|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11**  6th Meeting: Torino, 14-22 July, 2011 | Document: JCTVC-F298 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **Run-level Table Reduction for CAVLC** | | |
| *Status:* | Input Document to JCT-VC | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Liwei Guo,  Xianglin Wang  Marta Karczewicz 5775 Morehouse Drive San Diego, CA 92121 USA | Tel: Email: | 1-858-651-8125 [liweig@qualcomm.com](mailto:liweig@qualcomm.com) |
| *Source:* | Qualcomm Incorporated | | |

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

# Abstract

In current HM software, a run-level table of size 434 is used in the coding of Inter block coefficients (as well as Intra Chroma block coefficients). This contribution reduces the run-level related table sizes from 434 to 148. The Luma B-D rate changes for different cases are 0.00% AI, -0.14% RA, 0.03%LD, and 0.06% LDP.

# Introduction

According to the current HM [1], a table *g\_auiLumaRun8x8[maxrun][run]* is used in the run-level coding of inter block coefficients (both luma and chroma) and intra block chroma coefficients coding. This table is applied for *maxrun* = 0-27, and the range of *run* is 0 to *maxrun+1*, so the table size is 434. Codeword number cn is calculated as follows:

When *maxrun* <=27

*If lev = 0,*

*cn = g\_auiLumaRun8x8[maxrun][run]*

*if lev = 1,*

*cn = maxrun +1+g\_auiLumaRun8x8[maxrun][run]*

When *maxrun* > 27,

*If lev = 0, cn = run;*

*If lev = 1, cn =marxun+ 2+ run;*

# Proposal

## Run-level coding for Luma

For luma block, codenumber *cn* calculation for *maxrun* = 0-9 and *maxrun* > 27 are not changed. For *maxrun* = 10-27, we store *cn* for (*run=maxrun+1, lev = 0)*, and let *cn* increase with *run* for other *(run, level)*. The following are the pseudo codes for *cn* calculationof *maxrun = 10-27:*

*If lev = 0,*

*If run = maxrun+1,*

*cn = g\_auiMaxRunPlusOneLuma[maxrun-10];*

*else*

*cn = (run < g\_auiMaxRunPlusOneLuma[maxrun-10])?run: run+1;*

*If lev = 1,*

*cn = maxrun + 2 + run;*

The size of *g\_auiLumaRun8x8[maxrun][run]* (*maxrun < 10)* is 65, and the size of *g\_auiMaxRunPlusOneLuma[maxrun-10] (*maxrun = 10-27) is 18. So the total storage is 83.

## Run-level coding for Chroma

For chroma block (applied to both intra and inter), code number *cn* for *maxrun* = 0-9 is not changed but with a new run-level table *g\_auiChromaRun8x8.* For *maxrun* >= 10, *cn is 0* is for (*run=maxrun+1, lev = 0)*, and *cn* increases with *run* for other *(run level).* The following are the pseudo codes for *maxrun* >= 10:

*If lev = 0,*

*cn = (run==maxrun+1)? 0: run+1;*

*else*

*cn = maxrun + 2 + run;*

The size of *g\_auiChromaRun8x8[maxrun][run]* for *maxrun < 10* is 65. This is the only storage required for Chroma.

The total storage for Lum and Chroma is 83+65=148.

# Experimental results

We implement the presented run-level table reduction on top of HM 3.0. The simulations are conducted under low-complexity common conditions specified in [2]. The coding efficiency is measured by using 4 point BD-Rate. The encoding time is measured on clusters with same CPU, and the decoding time is measured by decoding all sequences on a single PC.

Table I summarizes the results of the proposed run-level table reduction. The Y bitrate reduction for AI, RA, LD and LDP are 0.00%, -0.14%, 0.03% and 0.06%. The change of UV bitrate is very negligible. Furthermore, there is basically no change in encoding and decoding time.

# Conclusion

In this contribution, we present a method to reduce the run-level table storage from 434 to 148. We observe on average 0.14% bit rate reduction in random access case, and 0.03% and 0.06% in cares for low delay and low delay P cases. There is no change in encoding time and decoding time. Based on the results, we recommend the proposed run-level table reduction be adopted into HM.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | All Intra LC | | | Random Access LC | | |
| Y | U | V | Y | U | V |
| Class A | 0.00 | 0.05 | 0.09 | -0.13 | -0.13 | 0.04 |
| Class B | 0.00 | 0.01 | 0.03 | -0.18 | 0.05 | -0.10 |
| Class C | 0.00 | 0.02 | -0.02 | -0.13 | 0.04 | -0.07 |
| Class D | 0.00 | 0.01 | 0.02 | -0.10 | -0.11 | -0.06 |
| Class E | 0.00 | 0.03 | 0.02 |  |  |  |
| **Overall** | **0.00** | **0.02** | **0.03** | **-0.14** | **-0.03** | **-0.05** |
| Enc Time[%] | 99% | | | 100% | | |
| Dec Time[%] | 101% | | | 98% | | |
|  | Low delay B LC | | | Low delay P LC | | |
| Y | U | V | Y | U | V |
| Class A |  |  |  |  |  |  |
| Class B | -0.06 | 0.12 | -0.28 | -0.01 | 0.27 | 0.14 |
| Class C | 0.00 | 0.19 | -0.27 | 0.02 | -0.13 | -0.18 |
| Class D | 0.01 | -0.07 | -0.28 | 0.01 | -0.06 | -0.69 |
| Class E | 0.25 | 0.03 | -0.25 | 0.33 | -0.28 | -0.54 |
| **Overall** | **0.03** | **0.07** | **-0.27** | **0.06** | **-0.02** | **-0.28** |
| Enc Time[%] | 100% | | | 100% | | |
| Dec Time[%] | 100% | | | 98% | | |

Table Results of run-level table reduction

# Draft texts

#### Parsing process for run\_level\_one

Inputs to this process are bits from slice data, the scan position *n* of the previous non-zero transform coefficient in inverse scan order.

Output of this process is the syntax element run\_level\_one.

maxRun = n

#### When blockType > 2 and blockType is not equal to 5 (Luma)

* + The variable levelGreaterThanOneFlag is derived as

levelGreaterThanOneFlag = codeNum <= maxRun + 1 ? 0 : 1

* + If maxRun is less than 10, the variable runOfZeros is derived from maxRun and codeNum as shown in Table I g\_auiLumaRun8x8.
  + If maxRun is between 10-27, the variable runOfZeros is derived from maxRun and codeNum using Table II auiMaxRunPlusOneLuma as

If codeNum == g\_auiMaxRunPlusOneLuma[maxrun-10], runOfZeros = maxrun+1;

else if codeNum > maxrun+1,

runOfZeros = codeNum – maxrun-2

else

runOfZeros = (codeNum < g\_auiMaxRunPlusOneY[maxrun-10])? codeNum: codeNum -1

* + Otherwise (maxRun is not less than 28), the variable runOfZeros is derived as

runOfZeros = codeNum <= maxRun + 1 ? codeNum : codeNum – maxRun – 2

* + The syntax element run\_level\_one is derived as

run\_level\_one = levelGreaterThanOneFlag ? runOfZeros + n : runOfZeros

#### When blockType <2 (Chroma)

* + The variable levelGreaterThanOneFlag is derived as

levelGreaterThanOneFlag = codeNum <= maxRun + 1 ? 0 : 1

* + If maxRun is less than 10, the variable runOfZeros is derived from maxRun and codeNum as shown in Table III g\_auiChromaRun8x8.
  + Otherwise (maxRun is not less than 10), the variable runOfZeros is derived as

If codeNum>= maxRun + 1,

runOfZeros = codeNum – maxrun-2

else runOfZeros = codeNum == 0? maxRun +1: codeNum-1;

* + The syntax element run\_level\_one is derived as

run\_level\_one = levelGreaterThanOneFlag ? runOfZeros + n : runOfZeros

Table I g\_auiLumaRun8x8

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **codeNum%(maxRun+1)** | | | | | | | | | | |
| **maxRun** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| 0 | 0 | 1 | NA | NA | NA | NA | NA | NA | NA | NA | NA |
| 1 | 0 | 2 | 1 | NA | NA | NA | NA | NA | NA | NA | NA |
| 2 | 0 | 2 | 3 | 1 | NA | NA | NA | NA | NA | NA | NA |
| 3 | 1 | 0 | 4 | 2 | 3 | NA | NA | NA | NA | NA | NA |
| 4 | 0 | 5 | 3 | 2 | 4 | 1 | NA | NA | NA | NA | NA |
| 5 | 0 | 2 | 5 | 4 | 3 | 6 | 1 | NA | NA | NA | NA |
| 6 | 1 | 0 | 2 | 6 | 4 | 5 | 7 | 3 | NA | NA | NA |
| 7 | 0 | 3 | 2 | 1 | 5 | 6 | 7 | 8 | 4 | NA | NA |
| 8 | 0 | 2 | 3 | 6 | 5 | 4 | 7 | 9 | 8 | 1 | NA |
| 9 | 0 | 1 | 3 | 4 | 6 | 5 | 6 | 8 | 10 | 9 | 2 |

Table II g\_auiMaxRunPlusOneLuma

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Index** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|  | 6 | 6 | 7 | 5 | 5 | 4 | 6 | 6 | 8 |
| **Index** | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
|  | 3 | 1 | 4 | 7 | 8 | 12 | 5 | 2 | 1 |

Table III g\_auiChromaRun8x8

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **codeNum%(maxRun+1)** | | | | | | | | | | |
| **maxRun** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| 0 | 1 | 0 | NA | NA | NA | NA | NA | NA | NA | NA | NA |
| 1 | 2 | 1 | 0 | NA | NA | NA | NA | NA | NA | NA | NA |
| 2 | 1 | 3 | 2 | 0 | NA | NA | NA | NA | NA | NA | NA |
| 3 | 2 | 1 | 3 | 4 | 0 | NA | NA | NA | NA | NA | NA |
| 4 | 1 | 5 | 3 | 2 | 4 | 0 | NA | NA | NA | NA | NA |
| 5 | 1 | 2 | 6 | 5 | 3 | 4 | 0 | NA | NA | NA | NA |
| 6 | 2 | 1 | 3 | 5 | 4 | 7 | 6 | 0 | NA | NA | NA |
| 7 | 1 | 5 | 4 | 2 | 3 | 6 | 8 | 7 | 0 | NA | NA |
| 8 | 1 | 3 | 8 | 7 | 5 | 2 | 4 | 9 | 6 | 0 | NA |
| 9 | 1 | 2 | 5 | 9 | 8 | 7 | 3 | 4 | 10 | 6 | 0 |

# References

[1] M. Karczewitz, X. Wang, W.-J. Chien, “CE5: coefficient coding with LCEC for large blocks”, JCTVC-E383, JCT-VC Meeting, Geneva, CH, Mar. 2011.

[2] F.Bossen, “Common test conditions and software reference configurations”, JCTVC-E700, JCT-VC Meeting, Geneva, CH, Mar. 2011.

# Patent rights declaration(s)

**Qualcomm 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).**