The proposed working draft modifications are as follows.

H.8.4.4.2.5 Specification of Intra\_DC (1) prediction mode

If the sdc\_flag is equal to 1, this process does nothing. Else,

~~T~~the specifications in subclause 8.4.4.2.5 apply.

H.8.4.4.2.7 Specification of Intra\_DepthPartition (35, 36) prediction mode

…

This intra prediction mode is invoked when intraPredMode is equal to 35 or 36.

If the sdc\_flag is equal to 1, this process does nothing. Else, the following process applies.

…

H.8.4.4.2.12 Depth partition value derivation and assignment process

…

Output of this process is:

* predicted samples predSamples[ x ][ y ], with x, y =0..nT−1.

The depth partition value derivation process as specified in subclause .1 is invoked with the neighbouring samples p[ x ][ y ], the binary pattern wedgePattern [ xB ][ yB ], the transform size nT, the dcOffsetAvailFlag set equal to ( intraPredMode = = Intra\_DepthPartition(36) ), intraChainFlag set equal to 0, and the DC Offsets DcOffsetP0[ xB ][ yB ], and DcOffsetP1[ xB ][ yB ] as inputs and the output is assigned to predVal[ X ] with X = 0, 1.

The predicted sample values predSamples are derived as follows for x = 0..nT−1 and for y = 0..nT−1.

* + - When partitionPattern[ x ][ y ] is equal to X, the following applies.
      * predSamples[ x ][ y ] = predVal[ X ].

~~The variables vertEdgeFlag and horEdgeFlag are derived as specified in the following:~~

* 1. ~~vertEdgeFlag = ( partitionPattern[ 0 ][ 0 ]  !=  partitionPattern[ nT − 1 ][ 0 ] ) ? 1 : 0 (‑51)~~
  2. ~~horEdgeFlag = ( partitionPattern[ 0 ][ 0 ]  !=  partitionPattern[ 0 ][ nT − 1 ] ) ? 1 : 0 (‑52)~~

~~The variables dcVal0 and dcVal1 are derived as specified in the following:~~

* ~~If vertEdgeFlag is equal to horEdgeFlag, the following applies:~~
  + 1. ~~dcValBR = horEdgeFlag ? ( ( p[ −1 ][ nT − 1 ] + p[ nT − 1 ][ −1 ] ) >> 1 ) : ( 1 << ( BitDepth~~~~Y~~ ~~− 1 ) ) (‑53)~~
    2. ~~dcValLT = ( p[ −1 ][ 0 ] + p[ 0 ][ −1 ] ) >> 1 (‑54)~~
* ~~Otherwise ( horEdgeFlag is not equal to vertEdgeFlag), the following applies:~~
  + 1. ~~dcValBR = horEdgeFlag ? p[ −1 ][ nT − 1 ] : p[ nT − 1 ][ −1 ] (‑55)~~
    2. ~~dcValLT = horEdgeFlag ? p[ ( nT – 1 ) >> 1 ][ −1 ] : p[ −1 ][ ( nT – 1 ) >> 1 ] (‑56)~~

1. ~~The variable dcOffsetScale is derived as follows:~~
   1. ~~dcOffsetScale = intraChainFlag ? Clip3( 1, ( 1 << BitDepth~~~~Y~~ ~~) − 1, Round( 2~~ ~~(QP’~~~~Y~~ ~~/10)- 2~~ ~~) ) : 1 (‑57)~~
2. ~~The predicted sample values predSamples[ x ][ y ] are derived as specified in the following:~~

* ~~For x in the range of 0 to ( nT − 1 ), inclusive the following applies:~~ 
  + ~~For y in the range of 0 to ( nT − 1 ), inclusive the following applies:~~ 
    - ~~The variables predDcVal and dcOffset are derived as specified in the following:~~ 
      * 1. ~~predDcVal = ( partitionPattern[ x ][ y ] = = partitionPattern[ 0 ][ 0 ] ) ? dcValLT : dcValBR (‑58)~~
        2. ~~dcOffset = dcOffsetAvailFlag ? ( partitionPattern[ x ][ y ] = = 0 ? dcOffsetP0 : dcOffsetP1 ) : 0 (‑59)~~
        3. ~~dcOffset = dcOffset \* dcOffsetScale (‑60)~~
    - ~~If intraChainFlag is equal to 1 or dlt\_flag[ nuh\_layer\_id ] is equal to 0, the following applies:~~ 
      * 1. ~~predSamples[ x ][ y ] = predDcVal + dcOffset (‑61)~~
    - ~~Otherwise ( intraChainFlag is equal to 0 and dlt\_flag[ nuh\_layer\_id ] is equal to 1 ), the following applies:~~
      * 1. ~~predSamples[ x ][ y ] = Idx2DepthValue[ DepthValue2Idx[ predDcVal ] + dcOffset ] (‑62)~~

**H.8.4.4.2.12.1 Depth partition value derivation process**

Inputs to this process are:

* neighbouring samples p[ x ][ y ], with x, y = −1..2\*nT−1,
* a binary array partitionPattern[ x ][ y ], with x, y =0..nT−1, specifying a partitioning of the prediction block in a partition 0 and a partition 1.
* a variable nT specifying the transform size,
* a flag dcOffsetAvailFlag, specifying whether DC Offset values are available
* a flag intraChainFlag, specifying whether the current intra prediction mode is equal to Intra\_Chain(43,44)
* the variables dcOffsetP0 and dcOffsetP1, specifying the DC offsets for the block partitions

Output of this process is:

* predicted values predVal[ X ]., with X =0, 1.

The variables vertEdgeFlag and horEdgeFlag are derived as specified in the following:

vertEdgeFlag = ( partitionPattern[ 0 ][ 0 ]  !=  partitionPattern[ nT − 1 ][ 0 ] ) ? 1 : 0 (‑51)

horEdgeFlag = ( partitionPattern[ 0 ][ 0 ]  !=  partitionPattern[ 0 ][ nT − 1 ] ) ? 1 : 0 (‑52)

The variables dcVal0 and dcVal1 are derived as specified in the following:

* If vertEdgeFlag is equal to horEdgeFlag, the following applies:

dcValBR = horEdgeFlag ? ( ( p[ −1 ][ nT − 1 ] + p[ nT − 1 ][ −1 ] ) >> 1 ) : ( 1 << ( BitDepthY − 1 ) ) (‑53)

dcValLT = ( p[ −1 ][ 0 ] + p[ 0 ][ −1 ] ) >> 1 (‑54)

* Otherwise ( horEdgeFlag is not equal to vertEdgeFlag), the following applies:

dcValBR = horEdgeFlag ? p[ −1 ][ nT − 1 ] : p[ nT − 1 ][ −1 ] (‑55)

dcValLT = horEdgeFlag ? p[ ( nT – 1 ) >> 1 ][ −1 ] : p[ −1 ][ ( nT – 1 ) >> 1 ] (‑56)

The variable dcOffsetScale is derived as follows:

dcOffsetScale = intraChainFlag ? Clip3( 1, ( 1 << BitDepthY ) − 1, Round( 2 (QP’Y /10)- 2 ) ) : 1 (‑57)

The predicted sample values predVal[ X ] are derived as specified in the following:

* For X in the range of 0 to 1, inclusive the following applies:
  + - The variables predDcVal and dcOffset are derived as specified in the following:
      * 1. predDcVal = ( X == 0 ) ? dcValLT : dcValBR (‑58)
        2. dcOffset = dcOffsetAvailFlag ? ( X = = 0 ? dcOffsetP0 : dcOffsetP1 ) : 0 (‑59)
        3. dcOffset = dcOffset \* dcOffsetScale (‑60)
    - If intraChainFlag is equal to 1 or dlt\_flag[ nuh\_layer\_id ] is equal to 0, the following applies:
      * 1. predVal[ X ] = predDcVal + dcOffset (‑61)
    - Otherwise ( intraChainFlag is equal to 0 and dlt\_flag[ nuh\_layer\_id ] is equal to 1 ), the following applies:
      * 1. predVal[ X ] = Idx2DepthValue[ DepthValue2Idx[ predDcVal ] + dcOffset ] (‑62)

H.8.4.4.3 Depth value reconstruction process

Inputs to this process are:

…

* a variable intraPredMode specifying the prediction mode of the current prediction block
* the neighbouring samples Rec[ x ][ y ], with x, y = −1..2\*nT−1

…

* Otherwise ( dlt\_flag[ nuh\_layer\_id ] is equal to 1 ), the following applies:
  + ~~The variable log2SubSample is set equal to ( nT < 32 ) ? 0 : 1.~~
  + ~~For p in the range of 0 to 1, inclusive, the variable dcPred[ p ] is derived as specified in the following:~~

~~sumPred = 0  
 numPred = 0.   
 for( x = 0; x < ( nT >> log2SubSample ); x++ ) {   
 x~~~~S~~  ~~= x << log2SubSample    
 for ( y = 0; y < ( nT >> log2SubSample ) ;y++ ) {   
 y~~~~S~~ ~~= y << log2SubSample .  
 if ( p = = wedgePattern[ x~~~~S~~~~][ y~~~~S~~~~] ) {  
 sumPred += predSamples[ x~~~~S~~~~][ y~~~~S~~~~]  
 numPred += 1  
 }  
 }  
 }  
 dcPred[ p ] = ( numPred > 0 ) ? ( sumPred / numPred ) : 0~~

The array wedgePattern[ x ][ y ] with x, y =0..nT−1 specifying the binary segmentation pattern is initialized as follows

-For x, y = 0..nT−1 wedgePattern[ x ][ y ] is set equal to 0.

If intraPredMode is equal to Intra\_Planar (0) prediction mode,

dcPred[ 0 ] = dcPred[ 1 ] = ( Rec [ nT ][ −1 ] + Rec [ -1 ][ nT ] + 1 ) >> 1;

Else if intraPredMode is equal to Intra\_DC (1) prediction mode,

dcPred[ 0 ] = dcPred[ 1 ] = ( Rec [ 0 ][ −1 ] + Rec [ -1 ][ 0 ] + 1 ) >> 1;

Else if intraPredMode is equal to Intra\_DepthPartition (35) prediction mode,

The variable wedgePattern[ x ][ y ] with x, y =0..nT−1, is derived as.

wedgePattern = WedgePatternTable[ Log2( nT) ][ wedge\_full\_tab\_idx[ xB ][ yB ] ].

The depth partition value derivation process as specified in subclause .1 is invoked with the neighbouring samples Rec[ x ][ y ], the binary pattern wedgePattern [ xB ][ yB ], the transform size nT, the dcOffsetAvailFlag set equal to false, intraChainFlag set equal to 0, and the DC Offsets DcOffsetP0[ xB ][ yB ], and DcOffsetP1[ xB ][ yB ] as inputs and the output is assigned to dcPred.

If intraPredMode is equal to Intra\_DC (1) prediction mode, or intraPredMode is equal to Intra\_DepthPartition (35) prediction mode, the following applies for x, y = 0..nT−1:

predSamples[ x ][ y ]= dcPred[ wedgePattern[ x ][ y ] ].

…