


## 【H.264/AVC视频编解码技术详解】十二、解析H.264码流的宏块结构（下）：H.264帧内编码宏块的预测结构

---

 [blog.csdn.net/shagoneal/article/details/53959053](https://blog.csdn.net/shagoneal/article/details/53959053)

《H.264/AVC视频编解码技术详解》视频教程已经在“CSDN学院”上线，视频中详述了H.264的背景、标准协议和实现，并通过一个实战工程的形式对H.264的标准进行解析和实现，欢迎观看！

---

“纸上得来终觉浅，绝知此事要躬行”，只有自己按照标准文档以代码的形式操作一遍，才能对视频压缩编码标准的思想和方法有足够深刻的理解和体会！

---

链接地址：[H.264/AVC视频编解码技术详解](#)

---

GitHub代码地址：[点击这里](#)

---

在以H.264格式编码的视频码流中，宏块结构必然包含预测结构（I\_PCM模式除外），该结构中包含了像素块的预测模式等信息。对于不同预测模式的宏块，其预测结构是不同的。从上篇的宏块结构中，可以看出，对于部分模式，预测信息保存于mb\_pred()结构中，而对于另一部分模式则采用sub\_mb\_pred()结构。

macroblock_layer( ) {	<b>C</b>	<b>Descriptor</b>
<b>mb_type</b>	2	ue(v)   ae(v)
if( mb_type == I_PCM ) {		
while( !byte_aligned( ) )		
<b>pcm_alignment_zero_bit</b>	3	f(1)
for( i = 0; i < 256; i++ )		
<b>pcm_sample_luma[ i ]</b>	3	u(v)
for( i = 0; i < 2 * MbWidthC * MbHeightC; i++ )		
<b>pcm_sample_chroma[ i ]</b>	3	u(v)
} else {		
noSubMbPartSizeLessThan8x8Flag = 1		
if( mb_type != I_NxN && MbPartPredMode( mb_type, 0 ) != Intra_16x16 && NumMbPart( mb_type ) == 4 ) {		
sub_mb_pred( mb_type )	2	
for( mbPartIdx = 0; mbPartIdx < 4; mbPartIdx++ )		
if( sub_mb_type[ mbPartIdx ] != B_Direct_8x8 ) {		
if( NumSubMbPart( sub_mb_type[ mbPartIdx ] ) > 1 )		
noSubMbPartSizeLessThan8x8Flag = 0		
} else if( !direct_8x8_inference_flag )		
noSubMbPartSizeLessThan8x8Flag = 0		
} else {		
if( transform_8x8_mode_flag && mb_type == I_NxN )		
<b>transform_size_8x8_flag</b>	2	u(1)   ae(v)
mb_pred( mb_type )	2	
}		
if( MbPartPredMode( mb_type, 0 ) != Intra_16x16 ) {		
<b>coded_block_pattern</b>	2	me(v)   ae(v)
if( CodedBlockPatternLuma > 0 && transform_8x8_mode_flag && mb_type != I_NxN && noSubMbPartSizeLessThan8x8Flag && ( mb_type != B_Direct_16x16    direct_8x8_inference_flag ) )		
<b>transform_size_8x8_flag</b>	2	u(1)   ae(v)
}		
if( CodedBlockPatternLuma > 0    CodedBlockPatternChroma > 0    MbPartPredMode( mb_type, 0 ) == Intra_16x16 ) {		
<b>mb_qp_delta</b>	2	se(v)   ae(v)
residual( 0, 15 )	3   4	
}		
}		
}		
}		

<https://blog.csdn.net/shaqoneal>

在我们本系列的H.264分析器SimpleH264Analyzer项目中默认的全I帧测试码流中，我们所分析的第一个IDR帧的第一个宏块，其mb\_type为I\_NxN。实际上，对于除了I\_PCM模式之外的所有Intra宏块，其预测结构均采用mb\_pred()结构。

在标准文档中，mb\_pred()的定义如下表所示（只看Intra模式下）：

mb_pred( mb_type ) {	C	Descriptor
if( MbPartPredMode( mb_type, 0 ) == Intra_4x4    MbPartPredMode( mb_type, 0 ) == Intra_8x8    MbPartPredMode( mb_type, 0 ) == Intra_16x16 ) {		
if( MbPartPredMode( mb_type, 0 ) == Intra_4x4 )		
for( luma4x4BlkIdx=0; luma4x4BlkIdx<16; luma4x4BlkIdx++ ) {		
<b>prev_intra4x4_pred_mode_flag[ luma4x4BlkIdx ]</b>	2	u(1)   ae(v)
if( !prev_intra4x4_pred_mode_flag[ luma4x4BlkIdx ] )		
<b>rem_intra4x4_pred_mode[ luma4x4BlkIdx ]</b>	2	u(3)   ae(v)
}		
if( MbPartPredMode( mb_type, 0 ) == Intra_8x8 )		
for( luma8x8BlkIdx=0; luma8x8BlkIdx<4; luma8x8BlkIdx++ ) {		
<b>prev_intra8x8_pred_mode_flag[ luma8x8BlkIdx ]</b>	2	u(1)   ae(v)
if( !prev_intra8x8_pred_mode_flag[ luma8x8BlkIdx ] )		
<b>rem_intra8x8_pred_mode[ luma8x8BlkIdx ]</b>	2	u(3)   ae(v)
}		
if( ChromaArrayType == 1    ChromaArrayType == 2 )		
<b>intra_chroma_pred_mode</b>	2	ue(v)   ae(v)

从表中可以看出，Intra预测模式的结构主要有两组，分别表示4×4和8×8模式，每一组包含两个元素，分别表示预测模式标识位和预测模式值，以及最后的色度分量预测模式。

- **prev\_intra4x4\_pred\_mode\_flag和prev\_intra8x8\_pred\_mode\_flag**：表示帧内预测模式预测标识。如果该标识位为1，表示帧内预测模式的预测值就是实际的模式，否则就需要另外传递实际的帧内预测模式。
- **prev\_intra4x4\_pred\_mode\_flag和prev\_intra8x8\_pred\_mode\_flag**：表示额外传递的实际帧内预测模式。
- **intra\_chroma\_pred\_mode**：表示色度分量的预测模式，取值范围为[0,3]，分别代表DC、水平、垂直和平面模式。

在我们的demo中解析这部分的代码以下面的代码段实现：

```

if (m_mb_type == 25)
{
    // To do: I-PCM mode...
}
else if (m_mb_type == 0)
{
    // Intra_NxN mode...
    if (m_pps_active->Get_transform_8x8_mode_flag())
    {
        m_transform_size_8x8_flag = Get_bit_at_position(m_pSODB, m_byeOffset, m_bitOffset);
    }

    // Get prediction-block num...
    if (m_transform_size_8x8_flag)
    {
        // Using intra_8x8
        m_pred_struct = new IntraPredStruct[4];
    }
}

```

```

for (int luma8x8BlkIdx = 0; luma8x8BlkIdx < 4; luma8x8BlkIdx++)
{
    m_pred_struct[luma8x8BlkIdx].block_mode = 1;
    m_pred_struct[luma8x8BlkIdx].prev_intra_pred_mode_flag = Get_bit_at_position(m_pSODB,
m_byeOffset, m_bitOffset);
    if (!m_pred_struct[luma8x8BlkIdx].prev_intra_pred_mode_flag)
    {
        m_pred_struct[luma8x8BlkIdx].rem_intra_pred_mode = Get_uint_code_num(m_pSODB,
m_byeOffset, m_bitOffset, 3);
    }
}
else
{
    // Using intra_4x4
    m_pred_struct = new IntraPredStruct[16];
    for (int luma4x4BlkIdx = 0; luma4x4BlkIdx < 16; luma4x4BlkIdx++)
    {
        m_pred_struct[luma4x4BlkIdx].block_mode = 0;
        m_pred_struct[luma4x4BlkIdx].prev_intra_pred_mode_flag = Get_bit_at_position(m_pSODB,
m_byeOffset, m_bitOffset);
        if (!m_pred_struct[luma4x4BlkIdx].prev_intra_pred_mode_flag)
        {
            m_pred_struct[luma4x4BlkIdx].rem_intra_pred_mode = Get_uint_code_num(m_pSODB,
m_byeOffset, m_bitOffset, 3);
        }
    }
}

// intra_chroma_pred_mode
m_intra_chroma_pred_mode = Get_uev_code_num(m_pSODB, m_byeOffset, m_bitOffset);
}
else
{
    // To do: Intra_16x16 mode
}

    • 1
    • 2
    • 3
    • 4
    • 5
    • 6
    • 7
    • 8
    • 9
    • 10
    • 11
    • 12
    • 13
    • 14
    • 15
    • 16
    • 17
    • 18

```

- 19
- 20
- 21
- 22
- 23
- 24
- 25
- 26
- 27
- 28
- 29
- 30
- 31
- 32
- 33
- 34
- 35
- 36
- 37
- 38
- 39
- 40
- 41
- 42
- 43
- 44
- 45
- 46
- 47
- 48
- 49

更详细的信息可以到github下载完整的工程：

<https://github.com/yinwenjie/SimpleH264Analyzer>