**Working draft modification for JCTVC-I0224**

All the modifications are highlighted by yellow.

### 7.3.2 Raw byte sequence payloads and RBSP trailing bits syntax

#### 7.3.2.1 Sequence parameter set RBSP syntax

|  |  |
| --- | --- |
| seq\_parameter\_set\_rbsp( ) { | Descriptor |
| **profile\_idc** | u(8) |
| **reserved\_zero\_8bits** /\* equal to 0 \***/** | u(8) |
| **level\_idc** | u(8) |
| **seq\_parameter\_set\_id** | ue(v) |
| **chroma\_format\_idc** | ue(v) |
| ... |  |
| **sample\_adaptive\_offset\_enabled\_flag** | u(1) |
| **adaptive\_loop\_filter\_enabled\_flag** | u(1) |
| if( adaptive\_loop\_filter\_enabled\_flag ) |  |
| **alf\_coef\_in\_slice\_flag** | u(1) |
| if ( pcm\_enabled\_flag ) |  |
| **pcm\_loop\_filter\_disable\_flag** | u(1) |
| **non\_square\_intra\_enabled\_flag** | u(1) |
| **temporal\_id\_nesting\_flag** | u(1) |
| if( log2\_min\_coding\_block\_size\_minus3 = = 0 ) |  |
| **inter\_4x4\_enabled\_flag** | u(1) |
| ... |  |
| rbsp\_trailing\_bits( ) |  |
| } |  |
|  |  |
|  |  |

### 7.3.6 Coding unit syntax

|  |  |
| --- | --- |
| coding\_unit( x0, y0, log2CbSize ) { | Descriptor |
| CurrCbAddrTS = MinCbAddrZS[ x0 >> Log2MinCbSize ][ y0 >> Log2MinCbSize ] |  |
| nsIntraFlag = non\_square\_intra\_enabled\_flag && log2CbSize < 6 |  |
| if( slice\_type != I ) |  |
| **skip\_flag[** x0 **][** y0 **]** | ae(v) |
| if( skip\_flag[ x0 ][ y0 ] ) |  |
| prediction\_unit( x0, y0 , log2CbSize ) |  |
| else if( slice\_type != I | | log2CbSize = = Log2MinCbSize | | nsIntraFlag ) { |  |
| if( slice\_type != I ) |  |
| **pred\_mode\_flag** | ae(v) |
| if( PredMode != MODE\_INTRA | | log2CbSize = = Log2MinCbSize | | nsIntraFlag ) |  |
| **part\_mode** | ae(v) |
| x1 = x0 + ( ( 1 << log2CbSize ) >> 1 ) |  |
| y1 = y0 + ( ( 1 << log2CbSize ) >> 1 ) |  |
| x2 = x1 − ( ( 1 << log2CbSize ) >> 2 ) |  |
| y2 = y1 − ( ( 1 << log2CbSize ) >> 2 ) |  |
| x3 = x1 + ( ( 1 << log2CbSize ) >> 2 ) |  |
| y3 = y1 + ( ( 1 << log2CbSize ) >> 2 ) |  |
| if( PartMode == PART\_2Nx2N ) { |  |
| prediction\_unit( x0, y0 , log2CbSize ) |  |
| } else if( PartMode == PART\_2NxN ) { |  |
| prediction\_unit( x0, y0 , log2CbSize ) |  |
| prediction\_unit( x0, y1 , log2CbSize ) |  |
| } else if( PartMode == PART\_Nx2N ) { |  |
| prediction\_unit( x0, y0 , log2CbSize ) |  |
| prediction\_unit( x1, y0 , log2CbSize ) |  |
| } else if( PartMode == PART\_2NxnU ) { |  |
| prediction\_unit( x0, y0 , log2CbSize ) |  |
| prediction\_unit( x0, y2 , log2CbSize ) |  |
| } else if( PartMode == PART\_2NxnD ) { |  |
| prediction\_unit( x0, y0 , log2CbSize ) |  |
| prediction\_unit( x0, y3 , log2CbSize ) |  |
| } else if( PartMode == PART\_nLx2N ) { |  |
| prediction\_unit( x0, y0 , log2CbSize ) |  |
| prediction\_unit( x2, y0 , log2CbSize ) |  |
| } else if( PartMode == PART\_nRx2N ) { |  |
| prediction\_unit( x0, y0 , log2CbSize ) |  |
| prediction\_unit( x3, y0 , log2CbSize ) |  |
| } else { /\* PART\_NxN \*/ |  |
| prediction\_unit( x0, y0 , log2CbSize ) |  |
| prediction\_unit( x1, y0 , log2CbSize ) |  |
| prediction\_unit( x0, y1 , log2CbSize ) |  |
| prediction\_unit( x1, y1 , log2CbSize ) |  |
| } |  |
| if( !pcm\_flag ) { |  |
| transform\_tree( x0, y0, x0, y0, log2CbSize, log2CbSize, log2CbSize, 0, 0 ) |  |
| } |  |
| } |  |
| } |  |

### 7.3.7 Prediction unit syntax

|  |  |
| --- | --- |
| prediction\_unit( x0, y0, log2CbSize ) { | Descriptor |
| if( skip\_flag[ x0 ][ y0 ] ) { |  |
| if( MaxNumMergeCand > 1 ) |  |
| **merge\_idx[** x0 **][** y0 **]** | ae(v) |
| } else if( PredMode = = MODE\_INTRA ) { |  |
| if( PartMode = = PART\_2Nx2N && pcm\_enabled\_flag &&  log2CbSize >= Log2MinIPCMCUSize &&  log2CbSize <= Log2MaxIPCMCUSize ) |  |
| **pcm\_flag** | ae(v) |
| if( pcm\_flag ) { |  |
| ... |  |
| } else { |  |
| **prev\_intra\_luma\_pred\_flag[** x0 **][** y0 **]** | ae(v) |
| if( prev\_intra\_luma\_pred\_flag[ x0 ][ y0 ] ) |  |
| **mpm\_idx[** x0 **][** y0 **]** | ae(v) |
| else |  |
| **rem\_intra\_luma\_pred\_mode[** x0 **][**y0 **]** | ae(v) |
| p0 = ( ( 1 << log2CbSize ) >> 1 ) |  |
| xOffset = ( (x0 % (p0 << 1) !=0 ) \* p0 |  |
| yOffset = ( (y0 % (p0 << 1) !=0 ) \* p0 |  |
| bChroma = (PartMode == PART\_NxN) ? (xOffset && yOffset) : (xOffset | | yOffset) |  |
| if ( PartMode  == PART\_2Nx2N | | bChroma ) |  |
| **intra\_chroma\_pred\_mode**[ x0 – xOffset ][ y0 – yOffset ] | ae(v) |
| ~~SignalledAsChromaDC =   ( chroma\_pred\_from\_luma\_enabled\_flag ?~~  ~~intra\_chroma\_pred\_mode[ x0 ][ y0 ] = = 3 :~~  ~~intra\_chroma\_pred\_mode[ x0 ][ y0 ] = = 2 )~~ |  |
| } |  |
| } else { /\* MODE\_INTER \*/ |  |
| **merge\_flag[** x0 **][** y0 **]** | ae(v) |
| .... |  |
| } |  |
| } |  |

Note: Text highlighted in green is related to a bug fix, which is also reported in JCTVC-I0302.

### 7.3.8 Transform tree syntax

|  |  |
| --- | --- |
| transform\_tree( x0, y0, xC, yC, log2CbSize, log2TrafoWidth, log2TrafoHeight, trafoDepth, blkIdx ) { | Descriptor |
| if( trafoDepth = = 0 && IntraSplitFlag = = 0 && PredMode != MODE\_INTRA &&   !(PartMode = = PART\_2Nx2N && merge\_flag[x0][y0]) ) |  |
| **no\_residual\_data\_flag** | ae(v) |
| if( !no\_residual\_data\_flag ) { |  |
| log2TrafoSize = ( log2TrafoWidth + log2TrafoHeight ) >> 1 |  |
| intraNsqtFlag = ( PredMode == MODE\_INTRA && (PartMode == PART\_2NxN | | PartMode == PART\_Nx2N) ) |  |
| intraSplitFlag = ( IntraSplitFlag && trafoDepth = = 0 ? 1 : 0 ) |  |
| interSplitFlag = ( max\_transform\_hierarchy\_depth\_inter = =0 &&  PredMode = = MODE\_INTER && PartMode != PART\_2Nx2N &&  trafoDepth = = 0 ) |  |
| maxDepth = ( PredMode = = MODE\_INTRA ?   max\_transform\_hierarchy\_depth\_intra + IntraSplitFlag :   max\_transform\_hierarchy\_depth\_inter + InterSplitFlag ) |  |
| xBase = x0 − ( x0 & ( 1 << log2TrafoWidth ) ) |  |
| yBase = y0 − ( y0 & ( 1 << log2TrafoHeight ) ) |  |
| if( log2TrafoSize <= Log2MaxTrafoSize &&   log2TrafoSize > Log2MinTrafoSize &&  trafoDepth < maxDepth && !intraSplitFlag && !interSplitFlag && !intraNsqtFlag) |  |
| **split\_transform\_flag**[ x0 ][ y0 ][ trafoDepth ] | ae(v) |
| if( log2TrafoSize <= Log2MaxTrafoSize ) { |  |
| firstChromaCbf = ( log2TrafoSize = = Log2MaxTrafoSize | |  trafoDepth = = 0 ) ? 1 : 0 |  |
| if( firstChromaCbf | | (log2TrafoSize > Log2MinTrafoSize && !intraNsqtFlag)) { |  |
| if( firstChromaCbf | | cbf\_cb[ xBase ][ yBase ][ trafoDepth − 1 ] ) { |  |
| ... |  |
| } |  |
| if( firstChromaCbf | | cbf\_cr[ xBase ][ yBase ][ trafoDepth − 1 ] ) { |  |
| ... |  |
| } |  |
| } |  |
| } |  |
| if( split\_transform\_flag[ x0 ][ y0 ][ trafoDepth ] ) { |  |
| if( InterTUSplitDirection = = 2 ) { |  |
| ... |  |
| } else { |  |
| ... |  |
| } |  |
| transform\_tree( x0, y0, x0, y0, log2CbSize, log2TrafoWidth − 1, log2TrafoHeight − 1,   trafoDepth + 1, 0 ) |  |
| transform\_tree( x1, y1, x0, y0, log2CbSize, log2TrafoWidth − 1, log2TrafoHeight − 1,   trafoDepth + 1, 1 ) |  |
| transform\_tree( x2, y2, x0, y0, log2CbSize, log2TrafoWidth − 1, log2TrafoHeight − 1,   trafoDepth + 1, 2 ) |  |
| transform\_tree( x3, y3, x0, y0, log2CbSize, log2TrafoWidth − 1, log2TrafoHeight − 1,   trafoDepth + 1, 3 ) |  |
| } else { |  |
| ... |  |
| transform\_unit (x0, y0, xC, yC, log2TrafoWidth, log2TrafoHeight, trafoDepth, blkIdx) |  |
| } |  |
| } |  |
| } |  |

## 7.4 Semantics

**non\_square\_intra\_enabled\_flag** equal to 1 specifies that non-square intra prediction partitions may be used in treeblocks; non\_square\_intra\_enabled\_flag equal to 0 specifies that non-square intra prediction partitions can not be used in treeblocks.

Table 7‑10 ‑ Name association to prediction mode and partitioning type

|  |  |  |  |
| --- | --- | --- | --- |
| **PredMode** | **part\_mode** | **IntraSplitFlag** | **PartMode** |
| MODE\_INTRA | 0 | 0 | PART\_2Nx2N |
| 1 | 1 | PART\_2NxN |
| 2 | 1 | PART\_Nx2N |
| 3 | 1 | PART\_NxN |
| MODE\_INTER | 0 | 0 | PART\_2Nx2N |
| 1 | 0 | PART\_2NxN |
| 2 | 0 | PART\_Nx2N |
| 3 | 0 | PART\_NxN |
| 4 | 0 | PART\_2NxnU |
| 5 | 0 | PART\_2NxnD |
| 6 | 0 | PART\_nLx2N |
| 7 | 0 | PART\_nRx2N |

**7.4.8** **Transform tree semantics**

The variable InterTUSplitDirection defines how a transform block is split into four blocks with smaller horizontal or vertical size for the purpose of transform coding. The blocks are half horizontal and vertical size when ~~PredMode is equal to MODE\_INTRA or~~ InterTUSplitDirection is equal to 2, full horizontal and quarter vertical size when InterTUSplitDirection is equal to 0, quarter horizontal and full vertical size when InterTUSplitDirection is equal to 1. InterTUSplitDirection is specified as follows.

[Note: it is recommended to replace “InterTUSplitDirection” by “TUSplitDirection” in related text. (7.3.8, 7.4.8, 8.5.3 and 8.7.1)]

# Decoding process

## Decoding process for coding units coded in intra prediction mode

Inputs to this process are:

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

– a variable log2CbSize specifying the size of the current coding unit.

Output of this process is:

– a modified reconstructed picture before deblocking filtering.

A variable nS is set equal to ( 1 << log2CbSize ).

Depending on pcm\_flag and IntraSplitFlag and PartMode, the decoding process for luma samples is specified as follows.

– If pcm\_flag is equal to 1, the reconstucted picture is modified as follows:

recSamplesL[ xB + i, yB + j ] =   
 pcm\_sample\_luma[ ( nS \* j ) + i ] << ( BitDepthY – PCMBitDepthY ), with i, j = 0..nS-1 (8‑16)

– Otherwise (pcm\_flag is equal to 0), if IntraSplitFlag is equal to 0, the following ordered steps apply:

1. The derivation process for the intra prediction mode as specified in subclause 8.3.1 is invoked with the luma location ( xB, yB ) and the variable log2PUSize set equal to log2CbSize as well as IntraPredMode that is previously (in decoding order) derived for adjacent coding units as the input and the output is the variable IntraPredMode[ xB ][ yB ].
2. The decoding process for intra blocks as specified in subclause 8.3.3 is invoked with the luma location ( xB, yB ), the variable log2TrafoSize set equal to log2CbSize, the variable trafoDepth set equal to 0, the luma intra prediction mode IntraPredMode[ xB ][ yB ] and the variable cIdx set equal to 0 as the inputs and the output is a modified reconstructed picture before deblocking filtering.

– Otherwise (pcm\_flag is equal to 0 and IntraSplitFlag is equal to 1), for the variable blkIdx (proceeding over the values 0..1 for PartMode PART\_2NxN and PART\_Nx2N,0..3 for PartMode PART\_NxN), the following ordered steps apply:

1. Depending on PartMode, the variable xBS is set equal to the values as follows

xB + ( nS  >> 1 ) \* ( blkIdx % 2 ), when PartMode is equal to PART\_NxN

xB, when PartMode is equal to PART\_2NxN

xB + ( nS >> 1) \*  blkIdx, when PartMode is equal to PART\_Nx2N

1. Depending on PartMode, the variable yBS is set equal to the values as follows

yB + ( nS >> 1 ) \* ( blkIdx / 2 ), when PartMode is equal to PART\_NxN

yB + ( nS >> 1 ) \*  blkIdx, when PartMode is equal to PART\_2NxN

yB, when PartMode is equal to PART\_Nx2N

1. The derivation process for the intra prediction mode as specified in subclause is invoked with the luma location ( xBS, yBS ) and the variables

* log2PUWidth set equal to log2CbSize, log2PUHeight set equal to log2CbSize-1 when PartMode is equal to PART\_2NxN and log2CbSize is greater than 3,
* log2PUWidth set equal to log2CbSize-1, log2PUHeight set equal to log2CbSize when PartMode is equal to PART\_Nx2N and log2CbSize is greater than 3,
* log2PUWidth and log2PUHeight set equal to log2CbSize-1, otherwise,

as well as IntraPredMode that is previously (in decoding order) derived for adjacent coding units as the input and the output is the variable IntraPredMode[ xBS ][ yBS ].

1. The decoding process for intra blocks as specified in subclause is invoked with the luma location ( xB, yB ), the variables

* log2TrafoWidth set equal to log2CbSize, log2TrafoHeight set equal to log2CbSize-1 when PartMode is equal to PART\_2NxN and log2CbSize is greater than 3,
* the variables log2TrafoWidth set equal to log2CbSize-1, log2TrafoHeight set equal to log2CbSize when PartMode is equal to PART\_Nx2N and log2CbSize is greater than 3,
* log2TrafoWidth and log2TrafoHeight set equal to log2CbSize – 1, otherwise,

the variable trafoDepth set equal to 1, the prediction partition mode PartMode, the luma intra prediction mode IntraPredMode[ xBS ][ yBS ] and the variable cIdx set equal to 0 as the inputs and the output is a modified reconstructed picture before deblocking filtering.

Depending on pcm\_flag, the decoding process for chroma samples is specified as follows:

...

### Derivation process for luma intra prediction mode

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 log2PUWidth specifying the width of the current prediction unit,

– a variable log2PUHeight specifying the height 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 IntraPredMode[ xB ][ yB ].

### 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.

Depending on PartMode, the chroma intra prediction mode IntraPredModeC is derived as follows:

– If PartMode is equal to PART\_2Nx2N or PART\_NxN, the chroma intra prediction mode IntraPredModeC is derived as specifed in or with intra\_chroma\_pred\_mode, IntraPredMode[ xB ][ yB ] and chroma\_pred\_from\_luma\_enabled\_flag as inputs.

Table 8‑2 – Specification of IntraPredModeC according to the values of intra\_chroma\_pred\_mode and IntraPredMode[ xB ][ yB ] when chroma\_pred\_from\_luma\_enabled\_flag is equal to 1 and PartMode is equal to PART\_2Nx2N or PART\_NxN

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **intra\_chroma\_pred\_mode** | **IntraPredMode[ xB ][ yB ]** | | | | |
| **0** | **26** | **10** | **1** | **X ( 0 <= X < 35 )** |
| 0 | 34 | 0 | 0 | 0 | 0 |
| 1 | 26 | 34 | 26 | 26 | 26 |
| 2 | 10 | 10 | 34 | 10 | 10 |
| 3 | 1 | 1 | 1 | 34 | 1 |
| 4 | LM | LM | LM | LM | LM |
| 5 | 0 | 26 | 10 | 1 | X |

Table 8‑3 – Specification of IntraPredModeC according to the values of intra\_chroma\_pred\_mode and IntraPredMode[ xB ][ yB ] when chroma\_pred\_from\_luma\_enabled\_flag is equal to 0 and PartMode is equal to PART\_2Nx2N or PART\_NxN

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **intra\_chroma\_pred\_mode** | **IntraPredMode[ xB ][ yB ]** | | | | |
| **0** | **26** | **10** | **1** | **X ( 0 <= X < 35 )** |
| 0 | 34 | 0 | 0 | 0 | 0 |
| 1 | 26 | 34 | 26 | 26 | 26 |
| 2 | 10 | 10 | 34 | 10 | 10 |
| 3 | 1 | 1 | 1 | 34 | 1 |
| 4 | 0 | 26 | 10 | 1 | X |

– Otherwise (PartMode is equal to PART\_2NxN or PART\_Nx2N), IntraPredModeC is derived as specified in Table 8-3.1.

Table 8‑3.1 – Specification of IntraPredModeC according to the values of intra\_chroma\_pred\_mode when PartMode is equal to PART\_2NxN or PART\_Nx2N

|  |  |  |
| --- | --- | --- |
| **intra\_chroma\_pred\_mode** | chroma\_pred\_from\_luma\_enabled\_flag  == 1 | chroma\_pred\_from\_luma\_enabled\_flag  == 0 |
| 0 | Mode0 | Mode0 |
| 1 | Mode1 | Mode1 |
| 2 | Mode2 | Mode2 |
| 3 | Mode3 | Mode3 |
| 4 | LM | IntraPredMode[ xB ][ yB ] |
| 5 | IntraPredMode[ xB ][ yB ] | n/a |

Mode0, Mode1, Mode2 and Mode3 in Table 8-3.1 are derived as follows.

1. Derive IntraPredMode[ xB+xOff ][ yB+xOff ].

if(PartMode == PART\_2NxN) {

xOff = 0;

yOff = nS>>1; }

else if(PartMode == PART\_ Nx2N) {

xOff = nS>>1;

yOff = 0; }

1. Sort mode array {IntraPredMode[ xB+xOff ][ yB+xOff ], 0, 26, 10, 1, 34}. The first four different modes that are different from IntraPredMode[ xB ][ yB ] are assigned to Mode0, Mode1, Mode2 and Mode3.

### Decoding process for intra blocks

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 log2TrafoWidth specifying the width of the current block,

– a variable log2TrafoHeight specifying the height of the current block,

– a variable trafoDepth specifying the hierarchy depth of the current block relative to the coding unit,

– a variable PartMode specifying prediction partition mode of the current coding unit,

– a variable intraPredMode specifying the intra prediction mode.

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

Output of this process is:

– a modified reconstructed picture before deblocking filtering.

Depending split\_transform\_flag[ xB ][ yB ][ trafoDepth ], log2TrafoWidth, log2TrafoHeight and PartMode, the following applies:

– If split\_transform\_flag[ xB ][ yB ][ trafoDepth ] is equal to 1, and PartMode is equal to PART\_2Nx2N or log2TrafoSize equals to 3 (8x8 block), the following ordered steps apply:

1. The variable xB1 is set equal to xB + ( ( 1 << log2TrafoSize ) >> 1 ).
2. The variable yB1 is set equal to yB + ( ( 1 << log2TrafoSize ) >> 1 ).
3. The decoding process for intra blocks as specified in this subclause is invoked with the location ( xB, yB ), the variable log2TrafoSize set equal to log2TrafoSize − 1, the variable trafoDepth set equal to trafoDepth + 1, the intra prediction mode intraPredMode, and the variable cIdx as the inputs and the output is a modified reconstructed picture before deblocking filtering.
4. The decoding process for intra blocks as specified in this subclause is invoked with the location ( xB1, yB ), the variable log2TrafoSize set equal to log2TrafoSize − 1, the variable trafoDepth set equal to trafoDepth + 1, the intra prediction mode intraPredMode, and the variable cIdx as the inputs and the output is a modified reconstructed picture before deblocking filtering.
5. The decoding process for intra blocks as specified in this subclause is invoked with the location ( xB, yB1 ), the variable log2TrafoSize set equal to log2TrafoSize − 1, the variable trafoDepth set equal to trafoDepth + 1, the intra prediction mode intraPredMode, and the variable cIdx as the inputs and the output is a modified reconstructed picture before deblocking filtering.
6. The decoding process for intra blocks as specified in this subclause is invoked with the location ( xB1, yB1 ), the variable log2TrafoSize set equal to log2TrafoSize − 1, the variable trafoDepth set equal to trafoDepth + 1, the intra prediction mode intraPredMode, and the variable cIdx as the inputs and the output is a modified reconstructed picture before deblocking filtering.

– Otherwise, if split\_transform\_flag[ xB ][ yB ][ trafoDepth ] is equal to 1, and log2TrafoWidth is not equal to log2TrafoHeight, the following ordered steps apply:

1. If log2TrafoWidth is greater than log2TrafoHeight, the variables xB1, xB2, xB3 are set equal to xB. Otherwise, the variable xB1 is set equal to xB + ( ( 1 << log2TrafoWidth ) >> 2 ), the variable xB2 is set equal to xB1 + ( ( 1 << log2TrafoWidth ) >> 2 ), the variable xB3 is set equal to xB2 + ( ( 1 << log2TrafoWidth ) >> 2 ).
2. If log2TrafoWidth is greater than log2TrafoHeight, the variable yB1 is set equal to yB + ( ( 1 << log2TrafoHeight ) >> 2 ), the variable yB2 is set equal to yB1 + ( ( 1 << log2TrafoHeight ) >> 2 ), the variable yB3 is set equal to yB2 + ( ( 1 << log2TrafoHeight ) >> 2 ). Otherwise, the variables yB1, yB2, yB3 are set equal to yB.
3. If log2TrafoWidth is greater than log2TrafoHeight, the variable log2TrafoHeight is set equal to log2TrafoWidth - 2. Otherwise, the variable log2TrafoWidth is set equal to log2TrafoHeight - 2.
4. The decoding process for intra blocks as specified in this subclause is invoked with the location ( xB, yB ), the variables log2TrafoWidth and log2TrafoHeight, the variable trafoDepth set equal to trafoDepth + 1, the intra prediction mode intraPredMode, and the variable cIdx as the inputs and the output is a modified reconstructed picture before deblocking filtering.
5. The decoding process for intra blocks as specified in this subclause is invoked with the location ( xB1, yB1 ), the variables log2TrafoWidth and log2TrafoHeight, the variable trafoDepth set equal to trafoDepth + 1, the intra prediction mode intraPredMode, and the variable cIdx as the inputs and the output is a modified reconstructed picture before deblocking filtering.
6. The decoding process for intra blocks as specified in this subclause is invoked with the location ( xB2, yB2 ), the variables log2TrafoWidth and log2TrafoHeight, the variable trafoDepth set equal to trafoDepth + 1, the intra prediction mode intraPredMode, and the variable cIdx as the inputs and the output is a modified reconstructed picture before deblocking filtering.
7. The decoding process for intra blocks as specified in this subclause is invoked with the location ( xB3, yB3 ), the variables log2TrafoWidth and log2TrafoHeight, the variable trafoDepth set equal to trafoDepth + 1, the intra prediction mode intraPredMode, and the variable cIdx as the inputs and the output is a modified reconstructed picture before deblocking filtering.

* Otherwise (split\_transform\_flag[ xB ][ yB ][ trafoDepth ] is equal to 0), the following ordered steps apply:

1. The variable nW is set equal to 1 << log2TrafoWidth, the variable nH is set equal to 1 << log2TrafoHeight.
2. The intra sample prediction process as specified in subclause 8.4.3.1 is invoked with the location ( xB, yB ), the intra prediction mode intraPredMode, the prediction width nW, the prediction height nH and the variable cIdx as the inputs and the output is a (nW)x(nH) array predSamples.
3. The scaling and transformation process as specified in subclause 8.6.1 is invoked with the location ( xB, yB ), the variable trafoDepth, the variable cIdx, and the transform width trafoWidth set equal to nW, and the transform height trafoHeight set equal to nH as the inputs and the output is a (nW)x(nH) array resSamples.
4. The residual signal accumulation process as specified in subclause XXX is invoked with the variable arrayWidth set equal to nW, the variable arrayHeight set equal to nH, the (nW)x(nH) array predSamples, and the (nW)x(nH) array resSamples as the inputs and the output is a (nW)x(nH) array recSamples.
5. The picture reconstruction process for a component before deblocking filtering as specified in subclause XXX is invoked with the location ( xB, yB ), the variable arrayWidth set equal to nW, the variable arrayHeight set equal to nH, the variable cIdx set equal to 0, and the (nW)x(nH) array recSamples as the inputs and the output is a modified reconstructed picture before deblocking filtering.

#### 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 intra prediction mode,

– a variable nW specifying the prediction width,

– a variable nH specifying the prediction height,

– 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..nW-1 and y = 0...nH-1.

The variable nS is set as (nW+nH)>>1, the nS\*4+1 neighbouring samples p[ x, y ] that are constructed samples prior to the deblocking filter process, with x = -1, y = -1..nS\*2-1 and x = 0..nS\*2-1, y=-1, are derived as follows.

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

xBN = xB + x  (8‑29)

yBN = yB +y  (8‑30)

– Each sample p[ x, y ] with x = -1, y= -1..nS\*2-1  and x = 0..nS\*2-1, y = -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”
  1. the coding unit covering ( xBN, yBN ) is not available
  2. 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 ].

When chroma\_pred\_from\_luma\_enabled\_flag is equal to 1 and cIdx is equal to 0, the nS\*4+1 neighbouring samples PLM[ x, y ] that are constructed luma samples for Intra\_FromLuma prediction mode, with x = -1, y = -1..nS\*2-1 and x = 0..nS\*2-1, y=-1, are derived as following ordered steps:

1. If the sample p[ x, y ] is marked as “not available for intra prediction”, the sample PLM[ x, y ] is marked as “not available for intra prediction”, otherwise, the sample PLM[ x, y ] is marked as “available for intra prediction”, with x = -1, y = -1..nS\*2-1 and x = 0..nS\*2-1, y=-1.
2. For x = -1, y= 0..nS\*2-1 , if the sample PLM[ x, y ] is marked as “available for intra prediction”, the sample location at the location ( xB + x – 1, yB + y ) inside the treeblock tbAddrN is assigned to PLM[ x, y ].
3. For x = -1..nS\*2-1, y=-1 , if the sample PLM[ x, y ] is marked as “available for intra prediction”, the sample location at the location ( xB + x , yB + y ) inside the treeblock tbAddrN is assigned to PLM[ x, y ].

When at least one sample p[ x, y ] with x = -1, y = -1..nS\*2 1 and x = 0..nS\*2 1, y = -1 is marked as “not available for intra prediction,” the reference sample substitution process for intra sample prediction in subclause 8.3.3.1.1 is invoked with the samples p[ x, y ] with x = -1, y = -1..nS\*2 1 and x = 0..nS\*2 1, y = -1 as input and the modified samples p[ x, y ] with x = -1, y = -1..nS\*2 1 and x = 0..nS\*2 1, y = -1 as output.

When chroma\_pred\_from\_luma\_enabled\_flag is equal to 1, cIdx is equal to 0, and at least one sample PLM[ x, y ] with x = -1, y = -1..nS\*2 1 and x = 0..nS\*2 1, y = -1 is marked as “not available for intra prediction,” the reference sample substitution process for intra sample prediction in subclause 8.3.3.1.1 is invoked with the samples PLM[ x, y ] with x = -1, y = -1..nS\*2 1 and x = 0..nS\*2 1, y = -1 as input and the modified samples PLM[ x, y ] with x = -1, y = -1..nS\*2 1 and x = 0..nS\*2 1, y = -1 as output.

Depending on intraPredMode, the following ordered steps apply:

1. When cIdx is equal to 0, filtering process of neighbouring samples specified in 8.4.3.1.2 is invoked with the sample array p and the prediction width nW and height nH as the inputs and the output is reassigned to the sample array p.
2. Intra sample prediction process according to intraPredMode applies as follows:
   1. One of the intra prediction modes specified in subclause 8.4.3.1.3 to 8.4.3.1.8 is invoked with the sample location ( xB, yB ), the sample array p, the prediction width nW and height nH and the chroma component index cIdx as the inputs and the output is the predicted sample array predSamples according to intraPredMode.

##### Reference sample substitution process for intra sample prediction

##### Filtering process of neighbouring samples

Inputs to this process are:

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

– a variable nW specifying the prediction width,

Output of this process is:

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

The value of intraFilterType[ nW ][ IntraPredMode ] is derived as the following ordered steps:

1. The variable minDistVerHor is derived as:

minDistVerHor = Min( Abs( intraPredMode − 26 ), Abs( intraPredMode − 10 ) ) (8‑31)

1. If intraPredMode is equal to Intra\_DC, intraFilterType[ nW ][ IntraPredMode ] is set equal to 0,
2. Otherwise, if minDistVerHor is larger than intraHorVerDistThresh[ nW ], intraFilterType[ nW ][ IntraPredMode ] is set to 1, otherwise it is set to 0. intraHorVerDistThres[ nW ] is specified in .

Table 8‑4 – Specification of intraHorVerDistThres[ nW ] for various prediction unit sizes

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **nW = 4** | **nW = 8** | **nW = 16** | **nW = 32** | **nW = 64** |
| **intraHorVerDistThresh[ nW ]** | 10 | 7 | 1 | 0 | 10 |

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

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

pF[ -1, nS\*2-1 ] = p[ -1, nS\*2-1 ] (8‑32)

pF[ nS\*2-1, -1 ] = p[ nS\*2-1, -1 ] (8‑33)

pF[ -1, y ] = ( p[ -1, y+1 ] + 2\*p[ -1, y ] + p[ -1, y-1 ] + 2 ) >> 2 for y = nS\*2-2..0 (8‑34)

pF[ -1, -1] = ( p[ -1, 0 ] + 2\*p[ -1, -1] + p[ 0, -1 ] + 2) >> 2 (8‑35)

pF[ x, -1 ] = ( p[ x-1, -1 ] + 2\*p[ x, -1 ] + p[ x+1, -1 ] + 2 ) >> 2 for x = 0..nS\*2-2 (8‑36)

##### Specification Intra\_Angular (26) prediction mode

Inputs to this process are:

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

– a variable nW specifying the prediction width,

– a variable nH specifying the prediction height,

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

Output of this process is:

– predicted samples predSamples[ x, y ], with x =0..nW-1 and y = 0..nH-1.

This intra prediction mode is invoked when intraPredMode is equal to 0.

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

– If cIdx is equal to 0,

predSamples[ x, y ] = p[ x, -1 ], with x =0..nW-1 and y = 0..nH-1 (8‑37)

predSamples[ x, y ] = Clip1Y( p[ x, -1 ] + ( ( d[ y ] + ( d[ y ] < 0 ? 1 : 0 ) )>> 1 ) ), with x = 0, y = 0..nH-1

where d[ y ] = p[ -1, y ] – p[ -1, -1 ] (8‑37.1)

– Otherwise,

predSamples[ x, y ] = p[ x, -1 ], with x =0..nW-1 and y = 0..nH-1 (8‑38)

##### Specification of Intra\_Angular (10) prediction mode

Inputs to this process are:

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

– a variable nW specifying the prediction width,

– a variable nH specifying the prediction height,

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

Output of this process is:

– predicted samples predSamples[ x, y ], with x =0..nW-1 and y = 0..nH-1.

This intra prediction mode is invoked when intraPredMode is equal to 1.

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

– If cIdx is equal to 0,

predSamples[ x, y ] = p[ -1, y ], with x =0..nW-1 and y = 0..nH-1 (8‑39)

predSamples[ x, y ] = Clip1Y Clip1Y( p[ −1, y ] + ( ( p[ x, −1 ] – p[ −1, −1 ] ) >> 1 ), with x =0..nW-1, y = 0

– Otherwise,

predSamples[ x, y ] = p[ -1, y ], with x =0..nW-1 and y = 0..nH-1 (8‑40)

##### Specification of Intra\_DC prediction mode

Inputs to this process are:

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

– a variable nW specifying the prediction width,

– a variable nH specifying the prediction height,

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

Output of this process is:

– predicted samples predSamples[ x, y ], with x =0..nW-1 and y = 0..nH-1.

This intra prediction mode is invoked when intraPredMode is equal to 1.

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

1. A variable DCVal is derived as:

If nW is equal to nH,

DCVal = , with x, y = 0..nS-1 (8‑41)  
where k=log2(nS)

Otherwise, if nW is greater than nH

DCVal = , with x = 0.. nW-1 (8‑41.1)  
where k=log2(nW)

Otherwise, if nW is less than nH

DCVal = , with y = 0.. nH-1 (8‑41.2)  
where k=log2(nH)

1. Depending on the chroma component index cIdx, the following applies.

* If cIdx is equal to 0, the following applies.

predSamples[ 0, 0 ] = ( 1\*p[ -1, 0 ] + 2\*DCVal + 1\*p[ 0, -1 ] + 2 ) >> 2 (8‑42)  
predSamples[ x, 0 ] = ( 1\*p[ x, -1 ] + 3\*DCVal + 2 ) >> 2, with x = 1..nW-1 (8‑43)  
predSamples[ 0, y ] = ( 1\*p[ -1, y ] + 3\*DCVal + 2 ) >> 2, with y = 1..nH-1 (8‑44)  
predSamples[ x, y ] = DCVal, with x = 1.. nW-1, y = 1.. nH-1 (8‑45)

* Otherwise, the prediction samples predSamples[ x, y ] are derived as

predSamples[ x, y ] = DCVal, with x = 0.. nW-1, y = 0.. nH-1 (8‑46)

##### Specification of Intra\_Angular (2..9, 11..25, 27..34) prediction mode

Inputs to this process are:

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

– a variable nW specifying the prediction width,

– a variable nH specifying the prediction height.

Output of this process is:

– predicted samples predSamples[ x, y ], with x =0..nW-1 and y = 0..nH-1.

This intra prediction mode is invoked when intraPredMode is in the range of 2..9, 11..25 and 27..34.

illustrates the total 34 intra angles and specifies the mapping table between intraPredMode and the angle parameter intraPredAngle.



Figure ‑ – Intra prediction angle definition (informative)

Table 8‑5 – Specification of intraPredAngle

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

further specifies the mapping table between intraPredMode and the inverse angle parameter invAngle.

Table 8‑6 – Specification of invAngle

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **intraPredMode** | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| **invAngle** | −256 | −315 | −390 | −482 | −630 | −910 | −1638 | −4096 |
| **intraPredMode** | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 |
| **invAngle** | −315 | −390 | −482 | −630 | −910 | −1638 | −4096 | - |

The reference pixel array refMain[ x ], with x= -Max(nW,nH)..2\*nS is specified as follows.

– If intraPredOrder is equal or greater than 18,

refMain[ x ] = p[ -1+x, -1 ], with x=0..nW (8‑47)

* If intraPredAngle is less than 0,

refMain[ x ] = p[ -1, -1+( ( x\*invAngle+128 )>>8 ) ], with x=( nH\*intraPredAngle ) >>5..-1 (8‑48)

* Otherwise,

refMain[ x ] = p[ -1+x, -1 ], with x=nW+1..2\*nS (8‑49)

Otherwise,

refMain[ x ] = p[ -1, -1+x ], with x=0..nH (8‑50)

* If intraPredAngle is less than 0,

refMain[ x ] = p[ -1+( ( x\*invAngle+128 )>>8 ), -1 ], with x=( nW\*intraPredAngle ) >>5..-1 (8‑51)

* Otherwise,

refMain[ x ] = p[ -1, -1+x ], with x=nH+1..2\*nS (8‑52)

The values of the prediction samples predSamples[ x, y ], with x =0..nW-1 and y = 0..nH-1 are derived by the following procedures.

– The index variable iIdx and the multiplication factor iFact are derived by

iIdx = ( ( y + 1 )\*intraPredAngle ) >> 5 (8‑53)

iFact = ( ( y + 1 )\*intraPredAngle ) && 31 (8‑54)

– Depending on the value of iFact, the following applies.

* If iFact is not equal to 0, the value of the prediction samples predSamples[ x, y ] is derived by

predSamples[ x, y ] = ( ( 32 – iFact )\*refMain[ x+iIdx+1 ] + iFact\*refMain[ x+iIdx+2] + 16 ) >> 5 (8‑55)

* Otherwise, the value of the prediction samples predSamples[ x, y ] is derived by

predSamples[ x, y ] = refMain[ x+iIdx+1 ] (8‑56)

If intraPredMode is less than 18, the value of prediction samples predSamples[ x, y ] is swapped by that of predSamples[ y, x ] for y=0..nS−2, x=y+1..nS−1 when nW is equal to nH, and for y=0..nW-1, x=0..nH−1 when nW is not equal to nH.

##### Specification of Intra\_Planar prediction mode

Inputs to this process are:

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

– a variable nW specifying the prediction width,

– a variable nH specifying the prediction height.

Output of this process is:

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

This intra prediction mode is invoked when intraPredMode is equal to 34.

The values of the prediction samples predSamples[ x, y ], with x = 0.. nW-1, y = 0.. nH-1 are derived by

predSamples[ x, y ] = (  
 ( ( ( nW – 1 – x ) \* p[ -1, y ] + ( x + 1 ) \* p[ nW, -1 ] + nW/2 ) >> m ) +   
 ( ( ( nH – 1 – y ) \* p[ x ,-1 ] + ( y + 1 ) \* p[ -1, nH ] + nH/ 2 ) >> n ) +1 ) >> 1 (8‑57)  
with x = 0.. nW-1, y = 0.. nH-1 where m = log2( nW ), n = log2( nH )

##### Specification of Intra\_FromLuma prediction mode

## 8.6.4 Transformation process for scaled transform coefficients

Inputs of this process are:

– a variable nW specifying the width of the current transform unit,

– a variable nH specifying the height of the current transform unit,

– a (nW)x(nH) array d of scaled transform coefficients with elements dij.

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

Output of this process is residual samples as a (nW)x(nH) array r with elements rij.

Depending on PredMode and IntraPredMode, the following applies:

– If PredMode is equal to MODE\_INTRA and cIdx is equal to 0, if nW is equal to 4, the variables horizTrType is specified as with IntraPredMode as input and if nH is equal to 4, the variables vertTrType is specified as with IntraPredMode as input.

– Otherwise, the variables horizTrType and vertTrType are set equal to 0.

#### 9.2.2.7 Binarization process for part\_mode

Input to this process is a request for a binarization for the syntax element part\_mode and a variable cLog2CbSize specifying the current CU size.

Output of this process is the binarization of the syntax element.

The binarization for part\_mode is given by Table 9-34 depending on PredMode, cLog2CbSize and

inter\_4x4\_enabled\_flag.

Table 9‑34 – Binarization for part\_mode

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **PredMode** | **Value of part\_mode** | **PartMode** | **Bin string** | | |
| cLog2CbSize >  Log2MinCUSize | cLog2CbSize = = Log2MinCUSize | |
| cLog2CbSize = = 3 &&  !inter\_4x4\_enabled\_flag | cLog2CbSize > 3 | |  inter\_4x4\_enabled\_flag |
| MODE\_INTRA | 0 | PART\_2Nx2N | 1 | 1 | 1 |
| 1 | PART\_2NxN | 01 | 001 | 001 |
| 2 | PART\_Nx2N | 00 | 000 | 000 |
| 3 | PART\_NxN | - | 01 | 01 |
| MODE\_INTER | 0 | PART\_2Nx2N | 1 | 1 | 1 |
| 1 | PART\_2NxN | 011 | 01 | 01 |
| 2 | PART\_Nx2N | 001 | 00 | 001 |
| 3 | PART\_NxN | - | - | 000 |
| 4 | PART\_2NxnU | 0100 | - | - |
| 5 | PART\_2NxnD | 0101 | - | - |
| 6 | PART\_nLx2N | 0000 | - | - |
| 7 | PART\_nRx2N | 0001 | - | - |