|  |  |
| --- | --- |
| **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-E384 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **LCEC coefficient coding table reduction** | | |
| *Status:* | Input Document to JCT-VC | | |
| *Purpose:* | Proposal | | |
| *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 | | |

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

# Abstract

In this contribution, a scheme is proposed to reduce the size of VLC tables used in coding the {*lev, run*} pair in LCEC coefficient coding. By sharing a same mapping table for different values of *lev* and using formula to calculate the code number in certain conditions, the table size needed can be at least reduced to less than half of the original size. Simulation shows that the scheme only has 0.00%, 0.07% and 0.05% impact on coding performance with all intra, low delay and random access configurations respectively.

# Introduction

In HM2.0 LCEC coefficient coding, 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*. The *lev* indicates whether the absolute value of a coefficient level is 1 (*lev* =0) or great than 1 (*lev=1)*. *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

To minimize the required memory to store the table, it is proposed that for 8x8 inter blocks *g\_auiLumaRun8x8* table is used for code number *cn* assignment to the {*lev, run*} pair only 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* are obtained as follows:

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

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

# 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 with the proposed LCEC table reduction scheme, the impact on coding performance is 0.01%, 0.07% and 0.05% with all intra, low delay and random access configurations.

# Conclusion

This contribution presents a scheme to reduce the size of table needed for LCEC coefficient coding. Using the proposed scheme, the table size needed can be at least reduced to less than half of the original size. Results show that the proposed scheme has little impact on coding performance. Based on the results, we recommend the proposed table reduction scheme be adopted into HM.

# 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 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}

};