|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11**  5th Meeting: Geneva, 16-23 March, 2011 | Document: JCTVC-E242 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **On clipping in bi-directional averaging** | | |
| *Status:* | Input Document to JCT-VC | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Kemal Ugur Jani Lainema | Tel: Email: | +358 50 4860857 kemal.ugur@nokia.com |
| *Source:* | Nokia | | |

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

# Abstract

This contribution presents a minor simplification in clipping operations involved in bi-directional averaging. Using the proposal, the intermediate prediction values for each one of the directions are not clipped but a single clipping is performed after bi-directional averaging is performed. The change in coding efficiency using the proposal is reported to be negligible.

# Introduction



Currently, the bidirectional averaging in HM is performed as follows (for simplicity, let’s assume the motion vector points to location b0,0 and input bit-depth is 8):

**Step -1**: Compute prediction sample values for each one of the direction (PL0, PL1). These values are in 14 bit accuracy and it is clipped between [0 – 16383].

PL0 = Clip3(0, (1<<14) - 1, ( -A-3,0 + 4\*A-2,0 - 11\*A-1,0 + 40\*A0,0 +  40\*A1,0 - 11\*A2,0 + 4\*A3,0 - A4,0 ) )

PL1 = Clip3(0, (1<<14) - 1, ( -A-3,0 + 4\*A-2,0 - 11\*A-1,0 + 40\*A0,0 +  40\*A1,0 - 11\*A2,0 + 4\*A3,0 - A4,0 ) )

**Step - 2**: The final prediction sample P is computed as average of PL0 and PL1.

P = (PL0 + PL1 + 64) >> 7

The proposal makes a small modification to the above process as follows:

**Step – 1:** The prediction sample values for each one of the directions are not clipped.

PL0 = ( -A-3,0 + 4\*A-2,0 - 11\*A-1,0 + 40\*A0,0 +  40\*A1,0 - 11\*A2,0 + 4\*A3,0 - A4,0 )

PL1 = ( -A-3,0 + 4\*A-2,0 - 11\*A-1,0 + 40\*A0,0 +  40\*A1,0 - 11\*A2,0 + 4\*A3,0 - A4,0 )

**Step – 2:** The final prediction sample P is computed as average of PL0 and PL1 but clipped as [0 – 255]

P = Clip (PL0 + PL1 + 64) >> 7)

# Experimental Results

Proposal has negligible impact on coding efficiency (0.0% on average) as seen in the below results.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  | Random access |  |  | Random access LoCo |  |
| Y BD-rate | U BD-rate | V BD-rate | Y BD-rate | U BD-rate | V BD-rate |
| Class A | 0.0 | -0.1 | -0.1 | 0.1 | 0.0 | 0.0 |
| Class B | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| Class C | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| Class D | 0.0 | 0.0 | 0.0 | 0.0 | -0.1 | 0.0 |
| Class E |  |  |  |  |  |  |
| All | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Low delay | | | Low delay LoCo | | |
|  | Y BD-rate | U BD-rate | V BD-rate | Y BD-rate | U BD-rate | V BD-rate |
| Class A |  |  |  |  |  |  |
| Class B | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| Class C | 0.0 | -0.1 | -0.1 | 0.0 | 0.0 | 0.1 |
| Class D | 0.0 | 0.2 | 0.5 | 0.0 | 0.2 | 0.0 |
| Class E | 0.0 | 0.0 | 0.2 | 0.0 | -0.5 | -0.2 |
| All | 0.0 | 0.0 | 0.1 | 0.0 | 0.0 | 0.0 |

# Proposed text

**Changes in section 8.4.2.2.1**

a0,0 = ( -A-3,0 + 4\*A-2,0 - 10\*A-1,0 + 57\*A0,0 +  
  19\*A1,0 - 7\*A2,0 + 3\*A3,0 - A4,0 + offset1) >> shift1

b0,0 = ( -A-3,0 + 4\*A-2,0 - 11\*A-1,0 + 40\*A0,0 +  
  40\*A1,0 - 11\*A2,0 + 4\*A3,0 - A4,0 + offset1) >> shift1

c0,0 = ( -A-3,0 + 3\*A-2,0 - 7\*A-1,0 + 19\*A0,0 +  
  57\*A1,0 - 10\*A2,0 + 4\*A3,0 - A4,0 + offset1) >> shift1

d0,0 = ( -A0,-3 + 4\*A0,-2 - 10\*A0,-1 + 57\*A0,0 +  
  19\*A0,1 - 7\*A0,2 + 3\*A0,3 - A0,4 + offset1) >> shift1

h0,0 ( -A0,-3 + 4\*A0,-2 - 11\*A0,-1 + 40\*A0,0 +  
  40\*A0,1 - 11\*A0,2 + 4\*A0,3 - A0,4 + offset1) >> shift1

n0,0 = ( -A0,-3 + 3\*A0,-2 - 7\*A0,-1 + 19\*A0,0 +  
  57\*A0,1 - 10\*A0,2 + 4\*A0,3 - A0,4 + offset1) >> shift1

...

e0,0 = ( -d1-3,0 + 4\*d1-2,0 - 10\*d1-1,0 + 57\*d10,0 +  
  19\*d11,0 - 7\*d12,0 + 3\*d13,0 - d14,0 + offset2) >> shift2

f0,0 = ( - d1-3,0 + 4\*d1-2,0 - 11\*d1-1,0 + 40\*d10,0 +  
  40\*d11,0 - 11\*d12,0 + 4\*d13,0 - d14,0 + offset2) >> shift2

g0,0 = ( - d1-3,0 + 3\*d1-2,0 - 7\*d1-1,0 + 19\*d10,0 +  
  57\*d11,0 - 10\*d12,0 + 4\*d13,0 - d14,0 + offset2) >> shift2

i0,0 = ( - h1-3,0 + 4\*h1-2,0 - 10\*h1-1,0 + 57\*h10,0 +  
  19\*h11,0 - 7\*h12,0 + 3\*h13,0 - h14,0 + offset2) >> shift2

j0,0 = ( - h1-3,0 + 4\*h1-2,0 - 11\*h1-1,0 + 40\*h10,0 +  
  40\*h11,0 - 11\*h12,0 + 4\*h13,0 - h14,0 + offset2) >> shift2

k0,0 = ( - h1-3,0 + 3\*h1-2,0 - 7\*h1-1,0 + 19\*h10,0 +  
  57\*h11,0 - 10\*h12,0 + 4\*h13,0 - h14,0 + offset2) >> shift2

p0,0 = ( - n1-3,0 + 4\*n1-2,0 - 10\*n1-1,0 + 57\*n10,0 +  
  19\*n11,0 - 7\*n12,0 + 3\*n13,0 - n14,0 + offset2) >> shift2

q0,0 = ( - n1-3,0 + 4\*n1-2,0 - 11\*n1-1,0 + 40\*n10,0 +  
  40\*n11,0 - 11\*n12,0 + 4\*n13,0 - n14,0 + offset2) >> shift2

r0,0 = ( - n1-3,0 + 3\*n1-2,0 - 7\*n1-1,0 + 19\*n10,0 +  
  57\*n11,0 - 10\*n12,0 + 4\*n13,0 - n14,0 + offset2) >> shift2

**Changes in section 8.4.2.2.2**

ab0,0 = ( -3\*B-1,0 + 60\*B0,0 + 8\*B1,0 – B2,0 + offset1 ) >> shift1

ac0,0 = ( -4\*B-1,0 + 54\*B0,0 + 16\*B1,0 – 2\*B2,0 + offset1 ) >> shift1

ad0,0 = ( -5\*B-1,0 + 46\*B0,0 + 27\*B1,0 – 4\*B2,0 + offset1 ) >> shift1

ae0,0 = ( -4\*B-1,0 + 36\*B0,0 + 36\*B1,0 – 4\*B2,0 + offset1 ) >> shift1

af0,0 = ( -4\*B-1,0 + 27\*B0,0 + 46\*B1,0 – 5\*B2,0 + offset1 ) >> shift1

ag0,0 = ( -2\*B-1,0 + 16\*B0,0 + 54\*B1,0 – 4\*B2,0 + offset1 ) >> shift1

ah0,0 = ( -B-1,0 + 8\*B0,0 + 60\*B1,0 – 3\*B2,0 + offset1 ) >> shift1

ba0,0 = ( -3\*B0,-1 + 60\*B0,0 + 8\*B0,1 – B0,2 + offset1 ) >> shift1

ca0,0 = ( -4\*B0,-1 + 54\*B0,0 + 16\*B0,1 – 2\*B0,2 + offset1 ) >> shift1

da0,0 = ( -5\*B0,-1 + 46\*B0,0 + 27\*B0,1 – 4\*B0,2 + offset1 ) >> shift1

ea0,0 = ( -4\*B0,-1 + 36\*B0,0 + 36\*B0,1 – 4\*B0,2 + offset1 ) >> shift1

fa0,0 = ( -4\*B0,-1 + 27\*B0,0 + 46\*B0,1 – 5\*B0,2 + offset1 ) >> shift1

ga0,0 = ( -2\*B0,-1 + 16\*B0,0 + 54\*B0,1 – 4\*B0,2 + offset1 ) >> shift1

ha0,0 = ( -B0,-1 + 8\*B0,0 + 60\*B0,1 – 3\*B0,2 + offset1 ) >> shift1

…

Xb0,0 = ( -3\*Xa-1,0 + 60\*Xa0,0 + 8\*Xa1,0 – Xa2,0 + offset2 ) >> shift2

Xc0,0 = ( -4\*Xa-1,0 + 54\*Xa0,0 + 16\*Xa1,0 – 2\*Xa2,0 + offset2 ) >> shift2

Xd0,0 = ( -5\*Xa-1,0 + 46\*Xa0,0 + 27\*Xa1,0 – 4\*Xa2,0 + offset2 ) >> shift2

Xe0,0 = ( -4\*Xa-1,0 + 36\*Xa0,0 + 36\*Xa1,0 – 4\*Xa2,0 + offset2 ) >> shift2

Xf0,0 = ( -4\*Xa-1,0 + 27\*Xa0,0 + 46\*Xa1,0 – 5\*Xa2,0 + offset2 ) >> shift2

Xg0,0 = ( -2\*Xa-1,0 + 16\*Xa0,0 + 54\*Xa1,0 – 4\*Xa2,0 + offset2 ) >> shift2

Xh0,0 = ( -Xa-1,0 + 8\*Xa0,0 + 60\*Xa1,0 – 3\*Xa2,0 + offset2 ) >> shift2

...

**Changes in section 8.4.2.2.3**

– Otherwise (both predFlagL0 and predFlagL1 are equal to 1),

predSamples[ x, y ] = Clip1 C  
  ( ( predSamplesL0[ x, y ] + predSamplesL1[ x, y ]  ) + offset3 ) >> shift3

# Patent rights declaration(s)

**Nokia 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).**