

Video Coding Technology proposal by LG Electronics

JVET-J0017



Overview

- ❖ **Introduction**
- ❖ **Algorithm Description**
- ❖ **Compression Performance**
- ❖ **Complexity Analysis**
- ❖ **Conclusions**

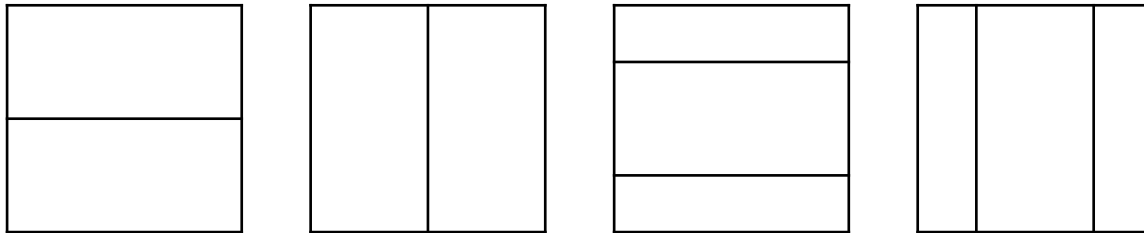
Introduction

❖ Key Coding Tools

- Partitioning structure based on Quad-tree plus binary tree and ternary tree (QTBTT)
- Linear interpolation intra prediction (LIP)
- Motion predictor candidate refinement (MPCR)
- Affine prediction modifications
- Simplified Core Transform
- Simplified Secondary Transform

Algorithm description: Partitioning structure

- ❖ Quad-tree plus binary tree and ternary tree (QTBT)
- A CTU is recursively split into sub-blocks based on quad-tree as HM
- A leaf node of QT is further split by either binary tree or ternary tree
 - ✓ Binary tree: $2N \times N$, $N \times 2N$
 - ✓ Ternary tree: $2N \times nNn$, $nNn \times 2N$



- Signaling
 - ✓ 2 bins for split type; No split, binary tree, or ternary tree
 - ✓ 1 bin for split direction; horizontal or vertical

Algorithm description: Partitioning structure

- Proposed QTBT parameters

QTBT parameters	I slices		P and B slices
CTUSize	128x128		128x128
	Luma	Chroma	
MinQTSIZE	8x8	4x4	8x8
MaxBTTSIZE	32x32	64x64	128x128
MaxBTDepth	3	3	3
MinBTTSIZE	4	4	4

- ✓ CTU size: the root node size of a quad-tree, the same concept as in HEVC
- ✓ MinQTSIZE: the minimum allowed the leaf node size of a quad-tree
- ✓ MaxBTTSIZE: the maximum allowed the root node size of binary and ternary trees
- ✓ MaxBTDepth: the maximum allowed depth of binary and ternary trees
- ✓ MinBTTSIZE: the minimum allowed leaf node size of binary and ternary trees

Algorithm description: Linear Intra prediction

❖ Generation of the opposite side reference sample array

- Step 1: compute the intensity value of the bottom right (BR) sample

$$BR(W, H) = \frac{W \times R(2W - 1, -1) + H \times R(-1, 2H - 1)}{(W + H)}$$

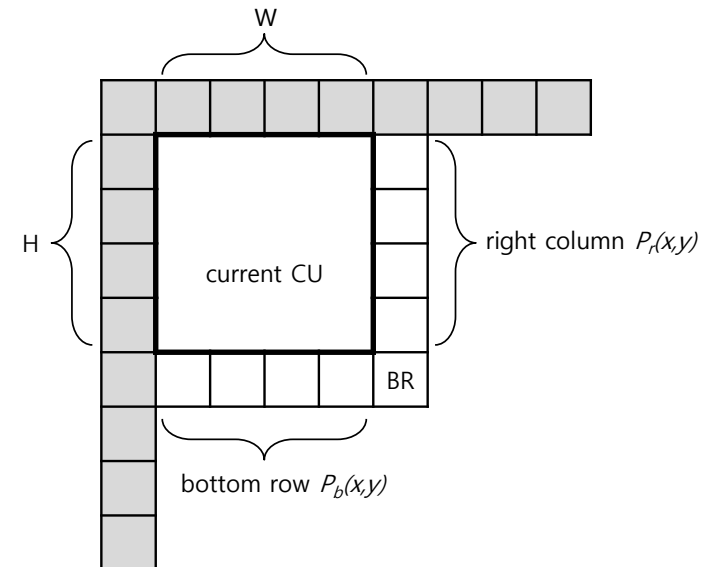
- $R(x, y)$: intensity value of the reconstructed neighboring sample at (x, y)
- W : block width, H : block height

- Step 2: generate the intensity values of the bottom row and the right column

$$P_r(W, y) = \frac{(H - 1 - y) \times R(W, -1) + (y + 1) \times BR(W, H)}{H}$$

$$P_b(x, H) = \frac{(W - 1 - x) \times R(-1, H) + (x + 1) \times BR(W, H)}{W}$$

- $P_b(x, y)$: intensity values of the bottom row samples
- $P_r(x, y)$: intensity values of the right column samples

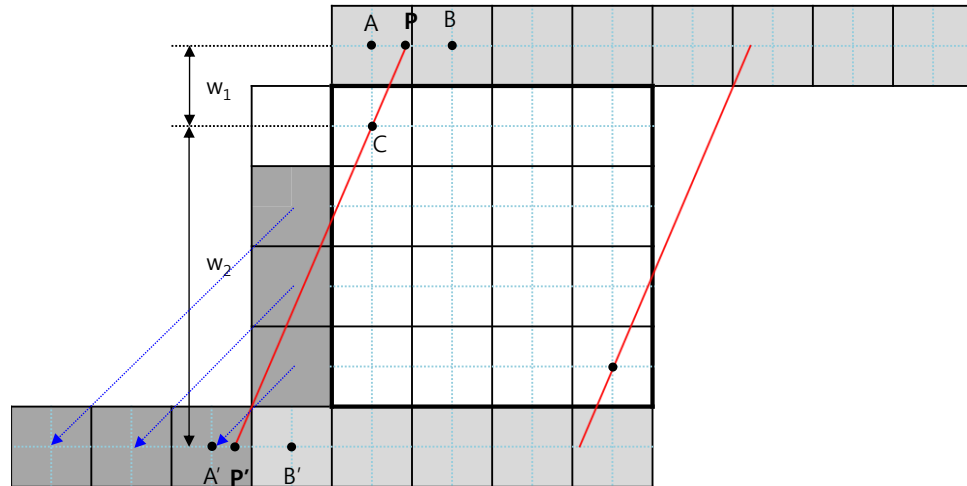


Algorithm description: Linear Intra prediction

❖ Linear interpolation intra prediction (LIP)

$$C = \frac{(W_1 \times P + W_2 \times P')}{(W_1 + W_2)}$$

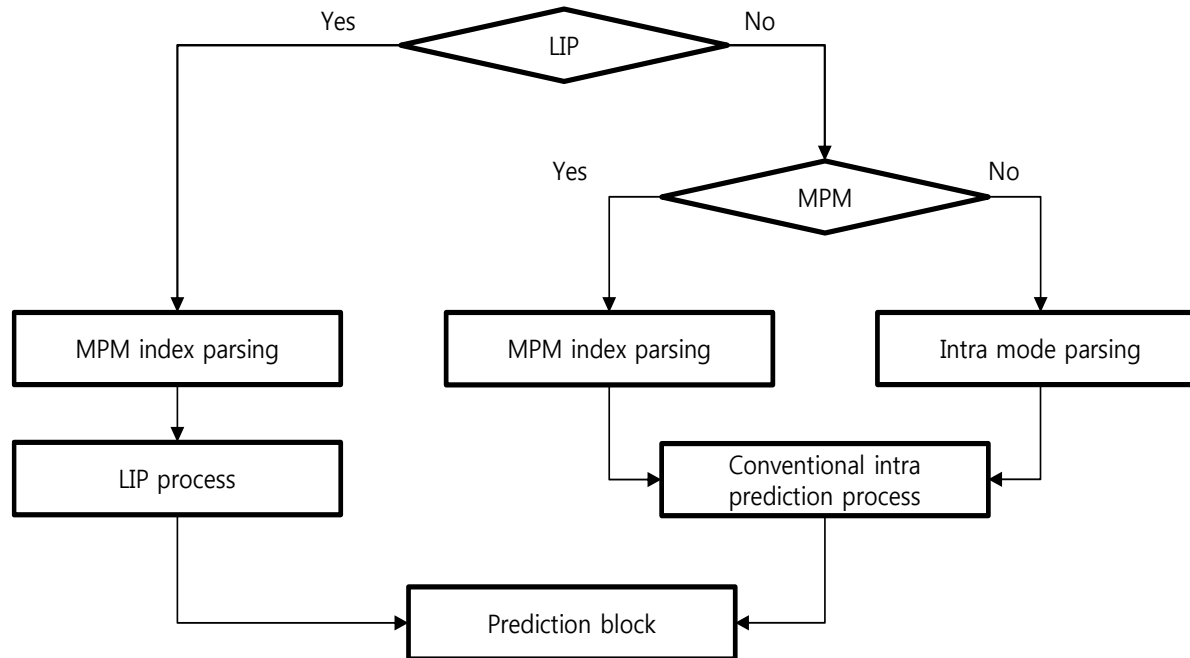
- C: predicted sample value
- P: predictor generated from the reference sample location
- P': predictor generated from the opposite side reference sample location
- W_1 and W_2 : weighting factors



Algorithm description: Linear Intra prediction

❖ Linear interpolation intra prediction mode signaling

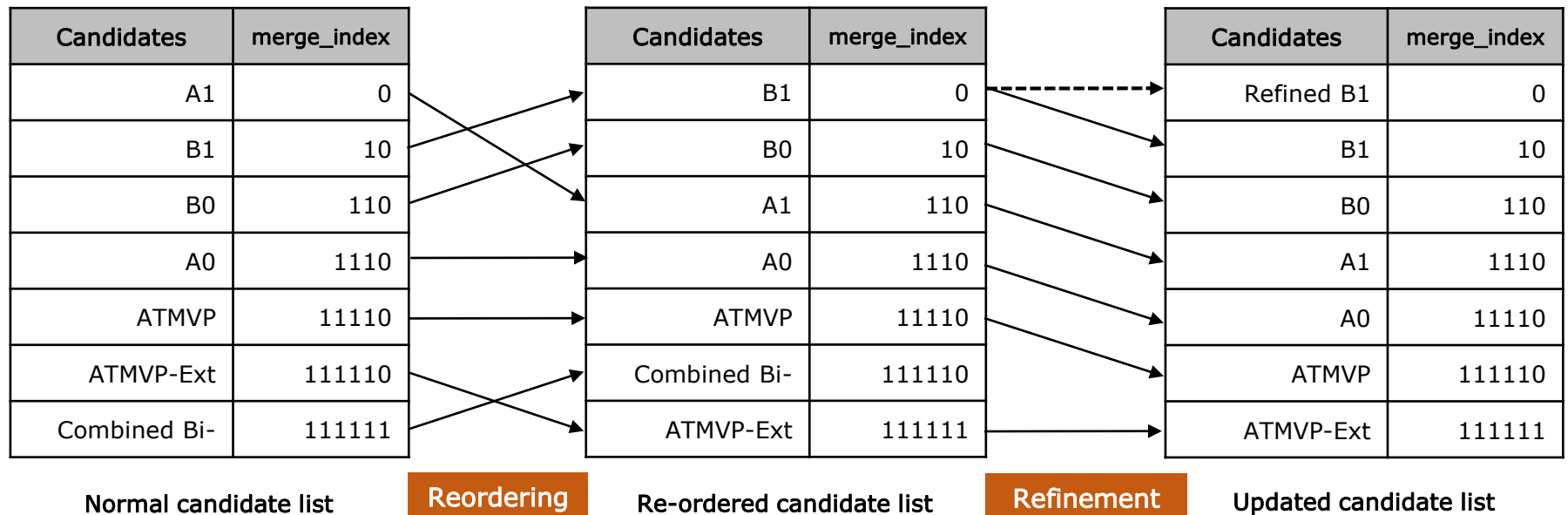
- CU level flag
- MPM flag checking is skipped when LIP mode is active



Algorithm description: Inter prediction (MPCR)

❖ Motion Predictor Candidate Refinement (MPCR)

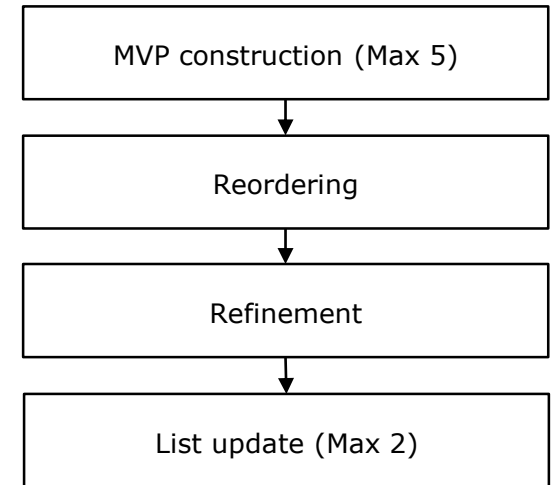
- Merge candidate list is reordered based on template matching cost
- The first candidate is refined and added to the list
- Remove the last candidate in the normal candidate list
- Replaced FRUC (no FRUC flag)



Algorithm description: Inter prediction (MPCR)

❖ Extension to AMVP

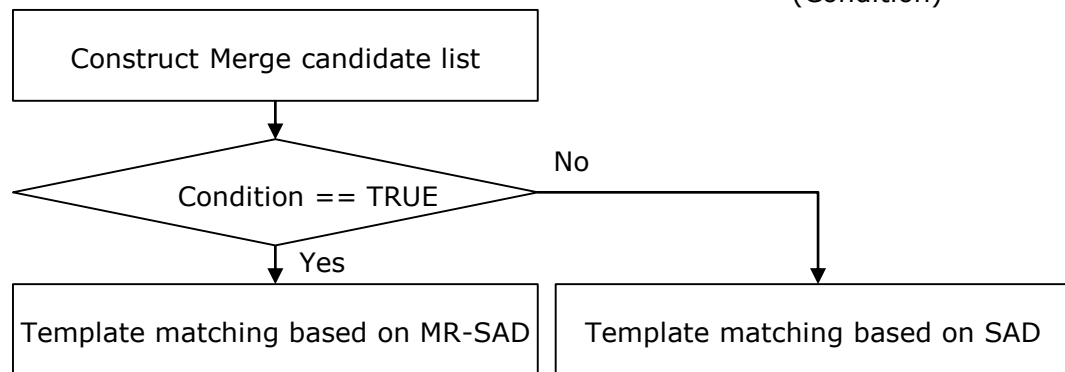
- Candidates are reordered and refined same as on MERGE
- MVP candidate construction process is modified
 - MVP candidate checking order is unified to the MERGE method
 - Availability checking order is modified
 - MVPs are constructed up to 5
- No changes on syntax elements



Proposed candidate construction method

❖ Interaction between MPCR and LIC

- Derive LIC flag for the MERGE candidates same as JEM.
- MR-SAD is used instead of SAD when the LIC flag of any candidate in the candidate list is true. (Condition)



Proposed harmonization with LIC

Algorithm description: Inter prediction (Affine)

❖ Affine merge candidate list construction

- Construct up to two candidates from neighboring affine coded blocks.
- Affine merge mode is skipped when there is no neighboring affine coded block.
- Affine merge index is signaled only when the available neighboring affine coded block is more than one (Figure 2)

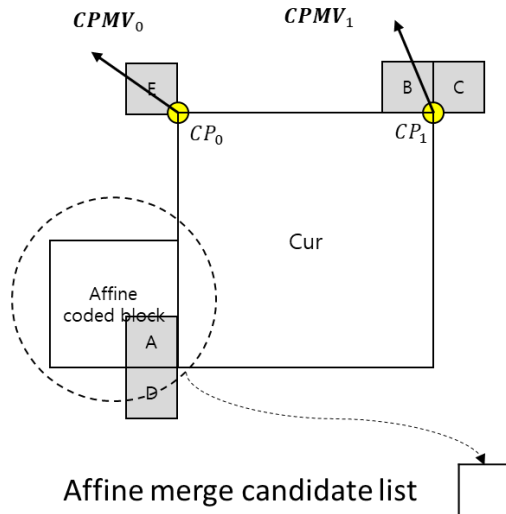


Figure 1

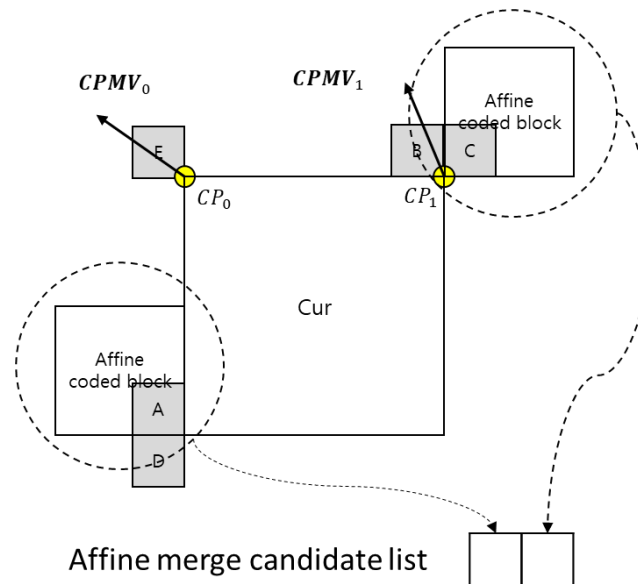


Figure 2

Algorithm description: Inter prediction (Affine)

❖ Affine MVP candidate list construction (Affine inter)

- Construct up to two candidates from the affine motion models of neighboring affine coded blocks (Figure 1 or 2).
- Perform JEM affine MVP process until the number of candidate is two

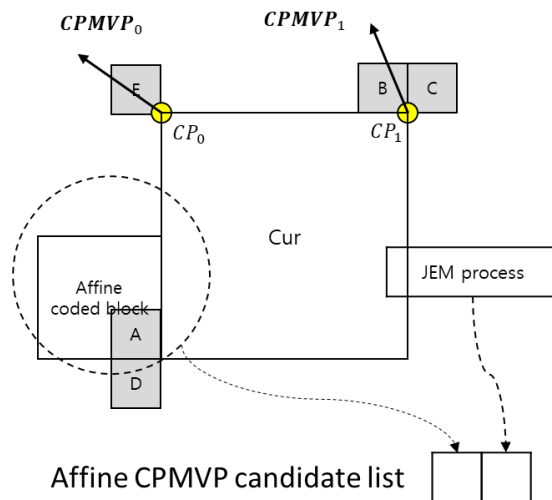


Figure 1

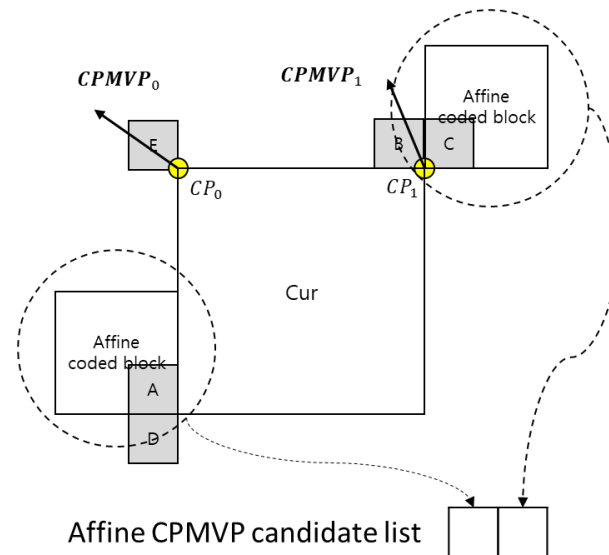
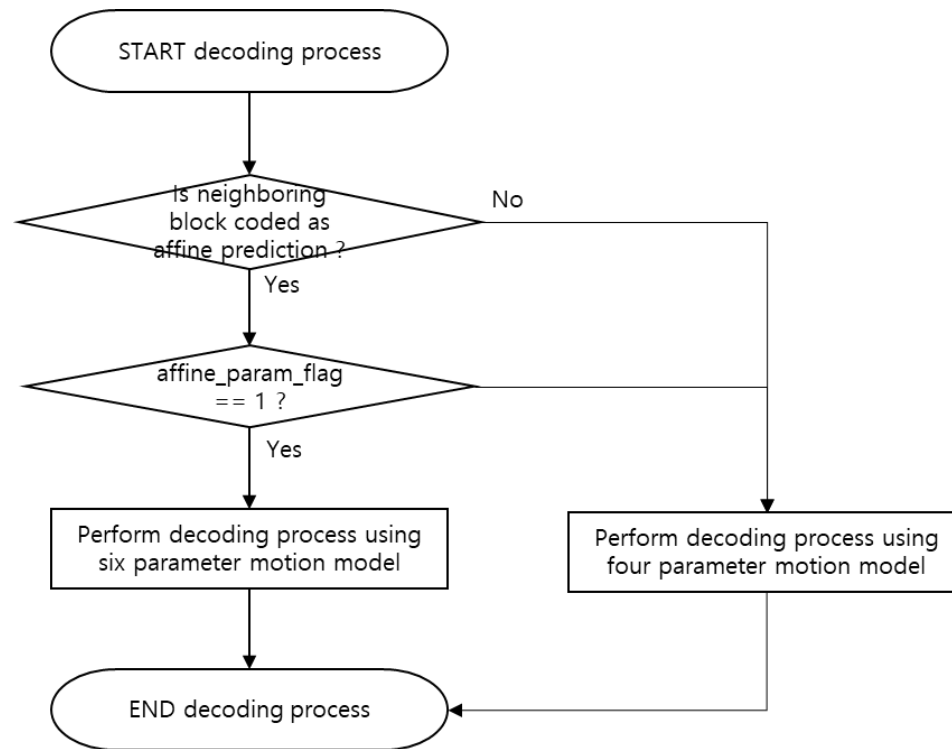
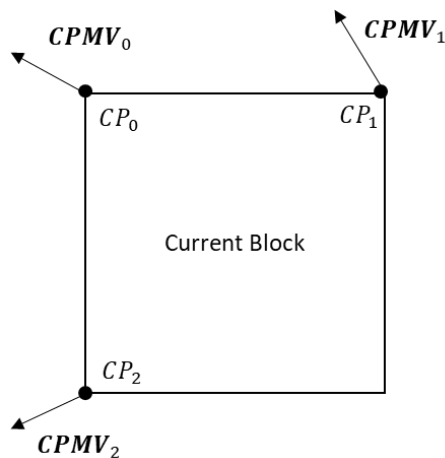


Figure 2

Algorithm description: Inter prediction (Affine)

❖ Adaptive four & six parameter motion model (Affine Inter)

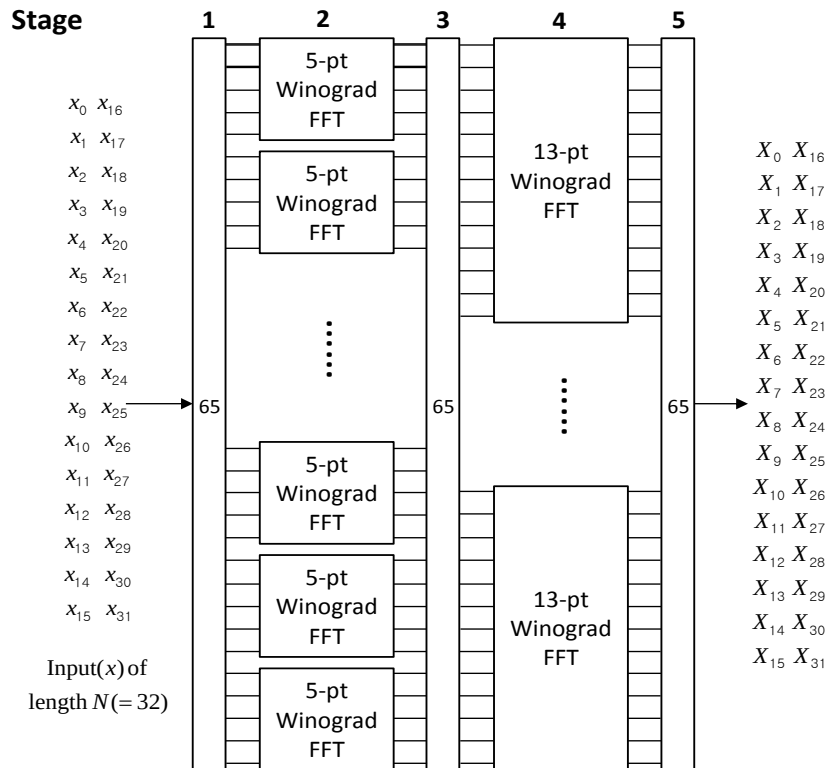
- Exploit six & four parameter motion models when at least one of the neighboring blocks is coded as affine prediction.
- Use four parameter motion model when no neighboring block is coded as affine prediction.



Algorithm description: Transform

❖ Primary transform

- Explicit Multiple Transform (EMT) with only DST7 and DCT8
- 32 x 32 max. EMT size
- Implementation of DST7 with Winograd FFTs
- DCT8: DST7 + simple pre-/post-processing



32 x 32 DST7 implementation

$$C_N^{VIII} = J_N S_N^{VII} D_N$$

$$[D_N]_{ij}, i, j=0, \dots, N-1 = \text{diag}((-1)^k), k=0, \dots, N-1 = \begin{cases} (-1)^i, & i=j \\ 0, & i \neq j \end{cases}$$

$$[J_N]_{ij}, i, j=0, \dots, N-1 = \begin{cases} 1, & j = N-1-i \\ 0, & \text{otherwise} \end{cases}$$

Relationship btw. DST7 and DCT8

Algorithm description: Transform

❖ Complexity characteristics of primary transform

- # of element operations required for transforming one row or column

Size	Proposed implementation		Matrix computation	
	# of mult.	# of add./sub.	# of mult.	# of add./sub.
8 x 8	21	77	64	56
16 x 16	42	125	256	240
32 x 32	87	279	1024	992

▪ Memory requirement

- # of kernel coefficients in all of DST7, DCT8, DST1, and DCT8 matrices of size up-to 64x64 (excl. 4x4)
 - $4 \times (8 \times 8 + 16 \times 16 + 32 \times 32 + 64 \times 64) = 5440$
- # of coefficients in all the Winograd FFTs employed

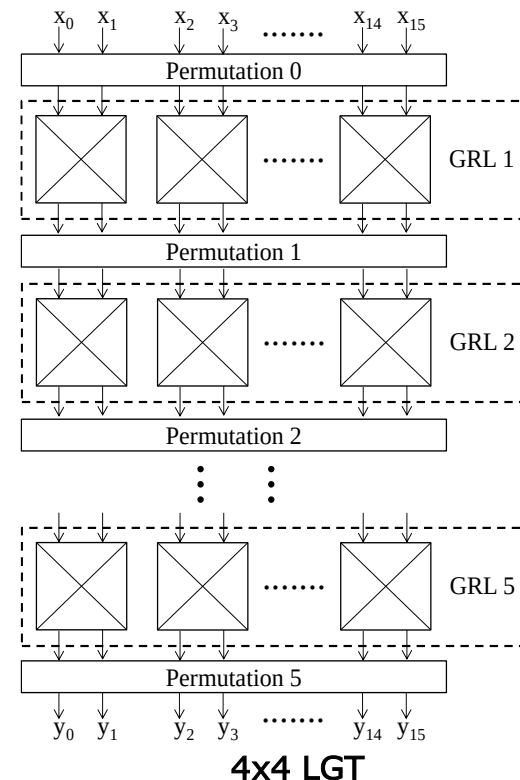
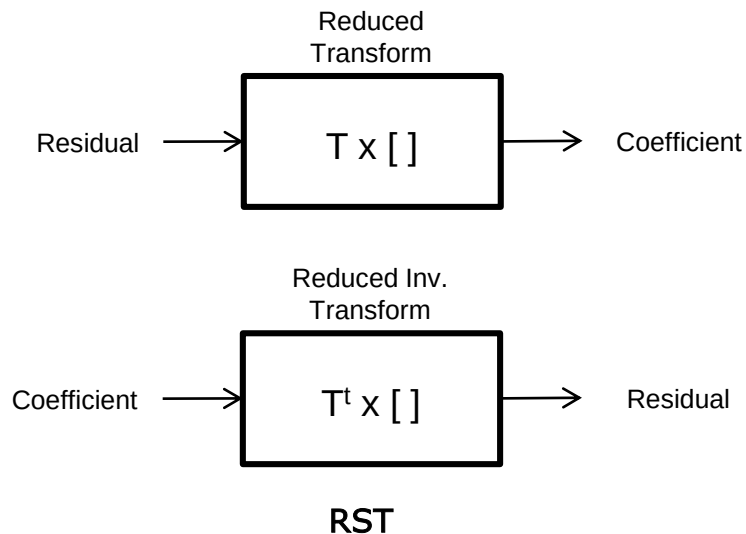
FFT length	# of coefficients
3	2
5	5
11	21
13	21
17	21
Total	70

Algorithm description: Transform

❖ Secondary transform

- 8x8 Reduced Secondary Transform (RST)
 - 16 bases of the 64 dimensional space (16x64 matrix)
 - ST_idx is inferred to be zero, if a non-zero coeff. in scanning positions from 17 to 64 is found
- 4x4 Layered Givens Transform (LGT)
 - 5 Givens rotation layers (GRLs) and 6 permutations

$$T_{R \times N} = \begin{bmatrix} t_{11} & t_{12} & t_{13} & \dots & t_{1N} \\ t_{21} & t_{22} & t_{23} & \dots & t_{2N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ t_{R1} & t_{R2} & t_{R3} & \dots & t_{RN} \end{bmatrix}$$



Algorithm description: Transform

❖ Complexity characteristics of Secondary transform

- 8x8 RST
 - **Fully implemented using direct non-square matrix multiply** (reduced memory complexity and computational complexity by 4 times compared to 8x8 NST)

	Memory Complexity (Bytes)	# of Multiplication	Multiple iterations
JEM NSST	816	3072	4-iterations
RST	1152	1152	No

- 4x4 layered Givens Transform (LGT)
 - 37.5% less computational load than JEM NSST (∵ 8 GRLs → 5 GRLs)
 - The increased memory usage can be regarded as the cost for reduction of computation.

Transform	Givens rotation	Permutation	Total
4x4 JEM NSST	512 bits	64 bits	576 bits
4x4 LGT	320 bits	384 bits	704 bits

Compression performance

- **Average bit-saving 34.75% for CS1 and 26.06% for CS2 (Compared to HM16.6)**

	Class A	Class B	Average
CS1	-37.60	-31.89	-34.75
CS2		-26.05	-26.06

- **Average bit-saving 2.52% for CS1 and 2.09% for CS2 (Compared to JEM7.0)**

	Class A	Class B	Average
CS1	-3.13	-1.91	-2.52
CS2		-2.09	-2.09

Complexity Analysis

- **Average decoding time 644% for CS1 and 577% for CS2 (Compared to HM16.6)**

	Enc.	Dec.
CS1	1523%	644%
CS2	1913%	577%

- **Average decoding time 84% for CS1 and 86% for CS2 (Compared to JEM7.0)**

	Enc.	Dec.
CS1	191%	84%
CS2	238%	86%

Conclusions

- ❖ In our response to the CfP, five key coding tools are developed considering higher compression efficiency minimizing complexity increase
- ❖ Our proposed coding tools are recommended to be considered in the core experiments for developing the next generation video coding standard with the JVET