HEVC
(High Efficiency Video Coding)

세종대학교
이영렬 교수
yllee@sejong.ac.kr

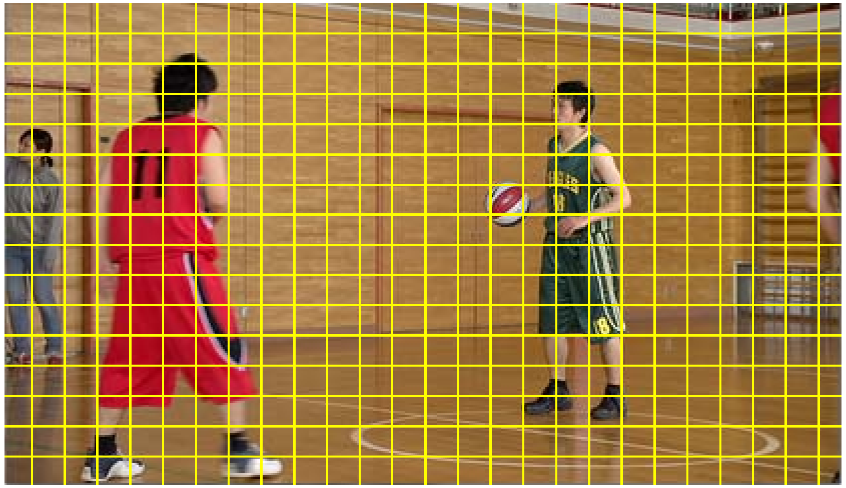
Lists

1	Tree Coding Units (8x8 ~ 64x64)
2	Prediction Units
3	DCT-based Interpolation Filter
4	Advanced Motion Vector Prediction
5	Motion Vector Merging
6	Temporal Prediction Structure

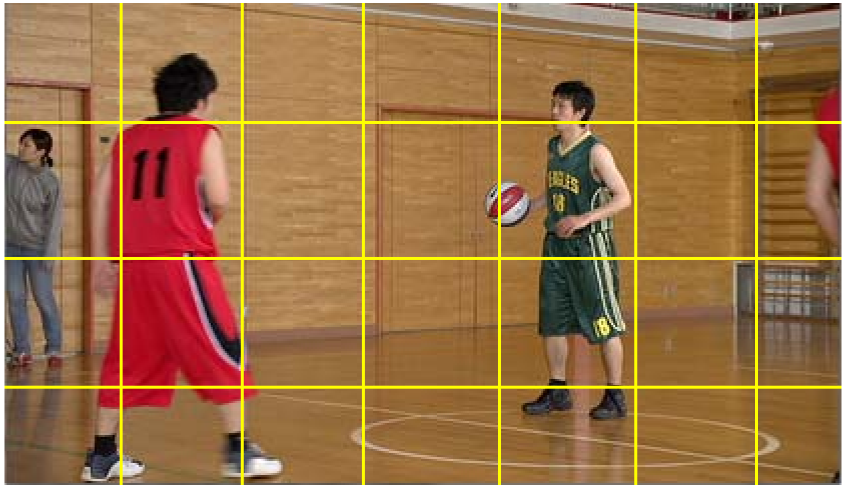
Tree Coding Units (8x8 ~ 64x64)

1. Tree Coding Units (8x8 ~ 64x64)

- 영상을 부호화하는 기본 단위의 크기를 비교
  - BasketballPass_416x240_50.yuv



H.264/AVC(16x16 MB)



HEVC(64x64 LCU)

- HEVC는 LCU를 시작으로 Quad-Tree 구조로 ~8x8까지 분할된 CU를 사용

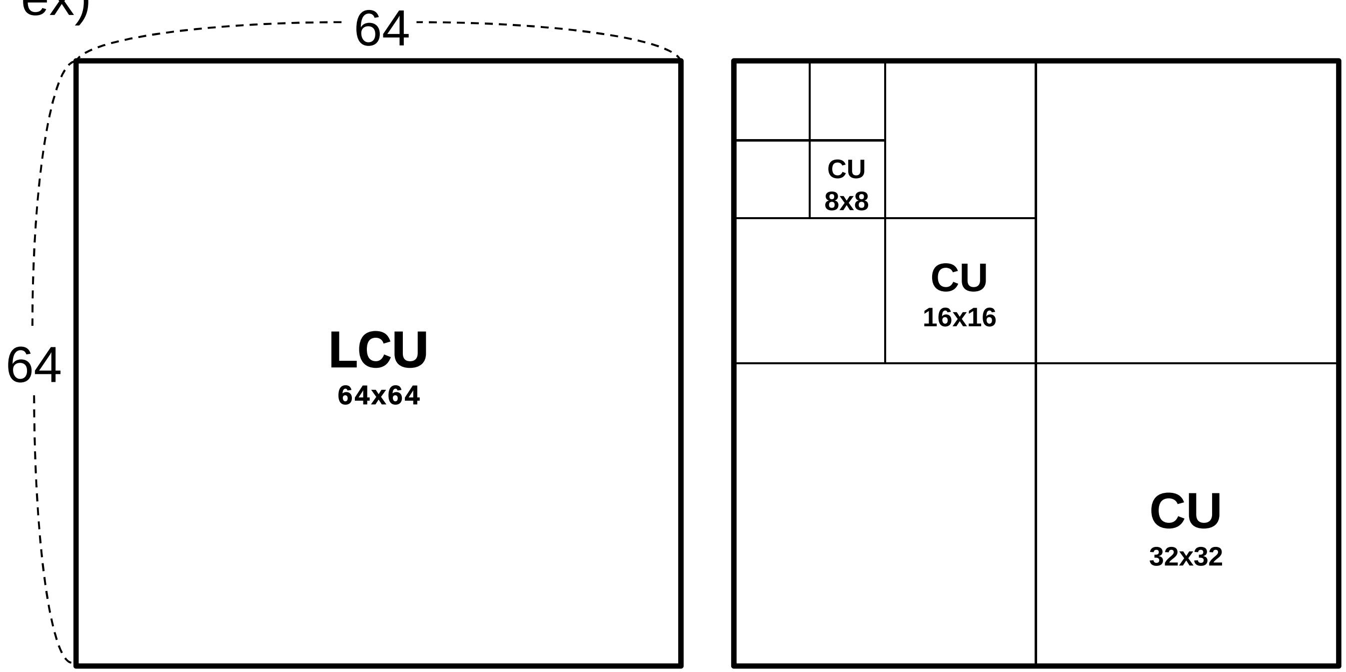
1. Tree Coding Units (8x8 ~ 64x64)

- **Tree Coding Units??**
  - Quad-Tree 구조를 사용함
  - All conditions (AI, LC, HE) 에서 동일함
    - ✓ 64x64~8x8까지 총 4개의 크기의 CU

Cofiguration file 참고

```
#======== Unit definition ================
MaxCUWidth          : 64      # Maximum coding unit width in pixel
MaxCUHeight         : 64      # Maximum coding unit height in pixel
MaxPartitionDepth   : 4       # Maximum coding unit depth
```

ex)

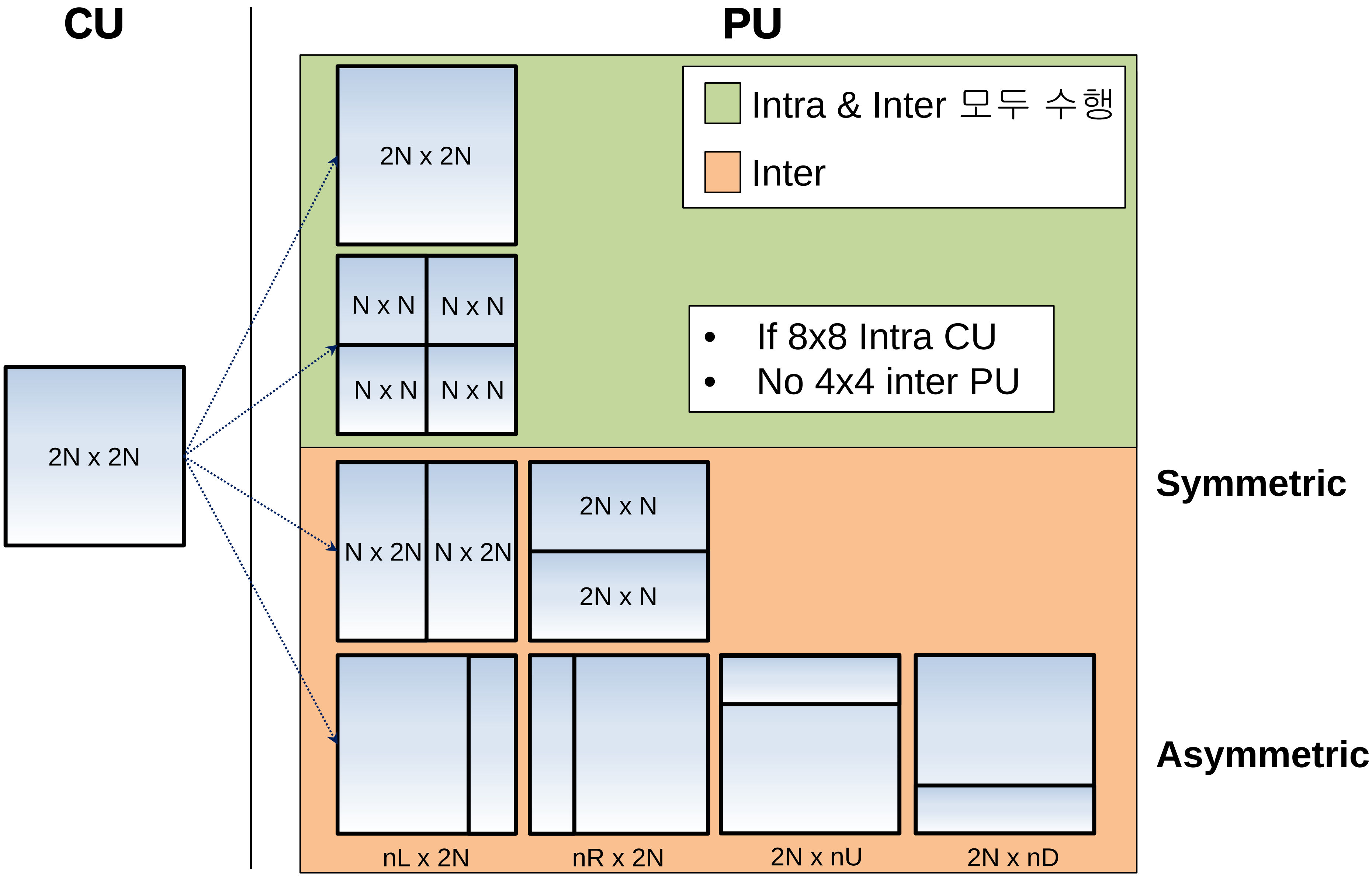


Prediction Units

2. Prediction Units

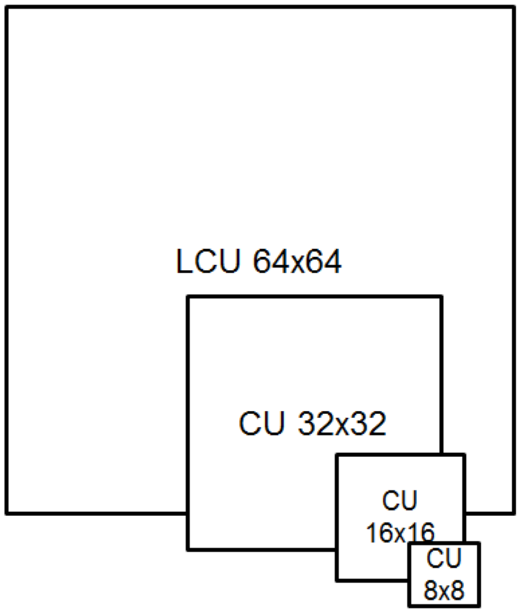
- **Prediction Units?**
  - 예측을 수행하는 단위
  - 각 CU에 대한 **Inter** prediction과 **Intra** prediction을 각각 수행

Inter Prediction Units	**Intra** Prediction Units
➢ Square, Rectangle • Symmetric • Asymmetric(Not Main Profile)	➢ Square

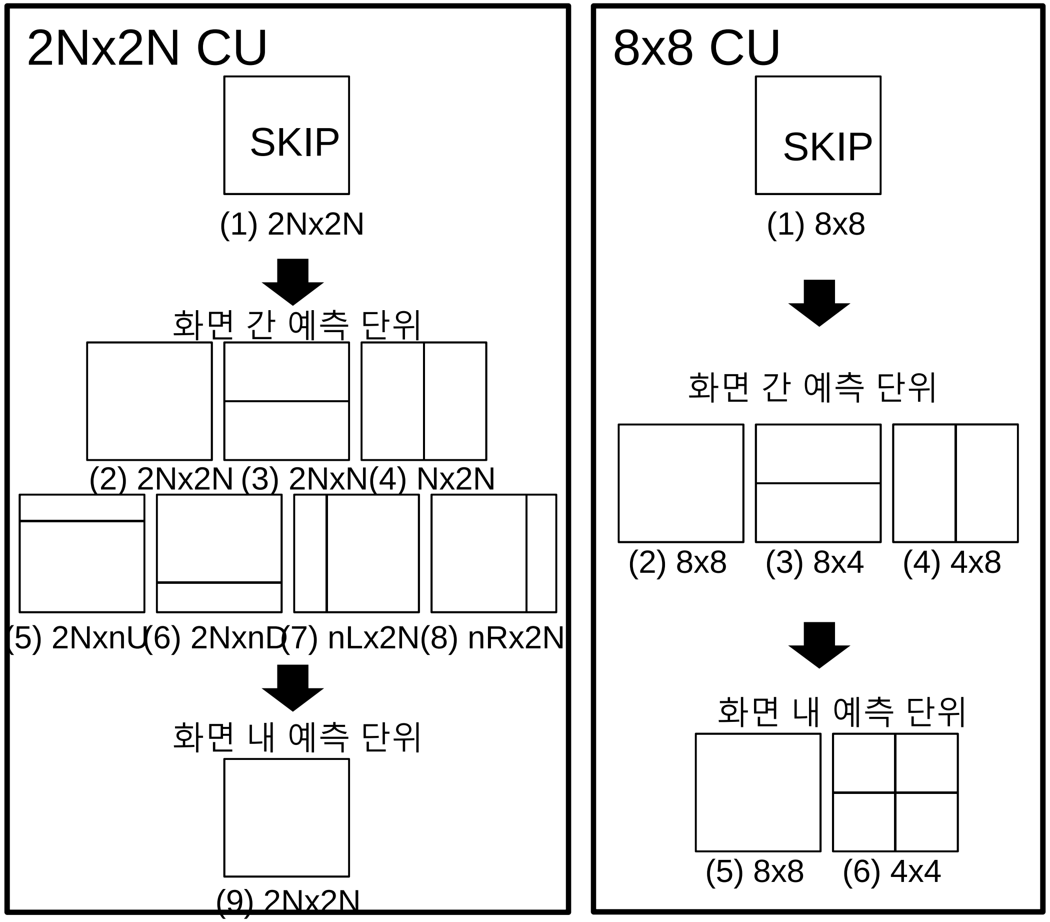

CU
PU
2N x 2N
2N x 2N
N x N
N x N
N x N
N x N
Intra & Inter 모두 수행
Inter
• If 8x8 Intra CU
• No 4x4 inter PU
N x 2N
N x 2N
2N x N
2N x N
Symmetric
Asymmetric
nL x 2N
nR x 2N
2N x nU
2N x nD

2. Prediction Units

- **PU encoding order**



Quad Tree



J0579 Bog on limits

- J0225 and J0335 16bit range constraint (clipping) for both horizontal and vertical MVs.

Item	Syntax Element	Type	Min Value	Max Value	Proposed Min	Proposed Max	Notes	Decision:
…								
16	abs_mvd _minus2	EGk(v)	0	??	0	Indirectly bound by re quiring -both mvd_x and mvd_y be in the range [-2^{15}, $2^{15}-1$]	Also need to bound motio n vector (m vLX and mv LY) to range [-2^{15}, $2^{15}-1$]	Resolved as noted abo ve this table.

J0086 disallow bi-predictive mode for 8x4 and 4x8 inter PUs.

- The CABAC binarization table of inter_pred_idcof 8x4 and 4x8 inter PUs

Slice_type	inter_pred_idc	Name of inter_pred_idc	bin string
P	inferred	Pred_L0	-
B	0	Pred_L0	0
	1	Pred_L1	1
	2	Pred_BI	-

- The CABAC binarization table for inter_pred_idc of inter PUs of 8x8 and larger (same as HM7.0)

Slice_type	inter_pred_idc	Name of inter_pred_idc	bin string
P	inferred	Pred_L0	-
B	0	Pred_L0	00
	1	Pred_L1	01
	2	Pred_BI	1

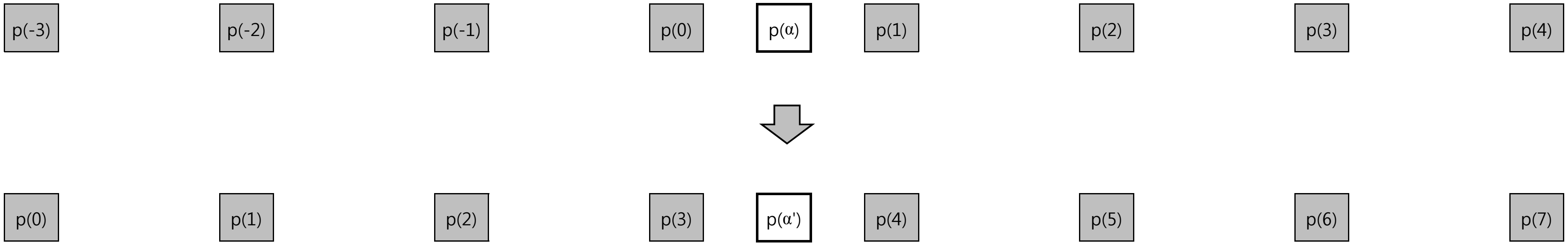
DCT-based Interpolation Filters

3. DCT-based Interpolation Filters

$A_{-1,-1}$				$A_{0,-1}$	$a_{0,-1}$	$b_{0,-1}$	$c_{0,-1}$	$A_{1,-1}$				$A_{2,-1}$
$A_{-1,0}$				$A_{0,0}$	$a_{0,0}$	$b_{0,0}$	$c_{0,0}$	$A_{1,0}$				$A_{2,0}$
$d_{-1,0}$				$d_{0,0}$	$e_{0,0}$	$f_{0,0}$	$g_{0,0}$	$d_{1,0}$				$d_{2,0}$
$h_{-1,0}$				$h_{0,0}$	$i_{0,0}$	$j_{0,0}$	$k_{0,0}$	$h_{1,0}$				$h_{2,0}$
$n_{-1,0}$				$n_{0,0}$	$p_{0,0}$	$q_{0,0}$	$r_{0,0}$	$n_{1,0}$				$n_{2,0}$
$A_{-1,1}$				$A_{0,1}$	$a_{0,1}$	$b_{0,1}$	$c_{0,1}$	$A_{1,1}$				$A_{2,1}$
$A_{-1,2}$				$A_{0,2}$	$a_{0,2}$	$b_{0,2}$	$c_{0,2}$	$A_{1,2}$				$A_{2,2}$

- **Integer samples**
  - shaded blocks with upper-case letters
- **Fractional sample positions**
  - Un-shaded blocks with lower-case letters
  - For quarter sample luma interpolation

3. DCT-based Interpolation Filters



- Forward DCT

$$F(u) = c(u)\sum_{l=0}^{N-1} p(l)\cos\left(\frac{(2l+1)u\pi}{2N}\right)$$

- Inverse DCT

$$p(x) = \sum_{u=0}^{N-1} c(u)F(u)\cos\left(\frac{(2x+1)u\pi}{2N}\right)$$

- α : Fractional point

$$p(\alpha') = \sum_{u=0}^{N-1} c(u)F(u)\cos\left(\frac{(2\alpha'+1)u\pi}{2N}\right)$$

$$N = Tap\ length$$

$$\alpha' = \alpha + \left(\tfrac{N}{2} - 1\right)$$

$$c(0) = \tfrac{1}{\sqrt{N}}, c(k) = \sqrt{\frac{2}{N}}, k = 1,\ldots N-1$$

3. DCT-based Interpolation Filters

- Interpolation filter coefficients
  - Luma

α	*filter*(α)
1/4	{-1, 4, -10, 58, 17, -5, 1, 0 }
1/2	{ -1, 4, -11, 40, 40, -11, 4, -1 }

  - Chroma

α	*filter* (α)
1/8	{ -2, 58, 10, -2,}
1/4	{ -4, 54, 16, -2,}
3/8	{ -6, 46, 28, -4,}
1/2	{ -4, 36, 36, -4,}

3. DCT-based Interpolation Filters

- Luma interpolation process
  - 1D interpolation filter
    - ✓ For fractional positions "$a_{(0,0)}$", "$b_{(0,0)}$" and "$c_{(0,0)}$", horizontal 1D filter is used.
    - ✓ For fractional positions "$d_{(0,0)}$", "$h_{(0,0)}$" and "$n_{(0,0)}$", vertical 1D filter is used.
    - ✓ The input of 1D interpolation function is integer position values.
    - ✓ The output is interpolated value X, which has fractional position α.
    - ✓ Ex. 1/2 position "$b_{(0,0)}$"
      - – 8-tap separable DCTIF coefficient of 1/2 position
        - » { -1, 4, -11, 40, 40, -11, 4, -1 }

$$b_{(0,0)} = \{-1\times A_{(-3,0)} + 4\times A_{(-2,0)} - 11\times A_{(-1,0)} + 40\times A_{(0,0)} + 40\times A_{(1,0)} - 11\times A_{(2,0)} + 4\times A_{(3,0)} - 1\times A_{(4,0)} + 32\} / 64$$

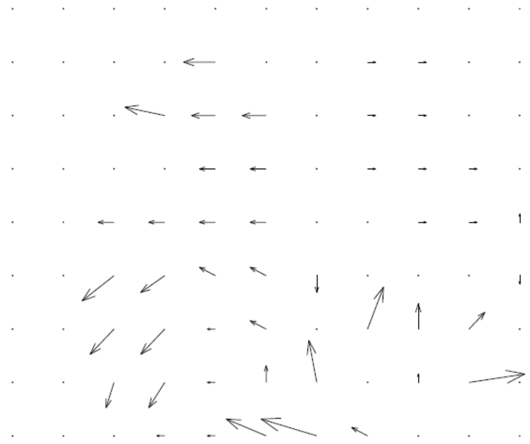
3. DCT-based Interpolation Filters

- 2D separable interpolation filter
  - ✓ For remaining positions first horizontal 1D filter is applied for extended block, and then vertical 1D filter is used.
  - ✓ Ex. 1/4 position "$e_{(0,0)}$"
    - 2D separable Interpolation
    - 8*horizontal 1D filter + 1*vertical 1D filter

- Chroma interpolation process is the same as Luma.

4. Advanced Motion Vector Prediction

4. Advanced Motion Vector Prediction



- **The reason for motion vector prediction**
  - High relevance with MV of neighboring partition.
  - Sending MVD is more efficient than sending MV.

- **Motion Vector Prediction**
  - The process of searching Motion Vector Predictor (MVP or pMV)
  - **Motion Vector Predictor (MVP)**
    - ✓ Predicted vector
    - ✓ HM4.0 : **3 MVP candidates**
  - **Motion Vector Difference (MVD)**
    - ✓ Difference between MVP and MV
    - ✓ MVD = MV – MVP

4. Advanced Motion Vector Prediction

- Decoder receives
  - ref_idx_l0, ref_idx_l1; reference index
  - mvd info
  - mvp_l0_flag, mvp_l1_flag

4. Advanced Motion Vector Prediction

mvLXA	mvLXB

1. Search for Spatial Candidates
2. Remove redundant MVPs
3. Temporal Candidate search if # of spatial candidates< 2
4. Additional Candidate list
   - Zero vector candidates are created by combining zero vector and refIdx

amvp_flag	L0
0	mvL0_A
1	-

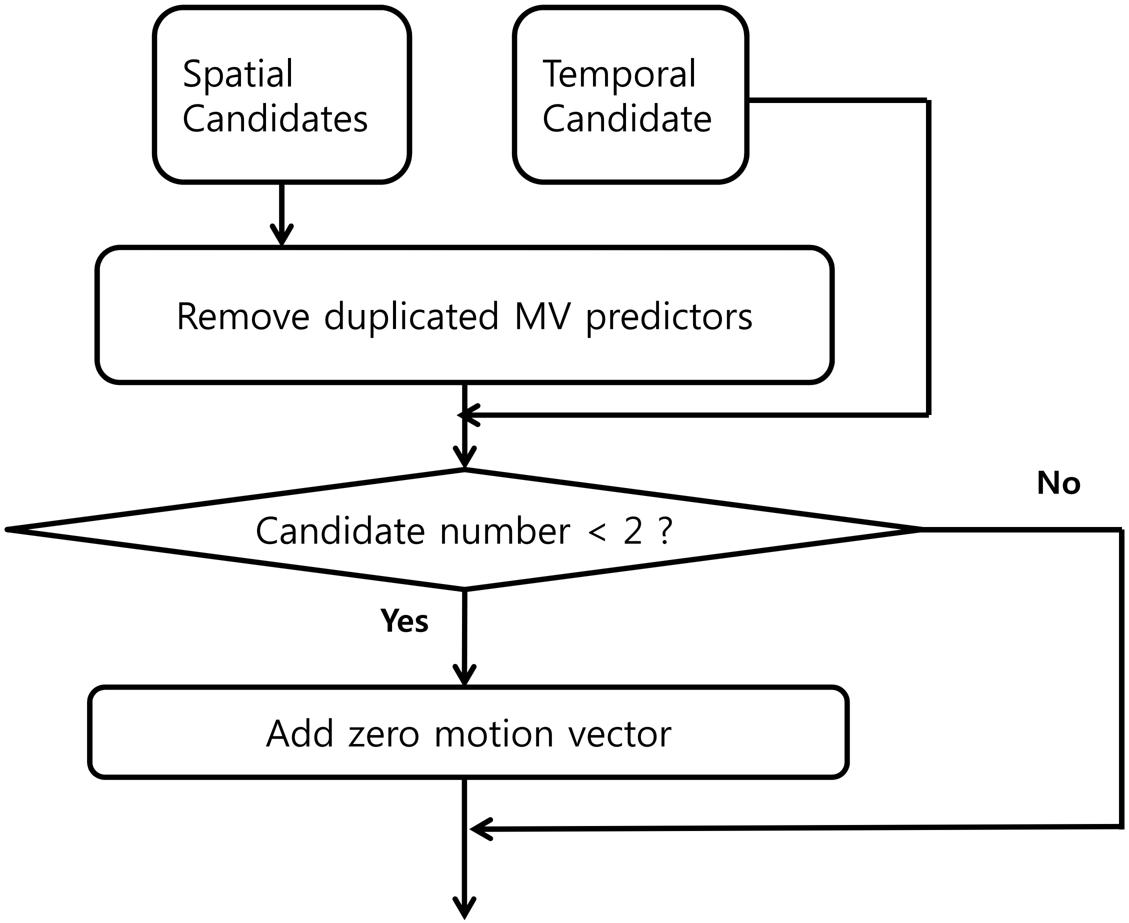
amvp_flag	L1
0	mvL1_A
1	-

Add zero vector

amvp_flag	L0
0	mvL0_A
1	(0, 0)

amvp_flag	L1
0	mvL1_A
1	(0, 0)

Simplification AMVP List Construction



4. Advanced Motion Vector Prediction

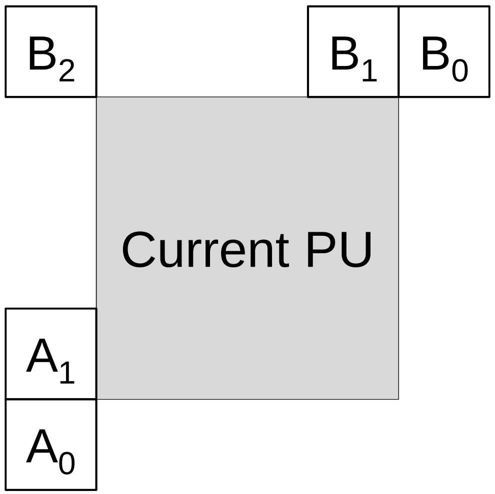
4. Decision of MVP before Motion Estimation (ME start decision from the previous MVP candidates)
   - Distortion : SAD
   - Rate : mvp_flag cost (1 bit)
   - RDCost = Distortion+(Bits*λ + 0.5)>>16 ; 2개후보에 대하여 2번계산
     - ✓ λ = 492942
5. Best MV 결정: ME algorithm – MV decision
   - SAD+(Bits*λ)
6. Decision of the best MVP candidate after Motion Estimation
   - MVP index 결정: Smallest MVD 결정 mvd=Best MV-MVP of mvp_index[i]

4. Advanced Motion Vector Prediction

7. Finally the best MVP with RDO value after competing against the merge modes with RDO values
   - RDO: SATD+ λ*R

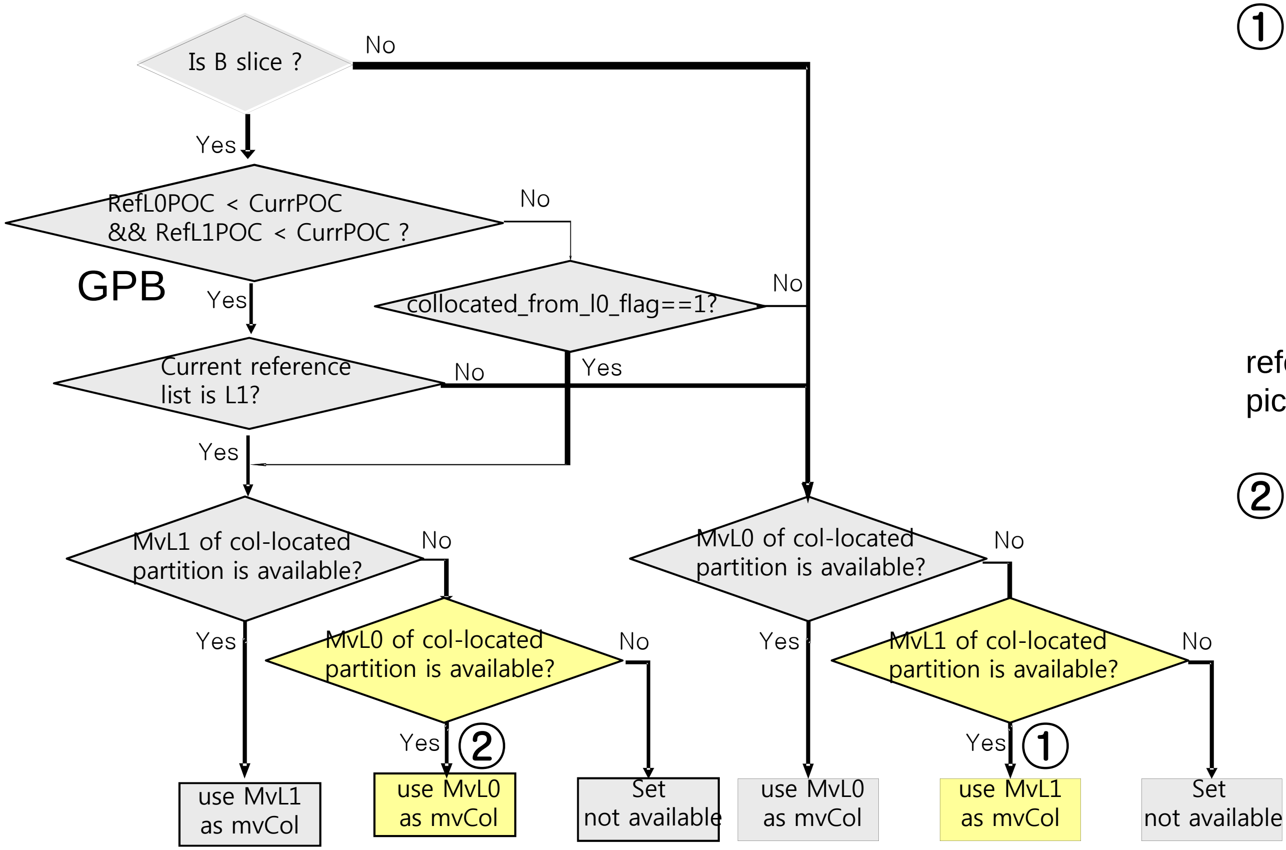
4. Advanced Motion Vector Prediction

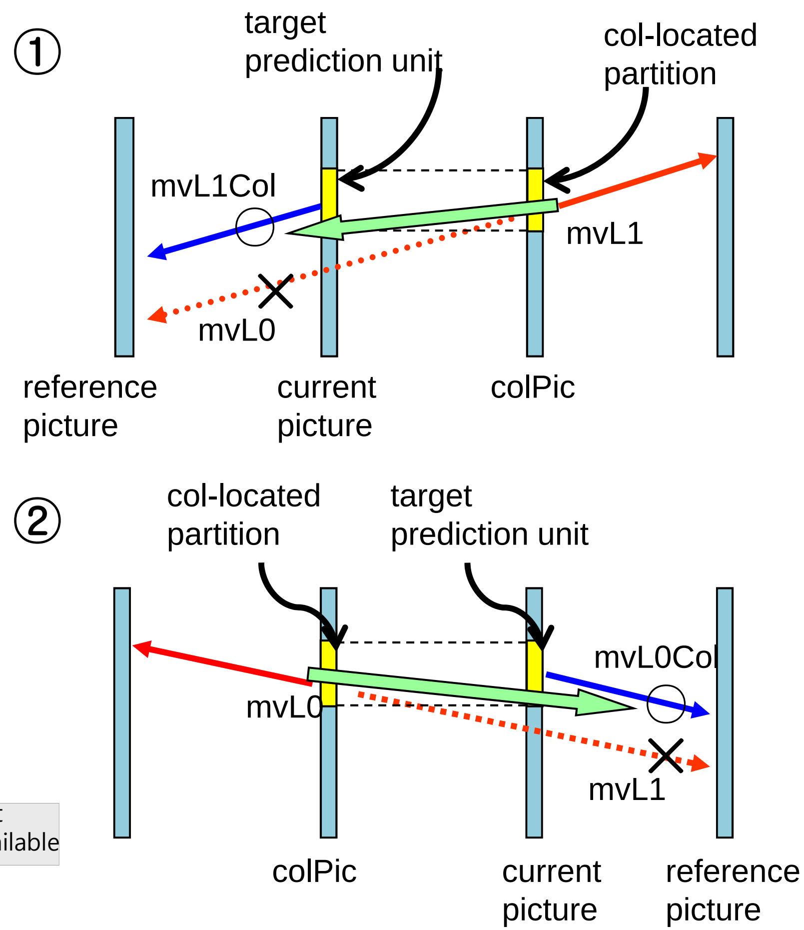
- Spatial MVP Candidates
  - mvLXA : Left spatial candidate
    - ✓ Derivation : (A_0 (→ A_1)) ➔ (scaling)(A_0 → A_1)
  - mvLXB : Above spatial candidate
    - ✓ Derivation : (B_0 → B_1→ B_2) ➔ (isScaledFlagLX == 0) (B_0 → B_1→ B_2)
      - isScaledFlagLX = (A_0 == available && A_0 != Intra) ||
        (A_1 == available && A_1 != Intra)



4. Advanced Motion Vector Prediction

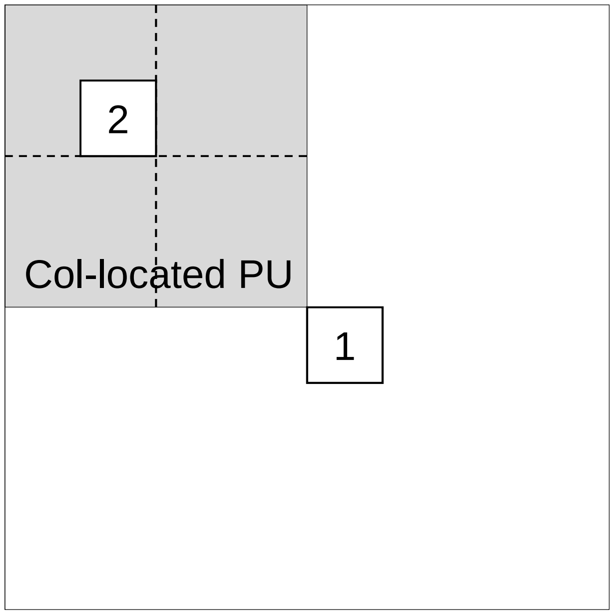
- Temporal MVP Candidate



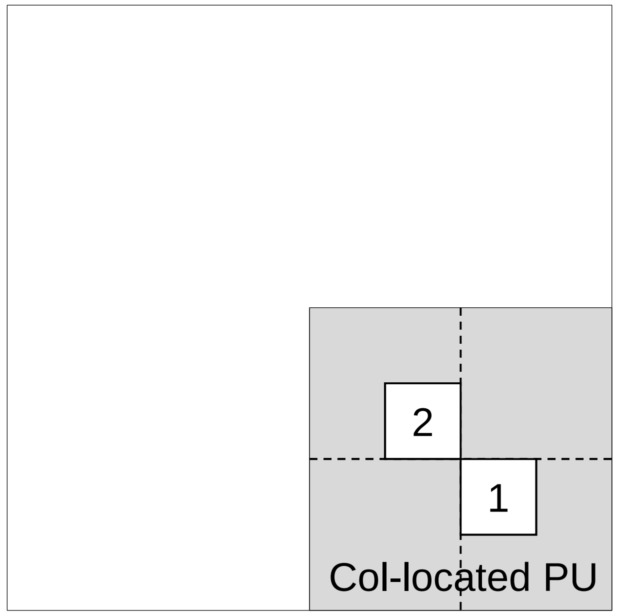


4. Advanced Motion Vector Prediction

- Position of temporal MVP Candidate



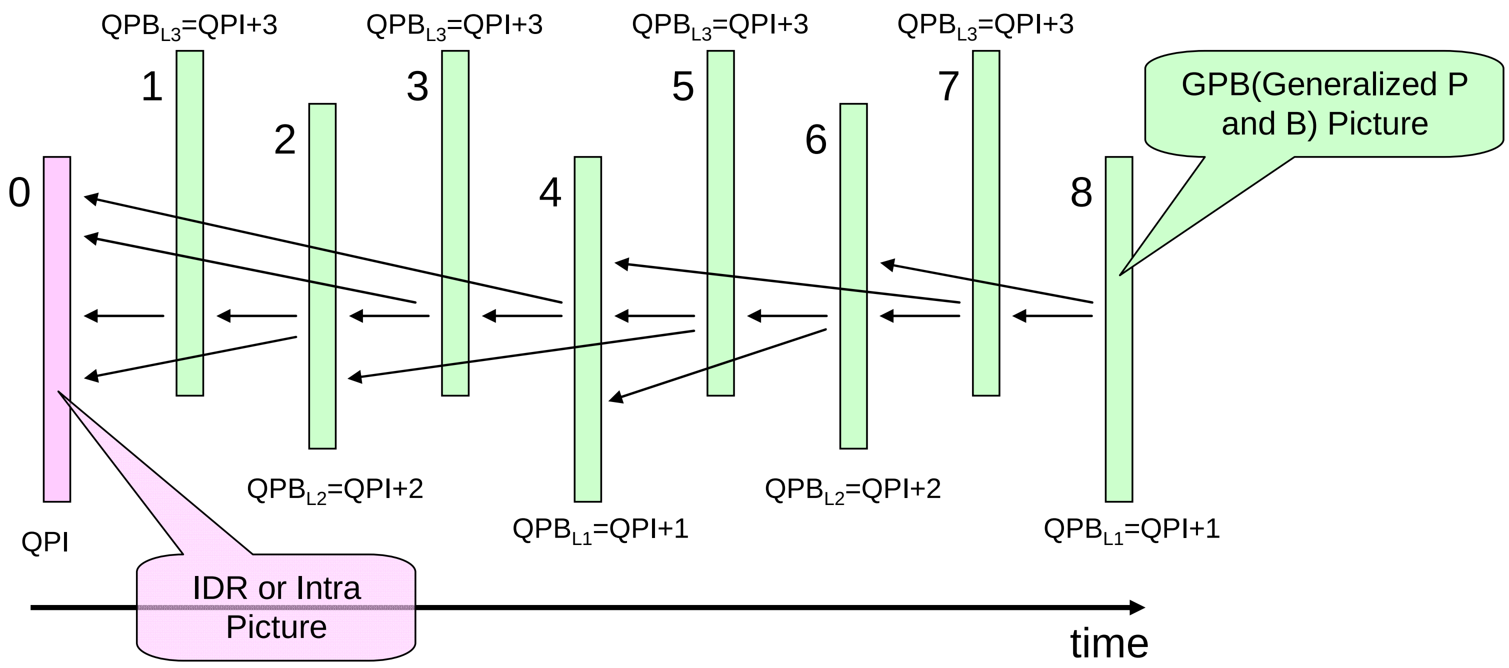
Normal case



Right-bottom PU in LCU

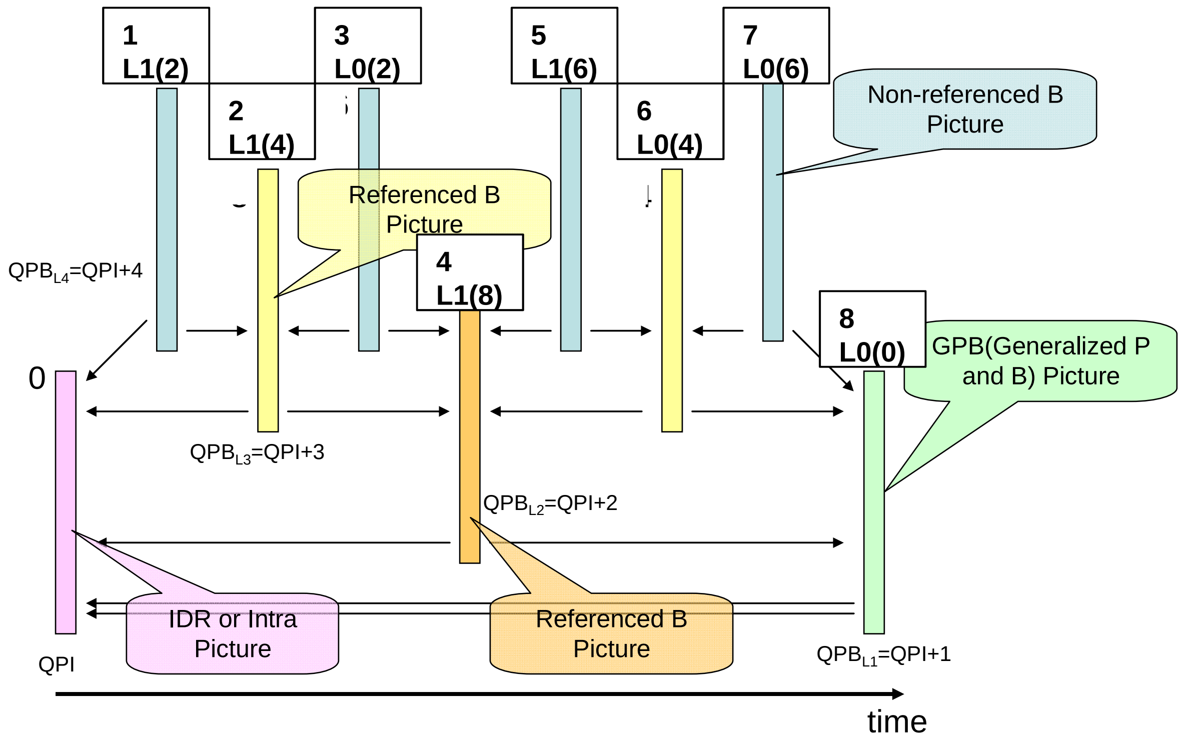
4. Advanced Motion Vector Prediction

- Low-delay 구조의 Col-located block
  - List0[0] Reference



4. Advanced Motion Vector Prediction

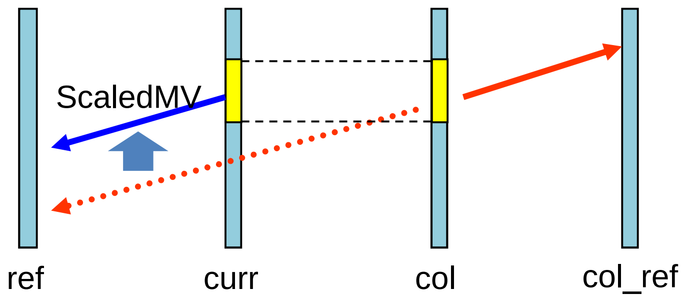
- Random access 구조의 Col-located block



- Computation by uni-directional prediction

4. Advanced Motion Vector Prediction

- MV Scaling



- ScaleFactor

$$ScaleFactor = \frac{POC_{curr} - POC_{ref}}{POC_{col} - POC_{col\_ref}} = \frac{TDB}{TDD}$$

Division-free Scaling

$$ScaleFactor = clip(-4096, 4095, (TDB \times tX + 32)) >> 6)$$

$$tX = \frac{2^{14} + \left|\frac{TDD}{2}\right|}{TDD}$$

- ScaledMV

$$ScaledMV = sign(ScaleFactor \times MV) \times ((abs(ScaleFactor \times MV) + 127) >> 8)$$

- ex)
  - ✓ $ScaleFactor \times MV$ = $1010000000_{(2)}$
  - ✓ $ScaledMV = 10_{(2)}$

5. Motion Vector Merging

5. Motion Vector Merging

- Decoder always receives
  - Merge_flag
  - Merge_index

- Merge Skip (2Nx2N)
  - 2Nx2N Merge mode로 5번 RDO를 수행 (when merge candidates=5)
    - ✓ cbf가 0일 경우 Merge Skip mode로 RDO 수행

5. Motion Vector Merging

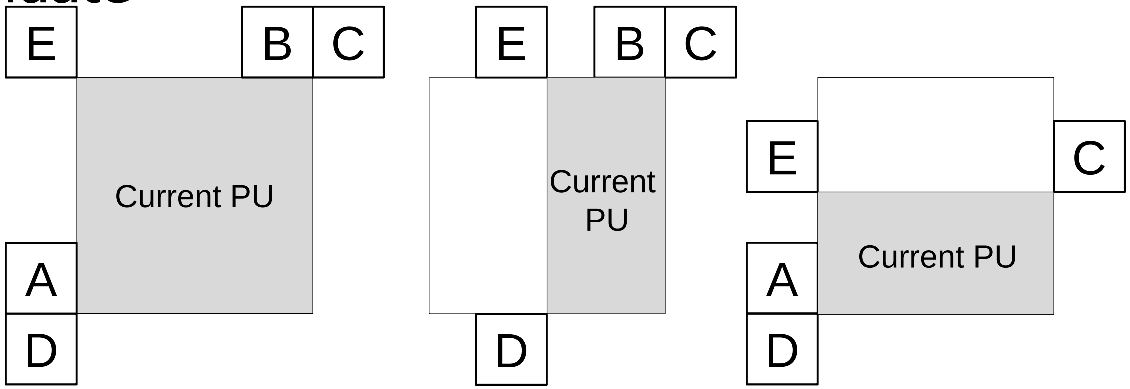
S_0	S_1	S_2	S_3	*Col*

1. Search for spatial candidates S_0, S_1, S_2, S_3
2. Remove redundant candidates
3. Search for temporal candidate *Col*
4. Add Candidate list
4. MRGCost = MRGError + λ*MRGBits ; 5번 실행 후 the best 선택
   - MRGError : Hadamard (SATD)
   - MRGBits : Truncated Unary Code, MaxNum is signaled at Slice header

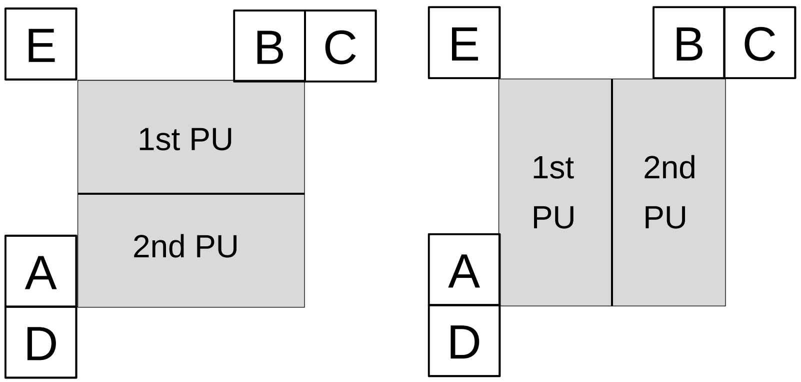
MrgIdx	MaxNum = 5	MaxNum = 4	MaxNum = 3	MaxNum = 2	MaxNum = 1
0	0	0	0	0	N/A
1	10	10	10	10	
2	110	110	110		
3	1110	1110			
4	11110				

5. Motion Vector Merging

- Spatial Merge Candidate
  - 16x16 – 64x64



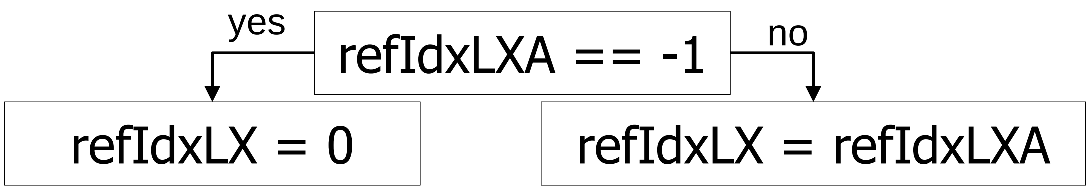
  - 8x8 (log2_parallel_merge_level_minus2 > 0)

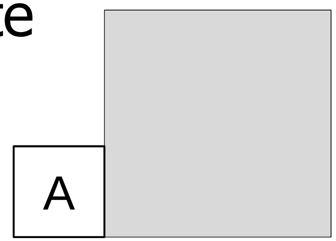


  - Derivation : A → B → C → D → E (one of A,B,C,D is not available)
    - ✓ available ,non-Intra candidate

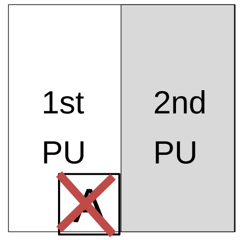
5. Motion Vector Merging

- To get the reference index of Temporal Merge Candidate
  - refIdxLX





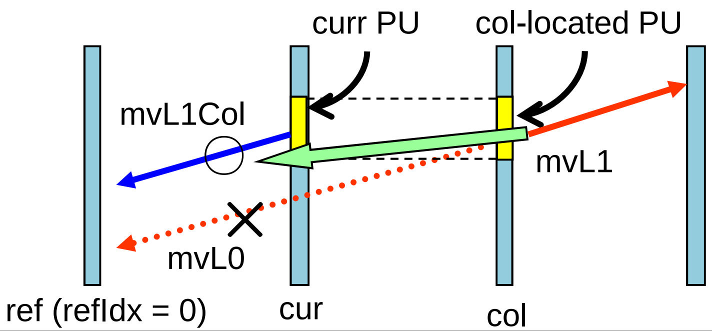
✓ 2nd PU (only in Nx2N) : refIdxLX = 0

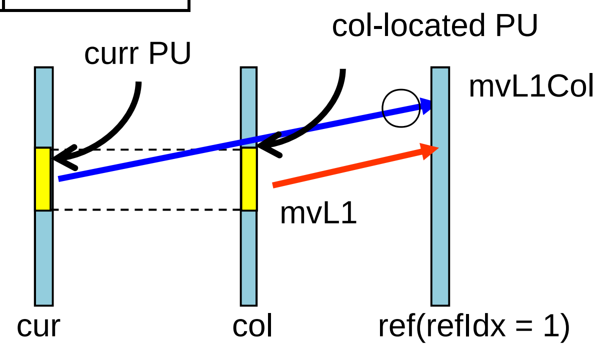


- Derivation of temporal merge candidate
  - Same process with TMVP

✓ ex) reference index :

reference Picture	List0	0
	List1	1





5. Motion Vector Merging

- Additional Candidate list
  1. Combined bi-directional Merge candidate
     - Two candidates in original candidates, which have mvL0 and refIdxL0 or

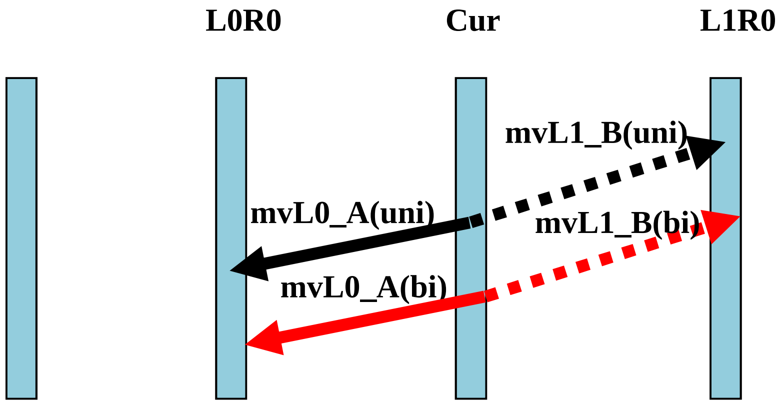
Original Merge candidate list

Merge_idx	L0	L1
0	mvL0_A, ref0	-
1	-	mvL1_B, ref0
2		
3		
4		

Merge candidate list after adding combined candidates

Merge_idx	L0	L1
0	mvL0_A, ref0	combine
1		mvL1_B, ref0
2	mvL0_A, ref0	mvL1_B, ref0
3		
4		

combine



5. Motion Vector Merging

- Additional Candidate list
  2. Zero vector Merge/AMVP candidate
     - ✓ Zero vector Merge/AMVP candidates are created by combining zero vector and refIdx

Original Merge candidate list

Merge_idx	L0	L1
0	mvL0_A, ref0	-
1	-	mvL1_B, ref0
2	mvL0_A, ref0	mvL1_B, ref0
3	-	-
4	-	-

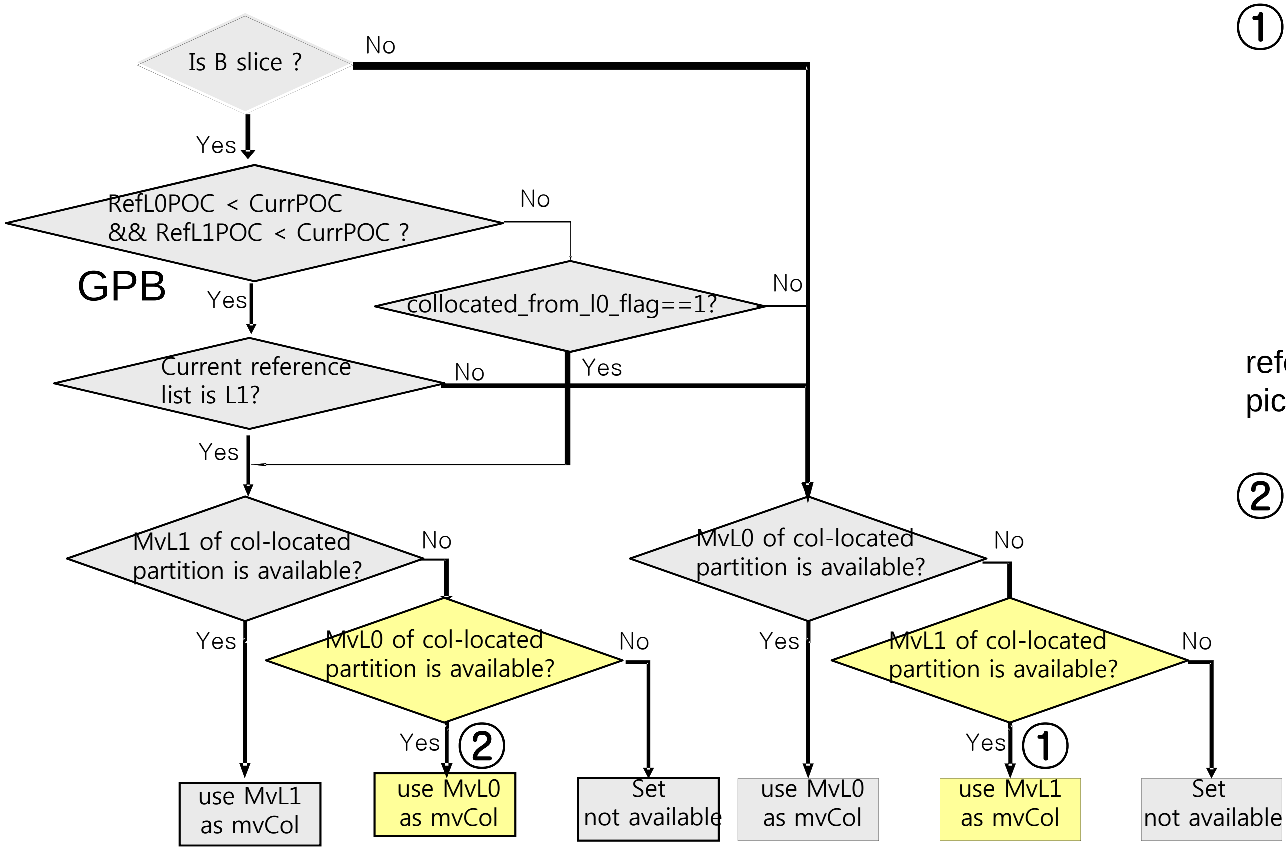
Merge candidate list after adding new ones

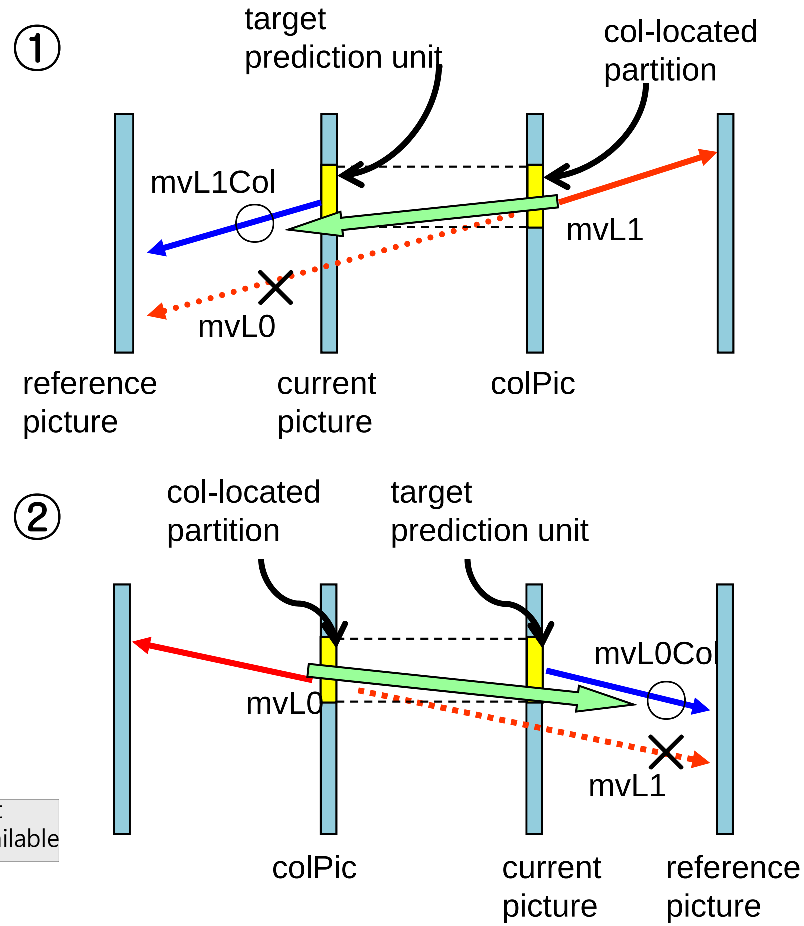
Merge_idx	L0	L1
0	mvL0_A, ref0	
1		mvL1_B, ref0
2	mvL0_A, ref0	mvL1_B, ref0
3	(0,0), ref0	(0, 0), ref0
4	(0,0), ref1	(0, 0), ref1

6. Temporal Prediction Structure

6. Temporal Prediction Structure

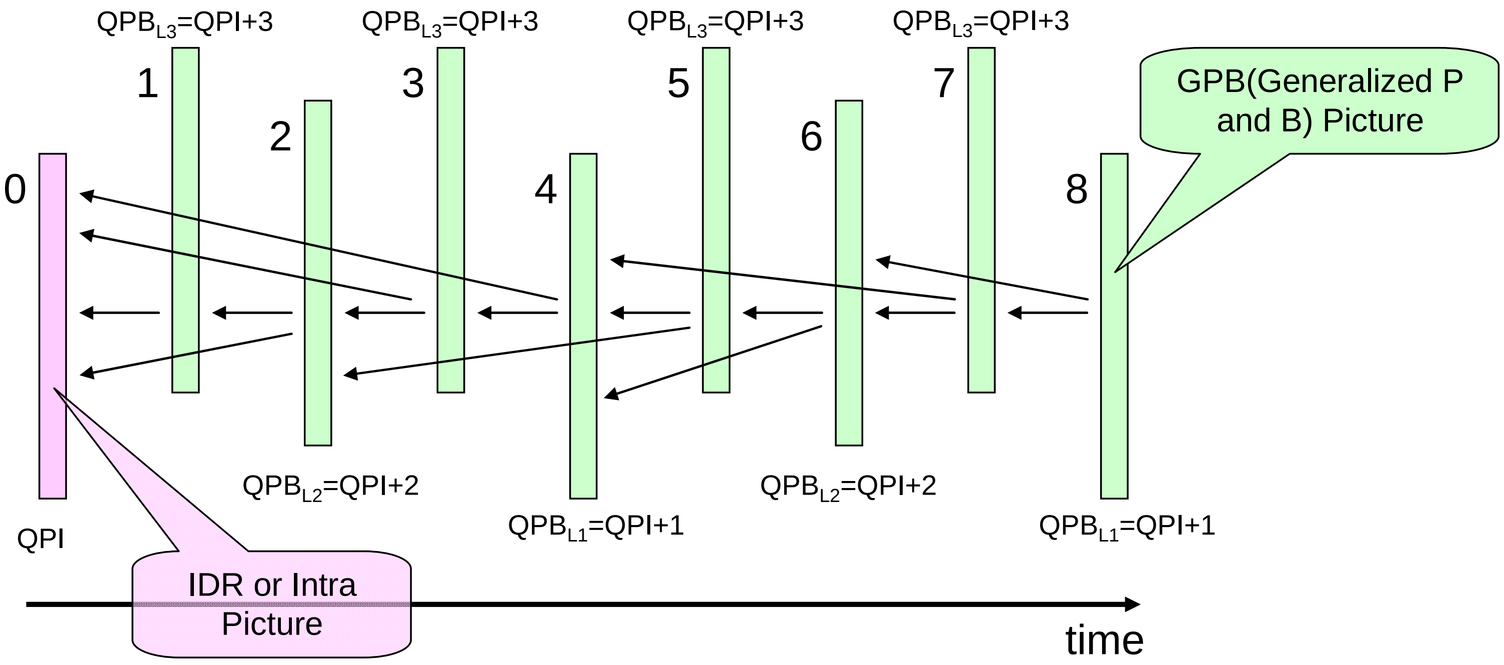
❖ Temporal MVP Candidate





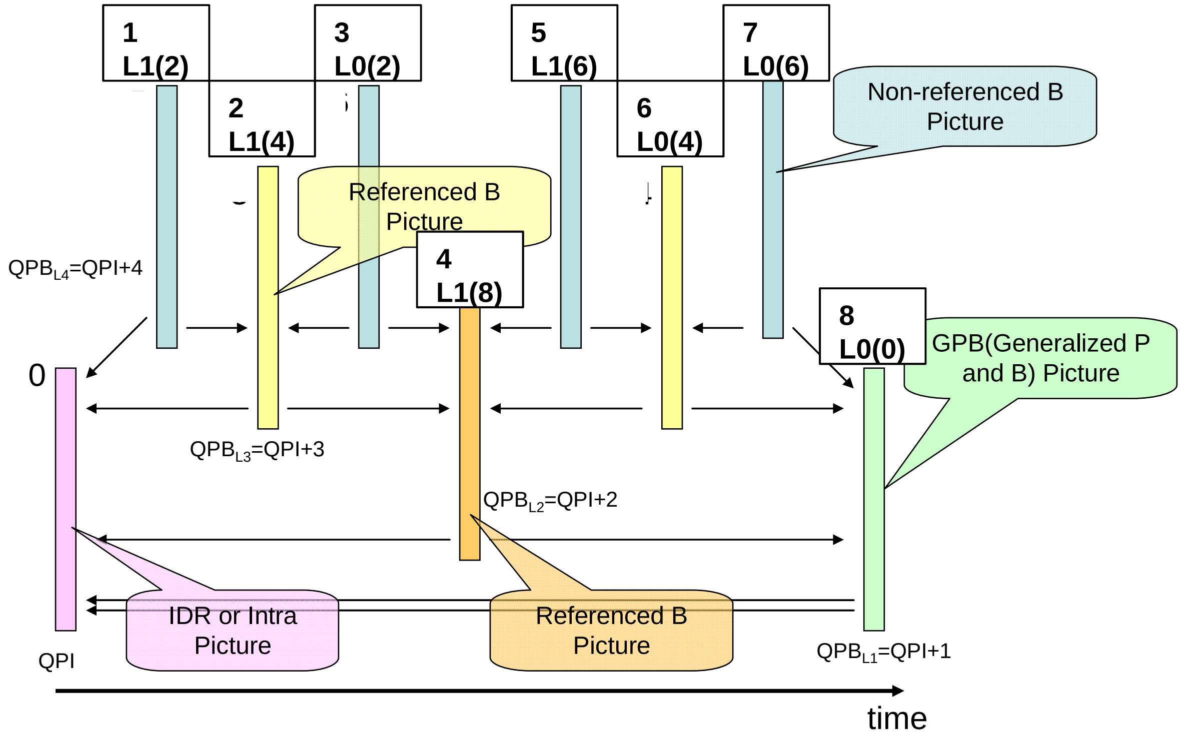
6. Temporal Prediction Structure

- **Col-located block**
  - Low-delay 구조
    - ✓ List0[0] Reference



6. Temporal Prediction Structure

- **Col-located block**
  - Random access 구조



  - Computation by uni-directional prediction

Main Profile

Main Profile	Profile not-defined
Coding Unit 8x8 up to 64x64 in tree structure	-
Prediction Unit 2Nx2N, 2NxN, Nx2N, NxN, Asymmetric Motion Partition(2NxnU, 2NxnD, nLx2N, nRx2N)	Inter4x4 PU
Transform unit tree(3 level max)	Non-square quadtree (4x16, 8x32)
Transform block size of 4x4 to 32x32 samples 4x4 intra mode dependent DCT/DST	-
Angular Intra Prediction (34 modes)	LM Chroma mode
Luma: DCT-based interpolation (Half pel : 8-tap, Quarter pel : 7-tap) Chroma: DCT-based interpolation filter (4-tap)	-
Advanced motion vector prediction PU based Motion vector merge/CU based skip	-
Simplified Deblocking Filter	-
Sample Adaptive Offset	Adaptive Loop Filter
CABAC	-
High-level parallelism : Tile, Wavefront	-