

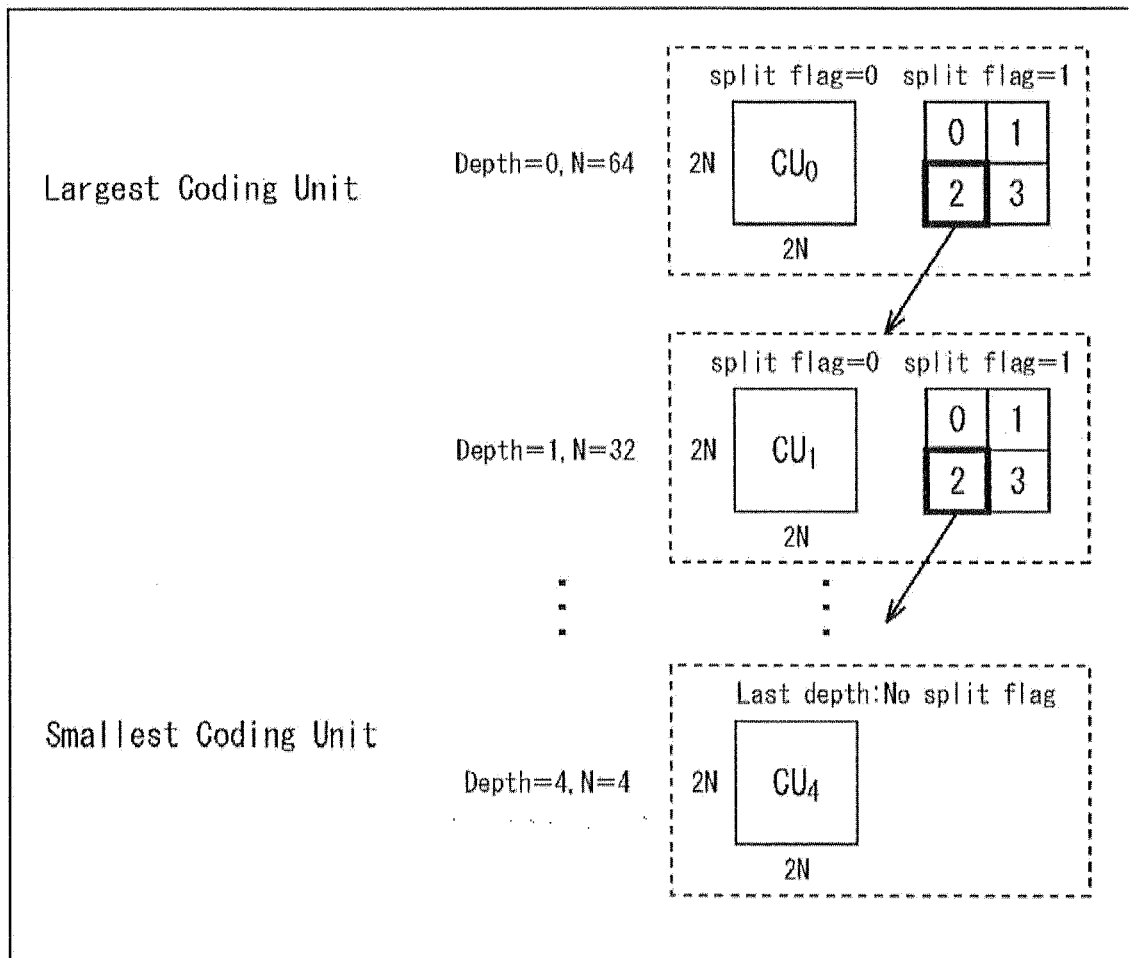
FIG. 1

FIG. 2

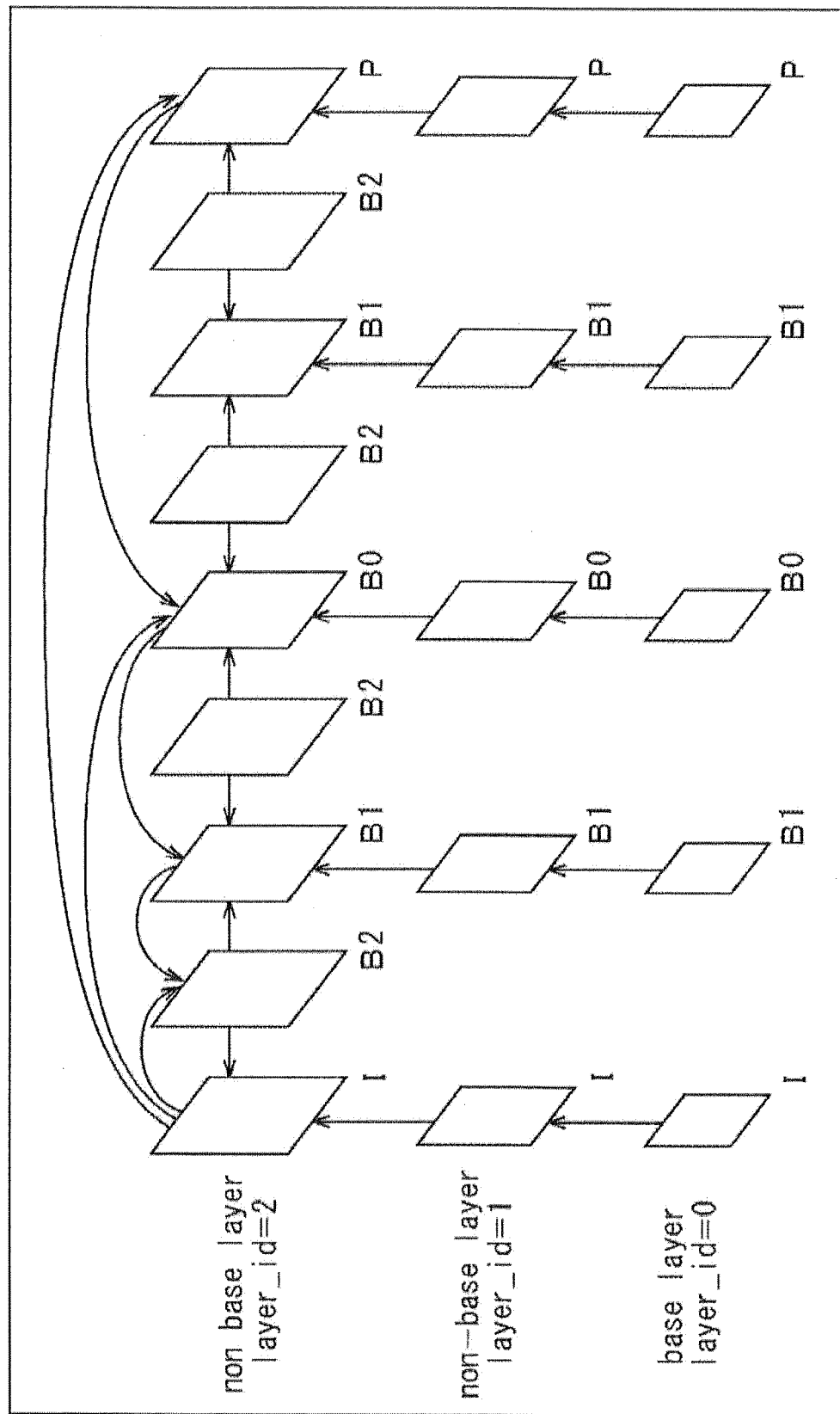


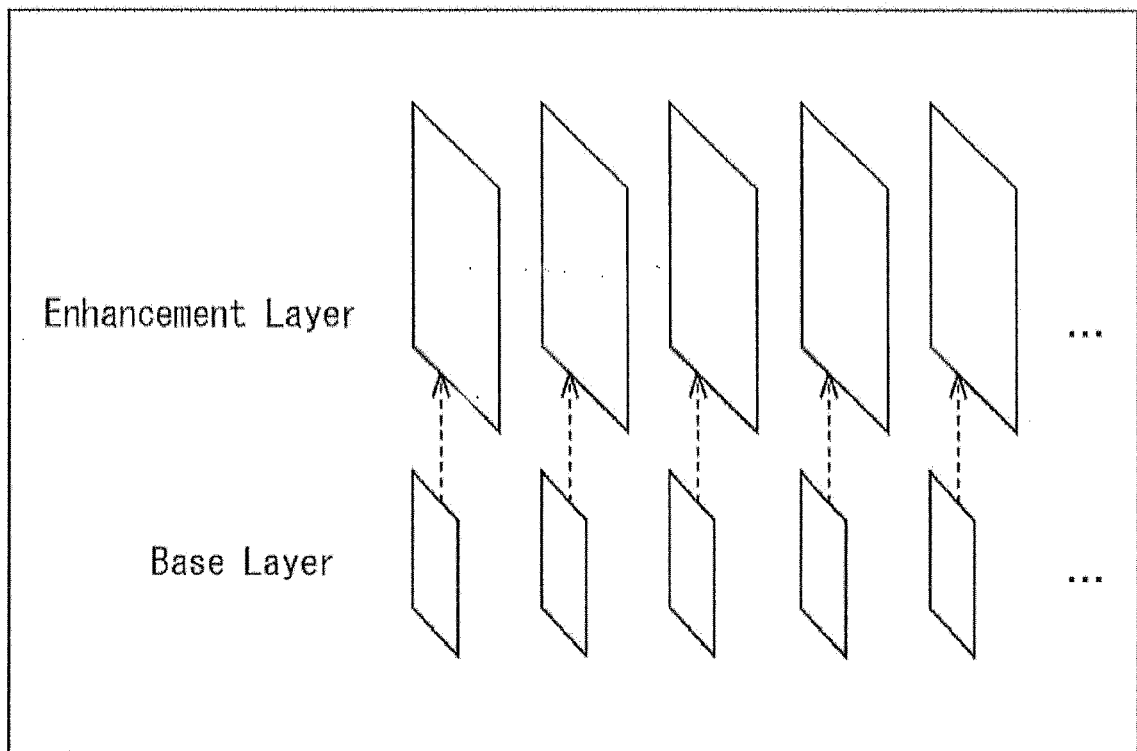
FIG. 3

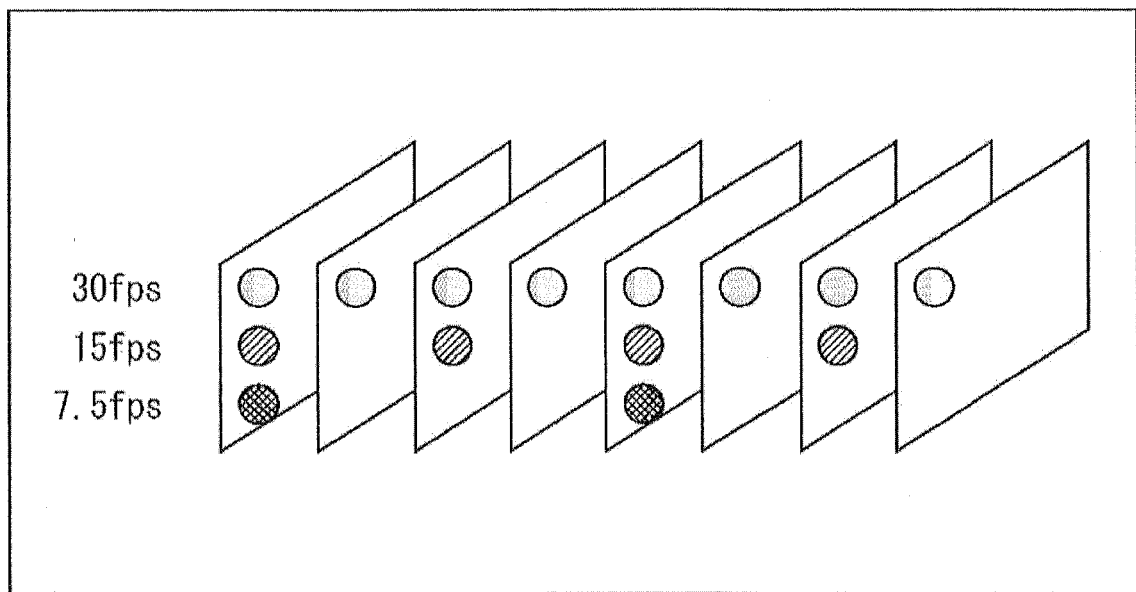
FIG. 4

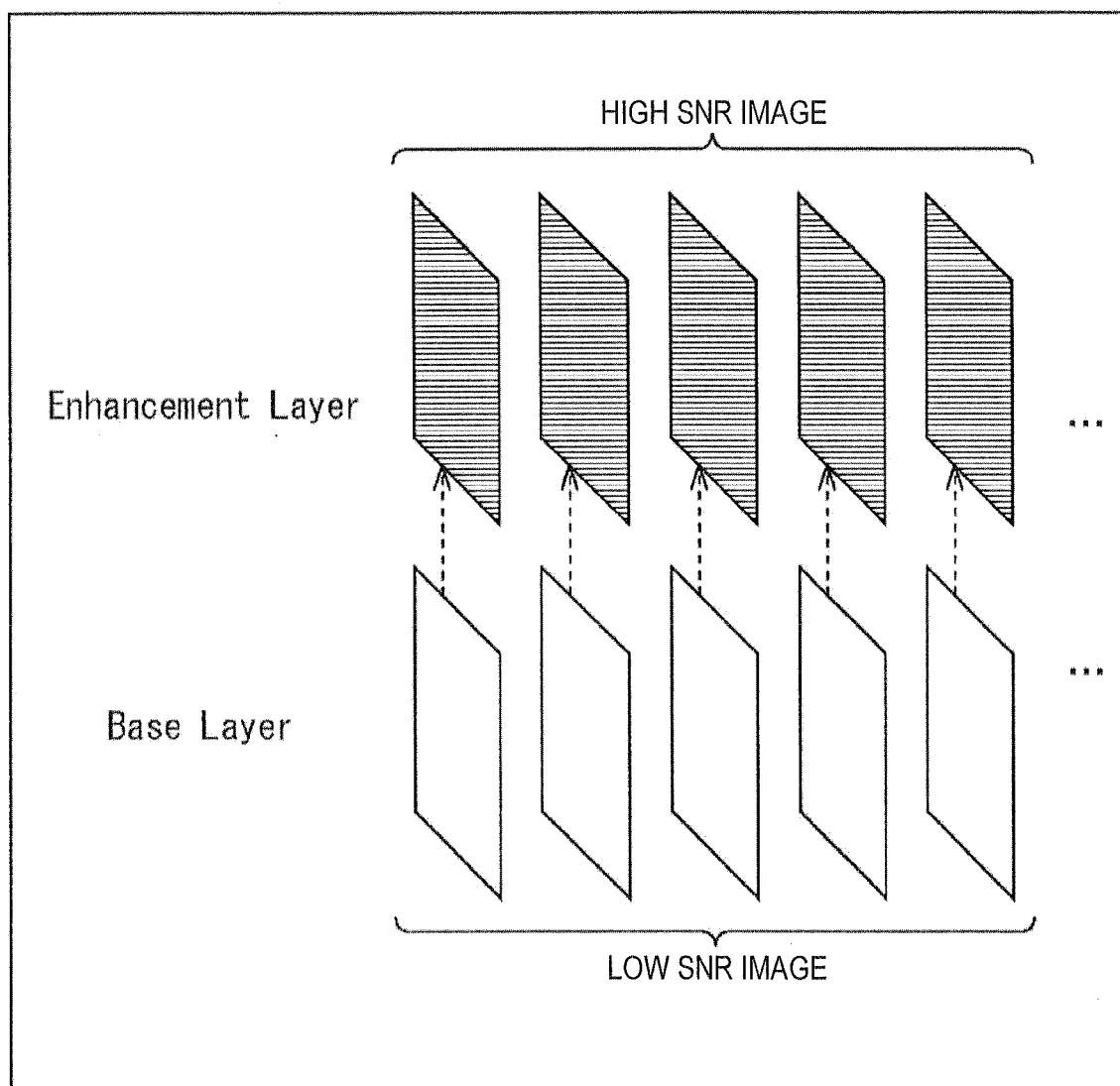
FIG. 5

FIG. 6

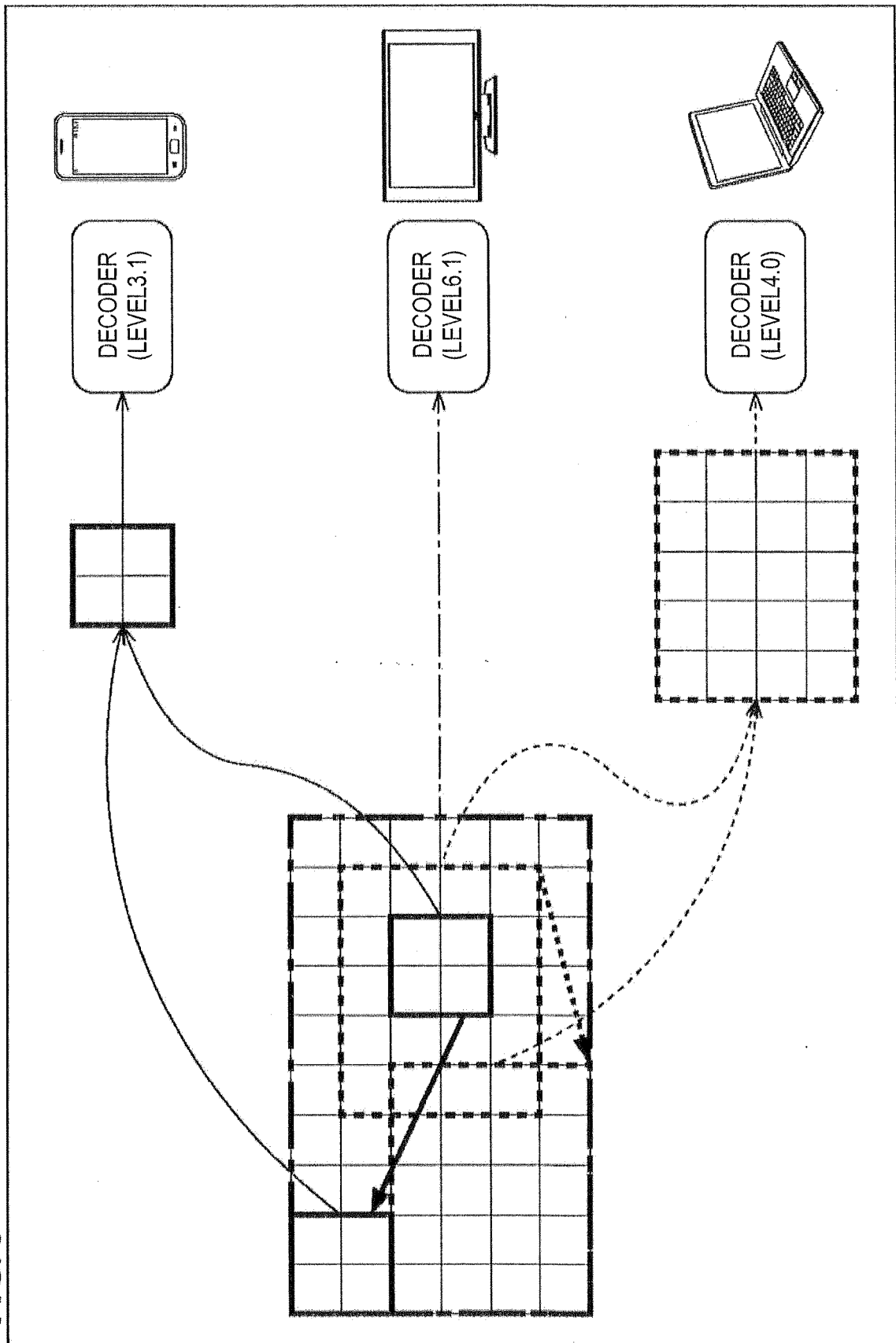


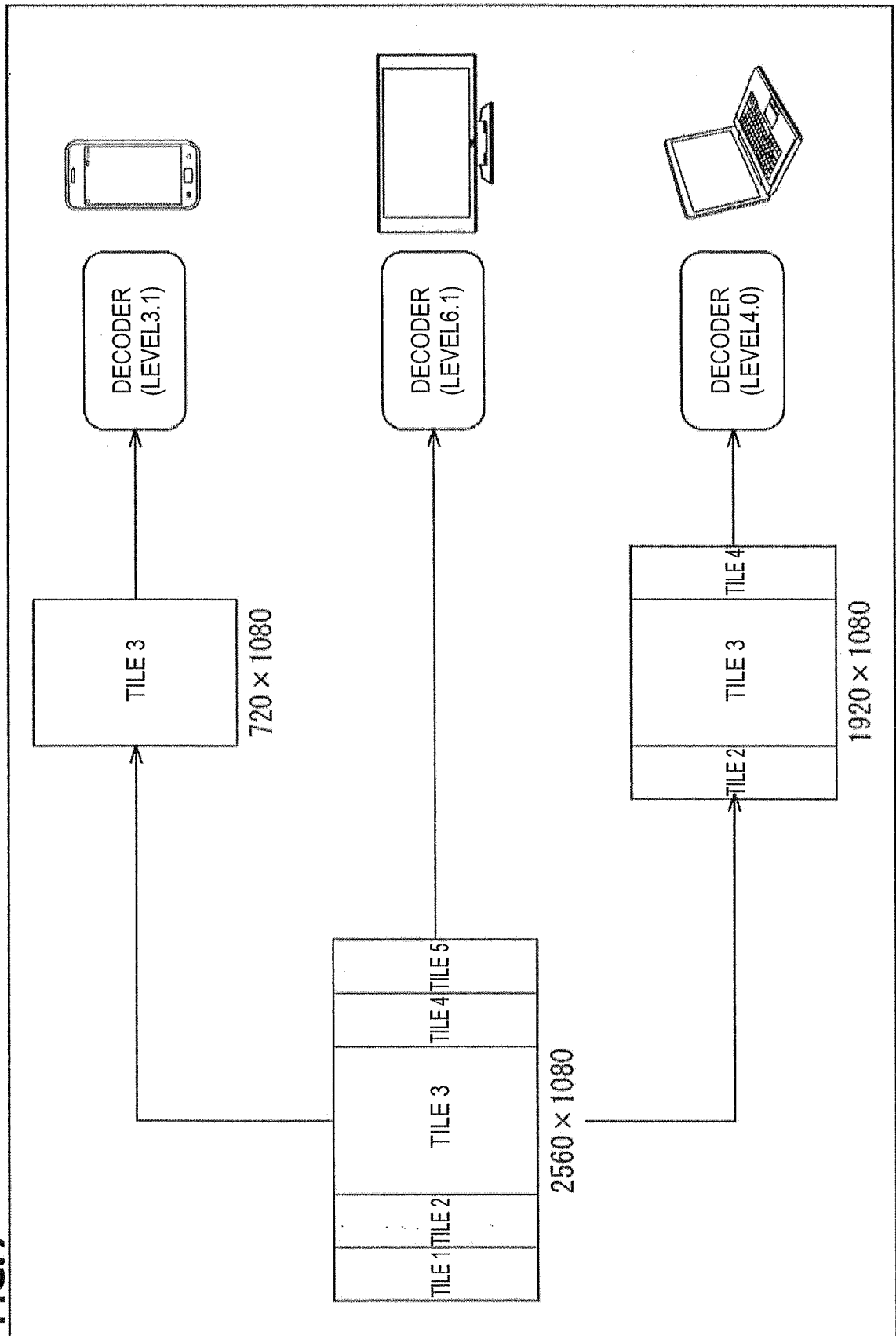
FIG. 7

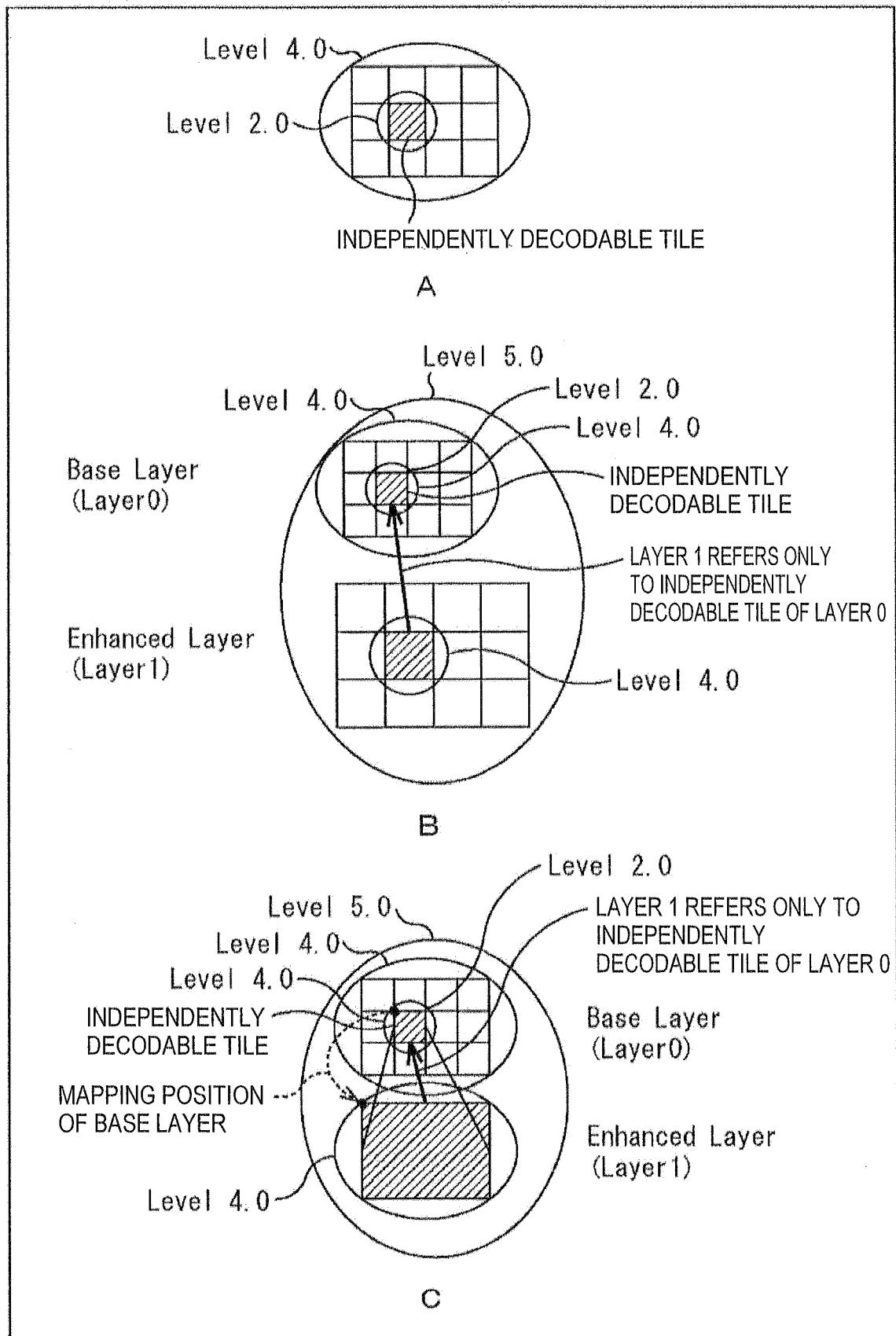
FIG. 8

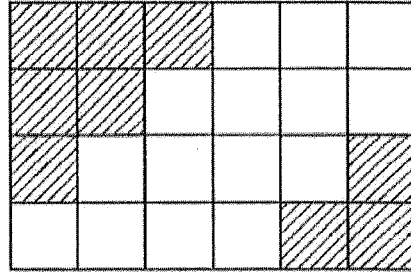
FIG. 9

motion_constrained_tile_sets(payloadSize){	Descriptor
num_sets_in_message_minus1	ue(v)
for(i=0;i<num_sets_in_message_minus1;i++){	
mcts_id[i]	ue(v)
num_tile_rects_in_set_minus1[i]	ue(v)
for(j=0;j<num_tile_rects_in_set_minus1[i];j++){	
top_left_tile_index[i][j]	ue(v)
bottom_right_tile_index[i][j]	ue(v)
}	
exact_sample_value_match_flag[i]	u(1)
mcts_level_idc[i]	ue(v)
}	
}	

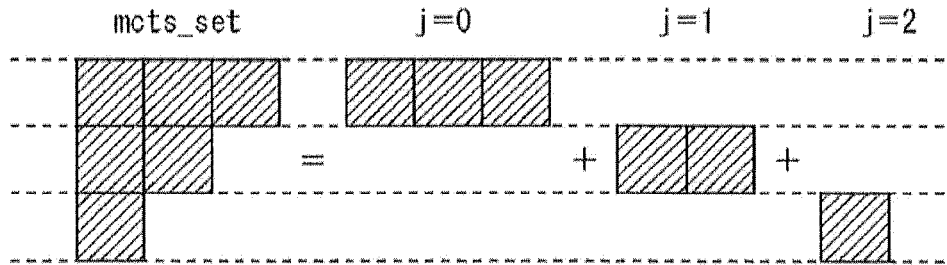
A

The semantics of the syntax element `mcts_level_idc[i]` is the same as the syntax element `general_level_idc` but apply to the *i*-th identified motion-constrained tile

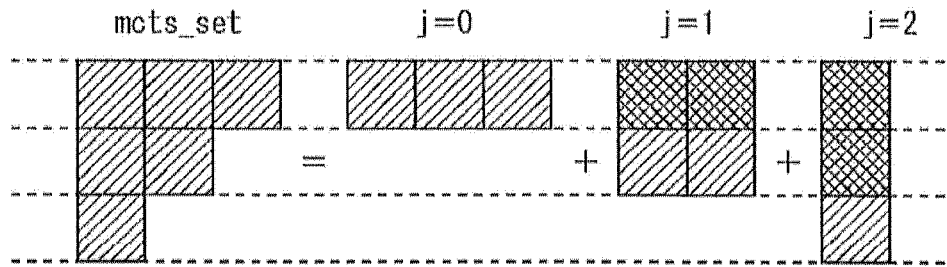
B

FIG. 10

A



B



C

$CtbAddrTsToRs[top_left_tile_index]$
 $CtbAddrTsToRs[bottom_right_tile_index]$
 $S_i = (X_{right, i} - X_{left, i}) * (Y_{top, i} - Y_{btm, i})$
 $S_{sum} = \sum S_i - Overwrap$
 $NumOfPixelToDecode = S_{sum} * CTUSize$

D

FIG. 11

motion_constrained_tile_sets(payloadSize) {	Descriptor
num_sets_in_message_minus1	ue(v)
for (i=0; i<=num_sets_in_message_minus1; i++) {	
mcts_id[i]	ue(v)
num_tile_rects_in_set_minus1[i]	ue(v)
for (j=0; j<=num_tile_rects_in_set_minus1[i]; j++) {	
top_left_tile_index[i][j]	ue(v)
bottom_right_tile_index[i][j]	ue(v)
}	
exact_sample_value_match_flag[i]	u(1)
for (j=0; j<=vps_num_layer_sets_minus1; j++) {	
mcts_level_idc[i][j]	ue(v)
}	
}	

A

The semantics of the syntax element `mcts_level_idc[i][j]` is the same as the syntax element `general_level_idc` but apply to the *i*-th identified motion-constrained tile of the *j*-th layer and lower.

B

FIG. 12

motion_constrained_tile_sets(payloadSize) {	Descriptor
num_sets_in_message_minus1	ue(v)
for (i=0; i<=num_sets_in_message_minus1; i++) {	
mcts_id[i]	ue(v)
num_tile_rects_in_set_minus1[i]	ue(v)
for (j=0; j<=num_tile_rects_in_set_minus1[i]; j++) {	
top_left_tile_index[i][j]	ue(v)
bottom_right_tile_index[i][j]	ue(v)
}	
exact_sample_value_match_flag[i]	u(1)
} // maxLumaPs_in_set [i]	ue(v)
}	
}	

A

maxLumaPs_in_set specifies the maximum luma picture size in the rectangular regions of tiles in the i-th identified motion-constrained tile set. The value of **maxLumaPs_in_set[i]** shall not be equal to 0 and shall be less than or equal to **PicSizeInSamplesY**.

B

FIG. 13

motion_constrained_tile_sets(payloadSize) {	Descriptor
num_sets_in_message_minus1	ue(v)
for (i=0; i<num_sets_in_message_minus1; i++) {	
mcts_id[i]	ue(v)
num_tile_rects_in_set_minus1[i]	ue(v)
for (j=0; j<num_tile_rects_in_set_minus1[i]; j++) {	
top_left_tile_index[i][j]	ue(v)
bottom_right_tile_index[i][j]	ue(v)
}	
exact_sample_value_match_flag[i]	u(1)
mcts_width_in_luma_samples[i]	ue(v)
mcts_height_in_luma_samples[i]	ue(v)
}	
}	

A

mcts_width_in_luma_samples specifies the total width of rectangular regions of tiles in the *i*-th identified motion-constrained tile set. The value of **mcts_width_in_luma_samples[i]** shall not be equal to 0 and shall be less than or equal to **pic_width_in_luma_samples** in active SPS.

mcts_height_in_luma_samples specifies the total height of rectangular regions of tiles in the *i*-th identified motion-constrained tile set. The value of **mcts_height_in_luma_samples[i]** shall not be equal to 0 and shall be less than or equal to **pic_height_in_luma_samples** in active SPS.

B

FIG. 14

	Descriptor
motion_constrained_tile_sets(payloadSize) {	
num_sets_in_message_minus1	ue (v)
for (i=0; i<=num_sets_in_message_minus1; i++) {	
mcts_id[i]	ue (v)
num_tile_rects_in_set_minus1[i]	ue (v)
for (j=0; j<=num_tile_rects_in_set_minus1[i]; j++) {	
top_left_tile_index[i][j]	ue (v)
bottom_right_tile_index[i][j]	ue (v)
}	
exact_sample_value_match_flag[i]	u(1)
if (nal_hrd_parameters_present_flag)	
mcts_hrd_parameters()	
if (vcl_hrd_parameters_present_flag)	
mcts_hrd_parameters()	
}	
}	

FIG.15

mcts_hrd_parameters()		Descriptor
mcts_bit_rate_value_minus1		ue (v)
mcts_cpb_size_value_minus1		ue (v)

A

mcts_bit_rate_value_minus1 (together with bit_rate_scale) specifies the maximum input bitrate for the CPB when the CPB operates the identified motion-constrained tile set. mcts_bit_rate_value_minus1[i] shall be in the range of 0 to $2^{32}-2$, inclusive.

The bit rate in bits per second is given by:

$$\text{mctsBitRate} = (\text{mcts_bit_rate_value_minus1} + 1) * 2^{(6 + \text{bit_rate_scale})}$$

mcts_cpb_size_value_minus1 is used together with cpb_size_scale to specify the CPB size when the CPB operates the identified motion-constrained tile set. mcts_cpb_size_value_minus1 shall be in the range of 0 to $2^{32}-2$, inclusive.

The CPB size in bits is given by:

$$\text{mctsCpbSize}[i] = (\text{mcts_cpb_size_value_minus1}[i] + 1) * 2^{(4 + \text{cpb_size_scale})}$$

B

FIG.16

sps_extension() [	Descriptor
inter_view_mv_vert_constraint_flag	u(1)
scaled_ref_layer_offset_present_flag	
if(scaled_ref_layer_offset_present_flag) {	
scaled_ref_layer_left_offset	se(v)
scaled_ref_layer_top_offset	se(v)
scaled_ref_layer_right_offset	se(v)
scaled_ref_layer_bottom_offset	se(v)
}	
...snip...	
mcts_present_flag	u(1)
...snip...	
}	

A

mcts_present_flag specifies whether the motion constrained tile set SEI is present.

B

FIG. 17

sps_extension() {	Descriptor
...snip...	
mcts_level_present_flag	u(1)
if (mcts_level_present_flag) {	
num_sets_in_message_minus1	ue(v)
for (i=0; i<=num_sets_in_message_minus1; i++) {	
mcts_level_idc[i]	ue(v)
}	
}	
}	

A

`mets_level_present_flag` equal to 1 specifies that level information is present for the representation of motion constrained tile sets.

`num_sets_in_message_minus1` plus 1 specifies the number of motion-constrained tile sets identified in the SEI message. The value of `num_sets_in_message_minus1` shall be in the range of 0 to 255, inclusive.

The semantics of the syntax element `mcts_level_idc[i]` is the same as the syntax element `general_level_idc` but apply to the i -th identified motion-constrained tile

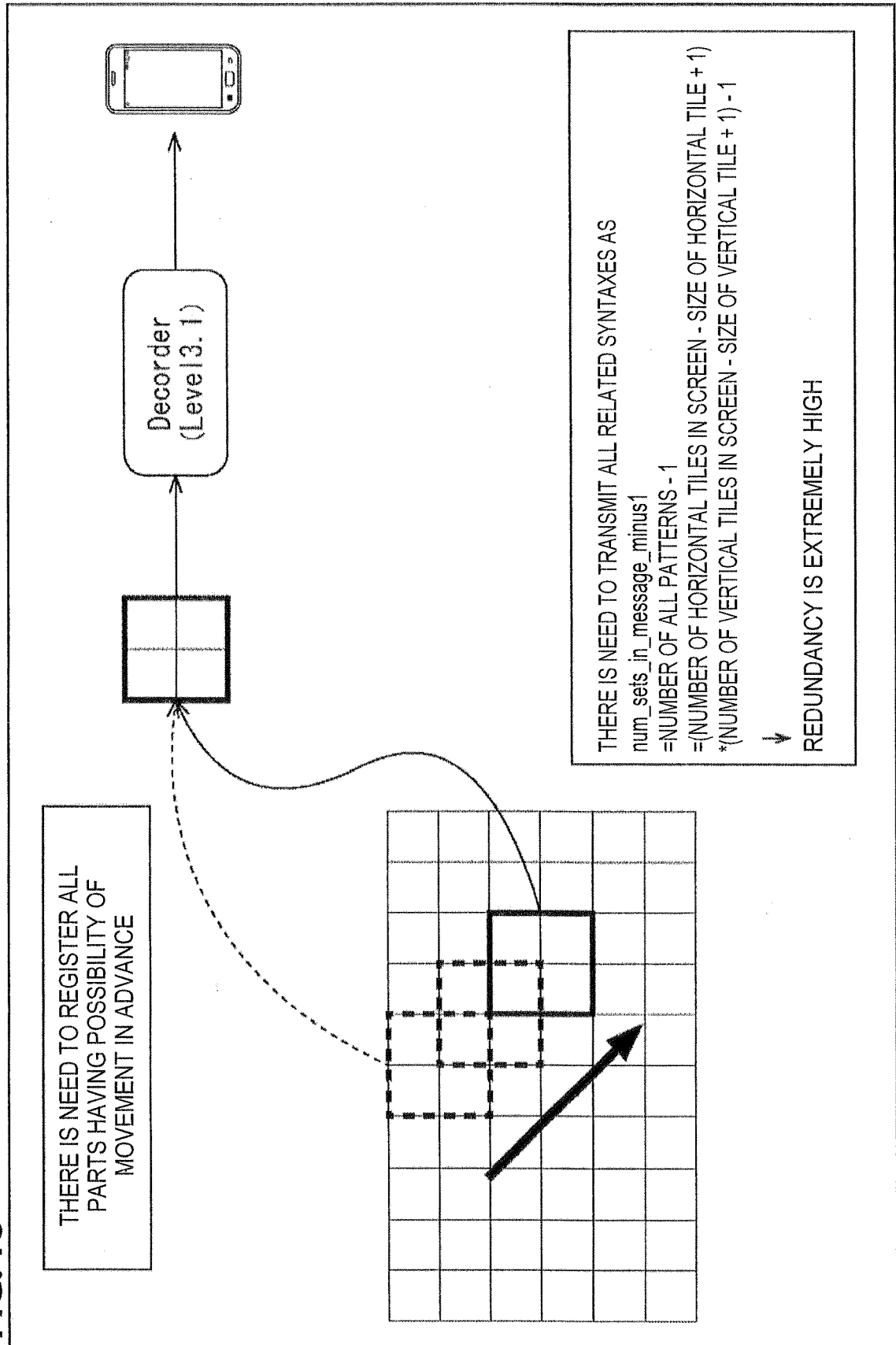
FIG. 18

FIG. 20

motion_constrained_tile_sets(payloadSize) {	Descriptor
mcts_mode	ue (v)
num_sets_in_message_minus1	ue (v)
if (mcts_mode == 0) { // mode for current ROI application	
for (i=0; i<=num_sets_in_message_minus1; i++) {	
mcts_id[i]	ue (v)
num_tile_rects_in_set_minus1[i]	ue (v)
for (j=0; j<=num_tile_rects_in_set_minus1[i]; j++) {	
top_left_tile_index[i][j]	ue (v)
bottom_right_tile_index[i][j]	ue (v)
}	
exact_sample_value_match_flag[i]	u (1)
mcts_level_idc[i]	ue (v)
}	
} else if (mcts_mode == 1) { // mode for DASH application	
for (i=0; i<=num_sets_in_message_minus1; i++) {	
mcts_id[i]	ue (v)
num_of_tiles_minus1[i]	ue (v)
mcts_level_idc[i]	ue (v)
}	
}	
}	

FIG. 21

temporal_motion_constrained_tile_sets(payloadSize) {	Descriptor
mc_all_tiles_exact_sample_value_match_flag	u(1)
each_tile_one_tile_set_flag	u(1)
mcts_level_idc_present_flag	u(1)
if(!each_tile_one_tile_set_flag) {	
num_sets_in_message_minus1	ue(v)
for (i=0; i<=num_sets_in_message_minus1; i++) {	
mcts_id[i]	ue(v)
num_tile_rects_in_set_minus1[i]	ue(v)
for (j=0; j<=num_tile_rects_in_set_minus1[i]; j++) {	
top_left_tile_index[i][j]	ue(v)
bottom_right_tile_index[i][j]	ue(v)
}	
}	
if(!mc_all_tiles_exact_sample_value_match_flag)	
exact_sample_value_match_flag[i]	u(1)
}	
if(mcts_level_idc_present_flag)	
mcts_level_idc	u(8)
else	
if(mcts_level_idc_present_flag) {	
num_mcts_level_idc_minus1	ue(v)
for (i=0; i<=num_mcts_level_idc_minus1; i++) {	
mcts_level_id[i]	ue(v)
num_mc_tile_columns_minus1[i]	ue(v)
num_mc_tile_rows_minus1[i]	ue(v)
mcts_level_idc[i]	u(8)
}	
}	
}	

A

mcts_level_idc is same as the syntax element **general_level_idc**, but applied to the representation of motion-constrained tile sets. **mcts_level_idc** indicates a level to which the motion-constrained tile sets conforms as specified in Annex A with the following additions.

num_mc_tile_columns_minus1[i] plus 1 specifies the number of tile columns partitioning the *i*-th identified motion-constrained tile sets. The value of **num_mc_tile_columns_minus1** shall be in the range of 0 to **num_tile_columns_minus1**, inclusive.

num_mc_tile_rows_minus1[i] plus 1 specifies the number of tile rows partitioning the *i*-th identified motion-constrained tile sets. The value of **num_tile_rows_minus1** shall be in the range of 0 to **num_tile_rows_minus1**, inclusive.

B

FIG. 22

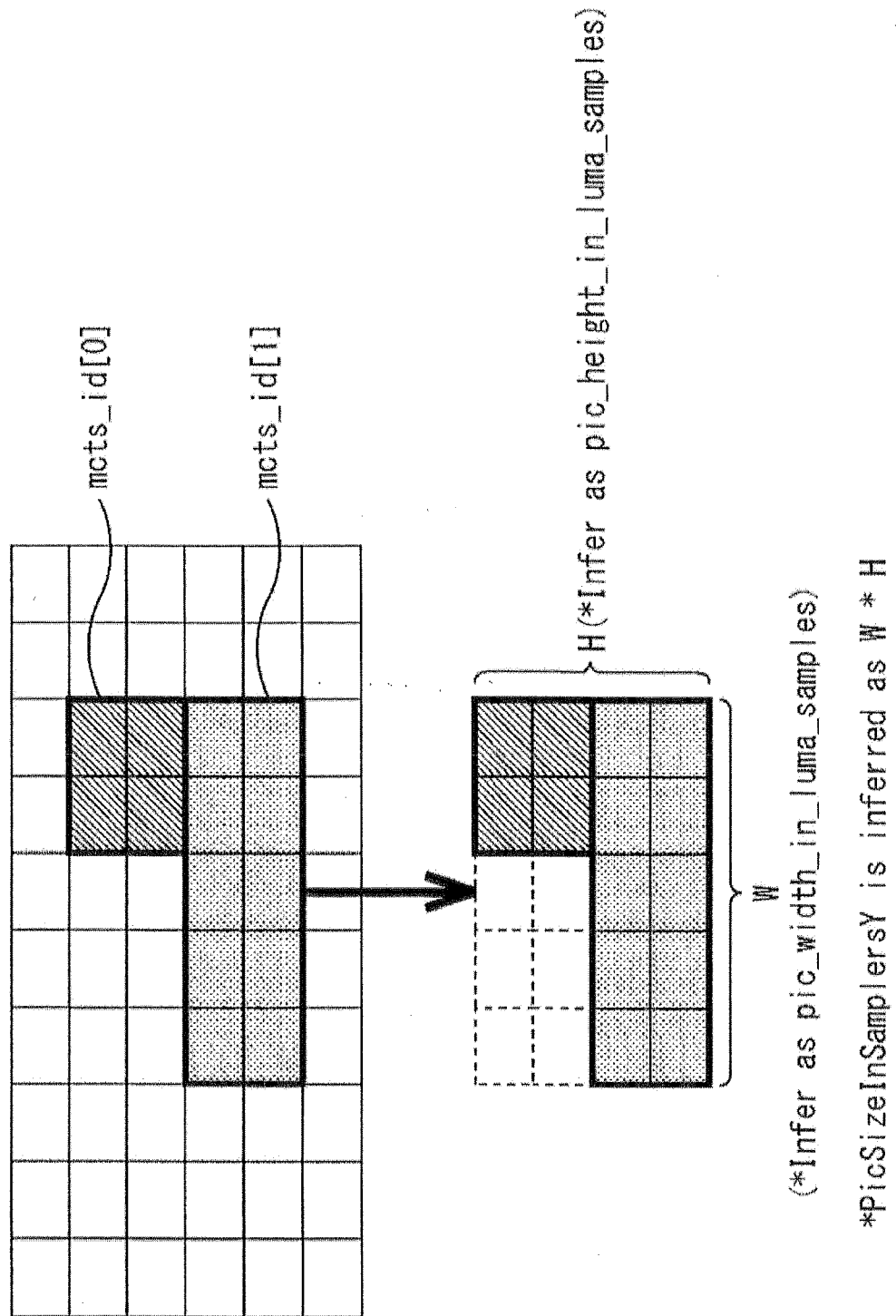


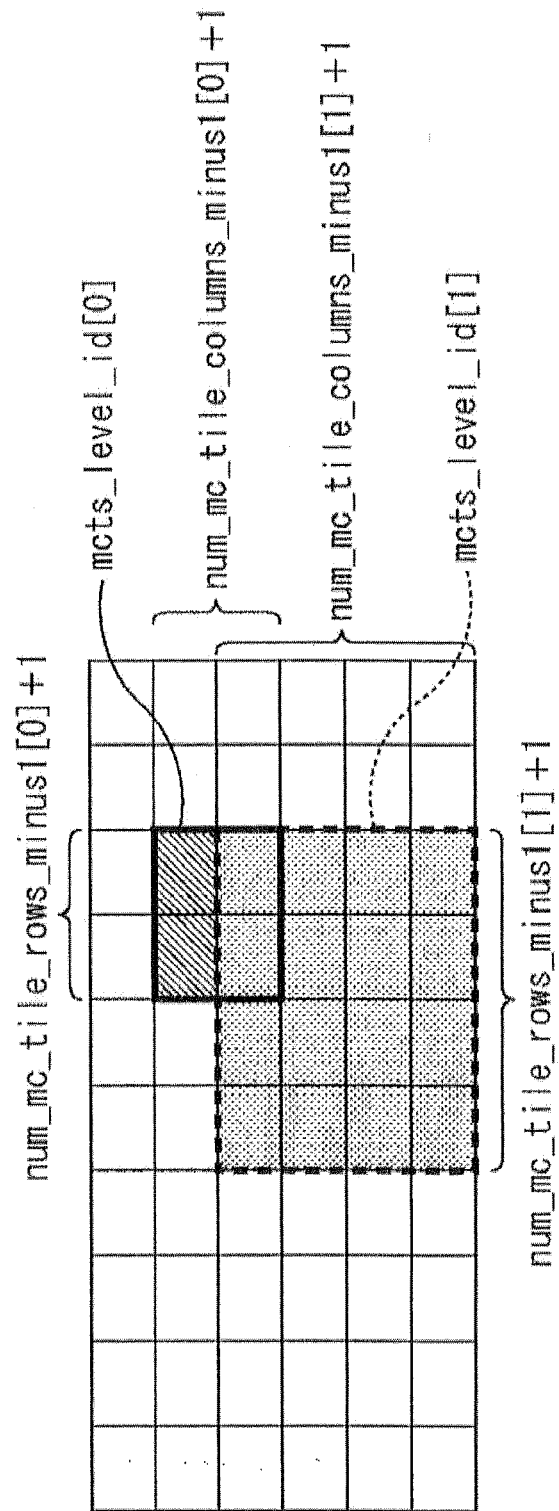
FIG. 23

FIG. 24

temporal_motion_constrained_tile_sets(payloadSize) {	Descriptor
mc_all_tiles_exact_sample_value_match_flag	u(1)
each_tile_one_tile_set_flag	u(1)
mcts_level_idc_present_flag	u(1)
if(!each_tile_one_tile_set_flag) {	
num_sets_in_message_minus1	ue(v)
for(i=0; i<=num_sets_in_message_minus1; i++) {	
mcts_id[i]	ue(v)
num_tile_rects_in_set_minus1[i]	ue(v)
for(j=0; j<=num_tile_rects_in_set_minus1[i]; j++) {	
top_left_tile_index[i][j]	ue(v)
bottom_right_tile_index[i][j]	ue(v)
}	
if(!mc_all_tiles_exact_sample_value_match_flag)	
exact_sample_value_match_flag[i]	u(1)
if(mcts_level_idc_present_flag)	
mcts_level_idc[i]	u(8)
}	
else {	
if(mcts_level_idc_present_flag)	
max_mcts_level_idc	ue(v)
}	

A

mcts_level_idc[i] is same as the syntax element **general_level_idc**, but applied to the representation of the *i*-th identified motion-constrained tile sets. The value of **mcts_level_idc** indicates a level to which the motion-constrained tile sets conforms as specified in Annex A with the following additions.

- The variables **PicSizeInSamplesY** should be inferred as total number of luma samples in motion-constrained tile sets.
- The value of **pic_width_in_luma_samples** and **pic_height_in_luma_samples** should be inferred as width and height of overall decoded region of motion-constrained tile sets in units of luma samples, respectively. In case the region of motion-constrained tile sets is not in rectangular shape (e.g. L/U shaped, apart from each other etc), the width and height should be inferred as value of a rectangle which covers the entire tile sets horizontally and vertically.
- The value of **num_tile_columns_minus1** plus 1 and **num_tile_rows_minus1** plus 1 should be inferred as total number of tile columns and rows in motion-constrained tile sets which the defined **mcts_level_idc**
- For the VCL HRD parameters and NAL HRD parameters, the values should be inferred as values when the HRD parameters are not present.

max_mcts_level_idc is same as the syntax element **mcts_level_idc[i]** for *i*-th identified motion-constrained tile set but applied to the representation of motion-constrained tile sets which is composed of one tile per one tile set. The value of **max_mcts_level_idc** should be the maximum **level_idc** value of tiles in the picture.

B

FIG. 25

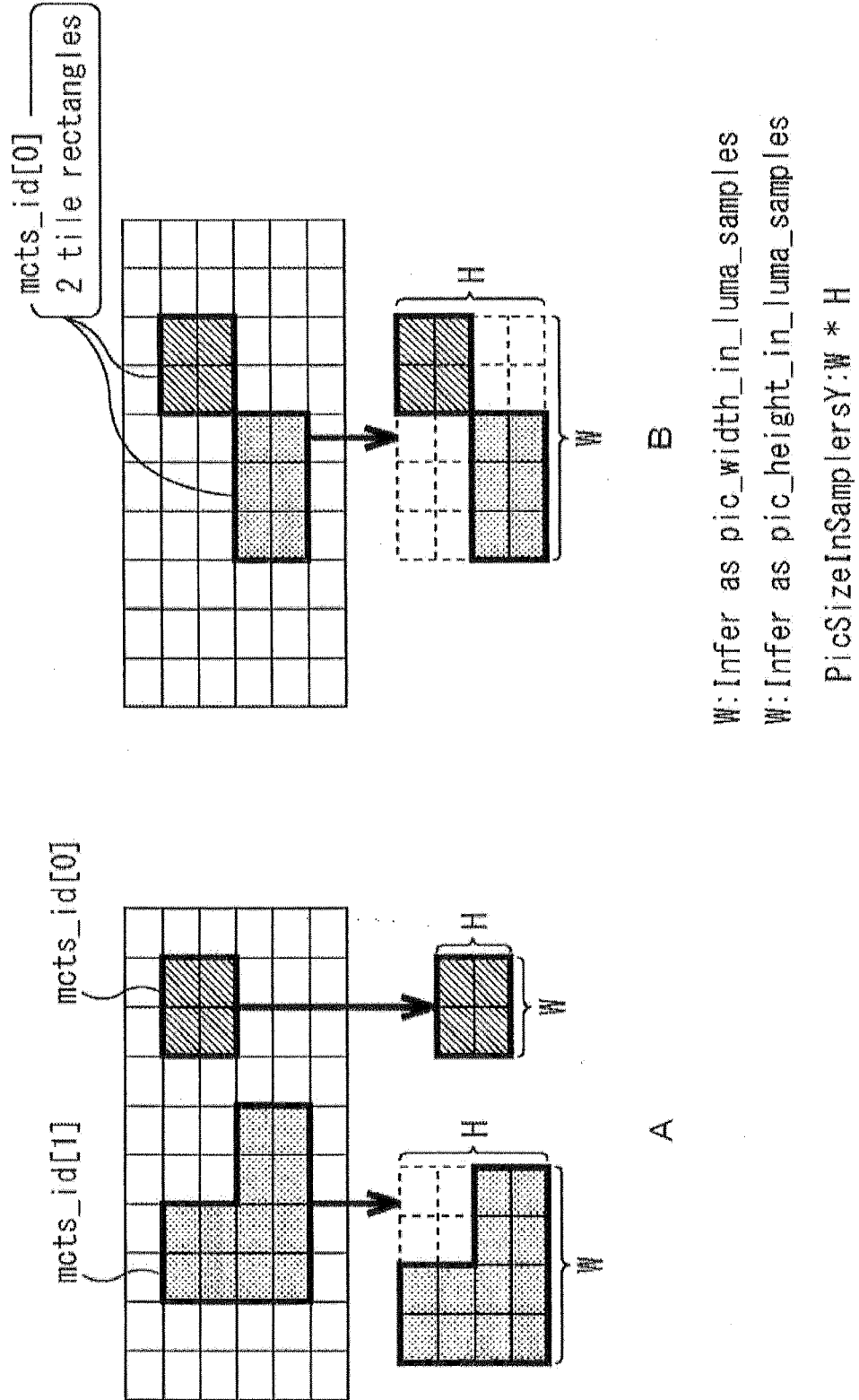


FIG. 27

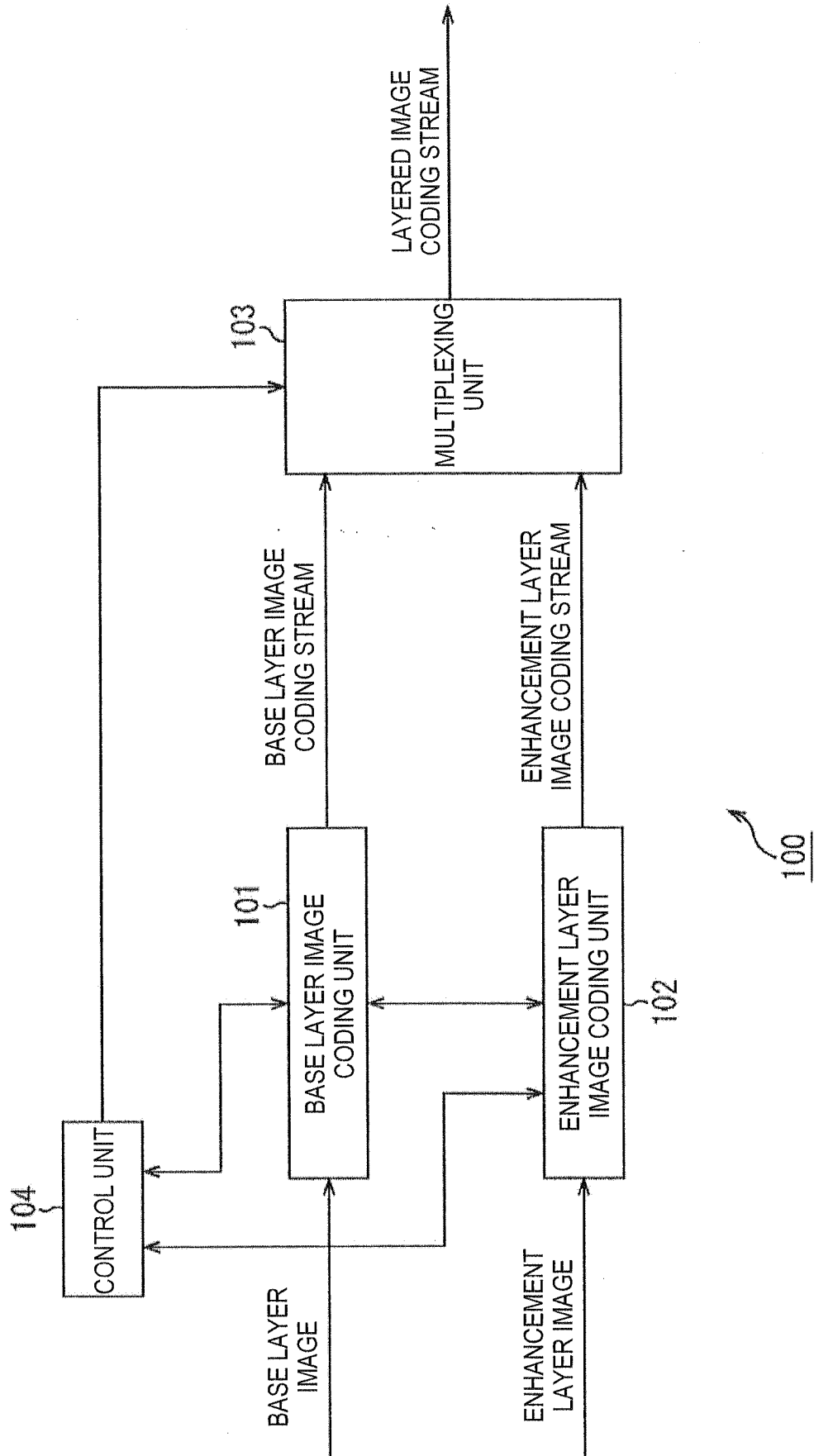


FIG. 28

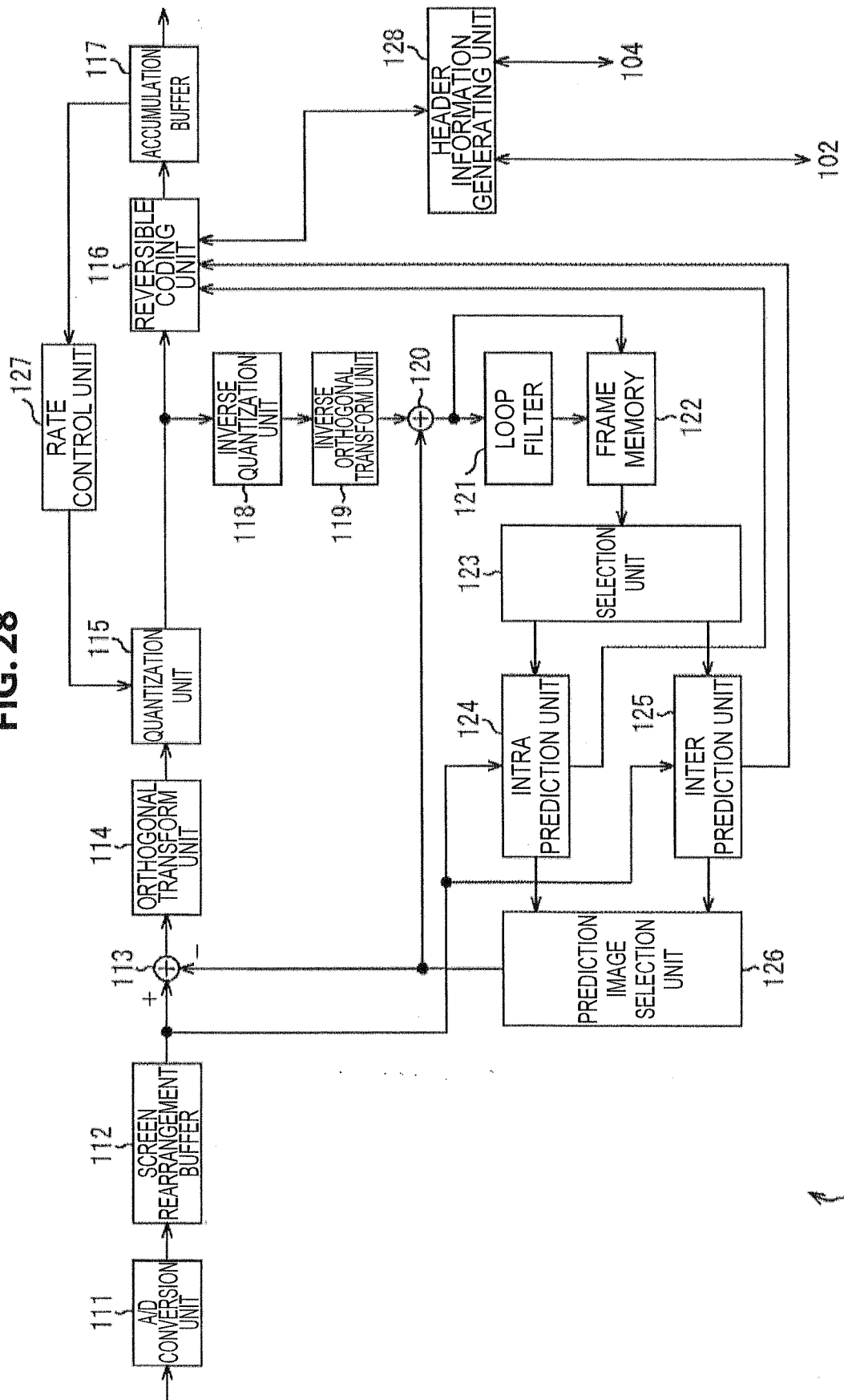


FIG. 29

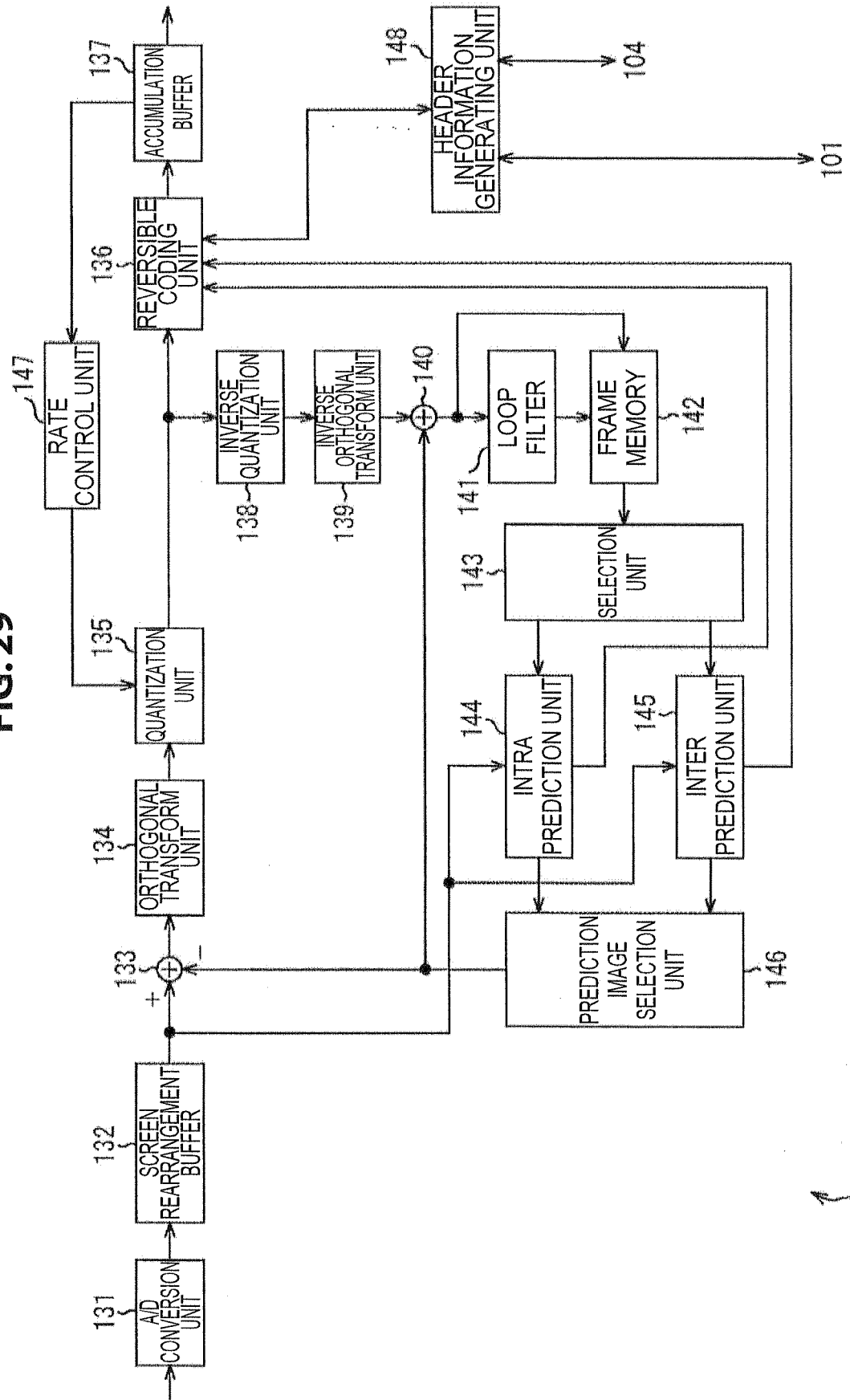
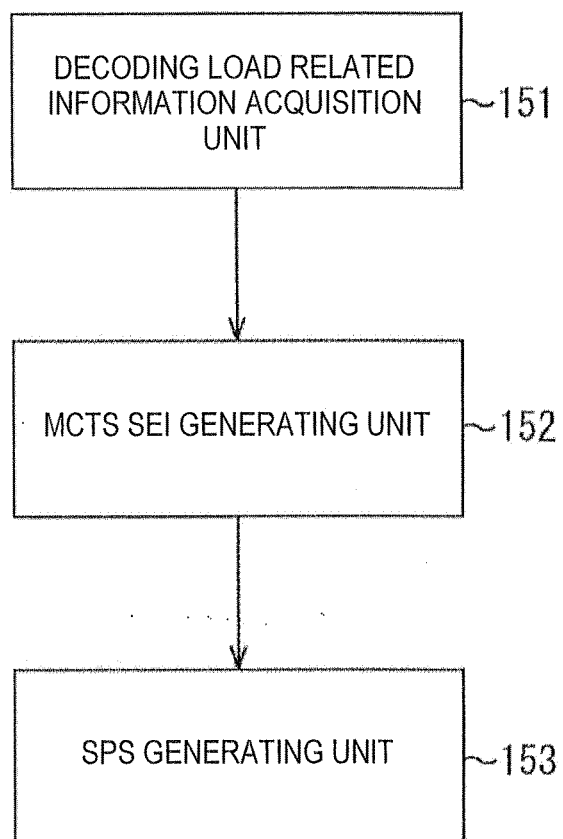


FIG. 30

128

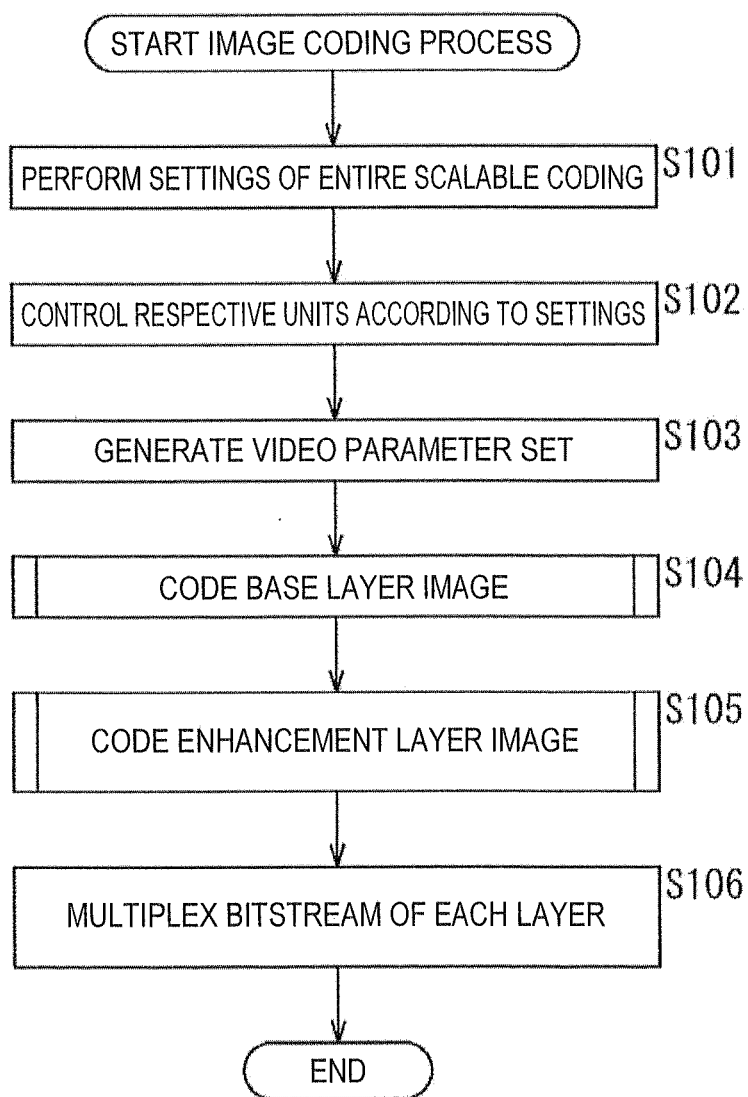
FIG. 31

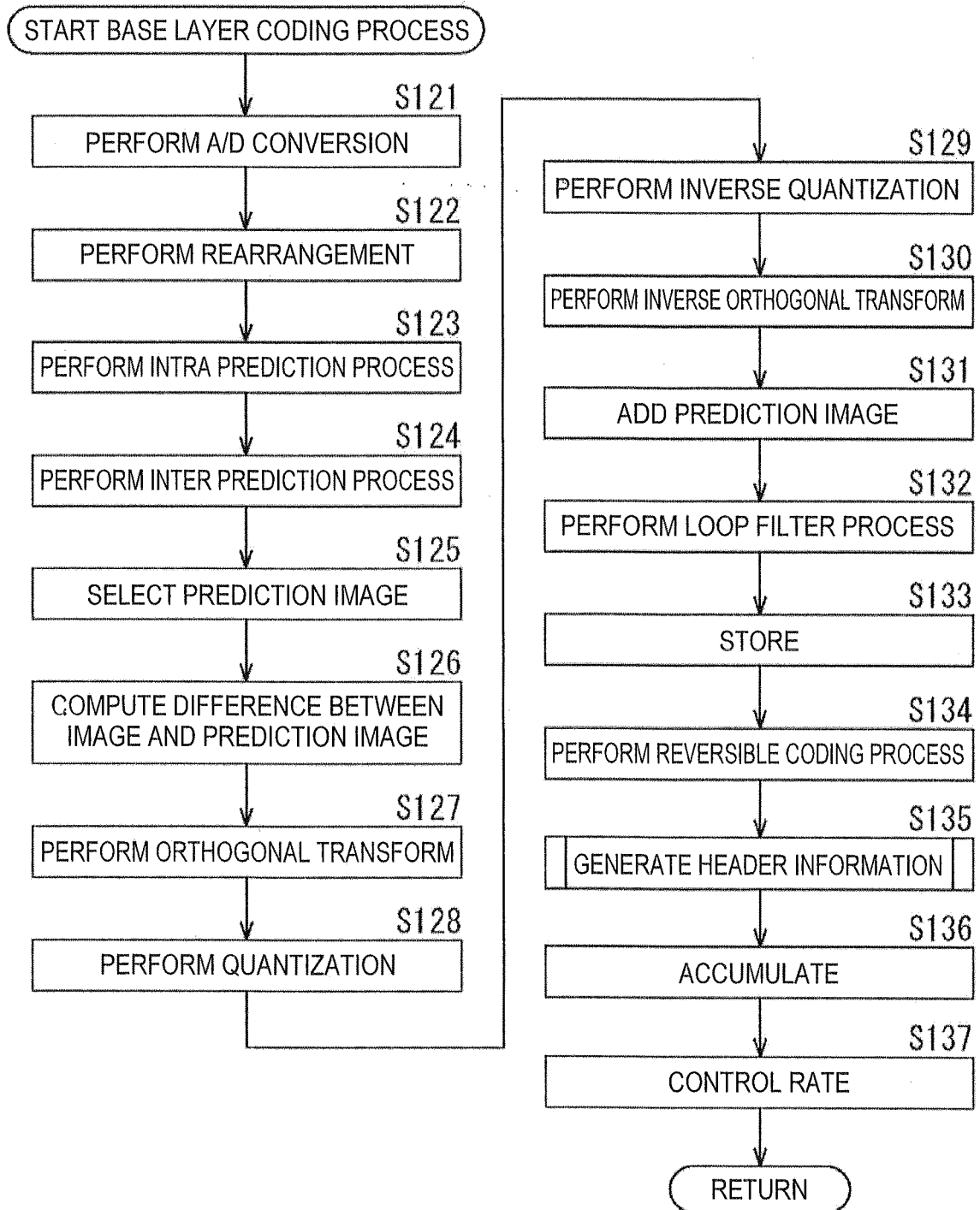
FIG. 32

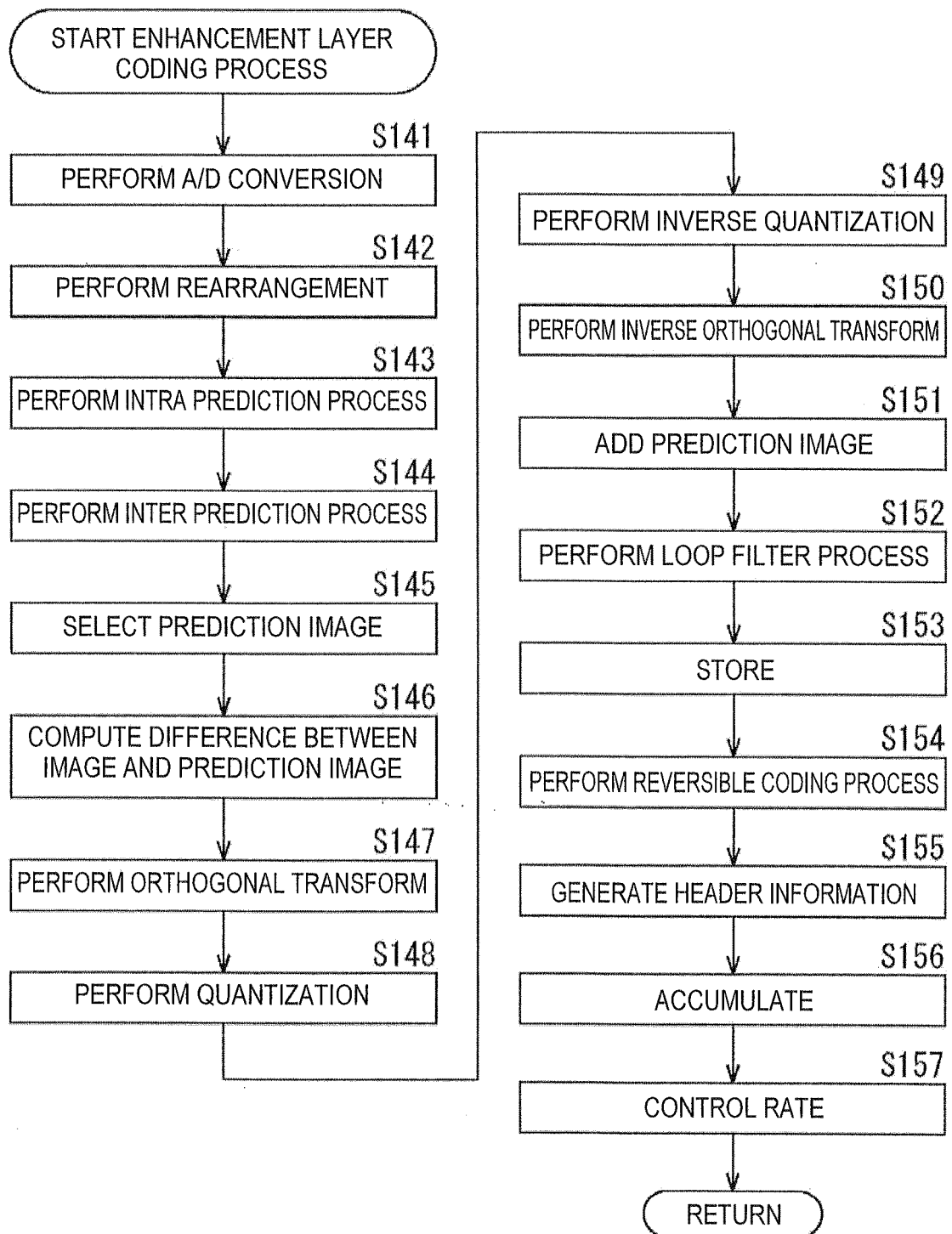
FIG. 33

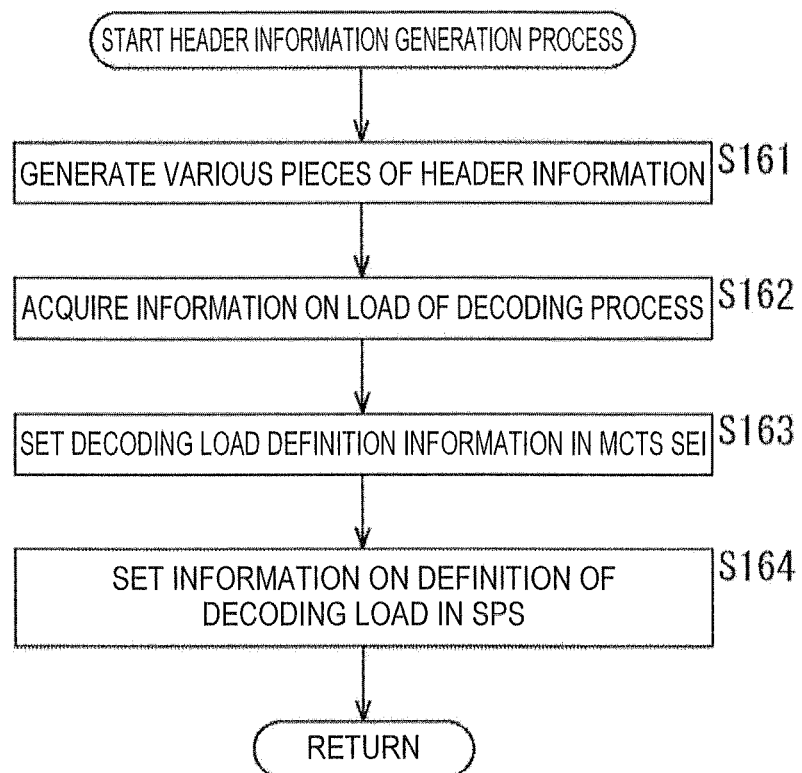
FIG. 34

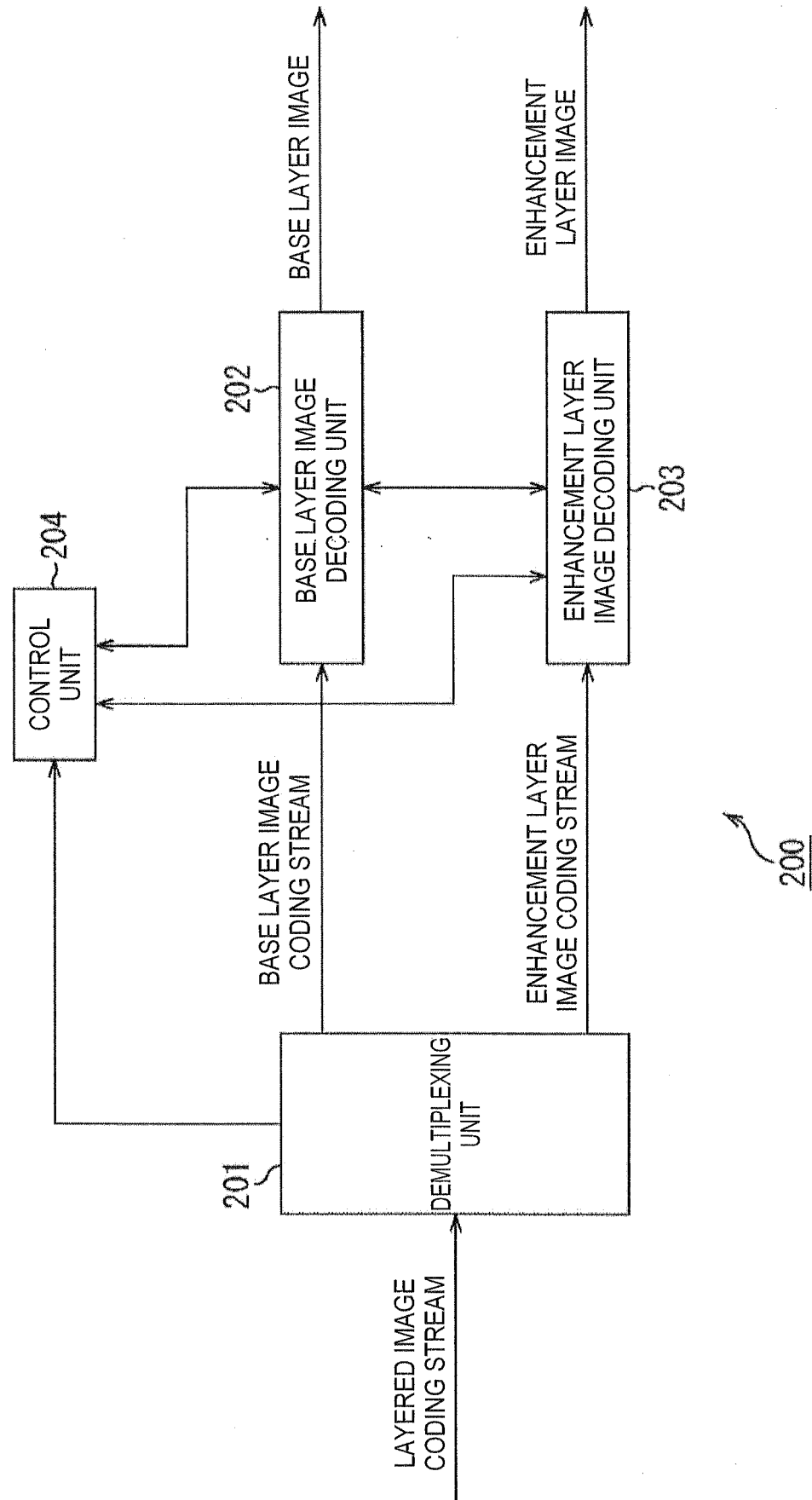
FIG. 35

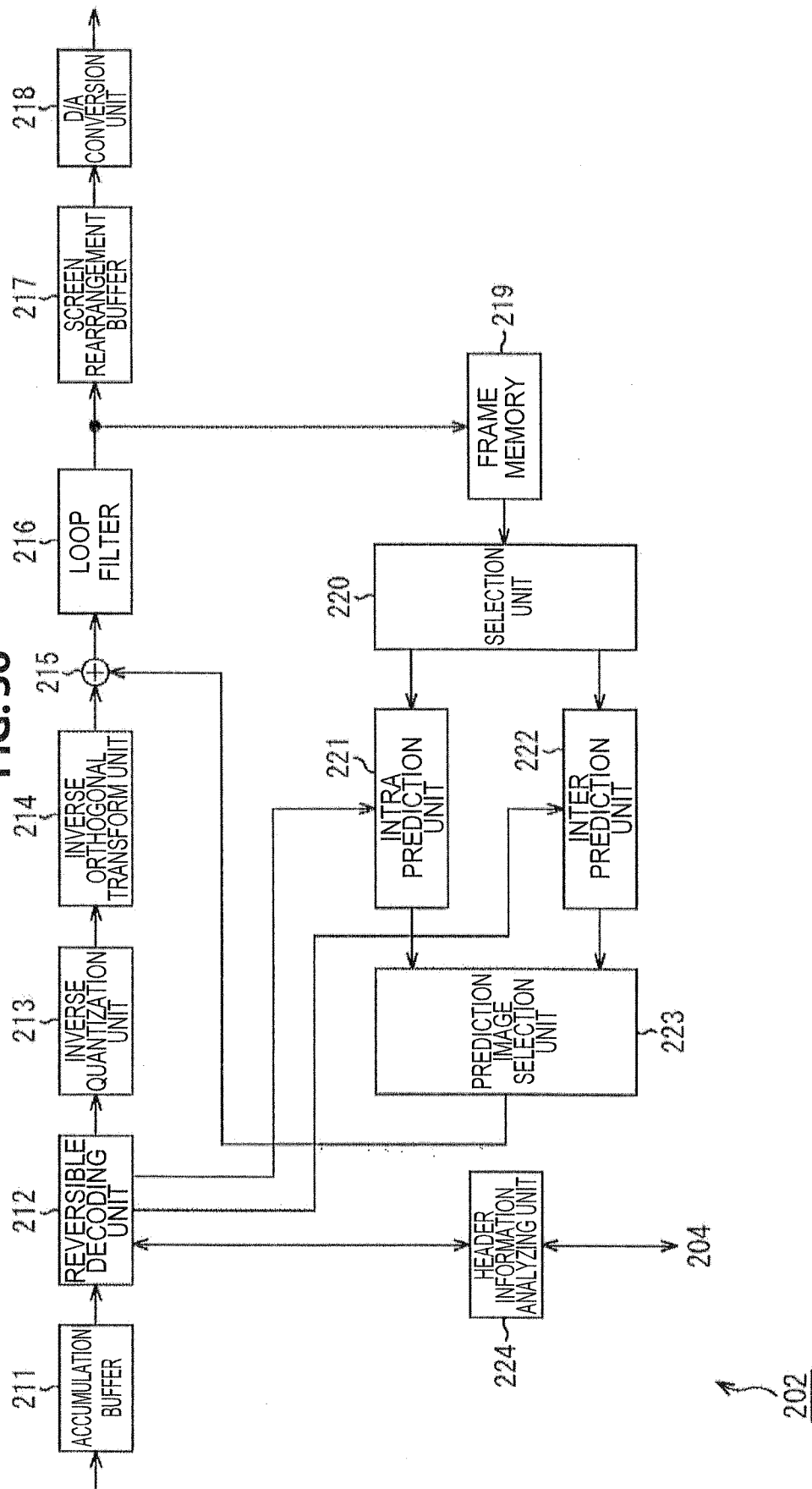
FIG. 36

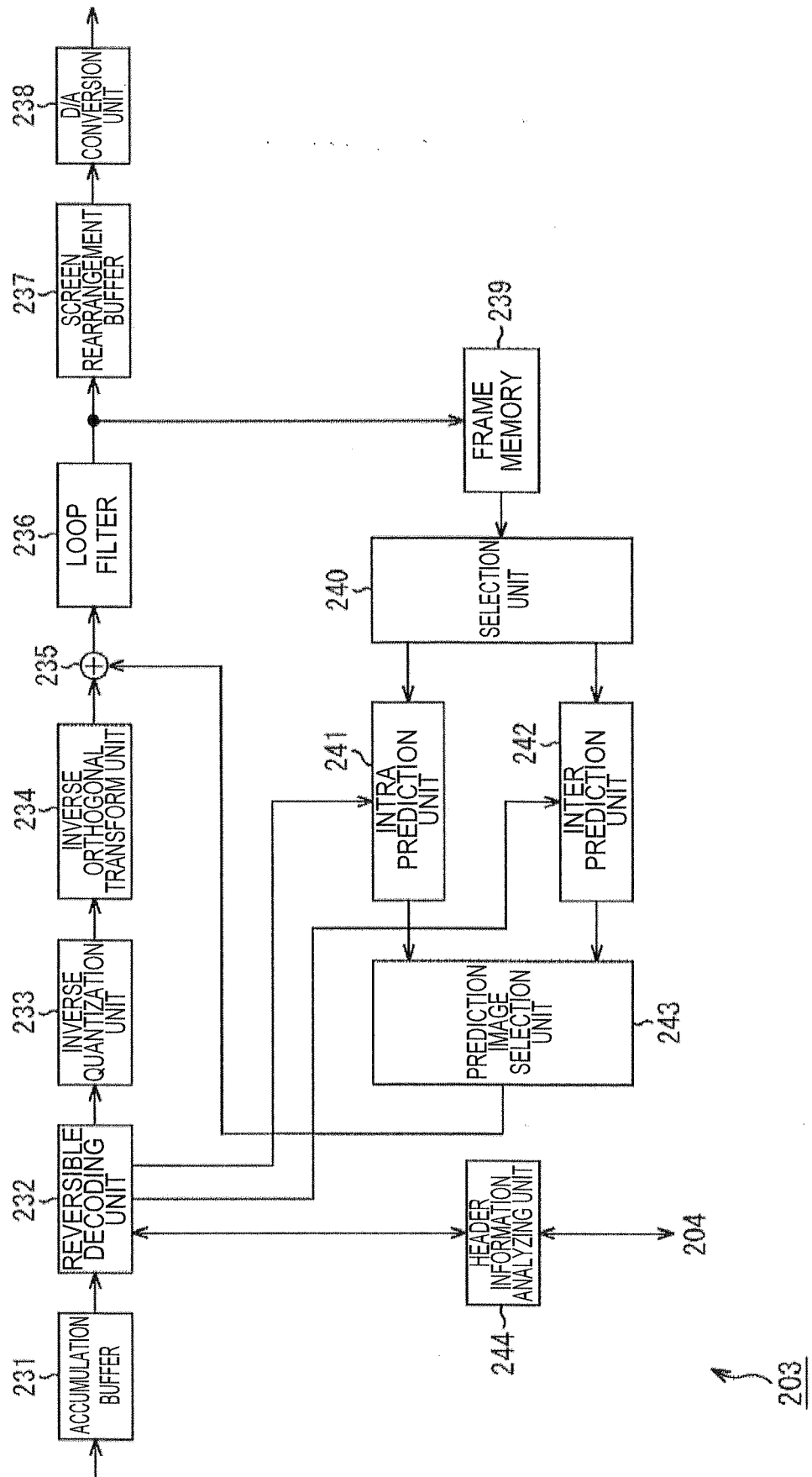
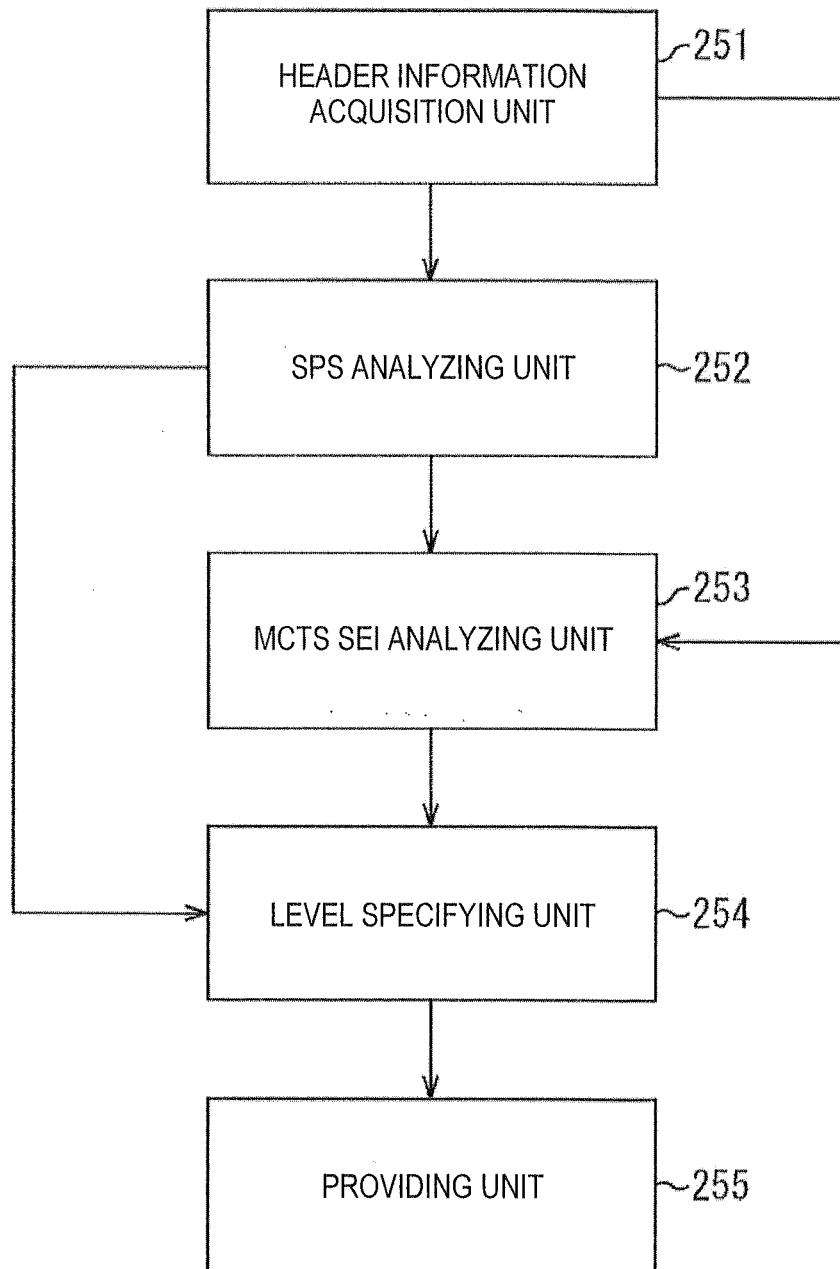
FIG. 37

FIG. 38

224

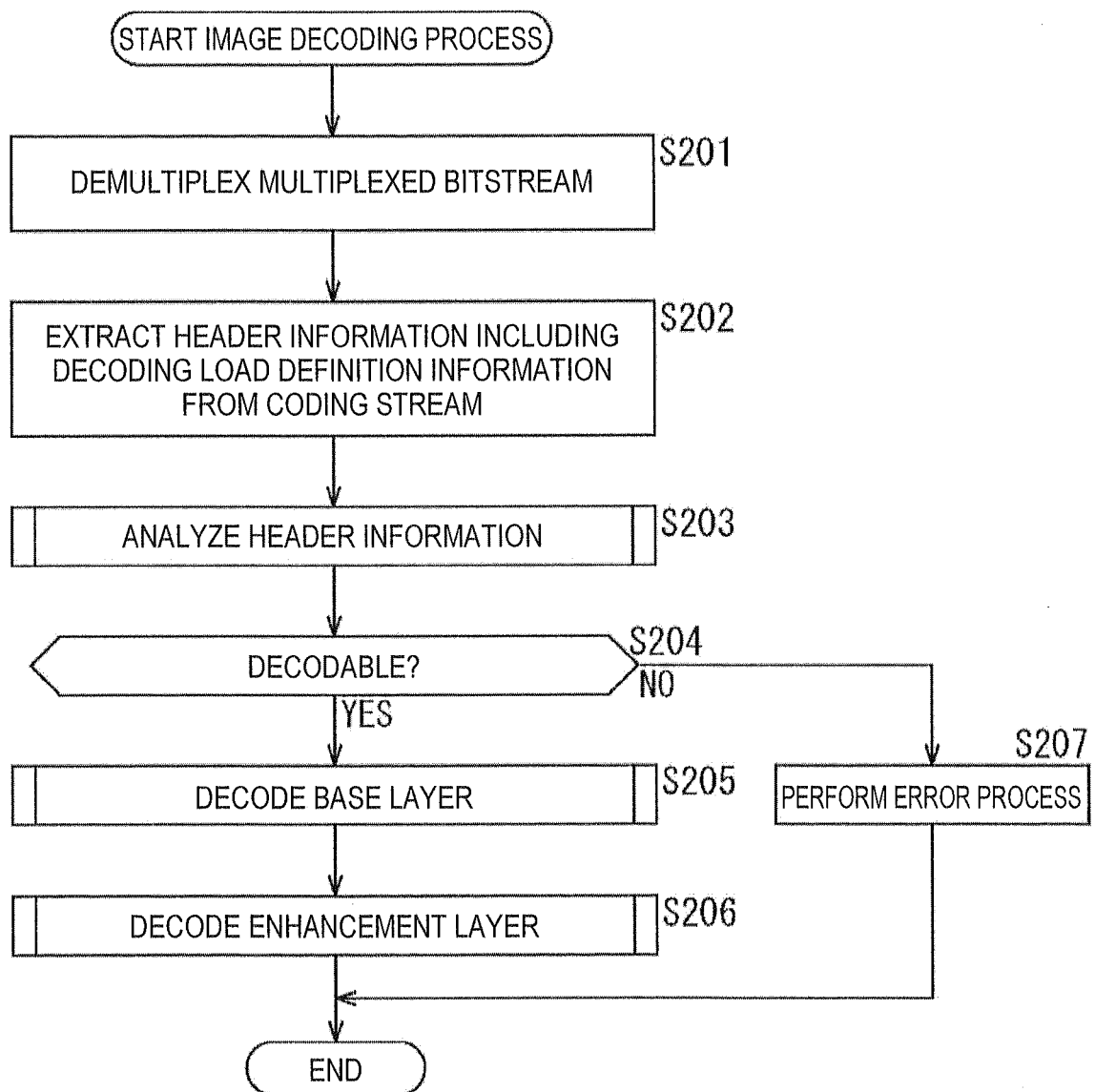
FIG. 39

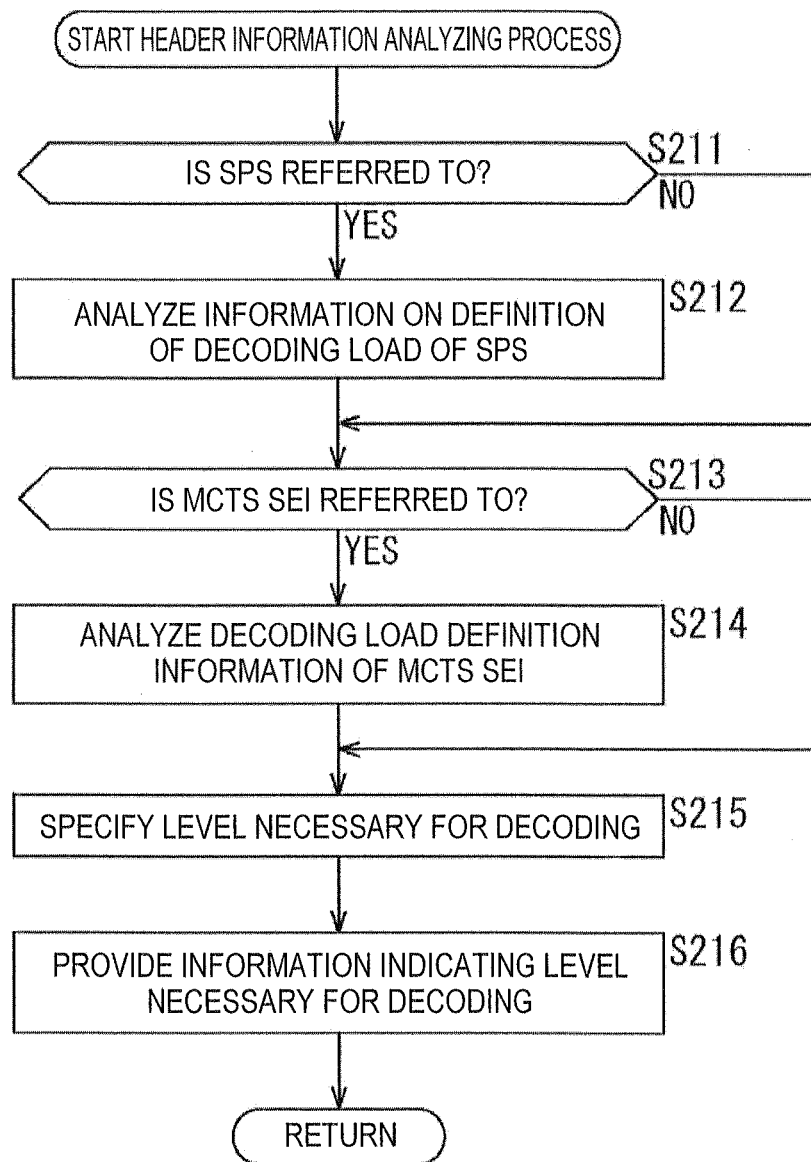
FIG. 40

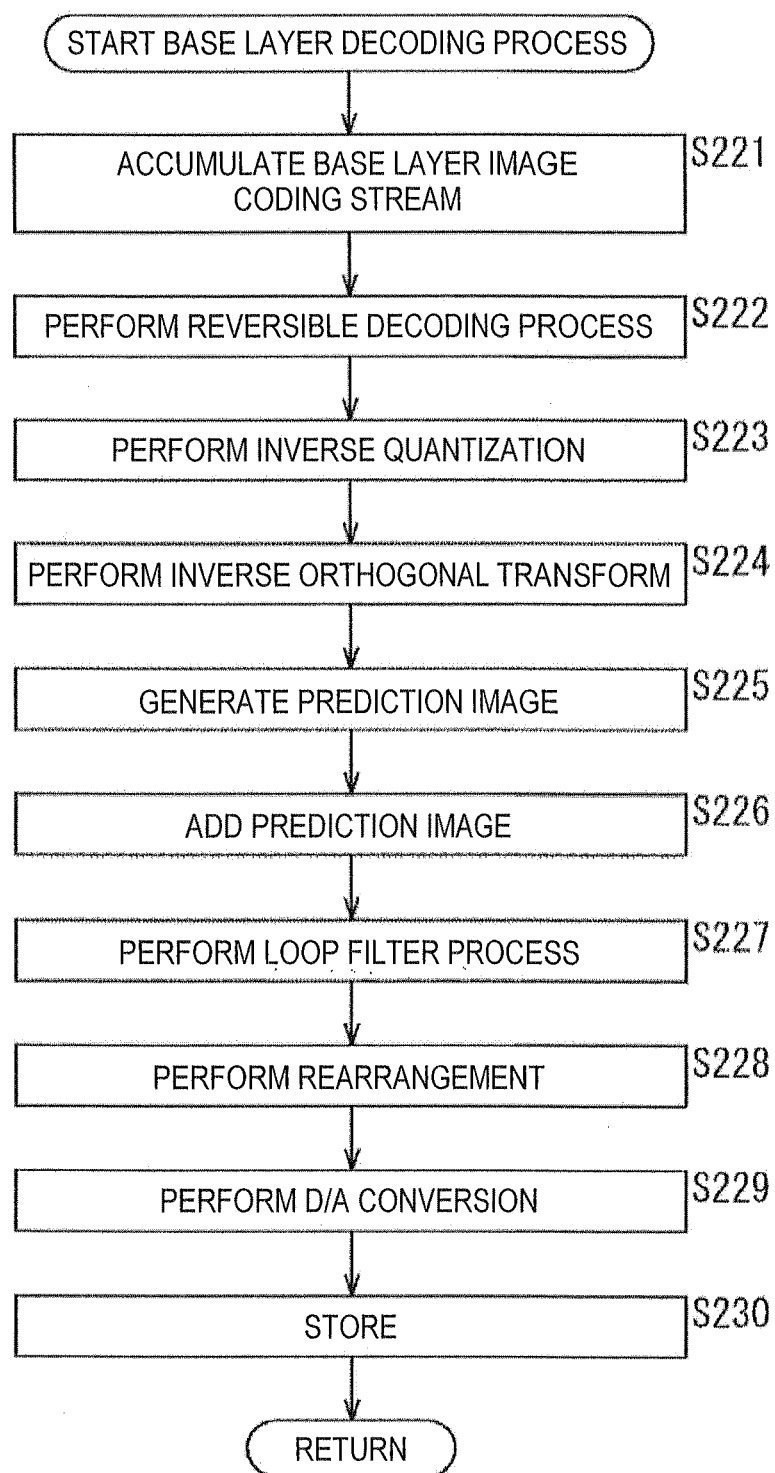
FIG. 41

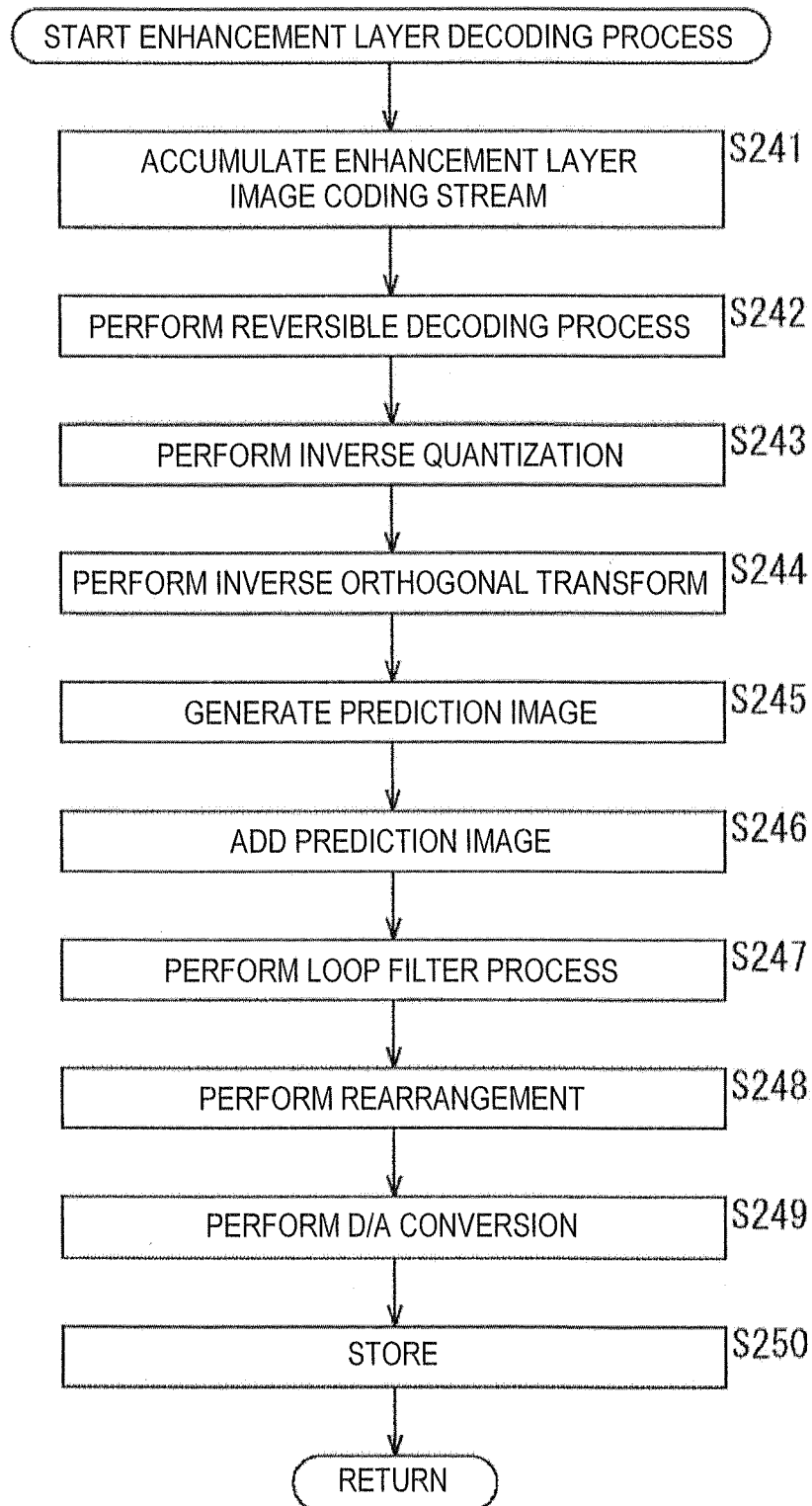
FIG. 42

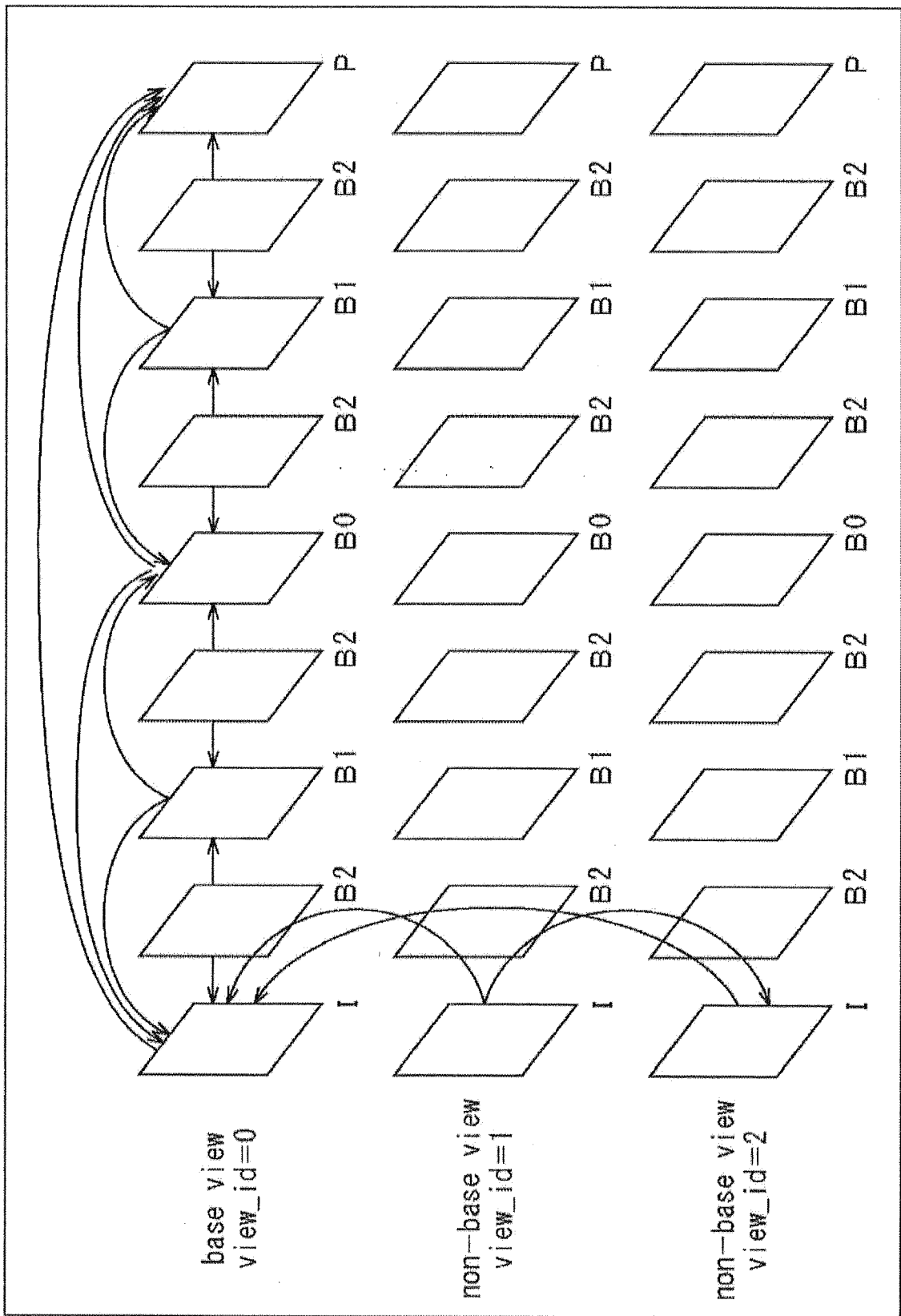
FIG. 43

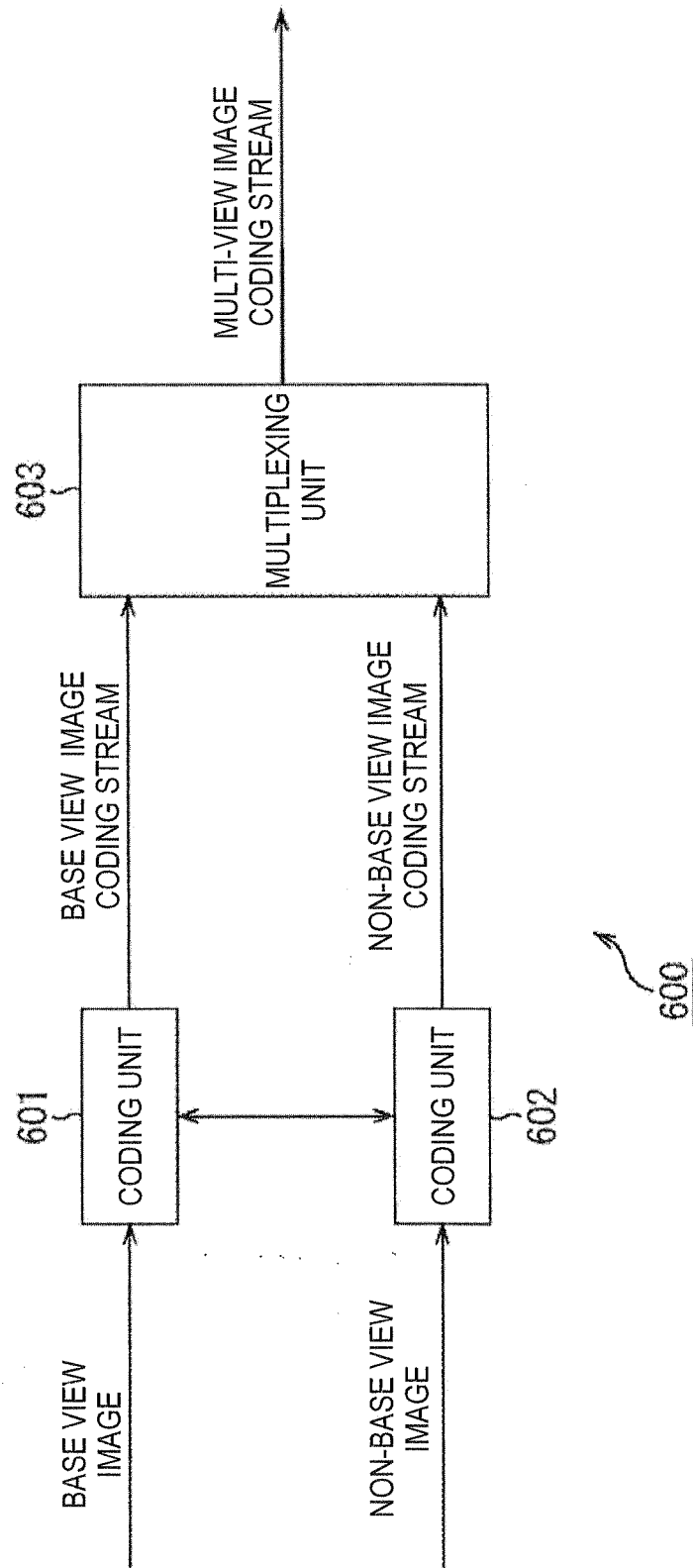
FIG. 44

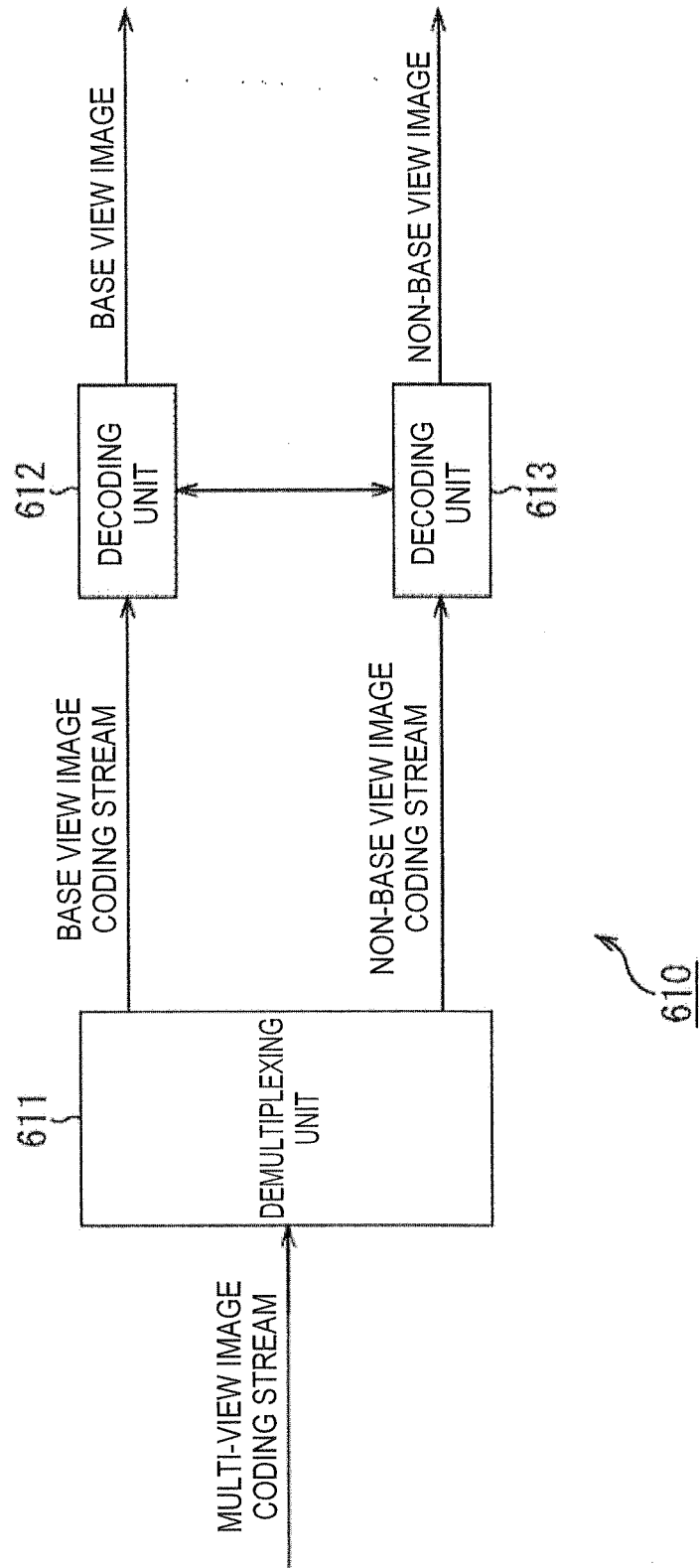
FIG. 45

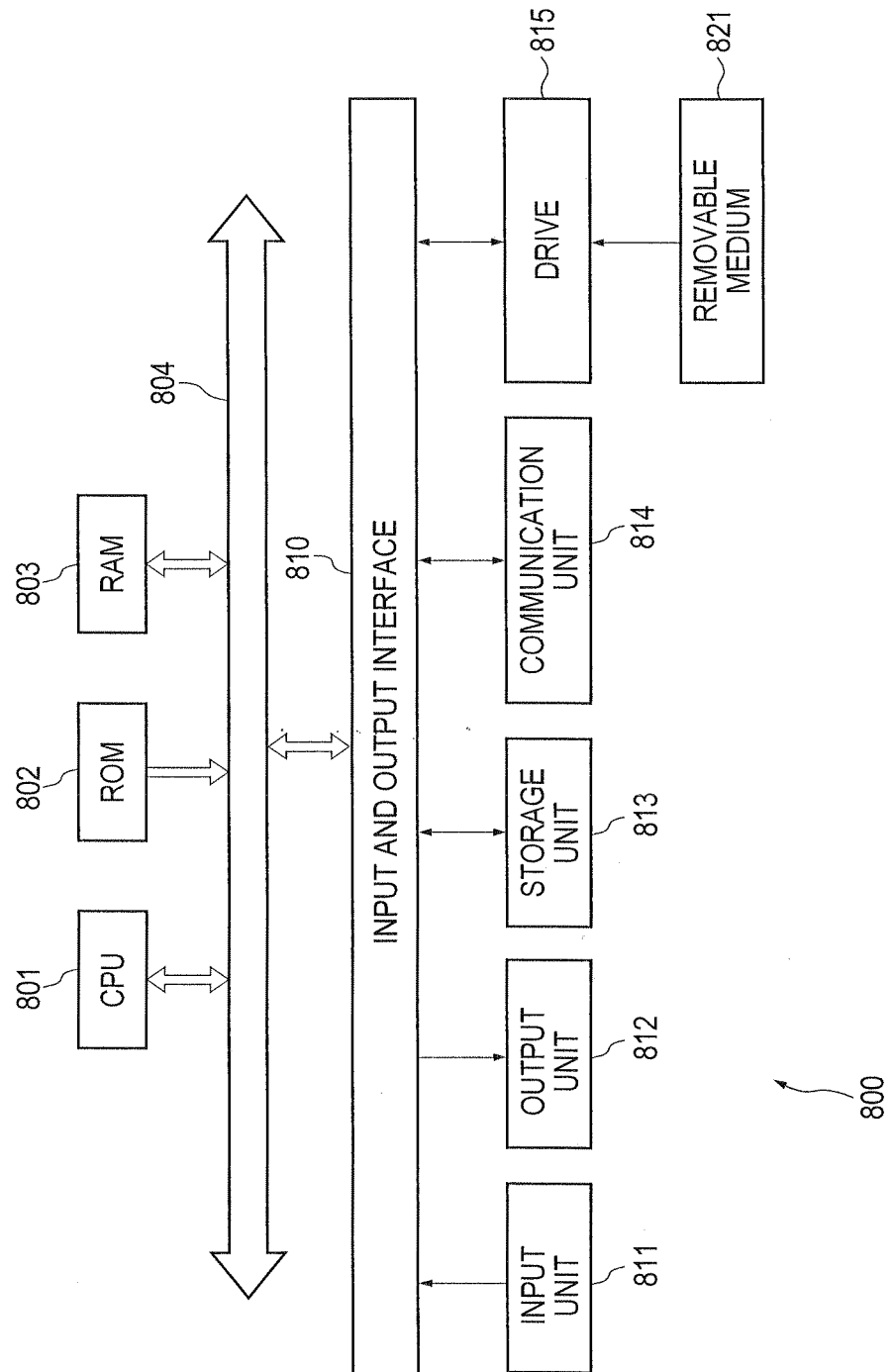
FIG. 46

FIG. 47

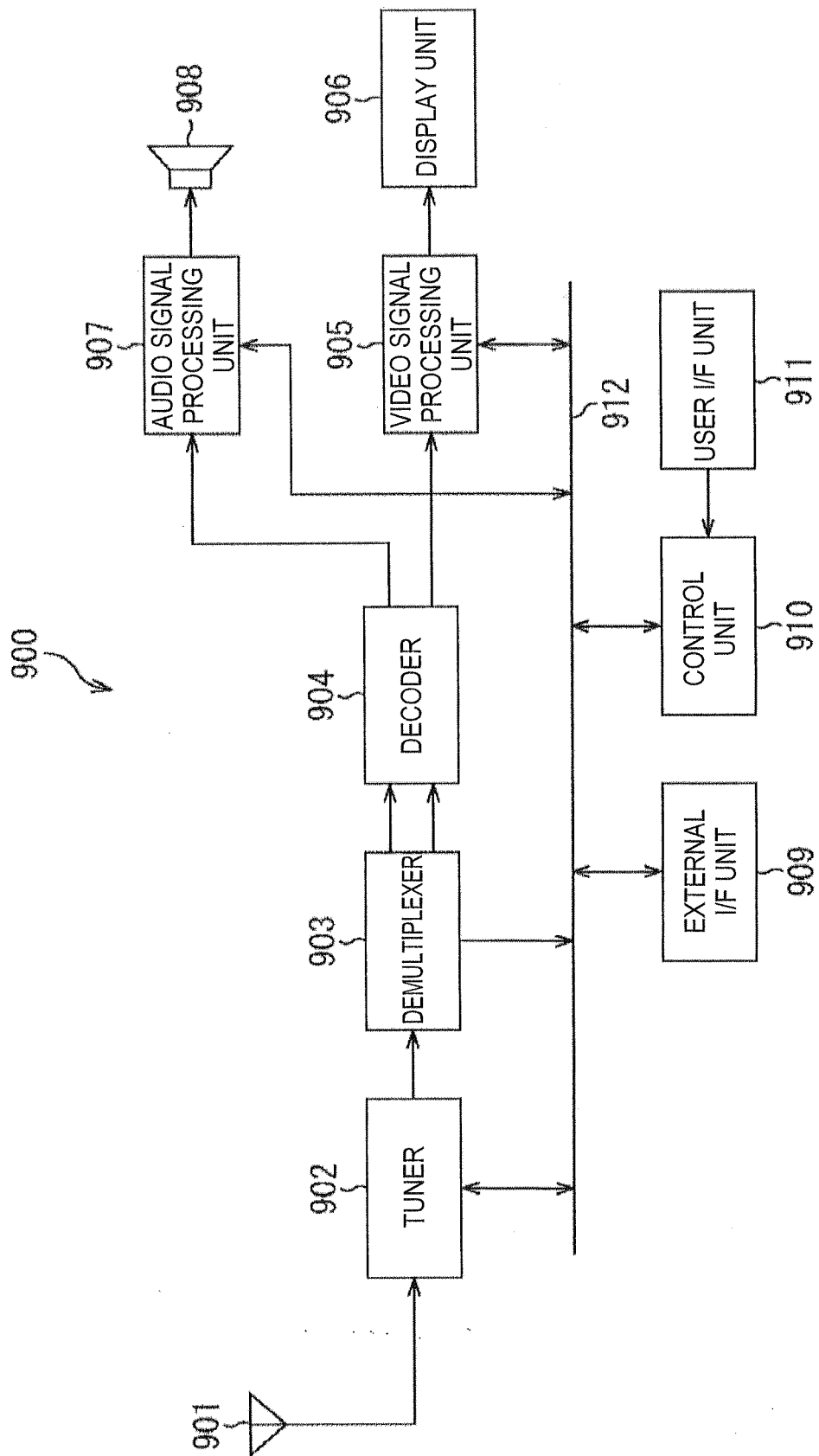


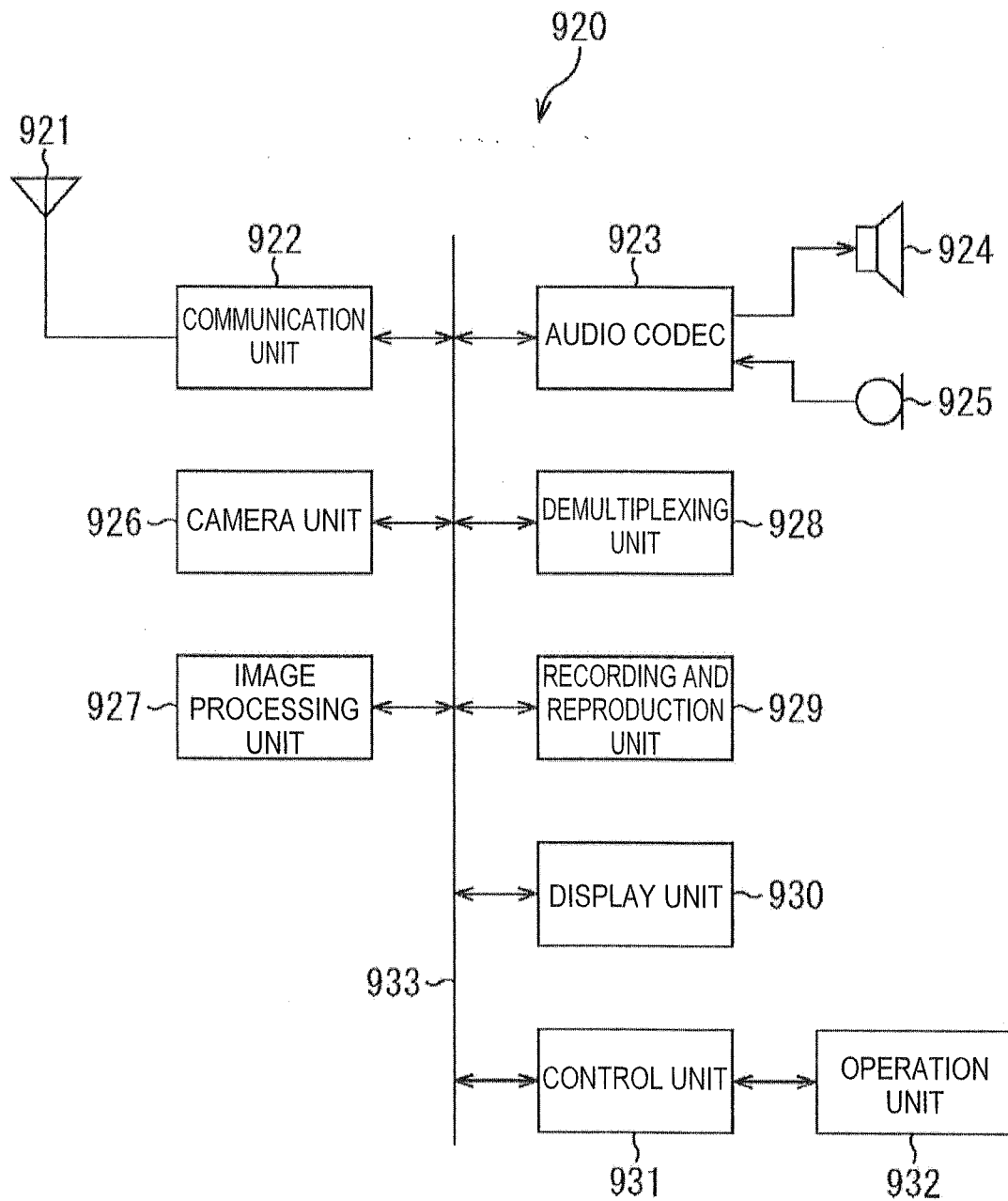
FIG. 48

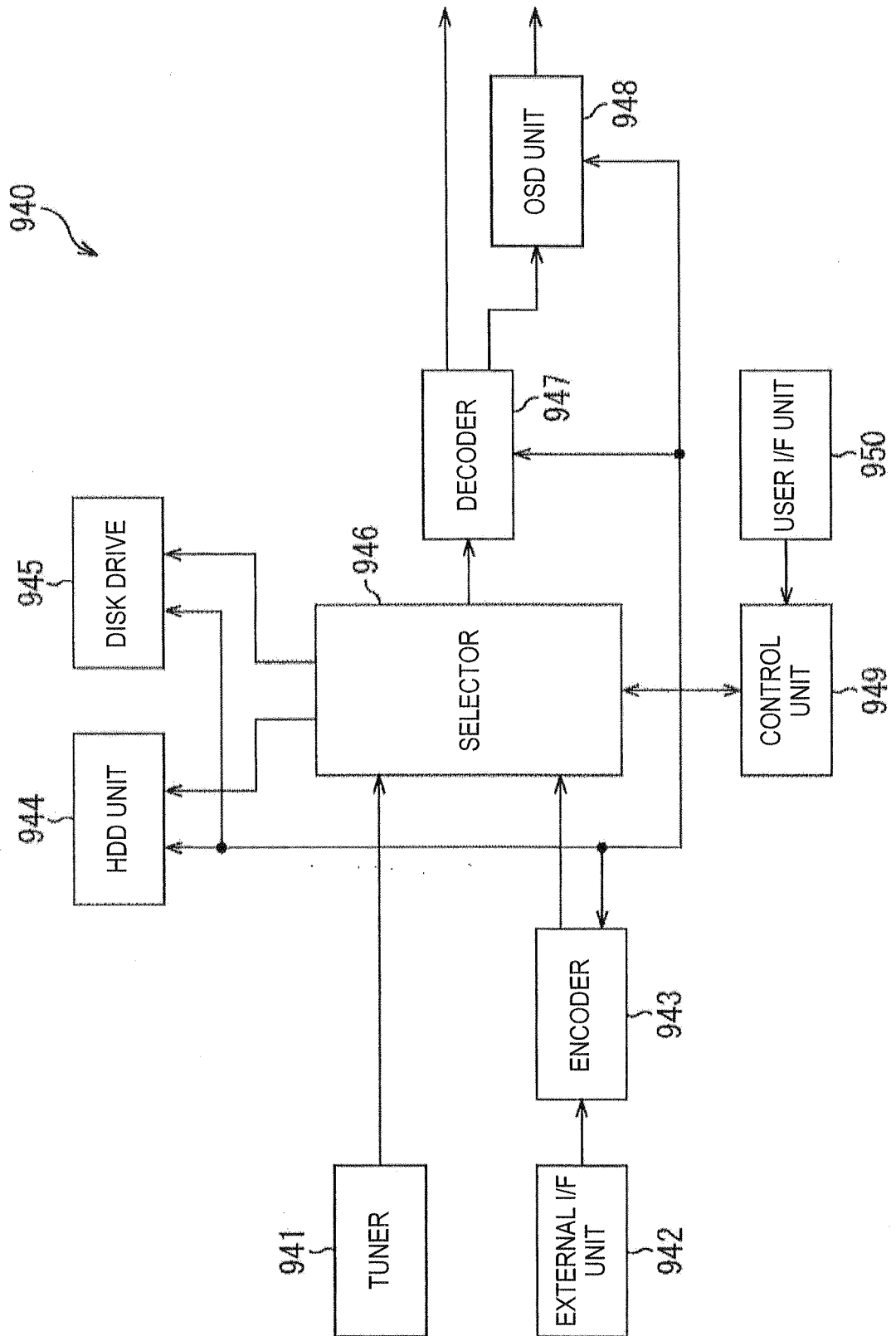
FIG. 49

FIG. 50

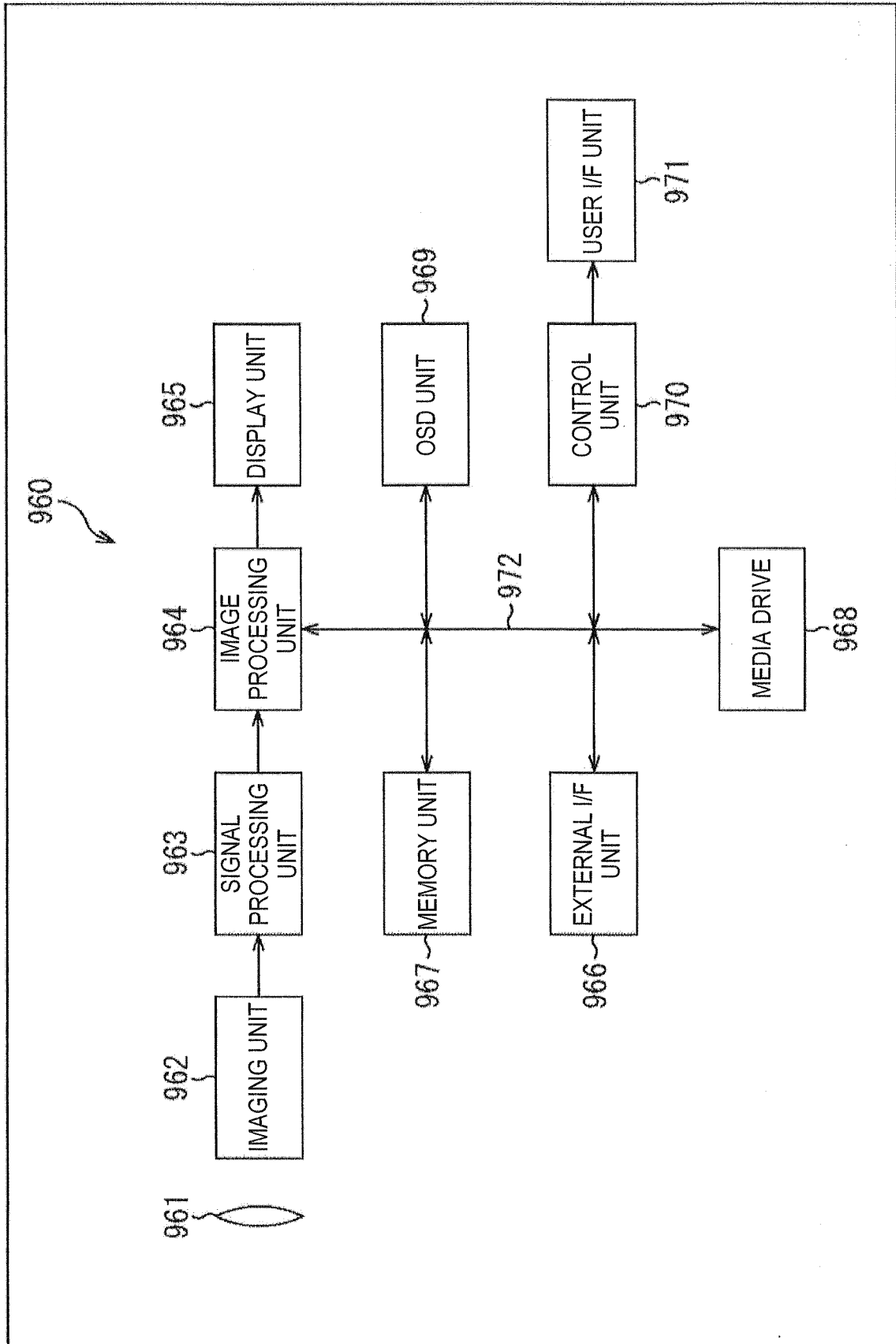


FIG. 51

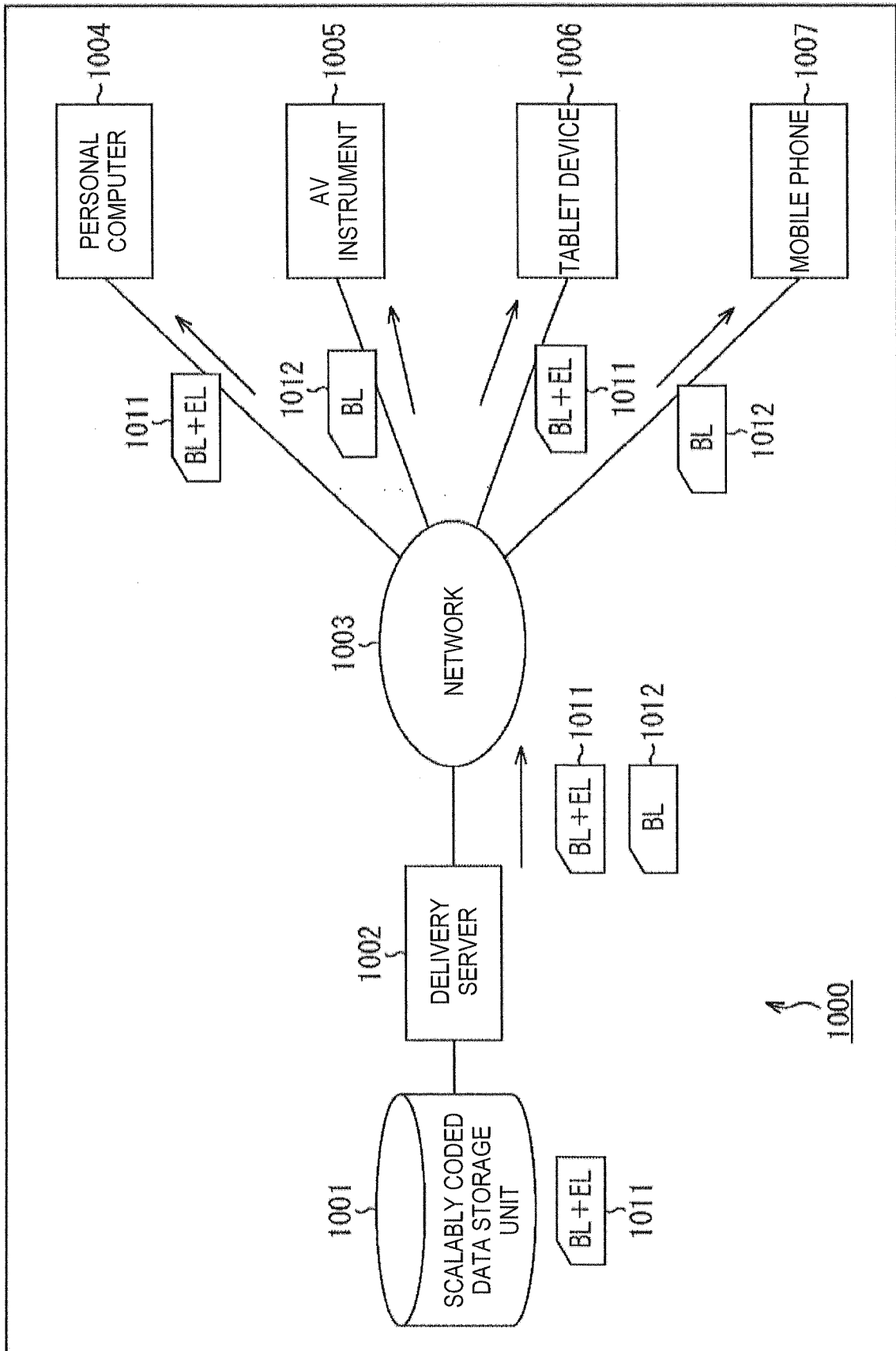


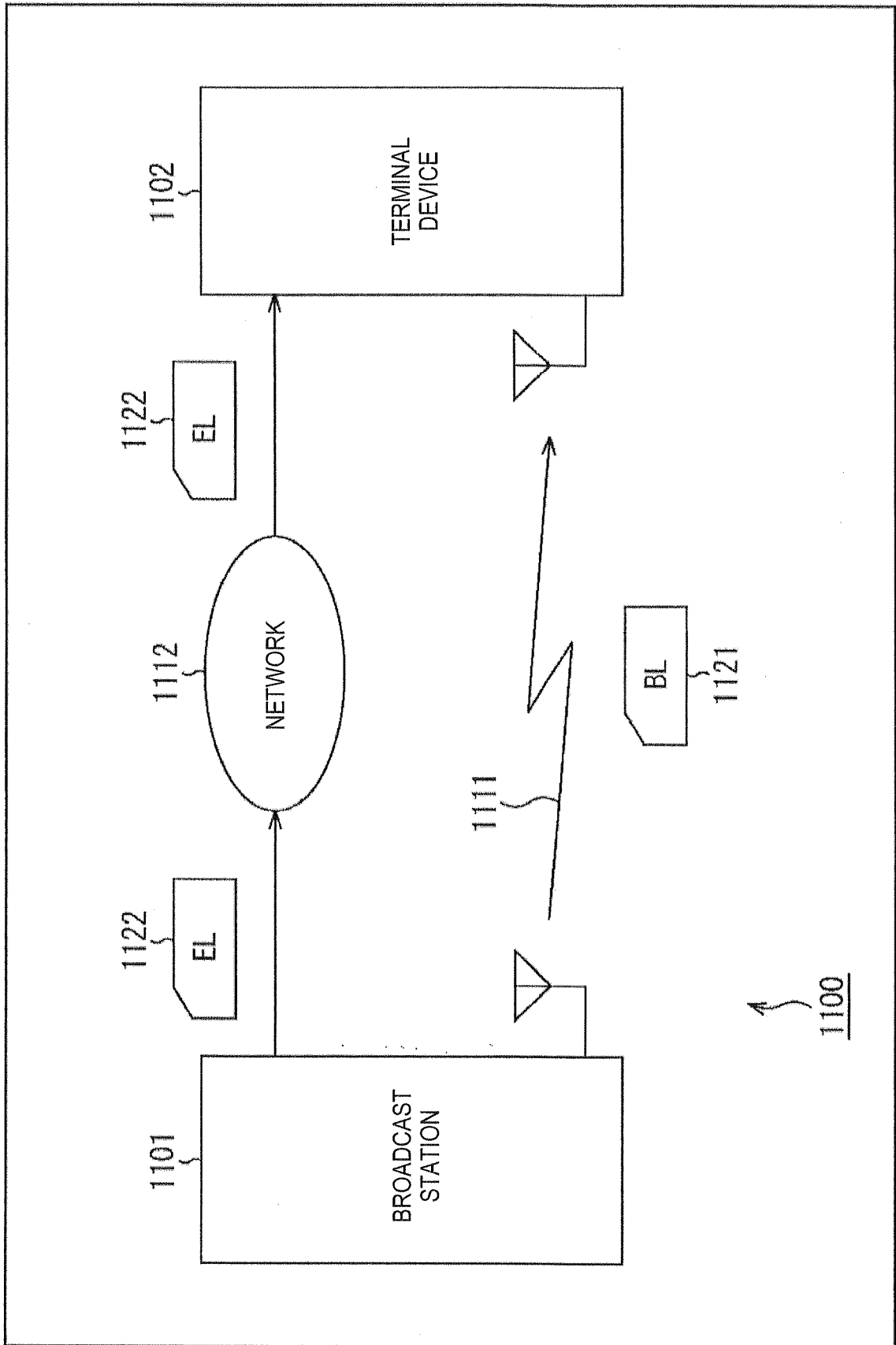
FIG. 52

FIG. 53

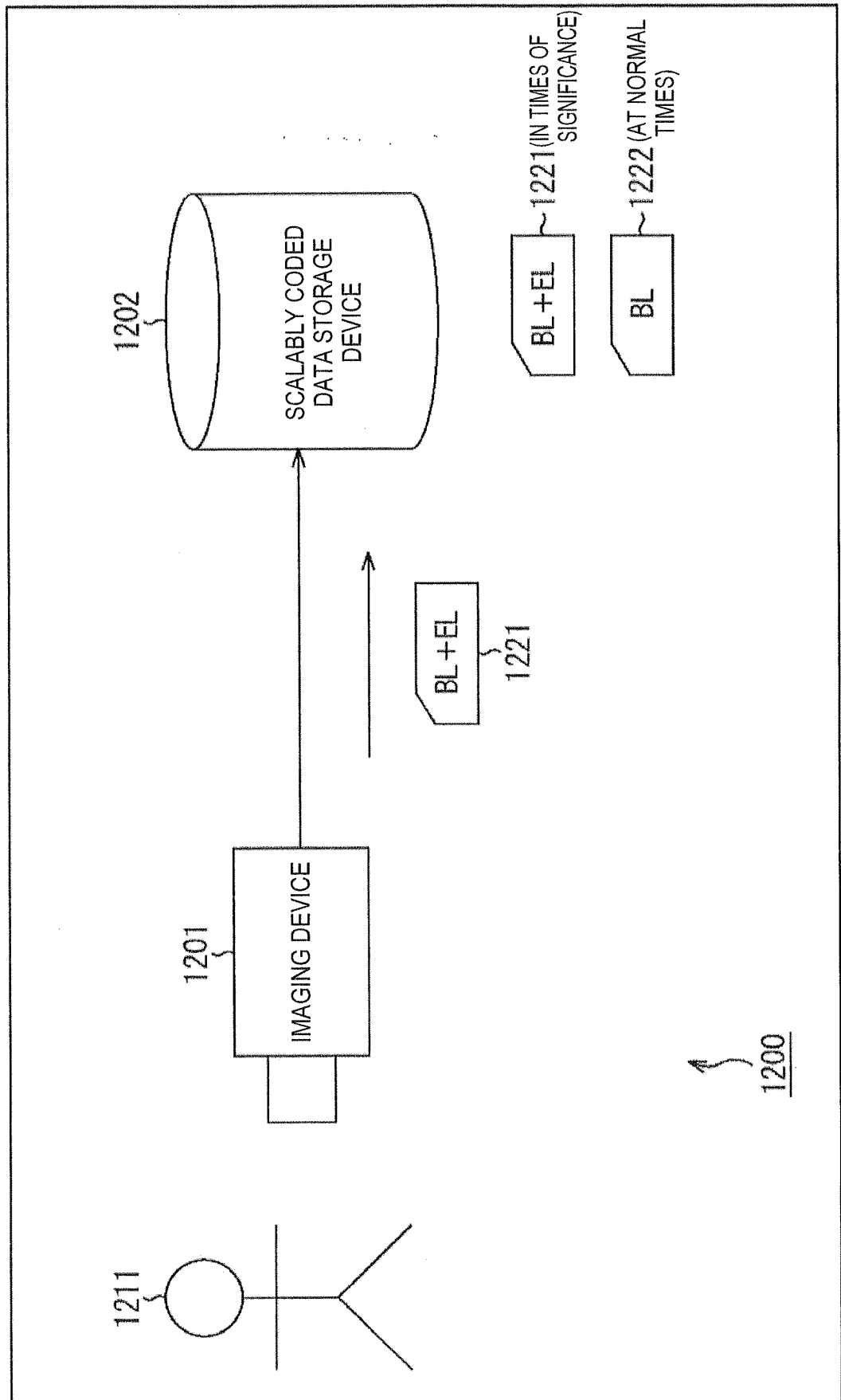


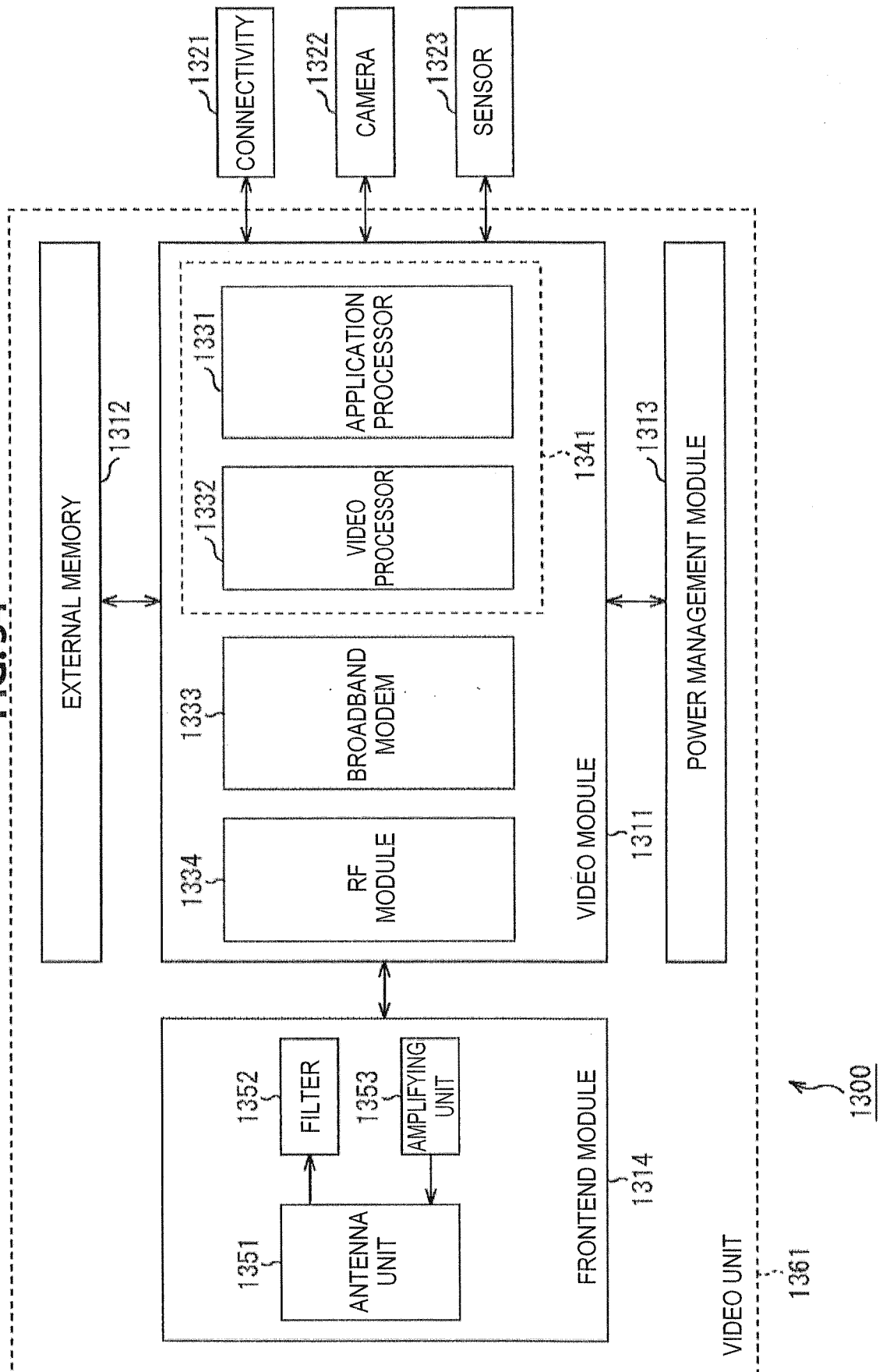
FIG. 54

FIG. 55

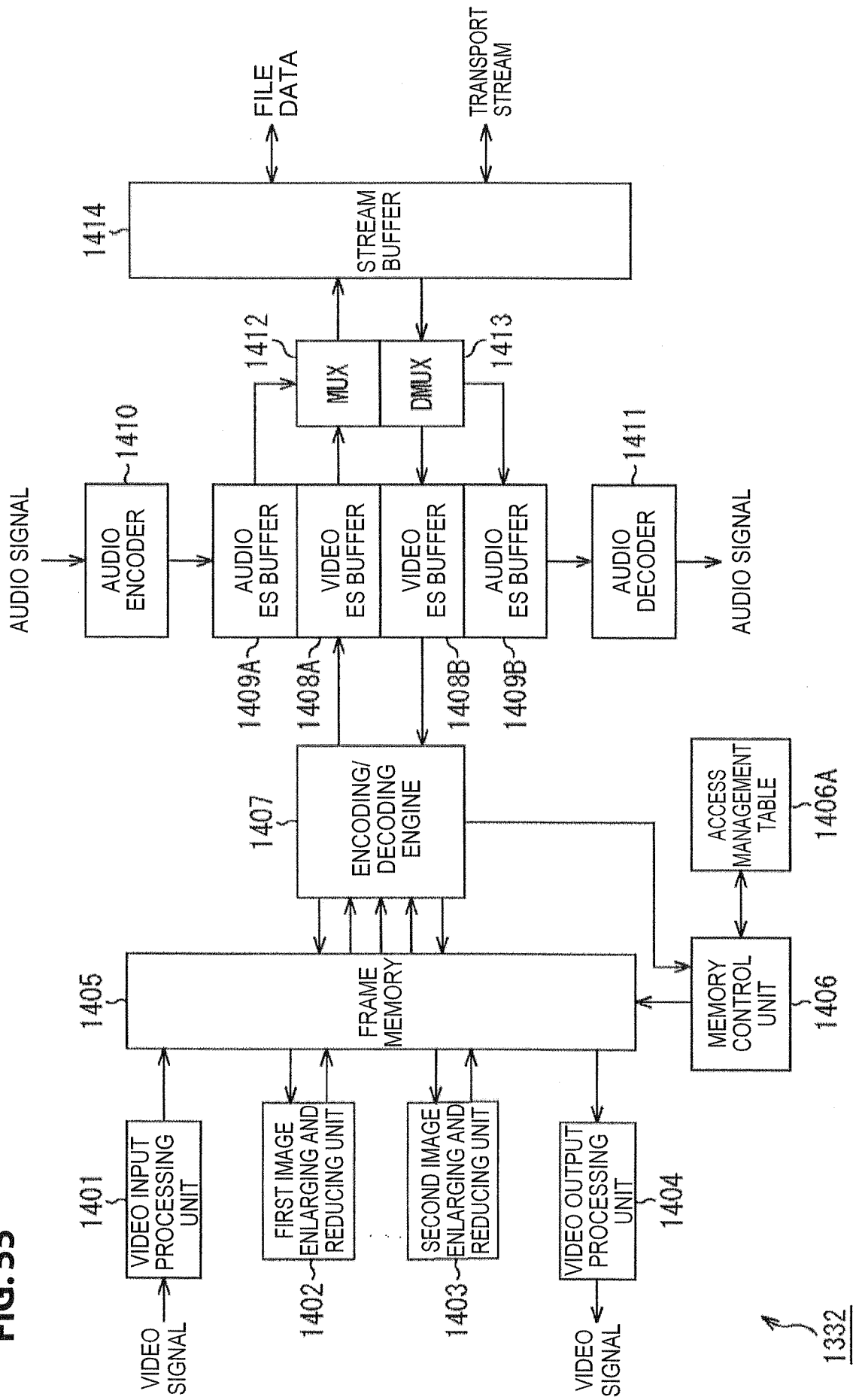


FIG. 56

