|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11**  5th Meeting: Geneva, CH, 16-23 March, 2011 | Document: JCTVC-E383 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **CE5: coefficient coding with LCEC for large blocks** | | |
| *Status:* | Input Document to JCT-VC | | |
| *Purpose:* | CE report | | |
| *Author(s) or Contact(s):* | Marta Karczewitz, Xianglin Wang  Wei-Jung Chen | Email: | [martak@qualcomm.com](mailto:martak@qualcomm.com)  [xianglin@qualcomm.com](mailto:xianglin@qualcomm.com)  [wchien@qualcomm.com](mailto:wchien@qualcomm.com) |
| *Source:* | Qualcomm Inc. | | |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Abstract

Coding results of extending LCEC coefficient coding to 16x16 and 32x32 blocks are presented. In addition this contribution proposes to remove VLC tables used to code the last active transform coefficient position for block sizes 8x8 and above and to reduce by 2 the size of the table used to code number of zeros between two nonzero coefficients counted in the reverse scan order.

# Introduction

According to the current HM, position of the last active coefficient *last\_pos* is coded jointly with *lev.* The *lev* indicates whether the absolute value of that coefficient level is 1 (*lev* =0) or great than 1 (*lev=1)*. To code {*lev, last\_pos*} pair for 4x4 and 8x8 blocks there are mapping tables, dependent on block type, between value of *m=N∙N∙lev + last\_pos* and code number *cn,* where N is equal to 4 or 8 depending on the block size. The mapping tables are adaptive. Hence there are 2 sets of 3 tables, 32 elements each for 4x4 blocks and 2 sets of 8 tables, 128 elements each for 8x8 blocks.

Remaining coefficients are coded in run mode followed by level mode. In run mode coding, the number of zero coefficients is signaled starting from the previous nonzero coefficient. The syntax element *isLevelOne\_run* combines information about the *run* and the *lev*. *Run* is defined as the number of zero coefficients between the current nonzero coefficient at position *k* and the next nonzero coefficient in the reverse scanning order. The mapping between {*lev, run*} pair and code number *cn* is also dependent on value *k*:

*cn=g\_auiLumaRun8x8[Min(k-1,28)][lev][run].*

The size of the *g\_auiLumaRun8x8 is 29∙2∙64.* However since many of thetable elements are unused with optimized storage it would require 968 bytes.

# Proposal

## Last Coding

Assignment of the code number *cn* to *{lev, last\_pos}* pair for 4x4 blocks is not modified. For block sizes *N=8,16* and *32* code number *cn* for *{lev, last\_pos}* pair is calculated as follows:

*if (lev ==0)*

*cn=((last\_pos + (last\_pos /N))/N)+ last\_pos;*

*else*

*if (last\_pos <N)*

*cn= N∙(last\_pos +1);*

*else*

*cn = N∙N∙lev + last\_pos;*

Two new VLC tables, VLC12 and VLC13, are used to code *cn* assigned to *{lev, last\_pos}* pair for blocks 16x16 and 32x32. VLC12 and VLC13 are Rice code of parameters 5 and 4, respectively:

VLC12: 1xxxxxx, 01xxxxxx, …

VLC13: 1xxxxx, 01xxxxx, …

In HM 2.0 the choice of the VLC table (number of the used VLC table, *vlcnum*) to code *cn* depends on the value of *m\_uiLastPosVlcIndex[n]*, where *n* specifies block size and type:

*vlcnum=g\_auiLastPosVlcNum[n][Min(16,m\_uiLastPosVlcIndex[n])];*

Value of m\_uiLastPosVlcIndex[n] is updated after each *cn* is encoded:

*m\_uiLastPosVlcIndex[n] +=*

*cn == m\_uiLastPosVlcIndex[n] ? 0 : (cn < m\_uiLastPosVlcIndex[n] ? -1 : 1);*

We propose to modify above formula for blocks 16x16 and 32x32 as follows:

*m\_uiLastPosVlcIndex[n] +=*

*(cn>>2) == m\_uiLastPosVlcIndex[n] ? 0 : ((cn>>2) < m\_uiLastPosVlcIndex[n] ? -1 : 1);*

## Run Coding

Since for 4x4 and 8x8 intra blocks code number *cn* assignment to the {*lev, run*} pair is already formula based, dependent on the position of the current nonzero coefficient, same formula is used for 16x16 and 32x32 blocks. For 8x8, 16x16 and 32x32 inter blocks *g\_auiLumaRun8x8* table is used when *k<29.* Otherwise

*if (lev==0)*

*cn=run;*

*else*

*cn=k +run+1;*

Hence the last row of *g\_auiLumaRun8x8* is never used in the proposed method.

It was further noticed that elements of the table *g\_auiLumaRun8x8* exhibit clear pattern. With some modifications to its entries we are able to store only its values for *lev=0*. The values for *lev=1* can be obtained as follows:

*cn=k + g\_auiLumaRun8x8[k-1][run].*

The modified table *g\_auiLumaRun8x8* can be stored using434 bytes.

Such a coding scheme is tested both with and without counter based adaptive scans. In case of adaptive scans three scans are used: horizontal, vertical and zigzag. The scan order for first 64 coefficients is adaptively adjusted based on previously coded coefficients (as described in the document JCTVC-C250).

# Coding results

Based on test conditions specified in CE5, simulations are performed using all three low complexity configurations. Detailed results can be found in the associated excel data sheet.

The results show that when extending the LCEC coefficient coding to 16x16 blocks, an average coding gain of 2.5%, 1.1% and 0.8% can be obtained respectively with all intra, low delay and random access configuration. When also extending the coding scheme to 32x32 blocks, an average coding gain of 2.9%, 1.4% and 1.0% can be obtained under the same three configurations.

# Conclusion

This contribution presents CE results on extending LCEC coefficient coding to large blocks. Results show that extending the coefficient coding to large blocks can bring substantial coding gain. Based on the results, we recommend the proposed coefficient coding extension be adopted into HM.

# References

[1] M. Karczewicz, X.Wang, W.-J. Chien, “CE5: Improved coefficient coding with LCEC”, JCTVC-D374, Daegu, Korea, Jan. 2011.

# Patent rights declaration(s)

**Qualcomm may have current or pending patent rights 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).**

# Annex

Values of table auiLastPosVlcNum:

g\_auiLastPosVlcNum[8][17] =

{

{10,10,10,10, 2,2,2,7,9,9,9,9,9,4,4,4,4},

{10,10,10,10,10,2,9,9,9,9,9,9,9,4,4,4,4},

{ 2, 2, 2, 2, 2,7,7,7,7,7,7,7,7,7,4,4,13},

{ 2, 2, 2, 2, 7,7,7,7,7,7,7,7,7,7,7,7,13},

{ 2, 2, 2, 2, 7,7,7,7,7,7,7,7,7,7,7,7,13},

{10,10,10, 4,4,4, 4,12,12,12,12,12,12,12,12,12,12},

{10,10,10,10,4,4,12,12,12,12,12,12,12,12,12,12,12},

{10,10,10,10,4,4,12,12,12,12,12,12,12,12,12,12,12},

};

Values of the table g\_auiLumaRun8x8:

g\_auiLumaRun8x8[28][29] =

{

{1, 0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},

{2, 1, 0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},

{1, 3, 2, 0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},

{2, 1, 3, 4, 0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},

{1, 5, 3, 2, 4, 0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},

{1, 2, 6, 5, 3, 4, 0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},

{2, 1, 3, 5, 4, 7, 6, 0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},

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

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

{1, 2, 5, 9, 8, 7, 3, 4, 10, 6, 0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},

{2, 1, 3, 4, 7, 8, 5, 6, 9, 11, 10, 0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},

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

{1, 2, 6, 7, 5, 3, 4, 8, 9, 13, 11, 12, 10, 0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},

{1, 2, 4, 8, 9, 7, 6, 3, 5, 12, 14, 13, 11, 10, 0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},

{1, 2, 3, 5, 9, 10, 8, 7, 4, 6, 12, 15, 14, 13, 11, 0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},

{1, 2, 3, 4, 5, 8, 10, 7, 6, 9, 11, 13, 16, 14, 15, 12, 0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},

{1, 3, 4, 2, 5, 6, 10, 9, 7, 8, 11, 12, 13, 15, 16, 17, 14, 0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},

{1, 2, 4, 7, 6, 3, 5, 8, 9, 10, 11, 17, 15, 13, 14, 16, 18, 12, 0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1},

{1, 2, 5, 7, 9, 10, 6, 3, 4, 8, 11, 16, 19, 17, 15, 13, 14, 18, 12, 0, -1, -1, -1, -1, -1, -1, -1, -1, -1},

{1, 3, 5, 7, 9, 14, 12, 8, 6, 2, 4, 15, 17, 19, 18, 16, 10, 11, 20, 13, 0, -1, -1, -1, -1, -1, -1, -1, -1},

{1, 2, 4, 6, 9, 8, 15, 14, 13, 7, 3, 5, 16, 18, 21, 19, 17, 10, 12, 20, 11, 0, -1, -1, -1, -1, -1, -1, -1},

{2, 1, 3, 4, 5, 8, 9, 11, 13, 10, 6, 7, 12, 14, 19, 22, 21, 17, 15, 18, 20, 16, 0, -1, -1, -1, -1, -1, -1},

{1, 5, 4, 2, 3, 6, 8, 7, 11, 12, 9, 10, 13, 14, 15, 18, 22, 21, 17, 19, 20, 23, 16, 0, -1, -1, -1, -1, -1},

{1, 4, 8, 6, 5, 2, 3, 7, 9, 12, 10, 11, 13, 16, 15, 14, 18, 20, 21, 19, 23, 22, 24, 17, 0, -1, -1, -1, -1},

{1, 3, 7, 12, 10, 6, 5, 2, 4, 8, 9, 11, 13, 16, 23, 20, 15, 14, 18, 19, 21, 24, 25, 22, 17, 0, -1, -1, -1},

{1, 2, 5, 9, 13, 11, 10, 8, 6, 3, 4, 7, 12, 15, 18, 25, 22, 19, 16, 14, 17, 21, 26, 23, 24, 20, 0, -1, -1},

{1, 2, 4, 7, 11, 15, 12, 9, 10, 8, 5, 3, 6, 14, 16, 22, 23, 24, 20, 19, 13, 18, 26, 27, 25, 21, 17, 0, -1},

{1, 2, 3, 5, 6, 10, 14, 13, 11, 9, 12, 7, 4, 8, 16, 18, 21, 23, 25, 20, 22, 15, 19, 27, 28, 26, 24, 17, 0}

};