JVET-AB0133: AHG12: Inter-RPL and 1-byte NAL unit headers

Rickard Sjöberg
Martin Pettersson
Jacob Ström
(Ericsson)

Summary

- Two modifications are proposed
  - Add prediction to the signaling of the RPLs in the SPS.
  - Make the NAL unit header in ECM 1 byte rather than 2 bytes for the 16 most common NAL unit types
- The slice data and therefore also all decoded sample values are unaffected by the proposal

	Over ECM-6.0		
	AI-Y	RA-Y	LDB-Y
Class A1	0.00%	-0.05%	
Class A2	0.00%	-0.04%	
Class B	-0.01%	-0.09%	-0.04%
Class C	-0.01%	-0.22%	-0.08%
Class E	-0.02%		-0.35%
Overall	-0.01%	-0.11%	-0.13%
Class D	-0.04%	-0.63%	-0.25%
Class F	-0.02%	-0.37%	-0.19%
Class TGM	0.00%	-0.12%	-0.07%

RPL background

The ECM-6.0 Decoded Picture Buffer (DPB)

- The DPB consists of decoded pictures and the current picture
- A current picture can only reference pictures that are present in the DPB

Reference picture lists in ECM-6.0

- One list of L0 lists and one list of L1 lists are decoded from the SPS
- In the slice header, there is a "ref_pic_list_idx" syntax element that selects L0 and L1 lists for the current picture

Proposal

- An alternative to the current ECM-6.0 POC based signaling is proposed
- By sending the SPS lists in decoding order and keeping track of a virtual DPB, the signaling can be done by indices to the virtual DPB
- This is similar to the Inter-RPS method in HEVC

list of L0 lists

0	-32	-64	-48	-40	-36
1	-16	-32	-48	-24	-20
2	-8	-24	-16	-40	-12

list of L1 lists

0	-32	-48
1	16	
2	8	24

Proposed SPS syntax

Conceptual proposed syntax

seq_parameter_set_rbsp( ) {	
...	
sps_inter_rpl_num_ref_pic_lists	ue(v)
for (rplIdx = 1 ; rplIdx < sps_inter_rpl_num_ref_pic_lists ; rplIdx ++) {	
for(lx = 0; lx < 2 ; lx++) {	
sps_num_ref_entries[lx]	se(v)
for (j=0 ; j< num_ref_entries[lx] ; j++)	
sps_inter_rpl_idx[lx][rplIdx][j]	u(v)
}	
}	
...	

Let RplList be the list of RPLs in the SPS and let RPL be an entry in RplList containing an L0 and L1 list.

sps_inter_rpl_num_ref_pic_lists is the number of RPLs in RplList.

sps_num_ref_entries[lx] is the number of entries for LX (L0 or L1).

sps_inter_rpl_idx[lx][rplIdx][j] is the index to the list of picture references identifying the value for LX[j] in the rplIdx'th entry of RplList.

list of L0 lists

0	**-32**	**-64**	**-48**	**-40**	**-36**
1	**-16**	**-32**	**-48**	**-24**	**-20**
2	**-8**	**-24**	**-16**	**-40**	**-12**

list of L1 lists

0	**-32**	**-48**
1	**16**	
2	**8**	**24**

RPL prediction details

In addition, the proposal includes the following to reduce the bit-cost further:

- One L0 flag and one L1 flag for each RPL that specifies whether automatically generated L0 and L1 lists will be used.
- A delta QP for each RPL that is later added to slice_qp_delta
- Signaling the log2 of the GOP size that may be used to derive POC delta values between RPL entries which are needed to maintain the virtual DPB.
- Delta coding of sps_num_ref_entries[lx]

NAL unit header

ECM-6.0 NAL unit header

1st byte

nal_unit_header( ) {	**Descriptor**
forbidden_zero_bit	f(1)
nuh_reserved_zero_bit	u(1)
nuh_layer_id	u(6)

2nd byte

...	
nal_unit_type	u(5)
nuh_temporal_id_plus1	u(3)
}	

Proposed NAL unit header

1st byte

nal_unit_header( ) {	**Descriptor**
zero_tid_required_flag	u(1)
nuh_temporal_id_plus1	u(3)
vcl_flag	u(1)
nal_unit_type_lsb	u(2)
nuh_extension_flag	u(1)

2nd byte

...	**Descriptor**
if (nuh_extension_flag) {	
nuh_layer_id	u(6)
nal_unit_type_bit	u(1)
nuh_reserved_zero_bit	u(1)
}	
}	

When **zero_tid_required_flag** is equal to 1, nuh_temporal_id_plus1 shall be equal to 1. (i.e., TemporalId shall be equal to 0). MPEG-2 PES startcode emulation is prevented.

If not present, **nal_unit_type_bit** and **nuh_layer_id** are both inferred to be equal to 0.

NalUnitType = (zero_tid_required_flag << 4) + (vcl_flag << 3) + (nal_unit_type_bit << 2) + nal_unit_type_lsb

1-byte NAL unit types

non-VCL:

- NAL_UNIT_PPS
- NAL_UNIT_PH
- NAL_UNIT_PREFIX_APS
- NAL_UNIT_SUFFIX_APS
- NAL_UNIT_DCI
- NAL_UNIT_SPS
- NAL_UNIT_EOS
- NAL_UNIT_EOB

VCL:

- NAL_UNIT_CODED_SLICE_TRAIL
- NAL_UNIT_CODED_SLICE_STSA
- NAL_UNIT_CODED_SLICE_RADL
- NAL_UNIT_CODED_SLICE_RASL
- NAL_UNIT_CODED_SLICE_IDR_W_RADL
- NAL_UNIT_CODED_SLICE_IDR_N_LP
- NAL_UNIT_CODED_SLICE_CRA
- NAL_UNIT_CODED_SLICE_GDR

The NAL unit header of these NAL unit types is 1 byte long when layer_id is equal to 0.

For layer_id not equal to 0, all NAL unit headers are 2 bytes.

Results

Full proposal

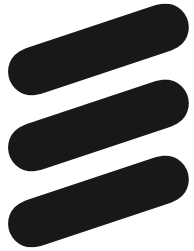
	Over ECM-6.0		
	AI-Y	RA-Y	LDB-Y
Class A1	0.00%	-0.05%	
Class A2	0.00%	-0.04%	
Class B	-0.01%	-0.09%	-0.04%
Class C	-0.01%	-0.22%	-0.08%
Class E	-0.02%		-0.35%
Overall	-0.01%	-0.11%	-0.13%
Class D	-0.04%	-0.63%	-0.25%
Class F	-0.02%	-0.37%	-0.19%
Class TGM	0.00%	-0.12%	-0.07%

1-byte NAL unit header only

	Over ECM-6.0		
	AI-Y	RA-Y	LDB-Y
Class A1	0.00%	-0.01%	
Class A2	0.00%	-0.01%	
Class B	-0.01%	-0.02%	-0.02%
Class C	-0.01%	-0.05%	-0.04%
Class E	-0.02%		-0.20%
Overall	-0.01%	-0.03%	-0.07%
Class D	-0.04%	-0.15%	-0.14%
Class F	-0.02%	-0.07%	-0.11%
Class TGM	0.00%	-0.01%	

Encoding and decoding times are estimated to be very similar to the anchor. Decoding seems to be a little faster for the proposal.

Only NAL unit headers and SPS data is modified by the proposal.



Results

Full proposal

	All Intra Main10 Over ECM-6.0				
	Y	U	V	EncT	DecT
Class A1	0.00%	0.00%	0.00%	94%	100%
Class A2	0.00%	0.00%	0.00%	99%	101%
Class B	-0.01%	-0.01%	-0.01%	98%	103%
Class C	-0.01%	-0.01%	-0.01%	86%	83%
Class E	-0.02%	-0.02%	-0.02%	86%	97%
Overall	-0.01%	-0.01%	-0.01%	93%	96%
Class D	-0.04%	-0.03%	-0.04%	100%	100%
Class F	-0.02%	-0.01%	-0.01%	91%	96%
Class TGM	0.00%	0.00%	0.00%	105%	104%

	Random Access Main 10 Over ECM-6.0				
	Y	U	V	EncT	DecT
Class A1	-0.05%	-0.04%	-0.04%	99%	88%
Class A2	-0.04%	-0.04%	-0.04%	96%	90%
Class B	-0.09%	-0.09%	-0.09%	96%	105%
Class C	-0.22%	-0.21%	-0.21%	84%	86%
Class E					
Overall	-0.11%	-0.10%	-0.10%	93%	93%
Class D	-0.63%	-0.62%	-0.62%	101%	95%
Class F	-0.37%	-0.35%	-0.36%	93%	96%
Class TGM	-0.12%	-0.12%	-0.12%	100%	108%

	Low delay B Main10 Over ECM-6.0				
	Y	U	V	EncT	DecT
Class A1					
Class A2					
Class B	-0.04%	-0.04%	-0.04%	97%	96%
Class C	-0.08%	-0.08%	-0.08%	95%	104%
Class E	-0.35%	-0.33%	-0.33%	92%	98%
Overall	-0.13%	-0.13%	-0.12%	95%	99%
Class D	-0.25%	-0.23%	-0.24%	93%	92%
Class F	-0.19%	-0.19%	-0.19%	89%	88%
Class TGM	-0.07%	-0.07%	-0.07%	109%	107%

1-byte NAL unit header only

	All Intra Main10 Over ECM-6.0				
	Y	U	V	EncT	DecT
Class A1	0.00%	0.00%	0.00%	#NUM!	93%
Class A2	0.00%	0.00%	0.00%	#NUM!	103%
Class B	-0.01%	-0.01%	-0.01%	#NUM!	99%
Class C	-0.01%	-0.01%	-0.01%	#NUM!	80%
Class E	-0.02%	-0.02%	-0.02%	#NUM!	87%
Overall	-0.01%	-0.01%	-0.01%	#NUM!	92%
Class D	-0.04%	-0.03%	-0.04%	#NUM!	105%
Class F	-0.02%	-0.01%	-0.01%	#NUM!	88%

	Random Access Main 10 Over ECM-6.0				
	Y	U	V	EncT	DecT
Class A1	-0.01%	-0.01%	-0.01%	#NUM!	91%
Class A2	-0.01%	-0.01%	-0.01%	#NUM!	91%
Class B	-0.02%	-0.02%	-0.02%	#NUM!	96%
Class C	-0.05%	-0.05%	-0.05%	#NUM!	79%
Class E					
Overall	-0.03%	-0.03%	-0.02%	#NUM!	89%
Class D	-0.15%	-0.15%	-0.15%	#NUM!	99%
Class F	-0.07%	-0.07%	-0.07%	#NUM!	85%

	Low delay B Main10 Over ECM-6.0				
	Y	U	V	EncT	DecT
Class A1					
Class A2					
Class B	-0.02%	-0.02%	-0.02%	#NUM!	102%
Class C	-0.04%	-0.04%	-0.04%	#NUM!	100%
Class E	-0.20%	-0.19%	-0.18%	#NUM!	88%
Overall	-0.07%	-0.07%	-0.07%	#NUM!	98%
Class D	-0.14%	-0.13%	-0.13%	#NUM!	94%
Class F	-0.11%	-0.11%	-0.10%	#NUM!	84%

Decoding complexity experiment

To estimate the decoding complexity, a laptop experiment where the encoder was modified to repeat RPL information in the bitstream 10,000 times, and the decoder was modified to decode it:

```
EncoderApp.exe  -c cfg/encoder_randomaccess_ecm.cfg -c cfg/per-sequence/BQSquare.cfg -q 37 -i black.yuv -wdt 64 -hgt 64 --
IntraPeriod=64 -f 65 -b bitstream.bit -dph 0 --ALF=0 --CCALF=0 --SAO=0 --CCSAO=0 > enc.txt
```

```
DecoderApp.exe -b bitstream.bit > dec.txt
```

The Run-time values in the table are "Total Time:" values from the dec.txt files.

We can observe that the modified proposal was faster to decode than the anchor, possibly since the number of bits used for RPL information is much lower in the proposal.

	Run-time anchor	Run-time proposal
Run 1	0.454 sec.	0.350 sec.
Run 2	0.452 sec.	0.261 sec.
Run 3	0.417 sec.	0.335 sec.
Run 4	0.453 sec.	0.265 sec.
Average	0.444 sec.	0.303 sec.