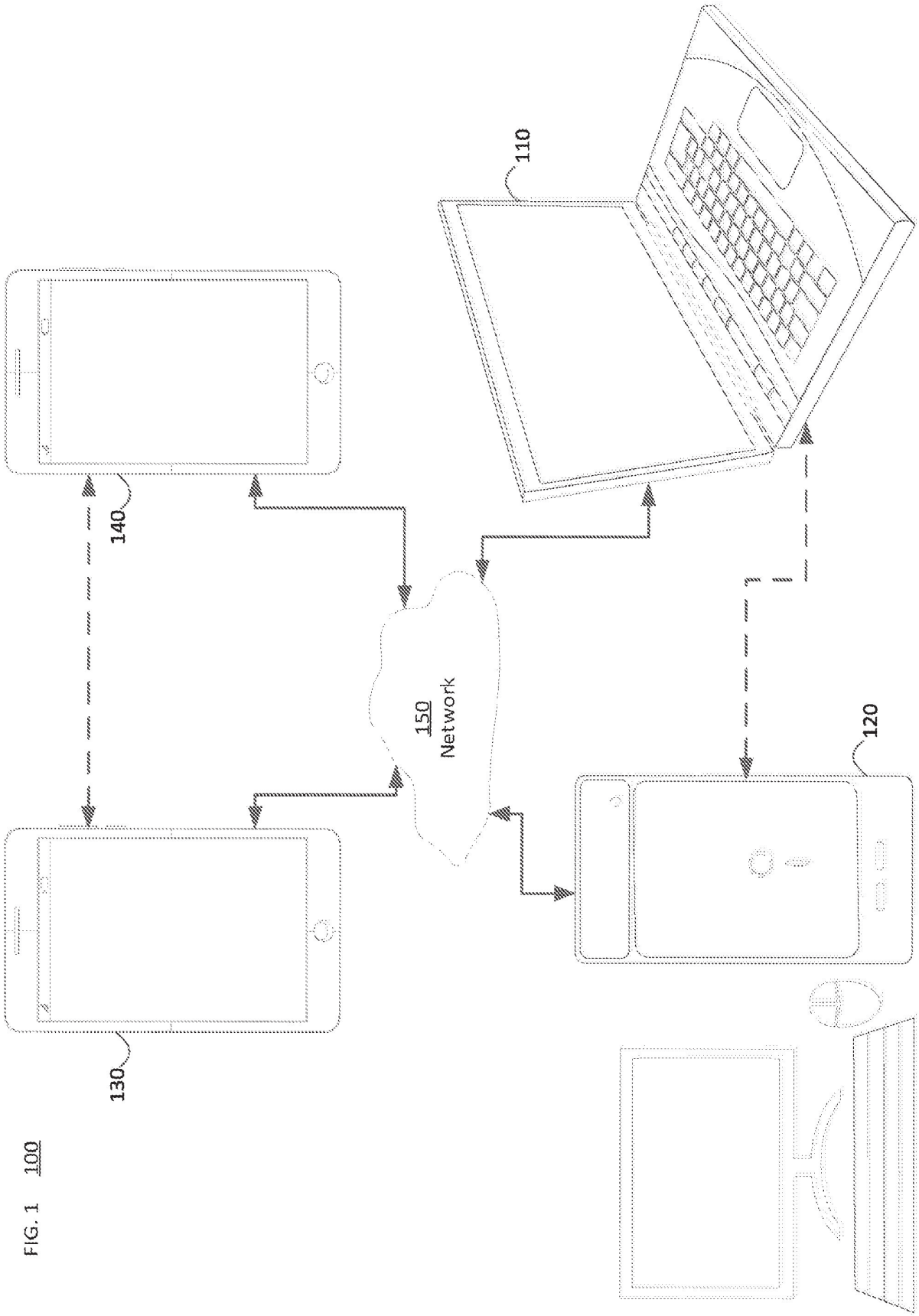


FIG. 1 100

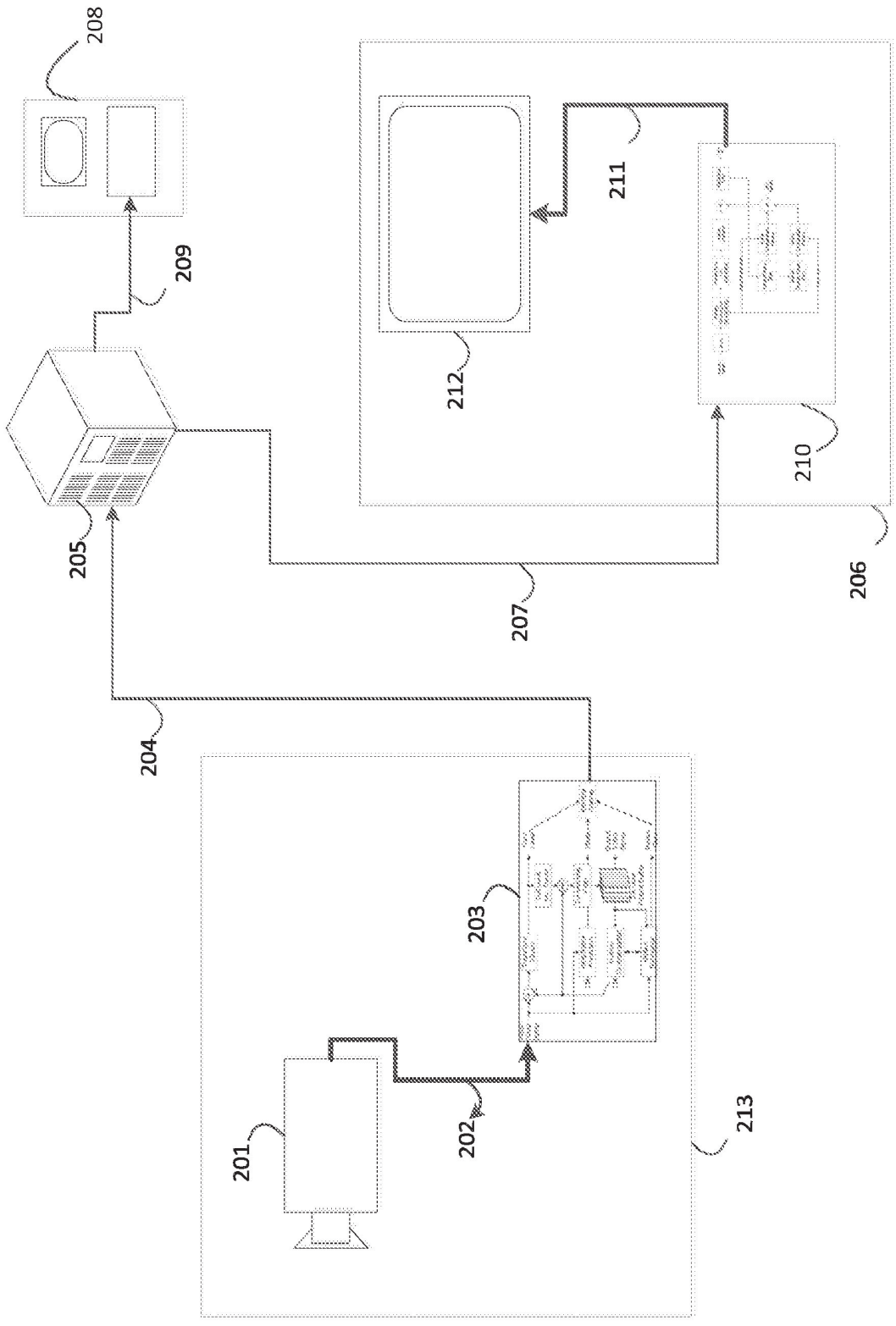


FIG. 2 Streaming System 200

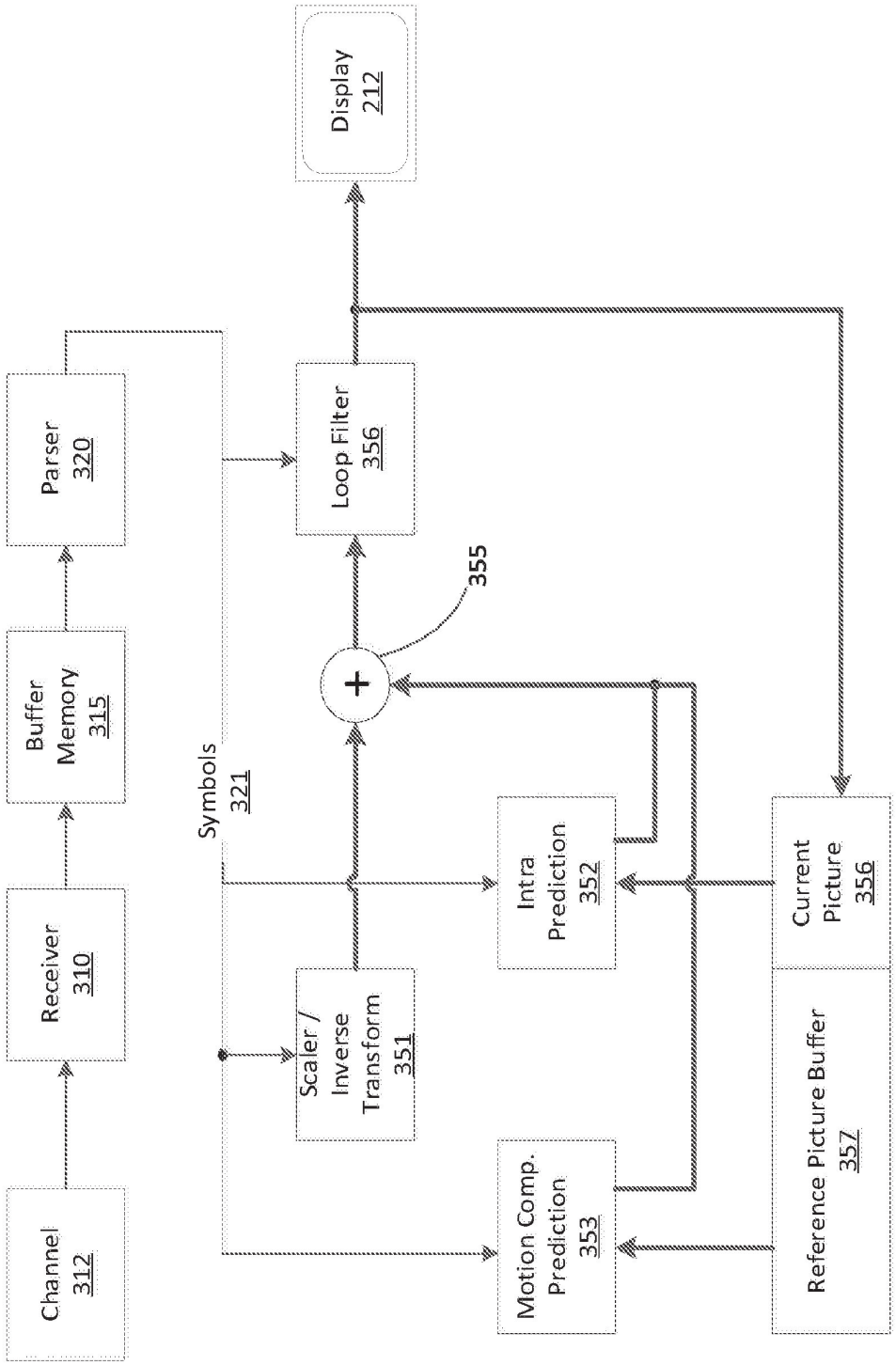


FIG. 3 Decoder 210

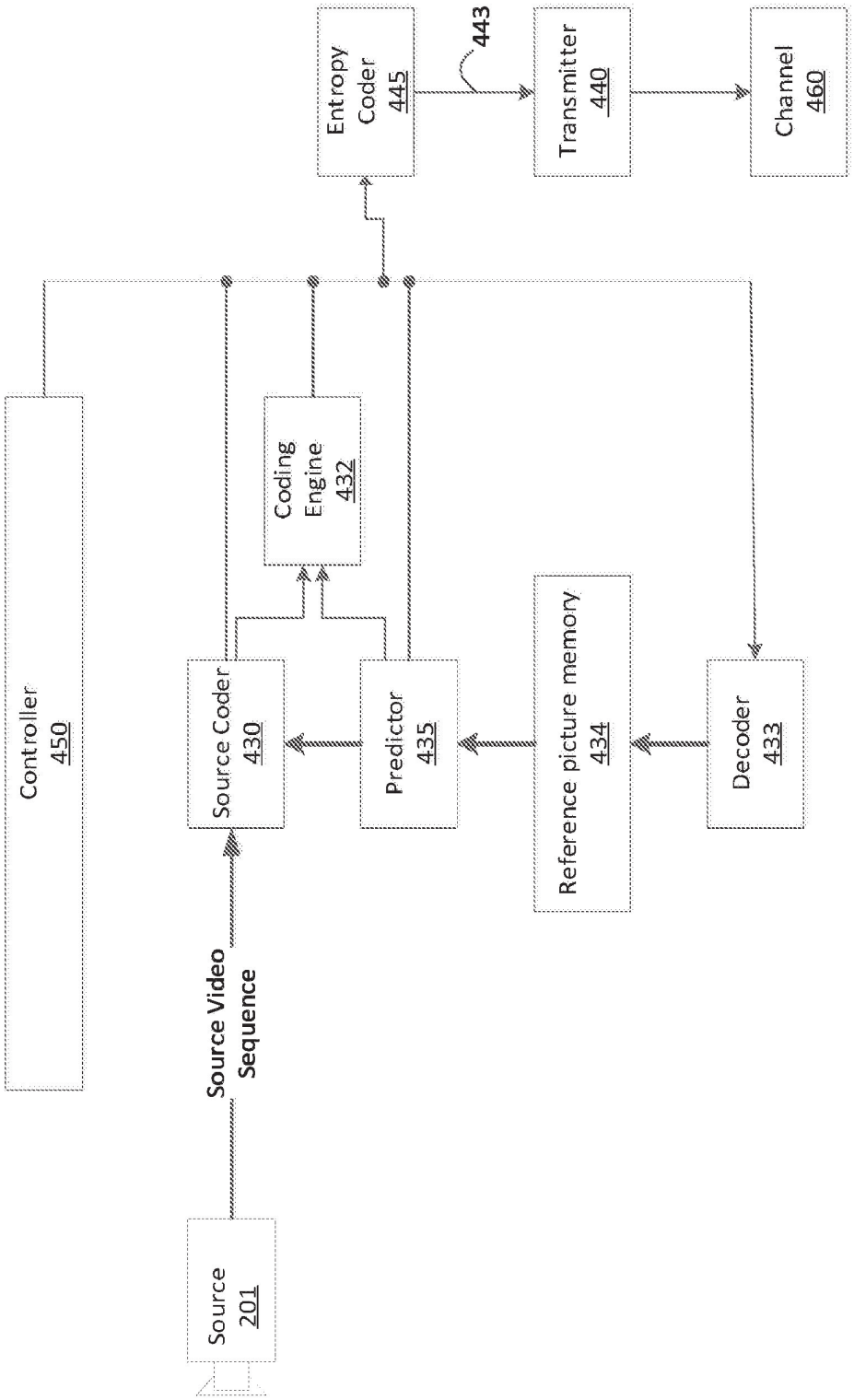


FIG. 4 Encoder 203

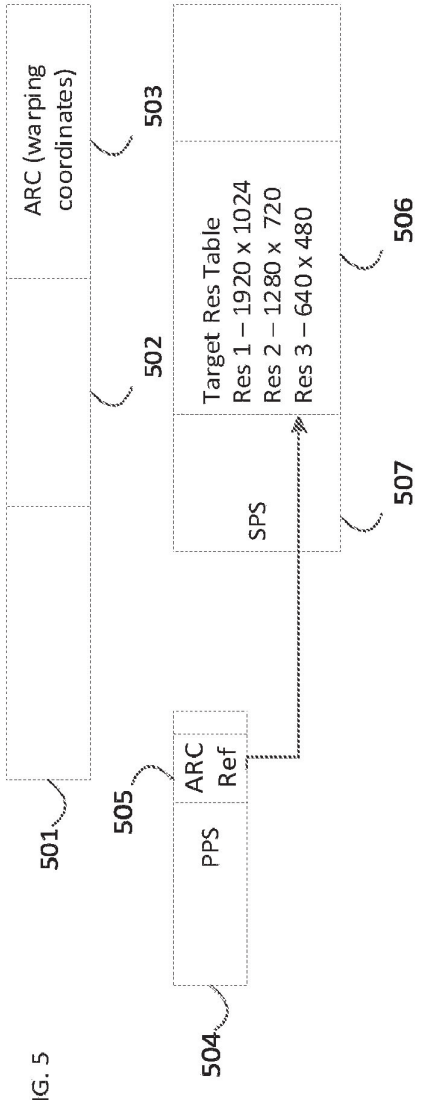

ARC (warping coordinates)
503
502
Target Res Table
Res 1 – 1920 x 1024
Res 2 – 1280 x 720
Res 3 – 640 x 480
506
SPS
507
501
505
ARC Ref
PPS
504

FIG. 5

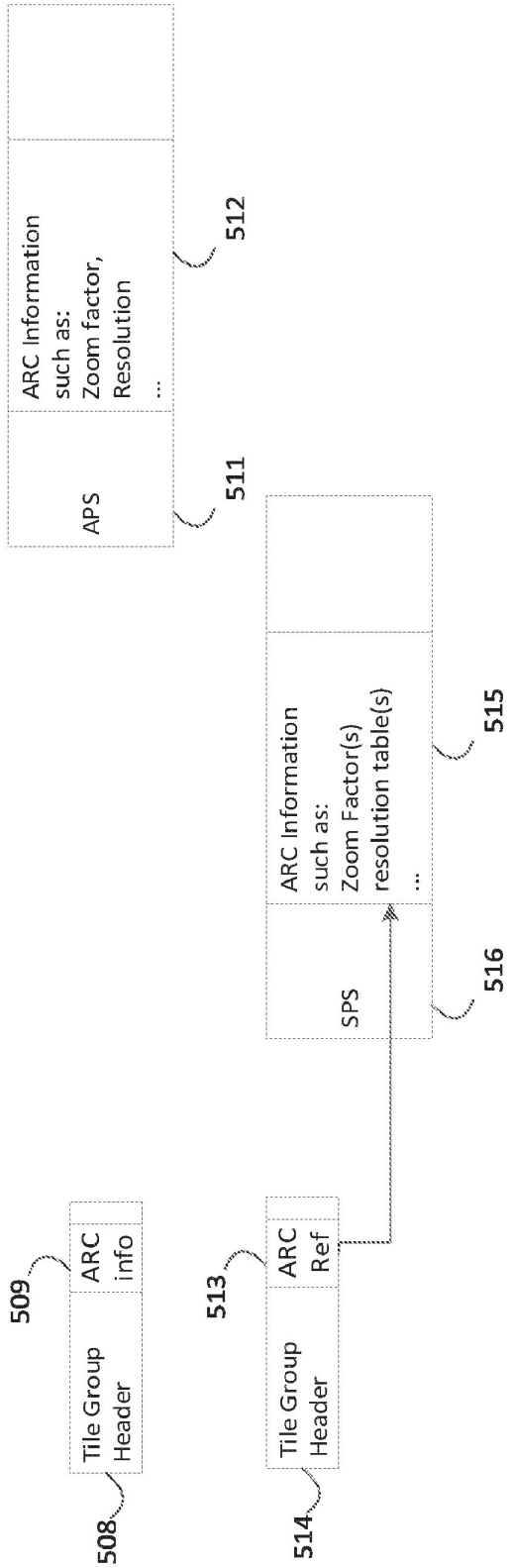

ARC Information such as:
Zoom factor,
Resolution
...
512
APS
511
509
ARC info
Tile Group Header
508
513
ARC Ref
Tile Group Header
514
ARC Information such as:
Zoom Factor(s)
resolution table(s)
...
515
SPS
516

Fig. 6

601 603 602

tile_group_header( ) {	
...	
if(adaptive_pic_resolution_change_flag) {	
dec_pic_size_idx	u(1)
}	
...	
}	

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)
}	
}	
...	
}	

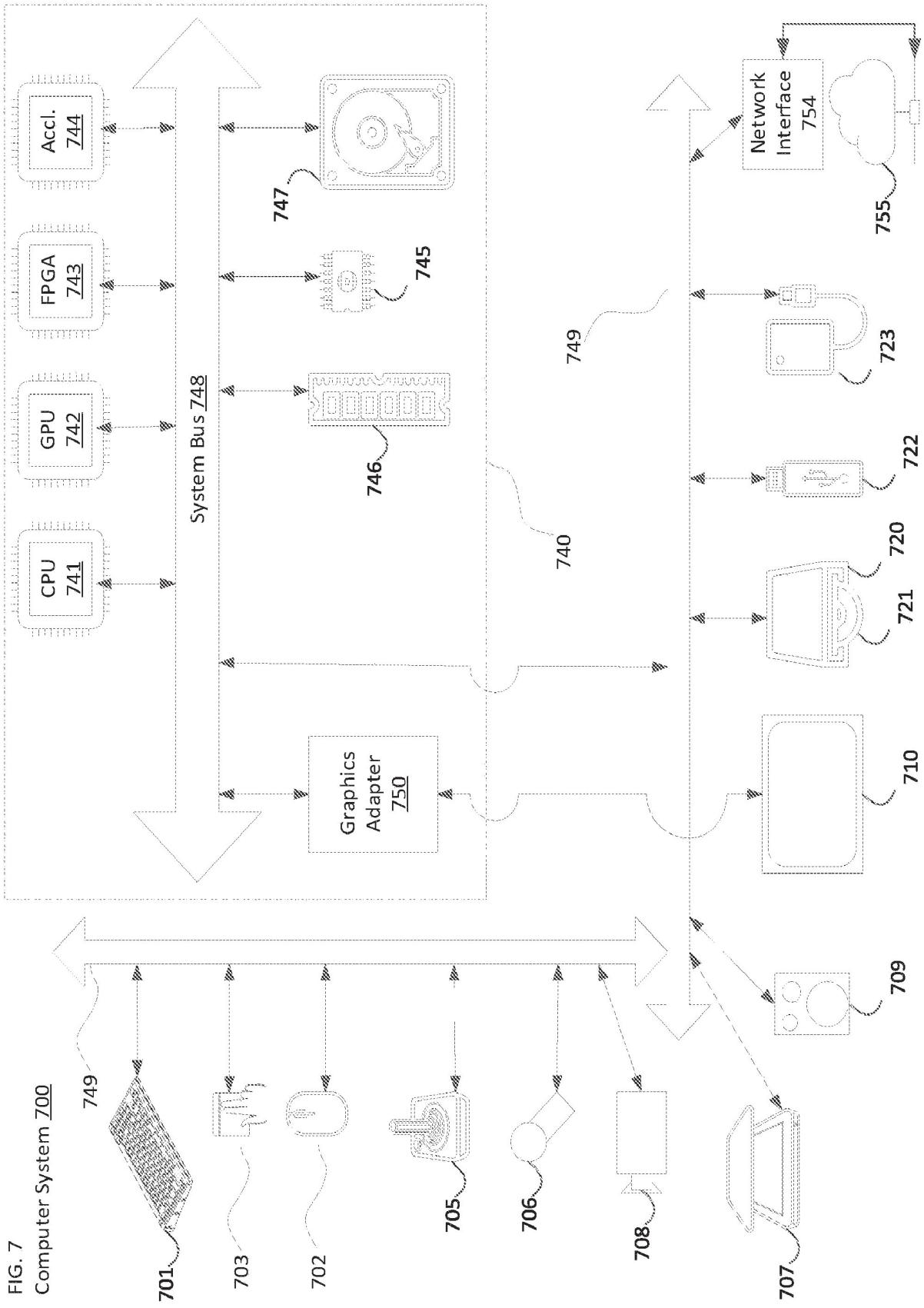

Computer System 700
CPU 741
GPU 742
FPGA 743
Accl. 744
System Bus 748
Graphics Adapter 750
740
746
745
747
749
Network Interface 754
755
723
722
720
721
710
709
701
703
702
705
706
708
707

FIG. 7

Fig. 8

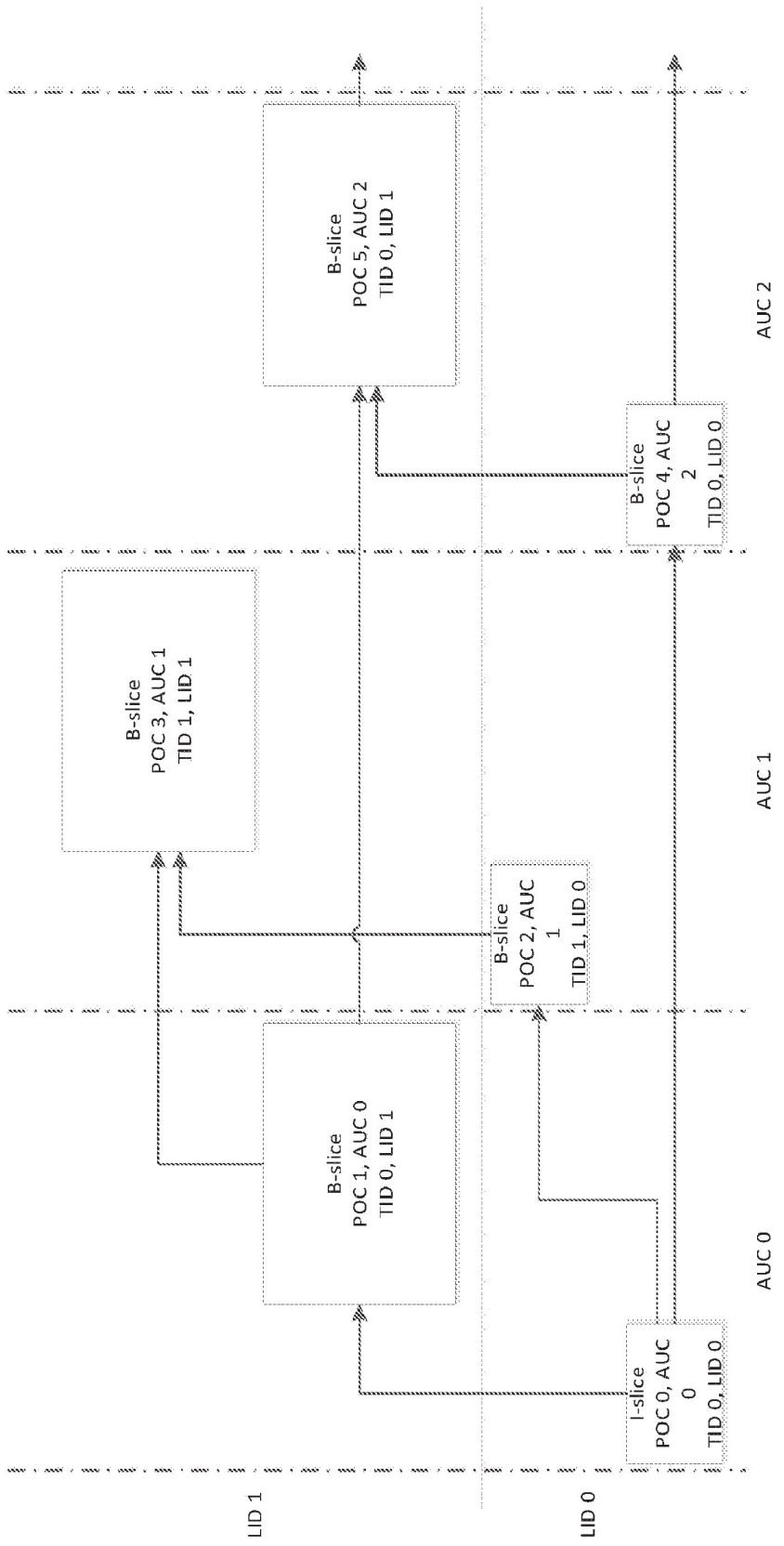

B-slice
POC 5, AUC 2
TID 0, LID 1
B-slice
POC 4, AUC 2
TID 0, LID 0
B-slice
POC 3, AUC 1
TID 1, LID 1
B-slice
POC 2, AUC 1
TID 1, LID 0
B-slice
POC 1, AUC 0
TID 0, LID 1
I-slice
POC 0, AUC 0
TID 0, LID 0
AUC 2
AUC 1
AUC 0
LID 1
LID 0
POC : Picture order count
AUC : Access unit count
TID : Temporal ID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

Fig. 9

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)
}	

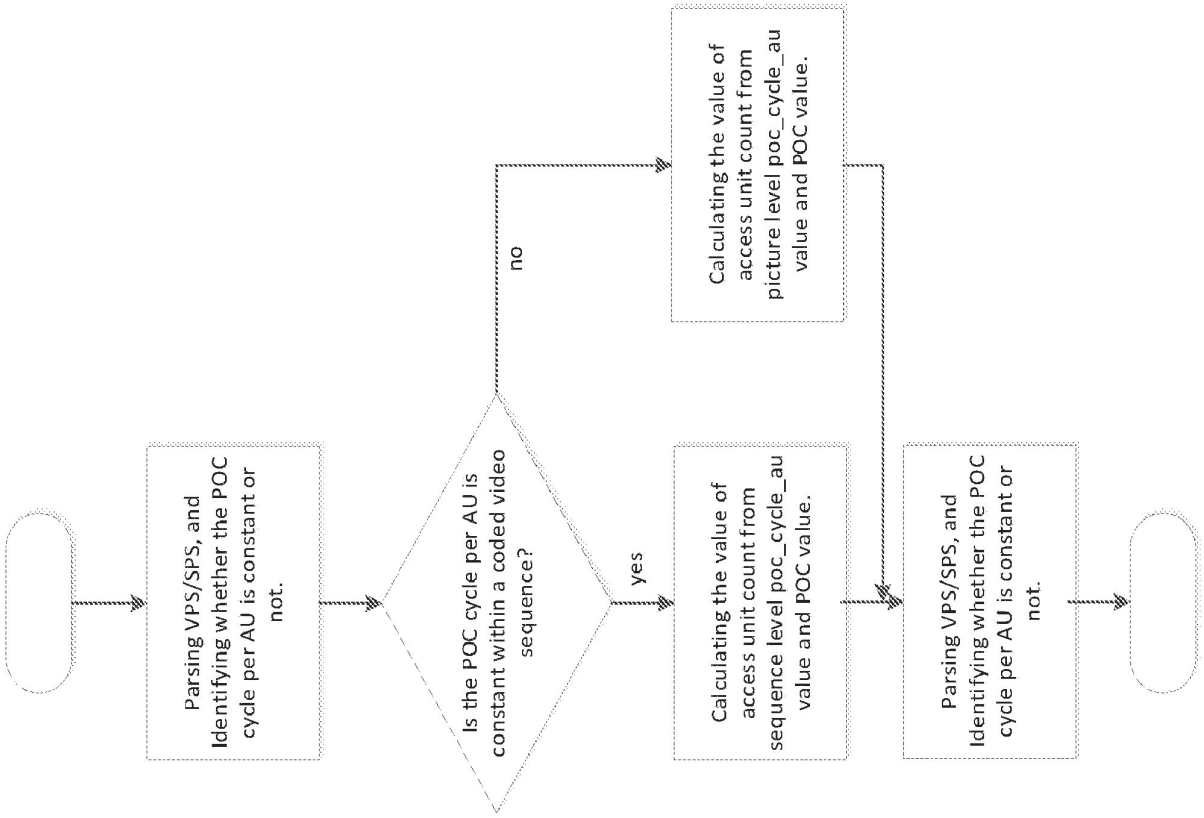

Parsing VPS/SPS, and Identifying whether the POC cycle per AU is constant or not.
Is the POC cycle per AU is constant within a coded video sequence?
yes
no
Calculating the value of access unit count from sequence level poc_cycle_au value and POC value.
Calculating the value of access unit count from picture level poc_cycle_au value and POC value.
Parsing VPS/SPS, and Identifying whether the POC cycle per AU is constant or not.

Fig. 10

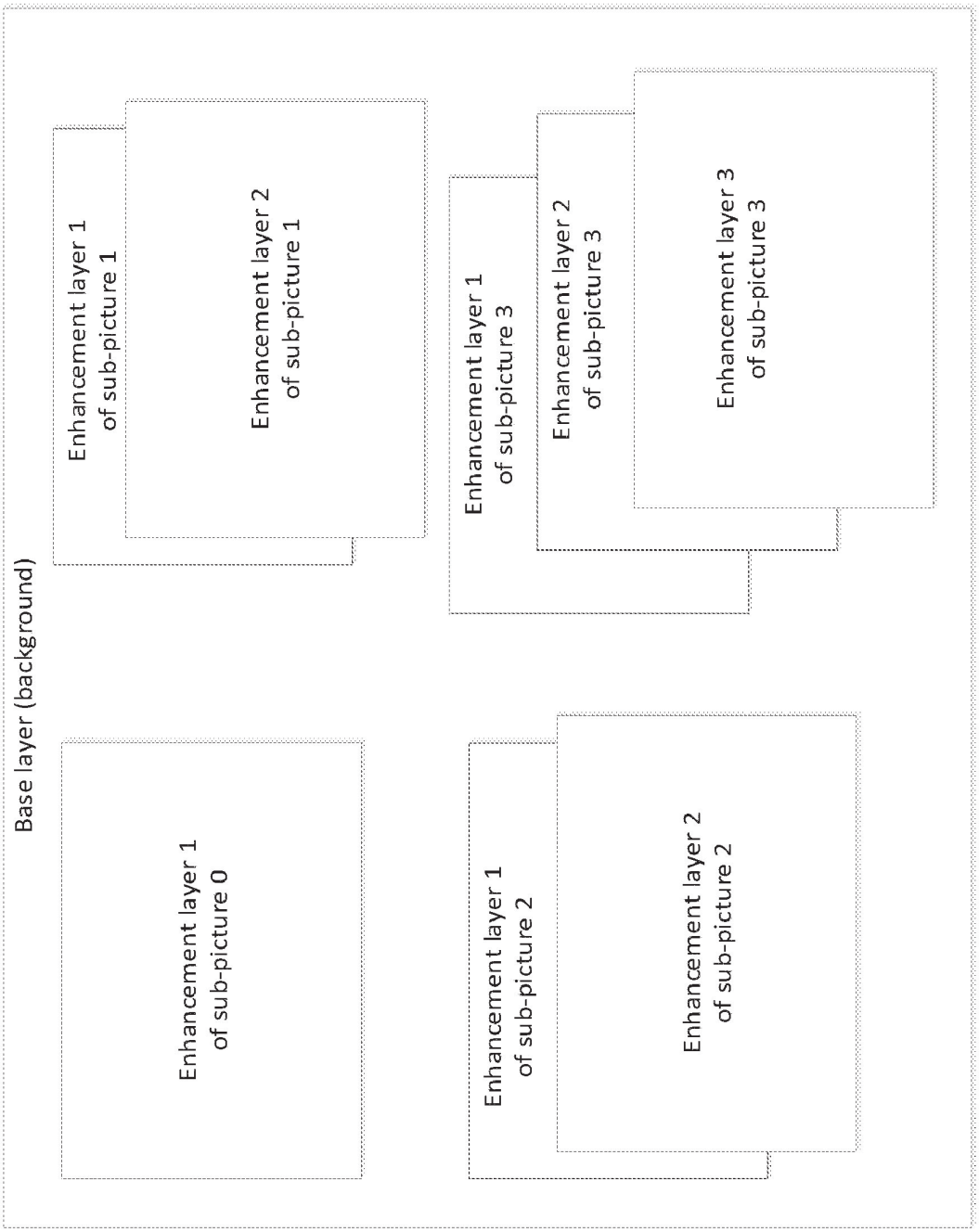

Base layer (background)
Enhancement layer 1 of sub-picture 0
Enhancement layer 1 of sub-picture 1
Enhancement layer 2 of sub-picture 1
Enhancement layer 1 of sub-picture 2
Enhancement layer 2 of sub-picture 2
Enhancement layer 1 of sub-picture 3
Enhancement layer 2 of sub-picture 3
Enhancement layer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

FIG. 11

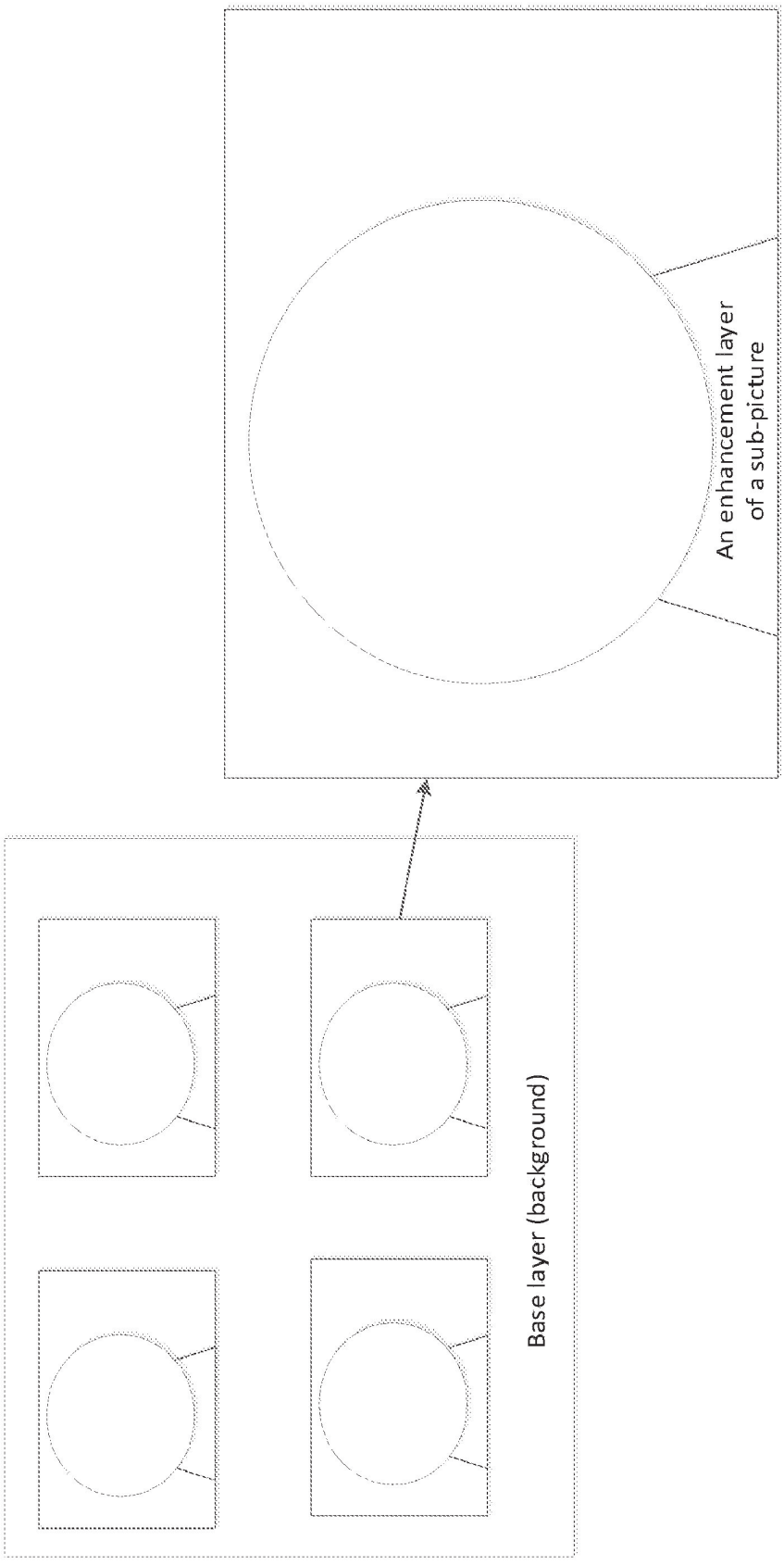

An enhancement layer
of a sub-picture
Base layer (background)

FIG. 12

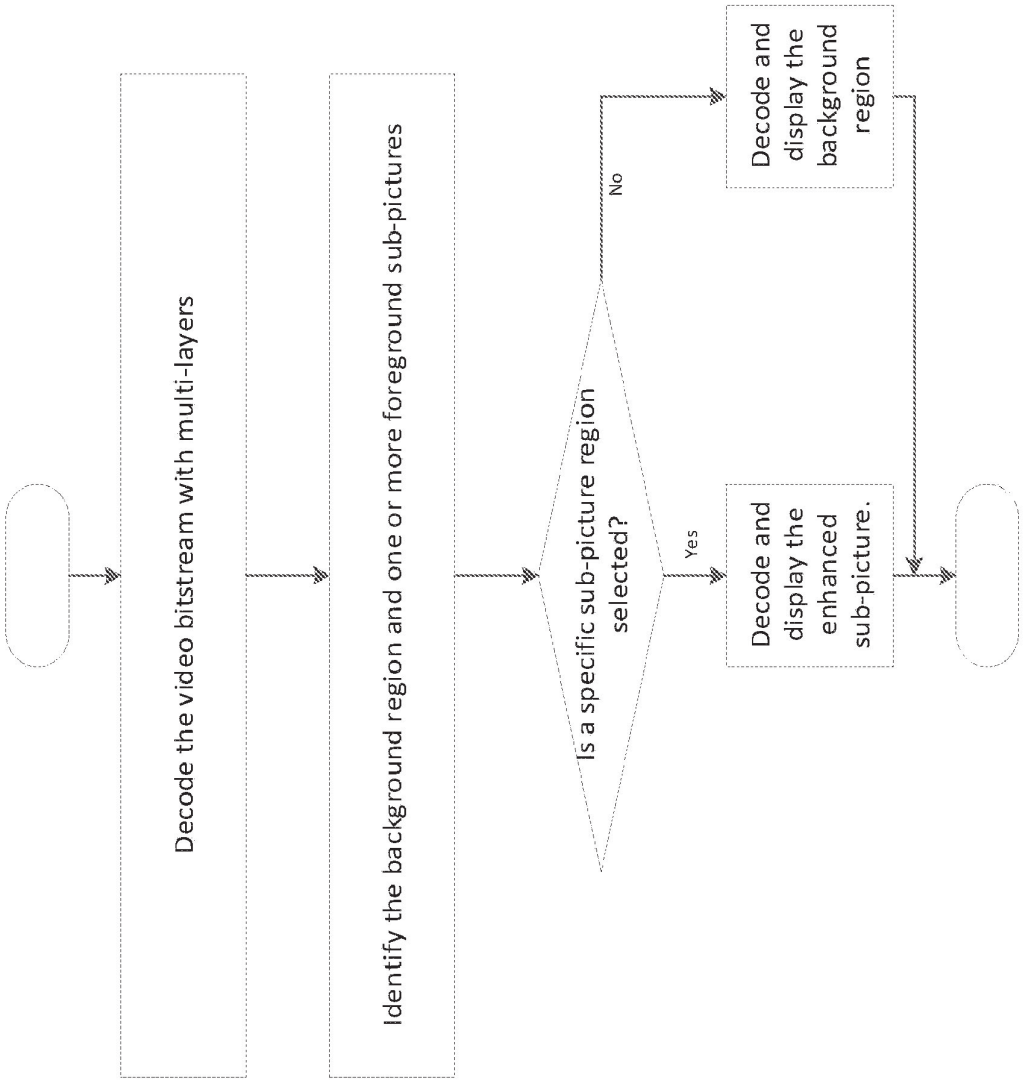

Decode the video bitstream with multi-layers
Identify the background region and one or more foreground sub-pictures
Is a specific sub-picture region selected?
Yes
No
Decode and display the enhanced sub-picture.
Decode and display the background region

Fig. 13

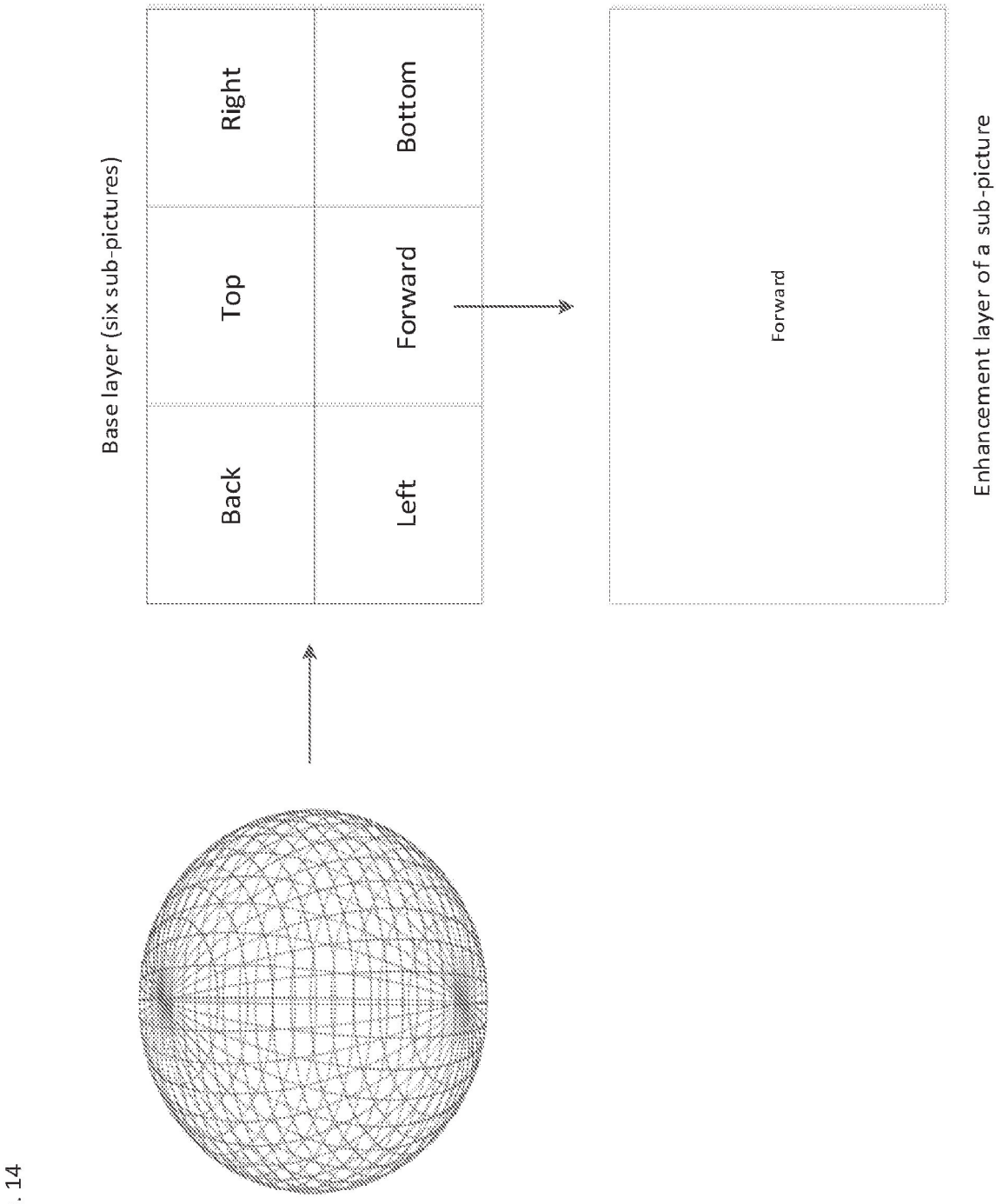
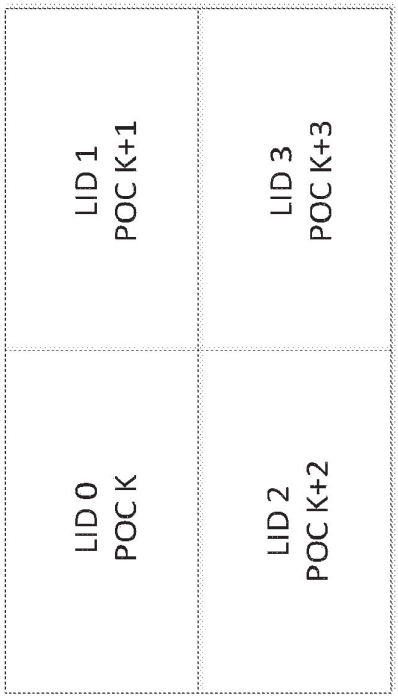
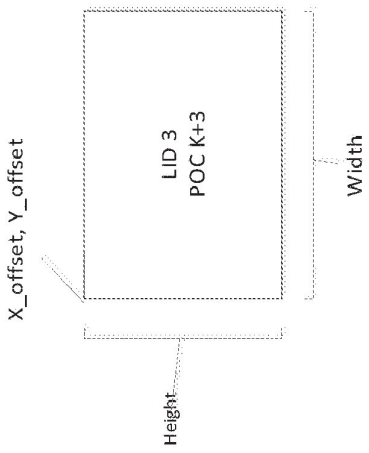
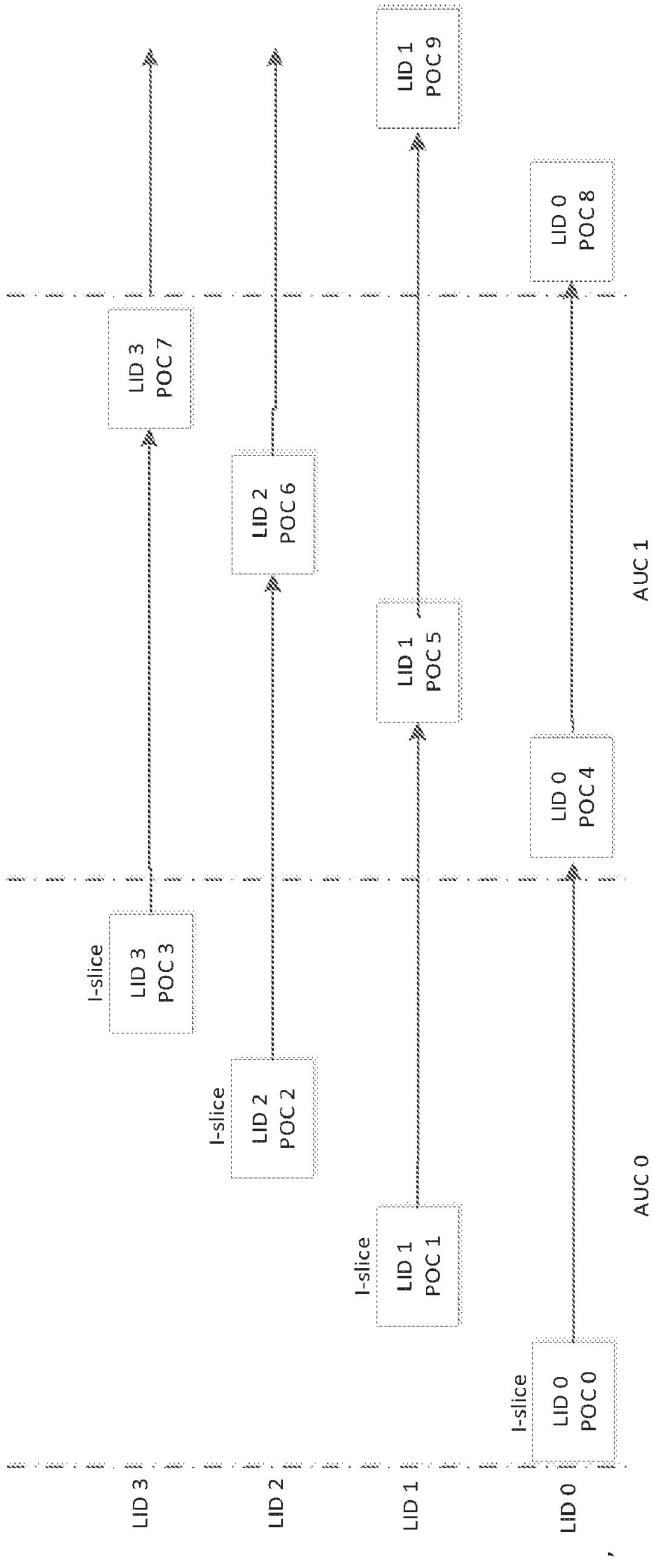

Base layer (six sub-pictures)
Back
Top
Right
Left
Forward
Bottom
Forward
Enhancement layer of a sub-picture

Fig. 14

Fig. 15


LID 0
POC K
LID 1
POC K+1
LID 2
POC K+2
LID 3
POC K+3


X_offset, Y_offset
LID 3
POC K+3
Height
Width


LID 3
LID 2
LID 1
LID 0
I-slice
LID 3
POC 3
I-slice
LID 2
POC 2
I-slice
LID 1
POC 1
I-slice
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

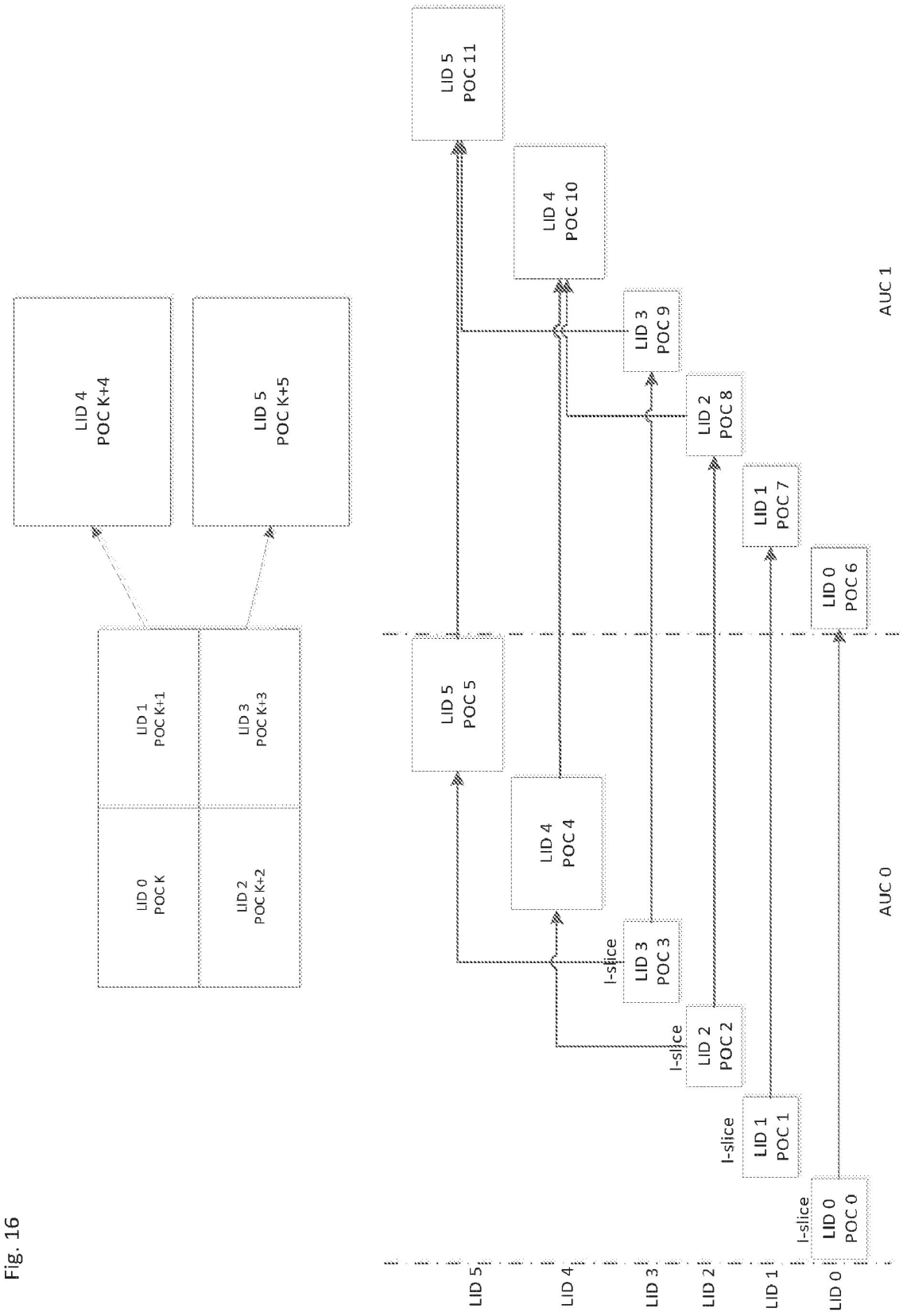

LID 4
POC K+4
LID 5
POC K+5
LID 1
POC K+1
LID 3
POC K+3
LID 0
POC K
LID 2
POC K+2
LID 5
POC 11
LID 4
POC 10
LID 3
POC 9
LID 2
POC 8
LID 1
POC 7
LID 0
POC 6
LID 5
POC 5
LID 4
POC 4
I-slice
LID 3
POC 3
I-slice
LID 2
POC 2
I-slice
LID 1
POC 1
I-slice
LID 0
POC 0
AUC 1
AUC 0
LID 5
LID 4
LID 3
LID 2
LID 1
LID 0

Fig. 16

Fig. 17

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)
}	
...	
}	

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. 18

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. 19

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. 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)
}	
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. 21

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. 22

video_parameter_set_rbsp( ) {	Descriptor
vps_video_parameter_set_id	u(4)
vps_max_layers_minus1	u(6)
vps_max_sublayers_minus1	u(3)
if(vps_max_layers_minus1 > 0 && vps_max_sublayers_minus1 > 0)	
vps_all_layers_same_num_sublayers_flag	u(1)
if(vps_max_layers_minus1 > 0)	
vps_all_independent_layers_flag	u(1)
for(i = 0; i <= vps_max_layers_minus1; i++) {	
vps_layer_id[i]	u(6)
if(i > 0 && !vps_all_independent_layers_flag) {	
vps_independent_layer_flag[i]	u(1)
if(!vps_independent_layer_flag[i]) {	
for(j = 0; j < i; j++)	
vps_direct_ref_layer_flag[i][j]	u(1)
max_tid_ref_present_flag[i]	u(1)
if(max_tid_ref_present_flag[i])	
max_tid_il_ref_pics_plus1[i]	u(3)
}	
}	
}	
if(vps_max_layers_minus1 > 0) {	
if(vps_all_independent_layers_flag)	
each_layer_is_an_ols_flag	u(1)
if(!each_layer_is_an_ols_flag) {	
if(!vps_all_independent_layers_flag)	
ols_mode_idc	u(2)
if(ols_mode_idc == 2) {	
num_output_layer_sets_minus1	u(8)
for(i = 1; i <= num_output_layer_sets_minus1; i++)	
for(j = 0; j <= vps_max_layers_minus1; j++)	
ols_output_layer_flag[i][j]	u(1)
}	
}	
}	
...	
}	

Fig. 23

video_parameter_set_rbsp( ) {	Descriptor
vps_video_parameter_set_id	u(4)
vps_max_layers_minus1	u(6)
vps_max_sublayers_minus1	u(3)
if(vps_max_layers_minus1 > 0 && vps_max_sublayers_minus1 > 0)	
vps_all_layers_same_num_sublayers_flag	u(1)
if(vps_max_layers_minus1 > 0)	
vps_all_independent_layers_flag	u(1)
for(i = 0; i <= vps_max_layers_minus1; i++) {	
vps_layer_id[i]	u(6)
if(i > 0 && !vps_all_independent_layers_flag) {	
vps_independent_layer_flag[i]	u(1)
if(!vps_independent_layer_flag[i]) {	
for(j = 0; j < i; j++)	
vps_direct_ref_layer_flag[i][j]	u(1)
max_tid_ref_present_flag[i]	u(1)
if(max_tid_ref_present_flag[i])	
max_tid_il_ref_pics_plus1[i]	u(3)
}	
}	
}	
if(vps_max_layers_minus1 > 0) {	
if(vps_all_independent_layers_flag)	
each_layer_is_an_ols_flag	u(1)
if(!each_layer_is_an_ols_flag) {	
if(!vps_all_independent_layers_flag)	
ols_mode_idc	u(2)
if(ols_mode_idc == 2 && vps_max_layers_minus1 > 1) {	
num_output_layer_sets_minus1	u(8)
for(i = 1; i <= num_output_layer_sets_minus1; i ++)	
for(j = 0; j <= vps_max_layers_minus1; j++)	
ols_output_layer_flag[i][j]	u(1)
}	
}	
}	
...	
}	

Fig. 24

picture_header_structure( ) {	**Descriptor**
gdr_or_irap_pic_flag	u(1)
if(gdr_or_irap_pic_flag)	
gdr_pic_flag	u(1)
ph_inter_slice_allowed_flag	u(1)
if(ph_inter_slice_allowed_flag)	
ph_intra_slice_allowed_flag	u(1)
non_reference_picture_flag	u(1)
…	
if(output_flag_present_flag && !non_reference_picture_flag)	
pic_output_flag	u(1)
if(rpl_info_in_ph_flag)	
ref_pic_lists( )	
…	
}	