

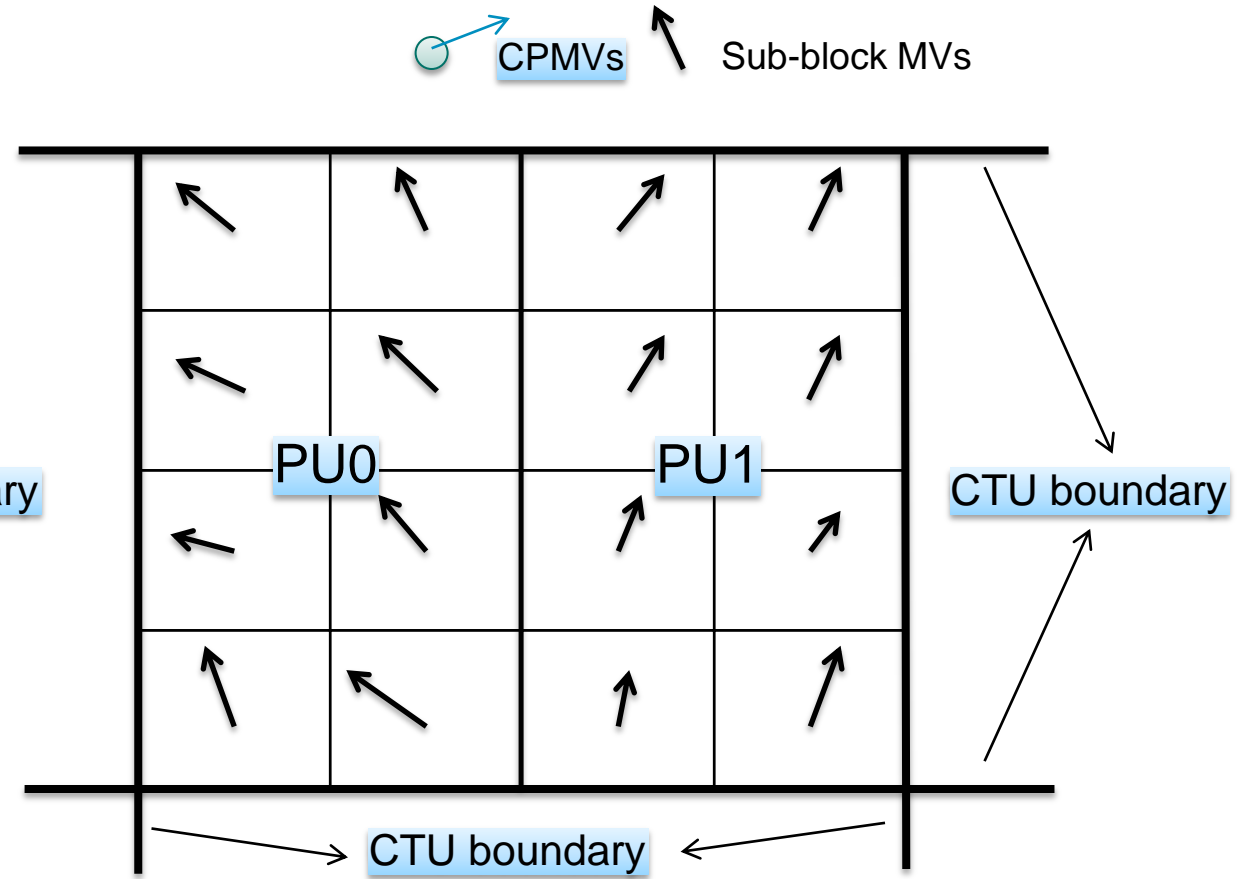
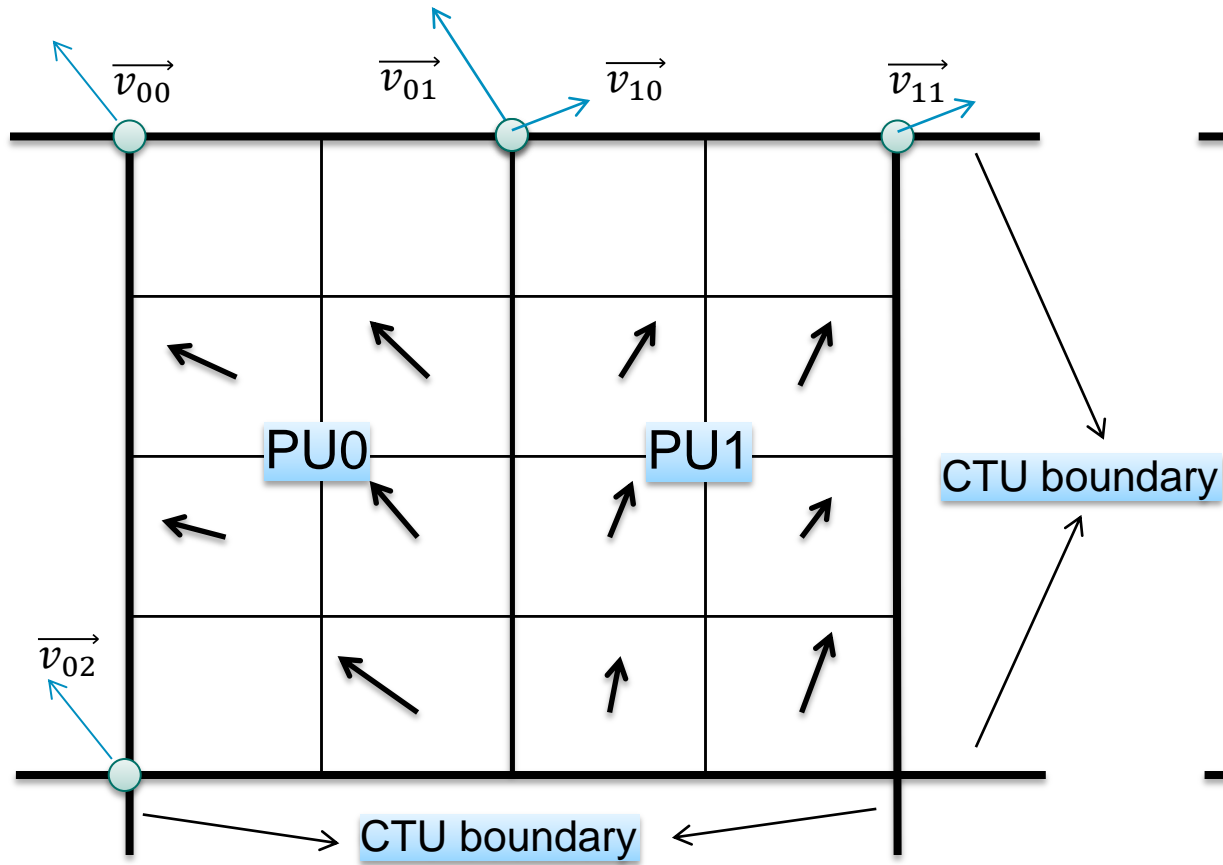
# A CLEAN UP FOR AFFINE MODE (JVET-L0047)



Minhua Zhou and Brian Heng

Broadcom Inc.

# Current Design of Affine Mode in VTM2.0.1



(a) MV field used for merge/skip/AMVP/TMVPs, affine inheritance

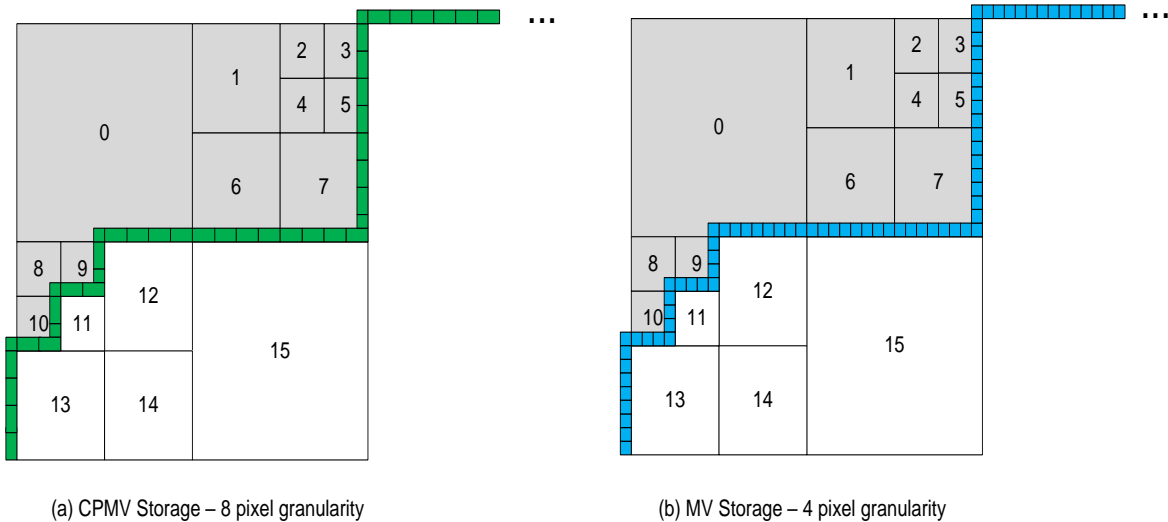
(b) MV field used for motion compensation

# Disadvantages of the Current Design of Affine Mode

- Sub-block motion vector field cannot be overwritten before it is consumed by the reference block pre-fetch and motion compensation
  - 1) Separate buffer is required to buffer the CPMVs (as opposed to embedding the CPMVs in the sub-block MV field as currently implemented in VTM2.0.1).
  - 2) More memory buffer needed due to the fact that the CPMVs are stored as TMVPs and the non-adjacent CPMVs cannot be disposed before they are compressed and written out.
  - 3) More conditional checking in the merge/skip/AMVP/affine merge/affine AMVP list derivation processes as fetch of neighboring spatial MV candidates needs to be switched back and forth between the CPMV buffer and sub-block MV buffer.
  - 4) Duplicated sub-block MV field derivation in VM2.0.1, one for the MC (computed on-the-fly), the other for the rest of process.

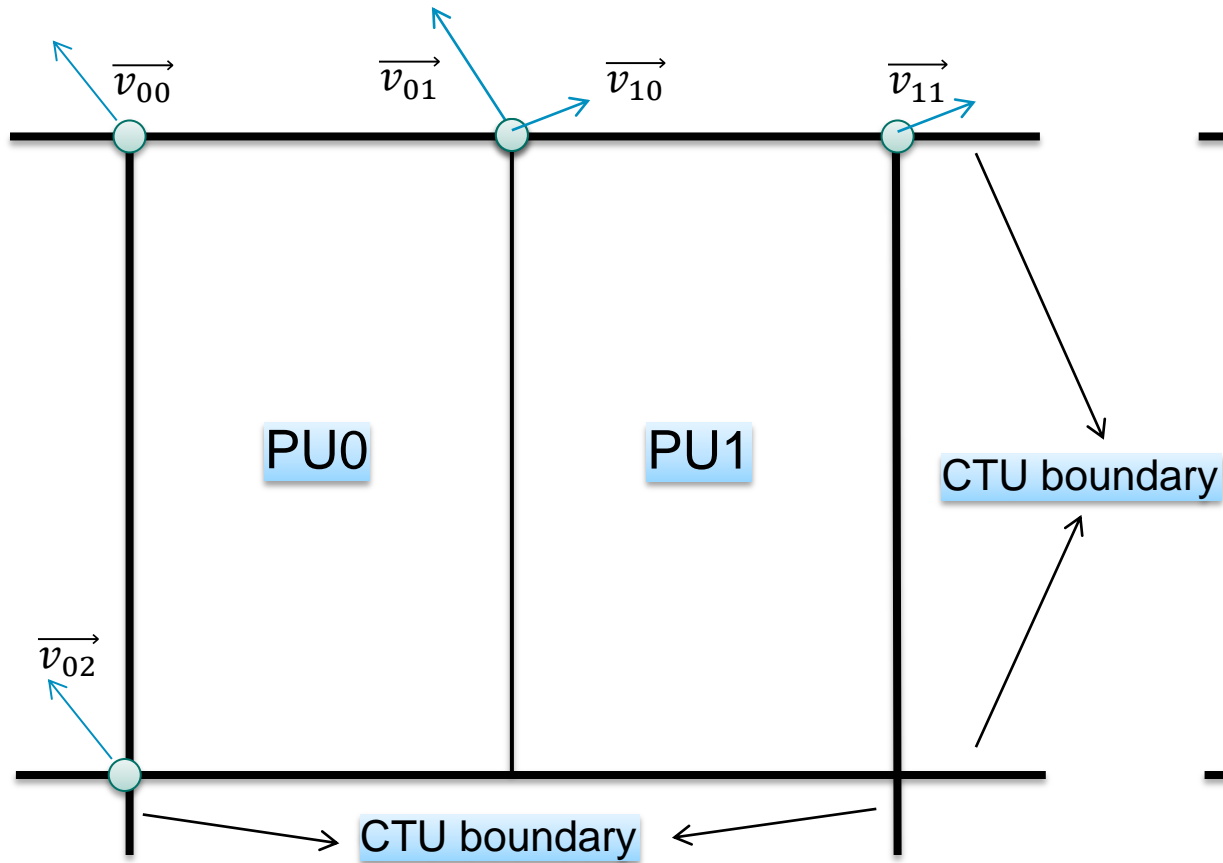
# Local Memory Usage Analysis for the Current Design

- Using local col/row CPMV and sub-block MV buffers and 64x64 decoder pipeline is more cost efficient for implementation
  - Frame-based or 128x128 CTU based implementation is unlikely used in product

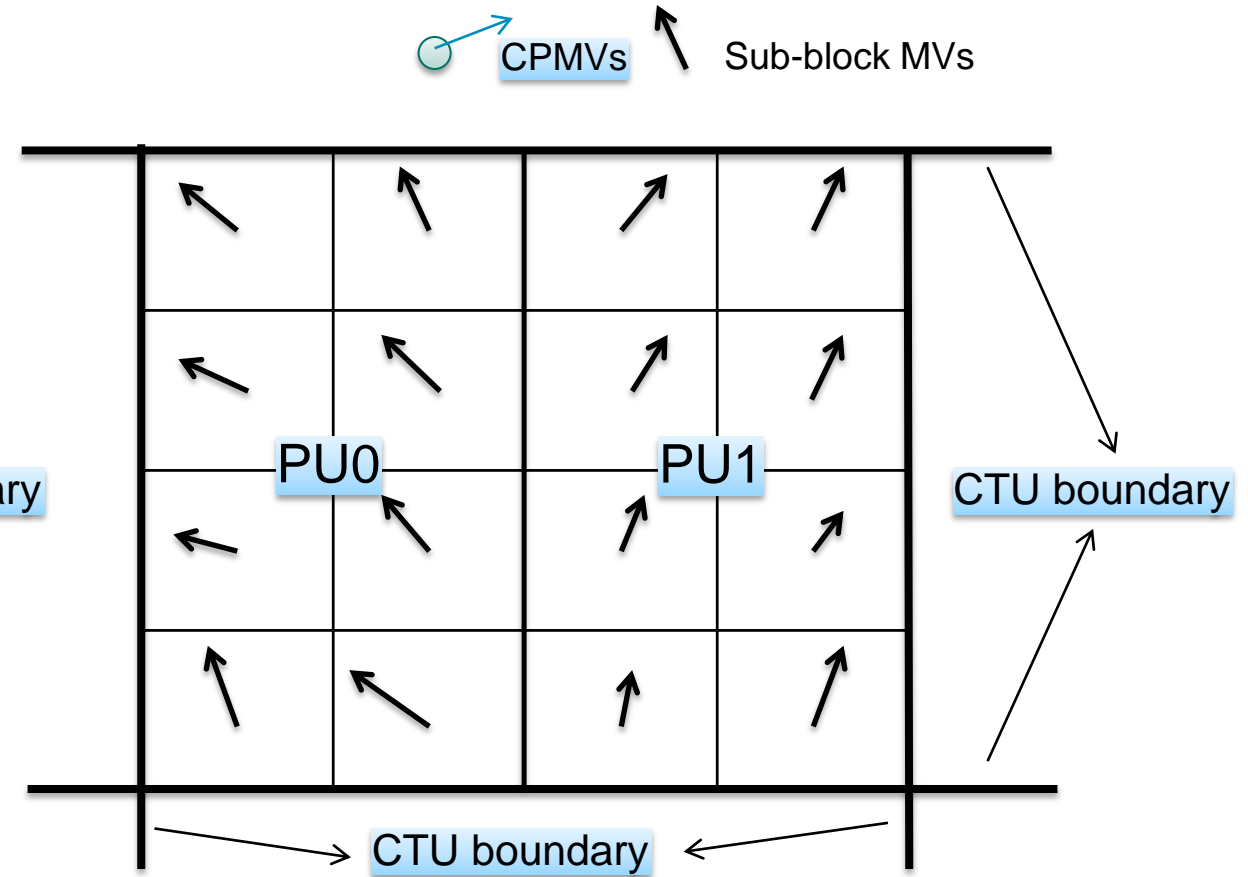


Method	Local sub-block MV buffer (bytes)	Local CPMV buffer (bytes)	Local buffers for left/top-left/top neighboring MV contexts (bytes)	Local buffers for left/top-left/top neighboring CPMV contexts (bytes)	Total local memory size (bytes)
128x128 CTU based VTM2.0.1	8,192	6,144	256 (left only)	384 (left only)	14,976
64x64 based VTM2.0.1	2,048	1,536	768	1,152	5,504

# Proposed Clean-up (Clean-up Method 1)



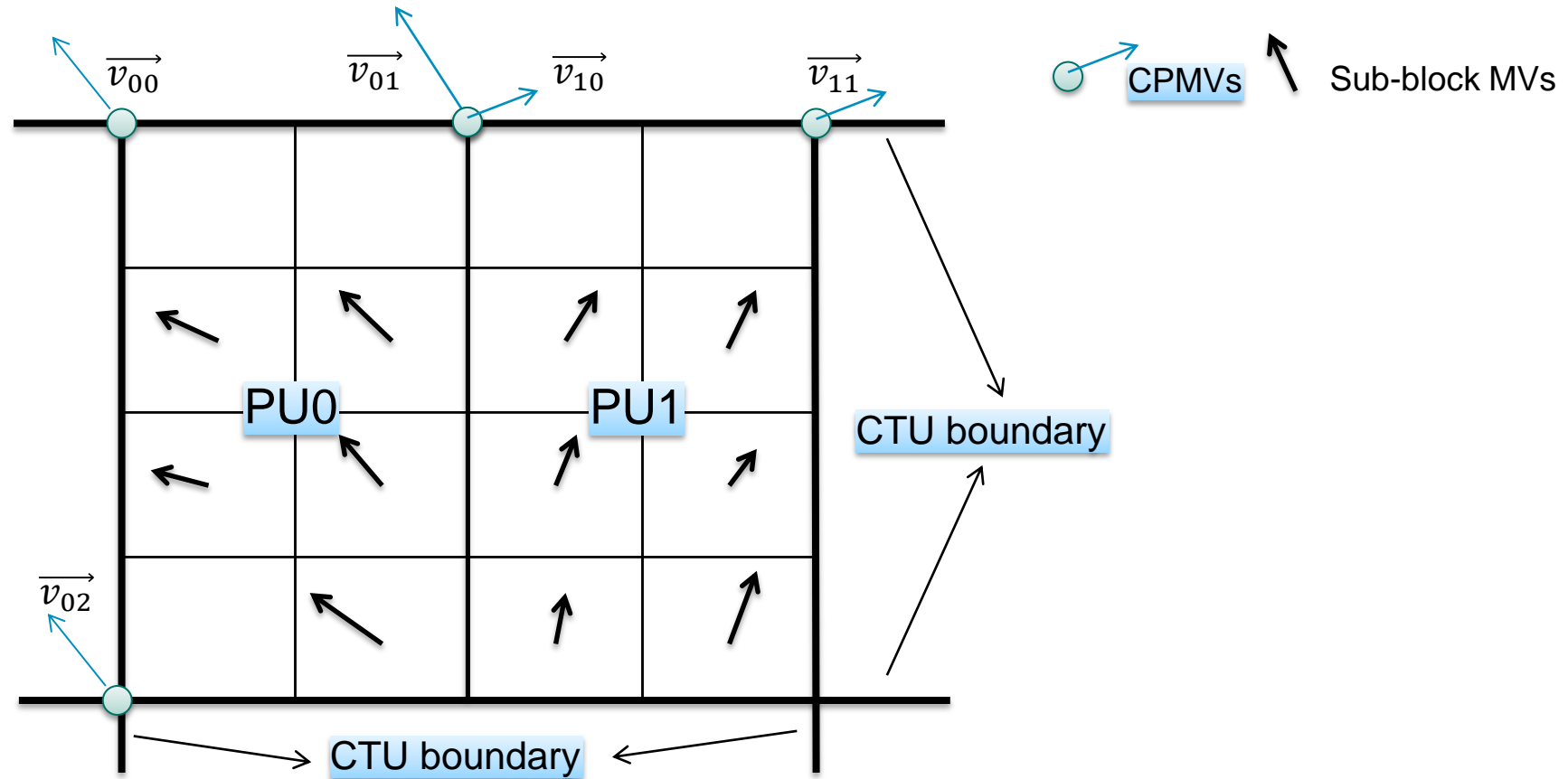
(a) MV field used for affine inheritance



(b) MV field used for motion compensation, merge/skip/AMVP/TMVPs

- *Non-adjacent CPMVs can be disposed because the CPMVs are not stored as TMVPs*

## Alternative Clean-up (Clean-up Method 2)



MV field used for merge/skip/AMVP/TMVPs, affine inheritance, and for motion compensation

- 2/3 of the local CPMV buffer can overlay on the local sub-block MV buffer

# Local Memory Usage Analysis for Different Methods

- Using local col/row CPMV and sub-block MV buffers and 64x64 decoder pipeline
- 2/3 of the local CPMV buffer overlays on the local sub-block MV buffer

Method	Local sub-block MV buffer (bytes)	Local CPMV buffer (bytes)	Local buffers for left/top-left/top neighboring MV contexts (bytes)	Local buffers for left/top-left/top neighboring CPMV contexts (bytes)	Total local memory size (bytes)
128x128 CTU based VTM2.0.1	8,192	6,144	256 (left only)	384 (left only)	14,976
64x64 based VTM2.0.1	2,048	1,536	768	1,152	5,504
64x64 based clean-up method 1	2,048		768	1,152	3,968
64x64 based clean-up method 2	2,048		768	384	3,200



# Experimental Results for Clean-up Methods 1 & 2

- Code base VTM2.0.1. Left method 1 (peak loss 0.07% in RA), right method 2 (peak loss 0.32 % in RA)

	Random Access Main 10				
	Over VTM-2.0.1				
	Y	U	V	EncT	DecT
Class A1	-0.01%	-0.04%	-0.05%	97%	97%
Class A2	0.03%	0.02%	-0.02%	97%	104%
Class B	0.00%	0.06%	0.06%	96%	101%
Class C	0.01%	0.12%	0.01%	99%	100%
Class E					
<b>Overall</b>	0.01%	0.05%	0.01%	97%	100%
Class D	0.03%	-0.04%	-0.01%	104%	103%
Class F (optional)	-0.01%	-0.02%	-0.02%	95%	94%

	Low delay B Main10				
	Over VTM-2.0.1				
	Y	U	V	EncT	DecT
Class A1					
Class A2					
Class B	0.03%	0.05%	-0.04%	102%	107%
Class C	0.00%	0.01%	-0.11%	88%	94%
Class E	-0.15%	0.44%	-0.14%	107%	107%
<b>Overall</b>	-0.03%	0.13%	-0.09%	99%	103%
Class D	0.05%	0.52%	-0.25%	112%	115%
Class F (optional)	-0.09%	0.08%	0.33%	96%	98%

	Low delay P Main10				
	Over VTM-2.0.1				
	Y	U	V	EncT	DecT
Class A1					
Class A2					
Class B	-0.02%	0.08%	-0.06%	92%	91%
Class C	0.02%	-0.11%	0.05%	98%	95%
Class E	0.09%	-0.06%	0.10%	96%	95%
<b>Overall</b>	0.02%	-0.02%	0.01%	95%	93%
Class D	0.04%	0.10%	-0.63%	102%	100%
Class F (optional)	0.04%	-0.05%	-0.11%	93%	94%

	Random Access Main 10				
	Over VTM-2.0.1				
	Y	U	V	EncT	DecT
Class A1	0.00%	0.13%	0.01%	93%	91%
Class A2	0.09%	0.02%	-0.06%	93%	93%
Class B	0.08%	0.09%	0.03%	99%	100%
Class C	0.07%	0.01%	0.02%	107%	114%
Class E					
<b>Overall</b>	0.06%	0.06%	0.01%	99%	100%
Class D	0.15%	0.00%	-0.03%	105%	113%
Class F (optional)	0.08%	0.06%	0.08%	101%	103%

	Low delay B Main10				
	Over VTM-2.0.1				
	Y	U	V	EncT	DecT
Class A1					
Class A2					
Class B	0.05%	0.03%	-0.18%	97%	102%
Class C	0.02%	-0.12%	-0.12%	84%	87%
Class E	-0.07%	-0.06%	0.02%	105%	106%
<b>Overall</b>	0.01%	-0.04%	-0.11%	95%	97%
Class D	0.04%	0.58%	0.42%	111%	109%
Class F (optional)	0.16%	0.32%	0.14%	103%	110%

	Low delay P Main10				
	Over VTM-2.0.1				
	Y	U	V	EncT	DecT
Class A1					
Class A2					
Class B	0.00%	0.26%	0.22%	95%	96%
Class C	0.10%	-0.04%	0.00%	97%	99%
Class E	-0.03%	0.46%	0.52%	89%	89%
<b>Overall</b>	0.03%	0.21%	0.22%	94%	95%
Class D	0.23%	0.34%	0.25%	98%	93%
Class F (optional)	0.19%	0.28%	-0.10%	102%	101%



# Recommendation

- Clean-up method 1 is recommended for a more consistent and implementation friendly design of affine mode without comprising coding efficiency
  - 1) Local memory saving by 1,536 bytes.
  - 2) Avoiding duplicated sub-block vector derivation.
  - 3) Less conditional checking in the merge/skip/AMVP/affine merge/affine AMVP list derivation of memory usage optimized decoder implementations.

*Thanks to InterDigital for cross-check*