|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11**  6th Meeting: Torino, IT, 14-22 July, 2011 | Document: JCTVC-F566  WG11 Number: m21000 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **CE6.c: Differential Coding of Intra Modes** | | |
| *Status:* | Input Document to JCT-VC | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Ehsan Maani  1730 N 1st Street San Jose, CA 95112  Tomoyuki Yamamoto  1-9-2 Nakase, Mihama-ku, Chiba-shi, Chiba 261-8520 JAPAN  Akiyuki Tanizawa, Taichiro Shiodera  1, Komukai-Toshiba-chou, Saiwai-ku, Kawasaki, Japan  Virginie Drugeon  Panasonic R&D Center Germany  Monzastr. 4c, 63225 Langen, Germany | Tel: Email:  Tel: Email  Tel: Email  Tel: Email: | (408) 352-4710 [ehsan.maani@am.sony.com](mailto:philippe.bordes@technicolor.com)  +81-43-299-8526 [yamamoto.tomoyuki@sharp.co.jp](mailto:ikai.tomohiro@sharp.co.jp)  +81-44-549-2152  [akiyuki.tanizawa@toshiba.co.jp](mailto:akiyuki.tanizawa@toshiba.co.jp)  +49 6103 766-240 [virginie.drugeon@eu.panasonic.com](mailto:virginie.drugeon@eu.panasonic.com) | |
| *Source:* | Sony Corp., SHARP, Toshiba Corp., Panasonic | | |

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

# Abstract

This document presents the description and results of Differential Coding of Intra Modes (DCIM) of JCTVC-B109. On average, using this technique, 0.8% and 0.7% gain is achieved compared to HM 3.0 anchors for High Efficiency (HE) and Low Complexity (LC) settings, respectively.

# Algorithm description

DCIM uses neighborhood edge estimation to predict the Intra prediction direction and differentially encodes the selected direction with respect to the predicted direction (i.e., given by the edge). This enables a higher accuracy in the Intra prediction directions (modes) without substantially increasing the mode signaling overhead. An additional flag bit is transmitted per Prediction Unit (PU) to signal to the decoder whether DCIM is used. The decoder needs to perform edge detection for the PUs for which the DCIM flag is on. Details of various parts of the proposed technique are described below.

## Edge Derivation

In order to find the dominant direction in the neighborhood of the block to be predicted, edge detection is used both at the encoder and decoder. A neighborhood of 2-pixels wide is considered in the reconstructed area around the block to be Intra predicted. Then, the following 2-tap filters are applies in the considered region to find the local image gradients.



,

Figure 1 illustrated the block to be predicted and the derived local image gradients in the reconstructed neighborhood. Next, to filter out noise and find a dominant direction the following two integer values are computed

,

where,  are the two components of the image gradient resulted from filtering with  and , and  represents the reconstructed neighborhood considered for edge derivation. The ratio  and the



Figure 1. An example of local image gradients in the reconstructed neighborhood.

signs of  and  are used with a look-up table of size 32 (integers) to find the index of the direction to be used for Intra prediction. Since  and  are both integers an 8-bit precision is considered for their ratio, i.e., . The index of the dominant direction is a value between 0-127, each corresponding to a direction of a line drawn between an integer location on the boundary of a 32x32 rectangle and its center.

## DCIM directions

In addition to the edge-derived direction and UI directions, a number of other DCIM directions, referred to as DCIM sub-directions (typically 2 directions on each side), are also considered around the detected edge. These directions consist of approximately equally distanced directions that are closest to the derived edge. Figure 2 demonstrated this concept.



Figure 2. Additional DCIM directions considered around the dominant edge direction.

## Interpolation

Conducting directional prediction using an arbitrary direction, often requires non-integer pixel values in the reconstructed boundary of the block. The values of these fractional-pel locations are therefore obtained using interpolation. In the current HM 3.0 a linear (2-pel) interpolation method is utilized. However, the DCIM directions perform better when used with a more accurate interpolation, such as 32 4-tap interpolation filters (shown in ). The main DCIM direction (edge-derived) always uses these 4-tap filters. However, the rest of the directions use both 4-tap and the default linear interpolation based on their mode number and a fixed look-up table. These interpolation filters are only used in certain experiments (when mentioned) along with DCIM.

Table 1. 4-tap interpolation filters (values scaled by 256).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Fractional-pel location | Filter coefficient | | | |
| 0/32 | 0 | 256 | 0 | 0 |
| 1/32 | -3 | 252 | 8 | -1 |
| 2/32 | -5 | 247 | 17 | -3 |
| 3/32 | -7 | 242 | 25 | -4 |
| 4/32 | -9 | 236 | 34 | -5 |
| 5/32 | -10 | 230 | 43 | -7 |
| 6/32 | -12 | 224 | 52 | -8 |
| 7/32 | -13 | 217 | 61 | -9 |
| 8/32 | -14 | 210 | 70 | -10 |
| 9/32 | -15 | 203 | 79 | -11 |
| 10/32 | -16 | 195 | 89 | -12 |
| 11/32 | -16 | 187 | 98 | -13 |
| 12/32 | -16 | 179 | 107 | -14 |
| 13/32 | -16 | 170 | 116 | -14 |
| 14/32 | -17 | 162 | 126 | -15 |
| 15/32 | -16 | 153 | 135 | -16 |
| 16/32 | -16 | 144 | 144 | -16 |
| *i*/32, *i* | Mirror of filter for (32-*i*)/32 | | | |

## Prediction

Once the index of the dominant direction is known the process of Intra prediction for the block is conducted similarly to that of Unified Intra (UI) directions. The only difference is that there are 32 possible UI directions (slopes) while the number of possible edge-derived directions is 128. Consequentially, the prediction for both edge-derived directions and UI directions can be performed using one function.

## Bi-prediction

* + 1. **Bidirectional intra prediction**

Bidirectional Intra Prediction (BIP) combines two unidirectional intra predictions (UIP) results by a weighted sum according to the distance between the predicted pixels and the reference pixel(s) used for prediction. The sample in the bidirectional prediction at pixel position (n), predBi[n], is derived from the following equation:

predBi[n] = ( w[n] \* predL0[n] + ((1<<w\_shift) - w[n]) \* predL1[n] + (1<<(w\_shift-1))) >> (w\_shift)

(1)

where predL0[n] and predL1[n] are the samples of the two UIP modes, and w[n] shows the weighting list for predL0[n] according to the sample position (n). w\_shift shows the shift value of the weighting list for the rounding operation and is set to 10. The weighting list w[n] is pre-determined according to the difference of distance from corresponding reference pixels of two prediction directions. The weighting lists are identical to one of JCTVC-D108 except for PU32x32. In order to reduce the number of multiplication in the equation (1), w[n] is set to (1<<(w\_shift-1)) for PU32x32.

* + 1. **Restriction of Unidirectional intra prediction used for Bidirectional intra prediction**

In JCTVC-D108, two kinds of UIP modes, intraPredModeL0 and intraPredModeL1, are selected from several UIP mode candidates for current PU. UIP modes except for 0, 1, 2, 3, 6 and 9 use bilinear filter to calculate the prediction value of fractional position. It may cause the large computational complexity to calculate the prediction value of BIP modes. Therefore, the UIP modes used for BIP modes are restricted by only 6 modes (0, 1, 2, 3, 6, 9), which mean to DC mode or just copy modes which copy the reference pixels. This restriction can reduce the computational complexity for worst case.

* + 1. **Reduction of Bidirectional intra prediction modes**

In order to reduce the computational complexity further, the number of BIP modes is reduced by 2 from 8 modes compared with JCTVC-D108. BIP modes are added when PU size is from 4x4 to 32x32. When the PU size is 64x64, the prediction mode set is same as set specified in original UIP (unidirectional only).

## Signaling of DCIM directions

For PU sizes of 4x4, 8x8, 16x16 and 32x32, DCIM modes can be signaled in addition to UI modes in HM. DCIM modes are correctively signaled as one of probable modes. Specifically, probable modes consist of (1) DCIM modes, (2) MPM, and (3) DC mode. As like HM-3.0, prev\_intra\_luma\_pred\_flag is first transmitted to signal whether one of probable modes is used. When prev\_intra\_luma\_pred\_flag is true, dcim\_flag follows, which indicates whether one of DCIM modes is used. When DCIM is used, information to select one of DCIM modes is additionally signaled. If dcim\_flag is false, the additional information is signaled to determine which probable mode (MPM or DC mode) is used. In case prev\_intra\_luma\_pred\_flag is false, rem\_intra\_luma\_pred\_mode is signaled to select one of remaining modes. The bit assignment to each intra mode is summarized in the following tables.

Table 2. Bit assignment for intra modes for Luma   
(Index k represents directions used for DCIM. K=2 for 16x16 and 32x32 PU and K=0 for other PU sizes.)

|  |  |  |  |
| --- | --- | --- | --- |
| **Intra mode** | **Bit assignment** | | |
|  | **prev\_intra\_luma \_pred\_flag** | **dcim\_flag** | **rem\_bits** |
| DCIM main direction (k=0) | 1 | 1 | 1 \*1 |
| DCIM sub directions (k=±1, ±2, … , ±K) | 1 | 1 | 0 + unary(abs(k)-1) + sign(k) |
| UI (MPM) | 1 | 0 | 0 \*2 |
| UI (DC/Planar) | 1 | 0 | 1 |
| UI (remaining modes) | 0 | - | rem\_intra\_luma\_pred\_modes |

\*1: This flag is omitted in PU 4x4 and 8x8 in which DCIM sub directions are not used.  
 \*2: This flag is omitted if MPM is either DC or Planar mode.

## Harmonization with other coding tools

Since adding DCIM modes changes distribution and availability of intra modes, some coding tools that use intra modes are modified to achieve bettern performance. For Mode Dependent Coefficient Scan (MDCS), when PU is coded in DCIM mode, scan direction is selcted based on the predicted UI mode that is derived based on the top and left non-DCIM PUs. For intra chroma mode based on luma UI mode, DC mode is derived when the corresponding luma component is coded with DCIM mode. For MDIS, DCT is always used in PU with DCIM mode. For mode dependent DST, the UI mode which is nearest to the derived edge is used to select a transform.

# Results

The proposed algorithm was implemented into HM-3.0 reference software. Based on the description in CE6.c, following configurations of DCIM were evaluated.

- Test 1. DCIM default (DCIM sub directions, BIP, and 4-tap filter are enabled)

- Test 2. DCIM no sub directions (DCIM sub directions is disabled relative to Test 1)

- Test 3. DCIM no BIP (BIP is disabled relative to Test 1)

- Test 4. DCIM no 4-tap filter (4-tap filter is disabled relative to Test 1)

Tables below summarize the perfomance of each configuration compared to anchor (HM-3.0).

**Table 3. Test 1. DCIM default compared with HM-3.0**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | All Intra HE | | | All Intra LC | | |
| Y | U | V | Y | U | V |
| Class A | -0.2 | 0.0 | 0.1 | -0.2 | 0.0 | 0.0 |
| Class B | -0.7 | -1.0 | -1.1 | -0.5 | -0.4 | -0.5 |
| Class C | -1.2 | -1.1 | -1.1 | -1.4 | -0.7 | -0.8 |
| Class D | -0.7 | -0.4 | -0.4 | -0.6 | -0.1 | -0.2 |
| Class E | -1.0 | -1.3 | -0.9 | -0.8 | -1.0 | -0.7 |
| **Overall** | **-0.8** | **-0.7** | **-0.7** | **-0.7** | **-0.4** | **-0.5** |
| Enc Time[%] | 113% | | | 124% | | |
| Dec Time[%] | 103% | | | 105% | | |

**Table 4. Test 2. DCIM no sub directions compared with HM-3.0**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | All Intra HE | | | All Intra LC | | |
| Y | U | V | Y | U | V |
| Class A | -0.1 | 0.3 | 0.3 | 0.0 | 0.2 | 0.2 |
| Class B | -0.3 | -0.3 | -0.2 | -0.1 | 0.0 | 0.0 |
| Class C | -0.8 | -0.5 | -0.5 | -1.1 | -0.3 | -0.4 |
| Class D | -0.6 | -0.2 | -0.2 | -0.6 | -0.1 | -0.1 |
| Class E | -0.3 | -0.1 | 0.1 | -0.1 | -0.3 | -0.1 |
| **Overall** | **-0.4** | **-0.2** | **-0.1** | **-0.4** | **-0.1** | **-0.1** |
| Enc Time[%] | 106% | | | 114% | | |
| Dec Time[%] | 103% | | | 106% | | |

**Table 5. Test 3. DCIM no BIP compared with HM-3.0**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | All Intra HE | | | All Intra LC | | |
| Y | U | V | Y | U | V |
| Class A | -0.2 | 0.1 | 0.0 | -0.1 | 0.1 | 0.1 |
| Class B | -0.7 | -1.0 | -1.2 | -0.4 | -0.3 | -0.5 |
| Class C | -1.1 | -1.1 | -1.0 | -1.2 | -0.5 | -0.6 |
| Class D | -0.6 | -0.4 | -0.4 | -0.4 | -0.1 | -0.1 |
| Class E | -1.0 | -1.2 | -0.8 | -0.6 | -0.9 | -0.7 |
| **Overall** | **-0.7** | **-0.7** | **-0.7** | **-0.5** | **-0.3** | **-0.3** |
| Enc Time[%] | 110% | | | 119% | | |
| Dec Time[%] | 103% | | | 104% | | |

**Table 6. Test 4. DCIM no 4-tap compared with HM-3.0**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | All Intra HE | | | All Intra LC | | |
| Y | U | V | Y | U | V |
| Class A | -0.2 | 0.2 | 0.1 | -0.2 | 0.0 | 0.0 |
| Class B | -0.3 | -0.6 | -0.7 | 0.1 | 0.1 | 0.0 |
| Class C | -0.6 | -0.5 | -0.4 | -0.5 | -0.1 | -0.1 |
| Class D | -0.3 | 0.1 | 0.1 | 0.0 | 0.3 | 0.2 |
| Class E | -0.4 | -0.8 | -0.3 | 0.1 | -0.4 | -0.1 |
| **Overall** | **-0.4** | **-0.3** | **-0.3** | **-0.1** | **0.0** | **0.0** |
| Enc Time[%] | 111% | | | 121% | | |
| Dec Time[%] | 103% | | | 105% | | |

# References

1. E. Maani, W. Liu, “Differential Coding of Intra Modes (DCIM)”, Doc. JCTVC-B109, Geneva, Switzerland, Jul 2010

# Patent rights declaration(s)

**Sony Corporation, SHARP Corporation, and Toshiba Corporation may have current or pending patent rights relating to the technology described in this contribution and, conditioned on reciprocity, are 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).**

**Panasonic Corporation 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).**

**Appendix A: Proposed syntax and semantics for DCIM**

The modified portions of JCTVC-E603 are written in blue in the present appendix.

1. **Changes in "3. Definitions"**

**differential coding of intra modes (DCIM)**: An *intra prediction* method that consists in deriving an edge in the neighboring signal of the same decoded *slice* and signal the direction of the *intra prediction* as a difference with the derived edge direction.

1. **Changes in PPS syntax**

|  |  |
| --- | --- |
| pic\_parameter\_set\_rbsp( ) { | **Descriptor** |
| **pic\_parameter\_set\_id** | ue(v) |
| **seq\_parameter\_set\_id** | ue(v) |
| **entropy\_coding\_mode\_flag** | u(1) |
| **num\_temporal\_layer\_switching\_point\_flags** | ue(v) |
| for( i = 0; i < num\_temporal\_layer\_switching\_point\_flags; i++ ) |  |
| **temporal\_layer\_switching\_point\_flag**[ i ] | u(1) |
| **num\_ref\_idx\_l0\_default\_active\_minus1** | ue(v) |
| **num\_ref\_idx\_l1\_default\_active\_minus1** | ue(v) |
| **pic\_init\_qp\_minus26** **/**\* relative to 26 \*/ | se(v) |
| **constrained\_intra\_pred\_flag** | u(1) |
| **slice\_granularity** | u(2) |
| **shared\_pps\_info\_enabled\_flag** | u(1) |
| if( shared\_pps\_info\_enabled\_flag ) |  |
| if( adaptive\_loop\_filter\_enabled\_flag ) |  |
| alf\_param( ) |  |
| if( cu\_qp\_delta\_enabled\_flag ) |  |
| **max\_cu\_qp\_delta\_depth** | u(4) |
| **dcim\_enabled\_flag** | u(1) |
| rbsp\_trailing\_bits( ) |  |
| } |  |

**dcim\_enabled\_flag** equal to 1 specifies that DCIM intra prediction is applied according to the signaled intra prediction mode.

1. **Changes in prediction unit syntax**

|  |  |
| --- | --- |
| prediction\_unit( x0, y0, log2PUWidth, log2PUHeight, PartIdx ,    InferredMergeFlag ) { | **Descriptor** |
| if( skip\_flag[ x0 ][ y0 ] ) { |  |
| if( NumMergeCand > 1 ) |  |
| **merge\_idx[** x0 **][** y0 **]** | ue(v) | ae(v) |
| } else if( PredMode = = MODE\_INTRA ) { |  |
| if( PartMode == PART\_2Nx2N &&  log2PUWidth >= Log2IPCMCUSize ) |  |
| **pcm\_flag** | u(1) | ae(v) |
| if( pcm\_flag ) { |  |
| while ( !byte\_aligned( ) ) |  |
| **pcm\_alignment\_zero\_bit** | u(v) |
| for( i = 0; i < 1 << ( log2CUSize << 1 ); i++ ) |  |
| **pcm\_sample\_luma**[ i ] | u(v) |
| for( i = 0; i < ( 1 << ( log2CUSize << 1 ) ) >> 1; i++ ) |  |
| **pcm\_sample\_chroma**[ i ] | u(v) |
| } else { |  |
| **prev\_intra\_luma\_pred\_flag[** x0 **][** y0 **]** | u(1) | ae(v) |
| if( prev\_intra\_luma\_pred\_flag[ x0 ][ y0 ] ) { |  |
| if ( log2PUWidth < 5 ) |  |
| **dcim\_flag[**x0**][**y0**]** | u(1) | ae(v) |
| if ( NumMPMCand > 1) |  |
| **mpm\_idx[** x0 **][** y0 **]** | u(1) | ae(v) |
| } |  |
| else |  |
| **rem\_intra\_luma\_pred\_mode[** x0 **][**y0 **]** | ce(v) | ae(v) |
| if( IntraPredMode[ x0 ][ y0 ] == 2 ) |  |
| **planar\_flag\_luma**[ x0 ][ y0 ] | u(1) | ae(v) |
| if(intraPredMode == Intra\_BIPRED&& NumBipredCand > 1) |  |
| **intra\_bipred\_luma\_pred\_flag[** x0 **][** y0 **]** | u(1) | ae(v) |
| if (dcim\_flag[ x0 ][ y0 ] && (log2PUWidth == 3 || log2PUWidth == 4 ) |  |
| { |  |
| **dcim\_main\_edge\_flag** | u(1) | ae(v) |
| if (dcim\_main\_edge\_flag[ x0 ][ y0 ]) |  |
| { |  |
| **dcim\_sub\_edge\_ idx\_abs\_minus\_one** | ce(v) | ae(v) |
| **dcim\_sub\_edge\_sign\_flag** | u(1) |
| **}** |  |
| } |  |
| **intra\_chroma\_pred\_mode**[ x0 ][ y0 ] | ue(v) | ae(v) |
| SignaledAsChromaDC =   ( chroma\_pred\_from\_luma\_enabled\_flag ?  intra\_chroma\_pred\_mode[ x0 ][ y0 ] == 3 :  intra\_chroma\_pred\_mode[ x0 ][ y0 ] == 2 ) |  |
| if( IntraPredMode[ x0 ][ y0 ] != 2 &&   IntraPredMode[ x0 ][ y0 ]!=34 && SignaledAsChromaDC ) |  |
| **planar\_flag\_chroma**[ x0 ][ y0 ] | u(1) | ae(v) |
| } |  |
| } else { /\* MODE\_INTER \*/ |  |
| if( !InferredMergeFlag ) |  |
| if( entropy\_coding\_mode\_flag || PartMode != PART\_2Nx2N ) |  |
| **merge\_flag[** x0 **][** y0 **]** | u(1) | ae(v) |
| if( merge\_flag[ x0 ][ y0 ] && NumMergeCand > 1 ) { |  |
| **merge\_idx[** x0 **][** y0 **]** | ue(v) | ae(v) |
| } else { |  |
| if( slice\_type = = B ) { |  |
| if(!entropy\_coding\_mode\_flag &&  use\_combined\_inter\_pred\_ref( x0, y0 ) ) { |  |
| **combined\_inter\_pred\_ref\_idx** | ue(v) |
| if( combined\_inter\_pred\_ref\_idx == MaxPredRef ) |  |
| **inter\_pred\_flag**[ x0 ][ y0 ] | ue(v) |
| } else |  |
| **inter\_pred\_flag[** x0 **][** y0 **]** | ue(v) | ae(v) |
| } |  |
| if( inter\_pred\_flag[ x0 ][ y0 ] = = Pred\_LC ) { |  |
| if( num\_ref\_idx\_lc\_active\_minus1 > 0 ) { |  |
| if( !entropy\_coding\_mode\_flag &&  use\_combined\_inter\_pred\_ref( x0, y0 ) ) { |  |
| if( combined\_inter\_pred\_ref\_idx == MaxPredRef ) |  |
| **ref\_idx\_lc\_minus4**[ x0 ][ y0 ] | ue(v) |
| } else |  |
| **ref\_idx\_lc[** x0 **][** y0 **]** | ae(v) |
| } |  |
| **mvd\_lc[** x0 **][** y0 **][** 0 **]** | se(v) | ae(v) |
| **mvd\_lc[** x0 **][** y0 **][** 1 **]** | se(v) | ae(v) |
| if( NumMVPCand( LcToLx ) > 1 ) |  |
| **mvp\_idx\_lc[ x0 ][ y0 ]** | ue(v) | ae(v) |
| } |  |
| else { /\* Pred\_L0 or Pred\_BI \*/ |  |
| if( num\_ref\_idx\_l0\_active\_minus1 > 0 ) { |  |
| if( !entropy\_coding\_mode\_flag &&  use\_combined\_inter\_pred\_ref( x0, y0 ) ) { |  |
| if( combined\_inter\_pred\_ref\_idx == MaxPredRef ) |  |
| **ref\_idx\_l0\_minusX**[ x0 ][ y0 ] | ue(v) |
| } else |  |
| **ref\_idx\_l0\_minusX**[ x0 ][ y0 ] | ue(v) | ae(v) |
| } |  |
| **mvd\_l0[** x0 **][** y0 **][** 0 **]** | se(v) | ae(v) |
| **mvd\_l0[** x0 **][** y0 **][** 1 **]** | se(v) | ae(v) |
| if( NumMVPCand( L0 ) > 1 ) |  |
| **mvp\_idx\_l0[ x0 ][ y0 ]** | ue(v) | ae(v) |
| } |  |
| if( inter\_pred\_flag[ x0 ][ y0 ] = = Pred\_BI ) { |  |
| if( num\_ref\_idx\_l1\_active\_minus1 > 0 ) { |  |
| if( !entropy\_coding\_mode\_flag &&  use\_combined\_inter\_pred\_ref( x0, y0 ) ) { |  |
| if( combined\_inter\_pred\_ref\_idx == MaxPredRef ) |  |
| **ref\_idx\_l1\_minusX**[ x0 ][ y0 ] | ue(v) |
| } else |  |
| **ref\_idx\_l1[** x0 **][** y0 **]** | ue(v) | ae(v) |
| } |  |
| **mvd\_l1[** x0 **][** y0 **][** 0 **]** | se(v) | ae(v) |
| **mvd\_l1[** x0 **][** y0 **][** 1 **]** | se(v) | ae(v) |
| if( NumMVPCand( L1 ) > 1 ) |  |
| **mvp\_idx\_l1[ x0 ][ y0 ]** | ue(v) | ae(v) |
| } |  |
| } |  |
| } |  |
| } |  |

**dcim\_flag** specifies whether the associated prediction unit is predicted with one of the DCIM modes. If the dcim\_flag is equal to 1, the associated prediction unit is predicted with one of the DCIM modes, otherwise the associated prediction unit is predicted with one of the remaining modes. When dcim\_flag is not present, it shall be inferred to be equal to 0.

**dcim\_main\_edge\_flag** specifies whether the associated prediction unit is predicted with the DCIM mode corresponding to the main edge direction which is derived as edgeSlope[ 0 ] in specification of edge detection process according to subclause 8.3.3.1.9.1. If dcim\_main\_edge\_flag is equal to 1, DCIM mode corresponding to the main edge direction is used to predict the associated prediction unit. Otherwise (dcim\_main\_edge\_flag is equal to 0), the DCIM mode corresponding to one of the sub edge direction is used to predict the associated prediction unit. When dcim\_main\_edge\_flag is not present, it shall be inferred to be equal to 1.

**dcim\_sub\_edge\_idx\_abs\_minus\_one** specifies the index to identify one of the DCIM mode corresponding to a sub edge direction that is used to predict the associated prediction unit. When dcim\_sub\_edge\_idx is not present, it shall be inferred to be equal to 0.

**dcim\_sub\_edge\_sign\_flag** indicates the side on which the sub direction of DCIM mode is located relative to the main edge.

The index of DCIM mode DcimDir that is used to identify one of the DCIM modes is derived as follows.

DcimDir[ x0 ][ y0 ] = dcim\_main\_edge\_flag ? 0 :   
 ((dcim\_sub\_edge\_idx\_abs\_minus\_one + 1)<<1 + (dcim\_sub\_edge\_sign\_flag ? 0 : 1))

**intra\_bipred\_luma\_pred\_flag** [ x0 ][ y0 ] specify the bidirectional intra prediction flag for luma samples. The array indices x0, y0 specify the location ( x0, y0 ) of the top-left luma sample of the considered prediction block relative to the top-left luma sample of the picture. When intra\_bipred\_luma\_pred\_flag [ x0 ][ y0 ] is equal to 1, the intra prediction mode is inferred from a bidirectional intra-predicted prediction unit according to subclause 8.3.1. Intra\_BIPRED specify one of intra prediction modes, which show that IntraBipredFlag is equal to 1.

1. **Changes in "8.3.1 Derivation process for luma intra prediction mode "**

Table 8‑5 specifies the value for the intra prediction mode and the associated names.

**Table 8‑5 – Specification of intra prediction mode and associated names**

|  |  |
| --- | --- |
| **Intra prediction mode** | **Associated names** |
| 0 | Intra\_Vertical |
| 1 | Intra\_Horizontal |
| 2 | Intra\_DC |
| 3..33) | Intra\_Angular |
| 34 | Intra\_Planar |
| 35 | Intra\_FromLuma (used only for chroma) |
| 36 | Intra\_DCIM |
| 37, 38 | Intra\_BIPRED |

IntraPredMode[ xB ][ yB ] is derived as the following ordered steps.

1. The derivation process for neighbouring treeblocks specified in subclause XXX with ( xB,  yB ) given as input and the output is assigned to tbAddrA and tbAddrB specifying the treeblock addresses of treeblocks covering ( xBA,  yBA ) and ( xBB, yBB ) respectively where ( xBA,  yBA ) is set equal to ( xB-1,  yB ) and ( xBB,  yBB ) is set equal to ( xB,  yB-1 ).
2. For N being either replaced A or B, the variables intraPredModeN are derived as follows.

* If the treeblock with address tbAddrN is not available, intraPredModeN is set equal to -1
* Otherwise, if the coding unit covering ( xBN,  yBN ) is not coded as intra mode, intraPredModeN is set equal to Intra\_DC,
* Otherwise, intraPredModeN is set equal to IntraPredMode[ xBN ][ yBN ], where IntraPredMode is the variable array assigned to the coding unit covering the luma location ( xBN, yBN ).

1. For N being either replaced A or B, the variables intraPredModeN are further modified to be equal to Intra\_DC when intraPredModeN is equal to Intra\_Planar.
2. For N being either replaced A or B, the variables candIntraPredModeN are derived as follows.

* If intraPredModeN is greater than or equal to intraPredModeNum
* If intraPredModeNum is equal to 4 then candIntraPredModeN is set equal to mapIntraPredMode3[ intraPredModeN ]
* Otherwise candIntraPredModeN is set equal to mapIntraPredMode9[ intraPredModeN ].
* Otherwise, candIntraPredModeN is set equal to intraPredModeN

1. The candModeList[x] and NumMPMCand are derived as follows:

* If dcim\_enabled\_flag is equal to 0, the following is applied:
  + If both candIntraPredModeN are not available, then the value 2 is assigned to candModeList[ 0 ] and NumMPMCand is set equal to 1
  + Otherwise, if only one candIntraPredModeN is available or if both candIntraPredModeN are the same, then this candIntraPredModeN is assigned to candModeList[ 0 ] and NumMPMCand is set equal to 1
  + Otherwise, both candIntraPredModeN are assigned to the candidate modes list with the smaller of the two candidates at candModeList[ 0 ] and the larger at candModeList[ 1 ] and NumMPMCand is set equal to 2.
* If dcim\_enabled\_flag is equal to 1, the following is applied:
  + If the smaller of the two candidates candIntraPredModeN is not equal to 2, candModeList[0] is assigned the value 2, candModeList[1] is assigned the smaller of the two candidates candIntraPredModeN and NumMPMCand is set equal to 2
  + If the smaller of the two candidates candIntraPredModeN is equal to 2, candModeList[0] is assigned the value 2 and NumMPMCand is set equal to 1.

1. IntraPredMode[ xB ][ yB ] is derived by applying the following procedure.

* If prev\_intra\_pred\_flag[ xB ][ yB ] is true and dcim\_flag[ xB ][ yB ] is false, the IntraPredMode[ xB ][ yB ] is set equal to candModeList[mpm\_idx[ xB ][ yB ]]
* If prev\_intra\_pred\_flag[ xB ][ yB ] is true and dcim\_flag[ xB ][ yB ] is true, the IntraPredMode[ xB ][ yB ] is set equal to Intra\_DCIM
* Otherwise IntraPredMode[ xB ][ yB ] is derived by applying the following procedure
* IntraPredMode[ xB ][ yB ] = rem\_intra\_pred\_mode
* for (cIdx = 0; cIdx < NumMPMCand; cIdx++)

if (IntraPredMode[ xB ][ yB ]>= candModeList[cIdx]) IntraPredMode[ xB ][ yB ]++

1. When IntraPredMode[ xB ][ yB ] is equal to Intra\_DC and planar\_flag\_luma is equal to 1, IntraPredMode[ xB ][ yB ] is further modified to be equal to Intra\_Planar.
2. **Changes in "8.3.2 Derivation process for chroma intra prediction mode "**

Input to this process is a luma location ( xB, yB ) specifying the top-left luma sample of the current block relative to the top‑left luma sample of the current picture.

Output of this process is the variable IntraPredModeC.

IntraPredModeC is derived as follows,

– If dcim\_flag is equal to 1, IntraPredModeC is set equal to Intra\_DC.

– Otherwise (dcim\_flag is equal to 0), the chroma intra prediction mode IntraPredModeC is derived as specifed in Table 8‑3 or Table 8‑4 with intra\_chroma\_pred\_mode, IntraPredMode[ xB ][ yB ] and chroma\_pred\_from\_luma\_enabled\_flag as inputs.

– IntraPredModeC is further modified to be equal to Intra\_Planar when IntraPredModeC is equal to Intra\_DC and planar\_flag\_chroma is equal to 1.

1. **Changes in "****8.3.3.1 Intra sample prediction"**

Inputs to this process are:

– a sample location ( xB, yB ) specifying the top-left sample of the current block relative to the top‑left sample of the current picture,

– a variable intraPredMode specifying the luma intra prediction mode,

– a variable nS specifying the prediction size.

– a variable cIdx specifying the chroma component of the current block,

Output of this process is:

– the predicted samples predSamples[ x, y ], with x, y =0..nS-1.

The nS\*12+9 neighbouring samples p[ x, y ] that are constructed luma samples prior to the deblocking filter process, with x = -3..-1, y = -3..nS\*2-1 and x = 0..nS\*2-1, y=-3..-1, are derived as follows.

– The luma location (xBN, yBN ) is specified by

xBN = xB + x  (8‑14)

yBN = yB +y  (8‑15)

– The derivation process for neighbouring locations in subclause XXX is invoked for sample location with ( xBN, yBN ) as input and the treeblock address tbAddrN as output.

– Each sample p[ x, y ] with x = -3..-1, y = -3..nS\*2-1 and x = 0..nS\*2-1, y=-3..-1 is derived as follows

* If any of the following condition is true, the sample p[ x, y ] is marked as “not available for intra prediction”
  + tbAddrN is not available
  + the coding unit covering ( xBN, yBN ) is not coded as intra mode and constrained\_intra\_pred\_flag is equal to 1
* Otherwise, the sample p[ x, y ] is marked as “available for intra prediction” and the sample at the location ( xBN, yBN ) inside the treeblock tbAddrN is assigned to p[ x, y ]

1. **Addition of new subsection "8.3.3.1.9 Specification of Intra\_DCIM prediction mode"**

Inputs to this process are:

– neighbouring samples p[ x, y ], with x, y = -3..2\*nS-1,

– a variable nS specifying the prediction block size.

Output of this process is:

– predicted samples predSamples[ x, y ], with x, y =0..nS-1.

This intra prediction mode is invoked when intraPredMode is equal to 36.

The values of the prediction samples predSamples[ x, y ], with x, y = 0..nS-1, are derived as the following ordered steps:

1. Edge detection process in the neighboring pixels specified in subclause 8.3.3.1.9.1 is invoked with the samples array p and the prediction size nS as the input and the output is an array containing edge information edgeSlope.
2. The process specified in subclause 8.3.3.1.9.3 is invoked with the samples array p, the prediction size nS and the array containing edge information edgeSlope as the input and the output is the predicted samples array predSamples.
3. **Addition of new subsection "8.3.3.1.9.1 Specification of edge detection process"**

Inputs to this process are:

– neighbouring samples p[ x, y ], with x, y = -3..2\*nS-1,

– a variable nS specifying the prediction block size.

Output of this process is:

* an array containing edge information edgeSlope.

The edge direction is calculated according to the following ordered steps:

1. The variables sigma[ k ] with k = 0..5 are set equal to 0.
2. The following ordered steps are applied for each values x = -3..2\*nS-1 and y = -3:
   1. The variables resx and resy are derived as:
      * If all the samples p[ x+i, y+j ] with i = 0..2 and j = 0..2 are marked as "available for intra prediction", the variable resx and resy are derived as:

resx = (-1)\*p[ x, y ] + (-2)\*p[ x+1, y ] + (-1)\*p[ x+2, y ] + 1\*p[ x, y+2 ] + 2\*p[ x+1, y+2 ] + 1\*p[ x+2, y+2 ]

resy = (-1)\*p[ x, y ] + (-2)\*p[ x, y+1 ] + (-1)\*p[ x, y+2 ] + 1\*p[ x+2, y ] + 2\*p[ x+2, y+1 ] + 1\*p[ x+2, y+2 ]

* + - Otherwise, the variables resx and resy are set equal to 0.
  1. The variables sigma[ k ] with k = 0..2 are modified as:

sigma[ 0 ] = sigma[ 0 ] – resx \* resy

sigma[ 1 ] = sigma[ 1 ] + resx \* resx

sigma[ 2 ] = sigma[ 2 ] + resy \* resy

1. The following ordered steps are applied for each values x = -3 and y = -2.. 2\*nS-1:
   1. The variables resx and resy are derived as:
      * If all the samples p[ x+i, y+j ] with i = 0..2 and j = 0..2 are marked as "available for intra prediction", the variable resx and resy are derived as:

resx = (-1)\*p[ x, y ] + (-2)\*p[ x+1, y ] + (-1)\*p[ x+2, y ] + 1\*p[ x, y+2 ] + 2\*p[ x+1, y+2 ] + 1\*p[ x+2, y+2 ]

resy = (-1)\*p[ x, y ] + (-2)\*p[ x, y+1 ] + (-1)\*p[ x, y+2 ] + 1\*p[ x+2, y ] + 2\*p[ x+2, y+1 ] + 1\*p[ x+2, y+2 ]

* + - Otherwise, the variables resx and resy are set equal to 0.
  1. The variables sigma[ k ] with k = 3..5 are modified as:

sigma[ 3 ] = sigma[ 3 ] – resx \* resy

sigma[ 4 ] = sigma[ 4 ] + resx \* resx

sigma[ 5 ] = sigma[ 5 ] + resy \* resy

1. The variables sigma[ 0 ] and sigma[ 3 ] are multiplied by 2.
2. The process to calculate the slope as specified in subsection 8.3.3.1.9.2 is invoked with sigma[ 0 ], sigma[ 1 ] and sigma[ 2 ] as input and the output is the value of an edge slope edgeSlope[ 1 ].
3. The process to calculate the slope as specified in subsection 8.3.3.1.9.2 is invoked with sigma[ 3 ], sigma[ 4 ] and sigma[ 5 ] as input and the output is the value of an edge slope edgeSlope[ 2 ].
4. The process to calculate the slope as specified in subsection 8.3.3.1.9.2 is invoked with sigma[ 0 ] + sigma[ 3 ], sigma[ 1 ] + sigma[ 4 ] and sigma[ 2 ] + sigma[ 5 ] as input and the output is the value of an edge slope edgeSlope[ 0 ].
5. **Addition of new subsection "8.3.3.1.9.2 Specification of process to calculate the slope"**

Inputs to this process are:

– values sigma[ 0 ], sigma[ 1 ] and sigma[ 2 ].

Output of this process is:

* a value containing the slope of an edge edgeSlope.

The value edgeSlope is derived as specified in the following ordered steps:

to do

1. **Addition of new subsection "8.3.3.1.9.3 Specification of edge based prediction mode"**

Inputs to this process are:

– neighbouring samples p[ x, y ], with x, y = -3..2\*nS-1,

* a variable nS specifying the prediction block size,
* an array containing edge information edgeSlope.

Output of this process is:

– predicted samples predSamples[ x, y ], with x, y =0..nS-1.

The values of the prediction samples predSamples[ x, y ], with x, y = 0..nS-1, are derived according to the following ordered steps:

to do

1. **Addition of new subsection "Derivation process for intraPredModeL0 and intraPredModeL1” (subclause 8.3.3.1.x)**

Inputs to this process are:

– a luma location ( xB, yB ) specifying the top-left luma sample of the current block relative to the top left luma sample of the current picture,

– a variable log2TrafoSize specifying the size of the current prediction unit,

– variable arrays IntraPredMode (If available) that are previously (in decoding order) derived for adjacent coding units.

Output of this process is the variable IntraPredModeL0[ xB ][ yB ] and IntraPredModeL1[ xB ][ yB ].

Table 8 – 1 specifies the number of luma intra prediction modes intraPredModeNum depending on log2TrafoSize.

Table 8 – 1 Specification of intraPredModeNum.

|  |  |
| --- | --- |
| **log2TrafoSize** | **intraPredModeNum** |
| 2 | 20 |
| 3 | 37 |
| 4 | 37 |
| 5 | 37 |
| 6 | 3 |

IntraPredModeL0[ xB ][ yB ] and IntraPredModeL1[ xB ][ yB ] is derived as follows.

Two Intra\_BIPRED mode (BIPRED1 and BIPRED2) and NumBipredCand which means the number of Intra\_BIPRED modes are set as follows.

1. Mode "a" and "b" is set to prediction modes of left neighboring PU A and above neighboring PU B respectively. If neighboring block A and B was coded using Intra\_BIPRED mode, mode "a" and "b" is set to IntraPredModeL0 of neighboring block A and B, respectively .
2. Restricted modes "ac" and "bc" are mapped from "a" and "b" in following equation;

ac = MappedDirTable [a], bc = MappedDirTable [b].

1. IntraPredModeL0 and IntraPredModeL1 for BIPRED1 and BIPRED2 are set as follows.

* BIPRED1 : combination of "a" and "b"

IntraPredModeL0 = a, IntraPredModeL1 = b

* BIPRED2 : combination of "min(a, b)" and neighboring direction of " min(a, b)"

IntraPredModeL0 = min(a, b), IntraPredModeL1 = MappedNeighborDirTable[min(a, b)]

where, MappedNeighborDirTable [x] is a table which map UIP mode "x" to the neighbor prediction direction of "x" in the range of available directions. Table A-5 specifies the details of tables MappedNeighborDirTable [].

1. NumBipredCand is set as follows.

* If the combination of BIPRED1 is equal to one of BIPRED2, NumBipredCand is set to 1.
* Otherwise, NumBipredCand is set to 2.

Table 8 – 2 specifies the mapped value of IntraPredModeLX[ xB ][ yB ].

Table 8 – 2 Spacification for MappedDirTable.

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

Table 8 – 2 Spacification for MappedNeighborDirTable.

|  |  |  |
| --- | --- | --- |
| IntraPredModeLX | MappedNeighborDirTable  [IntraPredModeLX]  for PU4x4, 8x8 | MappedNeighborDirTable  [IntraPredModeLX]  for PU16x16, 32x32 |
| 0 | 1 | 2 |
| 1 | 0 | 2 |
| 2 | 0 | 1 |
| 3 | 6 | 2 |
| 6 | 9 | 2 |
| 9 | 6 | 2 |

1. **Addition of new subsection "Specification of Intra\_BIPRED mode” (subclause 8.3.3.1.x)**

Inputs to this process are:

– predicted samples predL0[ x, y ] and predL1[ x, y ], with x, y = 0..nS-1,

– a variable nS specifying the prediction size.

Output of this process is:

– predicted samples predSamples[ x, y ], with x, y =0..nS-1.

This intra prediction mode is invoked when intraPredMode is equal to Intra\_BIPRED which means IntraBipredFlag is equal to 1.

The values of the prediction samples predSamples[ x, y ], with x, y = 0..nS-1, are derived as follows.

predSamples[ x, y ] = ( w[ x, y ] \* predL0[ x, y ] + ((1<<w\_shift) – w[ x, y ]) \* predL1[ x, y ]

+ (1<<(w\_shift –1)))>> (w\_shift) (8‑x)

w[ x, y ] shows the weighting list for predLX[ x, y ] according to the predSamples[ x, y ]. w\_shift specifies the shift value of the weighting list for the rounding operation and is set to 10. The weighting lists w[ x, y ] for PU32x32 is set to (1<<(w\_shift-1)).

1. **Changes in “9.2.2.5 Initialisation process for intraModeTable”**

Outputs of this process are initial values of the variable array intraModeTable.

The variable array intraModeTable is initialized as specified in Table 9‑6.

Table ‑ – Specification of intraModeTable[k][codeNum]

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **codeNum** | | | | | | | | | | | | | | | | |
| **k** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** | **17** |
| 0 | 0 | 15 | 11 | 10 | 16 | 13 | 7 | 9 | 4 | 14 | 2 | 3 | 6 | 8 | 5 | 12 | 1 | - |
| 1 | 14 | 10 | 9 | 0 | 15 | 13 | 7 | 2 | 8 | 3 | 12 | 6 | 4 | 11 | 1 | 5 | - | - |
| 2 | 2 | 0 | 29 | 30 | 20 | 1 | 33 | 21 | 28 | 15 | 7 | 16 | 8 | 11 | 31 | 22 | 19 | 32 |
| 3 | 2 | 1 | 28 | 0 | 29 | 20 | 32 | 27 | 19 | 51 | 21 | 7 | 14 | 10 | 11 | 30 | 31 | 18 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **codeNum** | | | | | | | | | | | | | | | | |
| **k** | **18** | **19** | **20** | **21** | **22** | **23** | **24** | **25** | **26** | **27** | **28** | **29** | **30** | **31** | **32** | **33** |  |  |
| 2 | 10 | 27 | 14 | 4 | 18 | 12 | 23 | 17 | 24 | 26 | 5 | 25 | 5 | 9 | 3 | 13 |  |  |
| 3 | 16 | 26 | 8 | 17 | 4 | 22 | 9 | 23 | 6 | 25 | 13 | 24 | 12 | 3 | 5 |  |  |  |

1. **Changes in “9.2.4.6 Parsing process for rem\_intra\_luma\_pred\_mode”**

This process is invoked when entropy\_coding\_mode\_flag is equal to 0 for parsing syntax element rem\_intra\_luma\_pred\_mode in subclause 7.3.7.

Inputs to this process are bits from slice data, a variable puSize specifying the size of the current prediction unit, NumMPMCand and a variable array intraModeTable.

Outputs of this process are the syntax element rem\_intra\_luma\_pred\_mode and value-updated intraModeTable.

The value of rem\_intra\_luma\_pred\_mode is derived as follows.

– Based on the input puSize, the value of a variable intraPredModeNum indicating the number of intra prediction modes for the given size of prediction unit is obtained according to Table 7‑11.

– If intraPredModeNum is equal to 3, read one bit and assign its value to rem\_intra\_luma\_pred\_mode. No further step is carried out. Othewise,

– The values of variable k and vlcNum are obtained as follows

* + If intraPredModeNum is equal to 17
    - If NumMPMCand is equal to 1, set vlcNum equal to 16 and k equal to 0.
    - Otherwise, set vlcNum equal to 17 and k equal to 1.
  + If intraPredModeNum is equal to 34
    - If NumMPMCand is equal to 1, set vlcNum equal to 18 and k equal to 2.
    - Otherwise, set vlcNum equal to 19 and k equal to 3.
  + The parsing process described in subclause 9.2.1 is invoked with vlcNum as input and the variable codeNum as output.
  + The variable rankIntraMode is set equal to intraModeTable[k][codeNum].
  + The variable lastRankIdx is set equal to (intraPredModeNum + (k==0 ? -1 : -2)).
  + If rankIntraMode is equal to lastRankIdx, a flag remFlag is parsed from slice data and the value of rem\_intra\_luma\_pred\_mode is set equal to (rankIntraMode + remFlag).
  + Otherwise (rankIntraMode is not equal to lastRankIdx), the value of rem\_intra\_luma\_pred\_mode is set equal to rankIntraMode.
  + The value of a variable counterNum is set to 0. The variable array intraModeTable[k] is updated by invoking process in subclause 9.2.3 with intraModeTable[k], codeNum and counterNum as inputs.

1. **Changes in “9.3.1.1 Initialization process for context variables”**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Syntax element** | | **ctxIdxTable** | **Slice Type** | | | | |
| **I** | **P** | | **B** | |
| prediction\_unit() | | prev\_intra\_luma\_pred\_flag |  | 0 | | 1 | | 2 |
| rem\_intra\_luma\_pred\_mode |  | 0 | | 1 | | 2 |
| dcim\_flag | x1 | 0..2 | | 3..5 | | 6..8 |
| dcim\_main\_edge\_flag | 11 | 0 | | 1 | | 2 |
| **dcim\_sub\_edge\_idx\_abs\_minus\_one** | 11 | 0 | | 1 | | 2 |
| **intra\_bipred\_luma\_pred\_flag** | x2 | 0 | | 1 | | 2 |

Table ‑11 – Values of variable m and n for rem\_intra\_luma\_pred\_mode, dcim\_main\_edge\_flag, dcim\_sub\_edge\_idx\_abs\_minus\_one ctxIdx

|  |  |  |  |
| --- | --- | --- | --- |
| **Initialisation variables** | **rem\_intra\_luma\_pred\_mode, dcim\_main\_edge\_flag, dcim\_sub\_edge\_idx\_abs\_minus\_one ctxIdx** | | |
| **0** | **1** | **2** |
| **m** | -3 | -2 | 1 |
| **n** | 65 | 61 | 55 |

Table 9‑x1 – Values of variable m and n for dcim\_flag ctxIdx

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| * 1. **Initialisation variables** | **dcim\_flag ctxIdx** | | | | | | | | |
| **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** |
| **m** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **n** | 64 | 64 | 64 | 64 | 64 | 64 | 64 | 64 | 64 |

Table 9‑x2 – Values of variable m and n for intra\_bipred\_luma\_pred\_flag ctxIdx

|  |  |  |  |
| --- | --- | --- | --- |
| **Initialisation variables** | **rem\_intra\_luma\_pred\_mode ctxIdx** | | |
| **0** | **1** | **2** |
| **m** | 0 | 0 | 0 |
| **n** | 64 | 64 | 64 |

1. **Changes in “9.3.2 Binarization process”**

| **Syntax element** |  | **Type of binarization** | **maxBinIdxCtx** | **ctxIdxTable** | **ctxIdxOffset** |
| --- | --- | --- | --- | --- | --- |
| **prev\_intra\_luma\_pred\_flag** | **I** | **FL, cMax = 1** | **0** |  | **0** |
|  | **P** |  | **0** |  | **1** |
|  | **B** |  | **0** |  | **2** |
| **rem\_intra\_luma\_pred\_mode** | **I** | **as specified in subclause** | **0** |  | **0** |
|  | **P** |  | **0** |  | **1** |
|  | **B** |  | **0** |  | **2** |
| **dcim\_flag** | **I** | **FL, cMax = 1** | **0** | **xx** | **0** |
|  | **P** |  | **0** | **xx** | **1** |
|  | **B** |  | **0** | **xx** | **2** |
| **dcim\_main\_edge\_flag** | **I** | **FL, cMax = 1** | **0** | **11** | **0** |
|  | **P** |  | **0** | **11** | **1** |
|  | **B** |  | **0** | **11** | **2** |
| **dcim\_sub\_edge\_idx\_abs\_minus\_one** | **I** | **TU, cMax = 1** | **0** | **11** | **0** |
|  | **P** |  | **0** | **11** | **1** |
|  | **B** |  | **0** | **11** | **2** |
| **intra\_bipred\_luma\_pred\_flag** | **I** | **FL, cMax = 1** | **0** | **xx** | **0** |
|  | **P** |  | **0** | **xx** | **1** |
|  | **B** |  | **0** | **xx** | **2** |

1. **Changes in “9.3.3.1Derivation process for ctxIdx”**

| **Syntax element** | **ctxIdxTable,  ctxIdxOffset** | | **binIdx** | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **>=4** |
| prev\_intra\_luma\_pred\_flag |  | 0 | 0 | na | na | na | na |
| 1 | 0 | na | na | na | na |
| 2 | 0 | na | na | na | na |
| rem\_intra\_luma\_pred\_mode |  | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 |
| dcim\_flag |  | 0 | 0 | na | na | na | na |
| 1 | 0 | na | na | na | na |
| 2 | 0 | na | na | na | na |
| **dcim\_main\_edge\_flag** | 11 | 0 | 0 | na | na | na | na |
| 1 | 0 | na | na | na | na |
| 2 | 0 | na | na | na | na |
| **dcim\_sub\_edge\_idx\_abs\_minus\_one** | 11 | 0 | 0 | na | na | na | na |
| 1 | 0 | na | na | na | na |
| 2 | 0 | na | na | na | na |
| **intra\_bipred\_luma\_pred\_flag** | xx | 0 | 0 | na | na | na | na |
| 1 | 0 | na | na | na | na |
| 2 | 0 | na | na | na | na |

1. **Changes in “9.3.3.1.1 Assignment process of ctxIdxInc using neighbouring syntax elements”**

Subclause 9.3.3.1.1.1 specifies the derivation process of ctxIdxInc for the syntax elements alf\_cu\_flag, split\_coding\_unit\_flag, skip\_flag, merge\_flag, intra\_chroma\_pred\_mode, inter\_pred\_flag, ref\_idx\_lc, ref\_idx\_l0, ref\_idx\_l1, mvd\_l0, mvd\_l1, mvd\_lc, no\_residual\_data\_flag, cbf\_luma, cbf\_cb and cbf\_cr, dcim\_flag.

1. **Changes in “9.3.3.1.1.1 Derivation process of ctxIdxInc using left and above syntax elements”**

|  |  |  |  |
| --- | --- | --- | --- |
| **Syntax element** | **condL** | **condA** | **ctxIdxInc** |
| dcim\_flag | dcim\_flag[ xL ][ yL ] | dcim\_flag[ xA ][ yA ] | ( condL && availableL ) + ( condA && availableA ) |