Joint Collaborative Team on Video Coding (JCT-VC)
of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11
7th Meeting: Geneva, 21-30 November, 2011

Document: JCTVC-G657

Title:	**Encoding and decoding significant coefficient flags for small Transform Units using partition sets**
Status:	Input Document to JCT-VC
Purpose:	Proposal

Author(s) or Contact(s): Gergely Korodi, Jinwen Zan, Da-ke He
295 Phillip Street
Waterloo, Ontario, Canada N2L 3W8
Tel: +1 519 888 7465
Email: {gkorodi, jzan, dhe}@rim.com

Source: Research In Motion Limited

Abstract

The current Working Draft of the HEVC video encoding standard encodes the significance flags of transform coefficients by assigning them to binary contexts based on the location of the coefficients in the Transform Unit. As of HM-4.0, a total of 62 binary contexts is allocated to the significance flags of 4x4 and 8x8 TUs: 15 for 4x4 luma, 15 for 4x4 chroma, 16 for 8x8 luma and 16 for 8x8 chroma. The present contribution introduces a new context allocation scheme, which reduces the overall number of contexts used for 4x4 and 8x8 significance maps (except for the last significant coefficients) by half to 31. The contribution of these contexts to the size of the encoded sequence is more balanced. New initialization values, which do not depend on QP, are given for the contexts. The proposed implementation matches the performance of HM-4.0 within a 0.1% accuracy of the BD-rate.

1 Introduction

An efficient model of a practical lossless compression system needs to find a balance among several constraints, namely complexity, accuracy, and adaptivity. This proposal provides a new context model to the significance map coding for 4x4 and 8x8 TUs, which model has been developed by balancing these constraints. Particularly, the goal was not to remove contexts based purely on the frequency of their usage, but rather to consolidate contexts with similar behavioral patterns. Following these guidelines, the proposal reduces the number of contexts used from 62 to 31.

To formalize the problem statement, let n be the size of the Transform Unit, which may be 4 and 8 for both luma and chroma. The objective is to losslessly encode the significant flags in the TU, except for the last one, which is already known. The last significant flag is determined by a known scan order, which may be diagonal, horizontal, or vertical. Based on this order, the last significant flag and all the zeros beyond it are removed from the TU, and the rest are processed in reversed scan order. To rephrase this task, we need to losslessly encode all 0 and 1 symbols of an input matrix M from $\{0, 1, X\}^{n*n}$, where the additional X serves as a placeholder for the omitted flags, which do not need to be encoded. The matrix M is associated with certain parameters, which are the slice type (intra or inter), text type (luma or chroma), and the QP value. The same values of these parameters are known to the decoder at the time of processing M, and may affect how M is encoded and decoded.

The outline of this proposal is as follows. Section 2 details the proposed solution, Section 3 the results on the HEVC test sequences and discussions on the principles that guided the model construction. Section 4 details the change requests.

2 Proposed solution

Each element of M, M[i][j], is encoded and decoded using one BAC context, after which that context is updated using the BAC state transitions and the value of M[i][j]. The context selected for M[i][j] is determined from the indices i and j.

2.1 *Context selection*

Let m be a positive integer. We call a mapping of the form P: {0, ..., n-1}x{0, ..., n-1} → {0, …, m-1} as a partition set; the numbers 0, …, m-1 identify different partitions. Each partition has one designated BAC context associated with it. This context is used exclusively for that partition. (Please note: our definition of partitions differs from the one used in combinatorics.)

For any two partition sets P and Q, if there is a mapping T such that T(P(i,j)) = Q(i,j) for all i and j, then we say that Q is a subset of P, or P is a refinement of Q.

Encoding works as follows: the TU of size nxn is assigned with a partition set P. The significant flags of the input matrix M are encoded in reversed (horizontal, vertical or diagonal) scan order $M[i_0][j_0]$, $M[i_1][j_1]$, …, M[0][0]. $M[i_k][j_k]$ is encoded in the BAC context corresponding to $P(i_k, j_k)$, and that context is updated using $M[i_k][j_k]$. Decoding is derived from the encoding procedure in a straightforward way.

We can use this framework to describe the current significant flags coding scheme in HM-4.0. Each of the 4x4 and 8x8 TUs is associated with a separate partition set, called P4 and P8, respectively. These are given as:

P4(i, j) = 4*i + j   i,j = 0, 1, 2, 3

[15 contexts total]

P8(i, j) = 4*[i/2] + [j/2]   i,j = 0, 1, 2, 3, 4, 5, 6, 7

[16 contexts total]

The same mappings are used for luma and chroma, but the contexts for luma and chroma are separate. Therefore, the total number of used contexts for these TUs is 15 + 15 + 16 + 16 = 62.

2.2 *New partitions*

In this section we present new 4x4 and 8x8 partition sets to be used for significance map coding. For a partition set of TU size nxn containing m partitions, we introduce the notation Pn-m. We propose the following sets:

```
P4-6: { 0, 1, 2, 4,
        1, 1, 2, 4,
        3, 3, 5, 5,
        4, 4, 5     },


P4-9: { 0, 1, 4, 5,
        2, 3, 4, 5,
        6, 6, 8, 8,
        7, 7, 8     },


P8-4: {  0,  0,  1,  1,  2,  2,  3,  3,
         0,  0,  1,  1,  2,  2,  3,  3,
         1,  1,  1,  1,  2,  2,  3,  3,
```

```
         1,  1,  1,  1,  2,  2,  3,  3,
         2,  2,  2,  2,  1,  2,  3,  3,
         2,  2,  2,  2,  2,  3,  3,  3,
         3,  3,  3,  3,  3,  3,  3,  3,
         3,  3,  3,  3,  3,  3,  3      },

P8-12: {  0,  1,  2,  2,  3,  3,  4,  4,
          1,  1,  2,  2,  3,  3,  4,  4,
          5,  5,  6,  6,  7,  7,  4,  4,
          5,  5,  6,  6,  7,  7,  4,  4,
          8,  8,  9,  9,  6,  7, 10, 10,
          8,  8,  9,  9,  9, 10, 10, 10,
          11, 11, 11, 11, 10, 10, 10, 10,
          11, 11, 11, 11, 10, 10, 10      }.
```

Note that P4-9 is a refinement of P4-6, and P8-12 is a refinement of P8-4. The partition sets are used for their respective TU sizes: P4-9 is for 4x4 luma, P4-6 is for 4x4 chroma, P8-12 is for 8x8 luma and P8-4 is for 8x8 chroma. The coarser partition sets for chroma is justified by the less impact of chroma values in the entire encoded sequence. The sets were designed in such a way that they provide good coding efficiency for both 4:2:0 and 4:4:4 modes, hence this set assignment is used for all configurations.

The special structure of the 4x4 partitions enables their compact expression using logic operations, avoiding the use of table look-ups. The following code returns the value of P4-9 at position (x, y):

```
if (x < 2) {
    if (y < 2)  return 2 * y + x;
    else return y + 4;
} else {
    if (y < 2) return x + 2;
    else return 8;
}
```

2.3 Partition initialization

Since each partition identifies a BAC state, which is used for encoding and decoding the bits in that partition, at the beginning of each slice the initial value of that state needs to be determined. The initial value is a BAC state, which in HM-4.0 terminology is an integer value in the interval {1, ..., 126}. The least significant bit of this value specifies the MPS, and the remaining 6 bits identify the probability of the LPS. The uniform state with MPS=0 and p(LPS)=0.5 is identified by the value 0.

Since the presented partition sets are highly adaptive, practical tests demonstrated that they perform most of the time with no more than 0.1% BD-rate loss as compared to HM-4.0, even if all the 31 contexts are initialized to the uniform state at the beginning of each slice. However, in line with the other context models used in HEVC, we do provide initialization values for the contexts in Section 5. Note that for added simplicity and to reduce the risk of over-training, we have designed the init values to be independent of QP (the linear coefficient as the function of QP is always 0).

2.4 *Low-description-cost variant*

To reduce the description cost of our proposal, we have considered using P4-9 for both 4x4 luma and chroma TUs, and the following partition set for both 8x8 luma and chroma TUs:

```
P8-10: {  0,  1,  2,  2,  3,  3,  4,  4,
          1,  1,  2,  2,  3,  3,  4,  4,
          5,  5,  6,  6,  3,  3,  4,  4,
          5,  5,  6,  6,  3,  3,  4,  4,
          7,  7,  7,  7,  8,  8,  8,  8,
          7,  7,  7,  7,  8,  8,  8,  8,
          9,  9,  9,  9,  8,  8,  8,  8,
          9,  9,  9,  9,  8,  8,  8      }.
```

P8-10 is essentially the upsampled version of P4-9, with the DC component separated in its own partition. Hence P8-10 can be easily expressed via P4-9, for example, CTX_IND_MAP_8x8[n] == n ? CTX_IND_MAP_4x4[((n>>4)<<2) + ((n&7)>>1)] + 1 : 0. In this variant we also use the HM-4.0 initialization values for the new contexts. The following arrays show how the HM-4.0 init values, corresponding to their position in the TU, are selected to initialize the proposed contexts marked by their indices:

```
4x4: { 0,  1,  2,  3,
       4,  5,  x,  x,
       6,  x,  7,  x,
       8,  x,  x    }
8x8: {0/1, 2,  3,  4,
       5,  6,  x,  x,
       7,  x,  8,  x,
       9,  x,  x,  x, }
```

In the 8x8 case 0/1 indicates that those contexts share the same HM-4.0 init value, since they are mapped to the same partition.

3 Results and discussions

The following tables show the BD-rate changes of this proposal on all of the JCT-VC HE test configurations [3], compared to HM-4.0. A negative value means a gain in BD-rate. In accordance to Section 3.2, we used P4-9 for 4x4 luma, P4-6 for 4x4 chroma, P8-12 for 8x8 luma and P8-4 for 8x8 chroma. In all tests the partitions were initialized to the values given in Section 5.

	All Intra HE		
	Y	U	V
Class A	0.0073%	-0.0178%	0.0030%
Class B	0.0327%	-0.0292%	-0.0192%
Class C	0.0009%	0.0013%	-0.0093%
Class D	0.0253%	-0.1486%	-0.0323%
Class E	0.0159%	-0.0197%	-0.1348%
Class F			
Overall	0.0173%	-0.0433%	-0.0328%
	0.0168%	-0.0434%	-0.0338%
Enc Time[%]	100%		
Dec Time[%]	100%		

	Low delay B HE		
	Y	U	V
Class A			
Class B	0.0446%	-0.4918%	-0.4922%
Class C	0.0184%	-0.5704%	-0.2322%
Class D	0.0816%	-0.6219%	-0.5706%
Class E	-0.0248%	0.0188%	0.1583%
Class F			
Overall	0.0343%	-0.4483%	-0.3248%
	0.0335%	-0.4422%	-0.3818%
Enc Time[%]	100%		
Dec Time[%]	100%		

	Random Access HE		
	Y	U	V
Class A	0.0531%	-0.0425%	-0.0873%
Class B	0.0221%	-0.0802%	-0.1535%
Class C	0.0275%	-0.2331%	-0.0475%
Class D	0.0808%	-0.3340%	-0.1135%
Class E			
Class F			
Overall	0.0445%	-0.1670%	-0.1036%
	0.0466%	-0.2003%	-0.1136%
Enc Time[%]	100%		
Dec Time[%]	100%		

	Low delay P HE		
	Y	U	V
Class A			
Class B	0.0325%	-0.3787%	-0.1939%
Class C	0.0393%	-0.3240%	-0.1517%
Class D	0.0857%	-1.2279%	-0.8639%
Class E	0.0323%	-0.2638%	-0.5085%
Class F			
Overall	0.0475%	-0.5558%	-0.4098%
	0.0434%	-0.5725%	-0.4477%
Enc Time[%]	101%		
Dec Time[%]	100%		

When class F is also added, the results become the following:

	All Intra HE		
	Y	U	V
Class A	0.0073%	-0.0178%	0.0030%
Class B	0.0327%	-0.0292%	-0.0192%
Class C	0.0009%	0.0013%	-0.0093%
Class D	0.0253%	-0.1486%	-0.0323%
Class E	0.0159%	-0.0197%	-0.1348%
Class F	-0.0337%	0.0453%	-0.0021%
Overall	0.0088%	-0.0285%	-0.0276%
	0.0087%	-0.0338%	-0.0367%
Enc Time[%]	100%		
Dec Time[%]	100%		

	Low delay B HE		
	Y	U	V
Class A			
Class B	0.0446%	-0.4918%	-0.4922%
Class C	0.0184%	-0.5704%	-0.2322%
Class D	0.0816%	-0.6219%	-0.5706%
Class E	-0.0248%	0.0188%	0.1583%
Class F	-0.2187%	-0.3779%	-0.0577%
Overall	-0.0163%	-0.4342%	-0.2714%
	-0.0178%	-0.4412%	-0.3201%
Enc Time[%]	100%		
Dec Time[%]	100%		

	Random Access HE		
	Y	U	V
Class A	0.0531%	-0.0425%	-0.0873%
Class B	0.0221%	-0.0802%	-0.1535%
Class C	0.0275%	-0.2331%	-0.0475%
Class D	0.0808%	-0.3340%	-0.1135%
Class E			
Class F	-0.0091%	0.0724%	-0.0060%
Overall	0.0343%	-0.1214%	-0.0850%
	0.0364%	-0.1454%	-0.0932%
Enc Time[%]	100%		
Dec Time[%]	100%		

	Low delay P HE		
	Y	U	V
Class A			
Class B	0.0325%	-0.3787%	-0.1939%
Class C	0.0393%	-0.3240%	-0.1517%
Class D	0.0857%	-1.2279%	-0.8639%
Class E	0.0323%	-0.2638%	-0.5085%
Class F	-0.1419%	-0.2520%	0.1943%
Overall	0.0096%	-0.4950%	-0.2890%
	0.0062%	-0.4977%	-0.3431%
Enc Time[%]	100%		
Dec Time[%]	101%		

The following tables show the BD-rate changes of this proposal compared to HM-4.0, with RDOQ turned off:

	All Intra HE		
	Y	U	V
Class A	-0.0046%	0.0693%	0.0050%
Class B	-0.0158%	0.0008%	0.0355%
Class C	-0.0715%	-0.0640%	-0.0129%
Class D	-0.1104%	-0.1068%	-0.0900%
Class E	-0.0316%	0.0064%	-0.0259%
Overall	-0.0460%	-0.0191%	-0.0146%

	Random Access HE		
	Y	U	V
Class A	0.0182%	-0.2259%	-0.0242%
Class B	0.0255%	-0.2249%	0.0186%
Class C	-0.0430%	-0.0297%	-0.0420%
Class D	-0.0485%	-0.0910%	-0.2906%
Overall	-0.0098%	-0.1477%	-0.0785%

	Low delay B HE		
	Y	U	V
Class B	-0.0089%	-0.6499%	-0.3846%
Class C	-0.0528%	-0.6362%	-0.4222%
Class D	0.0551%	-1.7431%	-1.4027%
Class E	-0.1109%	-0.2388%	-0.6388%
Overall	-0.0230%	-0.8427%	-0.6962%

The following tables show the performance of the low-description cost variant given in Section 2.4:

	All Intra HE		
	Y	U	V
Class A	0.0006%	-0.0783%	-0.0484%
Class B	0.0411%	-0.0257%	-0.0459%
Class C	-0.0022%	0.0380%	-0.0146%
Class D	-0.0125%	-0.0629%	-0.0009%
Class E	0.0049%	-0.0018%	-0.0058%
Overall	0.0082%	-0.0273%	-0.0251%

	Low delay B HE		
	Y	U	V
Class A			
Class B	0.0179%	-0.0090%	-0.3364%
Class C	-0.0189%	-0.0370%	-0.1466%
Class D	0.0250%	0.4752%	0.0808%
Class E	0.0664%	-0.3694%	0.7410%
Overall	0.0196%	0.0375%	0.0174%

	Random Access HE		
	Y	U	V
Class A	0.0152%	-0.0967%	-0.2010%
Class B	-0.0037%	0.0789%	-0.0663%
Class C	-0.0013%	-0.0795%	0.0516%
Class D	0.0222%	0.1633%	-0.2873%
Class E			
Overall	0.0074%	0.0202%	-0.1223%

As outlined in Section 1, the guiding principle of the model construction was to find a good balance between model complexity, accuracy, and adaptivity. Complexity describes the number of operations the model needs to carry out for a specific task, and also the size of the model expressed in a relevant unit. For the significance map coding in HM-4.0, this unit is the binary context. Models using more contexts tend to be either slower, more difficult, or more expensive to implement, hence the use of small models is preferred to large ones from this perspective.

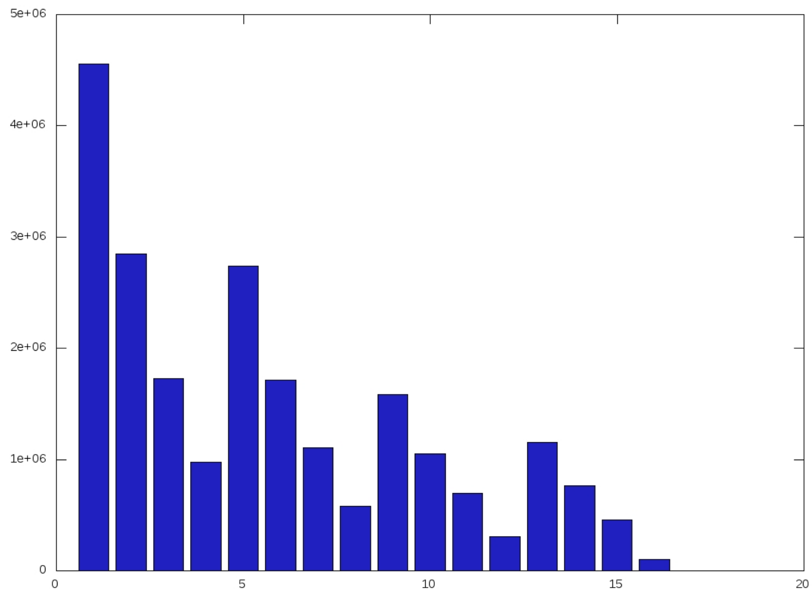
Accuracy describes how closely the states of the model can approximate the probability values of the true model, which was used to generate the data; in this case, the significance map. Accurate representation reduces the relative entropy between the current and the true model, which in turn leads to better compression performance, provided we have enough data to process. For significance map coding the true model is not known; for 4x4 and 8x8 TUs HM-4.0 approximates it by assuming that significant flags are the outcomes of independent and identically distributed processes assigned to each position of the TU. If this assumption was true, the most accurate model would use 15 contexts for 4x4 TUs, and 63 contexts for 8x8 TUs (this number is one less than the number of positions in the TU, since at the time of decoding the flags, the position of the last significant coefficient is already known). Based on this assumption, any model using fewer contexts trades compression efficiency for less complexity.

Accuracy is an asymptotic concept, which is relevant in practice only when the same model is used for a large amount of data. In the practice of significance map coding the data is limited, since models need to start from scratch at the beginning of each slice, and they forget everything they learned at the end of the slice. Even within the slice, when the approximation of the true model is not good, as it often happens in practice, the current model views that as if the model parameters are constantly changing. When the model needs to adjust its parameters to be in line with the observed data, the relative entropy increases and the coding efficiency drops. This stage is called the "learning process" of the model, and for better efficiency, it is best kept as short as possible. How fast the model can learn the changing statistics is called the adaptivity of the model. The more adaptive the model, the faster it can converge to its best representation of the true statistics, hence it achieves its best performance for a larger part of the data, resulting in better compression efficiency. As a rule of thumb, well-designed complex models tend to be more accurate, but at the same time less adaptive, than smaller models. Finding a good trade-off between complexity, accuracy and adaptivity thus leads to an important optimization problem.

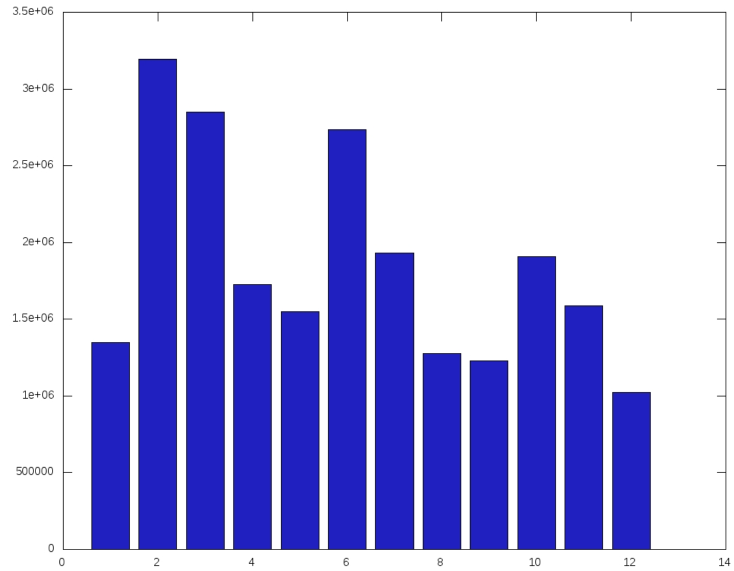
HM-4.0 uses the maximum number of 15 contexts for both luma and chroma 4x4 TUs, resulting in maximum accuracy, but also highest model cost, and slowest adaptivity. To address slow convergence, at the beginning of each slice the model is initialized with default parameters, which aim to provide an accurate representation for a large class of video sequences. However, relying on the efficiency of such values makes the model susceptible for over-training, and increases the worst-case expansion on different sequences. A better solution is to design more adaptive models. A common way to achieve this is via information sharing: contexts considered relevant to each other may adjust their parameters based on the states of their counterparts. To simplify this scheme to be practical for HEVC, positions in the TU are grouped together in partitions, based on the relative entropy among the distributions formed at each position, and the proportion that each position contributes to the overall encoded size. Complexity is decreased, and accuracy is left mostly unaffected by joining together rarely used positions. Accuracy is preserved, adaptivity is increased and complexity is decreased by joining together positions with similar distributions. An analysis carried out on practical sequences has revealed that similar positions may be separated far from each other, with irrelevant positions in between them; this manifests most conceivably from a symmetry on the main diagonal. However, in general the positions in the TU may be grouped together in arbitrary partitions, and still have the potential to offer practical benefits. As such, in the development of this contribution we have considered all combinations for partition sets.

To understand the complexity of this task, we remark that the number of essentially unique partition sets for any given TU size n and partition count m is given by the Stirling numbers of the second kind. Based on that formula, the total number of partition sets for 4x4 TUs, that is, 15 coefficients, is 1,382,958,545; the number of partition sets having exactly 5 partitions is 210,766,920, and those having exactly 10 partitions are 12,662,650. The corresponding numbers for 8x8 TUs (63 coefficients) are better expressed in exponential form: the total number of different partition sets is $8.2507717*10^{63}$, the number of sets having no more than 16 partitions is $3.5599620*10^{62}$, the number of sets having exactly 5 partitions is $9.0349827*10^{41}$, and those having exactly 10 partitions are $2.7197285*10^{56}$. Since any of these form legit partition sets for video compression, selecting the best ones from so many candidates is a complex task.

In HM-4.0 using the same context for several positions is already done for 8x8 TUs, where each context covers a 2x2 block of positions in a uniform grid, reducing the number of contexts from the maximum 63 to 16 per text type. However, this solution can be improved from the perspective of complexity. Whenever certain components of the model are used less frequently and contribute only to a small part of the encoded sequence, the model can be simplified by omitting those components, resulting in the smallest reduction in compression efficiency. As the following image shows, such is the case for the high-frequency coefficients of the 8x8 TUs. This image shows the counts of the number of encoded bits output at each context of the significance map, normalized with the total sum for all positions, for all Class C random access sequences. The contexts are listed in horizontal scan order.



The significantly lower values for the high-frequency coefficients indicate that their associated contexts are underused compared to the low-frequency contexts, although their contribution to the increasing model cost is the same. From the perspective of complexity alone, the normalized counts for the optimal model are uniform; when compression efficiency is also taken into account, the curve needs to be adjusted based on the relative entropy between positions that can be grouped in the same context. As such, the resulting curve will never be flat, but a smaller variance indicates a better cost-distribution from the point of view of complexity. For comparison, the next image shows the same normalized encoded bit counts for the partition set P8-12 (note the reduced number of contexts):



4 Specification

This section describes the change request of the current proposal with regard to the Working Draft 4 of HEVC [2]. New additions are highlighted in yellow, removed text is ~~struck out~~.

4.1 Proposed changes to "9.3.3.1.1.4 Derivation process of ctxIdxInc for the syntax element significant_coeff_flag"

Inputs to this process are the color component index cIdx, the current coefficient scan position (xC , yC) and the transform block size log2TrafoSize.

Output of this process is ctxIdxInc.

The variable sigCtx depends on the current position (xC, yC), the color component index cIdx, the transform block size and previously decoded bins of the syntax element significant_coeff_flag. For the derivation of sigCtx, the following applies.

– ~~If log2TrafoSize is less than or equal to 3, sigCtx is derived as follows.~~

– ~~shift = log2TrafoSize = = 3 ? 1 : 0~~
~~sigCtx = (shift * 15) + ((yC >> shift) << 2) + (xC >> shift) (9-1)~~

– If log2TrafoSize equals to 2, sigCtx is derived as follows.

sigCtx = CTX_IND_MAP_4x4[cIdx][(yC << 2) + xC] (9-2)

where CTX_IND_MAP_4x4 is defined as

```
static const UInt CTX_IND_MAP4x4[2][15] = {
    {   0, 1, 2, 3,    4, 5, 2, 3,    6, 6, 7, 7,    8, 8, 7,    },
    {   0, 1, 2, 3,    1, 1, 2, 3,    4, 4, 5, 5,    3, 3, 5     }
};
```

– Otherwise if log2TrafoSize equals to 3, sigCtx is derived as follows.

sigCtx = CTX_IND_MAP_8x8[cIdx][(yC << 3) + xC] (9-3)

where CTX_IND_MAP_8x8 is defined as

```
static const UInt CTX_IND_MAP8x8[2][63] = {
    {   0,  1,  2,  2,  3,  3,  4,  4,          1,  1,  2,  2,  3,  3,  4,  4,
        5,  5,  6,  6,  7,  7,  4,  4,          5,  5,  6,  6,  7,  7,  4,  4,
        8,  8,  9,  9,  6,  7, 10, 10,          8,  8,  9,  9,  9, 10, 10, 10,
       11, 11, 11, 11, 10, 10, 10, 10,         11, 11, 11, 11, 10, 10, 10      },
    {   0,  0,  1,  1,  2,  2,  3,  3,          0,  0,  1,  1,  2,  2,  3,  3,
        1,  1,  1,  1,  2,  2,  3,  3,          1,  1,  1,  1,  2,  2,  3,  3,
        2,  2,  2,  2,  1,  2,  3,  3,          2,  2,  2,  2,  2,  3,  3,  3,
        3,  3,  3,  3,  3,  3,  3,  3,          3,  3,  3,  3,  3,  3,  3   }
};
```

– Otherwise if xC + yC is less than 2, sigCtx is derived as follows.

sigCtx = 31 + (yC << 1) + xC (9-4)

– Otherwise if xC + yC is less than 5, sigCtx is derived as follows.

temp = significant_coeff_flag[xC + 1][yC] + significant_coeff_flag[xC + 2][yC] +
significant_coeff_flag[xC][yC + 1] + significant_coeff_flag[xC + 1][yC + 1] +
significant_coeff_flag[xC][yC + 2] (9-5)
sigCtx = 34 + Min(4, temp)

– Otherwise (xC + yC is greater than 4), sigCtx is derived using previously decoded bins of the syntax element significant_coeff_flag as follows.

  – The variable sigCtx is initialized as follows.

    sigCtx = 39 (9-6)

  – When xC is less than (1 << log2TrafoSize) − 1, the following applies.

    sigCtx = sigCtx + significant_coeff_flag[xC + 1][yC] (9-7)

  – When xC and yC are less than (1 << log2TrafoSize) − 1, the following applies.

    sigCtx = sigCtx + significant_coeff_flag[xC + 1][yC + 1] (9-8)

  – When xC is less than (1 << log2TrafoSize) − 2, the following applies.

    sigCtx = sigCtx + significant_coeff_flag[xC + 2][yC] (9-9)

  – When yC is less than (1 << log2TrafoSize) − 1, the following applies.

    sigCtx = sigCtx + significant_coeff_flag[xC][yC + 1] (9-10)

  – When yC is less than (1 << log2TrafoSize) − 2 and sigCtx is less than 43, the following applies.

    sigCtx = sigCtx + significant_coeff_flag[xC][yC + 2] (9-11)

The context index increment ctxIdxInc is derived using the color component index cIdx, the transform block size log2TrafoSize, sigCtx and the partition sets as follows.

- ~~If cIdx is equal to 0, ctxIdxInc is derived as follows.~~
- ~~ctxIdxInc = sigCtx (9-~~12~~)~~
- ~~Otherwise (cIdx is greater than 0), ctxIdxInc is derived as follows.~~
- ~~ctxIdxInc = 44 + sigCtx (9-~~13~~)~~

ctxIdxInc = ctxOffset[max(log2TrafoSize-2, 2)][cIdx] + sigCtx

where the table ctxOffset is given by

max(log2TrafoSize-2, 2)	cIdx=0	cIdx=1
0	0	0
1	9	6
2	21	10

4.2 *Proposed changes to "9.3.1.1 Initialisation process for context variables"*

The changes here affect Tables 9-18, 9-40, 9-41 and 9-42. The updated tables are shown below.

Table 9-18 – Association of ctxIdx and syntax elements for each slice type in the initialisation process

	Syntax element	ctxIdxTable	Slice Type		
			I	**P**	**B**
slice_header()	alf_cu_flag	Error: Reference source not found	0 2	3..5	6..8
coding_tree()	split_coding_unit_flag	Error: Reference source not found	0...2	3..5	6..8
coding_unit()	skip_flag	Error: Reference source not found		0..2	3..5
	cu_qp_delta	Error: Reference source not found	0..3	4..7	8..11
	pred_type	Error: Reference source not found	0	1..4	5..9
prediction_unit()	prev_intra_luma_pred_flag	Error: Reference source not found	0	1	2
	rem_intra_luma_pred_mode	Error: Reference source not found	0	1	2
	intra_chroma_pred_mode	Error: Reference source not found	0..3	4..7	8..11
	merge_flag	Error: Reference source not found		0..2	3..5
	merge_idx	Error: Reference source not found		0..3	4..7
	inter_pred_flag	Error: Reference source not found			0..2
	ref_idx_lc, ref_idx_l0, ref_idx_l1	Error: Reference source not found		0..5	6..11
	mvd_l0[ ][ ][0]	Error: Reference source not found		0..6	14..20
	mvd_lc[ ][ ][0], mvd_l1[ ][ ][0]	Error: Reference source not found			14..20
	mvd_l0[ ][ ][1]	Error: Reference source not found		7..13	21..27
	mvd_lc[ ][ ][1], mvd_l1[ ][ ][1]	Error: Reference source not found			21..27
	mvp_idx_lc, mvp_idx_l0, mvp_idx_l1	Error: Reference source not found		0..1	2..3
transform_tree()	no_residual_data_flag	Error: Reference source not found		0..3	4..7
	split_transform_flag	Error: Reference source not found	0..3	4..7	8..11
	cbf_luma	Error: Reference source not found	0..3	4..7	8..11
	cbf_cb	Error: Reference source not found	0..3	4..7	8..11
	cbf_cr	Error: Reference source not found	0..3	4..7	8..11
residual_coding()	last_significant_coeff_x	Error: Reference source not found	0..40	41..81	82..122
	last_significant_coeff_y	Error: Reference source not found	0..40	41..81	82..122
	significant_coeff_flag (I)	Table 9-40	0..56		
	significant_coeff_flag (B)	Table 9-41		0..56	
	significant_coeff_flag (P)	Table 9-42			0..56

	coeff_abs_level_greater1_flag	Error: Reference source not found	0..79	80..159	160..239
	coeff_abs_level_greater2_flag	Error: Reference source not found	0..79	80..159	160..239

Table 9-40 – Values of variable m and n for significant_coeff_flag ctxIdx (I)

Initialisation variables	**significant_coeff_flag ctxIdx**															
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
m	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
n	77	71	66	61	71	67	66	65	61	71	67	59	53	45	59	55
	16	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
m	0	0	0	0	0	-15	-14	-15	-4	0	-2	-7	-15	-4	1	-4
n	51	53	51	42	45	119	104	106	49	62	72	88	112	28	54	72
	32	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**	**45**	**46**	**47**
m	-7	-10	0	0	0	0	0	0	0	0	0	0	15	7	5	14
n	82	96	67	60	55	46	55	54	62	48	41	33	59	56	57	11
	48	**49**	**50**	**51**	**52**	**53**	**54**	**55**	**56**							
m	10	7	5	11	-9	5	7	10	13							
n	45	53	61	59	38	46	49	48	47							

Table 9-41 – Values of variable m and n for significant_coeff_flag ctxIdx (B)

Initialisation variables	**significant_coeff_flag ctxIdx**															
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
m	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
n	61	56	52	51	56	54	52	55	51	59	52	45	38	37	45	40
	16	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
m	0	0	0	0	0	0	0	-3	2	3	0	-3	-9	-4	4	1
n	37	38	37	40	37	78	66	68	31	53	65	74	93	20	44	57
	32	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**	**45**	**46**	**47**
m	0	-1	0	0	0	0	0	0	0	0	0	0	28	16	11	26
n	65	72	60	49	43	36	43	48	56	37	27	25	29	35	39	-18
	48	**49**	**50**	**51**	**52**	**53**	**54**	**55**	**56**							
m	10	4	-2	-11	0	5	5	9	20							
n	44	58	71	94	0	45	49	45	32							

Table 9-42 – Values of variable m and n for significant_coeff_flag ctxIdx (P)

Initialisation variables	significant_coeff_flag ctxIdx															
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
m	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
n	62	57	54	51	57	55	54	55	51	60	54	47	42	39	47	43
	16	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
m	0	0	0	0	0	0	0	-3	2	3	0	-3	-9	-4	4	1
n	41	42	41	41	39	78	66	68	31	53	65	74	93	20	44	57
	32	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**	**45**	**46**	**47**
m	0	-1	0	0	0	0	0	0	0	0	0	0	28	16	11	26
n	65	72	61	51	43	34	43	48	55	37	27	21	29	35	39	-18
	48	**49**	**50**	**51**	**52**	**53**	**54**	**55**	**56**							
m	10	4	-2	-11	0	5	5	9	20							
n	44	58	71	94	0	45	49	45	32							

4.3 Proposed changes to software implementation

The new constant definitions to be added are as follows.

CTX_IND_MAP_4x4 specifies the partition sets for the 4x4 luma and chroma TUs.

```
static const UInt CTX_IND_MAP_4x4[2][15] =
{
        //LUMA map
        {
                 0,  1,  2,  3,
                 4,  5,  2,  3,
                 6,  6,  7,  7,
                 8,  8,  7,
        },
        //CHROMA map
```

```
    {
              0,  1,  2,  3,
              1,  1,  2,  3,
              4,  4,  5,  5,
              3,  3,  5
    }
};
```

CTX_IND_MAP_8x8 specifies the partition sets for the 8x8 luma and chroma TUs.

```
static const UInt CTX_IND_MAP_8x8[2][63] =
{
       //LUMA map
       {
              0,  1,  2,  2,  3,  3,  4,  4,
              1,  1,  2,  2,  3,  3,  4,  4,
              5,  5,  6,  6,  7,  7,  4,  4,
              5,  5,  6,  6,  7,  7,  4,  4,
              8,  8,  9,  9,  6,  7, 10, 10,
              8,  8,  9,  9,  9, 10, 10, 10,
             11, 11, 11, 11, 10, 10, 10, 10,
             11, 11, 11, 11, 10, 10, 10

       },
       //CHROMA map
       {
              0,  0,  1,  1,  2,  2,  3,  3,
              0,  0,  1,  1,  2,  2,  3,  3,
              1,  1,  1,  1,  2,  2,  3,  3,
              1,  1,  1,  1,  2,  2,  3,  3,
              2,  2,  2,  2,  1,  2,  3,  3,
              2,  2,  2,  2,  2,  3,  3,  3,
              3,  3,  3,  3,  3,  3,  3,  3,
              3,  3,  3,  3,  3,  3,  3
       }
};
```

NUM_SIG_FLAG_CTX_4x4 is the maximum number of partitions in any 4x4 set.

const UInt NUM_SIG_FLAG_CTX_4x4 = 9;

All of the algorithmic changes are in the TComTrQuant::getSigCtxInc function, and are shown below.

```
getSigCtxInc(pcCoeff, uiPosX, uiPosY, uiLog2BlkSize, uiStride, eTType) {
   eTType = eTType == TEXT_LUMA ? TEXT_LUMA : eTType == TEXT_NONE ?
            TEXT_NONE : TEXT_CHROMA
   L_C = eTType != TEXT_LUMA
   if (uiLog2BlkSize == 2) {
      return CTX_IND_MAP_4x4[L_C][(uiPosY << 2) + uiPosX]
   }
   if (uiLog2BlkSize == 3) {
      return NUM_SIG_FLAG_CTX_4x4 + CTX_IND_MAP_8x8[L_C] [(uiPosY << 3) + uiPosX]
   }
   // The rest of the function is unchanged
}
```

5 References

[1] D. Marpe, H. Schwarz, and T. Wiegand, “Context-based adaptive binary arithmetic coding in the h.264/AVC video compression standard,” *IEEE Transactions on Circuits and Systems for Video Technology*, 13(7):620–636, Jul. 2003.

[2] B. Bross, W-J Han, J-R Ohm, G. J. Sullivan, and T. Wiegand, “WD4: Working Draft 4 of High-Efficiency Video Coding,” JCT-VC of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, 6th Meeting, Torino, July 2011.

[3] F. Bossen, “Common test conditions and software reference configurations,” JCT-VC of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, 6th Meeting, Torino, July 2011.

6 Patent rights declaration(s)

Research In Motion Ltd. may have IPR relating to the technology described in this contribution and, conditioned on reciprocity, is prepared to grant licenses under reasonable and non-discriminatory terms as necessary for implementation of the resulting ITU-T Recommendation | ISO/IEC International Standard (per box 2 of the ITU-T/ITU-R/ISO/IEC patent statement and licensing declaration form).