|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11**  11th Meeting: Shanghai, CN, 10–19 Oct. 2012 | Document: JCTVC-K0112 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **DQP Buffering Fix** | | |
| *Status:* | Input Document to JCT-VC | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Pieter Kapsenberg (Intel Corporation)  Wenhao Zhang (Intel Corporation)  CH7-427 5000 W Chandler Blvd Chandler AZ, 85226  Geert Van der Auwera, Rajan Joshi, Marta Karczewicz (Qualcomm Inc.)  5775 Morehouse Drive  San Diego, CA 92121 | Tel: Email:  Email: | +1 480 552 5481 pieter.kapsenberg@intel.com  geertv@qti.qualcomm.com |
| *Source:* | Intel Corp, Qualcomm Inc. | | |

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

# Abstract

This contribution claims that the adopted proposal JCTVC-I0219 – while being an important solution to enable CU-level processing – does not alleviate the worst case that would only be caused by a specially designed “evil” bitstream. Two solutions are proposed. The first solution stores QP per 8x8 region, while the second solution signals QP delta at the beginning of the coding unit depending on the no\_residual\_syntax\_flag for both inter- and intra-coded CUs.

# Introduction

The adopted proposal JCTVC-I0219 allows for CU-level processing. However the current design still places an unnecessary burden on hardware designs which must account for the worst case allowed by the standard. Consider the following scenario:

* CTB size is 64x64
* CU is not split (size is 64x64)
* CU is intra-coded (all boundaries require deblocking)
* diff\_cu\_qp\_delta\_depth = 0 (quantization group is 64x64)
* Fully split TU tree, 256 luma 4x4 partitions, 64 + 64 chroma partitions.
  + This could also happen implicitly with Log2MaxTrafoSize = 2
* Only last TU has cbf = 1, so only then is the QP value derived.

Since the QP value for the CU must be known to perform deblocking filtering, this operation must wait until the parsing process has traversed the whole tree. In the meantime the TU structure must be saved, requiring a 384 entry buffer that under normal circumstances will hardly ever be used. This results in wasted area in a hardware design.

# Proposed solution 1: Set QP per 8x8 region

By considering 8x8 QP regions within a CU that occur prior to the decoding of the QP delta to have a QP value equal to the predicted QP value, the situation above is avoided. Additionally, it better reflects the intention of QP-based deblocking of boundaries that were coded with no residuals.

This solution requires that QP values are stored per 8x8 region, as opposed to per-CU as in the current design. However the worst case QP storage for the current design is already 8x8, which occurs when the quantization group size is 8x8, so no extra storage is required by a Main-Profile compliant decoder. This also neatly matches the 8x8 grid used by the deblocking filter, and the line buffer structures.

In the diagram below the resulting process is illustrated. The purple TU is the first TU in Z-order within the CU to have a non-zero cfb, and the subsequent TUs are colored differently to illustrate the new QP value:



Figure - Proposed QP storage

The deblocking filter would only have to wait for at most the four luma TUs and the first chroma TU before being able to resume. Thus the TU buffering requirement has been reduced from 380 to 3.

# Proposed solution 2: Signal QP delta at beginning of CU

It is proposed to signal QP delta at the beginning of the CU. However, if there is no coded residual in the CU, the QP delta represents an overhead compared with the current HEVC syntax. Therefore, it is proposed to signal the no\_residual\_syntax\_flag in the CU syntax [1] both for inter- and intra-coded CUs, if QP delta signalling is enabled. QP delta is signalled only if the no\_residual\_syntax\_flag is equal to 0. As is the case currently in HEVC, the delta QP is signalled once per quantization group. In addition, if no\_residual\_syntax\_flag is equal to 1 for an intra-coded CU, then the signalling of the cbf flags for luma and chroma is disabled.

# Simulations results

The configuration used for the simulations is the CTC modified as follows:

* DQP depth = 1 (32x32 Quantization Groups)
* Max delta QP is 3.

Because of the large runtime required when dqp is enabled, simulations were run with only the first 10% of the frames of each sequence and Class A & B streams were omitted.

## Solution 1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **All Intra Main** | | | **All Intra HE10** | | |
|  | Y | U | V | Y | U | V |
| Class A |  |  |  |  |  |  |
| Class B |  |  |  |  |  |  |
| Class C | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% |
| Class D | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% |
| Class E | 0.0% | 0.0% | 0.1% | 0.0% | 0.1% | 0.0% |
| Class F | 0.0% | 0.1% | 0.0% | 0.0% | 0.1% | 0.0% |
| **Overall** |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| Enc Time[%] |  | | |  | | |
| Dec Time[%] |  | | |  | | |
|  |  |  |  |  |  |  |
|  | **Random Access Main** | | | **Random Access HE10** | | |
|  | Y | U | V | Y | U | V |
| Class A |  |  |  |  |  |  |
| Class B |  |  |  |  |  |  |
| Class C | -0.1% | -0.1% | -0.2% | 0.1% | 0.0% | -0.1% |
| Class D | 0.0% | -0.3% | -0.1% | 0.0% | 0.1% | 0.4% |
| Class E |  |  |  |  |  |  |
| Class F | -0.1% | -0.2% | -0.4% | -0.1% | -0.2% | -0.3% |
| **Overall** |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| Enc Time[%] |  | | |  | | |
| Dec Time[%] |  | | |  | | |
|  |  |  |  |  |  |  |
|  | **Low delay B Main** | | | **Low delay B HE10** | | |
|  | Y | U | V | Y | U | V |
| Class A |  |  |  |  |  |  |
| Class B |  |  |  |  |  |  |
| Class C | -0.1% | 0.1% | 0.1% | 0.0% | 0.1% | 0.1% |
| Class D | 0.1% | 0.0% | -0.3% | 0.0% | 0.1% | 0.2% |
| Class E | 0.0% | 0.3% | -0.3% | -0.3% | 0.1% | 0.4% |
| Class F | -0.1% | -0.1% | 0.0% | 0.0% | -0.6% | -0.6% |
| **Overall** |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| Enc Time[%] |  | | |  | | |
| Dec Time[%] |  | | |  | | |
|  |  |  |  |  |  |  |
|  | **Low delay P Main** | | | **Low delay P HE10** | | |
|  | Y | U | V | Y | U | V |
| Class A |  |  |  |  |  |  |
| Class B |  |  |  |  |  |  |
| Class C | 0.0% | 0.2% | 0.1% | 0.0% | 0.1% | -0.3% |
| Class D | 0.0% | 0.1% | -0.1% | 0.1% | 0.5% | 0.5% |
| Class E | -0.2% | -0.1% | -0.1% | -0.1% | -0.3% | -0.2% |
| Class F | 0.0% | -0.1% | 0.1% | 0.1% | -0.1% | -0.1% |
| **Overall** |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| Enc Time[%] |  | | |  | | |
| Dec Time[%] |  | | |  | | |

## Solution 2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **All Intra Main** | | | **All Intra HE10** | | |
|  | Y | U | V | Y | U | V |
| Class A |  |  |  |  |  |  |
| Class B |  |  |  |  |  |  |
| Class C | 0.13% | -0.07% | -0.12% | 0.11% | -0.10% | -0.01% |
| Class D | 0.14% | -0.06% | -0.08% | 0.15% | 0.02% | -0.09% |
| Class E | 0.45% | -0.15% | -0.03% | 0.39% | -0.36% | -0.18% |
| Class F | 0.23% | 0.43% | 0.35% | 0.21% | 0.54% | 0.21% |
| **Overall** |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| Enc Time[%] |  | | |  | | |
| Dec Time[%] |  | | |  | | |
|  |  |  |  |  |  |  |
|  | **Random Access Main** | | | **Random Access HE10** | | |
|  | Y | U | V | Y | U | V |
| Class A |  |  |  |  |  |  |
| Class B |  |  |  |  |  |  |
| Class C | 0.09% | -0.08% | -0.09% | 0.17% | 0.01% | 0.20% |
| Class D | 0.19% | -0.09% | 0.60% | -0.02% | 0.56% | 0.54% |
| Class E |  |  |  |  |  |  |
| Class F | 0.10% | -0.36% | -0.05% | 0.10% | 0.36% | 0.05% |
| **Overall** |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| Enc Time[%] |  | | |  | | |
| Dec Time[%] |  | | |  | | |
|  |  |  |  |  |  |  |
|  | **Low delay B Main** | | | **Low delay B HE10** | | |
|  | Y | U | V | Y | U | V |
| Class A |  |  |  |  |  |  |
| Class B |  |  |  |  |  |  |
| Class C | 0.01% | 0.24% | 0.19% | 0.09% | 0.23% | 0.58% |
| Class D | 0.16% | 0.26% | 0.29% | -0.01% | 0.06% | -0.67% |
| Class E | 0.19% | 0.36% | -0.37% | -0.12% | -0.66% | -1.41% |
| Class F | 0.37% | 0.18% | 0.11% | -0.03% | -0.04% | -0.43% |
| **Overall** |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| Enc Time[%] |  | | |  | | |
| Dec Time[%] |  | | |  | | |
|  |  |  |  |  |  |  |
|  | **Low delay P Main** | | | **Low delay P HE10** | | |
|  | Y | U | V | Y | U | V |
| Class A |  |  |  |  |  |  |
| Class B |  |  |  |  |  |  |
| Class C | 0.10% | 0.56% | 0.38% | 0.11% | 0.49% | 0.27% |
| Class D | -0.16% | -0.08% | 0.42% | 0.07% | 0.41% | -0.22% |
| Class E | 0.15% | -0.30% | -0.18% | 0.03% | -0.75% | -1.00% |
| Class F | 0.30% | 0.37% | -0.11% | 0.26% | 0.26% | 0.84% |
| **Overall** |  |  |  |  | #VALUE! | #VALUE! |
|  |  |  |  |  | #VALUE! | #VALUE! |
| Enc Time[%] |  | | |  | | |
| Dec Time[%] |  | | |  | | |

# Conclusion

Two solutions are proposed to alleviate the worst case QP buffering for deblocking. The first solution stores QP per 8x8 region, while the second solution signals QP delta at the beginning of the coding unit depending on the no\_residual\_syntax\_flag for both inter- and intra-coded CUs.

It is recommended to adopt one of the proposed solutions into the HEVC design.

# References

[1] B. Bross, W.-J. Han, J.-R. Ohm, G. J. Sullivan, T. Wiegand, “High efficiency video coding (HEVC) text specification draft 8,” 10th JCT-VC meeting, Stockholm, Sweden, July 2012, Doc. JCTVC-J1003\_d7

# Draft text changes for solution 1

The following DIS text changes are based on [1].

“**7.4.2.3 Picture parameter set RBSP semantics**

…

**pic\_init\_qp\_minus26** specifies the initial value minus 26 of SliceQPY for each slice. The initial value is modified at the slice layer when a non-zero value of slice\_qp\_delta is decoded, and is modified further when a non-zero value of cu\_qp\_delta\_abs is decoded at the transform~~coding~~ unit layer. The value of pic\_init\_qp\_minus26 shall be in the range of −(26 + QpBdOffsetY ) to +25, inclusive.

…”

**“7.4.5.1 General slice header semantics**

**...**

**slice\_address** specifies the address of the first coding tree block in the slice. The length of the slice\_address syntax element is Ceil( Log2( PicSizeInCtbsY ) ) bits. The value of slice\_address shall be in the range of 1 to PicSizeInCtbsY – 1, inclusive. When slice\_address is not present, it is inferred to be equal to 0.

The variable CtbAddrRS, specifying a coding tree block address in coding tree block raster scan order, is set equal to slice\_address. The variable CtbAddrTS, specifying a coding tree block address in coding tree block tile scan order, is set equal to CtbAddrRStoTS[ CtbAddrRS ].The variable CuQpDelta, specifying the difference between a luma quantization parameter for the quantization group ~~coding unit~~ containing cu\_qp\_delta\_abs and its prediction, is set equal to 0.

…

**slice\_qp\_delta** specifies the initial value of QPY to be used for the coding blocks in the slice until modified by the value of CuQpDelta in the transform~~coding~~ unit layer. The initial QPY quantization parameter for the slice is computed as

SliceQPY = 26 + pic\_init\_qp\_minus26 + slice\_qp\_delta

The value of slice\_qp\_delta shall be limited such that SliceQPY is in the range of −QpBdOffsetY to +51, inclusive.

…”

“**7.4.11 Transform unit semantics**

…

**cu\_qp\_delta\_abs** specifies the absolute value of the difference between a luma quantization parameter for the quantization group ~~coding unit~~ containing cu\_qp\_delta\_abs and its prediction.

**cu\_qp\_delta\_sign** specifies the sign of a CuQpDelta as follows.

* If cu\_qp\_delta\_sign is equal to 0, the corresponding CuQpDelta has a positive value.
* Otherwise (cu\_qp\_delta\_sign is equal to 1), the corresponding CuQpDelta has a negative value.

When cu\_qp\_delta\_sign is not present, it is inferred to be equal to 0.

When cu\_qp\_delta\_abs is present, the variables IsCuQpDeltaCoded and CuQpDelta are derived as follows.

IsCuQpDeltaCoded = 1

CuQpDelta = cu\_qp\_delta\_abs \* ( 1 − 2 \* cu\_qp\_delta\_sign )

The decoded value of CuQpDelta shall be in the range of −( 26+ QpBdOffsetY / 2 ) to +( 25+ QpBdOffsetY / 2 ), inclusive.

…”

**“8.4 “****Decoding process for coding units coded in intra prediction mode**

**8.4.1 General decoding process for coding units coded in intra prediction mode**

Inputs to this process are:

– a luma location ( xC, yC ) specifying the top-left sample of the current luma coding block relative to the top‑left luma sample of the current picture,

– a variable log2CbSize specifying the size of the current luma coding block.

Output of this process is:

– a modified reconstructed picture before deblocking filtering.

The derivation process for quantization parameters as specified in subclause 8.6.1 is invoked with the luma location ( xC, yC ) as input.

…”

**“…**

**8.6 Scaling, transformation and array construction process prior to deblocking filter process**

**8.6.1 Derivation process for quantization parameters**

Input of this process is:

– a luma location ( xC, yC ) specifying the top-left sample of the current luma coding block relative to the top left luma sample of the current picture.The luma location ( xQG, yQG ), specifies the top-left luma sample of the current quantization group relative to the top‑left luma sample of the current picture. The horizontal and vertical positions xQG and yQG are set equal to ( xC − ( xC & ((1 << Log2MinCuQPDeltaSize) − 1) ) ) and ( yC − ( yC & ((1 << Log2MinCuQPDeltaSize) − 1) ) ), respectively.A Qp region within the current quantization group includes a square luma block with dimension ( 1<<log2QprSize ) and the two corresponding chroma blocks. log2QprSize is set equal to Max( 3, Log2MinTrafoSize ). The luma location ( xQ, yQ ) specifies the top-left luma sample of the Qp region relative to ( xQG, yQG ), with xQ and yQ equal to ( iq<<log2QprSize ) and ( jq<<log2QprSize ), respectively, with iq and jq=0..( (1 << Log2MinCuQPDeltaSize) >> log2QprSize ) - 1. The z-scan order address zq of the Qp region (iq, jq) within the quantization group is set equal to MinTbAddrZS[iq][jq].

The luma location ( xT, yT ) specifies the top-left sample of the luma transform block in the transform unit containing syntax element cu\_qp\_delta\_abs within the current quantization group relative to the top-left luma sample of the current picture. If cu\_qp\_delta\_abs is not decoded, then ( xT, yT ) is set equal to ( xQG, yQG ).

The z-scan order address zqT of the Qp region covering the luma location ( xT-xQG, yT-yQG ) within the current quantization group is set equal to MinTbAddrZS[ (xT-xQG) >> log2QprSize ][ (yT-yQG) >> log2QprSize ].

The predicted luma quantization parameter qPY\_PRED is derived by the following ordered steps:

1. The variable qPY\_PREV is derived as follows.

– If one or more of the following conditions are true, qPY\_PREV is set equal to SliceQPY.

– The current quantization group is the first quantization group in a slice.

– The current quantization group is the first quantization group in a tile.

– The current quantization group is the first quantization group in a coding tree block row and tiles\_or\_entry\_coding\_sync\_idc is equal to 2.

– Otherwise, qPY\_PREV is set equal to the luma quantization parameter QPY of the last Qp region within the previous coding unit in decoding order, respectively.

1. The availability derivation process for a block in z-scan order as specified in subclause 6.4.1 is invoked with the location ( xCurr, yCurr ) set equal to ( xB, yB ) and the neighbouring location ( xN, yN ) set equal to ( xQG−1, yQG ) as the input and the output is assigned to availableA. The variable qPY\_A is derived as follows.

– If availableA is equal to FALSE or the coding tree block address of the coding tree block containing the luma coding block covering ( xQG − 1,  yQG ) ctbAddrA is not equal to CtbAddrTS, qPY\_A is set equal to qPY\_PREV.

– Otherwise, qPY\_A is set equal to the luma quantization parameter QPY of the Qp region ~~coding unit containing the luma coding block~~ covering ( xQG − 1,  yQG ).

1. The availability derivation process for a block in z-scan order as specified in subclause 6.4.1 is invoked with the location ( xCurr, yCurr ) set equal to ( xB, yB ) and the neighbouring location ( xN, yN ) set equal to ( xQG, yQG−1 ) as the input and the output is assigned to availableB. The variable qPY\_B is derived as follows.

– If availableB is equal to FALSE or the coding tree block address of the coding tree block containing the luma coding block covering ( xQG, yQG − 1 ) ctbAddrB is not equal to ctbAddrTS, qPY\_B is set equal to qPY\_PREV.

– Otherwise, qPY\_B is set equal to the luma quantization parameter QPY of the Qp region ~~coding unit containing the luma coding block~~ covering ( xQG, yQG − 1 ).

1. The predicted luma quantization parameter qPY\_PRED is derived as:

qPY\_PRED =  (qPY\_A + qPY\_B + 1) >> 1

The variable QPY of a Qp region with z-scan index zq within the current quantization group and within the current coding unit is derived as:

– If index zq is greater than or equal to zqT and CuQpDelta is non-zero,

QPY = ( ( ( qPY\_PRED + CuQpDelta +52+ 2\*QpBdOffsetY )%( 52 + QpBdOffsetY ) ) − QpBdOffsetY

– Otherwise:

QPY = qPY\_PRED

The luma quantization parameter QP’Y is derived as

QP’Y = QPY + QpBdOffsetY

The variables qPCb and qPCr are set equal to the value of QPC as specified in Table 8‑9 based on the index qPiequal to qPiCb and qPiCr derived as:

qPiCb = Clip3( −QpBdOffsetC, 57, QPY + pic\_cb\_qp\_offset + slice\_cb\_qp\_offset )

qPiCr = Clip3( −QpBdOffsetC, 57, QPY + pic\_cr\_qp\_offset + slice\_cr\_qp\_offset )

The chroma quantization parameters for Cb and Cr components, QP’Cb and QP’Cr are derived as:

QP’Cb = qPCb + QpBdOffsetC

QP’Cr = qPCr + QpBdOffsetC

Table 8‑9 – Specification of QPC as a function of qPi

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| qPi | <30 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | >43 |
| QPC | = qPi | 29 | 30 | 31 | 32 | 33 | 33 | 34 | 34 | 35 | 35 | 36 | 36 | 37 | 37 | = qPi − 6 |

…”

**“8.7.2.4.3 Decision process for luma block edge**

…

The variables QPQ and QPP are set equal to the QPY values of the ~~coding units which include the coding blocks~~ Qp regions containing the sample q0,0 and p0,0, respectively, as specified in subclause 8.6.1 with as inputs the luma location of the coding units which include the coding blocks containing the sample q0,0 and p0,0, respectively.

…”

**“8.7.2.4.5 Filtering process for chroma block edge**

…

The variables QPQ and QPP are set equal to the QPY values of the ~~coding units which include the coding blocks~~Qp regions containing the sample q0,0 and p0,0, respectively, as specified in subclause 8.6.1 with as inputs the luma location of the coding units which include the coding blocks containing the sample q0,0 and p0,0, respectively.

…”

# Draft text changes for solution 2

The following DIS text changes are based on [1].

Table

|  |  |
| --- | --- |
| coding\_unit( x0, y0, log2CbSize ) { | Descriptor |
| if( transquant\_bypass\_enable\_flag ) { |  |
| **cu\_transquant\_bypass\_flag** | ae(v) |
| } |  |
| if( slice\_type != I ) |  |
| **skip\_flag**[ x0 ][ y0 ] | ae(v) |
| if( skip\_flag[ x0 ][ y0 ] ) |  |
| prediction\_unit( x0, y0, log2CbSize ) |  |
| else { |  |
| nCbS = ( 1 << log2CbSize ) |  |
| if( slice\_type != I ) |  |
| **pred\_mode\_flag** | ae(v) |
| if( PredMode[ x0 ][ y0 ] != MODE\_INTRA | | log2CbSize = = Log2MinCbSize ) |  |
| **part\_mode** | ae(v) |
| if( PredMode[ x0 ][ y0 ] = = MODE\_INTRA ) { |  |
| if( PartMode = = PART\_2Nx2N && pcm\_enabled\_flag &&  log2CbSize >= Log2MinIPCMCUSize &&  log2CbSize <= Log2MaxIPCMCUSize ) |  |
| **pcm\_flag** | ae(v) |
| if( pcm\_flag ) { |  |
| **num\_subsequent\_pcm** | tu(3) |
| NumPCMBlock = num\_subsequent\_pcm + 1 |  |
| while( !byte\_aligned( ) ) |  |
| **pcm\_alignment\_zero\_bit** | f(1) |
| pcm\_sample( x0, y0, log2CbSize ) |  |
| } else { |  |
| pbOffset = ( PartMode = = PART\_NxN ) ? ( nCbS / 2 ) : 0 |  |
| for( j = 0; j <= pbOffset; j = j + pbOffset ) |  |
| for( i = 0; i <= pbOffset; i = i + pbOffset ) { |  |
| **prev\_intra\_luma\_pred\_flag**[ x0 + i ][ y0+ j ] | ae(v) |
| } |  |
| for( j = 0; j <= pbOffset; j = j + pbOffset ) |  |
| for( i = 0; i <= pbOffset; i = i + pbOffset ) { |  |
| if( prev\_intra\_luma\_pred\_flag[ x0 + i ][ y0+ j ] ) |  |
| **mpm\_idx**[ x0 + i ][ y0+ j ] | ae(v) |
| else |  |
| **rem\_intra\_luma\_pred\_mode**[ x0 + i ][ y0+ j ] | ae(v) |
| } |  |
| **intra\_chroma\_pred\_mode**[ x0 ][ y0 ] | ae(v) |
| } |  |
| } else { |  |
| if( PartMode = = PART\_2Nx2N ) |  |
| prediction\_unit( x0, y0, nCbS, nCbS ) |  |
| else if( PartMode = = PART\_2NxN ) { |  |
| prediction\_unit( x0, y0, nCbS, nCbS / 2 ) |  |
| prediction\_unit( x0, y0 + ( nCbS / 2 ), nCbS, nCbS / 2 ) |  |
| } else if( PartMode = = PART\_Nx2N ) { |  |
| prediction\_unit( x0, y0, nCbS / 2, nCbS ) |  |
| prediction\_unit( x0 + ( nCbS / 2 ), y0, nCbS / 2, nCbS ) |  |
| } else if( PartMode = = PART\_2NxnU ) { |  |
| prediction\_unit( x0, y0, nCbS, nCbS / 4 ) |  |
| prediction\_unit( x0, y0 + ( nCbS / 4 ), nCbS, nCbS \*3 / 4 ) |  |
| } else if( PartMode = = PART\_2NxnD ) { |  |
| prediction\_unit( x0, y0, nCbS, nCbS \*3 / 4 ) |  |
| prediction\_unit( x0, y0 + ( nCbS \* 3 / 4 ), nCbS, nCbS / 4 ) |  |
| } else if( PartMode = = PART\_nLx2N ) { |  |
| prediction\_unit( x0, y0, nCbS /4, nCbS ) |  |
| prediction\_unit( x0 + ( nCbS / 4 ), y0, nCbS \*3 / 4, nCbS) |  |
| } else if( PartMode = = PART\_nRx2N ) { |  |
| prediction\_unit( x0, y0, nCbS \*3 / 4, nCbS ) |  |
| prediction\_unit( x0 + ( nCbS \* 3 / 4 ), y0, nCbS / 4, nCbS ) |  |
| } else { /\* PART\_NxN \*/ |  |
| prediction\_unit( x0, y0, nCbS / 2, nCbS / 2) |  |
| prediction\_unit( x0 + ( nCbS / 2 ), y0, nCbS / 2, nCbS / 2 ) |  |
| prediction\_unit( x0, y0 + ( nCbS / 2 ), nCbS / 2, nCbS / 2 ) |  |
| prediction\_unit( x0 + ( nCbS / 2 ), y0 + ( nCbS / 2 ), nCbS / 2, nCbS / 2 ) |  |
| } |  |
| } |  |
| if( !pcm\_flag ) { |  |
| ~~if( PredMode[ x0 ][ y0 ] != MODE\_INTRA &&   !(PartMode = = PART\_2Nx2N && merge\_flag[x0][y0]) )~~ |  |
| if( !(PartMode = = PART\_2Nx2N && merge\_flag[x0][y0]) ) | | (MODE\_INTRA && cu\_delta\_qp\_enabled) |  |
| **no\_residual\_syntax\_flag** | ae(v) |
| ~~if( !no\_residual\_syntax\_flag ) {~~ |  |
| if( !no\_residual\_syntax\_flag | | PredMode[ x0 ][ y0 ] = = MODE\_INTRA ) { |  |
| MaxTrafoDepth = ( PredMode[ x0 ][ y0 ] = = MODE\_INTRA ?   max\_transform\_hierarchy\_depth\_intra + IntraSplitFlag :   max\_transform\_hierarchy\_depth\_inter ) |  |
| if(!no\_residual\_syntax\_flag && cu\_qp\_delta\_enabled\_flag && !IsCuQpDeltaCoded ) { |  |
| **cu\_qp\_delta\_abs** | ae(v) |
| if( cu\_qp\_delta\_abs ) |  |
| **cu\_qp\_delta\_sign** | ae(v) |
| } |  |
| transform\_tree( x0, y0 x0, y0, log2CbSize, 0, 0 ) |  |
| } |  |
| } |  |
| } |  |
| } |  |

Table

|  |  |
| --- | --- |
| transform\_unit( x0, y0, xBase, yBase, log2TrafoSize, trafoDepth, blkIdx ) { | Descriptor |
| if( cbf\_luma[ x0 ][ y0 ][ trafoDepth ] | | cbf\_cb[ x0 ][ y0 ][ trafoDepth ] | |  cbf\_cr[ x0 ][ y0 ][ trafoDepth ] ) { |  |
| ~~if( cu\_qp\_delta\_enabled\_flag && !IsCuQpDeltaCoded ) {~~ |  |
| **~~cu\_qp\_delta\_abs~~** | ~~ae(v)~~ |
| ~~if( cu\_qp\_delta\_abs )~~ |  |
| **~~cu\_qp\_delta\_sign~~** | ~~ae(v)~~ |
| ~~}~~ |  |
| if( cbf\_luma[ x0 ][ y0 ][ trafoDepth ] ) |  |
| residual\_coding( x0, y0, log2TrafoSize, 0 ) |  |
| if( log2TrafoSize > 2 ) { |  |
| if( cbf\_cb[ x0 ][ y0 ][ trafoDepth ] ) |  |
| residual\_coding( x0, y0, log2TrafoSize, 1 ) |  |
| if( cbf\_cr[ x0 ][ y0 ][ trafoDepth ] ) |  |
| residual\_coding( x0, y0, log2TrafoSize, 2 ) |  |
| } else if( blkIdx = = 3 ) { |  |
| if( cbf\_cb[ xBase ][ yBase ][ trafoDepth ] ) |  |
| residual\_coding( xBase, yBase, log2TrafoSize, 1 ) |  |
| if( cbf\_cr[ xBase ][ yBase ][ trafoDepth ] ) |  |
| residual\_coding( xBase, yBase, log2TrafoSize, 2 ) |  |
| } |  |
| } |  |
| } |  |

Table

|  |  |
| --- | --- |
| transform\_tree( x0, y0, xBase, yBase, log2TrafoSize, trafoDepth, blkIdx ) { | Descriptor |
| if( log2TrafoSize <= Log2MaxTrafoSize &&   log2TrafoSize > Log2MinTrafoSize &&  trafoDepth < MaxTrafoDepth && !(IntraSplitFlag && trafoDepth = = 0) ) |  |
| **split\_transform\_flag**[ x0 ][ y0 ][ trafoDepth ] | ae(v) |
| if( ( trafoDepth = = 0 | | log2TrafoSize > 2 ) && !no\_residual\_syntax\_flag ) { |  |
| if( trafoDepth = = 0 | | cbf\_cb[ xBase ][ yBase ][ trafoDepth − 1 ] ) |  |
| **cbf\_cb**[ x0 ][ y0 ][ trafoDepth ] | ae(v) |
| if( trafoDepth = = 0 | | cbf\_cr[ xBase ][ yBase ][ trafoDepth − 1 ] ) |  |
| **cbf\_cr**[ x0 ][ y0 ][ trafoDepth ] | ae(v) |
| } |  |
| if( split\_transform\_flag[ x0 ][ y0 ][ trafoDepth ] ) { |  |
| x1 = x0 + ( ( 1 << log2TrafoSize ) >> 1 ) |  |
| y1 = y0 + ( ( 1 << log2TrafoSize ) >> 1 ) |  |
| transform\_tree( x0, y0, x0, y0, log2TrafoSize − 1, trafoDepth + 1, 0 ) |  |
| transform\_tree( x1, y0, x0, y0, log2TrafoSize − 1 trafoDepth + 1, 1 ) |  |
| transform\_tree( x0, y1, x0, y0, log2TrafoSize − 1, trafoDepth + 1, 2 ) |  |
| transform\_tree( x1, y1, x0, y0, log2TrafoSize − 1, trafoDepth + 1, 3 ) |  |
| } else { |  |
| if( ( PredMode[ x0 ][ y0 ] = = MODE\_INTRA | | trafoDepth != 0 | |  cbf\_cb[ x0 ][ y0 ][ trafoDepth ] | | cbf\_cr[ x0 ][ y0 ][ trafoDepth ] ) && !no\_residual\_syntax\_flag ) |  |
| **cbf\_luma**[ x0 ][ y0 ][ trafoDepth ] | ae(v) |
| transform\_unit (x0, y0, xBase, yBase, log2TrafoSize, trafoDepth, blkIdx) |  |
| } |  |
| } |  |

| Table 9‑32 – Syntax elements and associated types of binarization, maxBinIdxCtx, ctxIdxTable, and ctxIdxOffset | | | | | |
| --- | --- | --- | --- | --- | --- |
| no\_residual\_syntax\_flag | 0 | FL, cMax = 1 | 0 | Table 9‑21 | 0 |
| 1 | 0 | Table 9‑21 | 1 |
| 2 | 0 | Table 9‑21 | 2 |

Table 9‑21 – Values of variable initValue for no\_residual\_syntax\_flag ctxIdx

|  |  |  |  |
| --- | --- | --- | --- |
| **Initialization variable** | **no\_residual\_syntax\_flag ctxIdx** | | |
| **0** | **1** | **2** |
| **initValue** | 79 | 79 | 79 |

# Patent rights declarations

**Intel Corporation may have current or pending patent rights relating to the technology described in this contribution and, conditioned on reciprocity, is prepared to grant licenses under reasonable and non-discriminatory terms as necessary for implementation of the resulting ITU-T Recommendation | ISO/IEC International Standard (per box 2 of the ITU-T/ITU-R/ISO/IEC patent statement and licensing declaration form).**

**Qualcomm Incorporated may have current or pending patent rights relating to the technology described in this contribution and, conditioned on reciprocity, is prepared to grant licenses under reasonable and non-discriminatory terms as necessary for implementation of the resulting ITU-T Recommendation | ISO/IEC International Standard (per box 2 of the ITU-T/ITU-R/ISO/IEC patent statement and licensing declaration form).**