|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11**  6th Meeting: Torino, IT, 14-22 July, 2011 | Document: JCTVC-F608 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **Removing Chroma Zonal Coding in CAVLC** | | |
| *Status:* | Input Document to JCT-VC | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Marta Karczewicz  Xianglin Wang  Wei-Jung Chien  Liwei Guo  5775 Morehouse Dr.  San Diego, CA 92121, USA | Tel:  Email: | 1-858-658-5673 [martak@qualcomm.com](mailto:martak@qualcomm.com)  [xianglin@qualcomm.com](mailto:xianglin@qualcomm.com)  [wchien@qualcomm.com](mailto:wchien@qualcomm.com)  [liweig@qualcomm.com](mailto:liweig@qualcomm.com) |
| *Source:* | Qualcomm Inc. | | |

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

# Abstract

This contribution proposes removing the CAVLC zonal coding currently applied to chroma block with a size of 16x16 or larger. Instead, code those chroma blocks normally with coefficients at every frequency location included.

# Introduction

In HM3.0, when CAVLC coefficient coding is used to code a chroma block with a size of 16x16 or larger, only the first 64 coefficients along the scan order are coded using the same coefficient coding scheme designed for 8x8 block. In case of intra blocks 16x16 or larger zig-zag scan is used. In case of inter blocks 8x8 zig-zag scan is used (upper, left corner of the 16x16 block is canned). The remaining coefficients are all assumed to be zero.

# Proposed Method

We propose to remove restriction on the number of coefficients coded for the chroma blocks. The coding process remain unchanged. As a result, no more zonal selection is needed and coefficients at every frequency location of a chroma block may be coded.

# Codeword Length Handling

When coefficient coding for chroma blocks is extended to 16x16 blocks codeword longer than 32 bits can be generated. To avoid it two modifications are proposed.

The index of the VLC table, vlcNum, used to code syntax element last\_pos\_level\_one for chroma is derived as

vlcNum = lastPosVlcNumTable[blockType][vlcNumIdx].

It is proposed to set the value of the array lastPosVlcNumTable for blockType=0 and vlcNumIdx=7 to 2 insted of 7.

We further propose to modify VLC table for vlcNum=9, by concatenating Golomb codes with two different parameters k. Currently Golomb code with k=4 is used, we propose to concatenate it with Golomb code k=7. The new decoding process for vlcNum=9 is:

– If leadingZeroBits is equal to 0,

b = read\_bits(1)

if (b) {

b = read\_bits(1)

if (b) {

codeNum = 3 + read\_bits(3)

}

else{

b = read\_bits(1)

codeNum = b ? 1 + read\_bits(1) : 0

}

}

– Otherwise (leadingZeroBits is less than than 16),

codeNum = (leadingZeroBits << 4) + read\_bits(4) + 11

– Otherwise,

codeNum = ((leadingZeroBits-16) << 7) + read\_bits(7) + 267

It should be noted that codeword longer than 32 bits can be generated in HM 3.0 also when encoding luma blocks.

# Simulation Results

Using HM3.0, simulation is performed under the common test conditions [1]. Results in Table 1 show that removing zonal coding of chroma block does not change coding performance much.

Table 1. Simulation results with reference to HM3.0

|  |  |  |  |
| --- | --- | --- | --- |
|  | LC | | |
| Y | U | V |
| AI | 0.0 | -0.1 | -0.1 |
| RA | 0.0 | -0.2 | -0.4 |
| LB | 0.0 | -0.2 | -0.2 |

Adding the changes to remove possibility of the codewords longer than 32 bits changes has only minor influence on the results:

Table 2. Simulation results with reference to HM3.0

|  |  |  |  |
| --- | --- | --- | --- |
|  | LC | | |
| Y | U | V |
| AI | 0.0 | -0.1 | -0.1 |
| RA | 0.0 | -0.2 | -0.4 |
| LB | 0.0 | -0.1 | -0.3 |

# Conclusion

Removal of the CAVLC zonal coding for chroma block of 16x16 or larger makes the overall design more consistent. We recommend such extension of CAVLC coefficient coding to be adopted into HM.

We further recommend that possibility of generating codewords longer than 32 bits will be examined for all the syntax elements and unified solution developed within the CE.

# Reference

1. Frank Bossen, “Common test conditions and software reference configurations”, JCTVC-E700, JCT-VC 5th Meeting: Geneva, March, 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: proposed text change

## Parsing process for last\_pos\_level\_one

Inputs to this process are bits from RBSP and the variable arrays lastPosVlcNumIndex and lastPosTable.

Output of this process is the syntax element last\_pos\_level\_one, the variables trOne and vlcNumLevel, and the variable arrays lastPosVlcNumIndex and lastPosTable.

The parsing process for last\_pos\_level\_one is specified as follows.

* The variable N is set equal to (1 << log2TrafoSize) >> (cIdx > 0 ? 2 : 0).
* If N is greater than 4, last\_pos\_level\_one is derived in the following ordered steps:

1. The variable blockType is derived as

blockType = (cIdx == 0 ? (PredMode==MODE\_INTRA ? 0 : slice\_type + 1) + (N > 8 ? 5 : 2) : cIdx – 1)

…

1. The variable array lastPosVlcNumIndex[blockType] is updated as follows.

if ((N = = 8 | | blockType < 2? codeNum : codeNum>>2) < lastPosVlcNumIndex[blockType])  
 lastPosVlcNumIndex[blockType] -= 1 (9‑29)  
 else if ((N = = 8 | | blockType < 2? codeNum : codeNum>>2) > lastPosVlcNumIndex[blockType])  
 lastPosVlcNumIndex[blockType] += 1

…

Table 9‑8 – Specification of lastPosVlcNumTable[blockType][vlcNumIdx]

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **vlcNumIdx** | | | | | | | | | | | | | | | | |
| **blockType** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** |
| 0 | 10 | 10 | 10 | 10 | 2 | 2 | 2 | ~~7~~2 | 9 | 9 | 9 | 9 | 9 | 4 | 4 | 4 | 4 |
| 1 | 10 | 10 | 10 | 10 | 10 | 2 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 4 | 4 | 4 | 4 |
| 2 | 2 | 2 | 2 | 2 | 2 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 4 | 4 | 13 |
| 3 | 2 | 2 | 2 | 2 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 13 |
| 4 | 2 | 2 | 2 | 2 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 13 |
| 5 | 10 | 10 | 10 | 4 | 4 | 4 | 4 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 |
| 6 | 10 | 10 | 10 | 10 | 4 | 4 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 |
| 7 | 10 | 10 | 10 | 10 | 4 | 4 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 | 12 |

## Parsing process for VLC codes

Inputs to this process are bits from the RBSP and the parameter vlcNum specifying the VLC code. If vlcNum is equal to 14, inputs to this process additionally include a parameter cMax.

…

* Otherwise, if vlcNum is equal to 9,
* If leadingZeroBits is equal to 0,

b = read\_bits(1)  
 if (b) {  
 b = read\_bits(1)  
 if (b) {  
 codeNum = 3 + read\_bits(3) (9‑11)  
 }  
 else{  
 b = read\_bits(1)  
 codeNum = b ? 1 + read\_bits(1) : 0  
 }  
 }

* ~~Otherwise (leadingZeroBits is greater than 0),~~

~~codeNum = (leadingZeroBits << 4) + read\_bits( 4) + 11~~

* Otherwise , if leadingZeroBits is less than 16,

codeNum = (leadingZeroBits << 4) + read\_bits( 4) + 11

* Otherwise

codeNum = ((leadingZeroBits – 16) << 7) + read\_bits( 7) + 267