

CE7-related: Simplifying the derivation process of ctxInc for residual coding (JVET-P0587)

Shih-Ta Hsiang, Tzu-Der Chuang, Yu-Wen Huang, and Shawmin Lei

Presented by Tzu-Der Chuang

Overall Summary

- Propose a modified derivation process of ctxInc for residual coding by re-organizing the indexing of the context variables associated the syntax elements par_level_flag and abs_level_gtx_flag
- The three context variables for entropy coding the values of abs_level_gtx_flag[n][0], par_level_flag[n], and abs_level_gtx_flag[n][1] for a current coefficient are grouped in a subset and accessed jointly.
- No impact on BD bitrate performance

Introduction

- In VVC Draft 6, the context variables for coding the syntax elements `abs_level_gtx_flag` and `par_level_flag` are grouped into separate context sets
- The variable `ctxInc` for `abs_level_gtx_flag` and `par_level_flag` for residual coding with `transform_skip_flag` equal to 0 is derived using the same method as follows:
 - The variable `ctxOffset` is set equal to $\text{Min}(\text{locSumAbsPass1} - \text{locNumSig}, 4)$.
 - The variable `d` is set equal to `xC + yC`.
 - If `xC` is equal to `LastSignificantCoeffX` and `yC` is equal to `LastSignificantCoeffY`, `ctxInc` is derived as follows:

$$\text{ctxInc} = (\text{cldx} == 0 ? 0 : 21)$$

- Otherwise, if `cldx` is equal to 0, `ctxInc` is derived as follows:

$$\begin{aligned} \text{ctxInc} = 1 + \text{ctxOffset} + \\ (\text{d} == 0 ? 15 : (\text{d} < 3 ? 10 : (\text{d} < 10 ? 5 : 0))) \end{aligned}$$

- Otherwise (`cldx` is greater than 0), `ctxInc` is derived as follows:

$$\text{ctxInc} = 22 + \text{ctxOffset} + (\text{d} == 0 ? 5 : 0)$$

Proposed Method

- For non-TS residual coding, the context variables associated with `par_level_flag` and `abs_level_gtx_flag` are grouped into one set
- The three context variables for entropy coding the values of `abs_level_gtx_flag[n][0]`, `par_level_flag[n]`, and `abs_level_gtx_flag[n][1]` for a current coefficient at the scanning position `n` can be grouped into a subset and accessed jointly

Syntax Table

residual_coding(x0, y0, log2TbWidth, log2TbHeight, cIdx) {	Descriptor
....	
for(n = firstPosMode0; n >= 0 && remBinsPass1 >= 4; n--) {	
if(sig_coeff_flag[xC][yC]) {	
abs_level_gtx_flag[n][0]	ae(v)
remBinsPass1--	
if(abs_level_gtx_flag[n][0]) {	
par_level_flag[n]	ae(v)
remBinsPass1--	
abs_level_gtx_flag[n][1]	ae(v)
remBinsPass1--	
}	
if(lastSigScanPosSb == -1)	
lastSigScanPosSb = n	
firstSigScanPosSb = n	
}	
.....	
}	
.....	
}	

ctxInc = ctxSubSetOffset
+ ctxSyntaxId,
ctxSyntaxId = 0 .. 2

Proposed Modifications: Non-TS residual coding

- The context variables associated with `abs_level_gtx_flag[n][0]`, `par_level_flag[n]`, and `abs_level_gtx_flag[n][1]` for non-TS residual coding are grouped into one set with the variable `ctxInc` derived as follows:
 - The variable `ctxSyntaxId` is set equal to 0 if the syntax element is `abs_level_gtx_flag[n][0]`, equal to 1 if the syntax element is `par_level_flag`, and equal to 2, otherwise.
 - The variable `ctxOffset` is set equal to $\text{Min}(\text{locSumAbsPass1} - \text{locNumSig}, 4)$.
 - The variable `d` is set equal to `xC + yC`.
 - If `xC` is equal to `LastSignificantCoeffX` and `yC` is equal to `LastSignificantCoeffY`, `ctxInc` is derived as follows:
$$\text{ctxInc} = ((\text{cldx} == 0 ? 0 : 21)) * 3 + \text{ctxSyntaxId}$$
 - Otherwise, if `cldx` is equal to 0, `ctxInc` is derived as follows:
$$\text{ctxInc} = (1 + \text{ctxOffset} + (\text{d} == 0 ? 15 : (\text{d} < 3 ? 10 : (\text{d} < 10 ? 5 : 0)))) * 3 + \text{ctxSyntaxId}$$
 - Otherwise (`cldx` is greater than 0), `ctxInc` is derived as follows:
$$\text{ctxInc} = (22 + \text{ctxOffset} + (\text{d} == 0 ? 5 : 0)) * 3 + \text{ctxSyntaxId}$$