

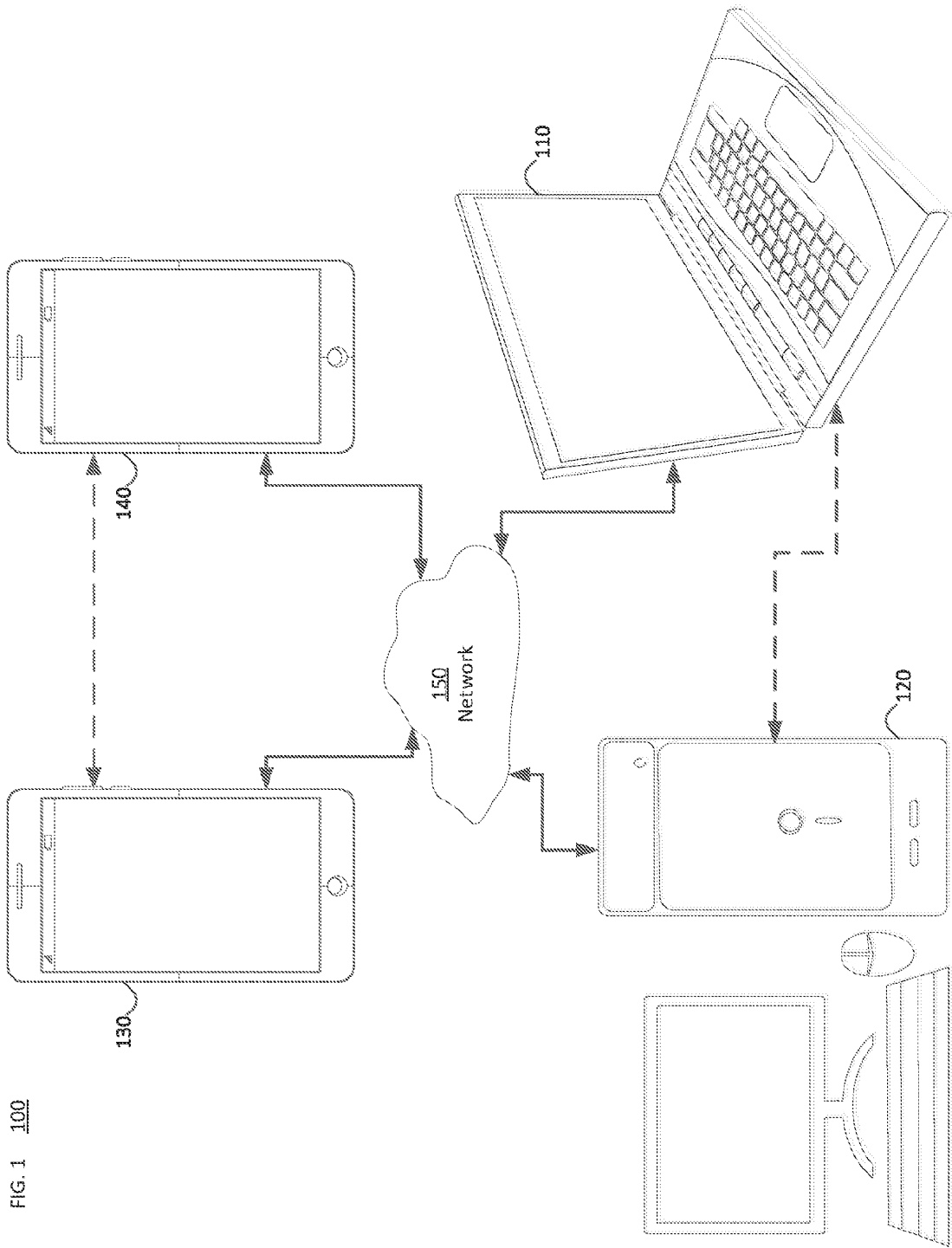


FIG. 1 100

FIG. 2 Streaming System 200

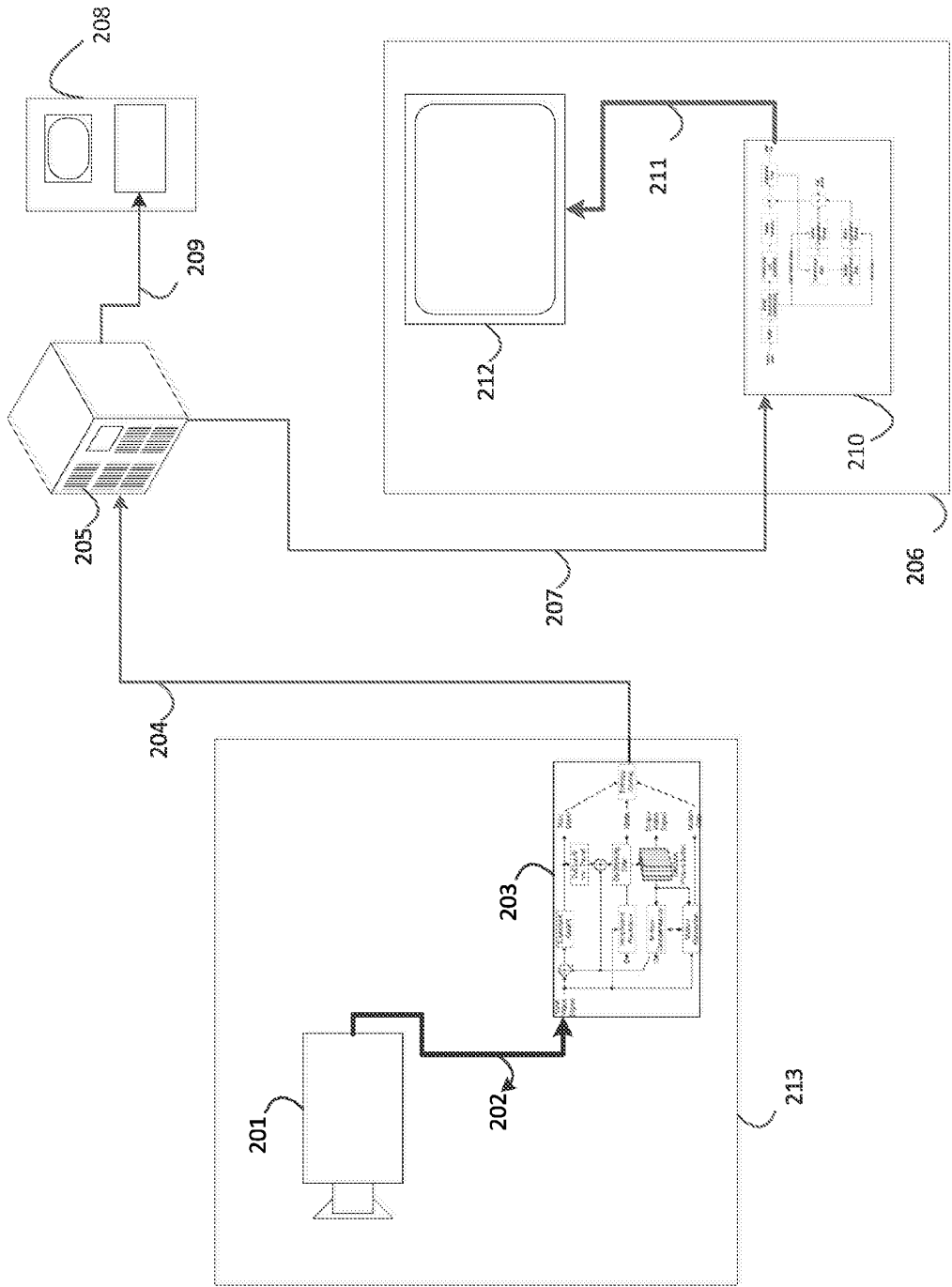


FIG. 3 Decoder 210

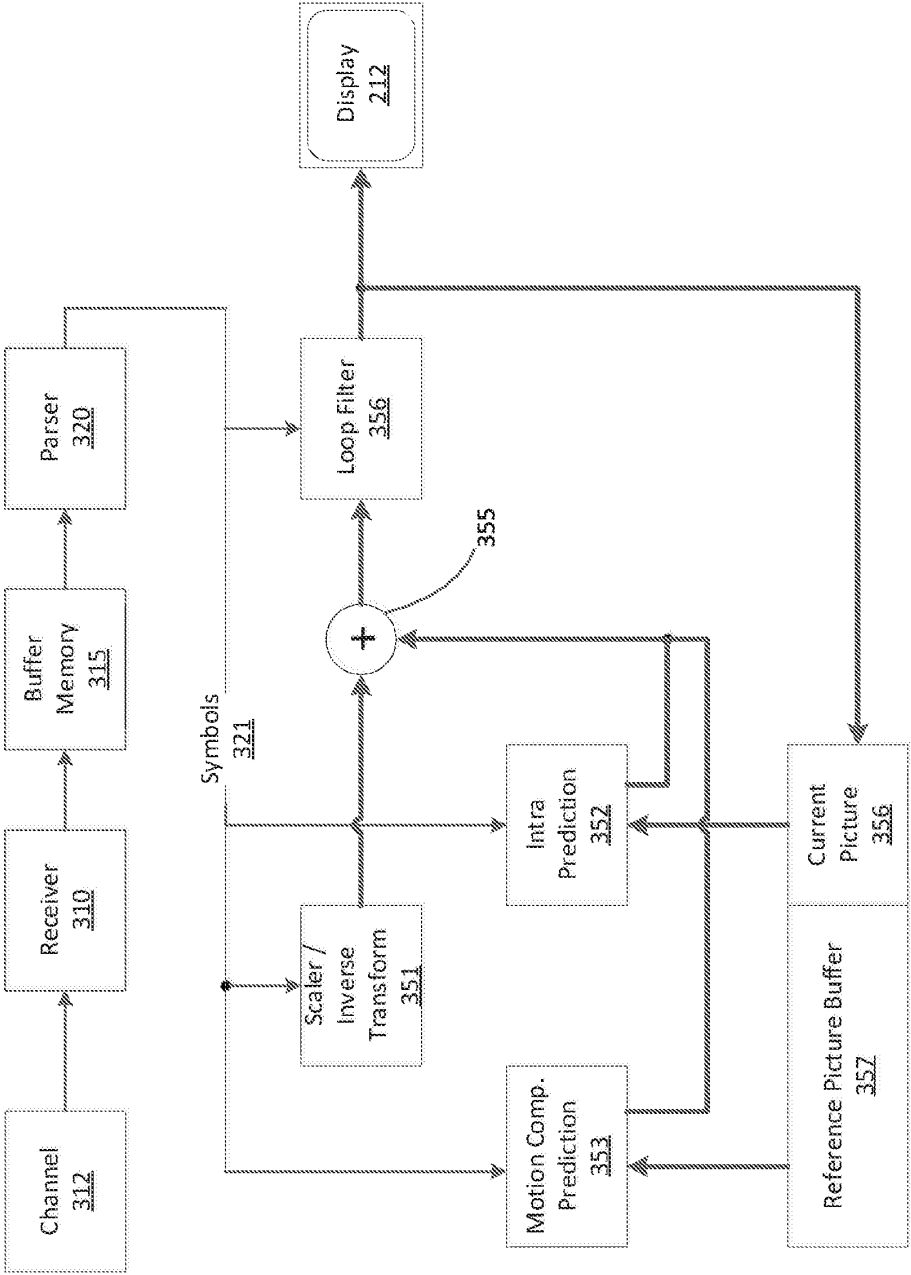
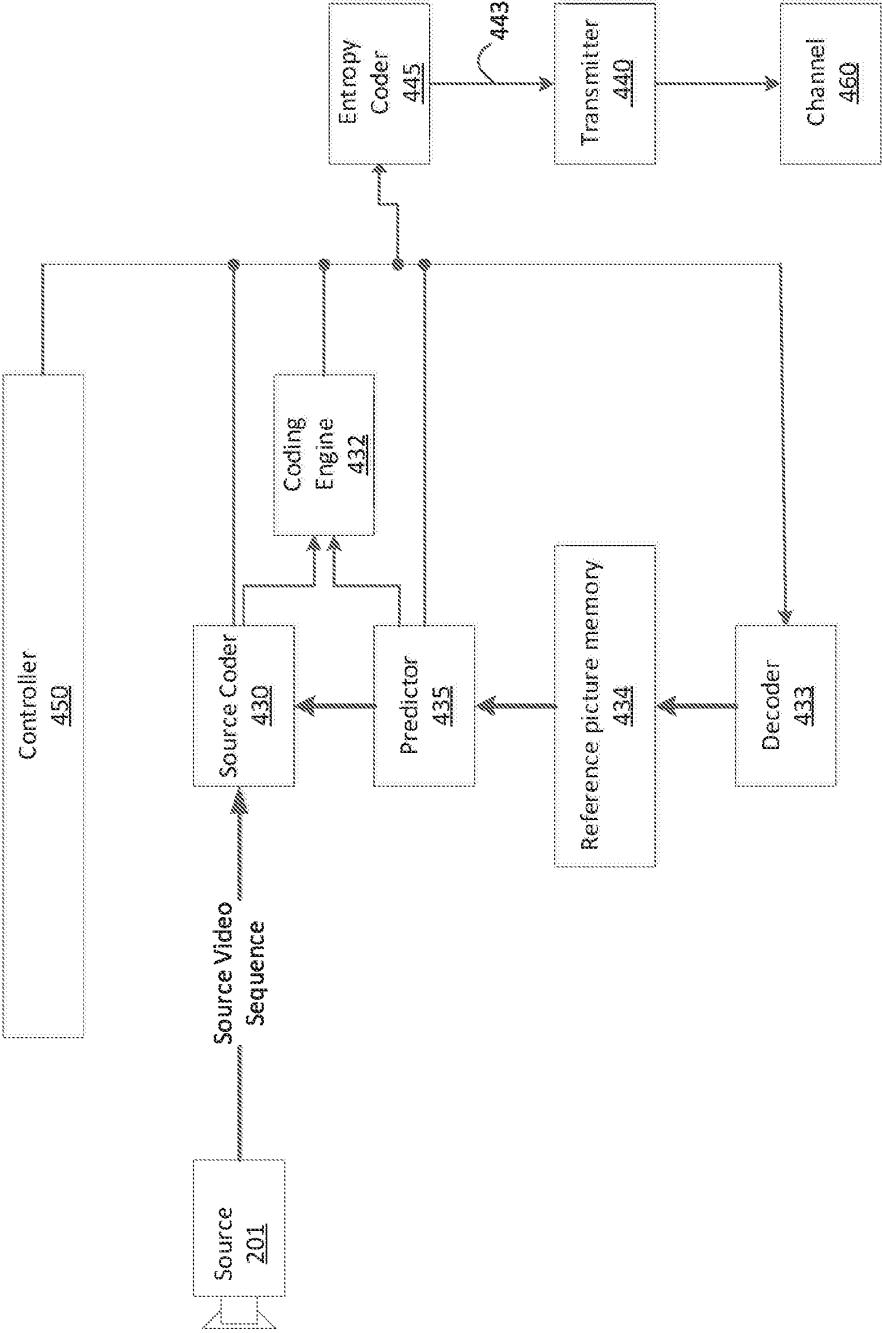


FIG. 4 Encoder 203



5.
6.
7.

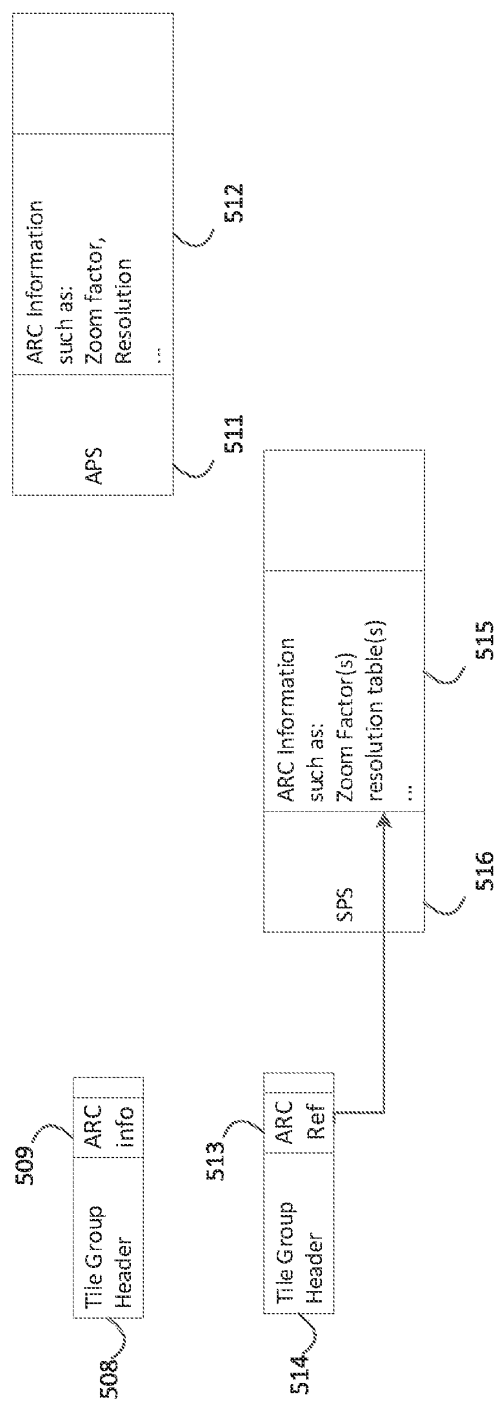
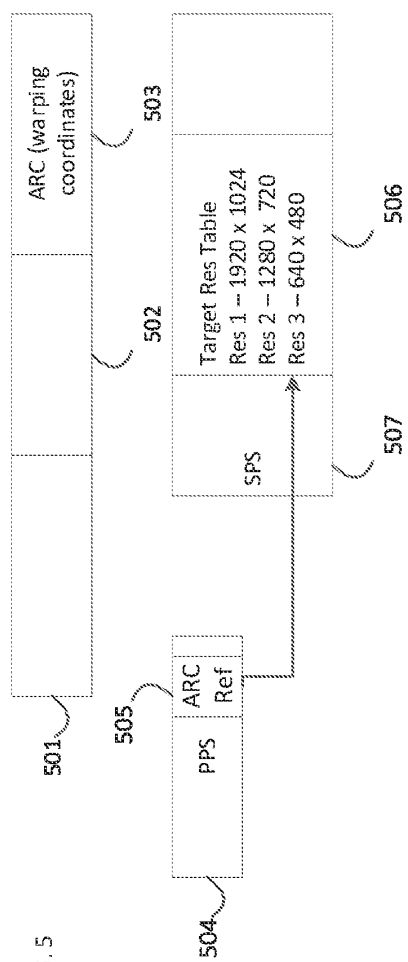


Fig. 6

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)
	}		
	}		
	...		
	}		

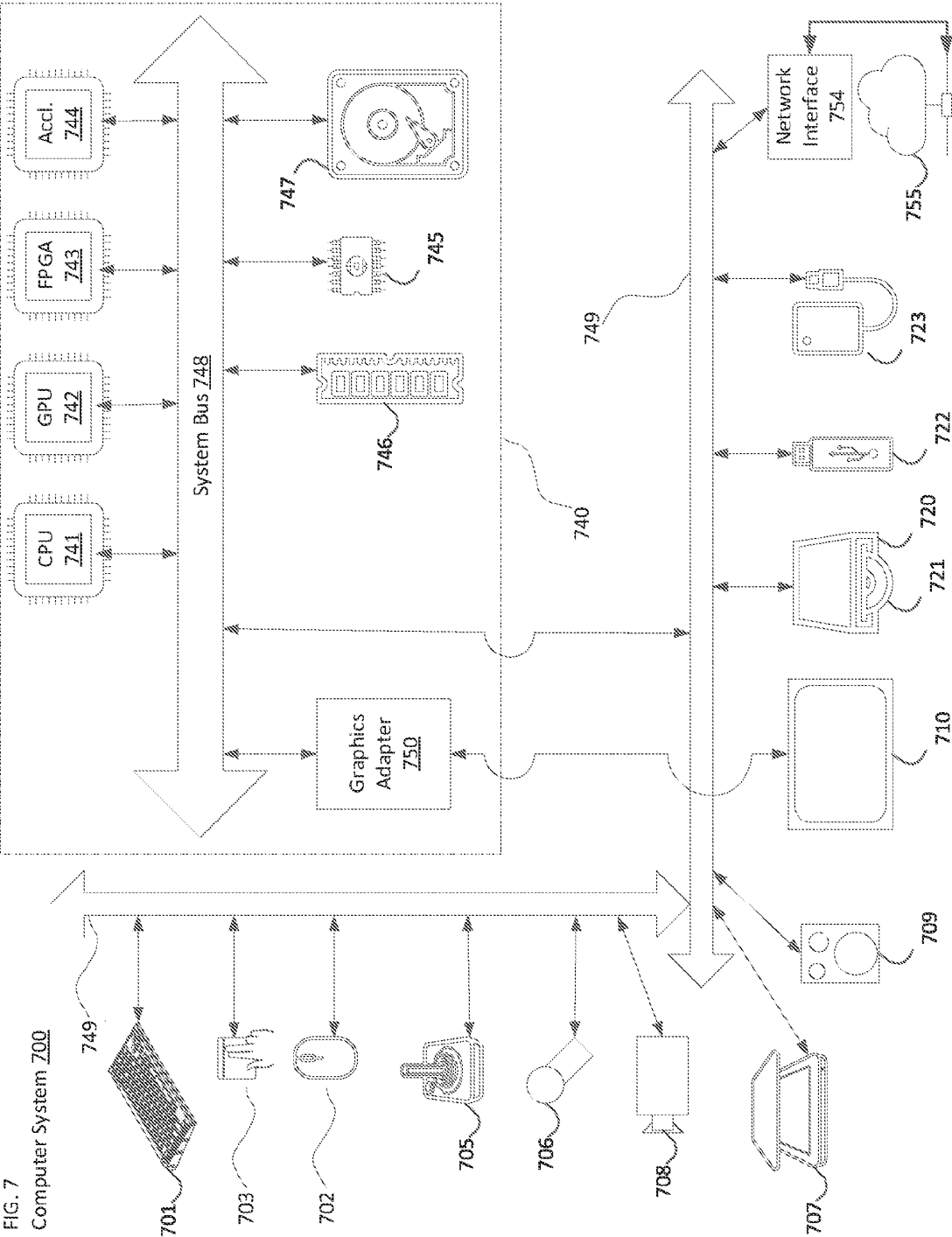


Fig. 8

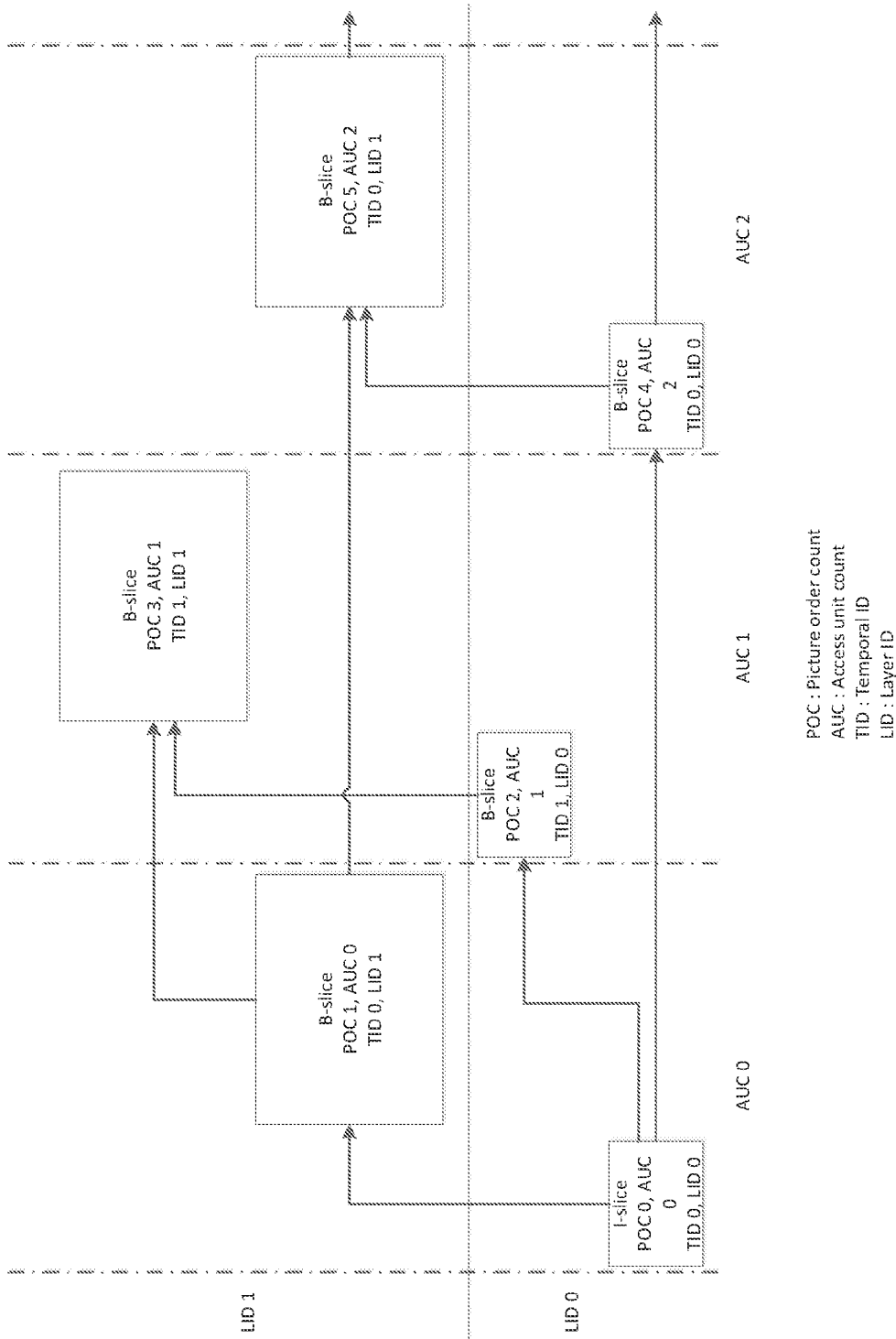


Fig. 9

<code>video_parameter_set_flag() {</code>	Descriptor
<code> vps_video_parameter_set_id</code>	<code>u(4)</code>
<code> vps_max_layers_minus1</code>	<code>u(8)</code>
<code> for(i = 0; i <= vps_max_layers_minus1; i++) {</code>	<code></code>
<code> vps_included_layer_id[i]</code>	<code>u(7)</code>
<code> vps_reserved_zero_bit</code>	<code>u(1)</code>
<code> }</code>	<code></code>
<code> vps_constraint_info_present_flag</code>	<code>u(1)</code>
<code> vps_constant_poc_cycle_per_aud</code>	<code>u(1)</code>
<code> if(vps_constant_poc_cycle_per_aud)</code>	<code></code>
<code> vps_poc_cycle_aud</code>	<code>u(8)</code>
<code> ...</code>	<code></code>
<code>}</code>	<code></code>

<code>slice_header() {</code>	Descriptor
<code> slice_pic_parameter_set_id</code>	<code>ue(4)</code>
<code> if(rect_slice_flag NumBricksInPic > 1)</code>	<code></code>
<code> slice_address</code>	<code>u(1)</code>
<code> if(rect_slice_flag && !single_brick_per_slice_flag)</code>	<code></code>
<code> num_bricks_in_slice_minus1</code>	<code>ue(4)</code>
<code> slice_type</code>	<code>ue(4)</code>
<code> if(NumUnitType == GRA_NUT)</code>	<code></code>
<code> recovery_poc_cnt</code>	<code>se(1)</code>
<code> slice_pic_order_cnt_lsb</code>	<code>u(1)</code>
<code> ...</code>	<code></code>
<code> if(!vps_constant_poc_cycle_per_aud)</code>	<code></code>
<code> slice_poc_cycle_aud</code>	<code>u(8)</code>
<code> ...</code>	<code></code>
<code>}</code>	<code></code>

10/24

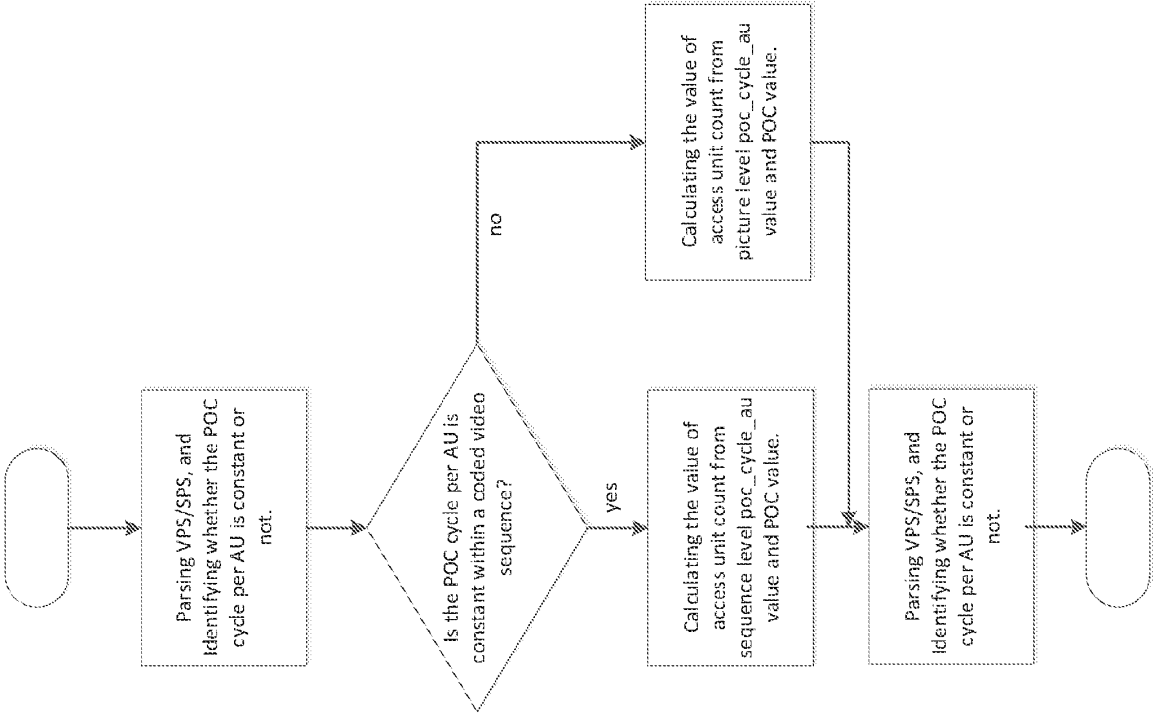
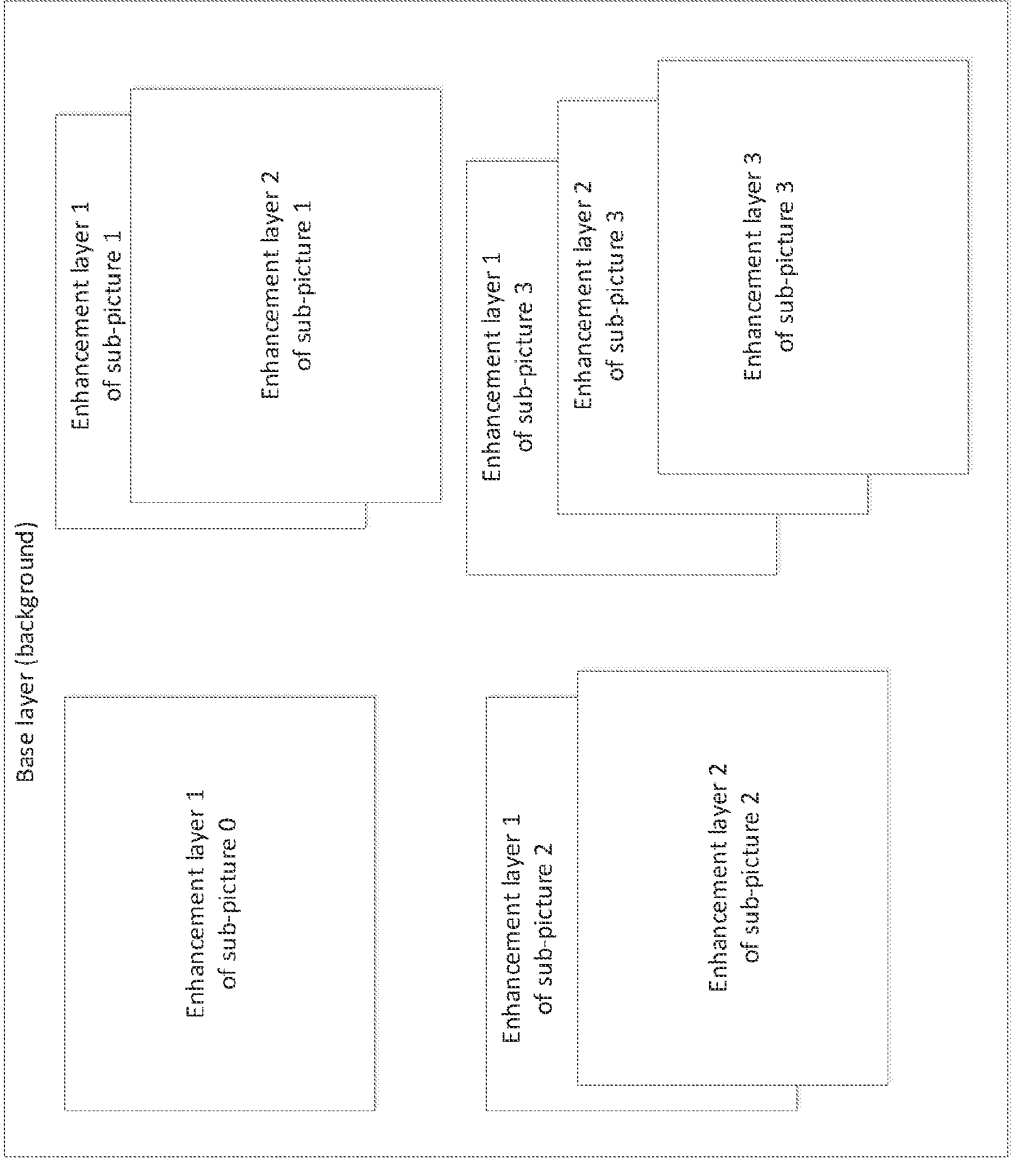


Fig. 10

FIG. 11



12/24

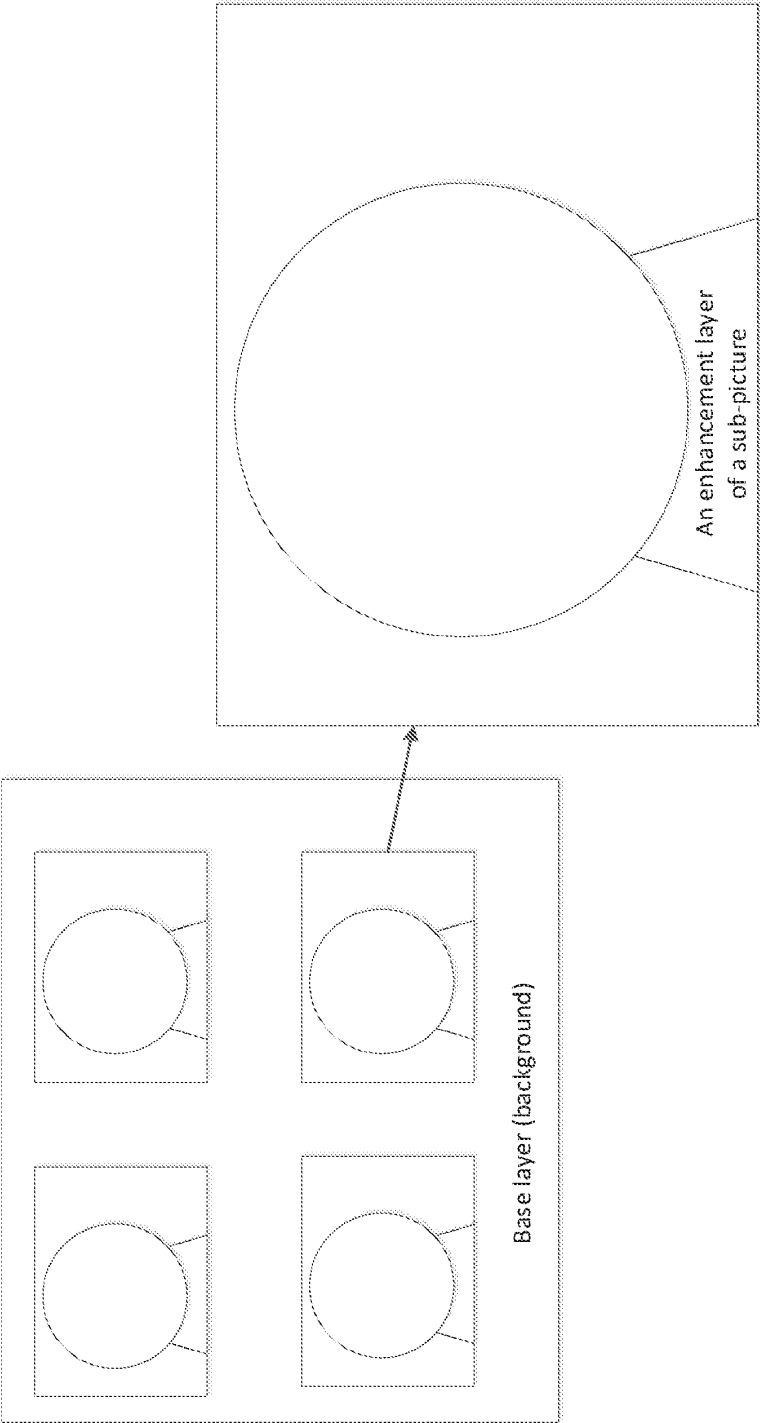
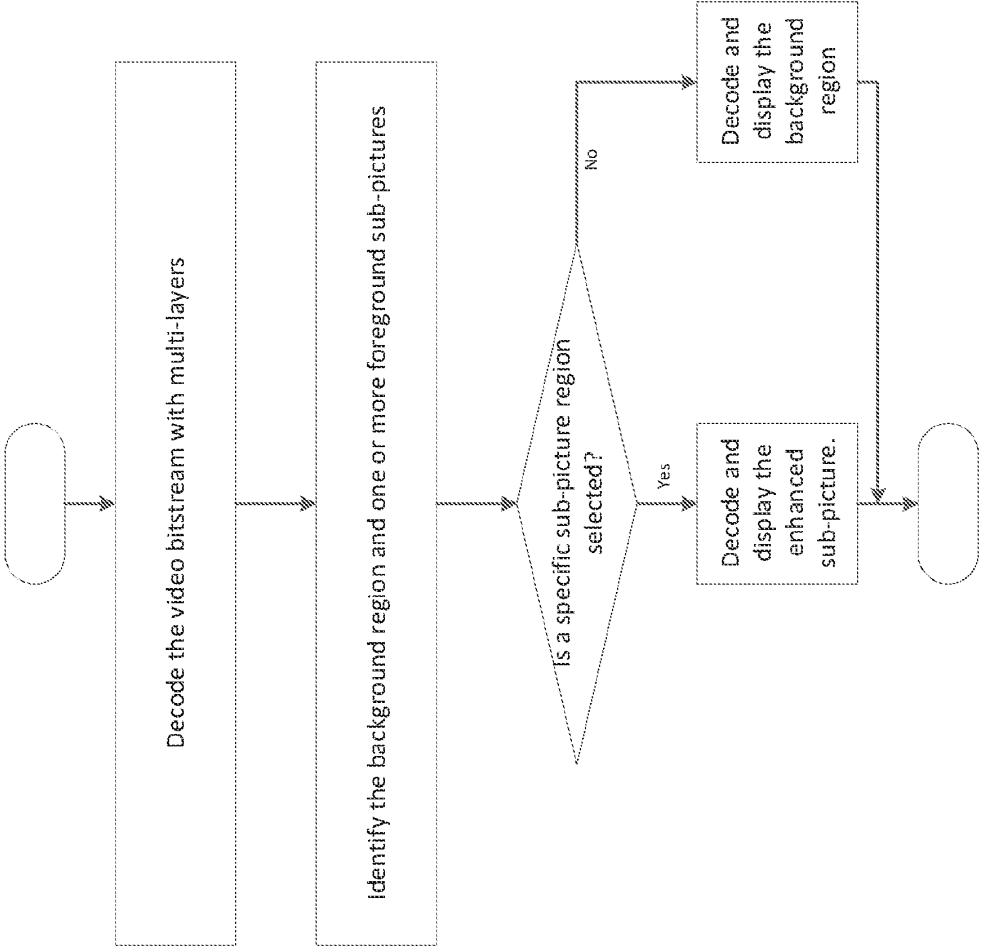


FIG. 12

Fig. 13



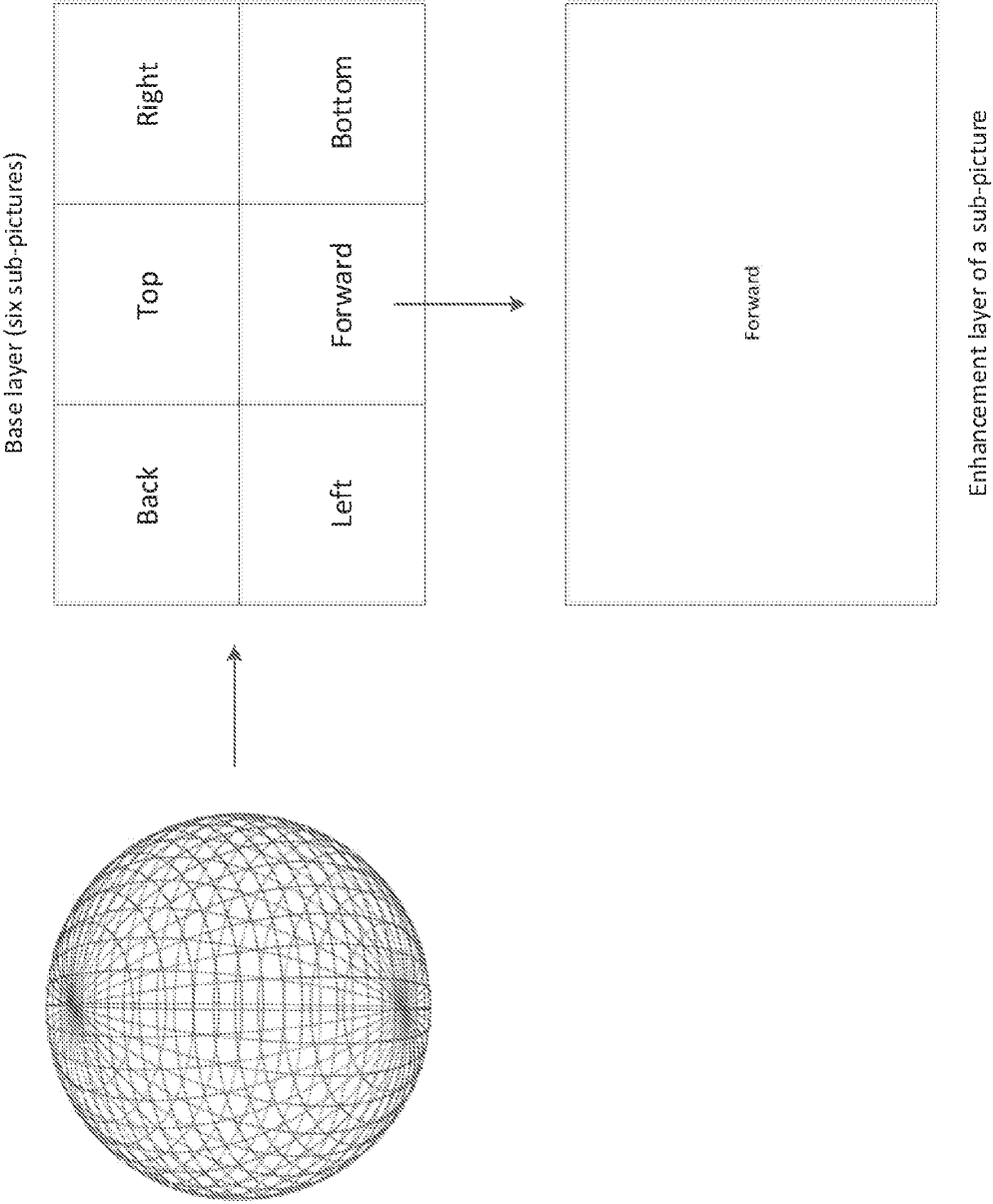


Fig. 14

Fig. 15

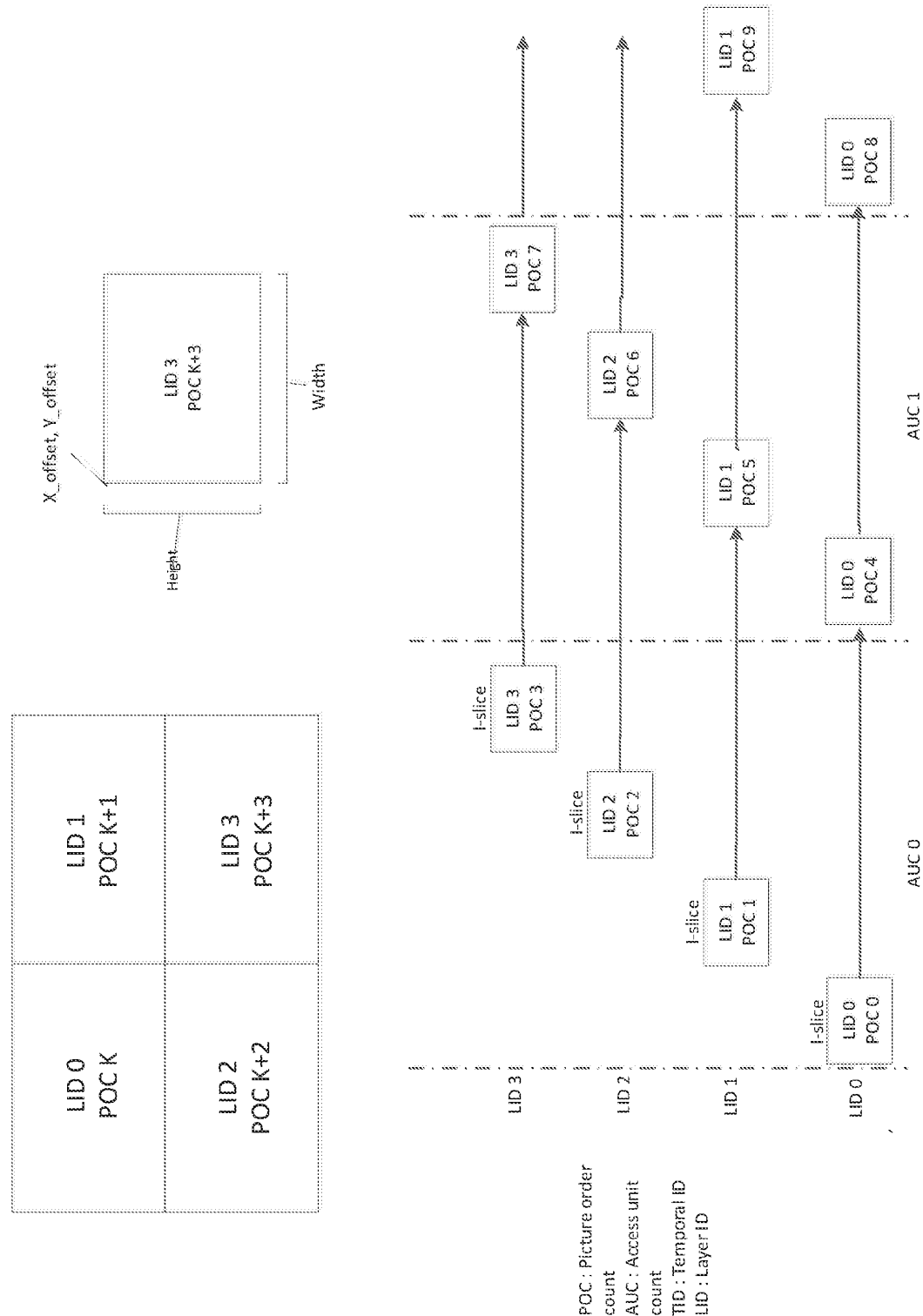


Fig. 16

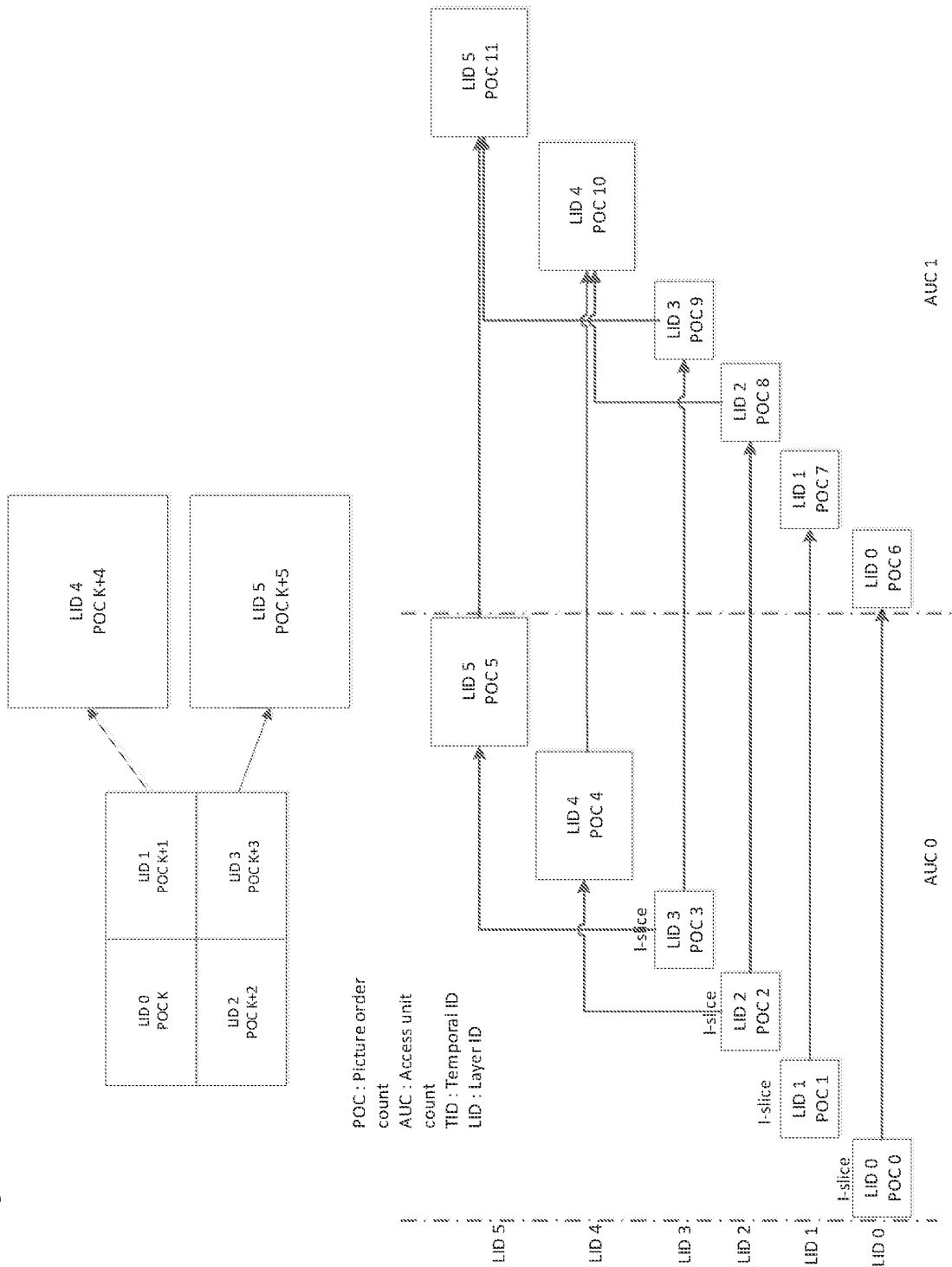


Fig. 18

sub_region_partitioning_info.payloadSize) { ⁴	Descriptor ⁴
num_sub_region ⁴	ug(v) ⁴
num_layers ⁴	ug(v) ⁴
for(i = 0; i <= num_layers; i++) ⁴	⁴
layer_id [i] ⁴	ug(v) ⁴
for(i = 1; i < num_layers; i++) ⁴	⁴
for(j = 0; j < i; j++) ⁴	⁴
direct_dependency_flag[i][j] ⁴	u(1) ⁴
for(i = 0; i < num_sub_region; i++) { ⁴	⁴
num_layers_for_region[i] ⁴	ug(v) ⁴
for(j = 0; j < num_layers_for_region[i]; j++) ⁴	⁴
sub_region_layer_id[i][j] ⁴	ug(v) ⁴
sub_region_offset_x[i] ⁴	ug(v) ⁴
sub_region_offset_y[i] ⁴	ug(v) ⁴
sub_region_width[i] ⁴	ug(v) ⁴
sub_region_height[i] ⁴	ug(v) ⁴
} ⁴	⁴
... ⁴	⁴
} ⁴	⁴

Fig. 19

	Descriptor
video_parameter_set_hdr() {	
...	
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 < NumLayersInList[i]; j++) {	
output_layer_flag[i][j]	u(1)
profile_tier_level_idx[i][j]	u(v)
}	
...	
}	

Fig. 20

	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)
}	
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 < NumLayersInList[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 < NumLayersInIdLast[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

	Descriptor
pic parameter set rbsp() {	
pps pic parameter set id	ue(v)
pps seq parameter set id	u(4)
mixed_nalu_types_in_pic_flag	u(1)
pic_width_in_luma_samples	ue(v)
pic_height_in_luma_samples	ue(v)
pps_conformance_window_flag	u(1)
if(pps_conformance_window_flag) {	
pps_conf_win_left_offset	ue(v)
pps_conf_win_right_offset	ue(v)
pps_conf_win_top_offset	ue(v)
pps_conf_win_bottom_offset	ue(v)
}	
scaling_window_explicit_signalling_flag	u(1)
if(scaling_window_explicit_signalling_flag) {	
scaling_win_left_offset	ue(v)
scaling_win_right_offset	ue(v)
scaling_win_top_offset	ue(v)
scaling_win_bottom_offset	ue(v)
}	
output_flag_present_flag	u(1)
subpic_id_mapping_in_pps_flag	u(1)
if(subpic_id_mapping_in_pps_flag) {	
pps_num_subpics_minus1	ue(v)
pps_subpic_id_len_minus1	ue(v)
for(i = 0; i <= pps_num_subpic_minus1; i++)	
pps_subpic_id[i]	ue(v)
}	
...	
}	

23/24

Fig. 23

	Descriptor
seq_parameter_set_rbsp() {	
...	
subpic_info_present_flag	u(1)
if (subpic_info_present_flag) {	
sps_num_subpics_minus2	ue(v)
sps_independent_subpics_flag	u(1)
for (i = 0; i <= sps_num_subpics_minus2 + 1; i++) {	
if (i > 0 && pic_width_max_in_luma_samples > CtbSizeY) {	
subpic_ctu_top_left_x[i]	u(v)
if (i > 0 && pic_height_max_in_luma_samples > CtbSizeY) {	
subpic_ctu_top_left_y[i]	u(v)
if (i < sps_num_subpics_minus2 + 1 && pic_width_max_in_luma_samples > CtbSizeY) {	
subpic_width_minus1[i]	u(v)
if (i < sps_num_subpics_minus2 + 1 && pic_height_max_in_luma_samples > CtbSizeY) {	
subpic_height_minus1[i]	u(v)
if (sps_independent_subpics_flag) {	
subpic_treated_as_pic_flag[i]	u(1)
loop_filter_across_subpic_enabled_flag[i]	u(1)
}	
}	
sps_subpic_id_len_minus1	ue(v)
subpic_id_mapping_explicitly_signalled_flag	u(1)
if (subpic_id_mapping_explicitly_signalled_flag) {	
subpic_id_mapping_in_sps_flag	u(1)
if (subpic_id_mapping_in_sps_flag) {	
for (i = 0; i <= sps_num_subpics_minus2 + 1; i++) {	
sps_subpic_id[i]	u(v)
}	
}	
...	
}	

Fig. 24

	Descriptor
pic_parameter_set_rbsp() {	
...	
mixed_nalu_types_in_pic_flag	u(1)
...	
subpic_id_mapping_in_pps_flag	u(1)
if(subpic_id_mapping_in_pps_flag) {	
pps_num_subpics_minus1	ue(v)
pps_subpic_id_len_minus1	ue(v)
for(i = 0; i <= pps_num_subpics_minus1; i++)	
pps_subpic_id[i]	u(v)
}	
pps_independent_subpics_flag	u(1)
if(!pps_independent_subpics_flag) {	
if(subpic_id_mapping_in_pps_flag)	
pps_num_subpics_minus1	ue(v)
for(i = 0; i <= pps_num_subpics_minus1; i++) {	
subpic_treated_as_pic_flag[i]	u(1)
loop_filter_across_subpic_enabled_flag[i]	u(1)
}	
}	
...	
}	