

**I. On informing the discrepancy between
HM S/W and WD at IDR picture**

**II. Syntax for clean random access pictures
(JCTVC-G533)**

**Youngo Park
Heechul Yang
Chanyul Kim
Kwang Pyo Choi
JeongHoon Park
(Samsung)**

**I. On informing the discrepancy between
HM S/W and WD at IDR picture**

**II. Syntax for clean random access pictures
(JCTVC-G533)**

POC (Picture Order Count) in AVC

- ❖ POC (Picture Order Count) Type 0 (Most common used in industry)
 - $POC = POCLsb + POCLsb$
 - POCLsb is sent in each slice header
 - POCMSb is changed dependent on the difference b/w PrevPOCLsb and POCLsb
 - POC never wraps around
 - POC should be reset to 0 at IDR or memory_management_control_operation is equal to 5

❖ Example – POC

POCLsb wraps around at MaxPOCLsb

log2_max_pic_order_cnt_lsb_minus4 = 0
 POCLsb = [0,15] and MaxPOCLsb = 16

POCLsb (signaled)	0	1	2	3	...	14	15	0	1	2	...	14	15	0	1	2	...
POCMSb (derived)	0	0	0	0	...	0	0	16	16	16	...	16	16	32	32	32	...
POC (derived)	0	1	2	3	...	14	15	16	17	18	...	30	31	32	33	34	...

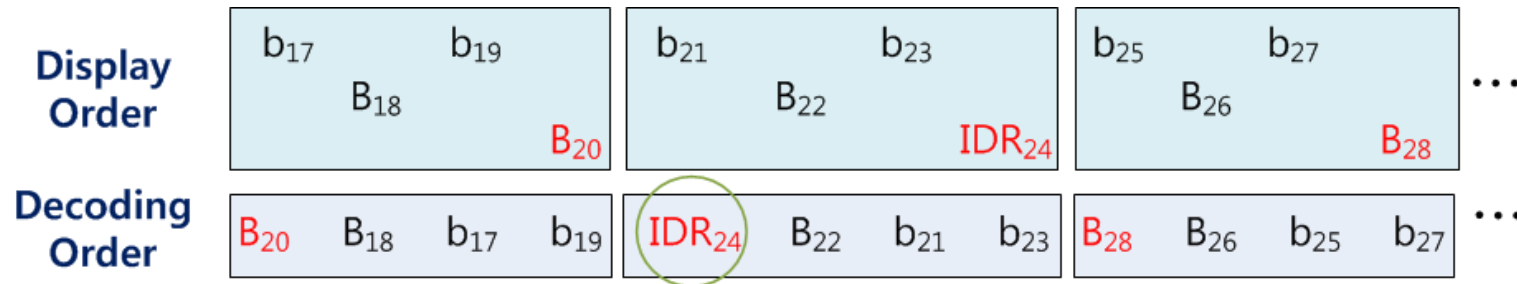
❖ Example – AVC Bitstream

844	F	P	845	F	B	846	F	IDR	847	F	P	848	F	P	849	F	P	850	F	P	851	F	P	852	F	P	853	F	I	854	F	P
00:00:28:166		00:00:28:133		00:00:28:200		00:00:28:233		00:00:28:266		00:00:28:300		00:00:28:333		00:00:28:366		00:00:28:400		00:00:28:433		00:00:28:466		00:00:28:500		00:00:28:533		00:00:28:566		00:00:28:600		00:00:28:633		00:00:28:666
POC = 130		POC = 128		POC = 0		POC = 2		POC = 4		POC = 6		POC = 8		POC = 10		POC = 12		POC = 14		POC = 16		POC = 18		POC = 20		POC = 22		POC = 24		POC = 26		POC = 28

- Note: AVC POC increases by two due to field coding

POC (Picture Order Count) in HM S/W

- ❖ POC is not reset to 0 at IDR picture in HM S/W

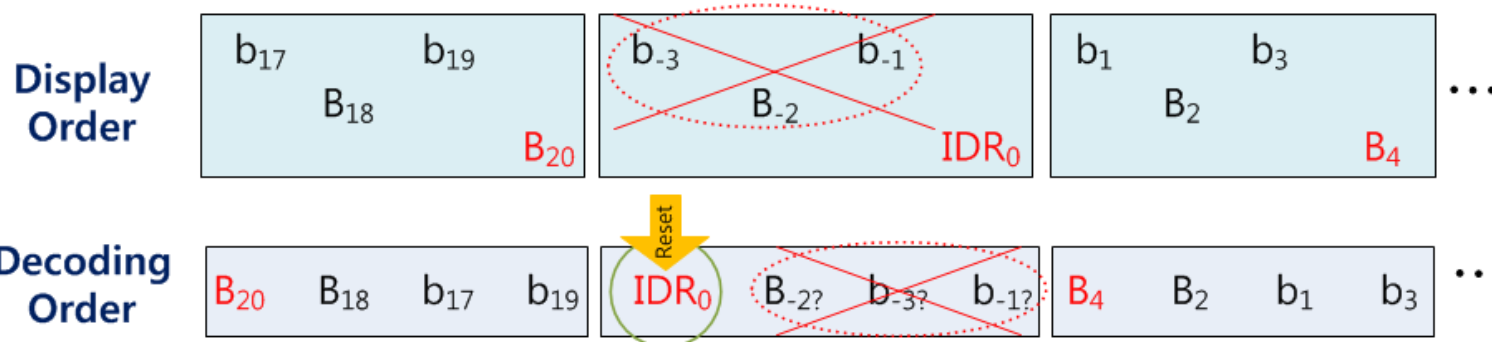


- ❖ If POC is reset to 0 at IDR complying with WD, abnormal case (negative POC) should be appeared to keep current display order

If POC is reset to 0, B_{22} is interpreted as B_6 ,
because POCMSb changed from 16 to 0 ($22=16(\text{MSB})+6(\text{LSB})$, $6=0(\text{MSB})+6(\text{LSB})$)

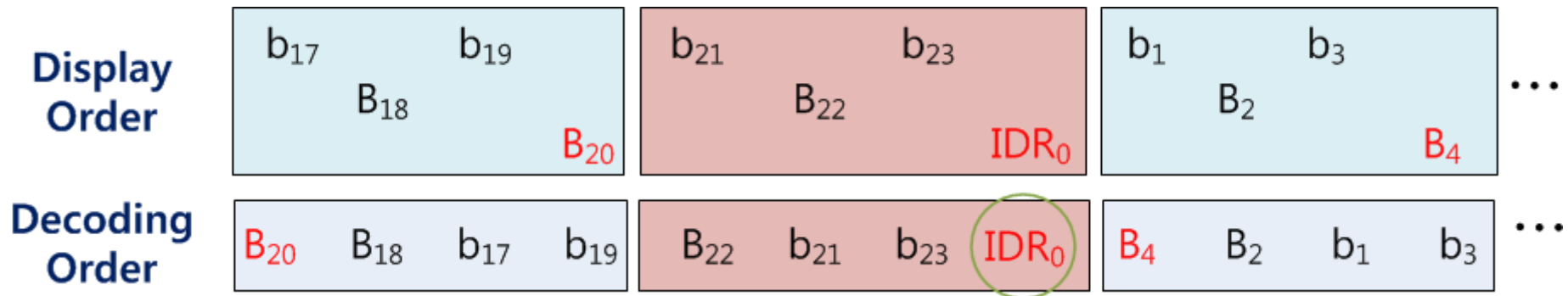


To keep same display order, POC should be negative value



POC (Picture Order Count) in S/W

- ❖ To comply with WD and display order for RA test condition, IDR picture **should be decoded in last order** when the GOP includes IDR picture



- ❖ Experimental Results

	Random Access HE			Random Access LC		
	Y	U	V	Y	U	V
Class A	-0.3%	0.6%	0.2%	-0.3%	0.4%	-0.1%
Class B	0.7%	1.3%	1.2%	0.6%	1.1%	0.7%
Class C	0.4%	1.1%	0.9%	0.4%	0.8%	0.6%
Class D	0.7%	1.4%	1.2%	0.7%	1.1%	1.0%
Class E						
Overall	0.4%	1.1%	0.9%	0.4%	0.9%	0.5%
	0.4%	1.1%	0.9%	0.4%	0.8%	0.5%
Enc Time[%]	98%			98%		
Dec Time[%]	101%			101%		

- LGE, Thanks for cross-checking

Conclusion

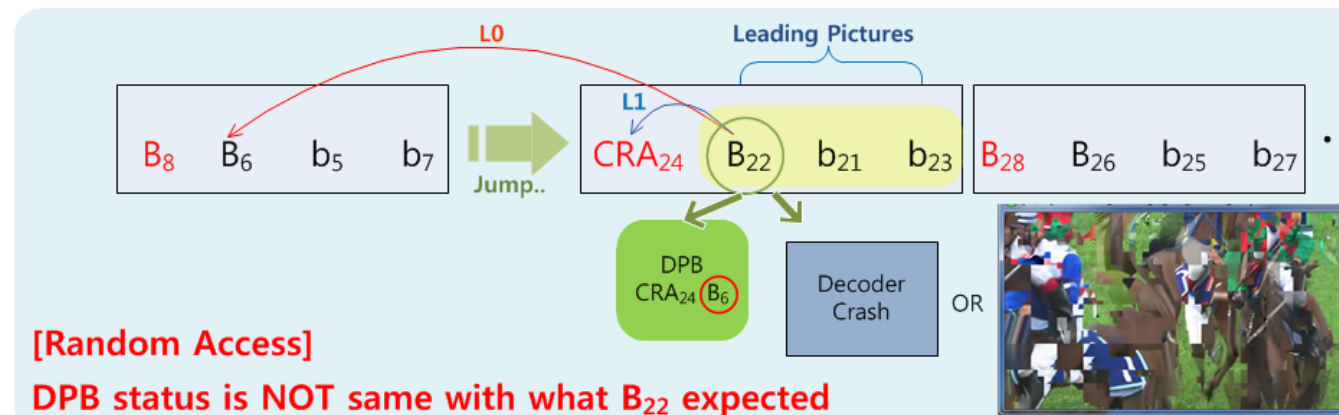
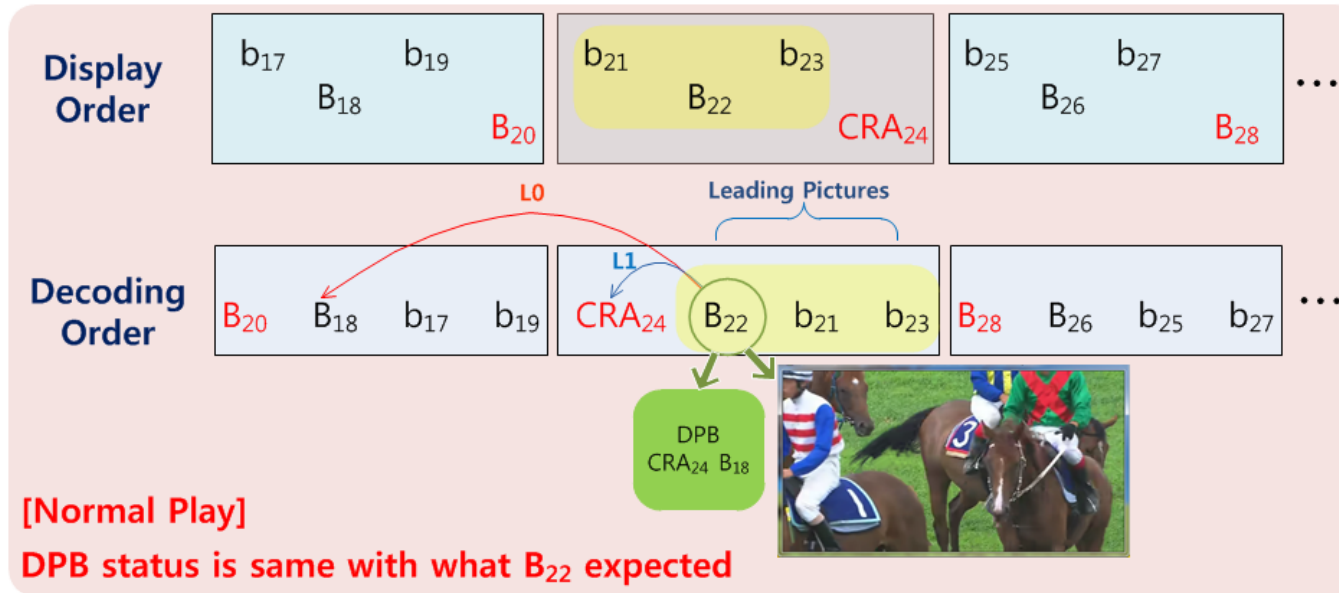
- ❖ Current HM S/W does not comply with WD POC (MSM/LSB) when IDR picture is decoded at RA test condition
 - POC should be reset to 0 at IDR picture
 - Negative POC should not be allowed
- ❖ Recommendation
 - It is recommended that the decoding order should be changed for GOP including IDR picture in HM S/W
 - S/W is ready

**I. On informing the discrepancy between
HM S/W and WD at IDR picture**

**II. Syntax for clean random access pictures
(JCTVC-G533)**

CRA picture at Random Access

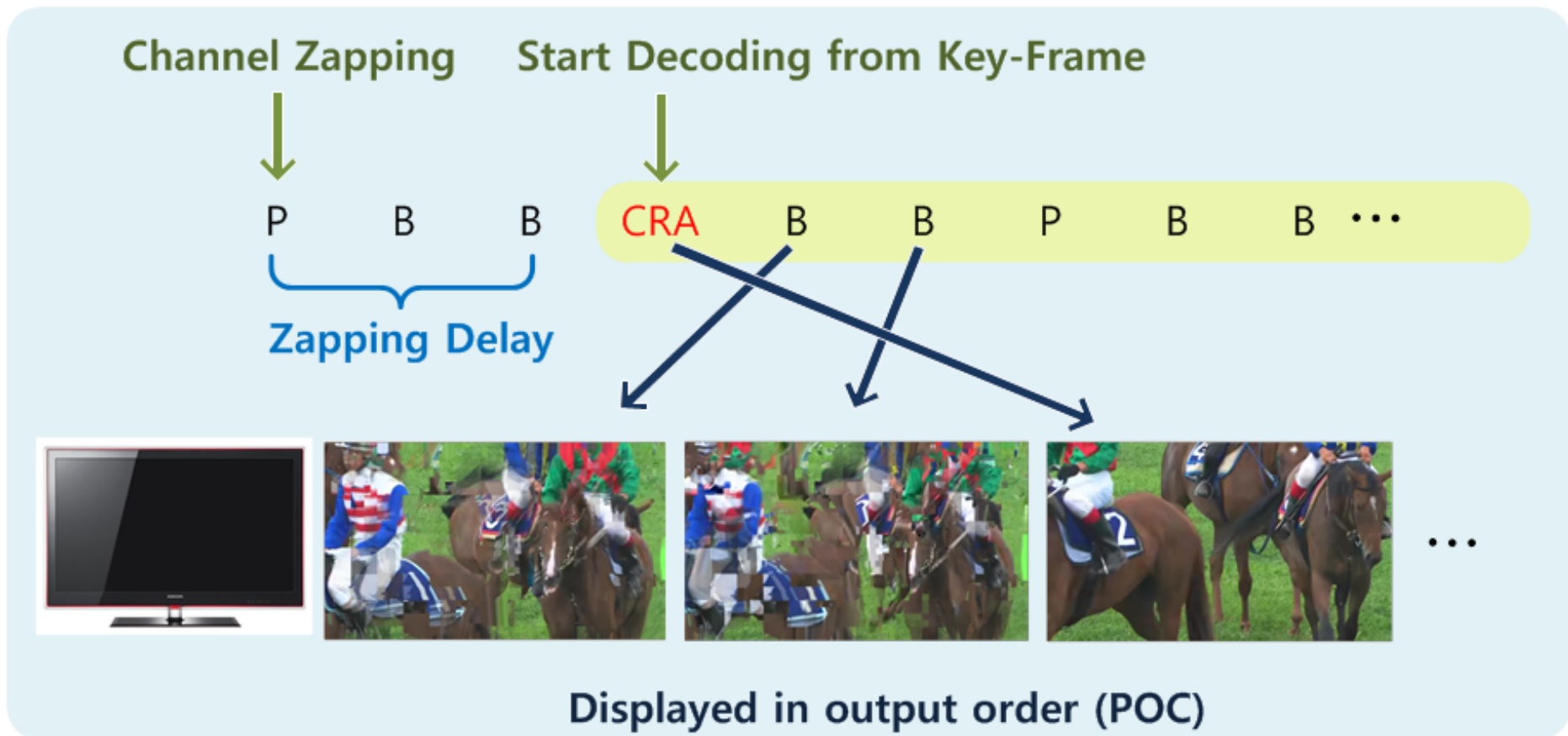
- ❖ At random access (Seek), **decoder could crash** or **produces corrupted picture** at leading picture decoding due to unexpected DPB status



CRA at Random Access – Scenario I

❖ TV Channel Zapping

- When leading picture is decoded, the DPB only has CRA picture that is not expected status
- The TV will start with corrupted pictures



CRA at Random Access – Scenario II

❖ Video Editing Application

- Video Editing is getting popular
- Video Editing is same concept with Random Access (Seek or Jump) in the sense of **finding a key frame** (random access point – IDR or CRA) and **copy & paste the bitstream from the key frame at any order**



CRA at Random Access – Scenario II

❖ Video Editing Application

[IDR Video Editing]

- Bitstream Copy & Paste would be ENOUGH to generate a new bitstream

Original
Bitstream

IDR₀ P₂ B₁ P₃ P₄ IDR₀ P₄ B₁ B₂ I₃ IDR₀ P₁ P₃ B₂

Edited
Bitstream

IDR₀ P₁ P₃ B₂ IDR₀ P₂ B₁ P₃

[CRA Video Editing]

- The bitstream of leading pictures should be DISCARDED by video editor on purpose
- The POC should be REGENERATED by video editor for continuous play of edited bitstream
(This is the reason why POC reset at key frame is recommend)

Original
Bitstream

CRA₀ P₂ B₁ P₃ CRA₆ B₄ B₅ P₇ I₈ CRA₁₀ B₉ P₁₂ B₁₁

Leading Pictures Leading Pictures

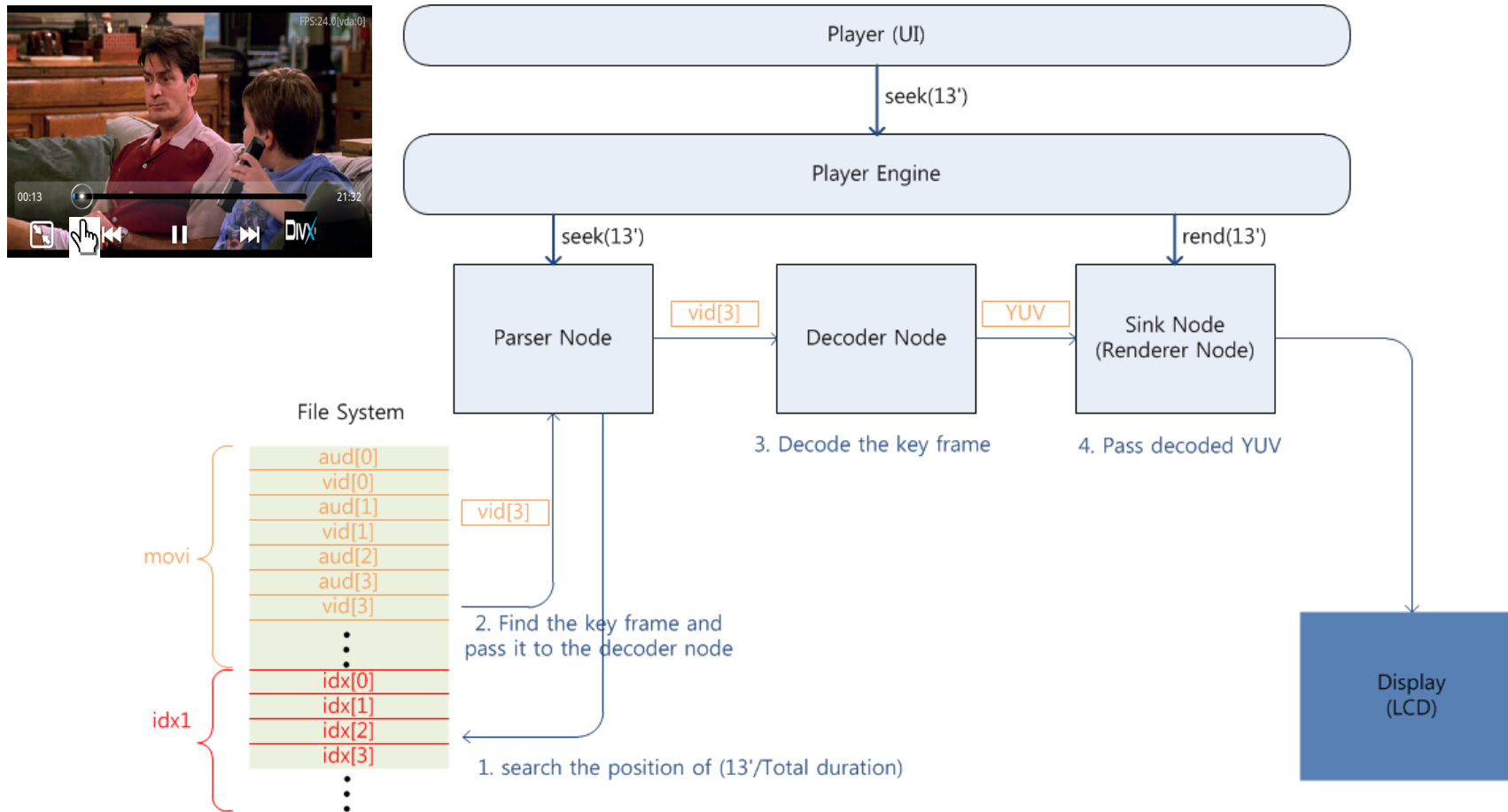
Edited
Bitstream

CRA₀ X P₂ B₁ CRA₃ X X P₄ I₅

CRA at Random Access – Scenario III

❖ Media Framework (DShow, OpenCore, GStreamer)

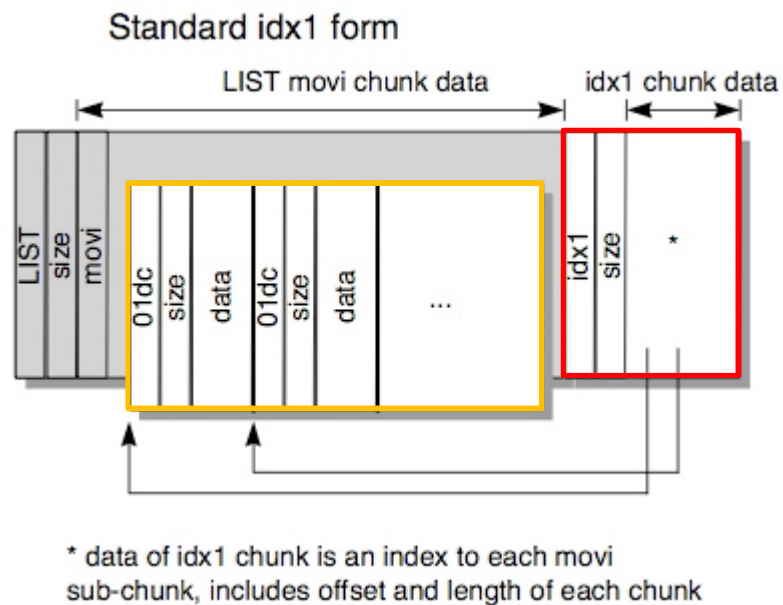
- The duty of “Decoder Node” is just to passively decode bitstream which is handed over from “Parser Node”



CRA at Random Access – Scenario III

❖ Media Framework

- File format example - AVI



struct LISTHEADER list[5]	movi	2800h	741E8Ah
char id[4]	LIST	2800h	4h
uint32 datalen	7609986	2804h	4h
char type[4]	movi	2808h	4h
struct genericblock audio[0]	01wb 7000	280Ch	1B60h
struct genericblock video[0]	00dc 10266	436Ch	2822h
struct genericblock audio[1]	01wb 468	6B8Eh	1DCh
struct genericblock video[1]	00dc 9103	6D6Ah	2398h
struct genericblock audio[2]	01wb 467	9102h	1DCh
struct genericblock video[2]	00dc 1271	92DEh	500h
struct genericblock audio[3]	01wb 467	97DEh	1DCh
struct genericblock video[3]	00dc 195	998Ah	CCh
struct genericblock audio[4]	01wb 467	9A86h	1DCh
struct genericblock video[4]	00dc 1738	9C62h	6D2h
struct genericblock audio[5]	01wb 467	A334h	1DCh
struct genericblock video[5]	00dc 400	A510h	198h

struct idx1HEADER idx1	idx1	74468Ah	187B8h
char id[4]	idx1	74468Ah	4h
uint32 datalen	100272	74468Eh	4h
struct AVIINDEXENTRY index[6267]		744692h	187B0h
struct AVIINDEXENTRY index[0]		744692h	10h
char ckid[4]	01wb	744692h	4h
DWORD dwFlags	16	744696h	4h
DWORD dwChunkOffset	4	74469Ah	4h
DWORD dwChunkLength	7000	74469Eh	4h
struct AVIINDEXENTRY index[1]		7446A2h	10h
char ckid[4]	00dc	7446A2h	4h
DWORD dwFlags	16	7446A6h	4h
DWORD dwChunkOffset	7012	7446AAh	4h
DWORD dwChunkLength	10266	7446AEh	4h
struct AVIINDEXENTRY index[2]		7446B2h	10h
char ckid[4]	01wb	7446B2h	4h
DWORD dwFlags	16	7446B6h	4h
DWORD dwChunkOffset	17286	7446BAh	4h
DWORD dwChunkLength	468	7446BEh	4h
struct AVIINDEXENTRY index[3]		7446C2h	10h
char ckid[4]	00dc	7446C2h	4h
DWORD dwFlags	0	7446C6h	4h
DWORD dwChunkOffset	17762	7446CAh	4h
DWORD dwChunkLength	9103	7446CEh	4h
struct AVIINDEXENTRY index[4]		7446D2h	10h
struct AVIINDEXENTRY index[5]		7446E2h	10h
struct AVIINDEXENTRY index[6]		7446F2h	10h
struct AVIINDEXENTRY index[7]		744702h	10h
struct AVIINDEXENTRY index[8]		744712h	10h

IDR picture vs. CRA picture

- ❖ IDR picture has two kinds of functionality
 1. Decoder Refresh functionality
 - When error is spread out in decoded picture, DPB buffer refresh can stop error propagation
 2. Random Access (Trick Modes, Seek) functionality
 - This is one of requirements in HEVC

- ❖ It is thought that CRA picture **has Decoder Refresh Functionality** but **has NOT Random Access Functionality**
 - **Independence from previous decoding status** is required in order to support Random Access Functionality
 - DPB should be refreshed
 - Leading picture should not be allowed
 - POC reset is recommended (video editing)
 - CRA picture is meaningful when it has leading picture

Solution

- ❖ To support random access at CRA picture, the system **should have a possibility to NOTIFY decoder** whether **the current decoding status is normal play or random access** via one flag in NAL unit header

- System

- If nal_unit_type is equal to 4 (CRA) and random access happens, then set the random_access_flag to one in system level. Otherwise set to zero

- Decoder

- If random_access_flag is set to 0
 - ✦ It means that current playing status is **normal play**, so **decode leading pictures and display them**
 - If random_access_flag is set to 1,
 - ✦ It meant that current playing status is **random access**, so just **skip leading pictures**

nal_unit(NumBytesInNALunit) {	Descriptor
forbidden_zero_bit	f(1)
nal_ref_idc	u(2)
nal_unit_type	u(5)
NumBytesInRBSP = 0	
nalUnitHeaderBytes = 1	
if(nal_unit_type == 1 nal_unit_type == 4 nal_unit_type == 5) {	
temporal_id	u(3)
output_flag	u(1)
reserved_one_4bits	u(4)
nalUnitHeaderBytes += 1	
}	



nal_unit(NumBytesInNALunit) {	Descriptor
forbidden_zero_bit	f(1)
nal_ref_idc	u(2)
nal_unit_type	u(5)
NumBytesInRBSP = 0	
nalUnitHeaderBytes = 1	
if(nal_unit_type == 1 nal_unit_type == 4 nal_unit_type == 5) {	
temporal_id	u(3)
output_flag	u(1)
random_access_flag	u(1)
reserved_one_3bits	u(3)
nalUnitHeaderBytes += 1	
}	

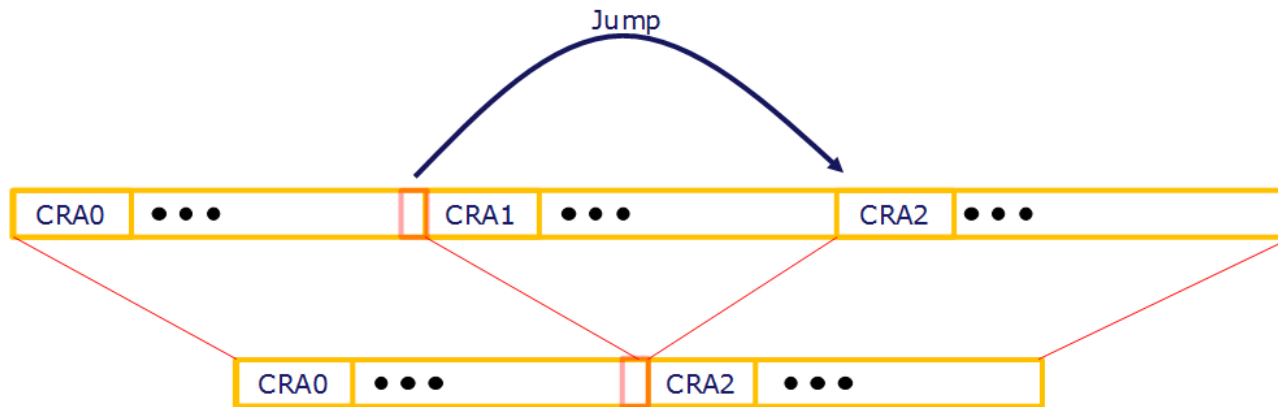
Conclusion

- ❖ We proposed two methods to give CRA pictures RA functionality
 - JCTVC-F604
 - The bitstream itself can notify decoder the current playing status w/o the help of system
 - ✦ (Drawback) The new syntax should be added in slice header and encoder/decoder implementation could be complex
 - JCTVC-G533
 - The system (parser, demuxer) should notify decoder the current playing status by "random_access_flag" in NAL header (It needs system help)
 - ✦ (Drawback) It looks simple but system should know and care about the bitstream
- ❖ Recommendation
 1. It is recommended changing the term from CRA (clean random access) to DDR (deferred decoder refresh) which was an initial term
 - CRA term is confusing. Is this for Random Access?
 - Current CRA picture still supports Decoder Refresh Functionality
 2. If not, it is noted that considering F604 or G533 to support Random Access Functionality for CRA picture
 - Regarding G533, It is true that the notification from system to codec looks like awkward so it is HIGHLY recommended being verified by system experts

Experiments

❖ Encoder

- Generate a truncated bitstream to simulate random access situation
- Set "random_access_flag" to one in case of CRA pictures



❖ Decoder

- Skip leading picture when "random_access_flag" is set to 1
- S/W and WD is available

❖ Cross-check

- Thanks to LGE