

# TOSHIBA

Leading Innovation >>>

---

JCTV-VC-C084:

Coding efficiency report of modification  
by TMuC draft005

TOSHIBA

# Summary

---

- **There was an agreement to move ALF filtering control map (ALF\_flag) to slice header at the Dresden meeting**
- **The text of TMuC draft005 incorporated the modification**
- **The software has been submitted to the SVN server as Toshiba branch of TMuC v0.8**
- **Experimental results of this modification is reported**
  - Coding efficiency loss is 0.04% on average, which corresponds to less than 1.5% (negligible) loss of overall coding efficiency gain of QC\_ALF.
  - Encoder implementation is simplified
  - Verified by Qualcomm (JCTVC-C243)

# Modification by TMuC draft005

## (Problem statement)

- Signaling ALF\_flag in CU header requires interleaving of the decision results of ALF\_flag after encoding whole slice at encoder. This interleaving makes encoder complex.

### 4.1.9 Coding unit syntax

[Ed. preliminary draft]

	C	Descriptor
coding_unit(x0, y0, currCodingUnitSize) {		
if(x0+currCodingUnitSize < PicWidthInSamples, && y0+currCodingUnitSize < PicHeightInSamples, && currCodingUnitSize > MinCodingUnitSize)		
split_coding_unit_flag	2	u(1) ae(v)
if(!split_coding_unit_flag && currCodingUnitSize == AlfMinCtrlCodingUnitSize)    (!split_coding_unit_flag && currCodingUnitSize > AlfCtrlMinCodingUnitSize)		
alf_flag	2	u(1) ae(v)
if(split_coding_unit_flag) {		
splitCodingUnitSize = currCodingUnitSize >> 1		

## (Modification)

- Signal ALF\_flag in slice header
- Gathering all ALF\_flags and signal them continuously
- Signal the number of ALF\_flags (add some **overhead**)

# Modification by TMuC draft005 (cont'd)

## 5.1.11.3 → Adaptive loop filter parameter syntax

Code Snippet	C	Descriptor
alf_param() {	C	Descriptor
adaptive_loop_filter_flag	2	u(1)
if (adaptive_loop_filter_flag) {		
.....		
→ <b>alf_cu_control_flag</b>	2	u(1)
if (alf_cu_control_flag) {		
→ <b>alf_cu_control_max_depth</b>	2	ue(v)
→ <u>if (alf_cu_control_max_depth) {</u>		
→ <u><b>alf length cu control info</b></u>	2	u(v)
→ }		
→ <u>for (i=0; i&lt;NumAlfCuFlag; i++)</u>		
→ <u><b>alf cu flag[i]</b></u>	2	u(1)   ae(v)
→ }		
→ }		
→ }		
}		

# Experimental results

---

- **Conditions**

- JCTVC-B300 and JCTVC-B310\_r3 (TMuC 0.8)
- High Efficiency, Random Access and Low Delay cases
- Anchor (reference): QC\_ALF, ALF\_flag signaled in CU header

High Efficiency	Random Access	Low Delay
Class A	0.01	N/A
Class B	0.03	0.07
Class C	0.04	-0.04
Class D	-0.02	0.03
Class E	N/A	0.26
Total	0.02 (-3.7)	0.07 (-5.2)
Encoding time	100	100
Decoding time	100	100

Values in the parenthesis are  $\Delta$ BD-rate of QC\_ALF (verified by TE12)

# Experimental results (backup)

- **Conditions**

- JCTVC-B300 and JCTVC-B310\_r3 (TMuC 0.8)
- High Efficiency, Random Access and Low Delay cases
- Anchor (reference): QC\_ALF, ALF\_flag signaled in CU header

High Efficiency	Random Access	Low Delay
Class A	0.01 (-4.7)	N/A
Class B	0.03 (-5.3)	0.07 (-5.6)
Class C	0.04 (-2.5)	-0.04 (-3.9)
Class D	-0.02 (-1.7)	0.03 (2.6)
Class E	N/A	0.26 (-7.7)
Total	0.02	0.07
Encoding time	100	100
Decoding time	100	100

Note: Values in parenthesis are  $\Delta$ BD-rate of the anchor (QC\_ALF on) against QC\_ALF\_off. Therefore, values in the parenthesis are slightly different from TE12 results.

# Conclusion

---

- **The text of TMuC draft005 incorporated the modification**
- **The software has been submitted to the SVN server as Toshiba branch of TMuC v0.8**
- **Experimental results of this modification is reported**
  - Coding efficiency loss is 0.04% on average, which corresponds to less than 1.5% (negligible) loss of overall coding efficiency gain of QC\_ALF.
  - Encoder implementation is simplified
  - Verified by Qualcomm
- **The results show the evidence of this modification to reduce the complexity at encoder without impact on the coding efficiency**