|  |  |
| --- | --- |
| **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-F556 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **CE6.b: SDIP Harmonization with Deblocking, MDIS and HE Residual Coding** | | |
| *Status:* | Input Document to JCT-VC | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Geert Van der Auwera, Joel Sole, Yunfei Zheng, Xianglin Wang, In Suk Chong, Rajan Joshi, Marta Karczewicz 5775 Morehouse Dr San Diego, CA 92121 USA | Email: | [geertv@qualcomm.com](mailto:geertv@qualcomm.com)  [joels@qualcomm.com](mailto:joels@qualcomm.com)  [zhengy@qualcomm.com](mailto:zhengy@qualcomm.com)  [xianglin@qualcomm.com](mailto:xianglin@qualcomm.com)  [martak@qualcomm.com](mailto:martak@qualcomm.com) |
| *Source:* | Qualcomm Inc. | | |

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

# Abstract

This contribution contains proposals on the harmonization of SDIP, in the context of CE6.b, with deblocking, mode-dependent intra smoothing (MDIS) and high efficiency residual coding. In addition, results are included for mode-dependent coefficient scanning (MDCS) and DC prediction filtering.

# CE6.b Test 7: SDIP Harmonization with Deblocking

## Summary

This contribution addresses the reduction of blocking artifacts of short-distance intra prediction (SDIP) partitions, which is an intra coding technique under study in CE6.b. The proposed SDIP deblocking method makes a distinction between the SDIP partition orientations with respect to the edge to be deblocked and adapts filters based on SDIP partition types. The method supports the deblocking parallelism of HM3. The average gain is -0.1% for both “All Intra” coding conditions, while the encoding and decoding times are comparable to HM3-SDIP.

## Problem Statement

The in-loop deblocking method included in the HM3 test model [1] computes decisions and performs filtering on the basis of a smallest block for deblocking of size 8×8 pixels. The deblocking decisions and filters for processing the edges of the smallest block are such that there is no overlap with other edges of prediction units (PU) or transform units (TU) besides the edge being deblocked, because a maximum of four pixels on either side of the edge are used and the smallest unit is of size 4×4 pixels. Decision and filtering results may become unreliable when other PU or TU edges are crossed and pixels from other block edge neighborhoods are used in the computations.

Short-distance intra prediction [2] (SDIP) partitions may have various rectangular sizes per coding unit (CU). enumerates examples of rectangular PU sizes of SDIP. Several of the PUs have a dimension that is smaller than the smallest 8×8 block size for deblocking. For example, PU sizes 1×16, 16×1, 2×8, 8×2 will cause the HM3 deblocking decisions and filters to cross one or several SDIP PU edges within the smallest block for deblocking.

Table 1 Examples of SDIP rectangular PU sizes (in pixels) per CU size

|  |  |
| --- | --- |
| CU 32×32 | 8×32, 32×8 |
| CU 16×16 | 4×16, 16×4, 1×16, 16×1 |
| CU 8×8 | 2×8, 8×2 |

The following are examples of deblocking decisions computed in the neighborhood of the current edge to be deblocked:

* Deblocking or no deblocking is decided for an edge consisting of 8 pixel lines
* Filter type decision: strong or weak filter

Deblocking uses pixels from the neighborhood of the current edge to be deblocked. Hence, both the deblocking decisions and filters have a certain support region on both sides of the edge. illustrates the filter and decision support region around the current edge to be deblocked. In this example, the support region contains an edge from an 8×2 SDIP PU and potentially the decisions and filters will cross this edge. Depending on the SDIP PU types in the neighborhood of the current edge, alternative deblocking decisions or filters with adapted support regions are required that do not cross the SDIP PU edges. This is detailed in the remainder of this proposal.

Figure 1 Filter and decision support region containing edge from 8×2 SDIP PU

8×8 CUs

8×2 SDIP PU

Current edge

Filter and decision support region

The proposed solution is to adapt the deblocking depending on the direction of the SDIP partitions with respect to the direction of the current edge to be deblocked. illustrates the SDIP partition directions for the vertical edge to be deblocked in between regions P and Q. For a horizontal edge in between vertically arranged P and Q regions, identical cases apply but rotated by 90 degrees. For example, the regions P and Q can be blocks of 8×8 pixels.

For each of the 9 cases identified in , the following deblocking strategies can be applied:

* (a): Both P and Q do not contain internal edges originating from SDIP partitions. Therefore, no special adaptation of deblocking decisions and filters is required.
* (b), (d), and (e): Regions P and/or Q contain internal edges from horizontal SDIP partitions. The internal SDIP edges are perpendicular to the current edge to be deblocked. Therefore, when horizontal deblocking (computation of decisions and filtering) is performed, no special adaptation of the deblocking decisions and filters is required.
* (c), (g), and (i): Regions P and/or Q contain internal edges from vertical SDIP partitions. The internal SDIP edges are parallel to the current edge to be deblocked. Therefore, when horizontal deblocking is performed, it is possible that the deblocking decisions and filters cross the internal SDIP edges. In this case, the decisions and filters may be adapted in order to avoid this problem.
* (f), (h): Regions P and Q contain internal edges from SDIP partitions with perpendicular directions with respect to each other. In this case, the deblocking decisions and filters may be adapted to avoid crossing the internal edges.

The adaptation of the deblocking method consists of modifying the deblocking decisions and the filters such that the support region size is decreased.

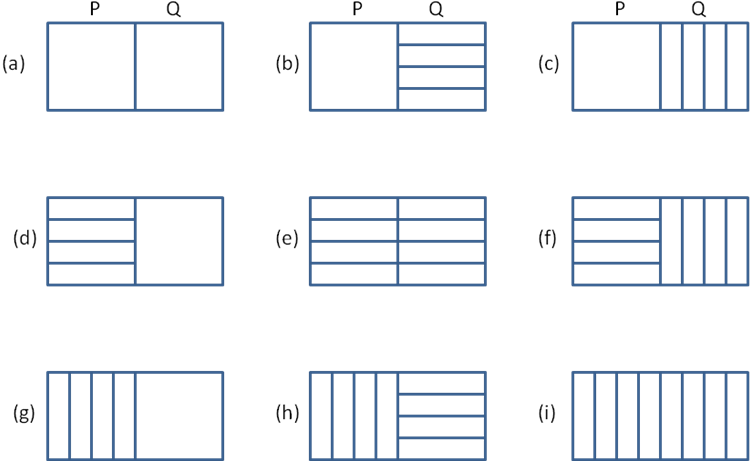


Figure 2 Several SDIP partition cases for the vertical edge to be deblocked in between regions P and Q. The cases for a horizontal edge in between a vertical arrangement of regions P and Q are similar.

## SDIP Deblocking Description

### Deblocking of CU-Boundary Edge with At Least One Adjacent SDIP Partition That Is Parallel with Edge

enumerates the deblocking filter types and deblocking decision conditions for deblocking of a CU-boundary edge (horizontal or vertical direction) between two blocks, block A and block B (can be on either side of edge), when at least one of the blocks contains a whole or partial SDIP-PU that is parallel with and adjacent to the current CU-boundary edge to be deblocked.

Deblocking filters and decision conditions are such that no SDIP-PU edges or other non-SDIP-PU or TU edges are crossed, which can lead to erroneous deblocking of the current edge. The proposed SDIP deblocking filtering supports parallel vertical and horizontal derivation of deblocking decision computations as is the case for HM3 deblocking, because pre-deblocked (reconstructed) samples are used. In addition, the parallel vertical or horizontal filtering of HM3 is also supported, because the filter results are computed independently from each other. Filters and decision conditions are further detailed below.

In case that neither block A nor block B is of the SDIP type, or the SDIP-PU orientation (longest dimension) is perpendicular to the CU-boundary edge to be filtered, then HM3 deblocking is applied. HM3 deblocking is applied to block edges that are unrelated to SDIP.

Table 2 Specification of SDIP deblocking filter types and decision conditions in case of a CU-boundary edge with at least one adjacent SDIP partition (SDIP-PU) that is parallel with the edge between blocks A and B. The edge has the horizontal or vertical direction.

|  |  |  |  |
| --- | --- | --- | --- |
| **Block A** | **Block B** | **Filter Type** | **Decision Conditions** |
| 32x8 | Not SDIP type or perpendicular SDIP-PU | Strong or Weak4 | Deblock edge on/off  Strong or weak filter |
| 32x8 | 32x8 | Strong or Weak4 | Deblock edge on/off  Strong or weak filter |
| 32x8 | 16x4 | Weak2 | Deblock edge on/off |
| 32x8 | 16x1 | Weak1 | Deblock edge on/off |
| 32x8 | 8x2 | Weak1 | Deblock edge on/off |
| 16x4 | Not SDIP type or perpendicular SDIP-PU | Weak2 | Deblock edge on/off |
| 16x4 | 16x4 | Weak2 | Deblock edge on/off |
| 16x4 | 16x1 | Weak1 | Deblock edge on/off |
| 16x4 | 8x2 | Weak1 | Deblock edge on/off |
| 8x2 | Not SDIP type or perpendicular SDIP-PU | Weak1 | Deblock edge on/off |
| 8x2 | 16x1 | Weak1 | Deblock edge on/off |
| 8x2 | 8x2 | Weak1 | Deblock edge on/off |
| 16x1 | Not SDIP type or perpendicular SDIP-PU | Weak1 | Deblock edge on/off |
| 16x1 | 16x1 | Weak1 | Deblock edge on/off |
| Not SDIP type or perpendicular SDIP-PU | Not SDIP type or perpendicular SDIP-PU | HM3 deblocking filters | HM3 deblocking decision conditions |

illustrates the naming and numbering of pixels on both sides of the current edge (for example, length 8 pixels) to be deblocked between blocks A and B. The naming and numbering is identical for a horizontal edge. The pixel values p or q can be either pre-deblocking (reconstructed) or deblocking filtered. The pixel values p’ and q’ represent the result after deblocking filtering.



Figure 3 Example of vertical edge of length 8 pixels between blocks A and B. The pixel rows are numbered from 0 to 7.

specifies the equations to compute the “Strong” (cfr. HM3 [1]), “Weak4” (cfr. [4]), “Weak2” (cfr. [4]), and “Weak1” deblocking filter types, while specifies the deblocking decision conditions (cfr. [3]) for determining whether to deblock the current edge (for example, length 8 pixels) or for selection between the “Strong” and “Weak4” filters.

Table 3 SDIP deblocking filters

|  |  |
| --- | --- |
| **Strong Filter** | p0’ =  ( p2 + 2\*p1 + 2\*p0 + 2\*q0 + q1 + 4 )/8  p1’ =  ( p2 + p1 + p0 + q0 + 2 )/4  p2’ =  ( 2\*p3 + 3\*p2 + p1 + p0 + q0 + 4 )/8  q0’ =  ( p1 + 2\*p0 + 2\*q0 + 2\*q1 + q2 + 4 )/8  q1’ =  ( p0 + q0 + q1 + q2 + 2 )/4  q2’ = ( p0 + q0 + q1 + 3\*q2 + 2\*q3 + 4 )/8 |
| **Weak4 Filter** | Δ =  ( 9\*(q0-p0) - 3\*(q1-p1) + 8 )/16  Δ = Clip( -tc, tc, Δ ) (tc is HM3 deblocking threshold [7], depending on QP)  p0’ = p0 + Δ  q0’ = q0 - Δ  Δp = Clip( - (tc/2), tc/2, ( (p2 + p0 + 1)/2 - p1 + Δ + 1 )/2 )  p1’ = p1 + Δp  Δq = Clip( - (tc/2), tc/2, ( (q2 + q0 + 1)/2 - q1 - Δ + 1)/2 )  q1’ = q1 + Δq |
| **Weak2 Filter** | Δ =  ( 9\*(q0 - p0) - 3\*(q1 - p1) + 8)/16  Δ = Clip( -tc, tc, Δ ) (tc is HM3 deblocking threshold [7], depending on QP)  p0’ = p0 + Δ  q0’ = q0 - Δ |
| **Weak1 Filter** | Δ =  ( 5\*(q0 - p0) + 8)/16  Δ = Clip( -tc, tc, Δ ) (tc is HM3 deblocking threshold [7], depending on QP)  p0’ = p0 + Δ  q0’ = q0 - Δ |

Table 4 Overview of deblocking decision conditions that are computed using pre-deblocking pixels

|  |  |
| --- | --- |
| **Deblock edge on/off** | |p0i – q0i| < α2  |p1i – p0i| < β2  |q1i – q0i| < β2  For i=2 and i=5 (see )  If all conditions are true, then edge is deblocked  α2 = (3\*α+2)>>2  β2 = (3\*β+2)>>2  (α and β are thresholds specified in [3], depending on QP) |
| **Strong or weak filter** | |p0 – q0| < δ  |q1 – q0| + |q2 – q0| + |q3 – q0| + |p1 – p0| + |p2 – p0| + |p3 – p0| < μ  Conditions tested on pixel line basis (see )  If conditions are true, then strong filter is applied to the line, else weak filter  δ = ( α2 >> 2 ) – (α2 >> 4)  μ = (β – 4) << 2 |

### Deblocking of Internal SDIP-CU Partition Boundaries of SDIP-Level 1

This section describes the deblocking of internal SDIP-CU partition boundaries of SDIP-Level 1. SDIP-Level 1 refers to the partitioning of, for example, 32x32, 16x16, or 8x8 CUs into SDIP partitions (horizontal or vertical) of sizes 32x8, 16x4, or 8x2, respectively. SDIP-Level 2 refers to the further partitioning of the SDIP-Level 1 partitions, for example, the 16x4 partitions may be further split into 16x1 partitions (horizontal and vertical), which is the topic of the next section.

There are three internal edges in case of SDIP-CU sizes 32x32, 16x16, or 8x8. The internal edges are SDIP-PU boundaries. specifies what filtering type or decision conditions to apply for the edge in between blocks A and B. In this section, blocks A and B contain a whole or partial SDIP-PU that is parallel with and adjacent to the current internal edge to be deblocked.

Table 5 Specification of deblocking filter types and decision conditions in case of deblocking of internal SDIP-CU partition boundaries of SDIP-Level 1

|  |  |  |  |
| --- | --- | --- | --- |
| **Block A** | **Block B** | **Filter Type** | **Decision Conditions** |
| 32x8 | 32x8 | Strong or Weak4 | Deblock edge on/off  Strong or weak filter |
| 16x4 | 16x4 | Weak2 | Deblock edge on/off |
| 16x4 | 16x1 | Weak1 | Deblock edge on/off |
| 16x1 | 16x1 | Weak1 | Deblock edge on/off |
| 8x2 | 8x2 | Weak1 | Deblock edge on/off |

### Deblocking of Internal SDIP-CU Partition Boundaries of SDIP-Level 2

For example, in the case of a 16x4 SDIP-PU (horizontal or vertical) that is further divided into 16x1 SDIP-PUs, the three internal edges are not deblocked in order to preserve texture detail ().

Table 6 Specification of deblocking filter types and decision conditions in case of deblocking of internal SDIP-CU partition boundaries of SDIP-Level 2

|  |  |  |  |
| --- | --- | --- | --- |
| **Block A** | **Block B** | **Filter Type** | **Decision Conditions** |
| 16x1 | 16x1 | No filtering | No conditions |

## Results

reports the BD-rate results of SDIP deblocking compared to HM3-SDIP (rev866). The gain is on average -0.1%. The encoding times and decoding times are comparable to HM3-SDIP.

An informal viewing experiment indicated that blocking artifacts originating from SDIP partitions are effectively reduced by the proposed SDIP deblocking method.

Table 7 BD-rate performance results, encoding and decoding times, of the SDIP deblocking method compared to the HM3-SDIP anchor

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | All Intra HE | | | All Intra LC | | |
| Y | U | V | Y | U | V |
| Class A | -0.1 | 0.0 | 0.0 | -0.1 | 0.0 | 0.0 |
| Class B | 0.0 | 0.0 | 0.0 | -0.1 | 0.0 | 0.0 |
| Class C | -0.1 | 0.0 | 0.0 | -0.2 | 0.0 | 0.0 |
| Class D | -0.1 | 0.0 | 0.0 | -0.1 | 0.0 | 0.0 |
| Class E | -0.1 | -0.1 | -0.1 | -0.2 | -0.2 | -0.1 |
| **Overall** | **-0.1** | **0.0** | **0.0** | **-0.1** | **0.0** | **0.0** |
| Enc Time[%] | 100% | | | 101% | | |
| Dec Time[%] | 100% | | | 101% | | |

## Conclusion of SDIP Harmonization with Deblocking

This contribution introduced a deblocking filter targeting the SDIP blocking artifacts. The deblocking filter has special adaptations for deblocking decisions and filters, so that no additional edges are included in the support regions of decisions and filters. The SDIP PU direction with respect to the edge to be deblocked is taken into account. In case of perpendicular orientation of the SDIP PU edges, HM3 deblocking is performed. In case of parallel orientation, deblocking decisions and filters are adapted to the SDIP PU types on both sides of the edge to be deblocked. The proposed SDIP deblocking filtering supports parallel vertical and horizontal implementation of deblocking decision computations, as is the case for HM3 deblocking. In addition, the parallel vertical or horizontal filtering of HM3 is also supported. The BD-rate performance of the SDIP deblocking method is -0.1% for “All intra” coding conditions with respect to the HM3-SDIP anchor, while the encoding and decoding times are unchanged.

# CE6.b Test 5: SDIP Harmonization with MDIS

## Summary

This section presents modifications to MDIS and a harmonized solution for SDIP, which is an intra-coding method being studied in CE6.b. An updated decision mapping table is introduced and planar mode is included. On average the BD-rate gain improvement of the harmonized MDIS modifications in the SDIP environment is -0.1% for both intra-coding test conditions, while execution times are unchanged.

## Introduction

This contribution proposes a solution for harmonizing the mode-dependent intra smoothing (MDIS) modifications from related document [5] with the short-distance intra prediction technique (SDIP) [2], which is under study in CE6.b (Test 5: Harmonization of SDIP and MDIS). The modifications proposed in [5] are an updated decision mapping table between intra prediction modes and PU sizes, and inclusion of planar mode [6]. In the present contribution the updated table is introduced for square PUs and rectangular SDIP PUs.

## MDIS Modifications and SDIP

specifies the updated decision mapping table from [5], where the entry “1” means that intra smoothing filtering ([1 2 1] filter) is applied for the specific intra prediction mode and dimension parameter.

Table 8 Decision mapping table between intra prediction mode and partition dimension

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Dim\Mode** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** | **17** | **18** | **19** | **20** | **21** | **22** | **23** | **24** | **25** | **26** | **27** | **28** | **29** | **30** | **31** | **32** | **33** | **34** |
| **4** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **8** |  |  |  | 1 |  |  | 1 |  |  | 1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 1 |
| **16** |  |  |  | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |  |  | 1 | 1 | 1 | 1 | 1 | 1 |  |  | 1 | 1 | 1 | 1 |
| **32** |  |  |  | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

In the context of SDIP harmonization, this contribution proposes to compute the dimension parameter as follows for modes 0-33:

Min( Hsize, Vsize ) ,

where Hsize and Vsize are respectively the horizontal and vertical size of the PU.

In case of planar mode 34, it is proposed to compute the dimension parameter as follows:

Max( Hsize, Vsize ) .

## Results

represents the BD-rate performance results of modified MDIS, harmonized with SDIP, with respect to HM3-SDIP (rev866). On average the coding gain is -0.1% for the high efficiency and low complexity intra-coding test conditions. The encoding and decoding times are found to be comparable to HM3-SDIP. In , the RD performance is illustrated compared with the HM3-SDIP anchor and MDIS (HM3 version) turned off (QC\_MDIS=0). The average BD-rate gains are respectively -0.3% and -0.4% for the high efficiency and low complexity intra-coding conditions.

Table 9 BD-rate performance results, encoding and decoding time comparison with HM3-SDIP

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | All Intra HE | | | All Intra LC | | |
| Y | U | V | Y | U | V |
| Class A | -0.1 | -0.2 | -0.1 | -0.1 | -0.1 | -0.1 |
| Class B | -0.1 | -0.2 | -0.2 | -0.1 | -0.1 | -0.1 |
| Class C | -0.1 | -0.1 | -0.1 | -0.1 | -0.1 | -0.1 |
| Class D | 0.0 | 0.0 | -0.1 | 0.0 | 0.0 | 0.0 |
| Class E | -0.1 | -0.3 | -0.2 | 0.0 | -0.2 | -0.2 |
| **Overall** | **-0.1** | **-0.2** | **-0.1** | **-0.1** | **-0.1** | **-0.1** |
| Enc Time[%] | 100% | | | 101% | | |
| Dec Time[%] | 99% | | | 99% | | |

Table 10 BD-rate performance results for modified MDIS, harmonized with SDIP, compared with the HM3-SDIP anchor with MDIS turned off (QC\_MDIS=0)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | All Intra HE | | | All Intra LC | | |
| Y | U | V | Y | U | V |
| Class A | -0.5 | -0.4 | -0.6 | -0.7 | -0.3 | -0.6 |
| Class B | -0.3 | -0.4 | -0.4 | -0.4 | -0.3 | -0.3 |
| Class C | -0.1 | -0.2 | -0.2 | -0.1 | -0.2 | -0.1 |
| Class D | -0.1 | -0.2 | -0.2 | -0.1 | -0.1 | -0.1 |
| Class E | -0.4 | -0.7 | -0.5 | -0.6 | -0.6 | -0.5 |
| **Overall** | **-0.3** | **-0.4** | **-0.4** | **-0.4** | **-0.3** | **-0.3** |

## Conclusion of SDIP Harmonization with MDIS

This contribution proposed a harmonized solution between the MDIS modifications from [5] and the SDIP method. An updated decision mapping table between intra prediction mode and a partition dimension parameter is introduced. The dimension parameter is derived as the minimum of the horizontal and vertical sizes of the PU, except for planar mode where the maximum size is used. The average BD-rate gain compared to the HM3-SDIP anchor is -0.1% for both HE and LC intra-coding test conditions, and respectively -0.3% and -0.4% compared to the HM3-SDIP anchor when the unmodified MDIS is turned off. The encoding and decoding times are similar to the anchor times.

# Additional Harmonization: HE Residual Coding

## Introduction

SDIP non-rectangular PUs are transformed by the corresponding non-rectangular transforms of the same size. The following transforms are currently applied: 32x8, 8x32, 16x4, 4x16, 8x2, 2x8 and 16x1 and 1x16. Entropy coding is applied to the resulting coefficients. However, the HE entropy coding has not been modified to deal with non-rectangular transform coefficients. Instead, a mapping is applied in order to re-use the existing method for rectangular blocks. The mapping is defined from a non-square block to a square block with the same number of pixels. The regular entropy coding and decoding are applied to the square block. Then, prior to the inverse transform, the inverse mapping is applied for the SDIP blocks from the square shape to the non-square original shape.

This method re-uses existing entropy coding, but fails to exploit the underlying correlation of the coefficients in a non-square block because the mapping ‘breaks’ some of the neighboring relationship and correlation among coefficients. Therefore, the current method can be improved by harmonizing the current entropy coding method for non-square blocks. This part describes such harmonization of the CABAC entropy coding with the non-square blocks.

In this part, the mapping between non-square blocks to square blocks is not used also because it introduces intermediate steps that add complexity to the coder. Instead, we propose several modifications in the current ‘square-based’ entropy coding to deal with the non-square coefficient block. In this way, the neighbouring relationship of the coefficients is kept and the performance improved. Modifications are in the following parts: scans, coding of the last significant coefficient, significance map coding and coefficient level coding.

## Scans

Similar to the square blocks, three scans for the non-square blocks: horizontal, vertical and a type of zig-zag. The scan depends on the intra mode. A table maps the intra mode to the one of the three scans to apply. The blocks that use the three scans are the smaller ones: the ones with 64 coefficients or less. Larger blocks use only one scan. The zig-zag scan is actually the fixed diagonal scan proposed in JCTVC-C227 [8] starting from the long side of the block.

## Last Significant Coefficient Position

For non-square blocks, in the signaling of the last significant coefficient, the context model depends on the length of the side of the coordinate being coded. In other words, the horizontal coordinate (X component of the last coefficient) has a context depending on the width of the block (and the bin of X being coded). Equivalently, the vertical coordinate (Y component) has a context set assigned depending on the height of the block.

The context models are the same as the square case except for the following case. For blocks of a side of length 2, a CABAC context is used to indicate whether it is the first row (or column) or the second one that has the last significant coefficient. The blocks of a side of length 1 only need to encode one coordinate.

## Significance Map

For the significance map of non-square blocks, blocks with the same lengths (i.e., 2x8 and 8x2) share the same contexts. This reduces the total number of additional contexts. In order to enhance performance, the context derivation and sharing is not done directly, but after transposing the coordinates and values of one of the rectangular shapes. For example, the 8x2 is encoded directly, but the 2x8 is ‘transposed’ before coding: coordinates X and Y and the width and height are swapped prior to coding. This exploits the fact that the statistics of the 8x2 block are similar to the statistics of the 2x8 block transposed. Same principle applies to the other block sizes.

Blocks with 64 coefficients or less use contexts for the significance map depending on the position (like in HEVC the 4x4 and 8x8 blocks). The other blocks (32x8/8x32) use the neighbor based derivation equivalent to the larger blocks in HEVC.

## Coefficient Level Coding

For the coefficient level coding, current HEVC relies on the division of the block in sub-blocks of 4x4 coefficients and performs the coding within each sub-block sequentially. This type of processing is not possible when the block size cannot fit in a 4x4 sub-block (i.e., when at least one of the sides of the non-square is less than 4). To overcome this issue, the coding process described in JCTVC-E335 [9] is used, which relies on the processing of the block dividing it in sets of 16 consecutive coefficients in the scan order. For the sake of consistency, this method is applied to both, square and non-square blocks.

## Results

The HE residue harmonization for SDIP has been implemented on the top of HM3.0 + SDIP, which is the anchor for the results in . The BD-rate for AI-HE is -0.21%, while the computational time is reduced thanks to avoiding the intermediate mappings between square and non-square blocks.

Table 11 BD-rate performance results for harmonized HE entropy coding

|  |  |  |  |
| --- | --- | --- | --- |
| BD-rate | **All Intra HE** | | |
| Y | U | V |
| Class A | -0.15 | 0.35 | 0.21 |
| Class B | -0.29 | -0.08 | -0.19 |
| Class C | -0.26 | -0.16 | -0.16 |
| Class D | -0.22 | -0.05 | -0.06 |
| Class E | -0.09 | 0.14 | 0.16 |
| All | -0.21 | 0.03 | -0.03 |
| Enc Time[%] | 92% | | |
| Dec Time[%] | 98% | | |

# CE6.b Test 3: Interaction of SDIP and MDCS

In this test, the interaction of SDIP and MDCS is investigated. In the SDIP software, there are two Marcos that control the MDCS functionality. One is QC\_MDCS, which controls the coefficient scanning in square block; the other is HHMTU\_SDIP\_MDCS that controls the coefficient scanning in non-square block. The following tests in are conducted.

Table 12 Test descriptions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Tests | HHMTU\_SDIP | QC\_MDCS | HHMTU\_SDIP\_MDCS | Anchor |
| Test 3.1 | 0 | 0 | 0 | HM 3.0 |
| Test 3.2 | 0 | 1 | 0 | (this is HM3.0) |
| Test 3.3 | 1 | 0 | 0 | HM3.0 + SDIP |
| Test 3.4 | 1 | 1 | 0 | HM3.0 + SDIP |

Test 3.1 shows the performance of MDCS in HM3.0; Test 3.2 is the HM3.0 anchor; Test 3.3 shows the performance of MDCS on both square and non-square blocks when SDIP is enabled; Test 3.4 shows the performance of MDCS on non-square block when SDIP is enabled. to list the results, which can be found in attached excel sheets in detailed.

Table 13 Test 3.1- performance of MDCS in HM3.0 (positive means gain)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | All Intra HE | | | All Intra LC | | |
| Y | U | V | Y | U | V |
| Class A | 0.5 | 0.4 | 0.3 | 0.2 | 0.0 | -0.1 |
| Class B | 0.9 | 0.7 | 0.7 | 1.7 | 0.8 | 0.9 |
| Class C | 1.4 | 1.1 | 1.2 | 1.9 | 1.2 | 1.2 |
| Class D | 1.3 | 1.4 | 1.6 | 1.7 | 1.0 | 1.1 |
| Class E | 1.9 | 0.4 | 0.5 | 2.2 | 0.1 | 0.2 |
| **Overall** | **1.2** | **0.8** | **0.9** | **1.5** | **0.7** | **0.7** |
| Enc Time[%] | 100% | | | 101% | | |
| Dec Time[%] | 100% | | | 99% | | |

Table 14 Test 3.3- performance of MDCS in SDIP (positive means gain)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | All Intra HE | | | All Intra LC | | |
| Y | U | V | Y | U | V |
| Class A | 0.3 | 0.1 | 0.2 | 0.0 | 0.1 | 0.1 |
| Class B | 0.5 | 0.3 | 0.4 | 0.4 | 0.5 | 0.5 |
| Class C | 0.9 | 0.8 | 0.8 | 0.9 | 0.8 | 0.8 |
| Class D | 0.9 | 1.1 | 1.3 | 0.8 | 0.7 | 0.7 |
| Class E | 0.9 | -0.2 | -0.2 | 0.6 | 0.1 | 0.2 |
| **Overall** | **0.7** | **0.5** | **0.5** | **0.5** | **0.5** | **0.5** |
| Enc Time[%] | 100% | | | 98% | | |
| Dec Time[%] | 100% | | | 97% | | |

Table 15 Test 3.4- performance of MDCS applied only to non-square block in SDIP (positive means gain)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | All Intra HE | | | All Intra LC | | |
| Y | U | V | Y | U | V |
| Class A | 0.0 | 0.1 | 0.0 | 0.0 | 0.1 | 0.1 |
| Class B | 0.0 | 0.1 | 0.0 | 0.1 | 0.1 | 0.1 |
| Class C | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 |
| Class D | 0.2 | 0.2 | 0.2 | 0.1 | 0.1 | 0.1 |
| Class E | 0.0 | -0.1 | 0.0 | 0.0 | 0.1 | 0.1 |
| **Overall** | **0.1** | **0.1** | **0.1** | **0.1** | **0.1** | **0.1** |
| Enc Time[%] | 100% | | | 99% | | |
| Dec Time[%] | 99% | | | 98% | | |

# CE6.b Test 8: Interaction of SDIP and DC Prediction Filtering

## Summary

This section presents results on the DC prediction filtering tool and its interaction with SDIP in the context of CE6.b Test 8. Compared to the HM3.0 anchor, the BD-rate gain of the tool is -0.1% for both the intra-coding test conditions. Compared to SDIP, the gain is -0.1% for high efficiency intra coding and -0.2% for low complexity.

## Description

The following specifies the cases that are studied in CE6.b Test 8 for the DC prediction filtering tool [14]:

1. Case 1: HM3.0 anchor versus HM3.0 with DC prediction filtering disabled (MN\_DC\_PRED\_FILTER=0)
2. Case 2: HM3-SDIP (rev866) versus HM3-SDIP with DC prediction filtering disabled (MN\_DC\_PRED\_FILTER=0)

and enumerate the BD-rate results and execution times for cases 1 and 2, respectively. The encoding times are measured on a computing cluster with variable speeds. The decoding times are measured on a single CPU.

Table 16 Results for case 1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | All Intra HE | | | All Intra LC | | |
| Y | U | V | Y | U | V |
| Class A | 0.1 | 0.0 | 0.1 | 0.1 | 0.2 | 0.3 |
| Class B | 0.1 | 0.1 | 0.0 | 0.2 | 0.1 | 0.2 |
| Class C | 0.1 | 0.1 | 0.1 | 0.1 | 0.2 | 0.1 |
| Class D | 0.1 | 0.1 | 0.1 | 0.1 | 0.2 | 0.1 |
| Class E | 0.0 | 0.0 | 0.1 | 0.1 | 0.0 | 0.0 |
| **Overall** | **0.1** | **0.1** | **0.1** | **0.1** | **0.2** | **0.2** |
| Enc Time[%] | 102% | | | 99% | | |
| Dec Time[%] | 100% | | | 100% | | |

Table 17 Results for case 2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | All Intra HE | | | All Intra LC | | |
| Y | U | V | Y | U | V |
| Class A | 0.1 | 0.1 | 0.2 | 0.2 | 0.2 | 0.4 |
| Class B | 0.2 | 0.1 | 0.1 | 0.2 | 0.2 | 0.2 |
| Class C | 0.2 | 0.1 | 0.2 | 0.2 | 0.2 | 0.2 |
| Class D | 0.1 | 0.1 | 0.1 | 0.2 | 0.1 | 0.2 |
| Class E | 0.1 | 0.0 | 0.1 | 0.2 | 0.1 | 0.1 |
| **Overall** | **0.1** | **0.1** | **0.1** | **0.2** | **0.2** | **0.2** |
| Enc Time[%] | 107% | | | 102% | | |
| Dec Time[%] | 100% | | | 99% | | |

# References

1. K. McCann, B. Bross, S. Sekiguchi, W.-J. Han, “HM3: High Efficiency Video Coding (HEVC) Test Model 3 Encoder Description,” 5th JVC-VC Meeting, Geneva, Switzerland, Mar. 2011, Doc. JCTVC-E602
2. X. Cao et al., “CE6.b1 Report on Short Distance Intra Prediction Method,” 5th JCT-VC Meeting, Geneva, CH, March 2011, Doc. JCTVC-E278
3. J. An, Q. Huang, X. Guo, Y.-W. Huang, S. Lei, “CE12 Subtest1: Improved Deblocking Filter,” 5th JCT-VC Meeting, Geneva, CH, March 2011, Doc. JCTVC-E079
4. Norkin, K. Andersson, R. Sjoberg, “CE12.1: Ericsson Deblocking Filter,” 5th JCT-VC Meeting, Geneva, CH, March 2011, Doc. JCTVC-E276
5. G. Van der Auwera, X. Wang, M. Karczewicz, “CE6.e: Mode-Dependent Intra Smoothing Modifications,” 6th JCT-VC Meeting, Torino, Italy, July 2011, Doc. JCTVC-F126
6. S. Kanumuri, F. Bossen, “CE6.e/f: Planar mode experiments and results,” 5th JCT-VC Meeting, Geneva, Switzerland, Mar. 2011, Doc. JCTVC-E321
7. T. Wiegand, W.-J. Han, B. Bross, J.-R. Ohm, G. J. Sullivan, “WD3: Working Draft 3 of High-Efficiency Video Coding”, 5th JCT-VC Meeting, Geneva, CH, March 2011, Doc. JCTVC-E603\_d8
8. V. Sze, M. Budagavi, “Parallelization of HHI\_TRANSFORM\_CODING,” 3rd JCT-VC Meeting, Guangzhou, China, October 2010, Doc. JCTVC-C227
9. J. Sole, R. Joshi, M. Karczewicz, “Parallelization Unified scans for the significance map and coefficient level coding in high coding efficiency,” 5th JCT-VC Meeting, Geneva, Switzerland, Mar. 2011, Doc. JCTVC-E335
10. K. Suginoto, et al., “CE6.f: LUT-based adaptive filtering on intra prediction samples,” 5th JCT-VC Meeting, Geneva, CH, March 2011, Doc. JCTVC-E069

# Appendix: Working Draft Descriptions

## SDIP and MDIS Harmonization

The following changes to the WD3 [7] reflect the modifications to MDIS from this proposal.

**8.3.3.1.2 Filtering process of neighbouring samples**

Inputs to this process are:

– neighbouring samples p[ x, y ], with x, y = -1..2\*nS-1,

– a variable nShor specifying the horizontal prediction size,

– a variable nSver specifying the vertical prediction size,

Output of this process is:

– filtered samples pF[ x, y ], with x, y = -1..2\*nS-1.

**Table 8-6 – Specification of intraFilterType[ nS ][ IntraPredMode ] for various prediction unit sizes**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **IntraPredMode** | **intraFilterType**  **for nS = 4** | **intraFilterType**  **for nS = 8** | **intraFilterType**  **for nS = 16** | **intraFilterType**  **for nS = 32** |
| 0-2 | 0 | 0 | 0 | 0 |
| 3 | 0 | 1 | 1 | 1 |
| 4, 5 | 0 | 0 | 1 | 1 |
| 6 | 0 | 1 | 1 | 1 |
| 7, 8 | 0 | 0 | 1 | 1 |
| 9 | 0 | 1 | 1 | 1 |
| 10-20 | 0 | 0 | 1 | 1 |
| 21, 22 | 0 | 0 | 0 | 1 |
| 23-28 | 0 | 0 | 1 | 1 |
| 29, 30 | 0 | 0 | 0 | 1 |
| 31-33 | 0 | 0 | 1 | 1 |
| 34 | 0 | 1 | 1 | 1 |
| 35 | n/a | n/a | n/a | n/a |

Variable nS is derived as follows:

– When IntraPredMode is equal to 34, then nS = max(nShor, nSver), otherwise nS = min(nShor, nSver)

Filtered sample array pF[ x, y ] with x = -1..nS\*2-1 and y = -1..nS\*2-1  are derived as follows:

– When intraFilterType[ nS ][ IntraPredMode ] is equal to 1, the following applies:

pF[ -1, nS\*2-1 ] = p[ -1, nS\*2-1 ] (7‑1)

pF[ nS\*2-1, -1 ] = p[ nS\*2-1, -1 ] (7‑2)

pF[ -1, y ] = ( p[ -1, y+1 ] + 2\*p[ -1, y ] + p[ -1, y-1 ] + 2 ) >> 2 for y = nS\*2-2..0 (7‑3)

pF[ -1, -1] = ( p[ -1, 0 ] + 2\*p[ -1, -1] + p[ 0, -1 ] + 2) >> 2 (7‑4)

pF[ x, -1 ] = ( p[ x-1, -1 ] + 2\*p[ x, -1 ] + p[ x+1, -1 ] + 2 ) >> 2 for x = 0..nS\*2-2

## SDIP and Deblocking Harmonization

The following changes to the WD3 [7] reflect the modifications to deblocking from this proposal.

**8.6.1.2 Derivation process of prediction unit boundary**

…

Depending on PartMode, the following applies:

– If PartMode is equal to PART\_2NxN or PART\_NxN, horEdgeFlags[ k ][ 1 << ( log2CUSize – 1 ) ] is set equal to FilterInternalEdgesFlag for k = 0.. ( 1 << log2CUSize ) – 1.

– If PartMode is equal to PART\_Nx2N or PART\_NxN, verEdgeFlags[ 1 << ( log2CUSize – 1 ) ][ k ] is set equal to FilterInternalEdgesFlag for k = 0.. ( 1 << log2CUSize ) – 1.

* If PartMode is equal to PART\_2NxhN, horEdgeFlags[ k ][ 1 << ( log2CUSize – 2 ) ], horEdgeFlags[ k ][ (1 << ( log2CUSize – 2 )) \* 2 ], horEdgeFlags[ k ][ (1 << ( log2CUSize – 2 )) \* 3 ] are set equal to FilterInternalEdgesFlag for k = 0.. ( 1 << log2CUSize ) – 1.
* If PartMode is equal to PART\_hNx2N, verEdgeFlags [ 1 << ( log2CUSize – 2 ) ][ k ], verEdgeFlags[ (1 << ( log2CUSize – 2 )) \* 2 ] [ k ], verEdgeFlags[ (1 << ( log2CUSize – 2 )) \* 3 ] [ k ] are set equal to FilterInternalEdgesFlag for k = 0.. ( 1 << log2CUSize ) – 1.

**8.6.1.3 Derivation of boundary filtering strength**

…

Let ( xEk, yEj ) with k = 0..nE-1 and j = 0..nE-1 specify a set of edge sample locations where nE is set equal to ( ( 1 << log2CUSize ) >> 1 ), xE0 = 0, yE0 = 0, xEk+1 = xEk + 2 and yEj+1 = yEj + 2.

For ( xEk, yEj ) with k = 0..nE-1 and j = 0..nE-1, the following applies.

* If horEdgeFlags[ xEk ][ yEj ] is equal to 1,
* Set sample p0 = recPicture[ xC + xEk ][ yC + yEj – 1 ] and q0 = recPicture[ xC + xEk ][ yC + yEj ].
* The variable filterDir is set equal to 1.
* Otherwise, if verEdgeFlags[ xEk ][ yEj ] is equal to 1,
* Set sample p0 = recPicture[ xC + xEk – 1 ][ yC + yEj ] and q0 = recPicture[ xC + xEk ][ yC + yEj ].
* The variable filterDir is set equal to 0.
* Depending on the value of filterDir, the variable bS[ filterDir ][ xEk ][ yEj ] is derived as follows.
* If the block edge is also a coding unit edge:
* If the sample p0 or q0 is in a coding unit coded with intra prediction mode and PartMode is not PART\_2NxhN or PART\_hNx2N, then the variable bS[ filterDir ][ xEk ][ yEj ] is set equal to 4
* If the sample p0 or q0 is in a coding unit coded with intra prediction mode and PartMode is PART\_2NxhN or PART\_hNx2N, then variables bSp[ filterDir ] and bSq[ filterDir ] are set equal to 4, and they are further determined as follows:
  + - If p0 is in a coding unit (size log2CUSizep) with intra prediction mode and PartMode is PART\_2NxhN or PART\_hNx2N:
      * If PartMode is PART\_2NxhN and filterDir is equal to 0, or PartMode is PART\_hNx2N and filterDir is equal to 1, then the variable bSp[ filterDir ] stays 4
      * If PartMode is PART\_2NxhN and filterDir is equal to 1, or PartMode is PART\_hNx2N and filterDir is equal to 0:
        + If log2CUSizep is equal to 5, then the variable bSp[ filterDir ] is set equal to 5
        + If log2CUSizep is equal to 4, then the variable bSp[ filterDir ] is set equal to 6
        + If log2CUSizep is equal to 3, then the variable bSp[ filterDir ] is set equal to 7
    - If q0 is in a coding unit (size log2CUSizeq) with intra prediction mode and PartMode is PART\_2NxhN or PART\_hNx2N:
      * If PartMode is PART\_2NxhN and filterDir is equal to 0, or PartMode is PART\_hNx2N and filterDir is equal to 1, then the variable bSq[ filterDir ] stays 4
      * If PartMode is PART\_2NxhN and filterDir is equal to 1, or PartMode is PART\_hNx2N and filterDir is equal to 0:
        + If log2CUSizep is equal to 5, then the variable bSq[ filterDir ] is set equal to 5
        + If log2CUSizep is equal to 4, then the variable bSq[ filterDir ] is set equal to 6
        + If log2CUSizep is equal to 3, then the variable bSq[ filterDir ] is set equal to 7
    - The variable bS[ filterDir ][ xEk ][ yEj ] is set equal to Max( bSp[ filterDir ], bSq[ filterDir ] )
* Otherwise, if block edge is not a coding unit edge, and if the following condition is true:
* If the sample p0 or q0 is in a coding unit coded with intra prediction mode and PartMode is not PART\_2NxhN or PART\_hNx2N, then the variable bS[ filterDir ][ xEk ][ yEj ] is set equal to 3
* If the sample p0 or q0 is in a coding unit coded (size log2CUSize) with intra prediction mode and PartMode is PART\_2NxhN or PART\_hNx2N, then the variable bS[ filterDir ][ xEk ][ yEj ] is determined as follows:
  + - If log2CUSize is equal to 5, then bS[ filterDir ][ xEk ][ yEj ] is set equal to 5
    - If log2CUSize is equal to 4, then bS[ filterDir ][ xEk ][ yEj ] is set equal to 6
    - If log2CUSize is equal to 3, then bS[ filterDir ][ xEk ][ yEj ] is set equal to 7
* Otherwise, …

Mappings of bS[ filterDir ][ xEk ][ yEj ] values to filters:

|  |  |  |
| --- | --- | --- |
| bS | Filter Type | Decision Conditions |
| 0…4 | HM3 [] | HM3 [] |
| 5 | Strong/Weak4 () | Deblock edge on/off (Table 4)  Strong or weak filter (Table 4) |
| 6 | Weak2 () | Deblock edge on/off (Table 4) |
| 7 | Weak1 () | Deblock edge on/off (Table 4) |

## SDIP and HE Residual Coding Harmonization

###### Derivation process of ctxIdxInc for the syntax elements last\_significant\_coeff\_x and last\_significant\_coeff\_y (9.3.3.1.1.5 in WD3)

Inputs to this process are the binIdx, the color component index cIdx and the transform block size log2TrafoSize.

Output of this process is ctxIdxInc.

Specifcation of lastCtx depending on binIdx

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **binIdx** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** |
| **lastCtx** | 0 | 1 | 2 | 3 | 3 | 4 | 4 | 5 | 5 | 5 | 5 | 6 | 6 | 6 | 6 | 7 |
| **binIdx** | **16** | **17** | **18** | **19** | **20** | **21** | **22** | **23** | **24** | **25** | **26** | **27** | **28** | **29** | **30** |  |
| **lastCtx** | 7 | 7 | 7 | 8 | 8 | 8 | 8 | 9 | 9 | 9 | 9 | 10 | 10 | 10 | 10 |  |

The variable lastCtx is dervied from binIdx as given by Table 9‑39. For the derivation of ctxIdxInc, the following applies. For non-square blocks, log2TrafoSize is equal to the length of the side being coded.

– If log2TrafoSize is equal to 1, ctxIdxInc is dervied as follows.

ctxIdxInc = lastCtx

– log2TrafoSize is equal to 2, ctxIdxInc is dervied as follows.

ctxIdxInc = lastCtx + 1 (9‑7)

– Otherwise if log2TrafoSize is equal to 3, ctxIdxInc is dervied as follows.

ctxIdxInc = lastCtx + 4 (9‑8)

– Otherwise if log2TrafoSize is equal to 4, ctxIdxInc is dervied as follows.

ctxIdxInc = lastCtx + 9 (9‑9)

– Otherwise (log2TrafoSize is greater than 4), ctxIdxInc is dervied as follows.

ctxIdxInc = lastCtx + 16 (9‑10)

When cIdx is greater than 0, ctxIdxInc is modified as follows.

ctxIdxInc = ctxIdxInc + 27 (9‑11)

###### Derivation process of ctxIdxInc for the syntax element significant\_coeff\_flag of non-square blocks (for 9.3.3.1.1.5 in WD3)

Inputs to this process are the color component index cIdx, the current coefficient scan position ( xC , yC ) and the transform size blockWidth and blockHeight.

Output of this process is ctxIdxInc.

The variable sigCtx depends on the current position ( xC, yC ), the transform block size and previsously decoded bins of the syntax element significant\_coeff\_flag. blockType is derived as follows:

* If blockWidth = 1 or blockHeight =1, then blockType = 0.
* If blockWidth = 2 or blockHeight =2, then blockType = 1.
* If blockWidth = 4 or blockHeight =4, then blockType = 2.
* Otherwise, blockType = 3.

If trWidth < trHeight, ( xC, yC ) = swap( xC, yC )

For the derivation of sigCtx, the following applies.

* If blockType is equal 0, sigCtx is derived as follows.  
   sigCtx = xC
* Otherwise, if blockType is equal 1, sigCtx is derived as follows.

sigCtx = ( xC << 3 ) + yC

* Otherwise, if blockType is equal 2, sigCtx is derived as follows.
  + If xC is less than 2 and yC is less than 2, sigCtx is derived as follows.

sigCtx = ( yC << 1 ) + xC

* + Otherwise, if xC is smaller than 8,

sigCtx = 3 + ( xC >> 1 ) + (( yC >> 1) << 2 )

* + Otherwise, sigCtx = 9 + ( xC >> 2 ) + (( yC >> 1) << 1 )
* Otherwise if xC is less than 2 and yC is less than 2, sigCtx is derived as follows.

sigCtx = ( yC << 1 ) + xC

* Otherwise ( xC, yC ) = swap( xC, yC ), if yC is equal to 0, sigCtx is derived as follows.

sigCtx = 4 + significant\_coeff\_flag[ xC-1 ][ yC ] + significant\_coeff\_flag[ xC-2 ][ yC ]

* Otherwise if xC is equal to 0, sigCtx is derived as follows.

sigCtx = 4 + significant\_coeff\_flag[ xC ][ yC-1 ] + significant\_coeff\_flag[ xC ][ yC-2 ]

* Otherwise (xC is greater than 1 and yC is greater than 1) , the following applies.
* The variable sigCtx is initialized using previously decoded bins of the syntax element significant\_coeff\_flag as follows.

sigCtx = significant\_coeff\_flag[ xC-1 ][ yC ] + significant\_coeff\_flag[ xC ][ yC-1 ] +   
 significant\_coeff\_flag[ xC-1 ][ yC-1 ]

* When xC is greater than 1, the following applies.

sigCtx = sigCtx + significant\_coeff\_flag[ xC-2 ][ yC ]

* When yC is greater than 1, the following applies.

sigCtx = sigCtx + significant\_coeff\_flag[ xC ][ yC-2 ]

* The variable sigCtx is dervied as follows.

sigCtx = 10 + min( 4, sigCtx )

The context index increment ctxIdxInc is derived using the non-square index blockType

ctxIdxInc = ( blockType \* 16 ) + sigCtx

###### Derivation process of ctxIdxInc for the syntax element coeff\_abs\_level\_greater1\_flag and coeff\_abs\_level\_greater2\_flag (9.3.3.1.1.5 in WD3)

Inputs to this processes are the color component index cIdx, the transform block size log2TrafoSize, the 16 coefficient subset index i and the current coefficient scan index n within the current subset.

Output of this process is ctxIdxInc.

The variable ctxSet specifies the current context set and for its derivation the following applies.

* If n is equal to 0 or all previous syntex elements coeff\_abs\_level\_greater1\_flag[ pos ] with pos less than n are derived to be equal to 0 instead of being explicitly parsed, the following applies.
* If the current subset index i is 0, then ctxSet = 0
* Otherwise, ctxSet = 3
* If the subset i is the first one in 9.3.3.1.1.5 , the variable greater2Ctx that has been derived during the last invocation of subclause 9.3.3.1.1.6 for the syntax element coeff\_abs\_level\_greater2\_flag for the subset i + 1 is set to greater2Ctx>>1.
* The variable lastGreater2Ctx is set equal to greater2Ctx and ctxSet is modified as follows
  + If greater2Ctx > 0, ctxSet++
  + If greater2Ctx > 3, ctxSet++

|  |  |
| --- | --- |
| residual\_coding\_cabac( x0, y0, log2TrafoSize, trafoDepth, scanIdx, cIdx ) { | Descriptor |
| **last\_significant\_coeff\_x** | ae(v) |
| **last\_significant\_coeff\_y** | ae(v) |
| n = 0 |  |
| xC = ScanOrder[ log2TrafoSize − 2 ][ scanIdx ][ n ][ 0 ] |  |
| yC = ScanOrder[ log2TrafoSize − 2 ][ scanIdx  ][ n ][ 1 ] |  |
| while( ( xC != last\_significant\_coeff\_x ) || ( yC != last\_significant\_coeff\_y ) ) { |  |
| **significant\_coeff\_flag[** xC **][** yC **]** | ae(v) |
| n++ |  |
| xC = ScanOrder[ log2TrafoSize − 2 ][ scanIdx ][ n ][ 0 ] |  |
| yC = ScanOrder[ log2TrafoSize − 2 ][ scanIdx  ][ n ][ 1 ] |  |
| } |  |
| numLastSubset = n >> 4 |  |
| for( i = numLastSubset; i >= 0; i−− ) { |  |
| offset = i << 4 |  |
| for( n = 15; n >= 0; n−− ) { |  |
| xC = ScanOrder[ log2TrafoSize − 2 ][ scanIdx ][ n + offset ][ 0 ] |  |
| yC = ScanOrder[ log2TrafoSize − 2 ][ scanIdx  ][ n + offset ][ 1 ] |  |
| if( significant\_coeff\_flag[ xC ][ yC ] ) |  |
| **coeff\_abs\_level\_greater1\_flag[** n **]** | ae(v) |
| } |  |
| for( n = 15; n >= 0; n−− ) { |  |
| if( coeff\_abs\_level\_greater1\_flag[ n ] ) { |  |
| **coeff\_abs\_level\_greater2\_flag[** n **]** | ae(v) |
| if( coeff\_abs\_level\_greater2\_flag[ n ] ) |  |
| **coeff\_abs\_level\_minus3[** n **]** | ae(v) |
| } |  |
| } |  |
| for( n = 15; n >= 0; n−−) { |  |
| xC = ScanOrder[ log2TrafoSize − 2 ][ scanIdx ][ n + offset ][ 0 ] |  |
| yC = ScanOrder[ log2TrafoSize − 2 ][ scanIdx  ][ n + offset ][ 1 ] |  |
| if( significant\_coeff\_flag[ xC ][ yC ] ) { |  |
| **coeff\_sign\_flag[** n **]** | ae(v) |
| transCoeffLevel[ x0 ][ y0 ][ trafoDepth ][ cIdx ][ n + offset ] =   ( coeff\_abs\_level\_minus3[ n ] + 3 ) \* ( 1 − 2 \* coeff\_sign\_flag[ n ] ) |  |
| } else |  |
| transCoeffLevel[ x0 ][ y0 ][ trafoDepth ][ cIdx ][ n + offset ] = 0 |  |
| } |  |
| } |  |
| } |  |

Table 18 Residual coding syntax table

### Transform coefficient semantics

For non-square blocks, set the transform size blockWidth and blockHeight. Then, blockType is derived as follows:

* If blockWidth = 1 or blockHeight =1, then blockType = 0.
* If blockWidth = 2 or blockHeight =2, then blockType = 1.
* If blockWidth = 4 or blockHeight =4, then blockType = 2.
* Otherwise, blockType = 3.

Then, the specify the block scan type given the intra mode, Table 7.15 is used. If blockType is less than 3, then the type is derived as ScanType[ 2 ][ IntraPredMode ]. Otherwise, the type is dervied as ScanType[ 4 ][ IntraPredMode ].

###### Scanning order array initialisation process for non-square blocks (for 6.7 in WD3)

blkWidth and blkHeight are input to this process. Output of this process is the array ScanOrderNonSq[ scanIdx ][ pos ][ comp ].

i = blkWidth \* blkHeight

if( blkWidth > blkHeight ) {

j = 0   
 z = 0  
 while( z < j ) {  
 x = z

y = 0  
 while( x < blkWidth) {  
 y++

x--  
 }

while( x >= 0 && y < blkHeight) {  
 ScanOrderNonSq[ 0 ][ i ][ 0 ] = x  
 ScanOrderNonSq[ 0 ][ i ][ 1 ] = y

z++

y++

x--  
 }  
 }

}

Else{

j = 0   
 z = 0  
 while( z < j ) {  
 y = z

x = 0  
 while( y < blkHeight) {  
 x++

y--  
 }

while( y >= 0 && x < blkWidth ) {  
 ScanOrderNonSq[ 0 ][ i ][ 0 ] = x  
 ScanOrderNonSq[ 0 ][ i ][ 1 ] = y

z++

x++

y--

}

}

}

# Patent rights declaration(s)

**Qualcomm 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).**