

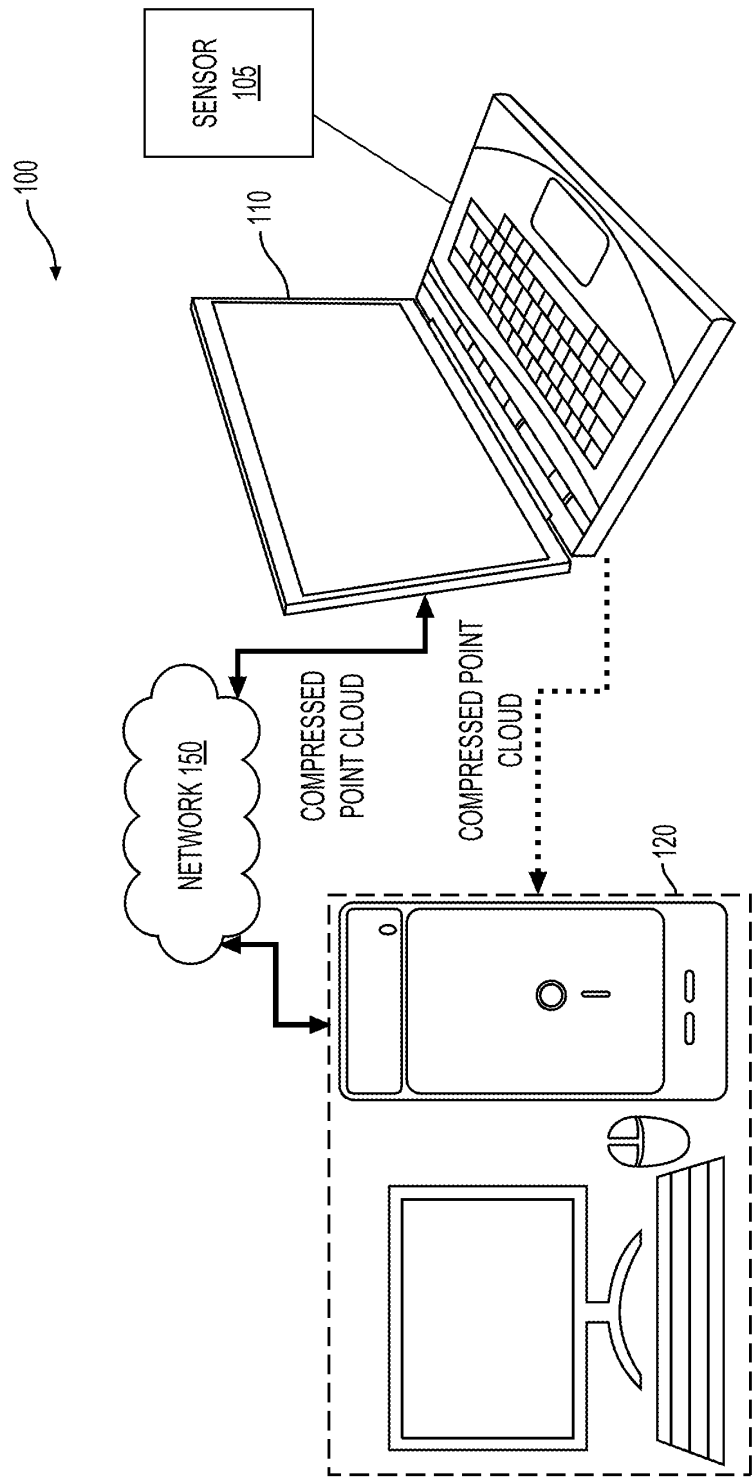


FIG. 1

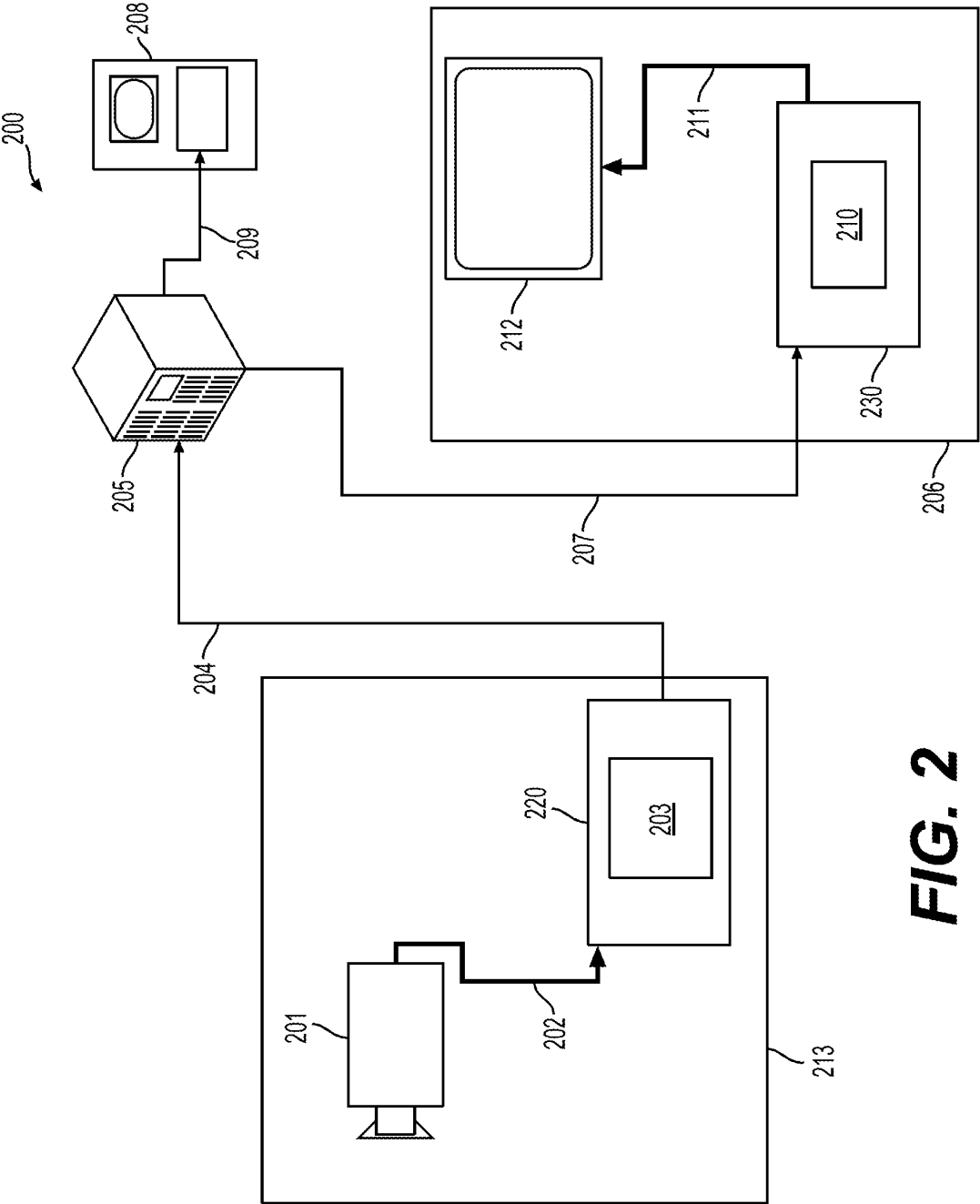


FIG. 2

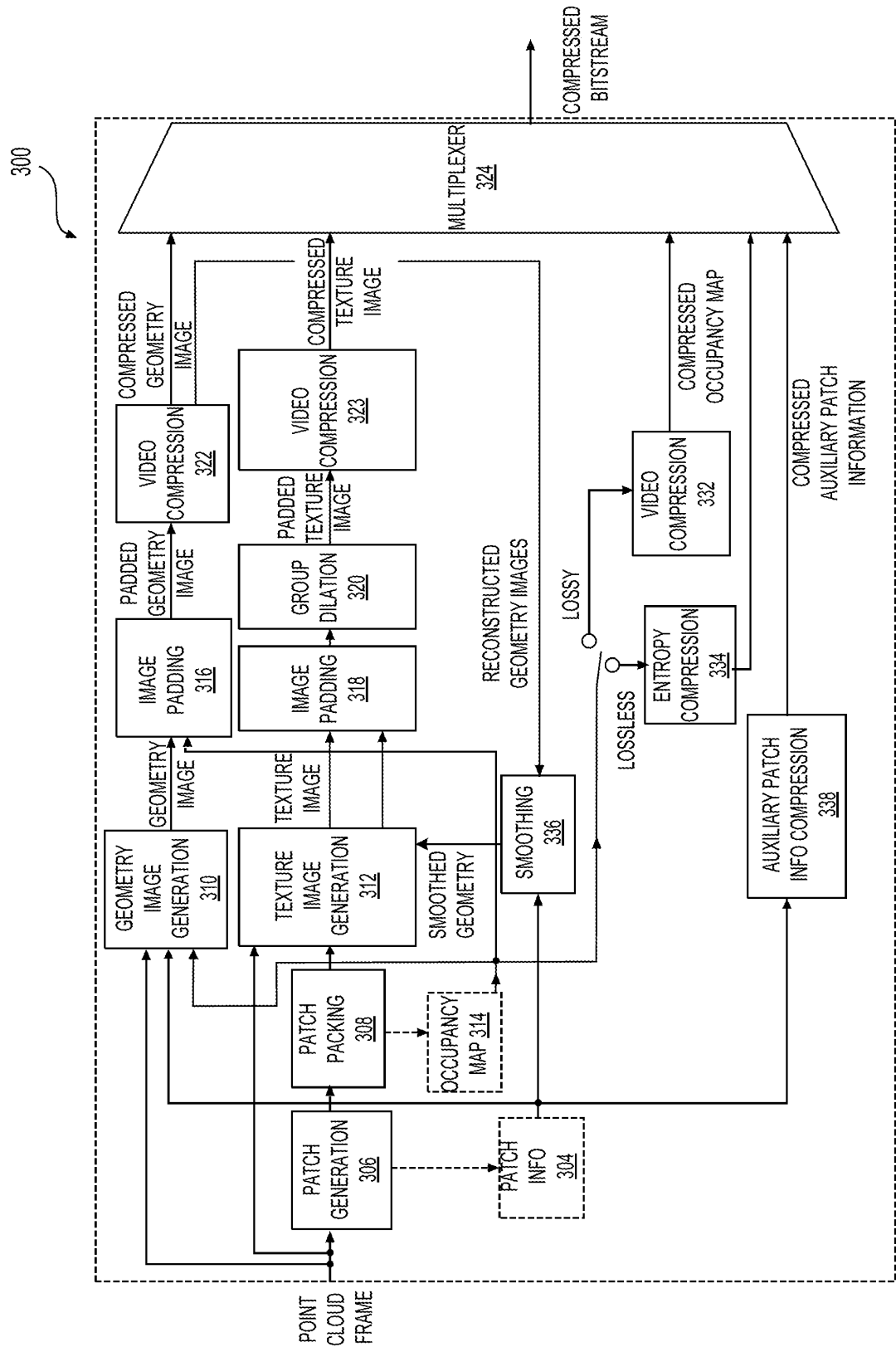


FIG. 3

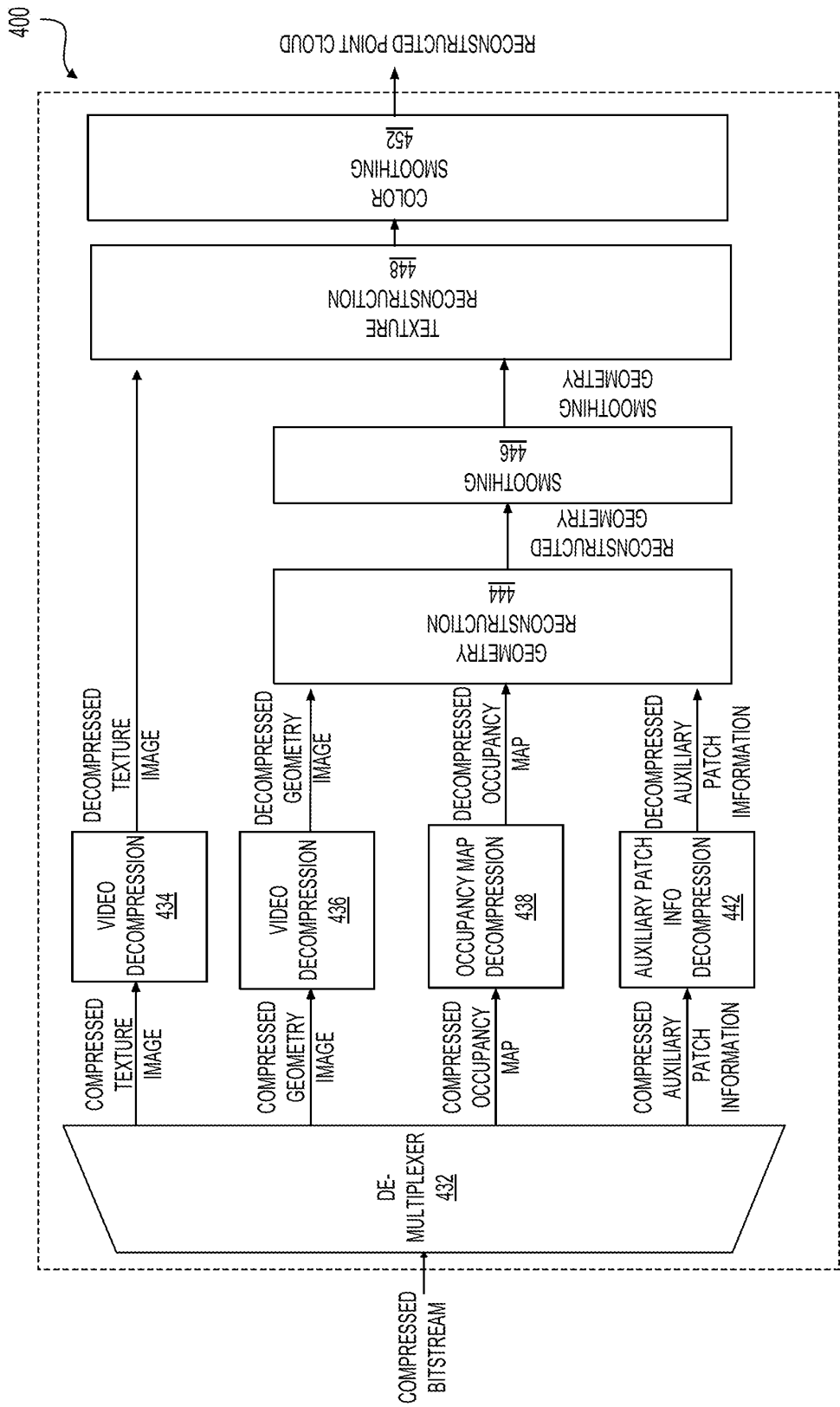


FIG. 4

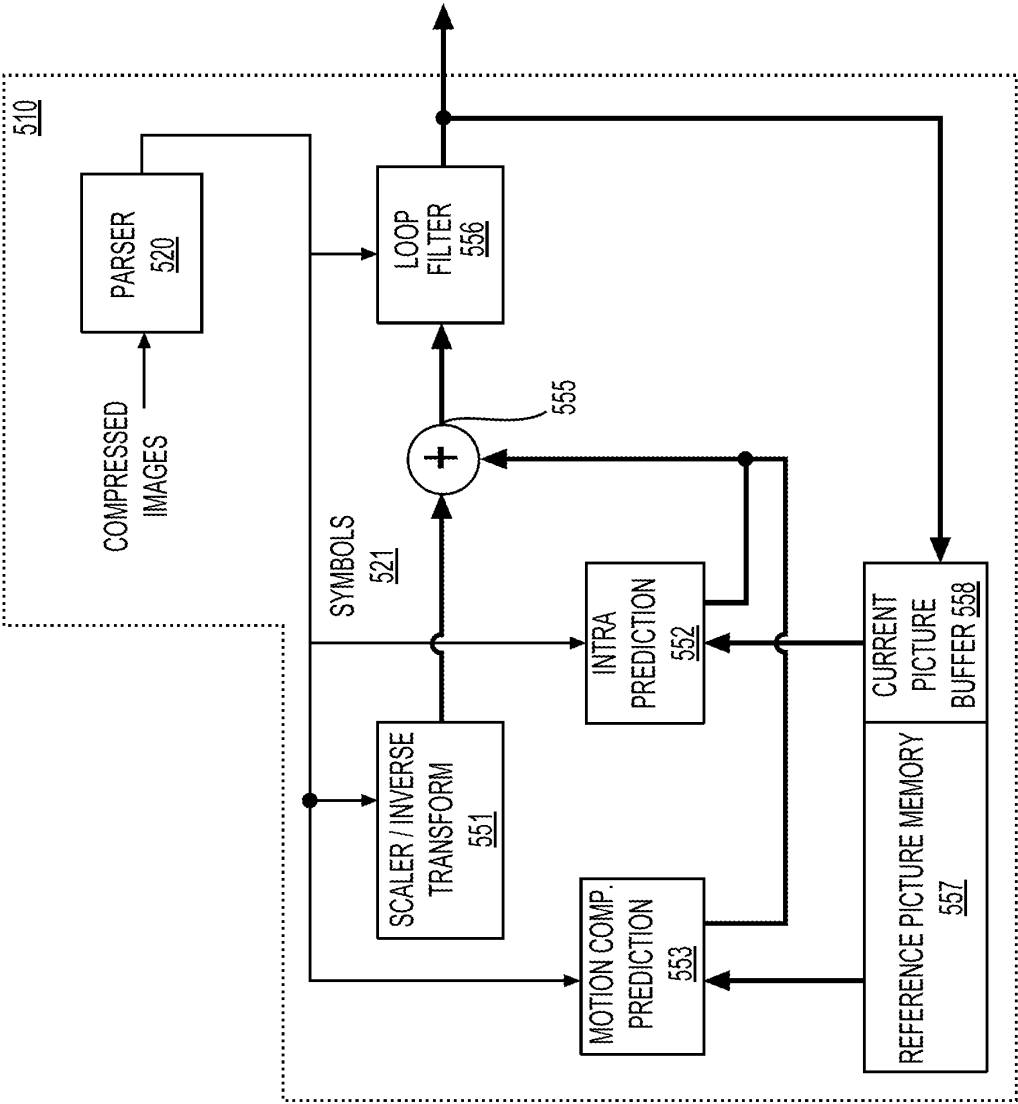


FIG. 5

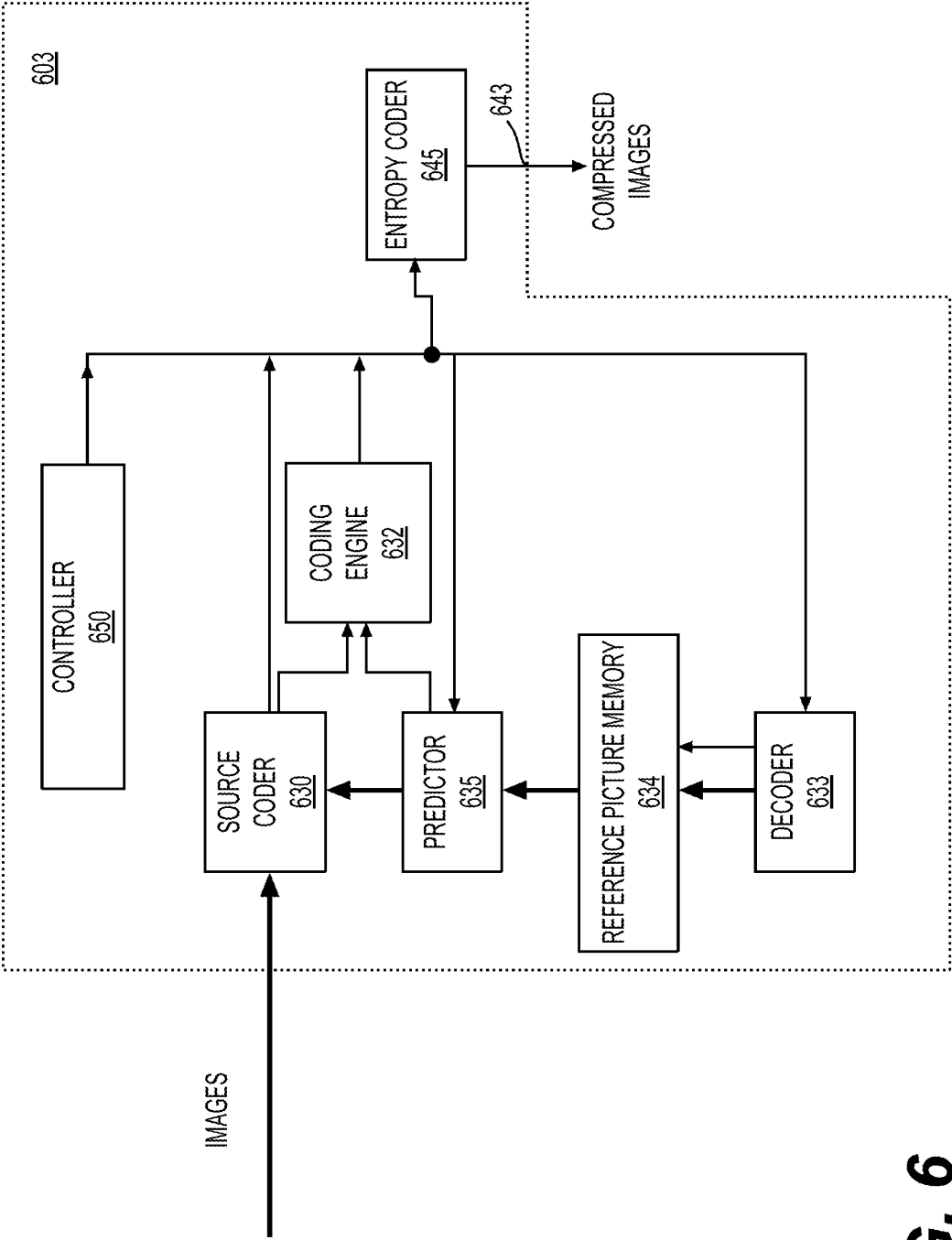


FIG. 6

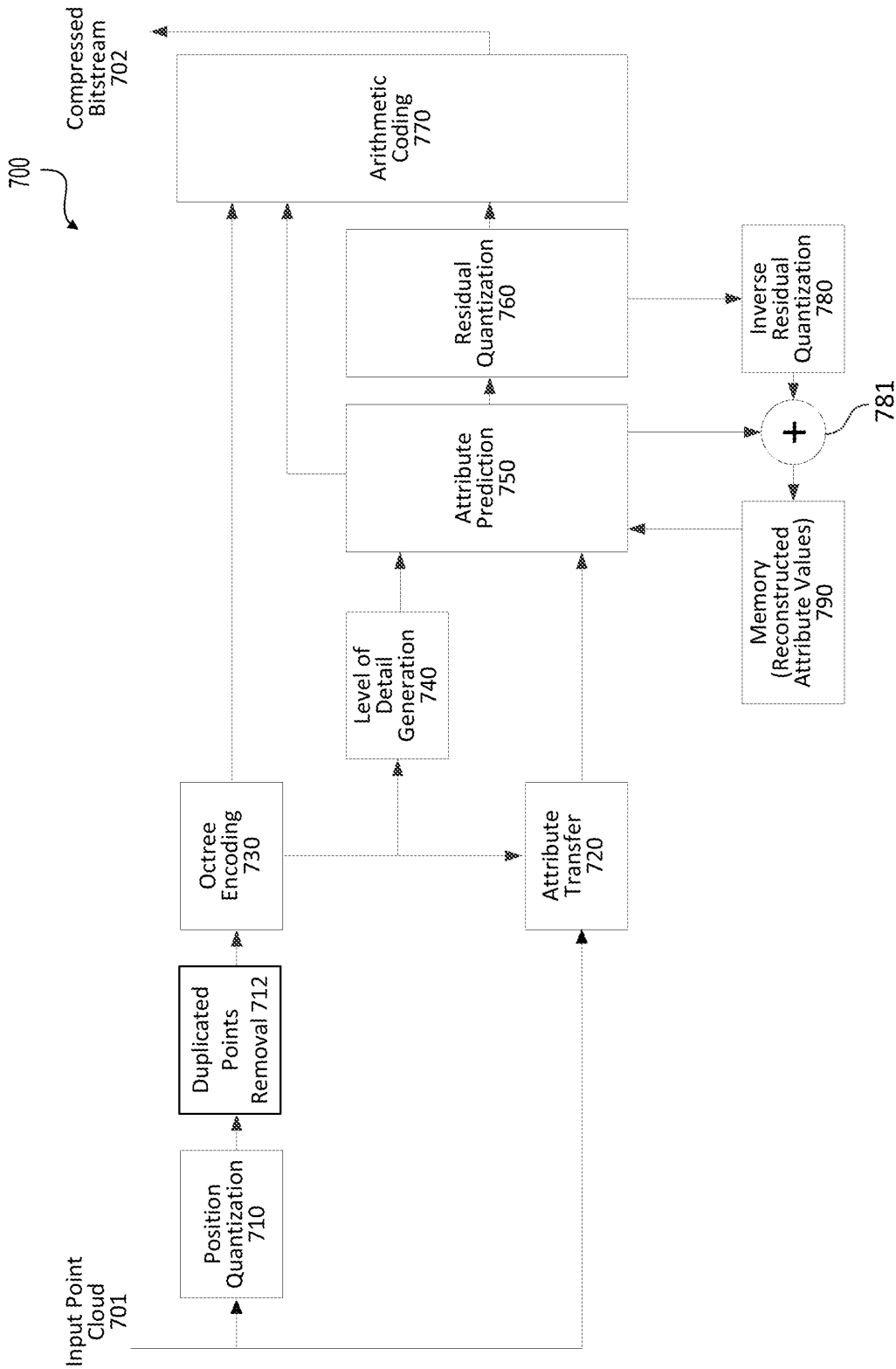


FIG. 7

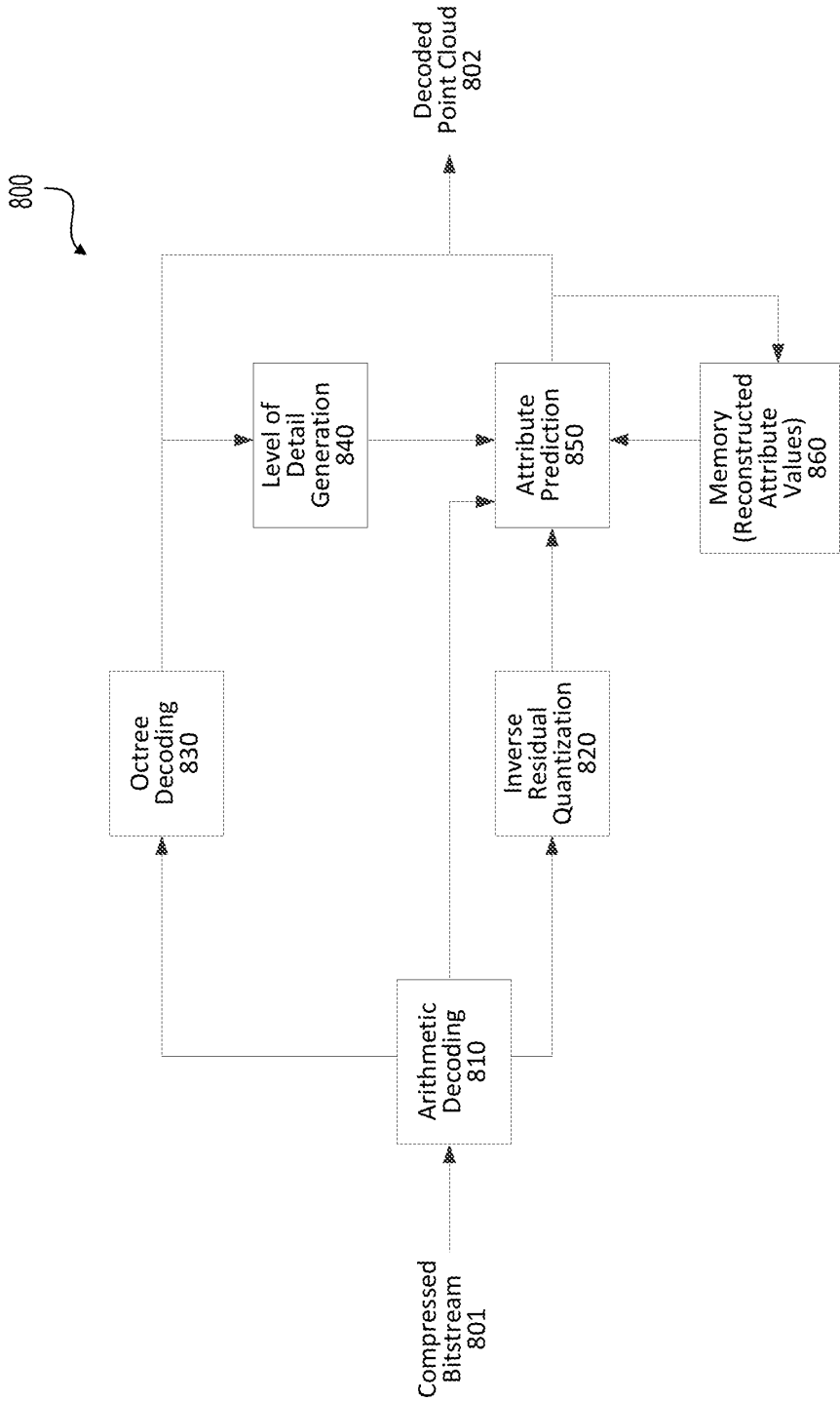


FIG. 8

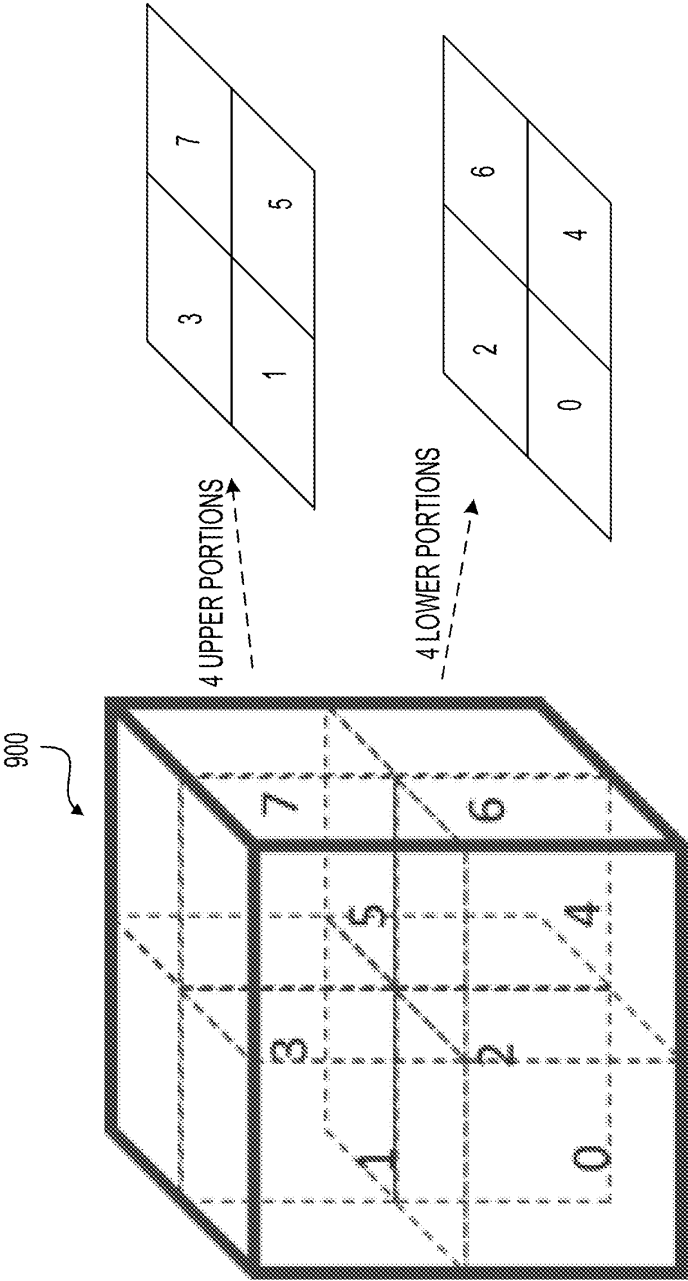


FIG. 9

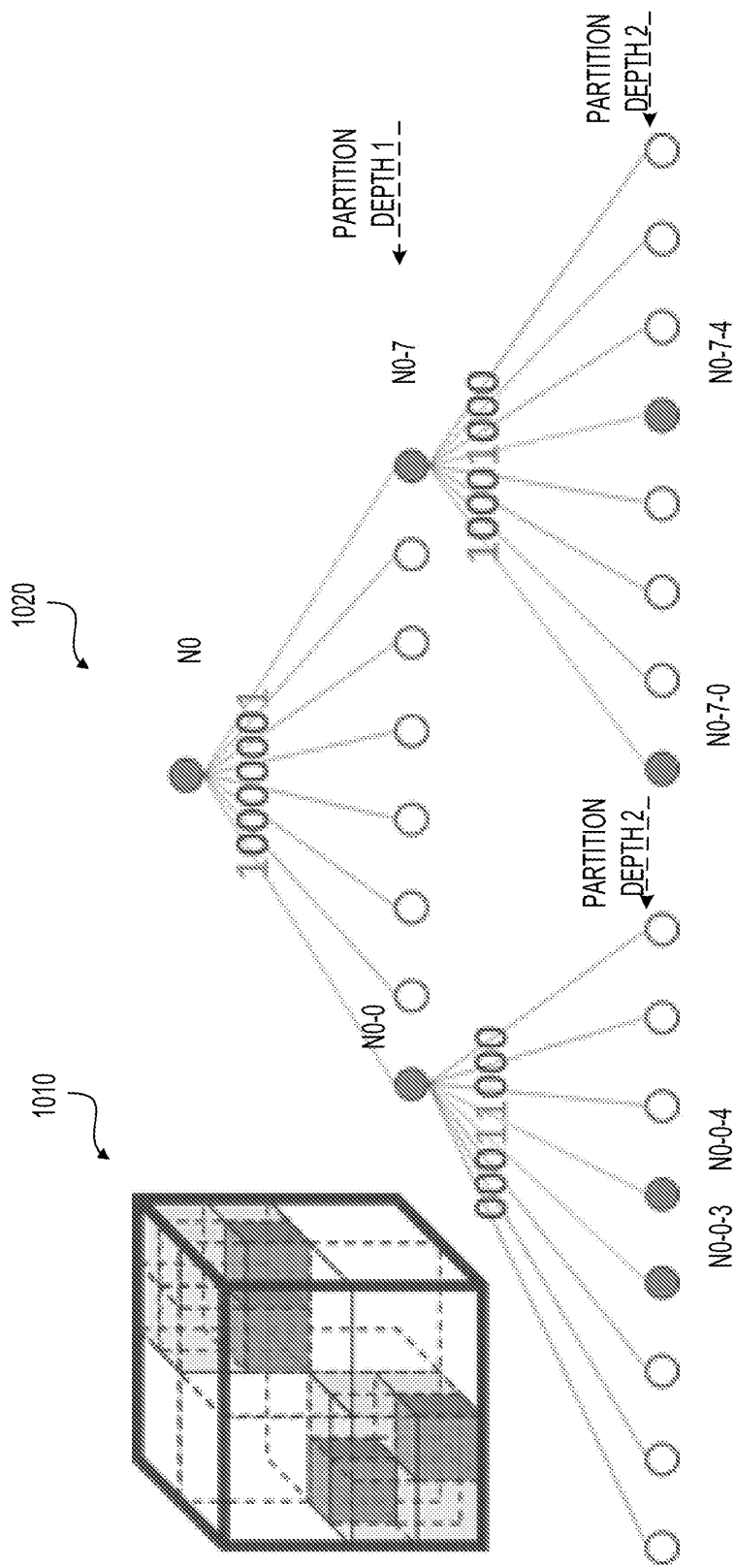


FIG. 10

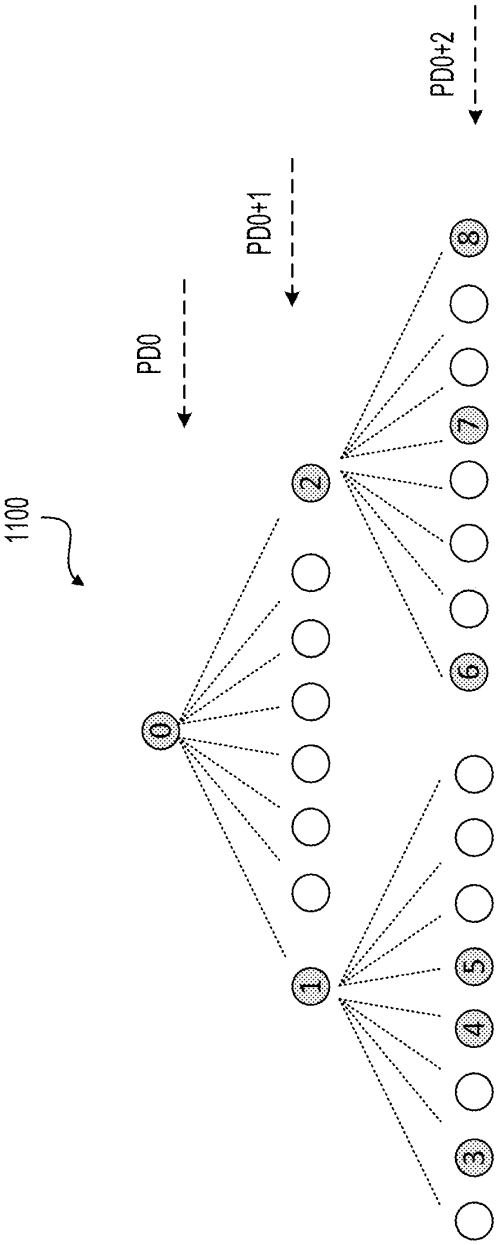


FIG. 11

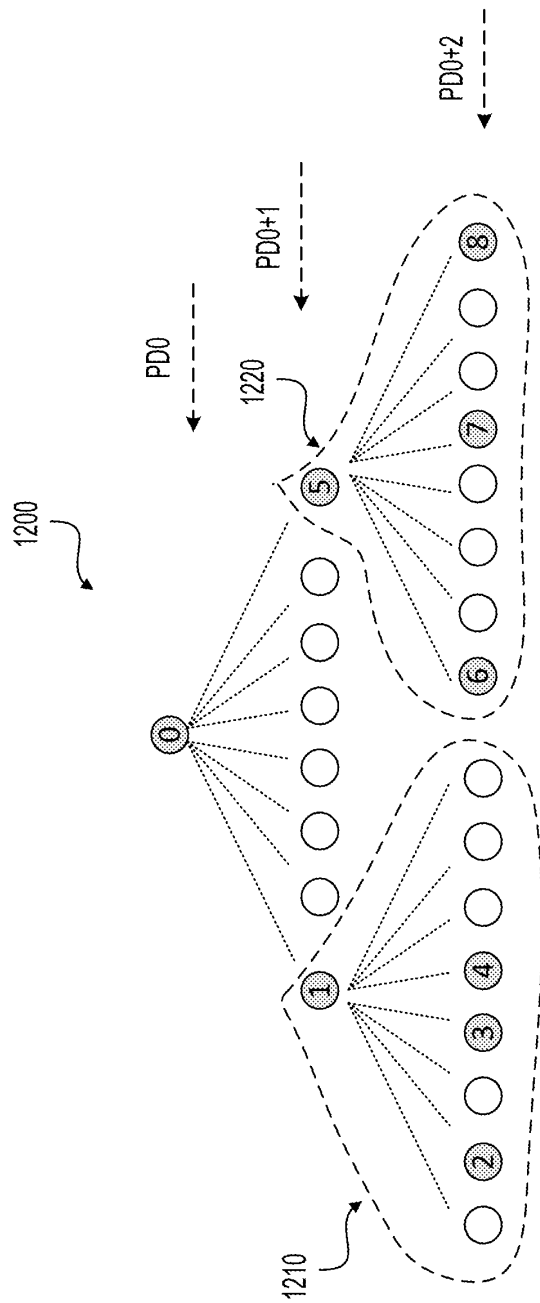


FIG. 12

	Descriptor
geometry_parameter_set() {	
gps_geom_parameter_set_id	ue(v)
gps_seq_parameter_set_id	ue(v)
gps_box_present_flag	u(1)
if(gps_box_present_flag){	
gps_gsh_box_log2_scale_present_flag	u(1)
if(gps_gsh_box_log2_scale_present_flag == 0)	
gps_gsh_box_log2_scale	ue(v)
}	
unique_geometry_points_flag	u(1)
neighbour_context_restriction_flag	u(1)
inferred_direct_coding_mode_enabled_flag	u(1)
bitwise_occupancy_coding_flag	u(1)
adjacent_child_contextualization_enabled_flag	u(1)
log2_neighbour_avail_boundary	ue(v)
log2_intra_pred_max_node_size	ue(v)
log2_trisoup_node_size	ue(v)
gps_depth_first_node_size_log2_minus_1	ue(v)
gps_extension_present_flag	u(1)
if(gps_extension_present_flag)	
while(more_data_in_byte_stream())	
gps_extension_data_flag	u(1)
byte_alignment()	
}	

FIG. 13

geometry_parameter_set() {		Descriptor
gps_geom_parameter_set_id		ue(v)
...		...
log2_trisoup_node_size		ue(v)
gps_hybrid_coding_order_flag		u(1)
if (gps_hybrid_coding_order_flag)		
gps_depth_first_node_size_log2_minus_2		ue(v)
...		...
byte_alignment()		
}		

FIG. 14

1500

geometry_slice_data() {	Descriptor
for(depth = 0; depth < MaxGeometryOctreeDepth; depth++) {	
for(nodeIdIdx = 0; nodeIdIdx < NumNodesAtDepth[depth]; nodeIdIdx++) {	
xN = NodeX[depth][nodeIdIdx]	
yN = NodeY[depth][nodeIdIdx]	
zN = NodeZ[depth][nodeIdIdx]	
if (depth >= MaxGeometryOctreeDepth - d_t) {	
geometry_node_depth_first (depth, nodeIdIdx, xN, yN, zN)	
} else {	
geometry_node(depth, nodeIdIdx, xN, yN, zN)	
}	
}	
}	
if (log2_trisoup_node_size > 0)	
geometry_trisoup_data()	
}	

1510

FIG. 15

1600

Descriptor
geometry_node_depth_first (depth, nodeIdx, xN, yN, zN) {
if(depth >= MaxGeometryOctreeDepth-1) {
return
}
geometry_node (depth, nodeIdx, xN, yN, zN)
nodeIdx = NumNodesAtDepth[depth + 1]
for(child = 0; child < GeometryNodeChildrenCnt; child++) {
childIdx = GeometryNodeChildren[child]
x = 2 × xN + (childIdx & 4 == 1)
y = 2 × yN + (childIdx & 2 == 1)
z = 2 × zN + (childIdx & 1 == 1)
geometry_node_depth_first (depth+1, nodeIdx, x, y, z)
nodeIdx++
}
NumNodesAtDepth[depth + 1] = nodeIdx
}

FIG. 16

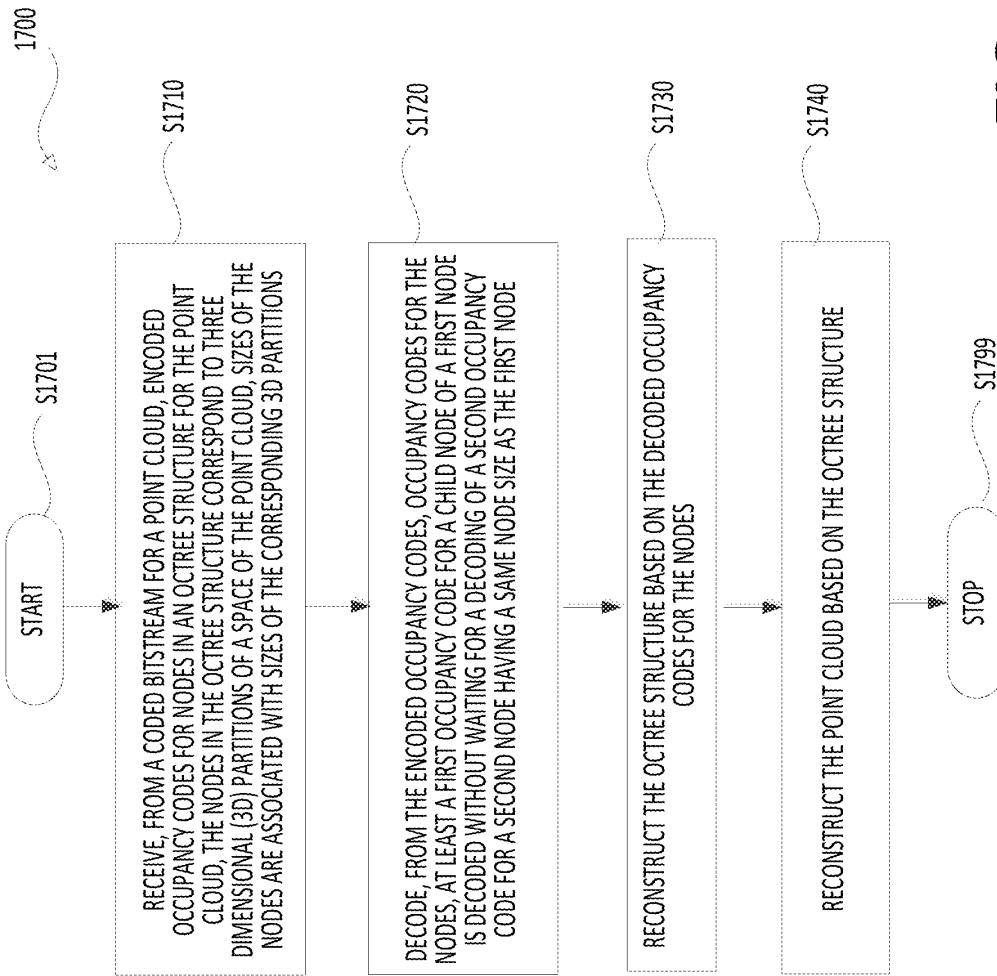


FIG. 17

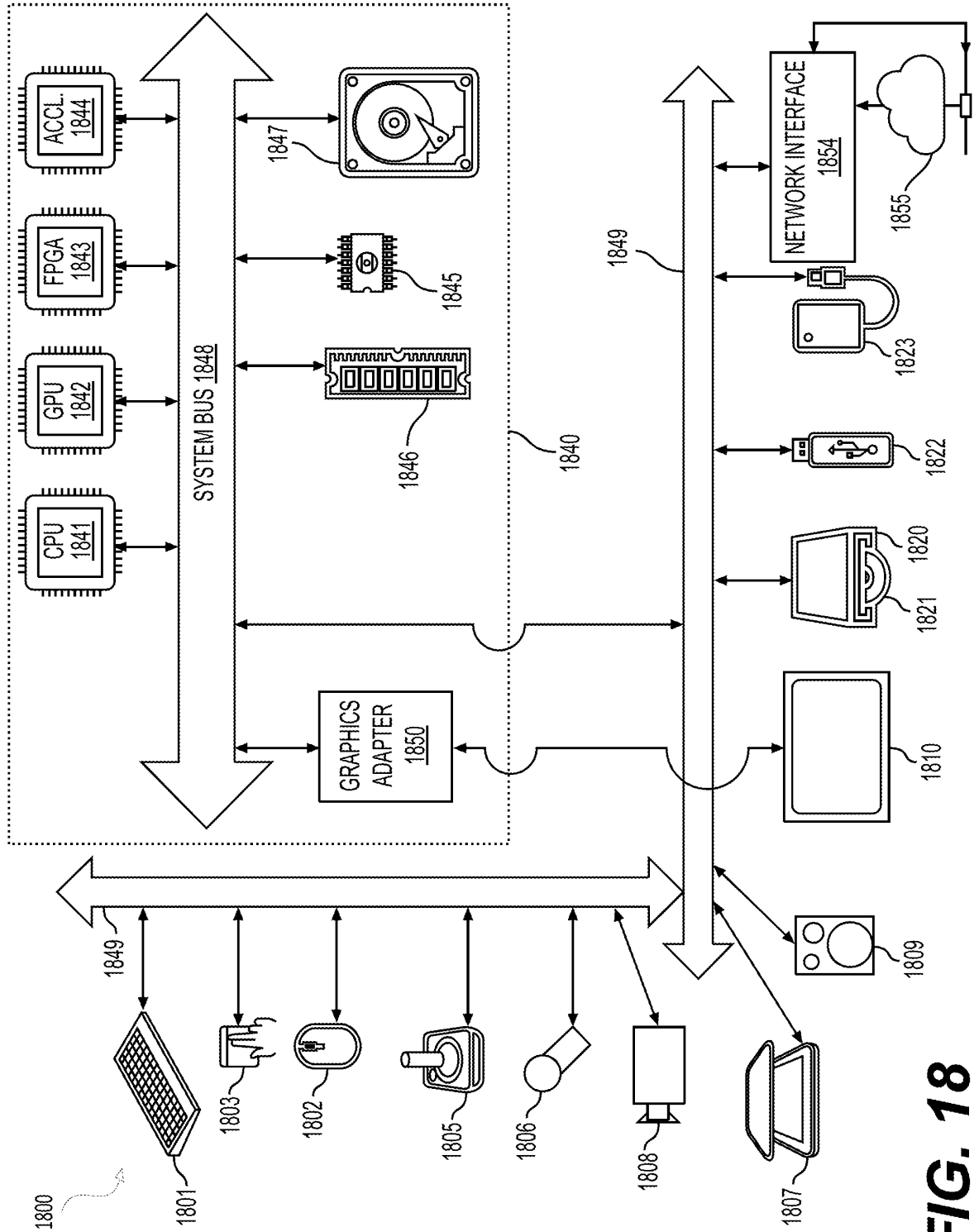


FIG. 18