

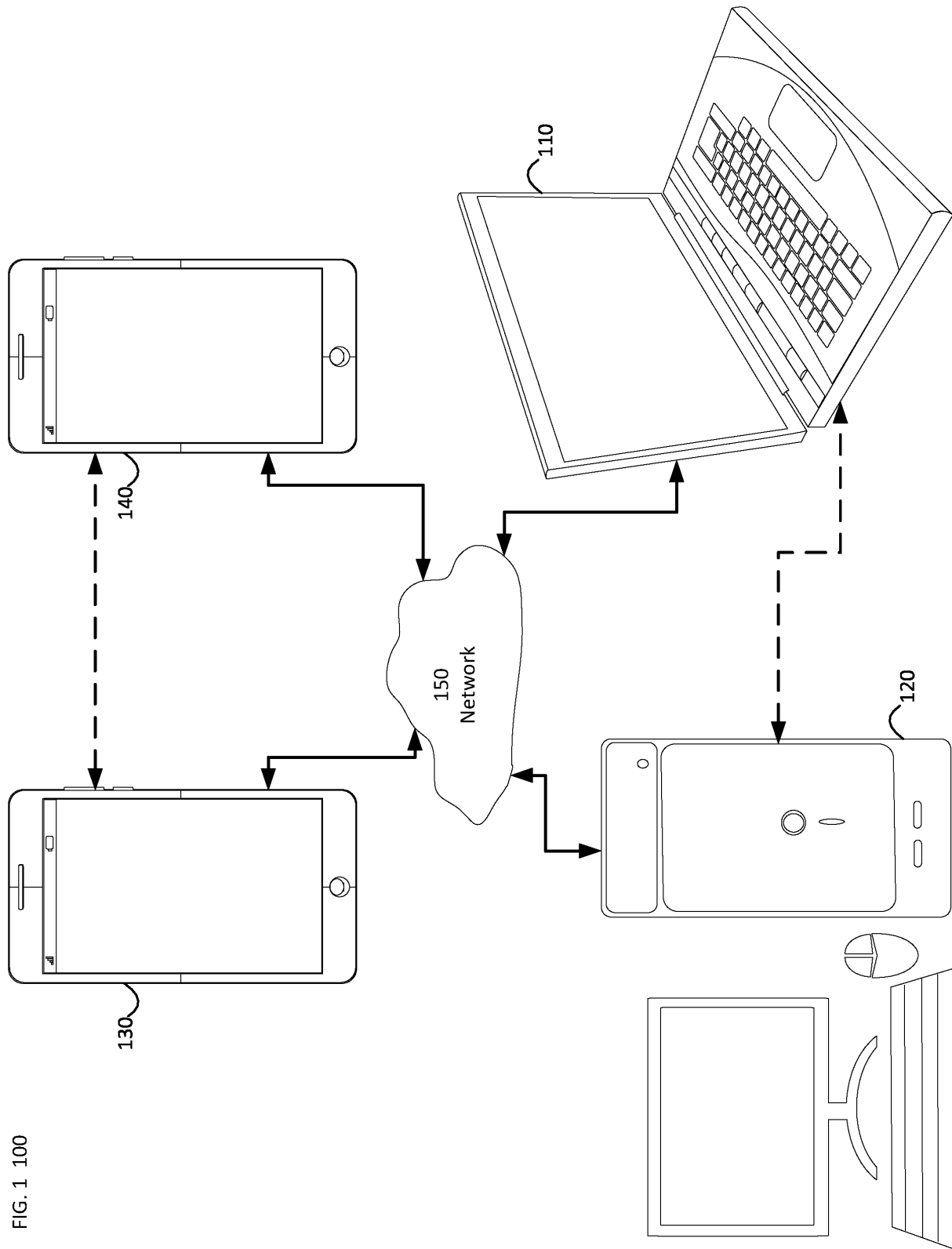


FIG. 1 100

FIG. 2

200

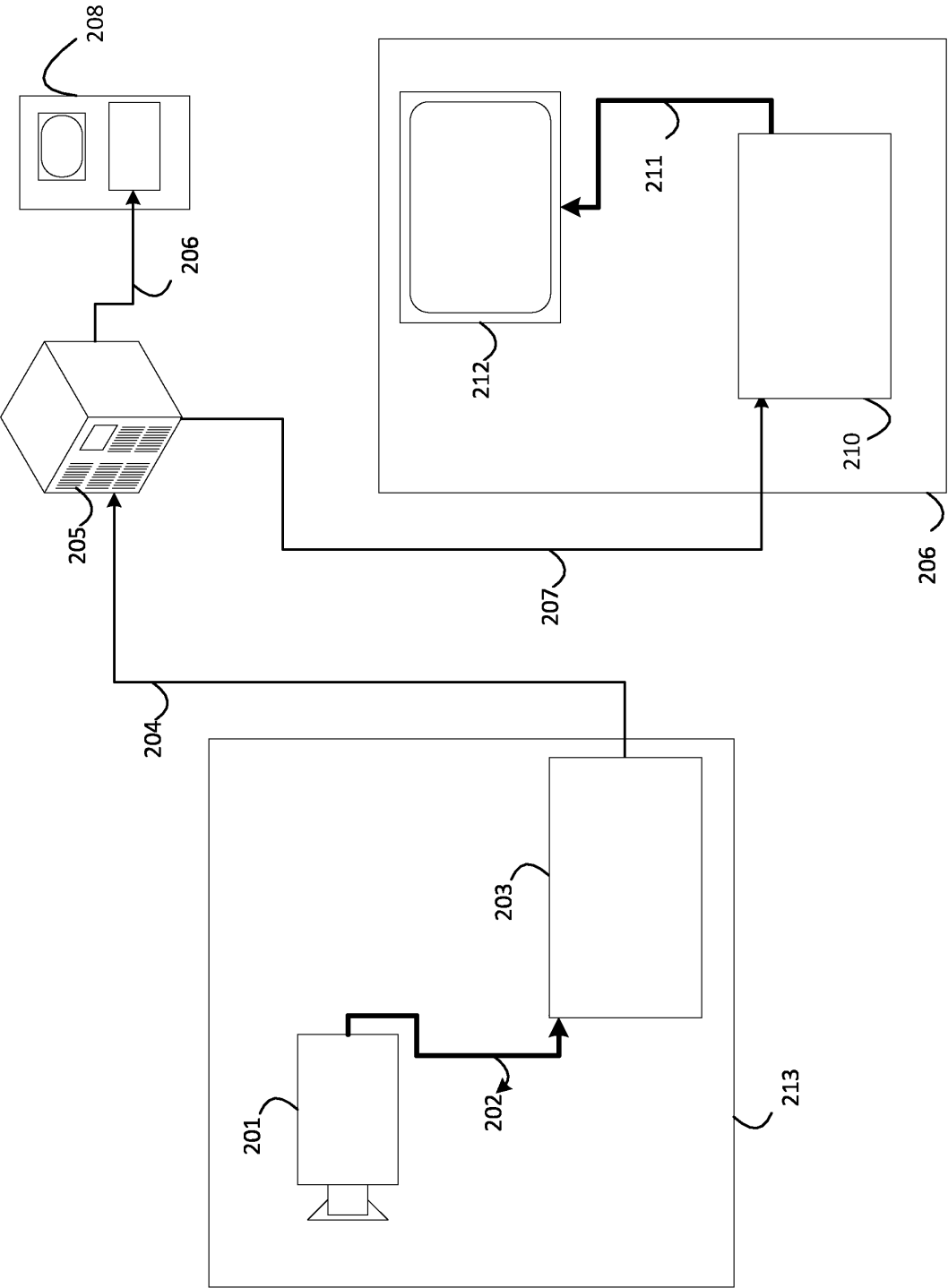


FIG. 3 Decoder 210

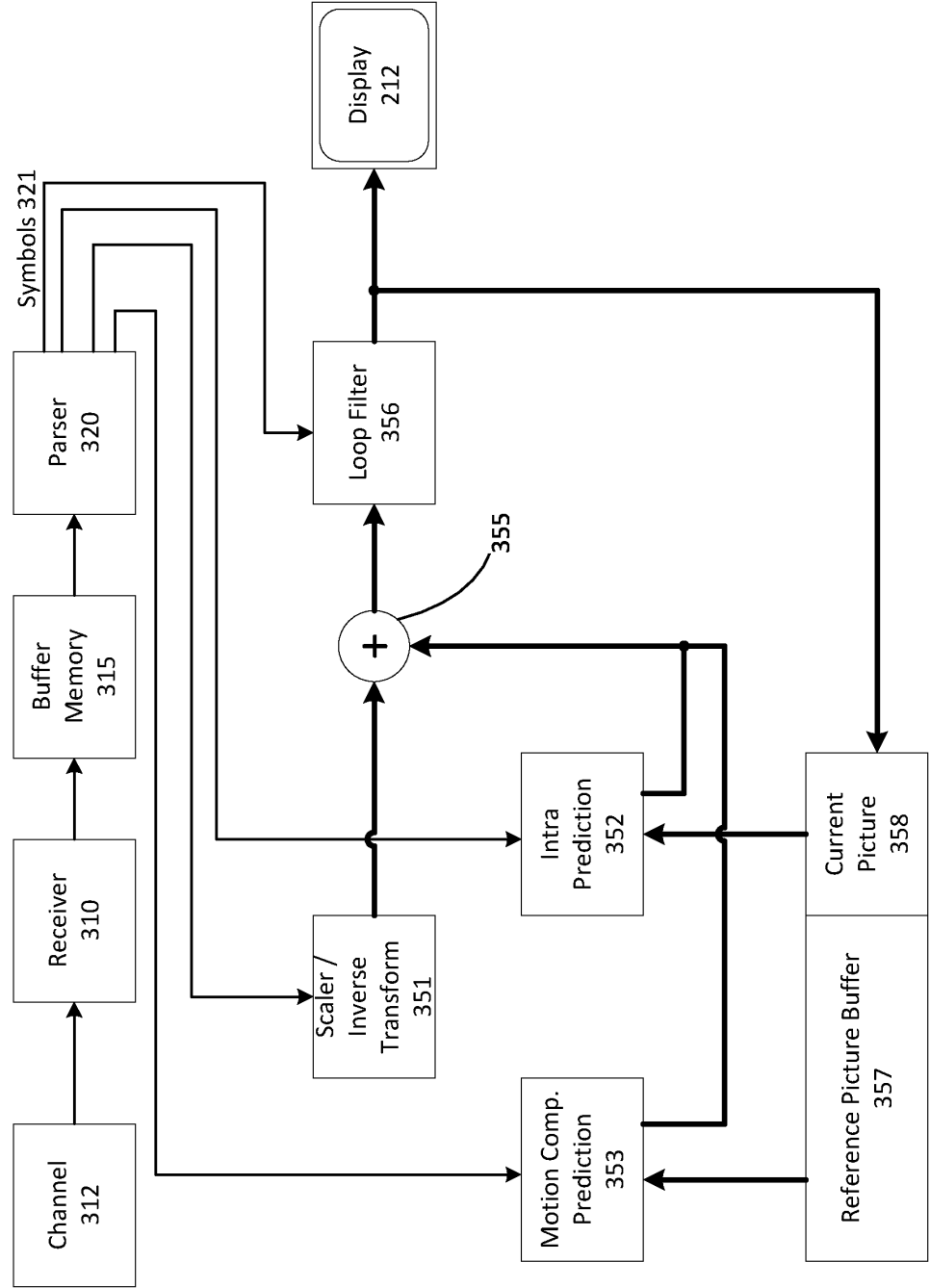
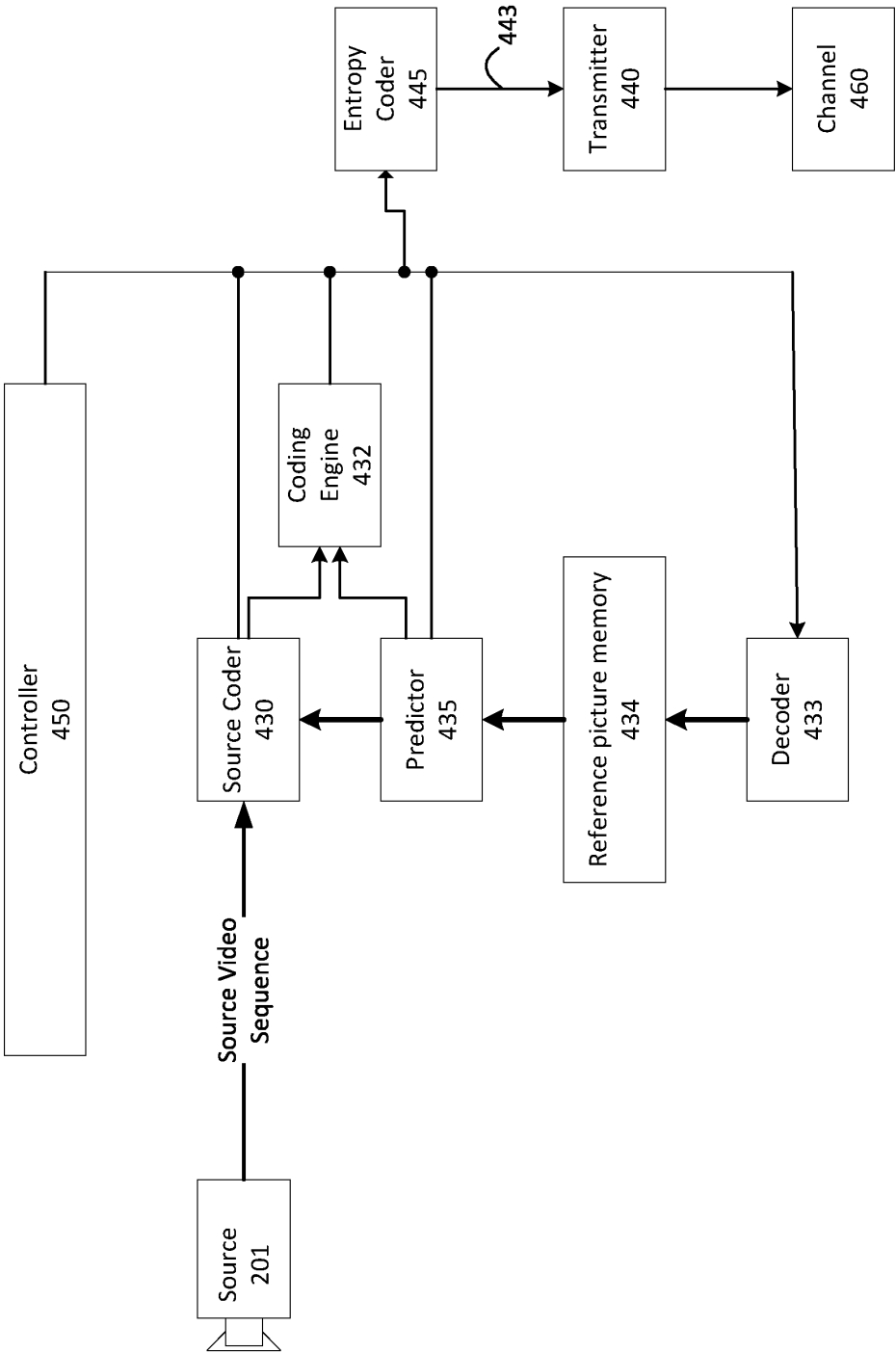


FIG. 4 Encoder 203



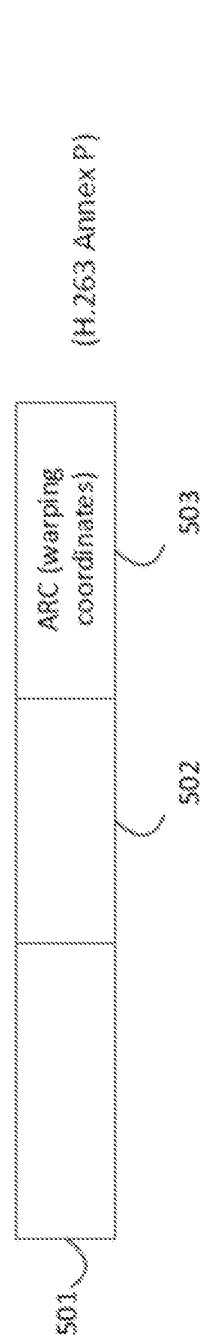


FIG. 5A

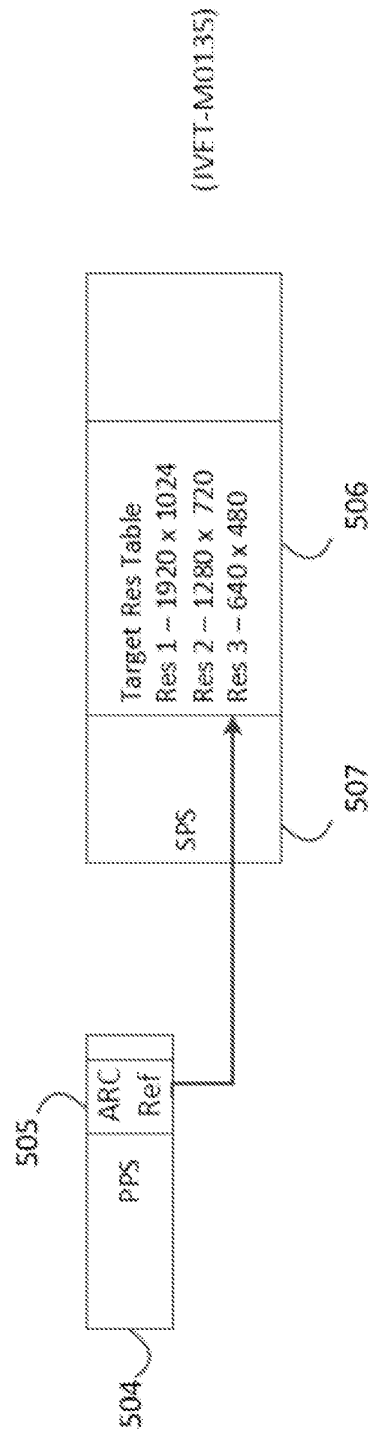


FIG. 5B

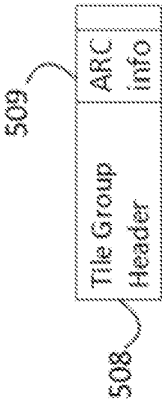


FIG. 5C

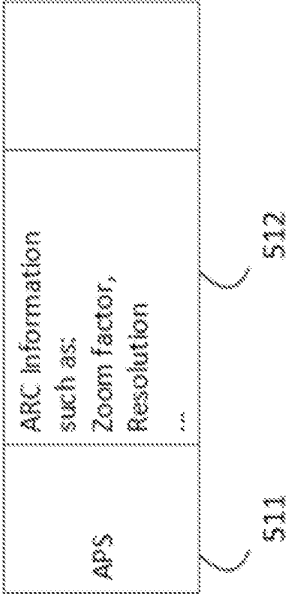


FIG. 5D

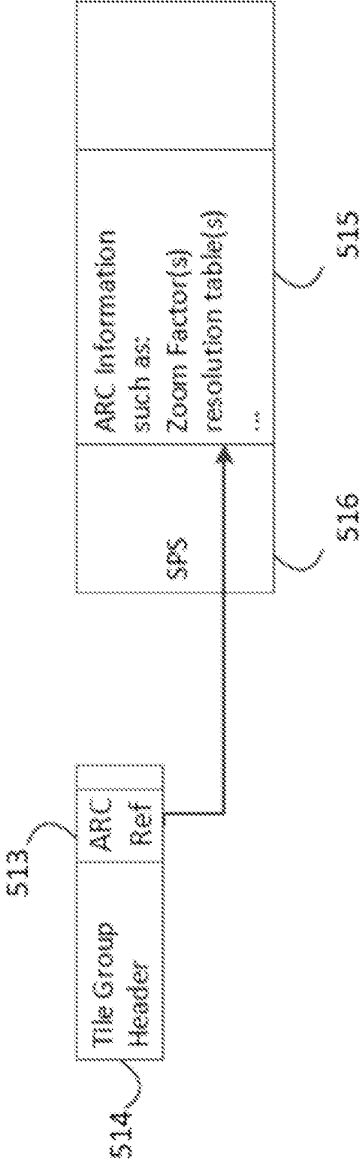


FIG. 5E

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

FIG. 6A

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)
	}	
616	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)
617	dec_pic_height_in_luma_samples[i]	ue(v)
	}	
	}	
	...	
	}	

FIG. 6B

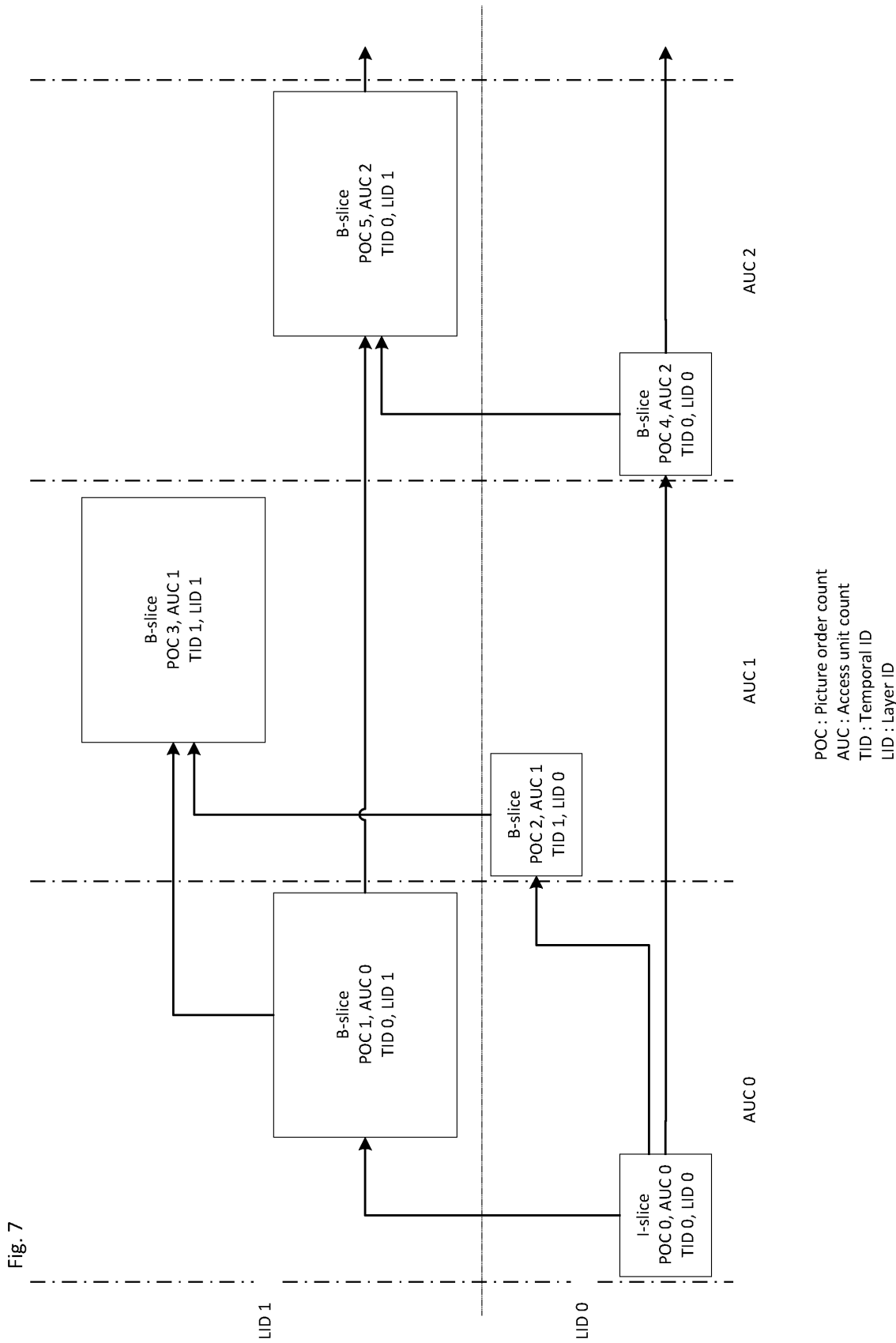


Fig. 8

<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(5)</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_au</code>	<code>u(1)</code>
<code> if(vps_constant_poc_cycle_per_au)</code>	<code></code>
<code> vps_poc_cycle_au</code>	<code>u(3)</code>
<code> ...</code>	<code></code>
<code> }</code>	<code></code>

<code> }</code>	Descriptor:
<code> slice_header() {</code>	<code>ue(v)</code>
<code> slice_pic_parameter_set_id</code>	<code></code>
<code> if(rect_slice_flag == NonBricksInPic > 1)</code>	<code></code>
<code> slice_address</code>	<code>u(v)</code>
<code> if(!rect_slice_flag && !single_brick_per_slice_flag)</code>	<code></code>
<code> num_bricks_in_slice_minus1</code>	<code>ue(v)</code>
<code> slice_type</code>	<code>ue(v)</code>
<code> if(NaiUnitType == GRA_NUT)</code>	<code></code>
<code> recovery_poc_cnt</code>	<code>se(v)</code>
<code> slice_pic_order_cnt_lsb</code>	<code>u(v)</code>
<code> ...</code>	<code></code>
<code> if(vps_constant_poc_cycle_per_au)</code>	<code></code>
<code> slice_poc_cycle_au</code>	<code>u(3)</code>
<code> }</code>	<code></code>
<code> }</code>	<code></code>
<code> }</code>	<code></code>

Fig. 9

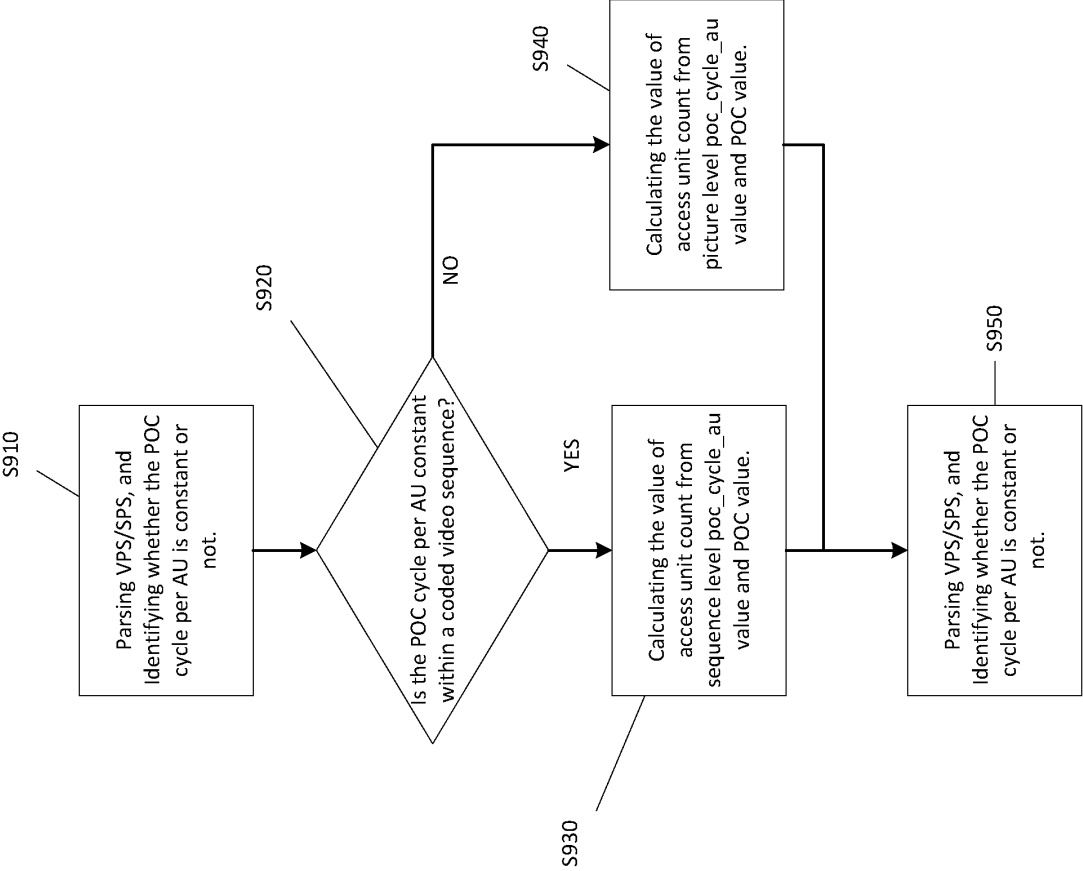
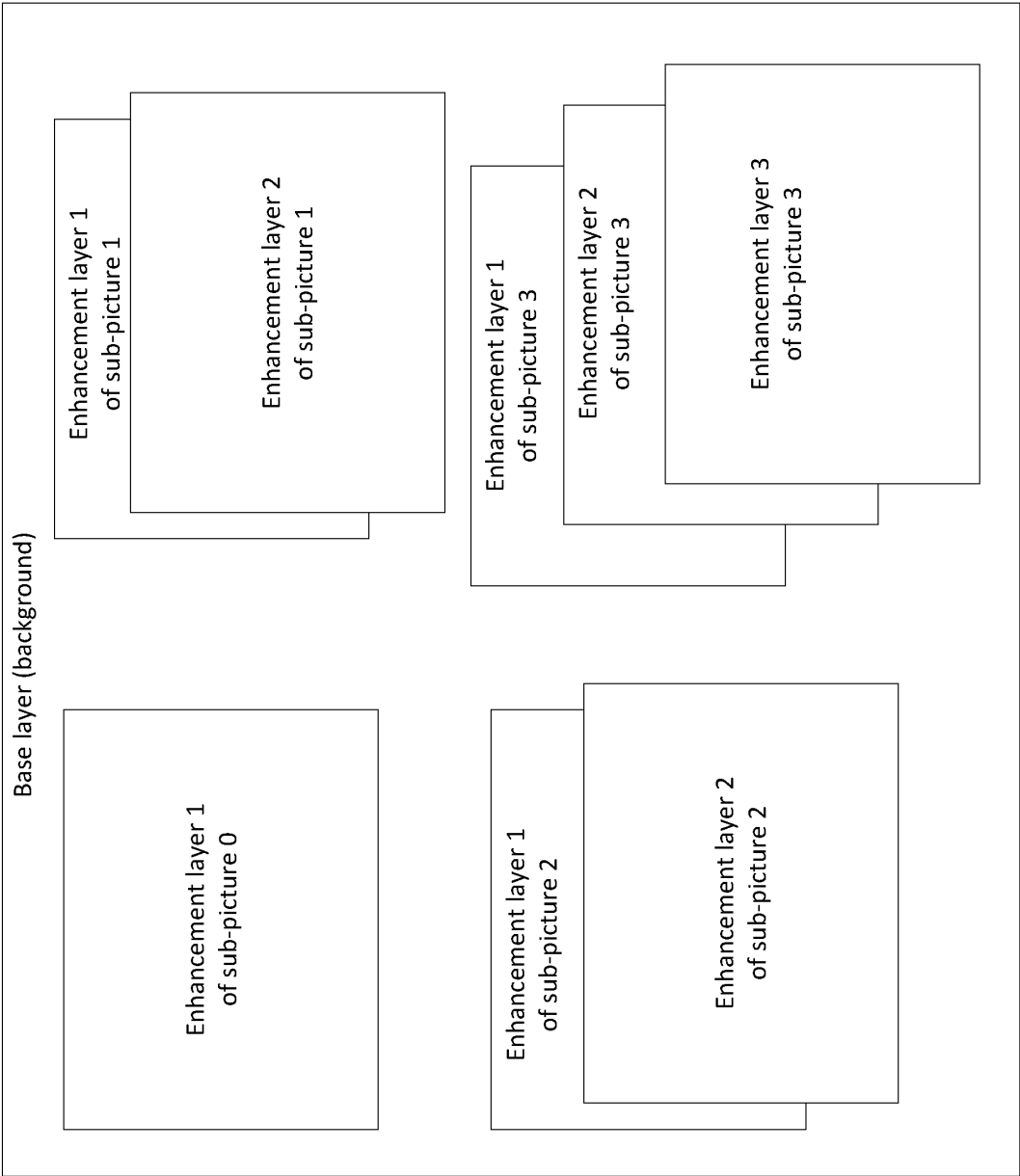


FIG. 10



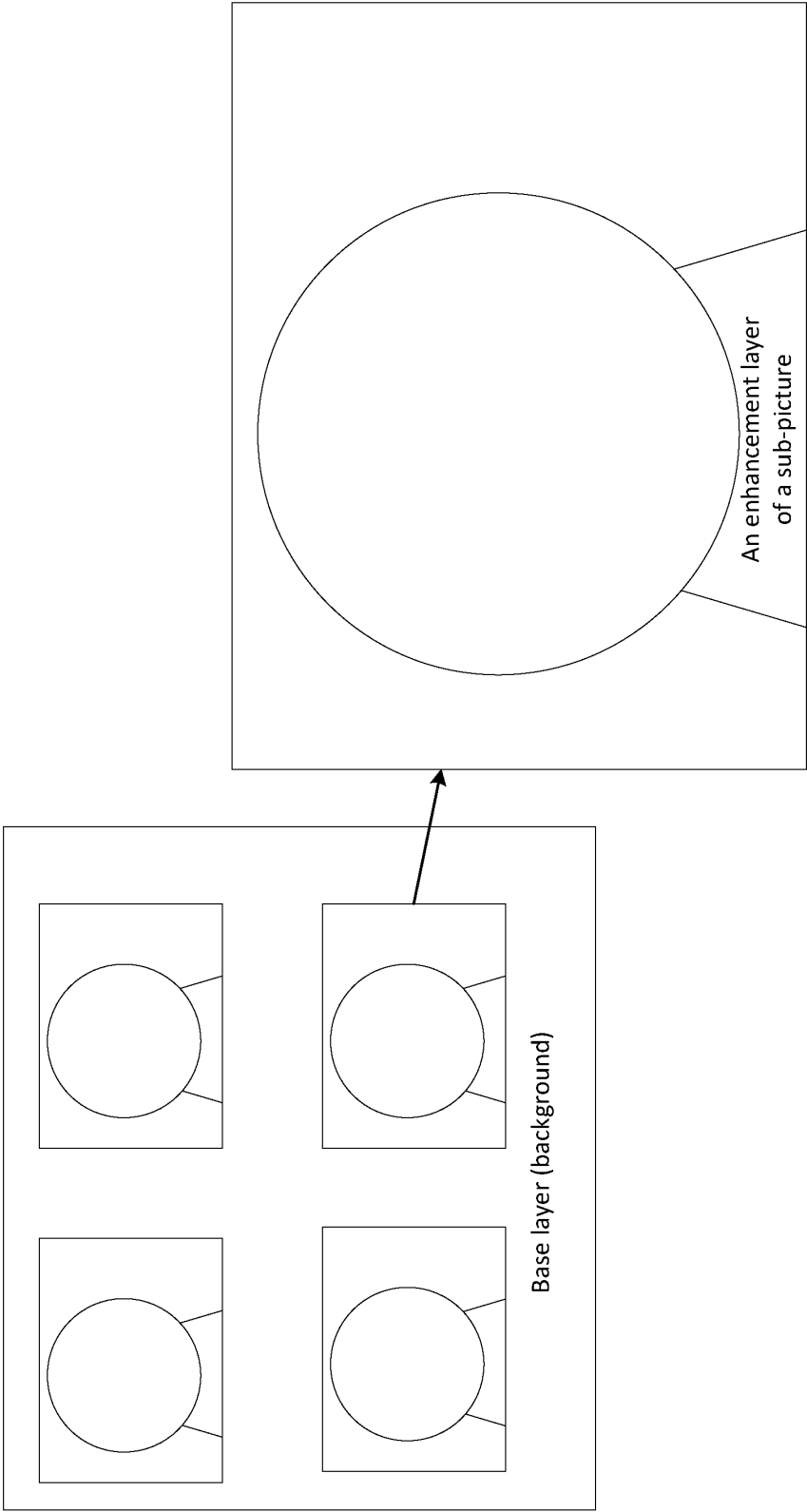


FIG. 11

Fig. 12

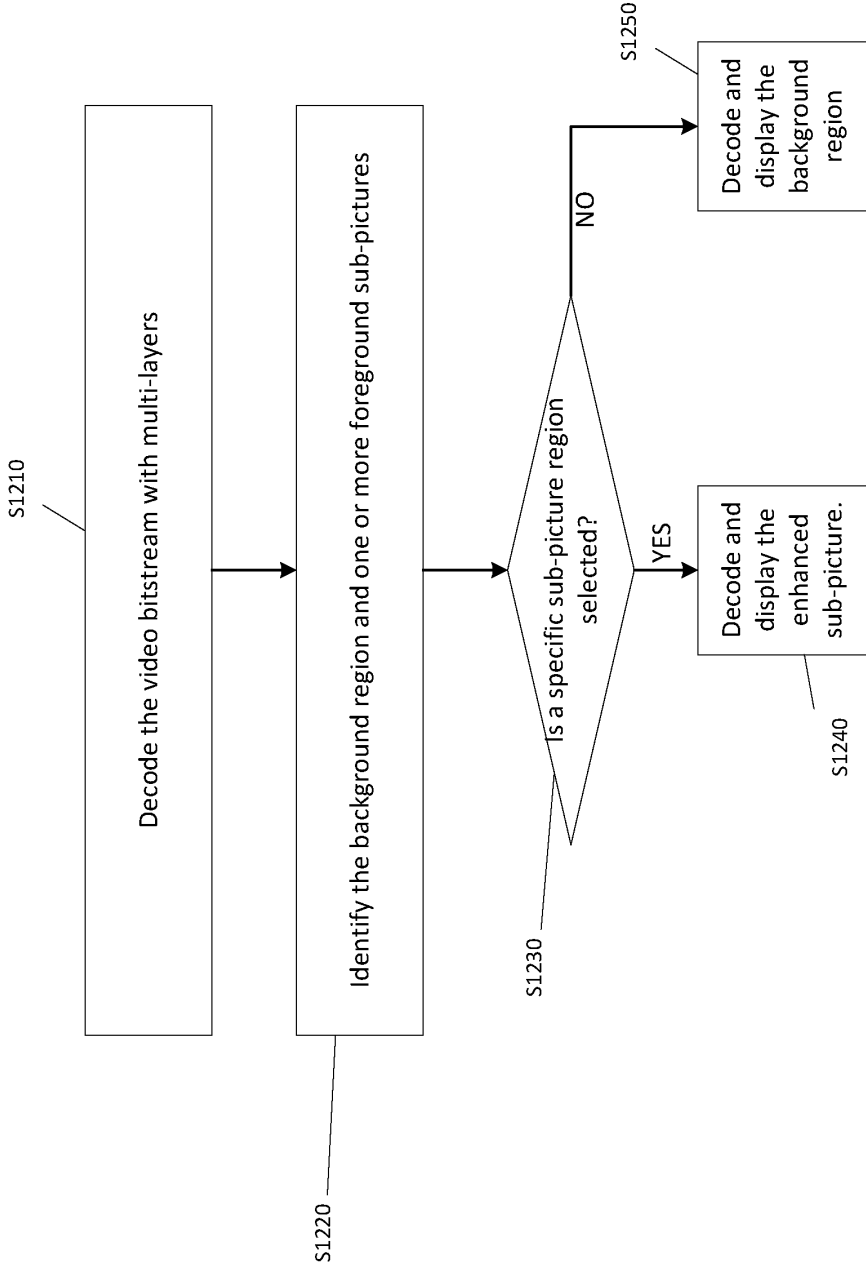


Fig. 13

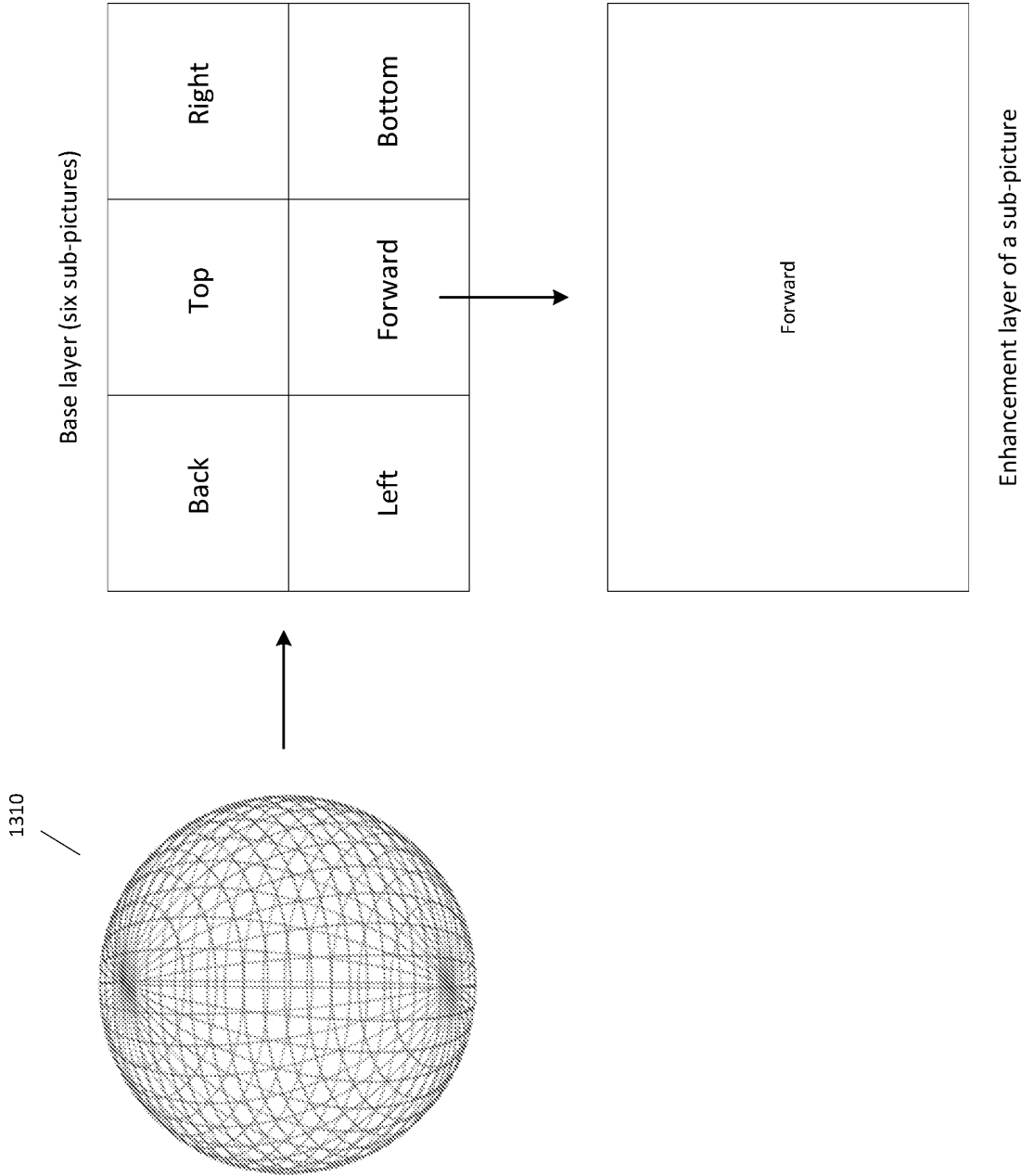
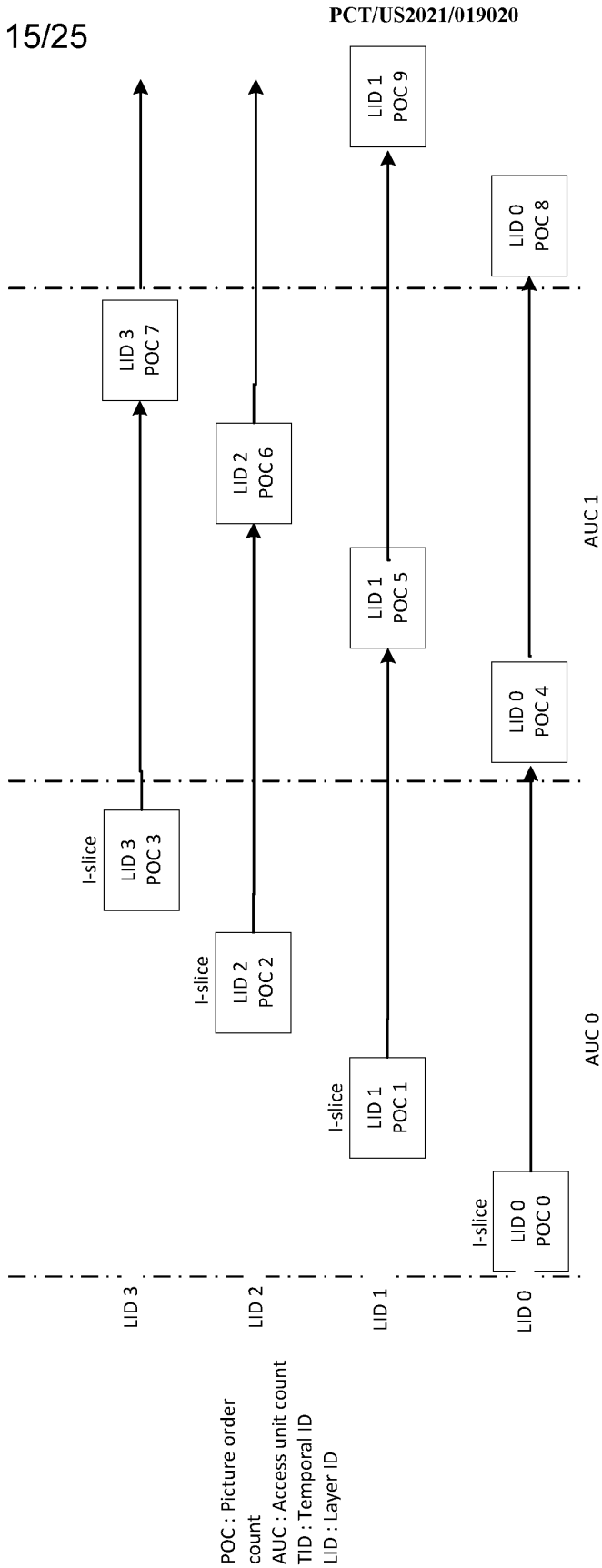
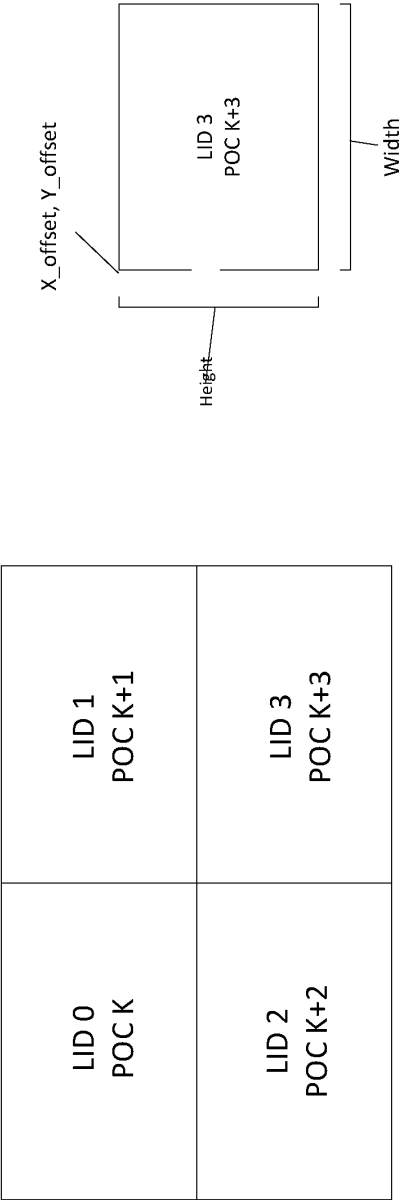


Fig. 14



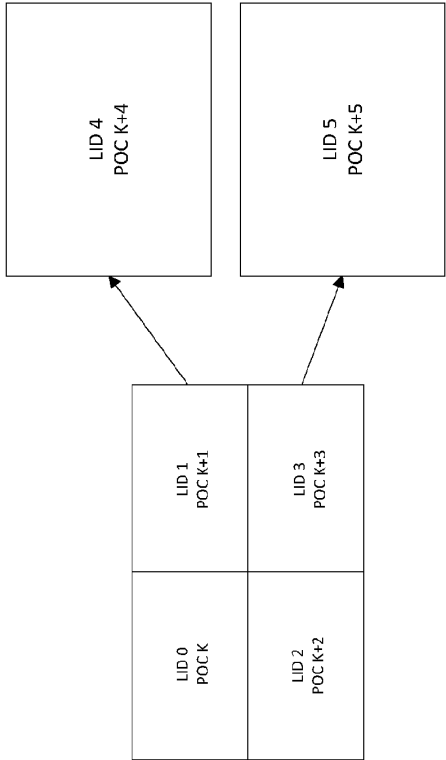
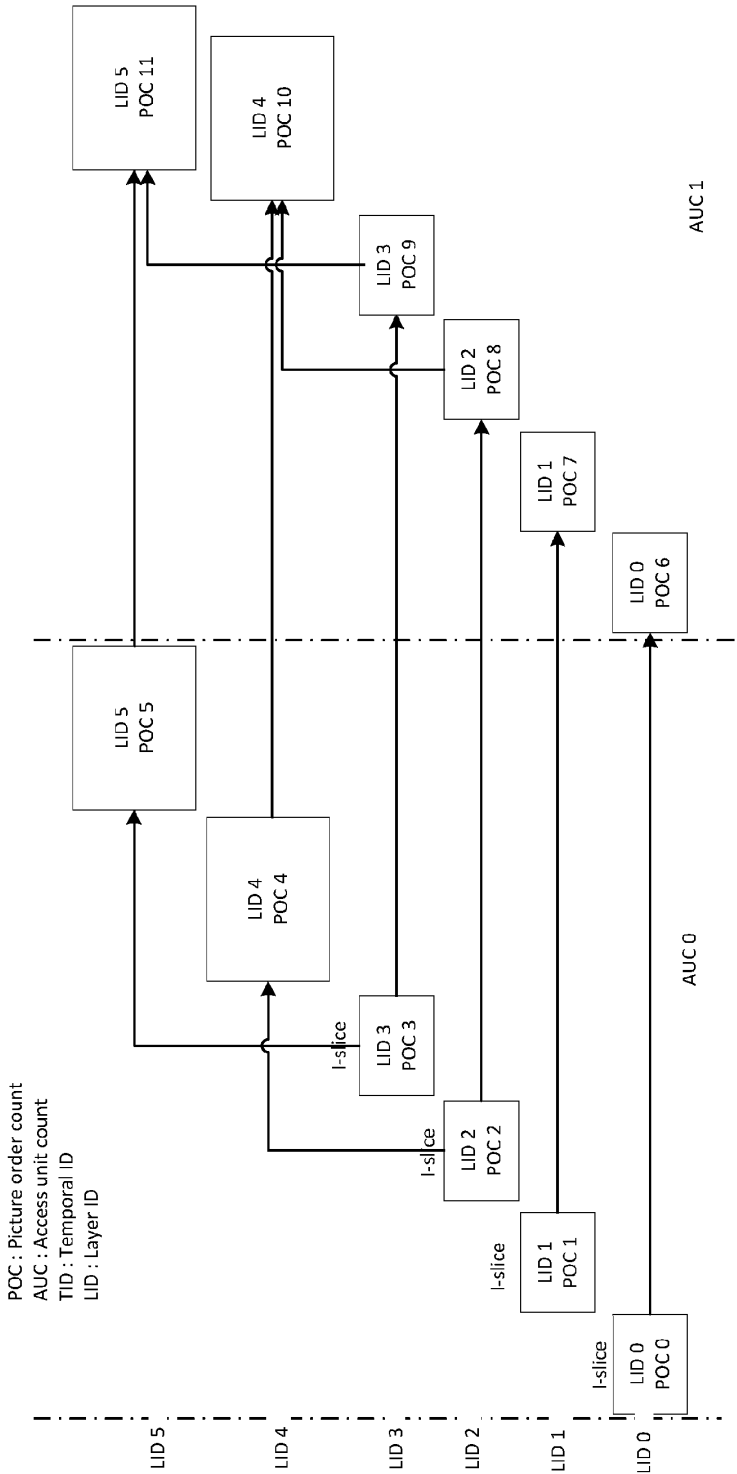


Fig. 15



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_sub_picture_dividing_flag; if(vps_sub_picture_dividing_flag) { vps_full_pic_width_in_luma_samples; vps_full_pic_height_in_luma_samples; } ... }
--

FIG. 16A

seq_parameter_set_rbsp() { seq_decoding_parameter_set_id; seq_video_parameter_set_id; sps_max_sub_layers_minus1; ... pic_width_in_luma_samples; pic_height_in_luma_samples; if(vps_sub_picture_dividing_flag) { pic_offset_x; pic_offset_y; } ... }

FIG. 16B

18/25

Fig. 17

sub_region_partitioning_info(payloadSize) { ²	Descriptor ²
num_sub_region ²	u8(v) ²
num_layers ²	u8(v) ²
for(i = 0; i <= num_layers; i++) ²	²
layer_id [i] ²	u8(v) ²
for(i = 1; i < num_layers; i++) ²	²
for(j = 0; j < i; j++) 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	u8(v) ²
for(j = 0; j < num_layers_for_region[i]; j++) ²	²
sub_region_layer_id[i][j] ²	u8(v) ²
sub_region_offset_x[i] ²	u8(v) ²
sub_region_offset_y[i] ²	u8(v) ²
sub_region_width[i] ²	u8(v) ²
sub_region_height[i] ²	u8(v) ²
} ²	²
... ²	²
} ²	²

Fig. 18

	Descriptor
video_parameter_set_rsp() {	
...	
vps_max_layers_minus1	u(G)
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(I)
profile_tier_level_idx[i][j]	u(V)
}	
...	
}	

Fig. 19

	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 < 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

	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[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_ids[i][j]	u(v)
}	
}	
...	
}	

Fig. 21

pic_parameter_set_rbsp() {		Descriptor
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_subpics_minus1; i++)		
pps_subpic_id[i]		u(v)
}		
...		
}		

seq_parameter_set_rbsp() {	Descriptor
...	
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_huma_samples > CbSizeY)	
subpic_ctu_top_left_x[i]	u(v)
if(i > 0 && pic_height_max_in_huma_samples > CbSizeY) {	
subpic_ctu_top_left_y[i]	u(v)
if(i < sps_num_subpics_minus2 + 1 && pic_width_max_in_huma_samples > CbSizeY)	
subpic_width_minus1[i]	u(v)
if(i < sps_num_subpics_minus2 + 1 && pic_height_max_in_huma_samples > CbSizeY)	
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. 22

FIG. 23 2300

