

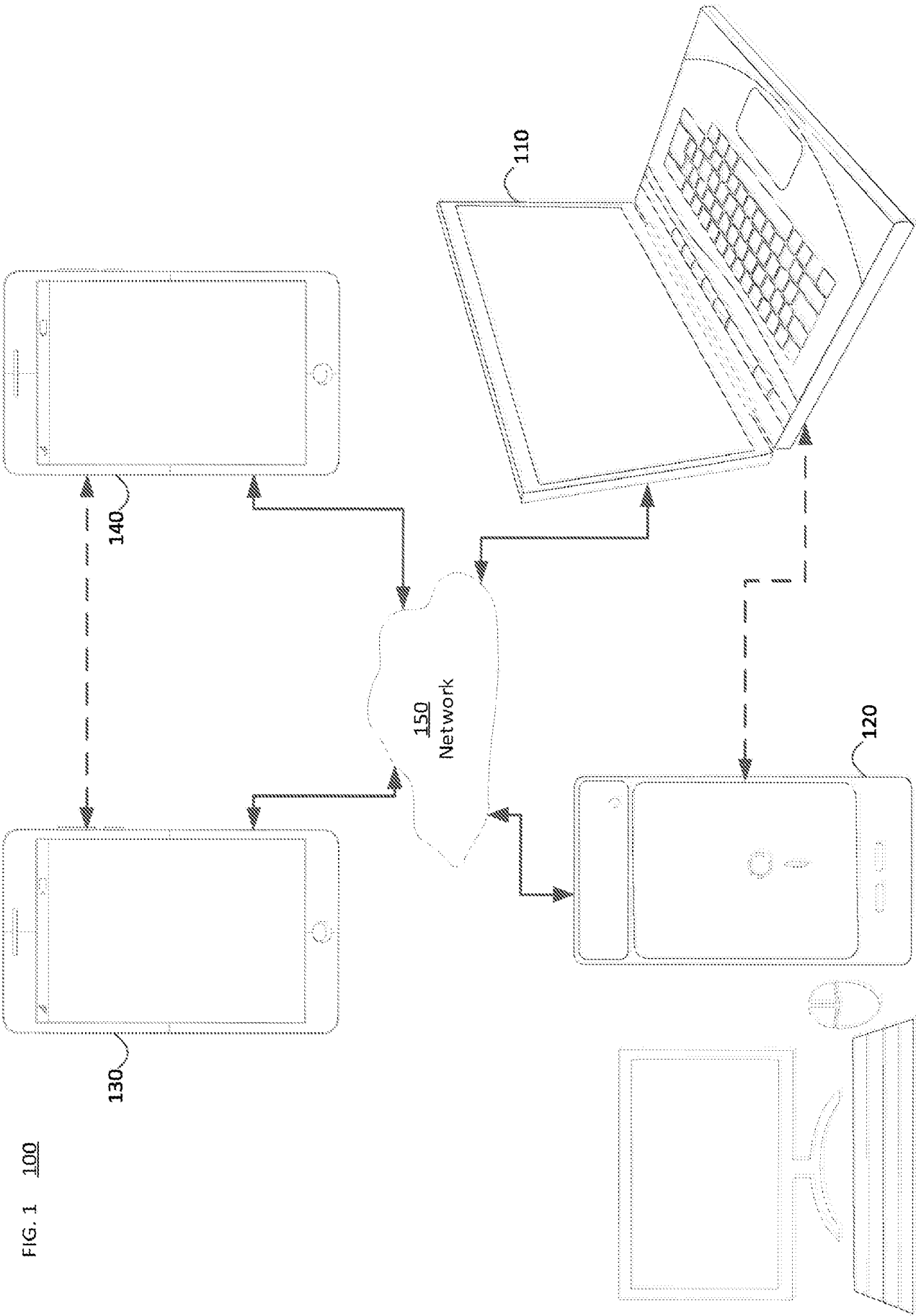


FIG. 2 Streaming System 200

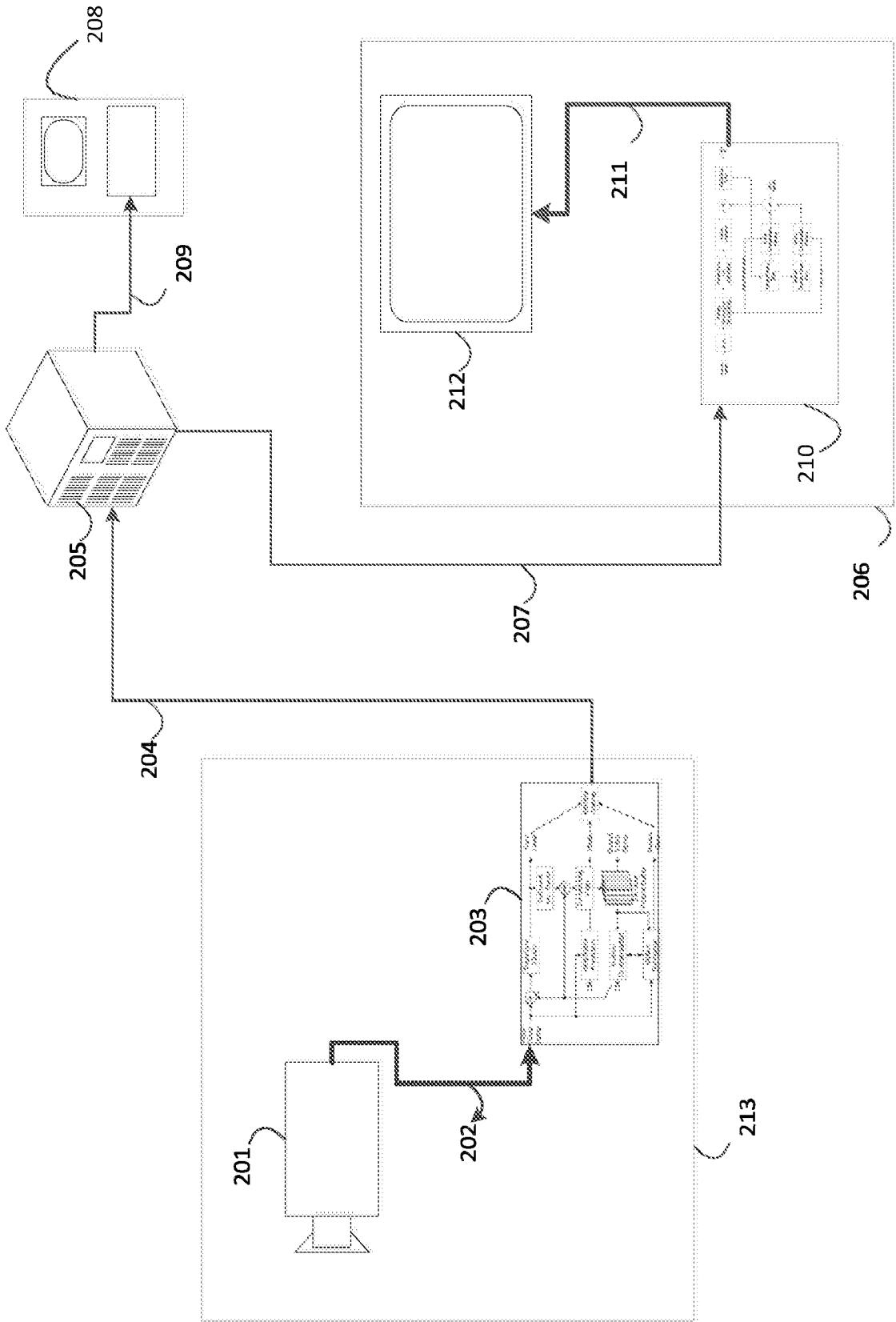


FIG. 3 Decoder 210

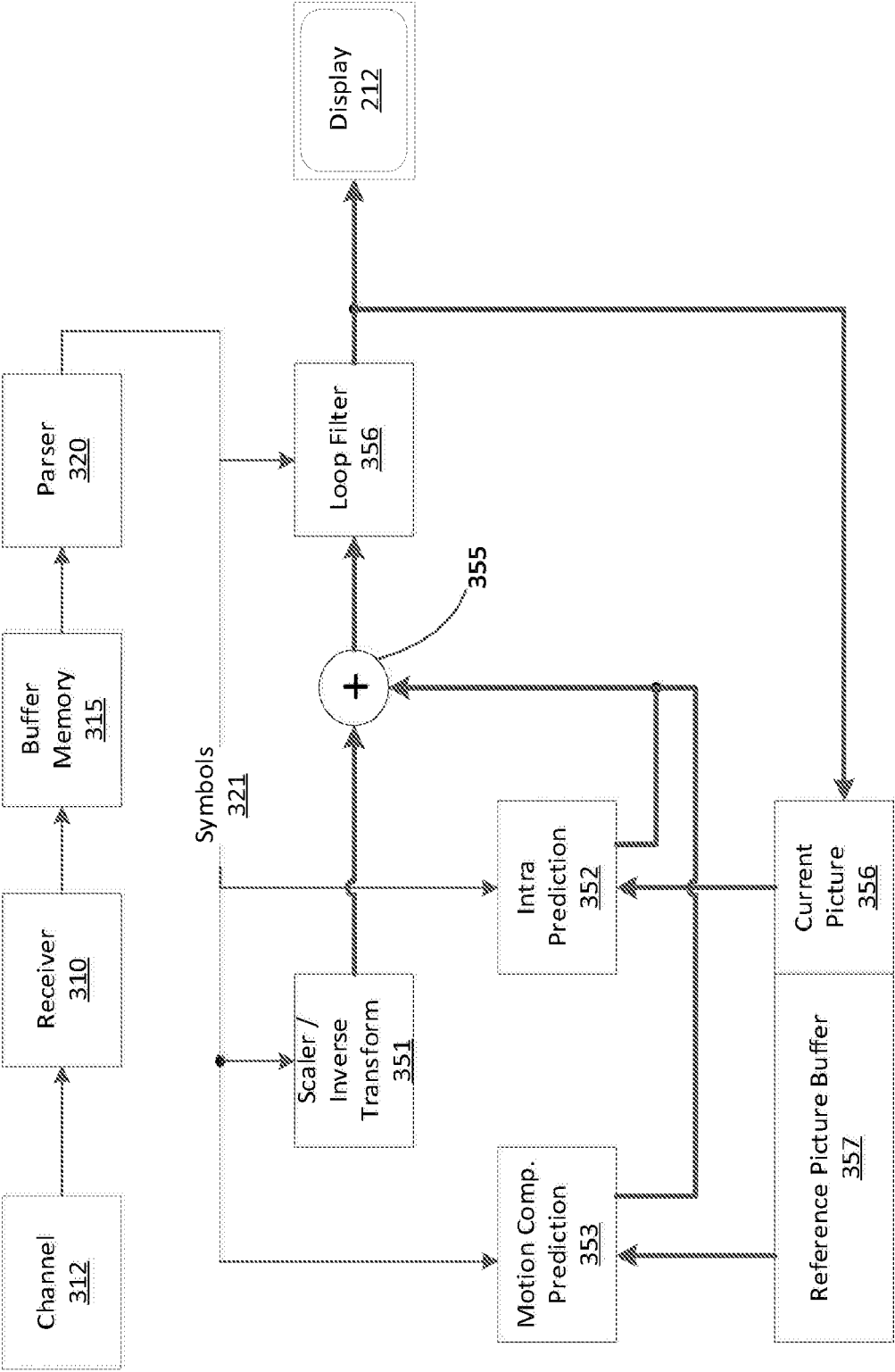
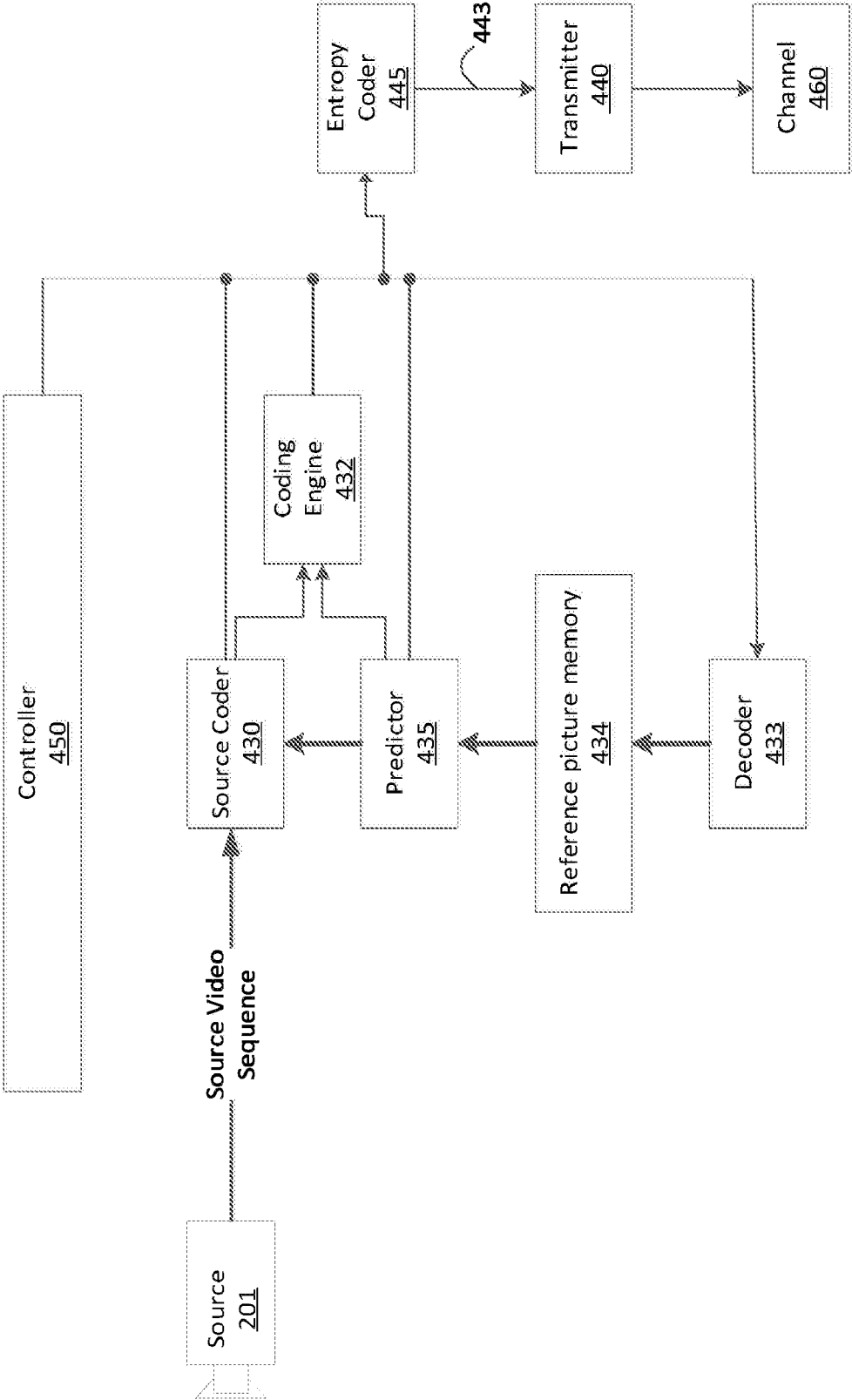


FIG. 4 Encoder 203



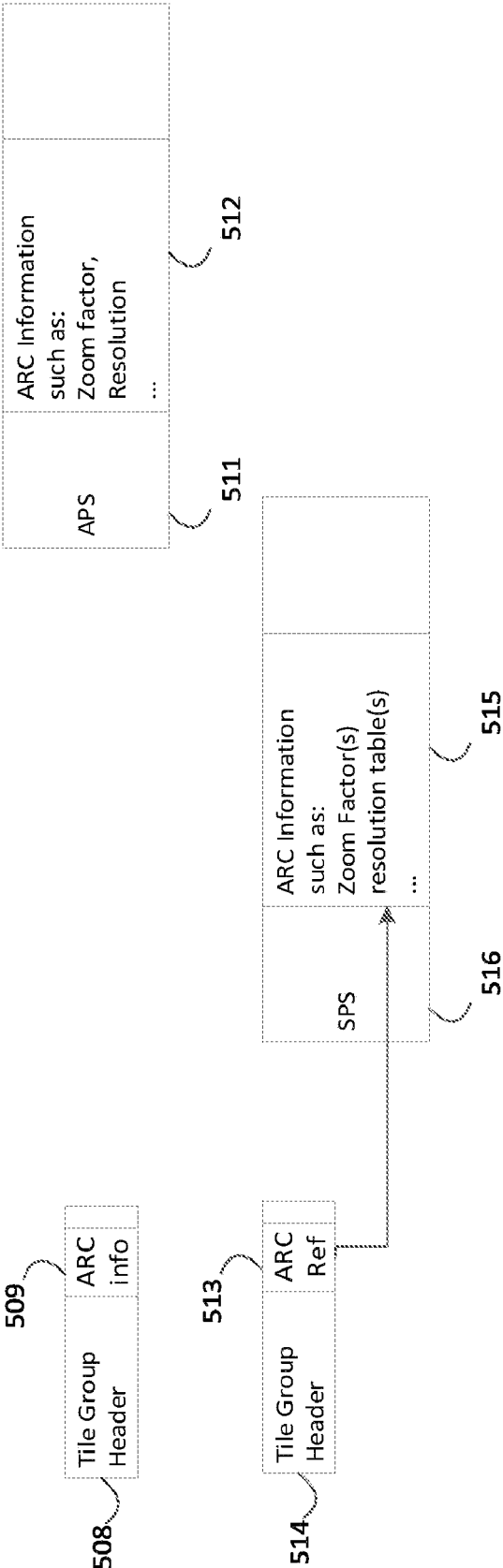
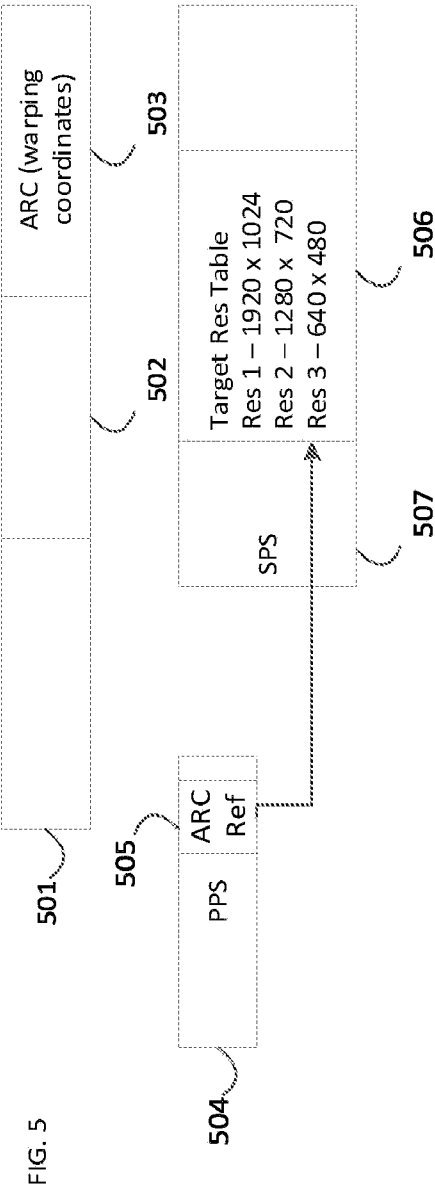


Fig. 6

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

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

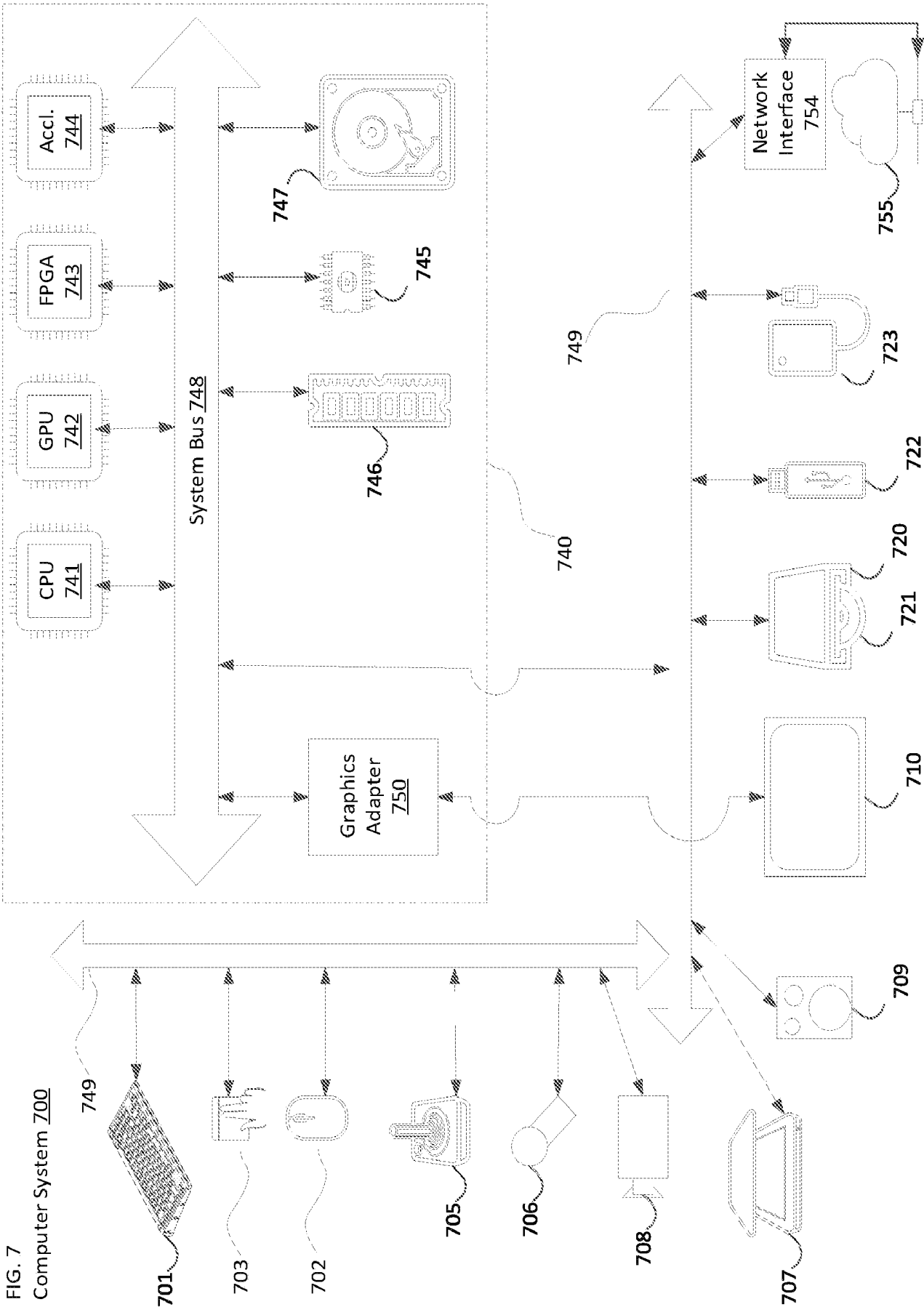


Fig. 8

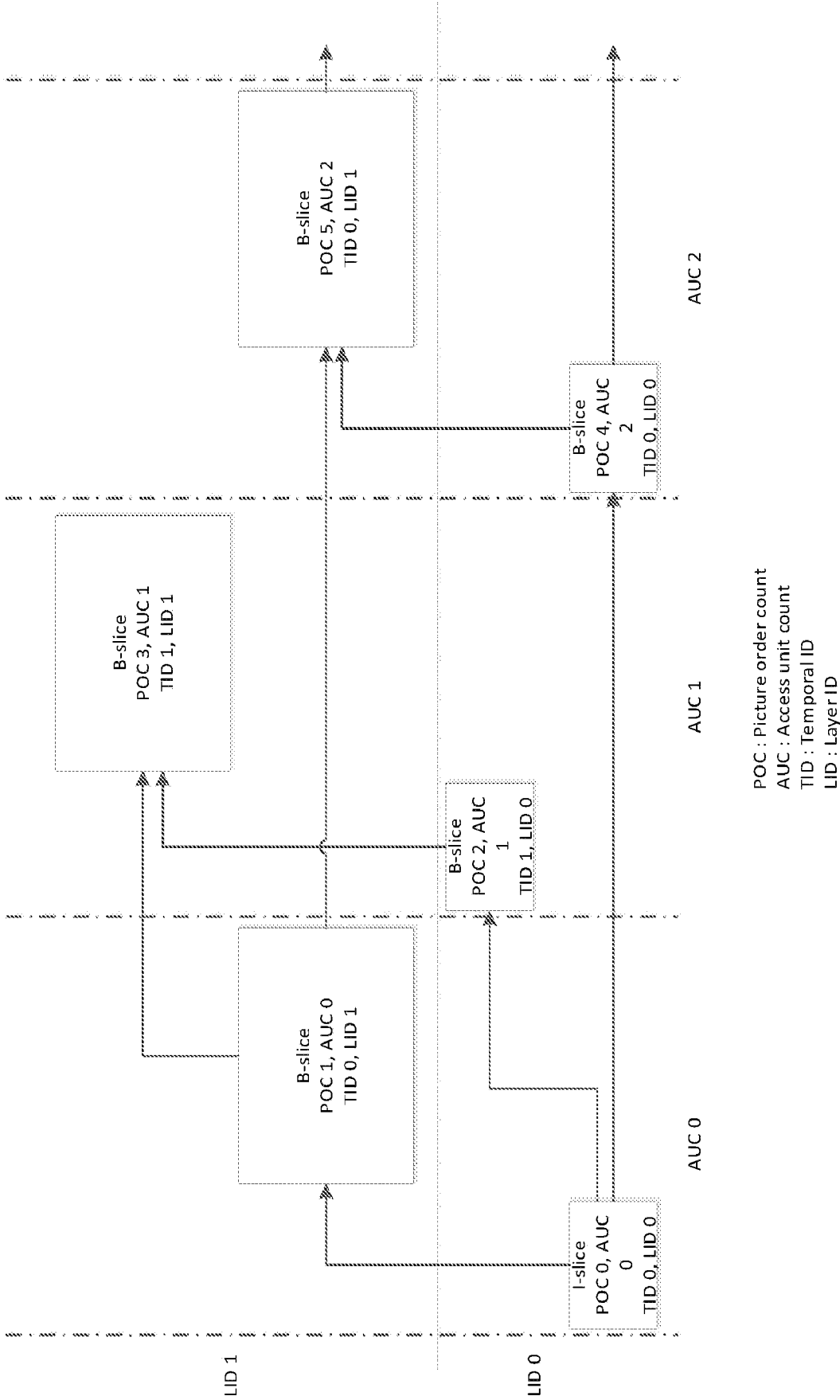


Fig. 9

video_parameter_set_rbsp() { vps_video_parameter_set_id vps_max_layers_minus1 for(i=0; i <= vps_max_layers_minus1; i++) { vps_included_layer_id[i] vps_reserved_zero_bit } vps_constraint_info_present_flag vps_constant_poc_cycle_per_au if(vps_constant_poc_cycle_per_au) vps_poc_cycle_au ... }	Descriptor: u(4) u(8) s u(7) u(1) u(1) u(1) s u(8) s s
slice_header() { slice_pic_parameter_set_id if(rect_slice_flag == NumBricksInPic > 1) slice_address if(!rect_slice_flag && !single_brck_per_slice_flag) num_bricks_in_slice_minus1 slice_type if(NaUIntType == GRA_NUT) recovery_poc_cnt slice_pic_order_cnt_lsb ... if(!vps_constant_poc_cycle_per_au) slice_poc_cycle_au ... }	Descriptor: ue(v) s u(v) ue(v) ue(v) s se(v) u(v) s s u(8) s s s

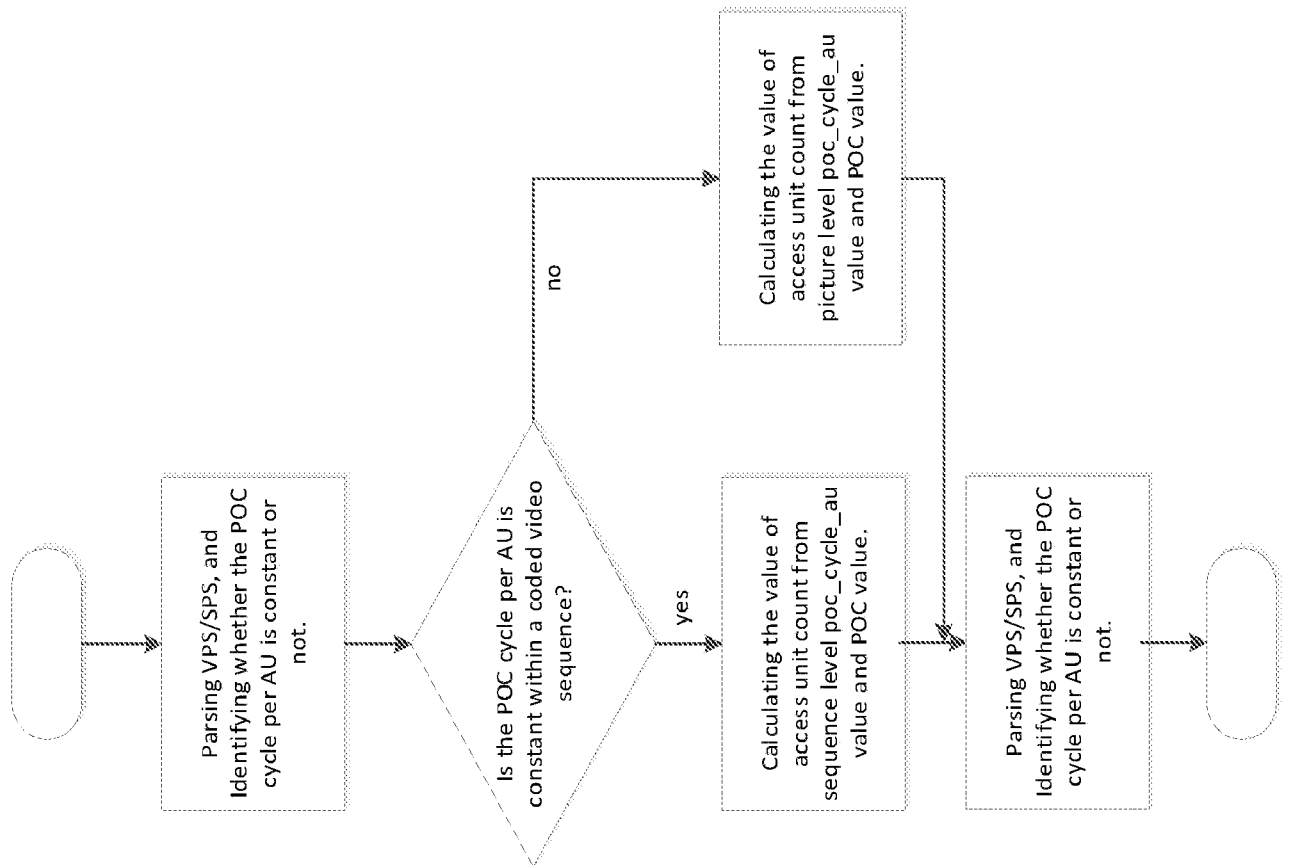


Fig. 10

FIG. 11

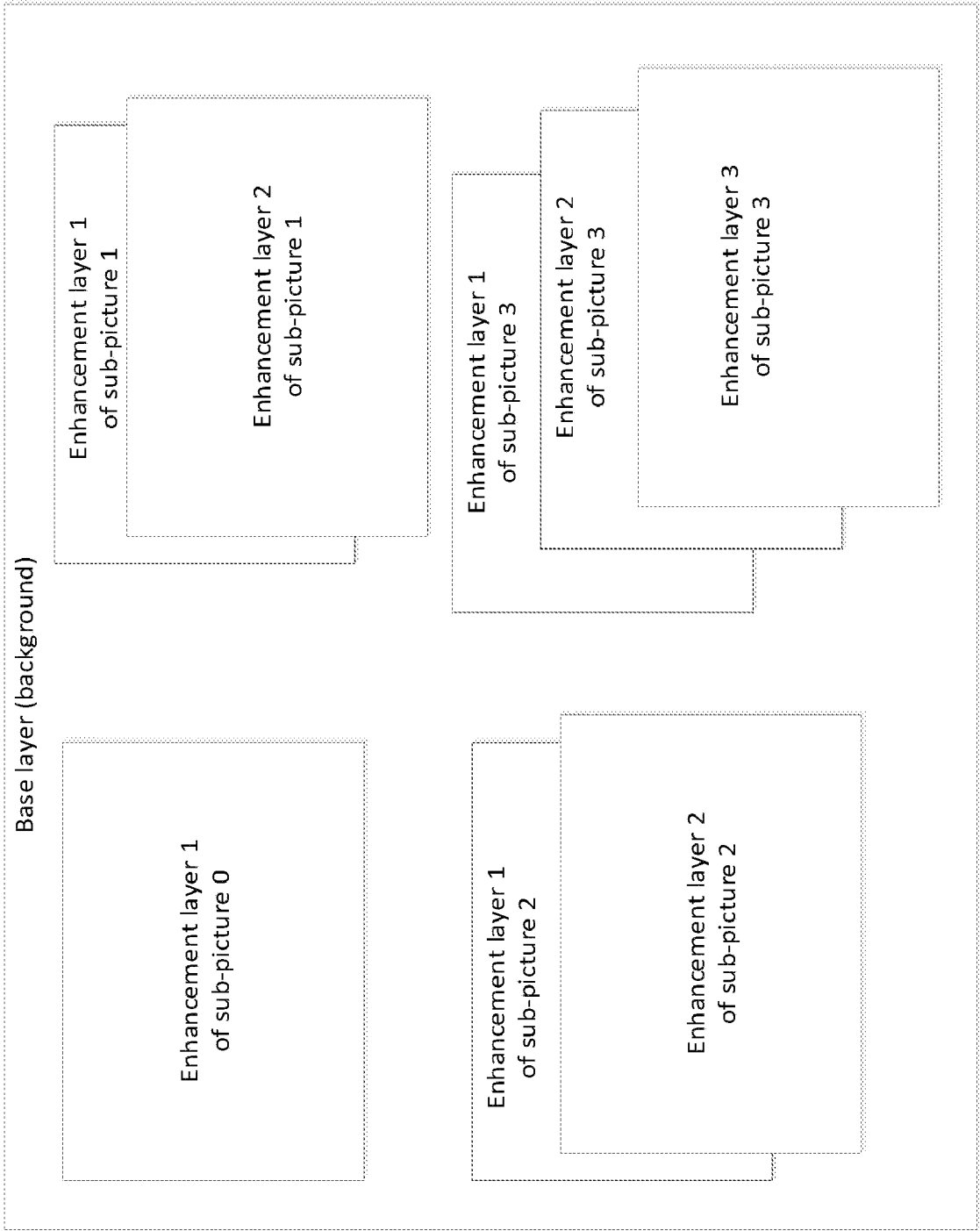


FIG. 12

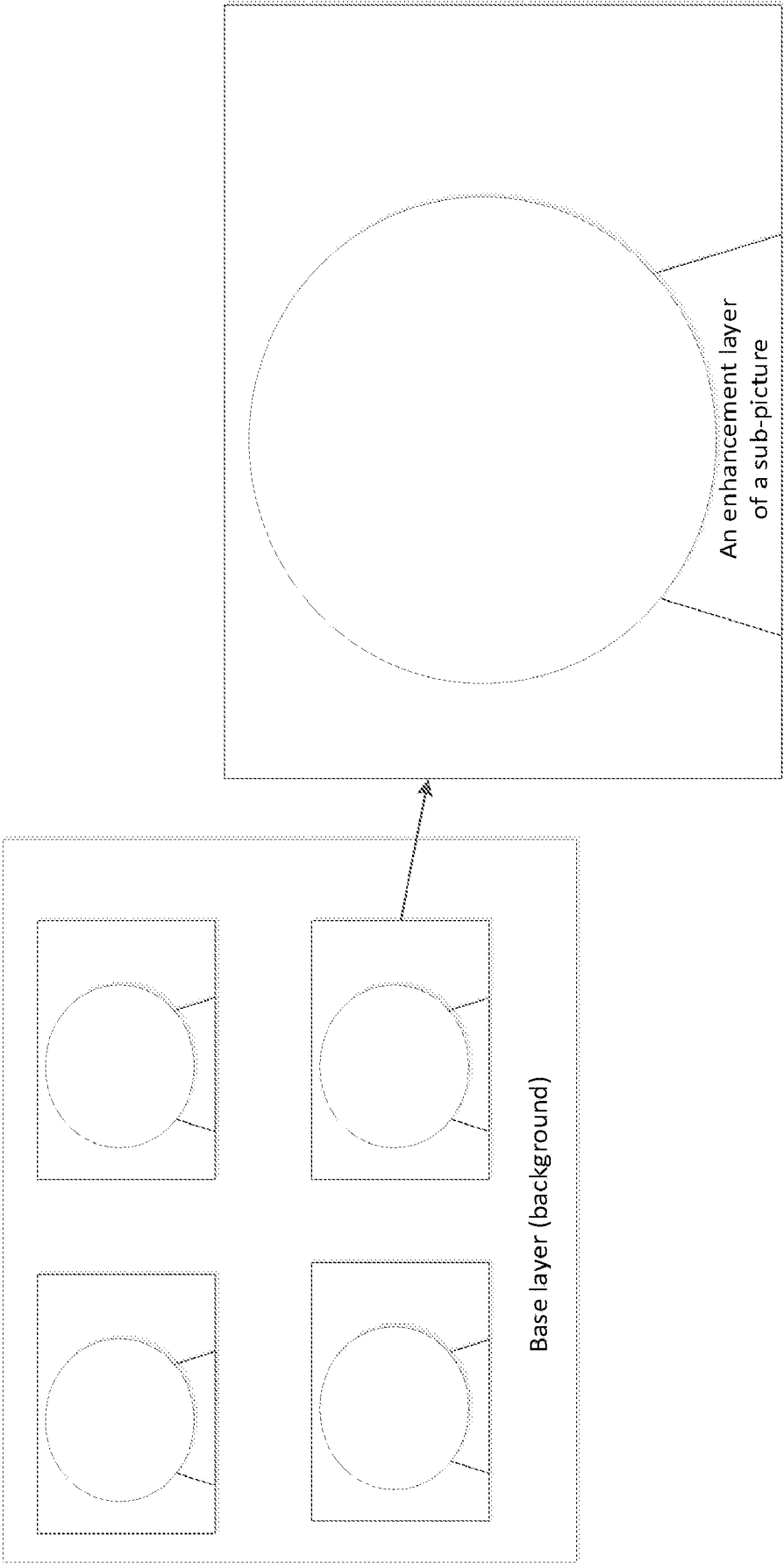
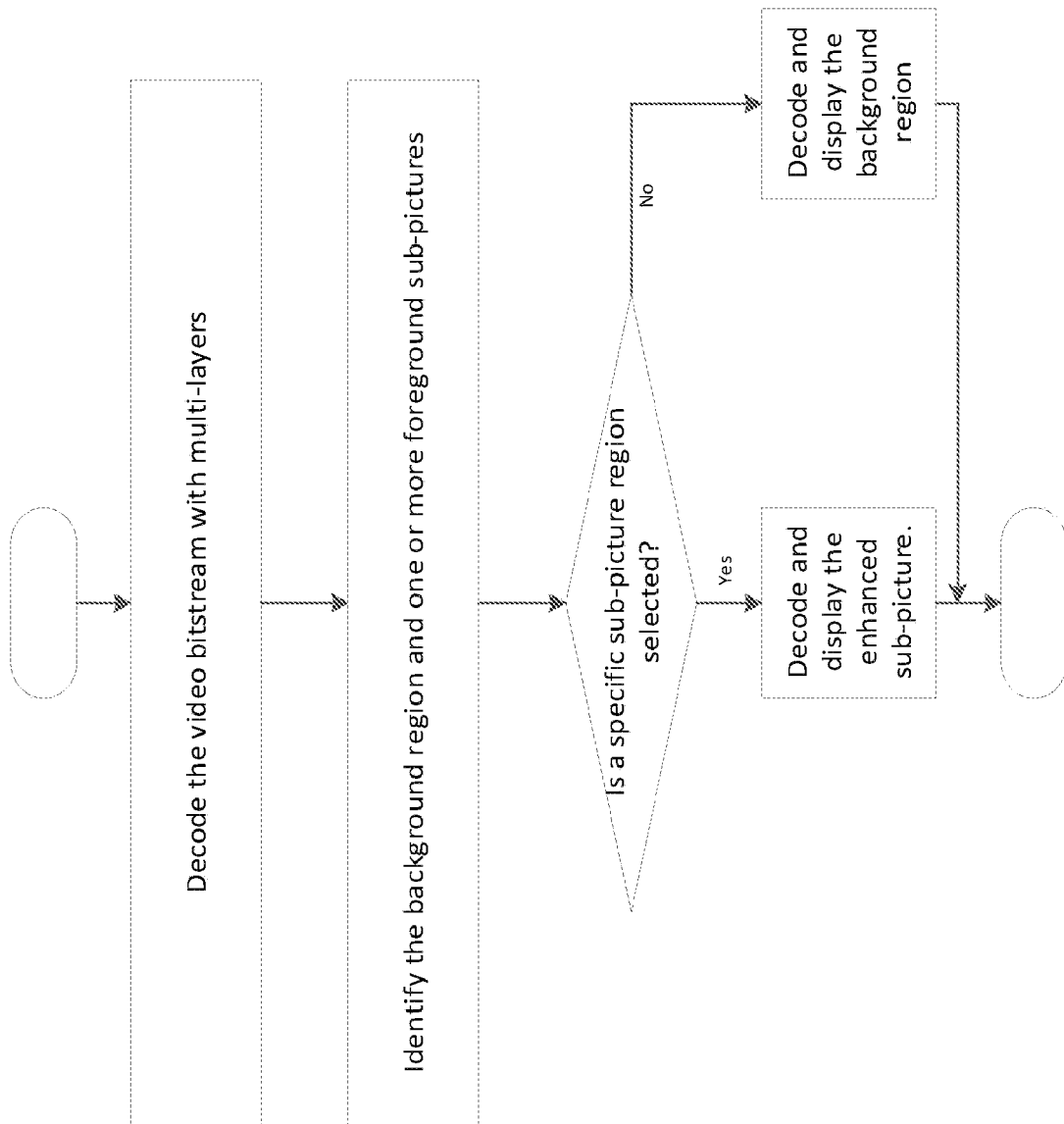


Fig. 13



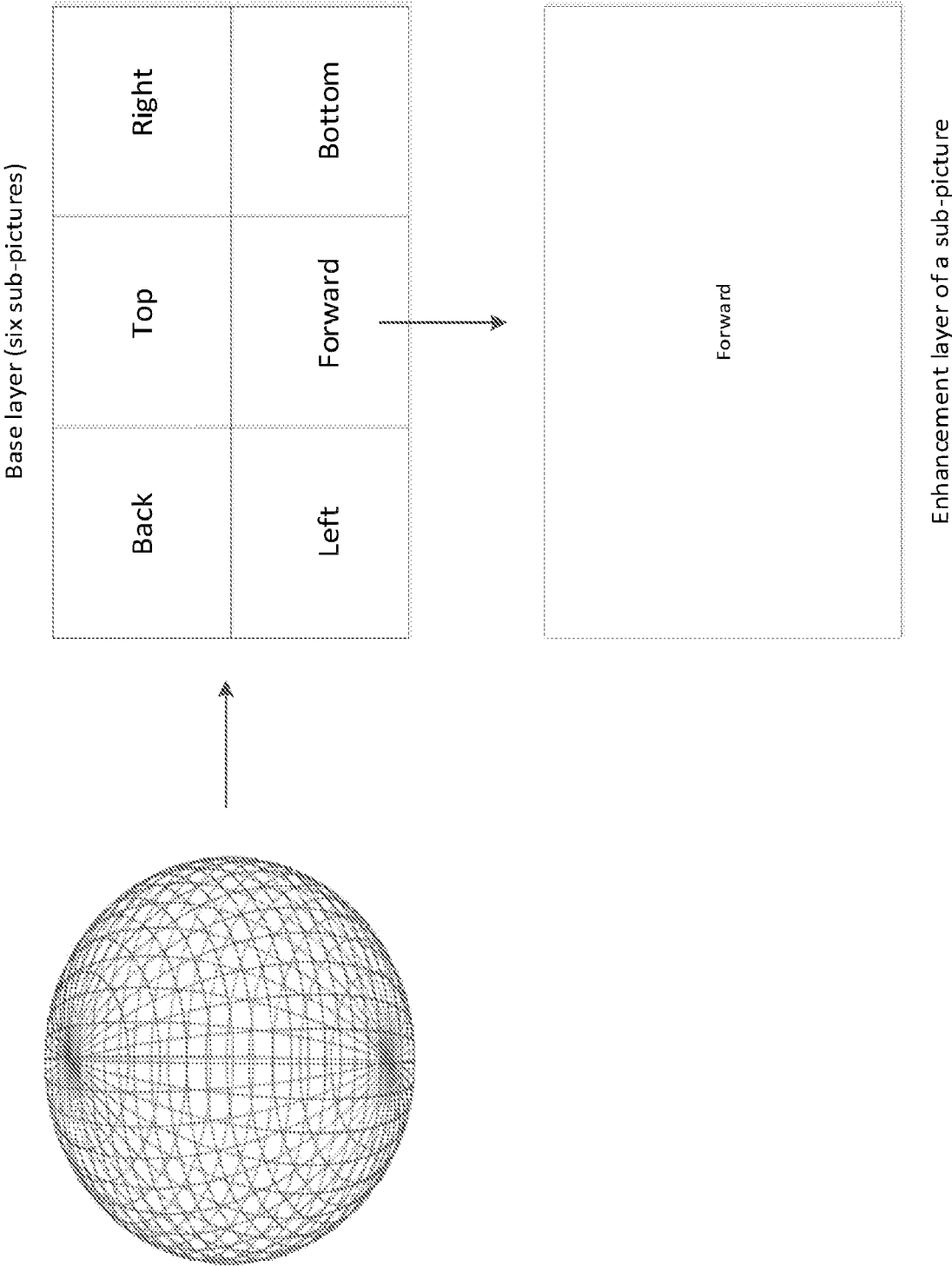


Fig. 14

Fig. 15

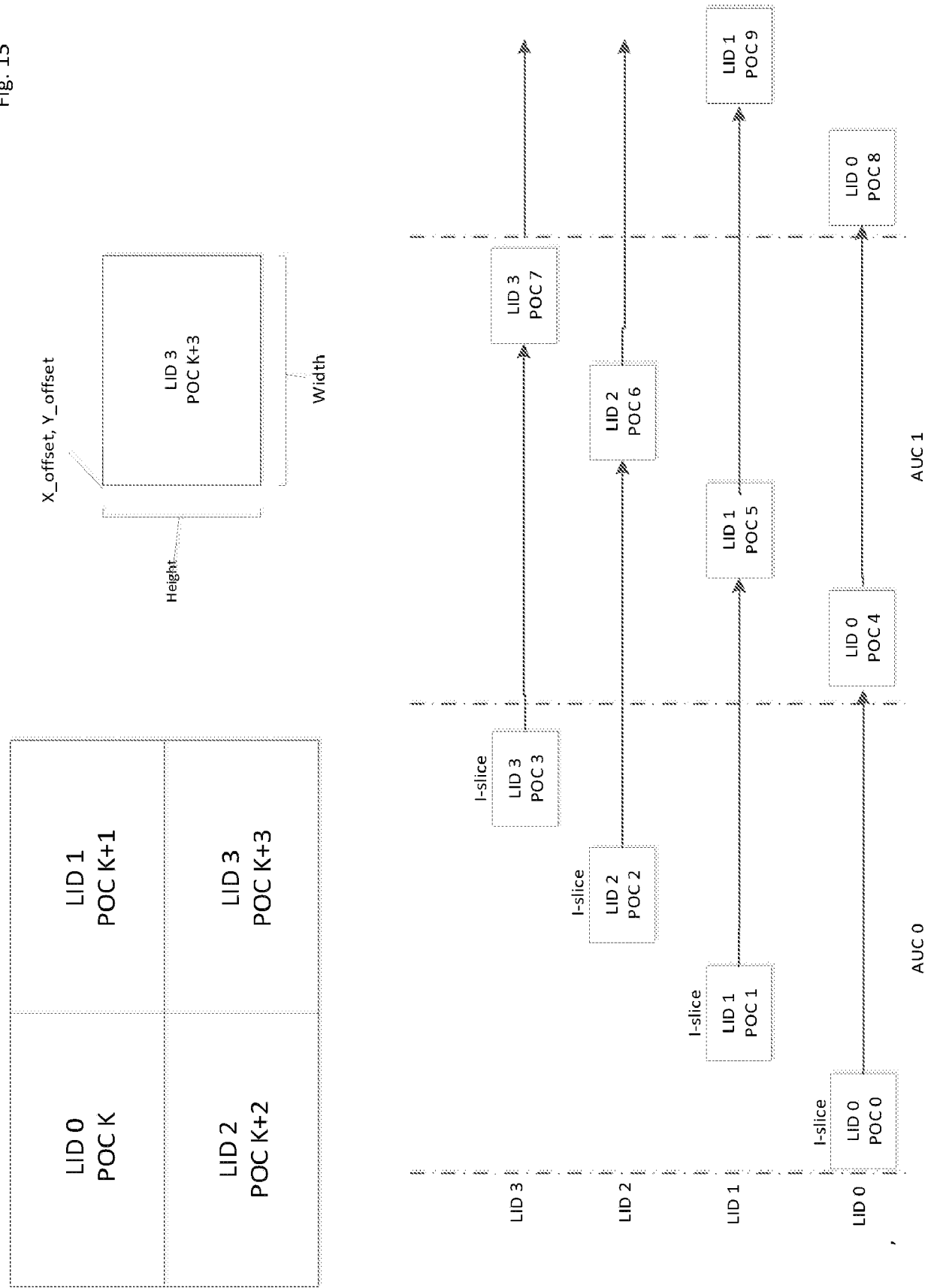


Fig. 16

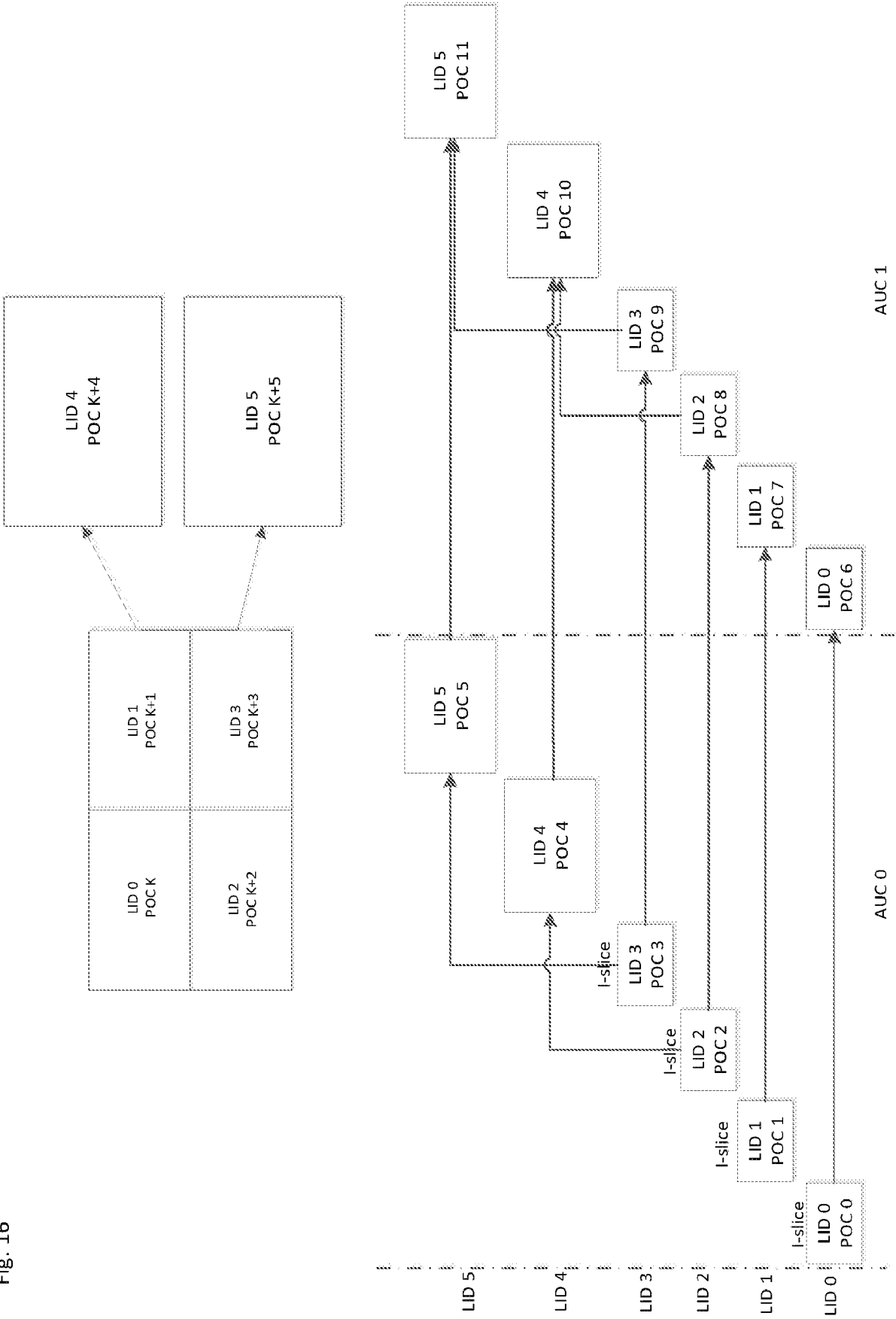


Fig. 17

video_parameter_set_rbsp() { ¹⁰	Descriptor ¹⁰
vps_video_parameter_set_id ¹⁰	u(4) ¹⁰
vps_max_layers_minus1 ¹⁰	u(3) ¹⁰
for(i = 0; i <= vps_max_layers_minus1; i++) { ¹⁰	¹⁰
vps_included_layer_id[i] ¹⁰	u(7) ¹⁰
vps_reserved_zero_bit ¹⁰	u(1) ¹⁰
} ¹⁰	¹⁰
... ¹⁰	¹⁰
vps_sub_picture_dividing_flag ¹⁰	u(1) ¹⁰
if(vps_sub_picture_dividing_flag) { ¹⁰	¹⁰
vps_full_pic_width_in_luma_samples ¹⁰	ue(1) ¹⁰
vps_full_pic_height_in_luma_samples ¹⁰	ue(1) ¹⁰
} ¹⁰	¹⁰
... ¹⁰	¹⁰
} ¹⁰	¹⁰
¹⁰	
• seq_parameter_set_rbsp() { ¹⁰	Descriptor ¹⁰
sps_decoding_parameter_set_id ¹⁰	u(4) ¹⁰
sps_video_parameter_set_id ¹⁰	u(4) ¹⁰
sps_max_sub_layers_minus1 ¹⁰	u(3) ¹⁰
... ¹⁰	¹⁰
pic_width_in_luma_samples ¹⁰	ue(1) ¹⁰
pic_height_in_luma_samples ¹⁰	ue(1) ¹⁰
if(vps_sub_picture_dividing_flag) { ¹⁰	¹⁰
pic_offset_x ¹⁰	ue(1) ¹⁰
pic_offset_y ¹⁰	ue(1) ¹⁰
} ¹⁰	¹⁰
... ¹⁰	¹⁰
} ¹⁰	¹⁰

Fig. 18

sub_region_partitioning_info(payloadSize) { ²	Descriptor ²
num_sub_region ²	us(v) ²
num_layers ²	us(v) ²
for(i = 0; i <= num_layers; i++) ²	²
layer_id [i] ²	us(v) ²
for(i = 1; i < num_layers; i++) ²	²
for(j = 0; j < i; i++) ²	²
direct_dependency_flag[i][j] ²	u(1) ²
for(i = 0; i < num_sub_region; i++) { ²	²
num_layers_for_region[i] ²	us(v) ²
for(i = 0; i < num_layers_for_region[i]; i++) ²	²
sub_region_layer_id[i][j] ²	us(v) ²
sub_region_offset_x[i] ²	us(v) ²
sub_region_offset_y[i] ²	us(v) ²
sub_region_width[i] ²	us(v) ²
sub_region_height [i] ²	us(v) ²
} ²	²
... ²	²
} ²	²

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() {	
...	
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

	Descriptor
video parameter set rbsp() {	
...	
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[j][i]	ue(v)
for(k = 0; k < num_output_subpic_layer[j][i]; k++)	
sub_pic_id_layer[j][i][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)
}	
}	
...	
}	

22/24

Fig. 22

Descriptor	
	video_parameter_sct_rbsp() {
u(4)	vps_video_parameter_set_id
u(6)	vps_max_layers_minus1
u(3)	vps_max_sublayers_minus1
	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)
u(1)	vps_all_independent_layers_flag
	for(i = 0; i <= vps_max_layers_minus1; i++) {
u(6)	vps_layer_id[i]
	if (i > 0 && !vps_all_independent_layers_flag) {
u(1)	vps_independent_layer_flag[i]
	if (!vps_independent_layer_flag[i]) {
	for(j = 0; j <= i; j++)
u(1)	vps_direct_ref_layer_flag[i][j]
u(1)	max_tid_ref_present_flag[i]
	if (max_tid_ref_present_flag[i])
u(3)	max_tid_ref_pics_plus1[i]
	}
	}
	}
	if (vps_max_layers_minus1 > 0) {
	if (vps_all_independent_layers_flag)
u(1)	each_layer_is_an_ols_flag
	if (!each_layer_is_an_ols_flag) {
u(2)	ols_mode_idc
	if (ols_mode_idc == 2) {
u(8)	num_output_layer_sets_minus1
	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

	Descriptor
picture header structure() {	
gdr or irap pic flag	u(1)
iff gdr or irap pic flag)	
gdr pic flag	u(1)
ph inter slice allowed flag	u(1)
iff ph inter slice allowed flag)	
ph intra slice allowed flag	u(1)
non reference picture flag	u(1)
...	
iff output flag present flag&&'non reference picture flag)	
pic output flag	u(1)
iff rpl info in ph flag)	
ref pic lists()	
...	
}	