

## PU structure for intra NxN

**JCTVC-H202**

Hiroya Nakamura, Shigeru Fukushima

JVC KENWOOD Corporation



# 1. Overview

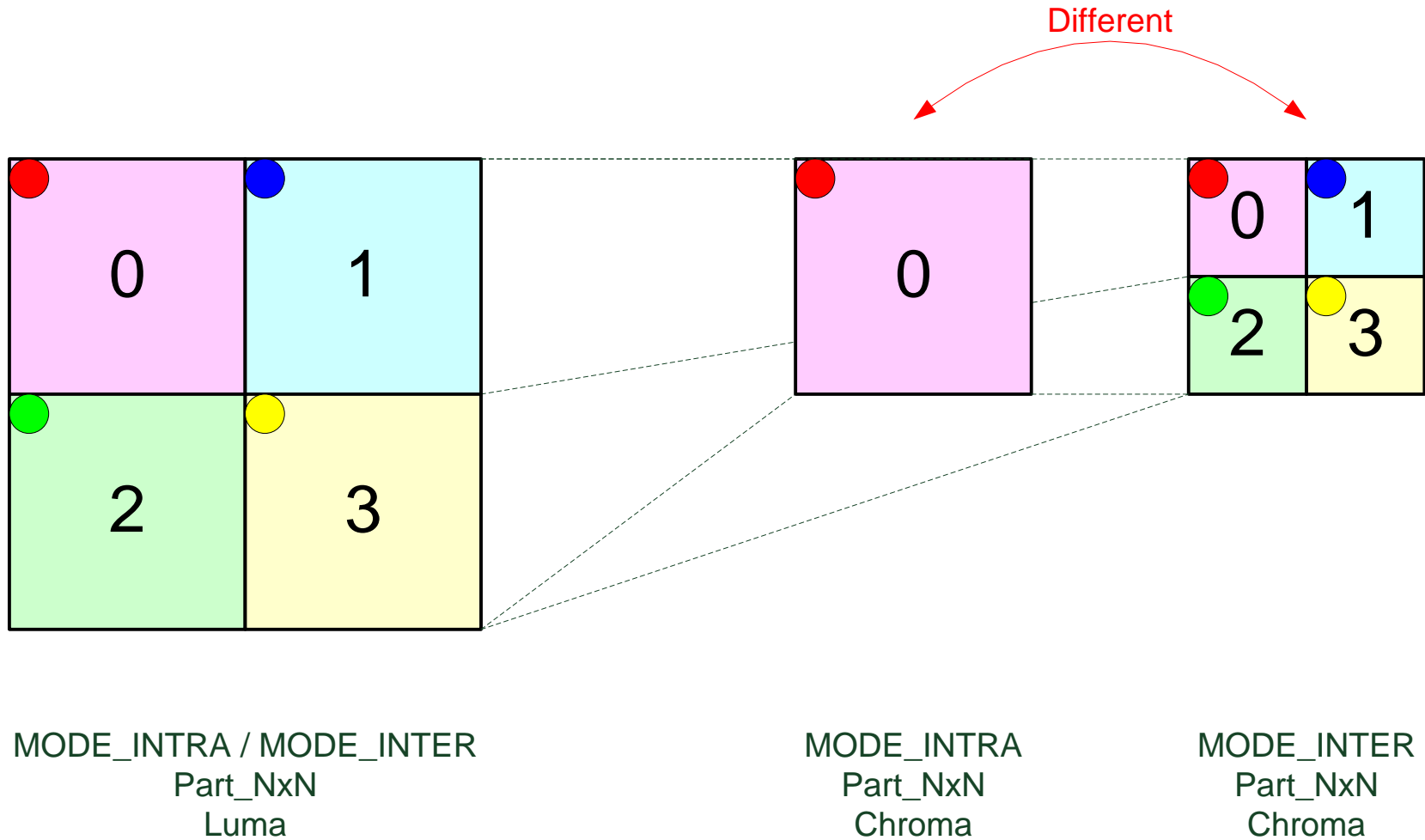
# Overview

- Current PU structure
  - In HM5.0
  - In WD5
- Proposed PU structure
  - In 4:2:0 sampling
  - In 4:2:2 and 4:4:4 sampling
- Cross-check
  - JCTVC-H0638 by Samsung
- Simulation results
  - No coding loss for AIHE/AILC settings



## 2. Current PU structure

# Intra and inter NxN PU structure in 4:2:0



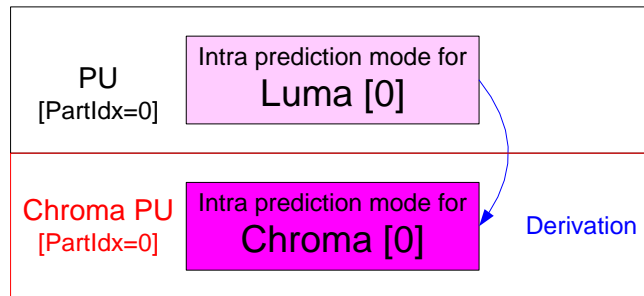
# The function of entropy decoding process in HM5.0 implementation

```
Void TDecEntropy::decodePredInfo    ( TComDataCU* pcCU, UInt uiAbsPartIdx, UInt uiDepth, TComDataCU*
    pcSubCU )
{
    .....
    if( eMode == SIZE_NxN )    // if it is NxN size, encode 4 intra directions.
    {
        .....
        // if it is NxN size, this size might be the smallest partition size.
        decodeIntraDirModeLuma( pcCU, uiAbsPartIdx,                uiDepth+1 );    PartIdx = 0 Luma
        decodeIntraDirModeLuma( pcCU, uiAbsPartIdx + uiPartOffset, uiDepth+1 );    PartIdx = 1 Luma
        decodeIntraDirModeLuma( pcCU, uiAbsPartIdx + uiPartOffset*2, uiDepth+1 );    PartIdx = 2 Luma
        decodeIntraDirModeLuma( pcCU, uiAbsPartIdx + uiPartOffset*3, uiDepth+1 );    PartIdx = 3 Luma
        decodeIntraDirModeChroma( pcCU, uiAbsPartIdx,                uiDepth );    PartIdx = 0 Chroma
    }
    else    // if it is not NxN size, encode 1 intra directions
    {
        decodeIntraDirModeLuma ( pcCU, uiAbsPartIdx, uiDepth );
        decodeIntraDirModeChroma( pcCU, uiAbsPartIdx, uiDepth );
    }

    .....
}
```

# PU structure in HM5.0 implementation - 1

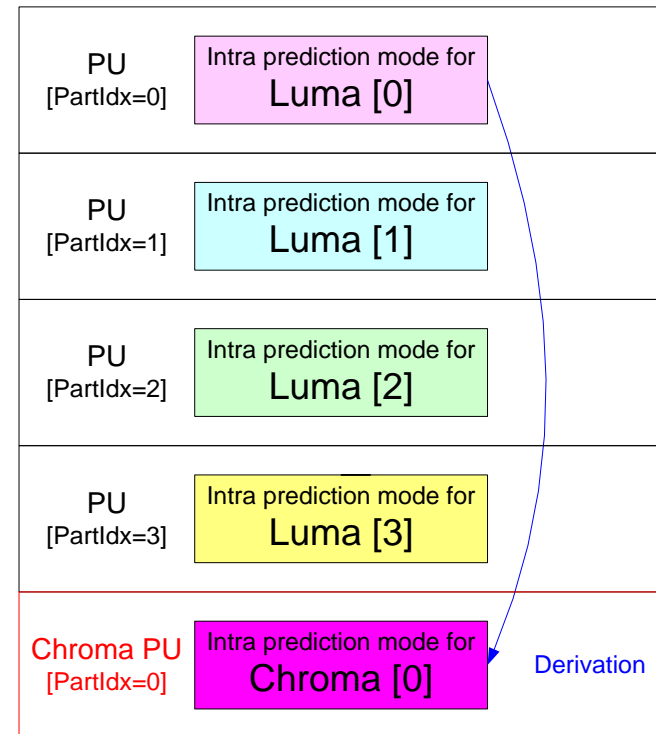
## MODE\_INTRA Part\_2Nx2N



It is not sure how the PU structure is defined in HM 5.0.

The PU for intra chroma is defined independently.

## MODE\_INTRA Part\_NxN



# CU and PU Syntax in HM5.0 implementation - 1

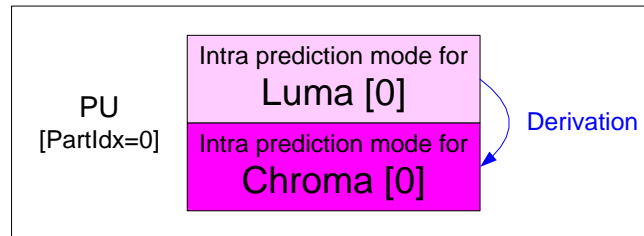
	Descriptor
<code>coding_unit( x0, y0, log2CUSize ) {</code>	
<code>if( slice_type != 1 )</code>	
<code>skip_flag[ x0 ][ y0 ]</code>	ae(v)
<code>IntraChroma = 0</code>	
<code>if( skip_flag[ x0 ][ y0 ] )</code>	
<code>prediction_unit( x0, y0, log2CUSize )</code>	
<code>else if( slice_type != 1    log2CUSize == Log2MinCUSize ) {</code>	
<code>if( slice_type != 1 )</code>	
<code>pred_mode_flag</code>	ae(v)
<code>if( PredMode != MODE_INTRA    log2CUSize == Log2MinCUSize )</code>	
<code>part_mode</code>	ae(v)
<code>x1 = x0 + ( ( 1 &lt;&lt; log2CUSize ) &gt;&gt; 1 )</code>	
<code>y1 = y0 + ( ( 1 &lt;&lt; log2CUSize ) &gt;&gt; 1 )</code>	
<code>.....</code>	
<code>if( PartMode == PART_2Nx2N ) {</code>	
<code>prediction_unit( x0, y0, log2CUSize )</code>	
<code>if( PredMode == MODE_INTRA )</code>	
<code>IntraChroma = 1</code>	
<code>prediction_unit( x0, y0, log2CUSize ) /* for chroma intra pred mode */</code>	
<code>}</code>	
<code>} else if( PartMode == PART_2NxN ) {</code>	
<code>prediction_unit( x0, y0, log2CUSize )</code>	
<code>prediction_unit( x0, y1, log2CUSize )</code>	
<code>} else if( PartMode == PART_Nx2N ) {</code>	
<code>prediction_unit( x0, y0, log2CUSize )</code>	
<code>prediction_unit( x1, y0, log2CUSize )</code>	
<code>} else if( PartMode == PART_2NxN ) {</code>	
<code>.....</code>	
<code>} else { /* PART_NxN */</code>	
<code>prediction_unit( x0, y0, log2CUSize )</code>	
<code>prediction_unit( x1, y0, log2CUSize )</code>	
<code>prediction_unit( x0, y1, log2CUSize )</code>	
<code>prediction_unit( x1, y1, log2CUSize )</code>	
<code>if( PredMode == MODE_INTRA )</code>	
<code>IntraChroma = 1</code>	
<code>prediction_unit( x0, y0, log2CUSize ) /* for chroma intra pred mode */</code>	
<code>}</code>	
<code>if( !pcm_flag ) {</code>	
<code>transform_tree( x0, y0, log2CUSize, log2CUSize, log2CUSize, 0, 0 )</code>	
<code>transform_coeff( x0, y0, x0, y0, log2CUSize, log2CUSize, 0, 0 )</code>	
<code>}</code>	
<code>}</code>	

	Descriptor
<code>prediction_unit( x0, y0, log2CUSize ) {</code>	
<code>if( skip_flag[ x0 ][ y0 ] ) {</code>	
<code>if( MaxNumMergeCand &gt; 1 )</code>	
<code>merge_idx[ x0 ][ y0 ]</code>	ae(v)
<code>} else if( PredMode == MODE_INTRA ) {</code>	
<code>if( PartMode == PART_2Nx2N &amp;&amp; pcm_enabled_flag &amp;&amp;</code>	
<code>log2CUSize &gt;= Log2MinIPCMCUSize &amp;&amp;</code>	
<code>log2CUSize &lt;= Log2MaxIPCMCUSize &amp;&amp;</code>	
<code>IntraChroma == 0 )</code>	
<code>pcm_flag</code>	ae(v)
<code>if( pcm_flag ) {</code>	
<code>if ( IntraChroma == 0 ) {</code>	
<code>while ( !byte_aligned( ) )</code>	
<code>pcm_alignment_zero_bit</code>	u(v)
<code>for( i = 0; i &lt; 1 &lt;&lt; ( log2CUSize &lt;&lt; 1 ); i++ )</code>	
<code>pcm_sample_luma[ i ]</code>	u(v)
<code>} else { /* IntraChroma == 1 */</code>	
<code>for( i = 0; i &lt; ( 1 &lt;&lt; ( log2CUSize &lt;&lt; 1 ) ) &gt;&gt; 1; i++ )</code>	
<code>pcm_sample_chroma[ i ]</code>	u(v)
<code>}</code>	
<code>} else {</code>	
<code>if ( IntraChroma == 0 ) {</code>	
<code>prev_intra_luma_pred_flag[ x0 ][ y0 ]</code>	ae(v)
<code>if( prev_intra_luma_pred_flag[ x0 ][ y0 ] )</code>	
<code>mpm_flag[ x0 ][ y0 ]</code>	ae(v)
<code>else</code>	
<code>rem_intra_luma_pred_mode[ x0 ][ y0 ]</code>	ae(v)
<code>} else { /* IntraChroma == 1 */</code>	
<code>intra_chroma_pred_mode[ x0 ][ y0 ]</code>	ae(v)
<code>SignaledAsChromaDC =</code>	
<code>( chroma_pred_from_luma_enabled_flag ?</code>	
<code>intra_chroma_pred_mode[ x0 ][ y0 ] == 3 :</code>	
<code>intra_chroma_pred_mode[ x0 ][ y0 ] == 2 )</code>	
<code>}</code>	
<code>}</code>	
<code>} else { /* MODE_INTER */</code>	
<code>.....</code>	
<code>}</code>	
<code>}</code>	

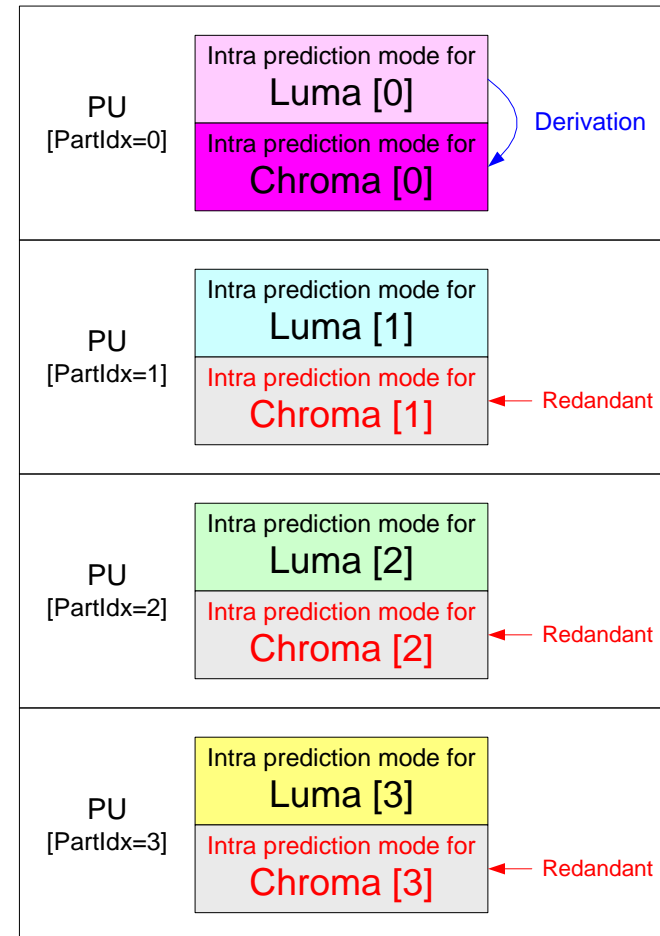


# PU structure in WD5

MODE\_INTRA  
Part\_2Nx2N



MODE\_INTRA  
Part\_NxN



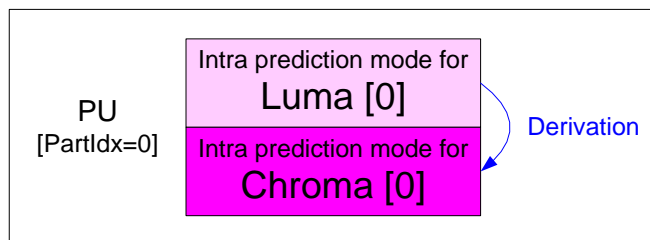
It seems that  
redundant syntax elements are coded in WD5.



## 3. Proposed PU structure

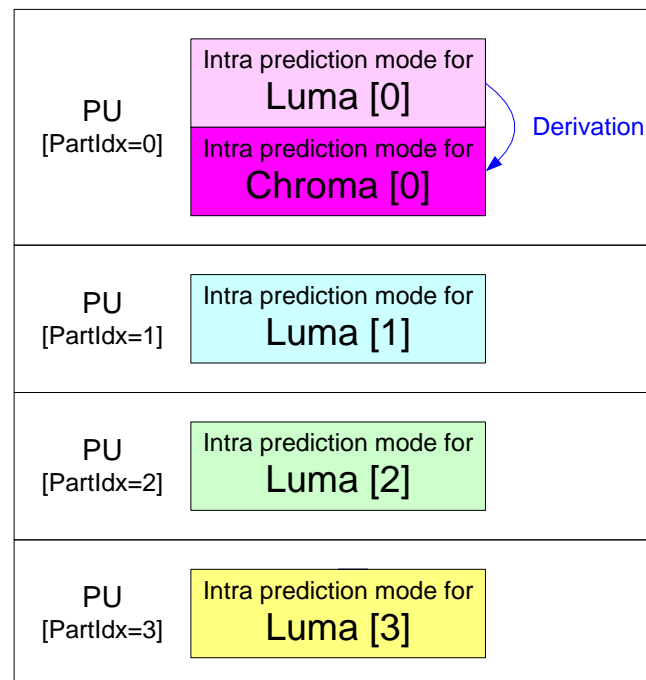
# Proposed PU structure

MODE\_INTRA  
Part\_2Nx2N



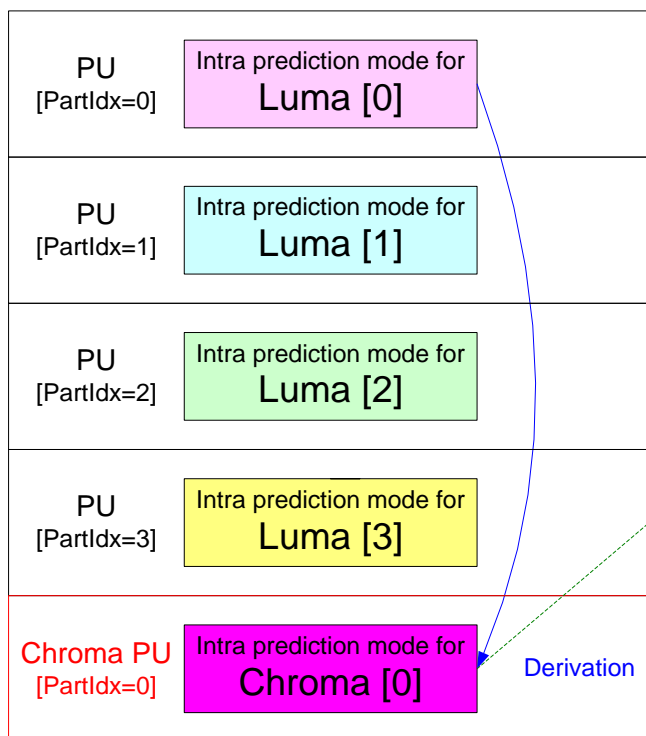
- The chroma intra prediction mode is
- coded after the luma intra prediction mode in the same PU.
  - derived using luma intra prediction mode in the same PU.

MODE\_INTRA  
Part\_NxN

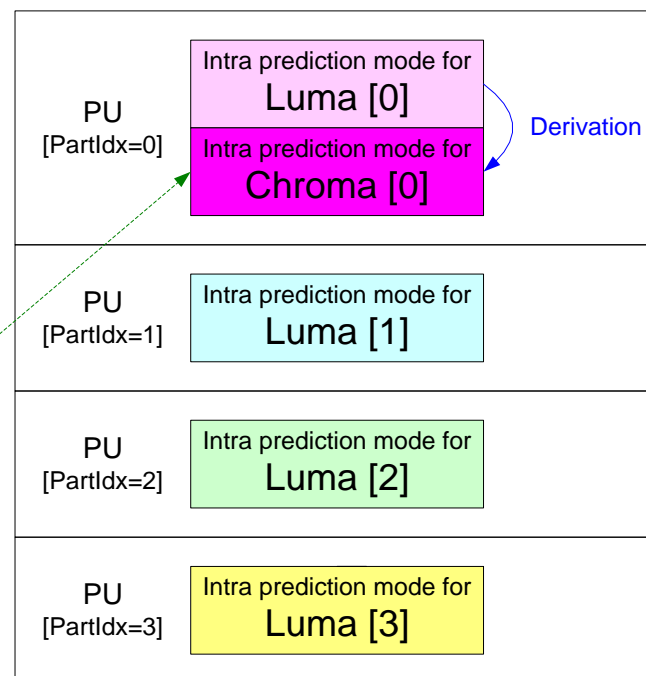


# Proposed PU structure

HM5.0

MODE\_INTRA  
Part\_NxN

Next HM


MODE\_INTRA  
Part\_NxN

# The function of entropy decoding process in Proposed HM implementation

```
Void TDecEntropy::decodePredInfo    ( TComDataCU* pcCU, UInt uiAbsPartIdx, UInt uiDepth, TComDataCU*
    pcSubCU )
{
    .....

    if( eMode == SIZE_NxN )    // if it is NxN size, encode 4 intra directions.
    {
        .....
        // if it is NxN size, this size might be the smallest partition size.
        decodeIntraDirModeLuma( pcCU, uiAbsPartIdx,                uiDepth+1 );    PartIdx = 0 Luma
        decodeIntraDirModeChroma( pcCU, uiAbsPartIdx,            uiDepth );    PartIdx = 0 Chroma
        decodeIntraDirModeLuma( pcCU, uiAbsPartIdx + uiPartOffset, uiDepth+1 );    PartIdx = 1 Luma
        decodeIntraDirModeLuma( pcCU, uiAbsPartIdx + uiPartOffset*2, uiDepth+1 );    PartIdx = 2 Luma
        decodeIntraDirModeLuma( pcCU, uiAbsPartIdx + uiPartOffset*3, uiDepth+1 );    PartIdx = 3 Luma
        decodeIntraDirModeChroma( pcCU, uiAbsPartIdx,            uiDepth );    PartIdx = 0 Chroma
    }
    else    // if it is not NxN size, encode 1 intra directions
    {
        decodeIntraDirModeLuma ( pcCU, uiAbsPartIdx, uiDepth );
        decodeIntraDirModeChroma( pcCU, uiAbsPartIdx, uiDepth );
    }

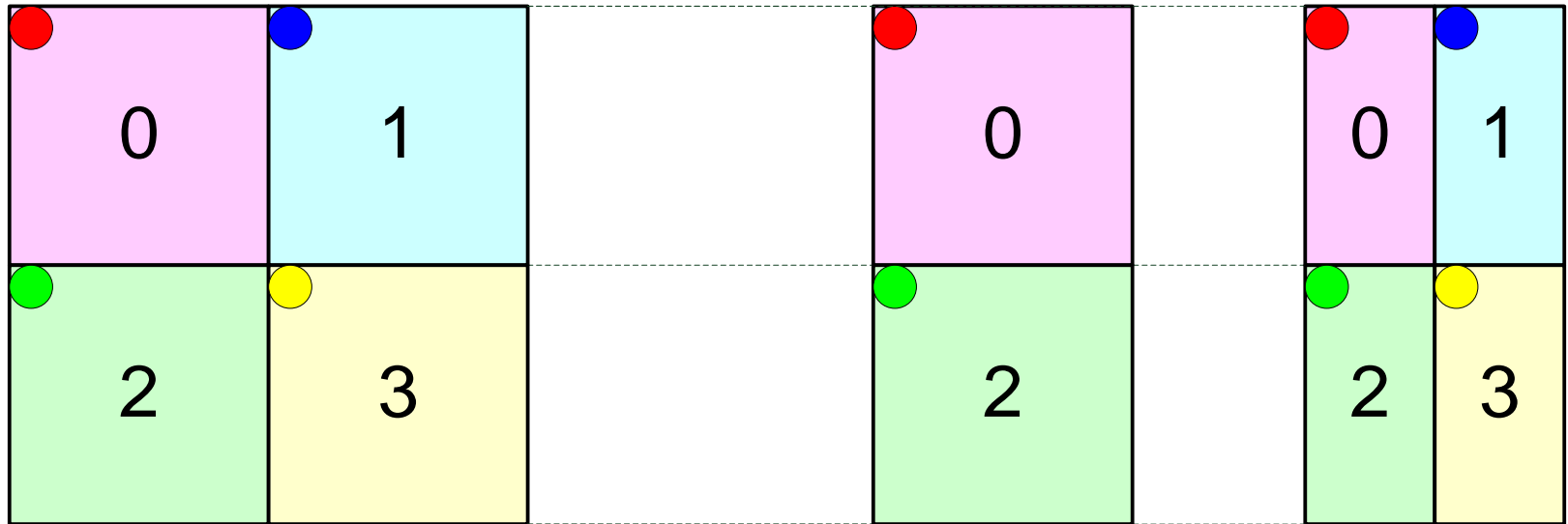
    .....
}
```



# Proposed Prediction Unit Syntax

	Descriptor
<code>prediction_unit( x0, y0, log2CUSize ) {</code>	
<code>if( skip_flag[ x0 ][ y0 ] ) {</code>	
<code>if( MaxNumMergeCand &gt; 1 )</code>	
<code>merge_idx[ x0 ][ y0 ]</code>	ae(v)
<code>} else if( PredMode == MODE_INTRA ) {</code>	
<code>if( PartMode == PART_2Nx2N &amp;&amp;</code>	
<code>log2CUSize &gt;= Log2MinIPCMCUSize )</code>	
<code>pcm_flag</code>	ae(v)
<code>if( pcm_flag ) {</code>	
<code>while ( !byte_aligned( ) )</code>	
<code>pcm_alignment_zero_bit</code>	u(v)
<code>for( i = 0; i &lt; 1 &lt;&lt; ( log2CUSize &lt;&lt; 1 ); i++ )</code>	
<code>pcm_sample_luma[ i ]</code>	u(v)
<code>for( i = 0; i &lt; ( 1 &lt;&lt; ( log2CUSize &lt;&lt; 1 ) ) &gt;&gt; 1; i++ )</code>	
<code>pcm_sample_chroma[ i ]</code>	u(v)
<code>} else {</code>	
<code>prev_intra_luma_pred_flag[ x0 ][ y0 ]</code>	ae(v)
<code>if( prev_intra_luma_pred_flag[ x0 ][ y0 ] )</code>	
<code>mpm_flag[ x0 ][ y0 ]</code>	ae(v)
<code>else</code>	
<code>rem_intra_luma_pred_mode[ x0 ][ y0 ]</code>	ae(v)
<code>if( PartIdx == 0 ) /* Part 2Nx2N and PART NxN */</code>	
<code>intra_chroma_pred_mode[ x0 ][ y0 ]</code>	ae(v)
<code>SignaledAsChromaDC =</code>	
<code>( chroma_pred_from_luma_enabled_flag ?</code>	
<code>intra_chroma_pred_mode[ x0 ][ y0 ] == 3 :</code>	
<code>intra_chroma_pred_mode[ x0 ][ y0 ] == 2 )</code>	
<code>}</code>	
<code>}</code>	
<code>} else { /* MODE_INTER */</code>	
<code>.....</code>	
<code>}</code>	
<code>}</code>	

# Intra and inter NxN PU structure in a coding unit in 4:2:2 chroma sampling

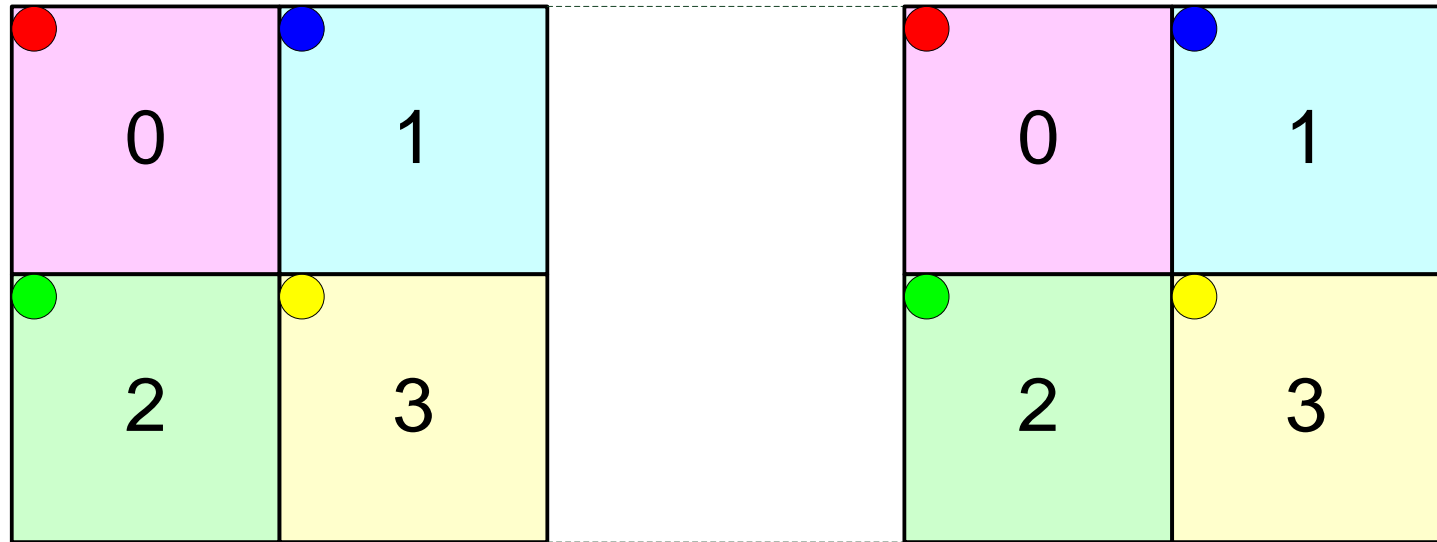


MODE\_INTRA / MODE\_INTER  
Part\_NxN  
Luma

MODE\_INTRA  
Part\_NxN  
Chroma

MODE\_INTER  
Part\_NxN  
Chroma

# Intra and inter NxN PU structure in a coding unit in 4:4:4 chroma sampling



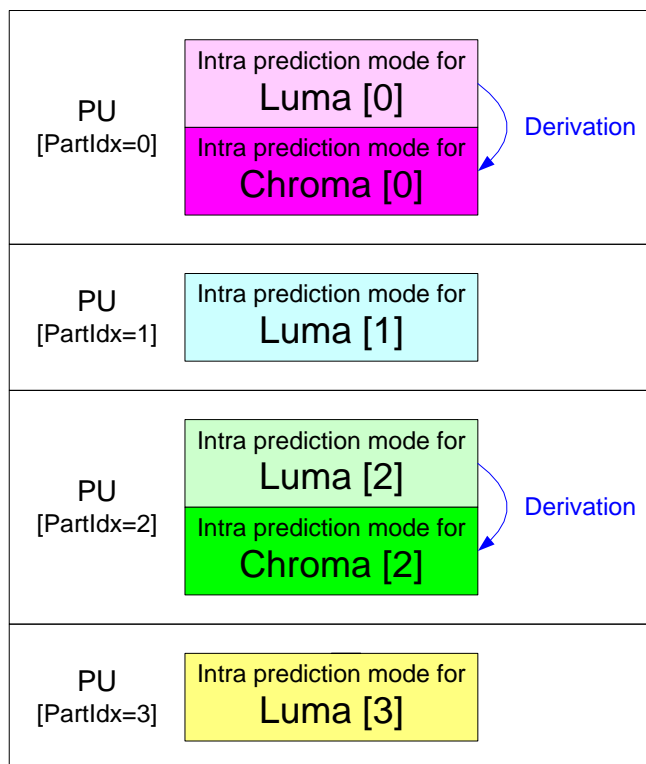
MODE\_INTRA / MODE\_INTER  
Part\_NxN  
Luma

MODE\_INTRA / MODE\_INTER  
Part\_NxN  
Chroma

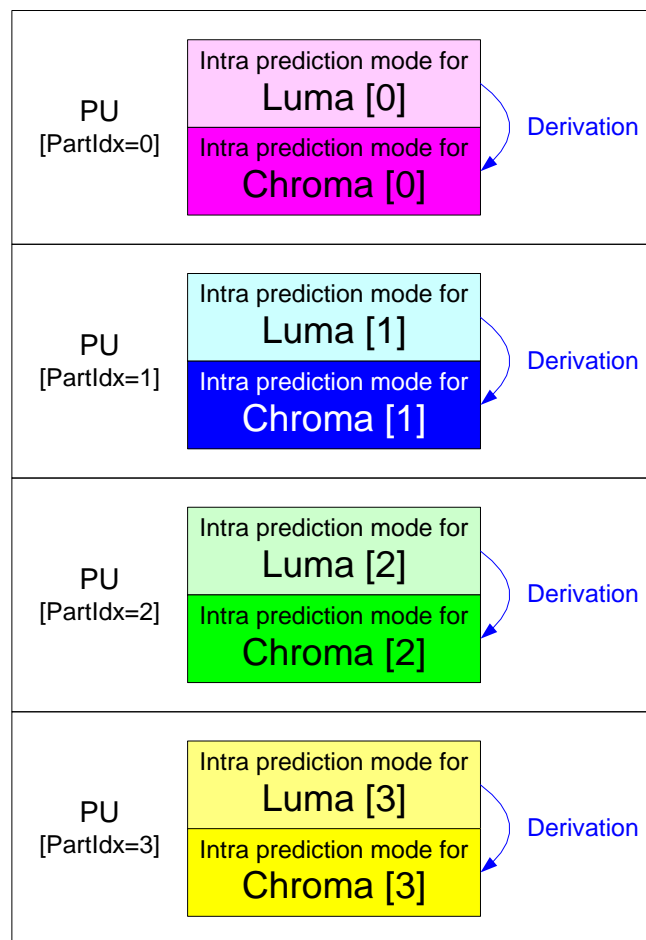


# Intra NxN PU structure in a coding unit in 4:2:2 and 4:4:4 chroma sampling

MODE\_INTRA  
Part\_NxN  
4:2:2



MODE\_INTRA  
Part\_NxN  
4:4:4



for 4:0:0, 4:2:2 and 4:4:4 sampling

	Descriptor
<b>prediction_unit</b> ( x0, y0, log2CUSize ) {	
if( skip_flag[ x0 ][ y0 ] ) {	
if( MaxNumMergeCand > 1 )	
<b>merge_idx</b> [ x0 ][ y0 ]	ae(v)
} else if( PredMode == MODE_INTRA ) {	
if( PartMode == PART_2Nx2N && log2CUSize >= Log2MinIPCMCSize )	
<b>pcm_flag</b>	ae(v)
if( pcm_flag ) {	
while ( !byte_aligned( ) )	
<b>pcm_alignment_zero_bit</b>	u(v)
for( i = 0; i < 1 << ( log2CUSize << 1 ); i++ )	
<b>pcm_sample_luma</b> [ i ]	u(v)
for( i = 0; i < ( 1 << ( log2CUSize << 1 ) ) >> 1; i++ )	
<b>pcm_sample_chroma</b> [ i ]	u(v)
} else {	
<b>prev_intra_luma_pred_flag</b> [ x0 ][ y0 ]	ae(v)
if( prev_intra_luma_pred_flag[ x0 ][ y0 ] )	
<b>mpm_flag</b> [ x0 ][ y0 ]	ae(v)
else	
<b>rem_intra_luma_pred_mode</b> [ x0 ][ y0 ]	ae(v)
if( ( ChromaArrayType != 0 && PartIdx == 0 )    ( ChromaArrayType == 2 && PartIdx == 2 )    ChromaArrayType == 3 )	
<b>intra_chroma_pred_mode</b> [ x0 ][ y0 ]	ae(v)
SignaledAsChromaDC = ( chroma_pred_from_luma_enabled_flag ? intra_chroma_pred_mode[ x0 ][ y0 ] == 3 : intra_chroma_pred_mode[ x0 ][ y0 ] == 2 )	
}	
}	
} else { /* MODE_INTER */	
.....	
}	
}	



# 4. Experiments

# Overview

- Implementation software: HM5.0
- Simulation Condition: AIHE, AILC
- Cross-check: JCTVC-H0638 by Samsung
- Simulation results

	All Intra HE			All Intra LC		
	Y	U	V	Y	U	V
Class A (8bit)	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Class B	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
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%	0.0%	0.0%	0.0%	0.0%
<b>Overall</b>	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Class F	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Enc Time[%]	100%			100%		
Dec Time[%]	100%			100%		

# 5

## 5. Conclusion

# Conclusions

Our Recommendations are as follows:

- The **Prediction Unit Syntax** described in CD should correspond exactly to next HM **by the smallest possible change**.
- The proposed PU structure is adopted in CD and next HM.
  - CD: **Add the conditional statement for intra NxN in Prediction Unit Syntax**
  - HM: **Modify the coding order of intra prediction mode**

**JVC KENWOOD**