Understanding in-loop filtering in the HEVC video standard

Mihir Mody - June 21, 2013

1.0 Introduction

High Efficiency Video Coding (HEVC) is a video compression standard, a successor to H.264/MPEG-4 AVC (Advanced Video Coding), jointly developed by the ISO/IEC Moving Picture Experts Group (MPEG) and ITU-T Video Coding Experts Group (VCEG) as ISO/IEC 23008-2 MPEG-H Part 2 and ITU-T H.265. HEVC promises half bit-rate compared to current de-facto standard H.264 at a similar video quality and is expected to be deployed in a wide variety of video applications ranging from cell phones, broadcast, set-top box, video conferencing, video surveillance, automotive, etc.

The figure below shows the block diagram of the HEVC Video decoder with loop filtering, as shown with loop filtering highlighted. As shown, it is a cascading of two stages - namely de-blocking filtering (DBLK) and Sample adaptive offset (SAO) filtering to remove blocking artifacts causing during video encoding. The next two sections describe these two stages in detail.

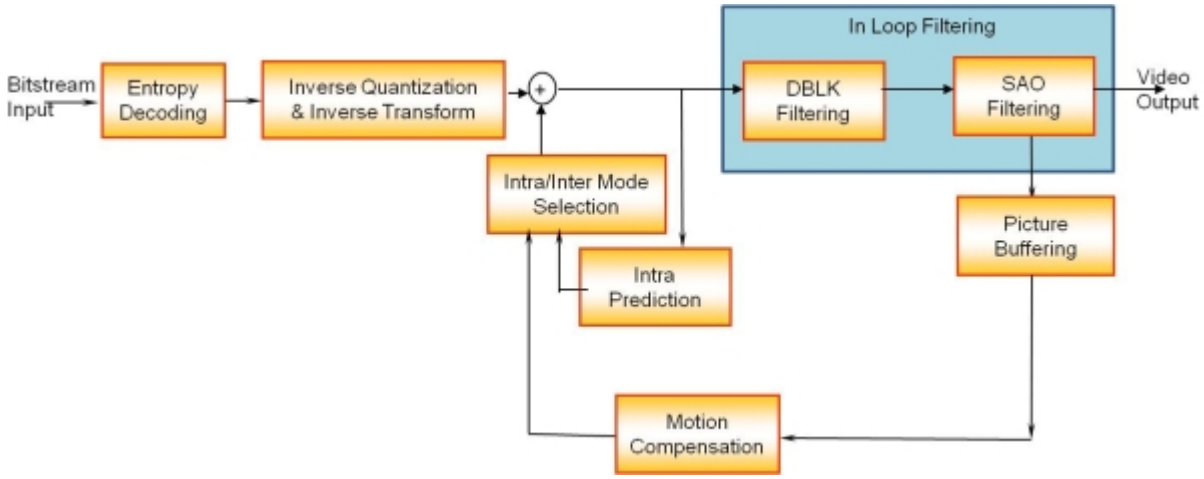

Figure 1: HEVC Video Decoder Block diagram

2.0 DB-Blocking Filtering (DBLK)

This section explains de-blocking filtering as described in the HEVC Video standard [1]. De-block filter operation can broadly be divided into two parts:

1. Boundary Strength (aka BS) Computation on filter edge
2. Actual Filter operation

Throughout the de-block filter operation the following convention has been used in the literature: Q pixels belong to the right side and P pixels belong to the left side of the “vertical” filter edge. Similarly, Q belongs to pixels below the horizontal edge and P above it for the “horizontal” filter edge.

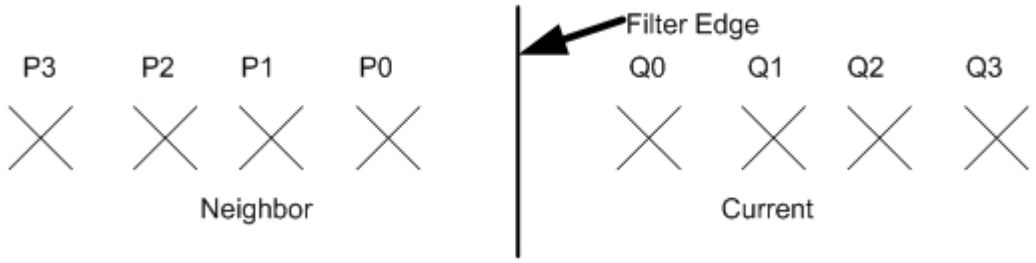

Figure 2: Neighbor pixels naming convention in in-loop de-blocking filtering

2.1 Filter Edge Strength (aka BS)

BS [hor/ver][xpos][ypos] is computed on an 8x8 grid and takes values 0, 1 & 2 as shown in the figure below:

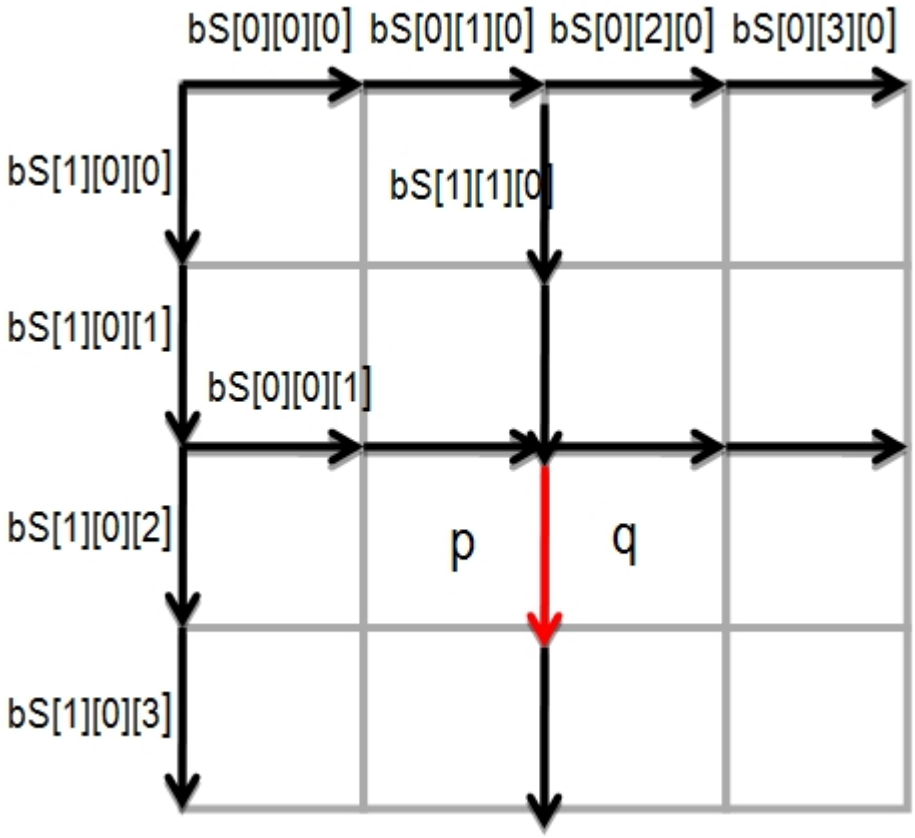

Figure 3: Boundary Strength (BS) diagram showing edges on 8x8 grid and its numbering

Although the filter is applied on every pixel edge on 8x8 grid (8x8 block contains 4 pixel edges), the property of boundary strength computation can be combined for all pixels for a 4-pixel segment. For

strength computation, we assume every 4-pixel segment as one edge. Please note that Boundary strength computation depends on Current, Left and Top LCU LCUinfo.

2.1.1 BS function Overview

BS is computed on an 8x8 grid, set to zero for picture and slice boundaries (if loop_filter_across_slice_enabled_flag = 0) and Tile boundaries (loop_filter_across_tile_enabled_flag = 0).

Only 8x8 pixel boundaries are filtered, which are Prediction Unit (PU) and/or Transform Unit (TU) boundaries, and get filtered as shown below.

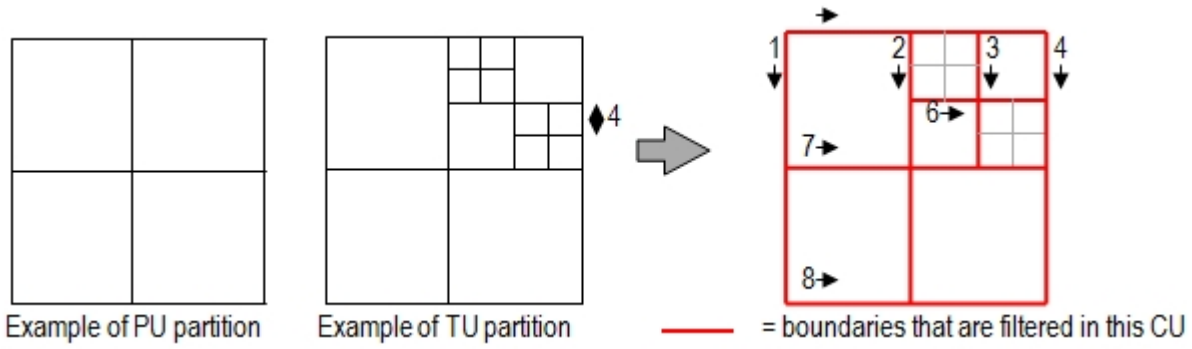

Figure 4: Boundary strength (BS) and edges alignment to PU and TU partitions

BS derivation rules:

BS is set to 0 if an edge is neither a TU edge nor a PU edge of a CU or not 8x8 edge or picture boundary or slice boundary with across slice filtering off or tile boundary with across tile boundary filtering off
else if block p or q is intra-coded, set BS to 2
else if block p or q is in a TU containing non-zero coefficients, set BS to 1
else if block p and q has different reference pictures or different number of MVs, BS=1
else if block p and q have a single MV, and vertical or horizontal MV difference ≥ 4, BS=1
else if block p and q have 2 MVs, and at least one MV has vertical/hor. MV difference ≥ 4, BS=1
else BS = 0

Readers interested in more details can refer to HEVC Specification [1] for the following:

- Derivation of TU Boundaries for BS : Section 8.7.2.1
- Derivation of PU Boundaries for BS : Section 8.7.2.2
- Derivation of BS: Section 8.7.2.3

Chroma BS derivation

2.1.2 Chroma BS derivation

Only PU and/or TU boundaries on 8x8 chroma pixel grid is filtered. Chroma BS values are derived from Luma BS values. For 4:2:0 (q0, p0) of chroma filter samples, derives BS from corresponding

(2q0, 2p0) Luma samples i.e downsampling of factor 2 as shown in below figure i.e BS{Hor/ver}[xpos/2][ypos/2].

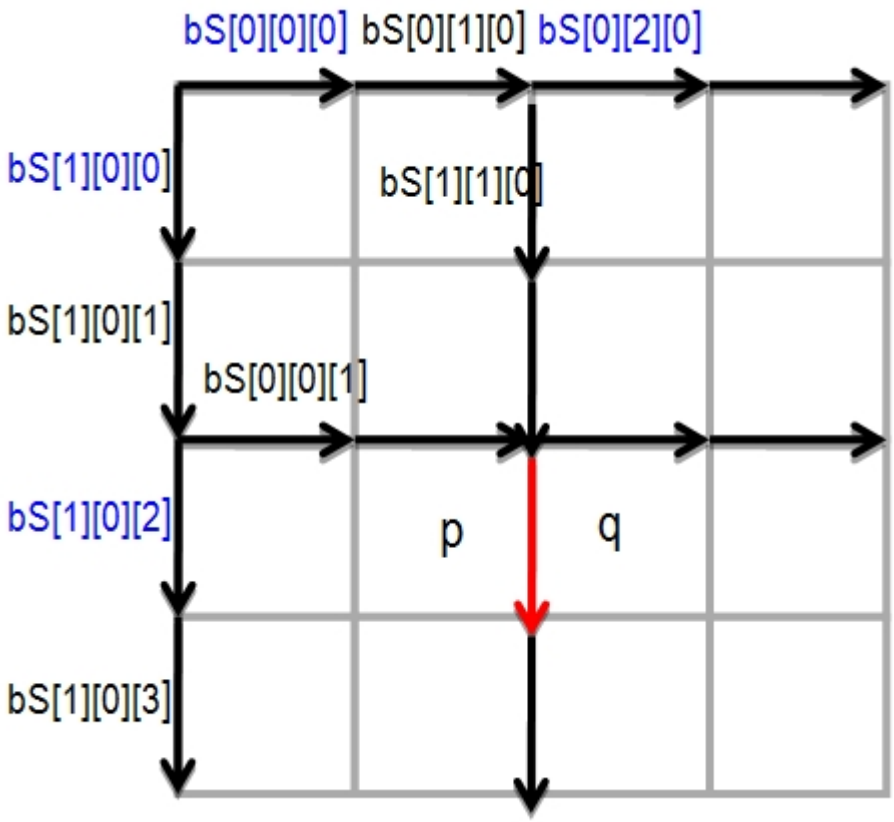

Figure 5: Chroma Boundary Strength mapping: Aligned to 8x8 and decimated by 2

2.2 Filter Operation

This section explains actual filtering to remove blocking artifacts as specified in the HEVC video standard.

2.2.1 Filter Order

With HEVC, the filter order is specified at the Frame level (not LCU level) as below:

- H filtering is performed across all block vertical edges in the **entire** processing frame
- V filtering is performed across all block horizontal edges in the **entire** processing frame

Filtering is fully independent from one 8x8 filtering segment to the next as shown in Figure 6:

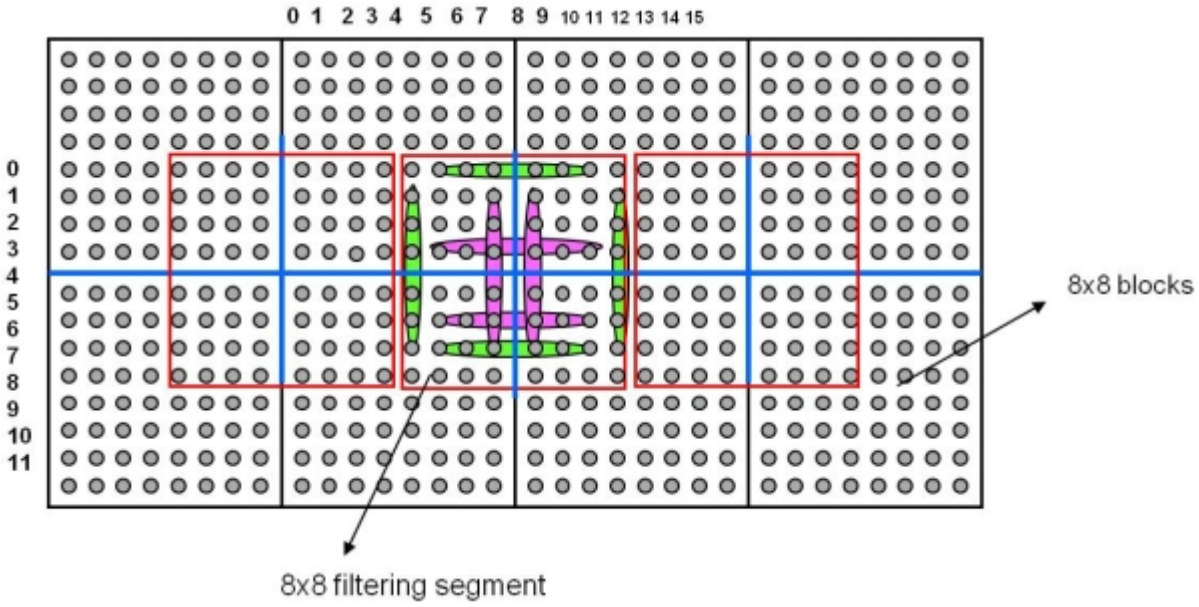

Figure 6: 8x8 block-level independent filtering

The filtering for deciding the need for Luma/Chroma (on/off decision), level of filtering (weak/strong), and finally actual filtering operations are described in the sections below.

2.2.2 Luma Filter ON/OFF and Weak/Strong decision

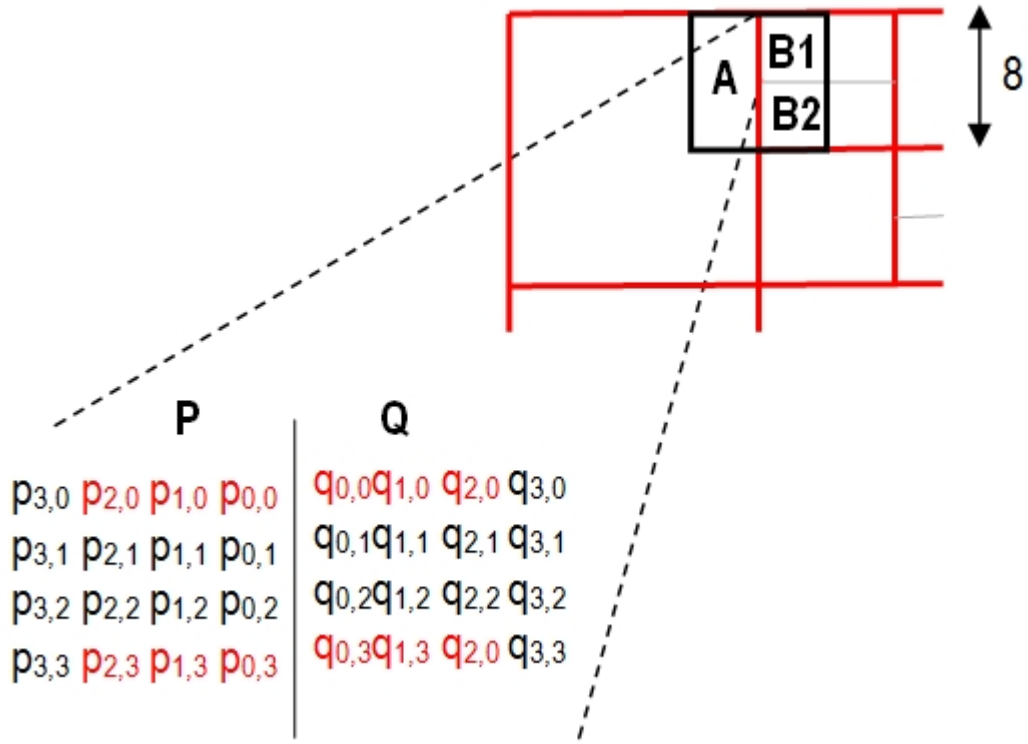

Figure 7: Pixels usage in filtering on/off and strength decision in de-blocking

- Filter on/off decision and Luma strong/weak decision are performed on four columns/lines segment as shown in the above diagram.
- If BS is not equal to zero, the following ordered steps apply
  - qPL= ((QPP+QPQ+1)>>1), QPPand QPQ are luma QPs
  - β = BETA_TABLE[Clip3(0,51,qPL+(beta_offset_div2<<1))]
  - tc = TC_TABLE[Clip3(0,53,qPL+2*(bS-1)+(tc_offset_div2<<1))]
  - dp0 = |p2,0-2*p1,0+p0,0| ; dp3 =|p2,3-2*p1,3+p0,3|
  - dq0 = |q2,0-2*q1,0+q0,0| ; dq3 = |q2,3-2*q1,3+q0,3|
  - dpq0 = dp0+dq0 ; dpq3 = dp3+dq3
  - dp = dp0+dp3 ; dq = dq0+dq3
  - dE = dEp = dEq = 0
  - If (dpq0 + dpq3 < β) // **filter on/off decision**
    - If ((2*dpq0 <(β>>2)) && (|p3,0- p0,0| + |q3,0- q0,0| <(β>>3)) && (| p0,0- q0,0| <((5*tc+1)>>1))) dSam0 = 1
    - If ((2*dpq3 <(β>>2)) && (|p3,3- p0,3| + |q3,3- q0,3| <(β>>3)) && (| p0,3- q0,3| <((5*tc+1)>>1))) dSam3 = 1
    - If (dSam0 ==1 && dSam3 ==1) dE = 2 (strong filter); else dE = 1 (weak filter);
    - If (dp < ((β+(β>>1))>>3)) dEp = 1 // number of filtered samples for weak filter
    - If (dq < ((β+(β>>1))>>3)) dEq = 1 // number of filtered samples for weak filter

Table 1: Relationship between qp, tc and β(BETA_TABLE and TC_TABLE)

Q	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
β'	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	7	8
t_C'	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Q	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
β'	9	10	11	12	13	14	15	16	17	18	20	22	24	26	28	30	32	34	36
t_C'	1	1	1	1	1	1	1	1	2	2	2	2	3	3	3	3	4	4	4
Q	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53			
β'	38	40	42	44	46	48	50	52	54	56	58	60	62	64	-	-			
t_C'	5	5	6	6	7	8	9	10	11	13	14	16	18	20	22	24			

Luma Strong & Weak Filtering

2.2.3 Luma Strong & Weak Filtering

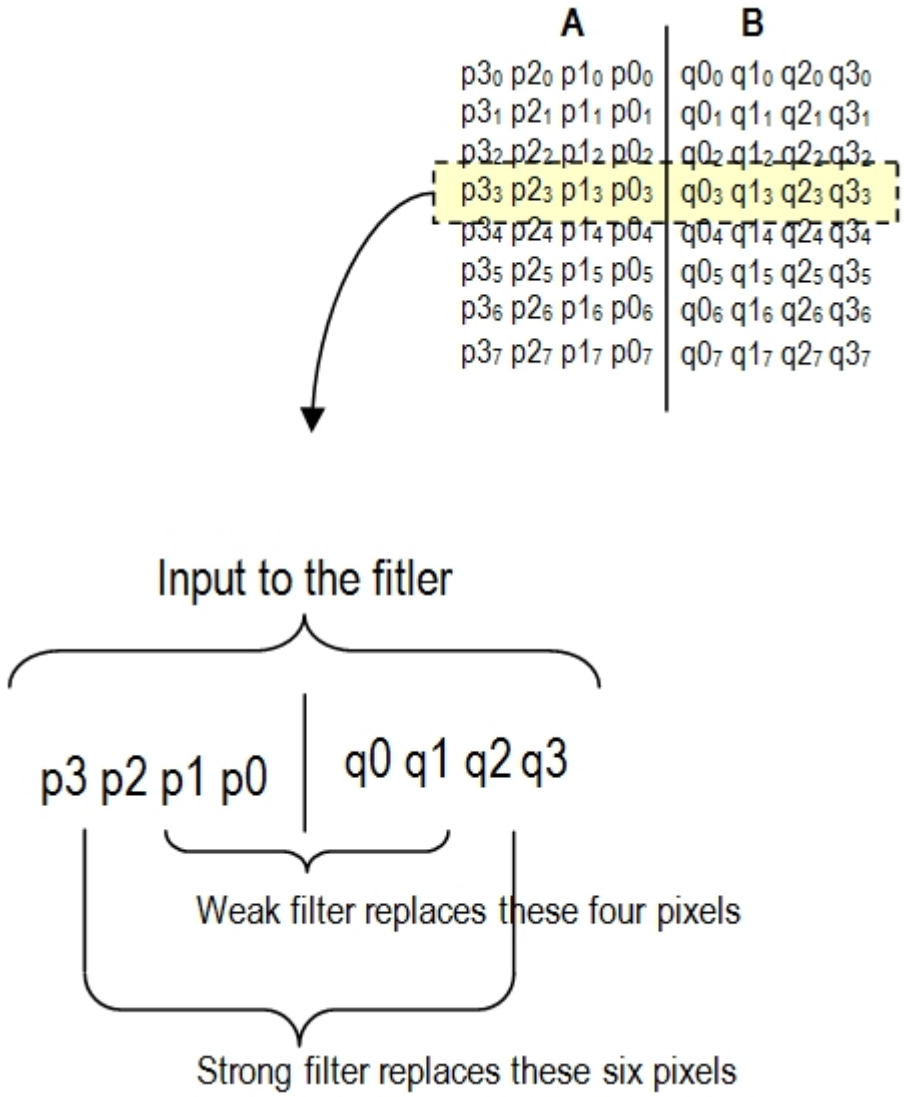

Figure 8: Luma DBLK Filtering and usage of pixels around edges

2.2.3.1 Luma Strong Filtering Mechanism

- 4-pixel segment shares the same decision (dE, dEp, dEq)
- If (dE ==2), strong filter applies to modify three pixels each side
  - p0' = Clip3(p0?2*tc, p0+2*tc, (p2 + 2*p1 + 2*p0 + 2*q0 + q1 + 4) >> 3)
  - p1' = Clip3(p1?2*tc, p1+2*tc, (p2 + p1 + p0 + q0 + 2) >> 2)
  - p2' = Clip3(p2?2*tc, p2+2*tc, (2*p3 + 3*p2 + p1 + p0 + q0 + 4) >> 3)
  - q0' = Clip3(q0?2*tc, q0+2*tc, (p1 + 2*p0 + 2*q0 + 2*q1 + q2 + 4) >> 3)
  - q1' = Clip3(q1?2*tc, q1+2*tc, (p0 + q0 + q1 + q2 + 2) >> 2)
  - q2' = Clip3(q2?2*tc, q2+2*tc, (p0 + q0 + q1 + 3*q2 + 2*q3 + 4) >> 3)

2.2.3.2 Luma Weak Filtering Mechanism

- 4-pixel segment shares a same decision (dE, dEp, dEq)
- If (dE ==1), weak filter applies to modify one or two pixels each side

- D = (9*(q0– p0)-3*(q1–p1)+8)>>4
- If (abs(Δ) < tc*10), the following ordered steps apply:
  - Filtered sample value p0’ and q0’ are specified as:
    Δ = Clip3(-tc,tc,Δ)
    p0’ = Clip1Y(p0+Δ)
    q0’ = Clip1Y(q0-Δ)
  - If dEp is equal to 1, the filtered sample value p1’ is specified as follows:
    Δp = Clip3(-(tc>>1),tc>>1,(((p2+p0+1)>>1)–p1+Δ)>>1)
    p1’ = Clip1Y(p1+Δp)
  - if dEq is equal to 1, the filtered sample value q1’ is specified as follows:
    Δq = Clip3(-(tc>>1),tc>>1,(((q2+q0+1)>>1)–q1–Δ)>>1)
    q1’ = Clip1Y(q1+Δq))

2.2.4 Chroma filtering

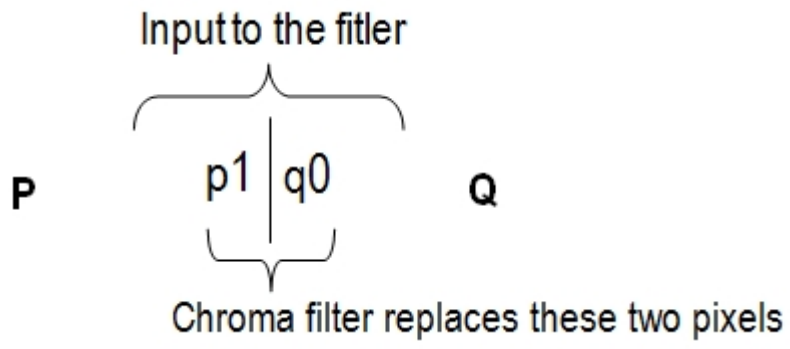

Figure 9: Chroma DBLK Filtering and usage of pixels around edges

- Only PU and/or TU boundaries on 8x8 chroma pixel grid are filtered
- Boundary strength bS is inherited from luma (factor 2 down-sampling each direction) (Refer to Chroma BS derivation sub-section)
- qPI = (((QPQ + QPP + 1) >> 1) + cqp_offset), where here cqp_offset represents pic_cb_qp_offset and pic_cr_qp_offset for components Cb/U and Cr/V respectively
- qPC is converted using qPI using Table as below

Table 2: QPc to qPi Conversion

qPi	<30	30	31	32	33	34	35	36	37	38	39	40	41	42	43	>43
QPc	= qPi	29	30	31	32	33	33	34	34	35	35	36	36	37	37	= qPi – 6

- tc = TC_TABLE[Clip3(0,53,qPC+2*(bS-1)+(tc_offset_div2<<1))]
- Chroma filter is carried out when bS > 1 Δ = Clip3(-tC,tC,((((q0–p0)<<2)+p1–q1+4)>>3))
  p0’ = Clip1C(p0+Δ)
  q0’ = Clip1C(q0-Δ)

2.2.5 Handling of IPCM and TQBypass

In the case of the edge containing IPCM CU with pcm_loop_filter_disable_flag =0 OR TQByapss (i.e., lossless), pixels (lying inside IPCM/TQBypass) remain unfiltered, while the other half gets filtering. This means the edge is filtered only on one side. This can be done by following two methods:

1. Skipping filtering of one half of pixel
2. Filter on both sides and later replace with Recon pixel back to half containing IPCM/TQBypass CU.

2.3 DBLK syntax element

The following syntax elements affect the DBLK operations and are summarized in the table below with range and description.

Table 3: DBLK Syntax Elements

Granularity	Syntax Element & Range	Comment
SPS	pcm_loop_filter_disable_flag (0/1)	Disable DBLK Filtering in case of CU type = IPCM
PPS	transquant_bypass_enable_flag (0/1)	Disable DBLK in case of CU type = Lossless (TQbypass)
	tiles_enabled_flag (0/1)	Tiles are present in given frame
	loop_filter_across_tiles_enabled_flag (0/1)	Across Tiles filtering on/off
	loop_filter_across_slices_enabled_flag (0/1)	Across Slice on/off
	deblocking_filter_control_present_flag (0/1)	Presence of Beta & Tc in PPS /SH
	deblocking_filter_override_enabled_flag (0/1)	Presence of override in SH
	pps_disable_deblocking_filter_flag (0/1)	Disable DBLK for entire picture
	beta_offset_div2 [-6,6]	Used in Filtering (ON/OFF/Weak/Strong Filtering)
	tc_offset_div2 [-6, 6]	Used in Filtering (ON/OFF/Weak/Strong Filtering)
SH	slice_header_disable_deblocking_filter_flag (0/1)	Disable DBLK for all LCU in slice
	beta_offset_div2 [-6,6]	Used in Filtering (ON/OFF/Weak/Strong Filtering)
	tc_offset_div2 [-6,6]	Used in Filtering (ON/OFF/Weak/Strong Filtering)
	slice_loop_filter_across_slices_enabled_flag (0/1)	Across Slice on/off (Controls Left & top boundary)
LCU level	----	

SAO-Decoder Filtering

3.0 SAO-Decoder Filtering

This section explains SAO filtering as described in the HEVC Video standard [1]. It is the process of adding offset to De-blocked pixel values according to SAO type - i.e., based on edge direction/shape (Edge Offset aka EO) and pixel level (Band Offset aka BO) or unchanged (OFF).

The offset range has the following characteristics:

- Offset magnitude range is [0, 7] for 8 bpp and [0, 31] for 10 bpp
- In case of BO, sign is signaled for offset with nonzero magnitude
- In case of EO, sign is inferred from edge category ((+) for 1 and 2, (-) for 3 and 4)

3.1 Edge offset (EO)

There are four types of Edge offset according to edge directions (0, 90, 135, 45 degrees) as shown on the diagram below, respectively.

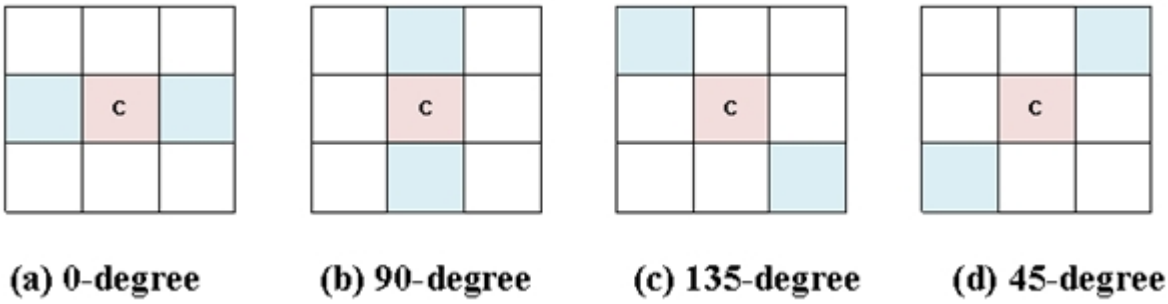

Figure 10: EO types namely 0,90, 135 and 45 degree

There are four offsets corresponding to four edge shapes (i.e., categories/edge index) for the selected direction. The edge index is referred to as SAO-subtype for EO. The figure below shows the different types of categories.

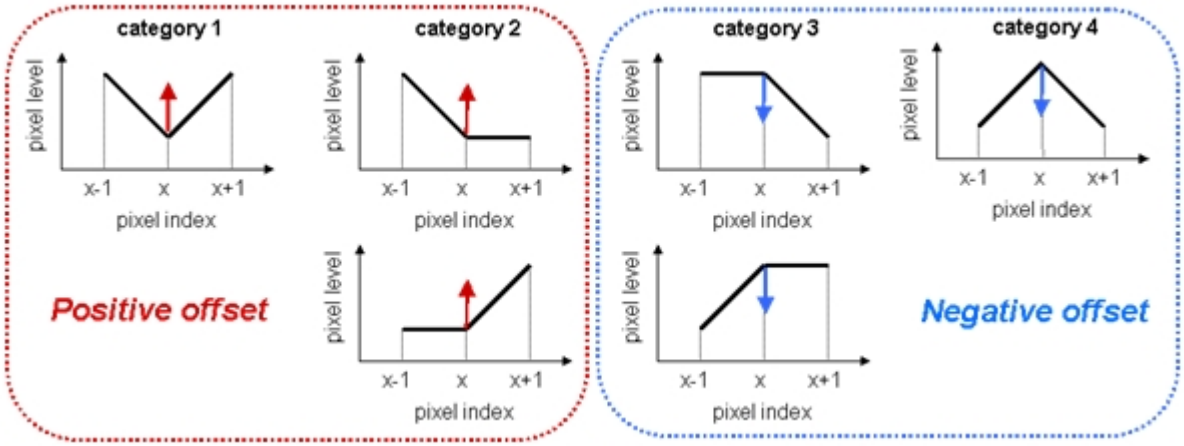

Figure 11: EO categories/edges index and pixel mapping

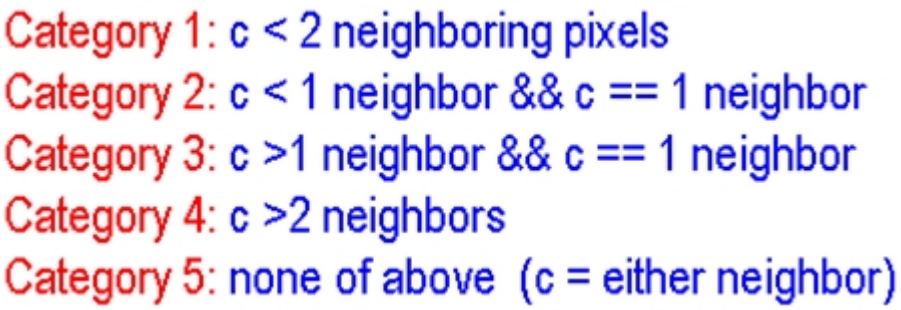

Figure 12: EO categories selection logic or formula

Offset value is added to each De-blocked pixel value if it belongs in a given EO type and category, otherwise it is not altered.

3.1.1 Band offset (BO)

The whole pixel range (0-255 for 8 bpp) is equally divided into 32 bands as shown in the figure below.

Figure 13: Bands classification (32 bands for pixel range)

There are four offsets provided for four consecutive bands from the start band. The Start band number is signaled along with four offset values. The start band position is also referred to as the SAO sub-type for BO.

Offset value is added to each De-blocked pixel value if it belongs to the range covered by one of the four bands, otherwise pixel value is not altered.

3.1.2 SAO syntax elements

The syntax elements that affect the SAO-Decoder operations are summarized in the table below along with with range and description.

Table 4: SAO Syntax Elements

Granularity	Syntax Element and Range	Comment
SPS	pcm_loop_filter_disable_flag (0/1)	Disable SAO-Decoder in case of CU type = IPCM
(Sequence)	sample_adaptive_offset_enabled_flag (0/1)	Enable SAO-Decoder (Bypass entire SAO-D block in case it is disabled)
PPS (Frame)	transquant_bypass_enable_flag (0/1)	Disable SAO-D in case of CU type = Lossless (TQbypass)
	tiles_enabled_flag (0/1)	Tiles are present in given frame
	loop_filter_across_tiles_enabled_flag (0/1)	Across Tiles filtering on/off
	loop_filter_across_slices_enabled_flag (0/1)	Across Slice on/off (Controls Left & top boundary)
Slice Header	slice_loop_filter_across_slices_enabled_flag (0/1)	Across Slice on/off (Controls Left & top boundary)
	slice_sao_luma_flag (0/1)	Slice level on/off
	slice_sao_chroma_flag (0/1)	Slice level on/off
LCU level	sao_merge_left_flag (0/1)	Common for all i.e. Y, Cb & Cr
	sao_merge_up_flag (0/1)	
	sao_type_idx_luma [0,2]	OFF, EO or BO
	sao_type_idx_chroma [0,2]	Common for Cb & Cr, Independent of Luma
	sao_offset_abs [0,7]	Common for EO & BO Separater for Y, CB & Cr
	sao_offset_sign (0/1)	Only in case of BO
	sao_band_position [0,28]	Only in case of BO
	sao_eo_class_luma [0,3]	Only in case of EO
	sao_eo_class_chroma [0,3]	EO Only, Common for Cb & Cr

3.1.3 Boundary condition processing

There are three conditions that require conditional processing:

1. Picture boundary (left, right, top & bottom)
2. Slice boundary and **slice_loop_filter_across_slices_enabled_flag =0**. The applicability of across slice filtering flag is for left and top edges for given slice boundary (and not all directions)
3. Tile boundary and **loop_filter_across_tiles_enabled_flag =0**

In the above scenario, pixels along slice boundary are not processed depending on SAO type.

- For BO, all pixels are processed
- For EO, pixel availability is checked according to SAO type and if pixel is not available in boundary condition it is skipped for processing (i.e., offset of 0)

The below figure depicts boundary condition processing.

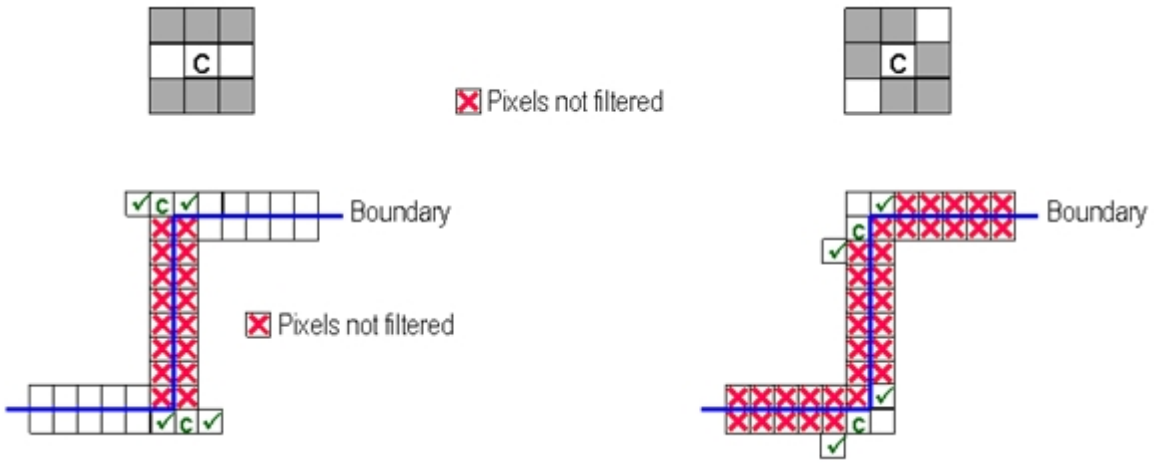

Figure 14: SAO-Decoder's Boundary Processing example

3.1.4 Conditional processing

In the following condition the SAO filtering is switched off.

- SAO Type idx = OFF
- CU type = PCM and Loop filter is disabled for PCM type
- CU type = TQBypss (lossless)
- Slice level SAO = OFF

References

[1] HEVC Standard FDIS (HM10.0): http://phenix.int-evry.fr/jct/doc_end_user/documents/12_Geneva/wg11/JCTVC-L1003-v34.zip
[2] JCT-VC document management system: http://phenix.int-evry.fr/jct/
[3] HEVC reference software: http://hevc.info/

The author would like to thank Minhua Zhou, Osamoto Akira and Woo-Shik Kim for suggestion and input provided for this article.

More about Mihir Mody