CTU：coding tree unit，编码树单元，LCU

对于YUV=420格式的彩色视频：一个CTU由一个CTB of the luma samples 、2个CTBs of the choma samples和相关的语法元素组成。Luma CTB是一个2^N x 2^N的像素区域，而相应的Choma CTB是2^(N-1) x 2^(N-1)的像素区域，N的值在编码器中确定，并在SPS(sequence parameter set)中传输。N可选4，5，6，表示CTU的大小可取16、32、64。

CTU相当于H.264中的MarcoBlock划分图片的概念，是在编码过程中的独立编码单位，然后可以递归划分成CU。

CU：coding unit，编码单元

每一个CTU，可以进一步均匀划分成4个square CUs，一个CU又可以递归按四叉树结构划分成4个小的CUs。对于YUV=420的彩色视频：一个CU由一个CB of the luma samples、2个CBs of the choma samples和相关的语法元素。一个Luma CB是2^N x 2^N（此处的N与CTU中的N大小不同）的像素区域，而相应的choma CB是2^(N-1) x 2^(N-1)的像素区域，N的值同样在编码器中确定，并在SPS中传输。

编码时，在CTU level，通过传输split_cu_flags标志指明CTU是否进一步划分成四个CU。类似地，对于一个CU，也通过一个split_cu_flags标志指明是否进一步划分成子CU。CU通过split_cu_flags标志指示进行递归的划分，直到split_cu_flags==0或者达到最小的CU尺寸（mininum CU size），对于达到最小尺寸的CU，不需要传输split_cu_flags标志，CU的最小尺寸参数（通过CTU深度确定）在编码器中确定，并在SPS中进行传输。

所以CU的大小范围是：minunum size CU ~CTU，一般情况设置CTU为64，最小CU为8（通过CTU深度确定），所以此时CU大小可取8、16、32、64。一个CTU进行编码时，是按照深度优先的顺序进行CU编码，类似于z-scan，如下图：右边表示CTU的递归四叉树划分，左边表示CTU中CU的编码顺序。

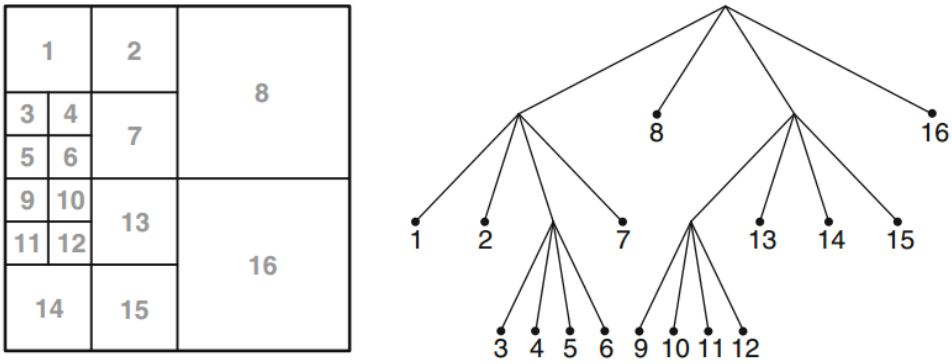


Fig. 3.4 Example for the partitioning of a 64 × 64 coding tree unit (CTU) into coding units (CUs) of 8 × 8 to 32 × 32 luma samples. The partitioning can be described by a quadtree, also referred to as coding tree, which is shown on the *right*. The numbers indicate the coding order of the CUs

视频序列的分辨率（长和宽参数）也会在SPS中传输，要求长宽必须是mininum CU size的整数倍，但是可以不是 CTU size的整数倍。对于长宽不是CTU size整数倍的情况，图像边界处的CTU被认为已经分割成和图像边界重合（the CTUs at the borders are inferred to be split until the boundaries of the resulting blocks coincide with the picture boundary），对于这种边界处默认的分割，不需要传输split_cu_flags标志。

CU块是进行决策帧间、帧内、Skip/Merge模式的基本单元。

PU：prediction unit，预测单元

在CU level决定prediction mode，并将一个CU的prediction mode传输在bitstream中。而PU是是进行预测的基本单元，有一个PB of the luma、2个PB of the choma和相应的语法元素组成。

如果一个CU的prediction mode是intra prediction（帧内预测）：

对于luma CU：有35个可选的帧内预测方向（Plannar(0)、DC(1)和方向预测(2~34)），对于mininum size的luma CB，可以平均划分成4个方形的subblocks，对于每个subblock进行独立的帧内预测，有独立的intra prediction mode。也就是说对于帧内预测的CU，可以进行2Nx2N和NxN两种PU划分模式，且NxN模式只有对mininum size CB可以使用。

一个帧内luma PU块，预测模式确定之后，需要对预测模式进行编码。HEVC中在进行帧内预测模式编码时，先为每个intra PU确定3个最可能模式（确定策略后面介绍），假设为S={M1，M2，M3}。然后通过判断luma PU的帧内预测模式是否在S中，如果在S中，则需要2bit编码预测模式在S中的索引，否则需要5bit编码预测模式在另外32种模式中的索引。

对于luma PU，确定最可能3个预测模式是根据当前PU左边和上边的预测模式，假设左边和上边的预测模式分别是A和B，如果左边或上边PU不是帧内预测模式或是PCM模式，则A或B为DC；另外，如果上边PU块不在当前CTU内，那么B也为DC。确定好A和B之后：

当A=B时，如果A，B都大于2，即A和B都不是Planar或DC，那么：

M1=A；

M2=2+（（A-2-1+32）%32）

M3=2+（（A-2+1）%32）

当A=B时，如果A，B至少有一个小于2，即A或B是Planar或DC，那么：

M1=Planar，M2=DC，M3=26（竖直方向预测）

当A！=B时，M1=A，M2=B，对于M3按照下面规则决定：

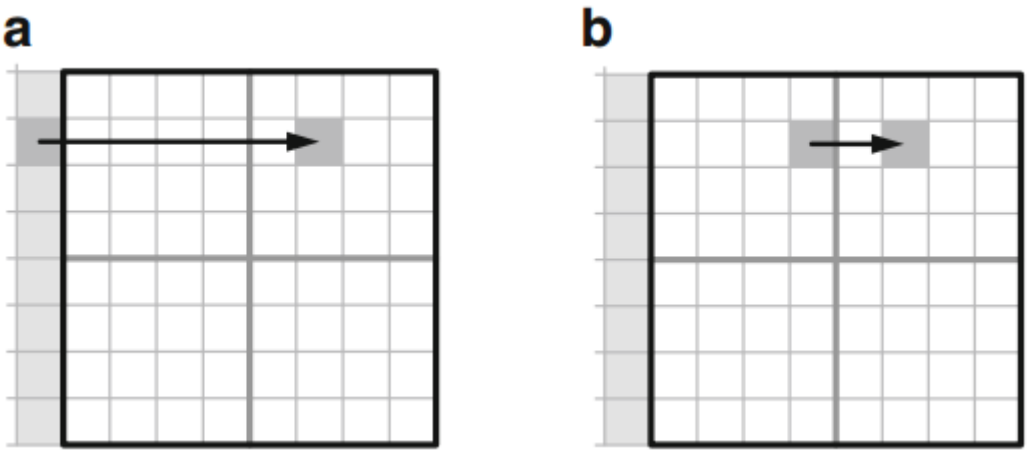
如果A和B都不是Planar，那么M3=Planar；

如果A和B都不是DC，那么M3=DC；

否则，说明{A，B}={Planar，DC}，那么M3=26。

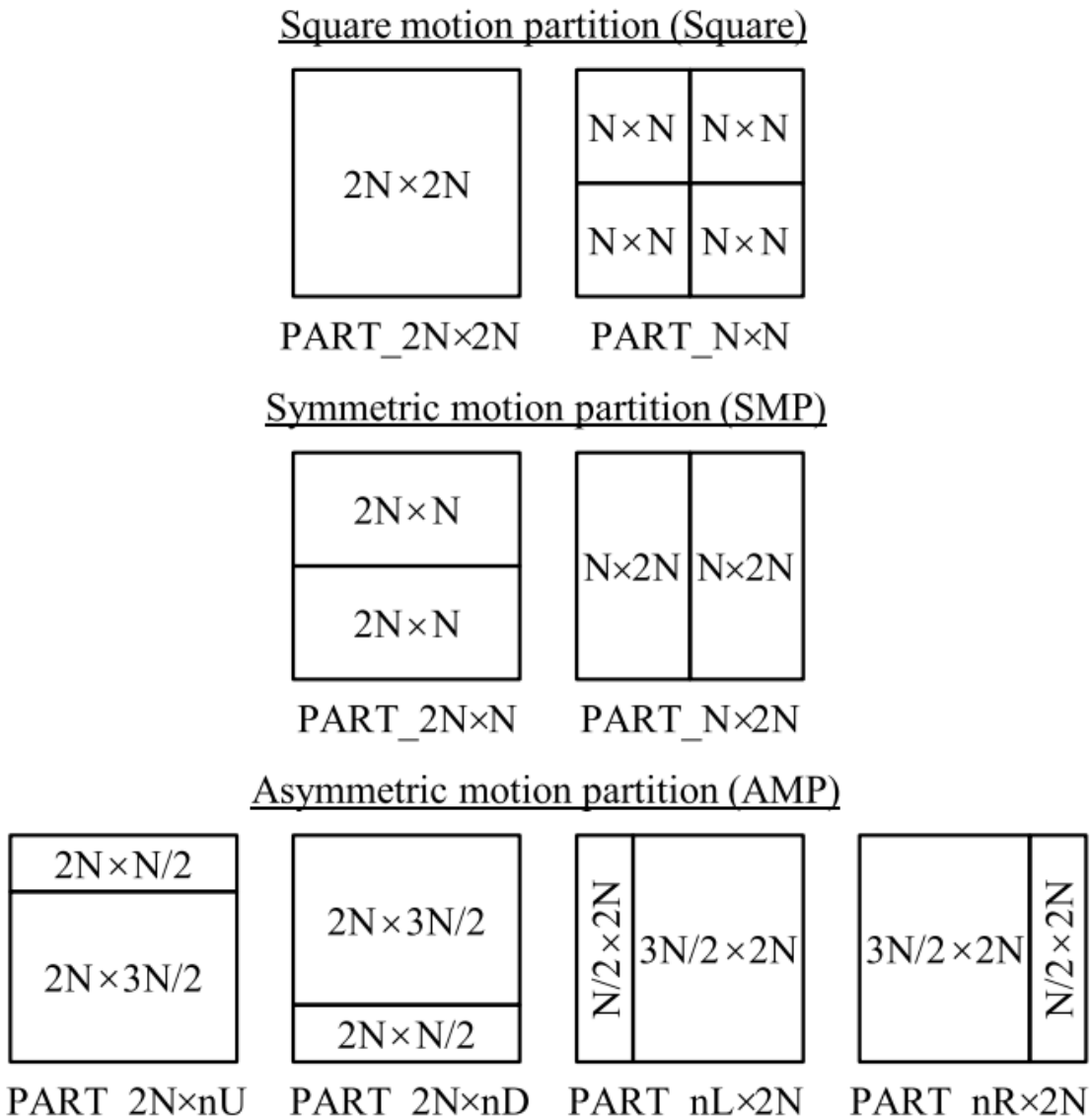
对于choma luma：有5个可选的帧内预测方向（Planar/0、DC/1、Vertical/26、Horizontal/10和luma PU的预测方向）。对于预测模式的编码，通过0表示luma PU的预测方向，100、111、101和110分别表示Planar/0、DC/1、Vertical/26和Horizontal/10。

另外，在进行帧内预测时，如果CU是mininum size CU，且将CU划分成4个PU时，那么要保证TU小于等于PU，如下图：表示一个8x8的CU块分成4个PU，那么必须分成四个4x4的TU块，至于每个TU是否进一步划分成更小的TU不作限定，只根据正常TU划分的条件判断。这是为了提高intra预测的精确度。图a表示如果CU不化成4个TU，那么intra预测的距离就会较远。图b则表示了将CU划分成4个TU，这时候预测右边的小PU时，左边的PU已经预测完成，并进行了变换和重建，可以保证预测距离更近。



如果一个CU的prediction mode是inter prediction（帧间预测）：

对于inter PU，luma PB和choma PBs拥有相同的PU划分模式和motion parameters（包括运动估计方向数目(1/2)，参考帧索引，和对每个运动估计方向的运动矢量MV）。HEVC中有8中PU划分模式(2Nx2N、NxN、2个SMP和4个AMP)，如下图所示：



img

对于NxN模式，只有mininum size CU可以使用，且8x8CU不能使用。

对于AMP模式，只有32x32和16x16的CU可以使用，8x8和64x64的CU不能使用，所以inter PU的最小尺寸为8x4和4x8，这是因为TU最小尺寸为4x4，进行变换的最小单元也是4x4。另外，HEVC可以在SPS中通过一个syntax禁用AMP。

从H.262到HEVC过程中，PU的可选大小变化如下图：

Video coding standard	Supported block sizes for motion-compensated prediction
H.262 | MPEG-2 Video	16 × 16
H.263	16 × 16, 8 × 8
MPEG-4 Visual	16 × 16, 8 × 8
H.264 | MPEG-4 AVC	16 × 16, 16 × 8, 8 × 16, 8 × 8, 8 × 4, 4 × 8, 4 × 4
HEVC	64 × 64, 64 × 48, 64 × 32, 64 × 16, 48 × 64, 32 × 64, 16 × 64, 32 × 32, 32 × 24, 32 × 16, 32 × 8, 24 × 32, 16 × 32, 8 × 32, 16 × 16, 16 × 12, 16 × 8, 16 × 4, 12 × 16, 8 × 16, 4 × 16, 8 × 8, 8 × 4, 4 × 8

如果一个CU的prediction mode是Skip：

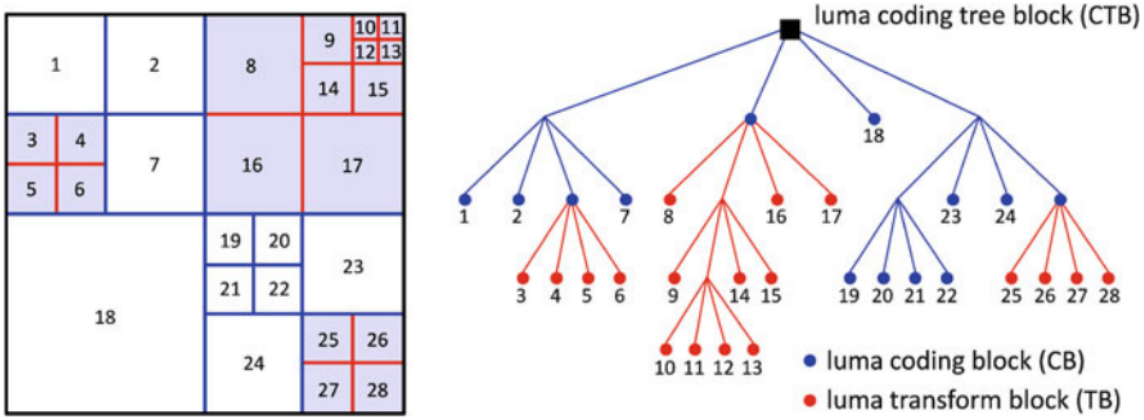
那么PU的划分模式只能是2N x 2N。

PS：对于4x8和8x4，HEVC规定只能用单向预测，不能用双向预测。

在HM1中，实际可以通过inter_4x4_enabled_flag（在SPS中）指示是否使用4x4的PU。

TU：transform unit，变换单元

对于是进行变量的单元，一个CU可以递归按照四叉树结构划分成TUs，CU作为四叉树的root，如下图表示一个CU划分成TUs的结构：



CU划分成TUs中，TU的大小范围取决于max TU size、min TU size和max TU depth三个参数决定，这三个参数在SPS level进行传输。max TU size为5表示最大TU是32x32，min TU size为2表示最小TU是4x4。max depth为3表示CU划分成TU最多划分成3层（如上图10、11、12、13就在第3层）。对于intra predition，要确保PU大于等于TU（即TU不跨多个intra PU），而inter predition没有相应的限制。

另外，对于一个CU，最多有一个trasform tree syntax，所以一个CU的luma CB 和choma CBs拥有相同的TU划分。但是除了对于8x8的luma CB划分成4x4的TB时，4x4的choma CBs不会划分成2x2的TB。

在进一步看代码前，先了解一下图像划分方式：

HEVC中，一帧图像分为多个slice，每个slice进行独立编解码。每个slice分为多个树形编码单元CTU，

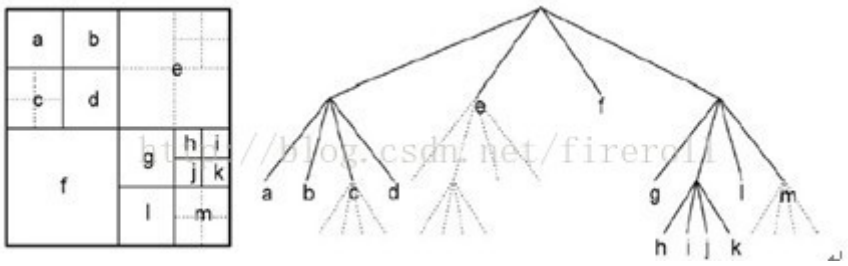
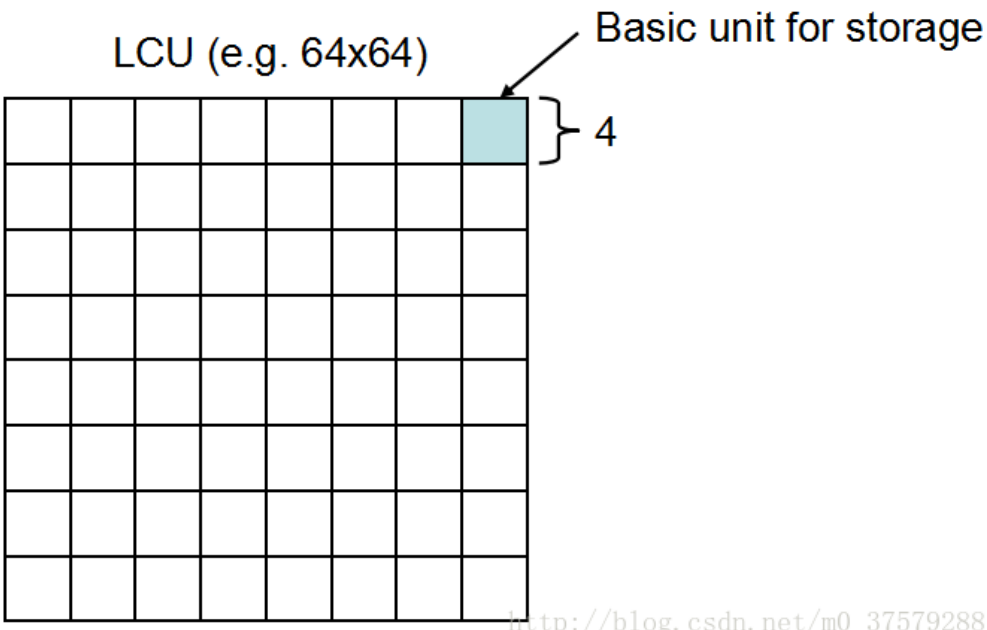


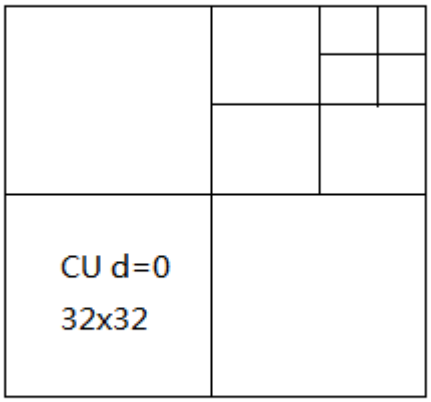
Fig. 灵活的块结构示意图

一个CTU分为一个亮度CTB和两个色度CTB，CTB大小有16、32、64（CTU同）。进一步再划分为CU编码单元、PU预测单元、TU变换单元，使得编码、预测、变换分离，处理的时候更灵活。它们的关系时，CTU以四叉树方式划分为CU，CU最大为64x64，最小为8x8，CU以四叉树方式划分为TU、PU，TU最大为32x32，最小为8x8，其中PU与TU无确定关系，允许TU跨越多个PU，但在帧内预测中一个PU可对应多个TU，一个TU至多对应一个PU。另外，HM中数据最小处理单元为4x4，而不是每次处理一个像素。



图一.4x4最小单元

CTU与CU的关系如图：



图二 CTU中CU的划分

CU的大小在代码中用划分深度（下一篇代码中可见）来表示，PU与TU的划分均基于CU。（Visio还没装上，这图抄网上的也不规范）

扫描方式：

前辈HEVC_CJL提到了帧内预测的扫描方式，感谢前辈，原文链接：http://blog.csdn.net/hevc_cjl/article/details/8183144

HEVC对像素有两种扫描方式：光栅扫描和Z扫描。指对像素的读取顺序，两种组织顺序如下：

0	1	4	5	16	17	20	21
2	3	6	7	18	19	22	23
8	9	12	13	24	25	28	29
10	11	14	15	26	27	30	31
32	33	36	37	48	49	52	53
34	35	38	39	50	51	54	55
40	41	44	45	56	57	60	61
42	43	46	47	58	59	62	63

g_auiZscanToRaster →

← g_auiRasterToZscan

0	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	50	51	52	53	54	55
56	57	58	59	60	61	62	63

图三 Z扫描与光栅扫描

如图，左边为Z扫描，右边为光栅扫描。HEVC中为方便两种扫描方式数据转换，定义了转换数组g_auiRasterToZscan, g_auiZscanToRaster, g_auiRasterToX, g_auiRasterToY，即将上图中数据放到另一个组中对应的位置。转换数组如下：

RasterToZscan

0	1	4	5	16	17	20	21	64	65	68	69	80	81	84	85
2	3	6	7	18	19	22	23	66	67	70	71	82	83	86	87
8	9	12	13	24	25	28	29	72	73	76	77	88	89	92	93
10	11	14	15	26	27	30	31	74	75	78	79	90	91	94	95
32	33	36	37	48	49	52	53	96	97	100	101	112	113	116	117
34	35	38	39	50	51	54	55	98	99	102	103	114	115	118	119
40	41	44	45	56	57	60	61	104	105	108	109	120	121	124	125
42	43	46	47	58	59	62	63	106	107	110	111	122	123	126	127
128	129	132	133	144	145	148	149	192	193	196	197	208	209	212	213
130	131	134	135	146	147	150	151	194	195	198	199	210	211	214	215
136	137	140	141	152	153	156	157	200	201	204	205	216	217	220	221
138	139	142	143	154	155	158	159	202	203	206	207	218	219	222	223
160	161	164	165	176	177	180	181	224	225	228	229	240	241	244	245
162	163	166	167	178	179	182	183	226	227	230	231	242	243	246	247
168	169	172	173	184	185	188	189	232	233	236	237	248	249	252	253
170	171	174	175	186	187	190	191	234	235	238	239	250	251	254	255

ZscanToRaster

0	1	16	17	2	3	18	19	32	33	48	49	34	35	50	51
4	5	20	21	6	7	22	23	36	37	52	53	38	39	54	55
64	65	80	81	66	67	82	83	96	97	112	113	98	99	114	115
68	69	84	85	70	71	86	87	100	101	116	117	102	103	118	119
8 9	9 9	24	25	10	11	26	27	40	41	56	57	42	43	58	59
12	13	28	29	14	15	30	31	44	45	60	61	46	47	62	63
72	73	88	89	74	75	90	91	104	105	120	121	106	107	122	123
76	77	92	93	78	79	94	95	108	109	124	125	110	111	126	127
128	129	144	145	130	131	146	147	160	161	176	177	162	163	178	179
132	133	148	149	134	135	150	151	164	165	180	181	166	167	182	183
192	193	208	209	194	195	210	211	224	225	240	241	226	227	242	243
196	197	212	213	198	199	214	215	228	229	244	245	230	231	246	247
136	137	152	153	138	139	154	155	168	169	184	185	170	171	186	187
140	141	156	157	142	143	158	159	172	173	188	189	174	175	190	191
200	201	216	217	202	203	218	219	232	233	248	249	234	235	250	251
204	205	220	221	206	207	222	223	236	237	252	253	238	239	254	255

图四 Raster与Zscan的转换

RasterToPelX

0	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60
0	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60
0	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60
0	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60
0	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60
0	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60
0	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60
0	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60
0	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60
0	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60
0	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60
0	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60
0	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60
0	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60
0	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60
0	4	8	12	16	20	24	28	32	36	40	44	48	52	56	60

RasterToPelY

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8
12	12	12	12	12	12	12	12	12	12	12	12	12	12	12	12
16	16	16	16	16	16	16	16	16	16	16	16	16	16	16	16
20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20
24	24	24	24	24	24	24	24	24	24	24	24	24	24	24	24
28	28	28	28	28	28	28	28	28	28	28	28	28	28	28	28
32	32	32	32	32	32	32	32	32	32	32	32	32	32	32	32
36	36	36	36	36	36	36	36	36	36	36	36	36	36	36	36
40	40	40	40	40	40	40	40	40	40	40	40	40	40	40	40
44	44	44	44	44	44	44	44	44	44	44	44	44	44	44	44
48	48	48	48	48	48	48	48	48	48	48	48	48	48	48	48
52	52	52	52	52	52	52	52	52	52	52	52	52	52	52	52
56	56	56	56	56	56	56	56	56	56	56	56	56	56	56	56
60	60	60	60	60	60	60	60	60	60	60	60	60	60	60	60

图五 Raster按4x4块为单位的偏移

由上一篇的代码就能知道，HM处理数据按4x4块，如果不理解，去翻一下fillReferenceSample代码中参考像素不全部可用时的处理。