Abstract

This proposal describes a flexible video compression architecture, suitable for interlaced and non-interlaced video, which facilitates several types of scalability. The architecture is based on extensions to the MPEG-1 algorithm and syntax in a manner that minimizes decoder complexity, and supports resolution and bit-stream scalabilities, as well as the picture-rate scalability already inherent in the MPEG-1 algorithm. Because of the wide scope of potential applications of the MPEG-2 algorithm, this proposal emphasizes flexibility in the number and nature of the scales present in a bit-stream. It includes the case of a single layer of high quality, which is fully compatible with MPEG-1 syntax and decoding procedures.

AVC-151

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION

ORGANISATION INTERNATIONAL DE NORMALISATION

ISO/IEC JTC1/SC2/WG11

CODED REPRESENTATION OF PICTURE AND AUDIO INFORMATION

ISO-IEC/JTC1/SC2/WG11 N
MPEG 91/212, 91/244
November 8, 1991

Title: **A Proposal for MPEG-1 Coding with Scalable Extensions**

Source: **C. Gonzales and E. Viscito (IBM)**

1. INTRODUCTION

There are several conflicting requirements for the MPEG-2 video algorithm from the diverse MPEG-2 community. The main reason for these conflicts is the great variety of applications for which MPEG-2 is intended, all of which have different requirements in terms of compatibility, encoder and decoder implementation complexity, functionality, and image quality, among other things. We believe that these conflicting requirements cannot be satisfied by a single coding algorithm, but instead require a flexible architecture of algorithms which can be matched to the requirements of the specific application. A properly designed architecture will satisfy many of these conflicting requirements, while still preserving a great deal of compatibility among the different manifestations of the architecture. We believe, for example, that whereas different encoder implementations may be needed for different applications, it should be possible to implement a single decoding device which can decode all bit-streams that conform to the flexible architecture. Such a generic decoder should not be excessively complex.

This document describes a proposal for such an architecture. Because the MPEG-1 algorithm [1, 2] operating at MPEG-2 bit-rates provides very good quality video, and because of our desire to retain a degree of compatibility with MPEG-1, we have built the architecture by extending the MPEG-1 syntax and decoding methodology. The extensions permit the encoding of video that is scalable. By scalability, we mean the ability to have more than one image resolution and/or quality level readily available in the bit-stream. At its simplest the architecture supports an MPEG-1 compatible, non-scalable, bit-stream for progressive video. A slight extension, which is compatible with MPEG-1 syntax, permits the coding of non-scalable interlaced video. Finally, in its most developed manifestation, the architecture supports up to four bit-stream or resolution scales in a manner that retains a great deal of commonality with MPEG-1 syntax and core technology. A device capable of decoding any of the manifestations of the architecture is just slightly more complex than an equivalent MPEG-1 decoder.

2. ARCHITECTURE OPTIONS

2.1 NON-SCALABLE PROGRESSIVE VIDEO

In this case the bit-stream is fully compatible with MPEG-1 syntax, semantics, and decoding methodology.

2.2 NON-SCALABLE INTERLACED VIDEO

In this case the bit-stream is fully compatible with MPEG-1 syntax. There is, however, a different interpretation for motion compensation vector information. Coding takes place by combining the two fields together as a single picture; however, motion compensation is field-based. Motion vectors that are common to both fields are coded using MPEG-1 syntax and coding tables. These vectors are then separately applied to both fields. Further, whereas the horizontal component of motion vectors can be coded

with 1/2 pixel resolution, as in MPEG-1, the vertical component is always coded with integer pixel resolution. With this vertical component resolution, motion compensation can also be implemented in the full picture domain by simply doubling the vertical components of the motion vectors, interpreting them as integer pixel components, and applying them to the full picture. Other than this, the decoding methodology is the same as non-core MPEG-1.

2.3 SCALABLE INTERLACED OR PROGRESSIVE VIDEO

We summarize our understanding of three different modalities of scalability which are desirable for the MPEG-2 algorithm:

1. **Resolution Scaling:** By this we refer to the ability to generate a bit-stream that can be decoded at a multiplicity of spatial resolutions. This feature is desirable in some applications where multiple video windows must be displayed in a full resolution screen. It is also desirable because it permits the implementation of decoders of varying degrees of complexity, such that very simple decoders may be possible that decode only the lower spatial resolutions.
2. **Bit-stream Scaling:** By this we refer to the ability to generate a bit-stream in which some coded bits can be disregarded and a usable image still results. Bit-stream scalability is often associated with resolution scalability; however, we interpret it differently here. By bit-stream scalability we mean two or more layers of compressed data associated with a single spatial resolution. This is a useful feature for graceful degradation of the quality of decompressed video when some of the compressed bit-stream data are corrupted by noise.
3. **Rate Scaling:** By this we refer to the ability of generating a bit-stream that can be played at a multiplicity of temporal resolutions. This feature is inherent to the MPEG-1 algorithm, and therefore to the proposed architecture. By proper choice of the MPEG-1 M-parameter, and by selectively skipping over one or more of the B-pictures, the MPEG-1 syntax permits decoding at a multiplicity of temporal resolutions. Thus rate scaling is not discussed any further in this document.

The architecture supports all of these types of scaling by extending the MPEG-1 syntax and decoding methods, while retaining a great degree of commonality with MPEG-1 core technology. The architecture supports resolution and bit-stream scaling by hierarchical coding of the 8x8 DCT components in a manner that resembles progressive spectral selection. We support up to four levels of resolution scales although, in principle, eight levels are possible with the 8x8 DCT. The lowest possible resolution is attained by coding the equivalent of the DC component of an 8x8 DCT block; this resolution is 1/64 the original resolution. Next we support a resolution of 1/16 the original resolution by coding of the equivalent of the upper left 2x2 coefficients of the DCT block. Coding of the equivalent of the upper left 4x4 coefficients leads to a resolution of 1/4 the original resolution. Finally, coding of all 8x8 coefficients leads to the full resolution video.

The architecture is flexible: one or more of these resolution scales can be stacked up in order of increasing resolution, such that the reconstructed coefficients at one level of resolution are used to predict the corresponding coefficients at the next level of resolution. Thus, transform coefficients are coded differentially with respect to a prediction from lower resolution coefficients. Note that we support complete flexibility in the choice of resolution scales. An encoder might choose, for example, to generate a bitstream that only contains data for the full and the 1/16 resolution scales.

A fundamental characteristic of the architecture is that the identity of the MacroBlock (MB) structure of MPEG-1 is preserved across all resolution scales. Figure 1 shows how the MB identity is preserved by scaling across four levels of resolution. It is important to preserve this identity because a MB is associated with a series of attributes which contribute to the amount of overhead data incorporated in an MPEG-1 compressed data stream. Examples of these attributes are the MB address, type, and motion vectors. By preserving the MB identity across multiple levels of resolutions, all resolution levels can share these attributes, thus requiring their inclusion only once in the data stream. Preserving the MB identity also significantly simplifies the derivation of motion estimation vector data for all resolution scales other than the highest resolution. Essentially the motion vector data corresponding to any resolution scale can be derived from the highest resolution motion vector data by appropriate scaling. For example, the x- and y-motion-vector components at 1/4 resolution are 1/2 the corresponding full resolution components.

2.3.1 Implementation of resolution scaling

In MPEG-1 a MB is subdivided into six 8x8 blocks of luminance and chrominance information, each block being coded using the 8x8 Discrete Cosine Transform (DCT). In our architecture, each scaled MB is also divided into six blocks of luminance and chrominance information, each block also corresponding to a smaller DCT. The small-DCT coefficients can be derived from the 8x8 full resolution coefficients or, alternatively, they can be derived by direct computation. Thus for the 1/4 resolution MB in Fig. 1, we could use a DCT of size 4x4. These alternatives, which are encoder options, are discussed in more detail in section 5.

For decoding at a scaled resolution, however, we propose that a DCT of the appropriate size be used. Thus for decoding at 1/4 resolution a 4x4 inverse DCT should be used. The one dimensional DCT matrices appropriate for decoding at the three "scaled" resolutions we support are:

$$IDCT_1 = 1$$

$$IDCT_2 = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$IDCT_4 = \frac{1}{2}\begin{bmatrix} 1.00 & 1.31 & 1.00 & 0.54 \\ 1.00 & 0.54 & -1.00 & -1.31 \\ 1.00 & -0.54 & -1.00 & 1.31 \\ 1.00 & -1.31 & 1.00 & -0.54 \end{bmatrix}$$

Both $IDCT_1$ and $IDCT_2$ are trivial transforms and should be easy to implement, perhaps even by "software only" decoders. The $IDCT_4$ which gives SIF resolution from an 8x8 CCIR 601, is not trivial but it is still relatively simple.

2.3.2 Implementation of bit-stream scaling

The lower resolution DCTs of the previous section can be interpreted as a subset of 8x8 DCT coefficients. Thus full resolution images of lower quality can be reconstructed by applying the inverse 8x8 DCT to the coefficient data of less than full resolution scales.

There is an alternative, however, for implementing bit-stream scalability which is fully supported by the syntax of the architecture we describe here. By specifying multiple layers at the same resolution level, but with progressively finer quantization factors; these layers would represent video of the same spatial resolution, but of increasing quality.

3. DESCRIPTION OF MPEG-1 EXTENSIONS

3.1 MULTIPLEXING OF SCALING LAYERS

We multiplex data for the various scales at the level of the slice layer. A slice, in this case, is always a row of 44 macroblocks. To facilitate demultiplexing and decoding at the various resolution scales, data from the various scales is separated by byte aligned start codes. We define a macroslice layer to replace the normal slice layer of MPEG-1 [1]. The macroslice layer is defined as follows:

```
macroslice() {
  slice()
  do {
    slave_slice()
  } while (nextbits()==slave_slice_start_code)
}
```

The slice layer is as in MPEG-1, except that the imbedded macroblocks contain coefficient data for the lowest resolution scale only. The resolution of this lowest scale is passed as **extra_information_slice**, with a value of 1, 2, 4, or 8 for the 1x1, 2x2, 4x4, and 8x8 DCT sizes, respectively. The default, when no **extra_information_slice** is present, is a non-scalable 8x8 slice layer. DCT coefficients of the imbedded macroblocks are coded in the zig-zag order of the corresponding DCT size.

The slave_slices contain DCT coefficient data only for the blocks in the slice layer marked to contain data as specified by the coded block pattern (blocks in intra macroblocks are always assumed marked.) The slave_slice layer is defined as follows:

```
slave_slice() {
  slave_slice_start_code          /* 32 bits */
  slave_slice_vertical_position   /* 8 bits */
  quantizer_delta                 /* 5 bits */
  dct_size                        /* 5 bits */
  for (s=0; s<slice_size; s++) {
    for (i=0; i<6; i++) {
      if (pattern_code[s][i]) {
        while (more_coefs) {
          while (nextbits() != eob_code) {
            if (lower_scale) {
              dct_coef_differential
            } else {
              next_dct_coef
            }
          }
        }
      }
    }
  }
  next_start_code()
}
```

The following terms are additional to those of [1]:

- **slave_slice_start_code**: the X'000001B0 is used.
- **slave_slice_vertical_position**: the same vertical position of the corresponding slice_layer.
- **quantizer_delta**: specifies a delta that should be added to all macroblocks' **quantizer_scale** factors in the lowest layer, in order to arrive to the appropriate quantizer scaling factor of the slave_slice macroblock.
- **dct_size**: 2, 4, or 8.
- **slice_size**: the total number of macroblocks in the slice layer.
- **pattern_code(s)(i)**: the index "s" has been added to the normal definition of pattern_code in MPEG-1, to identify a specific macroblock in the slice layer.
- **more_coefs**: more_coefs is true if we have not reached the last coefficient in the block of DCT coefficients. For the 8x8 slave_slice, more_coefs is always true.
- **eob_code** An end_of_block Huffman code that depends on the specific resolution scale.
- **lower_scale**: is true if the DCT coefficient is predicted from the next lower resolution scale.
- **dct_coef_differential**: DCT prediction difference coded by run/amplitude or run/size VLC.
- **next_dct_coef**: DCT coefficient coded by run/amplitude or run/size VLC.

3.2 QUANTIZATION OF RESOLUTION AND BIT-STREAM SCALES

Except for fixed scaling factors, we propose to use the same quantization matrices for all resolution scales. If the DCT data for the various resolution scales is derived from the full resolution 8x8 DCT, then quantization is just as with MPEG-1. On the other

hand if these data are derived from DCT implementations of smaller sizes, then the appropriate elements of the full resolution quantizer matrices, Q8, should be scaled. The correct scaling factors are as follows:

Quantizer	DCT	Factor
Q1	1x1	1/8 Q8
Q2	2x2	1/4 Q8
Q4	4x4	1/2 Q8

At less than full resolution, the precision of the quantization process will not affect decodability of a bitstream. Nevertheless, we recommend that small-DCT coefficients be quantized by first scaling them up by the appropriate factor and then quantizing them using Q8.

On the other hand, the precision and process of dequantization at less than full resolution needs to be clearly specified. The method we have implemented is as follows:

- Dequantize using MPEG-1 rules and the full resolution quantizer matrix Q8
- Inverse transform with an IDCT of the appropriate size that incorporates a scale down of the results. In other words, if we express a two dimensional inverse-DCT in matrix notation by using the Kronecker product, $\otimes$, by

$$IDCT(N \times N) = (IDCT_N \otimes IDCT_N)$$

then we propose to use the following "unnormalized", $IDCT^*$ definitions:

$$IDCT^*(1 \times 1) = (IDCT_1 \otimes IDCT_1)/8$$
$$IDCT^*(2 \times 2) = (IDCT_2 \otimes IDCT_2)/4$$
$$IDCT^*(4 \times 4) = (IDCT_4 \otimes IDCT_4)/2$$

There are obviously other equivalent ways of implementing dequantization.

3.3 HIERARCHICAL PREDICTION OF RESOLUTION AND BIT-STREAM SCALES

DCT coefficients in a low resolution layer are used to predict the corresponding coefficients in the next (higher) resolution layer. An example is shown in Figure 2 where a hierarchy of 4 resolution layers is shown. The prediction algorithm, at this point, is a simple one-to-one mapping of the properly scaled coefficients. Further details are given in sections 4 and 5.

Bit-stream layers of the same resolution are predicted by one-to-one mapping of corresponding coefficients.

3.4 VLC CODING OF RESOLUTION SCALES

If a 1x1 layer is present, intra coefficients are coded with the dc-VLCs of MPEG-1. Non-intra coefficients are coded with another VLC similar to MPEG-1's dc-VLCs (see Annex.) Note that no eob_code is needed in this case.

If a 2x2 layer is present, we distinguish between two cases. If the layer follows a 1x1 layer, coefficients are coded in all cases by using a JPEG-like VLC of run/sizes [4], where the maximum run is 3 and the maximum logarithmic size is 8 (see Annex.) If the 2x2 layer is the first resolution layer, intra DC coefficients are treated as in MPEG-1, the rest of coefficients are coded with the same JPEG-like VLC.

If a 4x4 layer is present, we distinguish between two cases. If the layer follows a lower resolution layer, coefficients are coded in all cases by using a JPEG-like VLC of run/sizes, where the maximum run is 15 and the maximum logarithmic size is 8 (see Annex.) If the 4x4 layer is the first resolution layer, intra DC coefficients are treated as in MPEG-1, the rest of coefficients are coded with the same JPEG-like VLC.

Finally, if an 8x8 layer follows a lower resolution layer, coefficients are coded in all cases (intra macroblocks included) by using the standard MPEG-1 tables 2-B.5c-g, without the special-case treatment of the first run/amplitude event.

3.5 PROVISIONS FOR RESOLUTION AND BIT-STREAM SCALE RATE CONTROL

The slave_slice layer specification includes the quantizer_delta parameter. This delta is always specified with reference to the corresponding quantizer scale factor used in the slice layer. The delta is added to the corresponding quantizer_scale in the slice_layer, in order to produce a quantizer_scale factor appropriate to the corresponding slave_slice MB.

4. DECODER IMPLEMENTATIONS

4.1 NON-SCALABLE PROGRESSIVE/INTERLACED VIDEO AT CCIR 601 RESOLUTION

The decoder is essentially an MPEG-1 decoder used at CCIR 601 resolution. A minor difference relates to the interpretation of the vertical component of motion compensation vectors for interlaced video as explained in section 2.2.

As with MPEG-1, the vertical chrominance resolution is half the luminance resolution. We obtain CCIR 601 resolution by simply line replicating the chrominance data.

4.2 SCALABLE PROGRESSIVE/INTERLACED VIDEO

We stress that we favor a flexible architecture. However, for the purposes of illustration we describe here a three layer decoder that we will demonstrate at the Kurihama meeting. The decoder, which is shown in Figure 3a, supports 2x2 (low), 4x4 (medium), and 8x8 (high) resolution scales. Decoders supporting only one target resolution scale can be implemented by eliminating the boxes in Figure 3a that are not needed to achieve

this target. After demultiplexing and entropy decoding, there will be for every 8x8 block, corresponding 2x2 and 4x4 blocks, all of which are necessary to build the final 8x8 matrix of DCT coefficients. The low resolution 2x2 blocks are used as a prediction to the four lowest order coefficients of their corresponding 4x4 blocks. Similarly, the 4x4 blocks are used as prediction to the 16 lowest order coefficients of their corresponding 8x8 blocks.

Note that the DCT coefficients are only dequantized by the corresponding quantizer scale_factor as we rebuild the target matrix of coefficients (see section 3.5.) Dequantization by the corresponding quantization matrix is only needed once we reach the target resolution. This simplifying feature is possible because we use a common quantization matrix as described in section 3.2.

In summary to rebuild the 8x8 DCT coefficients, we take the following steps:

1. multiply the quantized 2x2 coefficients by qp_2
2. multiply the quantized 4x4 coefficients by qp_4. Where appropriate, add the coefficients of the previous step
3. multiply the quantized 8x8 coefficients by qp_8. Where appropriate, add the coefficients of the previous step
4. dequantize the 8x8 coefficients using the appropriate quantization matrix

The final 8x8 matrix of coefficients can now be used to reconstruct the full resolution picture using MPEG-1 techniques, including motion compensated prediction. In this regard, the 16x16 MCP unit in Figure 3a, represents a generic MPEG-1 Motion Compensation Prediction unit.

Note that similar MCP units at other resolutions share the same motion vector data, MV, to generate motion compensated predictions for the scaled MBs at the various resolution scales. A decoder operating at a lower than 8x8 resolution would only rebuild the DCT coefficients up to the required level, it would then apply the same MPEG-1 techniques referred to above. Of special note is that when using motion compensation techniques, the full resolution motion vector should be scaled appropriately to match the decoder resolution.

A full resolution decoder, which receives partially corrupted data, can still produce a degraded reconstruction by utilizing the uncorrupted DCT layers as long as the lowest resolution DCT layer--which contains the MB attributes--is not corrupted. If the lowest layer is corrupted, other concealment techniques are required.

An alternative implementation for bit-stream scalability is shown in Figure 3b. Only 8x8 IDCTs are used here and the various layers are representative of data at increasing levels of amplitude resolution.

5. ENCODER IMPLEMENTATIONS

5.1 NON-SCALABLE PROGRESSIVE/INTERLACED VIDEO AT CCIR 601 RESOLUTION

The encoder is essentially an MPEG-1 encoder used at CCIR 601 resolution. A minor difference relates to the generation of the vertical component of motion compensation vectors for interlaced video as explained in section 2.2.

As with MPEG-1, the vertical chrominance resolution is half the luminance resolution. We have averaged the chrominance data in the two fields of the CCIR 601 data.

5.2 SCALABLE PROGRESSIVE/INTERLACED VIDEO

There are many possible implementations of encoders compatible with the decoders described above. We only describe encoders for resolution scaling here. The general structure of a three resolution layer encoder is shown in Figure 4. We divide the encoder into three parts. The first part is a transform unit which takes the digital video input and outputs partially quantized DCT data for the three resolution layers: d(8x8), d(4x4), and d(2x2). The second part is a hierarchical prediction unit which takes the transform unit's DCT output, and outputs quantized differential DCT data at all resolution layers. This output is multiplexed and entropy coded in the third unit to generate the final compressed video. The prediction unit also generates partially reconstructed DCT data: b(8x8), b(4x4), and b(2x2), which is fed back to the transform unit to complete the loop of typical hybrid transform codecs.

Figure 5a shows a simple implementation of the transform unit. In this implementation the 8x8 layer contains all the elements needed for an MPEG-1 encoder. We derive the 2x2 and 4x4 coefficients by simply extracting them from the corresponding 8x8 coefficients. Obviously one could generalize this version of the transform unit by deriving the 2x2 and 4x4 coefficients through a reduction formula or a weighting of 8x8 coefficients.

Note that because there is no feedback loop in the lower resolution scales, this encoder will result in accumulation of quantization and motion compensation errors at these resolution scales. The error, however will be naturally reset back to zero whenever a new Group of Pictures starts. Whereas the quality of the lower resolution layers will be limited by this accumulation of errors, the simplicity of the encoder makes this approach attractive. In particular, if what is required is only bit-stream scalability, this approach is all that is needed. Simulation results will be shown at Kurihama.

Another implementation of the transform unit is shown in Figure 5b. At a cost of increased complexity, this version would generally produce better quality low-resolution pictures than those from the unit in Figure 5a. Essentially, every resolution layer is a self-contained motion compensated loop taking its input from an appropriately decimated version of the original video. The H_4 and H_2 boxes perform the decimation

function on the input video. Because there is a feedback loop at every resolution scale, coding errors will not accumulate more than one picture period. Note, however, that the results of motion estimation can be shared by all resolution loops since motion vectors are one of the attributes shared by MBs at all scales. This implementation may be more appropriate for applications where the quality of the low resolution video is important.

Regardless of which transform unit is implemented, there are at least two possible implementations for the prediction unit. Figure 6a shows one of them. In this implementation the differential DCT coefficients are generated, followed by quantization by the appropriate quantizer_scale factors. The front end of the decoder of figure 3 is then replicated to generate the DCT data fed back to the transform unit. Figure 6b shows another implementation, which should result in somewhat lower quantization noise.

6. ADDITIONAL FEATURES AND PERFORMANCES

6.1 COMPATIBILITY WITH MPEG-1

The system we propose is in all cases upward and forward compatible with MPEG-1. It is also downward compatible with non MPEG-1 decoders which could, nevertheless, be considered part of the DCT standards family.

6.2 RANDOM ACCESS

The size of the Group of Pictures for all the test and demonstration material we prepared is 12. The maximum random access time is then about 2/5 second.

6.3 CODING/DECODING DELAY

The coding/decoding delay for all the material we prepared for Kurihama is less than 150 ms. We used the VBV criteria of MPEG-1 with the following buffer sizes:

- 4 Mbit/s: 589824 bits
- 9 Mbit/s: 1359872 bits

6.4 FAST FORWARD/REVERSE

Achieved by accessing the intrapicture in a GOP.

7. FUTURE EXPERIMENTS

The codec described here can be modified in many ways.

1. Arithmetic coding could be used instead of fixed VLCs. This method results in compression improvements in the MPEG-1 algorithm of the order of 8-17%. Fur-

thermore, it also reduces the table storage requirements. A relevant reprint is being contributed separately to this meeting [5].

2. Generic quantization matrices could be added for each resolution scale. Such matrices would be inserted in the sequence layer. This would generally result in increased complexity for decoders and encoders.
3. A 16x16 resolution scale could be added for even higher resolutions.
4. More sophisticated predictions could be incorporated when rebuilding higher resolution layers.
5. For better rate control, the quantizer_scale factor for each resolution scale could be generated by a proportional factor instead of a delta.
6. Motion vectors can be coded hierarchically (at the present time we transmit the full resolution vectors together with the lowest hierarchical scale.)

REFERENCES

[1] ISO CD 11172-2, Draft for "Coding of Moving Pictures and Associated Audio--for Digital Storage Media at up to about 1.5 Mbit/s," ISO-IEC/JTC1/SC2/WG11, MPEG 91/ document, August 1991.

[2] B. Astle, "A Proposal for MPEG Video Report, Rev. 1," ISO-IEC/JTC1/SC2/WG11, MPEG 91/ document, August 1991.

[3] "Proposal Package Description for MPEG Phase 2," ISO-IEC/JTC1/SC2/WG11 N0098, MPEG 91/100 document, August 1991.

[4] ISO 10918-1 (JPEG) "Digital Compression and Coding of Continuous-tone still Images"

[5] E. Viscito and C. Gonzales, "Encoding of motion video sequences for the MPEG environment using arithmetic coding," SPIE Vol. 1360 Visual Communications and Image Processing '90, pp. 1572-1576, October 1990.

[6] VLSI Subgroup, "Paper Design for SM1," ISO/IEC/JTC1/SC2/WG8 MPEG 90/ 115, July 1990.

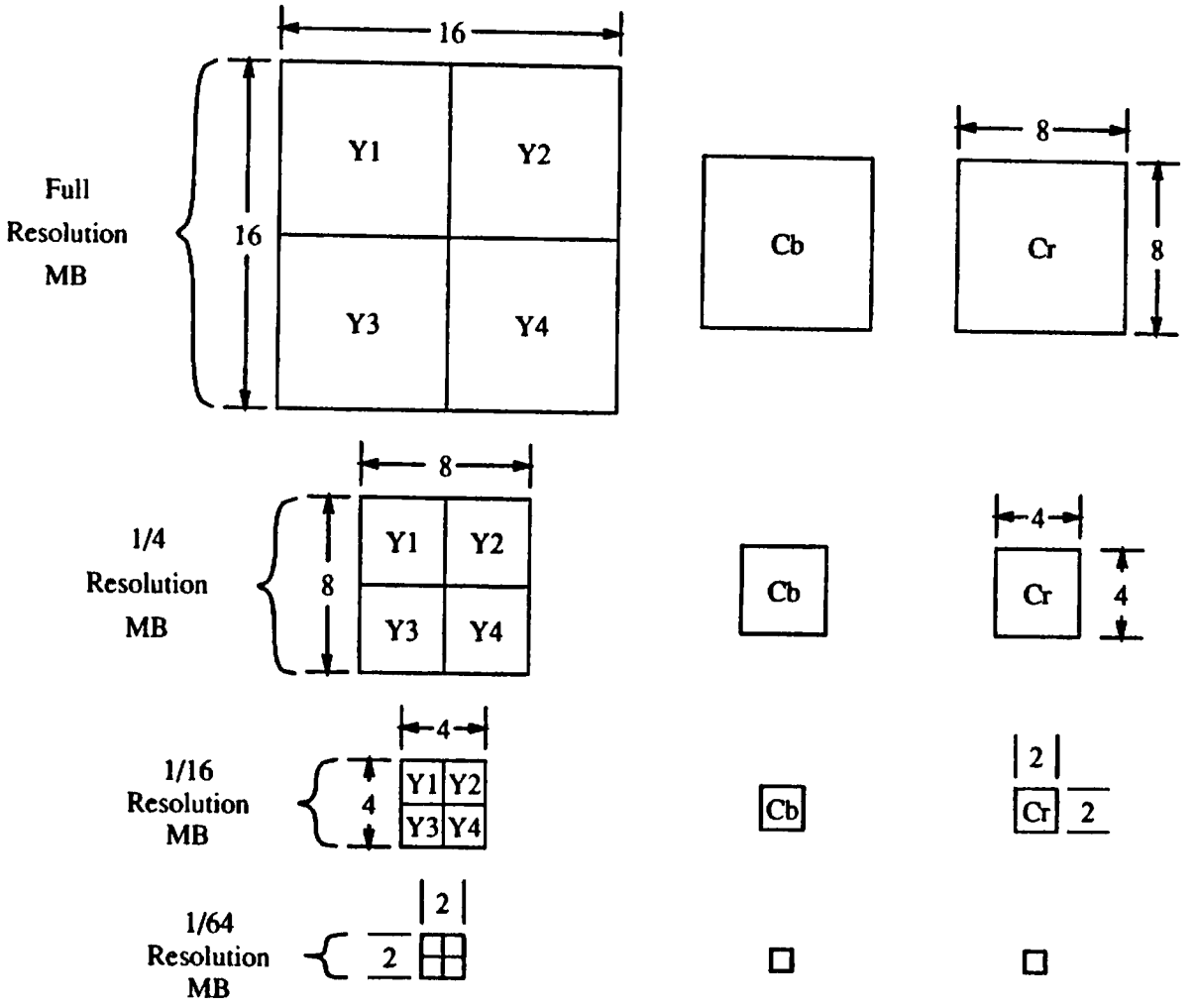

Figure 1: Macroblock respresentation at four resolution levels.

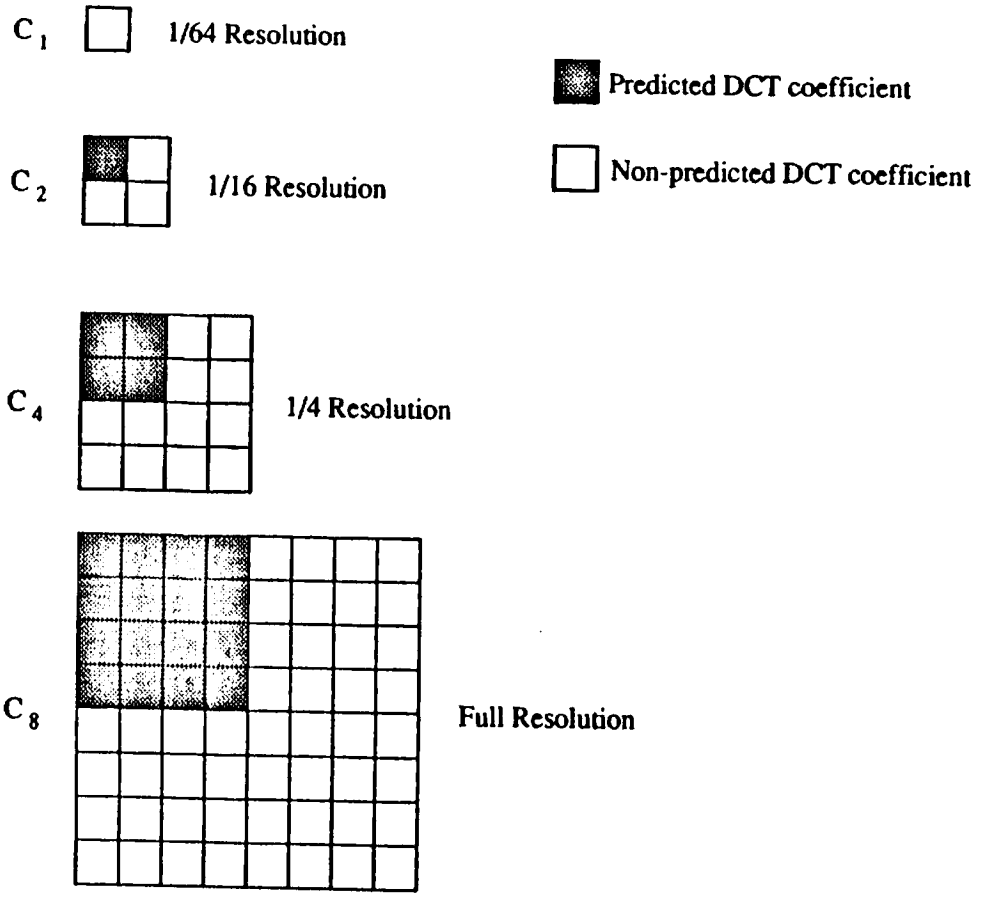

Figure 2: Pictorial illustration of hierarchical prediction of DCT coefficients.

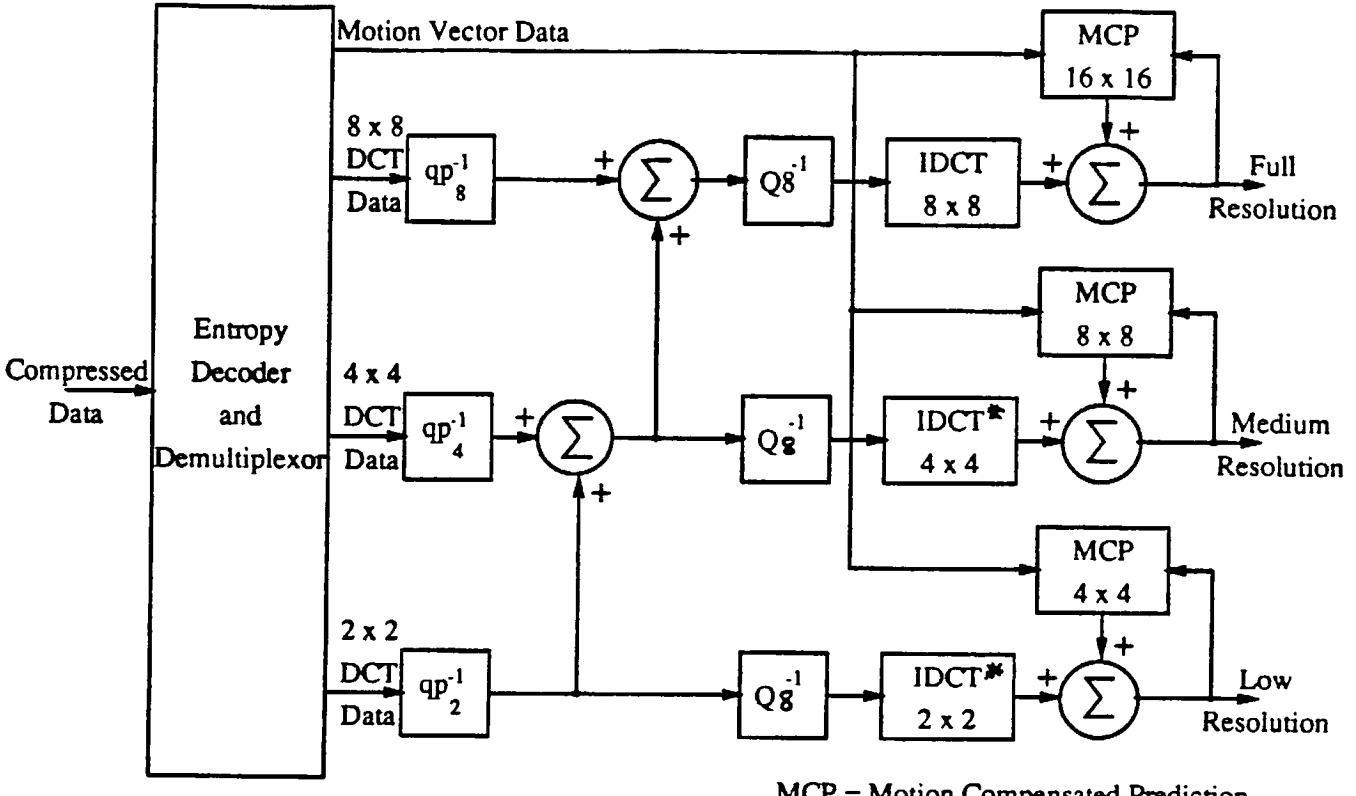

Figure 3a: A decoder with resolution scalability

Figure 4: Flexible scalable video compression encoder

Figure 5a: Transform Unit (Version A)

16

MCP = Motion Compensated Prediction
IDCT = Inverse Discrete Cosine Transform
Q_n^{-1} = Inverse Quantization at resolution scale n

Figure 5b: A decoder with bit-stream scalability

17

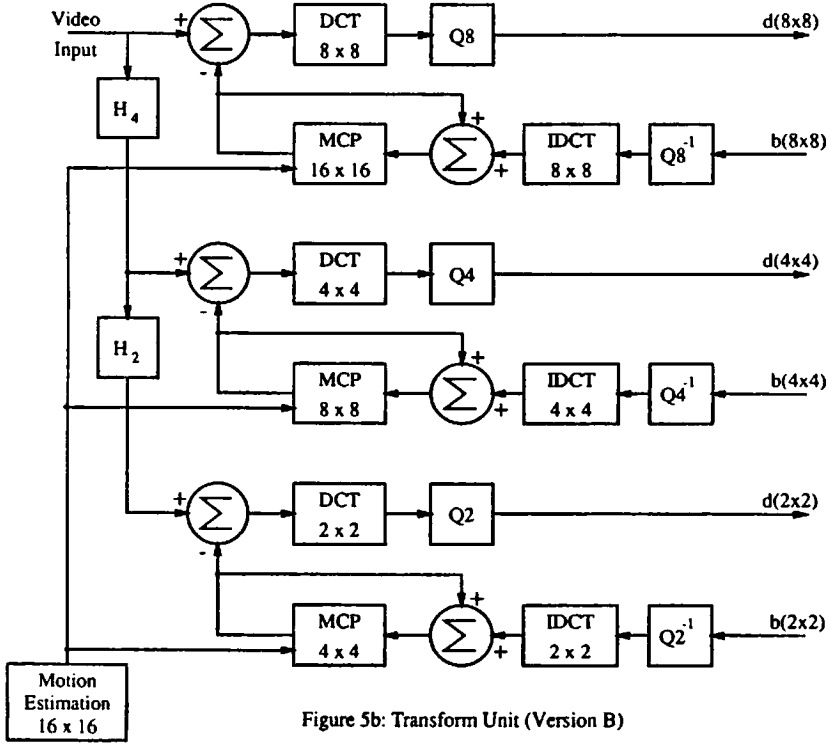

Figure 5b: Transform Unit (Version B)

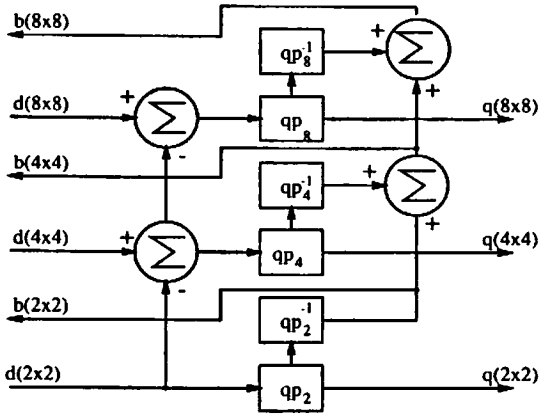

Figure 6a: Hierarchical Prediction Unit (Version A)

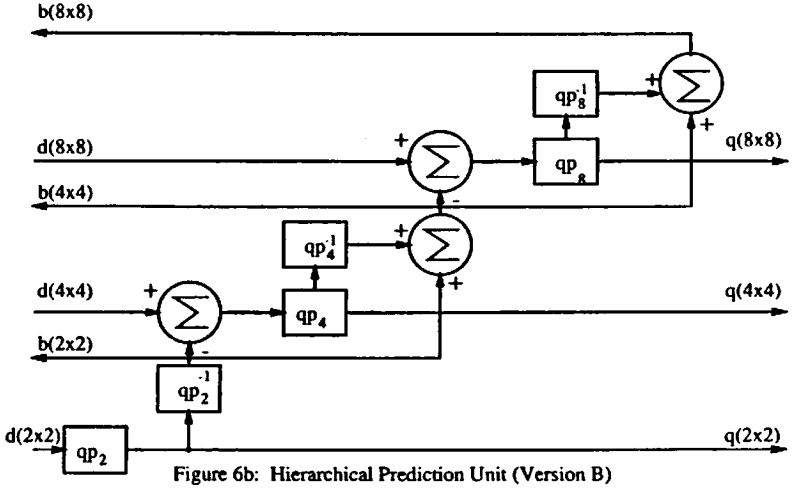

Figure 6b: Hierarchical Prediction Unit (Version B)

ANNEX 1
ADDITIONAL VLC TABLES

The VLC tables used for coding hierarchical 1x1, 2x2 and 4x4 layers are similar to the style specified in JPEG, i.e. they code run/log-size combinations. They can also be expressed compactly by specifying the array of code lenghts and the corresponding array of code symbols. To retrieve the exact amplitude additional bits are coded. The number of these additional bits is equal to the log-size value.

NOTE: THESE TABLES ARE STILL UNDER INVESTIGATION. THEY SIMPLY CORRESPOND TO THOSE USED IN OUR CURRENT SIMULATION FOR A SCALABLE DECODER.

1x1 non-intra VLC (size 8)

Not implemented at this time (expected size: 8 entries)

2x2 Scale VLC (size 33)

RUN	LOG-SIZE	LENGTH	CODE
0	1	2	01
0	2	4	0011
1	1	4	0010
0	3	5	00011
2	1	5	00010
1	2	6	000011
3	1	6	000010
0	4	7	0000011
1	3	7	0000010
2	2	7	0000001
0	5	9	000000011
1	4	9	000000010
3	2	9	000000001
2	3	10	0000000001
0	6	12	000000000011
1	5	12	000000000010
1	6	13	0000000000011
2	4	13	0000000000010
3	3	16	0000000000001111
2	5	16	0000000000001110
0	7	16	0000000000001101
0	8	16	0000000000001100
1	0	16	0000000000001011
1	7	16	0000000000001010
1	8	16	0000000000001001
2	6	16	0000000000001000
2	7	16	0000000000000111
2	8	16	0000000000000110
3	4	16	0000000000000101
3	5	16	0000000000000100
3	6	16	0000000000000011
3	7	16	0000000000000010
3	8	16	0000000000000001

EOB_SIZE= 1 EOB_CODE=1

This corresponds to the following specification to the JPEG generation algorithm:

```
Huffman code lengths
1 1 0 2 2 2 3 0
3 1 0 2 2 0 0 15

Huffman code symbols (decimal)
  1   2  17   3  33  18  49   4
 19  34   5  20  50  35   6  21
 22  36  51  37   7   8  16  23
 24  38  39  40  52  53  54  55
 56
```

Note that the normal JPEG tables have been complemented here to avoid start code emulation.

4x4 Scale VLC (size 129)

In this case, we only provide the specification for the JPEG generation algorithm:

```
Huffman code lengths
0 1 2 3 6 4 4 4
6 1 2 2 0 0 0 95

Huffman code symbols (decimal)
  1  49  17  33  81   2  50  65
 97 129 145  18  82 113 161   3
 34  51 177  19  66 193 225  52
 83  98 146 209 241 130   4  35
114  20  67 162  53  84 147  99
178  36 131 194   5 115 226  21
148 210  54  85   6 242   7   8
 16  22  23  24  37  38  39  40
 55  56  68  69  70  71  72  86
 87  88 100 101 102 103 104 116
117 118 119 120 132 133 134 135
136 149 150 151 152 163 164 165
166 167 168 179 180 181 182 183
184 195 196 197 198 199 200 211
212 213 214 215 216 227 228 229
230 231 232 243 244 245 246 247
248
```

As before, the codes are complemented with respect to normal JPEG codes to avoid start code emulation.

ANNEX 3
IMPLEMENTATION DATA

The proposal is for a flexible architecture that extends the MPEG-1 syntax and decoding methodology to support scalability by coding a variable number of hierarchical layers. As such, the complexity of decoders and encoders will depend on their specific configuration and the selected number of hierarchical layers.

We have simulated two configurations. The first configuration is a single layer, non-scalable codec for interlaced video, which we used to generate the test data for the Kurihama meeting. The second configuration implements three resolution scales as described in figure 3a. At Kurihama, we will demonstrate and compare the results of the non-scalable versus the scalable configurations.

Because the non-scalable configuration is compatible with frame based MPEG-1 coding, its decoder complexity can be evaluated by scaling the complexity of a core MPEG-1 decoder. The scalable decoder has only a relatively small increase in complexity as shown below. We use the "Paper design for SM1" in [6] appropriately updated and scaled to obtain the following results:

Decoder Computation Bandwidth

1. VBV size
   - At 4 Mbit/s: 589824 bits
   - At 9 Mbit/s: 1359872 bits
2. Frame assumptions

Frames required	2	
Frame size	495	Kbytes
Frame rate	30	
Component rate (Y+Cr+Cb)	14.5	Mbytes/s

3. Computations bandwidth
   - Operations/component

	Intra	Pred	Bidi	
Inverse Quantization (non-scalable)	2	2	2	Multiply
Inverse Quantization (3-scales)	2.31	2.31	2.31	Multiply
	0.34	0.34	0.34	Add
Addition for inter-blocks	0	1	1	Add
Half PEL filter (horizontal only)	0	1	2	Add
Interpolation	0	0	1	Add

   - Worst case computation bandwidth in MOP/s

Inverse Quantization (3-scales)	33.5	Multiply
	5	Add
Inverse DCT	87	Multiply/Accumulate
Addition for inter-blocks	14.5	Add
Half pel filter	29	Add
Interpolation	14.5	Add

   - Buffer Memory Bandwidth
     - Transfer Operations

	Intra	Pred	Bidi	
Predictor frame read		0	1.2	2.3 bytes/comp
Predictor frame write	1	1	0	bytes/comp
Display frame read	1	1	0	bytes/comp
Coded bitstream memory read (9Mbit/s)		900	500	140 Kbits/pic

     - Bandwidth

	Intra	Pred	Bidi	
Predictor frame read	0	17.4	33.4	Mbyte/s
Predictor frame write	14.5	14.5	0	Mbyte/s
Display frame read	14.5	14.5	0	Mbyte/s
Coded bitstream memory read (9Mbit/s) Mbyte/s		3.3	1.8	0.5
Total Buffer Memory Bandwidth	32	48	34	Mbyte/s

Since there are several options for implementing encoders which can vary substantially in their complexity, we have not carried out the evaluation here. Reference [6] shows, however, that motion estimation dominates the complexity of MPEG-1 encoders. We believe that practical motion estimation algorithms at CCIR 601 rates will be hierarchical (in fact our simulation encoder uses such and algorithm.)

Automatic Encoding

All sequences were processed equally. However, "Mobile and Calendar" used a different "intra" quantizer matrix. This matrix favored the low frequency coefficients more than the MPEG-1 default. In addition, we never use the default non-intra matrix.

This is: /images/stats/mega4.stats, printed on Thu Nov 07 18:40:12 EST 1991

RATE: 4 Mbit/s

			FLOWER GARDEN				TABLE TENNIS				MOBILE AND CALENDAR			
			All	Intra	Pred	Interp	All	Intra	Pred	Interp	All	Intra	Pred	Interp
Picture Count			151	13	38	100	151	13	38	100	151	13	38	100
Mean RMS for Y			10.33	8.65	10.38	10.53	7.64	7.05	7.66	7.72	11.49	11.73	11.36	11.51
Mean SNR for Y			27.90	29.42	27.84	27.73	31.07	31.73	31.01	31.01	26.98	26.75	27.05	26.99
Mean SNR for Cr			30.10	30.27	30.02	30.10	34.14	34.28	34.08	34.14	30.16	30.08	30.13	30.18
Mean SNR for Cb			29.26	29.55	29.13	29.27	34.09	34.21	34.03	34.10	29.95	29.84	29.92	29.98
M.V. of QP			12.89	12.18	8.15	14.78	7.13	7.89	6.43	7.29	11.55	11.73	8.14	12.83
M.V. of number of NonZero			3.96	9.56	6.61	2.23	4.62	9.90	5.70	3.52	4.83	10.09	5.65	3.83
M.V. of number of Zeros			12.77	14.50	16.55	11.10	19.10	17.10	18.74	19.50	16.34	11.74	17.35	16.56
MB type	Intra	Pred	Interp											
	-----	fixed	fixed	-----	-----	0	0	-----	-----	0	47	-----	-----	0	0
	I	I	I	-----	556	6	1	-----	1009	23	0	-----	591	5	0
	I+Q	I+Q	I+Q	-----	764	12	0	-----	311	23	0	-----	729	2	0
	-----	MC0	-----	-----	-----	3	-----	-----	-----	209	-----	-----	-----	6	-----
	-----	MC0+Q	-----	-----	-----	9	-----	-----	-----	3	-----	-----	-----	2	-----
	-----	MC	fwd	-----	-----	450	246	-----	-----	787	212	-----	-----	528	253
	-----	MC+Q	fwd+Q	-----	-----	740	0	-----	-----	37	0	-----	-----	676	0
	-----	MC!cbp	fwd!cbp	-----	-----	100	78	-----	-----	190	35	-----	-----	100	78
	-----	-----	bwd	-----	-----	-----	305	-----	-----	-----	382	-----	-----	-----	358
	-----	-----	bwd+Q	-----	-----	-----	0	-----	-----	-----	0	-----	-----	-----	0
	-----	-----	bwd!cbp	-----	-----	-----	119	-----	-----	-----	98	-----	-----	-----	149
	-----	-----	int	-----	-----	-----	91	-----	-----	-----	226	-----	-----	-----	53
	-----	-----	int+Q	-----	-----	-----	0	-----	-----	-----	0	-----	-----	-----	0
	-----	-----	int!cbp	-----	-----	-----	241	-----	-----	-----	206	-----	-----	-----	86
Number of Coded MB				1162	1320	1320	1081	1202	1320	1273	1160	1093	1320	1320	978
Number of Coded blocks				3063	7920	5299	1581	3230	7920	3872	2376	3071	7920	4586	1866
Number of bits	MB Attributes														
	MB_addr			1364	1320	1320	1387	1356	1320	1330	1371	1348	1320	1320	1362
	MB_type			3481	2084	4601	3237	2886	1631	2232	3298	3322	2049	4272	3126
	QP			1287	3819	3806	0	215	1554	317	2	1170	3643	3402	0
	MV			7941	0	9739	8290	7909	0	7691	9020	4677	0	5913	4815
	CBP			3374	0	5623	2958	3780	0	4814	3878	3872	0	6958	3203
	EOB			6125	15840	10598	3163	6459	15840	7744	4752	6143	15840	9171	3732
	DC_differential			4646	51232	822	44	3770	38626	1676	35	4897	55666	398	6
	Y			96239	330572	205471	24268	98476	352220	127968	54283	94233	316686	142830	46847
	Cr			2842	13945	5855	253	4069	16661	4980	2086	6313	32822	9833	1529
	Cb			4419	21420	8870	517	2914	14650	3191	1284	5704	30864	8668	1307
	Extra_data			1326	1418	1315	1318	1325	1420	1318	1316	1329	1459	1315	1317
	TOTAL			133043	441650	258020	45433	133160	443922	163261	81323	133006	460348	194081	67244

This is: /images/stats/mega9.stats, printed on Thu Nov 07 18:40:14 EST 1991

RATE: 9 Mbit/s

		FLOWER GARDEN				TABLE TENNIS				MOBILE AND CALENDAR				POPPLE			
		All	Intra	Pred	Interp	All	Intra	Pred	Interp	All	Intra	Pred	Interp	All	Intra	Pred	Interp
Picture Count		151	13	38	100	151	13	38	100	151	13	38	100	151	13	38	100
Mean RMS for Y		6.82	5.23	7.84	6.63	5.20	4.05	5.74	5.14	7.48	6.75	8.63	7.15	5.50	4.61	6.22	5.35
Mean SNR for Y		31.55	33.79	30.27	31.75	34.32	36.30	33.29	34.46	30.74	31.56	29.45	31.12	33.51	34.99	32.37	33.75
Mean SNR for Cr		30.76	31.05	30.56	30.80	34.91	35.17	34.74	34.95	31.62	31.91	31.34	31.68	34.12	34.47	33.79	34.21
Mean SNR for Cb		30.11	30.48	29.77	30.20	34.69	34.99	34.53	34.72	31.39	31.70	31.13	31.45	33.36	33.70	33.02	33.44
M.V. of QP		6.02	5.06	5.00	6.53	3.54	3.45	3.94	3.40	5.60	5.43	5.18	5.79	4.46	4.09	3.52	4.87
M.V. of number of NonZero		7.48	19.15	10.58	4.79	7.78	19.37	7.88	6.24	8.98	22.76	9.12	7.14	8.37	14.80	8.13	7.62
M.V. of number of Zeros		18.32	19.46	19.12	17.87	23.75	21.43	21.64	24.85	21.20	19.52	19.41	22.10	19.91	23.36	21.96	18.68
MB type	Intra	Pred	Interp														
	-----	fixed	fixed	-----	-----	0	0	-----	-----	0	28	-----	-----	0	0	-----	-----	0	2
	I	I	I	-----	560	6	1	-----	797	27	1	-----	591	5	0	-----	890	14	3
	I+Q	I+Q	I+Q	-----	760	11	0	-----	523	20	0	-----	729	2	0	-----	430	16	0
	-----	MC0	-----	-----	-----	3	-----	-----	-----	229	-----	-----	-----	6	-----	-----	-----	10	-----
	-----	MC0+Q	-----	-----	-----	9	-----	-----	-----	2	-----	-----	-----	2	-----	-----	-----	15	-----
	-----	MC	fwd	-----	-----	568	290	-----	-----	923	208	-----	-----	576	282	-----	-----	869	133
	-----	MC+Q	fwd+Q	-----	-----	720	0	-----	-----	33	0	-----	-----	681	0	-----	-----	392	0
	-----	MC!cbp	fwd!cbp	-----	-----	3	15	-----	-----	58	4	-----	-----	47	22	-----	-----	4	3
	-----	-----	bwd	-----	-----	-----	345	-----	-----	-----	375	-----	-----	-----	412	-----	-----	-----	149
	-----	-----	bwd+Q	-----	-----	-----	0	-----	-----	-----	0	-----	-----	-----	0	-----	-----	-----	0
	-----	-----	bwd!cbp	-----	-----	-----	21	-----	-----	-----	9	-----	-----	-----	44	-----	-----	-----	4
	-----	-----	int	-----	-----	-----	317	-----	-----	-----	643	-----	-----	-----	272	-----	-----	-----	603
	-----	-----	int+Q	-----	-----	-----	0	-----	-----	-----	0	-----	-----	-----	0	-----	-----	-----	0
	-----	-----	int!cbp	-----	-----	-----	261	-----	-----	-----	55	-----	-----	-----	168	-----	-----	-----	412
Number of Coded MB				1275	1320	1320	1252	1296	1320	1292	1295	1240	1320	1320	1200	1311	1320	1318	1308
Number of Coded blocks				4774	7920	6770	3606	5217	7920	5050	4929	4391	7920	5656	3452	4993	7920	6879	3896
Number of bits	MB Attributes																
	MB_addr			1343	1320	1320	1355	1330	1320	1327	1333	1357	1320	1320	1376	1320	1320	1320	1321
	MB_type			3576	2080	4324	3486	2910	1843	1985	3400	3524	2049	4186	3465	2950	1750	3096	3050
	QP			1259	3798	3702	0	295	2615	275	1	1176	3643	3428	0	717	2151	2114	0
	MV			9021	0	9739	9921	8817	0	7691	10390	5763	0	5913	6455	19734	0	15021	24091
	CBP			4701	0	6673	4563	5240	0	5575	5794	5200	0	7266	5090	5281	0	7163	5252
	EOB			9548	15840	13540	7212	10434	15840	10100	9857	8783	15840	11311	6904	9987	15840	13758	7793
	DC_differential			4648	51232	822	47	3770	38626	1676	35	4896	55666	398	6	4207	43405	1446	161
	Y			237193	740155	421411	101805	245806	710251	237676	188518	229051	797275	273346	138350	207313	499558	297614	135007
	Cr			12217	57536	21896	2648	11046	42850	9323	7567	19098	101803	23670	6609	23290	49926	29292	17546
	Cb			14601	65320	23355	4681	10166	45893	7294	6612	19007	105539	23115	6197	25987	52187	31888	20339
	Extra_data			1327	1425	1318	1317	1327	1422	1318	1318	1325	1414	1318	1316	1327	1417	1318	1319
	TOTAL			299434	938706	508100	137035	301140	860660	284239	234825	299180	1084548	355272	175768	302114	667554	404029	215878