* + - 1. General Coding unit syntax

|  |  |
| --- | --- |
| coding\_unit( x0, y0, log2CbSize , ctDepth) { | **Descriptor** |
| if( transquant\_bypass\_enable\_flag ) { |  |
| **cu\_transquant\_bypass\_flag** | ae(v) |
| } |  |
| if( slice\_type != I && !MotionInhFlag[ x0 ][ y0 ]) |  |
| **skip\_flag**[ x0 ][ y0 ] | ae(v) |
| if( skip\_flag[ x0 ][ y0 ] ) |  |
| prediction\_unit( x0, y0, log2CbSize ) |  |
| else { |  |
| if( !MotionInhFlag[ x0 ][ y0 ] ) { |  |
| nCbS = ( 1 << log2CbSize ) |  |
| if( slice\_type != I ) |  |
| **pred\_mode\_flag** | ae(v) |
| if( PredMode = = MODE\_INTRA && DepthFlag ) |  |
| **sdc\_flag**[ x0 ][ y0 ] | ae(v) |
| if( sdc\_flag[ x0 ][ y0 ] ) { |  |
| **sdc\_pred\_mode** | ae(v) |
| if( sdc\_pred\_mode = = 1 ) |  |
| **wedge\_full\_tab\_idx**[ x0 ][ y0 ] | ae(v) |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
| } |  |
| } else { |  |
| if( ( PredMode[ x0 ][ y0 ] ! = MODE\_INTRA | | log2CbSize = = Log2MinCbSize ) &&  !predPartModeFlag) |  |
| **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 |  |
| if ( enable\_DMM\_flag && log2CbSize <= Log2MaxDmmCbSize ) { |  |
| for( j = 0; j <= pbOffset; j = j + pbOffset ) |  |
| for( i = 0; i <= pbOffset; i = i + pbOffset ) |  |
| **dmm\_flag[** x0 + i ][ y0+ j ] | ae(v) |
| for( j = 0; j <= pbOffset; j = j + pbOffset ) |  |
| for( i = 0; i <= pbOffset; i = i + pbOffset ) |  |
| if (dmm\_flag**[** x0 + i ][ y0+ j ] ) |  |
| **dmm\_mode[** x0 + i ][ y0+ j ] | ae(v) |
| for( j = 0; j <= pbOffset; j = j + pbOffset ) |  |
| for( i = 0; i <= pbOffset; i = i + pbOffset ) |  |
| if ( dmm\_flag**[** x0 + i ][ y0+ j ] && (  dmm\_mode**[** x0 + i ][ y0+ j ] = = MODE\_DMM\_WFULL ||   dmm\_mode**[** x0 + i ][ y0+ j ] = = MODE\_DMM\_WFULLDELTA ) ) |  |
| **wedge\_full\_tab\_idx[**x0 + i **][**y0 + i **]** | ae(v) |
| for( j = 0; j <= pbOffset; j = j + pbOffset ) |  |
| for( i = 0; i <= pbOffset; i = i + pbOffset ) |  |
| if( dmm\_flag**[** x0 + i ][ y0+ j ] && (  dmm\_mode**[** x0 + i ][ y0+ j ] = = MODE\_DMM\_WPREDDIR ||   dmm\_mode**[** x0 + i ][ y0+ j ] = = MODE\_DMM\_WPREDDIRDELTA ) ) |  |
| **dmm\_delta\_end\_flag[**x0 + i **][**y0 + i **]** | ae(v) |
| for( j = 0; j <= pbOffset; j = j + pbOffset ) |  |
| for( i = 0; i <= pbOffset; i = i + pbOffset ) |  |
| if ( dmm\_delta\_end\_flag[ x0 + i ][ y0 + i ] ) |  |
| **dmm\_delta\_end\_abs\_minus1**[ x0 + i ][ y0 + i ] | ae(v) |
| for( j = 0; j <= pbOffset; j = j + pbOffset ) |  |
| for( i = 0; i <= pbOffset; i = i + pbOffset ) |  |
| if ( dmm\_delta\_end\_flag[ x0 + i ][ y0 + i ] ) |  |
| **dmm\_delta\_end\_sign\_flag**[ x0 ][ y0 ] | ae(v) |
| for( j = 0; j <= pbOffset; j = j + pbOffset ) |  |
| for( i = 0; i <= pbOffset; i = i + pbOffset ) |  |
| DmmDeltaFlag[ x0 + i ][ y0 + i ] = ( dmm\_flag[ x0 + i ][ y0+ j ] && (  dmm\_mode[ x0 + i ][ y0 + i ] = = MODE\_DMM\_WFULLDELTA ||   dmm\_mode[ x0 + i ][ y0 + i ] = = MODE\_DMM\_WPREDDIRDELTA ||   dmm\_mode[ x0 + i ][ y0 + i ] = = MODE\_DMM\_WPREDTEXDELTA ||   dmm\_mode[ x0 + i ][ y0 + i ] = = MODE\_DMM\_CPREDTEXDELTA ) ) |  |
| for( j = 0; j <= pbOffset; j = j + pbOffset ) |  |
| for( i = 0; i <= pbOffset; i = i + pbOffset ) |  |
| if ( DmmDeltaFlag[ x0 + i ][ y0 + i ] ) |  |
| **dmm\_dc\_1\_abs**[ x0 + i ][ y0 + i ] | ae(v) |
| for( j = 0; j <= pbOffset; j = j + pbOffset ) |  |
| for( i = 0; i <= pbOffset; i = i + pbOffset ) |  |
| if ( DmmDeltaFlag[ x0 + i ][ y0+ j ] &&  dmm\_dc\_1\_abs[ x0 + i ][ y0 + i ] != 0 ) |  |
| **dmm\_dc\_1\_sign\_flag**[ x0 + i ][ y0 + i ] | ae(v) |
| for( j = 0; j <= pbOffset; j = j + pbOffset ) |  |
| for( i = 0; i <= pbOffset; i = i + pbOffset ) |  |
| if ( DmmDeltaFlag[ x0 + i ][ y0+ j ] ) |  |
| **dmm\_dc\_2\_abs**[ x0 + i ][ y0 + i ] | ae(v) |
| for( j = 0; j <= pbOffset; j = j + pbOffset ) |  |
| for( i = 0; i <= pbOffset; i = i + pbOffset ) |  |
| if ( DmmDeltaFlag[ x0 + i ][ y0+ j ] &&   dmm\_dc\_2\_abs[ x0 + i ][ y0 + i ] != 0 ) |  |
| **dmm\_dc\_2\_sign\_flag**[ x0 + i ][ y0 + i ] | ae(v) |
| } else { |  |
| for( j = 0; j <= pbOffset; j = j + pbOffset ) |  |
| for( i = 0; i <= pbOffset; i = i + pbOffset ) { |  |
| if( !dmm\_flag**[** x0 + i ][ y0+ j ] ) |  |
| **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( !dmm\_flag**[** x0 + i ][ y0+ j ] ) { |  |
| 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) |
| if( rem\_intra\_luma\_pred\_mode[ x0 + i ][ y0+ j ]  = =  31 &&   ( log2CbSize + ( PartMode = = PART\_NxN ) ? − 1 : 0 ) <= 5 &&  ( log2CbSize + ( PartMode = = PART\_NxN ) ? − 1 : 0 ) >= 2 ) { |  |
| **edge\_intra\_flag**[ x0 + i ][ y0 + j ] | ae(v) |
| if( edge\_intra\_flag[ x0 + i ][  y0+ j  ] ) { |  |
| **edge\_start\_left\_flag**[ x0 + i ][ y0+ j ] | ae(v) |
| **edge\_start\_position**[ x0 + i ][  y0+ j ] | ae(v) |
| **num\_edge\_codes\_minus1**[ x0 + i ][  y0+ j ] | ae(v) |
| for( k = 0; k <= num\_edge\_codes\_minus1; k++ ) |  |
| **edge\_code**[ k ] | ae(v) |
| **edge\_dc\_flag**[ x0 + i ][ y0+ j ] | ae(v) |
| if( edge\_dc\_flag[ x0 + i ][ y0+ j ] ) { |  |
| **edge\_dc\_1\_abs**[ x0 + i ][ y0+ j ] | ae(v) |
| if( edge\_dc\_1\_abs[ x0 + i ][ y0+ j ] != 0 ) |  |
| **edge\_dc\_1\_sign\_flag**[ x0 + i ][  y0+ j ] | ae(v) |
| **edge\_dc\_2\_abs**[ x0 + i ][ y0+ j ] | ae(v) |
| if( edge\_dc\_2\_abs[ x0 + i ][ y0+ j ] != 0 ) |  |
| **edge\_dc\_2\_sign\_flag**[ 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 ( resPredEnableFlag ) |  |
| **res\_pred\_flag** | ae(v) |
| if ( icEnableFlag ) |  |
| **ic\_flag** | ae(v) |
| if ( MotionInhFlag[ x0 ][ y0 ] && TextureCtDepth[ x0 ][ y0 ] > ctDepth) { |  |
| coding\_tree( x0, y0, log2CbSize, ctDepth ) |  |
| } else { |  |
| if( !pcm\_flag ) { |  |
| if( PredMode[ x0 ][ y0 ] != MODE\_INTRA &&   !(PartMode = = PART\_2Nx2N && merge\_flag[x0][y0]) ) |  |
| **no\_residual\_syntax\_flag** | ae(v) |
| if( !no\_residual\_syntax\_flag && !sdc\_flag) { |  |
| MaxTrafoDepth = ( PredMode[ x0 ][ y0 ] = = MODE\_INTRA ?   max\_transform\_hierarchy\_depth\_intra + IntraSplitFlag :   max\_transform\_hierarchy\_depth\_inter ) |  |
| transform\_tree( x0, y0 x0, y0, log2CbSize, 0, 0 ) |  |
| } |  |
| if( sdc\_flag ) { |  |
| sdcNumSegments = ( sdc\_pred\_mode = = 1) ? 2 : 1 ) |  |
| for ( i = 0; i < sdcNumSegments ; i++) { |  |
| **sdc\_residual\_flag**[ x0 ][ y0 ][ i | ae(v) |
| if( sdc\_residual\_flag [ x0 ][ y0 ][ i ]) { |  |
| **sdc\_residual\_sign\_flag**[ x0 ][ y0 ][ i ] | ae(v) |
| **sdc\_residual\_abs\_minus1**[ x0 ][ y0 ][ i ] | ae(v) |
| } |  |
| } |  |
| } |  |
| } |  |
| } |  |
| } |  |
| } |  |

**sdc\_pred\_mode** shall be one of the values shown in Table G‑3. sdc\_pred\_mode specifies the intra prediction mode used for simplified depth coding.

Table G‑3 – Interpretation of sdc\_pred\_mode

|  |  |
| --- | --- |
| **sdc\_pred\_mode** | Associated Intra Prediction Mode |
| 0 | Intra\_DC |
| 1 | Intra\_DepthPartition( 35 ) |
|  |  |
| 2 | Intra\_Planar |

* If sdc\_flag[ xB ][ yB ] is equal to 1, IntraPredMode[ xB ][ yB ] is derived as follows.
  + If sdc\_pred\_mode[ xB ][ yB ] is equal to 0, IntraPredMode[ xB ][ yB ] is set equal to 1.
  + Otherwise, if sdc\_pred\_mode[ xB ][ yB ] is equal to 1, IntraPredMode[ xB ][ yB ] is set equal to 35.
  + Otherwise, if sdc\_pred\_mode [ xB ][ yB ] is equal to 2, IntraPredMode[ xB ][ yB ] is set equal to 0.

Table G‑6 – Association of ctxIdx and syntax elements for each initializationType in the initialization process

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Syntax element** | **ctxIdxTable** | **initType** | | |
| **0** | **1** | **2** |
| coding\_unit() | dmm\_flag |  | 0 | 1 | 2 |
| dmm\_mode |  | 0 | 1 | 2 |
| wedge\_full\_tab\_idx |  | 0 | 1 | 2 |
| dmm\_delta\_end\_flag dmm\_delta\_end\_abs\_minus1 |  | 0 | 1 | 2 |
| dmm\_dc\_1\_abs dmm\_dc\_2\_abs |  | 0 | 1 | 2 |
| res\_pred\_flag |  | 0 | 1 | 2 |
| ic\_flag |  |  | 0 | 1 |
| mvp\_l0\_idx mpv\_l1\_idx |  |  | 0..1 | 2..3 |
| sdc\_flag |  | 0..2 | 3..5 | 6..8 |
| sdc\_residual\_flag |  | 0..1 | 2..3 | 4..5 |
| sdc\_residual\_sign\_flag |  | 0 | 1 | 2 |
| sdc\_residual\_abs\_minus1 |  | 0..19 | 20..39 | 40..59 |
| sdc\_pred\_mode |  | 0..5 | 6..11 | 12..17 |

Table G‑20 – Syntax elements and associated types of binarization, maxBinIdxCtx, ctxIdxTable, and ctxIdxOffset

| **Syntax element** | **initType** | **Type of binarization** | **maxBinIdxCtx** | **ctxIdxTable** | **ctxIdxOffset** |
| --- | --- | --- | --- | --- | --- |
| sdc\_flag | 0 | FL, cMax = 1 | 0 |  | 0 |
| 1 | 0 |  | 3 |
| 2 | 0 |  | 6 |
| sdc\_residual\_flag | 0 | FL, cMax = 1 | 0 |  | 0 |
| 1 | 0 |  | 2 |
| 2 | 0 |  | 4 |
| sdc\_residual\_sign\_flag | 0 | FL, cMax = 1 | 0 |  | 0 |
| 1 | 0 |  | 2 |
| 2 | 0 |  | 4 |
| sdc\_residual\_abs\_minus1 | 0 | FL, cMax = dltFlag ? Floor( Log2( num\_depth\_values\_in\_dlt ) : BitDepthY | 9 |  | 0 |
| 1 | 9 |  | 20 |
| 2 | 9 |  | 40 |
| sdc\_pred\_mode | 0 | U | 0 |  | 0 |
| 1 | 0 |  | 6 |
| 2 | 0 |  | 12 |

Table G‑21 – Assignment of ctxIdxInc to syntax elements with context coded bins

| **Syntax element** | **binIdx** | | | | | |
| --- | --- | --- | --- | --- | --- | --- |
| **0** | **1** | **2** | **3** | **4** | **>=5** |
| sdc\_flag | 0,1,2 (subclause G.9.3.3.1.1) | na | na | na | na | na |
| sdc\_residual\_flag | 0,1 (subclause G.9.3.3.1.7) | na | na | na | na | na |
| sdc\_residual\_sign\_flag | 0,1 (subclause G.9.3.3.1.7) | na | na | na | na | na |
| sdc\_residual\_abs\_minus1 | 0,10 (subclause G.9.3.3.1.7) | | | | | |
| sdc\_pred\_mode | 0,1,2 | 3,4,5 | na | na | na | na |
|  |  |  |  |  |  |  |

* + - * 1. Derivation process of ctxIdxInc using left and above syntax elements

The specifications in subclause 9.3.3.1.1 apply with the following modifications.

– Table G‑22 is appended to the end of Table 9-38.

Table G‑22 – Specification of ctxIdxInc using left and above syntax elements

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

* + - * 1. .Derivation process of ctxIdxInc for the syntax elements last\_significant\_coeff\_x\_prefix and last\_significant\_coeff\_y\_prefix

The specifications in subclause 9.3.3.1.2 apply.

* + - * 1. Derivation process of ctxIdxInc using segment index

Input to this process is the luma location ( xC, yC ) specifying the top-left luma sample of the current luma coding block relative to the top-left sample of the current picture.

Output of this process is ctxIdxInc.

If IntraPredMode[ xC ][ yC ] is not equal to 0 and IntraPredMode[ xC ][ yC ] is not equal to 1, ctxIdxInc depends on the SdcSegmentIndex = 0..1, as specified in Table G‑23.

Otherwise, the ctxIdxInc is equal to 0.

Table G‑23 – Specification of ctxIdxInc using segment index

|  |  |  |  |
| --- | --- | --- | --- |
| **Syntax element** | **IntraPredMode** | **SdcSegmentIndex** | **ctxIdxInc** |
| sdc\_residual\_flag | IntraPredMode[ xC ][ yC ]  ==  0 || IntraPredMode[ xC ][ yC ]  ==  1 | 0 | 0 |
| IntraPredMode[ xC ][ yC ]  !=  0 && IntraPredMode[ xC ][ yC ]  !=  1 | 0 | 0 |
| 1 | 1 |
| sdc\_residual\_sign\_flag | IntraPredMode[ xC ][ yC ]  ==  0 || IntraPredMode[ xC ][ yC ]  ==  1 | 0 | 0 |
| IntraPredMode[ xC ][ yC ]  !=  0 && IntraPredMode[ xC ][ yC ]  !=  1 | 0 | 0 |
| 1 | 1 |
| sdc\_residual\_abs\_minus1 | IntraPredMode[ xC ][ yC ]  ==  0 || IntraPredMode[ xC ][ yC ]  ==  1 | 0 | 0 |
| IntraPredMode[ xC ][ yC ]  !=  0 && IntraPredMode[ xC ][ yC ]  !=  1 | 0 | 0 |
| 1 | 10 |