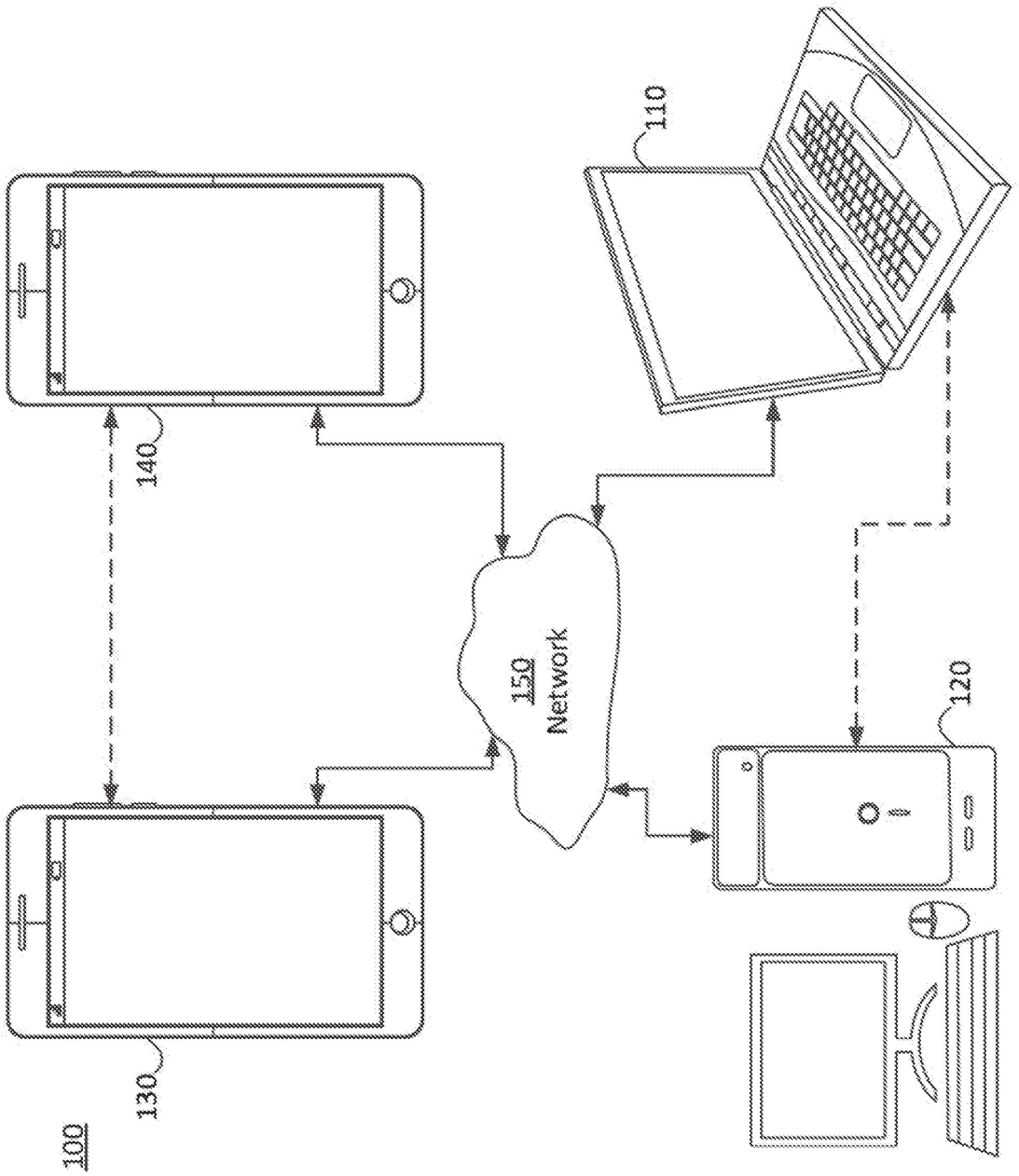


FIG. 1

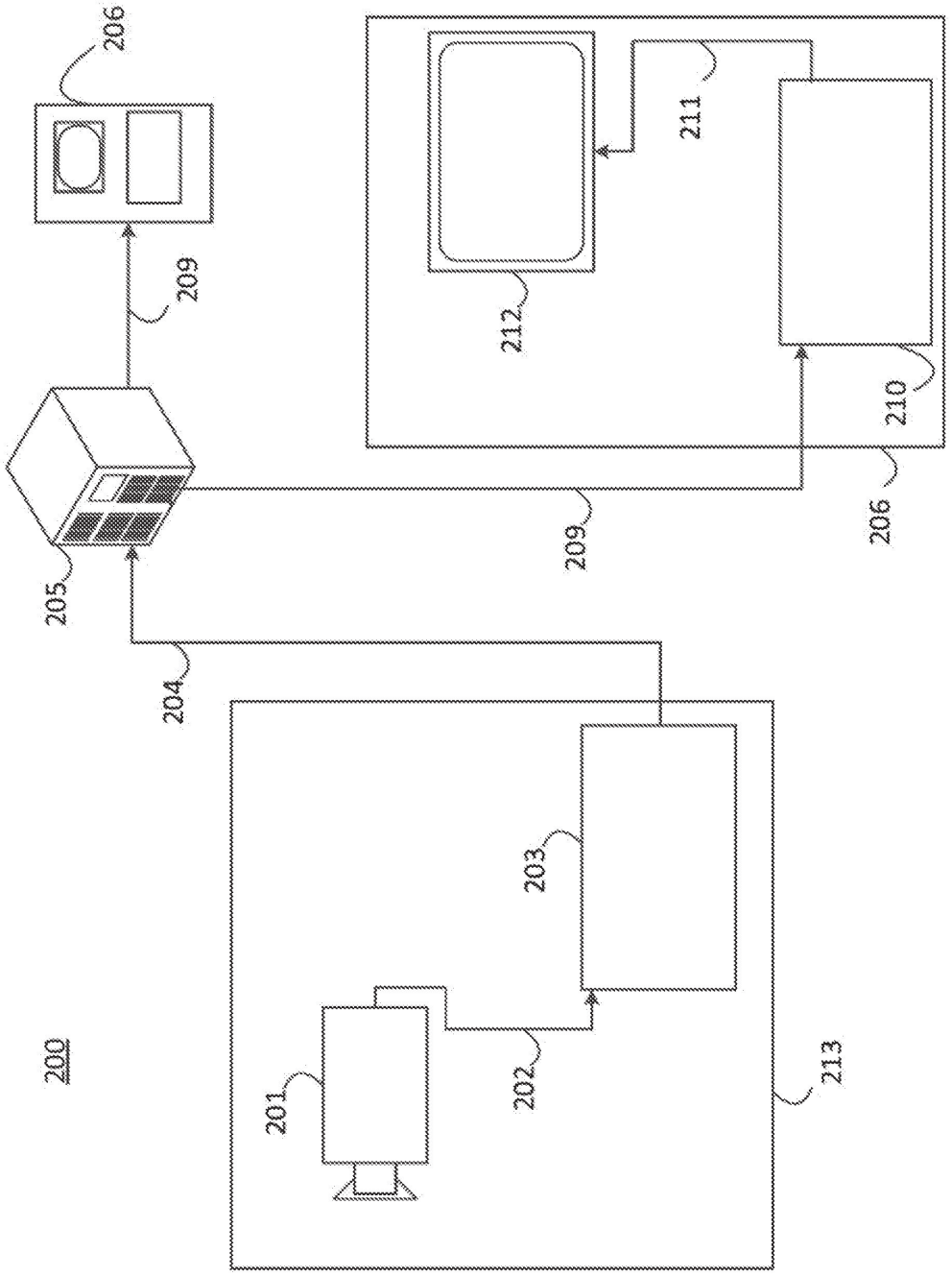
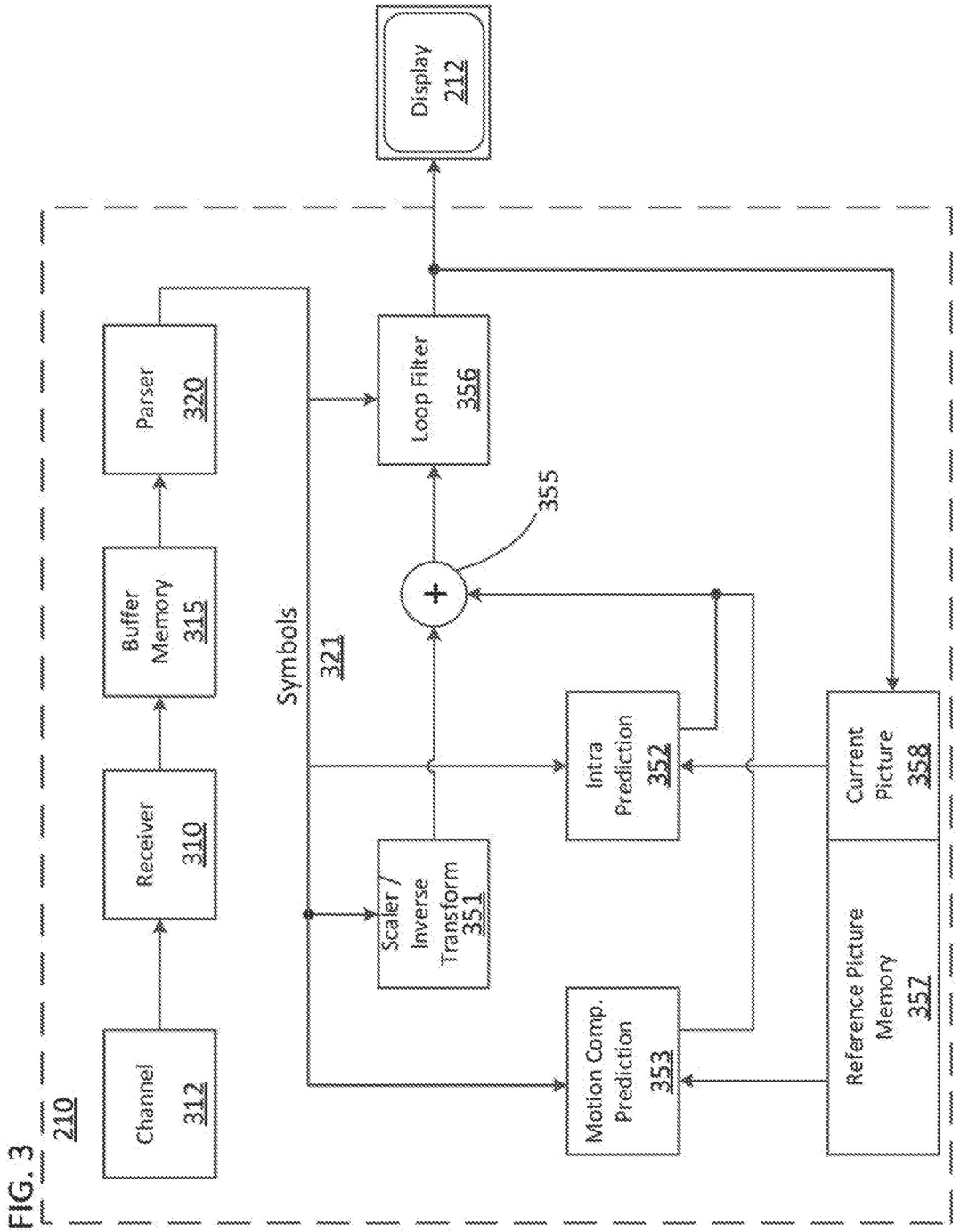


FIG. 2

FIG. 3


210
Channel
312
Receiver
310
Buffer
Memory
315
Parser
320
Symbols
321
Scaler /
Inverse
Transform
351
Intra
Prediction
352
Motion Comp.
Prediction
353
Loop Filter
356
355
Reference Picture
Memory
357
Current
Picture
358
Display
212

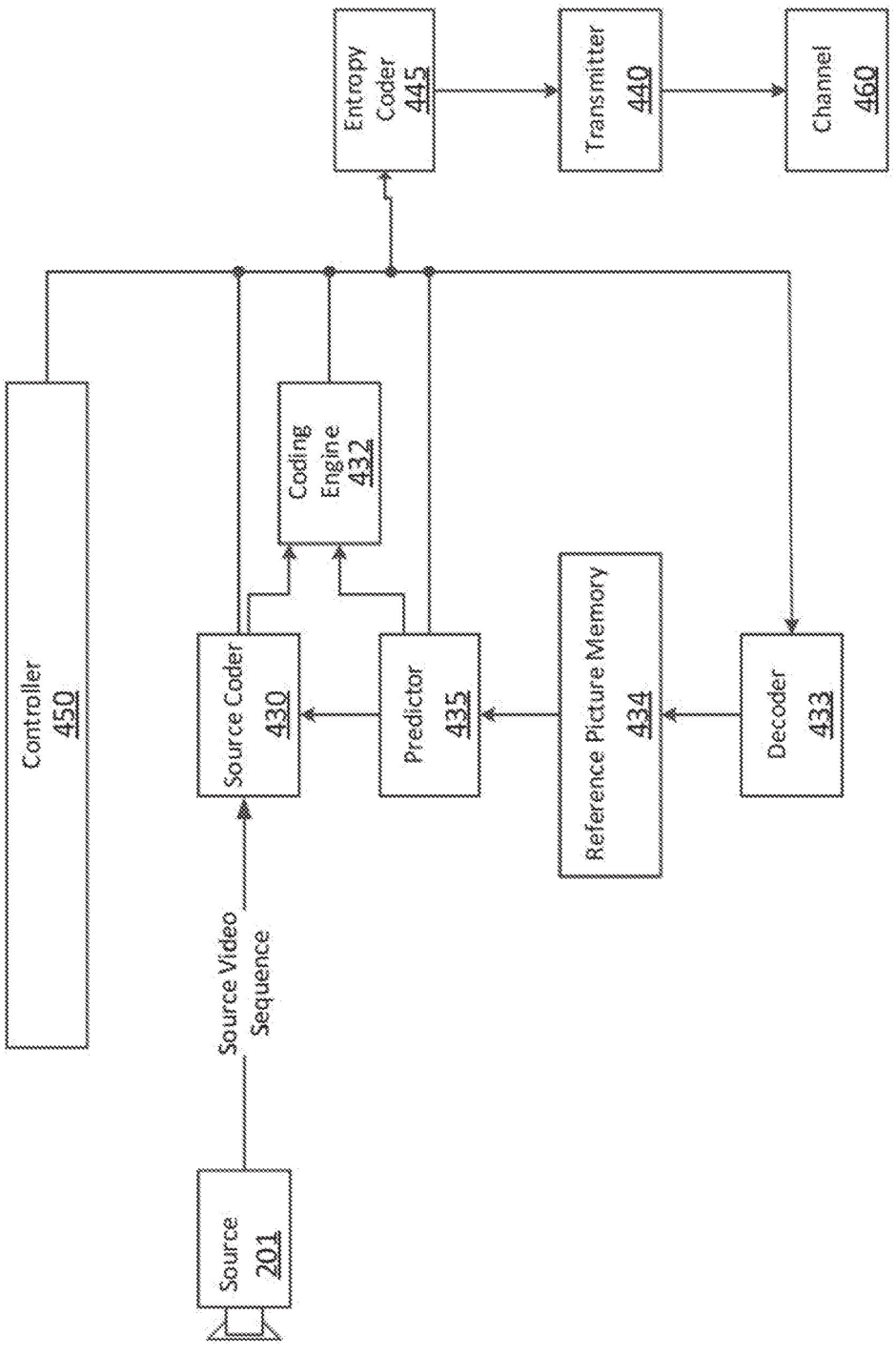

Source
201
Source Video
Sequence
Controller
450
Source Coder
430
Coding
Engine
432
Predictor
435
Reference Picture Memory
434
Decoder
433
Entropy
Coder
445
Transmitter
440
Channel
460

FIG. 4 203

FIG. 5A

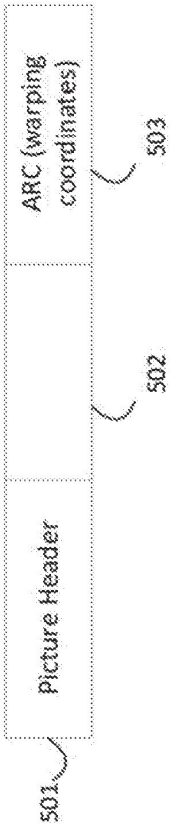

Picture Header
ARC (warping coordinates)
501
502
503

FIG. 5B

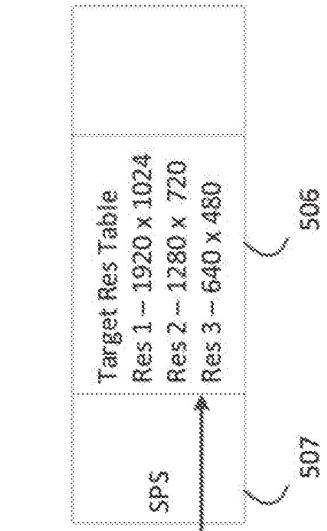

PPS
ARC Ref
SPS
Target Res Table
Res 1 – 1920 x 1024
Res 2 – 1280 x 720
Res 3 – 640 x 480
504
505
506
507

FIG. 6A

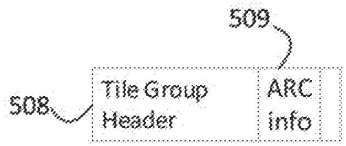


FIG. 6B

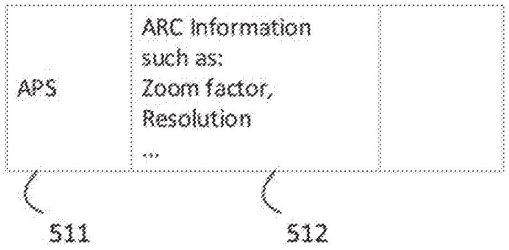


FIG. 6C

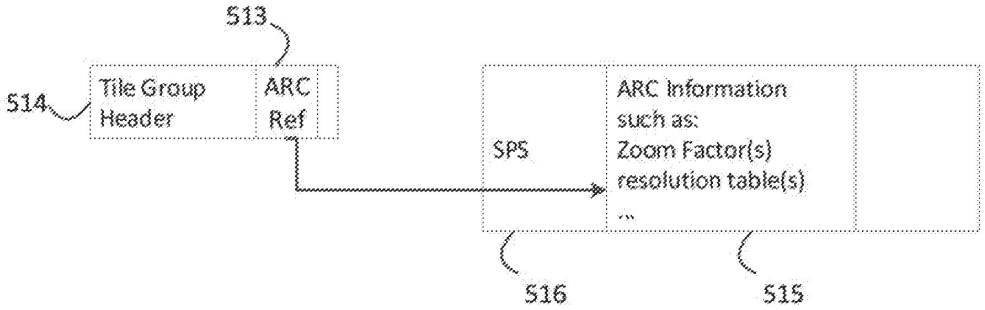


FIG. 7A

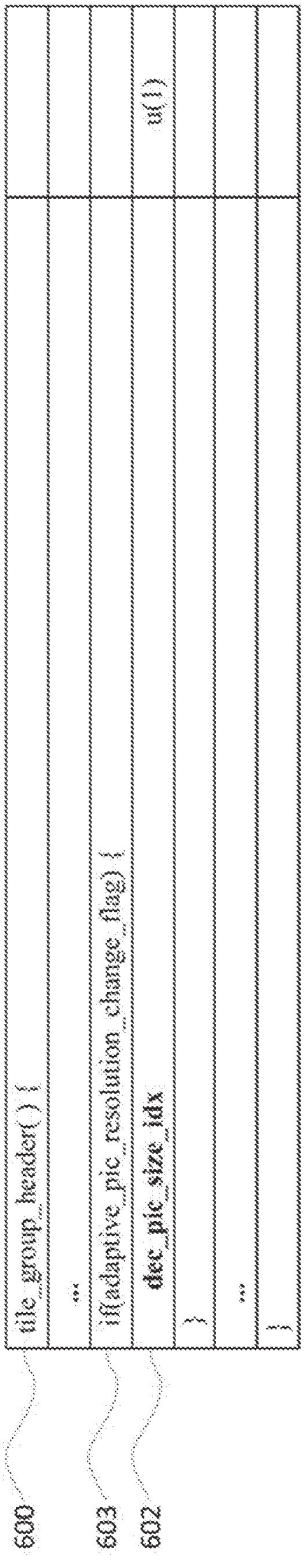
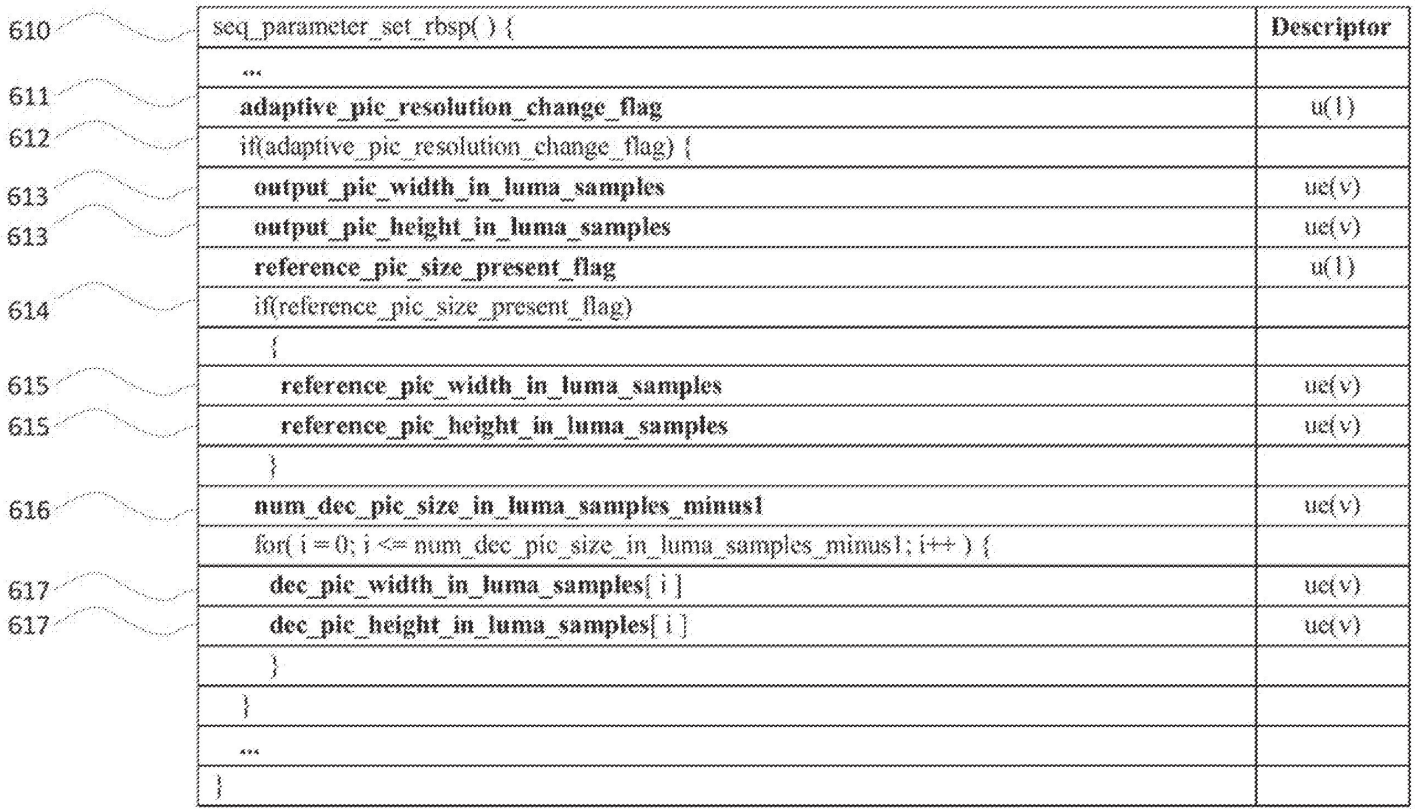

600
603
602
tile_group_header( ) {
...
if(adaptive_pic_resolution_change_flag) {
dec_pic_size_idx
u( ))
}
...
}

FIG. 7B



	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

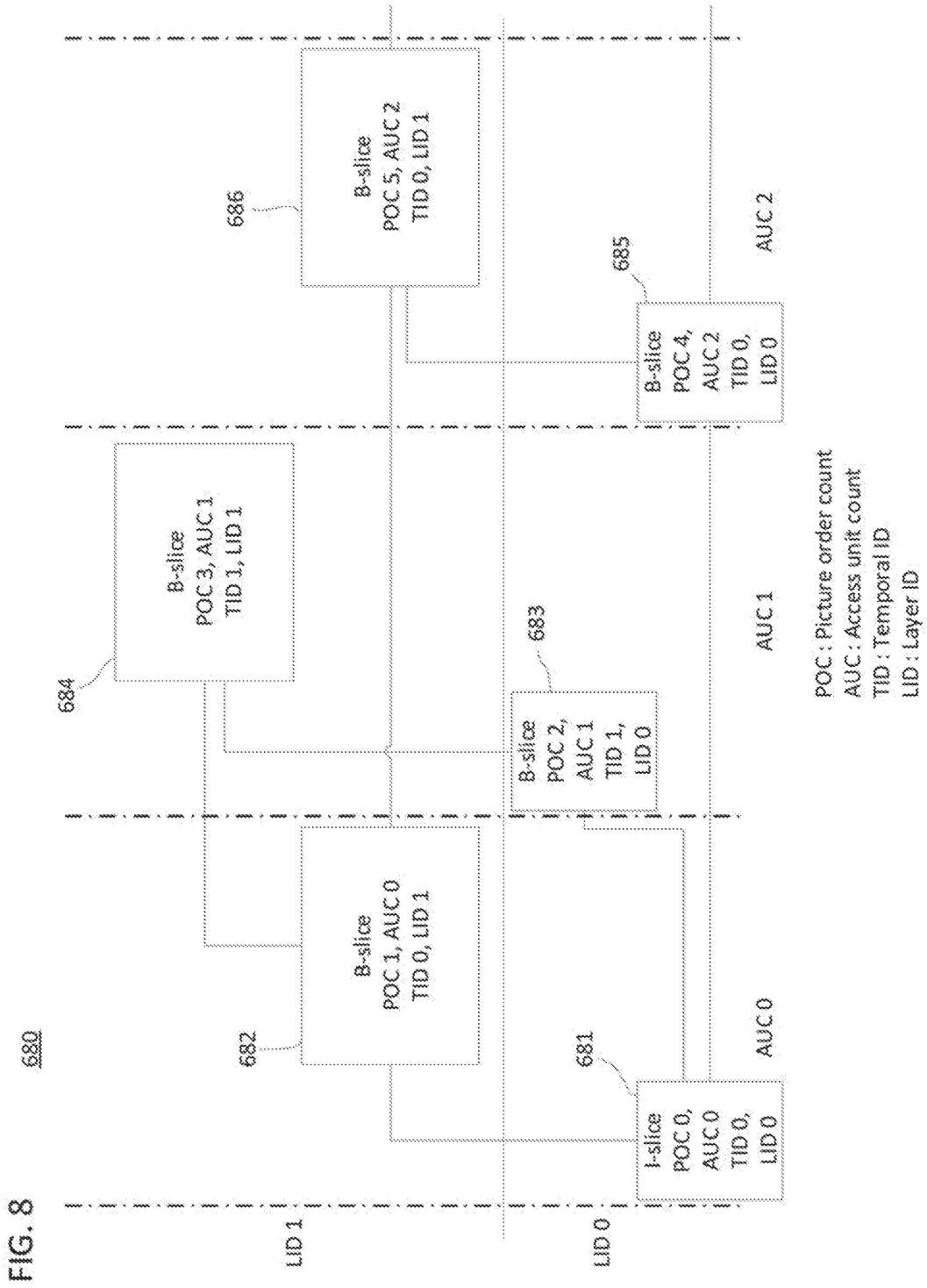
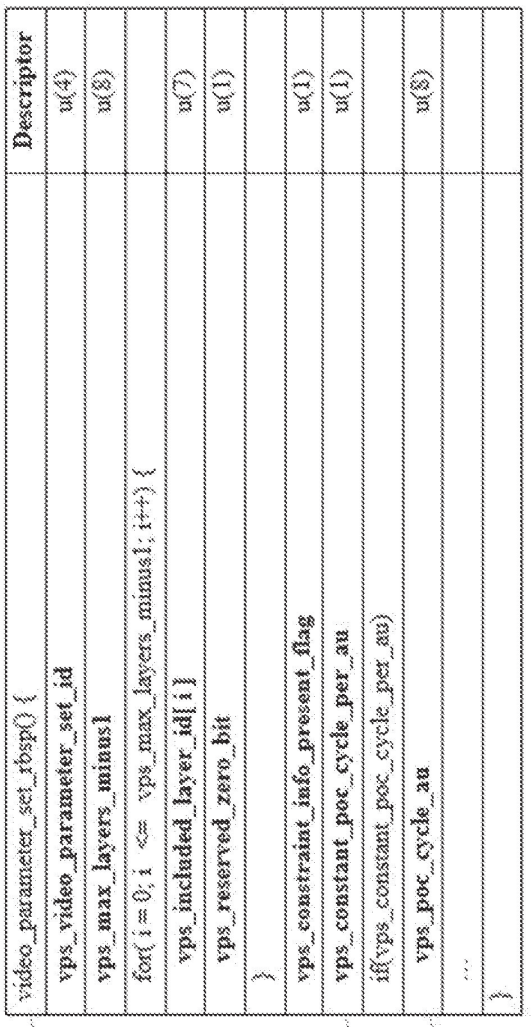

680
LID 1
LID 0
682
681
684
683
686
685
I-slice
POC 0,
AUC 0
TID 0,
LID 0
B-slice
POC 1, AUC 0
TID 0, LID 1
B-slice
POC 3, AUC 1
TID 1, LID 1
B-slice
POC 2,
AUC 1
TID 1,
LID 0
B-slice
POC 5, AUC 2
TID 0, LID 1
B-slice
POC 4,
AUC 2
TID 0,
LID 0
AUC 0
AUC 1
AUC 2
POC : Picture order count
AUC : Access unit count
TID : Temporal ID
LID : Layer ID

FIG. 9A



video_parameter_set_rbsp() {	Descriptor
vps_video_parameter_set_id	u(4)
vps_max_layers_minus1	u(8)
for(i = 0; i <= vps_max_layers_minus1; i++) {	
vps_included_layer_id[i]	u(7)
vps_reserved_zero_bit	u(1)
}	
vps_constraint_info_present_flag	u(1)
vps_constant_poc_cycle_per_au	u(1)
if(vps_constant_poc_cycle_per_au)	
vps_poc_cycle_au	u(8)
...	
}	

630

634

632

FIG. 9B

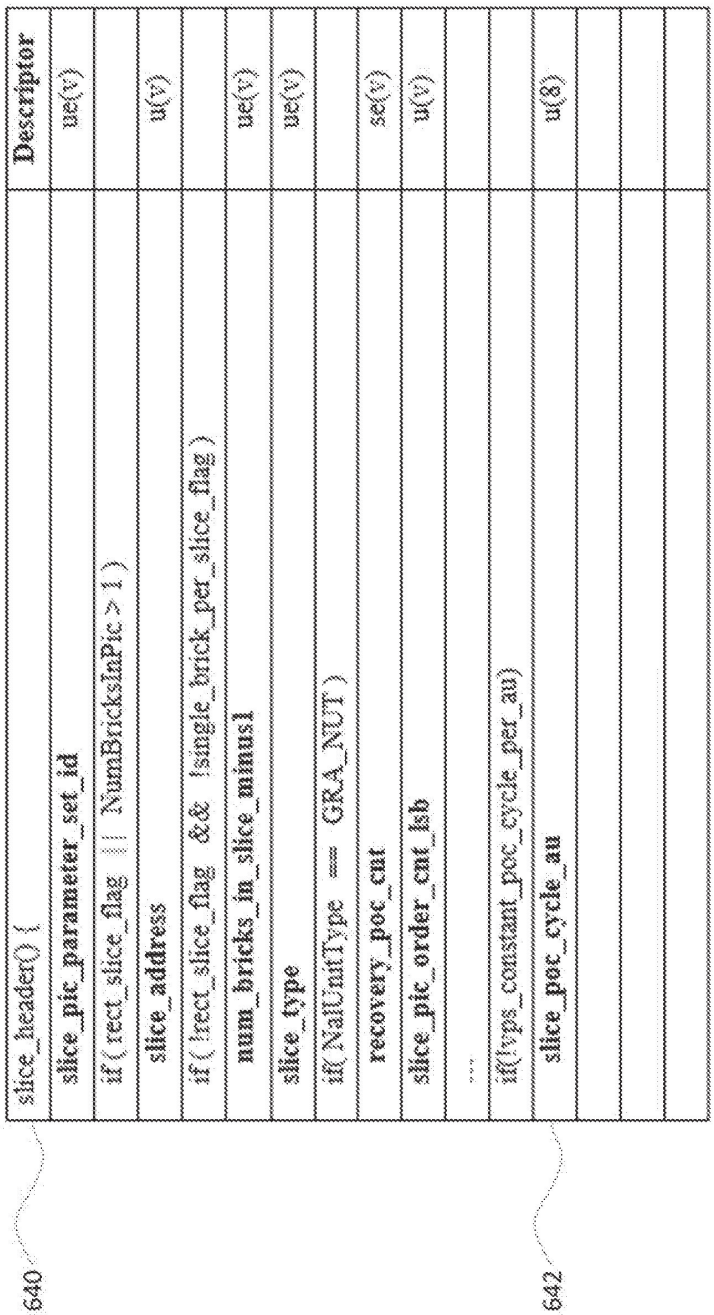

slice_header() {
slice_pic_parameter_set_id
if (rect_slice_flag || NumBricksInPic > 1)
slice_address
if (!rect_slice_flag && !single_brick_per_slice_flag)
num_bricks_in_slice_minus1
slice_type
if(NalUnitType == GRA_NUT)
recovery_poc_cnt
slice_pic_order_cnt_lsb
...
if(!vps_constant_poc_cycle_per_au)
slice_poc_cycle_au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

FIG. 10

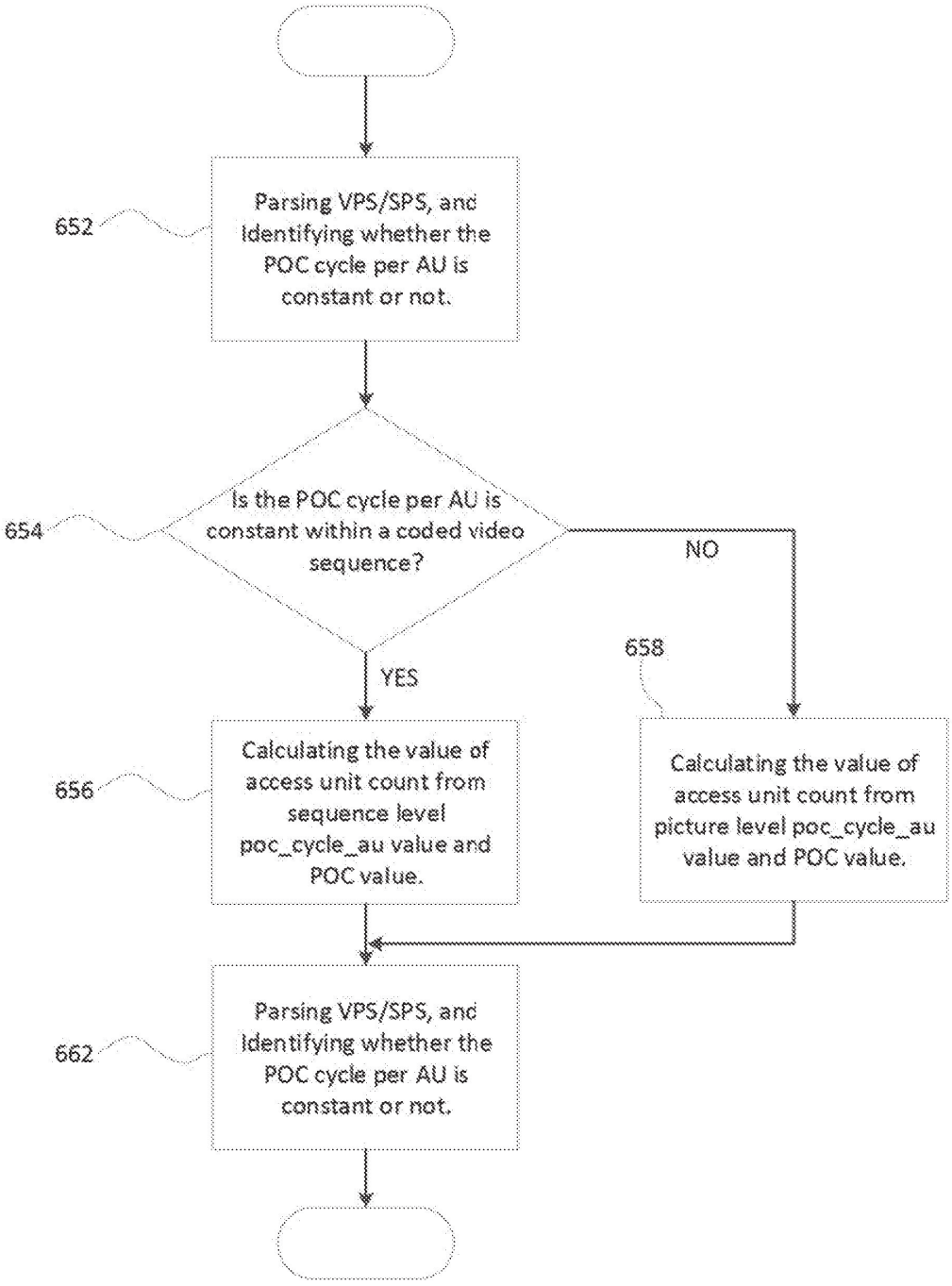
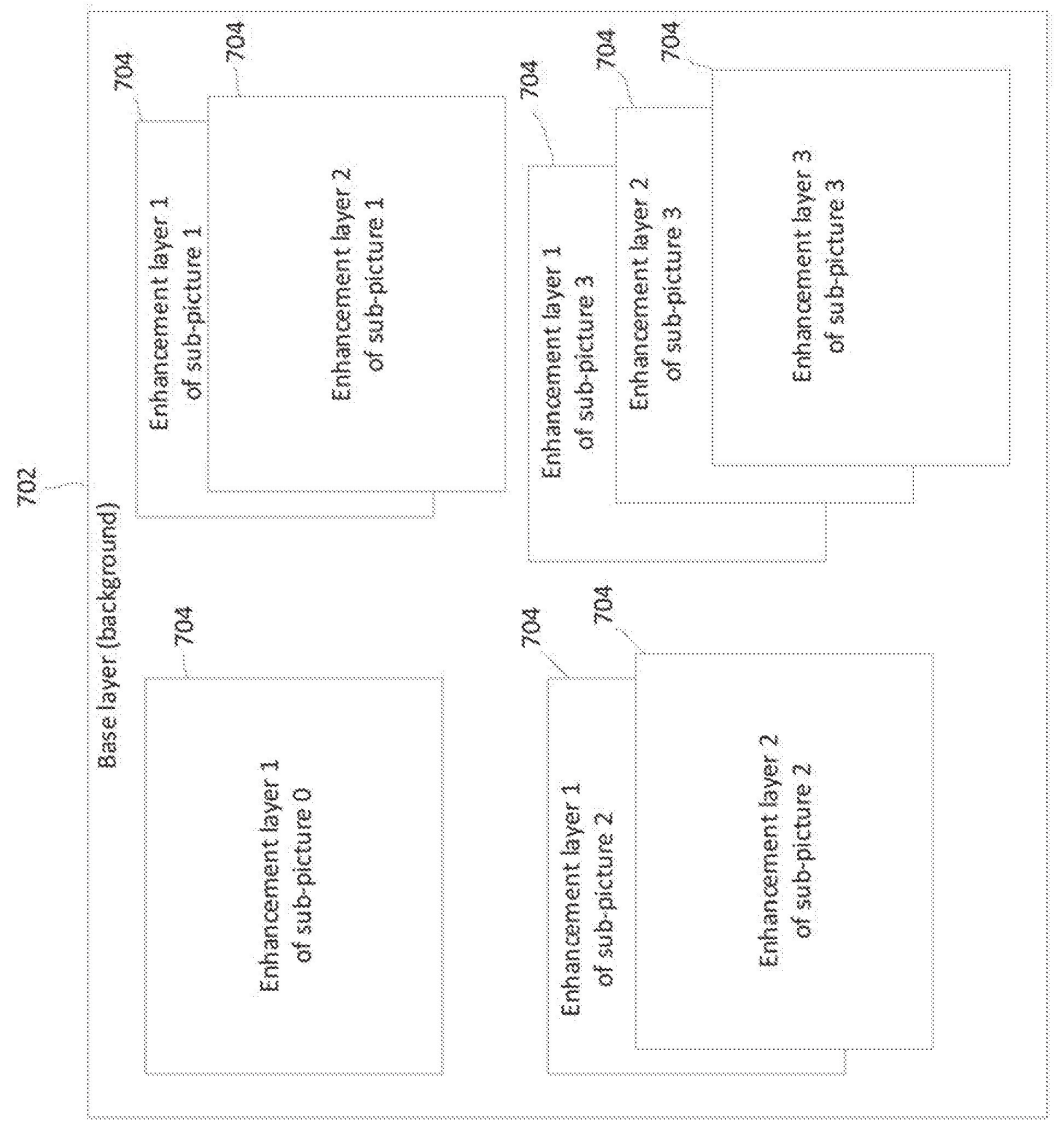

652
Parsing VPS/SPS, and Identifying whether the POC cycle per AU is constant or not.
654
Is the POC cycle per AU is constant within a coded video sequence?
NO
YES
658
656
Calculating the value of access unit count from sequence level poc_cycle_au value and POC value.
Calculating the value of access unit count from picture level poc_cycle_au value and POC value.
662
Parsing VPS/SPS, and Identifying whether the POC cycle per AU is constant or not.

FIG. 11


702
Base layer (background)
704
Enhancement layer 1
of sub-picture 0
704
Enhancement layer 1
of sub-picture 1
704
Enhancement layer 2
of sub-picture 1
704
Enhancement layer 1
of sub-picture 2
704
Enhancement layer 2
of sub-picture 2
704
Enhancement layer 1
of sub-picture 3
704
Enhancement layer 2
of sub-picture 3
704
Enhancement layer 3
of sub-picture 3

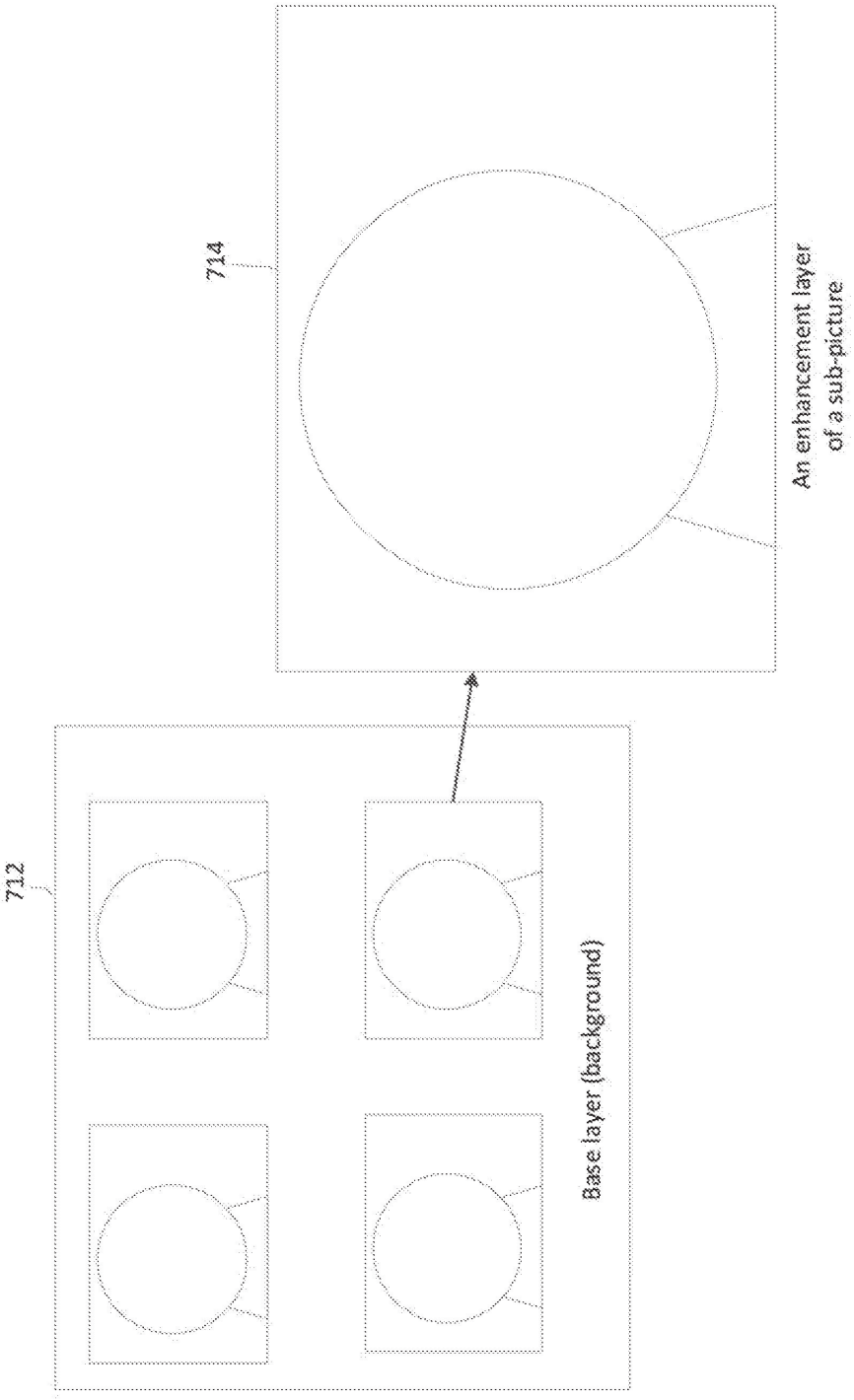

712
714
Base layer (background)
An enhancement layer
of a sub-picture

FIG. 12

FIG. 13

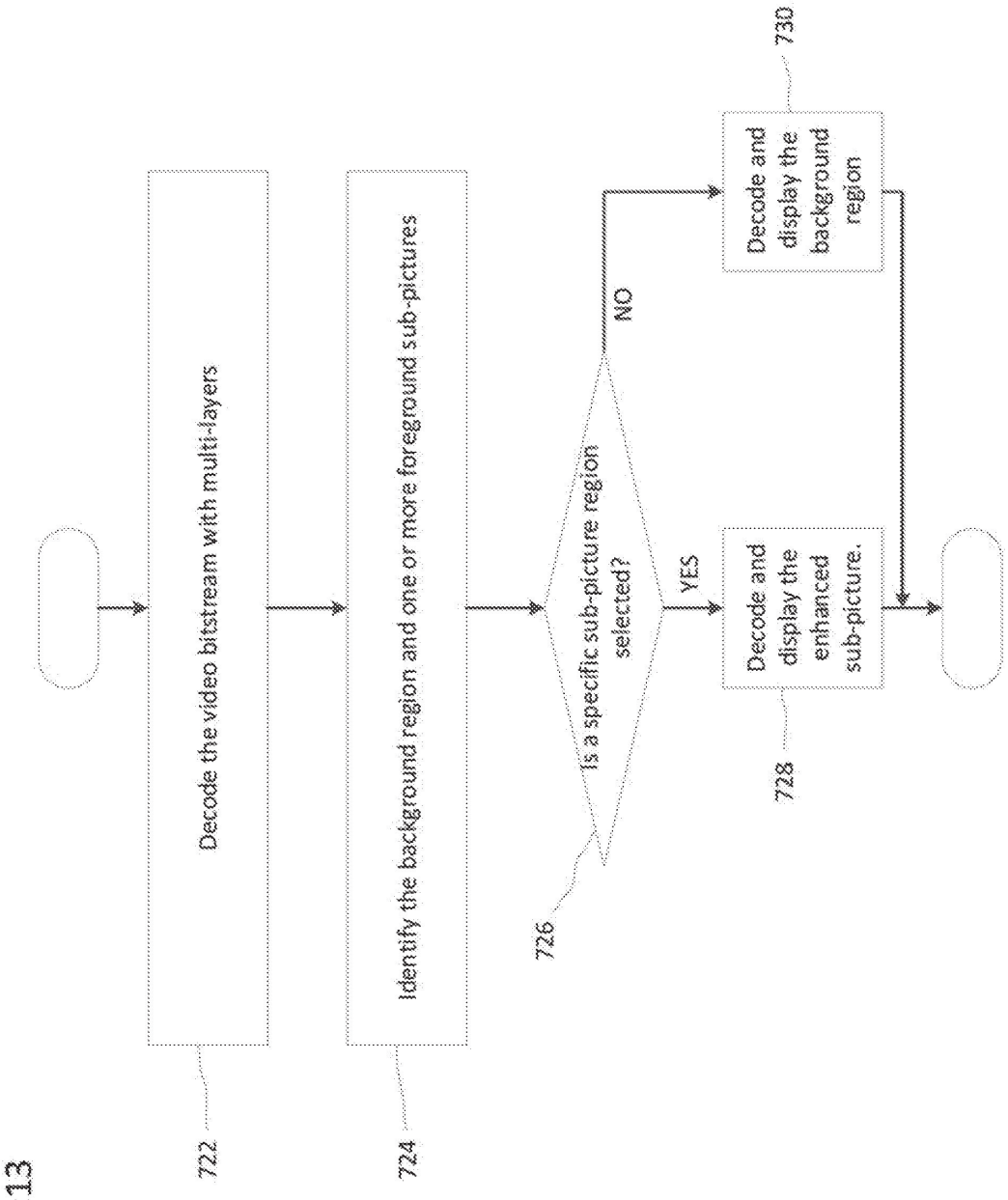

Decode the video bitstream with multi-layers
722
Identify the background region and one or more foreground sub-pictures
724
Is a specific sub-picture region selected?
726
NO
YES
Decode and display the enhanced sub-picture.
728
Decode and display the background region
730

FIG. 14

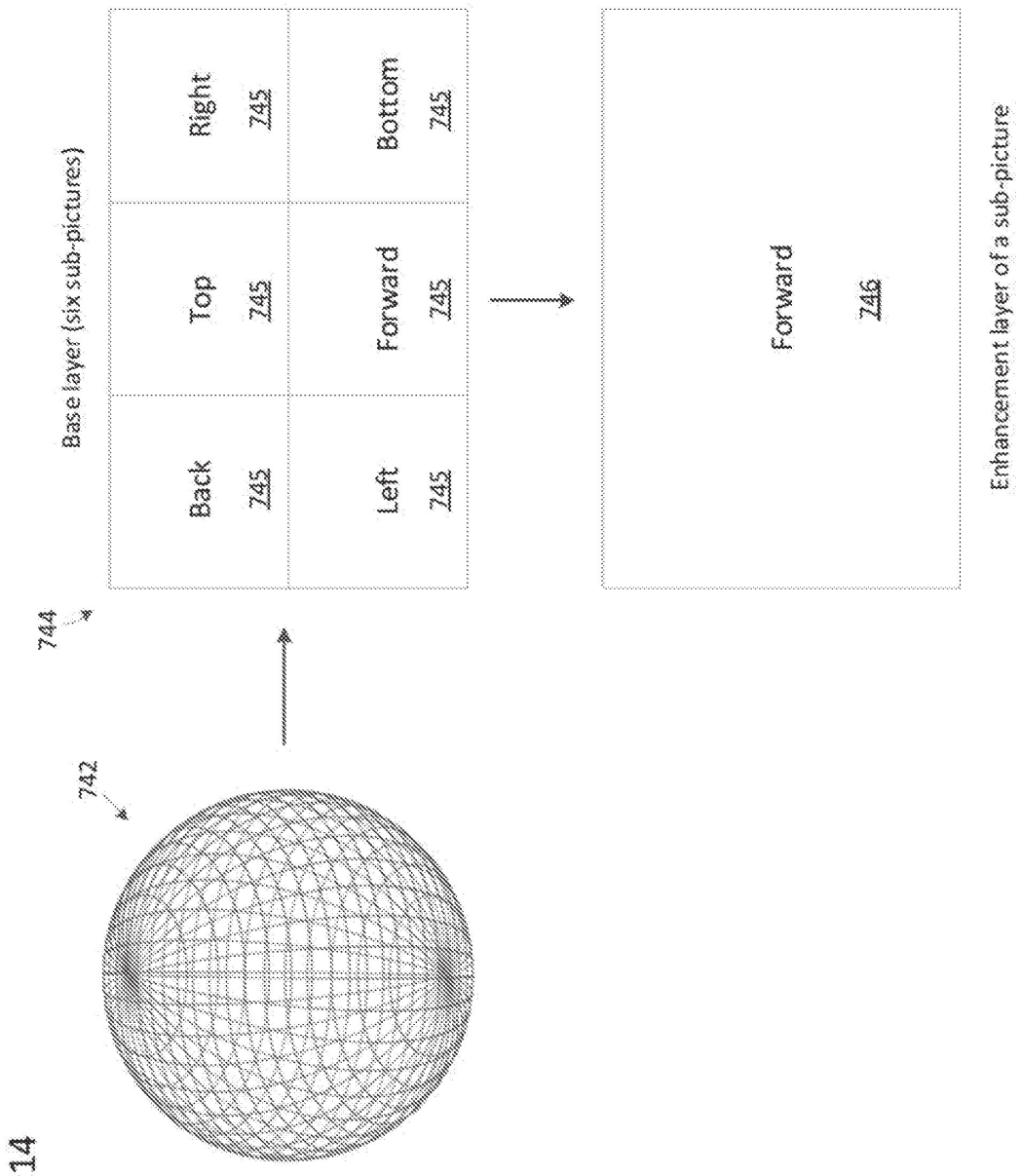

742
744
Base layer (six sub-pictures)
Back 745
Top 745
Right 745
Left 745
Forward 745
Bottom 745
Forward 746
Enhancement layer of a sub-picture

FIG. 15A

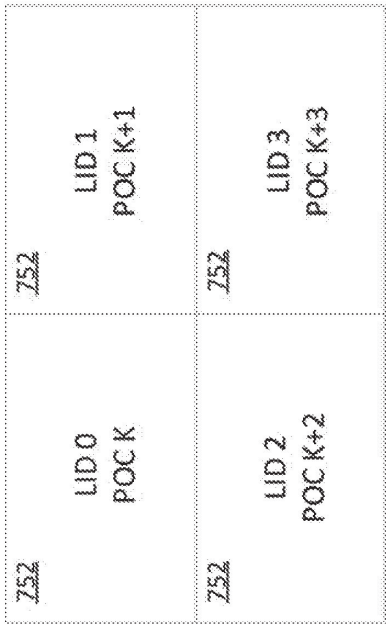


FIG. 15B

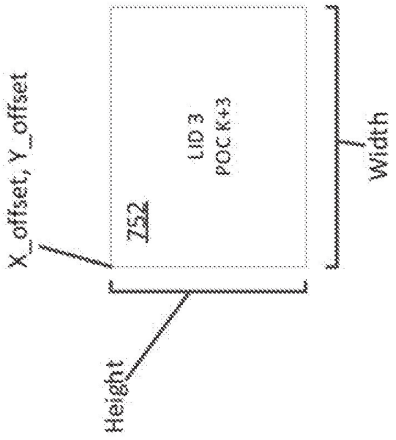
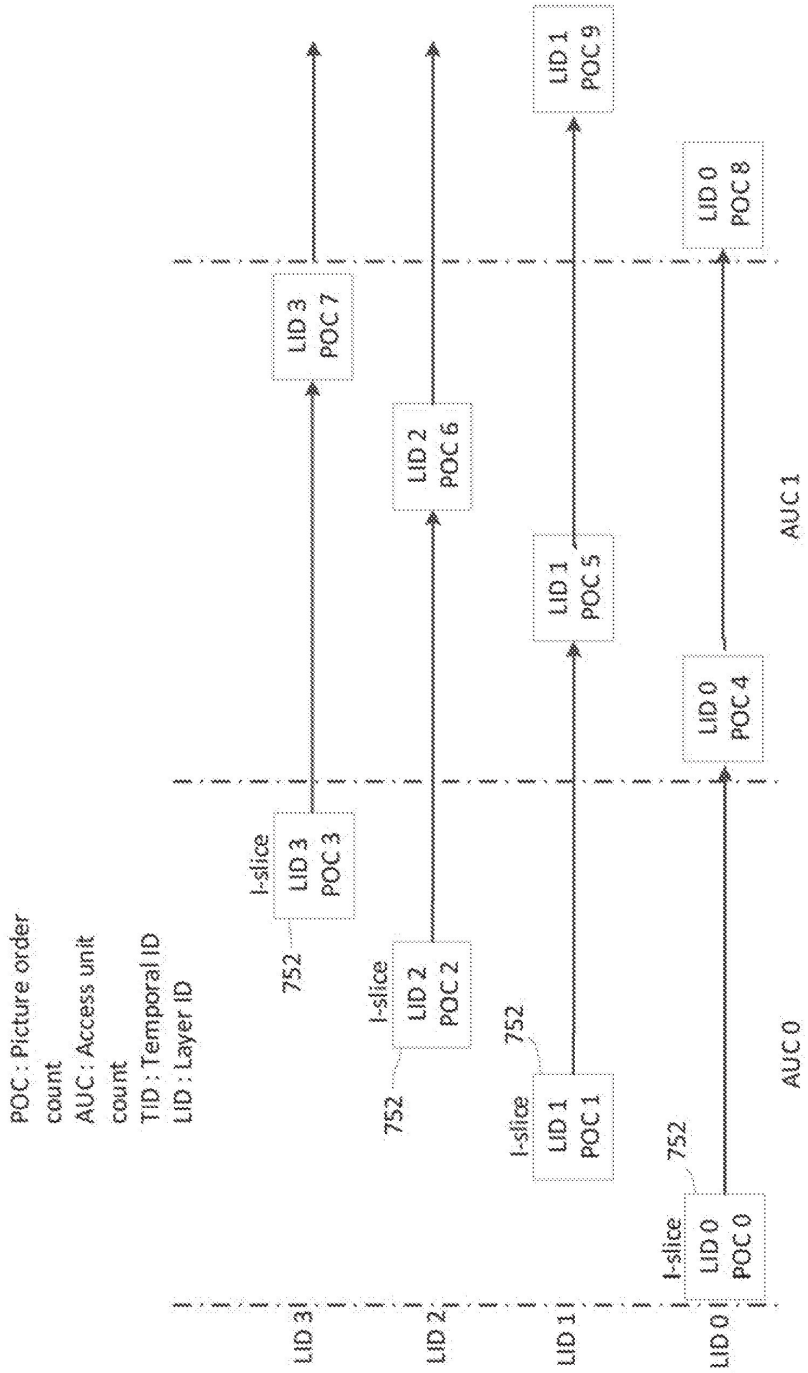


FIG. 16


POC : Picture order count
AUC : Access unit count
TID : Temporal ID
LID : Layer ID
LID 3
LID 2
LID 1
LID 0
I-slice
LID 0 POC 0
LID 1 POC 1
LID 2 POC 2
LID 3 POC 3
LID 0 POC 4
LID 1 POC 5
LID 2 POC 6
LID 3 POC 7
LID 0 POC 8
LID 1 POC 9
752
AUC 0
AUC 1

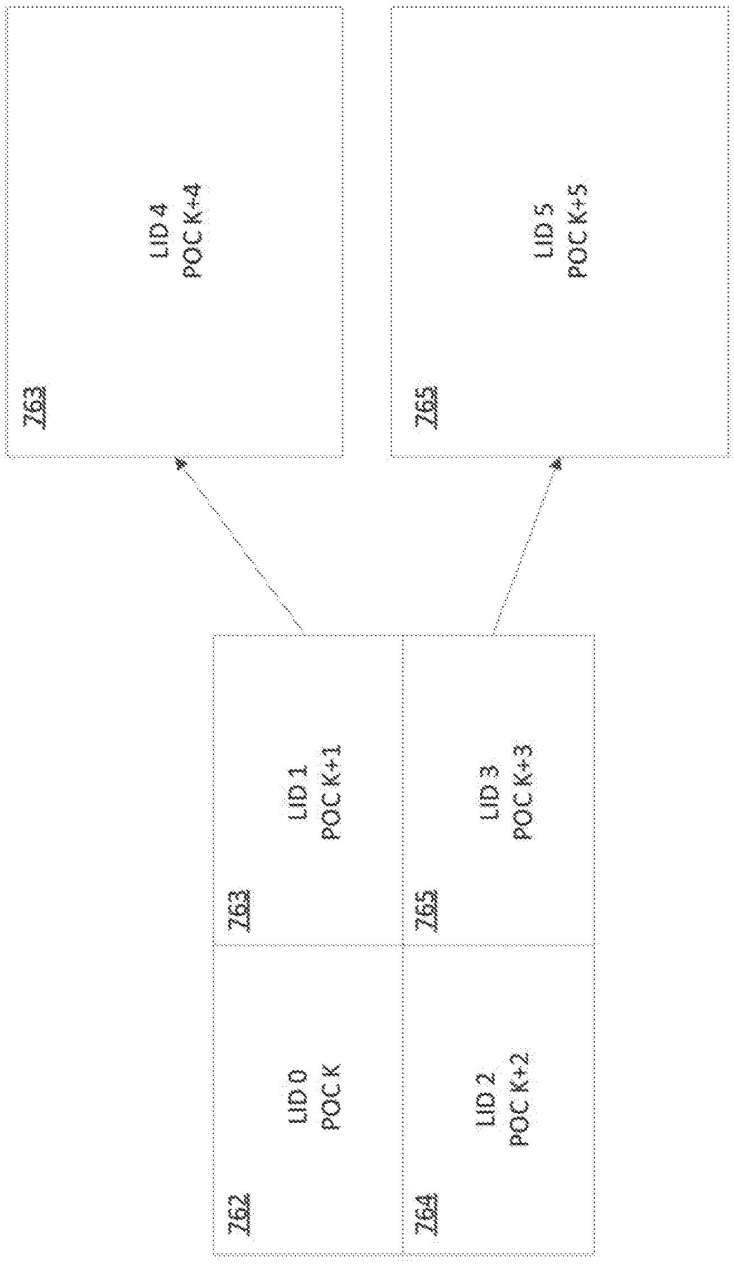

Z63
LID 4
POC K+4
Z65
LID 5
POC K+5
Z63
LID 1
POC K+1
Z65
LID 3
POC K+3
Z62
LID 0
POC K
Z64
LID 2
POC K+2

FIG. 17

FIG. 18

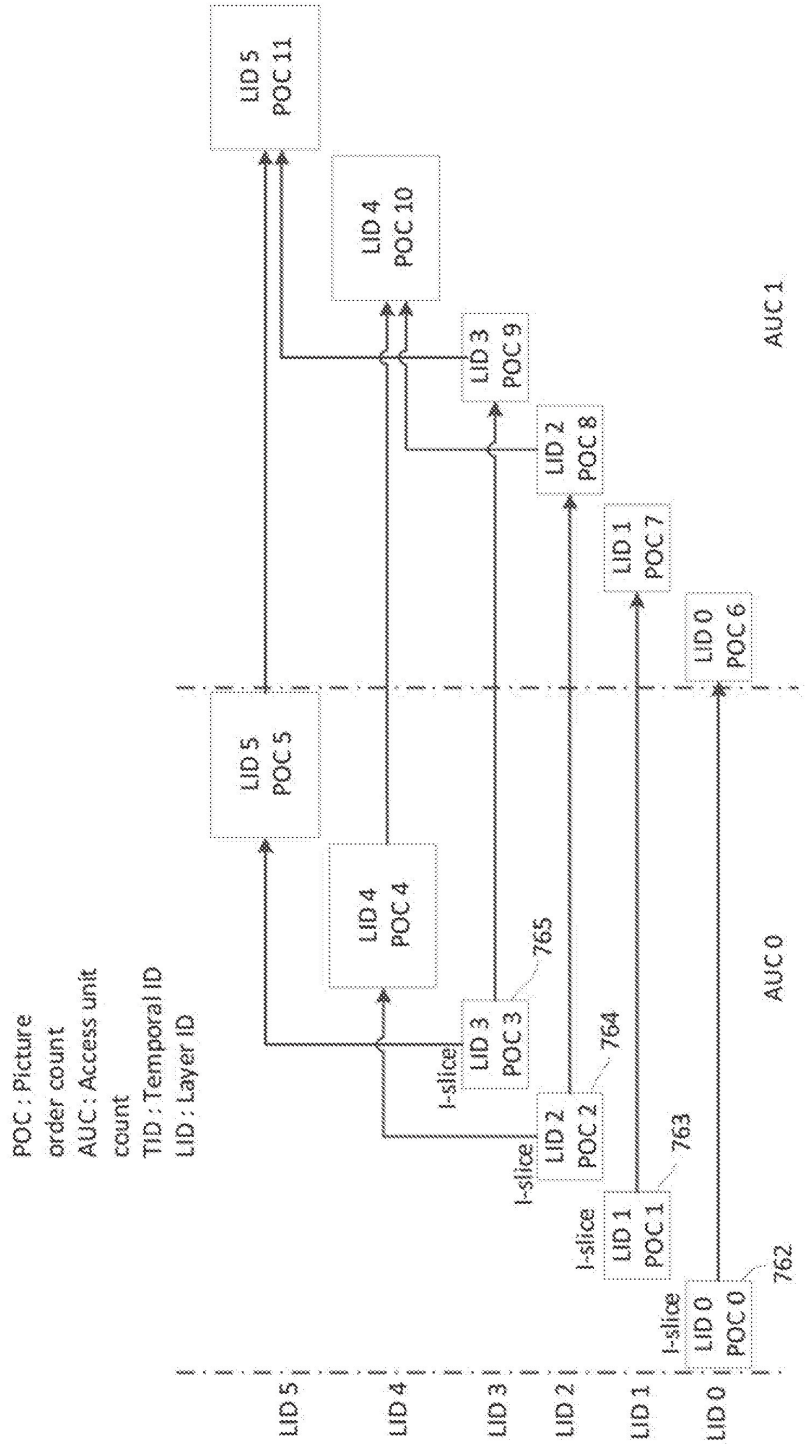
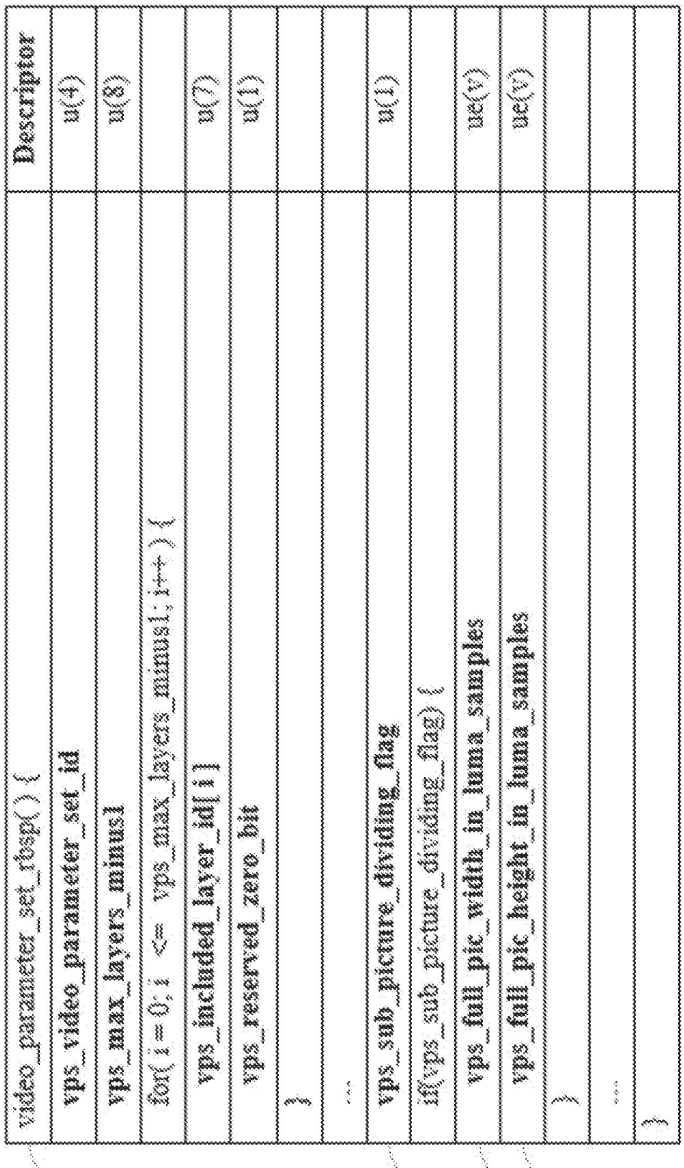

POC : Picture order count
AUC : Access unit count
TID : Temporal ID
LID : Layer ID
LID 5
LID 4
LID 3
LID 2
LID 1
LID 0
LID 0 POC 0
I-slice
LID 1 POC 1
I-slice
LID 2 POC 2
I-slice
LID 3 POC 3
LID 4 POC 4
LID 5 POC 5
LID 0 POC 6
LID 1 POC 7
LID 2 POC 8
LID 3 POC 9
LID 4 POC 10
LID 5 POC 11
762
763
764
765
AUC 0
AUC 1

FIG. 19A



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

FIG. 19B

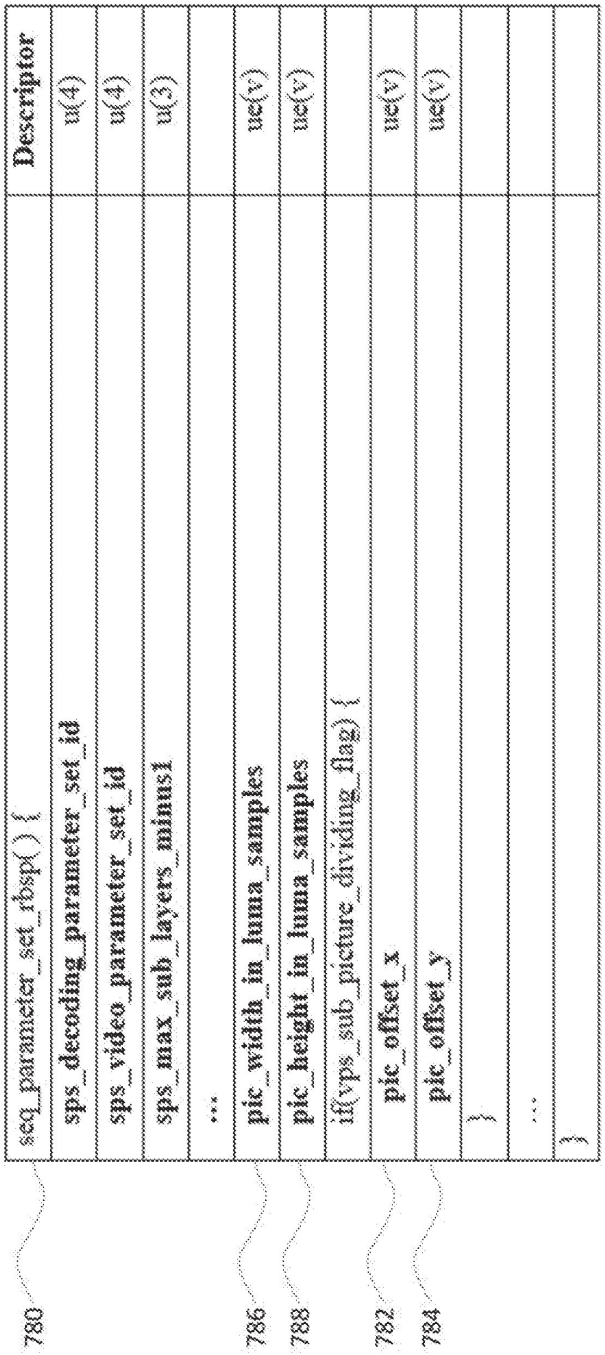
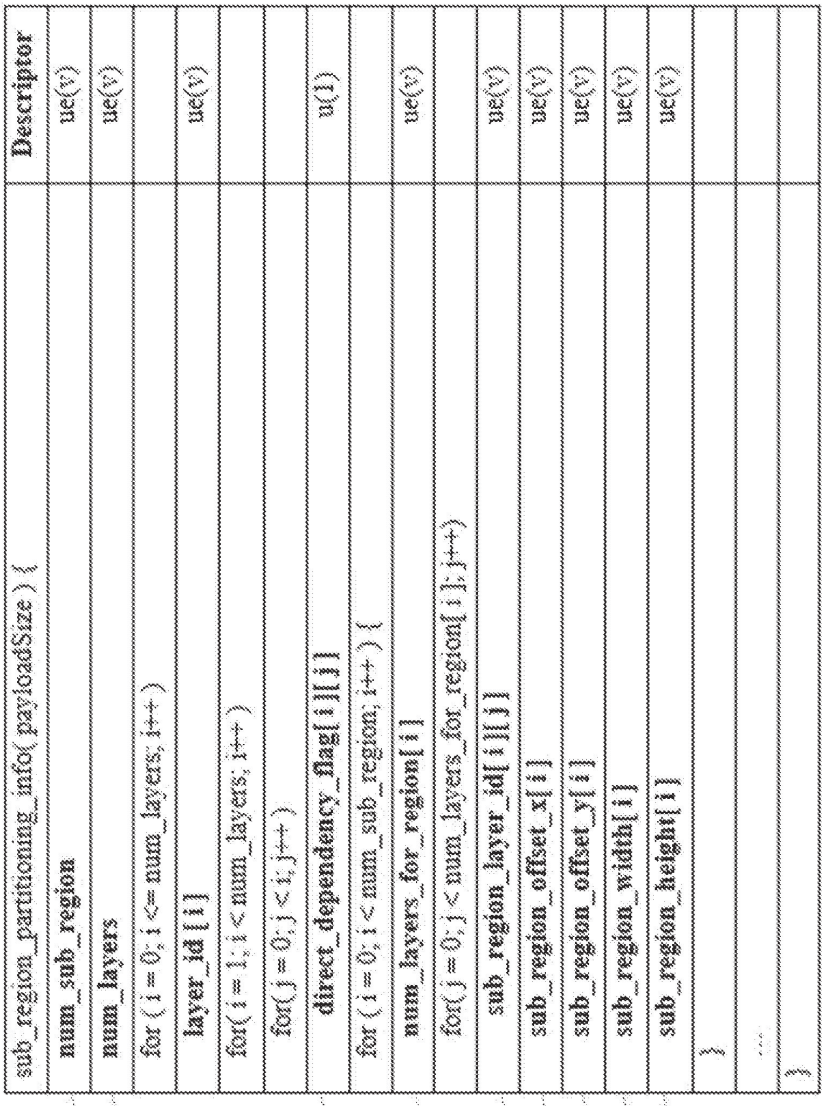
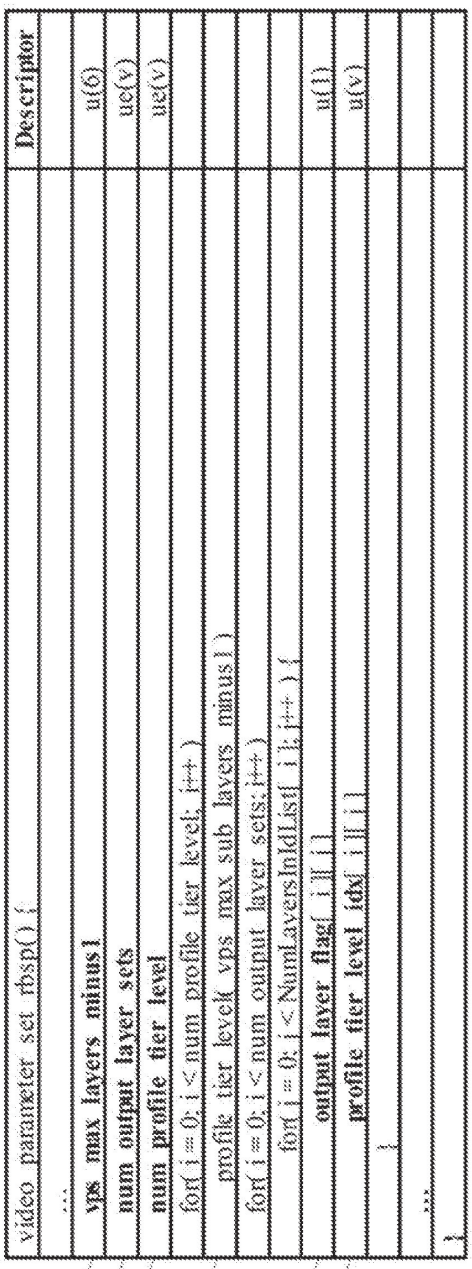

780
786
788
782
784
seq_parameter_set_rbsp( ) {
sps_decoding_parameter_set_id
sps_video_parameter_set_id
sps_max_sub_layers_minus1
...
pic_width_in_luma_samples
pic_height_in_luma_samples
if(vps_sub_picture_dividing_flag) {
pic_offset_x
pic_offset_y
}
...
}
Descriptor
u(4)
u(4)
u(3)
ue(v)
ue(v)
ue(v)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

FIG. 20



	sub_region_partitioning_info(payloadSize) {	Descriptor
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
...	
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(i = 0; i < NumLayersInIdList[i]; i++) {	
output_layer_flag[i][i]	u(1)
profile_tier_level_idx[i][i]	u(v)
}	
...	
}	

FIG. 22

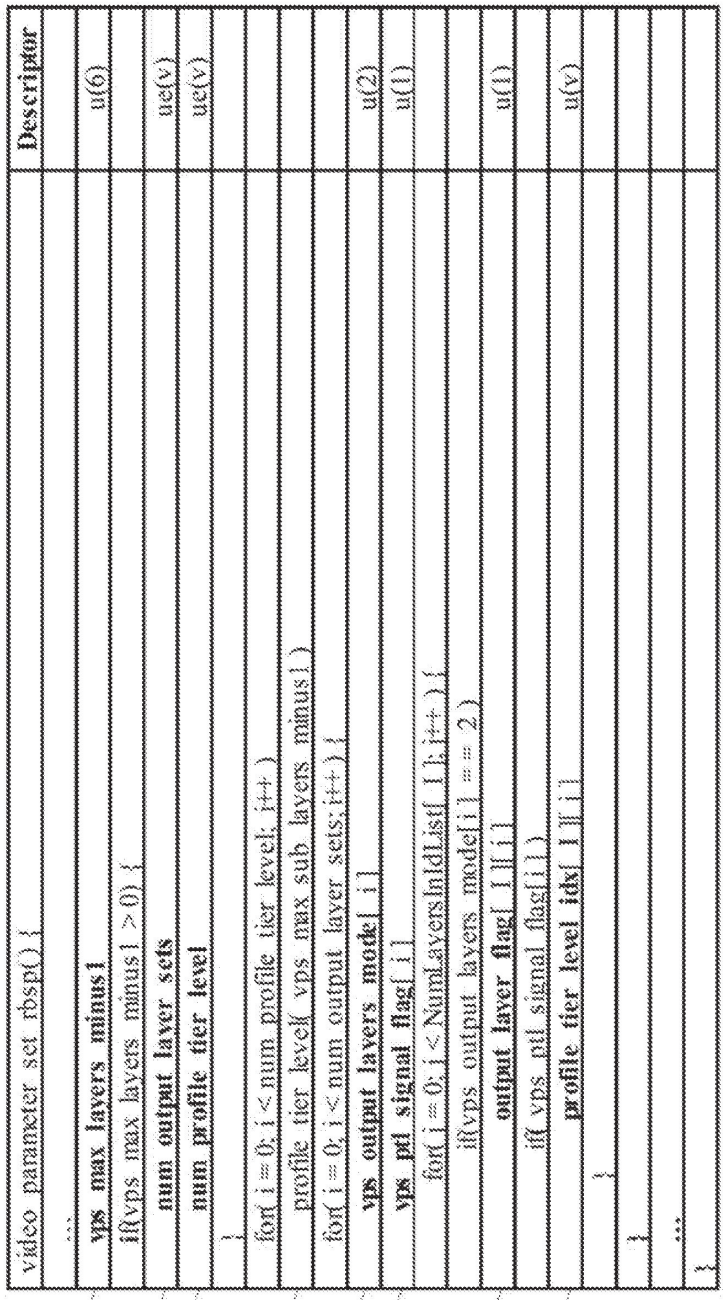
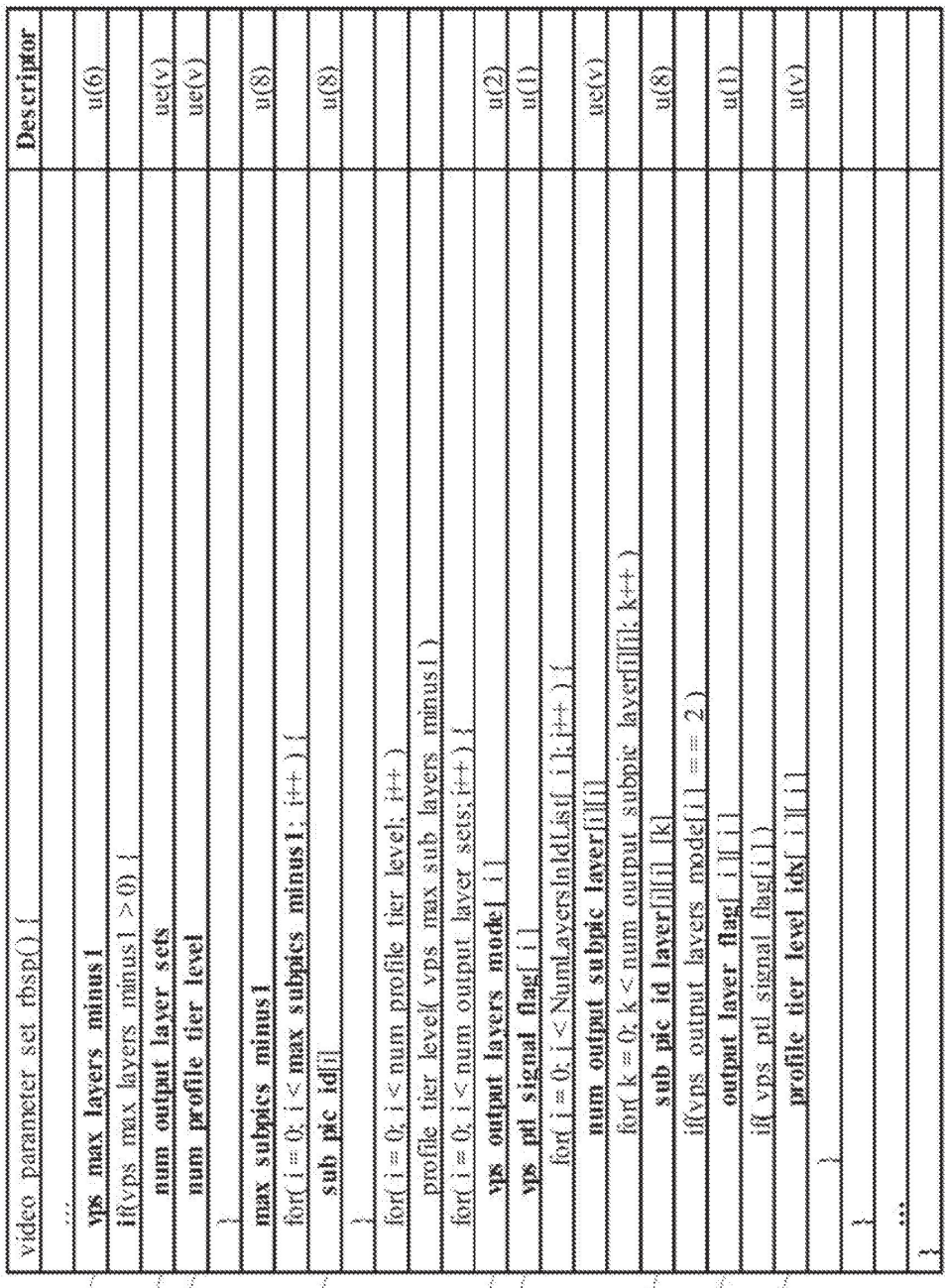

video parameter set rbsp( ) {
Descriptor
...
vps_max_layers_minus1
u(6)
if(vps_max_layers_minus1 > 0) {
num_output_layer_sets
ue(v)
num_profile_tier_level
ue(v)
}
for(i = 0; i < num_profile_tier_level; i++)
profile_tier_level(vps_max_sub_layers_minus1)
for(i = 0; i < num_output_layer_sets; i++) {
vps_output_layers_mode[i]
u(2)
vps_ptl_signal_flag[i]
u(1)
for(i = 0; i < NumLayersInIdList[i]; i++) {
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)
}
}
...
}
802
804
806
808
822
824
810
812

FIG. 23


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

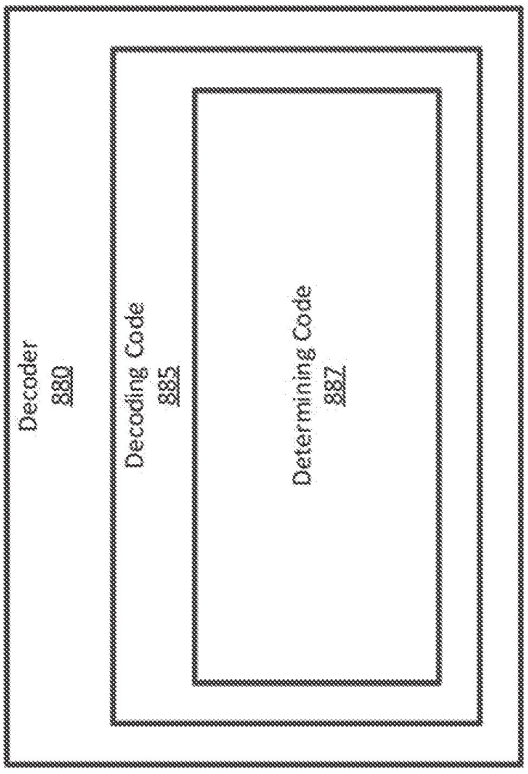


FIG. 24

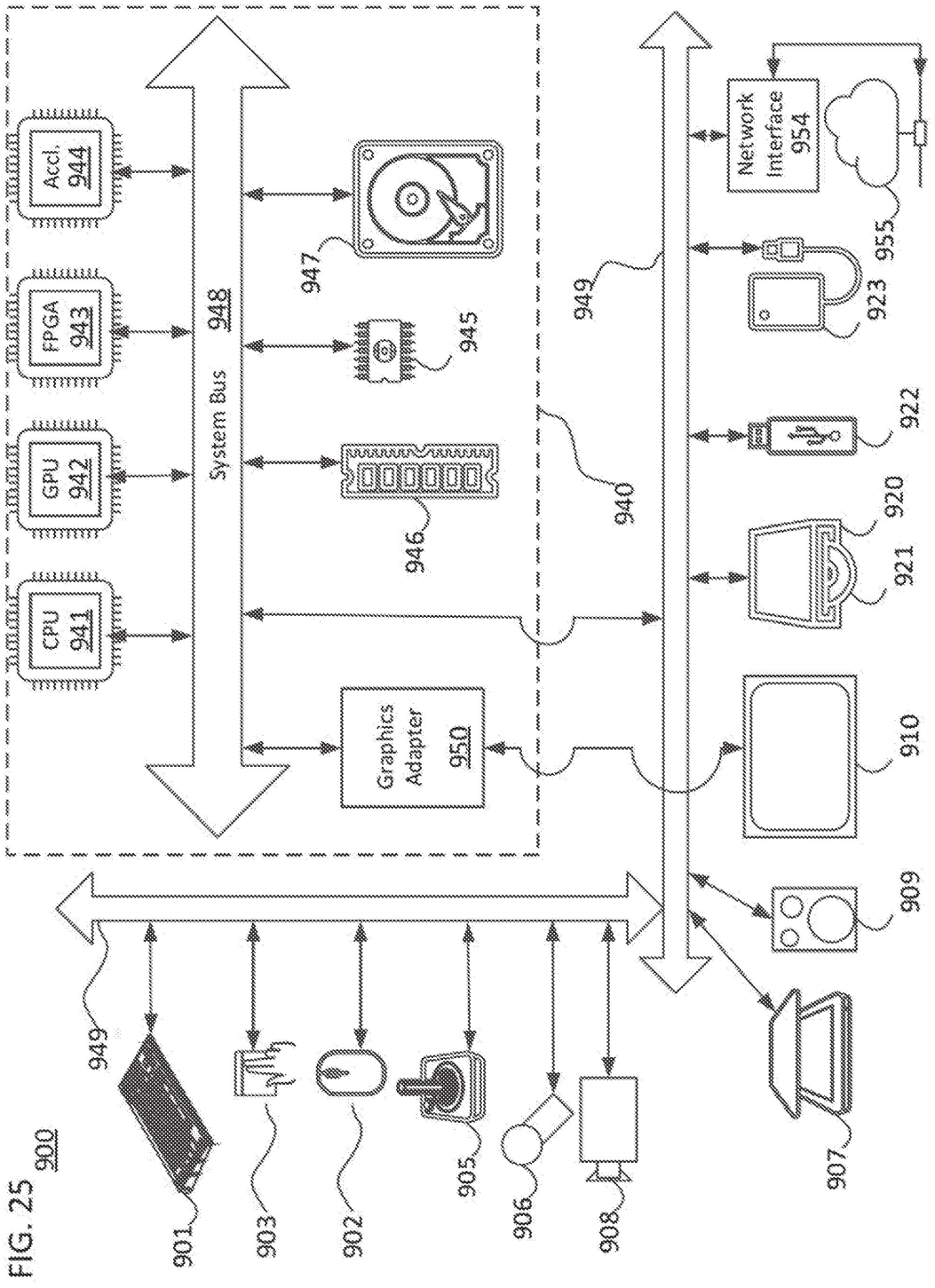

900
CPU 941
GPU 942
FPGA 943
Accl. 944
System Bus 948
Graphics Adapter 950
946
947
945
940
949
Network Interface 954
955
923
922
920
921
910
909
907
901
903
902
905
906
908

FIG. 25