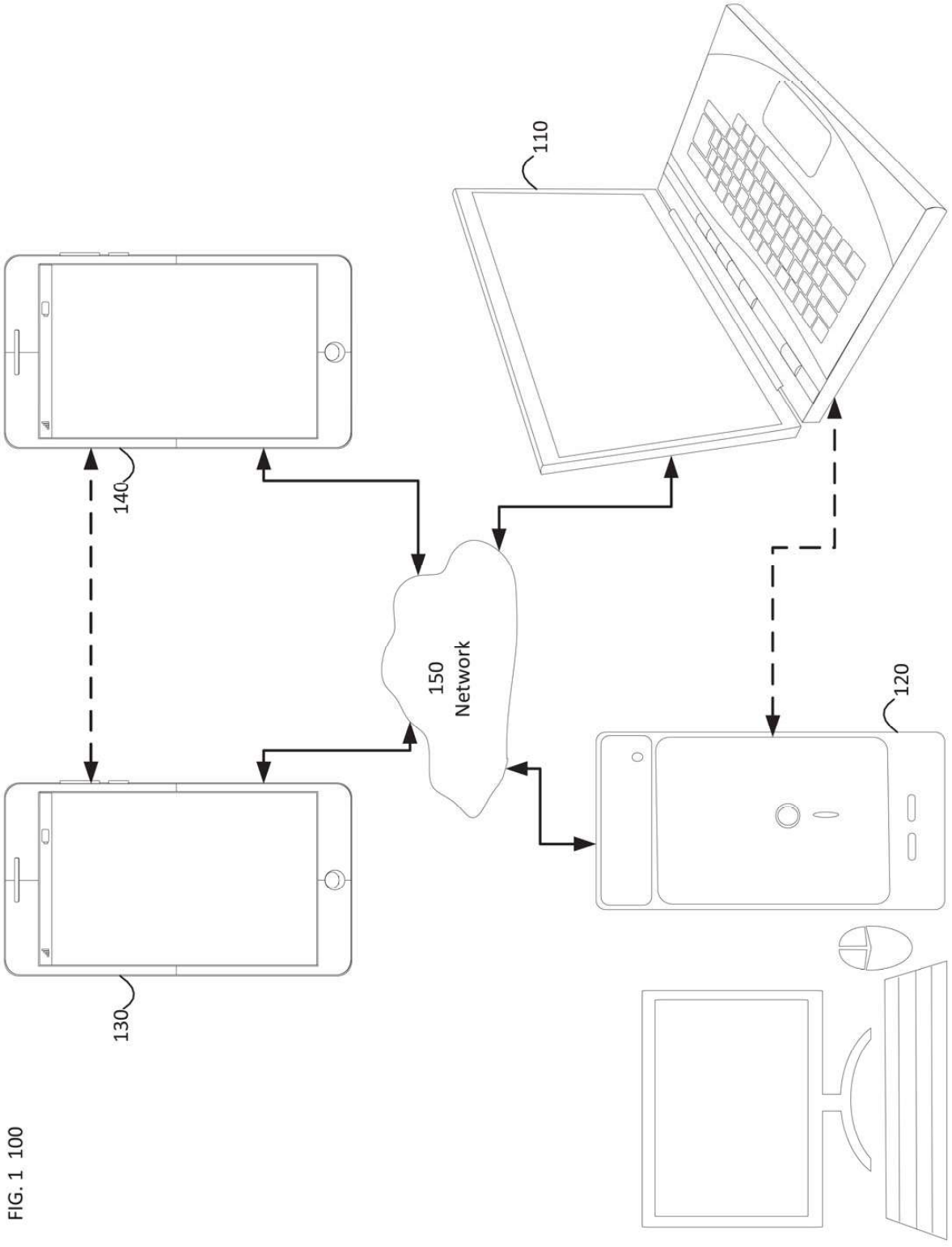


FIG. 1 100

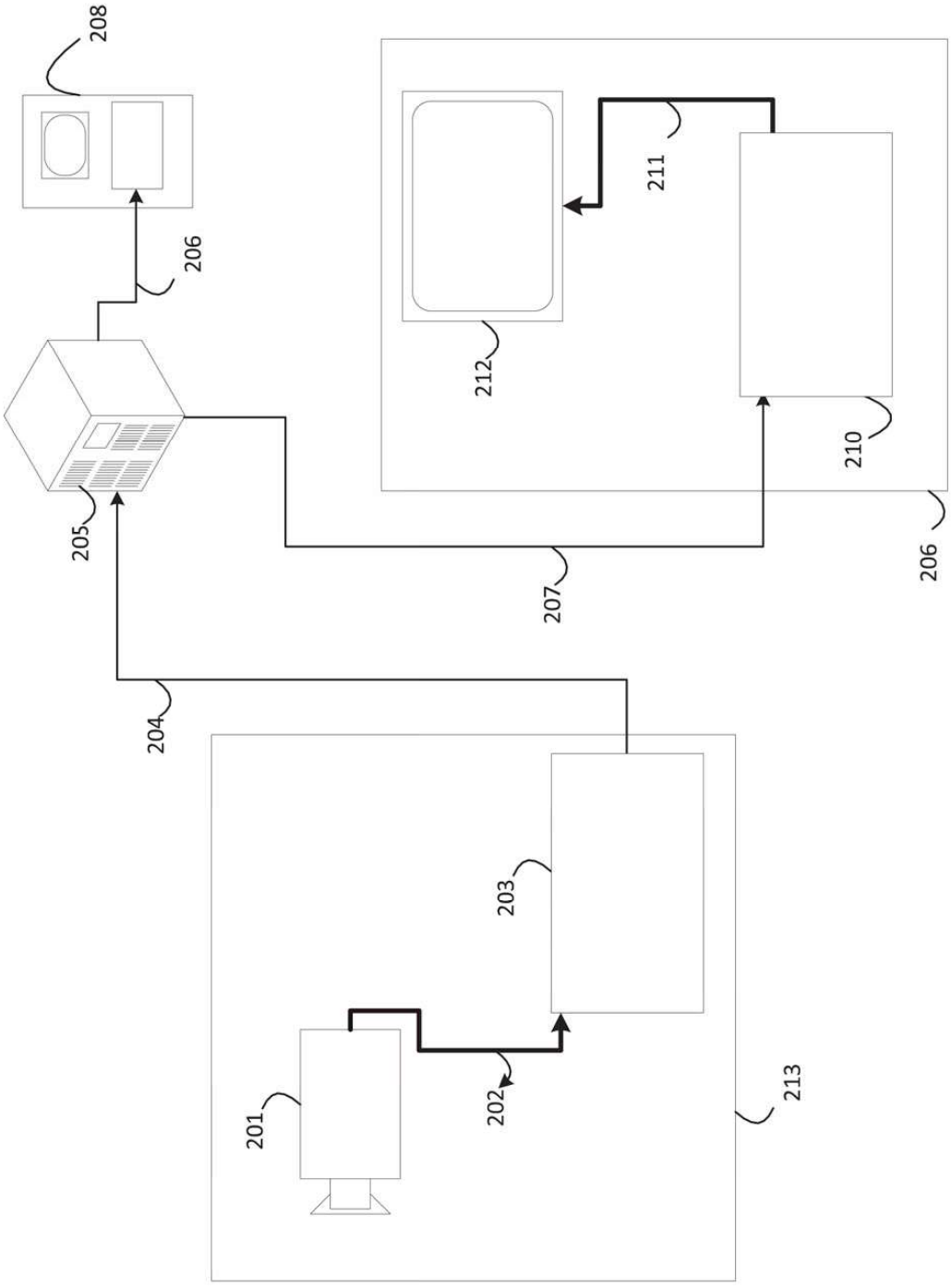


FIG. 2

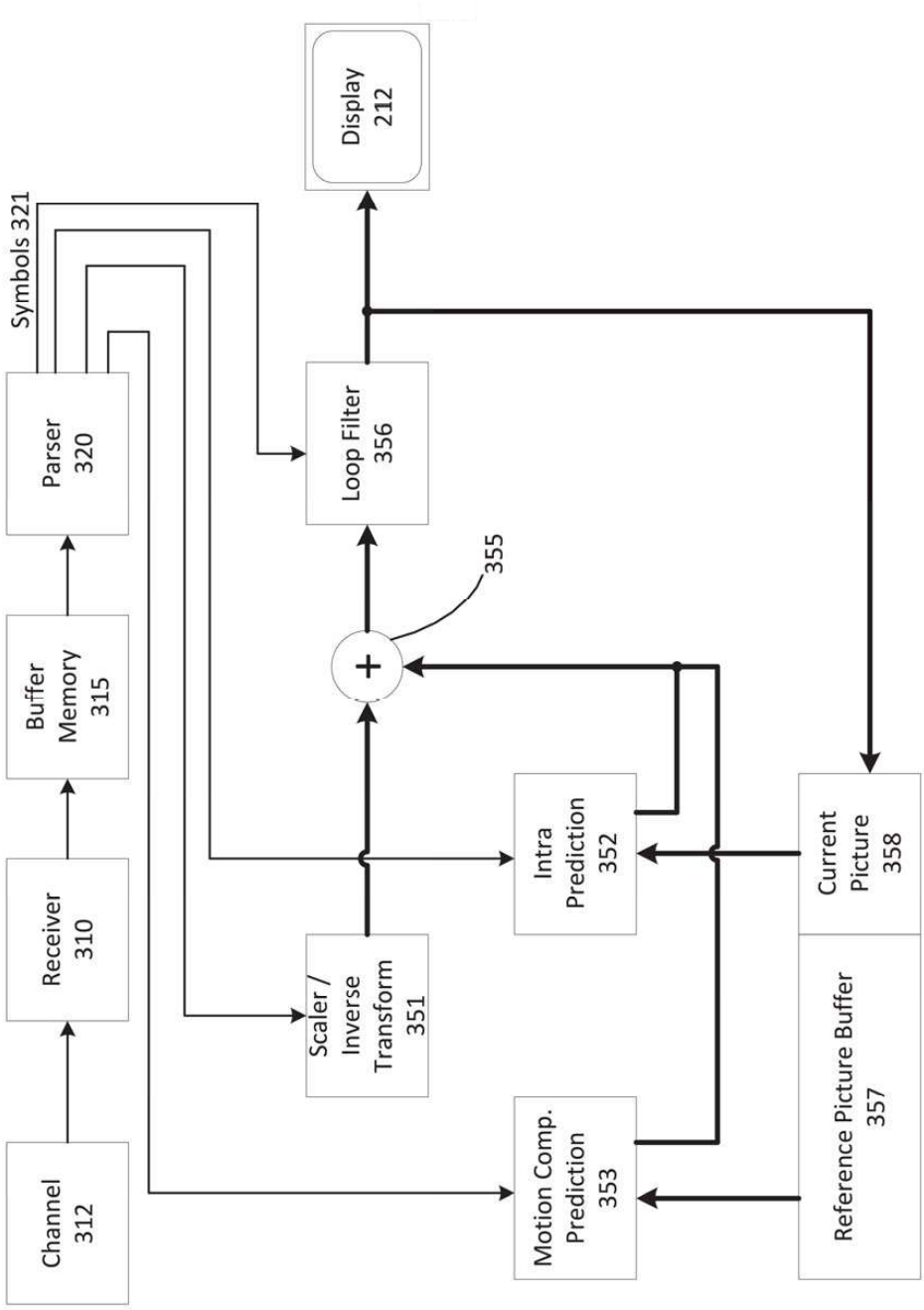


FIG. 3 Decoder 210

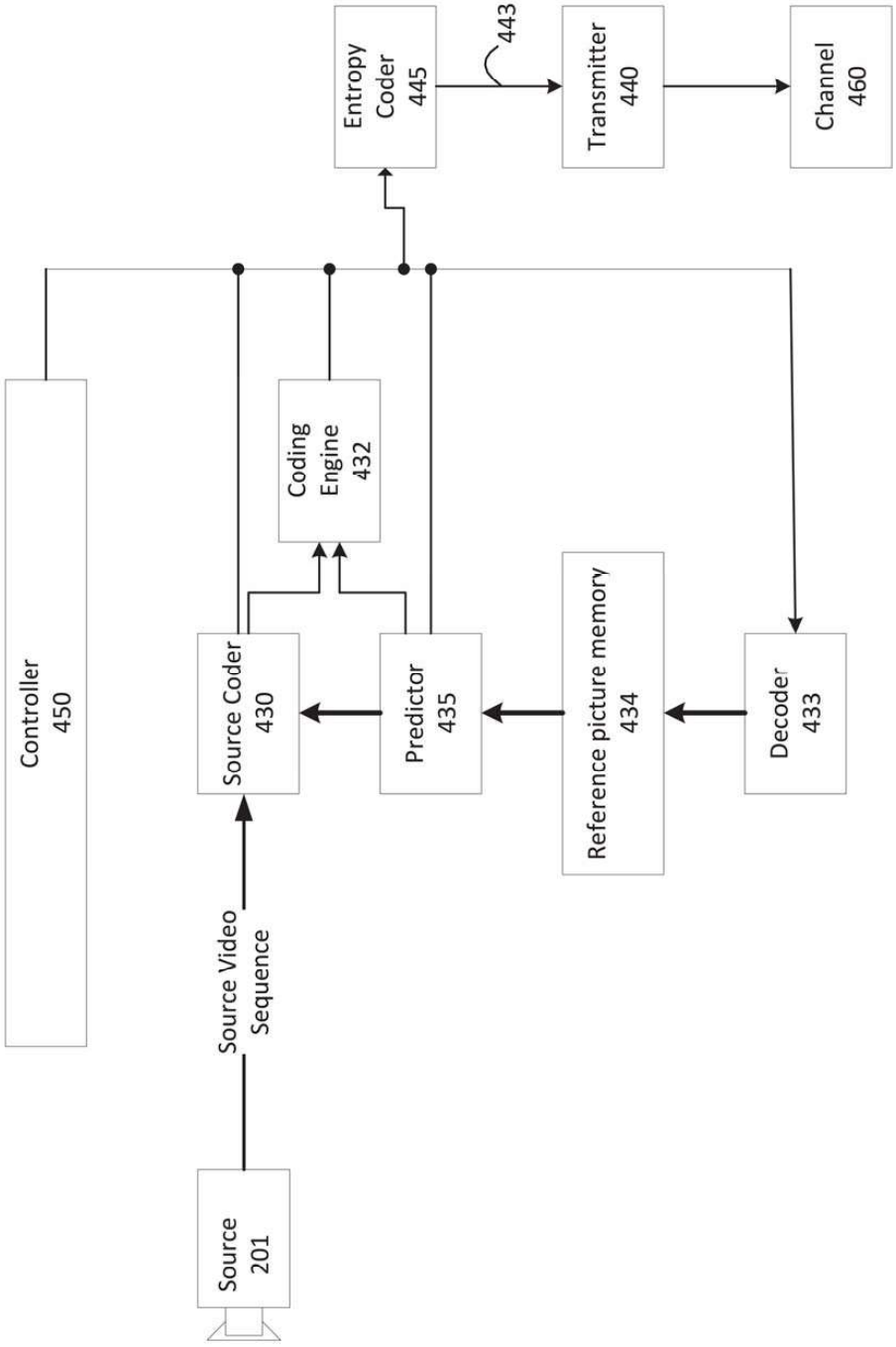


FIG. 4 Encoder 203

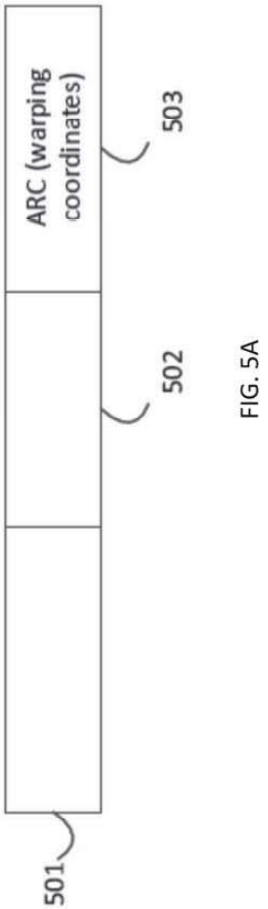

(H.263 Annex P)
ARC (warping coordinates)
503
502
501

FIG. 5A

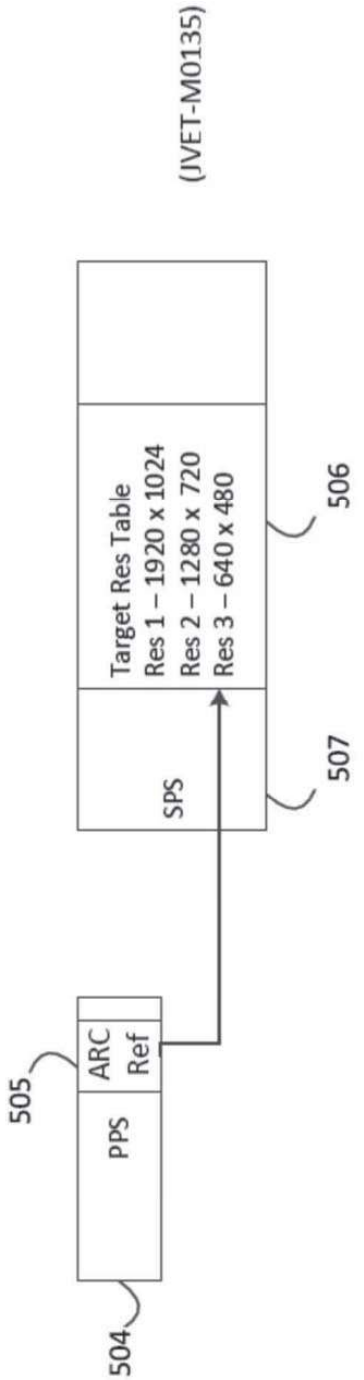

(JVET-M0135)
Target Res Table
Res 1 – 1920 x 1024
Res 2 – 1280 x 720
Res 3 – 640 x 480
506
SPS
507
ARC Ref
505
PPS
504

FIG. 5B

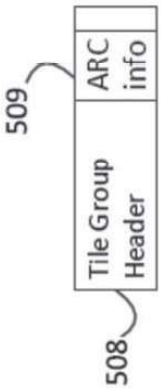


FIG. 5C



FIG. 5D

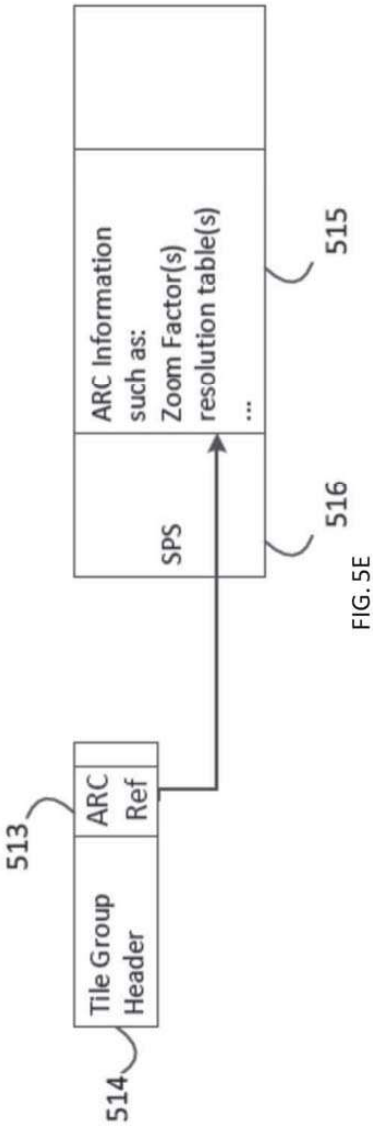


FIG. 5E

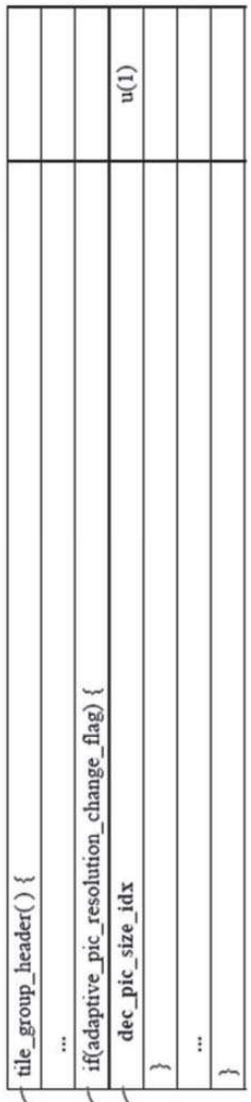

601
603
602
tile_group_header( ) {
...
if(adaptive_pic_resolution_change_flag) {
dec_pic_size_idx
u(1)
}
...
}

FIG. 6A

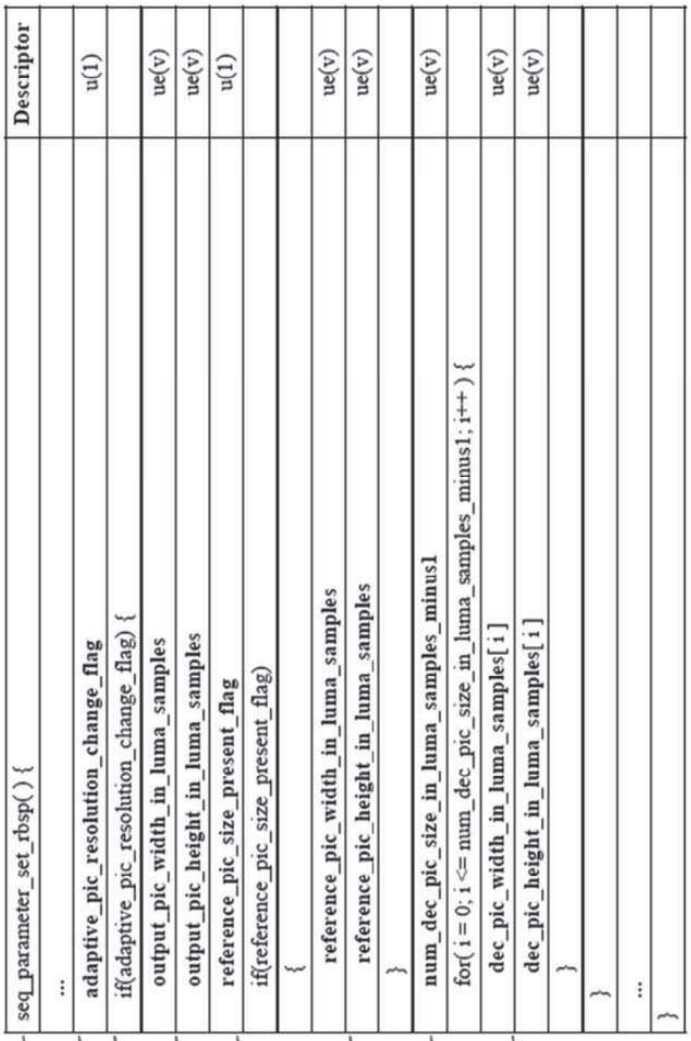

610
611
612
613
614
615
616
617
seq_parameter_set_rbsp( ) {
Descriptor
...
adaptive_pic_resolution_change_flag
u(1)
if(adaptive_pic_resolution_change_flag) {
output_pic_width_in_luma_samples
ue(v)
output_pic_height_in_luma_samples
ue(v)
reference_pic_size_present_flag
u(1)
if(reference_pic_size_present_flag)
{
reference_pic_width_in_luma_samples
ue(v)
reference_pic_height_in_luma_samples
ue(v)
}
num_dec_pic_size_in_luma_samples_minus1
ue(v)
for(i = 0; i <= num_dec_pic_size_in_luma_samples_minus1; i++) {
dec_pic_width_in_luma_samples[i]
ue(v)
dec_pic_height_in_luma_samples[i]
ue(v)
}
}
...
}

FIG. 6B

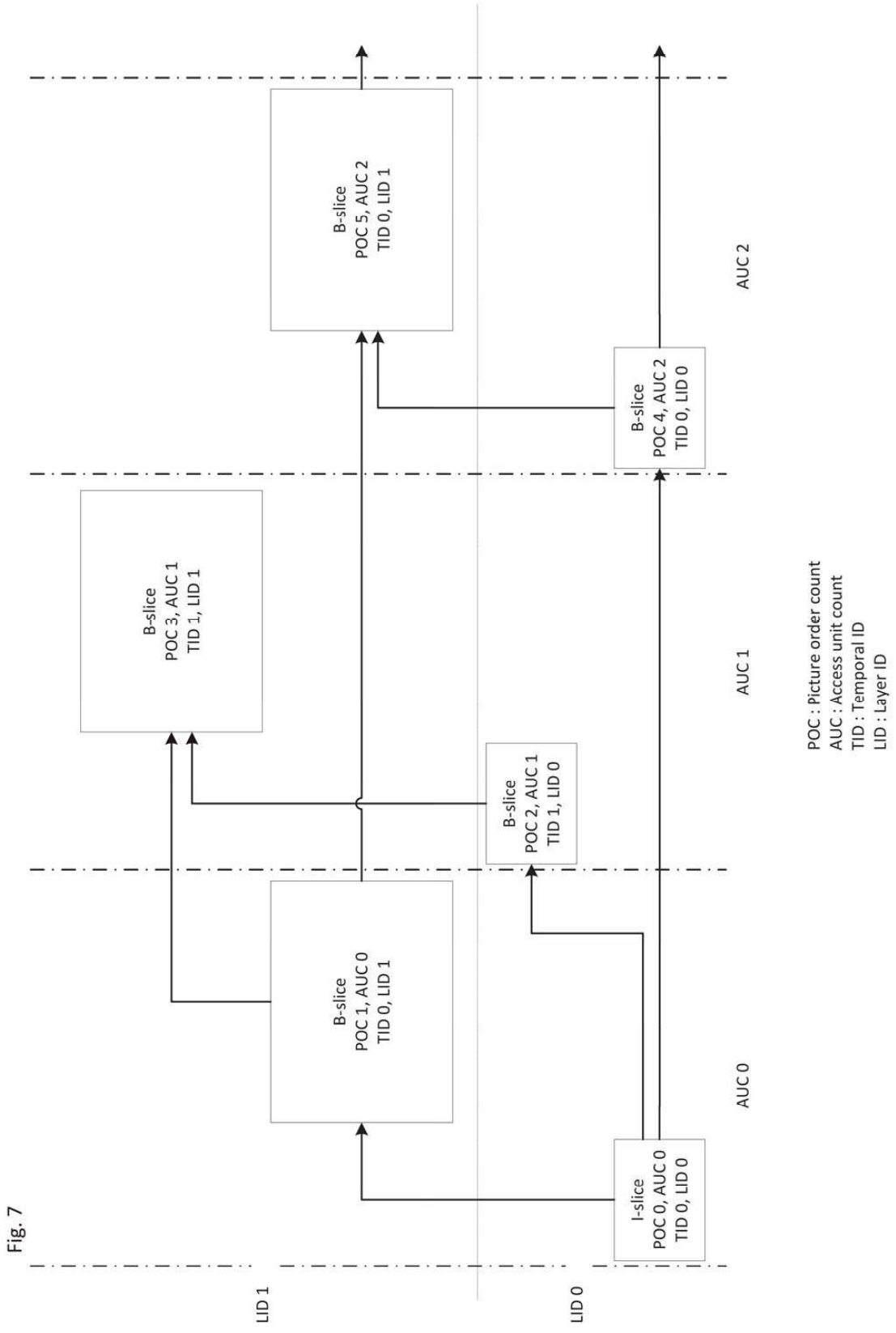

B-slice
POC 5, AUC 2
TID 0, LID 1
B-slice
POC 4, AUC 2
TID 0, LID 0
B-slice
POC 3, AUC 1
TID 1, LID 1
B-slice
POC 2, AUC 1
TID 1, LID 0
B-slice
POC 1, AUC 0
TID 0, LID 1
I-slice
POC 0, AUC 0
TID 0, LID 0
AUC 2
AUC 1
AUC 0
LID 1
LID 0
POC : Picture order count
AUC : Access unit count
TID : Temporal ID
LID : Layer ID

Fig. 7

Fig. 8

video_parameter_set_rbsp( ) {	Descriptor
vps_video_parameter_set_id	u(4)
vps_max_layers_minus1	u(8)
for(i = 0; i <= vps_max_layers_minus1; i++) {	
vps_included_layer_id[i]	u(7)
vps_reserved_zero_bit	u(1)
}	
vps_constraint_info_present_flag	u(1)
vps_constant_poc_cycle_per_au	u(1)
if(vps_constant_poc_cycle_per_au)	
vps_poc_cycle_au	u(8)
…	
}	

slice_header( ) {	Descriptor
slice_pic_parameter_set_id	ue(v)
if(rect_slice_flag || NumBricksInPic > 1)	
slice_address	u(v)
if(!rect_slice_flag && !single_brick_per_slice_flag)	
num_bricks_in_slice_minus1	ue(v)
slice_type	ue(v)
if(NalUnitType == GRA_NUT)	
recovery_poc_cnt	se(v)
slice_pic_order_cnt_lsb	u(v)
…	
if(!vps_constant_poc_cycle_per_au)	
slice_poc_cycle_au	u(8)
}	

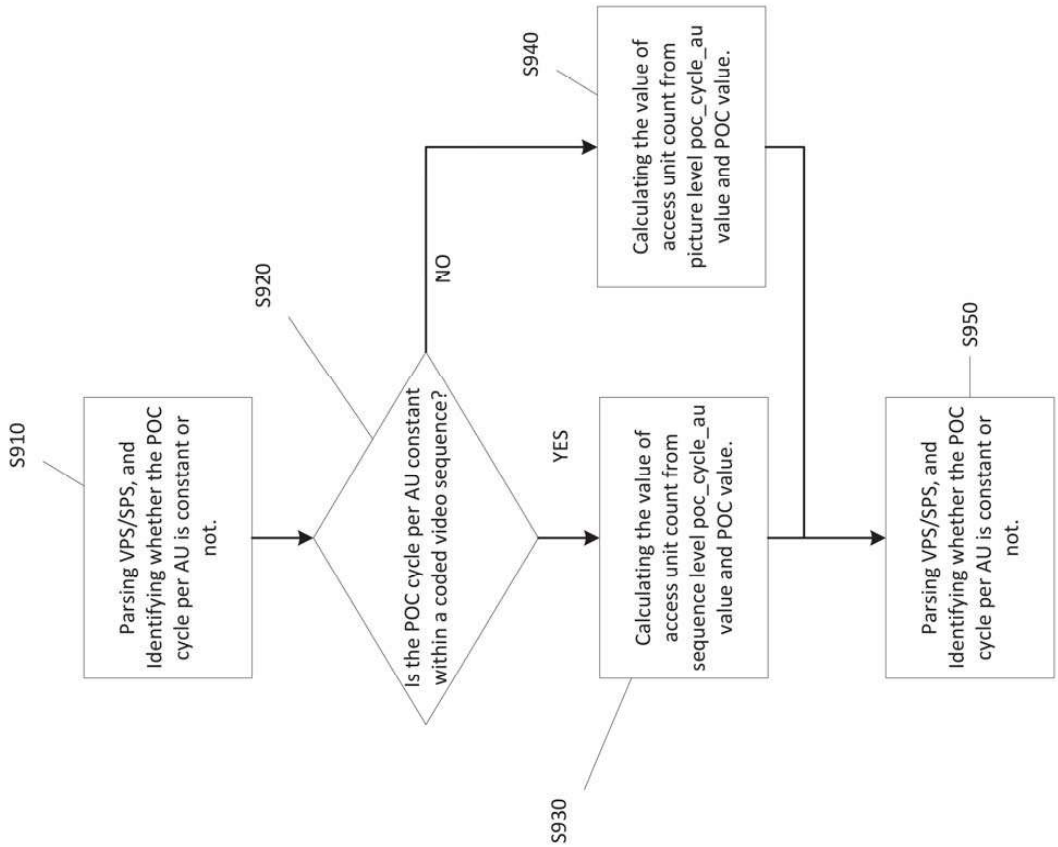

S910
Parsing VPS/SPS, and Identifying whether the POC cycle per AU is constant or not.
S920
Is the POC cycle per AU constant within a coded video sequence?
YES
NO
S930
Calculating the value of access unit count from sequence level poc_cycle_au value and POC value.
S940
Calculating the value of access unit count from picture level poc_cycle_au value and POC value.
S950
Parsing VPS/SPS, and Identifying whether the POC cycle per AU is constant or not.

Fig. 9

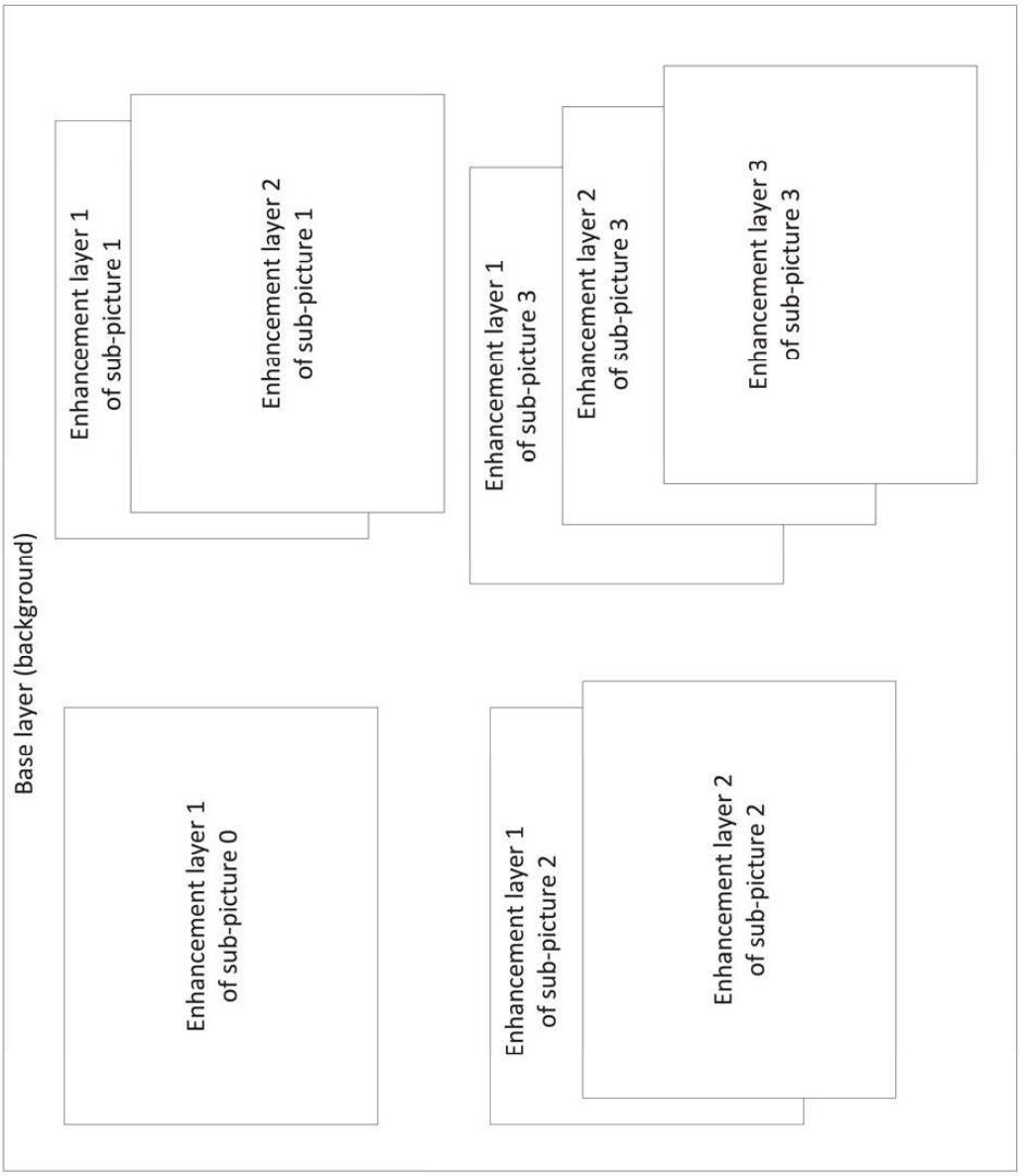

Base layer (background)
Enhancement layer 1 of sub-picture 0
Enhancement layer 1 of sub-picture 1
Enhancement layer 2 of sub-picture 1
Enhancement layer 1 of sub-picture 2
Enhancement layer 2 of sub-picture 2
Enhancement layer 1 of sub-picture 3
Enhancement layer 2 of sub-picture 3
Enhancement layer 3 of sub-picture 3

FIG. 10

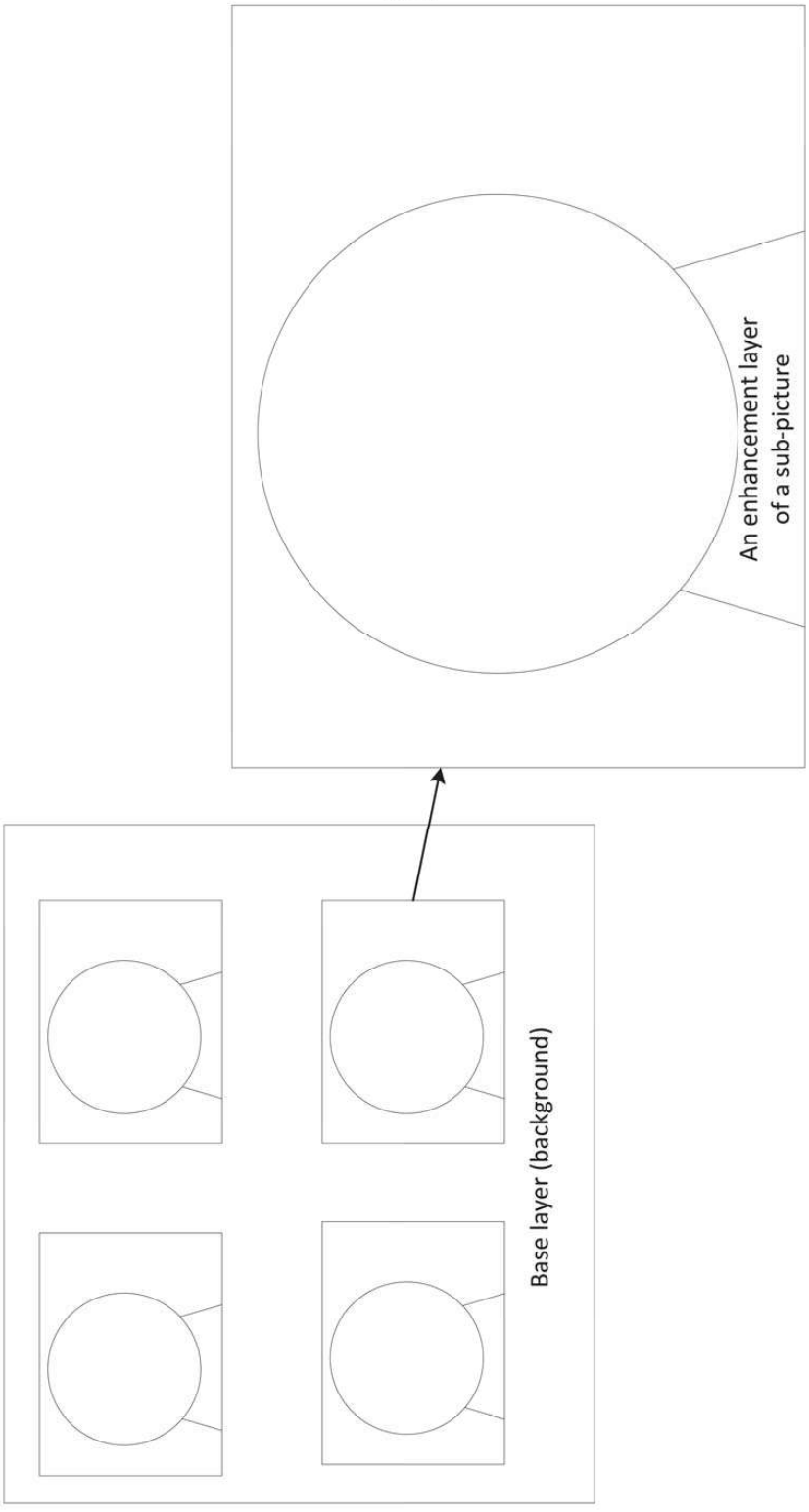

An enhancement layer
of a sub-picture
Base layer (background)

FIG. 11

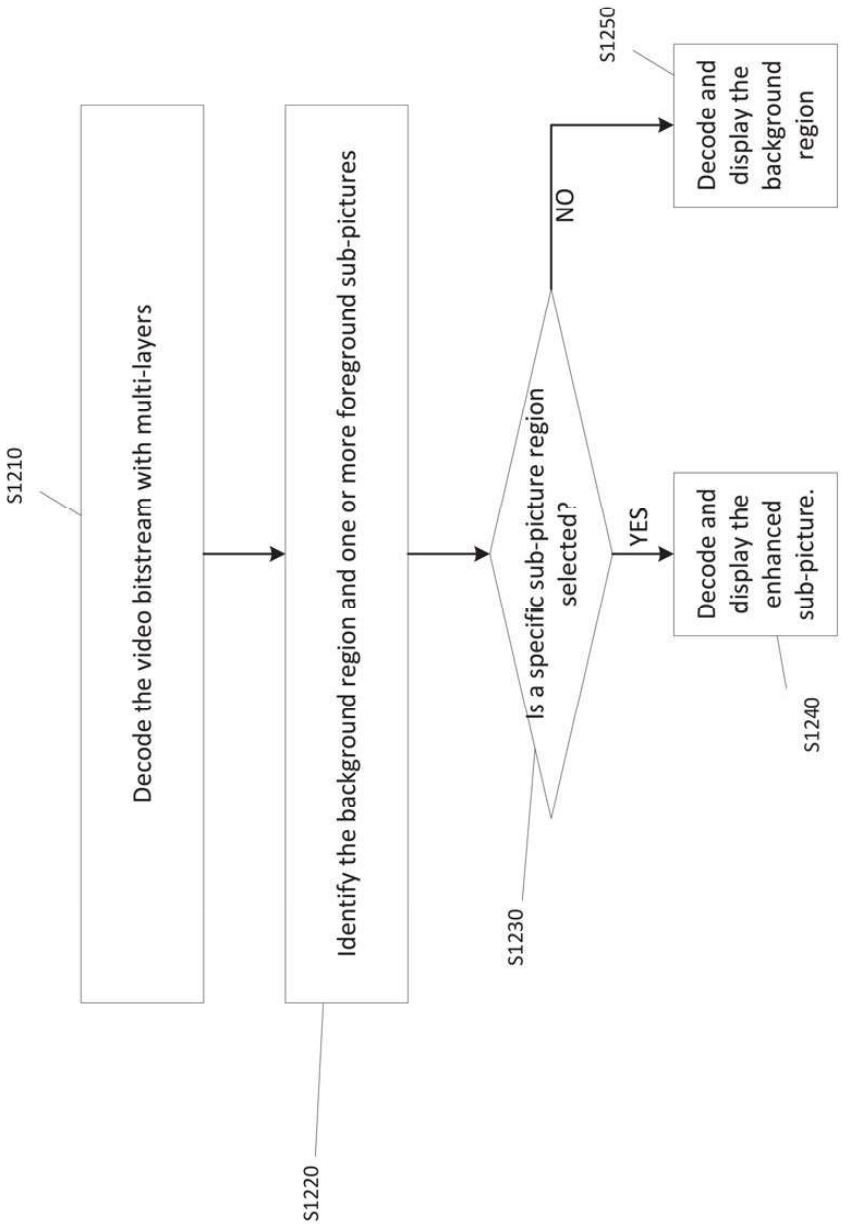

S1210
Decode the video bitstream with multi-layers
S1220
Identify the background region and one or more foreground sub-pictures
S1230
Is a specific sub-picture region selected?
YES
NO
S1240
Decode and display the enhanced sub-picture.
S1250
Decode and display the background region

Fig. 12

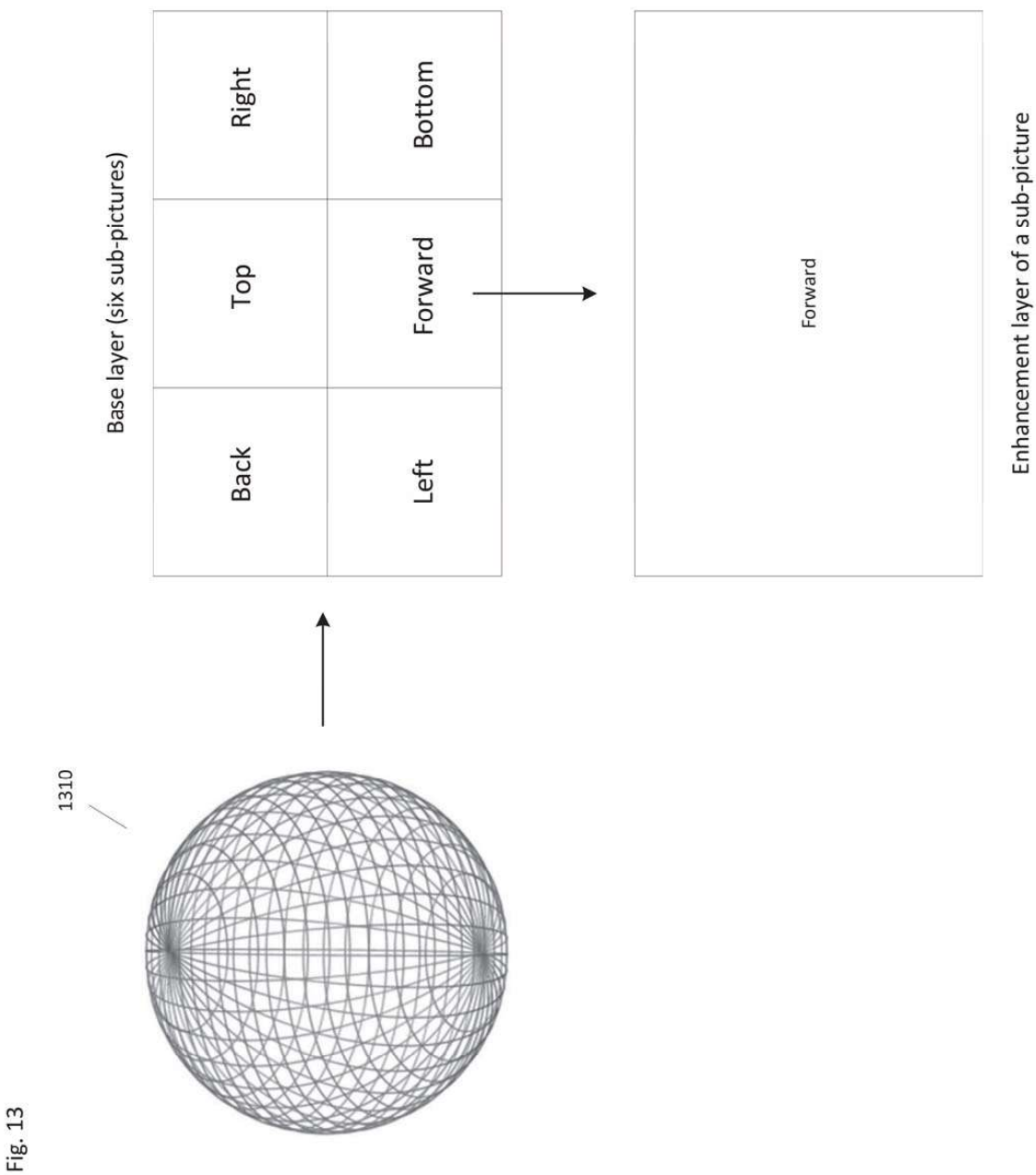
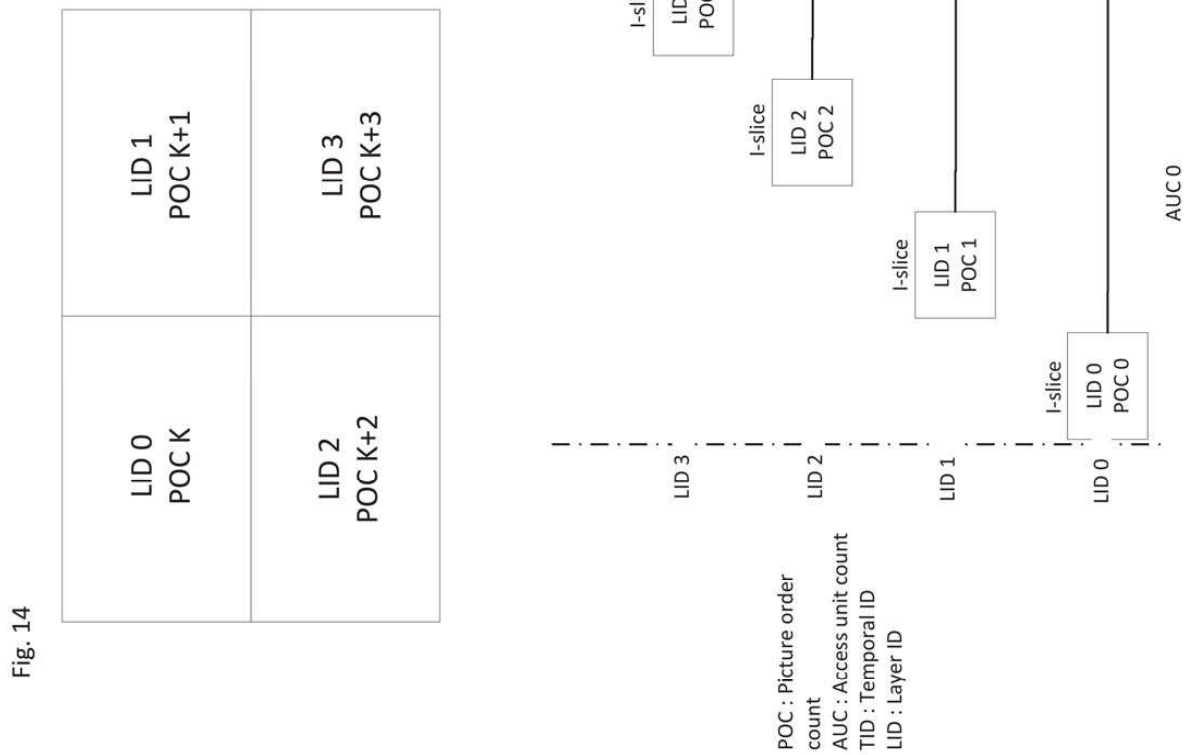

1310
Base layer (six sub-pictures)
Back
Top
Right
Left
Forward
Bottom
Forward
Enhancement layer of a sub-picture

Fig. 13

Fig. 14


LID 0
POC K
LID 1
POC K+1
LID 2
POC K+2
LID 3
POC K+3
X_offset, Y_offset
Height
Width
LID 3
POC K+3
POC : Picture order count
AUC : Access unit count
TID : Temporal ID
LID : Layer ID
LID 3
LID 2
LID 1
LID 0
I-slice
LID 3 POC 3
LID 2 POC 2
LID 1 POC 1
LID 0 POC 0
LID 3 POC 7
LID 2 POC 6
LID 1 POC 5
LID 0 POC 4
LID 1 POC 9
LID 0 POC 8
AUC 0
AUC 1

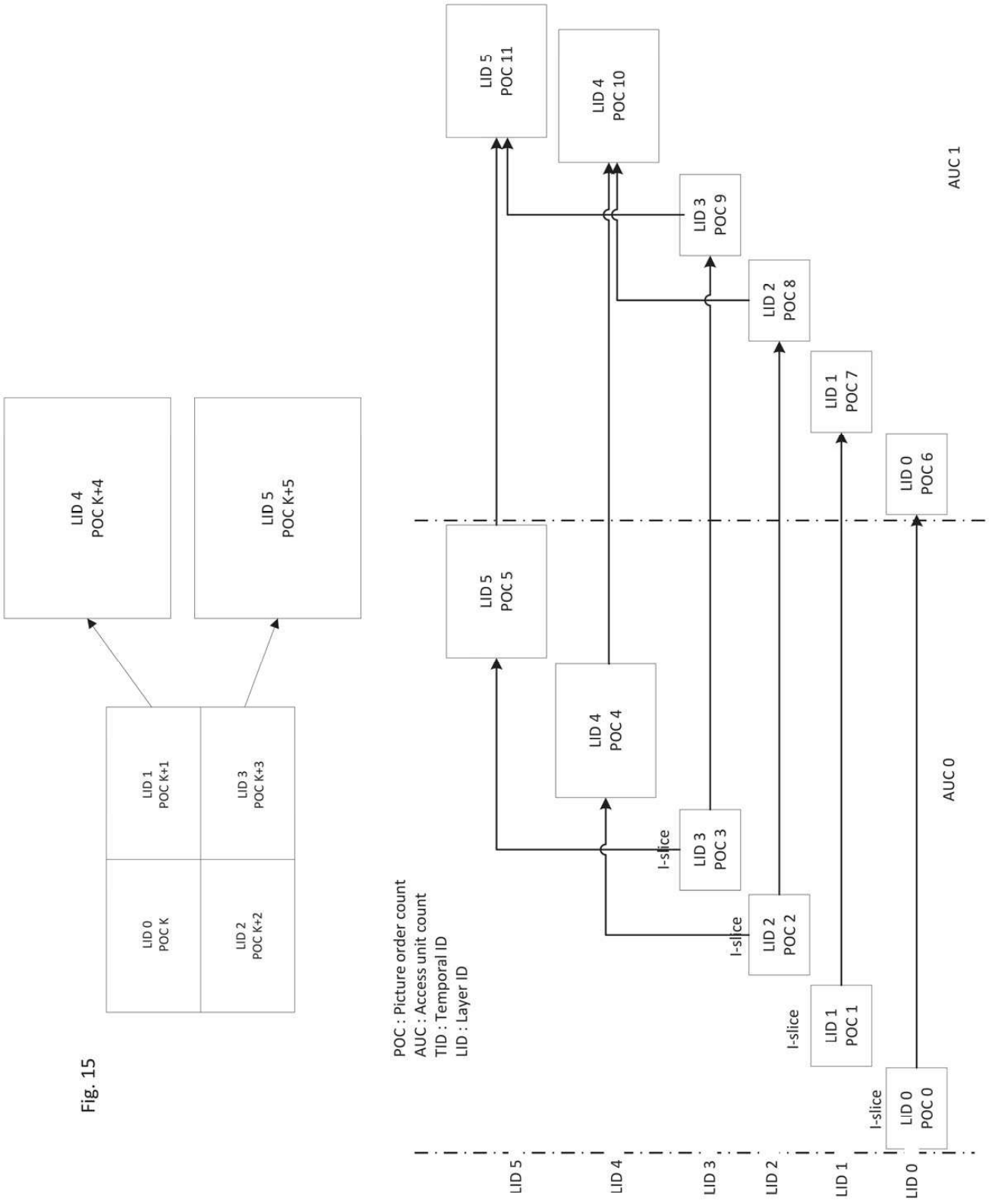

LID 0
POC K
LID 1
POC K+1
LID 2
POC K+2
LID 3
POC K+3
LID 4
POC K+4
LID 5
POC K+5
POC : Picture order count
AUC : Access unit count
TID : Temporal ID
LID : Layer ID
LID 5
LID 4
LID 3
LID 2
LID 1
LID 0
I-slice
LID 0
POC 0
LID 1
POC 1
LID 2
POC 2
LID 3
POC 3
LID 4
POC 4
LID 5
POC 5
LID 0
POC 6
LID 1
POC 7
LID 2
POC 8
LID 3
POC 9
LID 4
POC 10
LID 5
POC 11
AUC 0
AUC 1

Fig. 15

video_parameter_set_rbsp( ) {	Descriptor
vps_video_parameter_set_id	u(4)
vps_max_layers_minus1	u(8)
for(i = 0; i <= vps_max_layers_minus1; i++) {	
vps_included_layer_id[i]	u(7)
vps_reserved_zero_bit	u(1)
}	
...	
vps_sub_picture_dividing_flag	u(1)
if(vps_sub_picture_dividing_flag) {	
vps_full_pic_width_in_luma_samples	ue(v)
vps_full_pic_height_in_luma_samples	ue(v)
}	
...	
}	

FIG. 16A

▪seq_parameter_set_rbsp( ) {	Descriptor
sps_decoding_parameter_set_id	u(4)
sps_video_parameter_set_id	u(4)
sps_max_sub_layers_minus1	u(3)
...	
pic_width_in_luma_samples	ue(v)
pic_height_in_luma_samples	ue(v)
if(vps_sub_picture_dividing_flag) {	
pic_offset_x	ue(v)
pic_offset_y	ue(v)
}	
...	
}	

FIG. 16B

Fig. 17

sub_region_partitioning_info(payloadSize) {	Descriptor
num_sub_region	ue(v)
num_layers	ue(v)
for(i = 0; i <= num_layers; i++)	
layer_id [i]	ue(v)
for(i = 1; i < num_layers; i++)	
for(j = 0; j < i; j++)	
direct_dependency_flag[i][j]	u(1)
for(i = 0; i < num_sub_region; i++) {	
num_layers_for_region[i]	ue(v)
for(j = 0; j < num_layers_for_region[i]; j++)	
sub_region_layer_id[i][j]	ue(v)
sub_region_offset_x[i]	ue(v)
sub_region_offset_y[i]	ue(v)
sub_region_width[i]	ue(v)
sub_region_height [i]	ue(v)
}	
...	
}	

Fig. 18

video_parameter_set_rbsp() {	Descriptor
…	
vps_max_layers_minus1	u(6)
num_output_layer_sets	ue(v)
num_profile_tier_level	ue(v)
for(i = 0; i < num_profile_tier_level; i++)	
profile_tier_level(vps_max_sub_layers_minus1)	
for(i = 0; i < num_output_layer_sets; i++)	
for(j = 0; j < NumLayersInIdList[i]; j++) {	
output_layer_flag[i][j]	u(1)
profile_tier_level_idx[i][j]	u(v)
}	
…	
}	

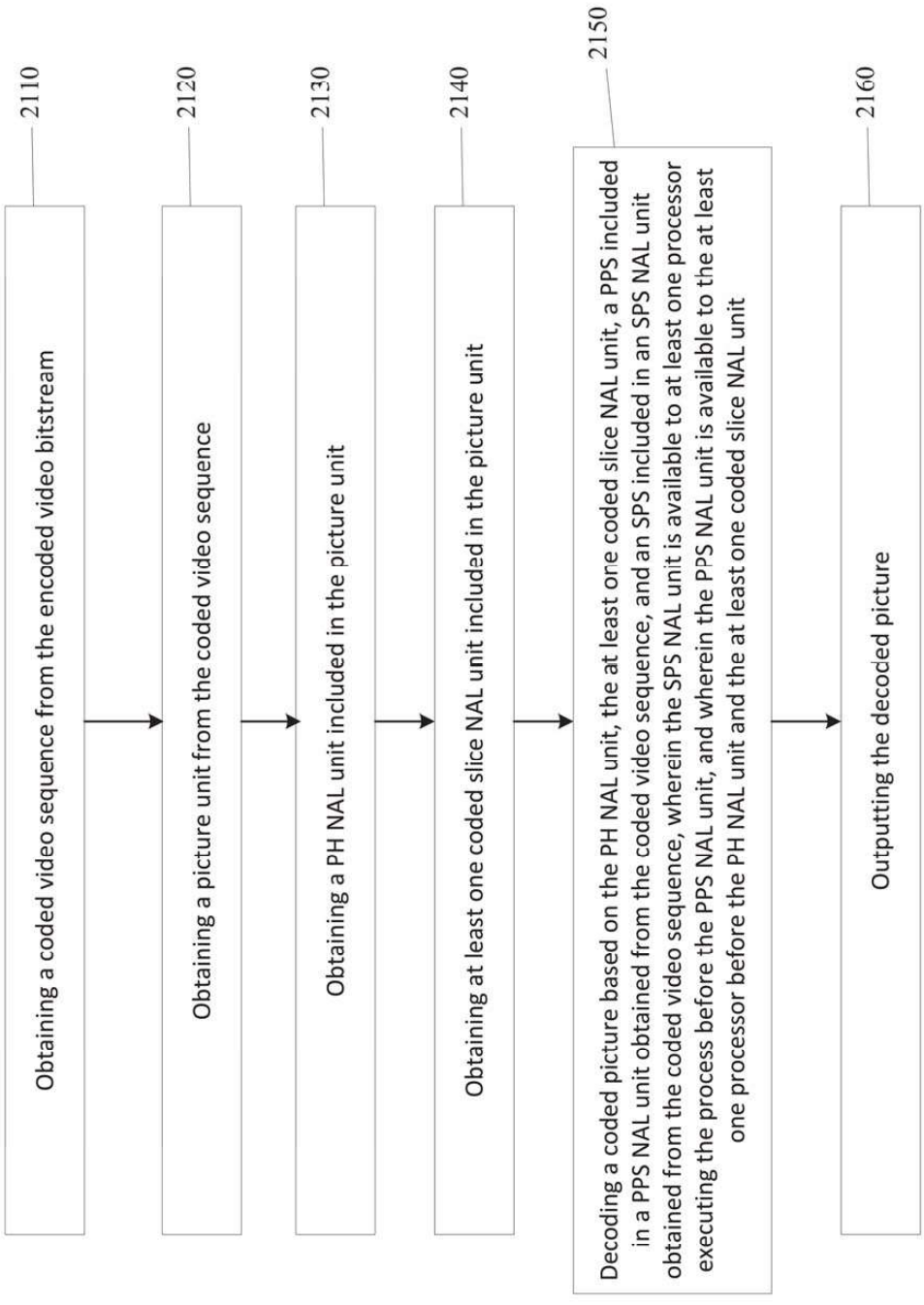
Fig. 19

video_parameter_set_rbsp() {	**Descriptor**
…	
vps_max_layers_minus1	u(6)
if(vps_max_layers_minus1 > 0) {	
num_output_layer_sets	ue(v)
num_profile_tier_level	ue(v)
}	
for(i = 0; i < num_profile_tier_level; i++)	
profile_tier_level(vps_max_sub_layers_minus1)	
for(i = 0; i < num_output_layer_sets; i++) {	
vps_output_layers_mode[i]	u(2)
vps_ptl_signal_flag[i]	u(1)
for(j = 0; j < NumLayersInIdList[I]; j++) {	
if(vps_output_layers_mode[i] = = 2)	
output_layer_flag[I][j]	u(1)
if(vps_ptl_signal_flag[i])	
profile_tier_level_idx[I][j]	u(v)
}	
}	
…	
}	

Fig. 20

video_parameter_set_rbsp() {	Descriptor
…	
vps_max_layers_minus1	u(6)
if(vps_max_layers_minus1 > 0) {	
num_output_layer_sets	ue(v)
num_profile_tier_level	ue(v)
}	
max_subpics_minus1	u(8)
for(i = 0; i < **max_subpics_minus1**; i++) {	
sub_pic_id[i]	u(8)
}	
for(i = 0; i < num_profile_tier_level; i++)	
profile_tier_level(vps_max_sub_layers_minus1)	
for(i = 0; i < num_output_layer_sets; i++) {	
vps_output_layers_mode[i]	u(2)
vps_ptl_signal_flag[i]	u(1)
for(j = 0; j < NumLayersInIdList[i]; j++) {	
num_output_subpic_layer[i][j]	ue(v)
for(k = 0; k < num_output_subpic_layer[i][j]; k++)	
sub_pic_id_layer[i][j] [k]	u(8)
if(vps_output_layers_mode[i] == 2)	
output_layer_flag[i][j]	u(1)
if(vps_ptl_signal_flag[i])	
profile_tier_level_idx[i][j]	u(v)
}	
}	
…	
}	

FIG. 21


2100
2110
Obtaining a coded video sequence from the encoded video bitstream
2120
Obtaining a picture unit from the coded video sequence
2130
Obtaining a PH NAL unit included in the picture unit
2140
Obtaining at least one coded slice NAL unit included in the picture unit
2150
Decoding a coded picture based on the PH NAL unit, the at least one coded slice NAL unit, a PPS included in a PPS NAL unit obtained from the coded video sequence, and an SPS included in an SPS NAL unit obtained from the coded video sequence, wherein the SPS NAL unit is available to at least one processor executing the process before the PPS NAL unit, and wherein the PPS NAL unit is available to the at least one processor before the PH NAL unit and the at least one coded slice NAL unit
2160
Outputting the decoded picture

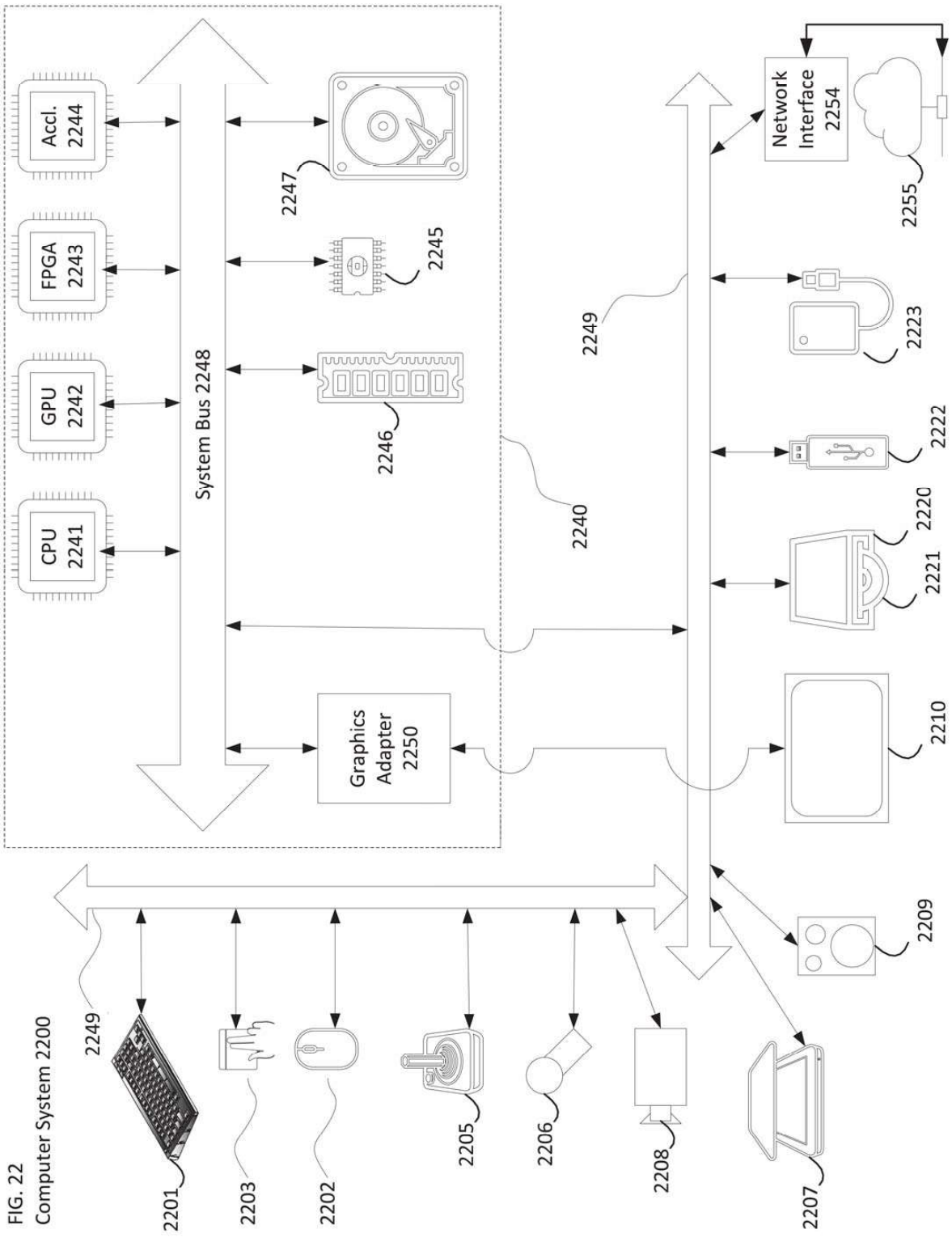


FIG. 22
Computer System 2200