

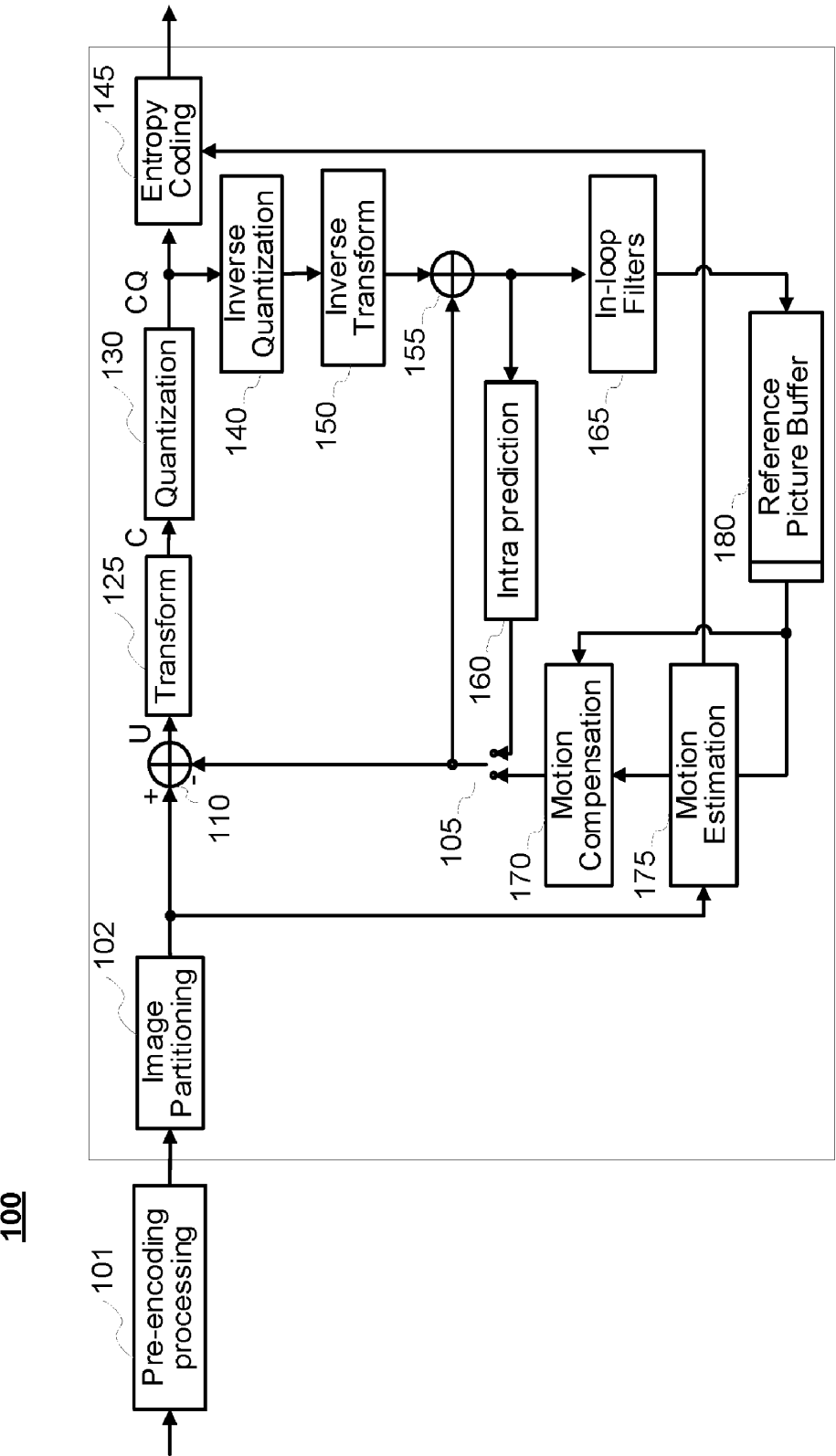


FIG. 1

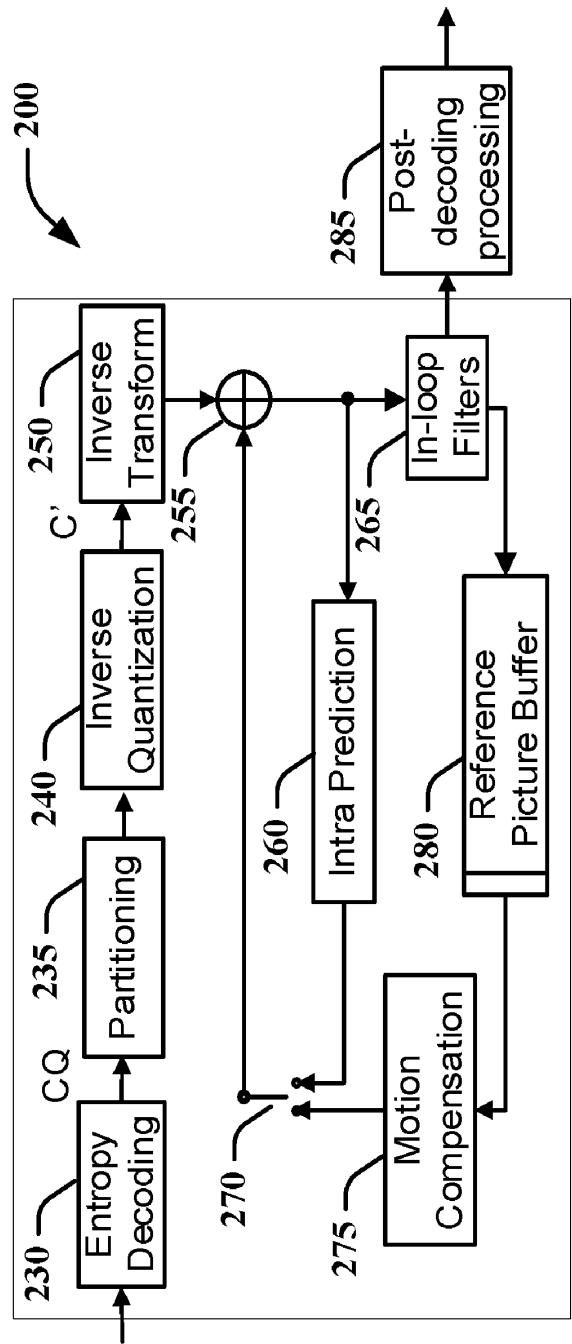


FIG. 2

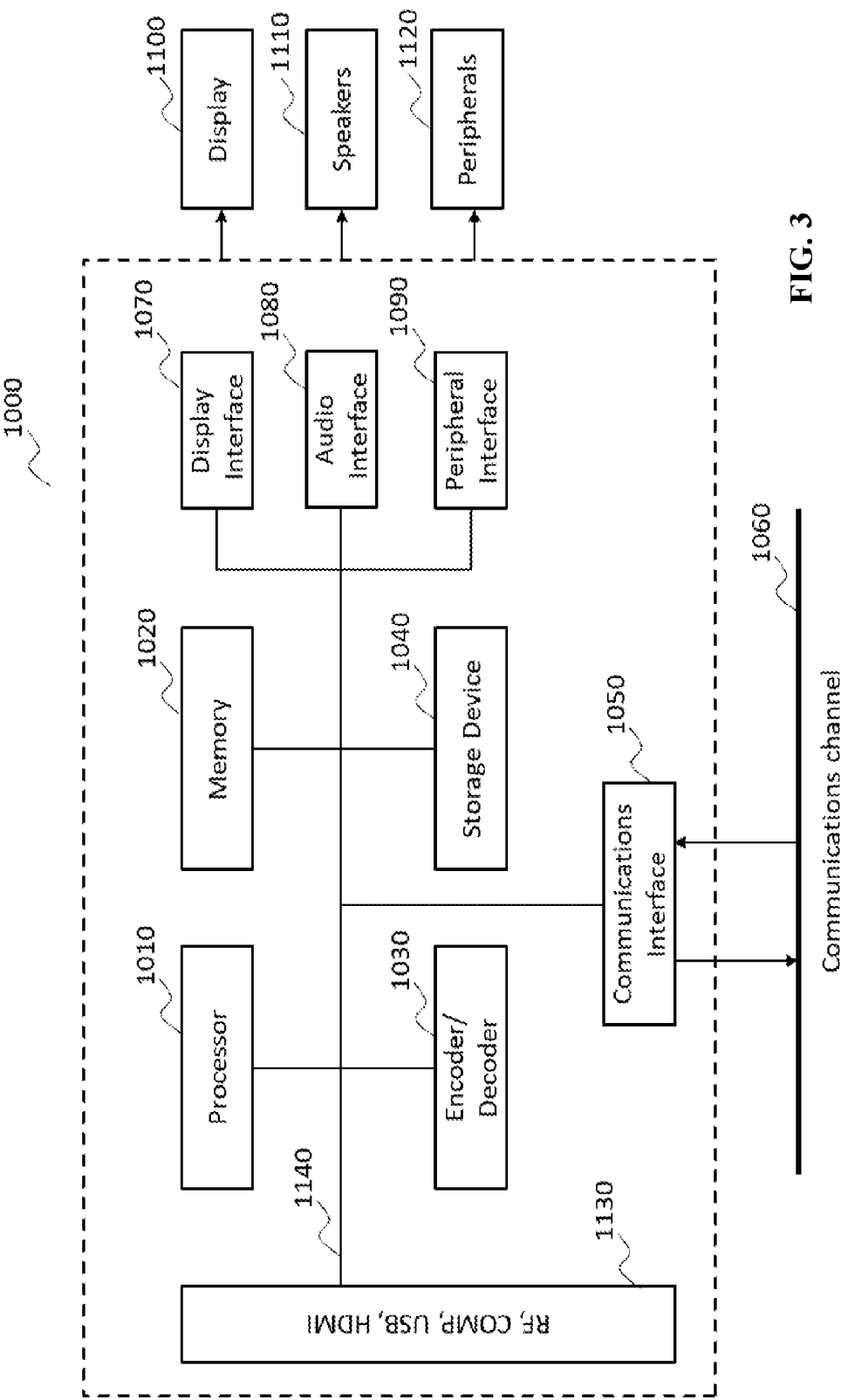


FIG. 3

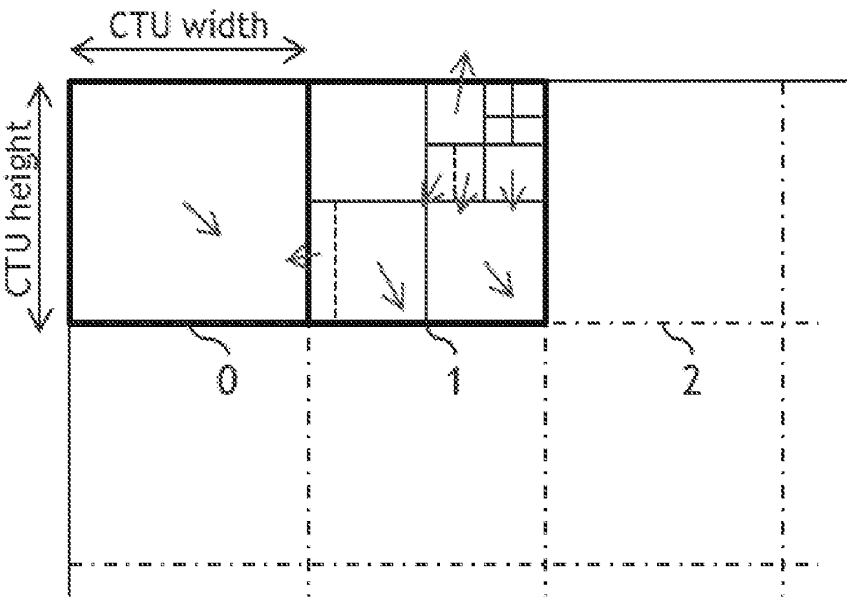


FIG. 4A

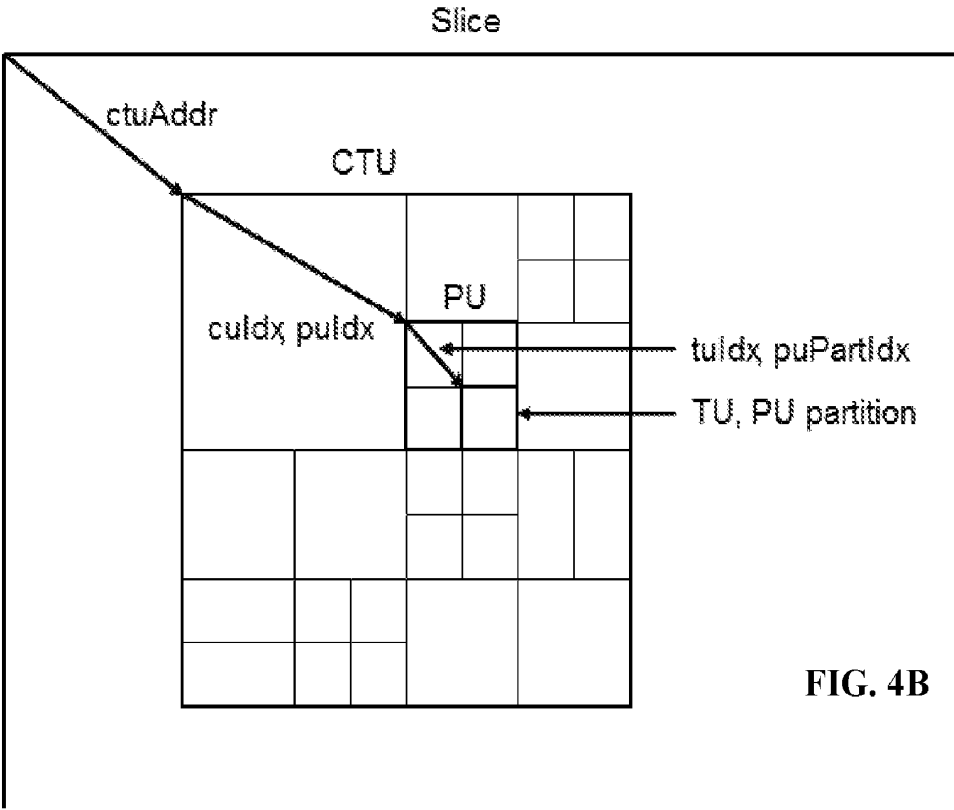


FIG. 4B

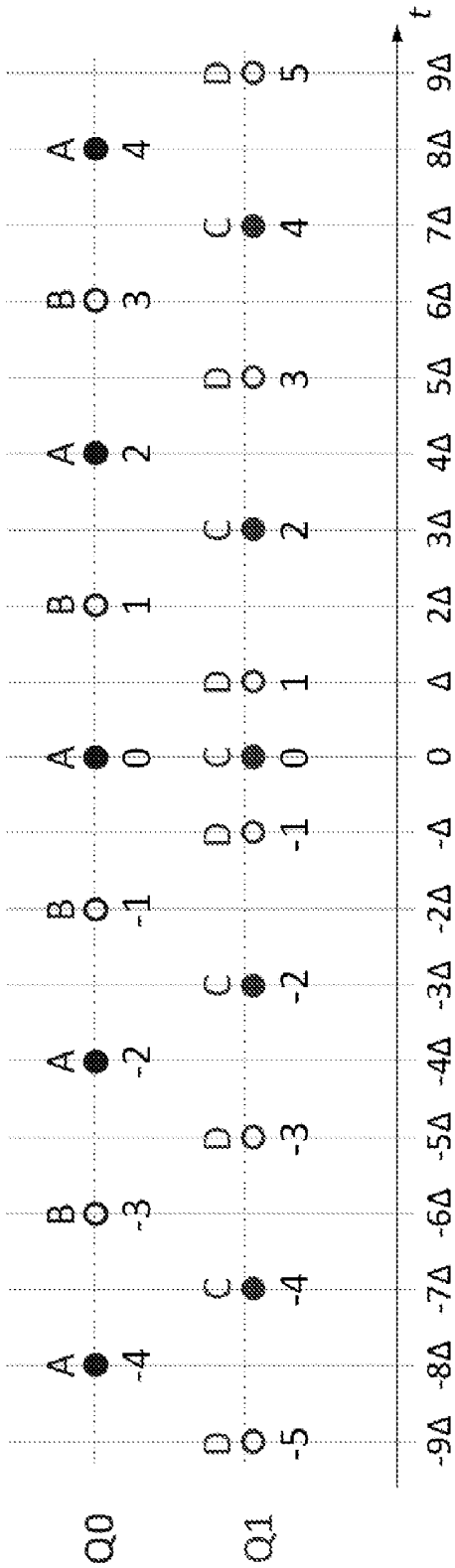


FIG. 5

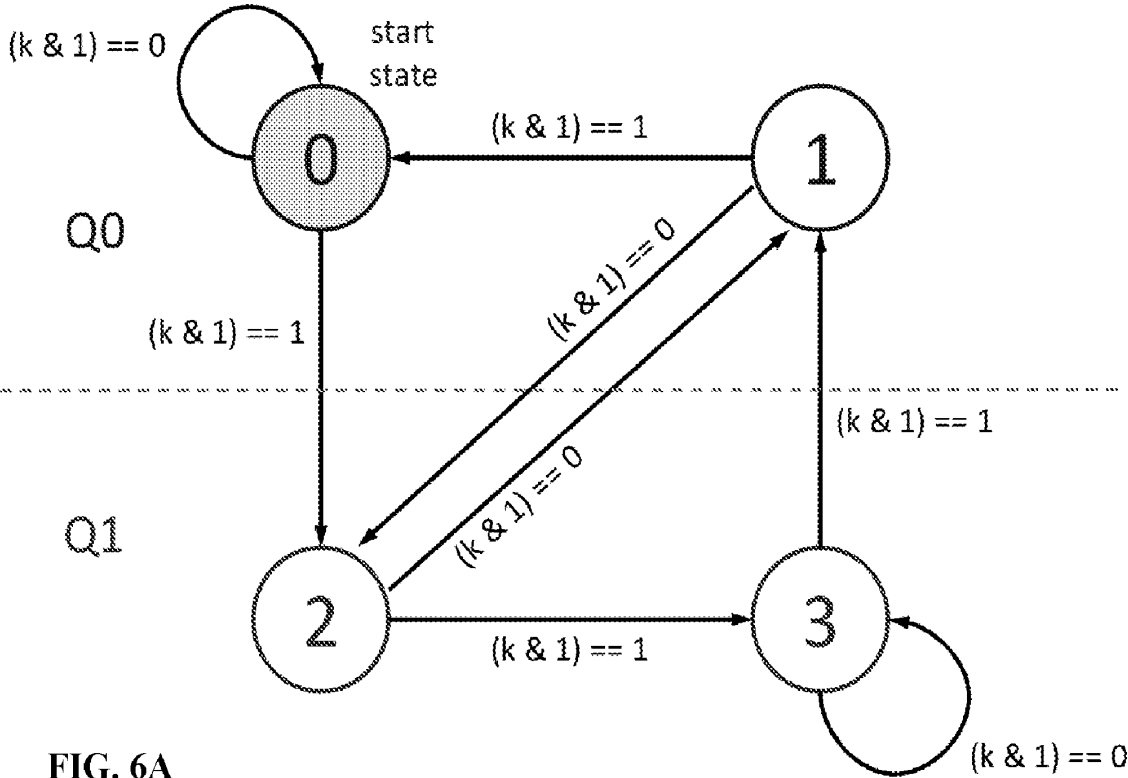


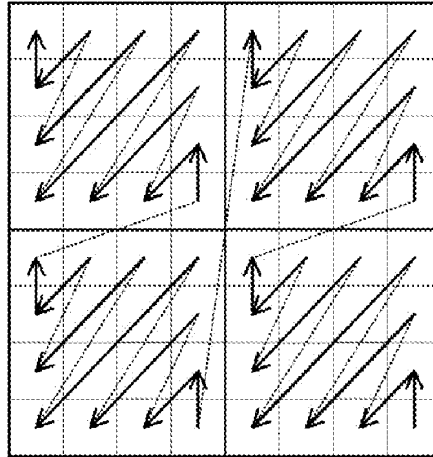
FIG. 6A

current state	next state for ...	
	$(k \& 1) == 0$	$(k \& 1) == 1$
0	0	2
1	2	0
2	1	3
3	3	1

FIG. 6B

7/26

FIG. 6C



```

residual_tb( ... )
{
    last_significant_coefficient_pos_x_prefix
    last_significant_coefficient_pos_y_prefix
    last_significant_coefficient_pos_x_suffix
    last_significant_coefficient_pos_y_prefix

    for( i=lastSubBlock ; i>=0; i-- ) {
        if( i < lastSubBlock ) {
            coded_sub_block_flag [ i ]
            if( codedSubBlockFlag [ i ] ) {
                residual_subblock ( ... )
            }
        }
    }
}

```

FIG. 7A

```
residual_subblock ( ... )
{
    numRegBins = is2x2CG ? 8 : 32

    for( k = firstPos ... ; remRegBins >= 4 ... ; k--) { // first pass
        sig_flag[ k ]      // ctx depends on prev. par_flag's
        remRegBins -- // decrement remRegBins
        if( sig_flag[ k ] ) {
            gt1_flag[ k ]
            if ( gt1_flag[k] ) {
                par_flag[ k ] // context-based coding (regular)
                remRegBins -- // decrement remRegBins
                gt3_flag[ k ] // context-based coding (regular)
                remRegBins -- // decrement remRegBins
            }
            coeff[k] = 1 + parFlag + gt1Flag + (gt3Flag << 1)
        }
        firstPosMode2 = k
        for( k=firstPos; k >firstPosMode2; k-- ) { // second pass (EP bins)
            if( coeff[k] >= 4 )
                remainder[ k ]
        }

        for ( k = firstPosMode2 ; k > lastPos ; k-- ) { // coeff bypass (EP bins)
            abs_level[ k ]
        }
        sign_data
    }
}
```

FIG. 7B

9/26

```
coding_unit( ... )
{
    cu_ckpt_flag
    If ( cu_skip_flag == 0 )
        cu_pred_mode_flag

    if( cu_skip_flag ==0 && cuPredMode != MODE_INTRA ){
        pred_mode_ibc_flag
    }

    If( cuPredMode == IMODE_NTRA ){
        intra_prediction_data
    }
    else { // MODE_INTER or MODE_IBC
        inter_prediction_data
    }

    If (!pcm_flag){
        if( cuPredMode != MODE_INTRA && merge_flag==0 )
            cu_cbf
            if ( cu_cbf ) {
                // SBT mode related signaling

                transform_tree( ... )
            }
    }
}
```

FIG. 8A

10/26

```
Transform_tree( ... )
{
    If ( IntraSubPartSize == NO_ISP_FLAG ){
        If ( IntraSubPartSize == NO_ISP_FLAG )
            if (width > MaxTbSizeX || height > MaxTbSizeY )
                quad-tree split into 4 sub-transform-trees
            }
        else {
            transform_unit( ... )
        }
    }
    Else if ( cu_sbt_fag ) {
        split into 2 transform-unit according to SBT syntax
        transform_unit( ... )
        transform_unit( ... )
    }
    else {
        split into 2 transform-unit according to ISP syntax
        transform_unit( ... )
        transform_unit( ... )
    }
}
```

FIG. 8B

11/26

```
transform_unit( ... )
{
    tu_cbf_luma
    tu_cbf_cb
    tu_cbf_cr

    if(tu_cbf_luma || tu_cbf_cb || tu_cbf_cr){
        cu_qp_delta_abs
        cu_qp_delta_sign_flag
    }

    if( width <= 32 && height <=32 ... ){
        transform_skip_flag
    }

    if( width <= 32 && height <= 32 && !transform_skip_flag){
        tu_mts_idx
    }

    if( tu_cbf_luma ){
        Residual_tb ( ... , compId=Y,..)
    }
    if( tu_cbf_cb ){
        Residual_tb ( ... , compId=Cb,..)
    }
    if( tu_cbf_cr ){
        Residual_tb ( ... , compId=Cr,..)
    }
}
```

FIG. 8C

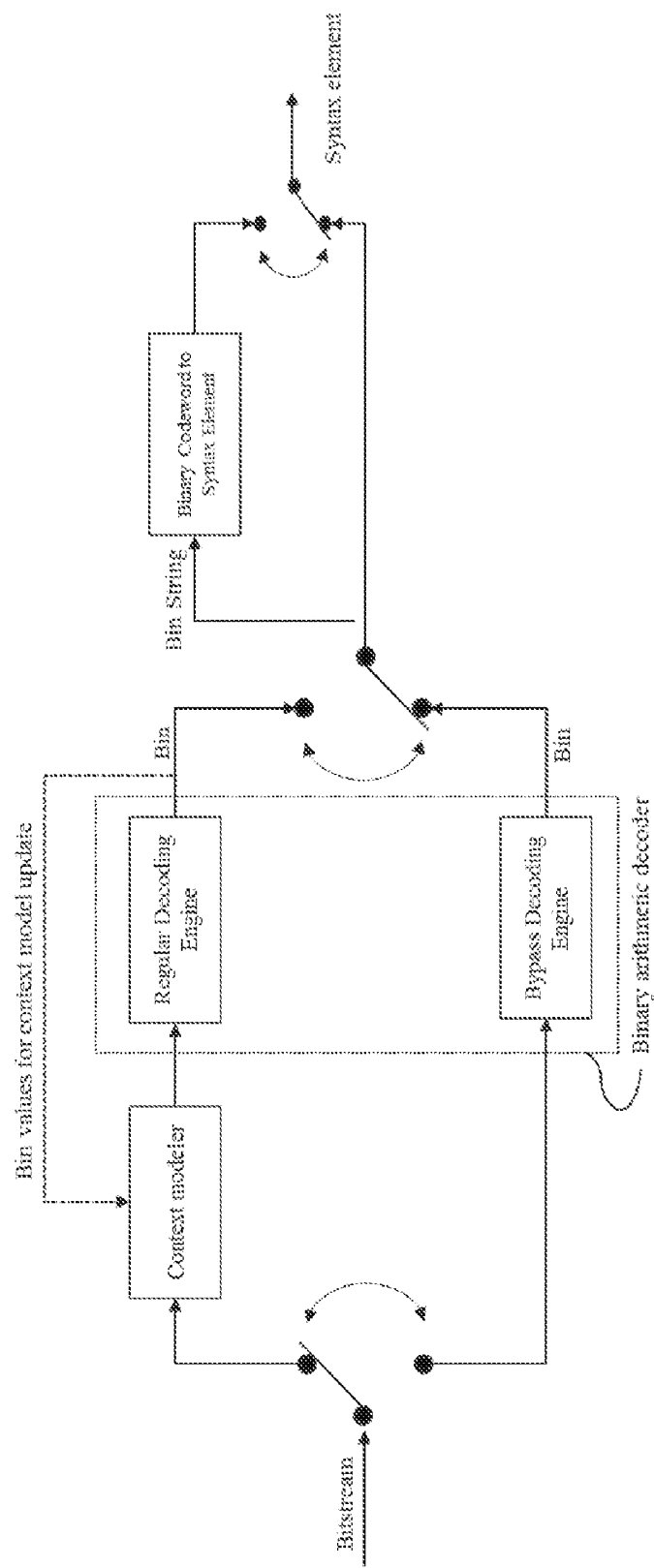


FIG. 9A

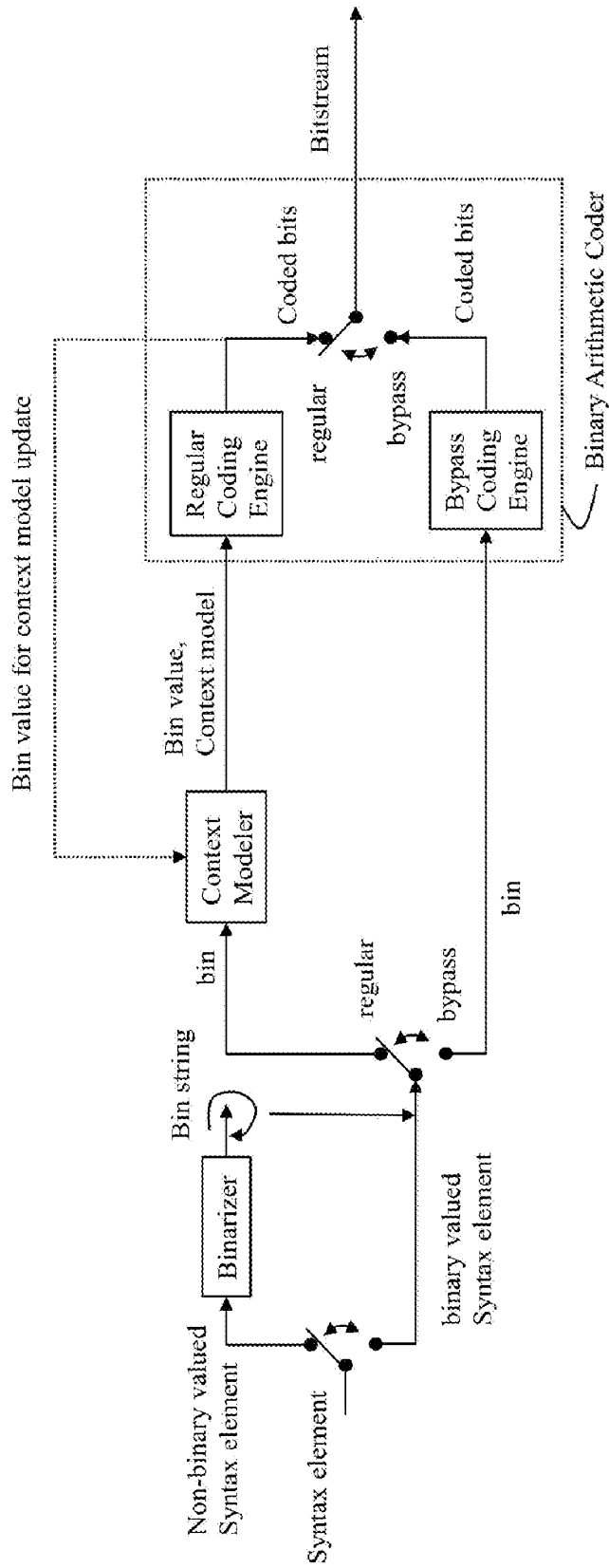


FIG. 9B

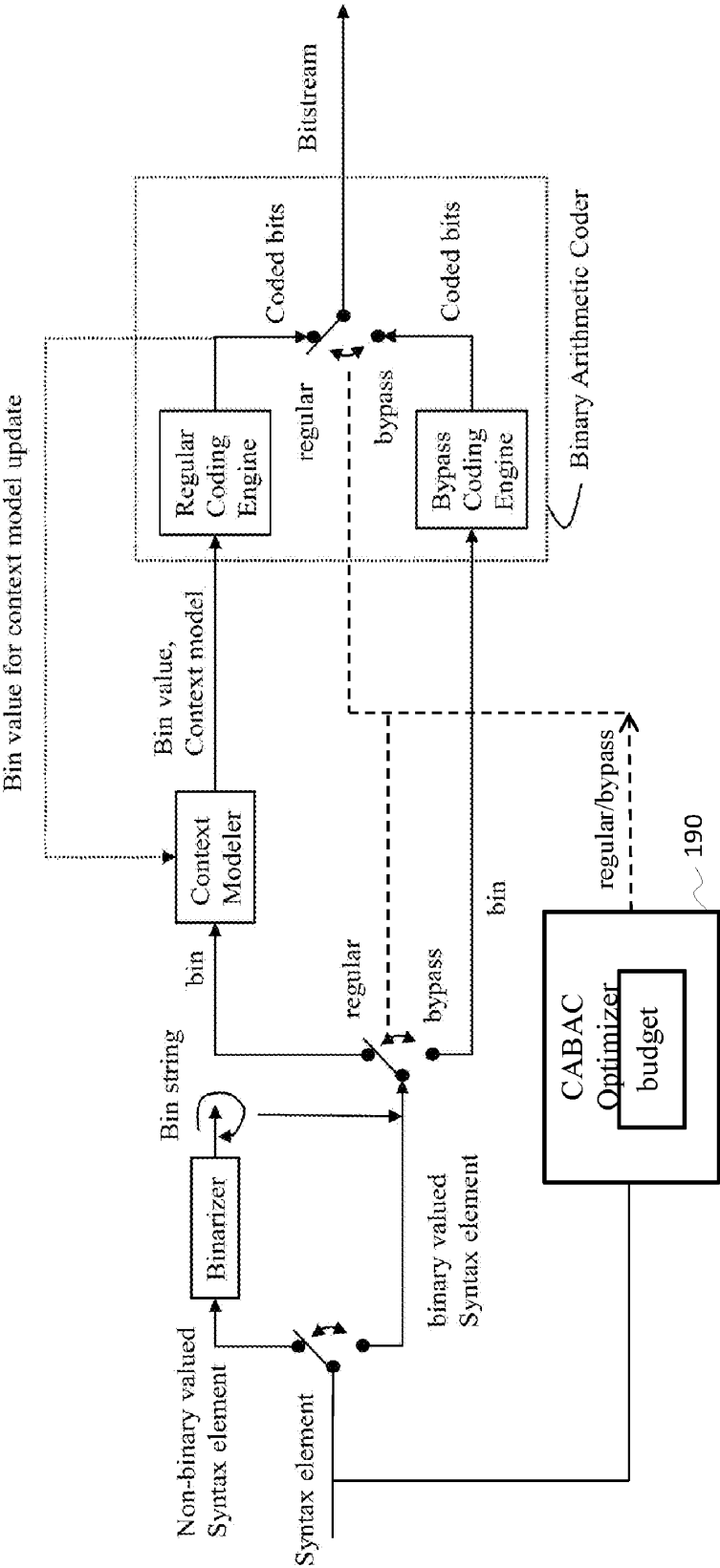


FIG. 10

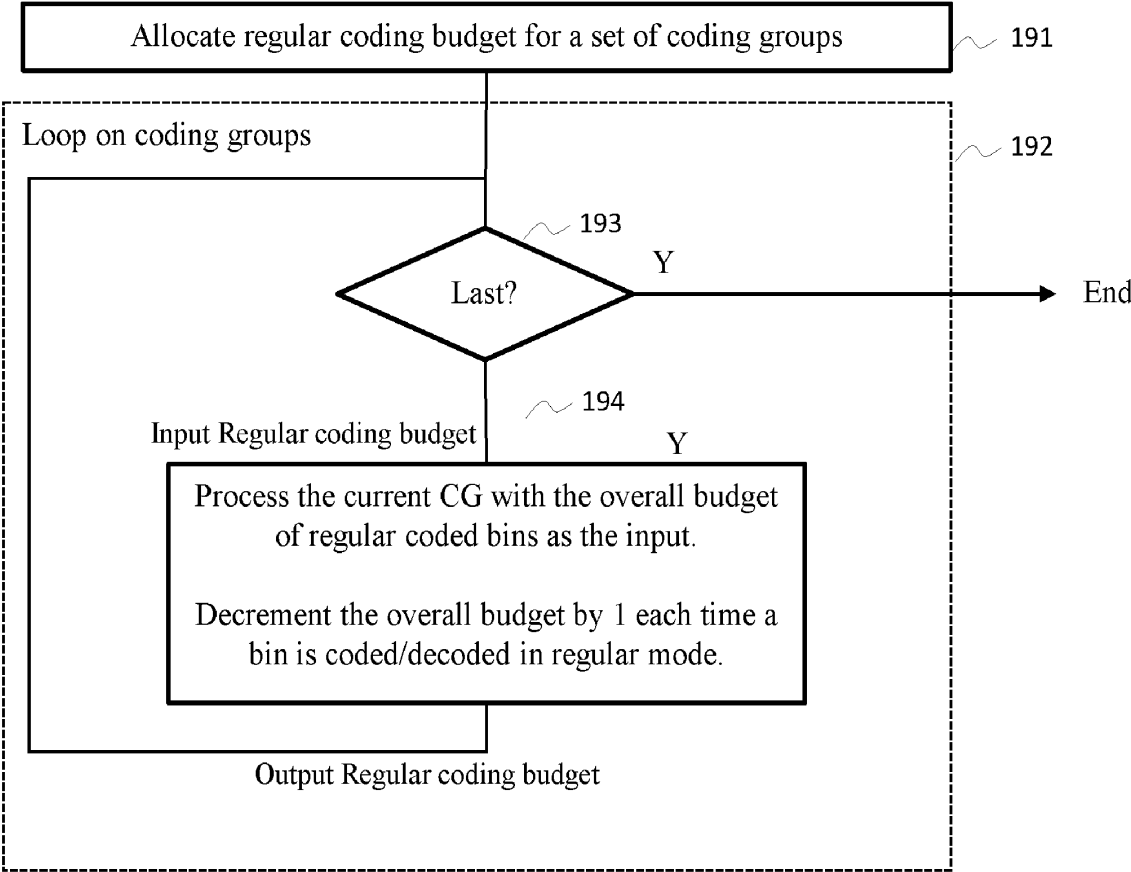


FIG. 11

```
residual_subblock ( numRegBins_in_out )
{
    remRegBins = numRegBins_in_out

    for( k = firstPos ... ; remRegBins >= 4 ... ; k--) { // first pass
        sig_flag[ k ]      // ctx depends on prev. par_flag's
        remRegBins -- // decrement remRegBins
        if( sig_flag[ k ] ) {
            gt1_flag[ k ]
            if ( gt1_flag[k] ) {
                par_flag[ k ]    // context-based coding (regular)
                remRegBins -- // decrement remRegBins
                gt3_flag[ k ]    // context-based coding (regular)
                remRegBins -- // decrement remRegBins
            }
            coeff[k] = 1 + parFlag + gt1Flag + (gt3Flag << 1)
        }
        firstPosMode2 = k
        for( k=firstPos; k >firstPosMode2; k-- ) { // second pass (EP bins)
            if( coeff[k] >= 4 )
                remainder[ k ]
        }

        for ( k = firstPosMode2 ; k > lastPos ; k-- ) { // coeff bypass (EP bins)
            abs_level[ k ]
        }
        sign_data
        numRegBins_in_out = remRegBins
    }
}
```

FIG. 12

```
residual_tb( ... )
{
    unitary_budget = 2;
    TB_level_remRegBins = unitary_budget * width*height

    last_significant_coefficient_pos_x_prefix
    last_significant_coefficient_pos_y_prefix
    last_significant_coefficient_pos_x_suffix
    last_significant_coefficient_pos_y_prefix

    for( i=lastSubBlock ; i>=0; i-- ) {
        if( i < lastSubBlock ) {
            coded_sub_block_flag [ i ]
            if( codedSubBlockFlag [ i ] ) {
                residual_subblock ( TB_level_remRegBins )
            }
        }
    }
}
```

FIG. 13A

```
residual_tb( ... )
{
    last_significant_coefficient_pos_x_prefix
    last_significant_coefficient_pos_y_prefix
    last_significant_coefficient_pos_x_suffix
    last_significant_coefficient_pos_y_prefix

    unitary_budget = 2;
    TB_level_remRegBins = unitary_budget * last_pos_x * last_pos_y

    for( i=lastSubBlock ; i>=0; i-- ) {
        if( i < lastSubBlock ) {
            coded_sub_block_flag [ i ]
            if( codedSubBlockFlag [ i ] ) {
                residual_subblock ( TB_level_remRegBins )
            }
        }
    }
}
```

FIG. 13B

```
transform_unit( ... )
{
    unitary_budget = 2; // average regular bins rate is 2 bins / sample
    TU_level_remRegBins = unitary_budget * width*height *3 / 2

    tu_cbf_luma
    tu_cbf_cb
    tu_cbf_cr

    if(tu_cbf_luma || tu_cbf_cb || tu_cbf_cr){
        cu_qp_delta_abs
        cu_qp_delta_sign_flag
    }

    if( width <= 32 && height <=32 ... ){
        transform_skip_flag
    }

    if( width <= 32 && height <= 32 && !transform_skip_flag){
        tu_mts_idx
    }

    if( tu_cbf_luma ){
        Residual_tb (TU_level_remRegBins, ... , compId=Y,..)
    }
    if( tu_cbf_cb ){
        Residual_tb ( TU_level_remRegBins, ... , compId=Cb,..)
    }
    if( tu_cbf_cr ){
        Residual_tb ( TU_level_remRegBins, ... , compId=Cr,..)
    }
}
```

FIG. 14A

```
transform_unit( ... )
{
    unitary_budget = 2; // average regular bins rate is 2 bins / sample
    TU_level_RegBins = unitary_budget * width*height *3 / 2

    tu_cbf_luma
    tu_cbf_cb
    tu_cbf_cr

    if(tu_cbf_luma || tu_cbf_cb || tu_cbf_cr){
        cu_qp_delta_abs
        cu_qp_delta_sign_flag
    }

    if( width <= 32 && height <=32 ... ){
        transform_skip_flag
    }

    if( width <= 32 && height <= 32 && !transform_skip_flag){
        tu_mts_idx
    }

    if( tu_cbf_luma ){
        bins_Y = TU_level_RegBins * 2 /3
        Residual_tb (bins_Y, ... , compId=Y,...)
    }
    if( tu_cbf_cb ){
        bins_Cb = (TU_level_RegBins - binsY ) /2
        Residual_tb ( bins_Cb, ... , compId=Cb,...)
    }
    if( tu_cbf_cr ){
        bins_Cr = TU_level_RegBins - binsY - binCb)
        Residual_tb ( bins_Cr, ... , compId=Cr,...)
    }
}
```

FIG. 14B

20/26

```
residual_tb ( numRegBins_in_out )
{
    last_significant_coefficient_pos_x_prefix
    last_significant_coefficient_pos_y_prefix
    last_significant_coefficient_pos_x_suffix
    last_significant_coefficient_pos_y_prefix

    for( i= lastSubBlock : i>=0; i-- ) {
        if( i < lastSubBlock ) {
            coded_sub_block_flag [ i ]
            if(codedSubBlockFlag [ i ] ) {
                residual_subblock (numRegBins_in_out )
            }
        }
    }
}
```

FIG. 14C

```
coding_unit( ... )
{
    unitary_budget = 2; // average regular bins rate is 2 bins / sample
    CU_level_remRegBins = unitary_budget *width*height *3 / 2

    cu_ckpt_flag
    If ( cu_skip_flag == 0 )
        cu_pred_mode_flag

    if( cu_skip_flag ==0 && cuPredMode != MODE_INTRA ){
        pred_mode_ibc_flag
    }

    If( cuPredMode == IMODE_NTRA ){
        intra_prediction_data
    }
    else { // MODE_INTER or MODE_IBC
        inter_prediction_data
    }

    If (lpcm_flag){
        if( cuPredMode != MODE_INTRA && merge_flag==0 )
            cu_cbf
            if ( cu_cbf ) {
                // SBT mode related signaling
            }

        transform_tree(CU_level_remRegBins , ... )
    }
}
```

FIG. 15A

```
Transform_tree(CU_level_remRegBins , ... )
{
    If ( IntraSubPartSize == NO_ISP_FLAG ){
        If ( IntraSubPartSize == NO_ISP_FLAG )
            if (width > MaxTbSizeX || height > MaxTbSizeY )
                quad-tree split into 4 sub-transform-trees
            }
        else {
            transform_unit(CU_level_remRegBins , ... )
        }
    }
    Else if ( cu_sbt_fag ) {
        split into 2 transform-unit according to SBT syntax
        transform_unit(CU_level_remRegBins , ... )
        transform_unit(CU_level_remRegBins , ... )
    }
    else {
        split into 2 transform-unit according to ISP syntax
        transform_unit(CU_level_remRegBins , ... )
        transform_unit(CU_level_remRegBins , ... )
    }
}
```

FIG. 15B

```
Transform_tree(CU_level_remRegBins , ... )
{
    If ( IntraSubPartSize == NO_ISP_FLAG ){
        If ( IntraSubPartSize == NO_ISP_FLAG )
            if (width > MaxTbSizeX || height > MaxTbSizeY )
                quad-tree split into 4 sub-transform-trees
            }
        else {
            transform_unit(CU_level_remRegBins , ... )
        }
    }
    Else if ( cu_sbt_fag ) {
        split into 2 transform-unit according to SBT syntax
        transform_unit(sbt_coded ? CU_level_remRegBins : 0, ... )
        transform_unit(sbt_coded ? CU_level_remRegBins : 0, ... )
    }
    else {
        split into 2 transform-unit according to ISP syntax
        regBins = CU_level_remRegBins/2
        transform_unit(regBins,... )
        transform_unit( (CU_level_remRegBins – regBins) , ... )
    }
}
```

FIG. 15C

```
transform_unit(TU_regBins , ... )
{
    tu_cbf_luma
    tu_cbf_cb
    tu_cbf_cr

    if(tu_cbf_luma || tu_cbf_cb || tu_cbf_cr){
        cu_qp_delta_abs
        cu_qp_delta_sign_flag
    }

    if( width <= 32 && height <=32 ... ){
        transform_skip_flag
    }

    if( width <= 32 && height <= 32 && !transform_skip_flag){
        tu_mts_idx
    }

    if( tu_cbf_luma ){
        Residual_tb (TU_regBins , ... , compId=Y,...)
    }
    if( tu_cbf_cb ){
        Residual_tb (TU_regBins , ... , compId=Cb,...)
    }
    if( tu_cbf_cr ){
        Residual_tb (TU_regBins , ... , compId=Cr,...)
    }
}
```

FIG. 15D

```
transform_unit(TU_RegBins , ... )
{
    tu_cbf_luma
    tu_cbf_cb
    tu_cbf_cr

    If(tu_cbf_luma || tu_cbf_cb || tu_cbf_cr){
        cu_qp_delta_abs
        cu_qp_delta_sign_flag
    }

    If( width <= 32 && height <=32 ... ){
        transform_skip_flag
    }

    If( width <= 32 && height <= 32 && !transform_skip_flag){
        tu_mts_idx
    }

    if( tu_cbf_luma ){
        bins_Y = TU_RegBins * 2 /3
        Residual_tb (bins_Y, ... , compId=Y,...)
    }
    if( tu_cbf_cb ){
        bins_Cb = (TU_RegBins - binsY ) /2
        Residual_tb ( bins_Cb, ... , compId=Cb,...)
    }
    if( tu_cbf_cr ){
        bins_Cr = TU_RegBins - binsY - binCb)
        Residual_tb ( bins_Cr, ... , compId=Cr,...)
    }
}
```

FIG. 15E

FIG. 16A

```
residual_subblock_all_bypass ()
{
    remRegBins = numRegBins_in_out

    K=firstPos
    firstPosMode2 = k

    for ( k = firstPosMode2 ; k > lastPos ; k-- ) {
        abs_level[ k ]
    }
    sign_data
    numRegBins_in_out = remRegBins
}
```

FIG. 16B

```
residual_subblock_all_regular ()
{
    for( k = firstPos ... ; remRegBins >= 4 ... ; k-- ) { // first pass
        sig_flag[ k ] // ctx depends on prev. par_flag's
        if( sig_flag[ k ] ) {
            gt1_flag[ k ]
            if ( gt1_flag[k] ) {
                par_flag[ k ] // context-based coding (regular)
                gt3_flag[ k ] // context-based coding (regular)
            }
            coeff[k] = 1 + parFlag + gt1Flag + (gt3Flag << 1)
        }
        firstPosMode2 = k
    }
    for( k=firstPos; k >lastPos; k-- ) { // second pass (EP bins)
        if( coeff[k] >= 4 )
            remainder[ k ]
    }

    sign_data
    numRegBins_in_out = remRegBins
}
```