

## JCTVC-Q0106 Rext HLS: On Lossless Coding



invention | collaboration | contribution

Xiaoyu Xiu, Yuwen He, Yan Ye  
InterDigital Communications, Inc.

17<sup>th</sup> JCT-VC meeting, Mar/Apr  
2014

# Introduction

- In Rext draft 6, lossy coding is enabled at sequence/picture level, whereas lossless coding is only enabled at CU level
  - Set `transquant_bypass_enabled_flag` at PPS level to 1
  - Set `cu_transquant_bypass_flag` at CU level to 1 → inverse quantization, inverse transform, deblocking, and SAO are all bypassed
  - When `transquant_bypass_enabled_flag` = 0, `cu_transquant_bypass_flag` is not sent and is inferred to be 0
- Proposal: enable sequence/picture level lossless coding for typical lossless applications with the following benefits
  - Simplified syntax parsing
  - More optimized decoder initialization
  - Encoder and decoder speed-up

# Option 1: signaling in PPS extension

pic_parameter_set_rbsp() {	Descriptor
...	
if( pps_extension_flag[ 0 ] ) {	
if( !transquant_bypass_enabled_flag )	
transquant_bypass_default_flag	u(1)
if( transform_skip_enabled_flag && !transquant_bypass_default_flag )	
log2_max_transform_skip_block_size_minus2	ue(v)
cross_component_prediction_enabled_flag	u(1)
if( !transquant_bypass_default_flag )	
chroma_qp_adjustment_enabled_flag	u(1)
if( chroma_qp_adjustment_enabled_flag ) {	
...	
}	
if( !transquant_bypass_default_flag ) {	
log2_sao_offset_scale_luma	ue(v)
log2_sao_offset_scale_chroma	ue(v)
}	
}	
...	
}	

**transquant\_bypass\_default\_flag** specifies the inferred value of **cu\_transquant\_bypass\_flag** when **transquant\_bypass\_enabled\_flag** is equal to 0. When **transquant\_bypass\_default\_flag** is not present, it is inferred to be equal to 0.

**cu\_transquant\_bypass\_flag** equal to 1 specifies that the scaling and transform process as specified in subclause 8.6 and the in-loop filter process as specified in subclause 8.7 are bypassed. When **cu\_transquant\_bypass\_flag** is not present, it is inferred to be equal to **transquant\_bypass\_default\_flag**.

# Option 1: bitstream conformance requirements

- PPS syntax elements related to transform, quantization, and loop filtering:
  - transform\_skip\_enabled\_flag
  - cu\_qp\_delta\_enabled\_flag
  - pps\_cb\_qp\_offset, pps\_cr\_qp\_offset
  - pps\_slice\_chroma\_qp\_offsets\_present\_flag
  - loop\_filter\_across\_tiles\_enabled\_flag
  - pps\_loop\_filter\_across\_slices\_enabled\_flag
  - pps\_scaling\_list\_data\_present\_flag
- Slice header syntax elements related to transform, quantization, and loop filtering:
  - slice\_sao\_luma\_flag
  - slice\_sao\_chroma\_flag
  - slice\_qp\_delta
- Proposed bitstream conformance requirements:
  - The above syntax elements shall be equal to 0 when transquant\_bypass\_default\_flag is equal to 1

## Option 2: signaling in SPS extension

seq_parameter_set_rbsp() {	Descriptor
...	
sps_extension_present_flag	u(1)
if( sps_extension_present_flag ) {	
for( i = 0; i < 1; i++ )	
sps_extension_flag[ i ]	u(1)
sps_extension_7bits	u(7)
if( sps_extension_flag[ 0 ] ) {	
<b>transquant_bypass_default_flag</b>	<b>u(1)</b>
transform_skip_rotation_enabled_flag	u(1)
transform_skip_context_enabled_flag	u(1)
...	
}	
if( sps_extension_7bits )	
while( more_rbsp_data( ) )	
sps_extension_data_flag	u(1)
}	
rbsp_trailing_bits( )	
}	

**transquant\_bypass\_default\_flag** specifies the inferred value of `cu_transquant_bypass_flag` when `transquant_bypass_enabled_flag` is equal to 0. When `transquant_bypass_default_flag` is not present, it is inferred to be equal to 0.

`cu_transquant_bypass_flag` equal to 1 specifies that the scaling and transform process as specified in subclause 8.6 and the in-loop filter process as specified in subclause 8.7 are bypassed. When `cu_transquant_bypass_flag` is not present, it is inferred to be equal to **transquant\_bypass\_default\_flag**.

## Option 2: bitstream conformance requirements

- SPS syntax elements related to transform, quantization, and loop filtering:
  - scaling\_list\_enabled\_flag
  - sample\_adaptive\_offset\_enabled\_flag
- PPS syntax elements related to transform, quantization, and loop filtering:
  - transform\_skip\_enabled\_flag
  - transquant\_bypass\_enabled\_flag
  - cu\_qp\_delta\_enabled\_flag
  - pps\_cb\_qp\_offset, pps\_cr\_qp\_offset
  - pps\_slice\_chroma\_qp\_offsets\_present\_flag
  - loop\_filter\_across\_tiles\_enabled\_flag
  - pps\_loop\_filter\_across\_slices\_enabled\_flag
  - pps\_scaling\_list\_data\_present\_flag
- Slice header syntax elements related to transform, quantization, and loop filtering:
  - slice\_qp\_delta
- Proposed bitstream conformance requirements:
  - The above syntax elements shall be equal to 0 when transquant\_bypass\_default\_flag is equal to 1

## Option 2: on transform\_tree() syntax

- transform\_tree() signals split\_transform\_flag, cbf\_luma, cbf\_cb and cbf\_cr flags, followed by the coefficients

transform_tree( x0, y0, xBase, yBase, log2TrafoSize, trafoDepth, blkIdx ) {	Descriptor
if( log2TrafoSize <= Log2MaxTrafoSize && log2TrafoSize > Log2MinTrafoSize && trafoDepth < MaxTrafoDepth && !( IntraSplitFlag && ( trafoDepth == 0 ) ) )	
split_transform_flag[ x0 ][ y0 ][ trafoDepth ]	ae(v)
....	
}	

- For lossless coding, bypassing TU quad-tree splitting can significantly speedup encoder by skipping the RDO process
  - Small performance loss
- Additional proposed bitstream conformance requirement:
  - When `transquant_bypass_default_flag` is equal to 1, `max_transform_hierarchy_depth_inter` and `max_transform_hierarchy_depth_intra` shall be 1, and `log2_diff_max_min_transform_block_size` in SPS shall be equal to 0

# Simulation settings

- The proposed SPS signaling (Option 2) was implemented in HM-13.0+RExt-6.0
- Two sets of simulations:
  - Set 1: bitstream restriction on transform splitting was not applied
  - Set 2: bitstream restriction on transform splitting was applied
- Option 1's performance and encoding/decoding speed should be similar to Set 1



# Simulation Results (1)

Set 1: same performance in AI, small performance gain in RA and LB

	Average bit-rate increase		
	AI	RA	LB
Class F	0.0%	-0.1%	-0.1%
Class B	0.0%	0.0%	0.0%
RGB 4:4:4 SC	0.0%	-0.1%	-0.1%
RGB 4:4:4 Animation	0.0%	0.0%	0.0%
YCbCr 4:4:4 SC	0.0%	-0.1%	-0.1%
YCbCr 4:4:4 Animation	0.0%	0.0%	0.0%
RangeExt	0.0%	0.0%	0.0%
RGB 4:4:4 SC (Optional)	0.0%	-0.1%	-0.3%
YCbCr 4:4:4 SC (Optional)	0.0%	-0.1%	-0.4%
Enc Time[%]	99%	99%	100%
Dec Time[%]	88%	90%	89%

More than 10% decoder speed up

## Simulation Results (2)

Set 2: AI 0.5%, RA 0.6%, LB 0.6% loss for mandatory sequences

	Average bit-rate increase		
	AI	RA	LB
Class F	0.5%	0.6%	0.5%
Class B	0.1%	0.2%	0.2%
RGB 4:4:4 SC	1.3%	1.5%	1.2%
RGB 4:4:4 Animation	0.2%	0.5%	0.6%
YCbCr 4:4:4 SC	1.4%	1.4%	1.4%
YCbCr 4:4:4 Animation	0.1%	0.4%	0.5%
RangeExt	0.0%	0.1%	0.1%
RGB 4:4:4 SC (Optional)	1.5%	1.3%	2.6%
YCbCr 4:4:4 SC (Optional)	1.5%	1.7%	0.3%
Enc Time[%]	70%	65%	68%
Dec Time[%]	72%	79%	73%

Significant encoder and decoder speed up

# Conclusion

- Typical lossless video applications use sequence/picture level lossless coding
- Propose to enable high level signaling of lossless coding
  - Add `transquant_bypass_default_flag` in PPS or SPS extension
  - Add bitstream conformance constraints to syntax elements related to transform, quantization, deblocking, and SAO
- Simulations show that
  - Without TU splitting constraint, >10% decoding speedup and small performance gain
  - With TU splitting constraint, >30% encoding speedup, >20% decoding speedup, and ~0.5% performance loss