|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11**  10th Meeting: Stockholm, Sweden, July 11-20, 2012 | Document: JCTVC-J0571  m26297 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **Side activity report on slice header parsing overhead reduction** | | |
| *Status:* | Input Document to JCT-VC | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Minhua Zhou Texas Instruments Inc., USA  Alexis Michael Tourapis  Apple Inc., USA | Tel: Email:  Tel  Email: | +1-214-480-3816 [zhou@ti.com](mailto:zhou@ti.com)  +1-408-783-5450  atourapis@apple.com |
| *Source:* | Texas Instruments Inc and Apple Inc. | | |

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

# Abstract

JCTVC-J0083 expresses concerns about the slice header parsing overhead in the evil case. The weighted prediction table is the most parsing intensive part of the slice header. Based on discussion on JCTVC-J0083, it is proposed to reduce the worst case number of weighted prediction tables from 32 in the current design to 8, and impose a limit on the sum of signaled luma/chroma weight flags (namely, luma\_weight\_l0\_flag, luma\_weight\_l1\_flag, chroma\_weight\_l0\_flag, and chroma\_weight\_l1\_flag) in pred\_weight\_table( ). Also, it is recommended to make the syntax of pred\_weight\_table( ) more parsing friendly by pulling luma/chroma weight flags out of the loop. The proposed solution does not change the slice header syntax.

# Introduction

Slice header parsing can be a significant overhead for real time decoding in the worst case. The weighted prediction table is the most parsing intensive part of the slice header. It is proposed to reduce the worst case number of weighted prediction tables from 32 in the current design to 8. In the proposed solution, the constraint is expressed by imposing a limit on the sum of signaled luma/chroma weight flags (namely, luma\_weight\_l0\_flag, luma\_weight\_l1\_flag, chroma\_weight\_l0\_flag, and chroma\_weight\_l1\_flag) in pred\_weight\_table( ). Also, the syntax of pred\_weight\_table( ) is made more parsing friendly by pulling luma/chroma weight flags out of the loop.

# CD text changes

Replace section **7.3.3.3 Prediction weight table syntax** with

|  |  |
| --- | --- |
| pred\_weight\_table( ) { | Descriptor |
| **luma\_log2\_weight\_denom** | ue(v) |
| if( chroma\_format\_idc != 0 ) |  |
| **delta\_chroma\_log2\_weight\_denom** | se(v) |
| if( slice\_type = = P | | slice\_type = = B ) { |  |
| for( i = 0; i <= num\_ref\_idx\_l0\_active\_minus1; i++ ) |  |
| **luma\_weight\_l0\_flag[** i **]** | u(1) |
| if( chroma\_format\_idc != 0 ) |  |
| for( i = 0; i <= num\_ref\_idx\_l0\_active\_minus1; i++ ) |  |
| **chroma\_weight\_l0\_flag[** i **]** | u(1) |
| for( i = 0; i <= num\_ref\_idx\_l0\_active\_minus1; i++ ) { |  |
| **~~luma\_weight\_l0\_flag~~** | ~~u(1)~~ |
| if( luma\_weight\_l0\_flag[ i ] ) { |  |
| **delta\_luma\_weight\_l0[** i **]** | se(v) |
| **luma\_offset\_l0[** i **]** | se(v) |
| } |  |
| ~~if( chroma\_format\_idc != 0 ) {~~ |  |
| **~~chroma\_weight\_l0\_flag~~** | ~~u(1)~~ |
| if( chroma\_weight\_l0\_flag[ i ] ) |  |
| for( j =0; j < 2; j++ ) { |  |
| **delta\_chroma\_weight\_l0[** i **][** j **]** | se(v) |
| **delta\_chroma\_offset\_l0[** i **][** j **]** | se(v) |
| } |  |
| } |  |
| } |  |
| If (slice\_type = = B ) { |  |
| for( i = 0; i <= num\_ref\_idx\_l1\_active\_minus1; i++ ) |  |
| **luma\_weight\_l1\_flag[** i **]** | u(1) |
| if( chroma\_format\_idc != 0 ) |  |
| for( i = 0; i <= num\_ref\_idx\_l1\_active\_minus1; i++ ) |  |
| **chroma\_weight\_l1\_flag[** i **]** | u(1) |
| for( i = 0; i <= num\_ref\_idx\_l1\_active\_minus1; i++ ) { |  |
| **~~luma\_weight\_l1\_flag~~** | ~~u(1)~~ |
| if( luma\_weight\_l1\_flag[i]) { |  |
| **delta\_luma\_weight\_l1[ i ]** | se(v) |
| **luma\_offset\_l1[ i ]** | se(v) |
| } |  |
| ~~if( chroma\_format\_idc != 0 ) {~~ |  |
| **~~chroma\_weight\_l1\_flag~~** | ~~u(1)~~ |
| if( chroma\_weight\_l1\_flag[i]) |  |
| for( j =0; j < 2; j++ ) { |  |
| **delta\_chroma\_weight\_l1[ i ][ j ]** | se(v) |
| **delta\_chroma\_offset\_l1[ i ][ j ]** | se(v) |
| } |  |
| } |  |
| } |  |
| } |  |

#### Add the following constraint at the end of section 7.4.3.3 Weighted prediction parameters semantics (changes marked in yellow)

**luma\_log2\_weight\_denom** is the base 2 logarithm of the denominator for all luma weighting factors. The value of luma\_log2\_weight\_denom shall be in the range of 0 to 7, inclusive.

**delta\_chroma\_log2\_weight\_denom** is the difference of the base 2 logarithm of the denominator for all chroma weighting factors.

The variable ChromaLog2WeightDenom is specified by luma\_log2\_weight\_denom + delta\_chroma\_log2\_weight\_denom and it shall be in the range of 0 to 7, inclusive.

**luma\_weight\_l0\_flag[** i **]** equal to 1 specifies that weighting factors for the luma component of list 0 prediction using RefPicList0[ i ] are present. luma\_weight\_l0\_flag[i] equal to 0 specifies that these weighting factors are not present.

**chroma\_weight\_l0\_flag[** i **]** equal to 1 specifies that weighting factors for the chroma prediction values of list 0 prediction using RefPicList0[ i ] are present. chroma\_weight\_l0\_flag[i] equal to 0 specifies that these weighting factors are not present. When chroma\_weight\_l0\_flag[i] is not present, it shall be inferred to 0.

**delta\_luma\_weight\_l0[** i **]** is the difference of the weighting factor applied to the luma prediction value for list 0 prediction using RefPicList0[ i ].

The variable LumaWeightL0[ i ] is specified by (1 << luma\_log2\_weight\_denom ) + delta\_luma\_weight\_l0[ i ]. When luma\_weight\_l0\_flag is equal to 1, the value of LumaWeightL0[ i ] shall be in the range of −128 to 127, inclusive. When luma\_weight\_l0\_flagis equal to 0, LumaWeightL0[ i ] is inferred to be equal to 2luma\_log2\_weight\_denom for RefPicList0[ i ].

**luma\_offset\_l0[** i **]** is the additive offset applied to the luma prediction value for list 0 prediction using RefPicList0[ i ]. The value of luma\_offset\_l0[ i ] shall be in the range of −128 to 127, inclusive. When luma\_weight\_l0\_flagis equal to 0, luma\_offset\_l0[ i ] is inferred as equal to 0 for RefPicList0[ i ].

**delta\_chroma\_weight\_l0[** i **][** j **]** is the difference of the weighting factor applied to the chroma prediction values for list 0 prediction using RefPicList0[ i ] with j equal to 0 for Cb and j equal to 1 for Cr.

The variable ChromaWeightL0[ i ][ j ] is specified by ( 1 << ChromaLog2WeightDenom ) + delta\_chroma\_weight\_l0[ i ][ j ]. When chroma\_weight\_l0\_flag is equal to 1, the value of ChromaWeightL0[ i ][ j ] shall be in the range of −128 to 127, inclusive. When chroma\_weight\_l0\_flag is equal to 0**,** ChromaWeightL0[ i ][ j ] is inferred to be equal to 2ChromaLog2WeightDenom for RefPicList0[ i ].

**delta\_chroma\_offset\_l0[** i **][** j **]** is the difference of the additive offset applied to the chroma prediction values for list 0 prediction using RefPicList0[ i ] with j equal to 0 for Cb and j equal to 1 for Cr.

The variable ChromaOffsetL0[ i ][ j ] is specified as follows:

shift = 1 << ( BitDepthC − 1 )

ChromaOffsetL0[ i ][ j ] = (delta\_chroma\_offset\_l0[i][j] –   
 ( (shift\*ChromaWeightL0[ i ][ j ]) >> ChromaLog2WeightDenom ) − shift ) (7‑63)

The variable ChromaOffsetL0[ i ][ j ] shall be in the range of −127 to 128, inclusive. When chroma\_weight\_l0\_flag is equal to 0**,** ChromaOffsetL0[ i ][ j ] is inferred to be equal to 0 for RefPicList0[ i ].

**luma\_weight\_l1\_flag, chroma\_weight\_l1\_flag**, **delta\_luma\_weight\_l1**, **luma\_offset\_l1**, **delta\_chroma\_weight\_l1**, **delta\_chroma\_offset\_l1** have the same semantics as luma\_weight\_l0\_flag, delta\_luma\_weight\_l0, luma\_offset\_l0, chroma\_weight\_l0\_flag, delta\_chroma\_weight\_l0, delta\_chroma\_offset\_l0, respectively, with l0, list 0, and List0 replaced by l1, list 1, and List1, respectively.

The variable sumWeightL0Flags is specified as the sum of luma\_weight\_l0\_flag[i] + 2\*chroma\_weight\_l0\_flag[i], for i = 0..num\_ref\_idx\_l0\_active\_minus1.

For B-slices, the variable sumWeightL1Flags is specified as the sum of luma\_weight\_l1\_flag[i] + 2\*chroma\_weight\_l1\_flag[i], for i = 0..num\_ref\_idx\_l1\_active\_minus1.

For P-slices, sumWeightL0Flags shall not exceed 24. For B-slices, the sum of sumWeightL0Flags and sumWeightL1Flags shall not exceed 24.

# References

[1] F. Bossen, “Common test conditions and software reference configurations,” JCT-VC Document, JCTVC-I1100, 9th Meeting: Geneva, Switzerland, 27 April – 07 May, 2012

[2] [B. Bross](mailto:benjamin.bross@hhi.fraunhofer.de), [W.-J. Han](mailto:wjhan.han@samsung.com), [J.-R. Ohm](mailto:ohm@ient.rwth-aachen.de), [G. J. Sullivan](mailto:garysull@microsoft.com), [T. Wiegand](mailto:thomas.wiegand@hhi.fraunhofer.de) “High Efficiency Video Coding (HEVC) text specification draft 7,” JCT-VC Document, JCTVC-I1003, 9th Meeting: Geneva, Switzerland, 27 April – 07 May, 2012.

[3] M. Zhou, “AHG9: on slice header parsing overhead reduction”, JCTVC Document, JCTVC-J0083, 10th meeting: Stockholm, Sweden, July 12 – 20, 2012

# Patent rights declaration(s)

**Texas Instruments, Inc. and Apple Inc. 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).**