

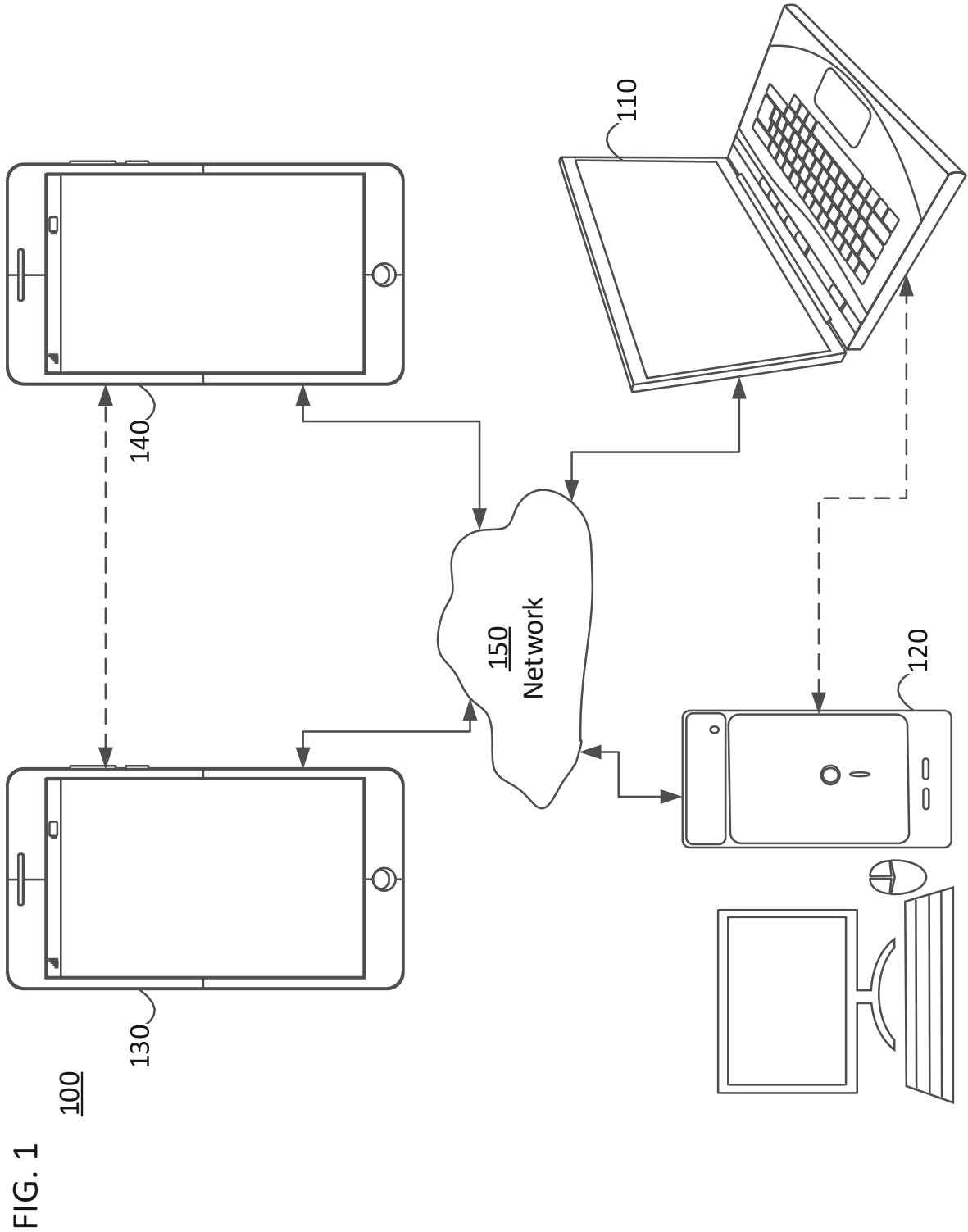
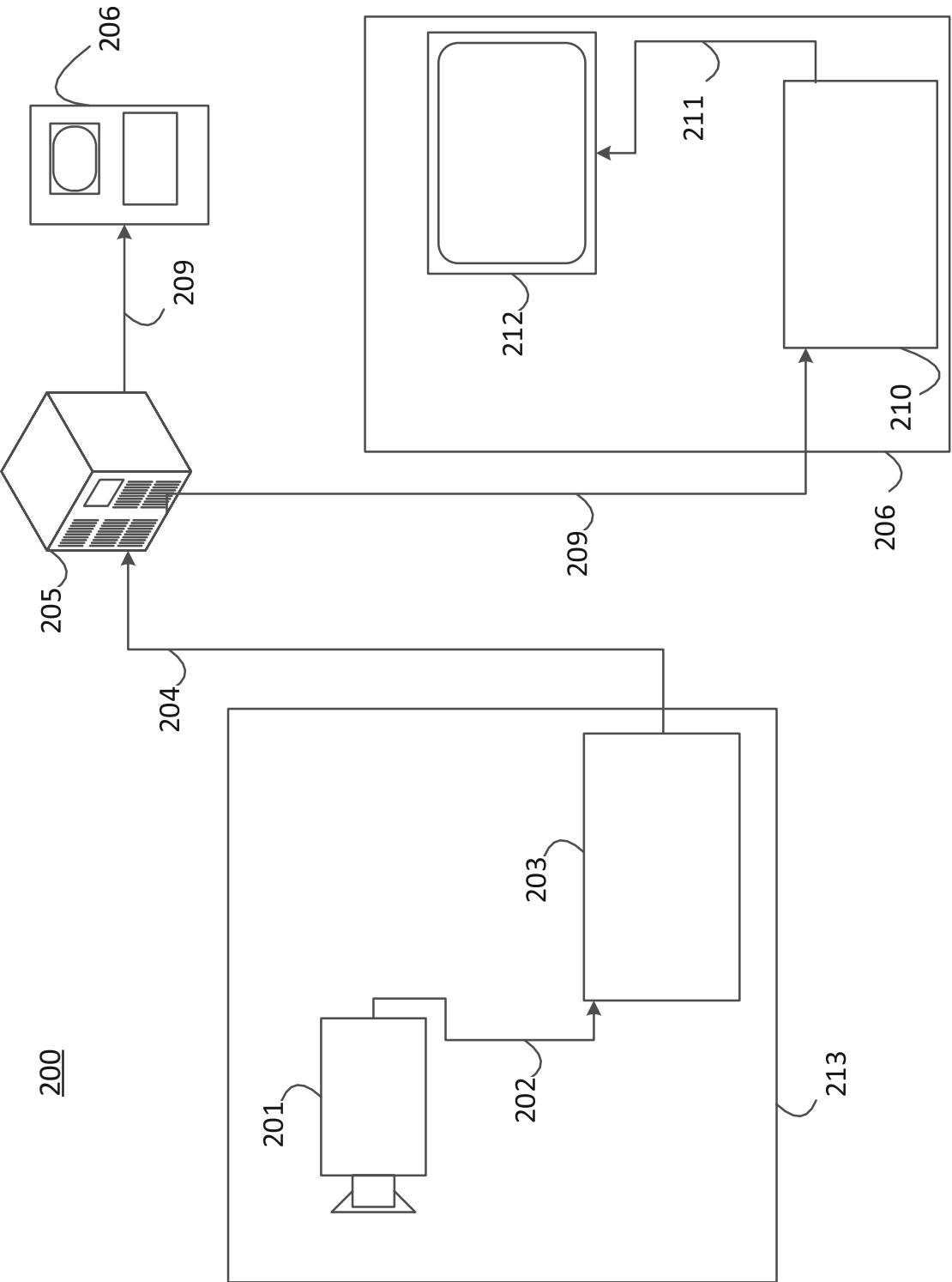


FIG. 2



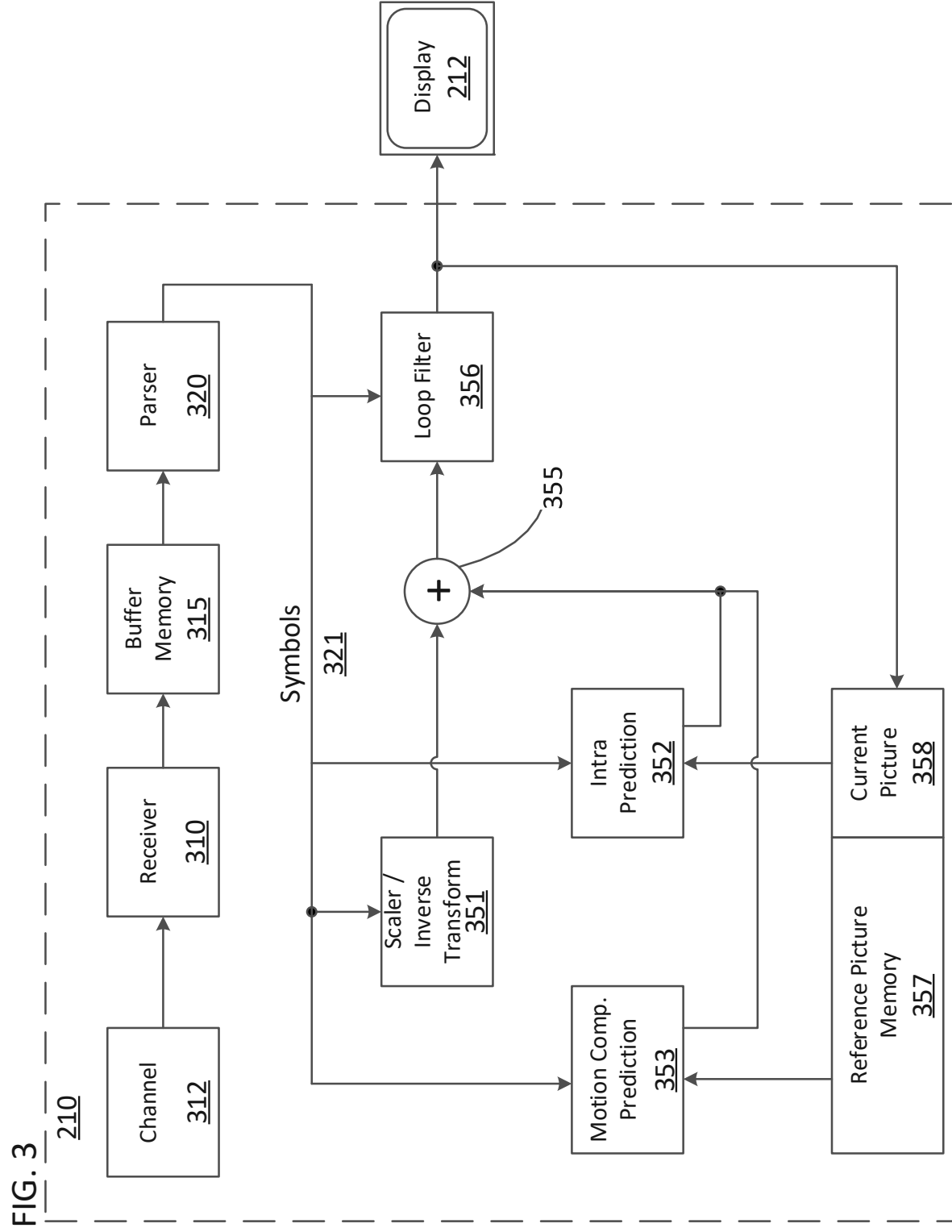


FIG. 4

203

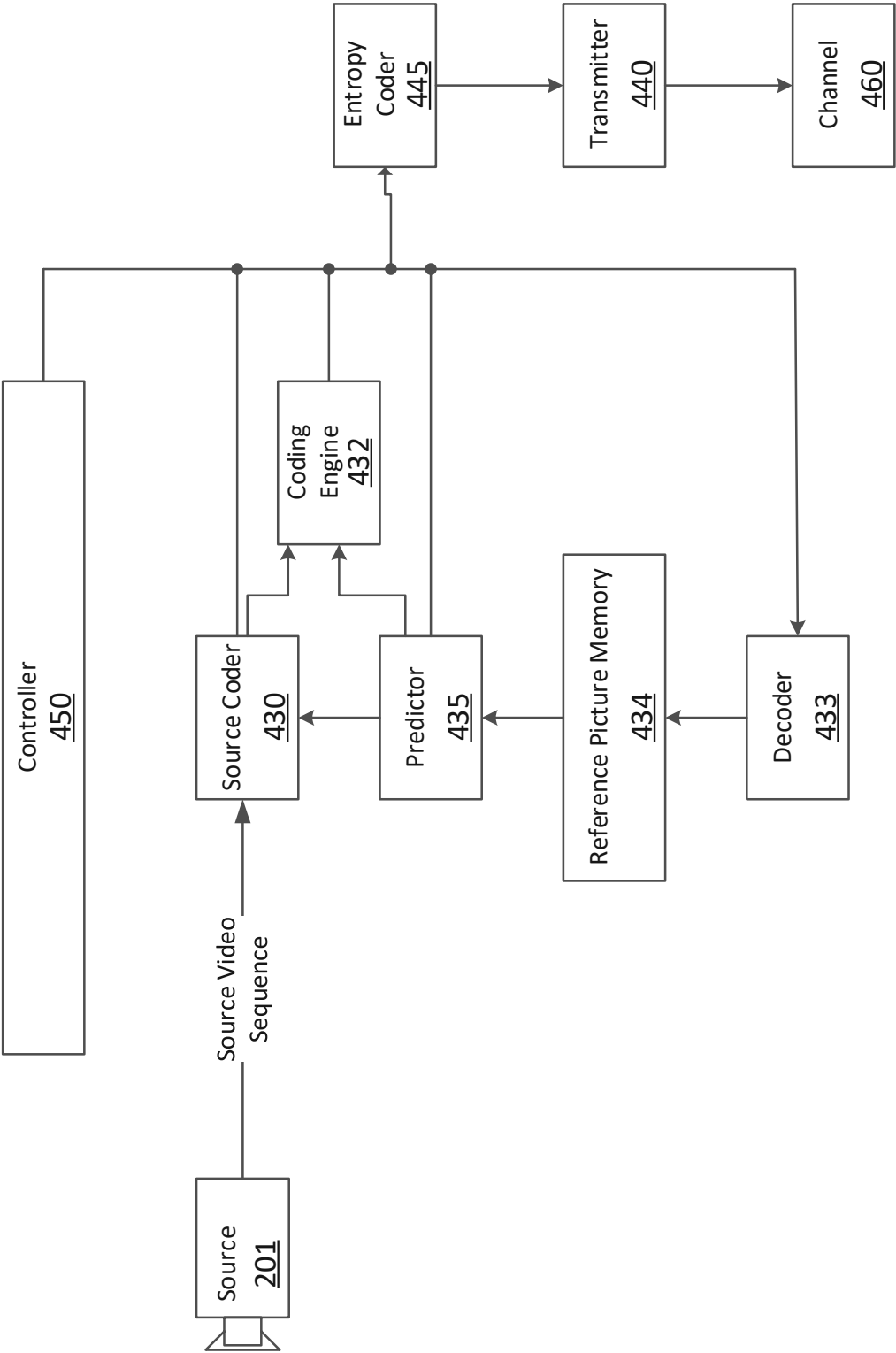


FIG. 5A

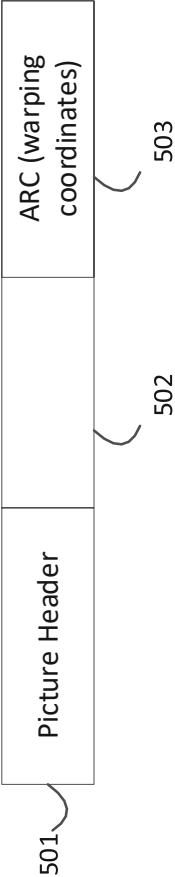


FIG. 5B

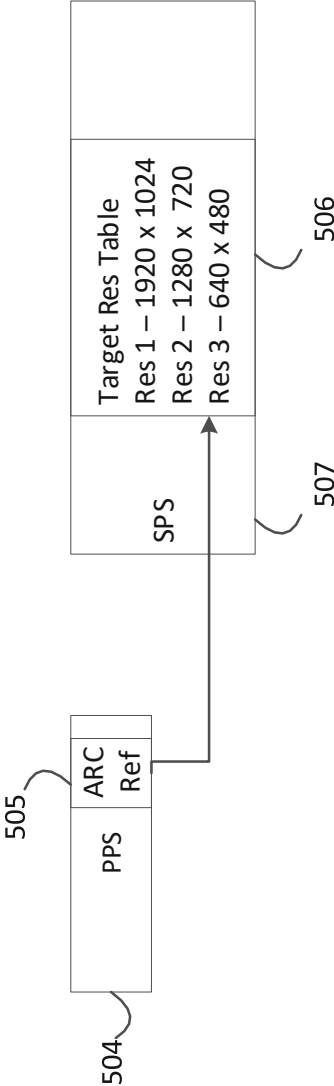


FIG. 6A

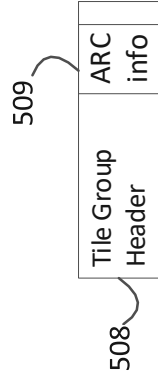


FIG. 6B



FIG. 6C

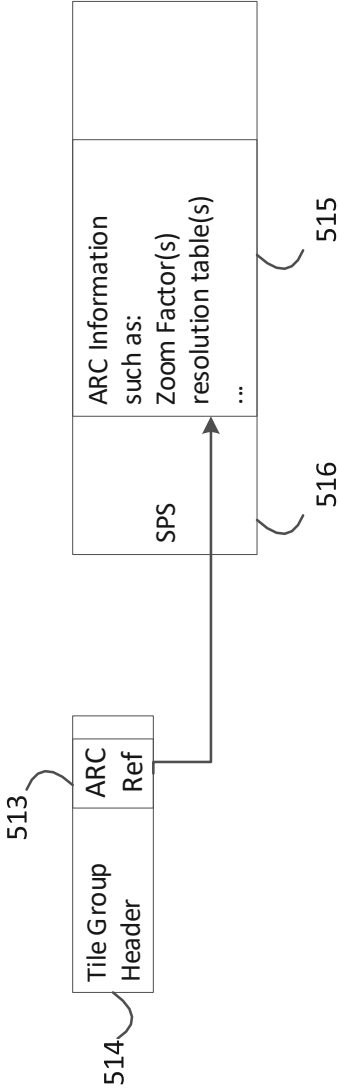


FIG. 7A

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

FIG. 7B

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)
613	output_pic_height_in_luma_samples	ue(v)
	reference_pic_size_present_flag	u(1)
614	if(reference_pic_size_present_flag)	
	{	
615	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++) {	
617	dec_pic_width_in_luma_samples[i]	ue(v)
617	dec_pic_height_in_luma_samples[i]	ue(v)
	}	
	}	
	...	
	}	

FIG. 8

680

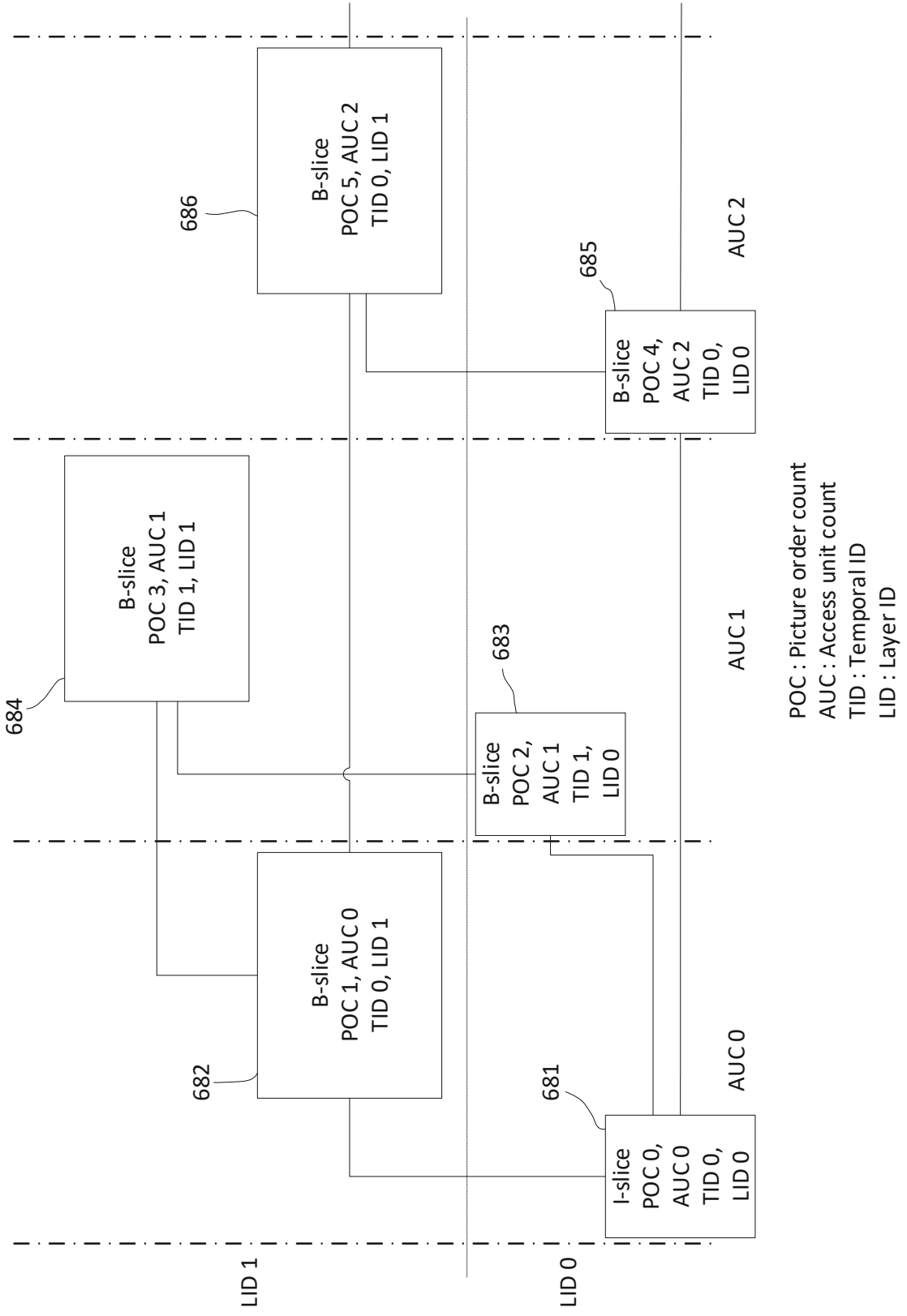


FIG. 9A

630	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) u(7) u(1) u(1) u(1) u(8)
634		
632		

FIG. 9B

640	slice_header() {	Descriptor
	slice_pic_parameter_set_id	ue(v)
	if (!rect_slice_flag NumBricksInPic > 1)	
	slice_address	u(v)
	if (!rect_slice_flag && !single_brick_per_slice_flag)	
	num_bricks_in_slice_minus1	ue(v)
	slice_type	ue(v)
	if(NalUnitType == GRA_NUT)	
	recovery_poc_cnt	se(v)
	slice_pic_order_cnt_lsb	u(v)
	...	
	if(!vps_constant_poc_cycle_per_au)	
642	slice_poc_cycle_au	u(8)

FIG. 10

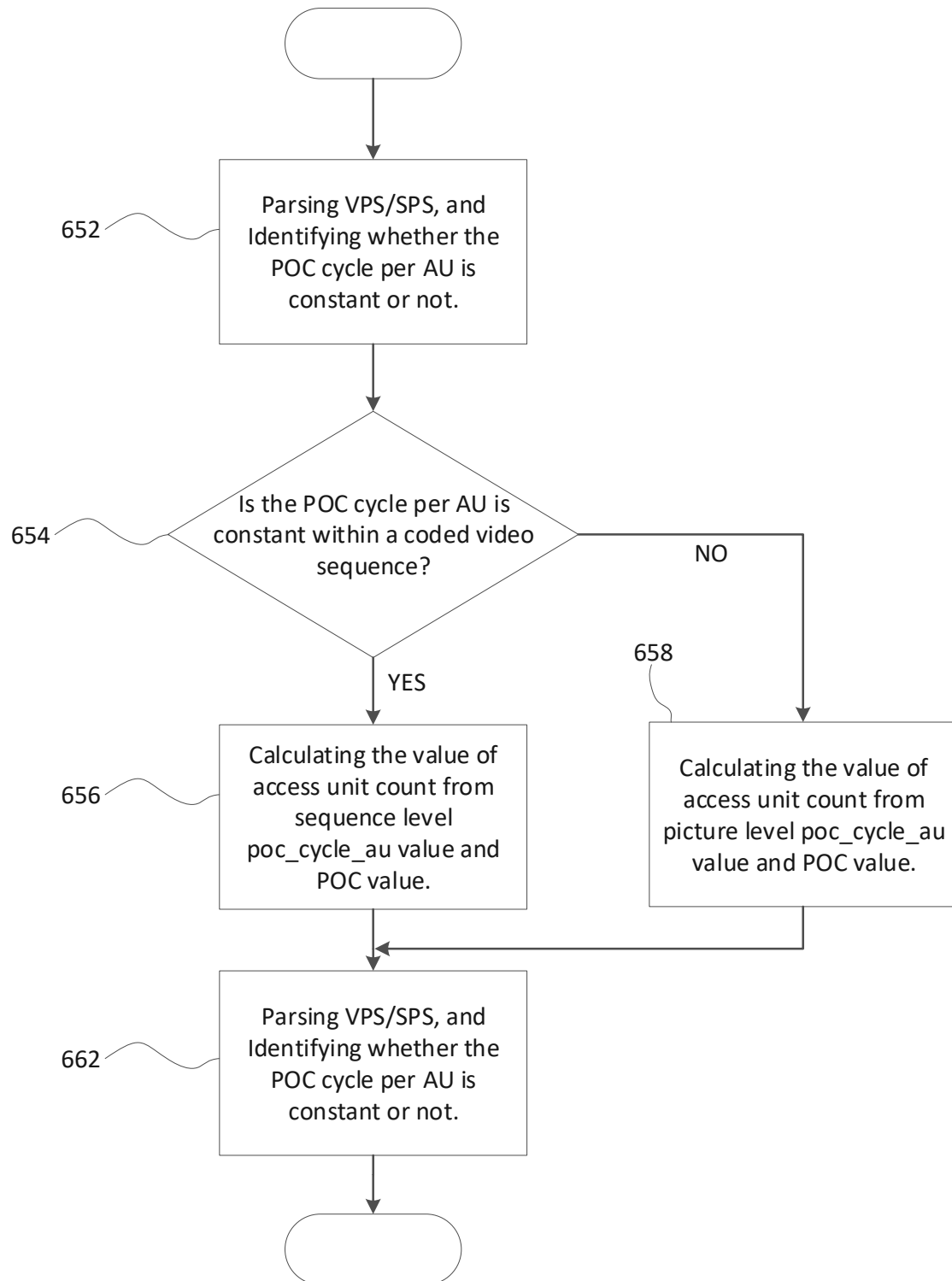


FIG. 12

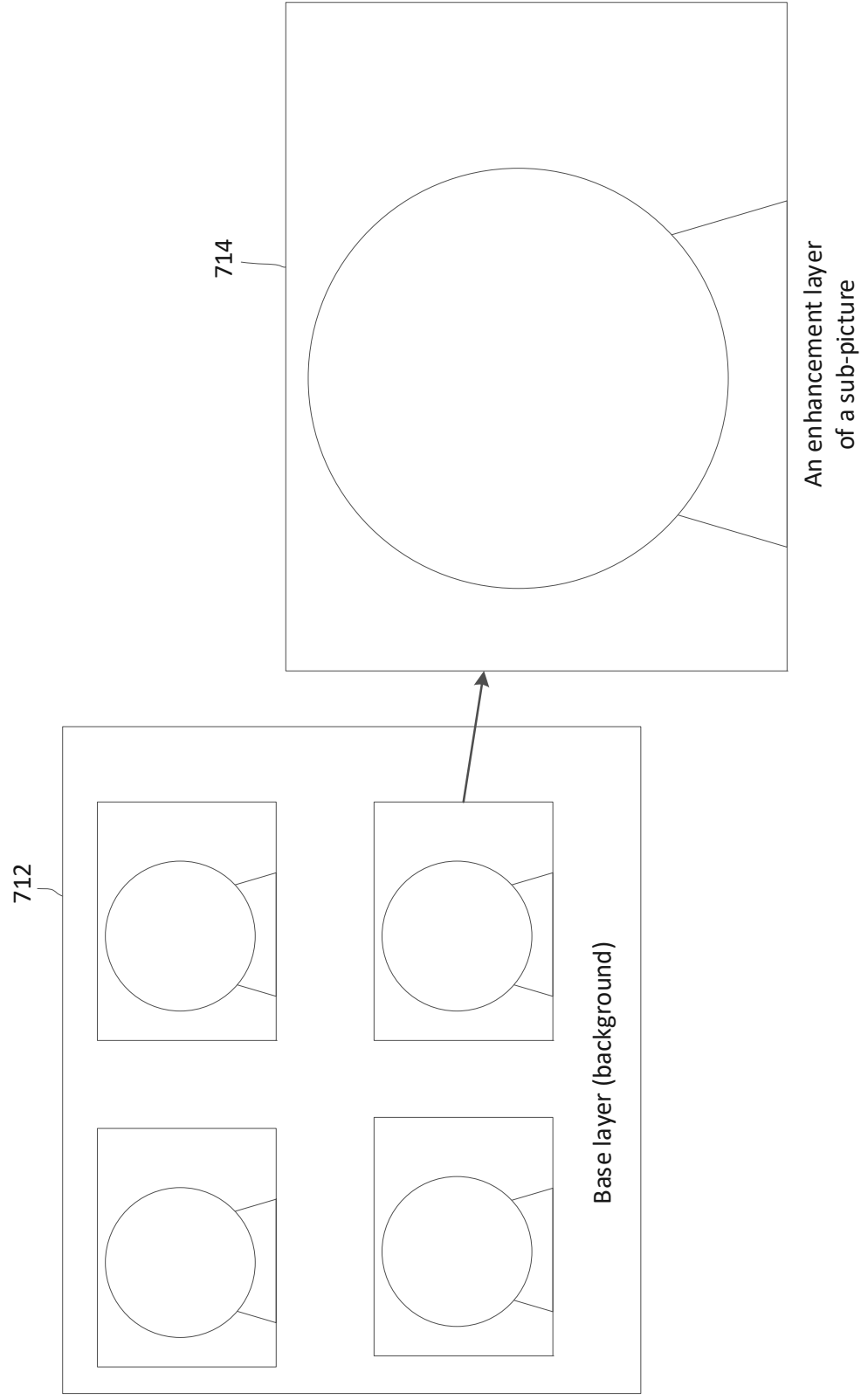


FIG. 13

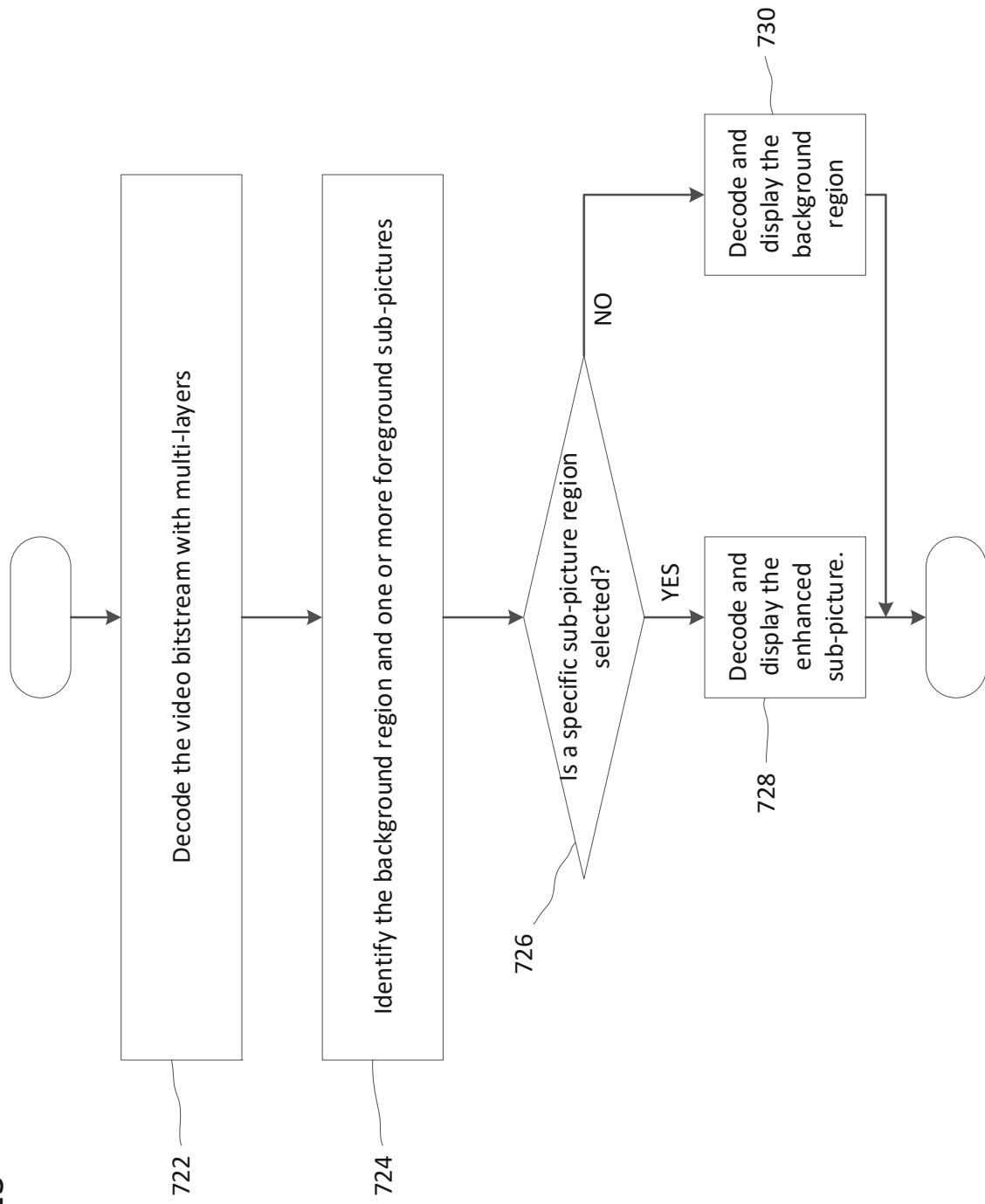


FIG. 14

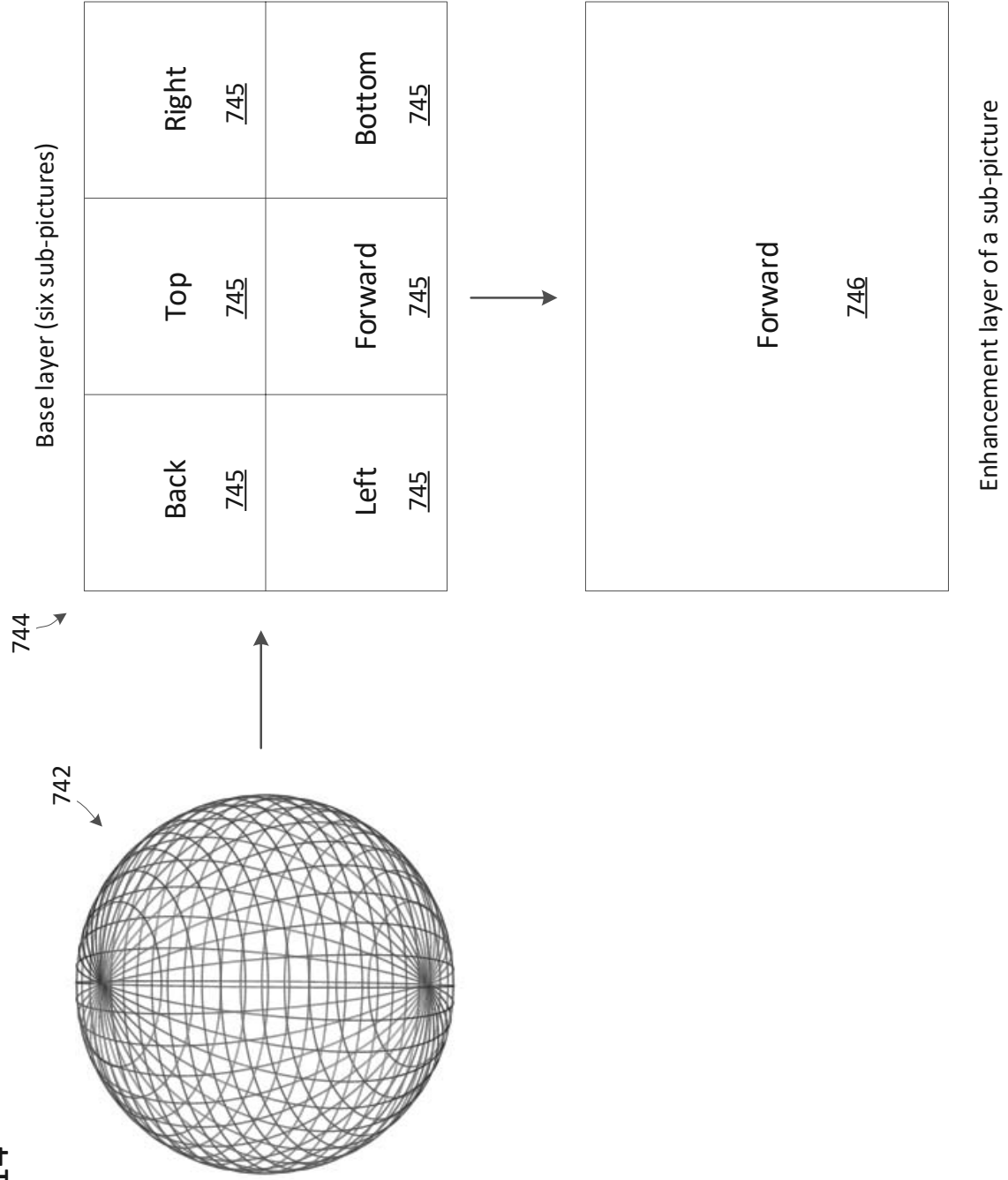


FIG. 15A

<u>752</u>	LID 0 POC K	<u>752</u>	LID 1 POC K+1
<u>752</u>	LID 2 POC K+2	<u>752</u>	LID 3 POC K+3

FIG. 15B

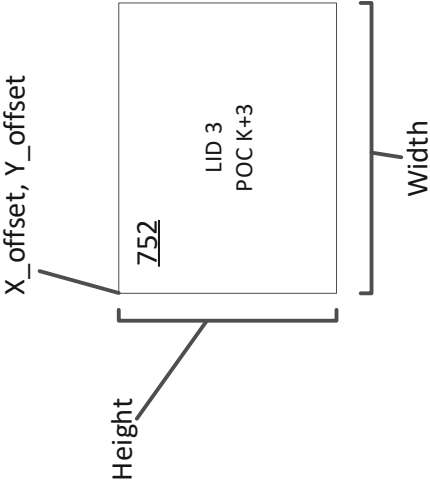


FIG. 16

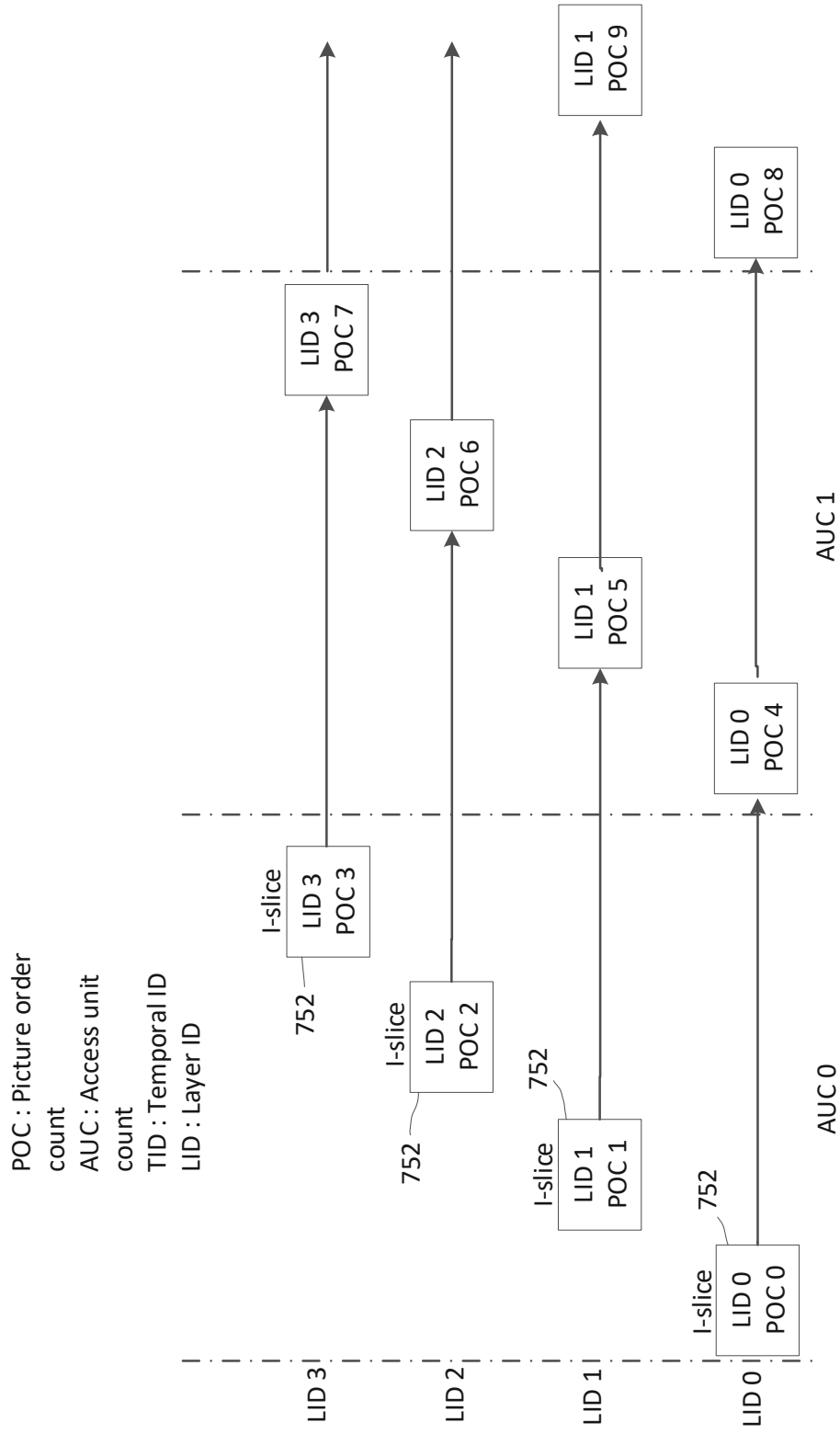


FIG. 17

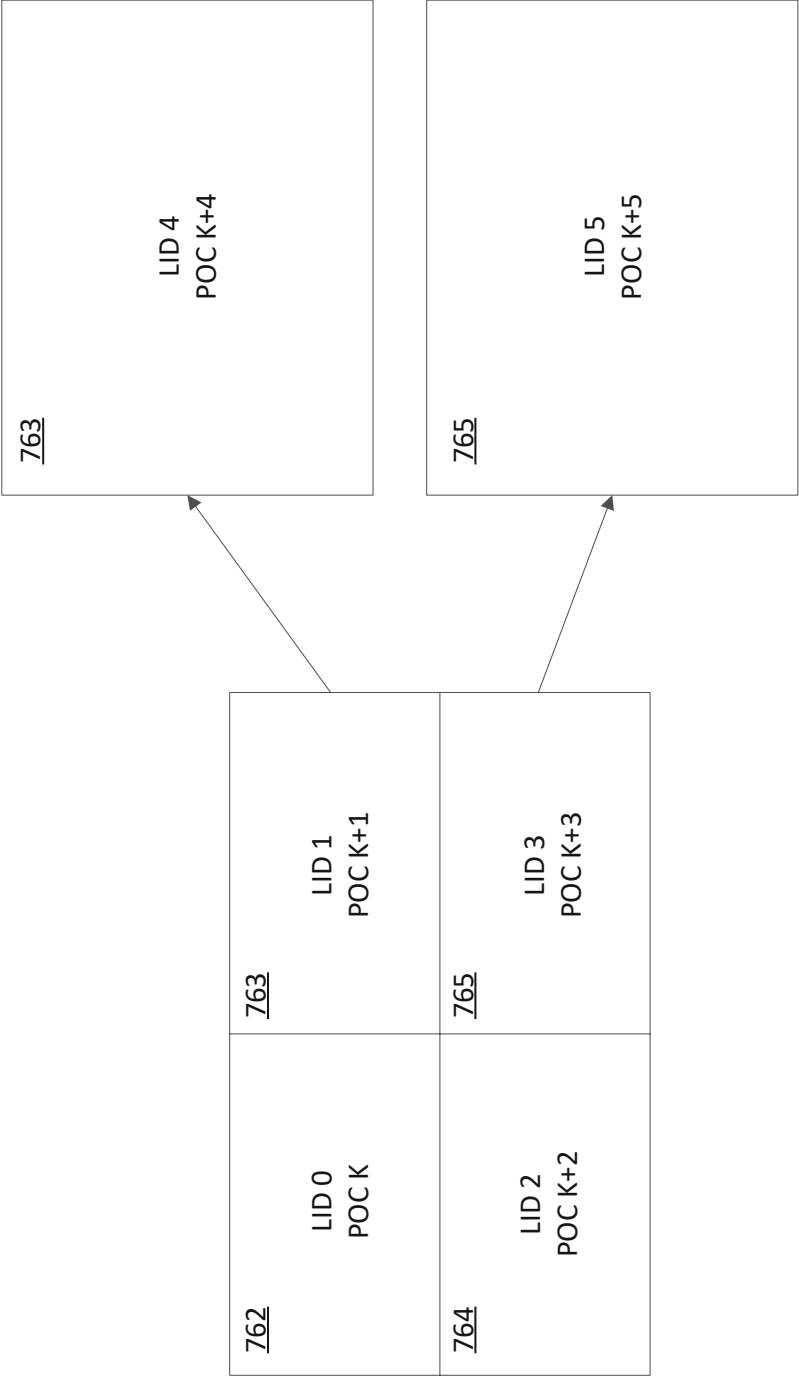


FIG. 18

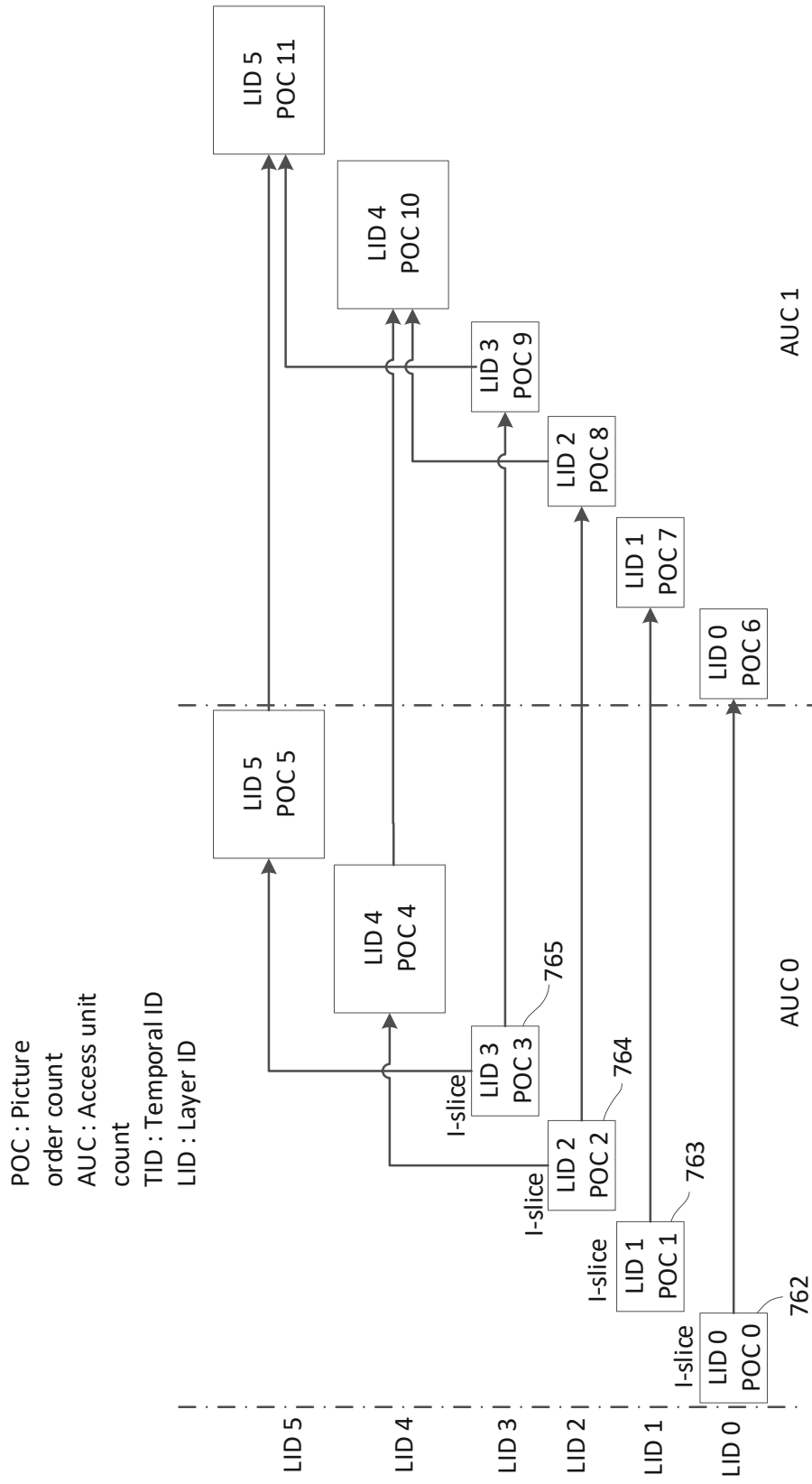


FIG. 19A

770	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 } ... }	Descriptor u(4) u(8) u(7) u(1) u(1) ue(v) ue(v)
772		
774		
776		

FIG. 19B

780	seq_parameter_set_rbsp() {	Descriptor
	sps_decoding_parameter_set_id	u(4)
	sps_video_parameter_set_id	u(4)
	sps_max_sub_layers_minus1	u(3)
	...	
786	pic_width_in_luma_samples	ue(v)
788	pic_height_in_luma_samples	ue(v)
	if(vps_sub_picture_dividing_flag) {	
782	pic_offset_x	ue(v)
784	pic_offset_y	ue(v)
	}	
	...	
	}	

FIG. 20

	sub_region_partitioning_info(payloadSize) {	
791	num_sub_region	ue(v)
792	num_layers	ue(v)
	for (i = 0; i <= num_layers; i++)	
	layer_id [i]	ue(v)
	for(i = 1; i < num_layers; i++)	
	for(j = 0; j < i; j++)	
793	direct_dependency_flag[i][j]	u(1)
	for (i = 0; i < num_sub_region; i++) {	
794	num_layers_for_region[i]	ue(v)
	for(j = 0; j < num_layers_for_region[i]; j++)	
795	sub_region_layer_id[i][j]	ue(v)
796	sub_region_offset_x[i]	ue(v)
797	sub_region_offset_y[i]	ue(v)
798	sub_region_width[i]	ue(v)
799	sub_region_height[i]	ue(v)
	}	
	...	
	}	

FIG. 21

	video_parameter_set_rbsp() {	Descriptor
	...	
802	vps_max_layers_minus1	u(6)
804	num_output_layer_sets	ue(v)
806	num_profile_tier_level	ue(v)
	for(i = 0; i < num_profile_tier_level; i++)	
808	profile_tier_level(vps_max_sub_layers_minus1)	
	for(i = 0; i < num_output_layer_sets; i++)	
	for(j = 0; j < NumLayersInList[i]; j++) {	
810	output_layer_flag[i][j]	u(1)
812	profile_tier_level_idx[i][j]	u(v)
	}	
	...	
	}	

FIG. 22

	video parameter set rbsp() {	Descriptor
	...	
802	vps max layers minus1	u(6)
	if(vps max layers minus1 > 0) {	
804	num output layer sets	ue(v)
806	num profile tier level	ue(v)
	}	
	for(i = 0; i < num profile tier level; i++)	
808	profile tier level(vps max sub layers minus1)	
	for(i = 0; i < num output layer sets; i++) {	
822	vps output layers mode[i]	u(2)
824	vps ptl signal flag[i]	u(1)
	for(j = 0; j < NumLayersInIdList[I]; j++) {	
	if(vps output layers mode[i][j] == 2)	
810	output layer flag[I][j]	u(1)
	if(vps ptl signal flag[i])	
812	profile tier level idx[I][j]	u(v)
	}	
	}	
	...	
	}	

FIG. 23

	video parameter set rbsp() {	Descriptor
	...	
802	vps max layers minus1	u(6)
	if(vps max layers minus1 > 0) {	
804	num output layer sets	ue(v)
806	num profile tier level	ue(v)
	}	
	max subpics minus1	u(8)
	for(i = 0; i < max subpics minus1; i++) {	
821	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++) {	
822	vps output layers mode[i]	u(2)
824	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++)	
826	sub pic id layer[i][j][k]	u(8)
	if(vps output layers mode[i] == 2)	
810	output layer flag[i][j]	u(1)
	if(vps_ptl signal flag[i])	
812	profile tier level idx[i][j]	u(v)
	}	
	}	
	...	
	}	

FIG. 24

	video_parameter_set_rbsp() {	Descriptor
842	vps video parameter set id	u(4)
802	vps max layers minus1	u(6)
846	vps max sublayers minus1	u(3)
	if(vps_max_layers_minus1 > 0 && vps_max_sublayers_minus1 > 0)	
848	vps_all_layers_same_num_sublayers_flag	u(1)
	if(vps_max_layers_minus1 > 0)	
850	vps all independent layers flag	u(1)
	for(i = 0; i <= vps_max_layers_minus1; i++) {	
852	vps layer id[i]	u(6)
	if(i > 0 && !vps_all_independent_layers_flag) {	
854	vps independent layer flag[i]	u(1)
	if(!vps_independent_layer_flag[i]) {	
	for(j = 0; j < i; j++)	
856	vps_direct_ref_layer_flag[i][j]	u(1)
858	max tid ref present flag[i]	u(1)
	if(max tid ref present flag[i])	
860	max tid il ref pics plus1[i]	u(3)
	}	
	}	
	}	
	if(vps_max_layers_minus1 > 0) {	
	if(vps_all_independent_layers_flag)	
862	each layer is an ols flag	u(1)
	if(!each layer is an ols flag) {	
	if(!vps_all_independent_layers_flag)	
864	ols mode idc	u(2)
	if(ols_mode_idc == 2) {	
866	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++)	
868	ols output layer flag[i][j]	u(1)
	}	
	}	
	}	
	...	
	}	

FIG. 25

	video_parameter_set_rbsp() {	Descriptor
842	vps video parameter set id	u(4)
802	vps max layers minus1	u(6)
846	vps max sublayers minus1	u(3)
	if(vps_max_layers_minus1 > 0 && vps_max_sublayers_minus1 > 0)	
848	vps_all_layers_same_num_sublayers_flag	u(1)
	if(vps_max_layers_minus1 > 0)	
850	vps all independent layers flag	u(1)
	for(i = 0; i <= vps_max_layers_minus1; i++) {	
852	vps layer id[i]	u(6)
	if(i > 0 && !vps_all_independent_layers_flag) {	
854	vps independent layer flag[i]	u(1)
	if(!vps_independent_layer_flag[i]) {	
	for(j = 0; j < i; j++)	
856	vps_direct_ref_layer_flag[i][j]	u(1)
858	max tid ref present flag[i]	u(1)
	if(max tid ref present flag[i])	
860	max tid il ref pics plus1[i]	u(3)
	}	
	}	
	}	
	if(vps_max_layers_minus1 > 0) {	
	if(vps_all_independent_layers_flag)	
862	each layer is an ols flag	u(1)
	if(!each layer is an ols flag) {	
	if(!vps_all_independent_layers_flag)	
864	ols mode idc	u(2)
	if(ols_mode_idc == 2 && vps_max_layers_minus1 > 1) {	
866	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++)	
868	ols output layer flag[i][j]	u(1)
	}	
	}	
	}	
	...	
	}	

FIG. 26

1	picture header structure() {	Descriptor
2	ph_gdr_or_irap_pic_flag	u(1)
	if(ph_gdr_or_irap_pic_flag)	
	ph_gdr_pic_flag	u(1)
	ph_inter_slice_allowed_flag	u(1)
	if(ph_inter_slice_allowed_flag)	
	ph_intra_slice_allowed_flag	u(1)
	ph_non_ref_pic_flag	u(1)
	ph_pic_parameter_set_id	ue(v)
	ph_pic_order_cnt_lsb	u(v)
	if(ph_gdr_or_irap_pic_flag)	
3	ph_no_output_of_prior_pics_flag	u(1)
	if(ph_gdr_pic_flag)	
	ph_recovery_poc_cnt	ue(v)
	...	
	}	

FIG. 27

10	access unit delimiter rbsp() {	Descriptor
12	aud_irap_or_gdr_au_flag	u(1)
	aud_pic_type	u(3)
	rbsp_trailing_bits()	
	}	

FIG. 28

1	picture header structure() {	Descriptor
6	ph irap pic flag	u(1)
	if(!ph irap pic flag)	
7	ph gdr pic flag	u(1)
	...	
	}	

FIG. 29

20	slice header() {	Descriptor
	sh picture header in slice header flag	u(1)
	if(sh picture header in slice header flag)	
	picture header structure()	
	...	
	if(nal_unit_type == IDR_W_RADL nal_unit_type == IDR_N_LP nal_unit_type == CRA_NUT)	
23	sh no output of prior pics flag	u(1)
	...	
	}	

FIG. 30

10	access_unit_delimiter_rbsp() {	Descriptor
16	aud irap au flag	u(1)
17	aud gdr au flag	u(1)
	aud pic type	u(3)
	rsbp_trailing_bits()	
	}	

FIG. 31

1	picture_header_structure() {	Descriptor
2	ph_gdr_or_irap_pic_flag	u(1)
	ph_non_ref_pic_flag	u(1)
7	if(ph_gdr_or_irap_pic_flag)	
	ph_gdr_pic_flag	u(1)
	...	
32	ph_pic_order_cnt_lsb	u(v)
	if(ph_gdr_or_irap_pic_flag)	
	ph_no_output_of_prior_pics_flag	u(1)
	for(i = 0; i < NumExtraPhBits; i++)	
	ph_extra_bit[i]	u(1)
	if(sps_poc_msb_cycle_flag) {	
	ph_poc_msb_cycle_present_flag	u(1)
	if(ph_poc_msb_cycle_present_flag)	
36	ph_poc_msb_cycle_val	u(v)
	}	
	if(ph_gdr_pic_flag)	
34	ph_recovery_poc_cnt	ue(v)
	...	
	}	

FIG. 32

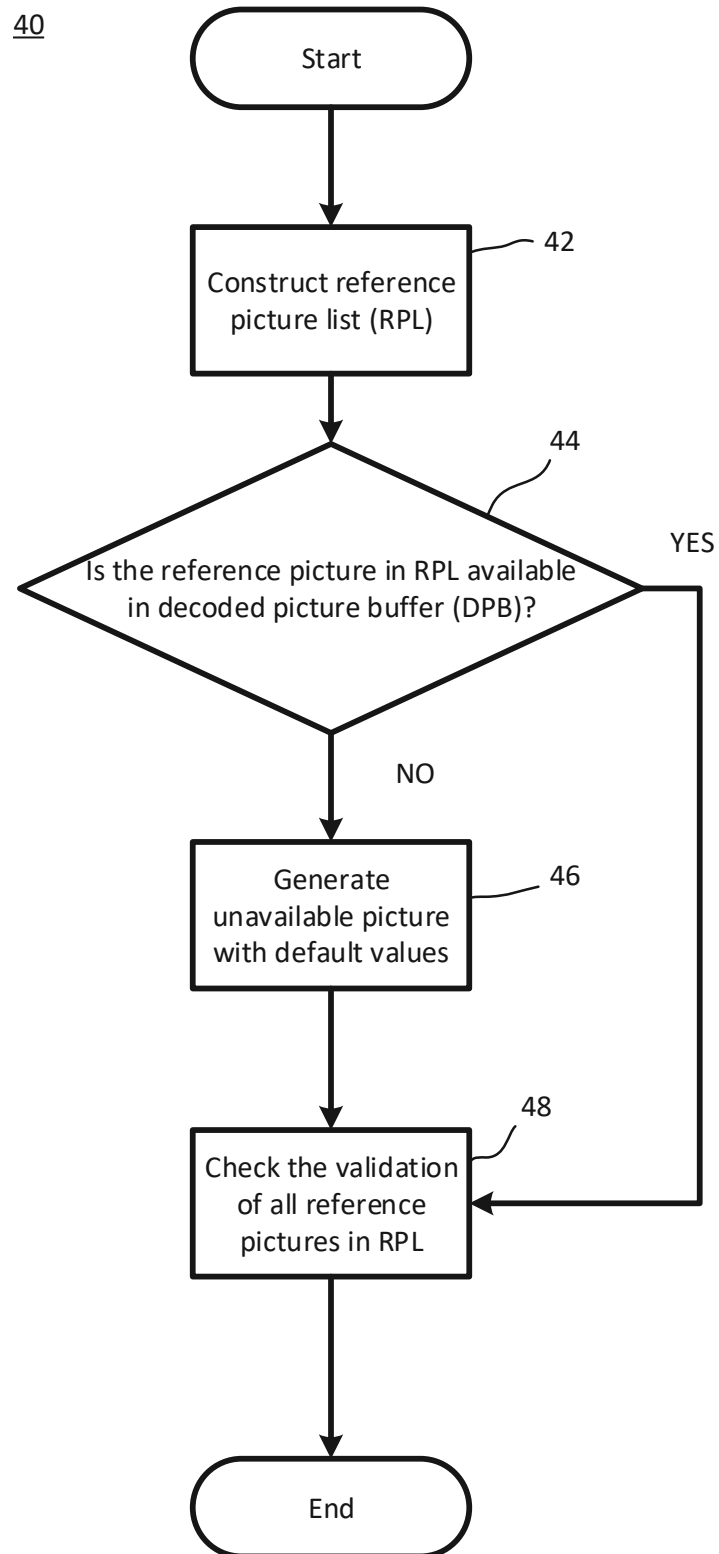


FIG. 33

