable side information, which allows some information such as new quantizer values, coded block patterns, and motion vector refinements to be transmitted in layers other than the base layer.

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.

The syntax allows the implementation of sub-band as well as pyramidal encoders that operate in the frequency domain. 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 sub-band implementation is useful for drift encoder schemes as outlined in Figure D.1.1. The pyramid version provides additional flexibility and functionality, such as the possibility to distribute bit rate between layers more efficiently. It is particularly recommended to use the pyramid implementation in combination with the drift-free encoding scheme outlined in Figure D.1.2.

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 systems level, according to the syntax in Chapter 9. Each **sequence_frequency_extension** has its own layer_id which identifies the layer that it represents. The base layer has a layer_id of 0. For each layer up the layer_id is incremented by 1. In a scalable bit-stream, the **picture layer** contains **scaled_slices** that carry data for the layer. 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 each layer is given by the fscale_code in the **sequence_frequency_extension** of the layer.

The **scaled_macroblock** syntax for base layer of 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 **scaled_slices** of the other layers contain **slave_macroblocks** which, in turn, contain additional DCT coefficient data in **slave_blocks**. 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 sequence of the non-scalable Test Model is replaced in the scalable Test Model by the base sequence containing 2x2 DCT data for QCIF resolution pictures and all the MB side information for

all layers; and two slave sequences, the first containing 4x4 DCT data for SIF resolution pictures, and the second, 8x8 DCT data for CCIR 601 resolution pictures.
Other more flexible layering scheme can be employed. Different number of layers can be used. The relative dct sizes of the layers can also be chosen depending on the application. For example, by choosing the same dct size for both layers we have SNR scalability. Interworking between QCif and Rec 601 can also be achieved by ignoring the 4x4 DCT layer.
In the case of scalable_side_information some MB side information may be transmitted in the slave sequences.

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 picture at scale-s.
Ps(x,y) = pixel x,y of motion-compensated prediction picture 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 picture 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 after dividing the luminance motion vector by 2. Thus, the precision used for chrominance is the same as that used for luminance.

D.4 MODES AND MODE SELECTION

The exact treatment of macroblock modes in frequency domain scalable video depends on whether scalable side information is being used and, to a lesser extent, on whether the source is interlaced.

D.4.1 No Scalable Side Information

The same picture types and coding modes used in the non-scalable Test Model are used in the frequency domain scalable extension without scalable side information. 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. All prediction and coding modes available in the Working Draft can be used. The same mode is inherited by all the slave macroblocks that are co-sited with the base macroblock. For interlaced input sources, the same motion compensation modes as used in the non-scalable Test Model are allowed in the frequency domain scalable Test Model. The decision rule is the same as in that case, except that frame motion vectors which are not a multiple of 4 frame lines are disallowed in a three-layer scheme to prevent field-flip from occurring in the luminance motion compensation.

D.4.2 Scalable Side Information

Scalable side information enables side information (address increment, macroblock type and coded block pattern) to be sent in the higher layers where it is required. Relative slave macroblock address are coded as described in section 8.1. The available slave macroblock types are

- coded coefficients with slave coded block pattern
- coded coefficients with coded block pattern inherited from previous layer
- coded coefficients with modified quantizer and slave coded block pattern
- coded coefficients with modified quantizer but no slave coded block pattern
- no coded coefficients and no slave coded block pattern (not sent)

The prediction modes and intra/inter decisions are determined by the first macroblock_type (in the order of layers, from low resolution to high resolution) transmitted for the macroblock. In each macroblock of non-base channels, the quantizer (slave_quantizer_scale) used is derived according to the following equation

slave_quantizer_scale = quantizer_scale * slave_slice_quantizer_ratio // 16.

The parameter slave_slice_quantizer_ratio is transmitted in the slice header, and is computed as described in the description for core experiment I.4. When macroblock_quant is 1 a new quantizer_scale is transmitted. When macroblock_quant is 0 the (slave_)quantizer_scale from the previous macroblock at the layer is inherited.

The coded block pattern can be inherited, changed or reset to zero depending on the slave macroblock type. Coded block pattern are determined by the quantized coefficients of the current layer. Incremental macroblocks are not coded when there is no new information to be transmitted.

D.4.3 Motion Refinement

When scalable_side_information is set the scaled motion vector is transmitted in the base layer, otherwise no scalling is done and the full resolution vector is transmitted. 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 **motion_refinement** 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.

In the case where the dct_size is the same as the layer below, no scaling of the motion vector is required. If motion refinement is not required then the **motion_refinement** flag is set to zero.

D.5 INTER-SCALE 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, for progressive sources, the 2x2 DCT coefficients of the Test Model base layer are used to predict the upper-left 2x2 coefficients of the corresponding coefficients in the 4x4 layer. Similarly, the 4x4 coefficients of the 4x4 layer are used to predict the upper-left 4x4 coefficients of the corresponding coefficients in the 8x8 layer. After computing the prediction differences, these differences together with the new (not predicted) coefficient data are coded using a layered sequential zig-zag scan pattern as shown in Figure D.3.

```
 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
```

Figure D.3: Scan patterns for layer-sequential scanning.

D.6 TRANSFORMATION AND QUANTIZATION

D.6.1 Transformation

In order to facilitate inter-scale 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 with a single motion-compensation loop and no layer rate control the same quantizer_scale 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 (encoder option).
(We emphasize that, the generality exists to use at least different quantizer_scale 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 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. dequantization of the DCT coefficients of the target scale and all lower scales using the appropriate quantizer scale factor "quantizer_scale_s" (quantizer_scale_2 for scale 2, quantizer_scale_4 for scale 4, and quantizer_scale_8 for scale 8), and quantization matrix, and
2. for appropriate coefficients, summation of the corresponding coefficient resulting from the previous step (inter-scale DCT coefficient prediction).

D.6.3 BANDWIDTH CONTROL OF RESOLUTION LAYERS

The additional channel layer specification includes a quantizer_ratio parameter. For each MB in a such a layer, this ratio is computed as given in section D.9, slice granularity approach. This parameter is used to derive the "quantizer_ratio" values used in the additional layers. For the basic encoder without layer rate control, quantizer_ratio is always 1.0.

D.7 DCT COEFFICIENT CODING

Coding of DCT coefficients is done using the default VLC tables MPEG-2 VLC tables. This is a new feature, as up until Rome, three special VLC tables existed for frequency scalable coding.

D.8 VIDEO MULTIPLEX CODER

See Test Model 5, Chapter 9 for a complete description of the syntax of frequency scalable systems used in the Test Model and core experiments.

D.9. RATE CONTROL AND QUANTIZATION CONTROL

Some Core Experiments require multi-layer rate and quantization control. There are two approaches: slice granular and macroblock granular. They are described below. For the higher resolution layers, each macroblock Mquant is the product of the multiplication factor and Mquant from the low resolution layer.

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 (Formula needs fixing word processing problem!)

$$T_i(l,m,h) = \max\left\{\frac{R(l,m,h)}{1+\frac{N_p X_p(l,m,h)}{X_i(l,m,h)K_p}+\frac{N_p X_b(l,m,h)}{X_i(l,m,h)K_b}}, \frac{br\%(l,m,h)*bit\_rate}{(8*picture\_rate)}\right\}$$

$$T_p(l,m,h) = \max\left\{\frac{R(l,m,h)}{N_p + \frac{N_b X_b(l,m,h) K_p}{X_p(l,m,h) K_b}}, \frac{br\%(l,m,h)*bit\_rate}{(8*picture\_rate)}\right\}$$

$$T_b(l,m,h) = \max\left\{\frac{R(l,m,h)}{N_b + \frac{N_p X_p(l,m,h) K_b}{X_b(l,m,h) K_p}}, \frac{br\%(l,m,h)*bit\_rate}{(8*picture\_rate)}\right\}$$

After coding a picture, $R(l,m,h) = R(l,m,h) - S_{ipb}(l,m,h)$.

Before encoding the first picture 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

$$dj_{ipb}(l,m,h) = dj0_{ipb}(l,m,h) + Bj\text{-}1(l,m,h) - \frac{T_{ipb}(l,m,h)*(j-1)}{MB\_cnt}$$

$$Qj(l,m,h) = \frac{dj(l,m,h) * 31}{br\%(l,m,h)*r}$$

Initial conditions:

$$d0_i(l,m,h) = \frac{10*br\%(l,m,h)*r}{31}$$

$$d0_p(l,m,h) = K_p * d0_i(l,m,h)$$

$$d0_b(l,m,h) = K_b * d0_i(l,m,h)$$

b) Slice granularity approach

$$ds_{ipb}(l,m,h) = ds0_{ipb}(l,m,h) + Bs\text{-}1(l,m,h) - \frac{T_{ipb}(l,m,h)*(s-1)}{Slc\_cnt}$$

$$Qs(l,m,h) = \frac{ds(l,m,h) * 31}{br\%(l,m,h)*r}$$

where Slc_cnt is the number of slices in a picture and s is the current slice index.

Initial conditions:

$$d0_i(l,m,h) = \frac{10*br\%(l,m,h)*r}{31}$$

$$d0_p(l,m,h) = K_p * d0_i(l,m,h)$$

$$d0_b(l,m,h) = K_b * d0_i(l,m,h)$$

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_actj 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: $M\_ratios(m,h) = \frac{Qs(m,h)}{Qs(l)}$

A 9 bit pattern (slice_quantizer_ratio)

16 8 4 2 1 0.5 0.25 0.125 0.0625

is used to represent M_ratios, e.g a slice_quantizer_ratio of 0.75 would be represented as 000001100. At the decoder the MB mquant is

$$mquant_j(m,h) = M\_ratios(m,h)*mquant_j(l).$$

D.10 Drift correction

In a simple single loop encoder- decoder approach for frequency scalability drift occurs in the lower layers with the decoding of the lower resolution video. This simple frequency scalable coding scheme has shown very good coding efficiency with good quality of the highest resolution reconstructed images. Postprocessing methods to reduce the drift at lower resolution images have been proposed and showed good performance depending on complexity of implementations. Note, that the postprocessing used at the decoder is not subject to standardisation. The postprocessing methods proposed are documented in MPEG'93/39, MPEG'93/121 and MPEG'92/466.

D.11 Interlaced quality on lower resolution layers

The drift-free encoding scheme outlined in Figure D.1.2 allows the application of any frequency or spatial domain decimation technique to provide sufficient interlaced input quality for the lower resolution layers and is not subject for standardisation. For services with high interlaced quality requirements on the lowest layers the decimation technique outlined in Chapter 3.5 of the Test Model may be applied.
For the drift encoding scheme in Figure D.1.1 improved interlaced quality on the lowest layer can be achieved by using pure field coding modes.(Figure 4.6 in the Test Model). This leads to an interlace quality which will be satisfactory for many applications at the expense of some reduction in the high layer coding efficiency due to the removal of the frame-based DCT mode. A more natural solution is to use an adaptive frame/field DCT approach (Section 6.5.6. of Test Model) and extract the appropriate DCT coefficients from the field-based or frame-based DCT blocks. The coding efficiency of the upper layer of the frequency scalable coder is improved at the cost of a small decrease in the quality of the lower layer service. The lower layer quality, however, remains adequate. Motion compensation is also performed on an adaptive field/frame basis.

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 bitstream 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 Bitstream specification

The coded bitstream 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.

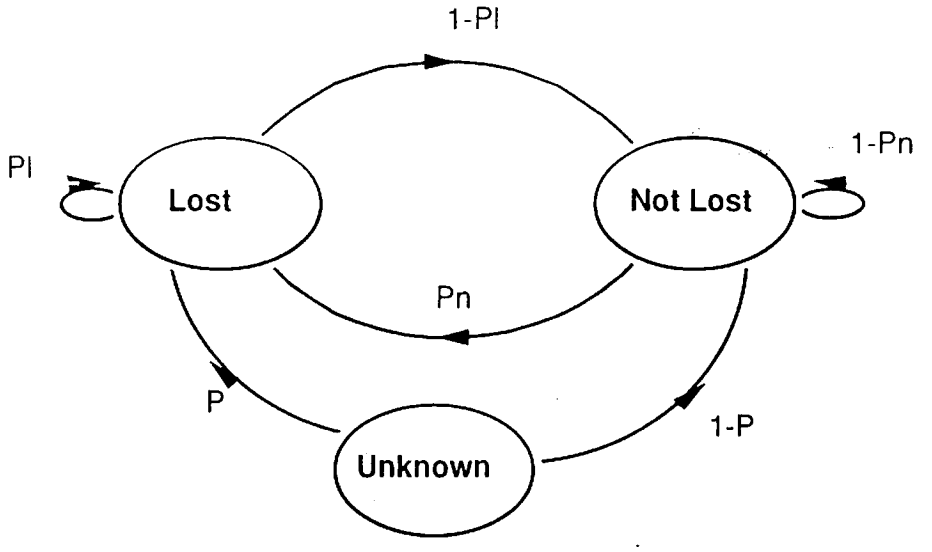
For a lost cell the cell is discarded.

F.1.2 Calculation of cell loss probabilities

This section outlines a method for determining whether any cell in a bitstream 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 1000 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 1000 times. If this is not done, the first few random numbers will be small, leading to the loss of the first cells in the bitstream.

With respect to the results of this model, it must be taken into account that this model has a limited accuracy, in particular, when the expected number of cells lost is less than 100.

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 DSM simulations, a burst length of 100 and cell loss rate of 10^{-5}should be used.

For all experiments the following table should be completed.

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

F.3 Concealment techniques

Three concealment methods are proposed.
1. Substitution of lost macro blocks with the co-located macro block of the previous frame.
2. Substitution of lost macro blocks with the motion compensated macro block of the previous frame if frametype is P or B, and co-located macroblock if frametype is I. The concealment motion vectors are obtained from the macro block above the lost macro block in the current frame.
3. As (2), and also intra macroblock motion vectors for concealment.

When layering is used the decoder will always first reconstruct the base layer (with partial data in case of Data Partitioning).

F.4 AC-Leaky Prediction

Organizations: AT&T, JVC, Sarnoff
Purpose: To verify the coding efficiency, visual quality and error resilience.

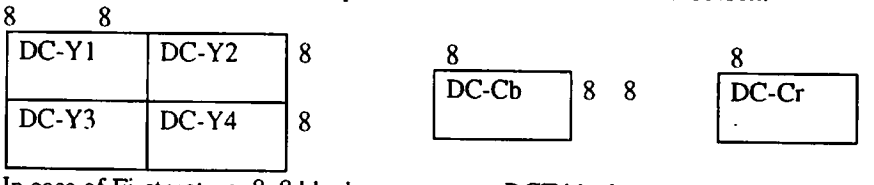
F.4.1 Description of AC Leak

1. Leaky prediction for AC component is used.
   (AC component is the residual after subtracting 8x8 mean.)
2. Only MC predictor is modified.
3. Apply to P-picture only, but not B-picture.

Procedure

Step 1. Calculation of DC

DC is the average value of 8x8 pixels, then 6 values are in Macroblock.



In case of Fi-structure; 8x8 blocks are same as DCT block.

In case of Fr-structure; 8x8 blocks are may not be same as DCT block. (If MC and DCT are coupled, it is same.)

Step 2. Subtraction of DC component

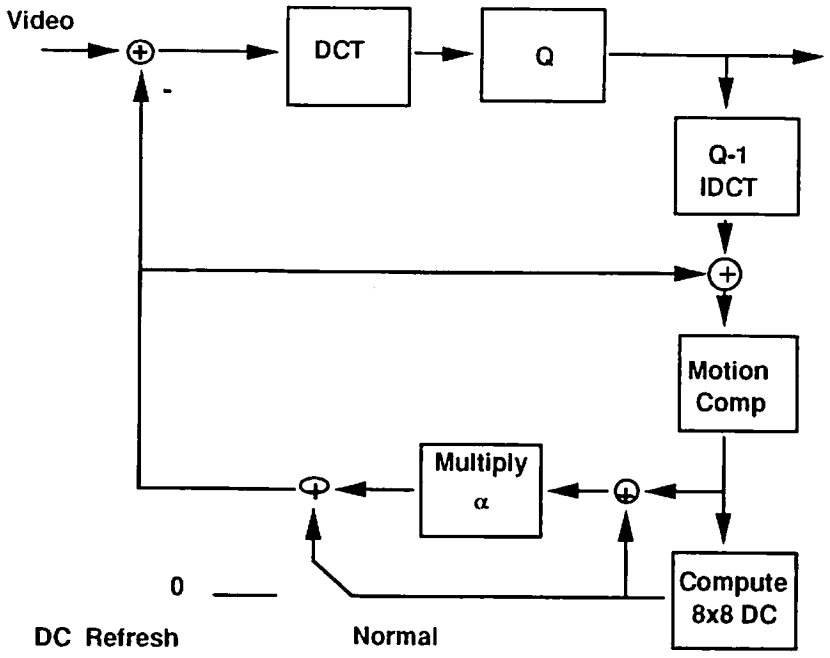
Subtract DC value of block from all pixels of that MC block. Get AC components.

Step 3. Multiplication of Leaky Factor

Multiply leaky factor (0.875 or 0.9375) to all AC components. Use shift and subtract.

Step 4. Addition of DC component

For normal picture, add DC component to AC components, obtaining the MC prediction. Decision of MB type is done before extraction of the DC component.



Encoder Block Diagram

For picture with DC Refresh

- Refresh of DC component is required, because DC component is non-leaky.
- Subtract DC value of block from all pixels of that MC block, but add 0 instead of the DC value to obtain the MC prediction. DC component of input signal remains in prediction residual.
- Quantize this DC component using the intra macroblock method.
- Encode quantized DC coefficients as in I-picture using the Differential DC tables.

With more frequent refresh (e.g., Ndc = 6), error resilience is better.

F.4.2 Core Experiments

The following parameters will be used for the experiments:

- Frame-structure with Fr/Fi or Field-structure
- 4Mbps
- Leaky Factor : 0.875 (M = 3) / 0.9375(M = 1)
- Ndc = 15 (12 in 50Hz)
- Ndc = 6 for faster refresh.
- Test sequences: Flower Garden, Mobile & Calendar

The following aspects will be tested:

- Coding performance (no losses)
- Cell loss concealment : compare with I-picture refresh and Intra MB motion vector method.

using motion compensated concealment in all cases. The cell losses will be generated individually for each bitstream using the same parameters.

- Demonstrate channel hopping using M=1, Ndc=6 for Flower Garden, and M=3, Ndc=15 for Mobile & Calendar.
- Scene change: Combine clips from two test sequences creating an artificial scene change, and compare the coding performance.

Note: If intra macroblocks are disabled, they should be disabled for all the experiments.

F.4.3 Syntax

The leaky factor for Leaky prediction "LF" is transmitted in the picture header of P-picture and is immediately followed by "dc_refresh_flag" using the following syntax.

```
if (picture_coding_type == 2) {
        leak_factor_code 3      uimsbf
        dc_refresh_flag  1      uimsbf
}
```

A leak_factor_code of 000 is not allowed, and a leak_factor_code of 111 means "without using leaky prediction (LF=1)". Otherwise, LF = 1-1/2^n where n = leak_factor_code. The value of LF is constant throughout the picture.

A dc-refresh-flag of 1 indicates DC-refresh picture and that of 0 is equivalent to non-DC-refresh. DC-refresh is applicable in the case of LF=1.

Reference ; MPEG 92 / 261, 442, 561, 93 / 166

F.5 Data Partitioning versus 1-layer cell loss experiment

Organizations: Sarnoff, Australian UVC Consortium

F.5.1 Introduction

The aim of this core experiment is to compare the performance of Data Partitioning for transmission as a method of spatial/temporal localisation of errors, as compared to normal Slice based transmission in TM.

F.5.2 Syntax and Semantics of Data Partitioning.

The syntax and semantics are described in Apendix L.

F.5.3 Experiment parameters

Because the prioritized transmission is particularly useful for high loss rates, experimenters are encouraged to use CLR=0.01. Two splits are proposed:
1. Use pbp=0 for I pictures, pbp=68 for P pictures, pbp=65 or 67 for B pictures.
2. Use a constant bitrate ratio for HP/LP partitions. Recommended ratios are 50/50 and 20/80. The breakpoints should be no lower than the first case (at least DC coefficients for I pictures, at least motion vectors for P pictures).

The bitrate ratios must be reported.

F.5.4 Concealment techniques

General guidelines in section F.3 should be observed. In addition, any incomplete data, such as motion vectors in P pictures, should be used for concealment.

APPENDIX G: COMPATIBILITY AND SPATIAL SCALABILITY

Scalability is achieved by spatial reduction in the pel and temporal domain. This is the only method that guarantees compatibility. The following section will describe in more detail the elements of spatial scalability/compatibility. See also Chapter J.

G.1 Spatial/Temporal Prediction

G.1.1 Introduction

Spatial scaling/compatible coding can be achieved by the use of layered coding. There is 'loose' coupling between the layers, that is , the coding algorithms used to code the layers can be chosen independently, but use is made of the reconstructed pictures produced by lower coding layers. The coding scheme used for the different layers can be chosen independently, as can the resolution for the layers, and the particular methods of up and down-sampling. Figure 1 below shows the method of 'loose' coupling.

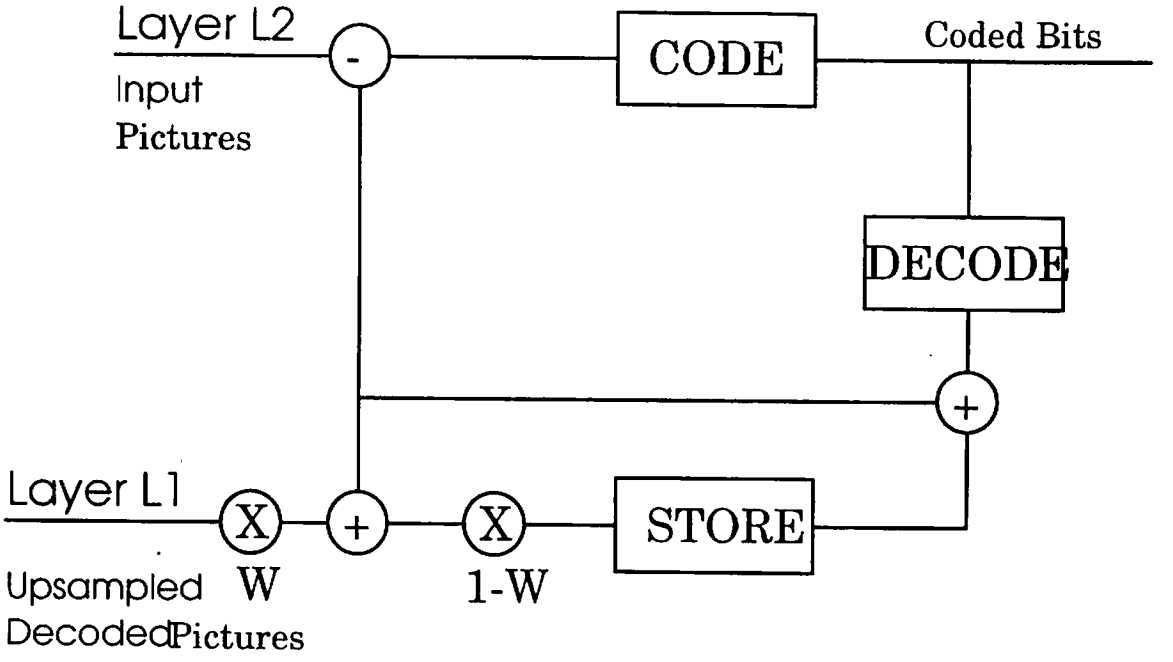


Figure G.1: Block Diagram of 'loose' coupling between layers

As depicted in figure G.1 coding layer L2 makes use of the reconstructed pictures produced by the next lower coding layer L1 by a method called 'spatio-temporal' weighting. This method performs a weighted combination of the temporal prediction from layer L2 and the up-converted spatial prediction from layer L1 on a macroblock basis. The weightings can be adapted and a choice from four can be made at any one time. This is signalled by **prediction_weight_code**.

For all the pels in a macroblock the corresponding weight is applied to the up-sampled spatial/compatible prediction and the chosen temporal prediction in the following manner:

```
for (pels in the macroblock)
        prediction[pel] = (W*compatible_prediction[pel] + (1-W)*normal_prediction[pel])
```

where the compatible_prediction[pel] is a pel from the upsampled layer 1 decoded picture, the normal_prediction[pel] is a pel from the prediction made from layer 2 (i.e. field/frame.. and half pel refinement], and prediction[pel] is the resulting final prediction used by layer 2.

G.1.2 Detail

The weighting is a pair of weights, one for each field. For each field in the macroblock the corresponding weight is applied to the up-sampled spatial/compatible prediction and the chosen temporal prediction in the following manner:

```
for (field 1 pels in the macroblock)
        prediction[pel] = (w1*compatible_prediction[pel] + (1-w1)*normal_prediction[pel])
for (field 2 pels in the macroblock)
        prediction[pel] = (w2*compatible_prediction[pel] + (1-w2)*normal_prediction[pel])
```

where w1 and w2 is the pair of weights identified by the **prediction_weight_code**. The weight codes are in table 1.

prediction_weight_code	Field 1 weight w1	Field 2 weight w2	compatible prediction
00	1	0	for Field 1 only
01	0	1	for Field 2 only
10	1	1	for Field 1 & Field 2
11	0.5	0.5	half compatible/half temporal for both fields

Table G.1: Prediction weight codes

[NOTE: There is an inconsistency with table G.1 and section 9.4, it should also be noted that these function should be made available for field-structure pictures]

There is one special case for the prediction_weight_code and that is in intra pictures. In the case of an intra picture there is no spatial/compatible weighted prediction. A macroblock in an intra picture is either pure intra or predicted from a lower layer (the weights in this case are "1"). Hence it is not necessary to transmit the prediction weight code.

Figure G.2 provides a more detailed block diagram of the compatible prediction method.

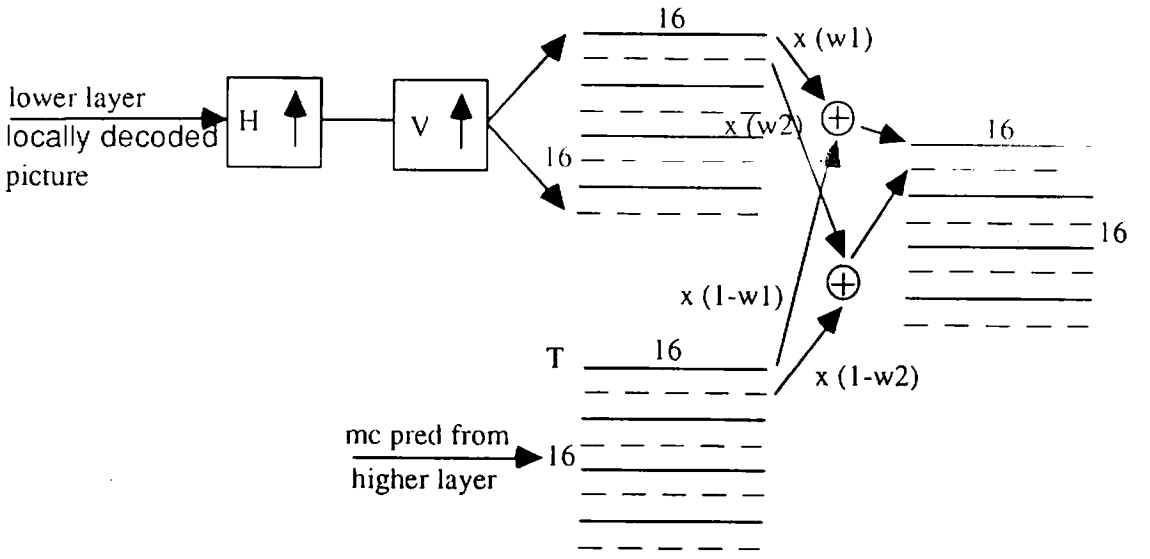


Figure G.2: Block diagram of compatible prediction method

G.2 Upsampling for Prediction

G.2.1 Introduction

The up-sampling for spatial scaling/compatible coding is performed on the locally decoded pictures of the lower layer. The up-sampling is different for the different picture formats in the higher and lower resolution layer.

G.2.2 Interlace to Interlace Up-sampling

The low resolution coded image should be padded with zeros to form a progressive grid (e.g 625 interlace to 625 progressive) and then filtered using the relevant two field aperture filter selected in table 2. The result is progressive fields at the same resolution. Vertical upsampling within each progressive field is performed using the filter in table 2.1

z (temporal)	y (vertical)	Filter for first field of frame i	Filter for second field of frame i
-1	-2	0	-a/2
-1	0	0	a
-1	+2	0	-a/2
0	-1	1/2	1/2
0	0	1	1
0	+1	1/2	1/2
+1	-2	-a/2	0
+1	0	a	0
+1	+2	-a/2	0

Table G.2 Vertical/Temporal upsampling filter. a=1/8

y (vertical)	Coefficient
-5	1
-3	-5
-1	20
0	32
+1	20
+3	-5
+5	1

/ 64

Table G.2.1 Vertical upsampling filter.

The horizontal upsampling filter is specified in table G.3. Then the picture can be re-interlaced to the output standard by the selection of the appropriate lines.

-12	0	140	256	140	0	-12

/256

Table G.3a Horizontal Interpolation Filter

0	0	1	2	1	0	0

/256

Table G.3b Horizontal Interpolation Filter

NOTE: Simplification of these filters is for further study, and might be relevant for future specification.

G.2.3 Progressive to Interlace Up-sampling

Assuming that the lower layer has coded odd fields only (field 1), derived by the method in section 3.3.2 the even field (field 2) is generated by line shifting the odd field.

For luminance, horizontal interpolation using the filter in table 4 followed by vertical interpolation using the filter in table 5.

-12	0	140	256	140	0	-12

/256

Table 4 Horizontal Interpolation Filter

-12	0	140	256	140	0	-12

/256

Table 5 Vertical Interpolation Filter

For chrominance, vertical interpolation using the filter in table 6 followed by horizontal interpolation using the filter in table 7 followed by vertical interpolation using the filter in table 8.

-12	0	140	256	140	0	-12

/256

Table 6 Vertical Interpolation Filter

-12	0	140	256	140	0	-12

/256

Table 7 Horizontal Interpolation Filter

-12	0	140	256	140	0	-12

/256

Table 8 Vertical Interpolation Filter

NOTE: Simplification of these filters is for further study, and might be relevant for future specification.

G.3 Coding of spatial scalable/compatible macroblocks

Spatial scalable/compatible macroblocks are coded as though they were non-intra macroblocks. That is to use the non-intra weighting matrix, non-intra quantizer and to signal CBP to indicate which blocks have coefficient data for them.

When the spatial scalable/compatible prediction is *fully spatial* (ie. no temporal prediction) then the motion vector predictors PMV are reset. Multiplexing of bitstreams is performed at the system level.

Macroblock Addressing and Skipped Macroblocks
In Intra pictures there are no skipped macroblocks. For compatibile macroblock_type a skipped macroblock indicates that the macroblock is using base layer prediction only, no coeficients are coded.

In predicted pictures 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. For compatible macroblock_type a skipped macroblock is not used.

In interpolated pictures, a macroblock is skipped if it has the same MTYPE or compatible_macroblock_type as the prior macroblock, its motion vectors are the same as the corresponding motion vectors in the prior macroblock, the compatible weight_codes are the same, all its quantized DCT coefficients are zero, and it is not the first or last macroblock in the slice.

Decision Tree and Selection Method
The decision tree is as follows:

- Selection of temporal prediction (frame, frame/field, special prediction mode) is based on decision methods documented in the relevant parts of the test model.
- Selection of spatial scalable/compatible prediction from the four weighting options. Note, this decision should be biased towards selection of all spatial prediction (code 10) because no motion vector information is then required in the higer layer. Optimum bias value to be determined.
- Selection of the best mode from the above two decisions based on lowest MSE (again biased towards possible all spatial prediction as for the frame/field motion compensation).
- Intra/Inter selection based on decision methods documented in the relevant parts of the test model.

G.3.1 Summary of previous scalability results

1) Good performance achieved using interframe upconversion derived from 3-field upconverter.
2) Only choice from 4 weight codes required.
3) Use of intra quantisation in spatially predicted blocks in I pictures not particularly beneficial

G.4 SNR Scalability

SNR scalability means compatibility between two layers having the same resolution. The difference of bitrate between the two layers is then used to provide a better coding quality for the upper layer, that is a better SNR. The lower layer is coded using the "MPEG2 basic mode", while the upper layer is coded using the SNR scalable option within the spatial scalable option. The intention of SNR scalability is to provide a mechanism for transmission of a service with two different picture quality levels.

Within spatial scalability context, SNR scalability can be achieved by two different quantization refinement structures. The first outlined in figure G.4.1 uses a quantizer refinement technique that has feedback into the prediction loop the second in figure G.4.2 has no feedback into the prediction loop.

Quantizer Refinement with Feedback

The encoder is a two layer encoder, the lower layer generating a pure "basic MPEG-2" bitstream. For the lower layer encoder, the Motion Compensation is performed from the decoded picture of the upper layer. Thus, a drift between Motion compensation at the encoder side and motion compensation at the decoder side (lower layer) appears. The drift is kept very acceptable at least with value of N up to 15 and M=3. The upper layer encoder simply encodes the residual prediction error of the lower layer encoder, and feeds back into the motion compensation loop.

For the decoder the lower Layer decoder is a pure "basic MPEG2" decoder. The upper Layer decoder is a pure "basic MPEG2" decoder PLUS a VLD and inverse quantization for refinement of DCT coefficients.

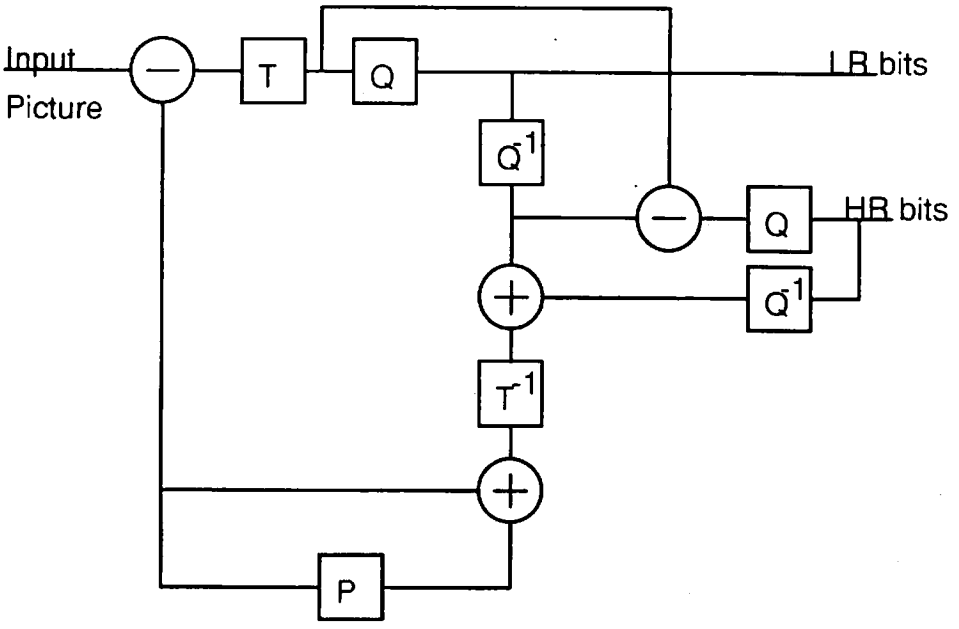


Figure G.4.1 SNR scalable encoder by quantizer refinement and feedback into the prediction loop

Quantizer Refinement with no Feedback

The encoder is a two layer encoder, the lower layer generating a pure "basic MPEG-2" bitstream. The upper layer encoder simply encodes the residual prediction error of the lower layer encoder.

For the decoder the lower Layer decoder is a pure "basic MPEG-2" decoder. The upper Layer decoder is a pure "basic MPEG-2" decoder PLUS a VLD, inverse quantization and inverse transform for refinement of the lower resolution picture.

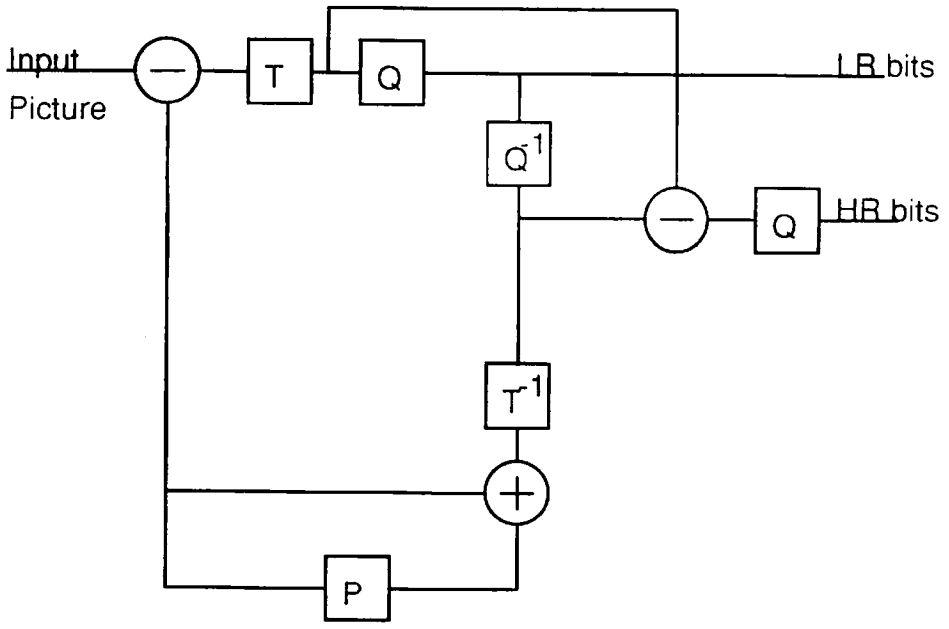


Figure G.4.2 SNR scalable encoder by quantizer refinement

Coding for quantisation refinement and feed-back :
The lower layer encoder works as a stand-alone MPEG2 encoder in terms of decisions, adaptive quantization, bit rate regulation etc. The only difference with a pure stand-alone encoder is that the lower layer encoder does not use the right prediction, i.e. it does not use the prediction corresponding to the one available by its decoder. Skipped Macroblocks are permitted for this layer, since the bitrate is rather low.

In the upper layer some Macroblocks can be skipped by the upper layer encoder. Then the same prediction error MB is decoded by both Low Quality and High quality decoders. For other Macroblocks the upper layer bitstream contains:
INTRA MB:

- No DC values for intra MB, possible CBP and DCT type, possible MQUANT.

INTER MB:

- Motion vectors are not transmitted since they are already available from the lower layer.
- If the Macroblock is not coded or skipped in the lower layer but coded in the upper layer, then CBP and DCT type has to be sent in upper layer.
- In other case the macroblock_type and motion vector information will be obtained from the lower layer. CBP may (whenever it saves bits) or may not be transmitted depending on the MB type.

The bitstream contains a MBtype, MB adress increment, mquant if any, CBP and DCT type if any, DC coefficient for inter MB, AC coefficients and EOB.

The MBtype codeword can signal the CBP to be transmitted and in these cases the DCT type is also transmitted.

When no CBP is transmitted for the upper layer the end of block codeword is always transmitted for each block (i.e. all blocks are coded). The "first coefficient trick" in the VLC table is thus not used.

When the CBP is transmitted the "first coefficient trick" in the VLC table is used as normally.

The MBtype codeword can signal the "MQUANT" (for adaptive quantization) to be transmitted by the

upper layer. This is done independently from the lower layer: since the quantization parameter for the upper layer is different from the one of the lower layer, and since a clipping is performed for the MQUANT calculation there is no reason why MQUANT decisions would correspond for both layers. (see formula for adaptive quantization in Rate Control and Adaptive quantization section of the TM: mquantj = Qj * N_actj followed by a clipping to the range [1...3

Improvement of the SNR encoder

- Skipped macroblocks are used for both layers. They are very usefull for the upper layer, since many macroblocks are not refined by this layer.
- Steeper weighting matrices are believed to be effective to improve the coding efficiency of the upper-layer (see doc. 93/375). However, this matrix should not be that steep that it degardes the inter-field motion rendition. The optimisation of such matrices is for further study.

G.4.1 Experiment on SNR-scalability

a) Targets of the experiment:

* optimise a coding scheme for SNR_scalability; for instance with respect to weighting matrices for the lower layer.
* evaluate the efficiency loss due to such a scalable coding by comparison against standalone coding (using the same coding parameters)
* another target is to compare with a data prioritisation approach, especially concerning the lower quality pictures.
* test SNR_scalability in respect to cell losses and bit errors as specified in appendix F.
* syntax cleaning is also to be considered in the frame of spatial scalability general syntax and also the commonalities between SNR and frequency scalability syntaxes are to be indentified.

b) Guidelines for Experiments:

Number of layers	Bit rate Layer 1 (LQ layer)	Bit rate Layer 2 (additional bit rate for HQ)
2	2.5Mbit/s	1.5 Mbit/s
2	9 Mbit/s *	4 Mbit/s *

Comparisons with standalone coding at the relevant bitrates (2.5, 5, 9 Mbit/s) will also be provided
Coding parameters and standards:
60 Hz: M = 3, N = 15
50 Hz: M = 3, N = 12.
Picture type: frame pictures only.
DCT and MC types: Frame/field DCT, Frame/field MC only (no special prediction mode for this experiment).
Sequences: Mobile and calendar, Flower Garden, Table Tennis, Football, Cheerleaders and Bicycle.
The three last sequences are of interest for a comparison with a "data prioritisation" approach, since complex\motions can be expected to introduce degradations for the LQ picture quality.

c) Optimisation of the SNR_scalable scheme
Weighting matrices for the Lower Layer:
A downloaded steep weighting matrix for Intra and Inter Macroblocks of the lower layer provide some benefits for the picture quality of both layers. For the lower layer, a very low bitrate is better accomodated by a steep weighting matrix. In addition, since high frequency coefficients are more often quantised to 0 in the lower layer, they are not transmitted twice (lower and upper layer) but only in the upper layer. Consequently, a gain in coding efficiency is achieved for the upper layer.
For the experiment, a steep weighting matrix will be used for the lower layer, while the by default

matrices will be used for the upper layer.
The matrix coefficients will be provided to experimenters by E-mail. Requests can be made to: nocture@lep.lep-philips.fr (Gilles Nocture, fax: + 33 1 4510 6960).

a) Guidelines for Experiments:

Number of layers	Bit rate Layer 1 (Lower layer)	Additional Bit rate Layer 2 (Higher layer)	
2	1.5Mbit/s	2.5 Mbit/s	SIF to 601
2	1.5 Mbit/s	2.5 Mbit/s	ISIF to 601

Coding parameters and standards:
60 Hz: M = 3, N = 15
50 Hz: M = 3, N = 12.
Picture type: frame or field pictures only.
DCT and MC types: Frame/field DCT, Frame/field MC only (no special prediction mode for this experiment).
Sequences TV: Flower Garden, Table Tennis, Football, Cheerleaders and Bicycle.

G.6 Experiments

1. Optimisation/Minimisation of upsampling filter . e.g.
2. For interlace-interlace..... minimise vertical extent of filter, examine intra slice upconversion filters
3. For progressive-interlace.. As for interlace-interlace
4. Optimisation macroblock layer syntax...e.g. only signalling changes in weight_code_mode
5. Comparison of different possible spatial prediction modes (dual-loop and single loop decoder modes) with eqquivalent frequency scalable approach. For description of spatial prediction modes see appendix J
6. comparison of spatial-temporal and spatial only
7. quantizer strategy for upper layer
8. upconversion

G.7 Chrominance scalability

G.7.1.Introduction

In this experiment, the spatial scalability of chroma formats and its simulation condition are described.This method is very usefull for the compatibility between contribution-quality and distribution-quality.

Additionally the picture format in core profile might be decided 4:2:0, however the hooks for the backward compatibility from the high quality format are necessary for the future.

G.7.2.Spatial scalable coding for Chroma formats

The block diagram of the spatial scalable coding for Chroma formats is specified in Fig.1. This idea is very simple, so this method is a kind of application of the spatial scalable coding. The conventional spatial scalable prediction is applied to the chroma signal only.

The input signal to the lower layer is formed into 4:2:0 format, which is down-sampled from the high-

quality 4:2:2 (4:4:4) format. The encoding process of the lower layer is the same as the normal MPEG-2 encoder. The 4:2:0 chroma signal which is decoded in the lower layer is provided to the upper layer encoder. At this time, the 4:2:0 decoded chroma signal is up-sampled by the spatial filter to adapt to the corresponding format in the upper layer. This up-sampled chroma signal is used for the prediction in the upper layer with the motion-compensated signal in the upper layer. The optimum prediction signal is selected from the spatial prediction and the temporal prediction.

This method is a kind of extension of the conventional spatial salable coding.

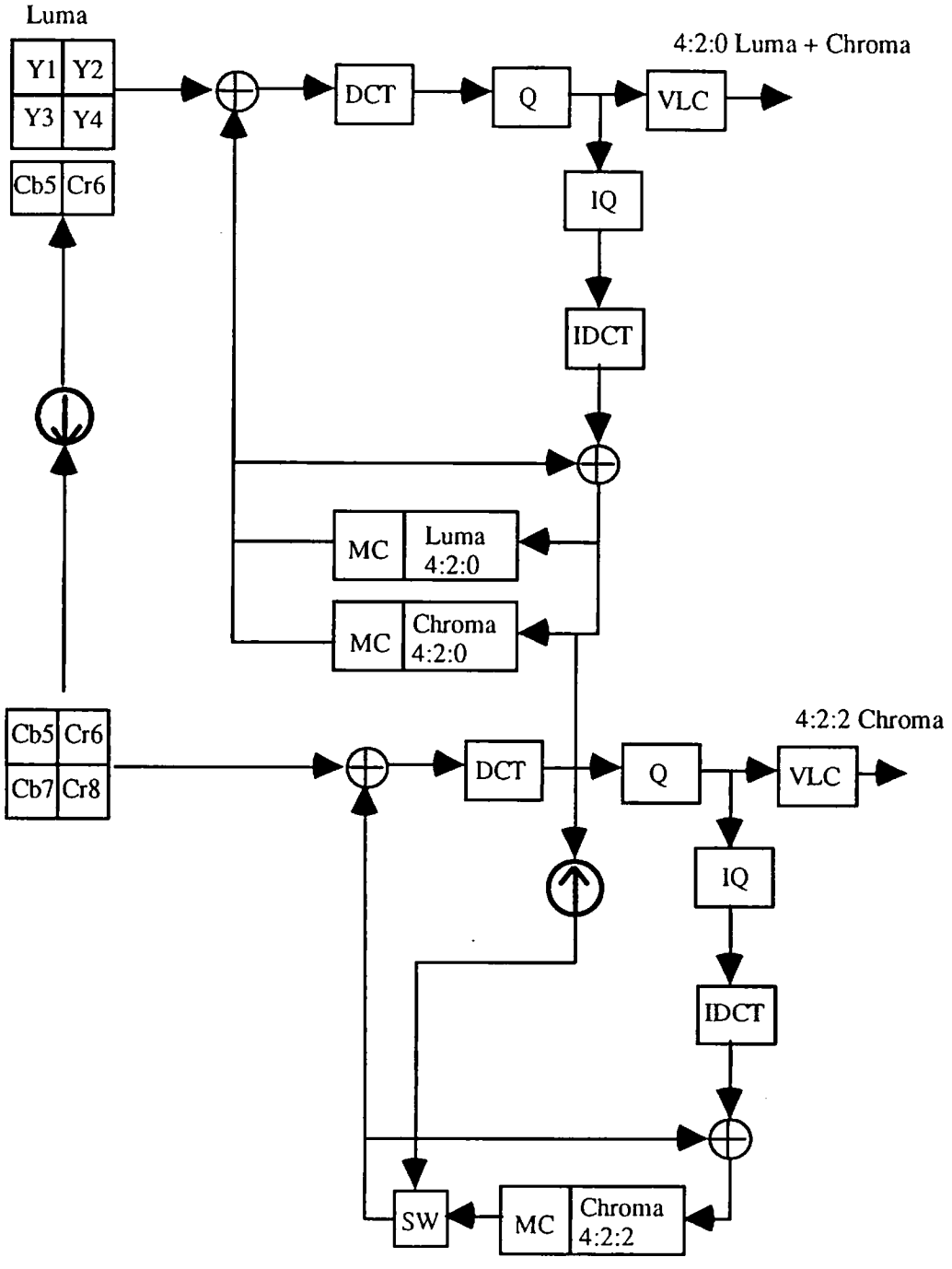


Fig. 1 Spatial Scalable Coding for Chroma Formats

G.7.3.simulation

[NOTE: if any are necessary]

Simulation condition

Picture format	:	Lower - 4:2:0
	:	Upper - 4:2:2 chroma
Rate Control	:	TM Rate Control with STEP3
Sequence	:	Mobile&Calender, Fountain, Popple
Bitrate Lower(4:2:0)	:	4Mbps, 9Mbps
Upper(4:2:2 Chroma):		2Mbps, ?Mbps
N(GOP)	:	15/12
M	:	M=3(bblbbpbbpbbpbbI...)
Number of frames	:	60

Other aspects are the same as the Spatial Scalable coding described in Appendix G.

SNR of 4:2:0 format is computed from the signal converted to 4:2:2 format. All SNR reference of the chroma signal is the original 4:2:2 chroma.

In the subjective test, the horizontal edges of the chroma signal will be improved. Especially in the red parts, the improvement of the quality will be detected easily.

G.8 Interlace to progressive compatibility

A near term possible solution to the problem of interlace/progressive interoperability may be 2-layer scalability. This is shown in the following diagrams that illustrate format scalable video coding. First, the principle of progressive to interlace is shown. Then interlace to progressive conversion is shown.

Format Scalable Codecs for Progressive Source

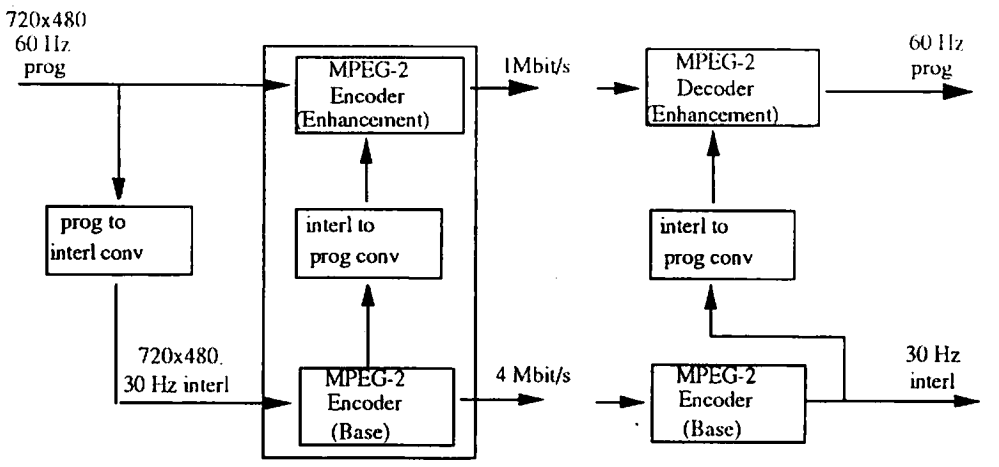


Fig 3(a) Interl base-layer and prog enhancement by snr scalability

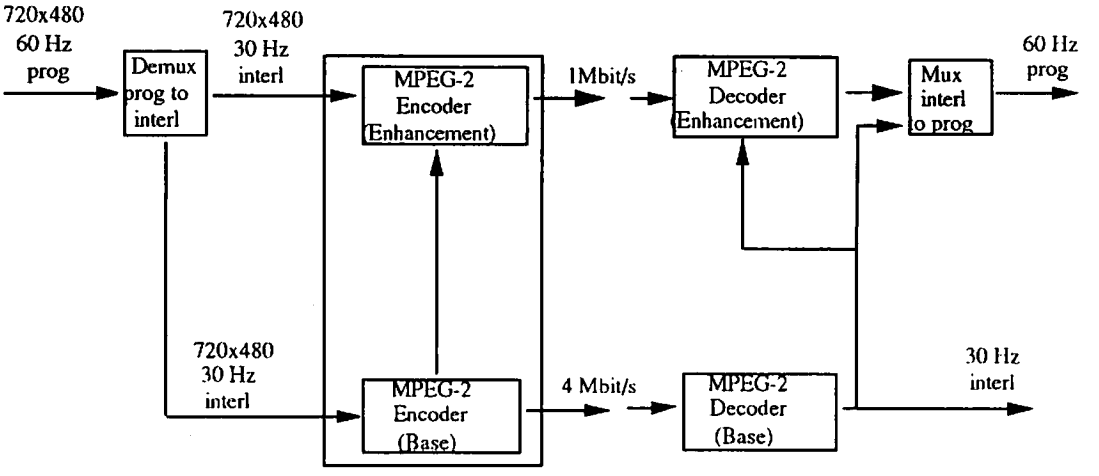


Fig 3(b) Interl base-layer and interl enhancement by temporal scalability

APPENDIX H: LOW DELAY CODING

H.1 Simulation guidelines for low delay c ıg.

H.1.1 Coding structure.

In the low delay coding 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. For low delay mode, forced intra slice/column is recommended for updating. With these updating method, it is assumed that the buffer delay may be 2fp.
- Delay resulting from frame/field display. This will be called basic delay.

To achive a maximum delay (maximum + buffer) of less then 150ms, the only realistic modes are frame and field structure with only I and P-pictures.

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

A major contribution to the buffer delay is 'large' pictures resulting from e.g. scene cuts. To get around this delay it is necessary for the low delay coding to have the possibility of picture skipping. The encoder may decide to skip frames after a "large" picture due to scene cut.

For the prediction of pictures following skipped ones, the reference is made to the re-constructed "large" picture.

VBV specification for skipped picture can be found in Annex. C.

H.1.3 How to handle the first picture using forced intra slice

In the low delay coding 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, and this picture is treated as P-picture in term of rate control.
- In evaluating the delay only buffer filling after the first picture is considered.

Note) In case of using intra picture, the first picture of the test sequence is coded as NORMAL Intra Picture, and its rate control follows NORMAL rate control described in TM.

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.

- Frame mode, M=1: Two slices pr. frame.
- Field mode, M=1: One slice every other field.

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.

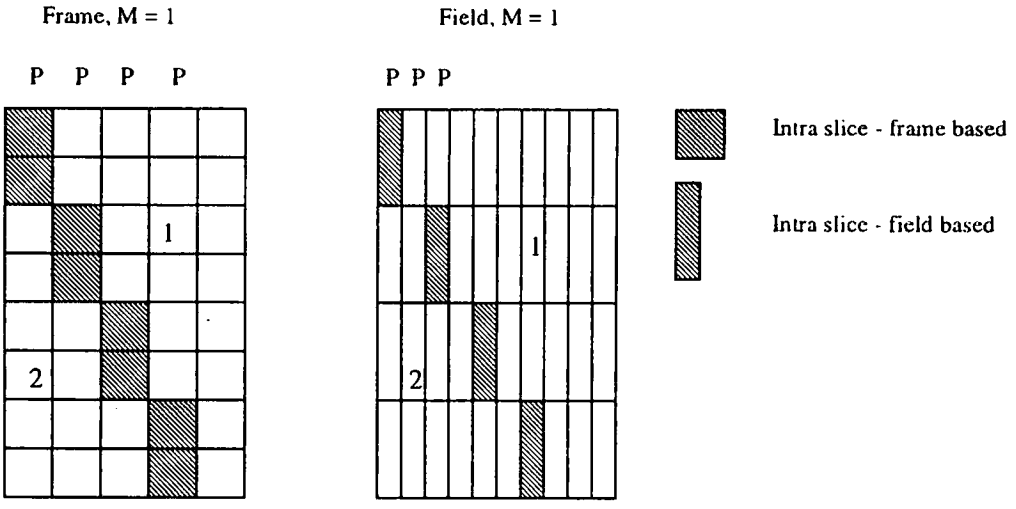


Figure H.1 Low Delay Coding stategy

H.1.5 Rate control.

INTRA SLICES

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

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

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

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

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 :

$$T_{p_p} = \frac{Number\_of\_P\_slices * Tp}{Number\_of\_P\_slices * Tp + Number\_of\_I\_slices * Ti} * T_p \quad (3)$$

$$T_{p_i} = \frac{Number\_of\_I\_slices * Ti}{Number\_of\_P\_slices * Tp + Number\_of\_I\_slices * Ti} * T_p \quad (4)$$

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

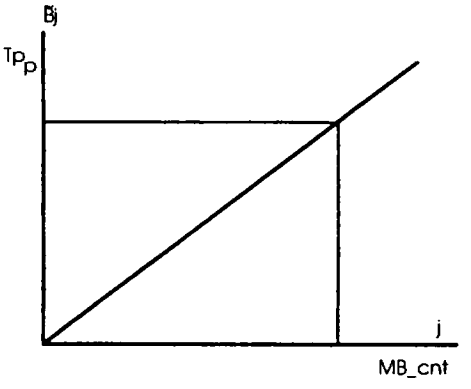


Figure H.2

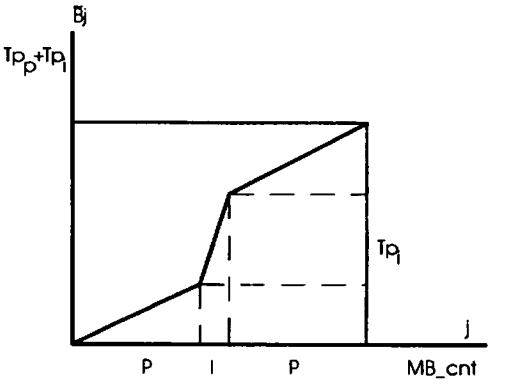


Figure H.3

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.
- The number of bits generated in the skipped picture is forced to set to zero.
- The Step 1 of rate control in TM is active even in the skipped pictures.
- The Step2 and Step3 of rate control are active only in the coded pictures.

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 coding. Forced INTRA slices could be replaced by leaky prediction. Possible advantages could be reduced buffer delay and no visible INTRA updating.

APPENDIX I : FREQUENCY DOMAIN SCALABILITY CORE EXPERIMENTS

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

Syntax extensions needed for this core experiment are now available in Section 9.

I.4.4. Coding

The semantics for this core experiment can now be found in the relevent subsections in Appendix D.

I.12 Single loop decoder structure

Purpose: To converge on the optimal single-decoder-loop structure.
To achieve a maximum lower layer quality the input signal for that layer is now being obtained the same way as defined for the interlace-interlace spatial scalable core experiment G.2.
Furthermore there is evidence that the upper layer quality will be maximised if there are no constraints put on the TM prediction and coding modes. Therefore all of them will be allowed in this revision of the Core Experiment, apart from Frame MC in those cases where a field flip occurs after scaling the vector for use in a lower layer. (This problem is already being reported to be present for chrominance vectors in the single layer coder. Since it has a bigger impact in the scalable coder, in those MBs where a field-flip occurs, the Frame MC vector should be replaced by the best available Field MC or SVMC same-field vector .)
Experiment parameters:

- 3 layers (scales 2, 4, 8)
- all DCT and MC modes enabled (except Frame MC if Field-Flip occurs)
- prediction DCT coefficients: NxN of lowest frequency (of Field or Frame DCT block)
- full-precision motion vectors on all layers
- N=12/15, M=3

- extended quantization range recommended by quantization Ad Hoc group
- spatio-/temporally decimated lower layer input signals (same as for G.2)
- bit rates:
  - 4.00/----/---- (Unconstrained lower layers)
  - 4.00/2.75/1.50
  - 4.00/1.75/0.75

Parameters under study:

- scalable side information
  - new macroblocks
  - refined motion vectors
  - refined macroblock types
- coefficient scanning and coding
  - zz on all layers vs layer-sequential scan

Coefficient scanning and coding will be evaluated in the first part of this experiment. The scale-sequential scan under consideration is given in Figure I.12.1. Afterwards the benefit of scalable side information will be evaluated under all three bit rate conditions.

```
 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
```

Figure I.12.1: Scan patterns for layer-sequential scanning.

For two layer interlace applications a further issue might be a comparison of the proposed extraction method for coefficient prediction to the one used before (indicated in Figure I.12.2). For this purpose only Frame DCT shall be used.

```
X X X X * * * *
X X X X * * * *
X X X X * * * *
* * * * * * * *
* * * * * * * *
* * * * * * * *
* * * * * * * *
X X X X * * * *
```

Figure I.12.2 DCT coefficient subset for interlace extraction.

I.13 Comparison of Data Partitioning vs. Simple Frequency Scalability Schemes

So far, two simple schemes namely data partitioning and single loop frequency scalable coding scheme have been identified to provide layering in frequency domain. Both schemes have shown promising results for cell loss concealment. Frequency Scalability, in addition, has been discussed for a variety of applications, like resolution and SNR scalability, fast forward/fast reverse trick modes etc..
In these Core Experiments, we intend to make a comparison between the two schemes both in terms of complexity and coding efficiency.

Experiment A)

For a total bit rate of 4 Mbps, use data partitioning to generate scale 2 and scale 8 layers. Repeat the same step for single loop encoder frequency scalable system with no rate control constraint on the lower scale (i.e. scale 2).
Compare the results in terms of SNR and bit rate at scale 2 and overall average SNR at scale 8.

Experiment B)

For a total bit rate of 4 Mbps and a given high vs. low priority output cell rate (e.g. %25 high vs. %75 low) compare the coding efficiency of data partitioning vs. single loop encoder frequency scalable system.

Experiment C)

Repeat Experiment B in the presence of random cell loss in an ATM channel. To generate random cell loss use the model described in
Appendix F of TM4 with the assumption that no high priority channels will be lost.

For the above experiments, all prediction modes outlined in MPEG2 Working Draft are to be considered. In addition it is desirable to investigate the layered sequential scan in combination with data partitioning. For frequency scalability implementations both subband and pyramid schemes should be considered.

I.14 Comparison of different encoder filters for inter-layer prediction.

I.14.1. Purpose

To compare the performance of different scale_2 and scale_4 prediction filters in a multiloop frequency scalable coder.

I.14.2. Experiment Details

Compare the PSNR of each layer, scale_2, scale_4 and scale_8 using different proposed prediction filters.

Experiment parameters:

- 3 layers (scale 2, 4 and 8)
- DCT mode shall be frame or field mode only.
- Full precision motion vectors on all layers.
- Slice granularity rate control is employed when the bit rate of scale_2 and scale_4 is constrained.
- N=12/15, M=3 and 1.
- bit rates:
  - 4.00/---/--- Mb/s(Unconstrained lower layers)
  - 4.00/2.75/1.5 Mb/s
- The scale_4 and scale_2 original sequences used in PSNR calculations are generated by DCT based extraction.
  - scale_4_orig DCT8X8----IDCT4X4 * 0.5
  - scale_2_orig DCT8X8----IDCT2X2 * 0.25
- no scalable side information
- MC_filters
  - 00 only 1/2 pel precision for each layer
  - 01 1/4 and 1/8 pel precision as defined in TM section D.3
  - 10 MPEG93/120 note: scale_2 filters can be supplied by email.

11 MPEG93/039 note: Pixels are replicated at the frame boundary.

- Test Sequences

Mobile and Calendar	Frame mode
Flower Garden	Frame mode
Cheer leaders	Field mode
Bus	Field mode

I.14.4 Syntax extensions

The frequency scalable syntax takes the form described in section 9 of TM 5. The extensions to this syntax for the purpose of this experiment follow.

```
sequence_frequency_extension () {
  .
  .
  .
  motion_compensation_loop                          1      uimsbf
  if (motion_compensation_loop == 1)
    mc_flag                                         2      uimsbf
  .
  .
  .
}
```

mc_flag	00	only 1/2 pel precision for each layer
	01	1/4 and 1/8 pel precision as defined in TM section D.3
	10	MPEG93/120 note: scale_2 filters can be supplied by email.
	11	MPEG93/039 note: Pixel are replicated

APPENDIX J: HARMONISED HYBRID SCALABILITY

J.1 Introduction

In the current stage of the MPEG/Experts group work the focus goes out to the main profile and its main level. The main profile is required to have error resilience, but scalability and compatibility are considered to fall in other profiles.

During the course of MPEG many proposals have been made which combine some of the **features of: low complexity, error resilience, image quality, scalability and compatibility.** Till so far no effort has been made to combine all proposals into one concise generic solution.

In this contribution a proposal will be made which combines most of the ideas, which were on the table at some time, into one concise generic solution.

J.2 Examples of the proposals currently on the table

The first example would be the most 'simple' decoder system currently proposed by Sarnoff and Thomson, which will provide error resilience. Data partitioning is used to achieve error resilience and high frequency coefficients (along the zigzag scan path) are merged with low frequency coefficients. In principle this merge can be seen as a addition, this is depicted in figure J.1.

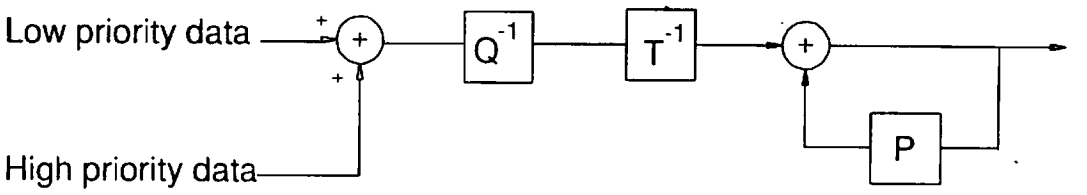


Figure J.1: Functional equivalent of the data partitioning decoder block diagram

A second scheme is using a sort of data partitioning, always taking the 4x4 low frequency components for a high priority channel to provide error resilience and scalability. The low frequency coefficients may be recoded again in the second layer. This is the scheme being proposed by the Australian consortium and is shown in figure J.2.

The third scheme, using SNR scalability to achieve error resilience, has been proposed by Philips LEP and CCETT, and is also depicted in figure J.2.

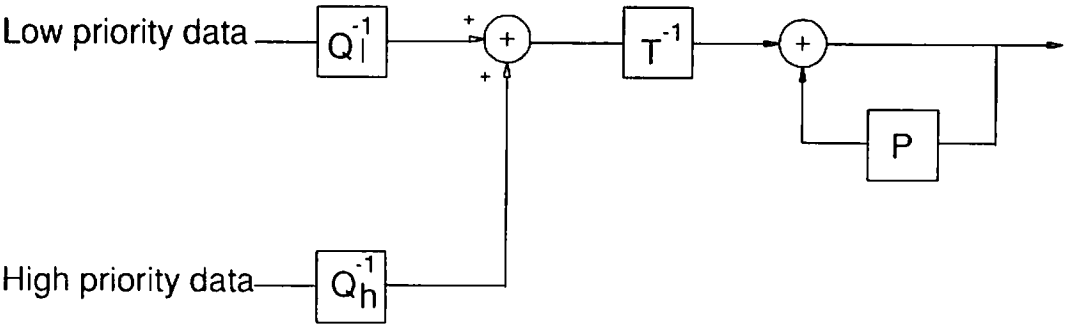


Figure J.2: Functional equivalent of the SNR scalable and frequency scalable decoder block diagram

It should be noted that figure 1 and figure 2 are identical, when one defines the Ql and Qh in figure 2 to be the same.

J.3 The concise generic decoder solution

It is even possible to shift the addition to after the inverse DCT and one could come to a figure depicted in figure J.3, which is still equivalent to figure 1 and figure 2 and has all functionality's as comes the functionality for the schemes from figure J.1 and figure J.2.

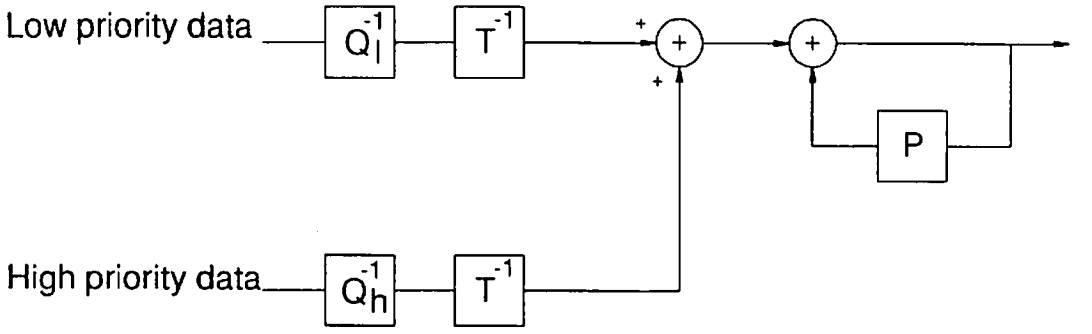


Figure J.3: Functional equivalent of figure 1 and figure 2

It should be noted that in Figures J.1,J.2 and J.3, the inverse transforms in both data streams have to operate at the same pixel rate. It also should also be noted that it is not possible to support an optimally down-converted image with these systems and therefore does not have compatibility and inter working with smaller coded images e.g. MPEG1, H.261 or even MPEG2.

What one can do to solve this problem is to generalise the model of Figure J.3 and allow the possibility of an up conversion in the system as shown in Figure 4. The system one then would get is only slightly more complex then the system depicted in figure 1 and 2, but will offer much more functionality then those systems. It will support all functionality of the data-partitioning system (error resilience), the SNR scalable system (watchable high priority data images) and the frequency scalable system ('simple' 'software' decoding of the high priority data images). **But more is offered,** this scheme does also provide in addition to all these features the possibility of being compatible (see figure J.4).

One form of decoder diagram is depicted in figure J.4a.

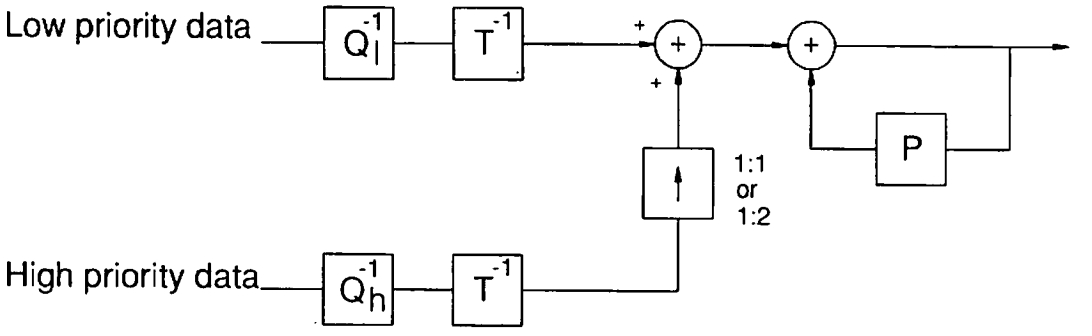


Figure J.4a: Harmonised scalable decoder

Note: the up converter supports 1:1 (no up conversion) as wells as 1:2 up conversion. It should be noted that the adder after the Inverse Transform can be shifted inside the loop, and can be combined with the Temporal-Spatial scalable weighter.

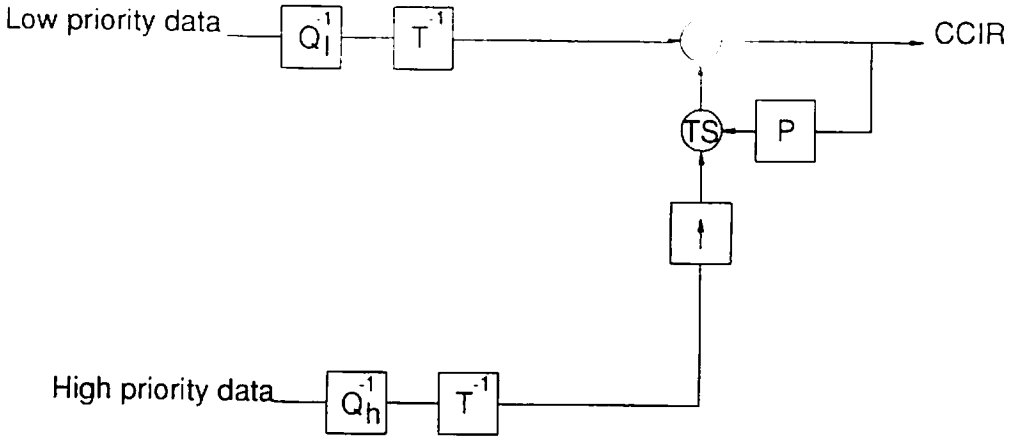


Figure J.4b: Equivalent of Harmonised scalable decoder

The above proposal can be combined with the current TM spatial scalable solution in one block diagram showing the two options. This is shown in Figure J.5.

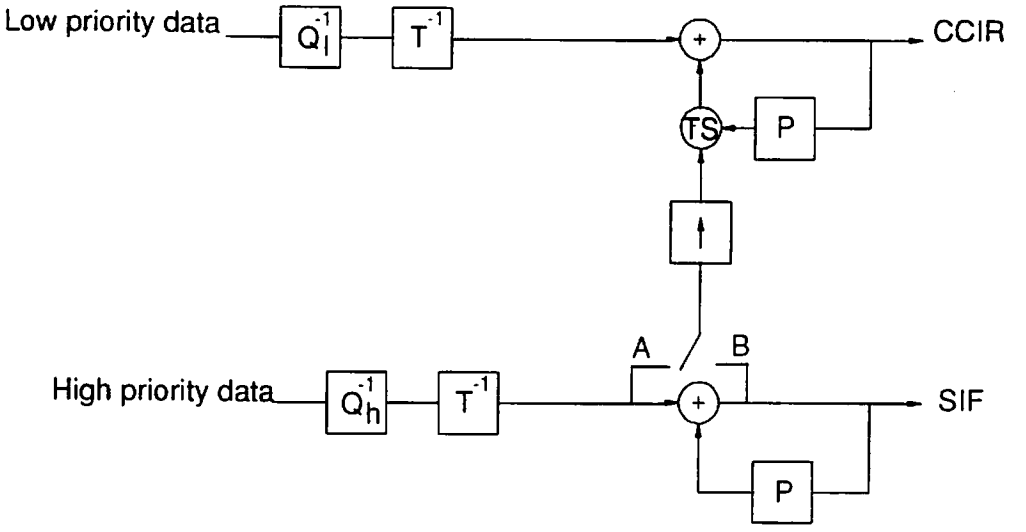


Figure J.5: Harmonised Hybrid decoder structure

TS - Temporal-Spatial weighting switch

Data paths A and B are not used concurrently. **It is proposed to signal in the sequence header which data path is used.** Data path A is selected for low complexity, error resilient decoders, data path B is selected for full scalable, compatible, higher image quality decoders.

NOTE: The up converters can be switched off (1:1 up conversion). Data path A's up converter for a 1:2 up conversion is a simple linear (1 2 1) filter inside a block for progressive to interlace (the temporal miss aligned field should not be predicted). Data path B's up converter for a 1:2 up conversion is a simple 1 2 1 filter inside a block for progressive to interlace.

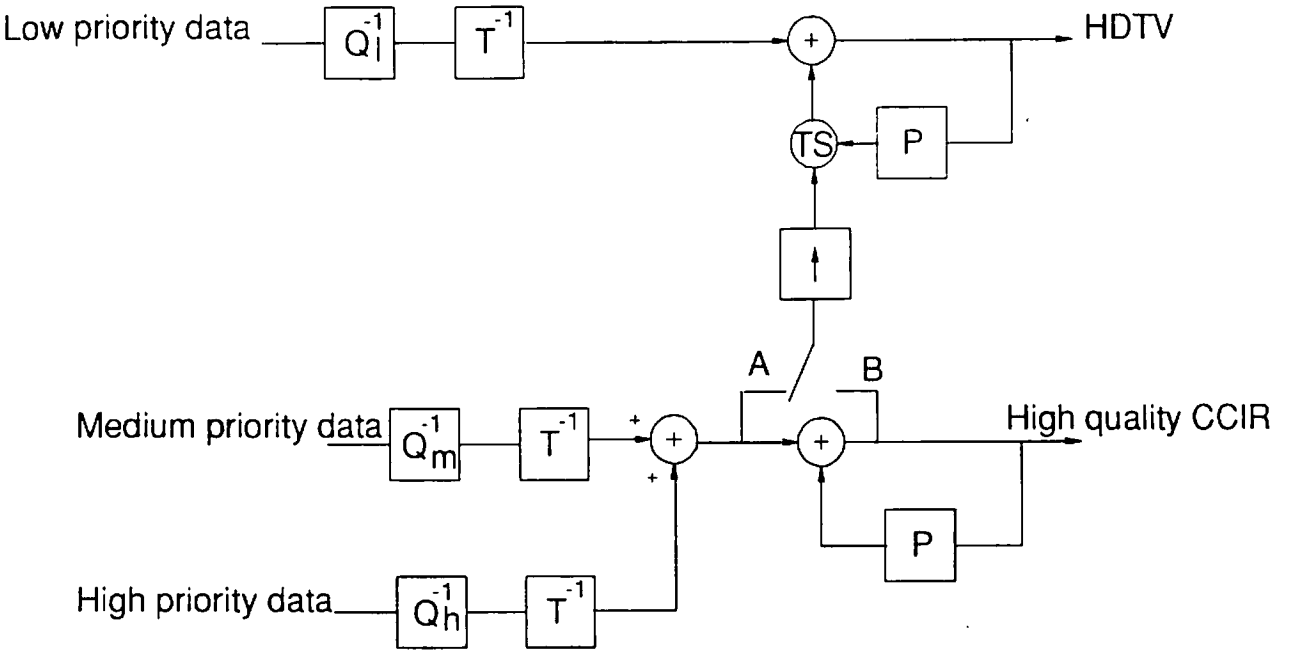


Figure J.6: Full three level HDTV decoder structure

TS - Temporal-Spatial weighting switch

Data paths A and B are not used concurrently. **It is proposed to signal in the sequence header which data path is used.** Data path A is selected for low complexity, error resilient decoders, data path B is selected for full scalable, compatible, higher image quality decoders.

J.4 Decoder conclusion

The proposed structure has merged the Thomson-Sarnoff error resilient system, the Philips/CCETT error resilient/SNR scalable system, the Australian low complexity/error resilient system, the BT/AT&T compatible system and the PTT low complexity/compatible system. In principle no changes to the syntax on the macroblock layer have to made. Signalling of the exact configuration of the encoder for will happen at the sequence layer.

J.5 The encoder

The main profile, main level encoder could be configured in many different ways, partly depending on the implementation of the targeted (subset of) main profile, main level decoder. A possible implementation of the MAIN encoder is depicted in figure J.6.

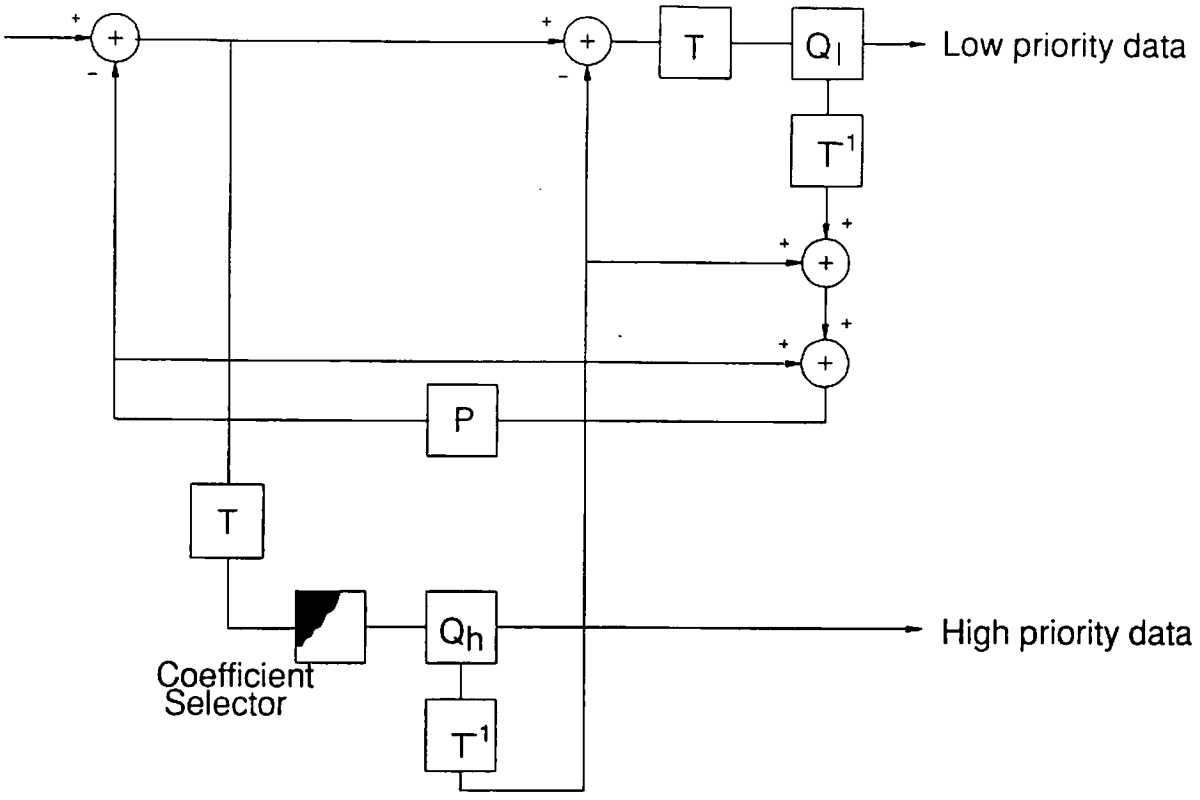


Figure J.7: Possible logic block diagram of encoder encapsulating 3 coding schemes

Figure 6 encapsulates 3 coding schemes currently on the table:

1. Sarnoff-Thomson's error resilient scheme, Incas Qh and Ql are identical: coefficient coded in the high priority data stream will generate zero's coefficients in the low priority data stream.
2. Philips' error resilient/SNR scalable scheme, In case the coefficient selector selects all coefficients, a course quantisation is performed by Qh and a finer re-quantisation is performed by Ql.
3. Australian scalable/error resilient scheme, In case the coefficient selector always selects the 4x4 low frequency coefficients, and Qh and Ql could be identical: coefficient coded in the high priority data stream will generate zero's coefficients in the low priority data stream. If Ql performs a courser quantisation then Qh, the 4x4 low frequency coefficients are refined.

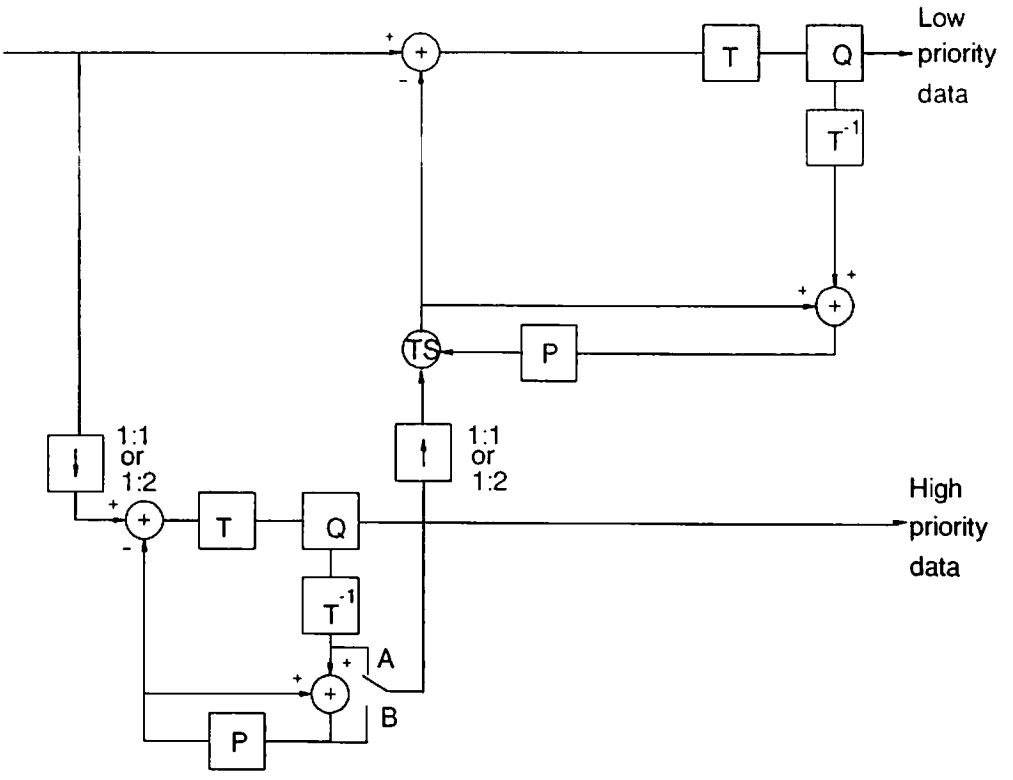


Figure J.8: Possible implementation of a hybrid harmonised scalable/compatible encoder depicted in figure J.4b and figure J.5 (switch set to A), able to generate an MPEG1/H.261 high priority data stream

Data paths A and B are not used concurrently. Data path A is selected for the lower complexity Harmonised Scalable decoder (figure J.4), data path B is selected for full scalable, compatible, higher image quality hybrid harmonised scalable decoders.

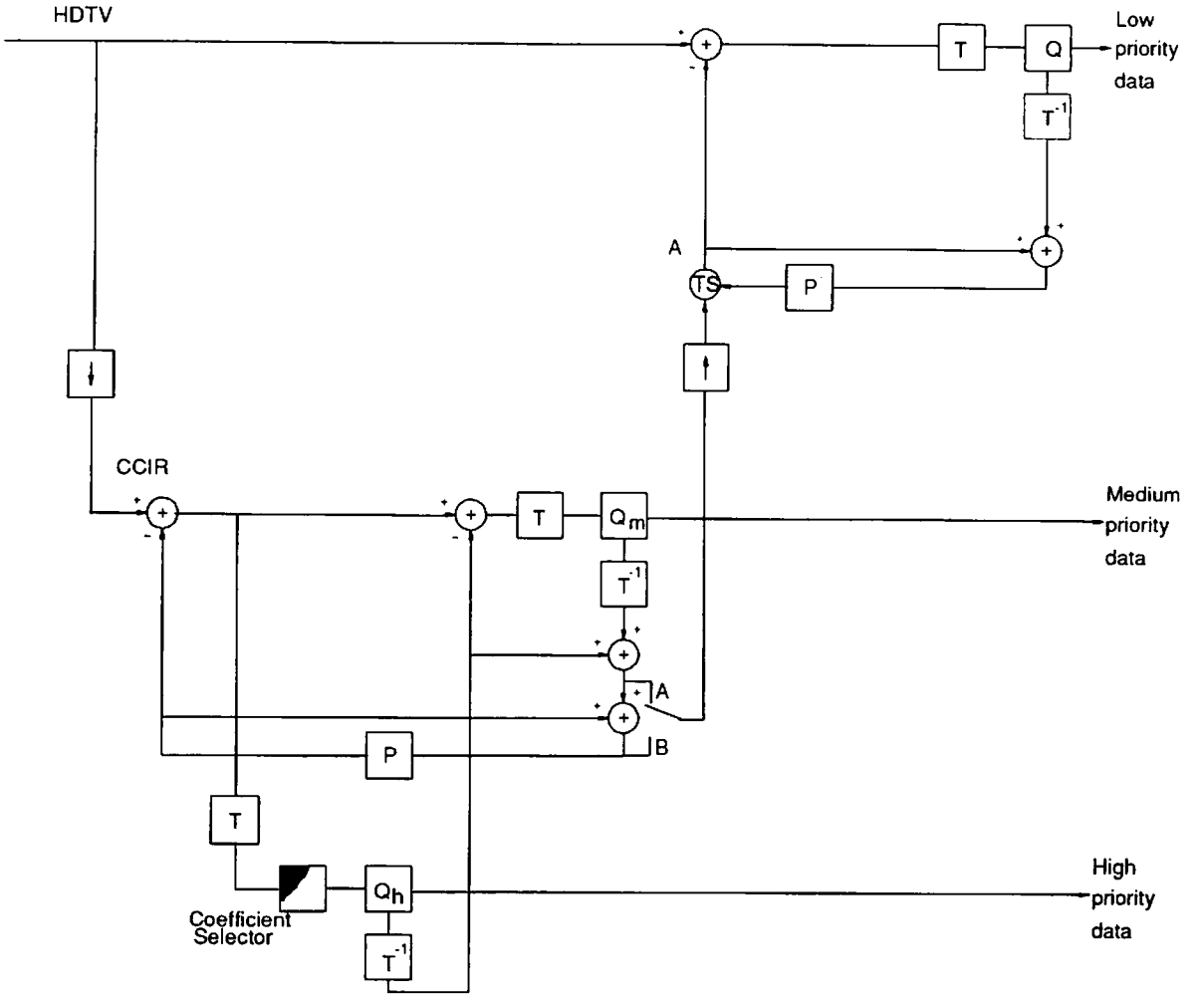


Figure J.9: Possible functional block diagram of 3 level HDTV encoder

J.6 Encoder conclusion

The proposed structure has merged the Thomson-Sarnoff error resilient system, the Philips error resilient/SNR scalable system, the Australian low complexity/error resilient system, the BT/AT&T compatible system and the PTT compatible system. No changes to the syntax on the macroblock layer have to made. Signalling of the exact configuration of the encoder for will happen at the sequence layer.

J.8 Upconversion prediction filter

The hardware complexity of the prediction upconversion filter is a matter of concern, and needs be studied further, for the upconversion in the spatial (Swith in B) and the error domain (Swith in A of for example figure J.8).

For the spatial domain some filter are defined in Appendix G.2. It should be noted that all these upconversion filters have to be standardised or they should be downloadable.

For the error domain upconversion prediction filter can be and linear interpolation ([1 2 1]-filter) within a macroblock or block, the upconversion should be done inter frame for efficiency reasons.

NOTE: Having inter block upconversions allow for less storage.

J.9 Harmonisation of Macroblock layer

An example of how to harmonise the macroblock syntax is given below.

macroblock() {	No. of bits	Mnemonic
if (<sequence extension was not present>)		
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	1-8	vlclbf
if (macroblock_compatible && picture_coding_type != 1 && compatible_mtype != '10')		
prediction_weight_code	1-3	vlclbf
if (macroblock_motion_forward ||		
macroblock_motion_backward) {		
if (picture_structure == 'frame') {		
if (frame_pred_frame_dct == 0)		
frame_motion_type	2	uimsbf
} else {		
field_motion_type	2	uimsbf
}		
}		
if ((picture_structure == 'frame') &&		
(frame_pred_frame_dct == 0) &&		
(macroblock_intra || macroblock_pattern))		
dct_type	1	uimsbf
} else {		
increment_macroblock_type	1-5	vlclbf
}		
if (macroblock_quant)		
quantizer_scale	5	uimsbf
if ((macroblock_motion_forward && motion_refinement)||		
(macroblock_intra && concealment_motion_vectors))		
forward_motion_vectors()	...	...
if (macroblock_motion_backward && motino_refinement)		
backward_motion_vectors()	...	...
if (macroblock_intra && concealment_motion_vectors)		
marker_bit	1	

if (macroblock_pattern)		
coded_block_pattern()	...	...
if (!chroma_scalable) {		
block_count_start = 0;		
block_count_end = block_count;		
} else {		
block_count_start = 4;		
if (chroma_format == 4:2:2) {		
block_count_end = 8;		
} else if (chroma_format == 4:4:4) {		
block_count_end = 12;		
}		
}		
for (i=block_count_start; i<block_count_end; i++) {		
if (frequency_scalable) {		
if (new_macroblock) {		
scaled_block(i)		
} else {		
if (coded_coefficiens) slave_block(i)		
}		
} else {		
if (compatible_mtype=="10" && !macroblock_pattern) {		
snr_block(i)		
} else {		
block(i)		
}		
}		
}		
if (picture_coding_type == 4)		
end_of_macroblock	1	"1"
}		

APPENDIX K: FAST FORWARD AND FAST REVERSE MODES

K.1 Conce . f FF/FR mode in DSMs

The target of "FF/FR mode in DSMs" is to provide the FF/FR mode for *cheap* DSM.

The *cheap* DSM means that DSM does not have neither MPEG decoder nor encoder. Therefore the DSM records the MPEG bitstream and playbacks the bitstream transparently. The bitstream is decoded by the MPEG decoder in a TV set.

In FF/FR mode, the playbacked bitstream should be close to valid MPEG2 bitstream as much as possible, so that "simple"(not necessary main profile) MPEG2 decoder can decode and display the pictures in FF/FR mode.

There are several proposed methods, and they are classified into two groups. One is Intra_slice approach, and the other is Data Partitioning approach.

K.2 Intra_slice approach

In FF/FR mode in DSMs, intra-slice is retrieved in burst condition. So there are some problems to be solved.
(1) A series of intra-slices has gaps between intra-slices.
(2) Sequence-layer, GOP-layer and picture-layer may be lost.

This type of bitstream is not allowed in MPEG1 syntax or semantics.

The "simple" decoder needs more flexible semantics than main decoder.

For example the decoder should deal with the gaps between intra-slices. This gap will be overwritten or filled with *error_start_code*, and the decoder should neglect the gap between intra-slices. And when the FF/FR mode is started, the decoder is recommended to use the same frame/field memory for reconstruction of image, so that the decoding process can be independent of the display process. Therefore the decoder can display the reconstructed image every 1/30 sec without picture_header.

K.3 Requirements for syntax extension

In FF-FR mode in DSMs, intra-slice is expected to be decoded independently. Therefore,
(1) DSM should be informed whether intra-slices exist or not in the bitstream.
(2) The decoder should be informed whether the bitstream is in FF/FR mode or not.
(3) Intra-slice is expected to be distinguished from non-intra-slice.
(4) Intra-slice should be decoded without information from upper layer, i.e., Picture-layer or Sequence-layer. Therefore slice-layer should have more information.

sequence(){
one flag for FF/FR DSM mode: This 1 bit has information of the existence of intra_slices.
}

```
slice(){
 slice_start_code                32
 quantizer_scale                  5

 if(nextbits()=='1'){
  extra_bit_slice                 1 "1"
  FF/FR_mode_flag                   1
  FF/FR_information_slice_1         7
  extra_bit_slice                 1 "1"
  FF/FR_information_slice_2         8
 }

 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()
}
```

K.4 Data Partitioning approach

K.4.1 FF/FR Operation by use of data partition

Fast forward/fast reverse can be implemented through a use of the data partition functionality within MPEG-2 syntax. For this operation the encoder would create a fast-scan partition which represents the data decodable during a fast scan operation. This data must be decodable independent of data outside the fast-scan partition. This data must also meet the data rate constraints for fast-scan playback operation.

K.4.2 VCR FF/FR Bitstream Delivery

The bitstream delivered by the VCR during FF/FR operation is a subset of the full encoded bitstream. Ideally a VCR could deliver 1/N times the original bitstream for an N times speed up. Considering the realistic constraints on VCR operation a more accurate model of FF/FR bitstream would be 1/(2N) of the original bitstream. For FF/FR operation by data partition, the encoder must manage the FF/FR partition buffer to maintain a lower layer bit-rate within the constraints of FF/FR playback.

The FF/FR bitstream need not have any other constraint upon it other than that any data below the picture layer must be extracted from the delivered bitstream, the FF/FR bit rate may not exceed 1/(2N) times the original bit-rate for a speed up factor of N, and it must be legally decodable within the syntax allowed for FF/FR operation. These constraints are met by the data partition syntax to FF/FR.

To maximize image quality during FF/FR, encoding full low-resolution intra-coded frames in the FF/FR partition is recommended. The presence of an entire frame provides a more coherent sequence in the presence of motion. If the FF/FR frames are a concatenation of data extracted from several frames motion within the frame tends to break up and can lead to chaotic images in the case of panning sequences and scene changes.

K.4.3 Syntax for FF/FR by Data Partition

In order to create the FF/FR partition, the encoder would select a priority-break point (PBP) for each slice based on its utility for FF/FR recording. Syntax and operation of the PBP is described in document MPEG 93/240.

The PBP is included in the slice extension as shown below:

```
slice() {
        slice_start_code                32
        quantizer_scale                 5

        if (nextbits()=='1') {  /* slice extension present */
                extra_bit                               1       '1'
                if (FF/FR_sequence) {  /* sequence FF/FR compatible */
                        PBP(1)          8
                        PBP(2) = 63;
                        p_num_max = 2;
                }
        }

        while (nextbits() == '1') {
                extra_bit                               1       '1'
                extra_slice_information;
        }

        extra_bit                                       1       '0'

        do {
                macroblock()
        } while (nextbits()!='000 0000 0000 0000 0000 0000 0000')
        next_start_code()
}
```

An example of a fast-scan partition would be setting the break point after the DC coefficient of every slice in a intra-picture and after the slice header for slices of all other pictures. This would provide for fast forward and fast reverse at the intra-picture rate.

```
if (picture_coding_type == I)
        PBP(1) = 0;
else
        PBP(1) = 67;
PBP(2) = 63;
```

Because the all layers from the slice header and above are included in the fast-scan partition. It is fully decodable as an lower layer partition bitstream. If further levels of data partitioning are needed for other applications they can be defined by there own partitioning algorithms as layers higher than the FF/FR partition. Multiple speed FF/FR might be optimized by several layers of data partitioning with each higher layer adapted to a slower FF/FR playback speed.

The decoding syntax is identical to the decoding syntax for a lower layer only decode of a PBP partitioned bitstream, with the caveat that frames which are skipped because of fast-scan are not concealed or

presented. In FF/FR playback decoding the **p_num_max** is set to 1 for decoding only the lowest layer. For normal playback decoding **p_num_max** is set to 2 for decoding the entire bitstream.

```
if (FF/FR_playback)
        p_num_max = 1;
else
        p_num_max = 2;
```

A possible method for indicating how FF/FR frames are to be presented would be an extension to the sequence header which indicated the speed up factor. This extension would be present when FF/FR playback is indicated in the sequence header.

APPENDIX L: DATA PARTITIONING

Data partitioning is an efficient layering method suitable for increased error resillience . provides some level of scalability. The bitstream consists of a number of layers (called *partitions*) that multiplexed at the System level, which identifies each partition. All startcodes and sequence, gop, picture, and slice headers are duplicated at all partitions.

L.1. Syntax for Data Partitioning

L.1.1 Sequence Layer Syntax

The following changes to the syntax are necessary at the sequence, picture, and slice headers.

```
sequence() {
...
        data_partition_flag             1       uimsbf
...
}

picture() {
...
        if (data_partition_flag)
                intra_pbp               8       uimsbf
...
}

slice() {
        slice_start_code        32      bslbf
        quantizer_scale         5       uimsbf
        if (data_partition_flag)
        pbp                     8       uimsbf
   ....
}
```

L.2 Description

In a partitioned bitsream, **data_partition_flag** is set to 1.

All MPEG syntax elements are grouped into *priority classes*, , as shown in Table 1. The *priority breakpoint* (**pbp**) indicates which priority classes are to be expected in the current partition. For example, **pbp**=2 means that the current partition contains codewords (headers or DCT coefficients) up to the first nonzero coefficient after the first DCT coefficient, i.e. those with priority class 6 or less. If there are no nonzero coefficients beyond **pbp**=1, then the **eob** codeword is placed in the partition.

In P or B pictures, the value of **intra_pbp** is used as the breakpoint for intra macroblocks, i.e. the **pbp** value is overridden.

In a non-partitioned bitstream (or for the last partition), **pbp** = 63.

pbp	Priority class	Data included in the priority class
65	0	All data at the Sequence, GOP, Picture and Slice layers down to the pbp code.
66	1	MB data starting with MB stuffing and ending with MB type
67	2	Data up to the first motion vector
68	3	Rest of the motion vectors
0	4	MB data starting with CBP and up to DC coefficient or first nonzero coefficient
1	5	First coefficient following DC or first nonzero coefficient after the first coefficient in the scan order
2	6	Second coefficient following DC or first nonzero coefficient after the first coefficient in the scan order
.	.	
j	j+4	j'th coefficient following DC or first nonzero coefficient after the (j-1)'st coefficient in the scan order

Table 1. PBP values and their semantic interpretation.

There is no additional syntax at the macroblock or block level, however the semantics of decoding have to be modified. The decoder needs to keep track of the current and previous breakpoints, and switch between partitionreakpoints are reached in the decoding process. When the decoding process reaches a point identified as the current breakpoint (pbp), the following actions are taken:

1. The next partition becomes the current partition. Thus, the bits for the next codeword are fetched from the next partition.
2. The current **pbp** is stored.
3. The **pbp** is set to the breakpoint specified in the new partition.

When the decoding procedure reaches the previous breakpoint, the following actions are taken.

1. The previous partition becomes the current partition. Thus, the bits for the next codeword are fetched from the previous partition.
2. The **pbp** is set to the breakpoint specified in the new partition.

Note that the decoder needs to keep track of only one breakpoint when there are two partitions. An example is presented in Figure L.1.

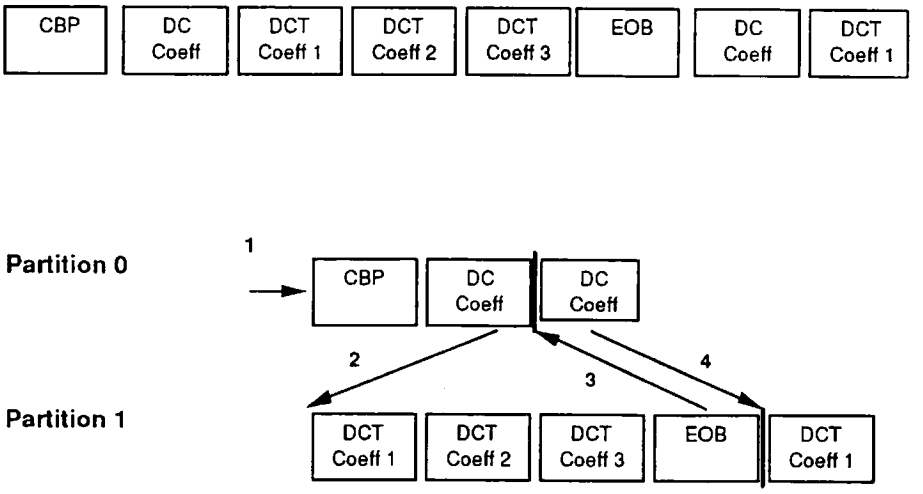


Figure L.1 A segment from a bitstream with two partitions, with **pbp** set to 0. The two partitions are shown, with arrows indicating how the decoder needs to switch between partitions.

L.3 Rate control

The breakpoint can be changed adaptively on a slice basis to provide tight rate control. Ultimately, the control is linked to the rate controller, but a simplified version can also be used. Two simple options are described here.

1. Deterministic method:
   Buffer one slice.
   Analyze the number of bits used by each priority class.
   Compute the breakpoint that results in the desired rate split.

2. Adaptive method:
   Start with the last breakpoint in the lmost recent picture of the same type (must be initialized depending on the desired rate split).
   If the last breakpoint resulted in more bits than desired in the lower partition, then decrease the breakpoint. Otherwise, increase the breakpoint.

When M=1 (no B pictures), equal allocation of bitrate ratios between I and P pictures works well. When M=3, few bits should be used by B pictures in the base partition.

L.4 Experiments

Experiments should focus on two aspects:
1. Cell loss performance (described in Appendix F).
2. Base partition only decoding. The results should be displayed both in original resolution and also in quarter resolution (i.e. CCIR-601 and SIF). Two methods are suggested:
   Use pbp=8 for I pictures, 6 for P pictures, with M=1 N=9.
   Use a 50/50 split and one of the rate control methods.

APPENDIX Q: QUANTISATION

Q 5. EXPERIMENT OF NON-8 x 8 DCT (Revised at Sydney)

Purpose : To verify the coding efficiency and improvement of visual quality

1. Horizontally one dimensional DCT (8x1 DCT)

The one-dimensional DCT is defined as;

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

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

where x = spatial coordinates in the pel domain

u = coordinates in the transform domain

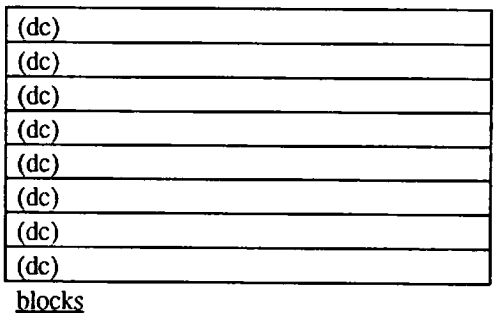
$$C(u) = \begin{cases} 1/\mathrm{sqrt}(2) & \text{for } u = 0 \\ 1 & \text{otherwise} \end{cases}$$

Coefficient range is - 724 ..... 724 (-2048 ..... 2048 in 8x8 DCT)

The one-dimensional IDCT is defined as;

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

Block size is same as normal 8x8 DCT, then 8 sets of horizontal one-dimensional DCT coeff. exist in each block independently. In case of Intra MB, 8 DC components are in a 8x8 block.



blocks

2. Quantization

Quantizing manner is same as 8x8 DCT by using one dimensional Matrix as below;

(These matrices are the same as a raw of 8x8 DCT matrices, except DC term for Intra-MB.)

8	22	26	27	29	34	37	40
8	22	26	27	29	34	37	40
8	22	26	27	29	34	37	40
8	22	26	27	29	34	37	40
8	22	26	27	29	34	37	40
8	22	26	27	29	34	37	40
8	22	26	27	29	34	37	40
8	22	26	27	29	34	37	40

Q-Mat. for Intra 8 x 1 DCT

20	21	22	23	25	26	27	28
20	21	22	23	25	26	27	28
20	21	22	23	25	26	27	28
20	21	22	23	25	26	27	28
20	21	22	23	25	26	27	28
20	21	22	23	25	26	27	28
20	21	22	23	25	26	27	28
20	21	22	23	25	26	27	28

Q-Mat. for Non-Intra 8 x 1 DCT

3. Scanning

Scanning for 8 x1 DCT block is vertical 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

4. Coding of coeff.

Coding of coeff. is basically same as 8x8 DCT.

a) Intra-MB

DC coeff.: Top one is a representative value for taking inter-block DC difference, and, coded using DC-VLC. Take difference with same type DCT only.

Other 7 DCs are calculated the difference from upper one in a 8x8 block, and, coded with AC coeff. using AC-VLC

AC coeff.: Same as 8x8 DCT

Totally, 63 Coeff.(7 DC and 56 AC) are coded AC-VLC.

b) Non-Intra-MB

Same as 8x8 DCT

Adaptation of 8 x 8 DCT / 8 x 1 DCT

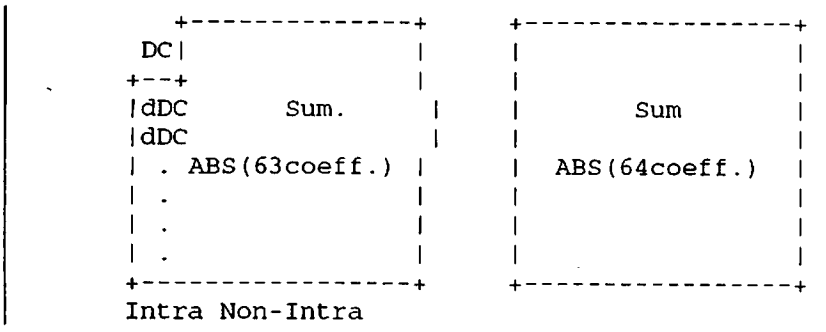
Switching unit

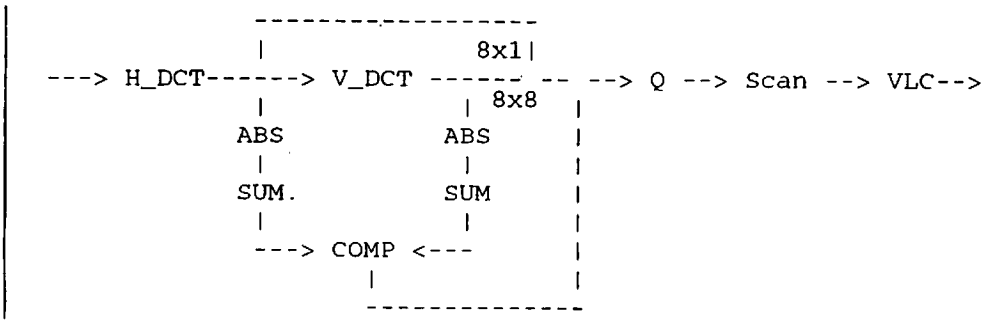
Block

Decision method

Use the summation of absolute coeff. before quantization

1. Calculate the summation of all absolute coeff. for both DCT types.
2. Select smaller one
3. If Intra-MB, take the difference from upper DC-component and reject top DC in 8x1DCT-block. Reject DC of 8x8DCT also. Add offset value "0" or "64" (1 for a pixel) on 8x1DCT-block.





Special Rate Control Step 3 (Adaptive Quantization)

1) For 8x8 DCT only
2) For 8x8/8x1 DCT
To be added (until end of May)

Core experiments

Frame-picture with Fr/Fi-MC & DCT, Field-picture with Fi/16x8-MC
M=1 and M=3, 4Mbps and 9Mbps

Syntax

```
block(i) {
        if (pattern_code[i]) {
            one_dimensional_DCT   1
            if (macroblock_intra) {
```

Reference

MPEG 92 / 093, 261, 444, 445, 562
MPEG 93 / 083, 167, 272, 401

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
