

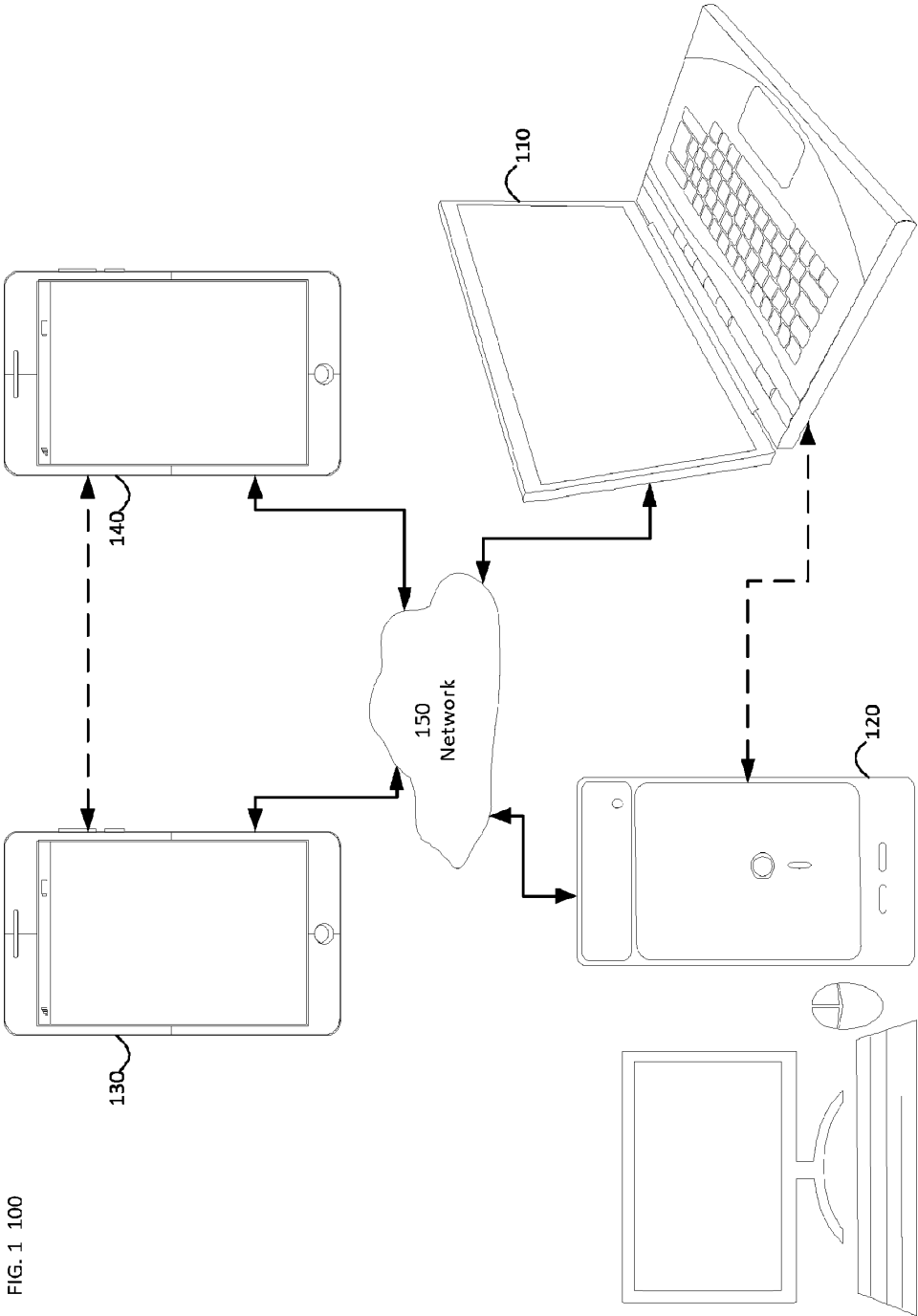


FIG. 1 100

FIG. 2

200

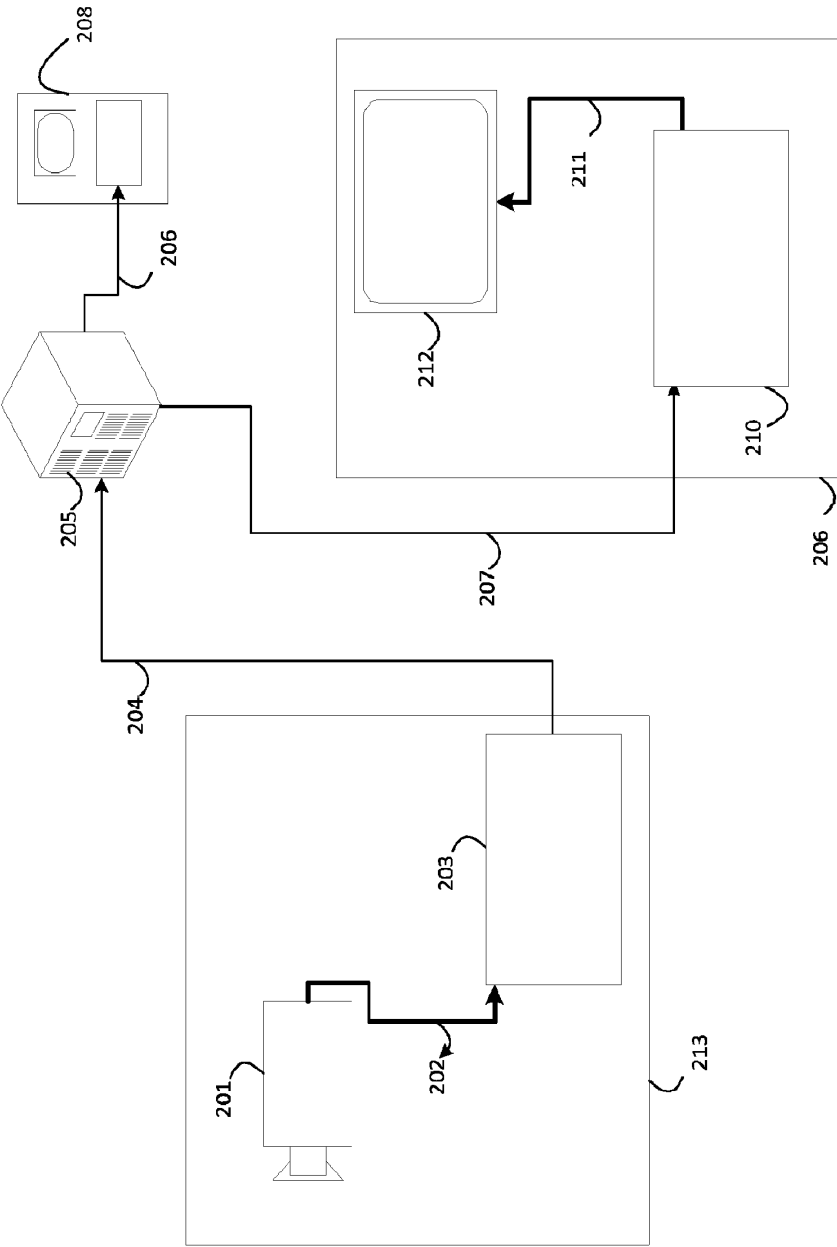


FIG. 3 Decoder 210

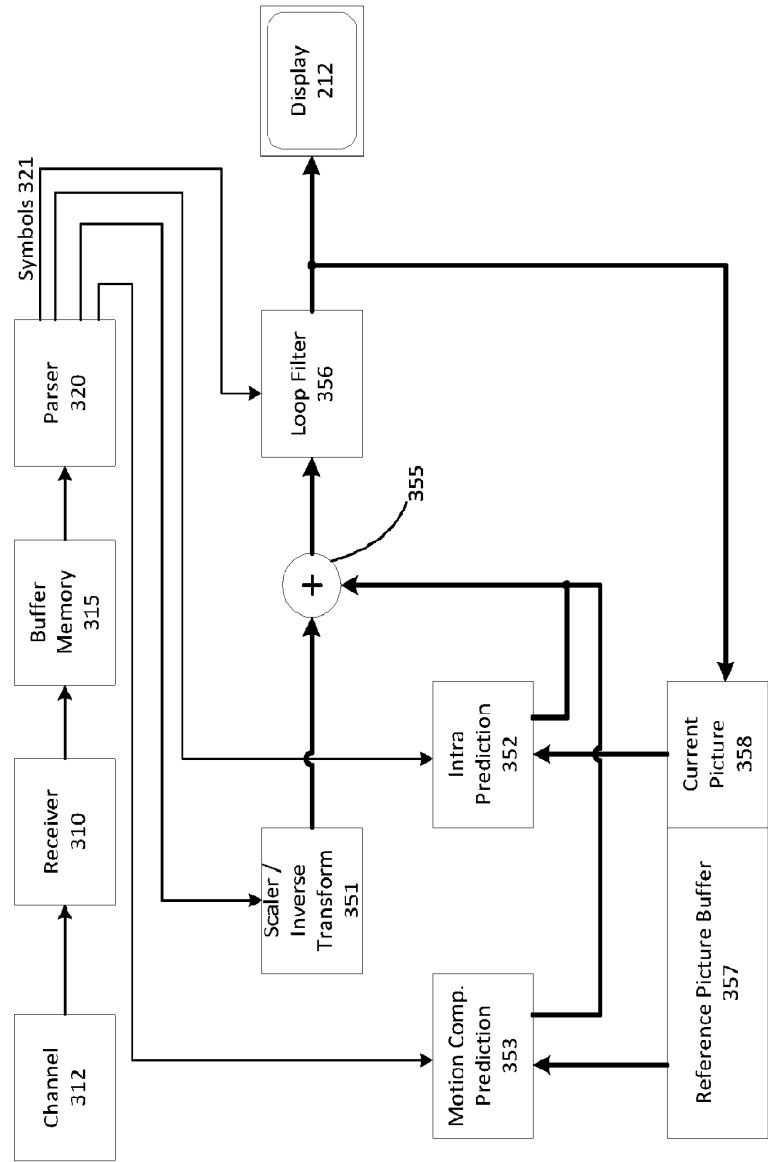
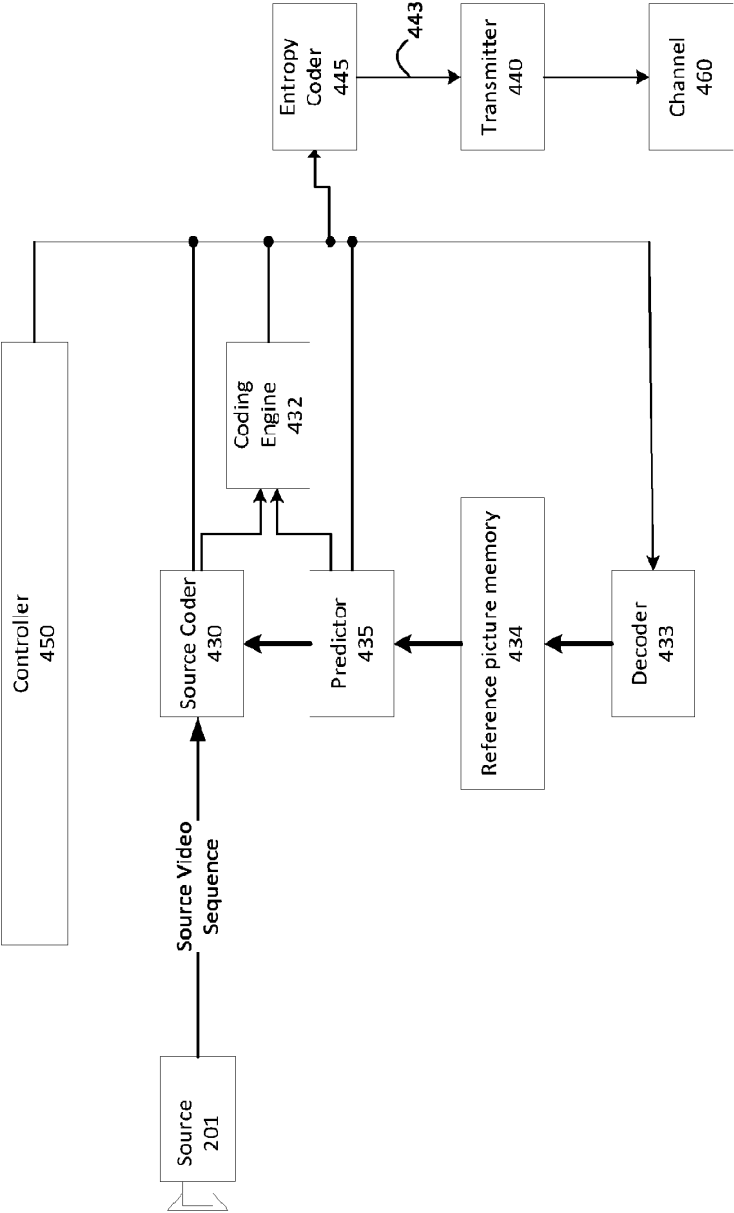
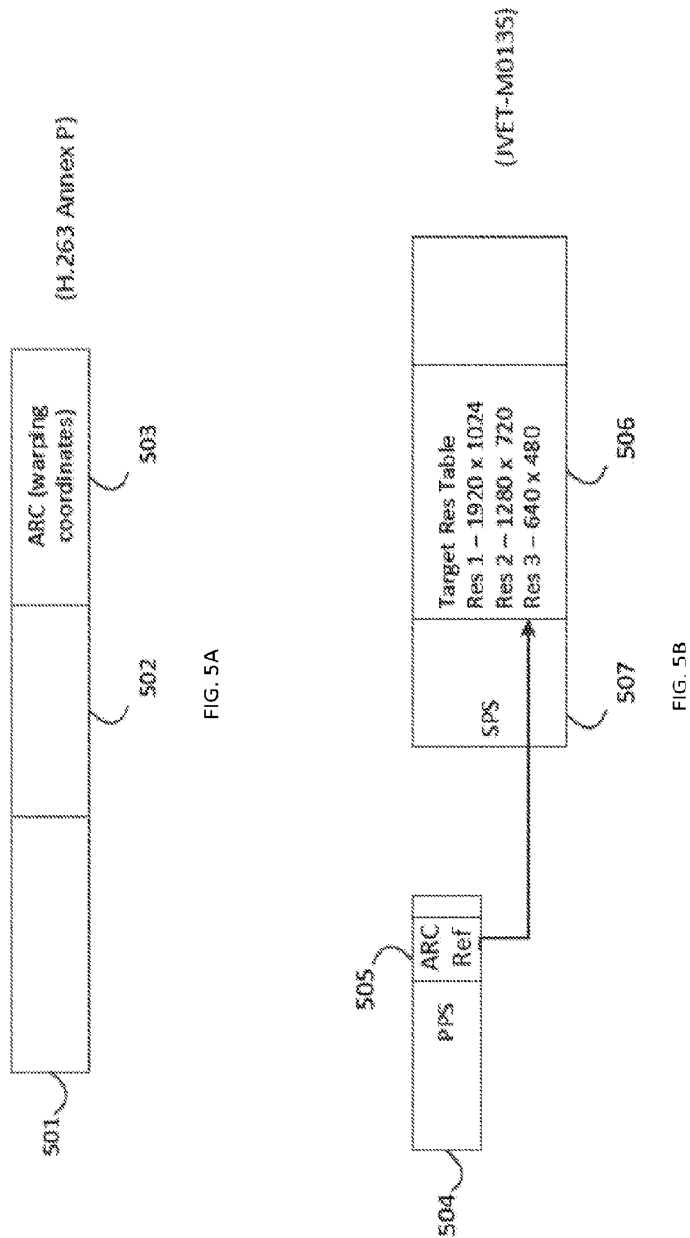


FIG. 4 Encoder 203





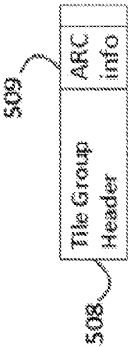


FIG. 5C

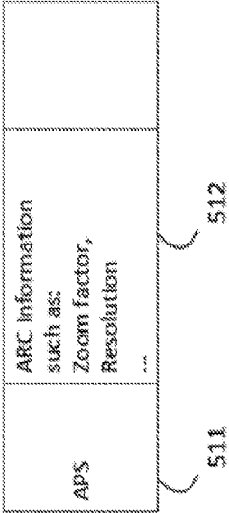


FIG. 5D

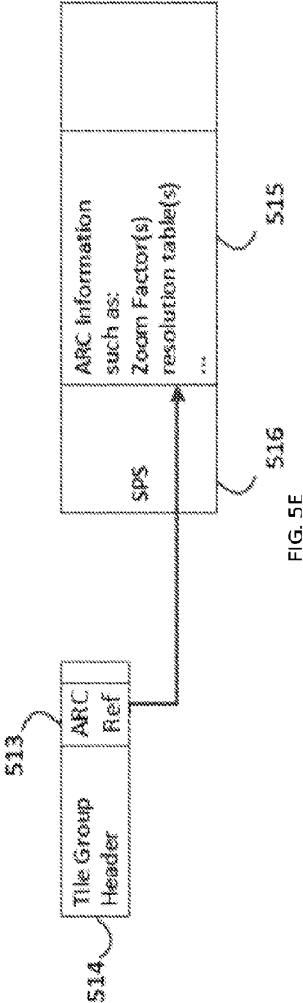


FIG. 5E

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) {	
613	output_pic_width_in_luma_samples	ue(v)
	output_pic_height_in_luma_samples	ue(v)
614	reference_pic_size_present_flag	u(1)
	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

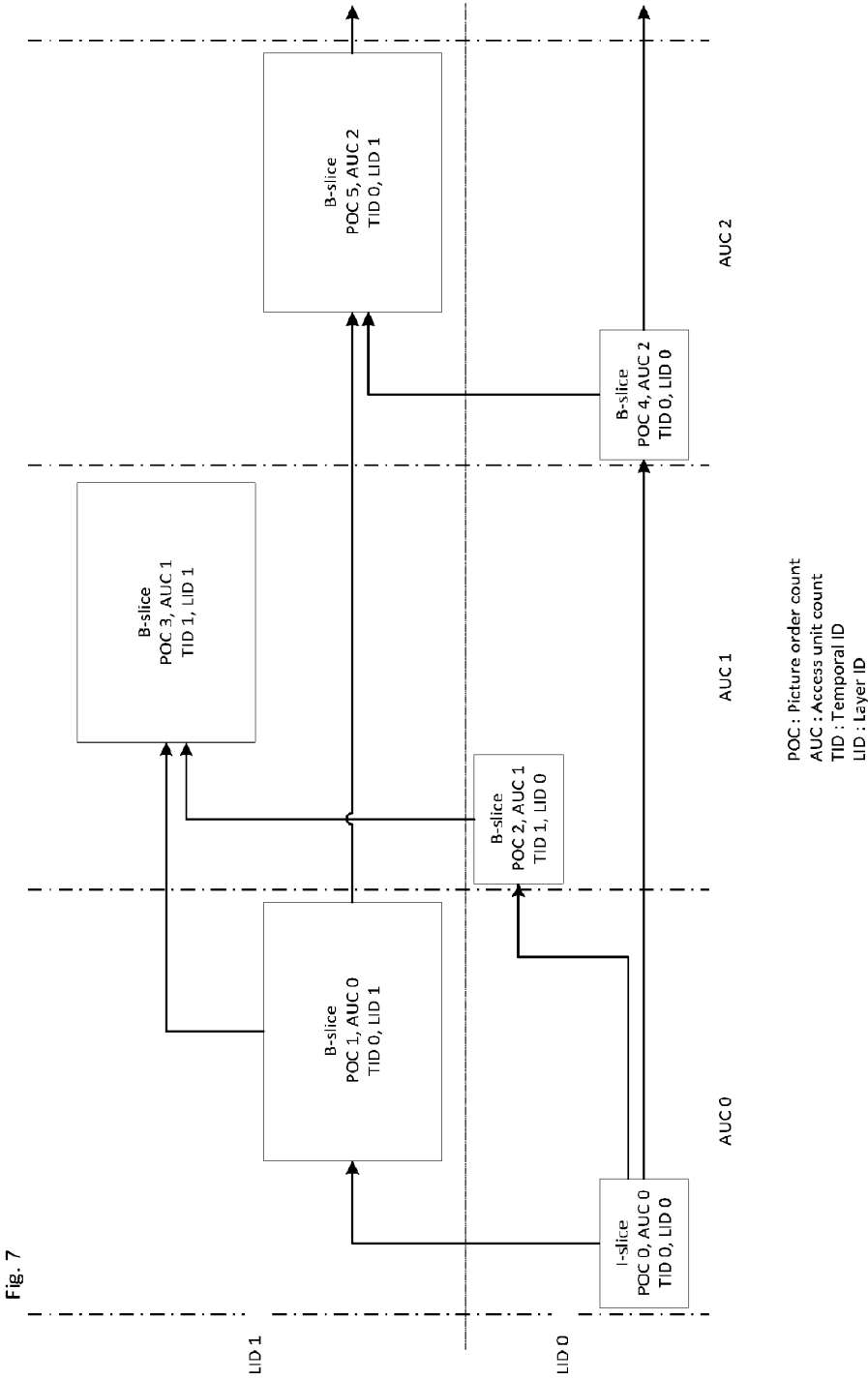


Fig. 8

video_parameter_set_id; { 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_aub; if(vps_constant_poc_cycle_per_aub) vps_poc_cycle_aub; ...; }	Descriptor: u(4) u(8) s u(7) u(1) s u(1) u(1) s u(8) s s s
slice_header() { slice_pic_parameter_set_id; if (set slice_flag == NumBricksInPic > 1) { slice_address; if (rect_slice_flag && (single_brick_per_slice_flag)) num_bricks_in_slice_minus1; slice_type; if (NalUnitType == GRA_NUT) { recovery_poc_cnt; slice_pic_order_lsb; ...; if(vps_constant_poc_cycle_per_aub) slice_poc_cycle_aub; ... }	Descriptor: ue(v) s u(v) s ue(v) ue(v) s ue(v) u(v) s u(8) s s s

Fig. 9

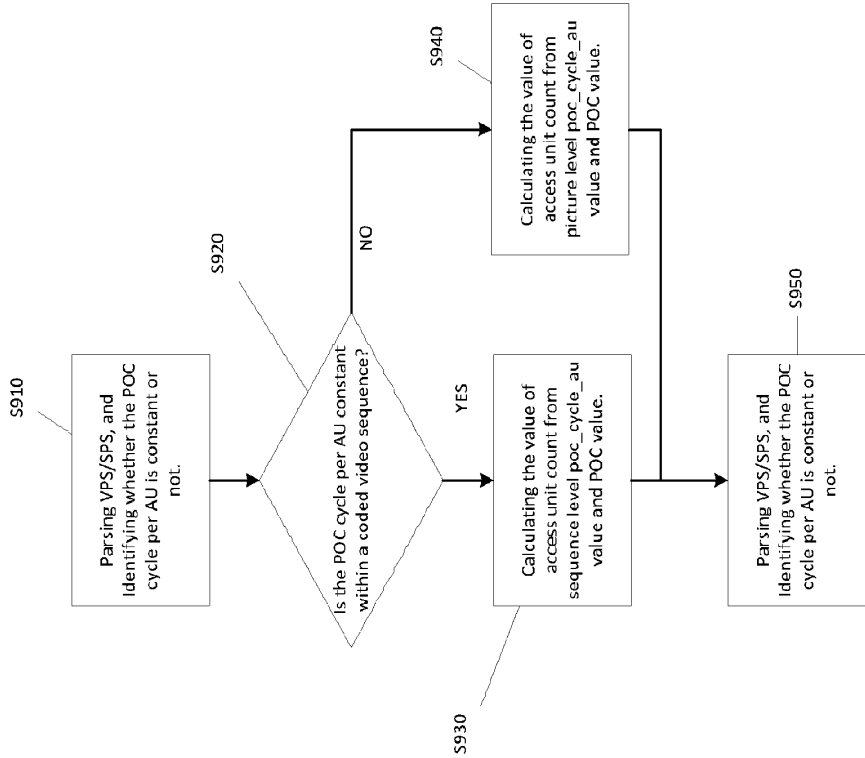
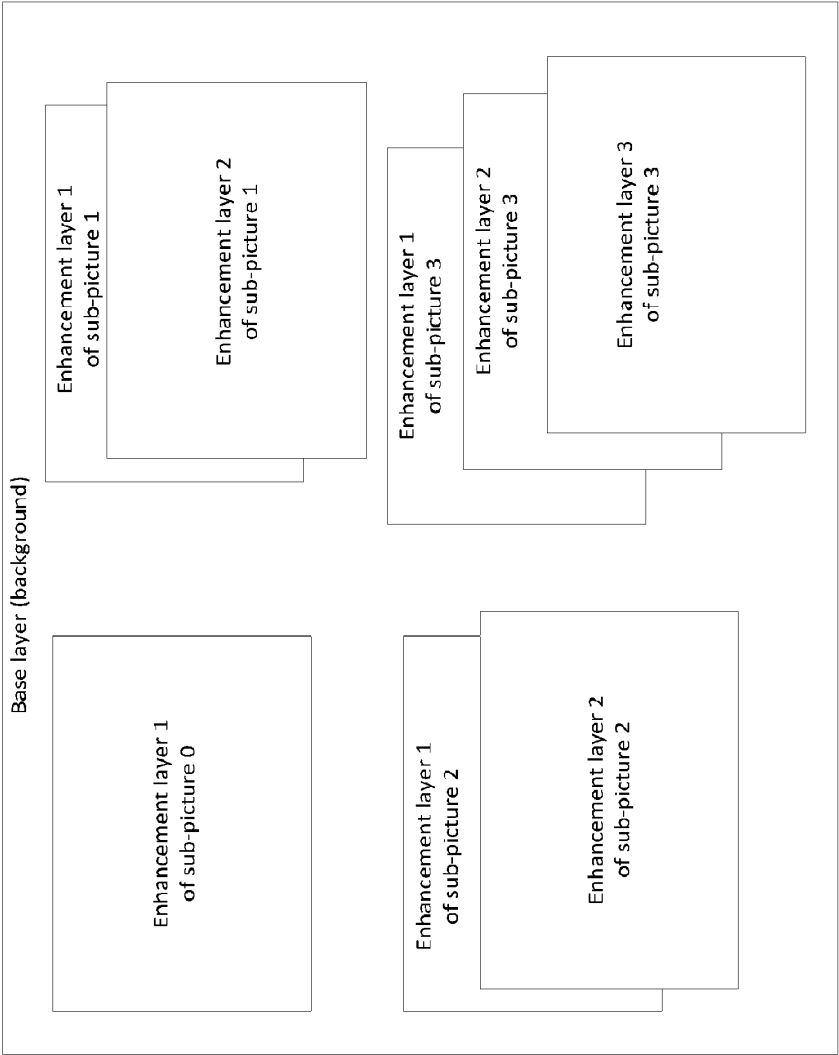
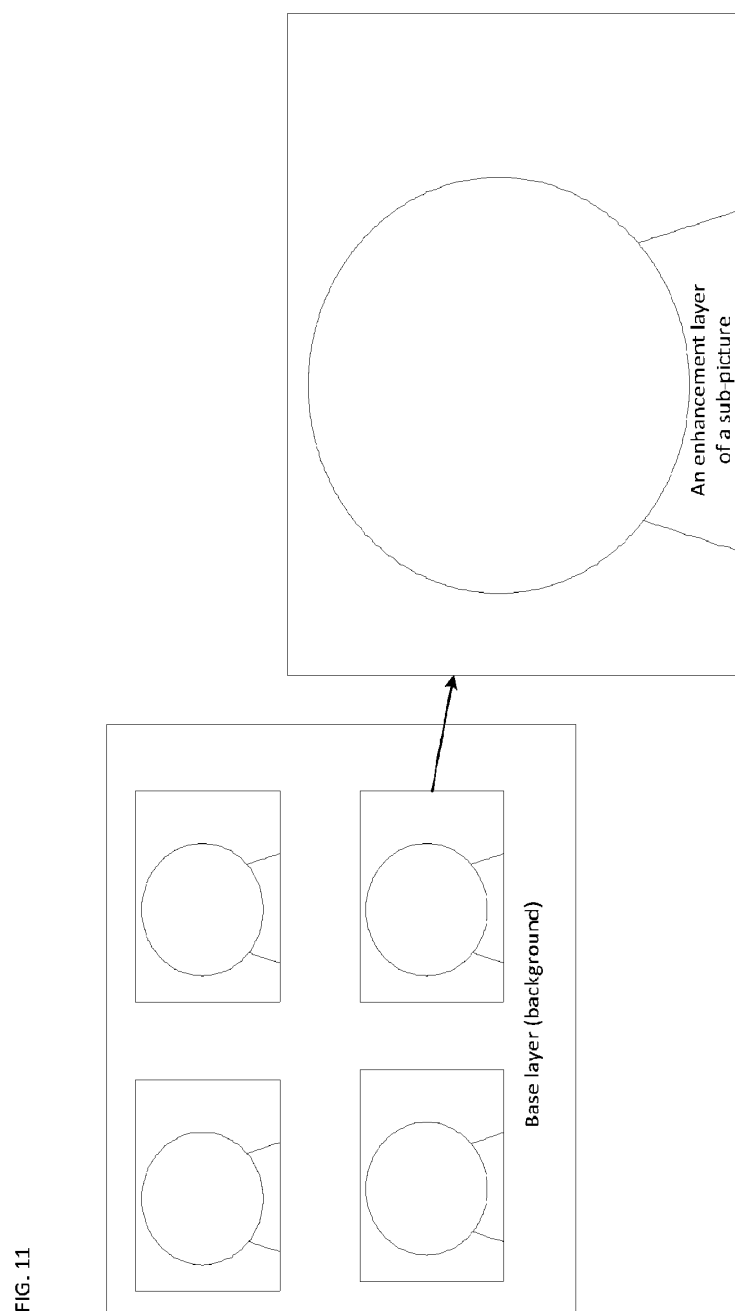


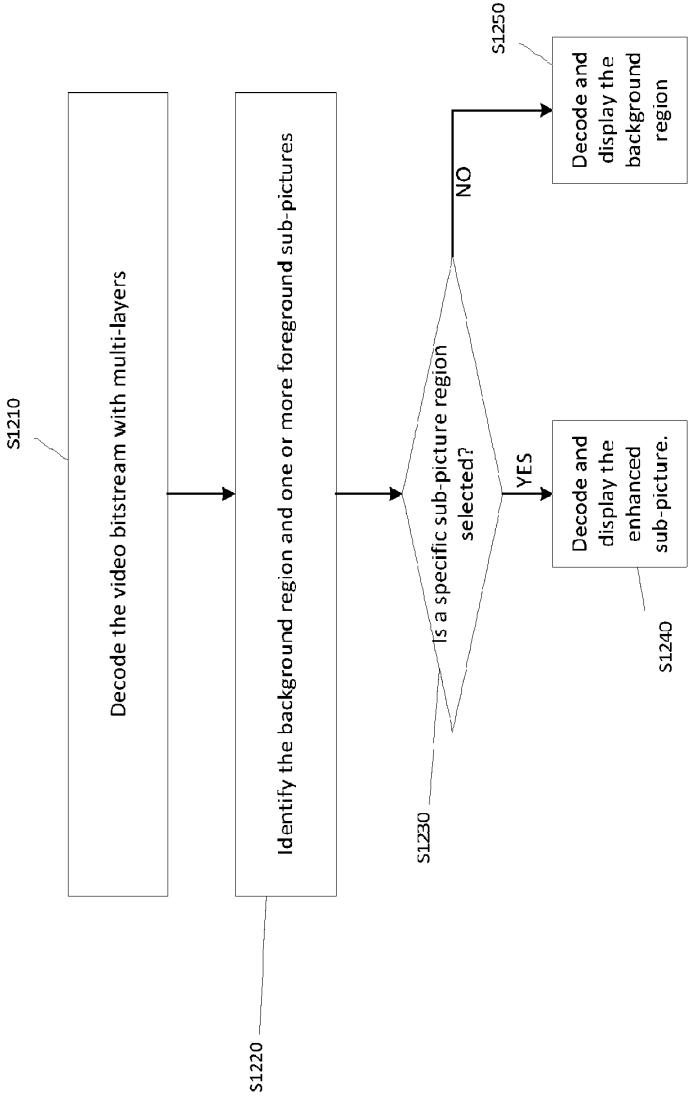
FIG. 10

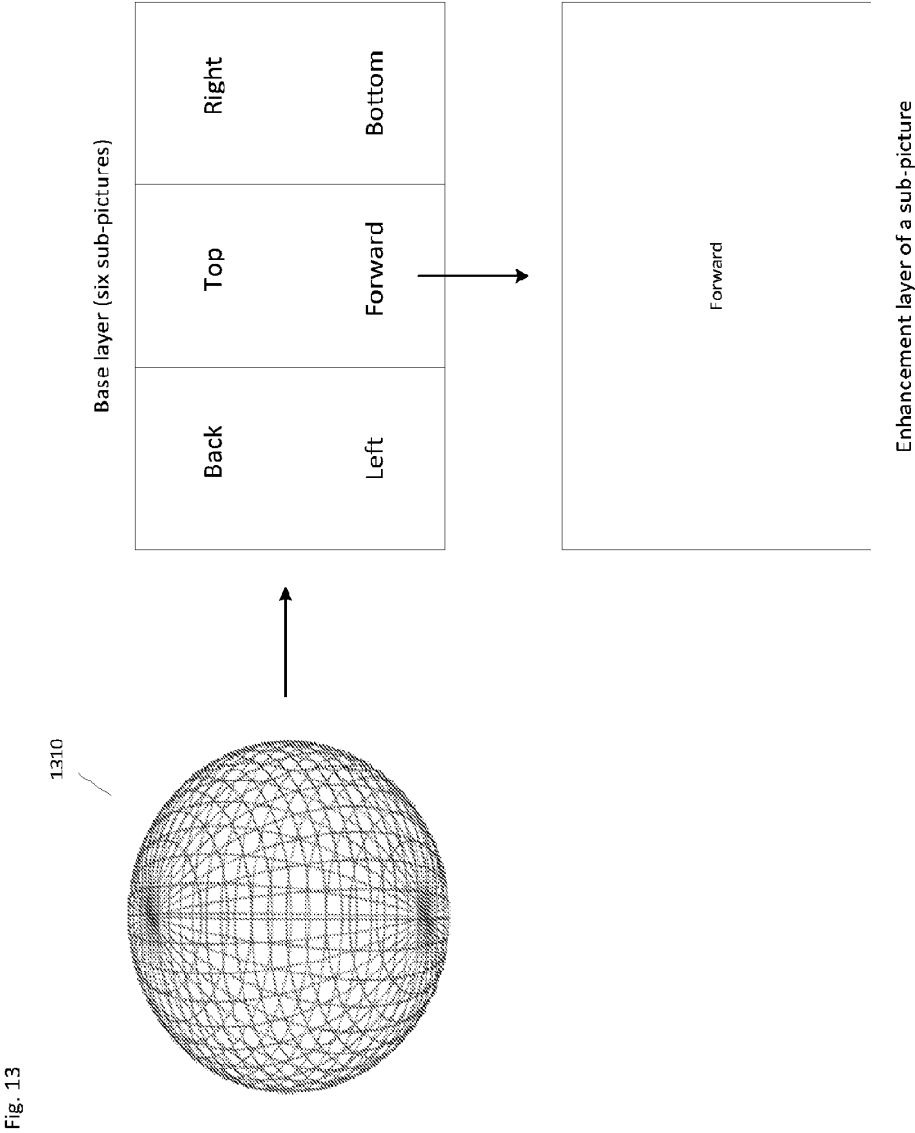


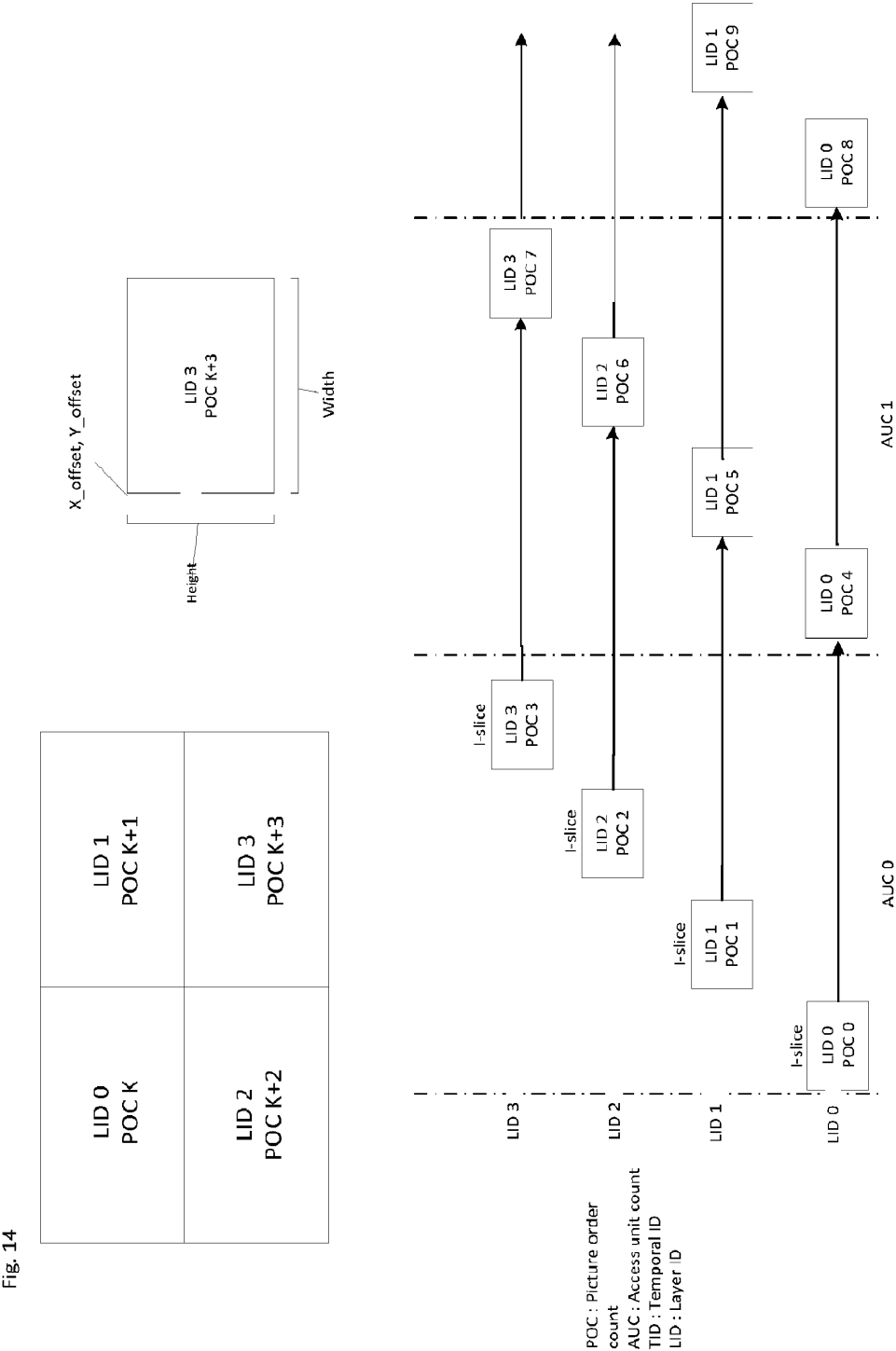


SUBSTITUTE SHEET (RULE 26)

Fig. 12







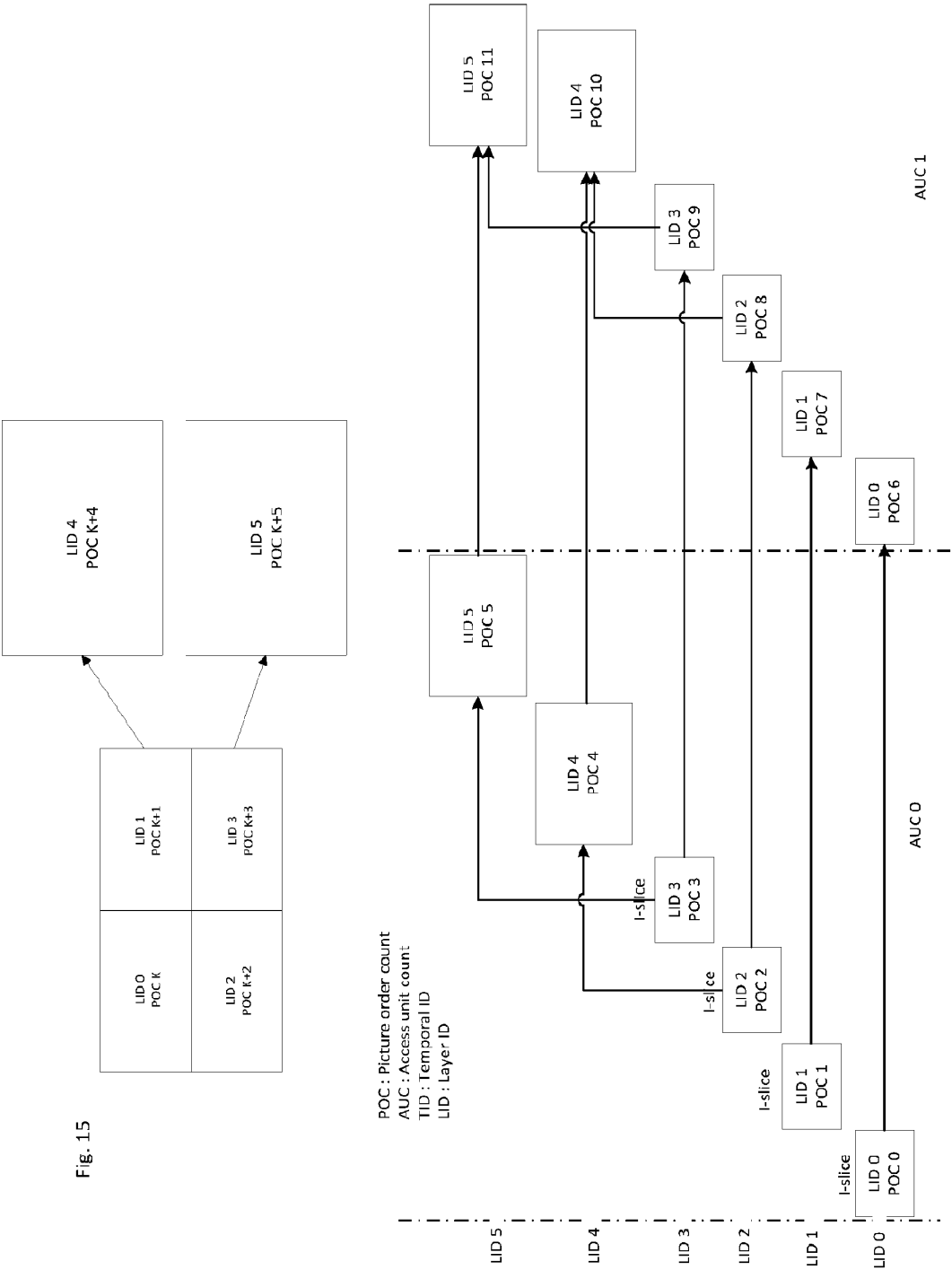


Fig. 15

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

FIG. 16A

	Descriptor
*seq_parameter_set_id { ^o	u(4) ^o
sps_decoding_parameter_set_id { ^o	u(4) ^o
sps_video_parameter_set_id { ^o	u(3) ^o
sps_max_sub_layers_minus1 { ^o	v ^o
... ^o	ue(v) ^o
pic_width_in_luma_samples { ^o	ue(v) ^o
pic_height_in_luma_samples { ^o	ue(v) ^o
if(vps_sub_picture_dividing_flag) { ^o	v ^o
pic_offset_x { ^o	ue(v) ^o
pic_offset_y { ^o	ue(v) ^o
} ^o	v ^o
... ^o	v ^o
} ^o	v ^o

FIG. 16B

Fig. 17

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

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 < NumLayersInProfileTierLevel(i); j++) {	
output_layer_flag[i][j]	u(1)
profile_tier_level_idx[i][j]	u(v)
}	
}	
}	
}	

Fig. 18

Fig. 19

	Descriptor
video_parameter_set { vps() {	
...	
vps_max_layers_minus1	u(0)
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 (j = 0; j < num_output_layer_sets; j++) {	
vps_output_layers_minus1	u(0)
vps_ptl_signal_flag[i]	u(1)
for (k = 0; k < NumLayersInIdTag[i]; k++) {	
if (vps_output_layers_minus1[k] == 2) {	
output_layer_flag[k]	u(1)
if (vps_ptl_signal_flag[i]) {	
profile_tier_level_idx[k]	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(i = 0; i < NumLayersInDLS[i]; i++) {	
num_output_subpic_layer[i][i]	ue(v)
for(k = 0; k < num_output_subpic_layer[i][i]; k++)	
sub_pic_id_layer[i][i][k]	u(8)
if(vps_output_layers_mode[i] == 2)	
output_layer_flag[i][i]	u(1)
if vps_ptl_signal_flag[i]	
profile_tier_level_ids[i][i]	u(v)
}	
}	
...	
}	

FIG. 21

2100

