

Tenth Meeting: Osaka, Japan, 16-19 May, 2000

Question: Q.15/SG16

Source: Byeungwoo Jeon
School of ECE, SKKU
300 Chunchun-dong, Jangan-gu
Suwon, Korea
Tel: +82-331-290-7144
Fax: +82-331-290-5819
Email: bjeon@yurim.skku.ac.kr

Title: Entropy Coding Efficiency of H.26L

Purpose: Information

1. Background

The entropy coding tool of the current TML2.0 [1] is the Huffman coding with a fixed single VLC table for all 10 syntax elements {MBtype, intraPredMode, MVDh, MVDv, CBPintra, CBPinter, TCOEFF_Luma_SimpleScan, TCOEFF_Luma_DoubleScan, TCOEFF_Chroma_AC, TCOEFF_Chroma_DC}. The VLC table is based on the universal VLC(UVLC) which constructs codewords by interleaving symmetric VLC code into fixed-length code(FLC) whose length is regulated properly with respect to the symmetric code [2]. The UVLC has several advantages as followings:

- easy extensibility of the VLC table using a straightforward construction rule
- no look-up-table necessary at encode and decoder
- easy decoding
- robustness to error

In spite of these outstanding features, its performance in terms of entropy coding efficiency has some space for further improvement since the codewords are not designed according to the symbol probabilities, but simply assigned according to the fixed construction rule. The actual symbol probabilities can be different for different Qp values, and also from one syntax element to another. The purpose of this document is to give information about the coding performance of the UVLC with respect to the syntax elements over various Qp values.

2. Description of Experiment 1

The purpose of the experiment is to find out how much compression performance will vary if the current single UVLC table is slightly modified to better suit the actual symbol probabilities. If the current UVLC table is good enough, there would be not significant performance gain. This experiment is performed for all Qp values from 1 to 31. We do not change the codewords themselves, however, we simply change the association between codewords and symbols. For this purpose, the TML1.0 [3] encoder encodes the test sequences below, and the generated bits are counted for each syntax elements and for each sequence. This is the number of bits which the current TML1.0 [3] coder generates and it is called S1 in eq.(1). The experiment is performed separately for each Qp from 1 to 31.

Foreman (QCIF frame rate=10)
News (QCIF frame rate=10)
Container (QCIF frame rate=10)
Silent (QCIF frame rate=15)
Mobile (CIF frame rate=30)

The bit counts are then used for estimating actual symbol probability of each syntax element. In counting the number of occurrences of symbols, 300 frames of each sequence are used. In counting, we exclude the first intra frame.

The number of symbol occurrences of each syntax element is then checked to sort the codewords in a decreasing order of their occurrences. Once the ordering is complete, codewords are re-assigned to the symbols in such a way that more frequent symbol is assigned to a shorter codeword. Subsequently, we code the symbols of each syntax element using a modified VLC code table which is a re-shuffled version of the given UVLC table. Note that the re-association just changes the mapping between codewords and symbols. This re-association is performed separately for each Qp value since we like to check the efficiency of the current VLC table with respect to Qp changes.

By coding the symbols using the modified VLC table, we can measure how much the actual statistics of each codewords are different from the statistics manifested in the given UVLC table. To quantify the change of coding efficiency, we measure a reduction ratio, denoted by R as below,

$$R \equiv \frac{S1 - S2}{S1} \quad (1)$$

where S1 is the number of bits generated by the current TML1.0 encoder, and S2 refers to the number of bits generated by the modified VLC table as described above. We also compute the coding efficiency as defined in eq.(2) for each syntax element.

$$E \equiv \frac{\text{total_symbol_count} \times \text{average_Entropy_of_given_symbol}}{S2} \quad (2)$$

Figure 1 shows a summary of coding efficiency. It suggests that a simple re-association of codewords and symbols can bring additional reduction by up to 12%. A brief overview of Fig.1 is that,

- Range 1 (Qp=1~12): 1 ~ 12 % reduction
- Range 2 (Qp=13~21): 1 ~ 2% reduction
- Range 3 (Qp=22~31): 1 ~ 6% reduction

This is a strong evidence of space existing for improving current entropy coding in TML model. That is, the current UVLC seems to match reasonably well the real statistics for Range 2 (Qp=13 ~ 21). However, in range 1 and 3, there is plenty of space for further compression possibly not sacrificing the advantages of the current UVLC scheme.

3. Description of Experiment 2

The purpose of the second experiment is to thoroughly analyze why there is such improvement as shown in Fig.1. In the first and second experiment, we have tested all 10 syntax elements {MBtype, intraPredMode, MVDh, MVDv, CBPintra, CBPinter, TCOEFF_Luma_SimpleScan, TCOEFF_Luma_DoubleScan, TCOEFF_Chroma_AC, TCOEFF_Chroma_DC} just to find out that the following three syntax elements, "MBtype, CBPinter, TCOEFF_Luma_SimpleScan" produce significantly different coding performance when the modified VLC table is used. Therefore, we show the reduction ratio R and entropy coding efficiency E only for those three in Fig.2 to 4. Table 1 and 2 show two examples of codeword-symbol assignments.

We also check how much portion each syntax element takes up among total number of bits. This data is shown in Fig.5. As expected, the TCOEFF_Luma_SimpleScan takes major portion(40% (Qp=31) ~ 80%(Qp=1)) of bit consumption irrespective of Qp. Portion of MBtype increases as Qp increases. When Qp is 31, MBtype data takes up about 20~30% of total bits. CBPinter takes about 2%(Qp=1)~12%(Qp=31).

The improvement of entropy coding using the modified VLC table is due to combined compression gain of mainly MBtype, CBPinter, and TCOEFF_Luma_SimpleScan as shown in Fig.2~4. Especially, the current entropy coding has much margin for improvement in Qp range 1 ($Qp < 13$) for all three syntax elements. Since the portion of TCOEFF_Luma_SimpleScan is very large in this range, the impact of improvement is expected to be significant. MBtype has non-significant margin for improvement even in Qp range 2 and 3.

3. Conclusion

As shown in the two experiments, there is room for improvement for entropy coding of the current H.26L TML. It is due to using a fixed single UVLC table which cannot effectively accommodate different symbol statistics with respect to various quantization step sizes of Qp. More work needs to be conducted for improvement of current entropy coding while keeping the benefit of the current UVLC scheme as much as possible.

References

- [1] Gisle Bjontegaard, "H.26L Test Model Long Term Number 1 (TML-2) draft 0," Q15-I-57d0, Red Bank, New Jersey, Oct. 1999.
- [2] Yuji Itoh, "Bi-directional motion vector coding using universal VLC," Signal Processing: Image Communication 14 (1999), pp.541-557.
- [3] Inge Lille-Langoy, "A software implementation of TML-1," Q15-I-13r0, Red Bank, New Jersey, Oct. 1999.

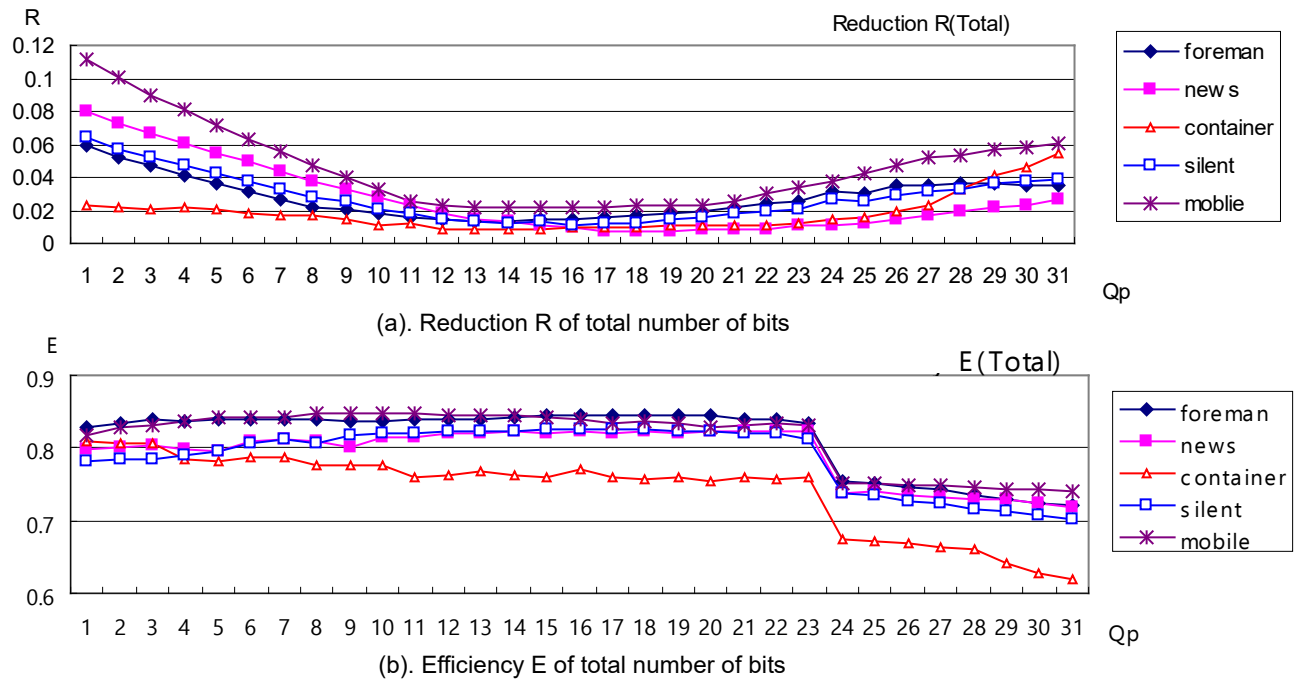


Figure 1. Reduction(R) and Efficiency(E) Comparison of Total number of Bits

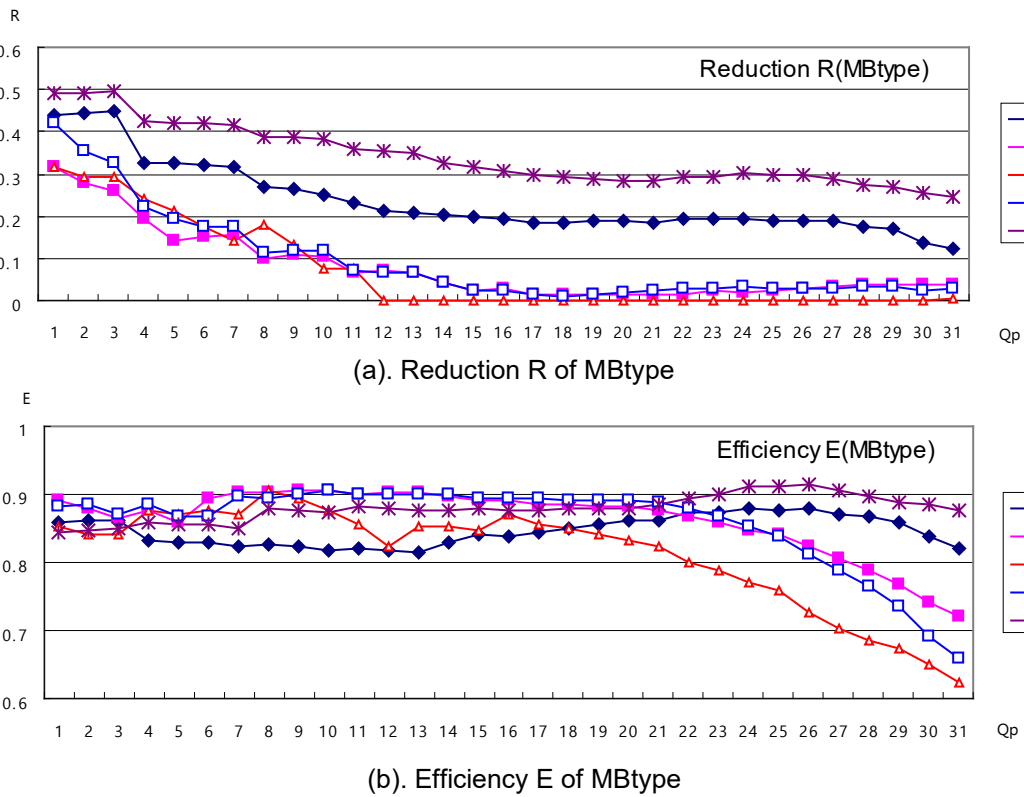


Figure 2. Reduction(R) and Efficiency(E) Comparison of "MBtype"

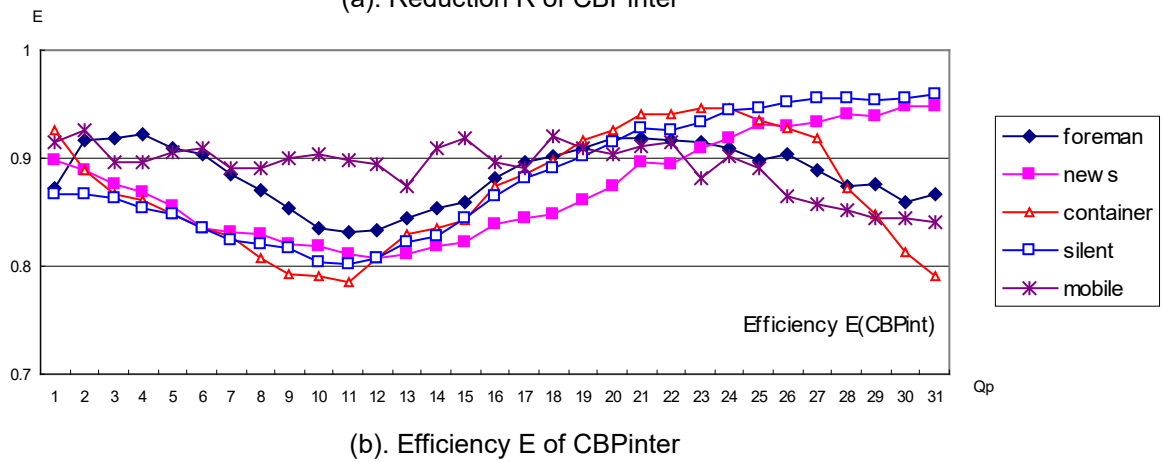
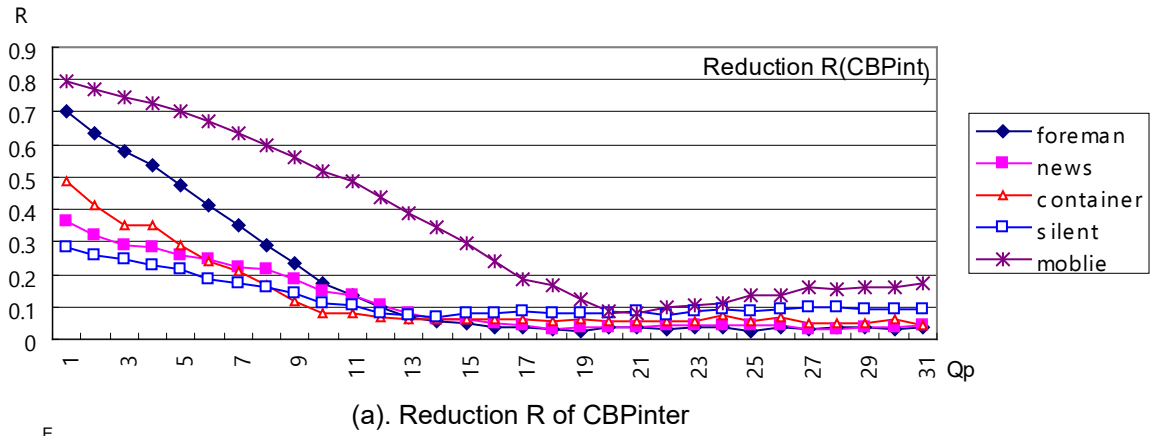


Figure 3. Reduction(R) and Efficiency(E) Comparison of “CBPinter”

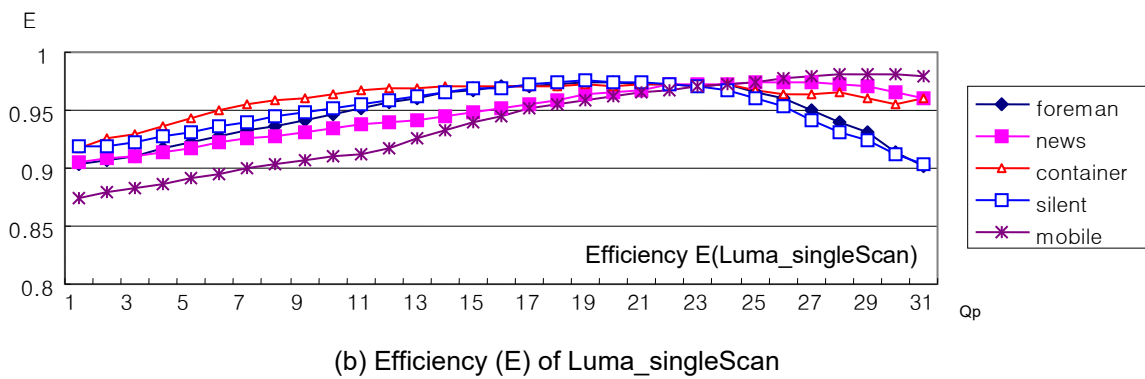
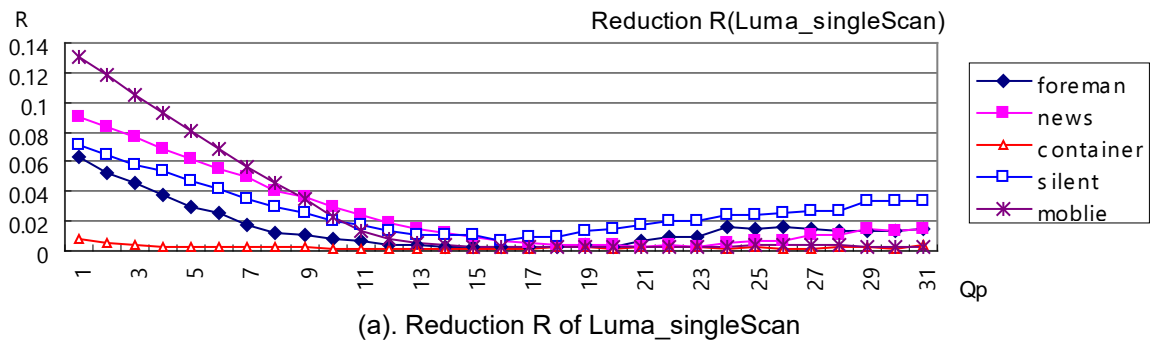


Figure 4. Reduction(R) and Efficiency(E) Comparison of “Luma_singleScan”

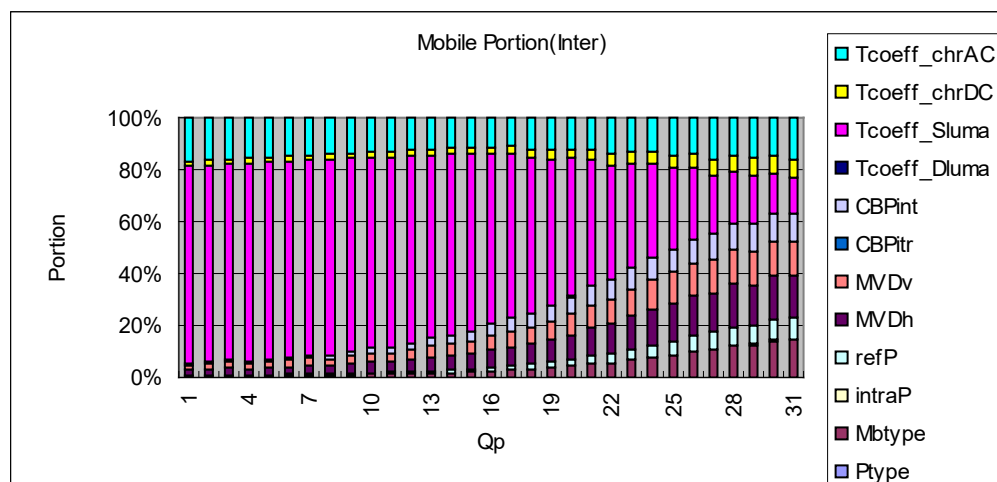
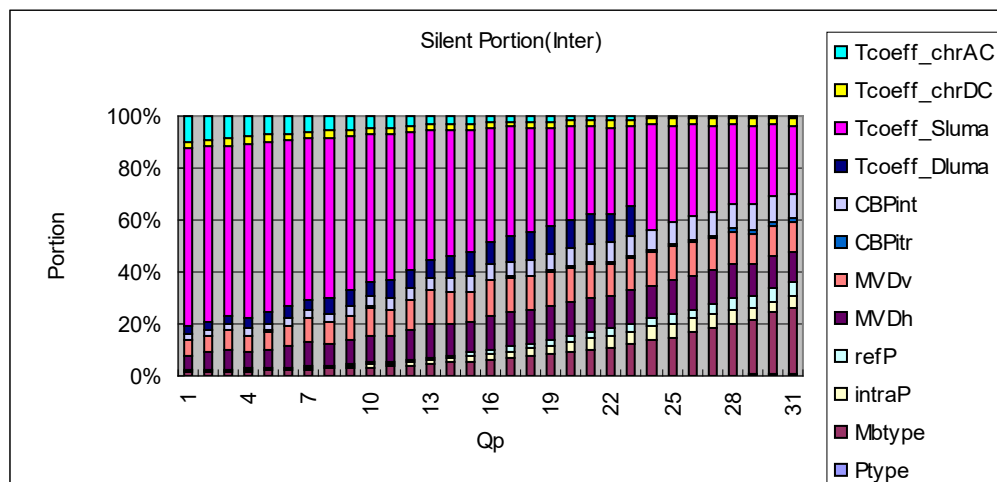
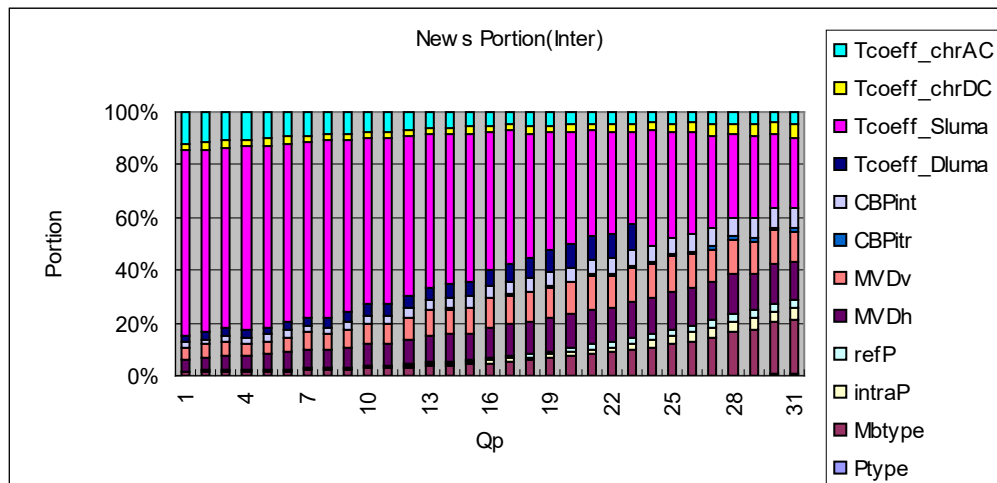


Figure 5. Relative portion of each syntax element

Table 1. Example of Codeword-Symbol Assignments (News Sequence)¹

Code number	MB_Type				CBPinter			
	UVLC	Qp=1	Qp=15	Qp=28	UVLC	Qp=1	Qp=15	Qp=28
0	skip(1)	16X16(3)	skip(1)	skip(1)	0(1)	47(7)	0(1)	0(1)
1	16X16(3)	4X4(7)	16X16(3)	16X16(3)	16(3)	16(3)	8(5)	8(5)
2	16X8(3)	skip(1)	4X8(5)	8X16(5)	1(3)	32(5)	47(7)	16(3)
3	8X16(5)	8X4(5)	4X4(7)	16X8(3)	2(5)	46(11)	10(7)	4(5)
4	8X8(5)	4X8(5)	8X16(5)	intra(7)	4(5)	45(9)	4(5)	1(3)
5	8X4(5)	8X8(5)	8X4(5)	4X8(5)	8(5)	42(9)	2(5)	32(5)
6	4X8(5)	16X8(3)	16X8(3)	8X8(5)	32(5)	37(9)	15(7)	2(5)
7	4X4(7)	8X16(5)	8X8(5)	8X4(5)	3(7)	11(7)	12(7)	10(7)
8		intra(7)	intra(7)	4X4(7)	5(7)	43(9)	5(7)	12(7)
9	intra(7)				10(7)	10(7)	1(3)	36(9)
10					12(7)	44(9)	44(9)	40(9)

Table 2. Example of Codeword-Symbol Assignments (Mobile Sequence)¹

Code number	MB_Type				CBPinter			
	UVLC	Qp=1	Qp=15	Qp=28	UVLC	Qp=1	Qp=15	Qp=28
0	skip(1)	4X4(7)	16X16(3)	16X16(3)	0(1)	47(7)	32(7)	0(1)
1	16X16(3)	16X16(3)	8X4(5)	8X16(5)	16(3)	46(11)	15(5)	32(5)
2	16X8(3)	8X4(5)	4X8(5)	16X8(3)	1(3)	45(7)	47(7)	16(3)
3	8X16(5)	4X8(5)	4X4(7)	skip(1)	2(5)	43(7)	39(9)	1(3)
4	8X8(5)	8X8(5)	8X16(5)	4X8(5)	4(5)	39(9)	43(9)	2(5)
5	8X4(5)	8X16(5)	8X8(5)	8X8(5)	8(5)	42(9)	45(9)	4(5)
6	4X8(5)	16X8(3)	16X8(3)	8X4(5)	32(5)	37(9)	46(11)	8(5)
7	4X4(7)	intra(7)	skip(1)	4X4(7)	3(7)	44(9)	44(9)	33(9)
8		skip(1)	intra(7)	intra(7)	5(7)	32(5)	35(9)	36(9)
9	intra(7)				10(7)	15(7)	37(9)	37(9)
10					12(7)	34(9)	42(9)	3(7)

¹ The column UVLC shows the symbol mapping as defined in current TML. The columns denoted by Qp=1, 15, and 28 show the modified mappings as described in experiment 1 and 2. The number in parenthesis is the length of the assigned codeword (which is shown in front of the parenthesis).