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

Inputs to this process are:

– a luma location ( xCb, yCb ) 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 ( xCb, yCb ) as input.

A variable nCbS is set equal to 1  <<  log2CbSize.

Depending on the values of pcm\_flag[ xCb ][ yCb ], palette\_mode\_flag[ xCb ][ yCb ], and IntraSplitFlag, the decoding process for luma samples is specified as follows:

– If pcm\_flag[ xCb ][ yCb ] is equal to 1, the reconstructed picture is modified as follows:

SL[ xCb + i ][ yCb + j ] =   
 pcm\_sample\_luma[ ( nCbS \* j ) + i ]  <<  ( BitDepthY − PcmBitDepthY ), with i, j = 0..nCbS − 1 (8‑12)

– Otherwise (pcm\_flag[ xCb ][ yCb ] is equal to 0), if palette\_mode\_flag[ xCb ][ yCb ] is equal to 1, the following order steps apply:

1. The decoding process for palette intra blocks as specified in subclause 8.4.5.2.8 is invoked with the luma location ( xCb, yCb ), nCbS, the variable cIdx set equal to 0, the palette modes array palette\_mode, the palette indices array paletteMap, and the array of quantized escape values palette\_escape\_val as inputs, and the output is an nCbS x nCbS array of reconstructed -palette sample values, recSamples[ x ][ y ], x, y = 0..nCbS − 1
2. The reconstructed picture is modified as follows:

– If palette\_transpose\_flag is true,

SL[ yCb + y ][ xCb + x ] = palette\_sample\_values[ x ][ y ]

– Otherwise (palette\_transpose\_flag is false)

SL[ xCb + x ][ yCb + y ] = palette\_sample\_values[ x ][ y ].

– Otherwise (pcm\_flag[ xCb ][ yCb ] is equal to 0, palette\_mode\_flag[ xCb ][ yCb ] 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 8.4.2 is invoked with the luma location ( xCb, yCb ) as input.
2. When intra\_bc\_flag[ xCb ][ yCb ] is equal to 1, the derivation process for block vector components in intra block copying prediction mode as specified in subclause 8.4.4 is invoked with the luma location ( xCb, yCb ) and variable log2CbSize as inputs, and the output being bvIntra.
3. If cu\_residual\_act\_flag[ xCb ][ yCb ] is equal to 1, the following applies:

– For the variable cIdx proceeding over the values 0..2, the following ordered steps apply:

* + The variable comp is set equal to (!cIdx ? L : (cIdx = =1 ? Cb : Cr).
  + The general decoding process for intra blocks as specified in subclause 8.4.4.1 is invoked with the location ( xCb, yCb ), the variable log2TrafoSize set equal to log2CbSize, the variable trafoDepth set equal to 0, the variable predModeIntra set equal to IntraPredModeY[ xCb ][ yCb ], the variable predModeIntraBc set equal to intra\_bc\_flag[ xCb ][ yCb ], the variable bvIntra, the variable cIdx, and variable controlParaACT equal to 1 as inputs, and the output is the residual sample array resSamplescomp.

~~– The residual modification process for residual blocks using adaptive colour transform as specified in subclause 8.6.8 is invoked with the variable blkSize set equal to nCbS, the (nCbS)x(nCbS) array r~~~~Y~~ ~~set equal to resSamples~~~~L~~~~, the (nCbS)x(nCbS) array r~~~~Cb~~ ~~set equal to resSamples~~~~Cb~~~~, and the (nCbS)x(nCbS) array r~~~~Cr~~ ~~set equal to resSamples~~~~Cr~~ ~~as inputs, and the output are modified versions of the (nCbS)x(nCbS) arrays resSamples~~~~L~~~~, resSamples~~~~Cb~~ ~~and resSamples~~~~Cr~~~~.~~

1. The general decoding process for intra blocks as specified in subclause 8.4.4.1is invoked with the luma location ( xCb, yCb ), the variable log2TrafoSize set equal to log2CbSize, the variable trafoDepth set equal to 0, the variable predModeIntra set equal to IntraPredModeY[ xCb ][ yCb ], the variable cIdx set equal to 0, and variable controlParaACT set to (cu\_residual\_act\_flag[ xCb ][ yCb ] ? 2 : 3 ) as inputs, and the output is a modified reconstructed picture before deblocking filtering.

– Otherwise (pcm\_flag[ xCb ][ yCb ] is equal to 0, palette\_mode\_flag[ xCb ][ yCb ] is equal to 0 and IntraSplitFlag is equal to 1), for the variable blkIdx proceeding over the values 0..3, the following ordered steps apply:

1. The variable xPb is set equal to xCb + ( nCbS  >>  1 ) \* ( blkIdx % 2 ).
2. The variable yPb is set equal to yCb + ( nCbS  >>  1 ) \* ( blkIdx / 2 ).
3. The derivation process for the intra prediction mode as specified in subclause 8.4.2 is invoked with the luma location ( xPb, yPb ) as input.
4. If cu\_residual\_act\_flag[ xCb ][ yCb ] is equal to 1, the following applies:

* For the variable cIdx proceeding over the values 0..2, the following ordered steps apply:
  + The variable comp is set equal to (!cIdx ? L : (cIdx = =1 ? Cb : Cr).
  + The general decoding process for intra blocks as specified in subclause 8.4.4.1is invoked with the luma location ( xPb, yPb ), the variable log2TrafoSize set equal to log2CbSize − 1, the variable trafoDepth set equal to 1, the variable predModeIntra set equal to IntraPredModeY[ xPb ][ yPb ], the variable predModeIntraBc set equal to 0, the variable cIdx, and variable controlParaACT set equal to 1 as inputs, and the output is the residual sample array resSamplescomp.
* ~~The variable nSubCbS is set equal to ( nCbS  >> 1 ) and the residual modification process for residual blocks using adaptive colour transform as specified in subclause 8.6.8 is invoked with the variable blkSize set equal to nSubCbS, the (nSubCbS)x(nSubCbS) array r~~~~Y~~ ~~set equal to resSamples~~~~L~~~~, the (nSubCbS)x(nSubCbS) array r~~~~Cb~~ ~~set equal to resSamples~~~~Cb~~~~, and the (nSubCbS)x(nSubCbS) array r~~~~Cr~~ ~~set equal to resSamples~~~~Cr~~ ~~as inputs, and the outputs are modified versions of the (nSubCbS)x(nSubCbS) arrays resSamples~~~~L~~~~, resSamples~~~~Cb~~ ~~and resSamples~~~~Cr~~

1. The general decoding process for intra blocks as specified in subclause 8.4.4.1is invoked with the luma location ( xPb, yPb ), the variable log2TrafoSize set equal to log2CbSize − 1, the variable trafoDepth set equal to 1, the variable predModeIntra set equal to IntraPredModeY[ xPb ][ yPb ], the variable cIdx set equal to 0, and variable controlParaACT set to ( cu\_residual\_act\_flag[ xCb ][ yCb ] ? 2 : 3 ) as inputs, and the output is a modified reconstructed picture before deblocking filtering.

When ChromaArrayType is not equal to 0, the following applies.

The variable log2CbSizeC is set equal to log2CbSize − ( ChromaArrayType  = =  3 ? 0 : 1 ).

Depending on the values of pcm\_flag[ xCb ][ yCb ] and IntraSplitFlag, the decoding process for chroma samples is specified as follows:

– If pcm\_flag[ xCb ][ yCb ] is equal to 1, the reconstructed picture is modified as follows:

SCb[ xCb / SubWidthC + i ][ yCb / SubHeightC + j ] =   
 pcm\_sample\_chroma[ ( nCbS / SubWidthC \* j ) + i ]  <<  ( BitDepthC − PcmBitDepthC ),  
 with i = 0..nCbS / SubWidthC − 1, and j = 0..nCbS / SubHeightC − 1 (8‑13)

SCr[ xCb / SubWidthC + i ][ yCb / SubHeightC + j ] =   
 pcm\_sample\_chroma[ ( nCbS / SubWidthC \* ( j + nCbS / SubHeightC ) ) + i ]  <<   ( BitDepthC − PcmBitDepthC ),  
 with i = 0..nCbS / SubWidthC − 1, and j = 0..nCbS / SubHeightC − 1 (8‑14)

– Otherwise (pcm\_flag[ xCb ][ yCb ] is equal to 0), if palette\_mode\_flag[ xCb ][ yCb ] is equal to 1 the following orderd steps apply:

1. The decoding process for palette intra blocks as specified in subclause 8.4.5.2.8 is invoked with the chroma location ( xCb, yCb ), nCbS, the variable cIdx set equal to 1, the palette modes array palette\_mode, the palette indices array paletteMap, and the array of quantized escape values palette\_escape\_val as inputs, and the output is an nCbS x nCbS array of reconstructed palette sample values, recSamples[ x ][ y ], x, y = 0..nCbS − 1.
2. The reconstructed picture is modified as follows:

– If palette\_transpose\_flag is true, SCb[ yCb + y ][ xCb + x ] is set equal to palette\_sample\_values[ x ][ y ],

– Otherwise (palette\_transpose\_flag is false), SCb[ xCb + x ][ yCb + y ] is set equal to palette\_sample\_values[ x ][ y ].

1. The decoding process for palette intra blocks as specified in subclause 8.4.5.2.8 is invoked with the chroma location ( xCb, yCb ), nCbS, the variable cIdx set equal to 2, the palette modes array palette\_mode, the palette indices array paletteMap, and the array of quantized escape values palette\_escape\_val as inputs, and the output is an nCbS x nCbS array of reconstructed palette sample values, recSamples[ x ][ y ], x, y = 0..nCbS − 1.
2. The reconstructed picture is modified as follows:

– If palette\_transpose\_flag is true, SCr[ yCb + y ][ xCb + x ] is set equal to palette\_sample\_values[ x ][ y ]

– Otherwise (palette\_transpose\_flag is false), SCr[ xCb + x ][ yCb + y ] is set equal to palette\_sample\_values[ x ][ y ]

– Otherwise (pcm\_flag[ xCb ][ yCb ] is equal to 0, palette\_mode\_flag[ xCb ][ yCb ] is equal to 0 ), if IntraSplitFlag is equal to 0 or ChromaArrayType is not equal to 3, the following ordered steps apply:

1. The derivation process for the chroma intra prediction mode as specified in 8.4.3 is invoked with the luma location ( xCb, yCb ) as input, and the output is the variable IntraPredModeC.
2. The general decoding process for intra blocks as specified in subclause 8.4.4.1is invoked with the chroma location ( xCb / SubWidthC, yCb / SubHeightC ), the variable log2TrafoSize set equal to log2CbSizeC, the variable trafoDepth set equal to 0, the variable predModeIntra set equal to IntraPredModeC, the variable cIdx set equal to 1, and variable controlParaACT set to ( cu\_residual\_act\_flag[ xCb ][ yCb ] ? 2 : 3 ) as inputs, and the output is a modified reconstructed picture before deblocking filtering.
3. The general decoding process for intra blocks as specified in subclause 8.4.4.1is invoked with the chroma location ( xCb / SubWidthC, yCb / SubHeightC ), the variable log2TrafoSize set equal to log2CbSizeC, the variable trafoDepth set equal to 0, the variable predModeIntra set equal to IntraPredModeC, the variable cIdx set equal to 2, and variable controlParaACT set to ( cu\_residual\_act\_flag[ xCb ][ yCb ] ? 2 : 3 ) as inputs, and the output is a modified reconstructed picture before deblocking filtering.

– Otherwise (pcm\_flag[ xCb ][ yCb ] is equal to 0, palette\_mode\_flag[ xCb ][ yCb ] is equal to 0, IntraSplitFlag is equal to 1 and ChromaArrayType is equal to 3), for the variable blkIdx proceeding over the values 0..3, the following ordered steps apply:

1. The variable xPb is set equal to xCb + ( nCbS  >>  1 ) \* ( blkIdx % 2 ).
2. The variable yPb is set equal to yCb + ( nCbS  >>  1 ) \* ( blkIdx / 2 ).
3. The derivation process for the chroma intra prediction mode as specified in 8.4.3 is invoked with the luma location ( xPb, yPb ) as input, and the output is the variable IntraPredModeC.
4. The general decoding process for intra blocks as specified in subclause 8.4.4.1 is invoked with the chroma location ( xPb, yPb ), the variable log2TrafoSize set equal to log2CbSizeC − 1, the variable trafoDepth set equal to 1, the variable predModeIntra set equal to IntraPredModeC, the variable cIdx set equal to 1, and variable controlParaACT set to ( cu\_residual\_act\_flag[ xCb ][ yCb ] ? 2 : 3 ) as inputs, and the output is a modified reconstructed picture before deblocking filtering.
5. The general decoding process for intra blocks as specified in subclause 8.4.4.1is invoked with the chroma location ( xPb, yPb ), the variable log2TrafoSize set equal to log2CbSizeC − 1, the variable trafoDepth set equal to 1, the variable predModeIntra set equal to IntraPredModeC, the variable cIdx set equal to 2, and variable controlParaACT set to ( cu\_residual\_act\_flag[ xCb ][ yCb ] ? 2 : 3 ) as inputs, and the output is a modified reconstructed picture before deblocking filtering.

#### 8.4.5.1 General decoding process for intra blocks

Inputs to this process are:

– a sample location ( xTb0, yTb0 ) specifying the top-left sample of the current transform block relative to the top‑left sample of the current picture,

– a variable log2TrafoSize specifying the size of the current transform block,

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

– a variable predModeIntra specifying the intra prediction mode,

– a variable predModeIntraBc specifying the intra block copying mode,

– a variable bvIntra specifying the intra block copying vector,

– a variable cIdx specifying the colour component of the current block, and

– a variable controlParaACT specifying the applicable processes.

Output of this process is a modified reconstructed picture before deblocking filtering when controlParaACT is not equal to 1, or residual sample array when controlParaACT is equal to 1.

The luma sample location ( xTbY, yTbY ) specifying the top-left sample of the current luma transform block relative to the top-left luma sample of the current picture is derived as follows:

( xTbY, yTbY ) = ( cIdx  = =  0 ) ? ( xTb0, yTb0 ) : ( xTb0 \* SubWidthC, yTb0 \* SubHeightC ) (8‑26)

The variable splitFlag is derived as follows:

– If cIdx is equal to 0, splitFlag is set equal to split\_transform\_flag[ xTbY ][ yTbY ][ trafoDepth ].

– Otherwise, if all of the following conditions are true, splitFlag is set equal to 1.

* cIdx is greater than 0
* split\_transform\_flag[ xTbY ][ yTbY ][ trafoDepth ] is equal to 1
* log2TrafoSize is greater than 2

– Otherwise, splitFlag is set equal to 0.

Depending on the value of splitFlag, the following applies:

– If splitFlag is equal to 1, the following ordered steps apply:

1. The variables xTb1 and yTb1 are derived as follows:
   * If either cIdx is equal to 0 or ChromaArrayType is not equal to 2, the following applies:
   * The variable xTb1 is set equal to xTb0 + ( 1  <<  ( log2TrafoSize − 1 ) ).
   * The variable yTb1 is set equal to yTb0 + ( 1  <<  ( log2TrafoSize − 1 ) ).
   * Otherwise (ChromaArrayType is equal to 2 and cIdx is greater than 0), the following applies:
   * The variable xTb1 is set equal to xTb0 + ( 1  <<  ( log2TrafoSize − 1 ) ).
   * The variable yTb1 is set equal to yTb0 + ( 2  <<  ( log2TrafoSize − 1 ) ).
2. The general decoding process for intra blocks as specified in this subclause is invoked with the location ( xTb0, yTb0 ), the variable log2TrafoSize set equal to log2TrafoSize − 1, the variable trafoDepth set equal to trafoDepth + 1, the intra prediction mode predModeIntra, and the variable cIdx as inputs, and the output is a modified reconstructed picture before deblocking filtering.
3. The general decoding process for intra blocks as specified in this subclause is invoked with the location ( xTb1, yTb0 ), the variable log2TrafoSize set equal to log2TrafoSize − 1, the variable trafoDepth set equal to trafoDepth + 1, the intra prediction mode predModeIntra, and the variable cIdx as inputs, and the output is a modified reconstructed picture before deblocking filtering.
4. The general decoding process for intra blocks as specified in this subclause is invoked with the location ( xTb0, yTb1 ), the variable log2TrafoSize set equal to log2TrafoSize − 1, the variable trafoDepth set equal to trafoDepth + 1, the intra prediction mode predModeIntra, and the variable cIdx as inputs, and the output is a modified reconstructed picture before deblocking filtering.
5. The general decoding process for intra blocks as specified in this subclause is invoked with the location ( xTb1, yTb1 ), the variable log2TrafoSize set equal to log2TrafoSize − 1, the variable trafoDepth set equal to trafoDepth + 1, the intra prediction mode predModeIntra, and the variable cIdx as inputs, and the output is a modified reconstructed picture before deblocking filtering.

– Otherwise (splitFlag is equal to 0), for the variable blkIdx proceeding over the values 0..( cIdx > 0  &&  ChromaArrayType  = =  2 ? 1 : 0 ), the following ordered steps apply:

1. The variable nTbS is set equal to 1  <<  log2TrafoSize.
2. The variable yTbOffset is set equal to blkIdx \* nTbS.
3. The variable yTbOffsetY is set equal to yTbOffset \* SubHeightC.
4. When controlParaACT is not equal to 2, the variable residualDpcm is derived as follows:
   * If all of the following conditions are true, residualDpcm is set equal to 1.
   * implicit\_rdpcm\_enabled\_flag is equal to 1.
   * either transform\_skip\_flag[ xTbY ][ yTbY + yTbOffsetY ][ cIdx ] is equal to 1, or cu\_transquant\_bypass\_flag is equal to 1.
   * either predModeIntra is equal to 10, or predModeIntra is equal to 26.
   * Otherwise, residualDpcm is set equal to explicit\_rdpcm\_flag[ xTbY ][ yTbY + yTbOffsetY ][ cIdx ].
5. When controlParaACT is not equal to 1, depending upon the value of predModeIntraBc, the following applies:

– When predModeIntraBc is equal to 0, the general intra sample prediction process as specified in subclause 8.4.4.2.1 is invoked with the transform block location ( xTb0, yTb0 + yTbOffset ), the intra prediction mode predModeIntra, the transform block size nTbS, and the variable cIdx as inputs, and the output is an (nTbS)x(nTbS) array predSamples.

– Otherwise (predModeIntraBc is equal to 1), the intra block copying process as specified in subclause 8.4.4.2.7 is invoked with the transform block location ( xTb0, yTb0 + yTbOffset ), the transform block size nTbS, the variable trafoDepth, the variable bvIntra, and the variable cIdx as inputs, and the output is an (nTbS)x(nTbS) array predSamples.

1. When controlParaACT is not equal to 2, the scaling and transformation process as specified in subclause 8.6.2 is invoked with the luma location ( xTbY, yTbY + yTbOffsetY ), the variable trafoDepth, the variable cIdx, and the transform size trafoSize set equal to nTbS as inputs, and the output is an (nTbS)x(nTbS) array resSamples.
2. When controlParaACT is not equal to 2 and residualDpcm is equal to 1, the directional residual modification process for blocks using a transform bypass as specified in subclause 8.6.5 is invoked with the variable mDir set equal to predModeIntra / 26, the variable nTbS, and the (nTbS)x(nTbS) array r set equal to the array resSamples as inputs, and the output is a modified (nTbS)x(nTbS) array resSamples.
3. When controlParaACT is not equal to 2 and cross\_component\_prediction\_enabled\_flag is equal to 1, and cu\_residual\_act\_flag[ xTbY][ yTbY ] is equal to 0, ChromaArrayType is equal to 3, and cIdx is not equal to 0, the residual modification process for transform blocks using cross-component prediction as specified in subclause 8.6.6 is invoked with the current luma transform block location ( xTbY, yTbY ), the variable nTbS, the variable cIdx, the (nTbS)x(nTbS) array rY set equal to the corresponding luma residual sample array resSamples of the current transform block, and the (nTbS)x(nTbS) array r set equal to the array resSamples as inputs, and the output is a modified (nTbS)x(nTbS) array resSamples.
4. When cu\_residual\_act\_flag[ xTbY][ yTbY ] is equal to 1, controlParaACT is equal to 1 and cIdx is equal to 2, the residual modification process for residual blocks using cross-component de-correlation as specified in subclause 8.6.8 is invoked with the current luma transform block location ( xTbY, yTbY ), the variable blkSize set equal to nTbS, the (nTbS)x(nTbS) array rY set equal to the corresponding luma residual sample array resSamples of the current transform block, the (nTbS)x(nTbS) array rCb set equal to the corresponding chroma residual sample array resSamples when cIdx is equal to 1 of the current transform block, and the (nTbS)x(nTbS) array rCr set equal to resSamples as inputs, and the output are modified versions of the three residual sample arrays.
5. When controlParaACT is not equal to 1, the picture construction process prior to in-loop filtering for a colour component as specified in subclause 8.6.7 is invoked with the transform block location ( xTb0, yTb0 + yTbOffset ), the variables nCurrSw and nCurrSh both set equal to nTbS, the variable cIdx, the (nTbS)x(nTbS) array predSamples, and the (nTbS)x(nTbS) array resSamples as inputs.

#### 8.5.4.1 General

Inputs to this process are:

– a luma location ( xCb, yCb ) 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.

Outputs of this process are:

– an (nCbSL)x(nCbSL) array resSamplesL of luma residual samples, where nCbSL is derived as specified below,

– when ChromaArrayType is not equal to 0, an (nCbSwC)x(nCbShC) array resSamplesCb of chroma residual samples for the component Cb, where nCbSwC and nCbShC are derived as specified below,

– when ChromaArrayType is not equal to 0, an (nCbSwC)x(nCbShC) array resSamplesCr of chroma residual samples for the component Cr, where nCbSwC and nCbShC are derived as specified below.

The variable nCbSL is set equal to 1  <<  log2CbSize. When ChromaArrayType is not equal to 0, the variable nCbSwC is set equal to nCbSL / SubWidthC and the variable nCbShC is set equal to nCbSL / SubHeightC.

Let resSamplesL be an (nCbSL)x(nCbSL) array of luma residual samples and when ChromaArrayType is not equal to 0, let resSamplesCb and resSamplesCr be two (nCbSwC)x(nCbShC) arrays of chroma residual samples.

Depending on the value of rqt\_root\_cbf, the following applies:

– If rqt\_root\_cbf is equal to 0 or cu\_skip\_flag[ xCb ][ yCb ] is equal to 1, all samples of the (nCbSL)x(nCbSL) array resSamplesL and when ChromaArrayType is not equal to 0, all samples of the two (nCbSwC)x(nCbShC) arrays resSamplesCb and resSamplesCr are set equal to 0.

– Otherwise (rqt\_root\_cbf is equal to 1), the following ordered steps apply:

1. The decoding process for luma residual blocks as specified in subclause 8.5.4.2 below is invoked with the luma location ( xCb, yCb ), the luma location ( xB0, yB0 ) set equal to ( 0, 0 ), the variable log2TrafoSize set equal to log2CbSize, the variable trafoDepth set equal to 0, the variable nCbS set equal to nCbSL, and the (nCbSL)x(nCbSL) array resSamplesL as inputs, and the output is a modified version of the (nCbSL)x(nCbSL) array resSamplesL.
2. When ChromaArrayType is not equal to 0, the decoding process for chroma residual blocks as specified in subclause 8.5.4.3 below is invoked with the luma location ( xCb, yCb ), the luma location ( xB0, yB0 ) set equal to ( 0, 0 ), the variable log2TrafoSize set equal to log2CbSize, the variable trafoDepth set equal to 0, the variable cIdx set equal to 1, the variable nCbSw set equal to nCbSwC, the variable nCbSh set equal to nCbShC, and the (nCbSwC)x(nCbShC) array resSamplesCb as inputs, and the output is a modified version of the (nCbSwC)x(nCbShC) array resSamplesCb.
3. When ChromaArrayType is not equal to 0, the decoding process for chroma residual blocks as specified in subclause 8.5.4.3 below is invoked with the luma location ( xCb, yCb ), the luma location ( xB0, yB0 ) set equal to ( 0, 0 ), the variable log2TrafoSize set equal to log2CbSize, the variable trafoDepth set equal to 0, the variable cIdx set equal to 2, the variable nCbSw set equal to nCbSwC, the variable nCbSh set equal to nCbShC, and the (nCbSwC)x(nCbShC) array resSamplesCr as inputs, and the output is a modified version of the (nCbSwC)x(nCbShC) array resSamplesCr.
4. ~~When cu\_residual\_act\_flag[ xCb ][ yCb ] is equal to 1, the residual modification process for residual blocks using adaptive colour transform as specified in subclause 8.6.8 is invoked with the variable blkSize set equal to nCbSL, the (nCbSL)x(nCbSL) array rY set equal to resSamplesL, the (nCbSL)x(nCbSL) array rCb set equal to resSamplesCb, and the (nCbSL)x(nCbSL) array rCr set equal to resSamplesCr as inputs, and the modified arrays resSamplesL, resSamplesCb and resSamplesCr as outputs.~~

#### 8.5.4.3 Decoding process for chroma residual blocks

This process is only invoked when ChromaArrayType is not equal to 0.

Inputs to this process are:

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

– a luma location ( xB0, yB0 ) specifying the top-left luma sample of the current chroma block relative to the top‑left sample of the current luma coding block,

– a variable log2TrafoSize specifying the size of the current chroma block in luma samples,

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

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

– the variables nCbSw and nCbSh specifying the width and height, respectively, of the current chroma coding block,

– an (nCbSw)x(nCbSh) array resSamples of chroma residual samples.

Output of this process is a modified version of the (nCbSw)x(nCbSh) array of chroma residual samples.

The variable splitChromaFlag is derived as follows:

– If split\_transform\_flag[ xCb + xB0 ][ yCb + yB0 ][ trafoDepth ] is equal to 1 and log2TrafoSize is greater than 3, splitChromaFlag is set equal to 1.

– Otherwise (split\_transform\_flag[ xCb + xB0 ][ yCb + yB0 ][ trafoDepth ] is equal to 0 or log2TrafoSize is equal to 3), splitChromaFlag is set equal to 0.

Depending on the value of splitChromaFlag, the following applies:

– If splitChromaFlag is equal to 1, the following ordered steps apply:

1. The variables xB1 and yB1 are derived as follows:

– The variable xB1 is set equal to xB0 + ( 1  <<  ( log2TrafoSize − 1 ) ).

– The variable yB1 is set equal to yB0 + ( 1  <<  ( log2TrafoSize − 1 ) ).

1. The decoding process for residual chroma blocks as specified in this subclause is invoked with the luma location ( xCb, yCb ), the luma location ( xB0, yB0 ), the variable log2TrafoSize set equal to log2TrafoSize − 1, the variable trafoDepth set equal to trafoDepth + 1, the variable cIdx, the variable nCbSw, the variable nCbSh, and the (nCbSw)x(nCbSh) array resSamples as inputs, and the output is a modified version of the (nCbSw)x(nCbSh) array resSamples.
2. The decoding process for residual chroma blocks as specified in this subclause is invoked with the luma location ( xCb, yCb ), the luma location ( xB1, yB0 ), the variable log2TrafoSize set equal to log2TrafoSize − 1, the variable trafoDepth set equal to trafoDepth + 1, the variable cIdx, the variable nCbSw, the variable nCbSh, and the (nCbSw)x(nCbSh) array resSamples as inputs, and the output is a modified version of the (nCbSw)x(nCbSh) array resSamples.
3. The decoding process for residual chroma blocks as specified in this subclause is invoked with the luma location ( xCb, yCb ), the luma location ( xB0, yB1 ), the variable log2TrafoSize set equal to log2TrafoSize − 1, the variable trafoDepth set equal to trafoDepth + 1, the variable cIdx, the variable nCbSw, the variable nCbSh, and the (nCbSw)x(nCbSh) array resSamples as inputs, and the output is a modified version of the (nCbSw)x(nCbSh) array resSamples.
4. The decoding process for residual chroma blocks as specified in this subclause is invoked with the luma location ( xCb, yCb ), the luma location ( xB1, yB1 ), the variable log2TrafoSize set equal to log2TrafoSize − 1, the variable trafoDepth set equal to trafoDepth + 1, the variable cIdx, the variable nCbSw, the variable nCbSh, and the (nCbSw)x(nCbSh) array resSamples as inputs, and the output is a modified version of the (nCbSw)x(nCbSh) array resSamples.

– Otherwise (splitChromaFlag is equal to 0), for the variable blkIdx proceeding over the values 0..( ChromaArrayType  = =  2 ? 1 : 0 ), the following ordered steps apply:

1. The variable nTbS is set equal to ( 1  <<  log2TrafoSize ) / SubWidthC.
2. The variable yBN is set equal to yB0 + blkIdx \* nTbS \* SubHeightC.
3. The scaling and transformation process as specified in subclause 8.6.2 is invoked with the luma location ( xCb + xB0, yCb + yBN ), the variable trafoDepth, the variable cIdx, and the transform size trafoSize set equal to nTbS as inputs, and the output is an (nTbS)x(nTbS) array transformBlock.
4. When explicit\_rdpcm\_flag[ xCb + xB0 ][ yCb + yBN ][ cIdx ] is equal to 1, the directional residual modification process for blocks using a transform bypass as specified in subclause 8.6.5 is invoked with the variable mDir set equal to explicit\_rdpcm\_dir\_flag[ xCb + xB0 ][ yCb + yBN ][ cIdx ], the variable nTbS, and the (nTbS)x(nTbS) array r set equal to the array transformBlock as inputs, and the output is a modified (nTbS)x(nTbS) array transformBlock.
5. When cross\_component\_prediction\_enabled\_flag is equal to 1, and cu\_residual\_act\_flag[ xCb ][ yCb ] is equal to 0, and ChromaArrayType is equal to 3, the residual modification process for transform blocks using cross-component prediction as specified in subclause 8.6.6 is invoked with the transform block location ( xCb + xB0, yCb + yB0 ), the variable nTbS, the variable cIdx, the (nTbS)x(nTbS) array rY set equal to the corresponding luma residual sample array transformBlock of the current transform block, and the (nTbS)x(nTbS) array r set equal to the array transformBlock as inputs, and the output is a modified (nTbS)x(nTbS) array resSamples.
6. The (nCbS)x(nCbS) residual sample array of the current coding block resSamples is modified as follows, for i = 0..nTbS − 1, j = 0..nTbS − 1:

resSamples[ ( xCb + xB0 ) / SubWidthC + i, ( yCb + yBN ) / SubHeightC + j ] = transformBlock[ i, j ] (8‑251)

1. When cu\_residual\_act\_flag[ xCb ][ yCb ] is equal to 1 and cIdx is equal to 2, the residual modification process for residual blocks using cross-component de-correlation as specified in subclause 8.6.8 is invoked with the transform block location ( xCb + xB0, yCb + yB0 ), the variable blkSize set equal to nTbS, the (nTbS)x(nTbS) array rY set equal to the corresponding luma residual sample array resSamples of the current transform block, the (nTbS)x(nTbS) array rCb set equal to the corresponding chroma residual sample array resSamples when cIdx is equal to 1 of the current transform block, and the (nTbS)x(nTbS) array rCr set equal to the array resSamples as inputs, and the output are modified versions of the three residual array.

### 8.6.8 Residual modification process for transform blocks using ~~adaptive colour transform~~ cross-component de-correlation

This process is only invoked when ChromaArrayType is equal to 3.

Inputs to this process are:

– a luma location ( xTbY, yTbY ) specifying the top-left sample of the current luma transform block relative to the top‑left luma sample of the current picture,

– a variable blkSize specifying the block size,

– an (blkSize)x(blkSize) array of luma residual samples rY with elements rY[ x ][ y ],

– an (blkSize)x(blkSize) array of chroma residual samples rCb with elements rCb[ x ][ y ],

– an (blkSize)x(blkSize) array of chroma residual samples rCr with elements rCr[ x ][ y ].

Outputs of this process are:

– an modified (blkSize)x(blkSize) array rY of luma residual samples,

– an modified (blkSize)x(blkSize) array rCb of chroma residual samples,

– an modified (blkSize)x(blkSize) array rCr of chroma residual samples.

The (blkSize)x(blkSize) arrays of residual samples rY, rCb and rCr are modified as follows:

* If cu\_transquant\_bypass\_flag is equal to 1, the (blkSize)x(blkSize) arrays of residual samples rY, rCb and rCr with  x = 0..blkSize − 1, y = 0..blkSize − 1 are modified as follows:

rCb[ x ][ y ] += ( ResScaleVal[ 1 ][ xTbY ][ yTbY ] \* ( ( rY[ x ][ y ]  <<  BitDepthC )  >>  BitDepthY ) ) >> 3

rCr[ x ][ y ] += ( ResScaleVal[ 2 ][ xTbY ][ yTbY ] \* ( ( rY[ x ][ y ]  <<  BitDepthC )  >>  BitDepthY ) ) >> 3

tmp = rY[ x ][ y ] − ( rCb[ x ][ y ]  >>  1 )  
 rY[ x ][ y ] = tmp + rCb[ x ][ y ]  
 rCb[ x ][ y ] = tmp − (  rCr[ x ][ y ]  >>  1 )  
 rCr[ x ][ y ] = rCb[ x ][ y ] + rCr[ x ][ y ]

* Otherwise (cu\_transquant\_bypass\_flag is equal to 0), the (blkSize)x(blkSize) arrays of residual samples rY, rCb and rCr with  x = 0..blkSize − 1, y = 0..blkSize − 1 are modified as follows:

bShift = BitDepthY + 3

wPara0 = 1 << bShift

wPara1 = ResScaleVal[ 1 ][ xTbY ][ yTbY ] << BitDepthc

wPara2 = ResScaleVal[ 2 ][ xTbY ][ yTbY ] << BitDepthc

resCoeffY = (wPara0 + wPara1) \* rY[ x ][ y ] + wPara0 \* rCb[ x ][ y ]

tmp0 = (wPara0 - wPara1) \* rY[ x ][ y ] - wPara0 \* rCb[ x ][ y ]

tmp1 = wPara2 \* rY[ x ][ y ] + wPara0 \* rCr[ x ][ y ]

resCoeffCb = tmp0 – tmp1

resCoeffCr = tmp0 + tmp1

rY[ x ][ y ] = resCoeffY >> bShift   
 rCb[ x ][ y ] = resCoeffCb >> bShift  
 rCr[ x ][ y ] = resCoeffCr >> bShift

~~tmp = rY[ x ][ y ] − rCb[ x ][ y ]   
rY[ x ][ y ] = rY[ x ][ y ] + rCb[ x ][ y ]  
rCb[ x ][ y ] = tmp − rCr[ x ][ y ]  
rCr[ x ][ y ] = tmp + rCr[ x ][ y ]~~