|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11**  8th Meeting: San José, CA, USA, 1–10 February, 2012 | Document: JCTVC-H0491 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **Non-CE6a: Remove the large multiplier for LM mode calculation** | | |
| *Status:* | Input Document to JCT-VC | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Lingzhi Liu (Hisilicon Technologies),  Guichun Li (Santa Clara University)  Yongbing Lin (Hisilicon Technologies) | Tel: Email: | +1-408-330-4996 [lingzhi.liu@huawei.com](mailto:lingzhi.liu@huawei.com)  [g1li@scu.edu](mailto:g1li@scu.edu)  [linyongbing@huawei.com](mailto:linyongbing@huawei.com) |
| *Source:* | Hisilicon Technologies & Santa Clara University | | |

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

# Abstract

This contribution provides a solution to simplify the Chroma from Luma intra prediction (LM mode) in HEVC standard. The proposed method removes the large 13 (AI-HE) or 15-bit (AI-HE10) multipliers by 8 or 10-bit multipliers through algorithm optimization. The BD-Rate for this optimization is: AI-HE: Y: 0.0%, U: 0.1%, V: 0.0% and AI-HE10: Y: 0.0%, U: 0.1%, V: 0.0%. These modifications will contribute to complexity reduction both for HW and SW implementation.

# Introduction

For chroma blocks, the current design of High Efficiency Video Coding (HEVC) has a chroma from luma intra prediction mode, which is also referred as LM mode, in addition to the directional modes and the DC mode.

According to the HEVC working draft [1] and the HM-5.0 software, the current design of LM mode uses reconstructed luma samples to generate chroma samples. The values of the prediction are derived as the following ordered steps [1]:

1. Variable k3 and the sample array pY’ are derived as:

k3 = Max( 0, BitDepthC + log2( nS ) – 14 ) (1)

pY’[ x, y ] = ( recSamplesL[ 2x-1, 2y+1 ] +   
2\*recSamplesL[ 2x, 2y+1 ] + recSamplesL[ 2x+1, 2y+1 ] + 2 ) >> 2, with x=0..nS-1, y = -1 (2)

pY’[ x, y ] = ( recSamplesL[ 2x, 2y ] + recSamplesL[ 2x, 2y+1 ] ) >> 1, with x=-1..nS-1, y = 0..nS-1 (3)

1. Variables L, C, LL, LC and k2 are derived as follows:

L =  (4)

C =  (5)

LL =  (6)

LC =  (7)

k2 = log2( (2\*nS) >> k3 ) (8)

1. Variables alpha, beta and k are derived as:

a1 = ( LC << k2 ) – L\*C (9)  
a2 = ( LL << k2 ) – L\*L (10)  
k1 = Max( 0, log2( abs( a2 ) ) – 5 ) – Max( 0, log2( abs( a1 ) ) – 14 ) + 2 (11)  
a1s = a1 >> Max(0, log2( abs( a1 ) ) – 14 ) (12)  
a2s = abs( a2 >> Max(0, log2( abs( a2 ) ) – 5 ) ) (13)  
a3 = a2s < 1 ? 0 : Clip3( -215, 215-1, a1s\*lmDiv + ( 1 << ( k1 – 1 ) ) >> k1 ) (14)

alpha = a3 >> Max( 0, log2( abs( a3 ) ) – 6 ) (15)  
k = 13 – Max( 0, log2( abs( alpha ) ) – 6 ) (16)

beta = ( L – ( ( alpha \* C ) >> k1 ) + ( 1 << ( k2 – 1 ) ) ) >> k2 (17)

where *lmDiv* is specified in a look-up table [1] with the input a2s.

1. The values of the prediction samples predSamples[ x, y ] are derived as:

predSamples[ x, y ] = Clip1C( ( ( pY’[ x, y ] \* alpha ) >> k ) + beta ), with x, y = 0..nS-1 (18)

# Computational complexity in current LM mode

In LM mode, the parameter of a1 and a2 are calculated through (9) and (10) respectively. Alpha is obtained through a3, where a3 is the division of a1/a2 which realized by a look-up table method according to (14). The calculation of (9) and (10) has the highest computational complexity in LM mode.

If the input bit-width is 8, both L and C will have maximum 13 bits, and both LC and LL will have maximum 21 bits. Because the maximum value of k2 is 5, each calculation of (9) and (10) then will have a 13-bit multiplier and a 26-bit subtractor.

If the input bit-width is 10, both L and C will have maximum 15 bits, and both LC and LL will have maximum 25 bits. Each calculation of (9) and (10) will have a 15-bit multiplier and a 30-bit subtractor.

The computation is probably the most complex among all the HEVC tools due to these large mathematical operations.

# Remove the computational complexity of LM mode

The calculation of (9) and (10) can be simplified by derivations as follows. And the large multipliers and substractors are shown to be simplified to get the alpha and beta.

Set



 (19)



 (20)

 (21)

where *avgY’* and *avgC* mean the floored average value of all the luma prediction samples and the reconstructed neighboring chroma samples respectively, while *RErrY’* and *RErrC* denote the corresponding rounding errors. The symbol ^ means bitwise exclusive or.

For easch Luma and chroma sample, we can set

 (22)

 (23)

,which are the rounding errors for each used samples. It is obvious that

 (24)

## Calculation of a2

From (4), (6), (8), (10) (19), (20) and (21) we have,

a2 = ( LL << k2 ) – L\*L





(25)

Ignoring the square of rounding error *RErrY’*, we have

 (26)

If the bit-width is 8, only 8-bit multipliers are needed for calculating a2’.

## Calculation of a1

Similarly, from (4), (5), (7), (8), (9), (19), (20) and (21) we have,

a1 = ( LC << k2 ) – L\*C





 (27)

Ignoring the multiplication of rounding error *RErrY’\*RErrC*, we have

 (28)

Also, if the bit-width is 8, only 8-bit multipliers are needed for the calculation of a1’.

## Calculation of alfa

Because alpha=a1/a2, using a1’ and a2’ to replace a1 and a2, from (26) and (28) we have

(29)

(30) implies that by introducing the parameters of *avgY’, RErrY’,* *avgC and RErrC*, no large multiplier more than 8-bit is required to calculate alpha and the subtrators are 16-bit.

## Calculation of beta

Beta is originally derived as follows [2]:

 (30)

Using the parameters of *avgY’*, *RErrY’, avgC, and RErrC*, the calculation is rewritten as bellowing

 (31)

We can also ignore the rounding errors of *RErrY’ and RErrC*, and have

 (32)

The bit-width of (32) is reduced compared with (17) and no shifting operation is needed.

# The new steps of LM mode

Based on the derivation of Section 3, the values of the prediction samples predSamples[ x, y ], are derived as the following new ordered steps:

……

1. Variables L, C, LL, and LC are derived as follows:

L =  (33)

C =  (34)

LL =  (35)

LC =  (36)

1. Variables alpha, beta and k are derived as:

avgY’ =  L >> log2( abs( 2\*nS ) )  (37)  
RErrY’= L ^ (avgY’<< log2( abs( 2\*nS ) ) ) (38)  
avgC =  C  >> log2( abs( 2\*nS ) )  (39)  
RErrC= C ^ ( avgC<< log2( abs( 2\*nS ) ) ) (40)  
a1 = LC– (2\*nS\*avgY’\*avgC+avgY’\*RErrC+avgC\*RErrY’) (41)  
a2 = LL – (2\*nS\*avgY’2+2\*avgY’\*RErrY’) (42)  
……

beta =  avgC – ( ( alpha \* avgY’ ) >> k1 )  (43)

……

From above we can see that the large multipliers in the original algorithm are removed, and the parameters of k2 and k3 are not necessary for the calculation. Because *avgY’, RErrY’, avgC*, and *RErrC* are all nonnegative integers, only 8 or 10-bit unsigned multipliers are needed for the calculations. Comparing the two algorithms, the calculation of beta in the new algorithm is also simpler than the original one.

# Another Expression for a1 and a2 calculation

From (23), (24), (25), (27) and (29), a1’ and a2’ can also be formed as



 (44)



 (45)

Then alpha’ in (30) is reformed as

 (46)

(46) means that alpha’ can also be obtained by a series of 8 or 10-bit signed multiplications.

The corresponding steps are

……

1. Variables avgY’, avgC are derived as follows:

avgY’ =  (47)

avgC =  (48)

1. Set ΔpY’[ -1, y ]= pY’[ -1, y ] - avgY’, ΔpY’[ x, -1 ]= pY’[ x, -1 ] - avgY’, Δp[ -1, y ]= p[ -1, y ] – avgC and Δp[ x, -1 ]= p[ x, -1 ] – avgC, with x=0..nS-1, y = 0..nS-1, Variables LL and LC are derived as follows:

LL =  (49)

LC =  (50)

1. Variables alpha, beta and k are derived as:

a1 = LC (51)  
a2 = LL (52)  
……

beta =  avgC – ( ( alpha \* avgY’ ) >> k1 )  (53)

……

The large multiplier and subtractor in the original algorithm are removed, and the parameters of L, C, k2 and k3 are not necessary for the calculation. In this algorithm, the differences between each pixel value and the average value are needed to be calculated, and the multipliers in LC calculation are signed.

# Complexity analysis

In this section, complexities of the original and proposed algorithms are analyzed by comparing the operation number.

Table 1 shows the operation number for the original LM mode and the proposed two methods. Proposed 1 is described in Section 4 and Proposed 2 is described in Section 5.

Tab.4 operation number of proposed LM mode (8-bit config)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Step | Original LM | | | Prop. 1 | | | Prop. 2 | | |
| Mul | Add | Shift | Mul | Add | Shift | Mul | Add | Shift |
| Cal. of L, C | - | 2\*2N M-b | - | - | 2\*2N M-b | - | - | 2\*2N M-b | - |
| Cal. of ΔpY, Δp | - | - | - | - | - | - | - | 2\*2N 8-b | - |
| Cal. of LL, LC | 2\*2N 8-b | 2\*2N M-b | - | 2\*2N 8-b | 2\*2N M-b |  | 2N 8-b 2N 8-b (signed) | 2\*2N M-b | - |
| Cal. of *avgY’, avgC* | - | - | - | - | - | 2 | - | - | 2 |
| Cal. of *RErrY’, RErrC* | - | - | - | - | 2 xor | 2 | - | - | - |
| Cal. of a1 | 1 13-b | 1 26-b | 1 | 3 8-b | 2 16-b 1 8-b | 2 | - | - | - |
| Cal. of a2 | 1 13-b | 1 26-b | 1 | 2 8-b | 2 16-b | 2 | - | - | - |
| Cal. of beta | 1 13-b | 2 13-b | 3 | 1 8-b | 1 8-b | 1 | 1 8-b | 1 8-b | 1 |
| **Total operations** | **4N 8-b 3 13-b** | **8N M-b 2 13-b 2 26-b** | **5** | **4N+6 8-b** | **8N M-b 2 8-b 4 16-b 2 xor** | **9** | **2N+1 8-b 2N 8-b (signed)** | **8N M-b 4N 8-b** | **3** |

Here N=4, 8 or 16, and M-b means multiple bit-length operators will be used.

# Simulation results

Simulations have been performed to test the performance of the new simplified algorithms.

Table 2 shows the result of comparing the new algorithm described in Section 4 and Section 5 with HM5.0 software anchor.

Table 2. Testing results of proposed algorithm in Section 4 and Section 5 vs. HM 5.0

|  |  |  |  |
| --- | --- | --- | --- |
|  | **All Intra HE** | | |
|  | Y | U | V |
| Class A (8bit) | 0.0% | 0.1% | 0.1% |
| Class B | 0.0% | 0.3% | 0.0% |
| Class C | 0.0% | 0.1% | 0.0% |
| Class D | 0.0% | 0.1% | 0.0% |
| Class E | 0.0% | 0.0% | 0.0% |
| **Overall** | 0.0% | 0.1% | 0.0% |
|  | 0.0% | 0.1% | 0.0% |
| Class F | 0.0% | 0.0% | 0.2% |
| Enc Time[%] | 99% | | |
| Dec Time[%] | 97% | | |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **All Intra HE10** | | |
|  | Y | U | V |
| Class A | 0.0% | 0.2% | 0.2% |
| Class B | 0.0% | 0.1% | 0.0% |
| Class C | 0.0% | 0.0% | 0.1% |
| Class D | 0.0% | 0.0% | 0.0% |
| Class E | 0.0% | 0.0% | 0.0% |
| **Overall** | 0.0% | 0.1% | 0.0% |
|  | 0.0% | 0.1% | 0.0% |
| Class F | 0.0% | 0.1% | 0.2% |
| Enc Time[%] | 103% | | |
| Dec Time[%] | 100% | | |

The performance is almost the same for the original and the proposed two cases which also confirmed that the simplification has neglectable impact on coding performance.

# Conclusion

It is proposed to consider adopting the optimization of removing the large multiplier and subtractor, and using 8-bit multiplication for LM mode calculation.

# Reference

[1] Benjamin Bross, et. Al, “WD5: Working Draft 5 of High-Efficiency Video Coding”, ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, JCTVC-G1103

[2] Jianle Chen, et. Al, “Complexity reduction of chroma intra LM prediction mode”, ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, JCTVC-F494

[3] Frank Bossen, “Common HM test conditions and software reference configurations”, ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11, JCTVC-G1200

# Patent rights declaration(s)

**Hisilicon Technologies and Santa Clara University 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).**