



Adaptive Loop Filter with Zero Pixel Line Buffers for LCU-based Decoding

Ching-Yeh Chen, Chia-Yang Tsai, Chih-Ming Fu, Yu-Wen Huang, and Shawmin Lei



Overall Summary

- Repetitive padding at virtual boundaries to remove pixel line buffers of adaptive loop filter (ALF)
 - To filter a pixel, only pixels on the same side of a virtual boundary as the to-be-filtered pixel can be used in the ALF process.
 - Unified filter shape (5x5D, 7x7D, 9x9D for luma and 7x7D for chroma)
 - A boundary smoothing technique is applied to remove visual artifacts.
- Performance
 - Removed all ALF line buffers (7 luma and 7 chroma lines in HM-3.0) for LCU-based decoding, as well as for picture-based decoding
 - Achieved the same subjective quality as the HM-3.0 anchor

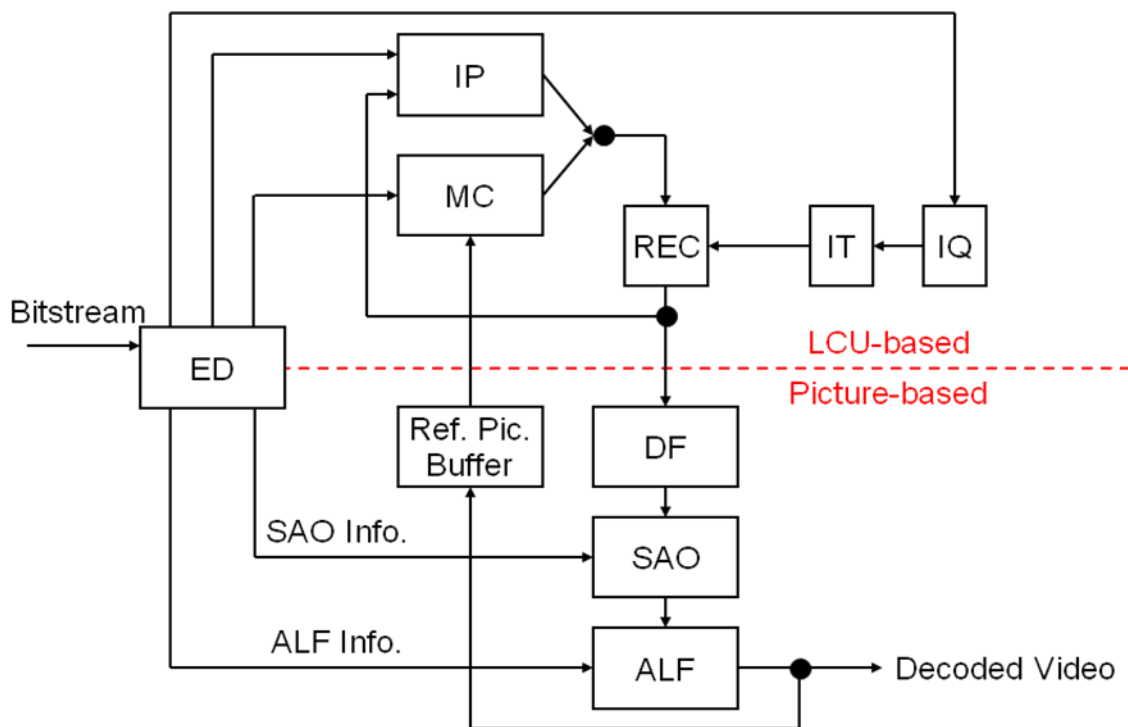
Anchor: JCTVC-E700	HE-AI	HE-RA	HE-LD
BD-Rate	0.1%	0.0%	-0.2%
Encoding Time	100%	100%	100%
Decoding Time	99%	100%	100%

Outline

- Picture-based and LCU-based decoding
- Analysis of line buffers for in-loop filtering in HM-3.0
- Proposed virtual boundaries
- Simulation results
- Conclusion

Current Implementation in HM-3.0

- LCU-based processing is used for intra prediction, motion compensation, inverse quantization, inverse transform, and reconstruction.
- Picture-based processing is used for deblocking filter (DF), sample adaptive offset (SAO), and adaptive loop filter (ALF).

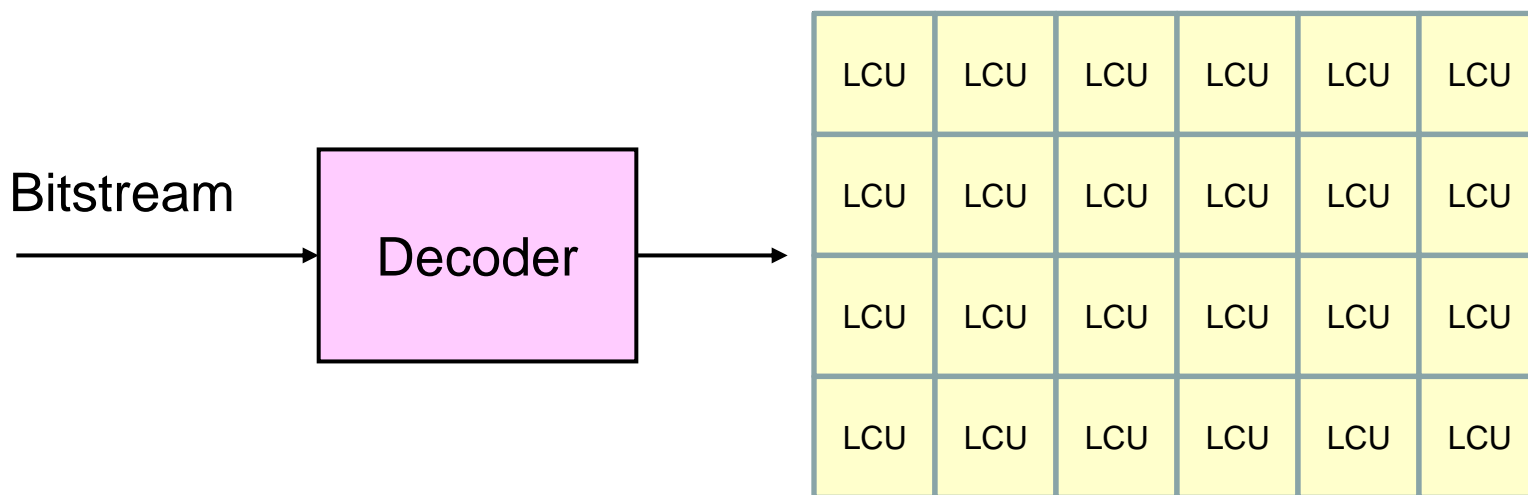


Picture-based Decoding

- Store the entire picture after REC
- Access the REC picture buffer to perform DF and store the entire picture after DF
- Access the DF picture buffer to perform SAO and store the entire picture after SAO
- Access the SAO picture buffer to perform ALF and store the entire picture after ALF
- Need additional picture buffers
- May be okay for PC-based software but very bad for embedded software and hardware
 - On-chip picture buffers usually require unacceptable area cost.
 - Off-chip picture buffers require significantly longer access time and higher power consumption in comparison with on-chip buffers and consumes a lot of precious external memory bandwidth

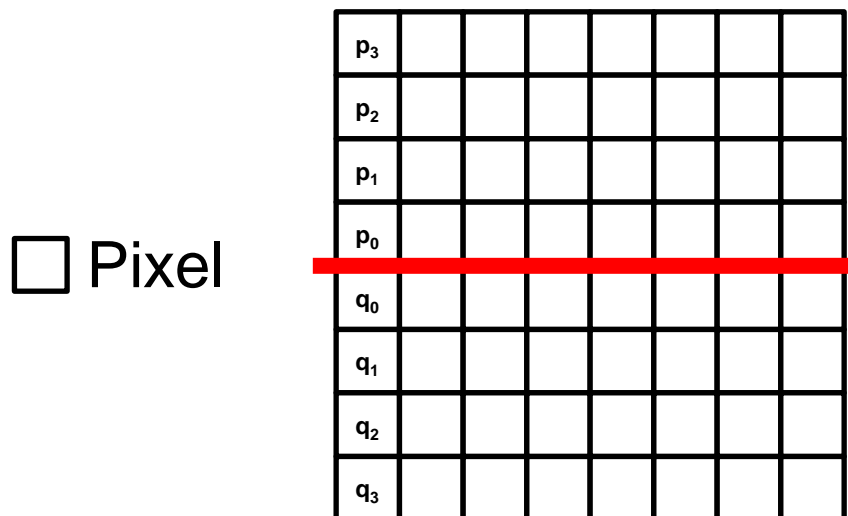
LCU-based Decoding

- Input bitstream
- Output decoded results LCU by LCU
- No picture buffers except for reference pictures
 - To minimize the external memory storage size and data access
- Mainstream for embedded software and hardware



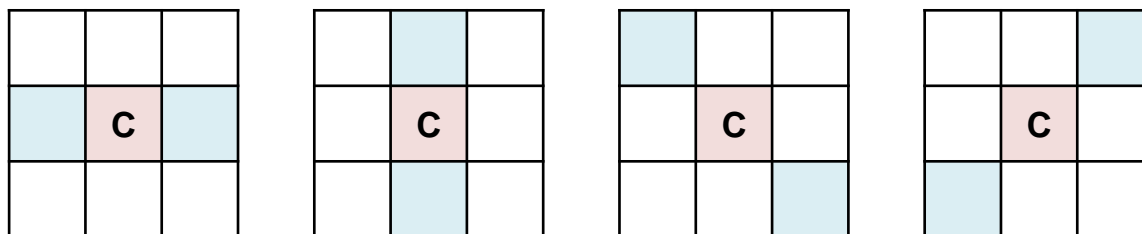
Deblocking Filter (DF)

- Luma vertical edges are first processed, and the results are DF intermediate pixels.
- Luma horizontal edges are then processed.
 - Use **reconstructed pixels** for filtering decisions (on/off & strong/weak)
 - Use **DF intermediate pixels** for filtering operations to generate **DF output pixels**
 - May change rows p_0 - p_2 & q_0 - q_2 for luma
- For chroma horizontal edges, only p_0 & q_0 may be changed, and only DF intermediate pixels are used in filtering decisions and filtering operations.



Sample Adaptive Offset (SAO)

- Applied only for luma in HM-3.0
- Each LCU belongs to one of the following SAO types
 - Central group band offset (BO), side group BO, 0-degree edge offset (EO), 45-degree EO, 90-degree EO, 135-degree EO, no processing
- BO uses the DF output pixel intensity for each pixel to perform pixel classification
- EO uses two neighboring DF output pixels for each pixel to perform pixel classification

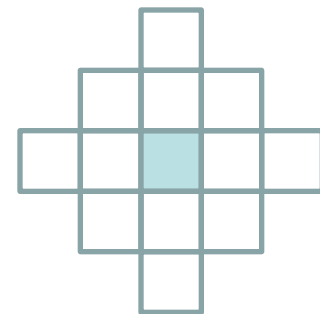
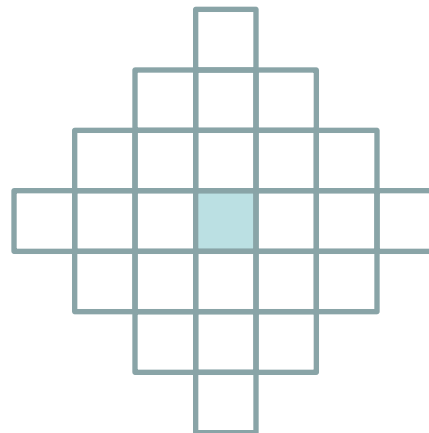
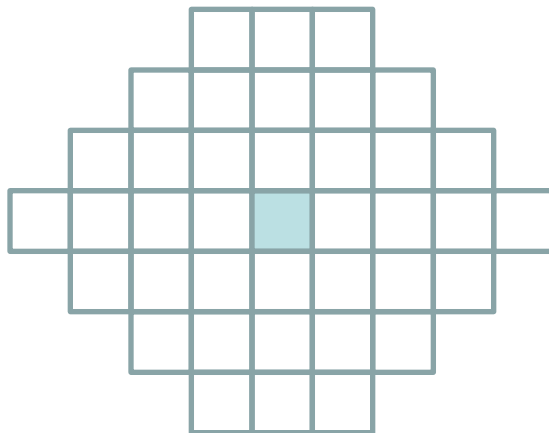


- After pixel classification, one offset is added for each DF output pixel to generate SAO output pixels

ALF – Filter Footprints

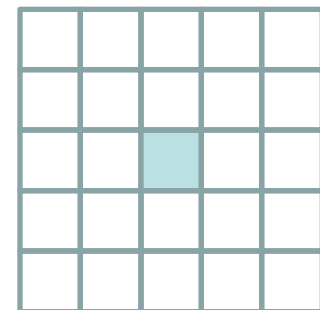
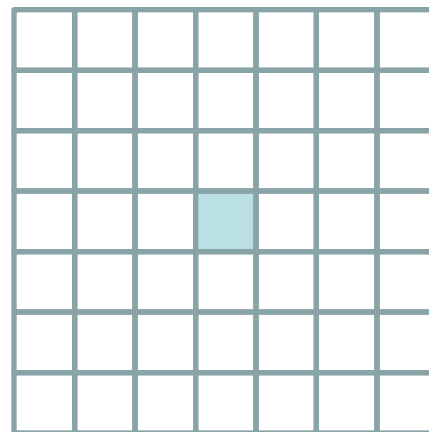
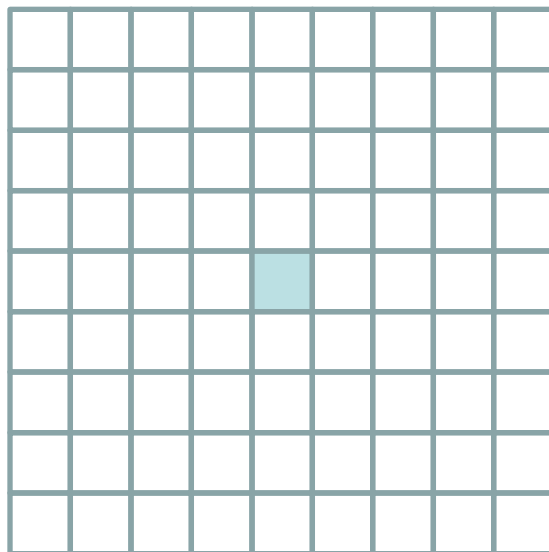
■ Luma

- 9x7D
- 7x7D
- 5x5D



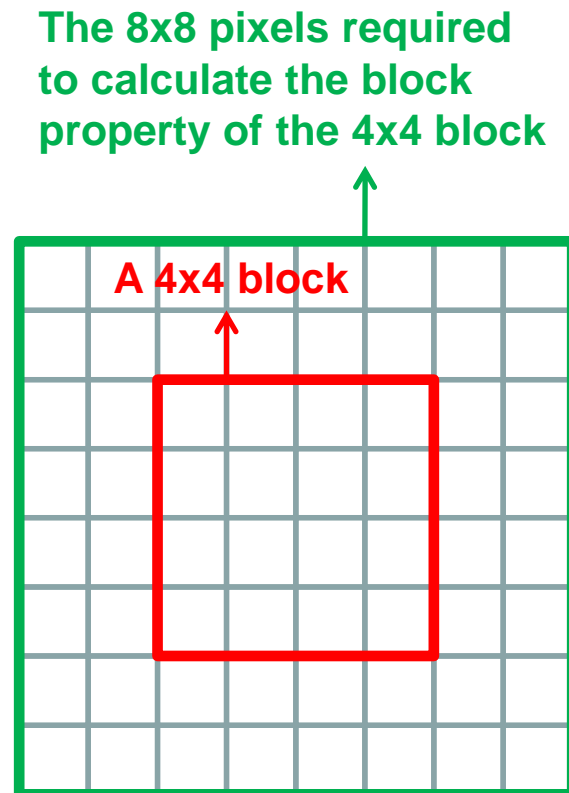
■ Chroma

- 9x9S
- 7x7S
- 5x5S



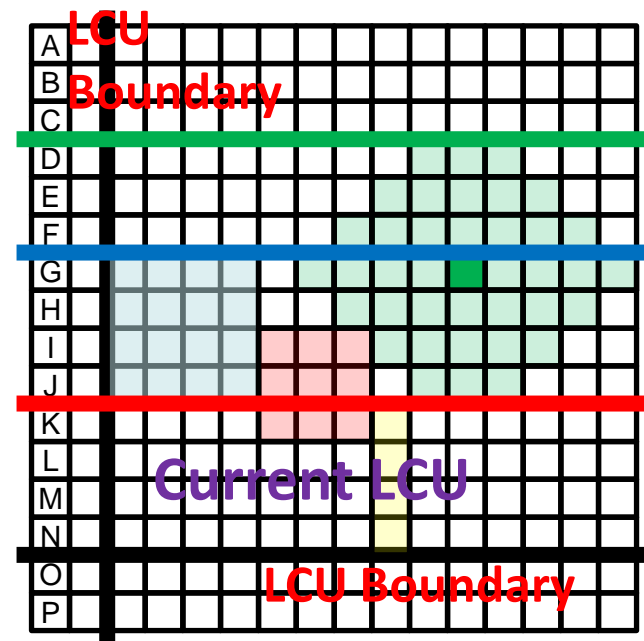
ALF – Filter Adaptation

- Luma: up to 16 filters
 - Region-based adaptation (RA)
 - Divide one picture into 16 regions
 - After region merging, apply one filter for each region
 - Block-based adaptation (BA)
 - Calculate a block property of every 4x4 block and classify all blocks into 15 groups
 - After group merging, apply one filter for each group
 - Can switch between RA and BA at the picture level
- Chroma: up to 1 filter shared by Cb and Cr



Luma Pixel Line Buffers in HM-3.0

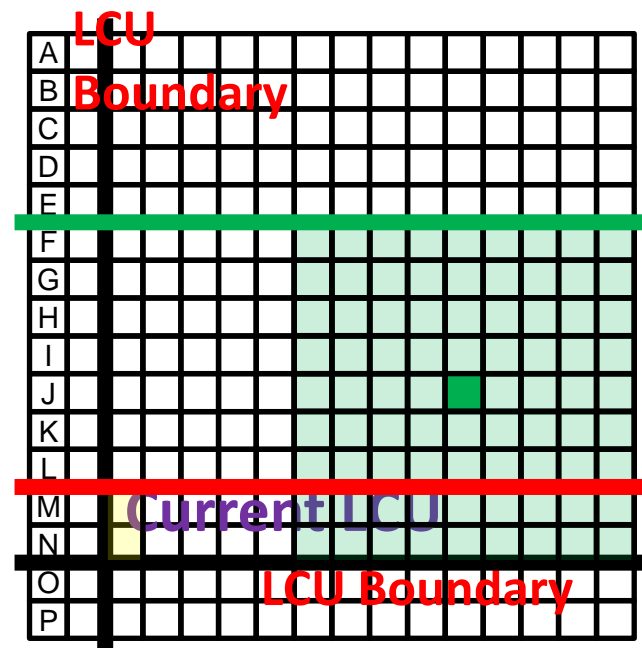
- Reconstructed pixels and DF intermediate pixels of lines K-N have to be stored for DF.
- Lines ...A-J can be processed by SAO because only lines L..N will be changed by vertical filter in DF
- Lines ...A-F can be processed by ALF, but lines G-J cannot because calculating block properties requires lines E-L
- When the lower LCU comes, lines K-P... can be processed by DF and SAO, and lines G-P... can be processed by ALF; however, DF output pixels of line J need to be stored for SAO, and SAO output pixels of lines D-J need to be stored.



In total
8 lines for DF
1 line for SAO
7 lines for ALF

Chroma Pixel Line Buffers in HM-3.0

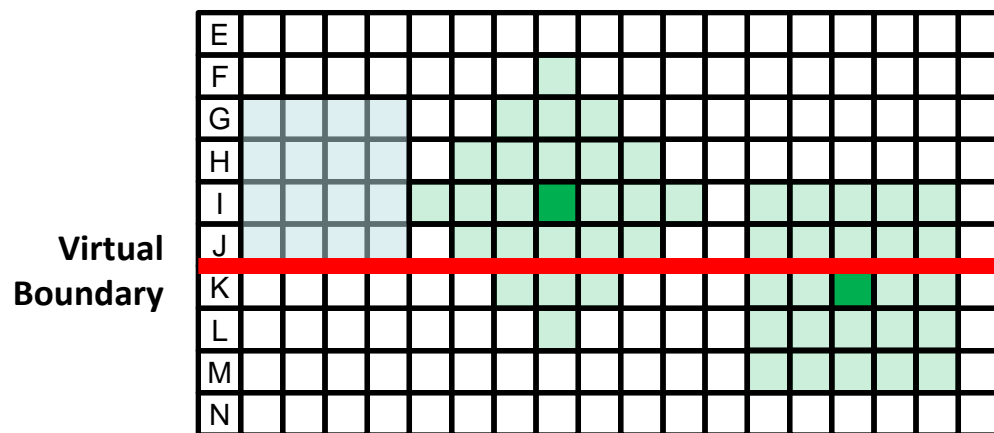
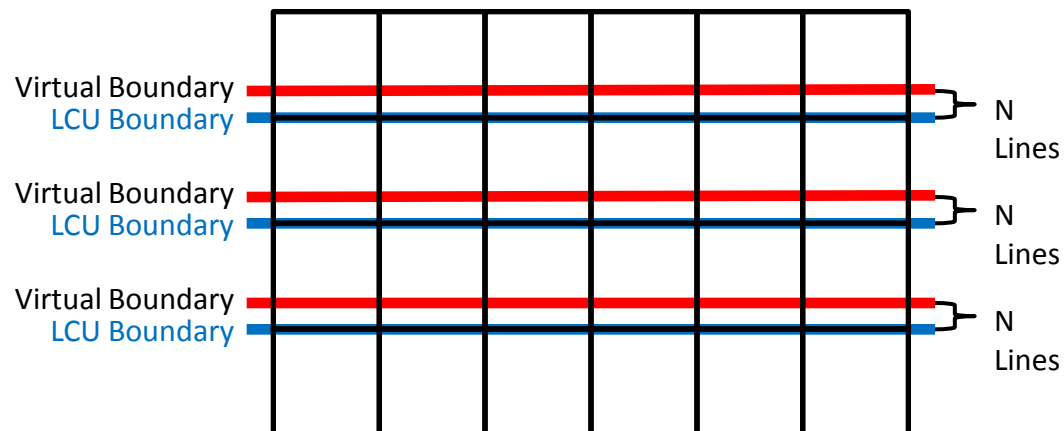
- DF intermediate pixels of lines M-N have to be stored for DF.
- Lines ...A-I can be processed by ALF because line M will not be further changed by DF
- When the lower LCU comes, lines M-P... can be processed by DF, and lines J-P... can be processed by ALF; however, **DF output pixels of lines F-L need to be stored.**



In total
2 lines for DF
0 line for SAO
7 lines for ALF

Proposed Virtual Boundaries for ALF

- Virtual boundaries (VBs) are upward shifted horizontal LCU boundaries by N pixels
 - N=4 for luma ([page 10](#))
 - N=2 for chroma ([page 11](#))
- For each to-be-filtered pixel, use repetitive padding to replace pixels on the other side of the virtual boundary
 - For both filtering and block property calculation
- No line buffer is required.**
 - Save 7 luma line buffers
 - Save 7 chroma line buffers



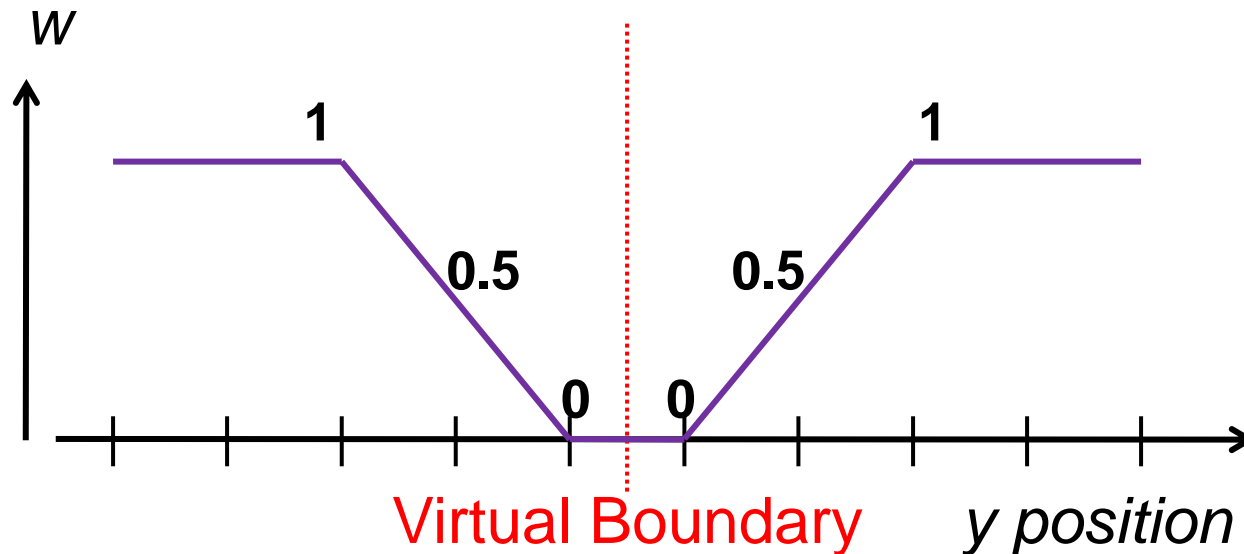
Proposed Unification of Filter Shape

- Since line buffers are no longer required, it is proposed to unify the filter shape
- Using 5x5 diamond, 7x7 diamond, and 9x9 diamond for luma and 7x7 diamond for chroma, as an example of unification

Proposed Boundary Smoothing

- Undesirable visual artifacts may exist near the virtual boundaries for some rare cases.
- A boundary smoothing technique is proposed to remove the possible visual artifacts as follows:

$$\text{FinalOutput} = \text{ALFOutput} * w + \text{SAOOutput} * (1-w)$$



Simulation Results

- Anchor: HM-3.0 under JCTVC-E700 common test conditions
- VB: virtual boundary; UFS: unified filter shape; BS: boundary smoothing

	HE-AI			HE-RA			HE-LD		
	Y	Cb	Cr	Y	Cb	Cr	Y	Cb	Cr
VB	0.1%	0.0%	0.1%	0.1%	0.0%	0.0%	0.2%	0.1%	-0.1%
	Enc: 100%			Enc: 99%			Enc: 100%		
	Dec: 100%			Dec: 100%			Dec: 100%		
VB + UFS	0.1%	-0.1%	-0.1%	-0.1%	-0.3%	-0.4%	-0.4%	-0.7%	-0.5%
	Enc: 101%			Enc: 99%			Enc: 99%		
	Dec: 100%			Dec: 100%			Dec: 100%		
VB + UFS + BS	0.1%	0.1%	0.1%	0.0%	-0.2%	-0.2%	-0.2%	-0.3%	-0.3%
	Enc: 100%			Enc: 100%			Enc: 100%		
	Dec: 99%			Dec: 100%			Dec: 100%		

Example of Boundary Smoothing

- Cactus, POC=311, QP=37, HE-LD

Enable boundary smoothing



Disable boundary smoothing

Subjective Quality Evaluation

- Blind tests
 - Not knowing which one is the anchor and which one is the proposed method
 - All tested videos with 4 QP values were evaluated by 5 people.
 - 4 tested videos were evaluated by more than 20 people including video experts and non-experts.
 - All bitstreams and decoders can be downloaded, and requests of the FTP site information can be sent to chingyeh.chen@mediatek.com
- Nobody could tell the difference between the anchor and the proposed method (repetitive padding at virtual boundaries with unified filter shape and boundary smoothing)

Crosscheck

- We thank Qualcomm (JCTVC-F367), Sharp (JCTVC-F389), and Toshiba (JCTVC-F665) for crosscheck.
- All objective and subjective results were confirmed.

Conclusions

- Proposed to apply repetitive padding at virtual boundaries to remove all ALF line buffers for both LCU-based decoding and picture-based decoding
- Used 5x5D, 7x7D, 9x9D for luma and 7x7D for chroma, as an example of filter shape unification
- Developed a boundary smoothing technique to remove possible visual artifacts
- Performance
 - Saved 7 luma line buffers and 7 chroma line buffers
 - No significant changes in BD-rates, in fact, achieved minor coding gain on average
 - No noticeable run time differences
 - Achieved the same subjective quality as anchor