|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11**  11th Meeting: Shanghai, CN, 10–19 Oct. 2012 | Document: JCTVC-K0101 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **Line buffer cleanup** | | |
| *Status:* | Input Document to JCT-VC | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Pieter Kapsenberg (Intel Corporation)  Wenhao Zhang (Intel Corporation)  CH7-427 5000 W Chandler Blvd Chandler AZ, 85226 | Tel: Email: | +1 480 552 5481 pieter.kapsenberg@intel.com |
| *Source:* | Intel Corp | | |

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

# Abstract

This contribution claims that the line buffer reduction technique adopted previously no longer has any benefit due to the removal of biprediction on 8x4 and 4x8 sized blocks. It is suggested that the relevant code and text is removed from the standard, which has the additional benefit of harmonizing the column buffer needed for in-loop filtering. This contribution includes a suggested HM-8.0 patch.

# Introduction

Presently, the draft standard includes a line buffer storage reduction technique between CTB rows as illustrated below:



Figure - Line buffer reduction technique

Now that 4x8 and 8x4 PUs are no longer allowed to be bidirectional, an 8 pixel region cannot have more than two upper motion vectors neighbors anyway:

* 8x8 or larger bidirectional PU
* Two 8x4 neighbors, which must be unidirectional.

Thus the benefit of this technique is negated, and current scheme only serves to complicate the standard.

# Proposed solution: Removal of reduction technique

Removing the reduction technique has the following benefits:

* Reduction of complexity, upper neighbor derivation is easy to understand and consistent throughout the picture.
* Storage of data for vertical tile boundaries (necessary for in-loop filtering) is harmonized with horizontal line storage.
* No extra bandwidth is incurred.

# Simulations results

Note that the encoder and decode times are not meaningful because of the variable load on the compute clusters.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **All Intra Main** | | | **All Intra HE10** | | |
|  | Y | U | V | Y | U | V |
| Class A | 0.0% | 0.0% | 0.0% | 0.0% | 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% | 0.0% | 0.0% |
| Class E | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% |
| Class F | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% |
| **Overall** | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% |
|  | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% |
| Enc Time[%] | 107% | | | 107% | | |
| Dec Time[%] | #NUM! | | | #NUM! | | |
|  |  |  |  |  |  |  |
|  | **Random Access Main** | | | **Random Access HE10** | | |
|  | Y | U | V | Y | U | V |
| Class A | 0.0% | 0.0% | 0.0% | 0.0% | 0.1% | -0.1% |
| Class B | 0.0% | 0.0% | 0.1% | 0.0% | 0.0% | 0.1% |
| Class C | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | -0.1% |
| Class D | 0.0% | -0.1% | 0.0% | -0.1% | -0.2% | -0.2% |
| Class E |  |  |  |  |  |  |
| Class F | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% |
| **Overall** | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% |
|  | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% |
| Enc Time[%] | 112% | | | 109% | | |
| Dec Time[%] | #NUM! | | | #NUM! | | |
|  |  |  |  |  |  |  |
|  | **Low delay B Main** | | | **Low delay B HE10** | | |
|  | Y | U | V | Y | U | V |
| Class A |  |  |  |  |  |  |
| Class B | 0.0% | 0.3% | -0.1% | 0.0% | 0.0% | 0.0% |
| Class C | 0.1% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% |
| Class D | -0.1% | 0.2% | -0.4% | 0.0% | -0.1% | -0.3% |
| Class E | 0.1% | 0.0% | 0.7% | 0.0% | 0.3% | -0.1% |
| Class F | 0.0% | 0.0% | 0.0% | -0.1% | 0.0% | -0.3% |
| **Overall** | 0.0% | 0.1% | 0.0% | 0.0% | 0.0% | -0.1% |
|  | 0.0% | 0.1% | 0.1% | 0.0% | 0.0% | -0.2% |
| Enc Time[%] | 108% | | | 108% | | |
| Dec Time[%] | #NUM! | | | #NUM! | | |
|  |  |  |  |  |  |  |
|  | **Low delay P Main** | | | **Low delay P HE10** | | |
|  | Y | U | V | Y | U | V |
| Class A |  |  |  |  |  |  |
| Class B | 0.0% | 0.0% | 0.6% | 0.0% | 0.1% | -0.4% |
| Class C | 0.0% | 0.0% | 0.2% | 0.0% | -0.1% | -0.1% |
| Class D | 0.0% | 0.6% | -0.1% | 0.1% | -1.0% | 0.0% |
| Class E | 0.0% | 0.4% | -0.5% | 0.1% | -0.5% | 0.5% |
| Class F | 0.0% | 0.3% | 0.4% | -0.1% | -0.1% | 0.0% |
| **Overall** | 0.0% | 0.2% | 0.2% | 0.0% | -0.3% | 0.0% |
|  | 0.0% | 0.2% | 0.2% | 0.0% | -0.3% | 0.0% |
| Enc Time[%] | 106% | | | 111% | | |
| Dec Time[%] | #NUM! | | | #NUM! | | |

# Draft Text changes for Solution

**8.5.2.1.6 Derivation process for motion vector predictor candidates**

<snip>

The motion vector mvLXB and the availability flag availableFlagLXB are derived in the following ordered steps:

1. Let a set of three sample location (xBk, yBk), with k = 0,1,2, specifies sample locations with xB0 = xP + nPbW, xB1 = xB0− 1, xB2 = xP − 1 and yBk = yP − 1. The set of sample locations ( xBk, yBk ) represent the sample locations immediately to the upper side of the above partition boundary and its extended line.
2. **~~When yP−1 is less than (( yC >> Log2CtbSizeY ) << Log2CtbSizeY), the following applies.~~**

**~~xB~~~~0~~ ~~= (xB~~~~0~~~~>>3)<<3) + ((xB~~~~0~~~~>>3)&1)\*7 (8‑131)  
xB~~~~1~~ ~~= (xB~~~~1~~~~>>3)<<3) + ((xB~~~~1~~~~>>3)&1)\*7 (8‑132)  
xB~~~~2~~ ~~= (xB~~~~2~~~~>>3)<<3) + ((xB~~~~2~~~~>>3)&1)\*7 (8‑133)~~**

1. The availability flag availableFlagLXB is set equal to 0 and the both components of mvLXB are set equal to 0.

**8.7.2.3 Derivation process of boundary filtering strength**

<snip>

Depending on p0 and q0, the variable bS[ xDi ][ yDj ] is derived as follows.

* If the sample p0 or q0 is in the luma coding block of a coding unit coded with intra prediction mode, the variable bS[ xDi ][ yDj ] is set equal to 2.
* Otherwise, if the block edge is also a transform block edge and the sample p0 or q0 is in a luma transform block which contains non-zero transform coefficient level.
* Otherwise, the following applies.
  + - **~~When edgeType is equal to EDGE\_HOR and yC + yD~~~~j~~~~− 1 is less than (( yC >> Log2CtbSizeY ) << Log2CtbSizeY), sample p0 = recPicture~~~~L~~~~[ xL ][ yC + yD~~~~j~~~~– 1 ] where xL is equal to ((( xC + xD~~~~i~~~~) >> 3) << 3) + ((( xC + xD~~~~i~~~~) >> 3) & 1) \* 7.~~**
    - If one or more of the following conditions are true, the variable bS[ xDi ][ yDj ] is set equal to 1.

# Patent rights declaration(s)

**Intel Corporation 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).**