

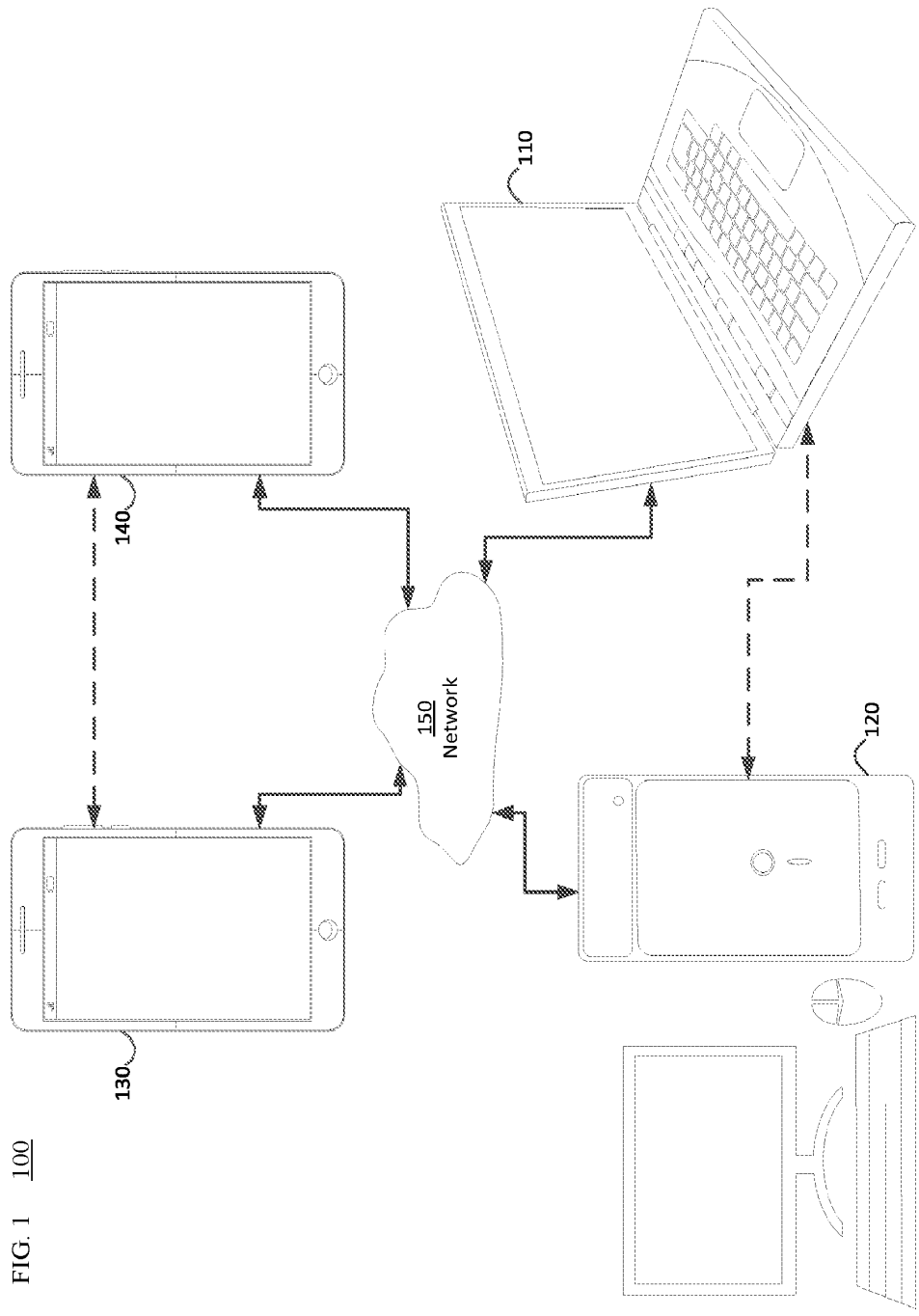


FIG. 1 100

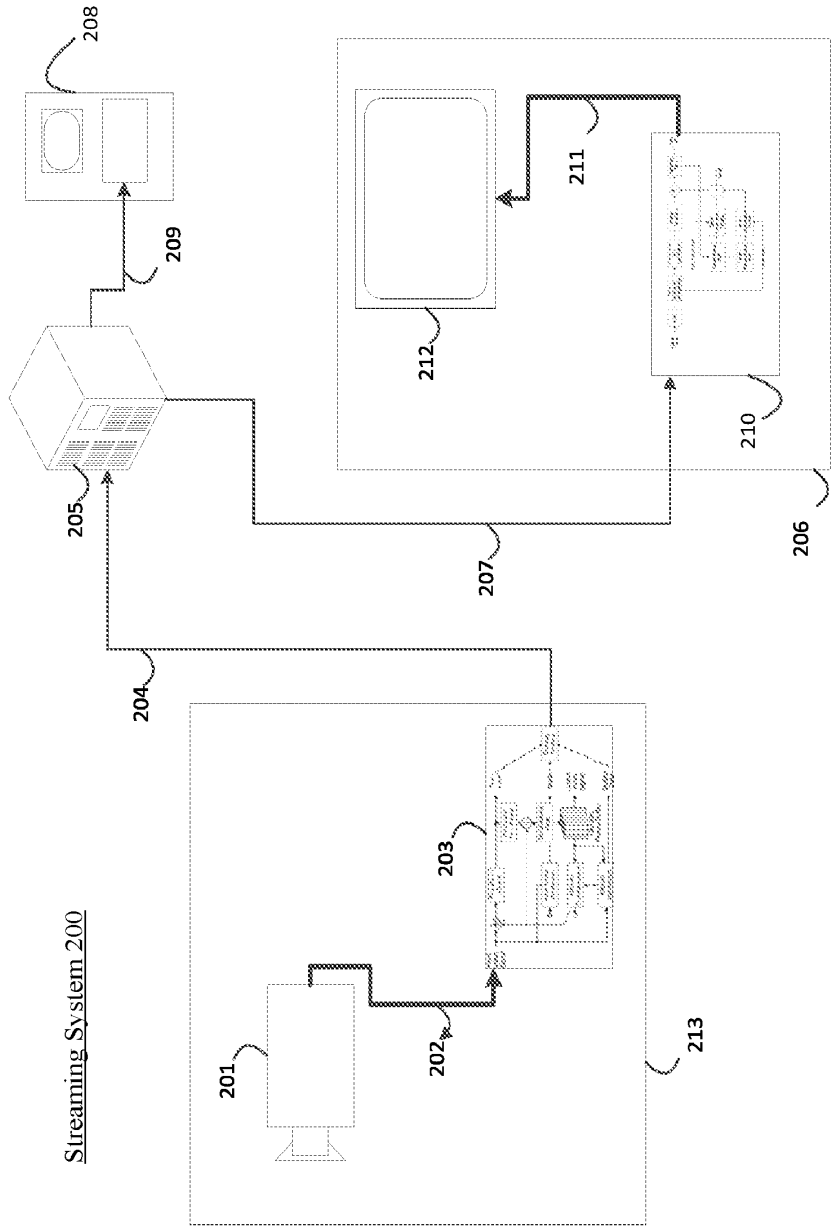


FIG. 2

Decoder 210

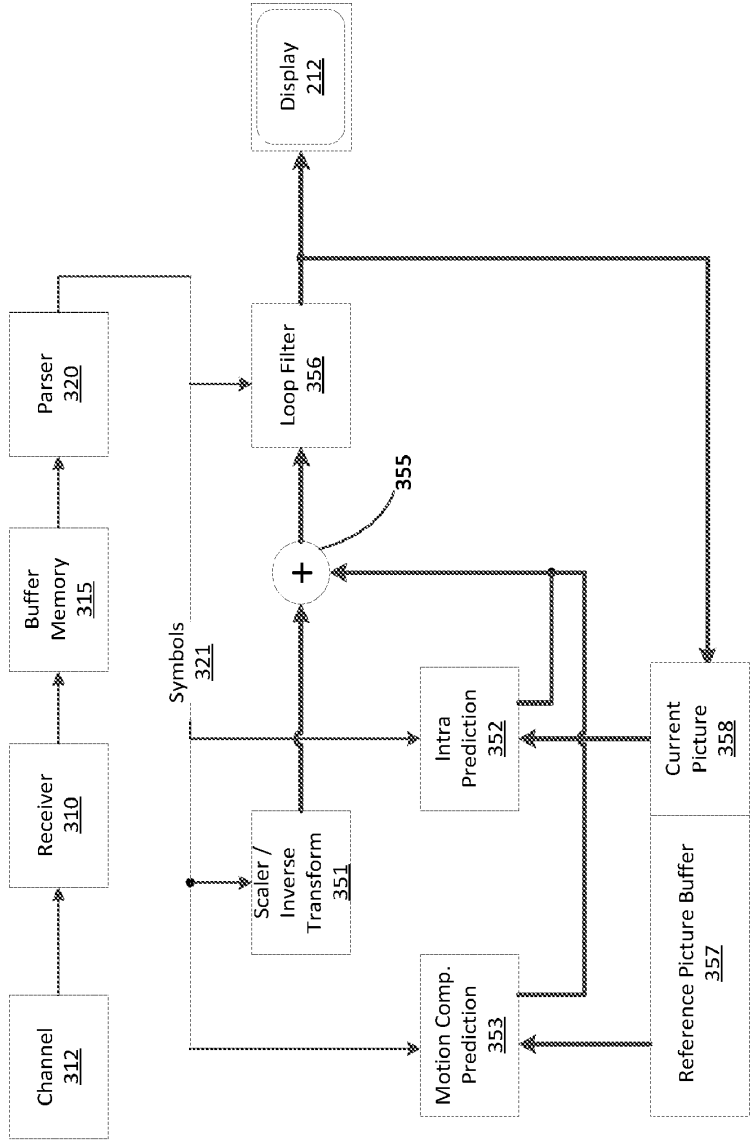


FIG. 3

Encoder 203

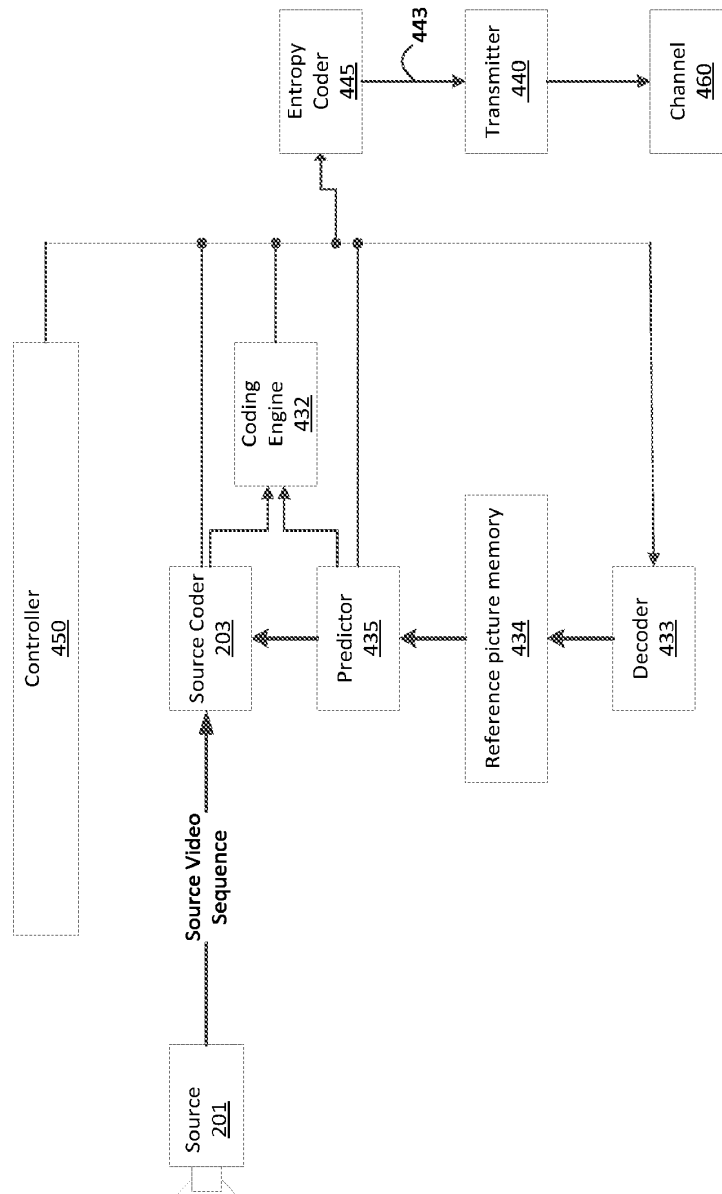


FIG. 4

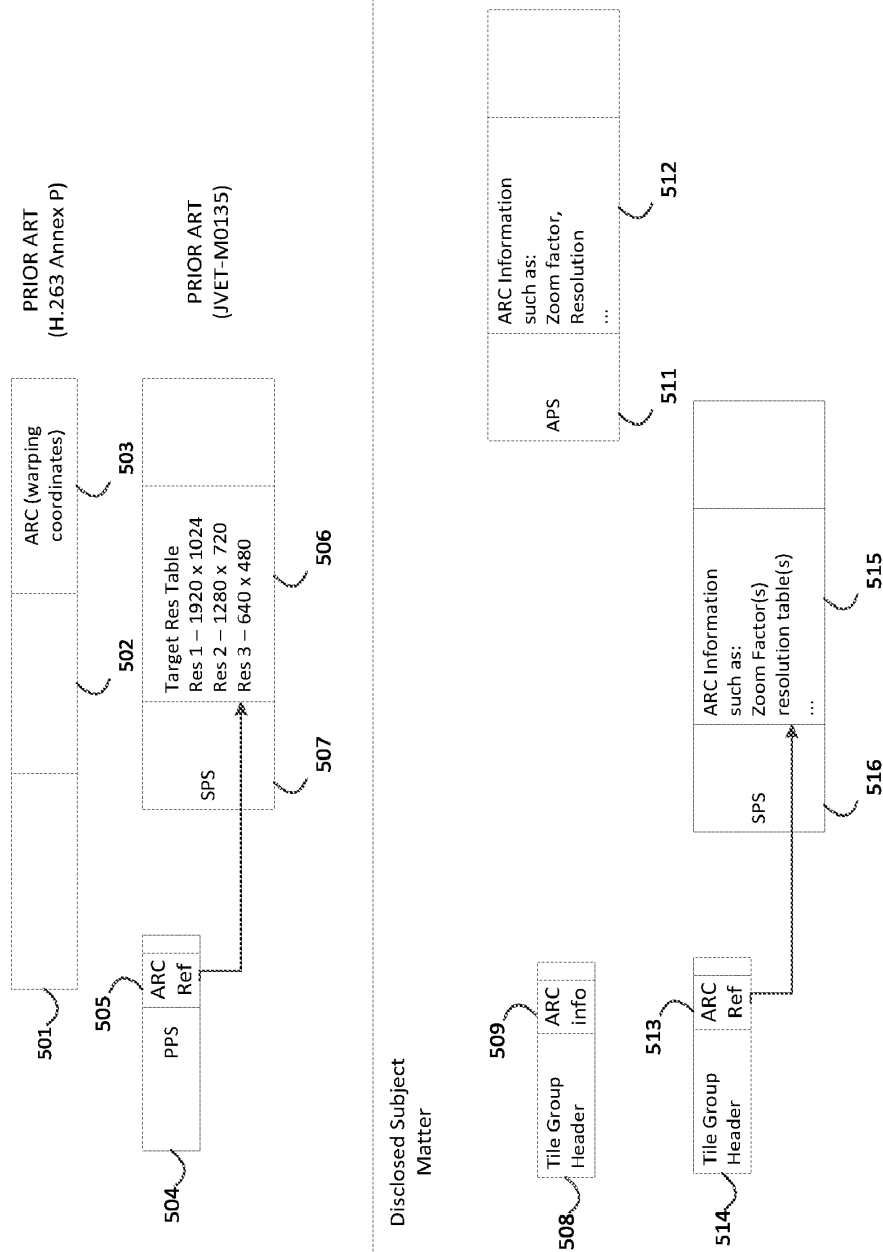


FIG. 5

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

	Descriptor
seq_parameter_set_rbsp() {	
...	
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)
}	
}	
...	
}	

FIG. 6

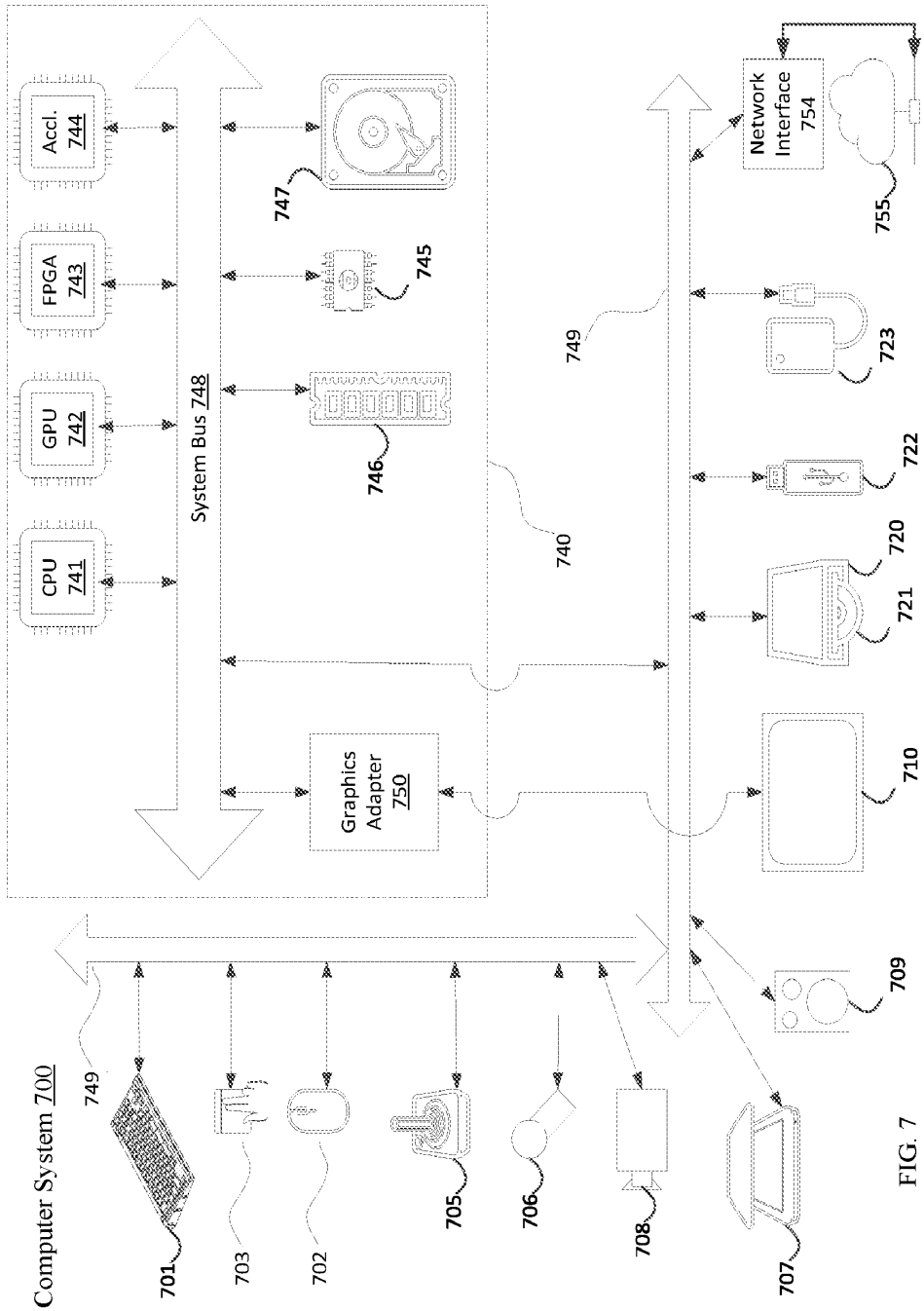


FIG. 7

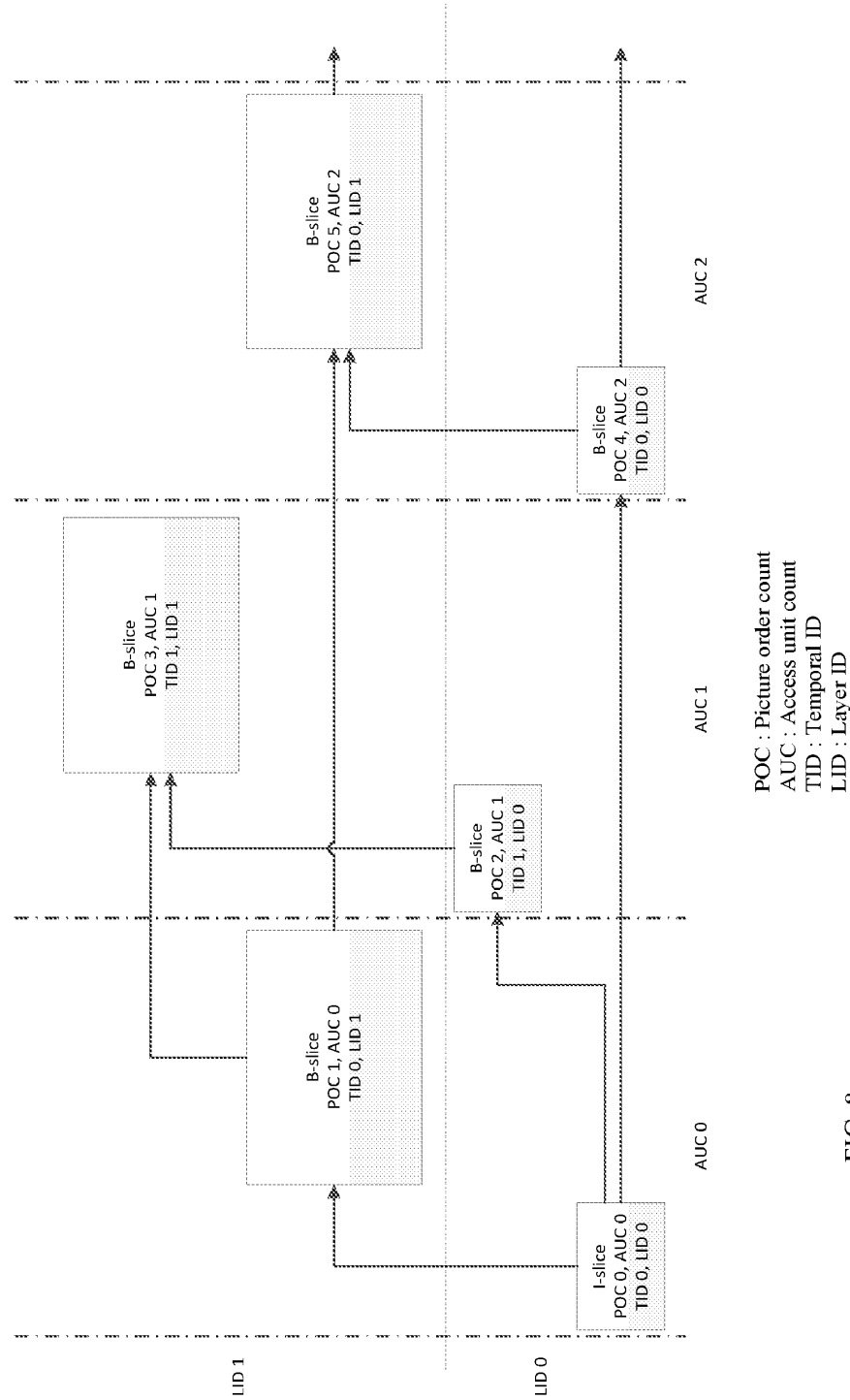


FIG. 8

video_parameters_set_top() { 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_constraint_poc_cycle_per_aub; if (vps_constraint_poc_cycle_per_aub) { vps_poc_cycle_aub; } }	Descriptor: u(4) u(8) s u(4) u(1) s u(1) u(1) u(1) u(8) s s
slice_header() { slice_pic_parameter_set_id; if (rect_slice_flag == NumBackshifts) { slice_address; if (rect_slice_flag && !single_block_per_slice_flag) { num_blocks_in_slice_minus1; } slice_type; if (NalUnitType == GRA_NUT) { recovery_poc_cnt; } slice_pic_order_cnt_lsb; } if (type_constraint_poc_cycle_per_aub) { slice_poc_cycle_aub; } }	Descriptor: u(4) s u(4) s u(4) u(4) s u(4) u(1) s u(1) s u(4) s s

FIG. 9

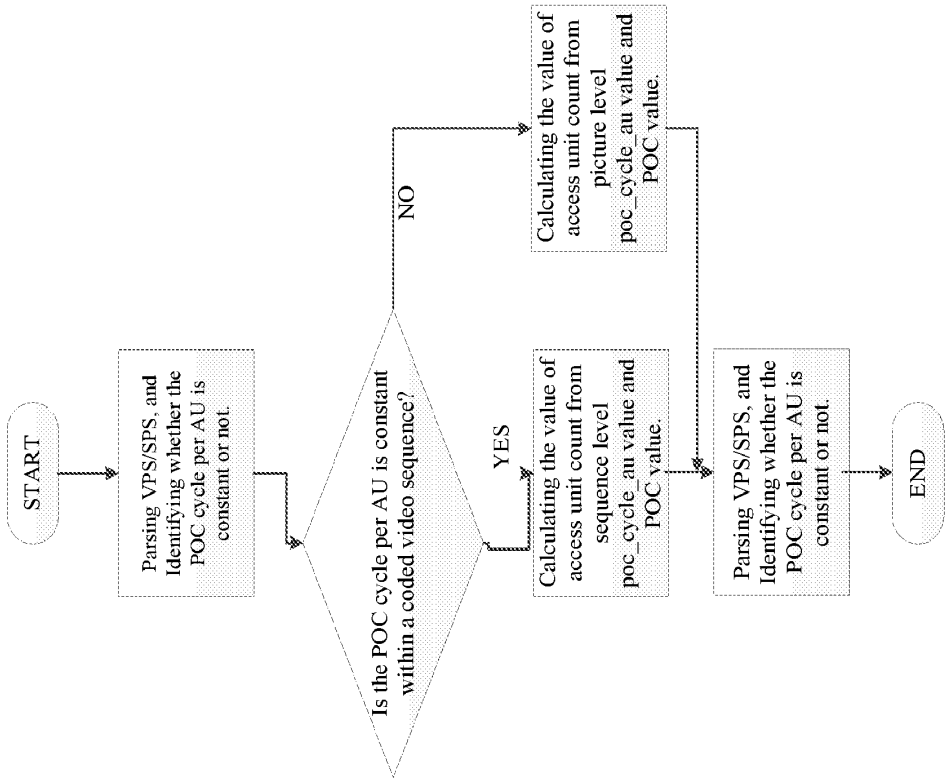


Fig. 10

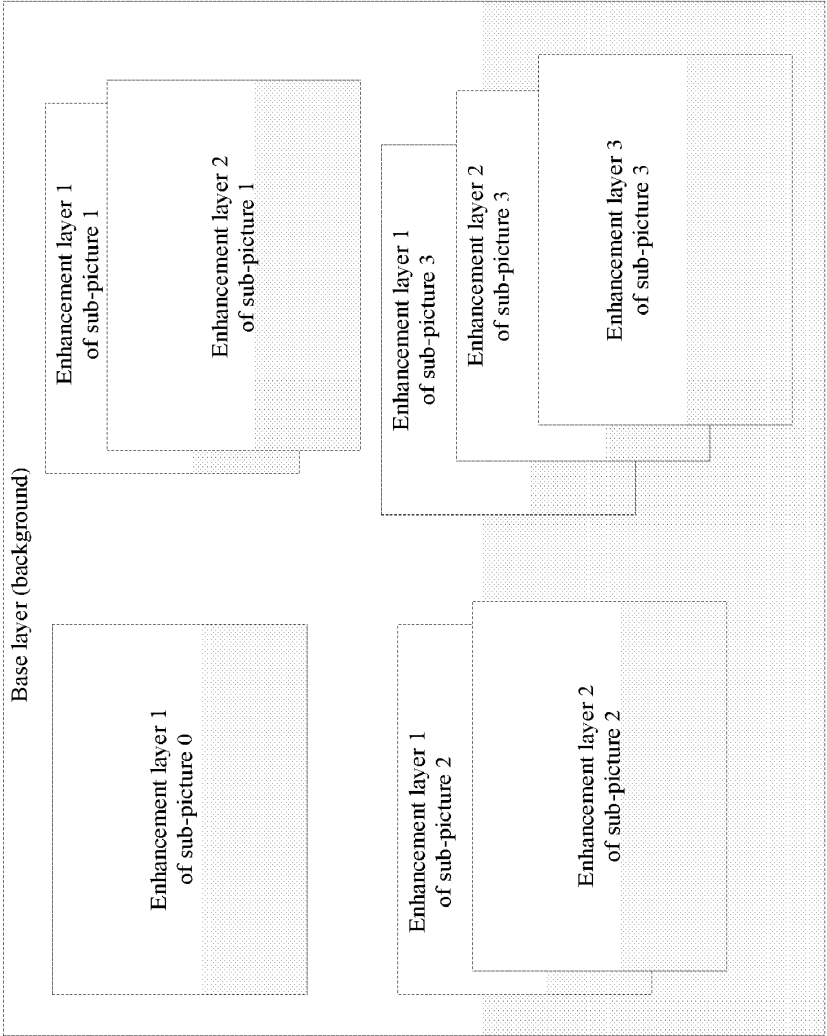


FIG. 11

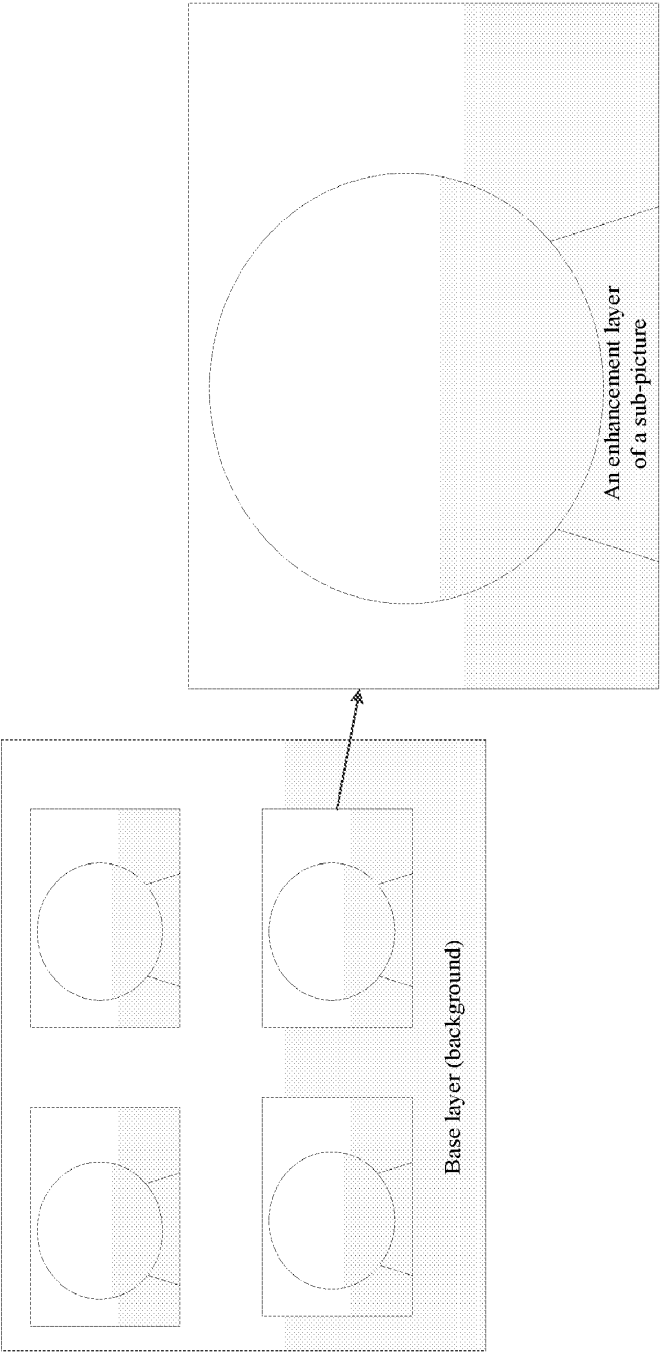


FIG. 12

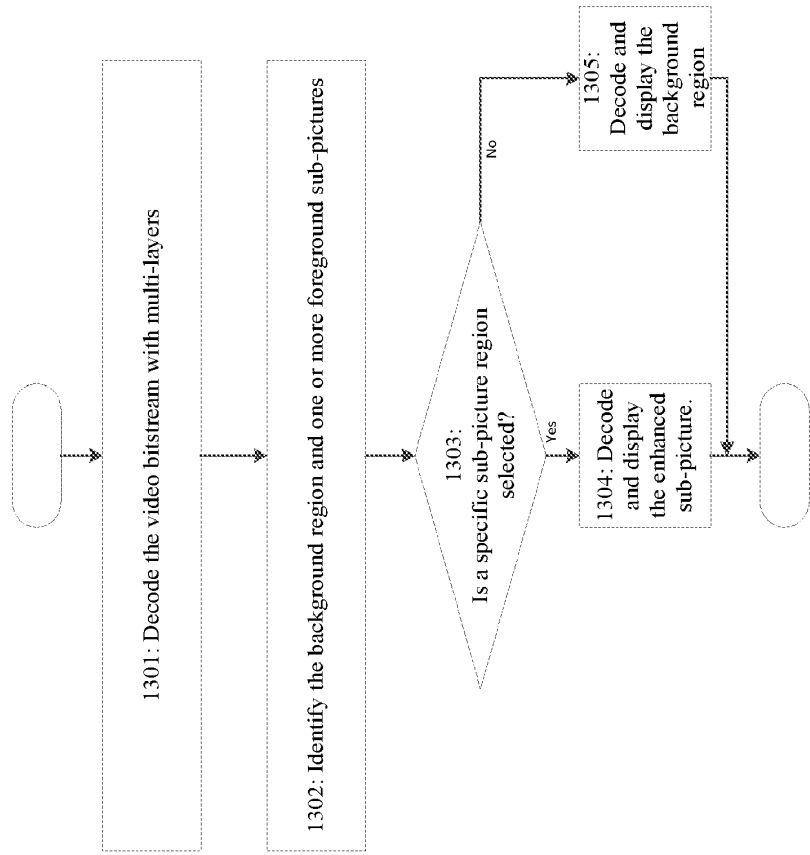


FIG. 13

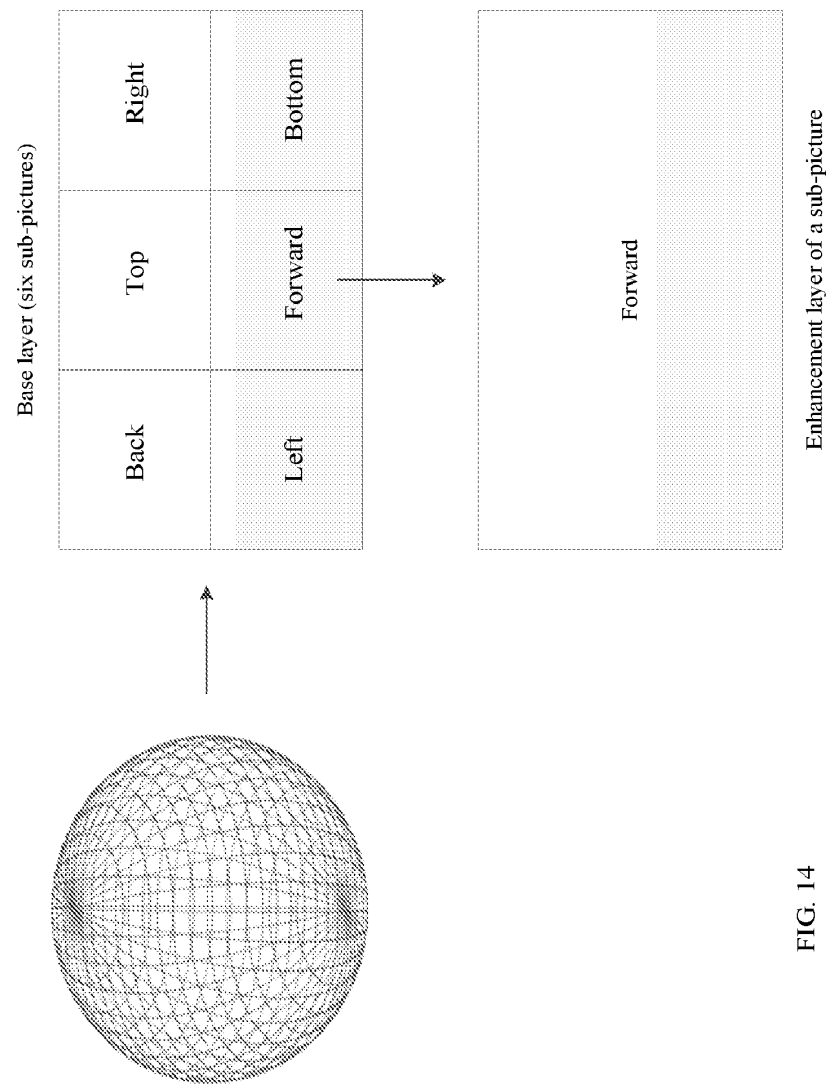


FIG. 14

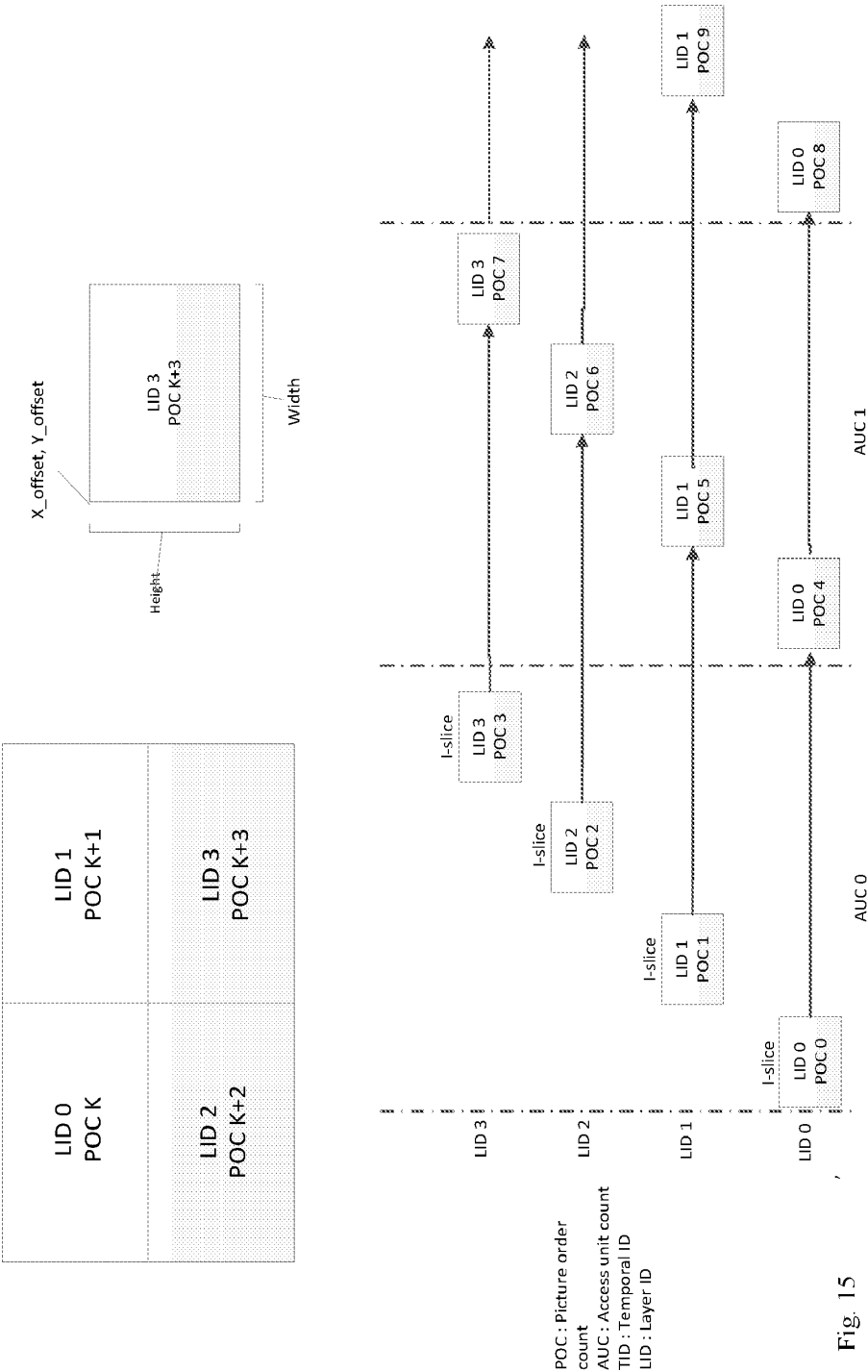
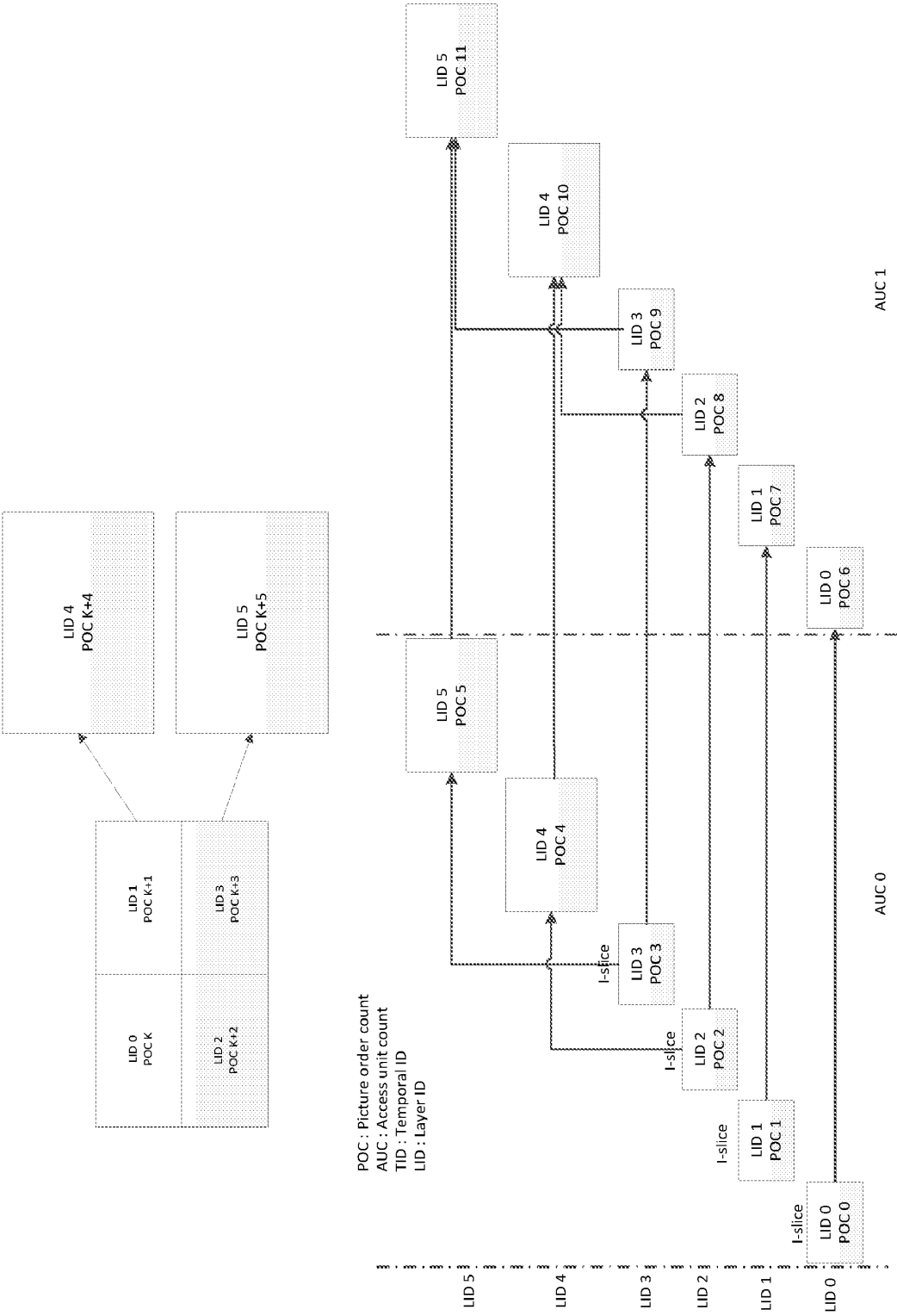


FIG. 16



Descriptor	Descriptor
video_parameter_set_flag() { 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; } ... }	seq_parameter_set_flag() { sps_decoding_parameter_set_id; sps_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; } ... }

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

FIG. 18

	Descriptor
video parameter set rbsp() { ... }	
vps max layers minus1	u(6)
num output layer sets	ue(V)
num profile tier level 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(i = 0; i < NumLayersInIdList[i]; i++) { output layer flag[i][i] profile tier level idx[i][i] } }	u(1) u(V)
...	
}	

FIG. 19

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_model[i]	u(2)
vps_ptl_signal_flag[i]	u(1)
for(i = 0; i < NumLayersInIdList[i]; i++) {	
if(vps_output_layers_model[i] == 2)	
output_layer_flag[i][i]	u(1)
if(vps_ptl_signal_flag[i])	
profile_tier_level_ids[i][i]	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 < NumLayersInIdList[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_idx[i][i]	u(v)
}	
}	
...	
}	

FIG. 21

	Descriptor
video_parameter_set_rbsp() {	
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(i = 0; i < i, i++)	
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_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. 22

video parameter set rsp() {		
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 && kbps all independent layers flag) {		
vps independent layer flag[i]		u(1)
if(kbps 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(kbps 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. 23

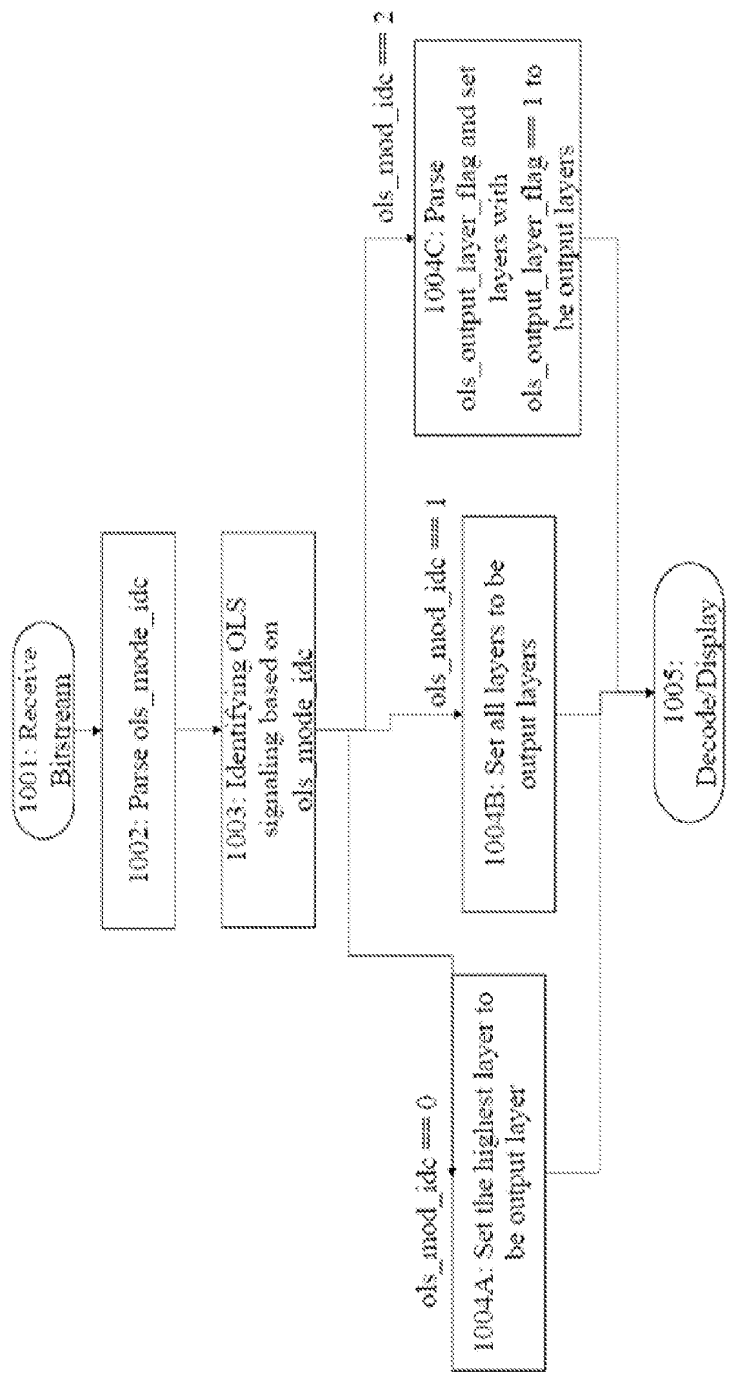


FIG. 24

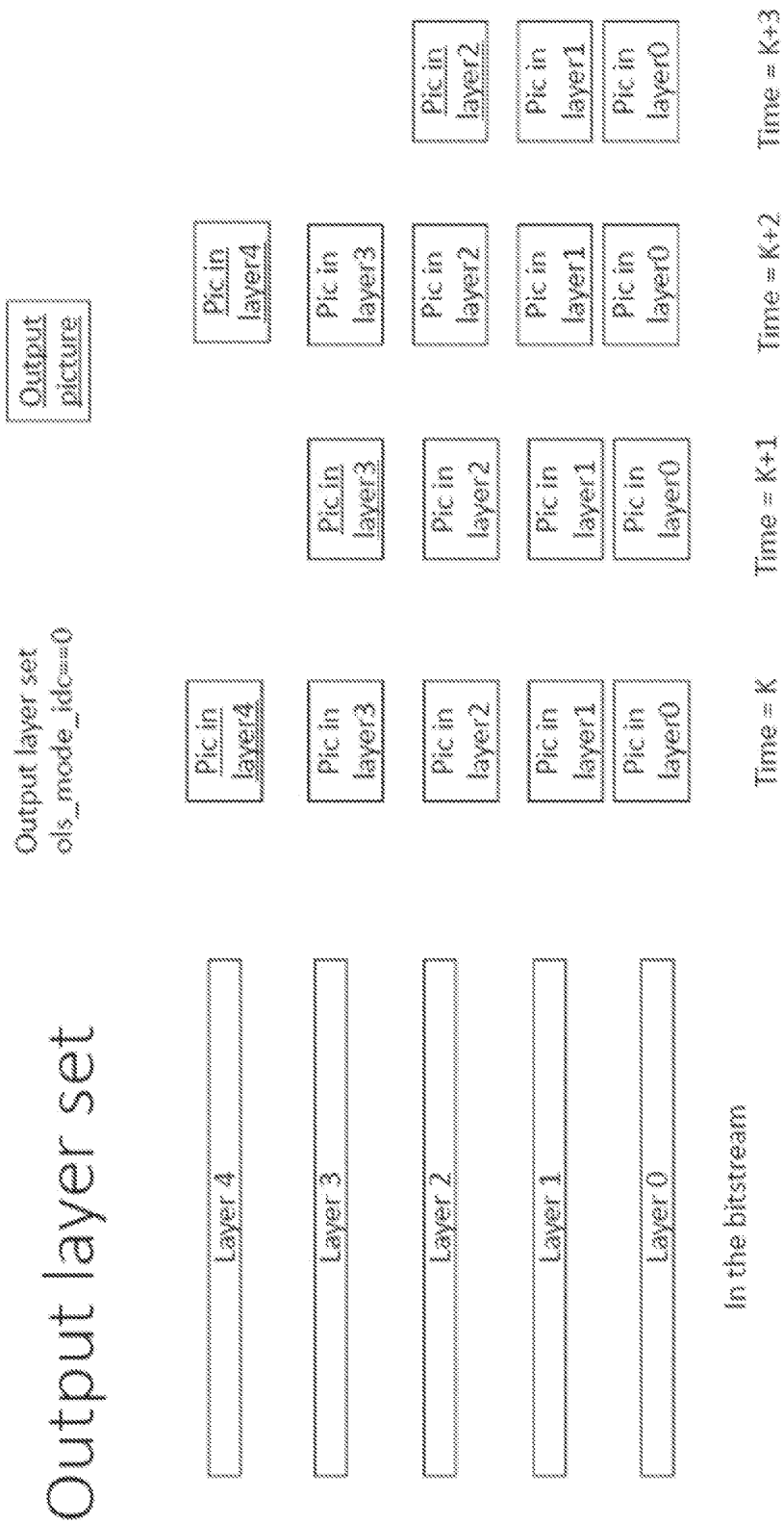


FIG. 25A

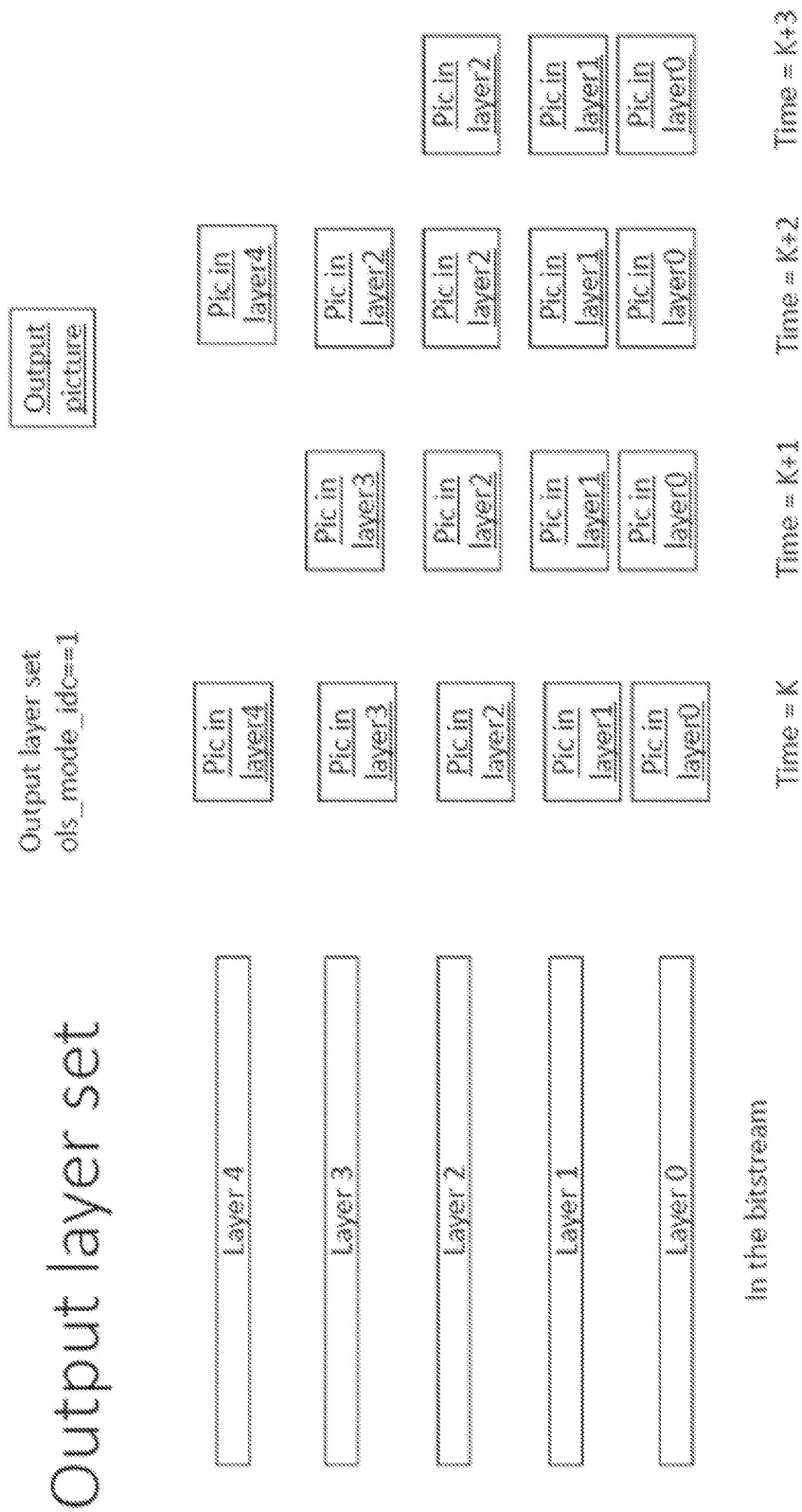


FIG. 25B

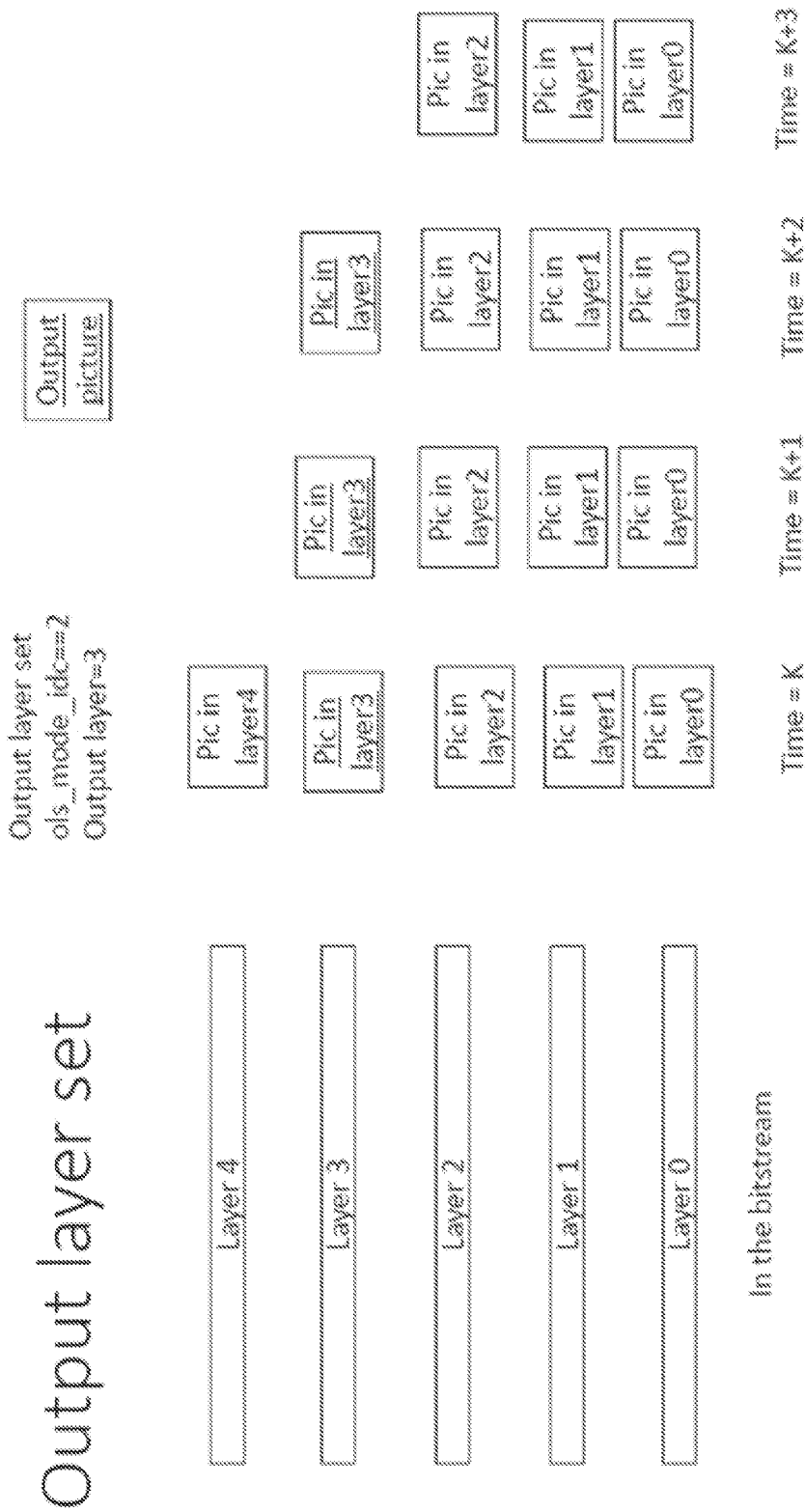


FIG. 25C