



JCTVC-I0189

On APS referencing and updating

N.Ouedroago, E.Francois

JCT-VC 9th Meeting, Geneva, April 2012

Proposal overview

- Proposal 1a–1b: Referencing multiple APS
 - Use of multiple identifiers in Slice Header
- Proposal 2: APS update
 - Conditional replenishment + reset mechanism
- Proposal 3: APS priority
 - APS for non-reference pictures
- Proposal 4: mechanism for APS buffer reset
 - Resynchronization points
 - Buffer period

Proposal overview

- **Proposal 1a–1b: Referencing multiple APS**
 - Use of multiple identifiers in Slice Header
- **Proposal 2: APS update**
 - Conditional replenishment + reset mechanism
- **Proposal 3: APS priority**
 - APS for non-reference pictures
- **Proposal 4: mechanism for APS buffer reset**
 - Resynchronization points
 - Buffer period

Proposal 1a: Referencing multiple APS

■ Classes of parameters

- QSM, DBF, SAO, ALF, ...

■ Direct referencing in SH to 1 APS for each class of params

- When $id=0$
→ params present in the Slice
- When $id > 0$
→ use params from APS with
 $aps_id = (id-1)$

slice_header() {	Descr
[...]	
if(scaling_list_enable_flag)	
scaling_list_param_id	ue(v)
if(deblocking_filter_in_aps_enabled_flag)	
deblocking_filter_param_id	ue(v)
if(sample_adaptive_offset_enabled_flag)	
sao_param_id	ue(v)
if(adaptive_loop_filter_enabled_flag)	
alf_param_id	ue(v)
[...]	
}	

■ Comments

- Simple design, which simplifies current signaling
- Enables for each tool to refer to different APSs

Proposal 1 b: Referencing multiple APS

- Refer in SH to multiple APS for sets of parameters classes
 - Several valid APS (**num_param_id**)
 - **param_id[i]**: APS identifier
 - When $id=0$ → params present in the Slice
 - When $id > 0$ → use params from APS with $aps_id = (param_id[i]-1)$
 - **identifier_set[i]**: specifies the set of used parameters classes from APS ($param_id[i]-1$)

slice_header() {	Descr
[...]	
num_param_id	u(v)
for (i = 0; i < num_param_id; i++) {	
param_id[i]	ue(v)
identifier_set[i]	u(v)
}	
[...]	
}	

identifier_set[i]	Set of APS parameters classes
0	ALF & SAO
1	ALF
2	SAO
3	DBF
4	QSM
5	SAO & QSM
6	SAO & ALF & QSM
7	ALL

Proposal 1 b: Referencing multiple APS

related APS Params Identifier	identifier_set[i]
ALF & SAO	0
ALF	1
SAO	2
DBF	3
QSM	4
SAO & QSM	5
SAO & ALF & QSM	6
ALL	7

APS – id=0	
Qmatrix_present	1
SAO_present	0
ALF_present	0
Qmatrix[0]	
No SAO	
No ALF	

APS – id=1	
Qmatrix_present	0
SAO_present	1
ALF_present	1
No Qmatrix	
SAO[0]	
ALF[0]	

Slice header	
num_valid_APS=2	
param_id[0]=2	
Identifier_set[0]=0	
param_id[1]=1	
Identifier_set[1]=4	
Qmatrix[0]	
SAO[1]	
ALF[1]	

■ Comments

- Same type of signaling as I0338, but inserted in SH instead of usage of a new type of NALU (GPS)
- Generalize signaling of I0338
- More future-proof than proposal 1 a

Proposal overview

- Proposal 1a–1b: Referencing multiple APS
 - Use of multiple identifiers in Slice Header
- Proposal 2: APS update
 - Conditional replenishment + reset mechanism
- Proposal 3: APS priority
 - APS for non-reference pictures
- Proposal 4: mechanism for APS buffer reset
 - Resynchronization points
 - Buffer period

Proposal 2: APS update

■ APS

- Container of parameters of different classes (QSM, DBF, SAO, ALF)
- For each class, presence flag indicates if params are present or not

● APS buffer

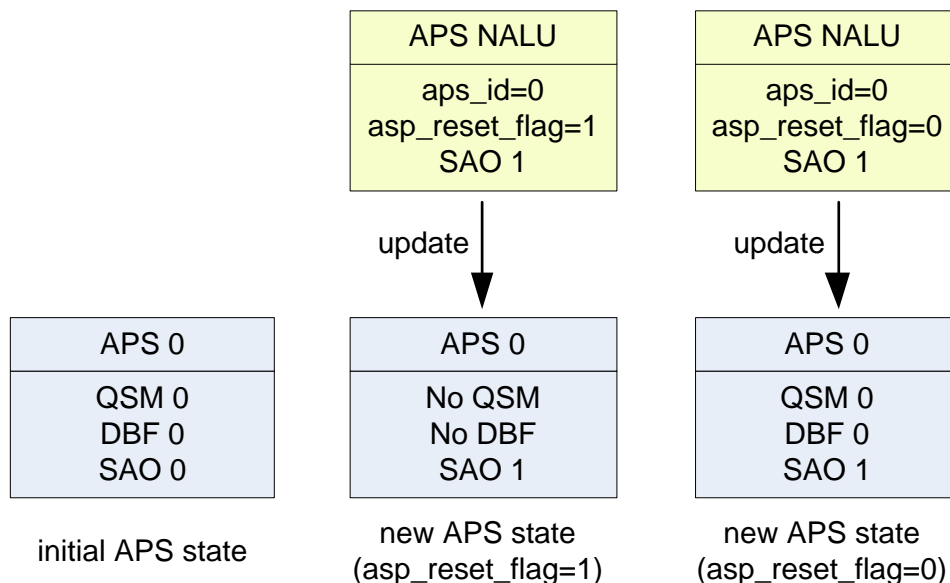
- APSs in buffer created and updated from APS NALUs

	Descriptor
aps_rbsp() {	
aps_id	ue(v)
aps_reset_flag	u(1)
aps_scaling_list_data_present_flag	u(1)
if(aps_scaling_list_data_present_flag)	
scaling_list_param()	
aps_deblocking_filter_data_present_flag	u(1)
if(aps_deblocking_filter_present_flag) {	
disable_deblocking_filter_flag	u(1)
if(!disable_deblocking_filter_flag) {	
beta_offset_div2	se(v)
tc_offset_div2	se(v)
}	
}	
aps_sample_adaptive_offset_data_present_flag	u(1)
if(aps_sample_adaptive_offset_data_present_flag) {	
aps_sample_adaptive_offset_flag	u(1)
if(aps_sample_adaptive_offset_flag)	
aps_sao_param()	
}	
aps_adaptive_loop_filter_data_present_flag	u(1)
if(aps_adaptive_loop_filter_data_present_flag) {	
aps_adaptive_loop_filter_flag	u(1)
if(aps_adaptive_loop_filter_flag)	
alf_param()	
}	
aps_extension_flag	u(1)
if(aps_extension_flag)	
while(more_rbsp_data())	
aps_extension_data_flag	u(1)
rbsp_trailing_bits()	
}	

Proposal 2: APS update

- Allow conditional replenishment for APS whose identifier=aps_id
- Condition replenishment controlled by **aps_reset_flag**
 - Equal to 1, reset the APS before update with parameters of the APS NALU
 - Equal to 0, all parameters of APS are conserved before update

aps_rbsp() {	Descr
aps_id	ue(v)
aps_reset_flag	u(1)
[...]	u(1)
}	



■ Comments

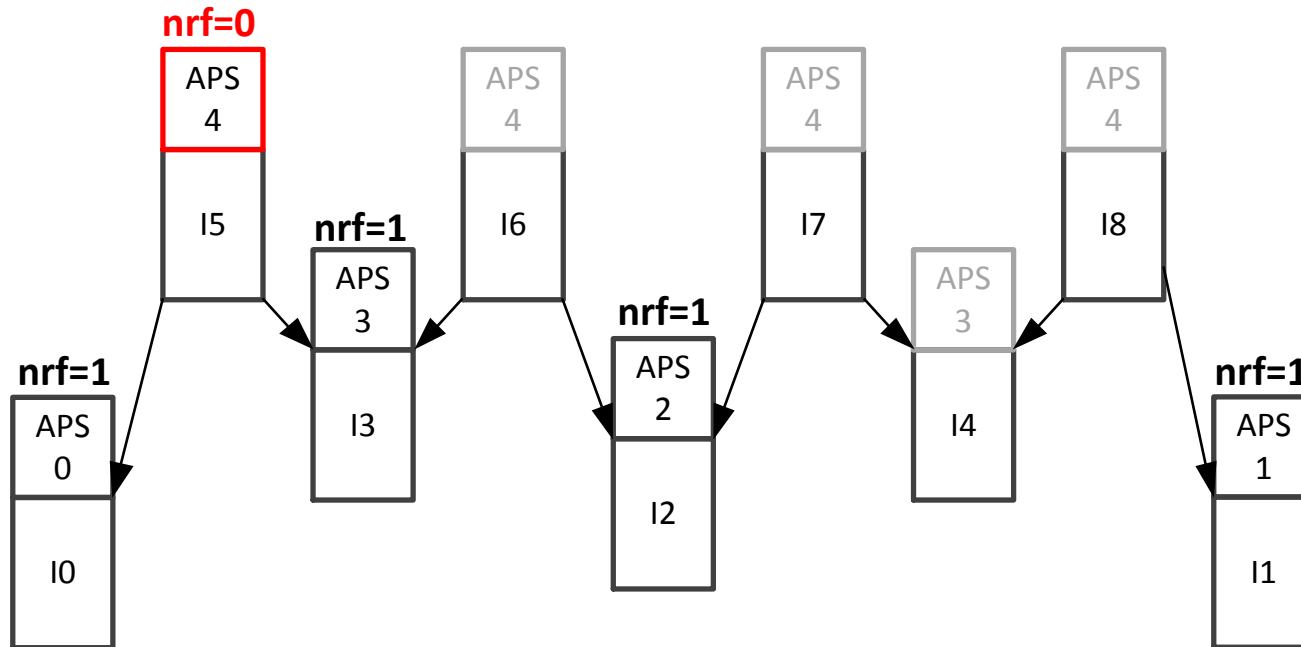
- Leads to equivalent behaviour as 'aps_intra_flag' introduced in I0081

Proposal overview

- Proposal 1a–1b: Referencing multiple APS
 - Use of multiple identifiers in Slice Header
- Proposal 2: APS update
 - Conditional replenishment + reset mechanism
- **Proposal 3: APS priority**
 - APS for non-reference pictures
- **Proposal 4: mechanism for APS buffer reset**
 - Resynchronization points
 - Buffer period

Proposal 3: APS for non-ref pictures

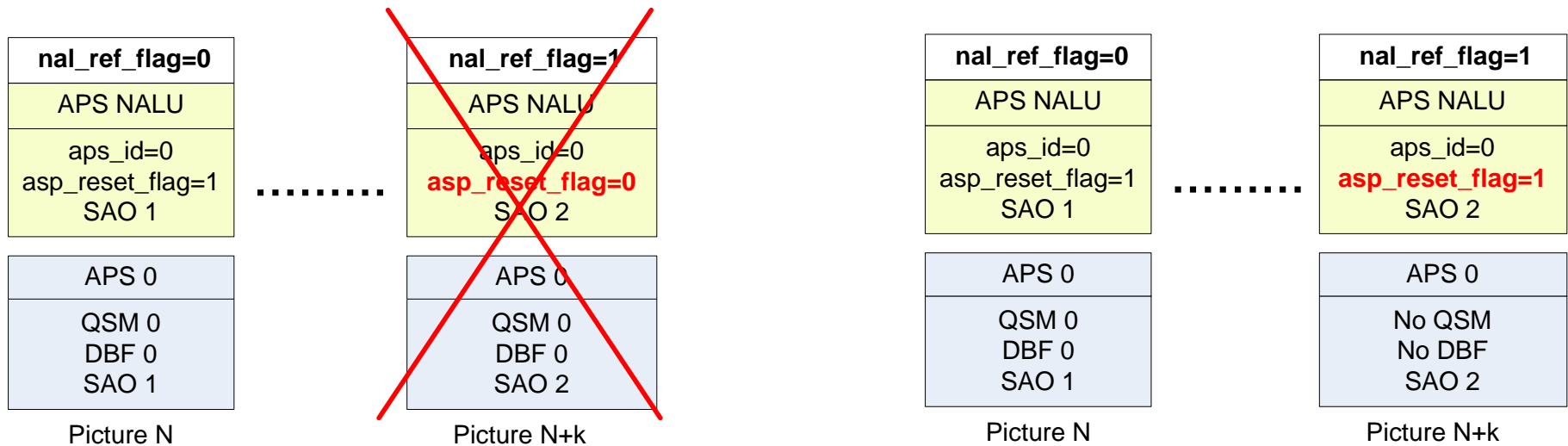
- Use `nal_ref_flag` to indicate priority of APS
 - For easy identification of ‘droppable’ APS
 - `nal_ref_flag=0` → only non-reference slices can refer to the APS
 - `nal_ref_flag=1` → can be referred by any slice



Proposal 3: APS for non-ref pictures

■ Case of conditional replacement

- An APS, previously 'droppable', cannot be partially updated and at the same time become non-droppable → must be reset
 - guarantees that all ref-slices refer to APSs entirely defined as non-droppable (no params were previously generated as droppable)



■ Comments

- Parsing: easy detection of the 'droppable/non-droppable' APS by checking `nal_ref` flag

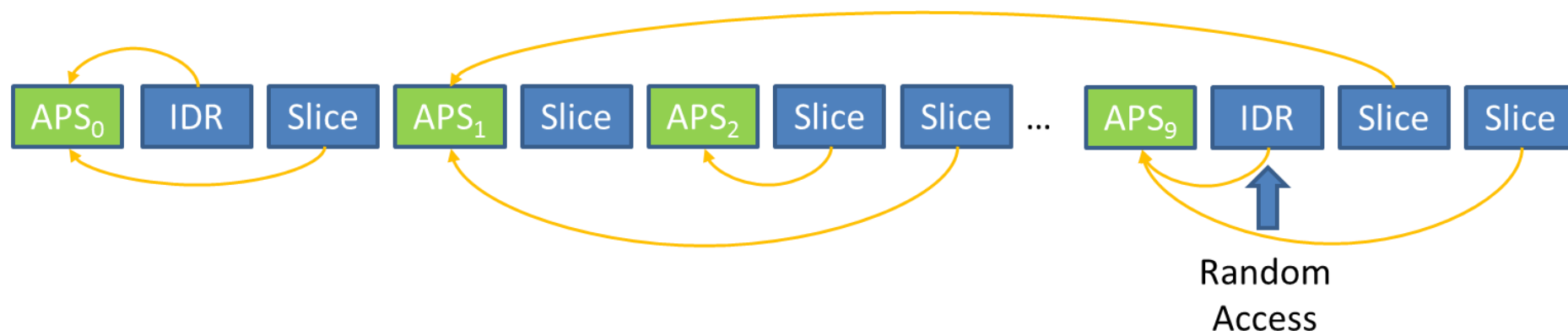
Proposal overview

- Proposal 1a–1b: Referencing multiple APS
 - Use of multiple identifiers in Slice Header
- Proposal 2: APS update
 - Conditional replenishment + reset mechanism
- Proposal 3: APS priority
 - APS for non-reference pictures
- **Proposal 4: mechanism for APS buffer reset**
 - Resynchronization points
 - Buffer period

Proposal 4: APS buffer reset

■ Buffer flushing necessary to guarantee random access points

- Transmission of APSs out-of-band not realistic (too many APSs, too frequent changing data)



- propose to refresh periodically the APS buffer to enable simple method to randomly access one picture
 - Buffer refreshed at each IDR or CRA
 - Or usage of Sliding window (predetermined buffering period)

Proposal 4: APS buffer reset

■ Define 2 syntax elements in SPS (or PPS)

- **aps_buffer_reset_at_idr_cra_flag**
- **aps_buffer_period** – indicates the maximum number of successive frames for which each APS NAL unit is valid

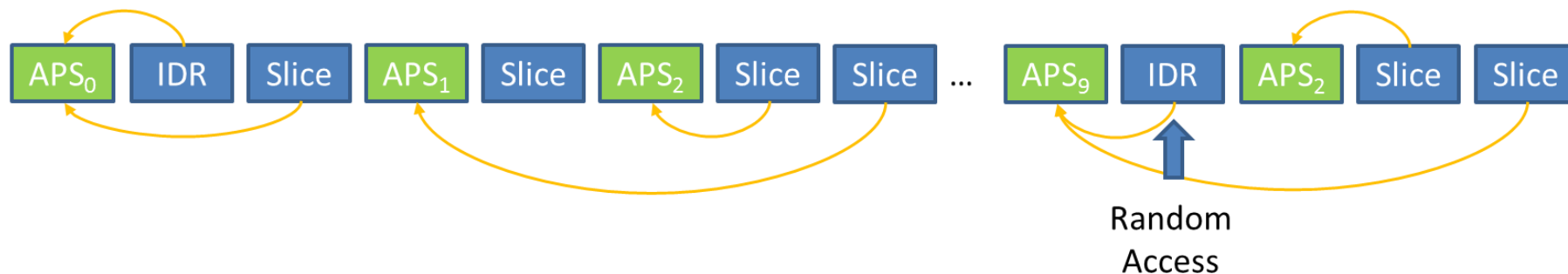
seq_parameter_set_rbsp() {	Descr
[...]	
aps_buffer_reset_at_idr_cra_flag	u(1)
if(aps_buffer_reset_at_idr_cdr_flag == 0)	
aps_buffer_period	ue(v)
[...]	

■ Comments

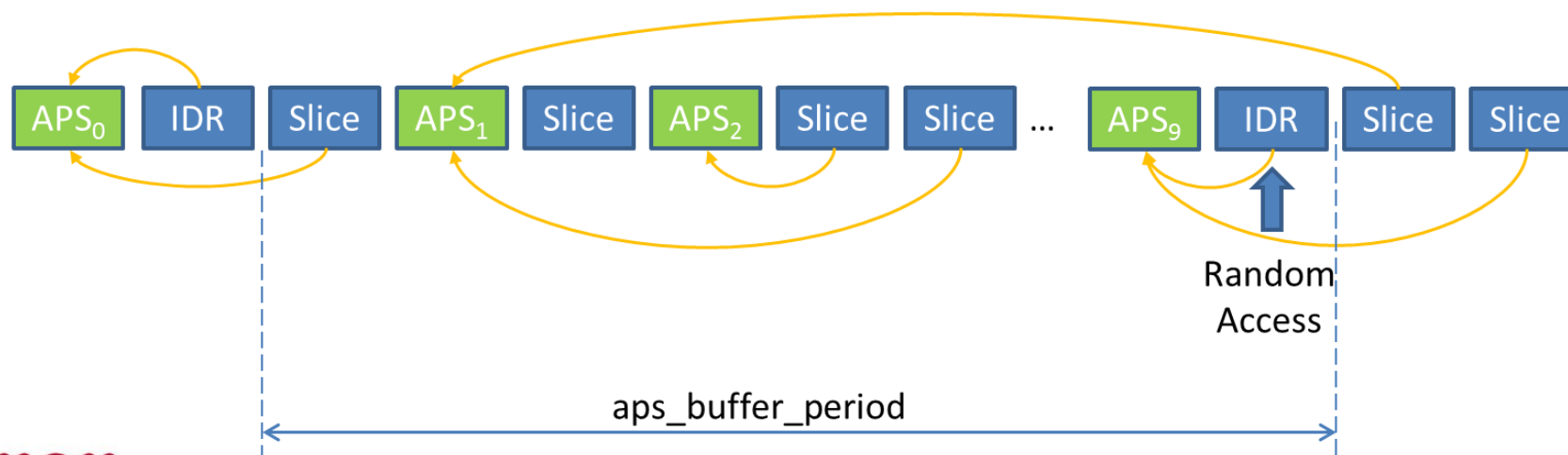
- Buffer period also present in other proposals, BUT period relates to latest aps_id (I0069) or nb of APS NALUs (I0082)
- No link with picture period → Issue: if APS NALUs are not regularly transmitted, need to parse far backward the stream to get these APS NALUs.

Proposal 4: APS buffer reset

■ Reset at IDR or CRA



■ Reset based on buffer period



Summary

■ Definition of APS buffer

- APSs of the buffer created or updated from APS NALUs

proposal	description	comments
1a	Direct referencing in SH to 1 APS per class of params	Signaling in SH
1b	Refer in SH to multiple APS for sets of params classes	Signaling in SH Generalize signaling principle of I0338
2	APS update with Reset mechanism (reset flag in APS NALU)	Similar to aps_intra_flag used in I0081 Also close behaviour to aps_update_flag from I0069
3	Usage of nal_ref_flag to manage APS priority (for ref/non-ref pictures)	
4	APS buffer reset (resync point for IDR/CRA, buffer period)	Resync at IDR/CRA similar to resync proposed in I0081 buffer period similar to I0069 and I0082, but counter based on pictures nb