Table 9-6 – Specification of intraModeTable[k][codeNum]

	codeNum																
k	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
0	0	15	11	10	13	7	9	4	14	2	3	6	8	5	12	1	
1	14	10	9	0	13	7	2	8	3	12	6	4	11	1	5		
2	2	0	29	30	20	1	21	28	15	7	16	8	11	31	22	19	32
3	2	1	28	0	29	20	27	19	51	21	7	14	10	11	30	31	18

	codeNum																
k	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**	**32**	**33**
2	10	27	14	4	18	12	23	17	24	26	5	25	5	9	3	13	
3	16	26	8	17	4	22	9	23	6	25	13	24	12	3	5		

9.2.2.6 Initialisation process for cbpSplitTransTable

Outputs of this process are initial values of the variable array cbpSplitTransTable and associated counters.

The values of the variable array cbpSplitTransTable[k][n] are initialized as specified in Table 9-7 where the values of k = 0...5 and n = 0,1. The number of counters used in each case for codeword index adaptive mapping is defined in Table 9-16. These counters are initialized according to subclause 9.2.2.1.

Table 9-7 – Specification of initial values of cbpSplitTransTable[flagPattern-8][n][codeNum]

	k (n = 0)						k (n = 1)					
codeNum	**0**	**1**	**2**	**3**	**4**	**5**	**0**	**1**	**2**	**3**	**4**	**5**
0	14	2	3	6	4	4	0	2	0	0	0	0
1	13	3	1	4	7	7	1	3	2	5	4	4
2	12	1	0	5	0	0	3	0	1	4	5	5
3	10	0	2	7	5	5	7		3	7	6	6
4	6			0	6	6	2			6	1	1
5	11			1	1	1	4			2	7	7
6	9			2	2	2	5			3	2	2
7	8			3	3	3	8			1	3	3
8	5						9					
9	4						11					
10	2						6					
11	7						10					
12	3						12					
13	1						13					
14	0						14					

9.2.3 Codeword index mapping update process

Inputs to this process are a VLC codeword index mapping table indexMappingTable that provides the mapping between VLC codeword indexes and syntax element values, a codeword index value codeNum, and a value counterNum that specifies the number of counters used. If counterNum is not equal to 0, inputs additionally include an array counterArray indexed from 0 to (counterNum-1) that saves the value of counters, a value counterSum that saves the number of all counter increments since its previous reset.

Outputs of this process are indexMappingTable, and if counterNum is not equal to 0 counterArray and counterSum with updated values.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000426

The process is specified in the following ordered steps:

– Syntax element value syntaxVal is set equal to indexMappingTable [codeNum].

– If codeNum is not less than 1, codeNumPre is set equal to (codeNum-1). Otherwise, codeNumPre is set equal to codeNum. Syntax element value syntaxValPre is set equal to indexMappingTable [codeNumPre].

– If codeNum is greater or equal to counterNum, the values of indexMappingTable[codeNumPre] and indexMappingTable[codeNum] are exchanged with each other. No further step is carried out. Otherwise

  – The value of counterArray[codeNum] is increased by 1.

  – If counterArray[codeNum] >= counterArray[codeNumPre], the counter values of counterArray[codeNum] and counterArray[codeNumPre] are swapped, and the values of indexMappingTable [codeNumPre] and indexMappingTable [codeNum] are also swapped.

  – If counterSum is not less than 15, the value of counterSum is reset to 0. For i = 0…(counterNum-1), counterArray[i] is set equal to (counterArray[i]>>1).

  – Otherwise, the value of counterSum is increased by 1.

9.2.4 Parsing process for transform coefficient syntax elements

This process is invoked when parsing syntax elements with descriptor equal to ce(v) in subclause 7.3.11 and 7.3.12 and when entropy_coding_mode_flag is equal to 0.

Inputs to this process are bits from RBSP, and the variables log2TrafoSize and cIdx, and the variable arrays lastPosVlcNumIndex and lastPosTable.

Outputs of this process are CAVLC syntax elements last_pos_level_one, level_minus2_and_sign, run_level_one, and level, and the variable arrays lastPosTable and lastPosVlcNumIndex.

9.2.4.1 Parsing process for last_pos_level_one

Inputs to this process are bits from RBSP and the variable arrays lastPosVlcNumIndex and lastPosTable.

Output of this process is the syntax element last_pos_level_one, the variables trOne and vlcNumLevel, and the variable arrays lastPosVlcNumIndex and lastPosTable.

The parsing process for last_pos_level_one is specified as follows.

– The variable N is set equal to (1 << log2TrafoSize) >> (cIdx > 0 ? 2 : 0).

– If N is greater than 4, last_pos_level_one is derived in the following ordered steps:

1. The variable blockType is derived as

$$blockType = (cIdx == 0\ ?\ (PredMode==MODE\_INTRA\ ?\ 0 : slice\_type + 1) + (N > 8\ ?\ 5 : 2) : cIdx - 1) \quad (9\text{-}23)$$

2. The variable vlcNumIndex is derived as

$$vlcNumIdx = Min(16, lastPosVlcNumIndex[blockType]) \quad (9\text{-}24)$$

3. The variable vlcNum is derived as

$$vlcNum = lastPosVlcNumTable[blockType][vlcNumIdx] \quad (9\text{-}25)$$

   The values of the array lastPosVlcNumTable are shown in Table 9-8.

4. The parsing process described in subclause 9.2.1 is invoked with vlcNum as input and the variable codeNum as output.

5. The variable lastPos is derived as follows

```
if (codeNum <= N+N*N)
    if (codeNum != 0 && (codeNum & (N-1)) = = 0)
        lastPos = (codeNum >> Log2(N)) – 1                    (9-26)
    else
        lastPos = codeNum – (codeNum >> Log2(N))
else
    lastPos = codeNum – N*N
```

6. The variable levelGreaterThanOneFlag is derived as

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000427

levelGreaterThanOneFlag = (codeNum > (N + N*N)) | | (codeNum != 0 && (codeNum & (N-1))) (9-27)

7. The syntax element last_pos_level_one is derived as

last_pos_level_one = levelGreaterThanOneFlag ? codeNum + N : codeNum (9-28)

8. The variable array lastPosVlcNumIndex[blockType] is updated as follows.

```
if ((N = = 8 ? codeNum : codeNum>>2) < lastPosVlcNumIndex[blockType])
    lastPosVlcNumIndex[blockType] -= 1
else if ((N = = 8 ? codeNum : codeNum>>2) > lastPosVlcNumIndex[blockType])
    lastPosVlcNumIndex[blockType] += 1
```
(9-29)

– Otherwise, (N equal to 4), last_pos_level_one is derived in the following ordered steps:

1. The parsing process described in subclause 9.2.1 is invoked with 2 as input and the variable codeNum as output.
2. The variable tableNum is derived as

tableNum = (PredMode==MODE_INTRA | | cIdx > 0) ? 0 : slice_type + 1 (9-30)

3. The syntax element last_pos_level_one is derived as

last_pos_level_one = lastPosTable[tableNum][codeNum] (9-31)

4. The variable array lastPosTable[tableNum] is updated by invoking process 9.2.3 with lastPosTable[tableNum], codeNum and 0 as inputs.

– The variable trOne is derived as

trOne = ! levelGreaterThanOneFlag (9-32)

– The variable vlcNumLevel is set equal to 0

Table 9-88 – Specification of lastPosVlcNumTable[blockType][vlcNumIdx]

	vlcNumIdx															
blockType	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
0	10	10	10	10	2	2	2	7	9	9	9	9	9	4	4	4	4
1	10	10	10	10	10	2	9	9	9	9	9	9	9	4	4	4	4
2	2	2	2	2	2	7	7	7	7	7	7	7	7	7	4	4	13
3	2	2	2	2	7	7	7	7	7	7	7	7	7	7	7	7	13
4	2	2	2	2	7	7	7	7	7	7	7	7	7	7	7	7	13
5	10	10	10	4	4	4	4	12	12	12	12	12	12	12	12	12	12
6	10	10	10	10	4	4	12	12	12	12	12	12	12	12	12	12	12
7	10	10	10	10	4	4	12	12	12	12	12	12	12	12	12	12	12

9.2.4.2 Parsing process for level_minus2_and_sign

Input to this process are bits from slice data.

Output of this process is level_minus2_and_sign.

The syntax element level_minus2_and sign is derived by invoking the parsing process of subclause 9.2.1 with 0 as input and level_minus2_and sign as output.

9.2.4.3 Parsing process for run_level_one

Inputs to this process are bits from slice data, the scan position n of the previous non-zero transform coefficient in inverse scan order, and the variable TrOne.

Outputs of this process are the syntax element run_level_one and the variable TrOne.

The value of run_level_one is derived as follows.

– The variable N is set equal to (1 << log2TrafoSize) >> (cIdx > 0 ? 2 : 0).

– The variable blockType is derived as

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000428

blockType = (cIdx == 0 ? (PredMode==MODE_INTRA ? 0 : slice_type + 1) + (N > 8 ? 5 : 2) : cIdx – 1) (9-33)

– The variable tableIdx is derived as

tableIdx = (blockType = = 2 | | blockType = = 5) ? (N <= 8 ? 0 : 1) : (N <= 8 ? 2 : 3) (9-34)

– The variable maxRun is set equal to n.

– The variable maxRunIdx is set equal to Min(28, maxRun).

– The variable vlcNum is derived from tableIdx and maxRunIdx as shown in Table 9-9.

– The parsing process described in subclause 9.2.1 is invoked with vlcNum as input and the variable codeNum as output.If blockType is equal to 2 or blockType is equal to 5, the value of run_level_one is derived as follows.

  – If N is equal to 4, the variable largeOnePos is derived from trOne and maxRunIdx as shown in Table 9-10.

  – Otherwise (N is greater than 4), the variable largeOnePos is derived from trOne and maxRunIdx as shown in Table 9-11.

  – The variables levelGreaterThanOneFlag and runOfZeros are derived as follows.

```
if ( largeOnePos > 0 ) {
    if( codeNum < min(largeOnePos, maxRunIdx + 2 ) ) {
        levelGreaterThanOneFlag = 0
        runOfZeros = codeNum
    } else if( codeNum < (maxRunIdx << 1 ) + 4 – largeOnePos ) {
        if(( codeNum + largeOnePos ) & 1 ) {
            levelGreaterThanOneFlag = 0
            runOfZeros = (codeNum + largeOnePos - 1) >> 1
        } else {
            levelGreaterThanOneFlag = 1
            runOfZeros = (codeNum - largeOnePos)>>1                    (9-35)
        }
    } else {
        levelGreaterThanOneFlag = 1
        runOfZeros = codeNum - maxRunIdx – 2
    }
} else {
    if( codeNum & 1) {
        levelGreaterThanOneFlag = 0
        runOfZeros = (codeNum - 1)>>1
    } else {
        runOfZeros = codeNum >> 1
        levelGreaterThanOneFlag = ((codeNum >> 1) <= maxRunIdx)?1:0
    }
}
```

  – The syntax element run_level_one is derived as

run_level_one = levelGreaterThanOneFlag ? runOfZeros + maxRun : runOfZeros (9-36)

– Otherwise (blockType is not equal to 2 and blockType is not equal to 5), the value of run_level_one is derived as follows.

  – The variable levelGreaterThanOneFlag is derived as

levelGreaterThanOneFlag = codeNum <= maxRun + 1 ? 0 : 1 (9-37)

  – If maxRun is less than 28, the variable runOfZeros is derived from maxRun and codeNum as shown in Table 9-12.

  – Otherwise (maxRun is not less than 28), the variable runOfZeros is derived as

runOfZeros = codeNum <= maxRun + 1 ? codeNum : codeNum – maxRun – 2 (9-38)

  – The syntax element run_level_one is derived as

run_level_one = levelGreaterThanOneFlag ? runOfZeros + n : runOfZeros (9-39)

– The variable trOne is derived as

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000429

trOne = (trOne = = 0 | | levelGreaterThanOneFlag) ? 0 : Max(4, trOne + 1) (9-40)

Table 9-90–93 – Derivation of vlcNum from tableIdx and maxRunIdx

	maxRunIdx														
tableIdx	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
0	8	0	0	5	5	5	5	5	5	5	6	6	6	6	6
1	8	0	0	5	5	5	5	5	5	5	6	6	6	6	6
2	8	0	0	0	0	5	5	5	5	5	5	5	5	5	5
3	8	0	1	1	1	1	2	2	2	2	2	2	2	6	6

	maxRunIdx														
tableIdx	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**
0	6	6	6	6	6	6	6	6	6	6	6	6	6	3	6
1	6	6	6	6	6	6	6	6	6	6	6	6	6	3	6
2	5	5	5	1	1	1	1	1	1	1	1	1	1	2	5
3	6	6	6	6	3	3	3	3	3	3	3	3	3	3	6

Table 9-104–108 – Derivation of largeOnePos from tableIdx and maxRunIdx when N is equal to 4

	maxRunIdx														
trOne	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0
1	2	3	4	5	6	5	6	7	7	7	7	7	6	4	2
2	2	2	2	3	3	3	3	4	4	4	3	3	3	2	NA
3	2	1	1	2	2	2	2	2	2	2	2	2	1	NA	NA
4	1	1	1	1	1	1	1	1	1	1	1	1	NA	NA	NA

Table 9-114–118 – Derivation of largeOnePos from tableIdx and maxRunIdx when N is greater than 4

	maxRunIdx														
trOne	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
0	0	1	2	2	2	2	2	4	5	5	6	6	6	6	6
1	2	4	4	6	6	8	8	10	11	13	15	13	14	15	16
2	2	3	4	5	4	5	6	6	7	8	8	9	9	10	10
3	2	1	2	2	3	3	4	4	5	5	5	6	6	7	7
4	2	1	2	2	2	2	3	3	3	3	4	4	4	4	4

	maxRunIdx													
trOne	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**
0	6	7	8	7	8	8	9	10	10	10	12	10	9	8
1	18	18	21	20	21	22	23	25	25	26	27	28	29	27
2	11	11	12	13	14	15	16	16	17	18	19	20	19	19
3	7	8	9	9	10	10	11	11	12	12	13	13	13	13
4	5	5	5	5	5	6	6	6	6	6	6	7	7	8

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000430

Table 9-[illegible] – Derivation of runOfZeros from maxRun and codeNum

	codeNum%(maxRun+1)														
maxRun	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
0	1	0	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
1	2	1	0	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
2	3	0	2	1	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
3	4	1	0	2	3	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
4	5	0	3	2	4	1	NA	NA	NA	NA	NA	NA	NA	NA	NA
5	6	0	1	4	5	3	2	NA	NA	NA	NA	NA	NA	NA	NA
6	7	1	0	2	4	3	6	5	NA	NA	NA	NA	NA	NA	NA
7	8	0	3	4	2	1	5	7	6	NA	NA	NA	NA	NA	NA
8	9	0	5	1	6	4	8	3	2	7	NA	NA	NA	NA	NA
9	10	0	1	6	7	2	9	5	4	3	8	NA	NA	NA	NA
10	11	1	0	2	3	6	7	4	5	8	10	9	NA	NA	NA
11	12	0	3	2	1	4	6	5	7	11	8	9	10	NA	NA
12	13	0	1	5	6	4	2	3	7	8	12	10	11	9	NA
13	14	0	1	7	2	8	6	5	3	4	13	12	9	11	10
14	15	0	1	2	8	3	9	7	6	4	5	14	10	13	12
15	16	0	1	2	3	4	8	7	5	9	6	10	15	11	13
16	17	0	3	1	2	4	5	8	9	7	6	10	11	12	16
17	18	0	1	5	2	6	4	3	7	8	9	10	17	13	14
18	19	0	1	7	8	2	6	3	9	4	5	10	18	15	16
19	20	0	9	1	10	2	8	3	7	4	16	17	6	19	5
20	21	0	1	10	2	11	3	9	5	4	17	20	18	8	7
21	22	1	0	2	3	4	10	11	5	6	9	7	12	8	13
22	23	0	3	4	2	1	5	7	6	10	11	8	9	12	13
23	24	0	5	6	1	4	3	7	2	8	10	11	9	12	15
24	25	0	7	1	8	6	5	2	9	10	4	11	3	12	17
25	26	0	1	9	10	2	8	11	7	3	6	5	12	4	19
26	27	0	1	11	2	10	12	3	9	7	8	4	6	20	13
27	28	0	1	2	12	3	4	11	13	9	5	8	10	7	6

	codeNum%(maxRun+1)													
maxRun	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**
0	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
1	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
2	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
3	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
4	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
5	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
6	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
7	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
8	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
9	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
10	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
11	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000431

12	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
13	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
14	11	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
15	14	12	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
16	13	14	15	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
17	12	15	11	16	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
18	14	11	13	17	12	NA	NA	NA	NA	NA	NA	NA	NA	NA
19	11	15	12	14	13	18	NA	NA	NA	NA	NA	NA	NA	NA
20	6	12	16	13	15	19	14	NA	NA	NA	NA	NA	NA	NA
21	18	21	17	19	14	20	16	15	NA	NA	NA	NA	NA	NA
22	14	22	18	15	19	20	17	16	21	NA	NA	NA	NA	NA
23	14	13	23	16	19	17	18	21	20	22	NA	NA	NA	NA
24	16	13	24	18	19	15	20	23	14	21	22	NA	NA	NA
25	13	18	20	14	17	25	21	16	23	24	15	22	NA	NA
26	5	14	26	21	19	18	25	15	16	17	24	22	23	NA
27	21	14	27	15	22	19	16	20	17	26	18	25	23	24

9.2.4.4 Parsing process for level

Inputs to this process are bits from slice data and the variable vlcNumLevel.

Outputs of this process are the syntax element level and the variable vlcNumLevel.

– The syntax element level is determined by invoking the parsing process described in subclause 9.2.1 with vlcNumLevel as input and level as output.

– The variable vlcNumLevel derived in the following ordered steps updated as follows.

1. The variable vlcIdx is set equal to vlcIdx = Min(3,vlcNumLevel)
2. The variable levelThreshold is derived from vlcIdx as shown in Table 9-13.
3. The variable vlcNumLevel is derived as

$$vlcNumLevel = vlcNumLevel + (level > levelThreshold \;\&\&\; vlcNumLevel < 4 \;?\; 1 : 0) \quad (9\text{-}41)$$

Table 9-138-139-13 – Derivation of levelThreshold from vlcIdx

vlcIdx	levelThreshold
0	4
1	6
2	14
3	28

9.2.4.5 Parsing process for cu_split_pred_part_mode

This process is invoked when entropy_coding_mode_flag is equal to 0 for parsing syntax element **cu_split_pred_part_mode** in subclause 7.3.5.

Inputs to this process are bits from slice data, splitPredPartModeTable.

Output of this process is the syntax element cu_split_pred_part_mode.

The value of cu_split_pred_part_mode is derived as follows.

– If log2CUSize is greater than Log2MinCUSize, the parsing process described in subclause 9.2.1 is invoked with vlcNum of 14 and cMax of 6 as inputs and the variable codeNum as output.

– Otherwise, the parsing process described in subclause 9.2.1 is invoked with vlcNum of 14 and cMax of 5 as inputs and the variable codeNum as output.

– The syntax element cu_split_pred_part_mode is derived as

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000432

cu_split_pred_part_mode = splitPredPartModeTable[Log2MaxCUSize – log2CUSize][codeNum] (9-42~~44~~)

– If log2CUSize is equal to Log2MinCUSize and cu_split_part_mode has a value of 5, PredMode and PartMode are determined as follows

1. Read one bit and assign its value to a variable b
2. If b is equal to 1, set PredMode to MODE_INTRA and PartMode to PART_2Nx2N
3. Otherwise, read the next one bit and assign its value to the variable b.
   – if b is equal to 1, set PredMode to MODE_INTRA and PartMode to PART_NxN
   – Otherwise, set PredMode to MODE_INTER and PartMode to PART_NxN

– The variable array splitPredPartModeTable[Log2MaxCUSize – log2CUSize] is updated by invoking process 9.2.3 with splitPredPartModeTable[Log2MaxCUSize – log2CUSize], codeNum and a value of 4 as inputs.

9.2.4.6 Parsing process for rem_intra_luma_pred_mode

This process is invoked when entropy_coding_mode_flag is equal to 0 for parsing syntax element rem_intra_luma_pred_mode in subclause 7.3.7.

Inputs to this process are bits from slice data, a variable puSize specifying the size of the current prediction unit, NumMPMCand and a variable array intraModeTable.

Outputs of this process are the syntax element rem_intra_luma_pred_mode and value-updated intraModeTable.

The value of rem_intra_luma_pred_mode is derived as follows.

– Based on the input puSize, the value of a variable intraPredModeNum indicating the number of intra prediction modes for the given size of prediction unit is obtained according to Table 7-12~~Table 7-11~~.

– If intraPredModeNum is equal to 3, read one bit and assign its value to rem_intra_luma_pred_mode. No further step is carried out. Othewise,

– The values of variable k and vlcNum are obtained as follows
  – If intraPredModeNum is equal to 17
    – If NumMPMCand is equal to 1, set vlcNum equal to 16 and k equal to 0.
    – Otherwise, set vlcNum equal to 17 and k equal to 1.
  – If intraPredModeNum is equal to 34
    – If NumMPMCand is equal to 1, set vlcNum equal to 18 and k equal to 2.
    – Otherwise, set vlcNum equal to 19 and k equal to 3.
  – The parsing process described in subclause 9.2.1 is invoked with vlcNum as input and the variable codeNum as output.
  – The value of rem_intra_luma_pred_mode is set equal to intraModeTable[k][codeNum].
  – The value of a variable counterNum is set to 0. The variable array intraModeTable[k] is updated by invoking process in subclause 9.2.3 with intraModeTable[k], codeNum and counterNum as inputs.

9.2.4.7 Parsing process for cbp_and_split_transform

This process is invoked when entropy_coding_mode_flag is equal to 0 for parsing syntax element cbp_and_split_transform in subclause 7.3.8.

Inputs to this process are bits from slice data, a variable cbpVlcNumIdx, trafoDepth and a variable array cbpSplitTransTable.

Outputs of this process are the syntax element cbp_and_split_transform and value-updated cbpVlcNumIndex and cbpSplitTransTable.

The value of cbp_and_split_transform is derived as follows.

– Obtain a 4-bit value flagPattern accoding to subclause 7.4.8.

– Derive the value of a variable k according to Table 9-16.

– Set the value of a variable n as

n = (PredMode = = MODE_INTRA ? 0 : 1) (9-43)

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000433

– Obtain a value of vlcNum as follows
  – If flagPattern is equal to 8,

$$vlcNum = (n == 0 \ ? \ cbpVlcNumTable[cbpVlcNumIdx]:11) \quad (9\text{-}44)$$

  – Otherwise, if flagPattern is equal to 11, 13 or 15 and n is equal to 0, set vlcNum to 15;
  – Otherwise, set vlcNum to 14.
– If vlcNum is euqal to 14, obtain the value of a variable cMax according to Table 9-15.
– The parsing process described in subclause 9.2.1 is invoked with vlcNum and cMax as inputs and the variable codeNum as output.
– Set the value of cbp_and_split_transform equal to cbpSplitTransTable[k][n][codeNum]
– If flagPattern is not equal to 10 or 12, a variable counterNum is obtained according to Table 9-16 and the variable array cbpSplitTransTable[k][n] is updated by invoking process 9.2.3 with cbpSplitTransTable[k][n], codeNum and counterNum as inputs.
– If flagPattern is euqal to 8 and n is equal to 0, the value of cbpVlcNumIdx is updated as follows

```
if(codeNum < cbpVlcNumIdx)
    cbpVlcNumIdx -= 1                                   (9-45)
else if (codeNum > cbpVlcNumIdx)
    cbpVlcNumIdx += 1
```

– If flagPattern is euqal to 15, the exact values of Cb and Cr in Table 7-14Table 7-13 are determined as follows.
  – The parsing process described in subclause 9.2.1 is invoked with vlcNum of 14 and cMax of 2 as inputs and the variable codeNum as output.
  – Set the value of variable b as

$$b = n? \ (codeNum + 1) : (3 - codeNum) \quad (9\text{-}46)$$

  – The values of Cb and Cr are set equal to

$$Cb = (b >> 1) \quad (9\text{-}47)$$

$$Cr = (b \ \& \ 0x01) \quad (9\text{-}48)$$

Table 9-149-143 – Specification of cbpVlcNumTable[cbpVlcNumIdx]

cbpVlcNumIdx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
vlcNum	1	2	2	2	11	11	11	11	11	11	11	11	11	11	11

Table 9-150-144 – Derivation of cMax from flagPattern and n

	flagPattern			
n	9	10, 12	11,13,15	14
0	3	3	-	7
1	2	3	6	7

Table 9-151-145 – Specification of cbpSplitTransTable and counterNum associated

flagPattern	8	9	10, 12	11, 13, 15	14 (trafoDepth=0)	14 (trafoDepth>0)
k	0	1	2	3	4	5
counterNum	2	3	0	4	4	4

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000434

9.3 CABAC parsing process for slice data

[Ed. (TW) needs to be extended]

This process is invoked when parsing syntax elements with descriptor ae(v) in subclauses 7.3.4 and 7.3.5 when entropy_coding_mode_flag is equal to 1.

Inputs to this process are a request for a value of a syntax element and values of prior parsed syntax elements.

Output of this process is the value of the syntax element.

When starting the parsing of the slice data of a slice in subclause 7.3.4, the initialisation process of the CABAC parsing process is invoked as specified in subclause 9.3.1.

The parsing of syntax elements proceeds as follows:

For each requested value of a syntax element a binarization is derived as described in subclause 9.3.2.

The binarization for the syntax element and the sequence of parsed bins determines the decoding process flow as described in subclause 9.3.2.1~~9.3.3~~.

For each bin of the binarization of the syntax element, which is indexed by the variable binIdx, a context index ctxIdx is derived as specified in subclause 9.3.3.1.

For each ctxIdx the arithmetic decoding process is invoked as specified in subclause 9.3.3.2.

The resulting sequence ($b_0..b_{binIdx}$) of parsed bins is compared to the set of bin strings given by the binarization process after decoding of each bin. When the sequence matches a bin string in the given set, the corresponding value is assigned to the syntax element.

In case the request for a value of a syntax element is processed for the syntax element mb_type and the decoded value of mb_type is equal to I_PCM, the decoding engine is initialised after the decoding of any pcm_alignment_zero_bit and all pcm_sample_luma and pcm_sample_chroma data as specified in subclause 9.3.1.2.

9.3.1 Initialisation process

Outputs of this process are initialised CABAC internal variables.

The processes in subclauses 9.3.1.1 and 9.3.1.2 are invoked when starting the parsing of the slice data of a slice in subclause 7.3.4.

9.3.1.1 Initialisation process for context variables

Outputs of this process are the initialised CABAC context variables indexed by ctxIdx.

Table 9-18~~Tables 9-5~~ to Table 9-43~~9-36~~ contain the values of the variables n and m used in the initialisation of context variables that are assigned to all syntax elements in subclauses 7.3.4 to 7.3.10 except for the end-of-slice flag and those related to CAVLC.

For each context variable, the two variables pStateIdx and valMPS are initialised.

NOTE 1 – The variable pStateIdx corresponds to a probability state index and the variable valMPS corresponds to the value of the most probable symbol as further described in subclause 9.3.3.2.

The two values assigned to pStateIdx and valMPS for the initialisation are derived from $SliceQP_Y$, which is derived in Equation $SliceQP_Y = 26 + pic\_init\_qp\_minus26 + slice\_qp\_delta$(7-12~~7-9~~). Given the two table entries (m, n), the initialisation is specified by the following pseudo-code process:

```
preCtxState = Clip3( 1, 126, ( ( m * Clip3( 0, 51, SliceQPY ) ) >> 4 ) + n )
if( preCtxState <= 63 ) {
    pStateIdx = 63 − preCtxState
    valMPS = 0                                                    (9-4~~9-4~~)
} else {
    pStateIdx = preCtxState − 64
    valMPS = 1
}
```

In Table 9-17~~Table 9-4~~, the ctxIdxTable and the ctxIdx for which initialisation is needed for each of the slice types are listed. The referenced context index tables ctxIdxTable include the values of m and n needed for the initialisation.

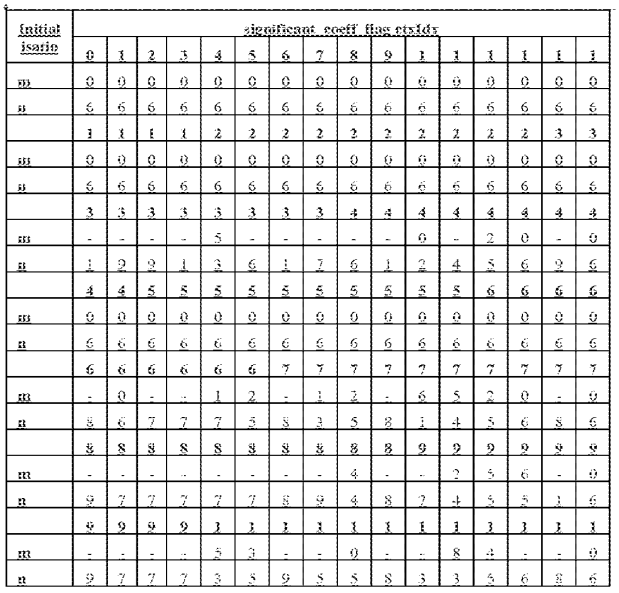
Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000435

Table 9-179-180-181 – Association of ctxIdx and syntax elements for each slice type in the initialisation process

	Syntax element	ctxIdxTable	Slice Type		
			I	P	B
slice_header()	alf_cu_flag	Table 9-187 table 9-5	0..2	3..5	6..8
coding_tree()	split_coding_unit_flag	Table 9-194 table 9-6	0...2	3..5	6..8
coding_unit()	skip_flag	Table 9-200 table 9-7		0..2	3..5
	cu_qp_delta	Table 9-214 table 9-8	0..3	4..7	8..11
	pred_type	Table 9-222 table 9-9	0	1..4	5..9
prediction_unit()	prev_intra_luma_pred_flag	Table 9-237 table 9-10	0	1	2
	rem_intra_luma_pred_mode	Table 9-243 table 9-11	0	1	2
	intra_chroma_pred_mode	Table 9-251 table 9-12	0..3	4..7	8..11
	merge_flag	Table 9-264 table 9-13		0..2	3..5
	merge_idx	Table 9-271 table 9-14		0..3	4..7
	inter_pred_flag	Table 9-284 table 9-15			0..2
	ref_idx_lc, ref_idx_l0, ref_idx_l1	Table 9-291 table 9-16		0..5	6..11
	mvd_l0[ ][ ][0]	Table 9-304 table 9-17		0..6	14..20
	mvd_lc[ ][ ][0], mvd_l1[ ][ ][0]	Table 9-304 table 9-17			14..20
	mvd_l0[ ][ ][1]	Table 9-304 table 9-17		7..13	21..27
	mvd_lc[ ][ ][1	Table			21..2

서식 있음: 글꼴: 8 pt

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000436

	], mvd_l1[ ][ ][1]	Table 9-47			7
	mvp_idx_lc, mvp_idx_l0, mvp_idx_l1	Table 9-48		0..1	2..3
transform_tree()	no_residual_data_flag	Table 9-49		0..3	4..7
	split_transform_flag	Table 9-50	0..3	4..7	8..11
	cbf_luma	Table 9-51	0..3	4..7	8..11
	cbf_cb	Table 9-52	0..3	4..7	8..11
	cbf_cr	Table 9-53	0..3	4..7	8..11
residual_coding( )	last_significant_coeff_x	Table 9-54	0..40	41..81	82..122
	last_significant_coeff_y	Table 9-55	0..40	41..81	82..122
	significant_coeff_flag	needs update	Table 9-39 Table 9-40		Table 9-41 Table 9-42

initialisatio	significant_coeff_flag ctxIdx															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
m	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
n	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
	1	1	1	1	2	2	2	2	2	2	2	2	2	2	3	3
m	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
n	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
	3	3	3	3	3	3	3	3	4	4	4	4	4	4	4	4
m	-	-	-	-	5	-	-	-	-	-	6	-	2	0	-	9
n	1	2	2	1	3	6	1	7	6	1	2	4	5	6	2	6
	4	4	5	5	5	5	5	5	5	5	5	5	6	6	6	6
m	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
n	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
	6	6	6	6	6	6	7	7	7	7	7	7	7	7	7	7
m	-	0	-	-	1	2	-	1	2	-	6	5	2	0	-	9
n	5	6	7	7	7	5	5	3	5	8	1	4	5	6	8	6
	8	8	8	8	8	8	8	8	8	8	9	9	9	9	9	9
m	-	-	-	-	-	-	-	-	4	-	-	2	5	6	-	9
n	9	7	7	7	7	7	8	9	4	8	7	4	5	5	1	6
	9	9	9	9	1	1	1	1	1	1	1	1	1	1	1	1
m	-	-	-	-	5	3	-	-	9	-	-	8	4	-	-	9
n	9	7	7	7	3	5	9	5	5	8	3	3	5	6	8	6

서식 있음: 글꼴: 8 pt

서식 있음: 글꼴: 8 pt

서식 있음: 글꼴: 8 pt

서식 있음: 글꼴: 8 pt

서식 있음: 글꼴: 8 pt

서식 있음: 글꼴: 8 pt

서식 있음: 글꼴: 8 pt

서식 있음: 글꼴: 8 pt

서식 있음: 글꼴: 8 pt, 강조

서식 있음: 글꼴: Times New Roman, 8 pt, 영어(영국), 강조

서식 있음: 글꼴: 8 pt, 강조

서식 있는 표

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000437

				Table 9-40 Table 9-33	
	coeff_abs_level_greater1_flag	Table 9-37 Table 9-29	0..79	80..159	160..239
	coeff_abs_level_greater2_flag	Table 9-37 Table 9-30	0..79	80..159	160..239

NOTE 2 – ctxIdxTable equal to 0 and ctxIdx equal to 0 are associated with the end_of_slice_flag. The decoding process specified in subclause 9.3.3.2.4 applies to ctxIdxTable equal to 0 and ctxIdx equal to 0. This decoding process, however, may also be implemented by using the decoding process specified in subclause 9.3.3.2.1. In this case, the initial values associated with ctxIdxTable equal to 0 and ctxIdx equal to 0 are specified to be pStateIdx = 63 and valMPS = 0, where pStateIdx = 63 represents a non-adapting probability state.

Table 9-188 – Values of variables m and n for alf_cu_flag ctxIdx

Initialisation variables	alf_cu_flag ctxIdx								
	0	1	2	3	4	5	6	7	8
m	0	0	0	0	0	0	0	0	0
n	64	64	64	64	64	64	64	64	64

Table 9-189 – Values of variables m and n for split_coding_unit_flag ctxIdx

Initialisation variables	split_coding_unit_flag ctxIdx								
	0	1	2	3	4	5	6	7	8
m	-7	-10	-10	-14	-6	-6	−14	−7	−10
n	68	87	105	71	73	91	71	74	92

Table 9-208 – Values of variables m and n for skip_flag ctxIdx

Initialisation variables	skip_flag ctxIdx					
	0	1	2	3	4	5
m	0	0	0	0	0	0
n	64	64	64	64	64	64

Table 9-210 – Values of variables m and n for cu_qp_delta ctxIdx

Initialisation variables	cu_qp_delta flag ctxIdx											
	0	1	2	3	4	5	6	7	8	9	10	11
m	0	0	0	0	0	0	0	0	0	0	0	0
n	64	64	64	64	64	64	64	64	64	64	64	64

서식 있음: 글꼴: Times New Roman, 8 pt, 영어(영국), 강조

서식 있음: 다음 단락과의 사이에 페이지 나누지 않음

서식 있음: 글꼴: 8 pt

서식 있음: 글꼴: 8 pt

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000438

Table 9-229-229-33 – Values of variables m and n for pred_type

Initialisation variables	pred_type ctxIdx									
	0	1	2	3	4	5	6	7	8	9
m	0	-25	-1	-3	6	6	-1	13	-11	-11
n	73	89	64	63	78	50	56	53	76	70

Table 9-239-239-33 – Values of variable m and n for prev_intra_luma_pred_flag ctxIdx

Initialisation variables	prev_intra_luma_pred_flag ctxIdx		
	0	1	2
m	2	0	0
n	54	50	51

Table 9-249-249-34 – Values of variable m and n for rem_intra_luma_pred_mode ctxIdx

Initialisation variables	rem_intra_luma_pred_mode ctxIdx		
	0	1	2
m	-3	-2	1
n	65	61	55

Table 9-259-259-35 – Values of variable m and n for intra_chroma_pred_mode ctxIdx

Initialisation variables	intra_chroma_pred_mode ctxIdx											
	0	1	2	3	4	5	6	7	8	9	10	11
m	0	0	0	0	0	0	0	0	0	0	0	0
n	64	64	64	64	64	64	64	64	64	64	64	64

Table 9-269-269-36 – Values of variable m and n for merge_flag ctxIdx

Initialisation variables	merge_flag ctxIdx					
	0	1	2	3	4	5
m	0	0	0	0	0	0
n	64	64	64	64	64	64

Table 9-279-279-37 – Values of variable m and n for merge_idx ctxIdx

Initialisation variables	merge_idx ctxIdx							
	0	1	2	3	4	5	6	7
m	0	0	0	0	1	6	-7	−4
n	64	64	64	64	65	42	75	72

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000439

Table 9-288-38[illegible] – Values of variable m and n for inter_pred_flag ctxIdx

Initialisation variables	inter_pred_flag ctxIdx		
	0	1	2
m	-2	-5	-9
n	58	70	85

Table 9-298-30[illegible] – Values of variable m and n for ref_idx_lc, ref_idx_l0, ref_idx_l1 ctxIdx

Initialisation variables	ref_idx_lc, ref_idx_l0, ref_idx_l1 ctxIdx											
	0	1	2	3	4	5	6	7	8	9	10	11
m	-6	-10	-8	-17	1	0	-9	-9	-9	-12	-18	0
n	59	75	75	96	59	64	55	71	76	86	55	64

Table 9-308-36[illegible] – Values of variable m and n for mvd_lc, mvd_l0, mvd_l1 ctxIdx

Initialisation variables	mvd_lc, mvd_l0, mvd_l1 ctxIdx													
	0	1	2	3	4	5	6	7	8	9	10	11	12	13
m	5	-2	-7	14	16	6	9	2	-3	-10	14	17	6	8
n	61	79	89	39	49	60	71	63	78	94	37	46	60	73
	14	15	16	17	18	19	20	21	22	23	24	25	26	27
m	1	-4	-10	11	14	4	7	1	-4	-10	11	15	5	8
n	67	83	97	49	56	69	77	65	81	96	48	52	67	76

Table 9-318-31[illegible] – Values of variable m and n for mvp_idx_lc, mvp_idx_l0, mvp_idx_l1 ctxIdx

Initialisation variables	mvp_idx_lc, mvp_idx_l0, mvp_idx_l1 ctxIdx			
	0	1	2	3
m	0	0	0	0
n	64	64	64	64

Table 9-328-33[illegible] – Values of variable m and n for no_residual_data_flag ctxIdx

Initialisation variables	no_residual_data_flag ctxIdx							
	0	1	2	3	4	5	6	7
m	-14	-12	-8	-25	-26	-16	-14	-27
n	72	86	78	128	88	91	87	132

Table 9-338-33[illegible] – Values of variable m and n for split_transform_flag ctxIdx

Initialisation variables	split_transform_flag ctxIdx											
	0	1	2	3	4	5	6	7	8	9	10	11
m	0	12	22	-2	0	-28	-30	-34	-11	-31	-42	-47
n	64	12	4	49	13	89	99	106	38	88	118	130

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000440

Table 9-348-34[illegible] – Values of variable m and n for cbf_luma ctxIdx

Initialisation variables	cbf_luma ctxIdx											
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
m	-22	-5	-16	-16	-18	-41	-29	-23	-11	-32	-19	-16
n	116	75	112	111	98	120	117	108	80	83	89	85

Table 9-359-35[illegible] – Values of variable m and n for cbf_cb ctxIdx

Initialisation variables	cbf_cb ctxIdx											
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
m	-35	-12	-9	-10	-46	-42	-11	-19	-22	-48	-7	-37
n	116	61	73	75	114	119	74	90	52	123	68	121

Table 9-369-36[illegible] – Values of variable m and n for cbf_cr ctxIdx

Initialisation variables	cbf_cr ctxIdx											
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**
m	-29	-12	-5	-6	-43	-41	-17	-25	-19	-48	-21	-9
n	104	59	65	67	107	118	86	101	45	123	94	73

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000441

Table 9-378-378-33 – Values of variable m and n for last_significant_coefficient_x ctxIdx

Initialisation variables	last_significant_coeff_y ctxIdx														
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
m	19	12	16	22	12	12	12	5	16	15	17	17	19	19	4
n	19	36	34	18	35	35	32	46	21	20	13	14	10	12	37
	15	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**
m	13	22	27	26	18	6	12	34	38	24	14	41	45	56	30
n	22	-4	-19	-12	6	27	10	-33	-42	-15	7	7	1	-9	22
	30	**31**	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**
m	14	23	19	25	29	29	19	10	12	-2	-37	15	14	15	25
n	40	24	32	26	10	4	20	37	38	60	114	29	36	41	8
	45	**46**	**47**	**48**	**49**	**50**	**51**	**52**	**53**	**54**	**55**	**56**	**57**	**58**	**59**
m	26	25	21	12	18	25	32	32	29	24	13	27	40	42	40
n	4	10	16	34	21	1	-12	-15	-11	-5	12	6	-32	-39	-39
	60	**61**	**62**	**63**	**64**	**65**	**66**	**67**	**68**	**69**	**70**	**71**	**72**	**73**	**74**
m	45	43	45	38	15	11	11	26	28	21	15	27	29	23	-1
n	-51	-51	-56	-46	-6	-4	-4	29	29	46	45	17	12	15	57
	75	**76**	**77**	**78**	**79**	**80**	**81**	**82**	**83**	**84**	**85**	**86**	**87**	**88**	**89**
m	6	22	31	37	43	43	48	15	14	15	25	26	25	21	12
n	65	17	-4	-25	-44	-46	-44	29	36	41	8	4	10	16	34
	90	**91**	**92**	**93**	**94**	**95**	**96**	**97**	**98**	**99**	**100**	**101**	**102**	**103**	**104**
m	18	25	32	32	29	24	13	27	40	42	40	45	43	45	38
n	21	1	-12	-15	-11	-5	12	6	-32	-39	-39	-51	-51	-56	-46
	105	**106**	**107**	**108**	**109**	**110**	**111**	**112**	**113**	**114**	**115**	**116**	**117**	**118**	**119**
m	15	11	11	26	28	21	15	27	29	23	-1	6	22	31	37
n	-6	-4	-4	29	29	46	45	17	12	15	57	65	17	-4	-25
	120	**121**	**122**												
m	43	43	48												
n	-44	-46	-44												

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000442

Table 9-388-388-xx – Values of variable m and n for last_significant_coefficient_y ctxIdx

Initialisation variables	last_significant_coeff_y ctxIdx														
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
m	19	12	16	22	12	12	12	5	16	15	17	17	19	19	4
n	19	36	34	18	35	35	32	46	21	20	13	14	10	12	37
	15	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**
m	13	22	27	26	18	6	12	34	38	24	14	41	45	56	30
n	22	-4	-19	-12	6	27	10	-33	-42	-15	7	7	1	-9	22
	30	**31**	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**
m	14	23	19	25	29	29	19	10	12	-2	-37	15	14	15	25
n	40	24	32	26	10	4	20	37	38	60	114	29	36	41	8
	45	**46**	**47**	**48**	**49**	**50**	**51**	**52**	**53**	**54**	**55**	**56**	**57**	**58**	**59**
m	26	25	21	12	18	25	32	32	29	24	13	27	40	42	40
n	4	10	16	34	21	1	-12	-15	-11	-5	12	6	-32	-39	-39
	60	**61**	**62**	**63**	**64**	**65**	**66**	**67**	**68**	**69**	**70**	**71**	**72**	**73**	**74**
m	45	43	45	38	15	11	11	26	28	21	15	27	29	23	-1
n	-51	-51	-56	-46	-6	-4	-4	29	29	46	45	17	12	15	57
	75	**76**	**77**	**78**	**79**	**80**	**81**	**82**	**83**	**84**	**85**	**86**	**87**	**88**	**89**
m	6	22	31	37	43	43	48	15	14	15	25	26	25	21	12
n	65	17	-4	-25	-44	-46	-44	29	36	41	8	4	10	16	34
	90	**91**	**92**	**93**	**94**	**95**	**96**	**97**	**98**	**99**	**100**	**101**	**102**	**103**	**104**
m	18	25	32	32	29	24	13	27	40	42	40	45	43	45	38
n	21	1	-12	-15	-11	-5	12	6	-32	-39	-39	-51	-51	-56	-46
	105	**106**	**107**	**108**	**109**	**110**	**111**	**112**	**113**	**114**	**115**	**116**	**117**	**118**	**119**
m	15	11	11	26	28	21	15	27	29	23	-1	6	22	31	37
n	-6	-4	-4	29	29	46	45	17	12	15	57	65	17	-4	-25
	120	**121**	**122**												
m	43	43	48												
n	-44	-46	-44												

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000443

Table 9-39 – Values of variable m and n for significant_coeff_flag ctxIdx (A)

Initialisation variables	significant_coeff_flag ctxIdx															
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
m	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
n	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64
	16	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
m	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
n	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64
	32	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**	**45**	**46**	**47**
m	-22	-14	-17	-26	5	-2	-32	-19	-7	-24	0	-1	2	0	-15	0
n	128	96	99	113	31	65	125	72	69	111	25	49	54	64	97	64
	48	**49**	**50**	**51**	**52**	**53**	**54**	**55**	**56**	**57**	**58**	**59**	**60**	**61**	**62**	**63**
m	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
n	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64
	64	**65**	**66**	**67**	**68**	**69**	**70**	**71**	**72**	**73**	**74**	**75**	**76**	**77**	**78**	**79**
m	-4	0	-4	-5	17	2	-8	1	2	-11	6	5	2	0	-8	0
n	88	68	75	74	7	56	80	36	52	85	12	41	54	65	82	64
	80	**81**	**82**	**83**	**84**	**85**	**86**	**87**	**88**	**89**	**90**	**91**	**92**	**93**	**94**	**95**
m	-8	-7	-7	-9	-18	-7	-11	-29	4	-14	-26	2	5	6	-22	0
n	95	77	73	74	77	72	81	92	47	83	79	46	50	52	104	64
	96	**97**	**98**	**99**	**100**	**101**	**102**	**103**	**104**	**105**	**106**	**107**	**108**	**109**	**110**	**111**
m	-7	-5	-6	-8	5	3	-18	-10	0	-11	-3	8	4	-1	-7	0
n	93	79	79	79	36	56	99	59	56	84	31	35	51	66	81	64

Table 9-40 – Values of variable m and n for significant_coeff_flag ctxIdx (B)

Initialisation variables	significant_coeff_flag ctxIdx															
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
m	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
n	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64
	16	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
m	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
n	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64
	32	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**	**45**	**46**	**47**
m	-22	-14	-17	-26	5	-2	-32	-19	-7	-24	0	-1	2	0	-15	0
n	128	96	99	113	31	65	125	72	69	111	25	49	54	64	97	64
	48	**49**	**50**	**51**	**52**	**53**	**54**	**55**	**56**	**57**	**58**	**59**	**60**	**61**	**62**	**63**
m	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
n	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64
	64	**65**	**66**	**67**	**68**	**69**	**70**	**71**	**72**	**73**	**74**	**75**	**76**	**77**	**78**	**79**
m	-4	0	-4	-5	17	2	-8	1	2	-11	6	5	2	0	-8	0
n	88	68	75	74	7	56	80	36	52	85	12	41	54	65	82	64
	80	**81**	**82**	**83**	**84**	**85**	**86**	**87**	**88**	**89**	**90**	**91**	**92**	**93**	**94**	**95**
m	-8	-7	-7	-9	-18	-7	-11	-29	4	-14	-26	2	5	6	-22	0
n	95	77	73	74	77	72	81	92	47	83	79	46	50	52	104	64
	96	**97**	**98**	**99**	**100**	**101**	**102**	**103**	**104**	**105**	**106**	**107**	**108**	**109**	**110**	**111**
m	-7	-5	-6	-8	5	3	-18	-10	0	-11	-3	8	4	-1	-7	0
n	93	79	79	79	36	56	99	59	56	84	31	35	51	66	81	64

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000444

Table 9-41[9-43] – Values of variable m and n for significant_coeff_flag ctxIdx (C)

Initialisation variables	significant_coeff_flag ctxIdx															
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
m	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
n	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64
	16	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
m	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
n	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64
	32	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**	**45**	**46**	**47**
m	-34	-20	-23	38	3	-4	-17	-9	-4	-16	-1	0	-1	-12	-14	0
n	146	108	111	41	34	69	103	50	66	97	24	50	61	90	97	64
	48	**49**	**50**	**51**	**52**	**53**	**54**	**55**	**56**	**57**	**58**	**59**	**60**	**61**	**62**	**63**
m	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
n	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64
	64	**65**	**66**	**67**	**68**	**69**	**70**	**71**	**72**	**73**	**74**	**75**	**76**	**77**	**78**	**79**
m	-12	-4	-8	-7	15	2	-10	4	-10	-9	-9	6	3	-1	-14	0
n	99	75	81	80	13	57	85	27	75	81	46	41	56	69	96	64
	80	**81**	**82**	**83**	**84**	**85**	**86**	**87**	**88**	**89**	**90**	**91**	**92**	**93**	**94**	**95**
m	-7	-22	-12	-35	-5	-16	-9	-26	-28	-38	-3	-25	-9	6	-15	0
n	85	102	79	121	55	88	78	73	99	123	35	97	75	54	92	64
	96	**97**	**98**	**99**	**100**	**101**	**102**	**103**	**104**	**105**	**106**	**107**	**108**	**109**	**110**	**111**
m	-9	-14	-4	-14	7	3	-7	-8	-7	-8	-2	4	1	-5	-10	0
n	94	94	74	92	37	57	78	54	70	78	31	49	60	75	87	64

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000445

Table 9-429-428-33 – Values of variable m and n for coeff_abs_level_greater1_flag ctxIdx

Initialisation variables	coeff_abs_level_greater1_flag ctxIdx														
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
m	-11	-20	-16	-13	-10	-5	-8	-8	-3	-9	0	-5	-12	-9	-1
n	87	64	68	71	73	67	26	37	36	56	63	39	56	57	52
	15	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**
m	-4	-19	-28	-23	-3	-2	-27	-22	-14	-1	-10	-22	-13	-6	-5
n	72	73	88	85	59	72	97	89	77	58	86	74	63	57	63
	30	**31**	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**
m	-4	16	9	5	-7	-6	-13	-12	-18	-14	-5	-8	-22	-35	-3
n	70	-2	23	41	67	63	-5	42	53	59	65	36	67	89	47
	45	**46**	**47**	**48**	**49**	**50**	**51**	**52**	**53**	**54**	**55**	**56**	**57**	**58**	**59**
m	-12	-20	-51	-44	-23	-1	-58	-71	-67	-91	-1	20	13	2	5
n	77	66	113	109	84	64	127	143	134	187	64	-3	21	46	48
	60	**61**	**62**	**63**	**64**	**65**	**66**	**67**	**68**	**69**	**70**	**71**	**72**	**73**	**74**
m	-4	29	1	-6	-9	-6	23	3	-5	5	-12	-2	-15	-11	-3
n	71	-5	45	58	67	66	-7	26	42	31	79	45	67	62	54
	75	**76**	**77**	**78**	**79**	**80**	**81**	**82**	**83**	**84**	**85**	**86**	**87**	**88**	**89**
m	-4	-13	-3	-26	-18	-9	-30	-24	-43	-6	4	34	21	14	5
n	70	68	100	91	84	83	103	90	122	66	59	-11	10	25	44
	90	**91**	**92**	**93**	**94**	**95**	**96**	**97**	**98**	**99**	**100**	**101**	**102**	**103**	**104**
m	-4	40	19	-11	-6	-11	12	17	-6	11	-11	1	-26	-28	-2
n	69	-33	8	65	59	68	-2	-10	34	14	70	27	71	79	47
	105	**106**	**107**	**108**	**109**	**110**	**111**	**112**	**113**	**114**	**115**	**116**	**117**	**118**	**119**
m	6	-23	-47	-55	-21	1	-38	-34	-45	10	9	41	32	14	17
n	47	67	107	117	83	59	82	77	95	25	46	-31	-17	19	18
	120	**129**	**130**	**131**	**132**	**133**	**134**	**135**	**136**	**137**	**138**	**139**	**140**	**141**	**142**
m	-2	18	10	-2	-7	-6	19	-5	-7	-4	-23	-3	2	-32	-16
n	65	22	33	55	64	67	11	50	53	54	99	51	41	102	79
	135	**136**	**137**	**138**	**139**	**140**	**141**	**142**	**143**	**144**	**145**	**146**	**147**	**148**	**149**
m	-8	-21	-26	-33	-4	-31	-34	-25	-43	-6	3	23	12	11	8
n	77	84	91	104	61	122	110	96	124	70	60	12	30	33	40
	150	**151**	**152**	**153**	**154**	**155**	**156**	**157**	**158**	**159**	**160**	**161**	**162**	**163**	**164**
m	-2	40	0	-5	7	-39	31	1	-35	-32	-15	16	-43	-75	-8
n	66	-20	46	54	37	116	-27	22	82	85	72	0	102	152	55
	165	**166**	**167**	**168**	**169**	**170**	**171**	**172**	**173**	**174**	**175**	**176**	**177**	**178**	**179**
m	-9	-12	-84	-93	25	-104	-40	-51	110	3	17	55	28	-5	29
n	68	54	171	186	12	222	92	93	-169	52	33	-45	1	55	-4

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000446

Table 9-439-438-33 – Values of variable m and n for coeff_abs_level_greater2_flag ctxIdx

Initialisation variables	coeff_abs_level_greater2_flag ctxIdx														
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
m	-12	-10	-11	-14	-35	-10	-1	-17	-5	-22	-13	-2	-13	-21	-3
n	72	79	87	94	136	58	54	86	70	105	70	59	81	96	73
	15	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**
m	-24	-19	-1	1	-17	-7	-20	-11	-27	-12	-9	-8	-4	-3	-9
n	90	88	63	60	97	69	95	84	110	96	72	76	75	76	94
	30	**31**	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**
m	-11	-4	-9	-24	-24	-18	-10	-19	-28	6	-12	-17	-8	-44	-17
n	65	66	82	107	111	58	61	82	97	47	50	73	66	115	86
	45	**46**	**47**	**48**	**49**	**50**	**51**	**52**	**53**	**54**	**55**	**56**	**57**	**58**	**59**
m	-26	-4	-43	-58	-51	-51	-85	-47	-95	-65	-2	7	-2	3	4
n	74	48	115	141	137	117	176	120	202	159	53	43	67	62	67
	60	**61**	**62**	**63**	**64**	**65**	**66**	**67**	**68**	**69**	**70**	**71**	**72**	**73**	**74**
m	-2	-13	-11	-13	-24	-4	-17	-16	-19	-28	-12	-11	-18	-6	-19
n	49	81	82	88	112	44	77	82	89	110	64	70	88	67	97
	75	**76**	**77**	**78**	**79**	**80**	**81**	**82**	**83**	**84**	**85**	**86**	**87**	**88**	**89**
m	-17	-27	-18	-13	-17	-11	-15	-10	-10	-12	-8	-7	-6	-8	-9
n	76	100	88	84	94	73	83	77	80	91	63	71	73	80	90
	90	**91**	**92**	**93**	**94**	**95**	**96**	**97**	**98**	**99**	**100**	**101**	**102**	**103**	**104**
m	14	-7	-4	-8	-50	2	11	-4	-41	-60	-12	-6	1	-5	-12
n	20	68	66	76	148	22	15	49	117	149	49	55	44	58	73
	105	**106**	**107**	**108**	**109**	**110**	**111**	**112**	**113**	**114**	**115**	**116**	**117**	**118**	**119**
m	13	5	-29	0	-23	-12	6	35	42	22	24	11	2	-5	4
n	9	26	101	46	92	57	30	-4	-24	24	0	34	61	75	62
	120	**129**	**130**	**131**	**132**	**133**	**134**	**135**	**136**	**137**	**138**	**139**	**140**	**141**	**142**
m	0	-6	-7	-31	-25	-20	-14	-20	-21	-6	-19	-34	-27	-7	-7
n	43	65	71	117	109	76	73	88	92	71	73	108	101	69	75
	135	**136**	**137**	**138**	**139**	**140**	**141**	**142**	**143**	**144**	**145**	**146**	**147**	**148**	**149**
m	-18	-7	-20	-9	-7	-26	-13	-6	-12	-2	-3	-6	-7	-6	-11
n	77	64	91	75	78	98	81	69	83	70	50	66	73	75	91
	150	**151**	**152**	**153**	**154**	**155**	**156**	**157**	**158**	**159**	**160**	**161**	**162**	**163**	**164**
m	19	2	4	-4	-32	13	-29	-64	-90	26	-24	8	-10	-16	-27
n	10	49	52	69	114	-1	85	143	186	4	68	30	61	78	96
	165	**166**	**167**	**168**	**169**	**170**	**171**	**172**	**173**	**174**	**175**	**176**	**177**	**178**	**179**
m	31	-12	-15	-68	31	-8	-32	-36	-53	36	26	19	-22	5	8
n	-16	54	70	158	12	44	63	81	106	12	-5	17	101	54	53

9.3.1.2 Initialisation process for the arithmetic decoding engine

This process is invoked before decoding the first treeblock of a slice.

Outputs of this process are the initialised decoding engine registers codIRange and codIOffset both in 16 bit register precision.

The status of the arithmetic decoding engine is represented by the variables codIRange and codIOffset. In the initialisation procedure of the arithmetic decoding process, codIRange is set equal to 510 and codIOffset is set equal to the value returned from read_bits(9) interpreted as a 9 bit binary representation of an unsigned integer with most significant bit written first.

The bitstream shall not contain data that result in a value of codIOffset being equal to 510 or 511.

NOTE – The description of the arithmetic decoding engine in this Recommendation | International Standard utilizes 16 bit register precision. However, a minimum register precision of 9 bits is required for storing the values of the variables codIRange and

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000447

codIOffset after invocation of the arithmetic decoding process (DecodeBin) as specified in subclause 9.3.3.2. The arithmetic decoding process for a binary decision (DecodeDecision) as specified in subclause 9.3.3.2.1 and the decoding process for a binary decision before termination (DecodeTerminate) as specified in subclause 9.3.3.2.4 require a minimum register precision of 9 bits for the variables codIRange and codIOffset. The bypass decoding process for binary decisions (DecodeBypass) as specified in subclause 9.3.3.2.3 requires a minimum register precision of 10 bits for the variable codIOffset and a minimum register precision of 9 bits for the variable codIRange.

9.3.2 Binarization process

Input to this process is a request for a syntax element.

Output of this process is the binarization of the syntax element, maxBinIdxCtx, ctxIdxOffset, and bypassFlag.

Table 9-31 specifies the type of binarization process, maxBinIdxCtx, ctxIdxTable, and ctxIdxOffset associated with each syntax element.

The specification of the unary (U) binarization process, the truncated unary (TU) binarization process, the Rice (R) binarization process, the truncated Rice (TR), the concatenated unary / k-th order Exp-Golomb (UEGk) binarization process, the concatenated truncated Rice / k-th order Exp-Golomb (TREGk) binarization process, and the fixed-length (FL) binarization process are given in subclauses 9.3.2.1 to 9.3.2.6, respectively. Other binarizations are specified in subclauses 9.3.2.7 to 9.3.2.10.

변경된 필드 코드

The binarizations for the syntax element coeff_abs_level_minus3 as specified in subclause 9.3.2.10 consist of bin strings given by a concatenation of prefix and suffix bit strings. The UEGk binarization as specified in subclause 9.3.2.3, which is used for the binarization of the syntax elements mvd_lX (X = 0, 1 C) also consist of a concatenation of prefix and suffix bit strings. For these binarization processes, the prefix and the suffix bit string are separately indexed using the binIdx variable as specified further in subclause 9.3.3. The two sets of prefix bit strings and suffix bit strings are referred to as the binarization prefix part and the binarization suffix part, respectively.

Associated with each binarization or binarization part of a syntax element is a specific value of the context index table (ctxIdxTable) variable and the related context index offset (ctxIdxOffset) variable and a specific value of the maxBinIdxCtx variable as given in 오류! 참조 원본을 찾을 수 없습니다. Table 9-33. When two values for each of these variables are specified for one syntax element in 오류! 참조 원본을 찾을 수 없습니다. Table 9-22, the value in the upper row is related to the prefix part while the value in the lower row is related to the suffix part of the binarization of the corresponding syntax element.

The use of the DecodeBypass process and the variable bypassFlag is derived as follows.

- If no value is assigned to ctxIdxOffset for the corresponding binarization or binarization part in 오류! 참조 원본을 찾을 수 없습니다. Table 9-22 labelled as "na", all bins of the bit strings of the corresponding binarization or of the binarization prefix/suffix part are decoded by invoking the DecodeBypass process as specified in subclause 9.3.3.2.3. In such a case, bypassFlag is set equal to 1, where bypassFlag is used to indicate that for parsing the value of the bin from the bitstream the DecodeBypass process is applied.
- Otherwise, for each possible value of binIdx up to the specified value of maxBinIdxCtx given in 오류! 참조 원본을 찾을 수 없습니다. Table 9-22, a specific value of the variable ctxIdx is further specified in subclause 9.3.3. bypassFlag is set equal to 0.

The possible values of the context index ctxIdx vary depending on the value of ctxIdxTable. The value assigned to ctxIdxOffset specifies the lower value of the range of ctxIdx assigned to the corresponding binarization or binarization part of a syntax element.

ctxIdxTable = 0 and ctxIdx = ctxIdxOffset = 0 are assigned to the syntax element end_of_slice_flag as further specified in subclause 9.3.3.1. For parsing the value of the corresponding bin from the bitstream, the arithmetic decoding process for decisions before termination (DecodeTerminate) as specified in subclause 9.3.3.2.4 is applied.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000448

Table 9-448-448-33 – Syntax elements and associated types of binarization, maxBinIdxCtx, ctxIdxTable, and ctxIdxOffset

Syntax element		Type of binarization	maxBinIdx Ctx	ctxIdxTable	ctxIdxOffset
alf_cu_flag	I	FL, cMax = 1	0	Table 9-18Table 9-5	0
	P		0	Table 9-18Table 9-5	3
	B		0	Table 9-18Table 9-5	6
end_of_slice_flag	all	FL, cMax = 1	0	0	0
split_coding_unit_flag	I	FL, cMax = 1	0	Table 9-12Table 9-6	0
	P		0	Table 9-12Table 9-6	3
	B		0	Table 9-12Table 9-6	6
skip_flag	P	FL, cMax = 1	0	Table 9-20Table 9-7	0
	B		0	Table 9-20Table 9-7	3
cu_qp_delta	I	as specified in subclause 9.3.2.7	2	Table 9-21Table 9-8	0
	P		2	Table 9-21Table 9-8	4
	B		2	Table 9-21Table 9-8	8
pred_type	I	as specified in subclause 9.3.2.8	0	Table 9-22Table 9-9	0
	P		3	Table 9-22Table 9-9	1
	B		4	Table 9-22Table 9-9	5
pcm_flag	all	FL, cMax = 1	0	0	0
prev_intra_luma_pred_flag	I	FL, cMax = 1	0	Table 9-23Table 9-10	0
	P		0	Table 9-23Table 9-10	1
	B		0	Table 9-23Table 9-10	2
rem_intra_luma_pred_mode	I	as specified in subclause 9.3.2.9	0	Table 9-24Table 9-11	0
	P		0	Table 9-24Table 9-11	1
	B		0	Table 9-24Table 9-11	2
intra_chroma_pred_mode (IntraPredMode < 4)	I	TU, cMax = 3	3	Table 9-25Table 9-12	0
	P		3	Table 9-25Table 9-12	4
	B		3	Table 9-25Table 9-12	8
intra_chroma_pred_mode (4 <= IntraPredMode < 34)	I	TU, cMax = 4	4	Table 9-25Table 9-12	0
	P		4	Table 9-25Table 9-12	4

서식 있음: 글꼴: 8 pt

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000449

Table 9-44~~8~~ – Syntax elements and associated types of binarization, maxBinIdxCtx, ctxIdxTable, and ctxIdxOffset

Syntax element		Type of binarization	maxBinIdx Ctx	ctxIdxTable	ctxIdxOffset
	B		4	~~Table 9-25~~Table 9-42	8
merge_flag	P	FL, cMax = 1	0	~~Table 9-26~~Table 9-43	0
	B		0	~~Table 9-26~~Table 9-43	3
merge_idx	P	TU, cMax = NumMergeCand	3	~~Table 9-27~~Table 9-44	0
	B		3	~~Table 9-27~~Table 9-44	4
inter_pred_flag	B	FL, cMax = 1	0	~~Table 9-28~~Table 9-45	0
ref_idx_lc	B	TU, cMax = num_ref_idx_lC_active_minus1	1	~~Table 9-29~~Table 9-46	6
ref_idx_l0	P	TU, cMax = num_ref_idx_l0_active_minus1	1	~~Table 9-29~~Table 9-46	0
	B		1	~~Table 9-29~~Table 9-46	6
ref_idx_l1	B	TU, cMax = num_ref_idx_l1_active_minus1	1	~~Table 9-29~~Table 9-46	6
mvd_l0[ ][ ][0]	P	prefix and suffix as given by UEG3 with signedValFlag=1, uCoff=9	prefix: 4 suffix: na	prefix: ~~Table 9-30~~Table 9-47 suffix: na	prefix: 0 suffix: na, (uses Decode Bypass)
mvd_l0[ ][ ][1]	P		prefix:4 suffix: na	prefix: ~~Table 9-30~~Table 9-47 suffix: na	prefix: 7 suffix: na, (uses Decode Bypass)
mvd_l0[ ][ ][0], mvd_lc[ ][ ][0], mvd_l1[ ][ ][0]	B		prefix: 4 suffix: na	prefix: ~~Table 9-30~~Table 9-47 suffix: na	prefix: 15 suffix: na, (uses Decode Bypass)
mvd_l0[ ][ ][1], mvd_lc[ ][ ][1], mvd_l1[ ][ ][1]	B		prefix: 4 suffix: na	prefix: ~~Table 9-30~~Table 9-47 suffix: na	prefix: 21 suffix: na, (uses Decode Bypass)
mvp_idx_lc	B	TU, cMax = NumMVPCand(LcToLx)	1	~~Table 9-31~~Table 9-48	2
mvp_idx_l0	P	TU, cMax = NumMVPCand(L0)	1	~~Table 9-31~~Table 9-48	0
	B		1	~~Table 9-31~~Table 9-48	2
mvp_idx_l1	B	TU, cMax = NumMVPCand(L1)	1	~~Table 9-31~~Table 9-48	2
no_residual_data_flag	P	FL, cMax = 1	0	~~Table 9-32~~Table 9-49	0
	B		0	~~Table 9-32~~Table 9-49	4
split_transform_flag	I	FL, cMax = 1	0	~~Table 9-33~~Table 9-50	0

서식 있음: 글꼴: 8 pt

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000450

Table 9-44[illegible] – Syntax elements and associated types of binarization, maxBinIdxCtx, ctxIdxTable, and ctxIdxOffset

Syntax element		Type of binarization	maxBinIdx Ctx	ctxIdxTable	ctxIdxOffset
	P		0	~~Table 9-33~~Table 9-20	4
	B		0	~~Table 9-33~~Table 9-20	8
cbf_luma	I	FL, cMax = 1	0	~~Table 9-34~~Table 9-21	0
	P		0	~~Table 9-34~~Table 9-21	4
	B		0	~~Table 9-34~~Table 9-21	8
cbf_cb	I	FL, cMax = 1	0	~~Table 9-35~~Table 9-22	0
	P		0	~~Table 9-35~~Table 9-22	4
	B		0	~~Table 9-35~~Table 9-22	8
cbf_cr	I	FL, cMax = 1	0	~~Table 9-36~~Table 9-23	0
	P		0	~~Table 9-36~~Table 9-23	4
	B		0	~~Table 9-36~~Table 9-23	8
last_significant_coeff_x	I	TU, cMax = (1 << log2TrafoSize) − 1	30	~~Table 9-37~~Table 9-24	0
	P		30	~~Table 9-37~~Table 9-24	41
	B		30	~~Table 9-37~~Table 9-24	82

서식 있음: 글꼴: 8 pt

HEVC

Table 9-44[illegible] – Syntax elements and associated types of binarization, maxBinIdxCtx, ctxIdxTable, and ctxIdxOffset

Syntax element		Type of binarization	maxBinIdx Ctx	ctxIdxTable	ctxIdxOffset
last_significant_coeff_y	I	TU, cMax = (1 << log2TrafoSize) − 1	30	Table 9-38Table 9-25	0
	P		30	Table 9-38Table 9-25	41
	B		30	Table 9-38Table 9-25	82
significant_coeff_flag	I		0	Table 9-39Table 9-26	0
	P		0	Table 9-40	0
	B		0	Table 9-41Table 9-28	0
coeff_abs_level_greater1_flag	I	FL, cMax = 1	0	Table 9-42Table 9-29	0

significant_coeff_flag ctxIdx

Initialisation variables	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
m	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
n	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
m	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
n	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64
	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
m	-22	-14	-17	-26	5	-2	-32	-19	-7	-24	0	-1	2	0	-15	0
n	128	96	99	133	31	65	125	72	69	111	25	89	54	64	97	64
	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
m	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
n	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64
	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
m	-4	0	-4	-5	17	1	-8	1	7	-11	6	5	2	0	-8	0
n	88	68	75	74	7	56	80	36	52	85	12	41	54	63	82	64
	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
m	-8	-7	-7	-9	-18	-7	-11	-29	4	-14	-26	2	5	0	-22	0
n	95	77	73	74	77	72	81	92	47	83	79	46	50	52	104	64
	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
m	-7	-5	-6	-8	5	1	-18	-10	0	-11	-3	8	4	-1	-7	0
n	93	72	72	79	36	56	99	59	56	84	31	35	51	66	81	64

서식 있음: 글꼴: 8 pt

서식 있음: 글꼴: 8 pt

서식 있음: 글꼴: 8 pt

서식 있음: 글꼴: 8 pt

서식 있음: 글꼴: Times New Roman, 8 pt, 영어(영국)

서식 있음: Table_Text, 왼쪽, 간격 앞: 0 pt, 단락 뒤: 0 pt, 줄 간격: 1줄, 다음 단락과의 사이에 페이지 나누기, 현재 단락을 나눔

서식 있는 표

서식 있음: 글꼴: Times New Roman, 8 pt, 영어(영국)

서식 있음: 글꼴: 8 pt

서식 있음: Table_Text, 가운데, 다음 단락과의 사이에 페이지 나누지 않음

서식 있음: 글꼴: 8 pt

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000452

Table 9-448-448-33 – Syntax elements and associated types of binarization, maxBinIdxCtx, ctxIdxTable, and ctxIdxOffset

Syntax element		Type of binarization	maxBinIdx Ctx	ctxIdxTable	ctxIdxOffset
	P		0	Table 9-42Table 9-33	60
	B		0	Table 9-42Table 9-29	120
coeff_abs_level_greater2_flag	I	FL, cMax = 1	0	Table 9-43Table 9-30	0
	P		0	Table 9-43Table 9-36	60
	B		0	Table 9-43Table 9-30	120
coeff_abs_level_minus3	all	prefix and suffix as specified in subclause 9.3.2.10	prefix: na suffix: na	prefix: na suffix: na	prefix: na, (uses Decode Bypass) suffix: na, (uses Decode Bypass)
coeff_sign_flag	all	FL, cMax = 1	na	na	na, (uses Decode Bypass)

[Ed. (BB): The TU binarization for last_significant_coeff_x and last_significant_coeff_y is using 0s and a terminating 1 instead of using 1s and a terminating 0.]

9.3.2.1 Unary (U) binarization process

Input to this process is a request for a U binarization for a syntax element.

Output of this process is the U binarization of the syntax element.

The bin string of a syntax element having (unsigned integer) value synElVal is a bit string of length synElVal + 1 indexed by binIdx. The bins for binIdx less than synElVal are equal to 1. The bin with binIdx equal to synElVal is equal to 0.

Table 9-45Table 9-35 illustrates the bin strings of the unary binarization for a syntax element.

Table 9-459-459-35 – Bin string of the unary binarization (informative)

Value of syntax element	Bin string					
0 (I_NxN)	0					
1	1	0				
2	1	1	0			
3	1	1	1	0		
4	1	1	1	1	0	
5	1	1	1	1	1	0
…						
binIdx	0	1	2	3	4	5

서식 있음: Table_Text, 가운데, 다음 단락과의 사이에 페이지 나누지 않음
서식 있음: 글꼴: 8 pt
서식 있음: 글꼴: 8 pt
서식 있음: Table_Text, 가운데, 다음 단락과의 사이에 페이지 나누지 않음
서식 있음: 글꼴: 8 pt
서식 있음: 글꼴: 8 pt
서식 있음: 글꼴: 8 pt
변경된 필드 코드

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000453

9.3.2.2 Truncated unary (TU) binarization process

Input to this process is a request for a TU binarization for a syntax element and cMax.

Output of this process is the TU binarization of the syntax element.

For syntax element (unsigned integer) values less than cMax, the U binarization process as specified in subclause 9.3.2.2.1 is invoked. For the syntax element value equal to cMax the bin string is a bit string of length cMax with all bins being equal to 1.

변경된 필드 코드

NOTE – TU binarization is always invoked with a cMax value equal to the largest possible value of the syntax element being decoded.

9.3.2.3 Truncated Rice (TR) binarization process

Input to this process is a request for a TR binarization for a syntax element, cRiceParam and cTRMax.

Output of this process is the TR binarization of the syntax element.

A TR bin string is a concatenation of a prefix bit string and a suffix bit string. The prefix of the binarization is specified by invoking the TU binarization process for the prefix part of the value specified by synElVal >> cRiceParam with cMax = cTRMax >> cRiceParam. The suffix of the TR bin string is the binary representation of synElVal – ((synElVal >> cRiceParam) << cRiceParam).

NOTE – For the input parameter cRiceParam = 0 the TR binarization is exactly the TU binarization.

9.3.2.4 k-th order Exp-Golomb (EGk) binarization process

Input to this process is a request for an EGk binarization for a syntax element and signedFlag.

Output of this process is the EGk binarization of the syntax element.

The bin string of the EGk binarization process of a syntax element synVal is specified by a process equivalent to the following pseudo-code:

```
absV = Abs( synVal )
stopLoop = 0
do {
    if( absV >= ( 1 << k ) ) {
        put( 1 )
        absV = absV – ( 1 << k )
        k++
    } else {
        put( 0 )
    (9-5)(5)
        while( k– – )
            put( ( absV >> k ) & 1 )
        stopLoop = 1
    }
} while( !stopLoop )
if( signedFlag && synVal != 0) {
    if( synVal > 0 )
        put( 0 )
    else
        put( 1 )
}
```

NOTE 1 – The specification for the k-th order Exp-Golomb (EGk) code uses 1's and 0's in reverse meaning for the unary part of the Exp-Golomb code of 0-th order as specified in subclause 9.1.

변경된 필드 코드

9.3.2.5 Concatenated unary/ k-th order Exp-Golomb (UEGk) binarization process

Input to this process is a request for a UEGk binarization for a syntax element, signedValFlag and uCoff.

Output of this process is the UEGk binarization of the syntax element.

A UEGk bin string is a concatenation of a prefix bit string and a suffix bit string. The prefix of the binarization is specified by invoking the TU binarization process for the prefix part Min(uCoff, Abs(synElVal)) of a syntax element value synElVal as specified in subclause 9.3.2.2 with cMax = uCoff, where uCoff > 0.

변경된 필드 코드

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000454

The variable k for a UEGk bin string is dependent on the syntax element for which a UEGk binarization is requested. 오류! 참조 원본을 찾을 수 없습니다.Table 9-34 specifies the associated types of binarization for syntax elements, including the value of k for syntax elements that use UEGk binarization.

NOTE 1 – For the syntax elements mvd_lc[ ][ ][ ], mvd_l0[ ][ ][ ] and mvd_l1[ ][ ][ ] a UEG3 binarization is used (k is equal to 3).

The UEGk bin string is derived as follows.

– If one of the following is true, the bin string of a syntax element having value synElVal consists only of a prefix bit string:
  – signedValFlag is equal to 0 and the prefix bit string is not equal to the bit string of length uCoff with all bits equal to 1,
  – signedValFlag is equal to 1 and the prefix bit string is equal to the bit string that consists of a single bit with value equal to 0.
– Otherwise, the bin string of the UEGk suffix part of a syntax element value synElVal is specified by invoking the EGk binarization process as specified in subclause 9.3.2.4 with k being initialised to the value that is specified in Table 9-3 for the requested UEGk binarization process.

서식 있음: 강조

9.3.2.6 Fixed-length (FL) binarization process

Input to this process is a request for a FL binarization for a syntax element and cMax.

Output of this process is the FL binarization of the syntax element.

FL binarization is constructed by using a fixedLength-bit unsigned integer bin string of the syntax element value, where fixedLength = Ceil(Log2(cMax + 1)). The indexing of bins for the FL binarization is such that the binIdx = 0 relates to the least significant bit with increasing values of binIdx towards the most significant bit.

9.3.2.7 Binarization process for cu_qp_delta

Input to this process is a request for a binarization for the syntax element mb_qp_delta.

Output of this process is the binarization of the syntax element.

For syntax elements mb_qp_delta, the U binarization process as specified in subclause 9.3.2.1 is invoked with the mapped value of the syntax element mb_qp_delta as input and the bin string as output. The assignment rule between the signed value of mb_qp_delta and its mapped value is given as specified in Table 9-3Table 9-3.

변경된 필드 코드

9.3.2.8 Binarization process for pred_type

Input to this process is a request for a binarization for the syntax element pred_type and a variable cLog2CUSize specifying the current CU size.

Output of this process is the binarization of the syntax element.

The binarization for pred_type is given by Table 9-16Table 9-6 depending on slice type and the size of the coding unit.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000455

Table 9-468-468-45 – Binarization for pred_type

Slice type	Value of pred_type	PredMode	PartMode	Bin string	
				cLog2CUSize > 3	cLog2CUSize = = 3e
I	0	MODE_INTRA	PART_2Nx2N	-	1
	1	MODE_INTRA	PART_NxN	-	0
P	0	MODE_INTER	PART_2Nx2N	01	01
	1	MODE_INTER	PART_2NxN	001	001
	2	MODE_INTER	PART_Nx2N	000	0001
	3	MODE_INTER	PART_NxN	-	0000
	4	MODE_INTRA	PART_2Nx2N	1	11
	5	MODE_INTRA	PART_NxN	-	10
B	0	MODE_INTER	PART_2Nx2N	1	1
	1	MODE_INTER	PART_2NxN	01	01
	2	MODE_INTER	PART_Nx2N	001	001
	3	MODE_INTER	PART_NxN	-	0001
	4	MODE_INTRA	PART_2Nx2N	000	0000 1
	5	MODE_INTRA	PART_NxN	-	0000 0

9.3.2.9 Binarization process for rem_intra_luma_pred_mode

Input to this process is a request for the syntax element rem_intra_luma_pred_mode and cNumBins.

Output of this process is the binarization of the syntax element.

The binarization for rem_intra_luma_pred_mode is given by Table 9-47Table 9-35.

Table 9-479-479-53 – Binarization for rem_intra_luma_pred_mode

Value of rem_intra_luma_pred_mode	Bin string
less than 32	FL, cMax = cNumBins
32	111110
33	111111

9.3.2.10 Binarization process for coeff_abs_level_minus3

Input to this process is a request for a binarization for the syntax element coeff_abs_level_minus3[n].

Output of this process is the binarization of the syntax element.

The variables cLastSE and cLastRiceParam are derived as follows.

– If n is equal to 0 or all previous syntex elements coeff_abs_level_minus3[pos] with pos less than n are derived to be equal to 0 instead of beeing explicitly parsed, cLastSE and cLastRiceParam are set equal to 0.

– Otherwise (n is not equal to 0), cLastSE is set equal to coeff_abs_level_minus3[n − 1] and cLastRiceParam is set equal to the value of cRiceParam that has been derived during the invocation of this subclause for the syntax element coeff_abs_level_minus2[n – 1] of the same transform block.

The variable cRiceParam is derived as follows.

– If cLastSE is equal to 0 or 1 and cLastRiceParam is equal to 0, cRiceParam is set equal to 0.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000456

– Otherwise if cLastSE is equal to 2 or 3 and cLastRiceParam is equal to 0 or 1, cRiceParam is set equal to 1.
– Otherwise if one of the following conditions is true, cRiceParam is set equal to 2.
  – cLastSE is in the range of 4 to 12 and cLastRiceParam is equal to 0.
  – cLastSE is in the range of 4 to 10 and cLastRiceParam is equal to 1.
  – cLastSE is in the range of 4 to 9 and cLastRiceParam is equal to 2.
– Otherwise, cRiceParam is set equal to 3.

The variable cTRMax is derived from cRiceParam as given in Table 9-48~~Table 9-36~~.

Table 9-48~~9-38~~ – Specifcation of cTRMax depending on cRiceParam

cRiceParam	cTRMax
0	7
1	20
2	42
3	70

The binarization of coeff_abs_level_minus3 consists of a prefix part and (when present) a suffix part. The prefix part of the binarization is derived by invoking the TR binarization process as specified in subclause 9.3.2.3 for the syntax element coeff_abs_level_minus3[n] with the variables cRiceParam and cTRMax as the inputs.

When the prefix bit string is not equal to the bin string that consists of (cTRMax + 1 >> cRiceParam) + cRiceParam ones, the bin string consists of a prefix bin string and a suffix bin string. The suffix bin string is derived using the EG0 binarization as specified in subclause 9.3.2.4.

9.3.3 Decoding process flow

Input to this process is a binarization of the requested syntax element, maxBinIdxCtx, bypassFlag, ctxIdxTable and ctxIdxOffset as specified in subclause 9.3.2.

Output of this process is the value of the syntax element.

This process specifies how each bit of a bit string is parsed for each syntax element.

After parsing each bit, the resulting bit string is compared to all bin strings of the binarization of the syntax element and the following applies.

– If the bit string is equal to one of the bin strings, the corresponding value of the syntax element is the output.
– Otherwise (the bit string is not equal to one of the bin strings), the next bit is parsed.

While parsing each bin, the variable binIdx is incremented by 1 starting with binIdx being set equal to 0 for the first bin.

When the binarization of the corresponding syntax element consists of a prefix and a suffix binarization part, the variable binIdx is set equal to 0 for the first bin of each part of the bin string (prefix part or suffix part). In this case, after parsing the prefix bit string, the parsing process of the suffix bit string related to the binarizations specified in subclauses 9.3.2.5 and 9.3.2.10 is invoked depending on the resulting prefix bit string as specified in subclauses 9.3.2.5 and 9.3.2.10.

Depending on the variable bypassFlag, the following applies.

– If bypassFlag is equal to 1, the bypass decoding process as specified in subclause 9.3.3.2.3 is applied for parsing the value of the bins from the bitstream.
– Otherwise (bypassFlag is equal to 0), the parsing of each bin is specified by the following two ordered steps:
  1. Given binIdx, maxBinIdxCtx, ctxIdxTable, and ctxIdxOffset, ctxIdx is derived as specified in subclause 9.3.3.1.
  2. Given ctxIdxTable and ctxIdx, the value of the bin from the bitstream as specified in subclause 9.3.3.2 is decoded.

9.3.3.1 Derivation process for ctxIdx

Inputs to this process are binIdx, maxBinIdxCtx, ctxIdxTable, and ctxIdxOffset.

Output of this process is ctxIdx.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000457

Table 9-49Table 9-37 shows the assignment of ctxIdx increments (ctxIdxInc) to binIdx for all ctxIdxTable and ctxIdxOffset.

The ctxIdx to be used with a specific binIdx is specified by first determining the ctxIdxTable and ctxIdxOffset associated with the given bin string or part thereof. The ctxIdxOffset is listed in Table 9-49Table 9-37, the ctxIdx for a binIdx is the sum of ctxIdxOffset and ctxIdxInc, which is found in Table 9-49Table 9-37. When more than one value is listed in Table 9-49Table 9-37 for a binIdx, the assignment process for ctxIdxInc for that binIdx is further specified in the subclauses given in parenthesis of the corresponding table entry.

All bins with binIdx greater than maxBinIdxCtx are parsed using the value of ctxIdx being assigned to binIdx equal to maxBinIdxCtx.

All entries in Table 9-49Table 9-37 labelled with "na" correspond to values of binIdx that do not occur for the corresponding ctxIdxOffset.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000458

Table 9-48 – Assignment of ctxIdxInc to binIdx for all ctxIdxTable and ctxIdxOffset values

Syntax element	ctxIdxTable, ctxIdxOffset		binIdx				
			0	1	2	3	>=4
alf_cu_flag	Table 9-8	0	0,1,2 (subclause 9.3.3.1.1.1)	na	na	na	na
	Table 9-5	3	0,1,2 (subclause 9.3.3.1.1.1)	na	na	na	na
		6	0,1,2 (subclause 9.3.3.1.1.1)	na	na	na	na
split_coding_unit_flag	Table 9-9	0	0,1,2 (subclause 9.3.3.1.1.1)	na	na	na	na
	Table 9-6	3	0,1,2 (subclause 9.3.3.1.1.1)	na	na	na	na
		6	0,1,2 (subclause 9.3.3.1.1.1)	na	na	na	na
skip_flag	Table 9-10	0	0,1,2 (subclause 9.3.3.1.1.1)	na	na	na	na
	Table 9-7	3	0,1,2 (subclause 9.3.3.1.1.1)	na	na	na	na
cu_qp_delta	Table 9-11	0	0	2	3	3	3
	Table 9-8	4	0	2	3	3	3
		8	0	2	3	3	3
pred_type	Table 9-12	0	0	na	na	na	na
	Table 9-9	1	0	1	2	3	na
		5	0	1	2	3	4
prev_intra_luma_pred_flag	Table 9-13	0	0	na	na	na	na
	Table 9-10	1	0	na	na	na	na
		2	0	na	na	na	na
rem_intra_luma_pred_mode	Table 9-14	0	0	0	0	0	0
	Table 9-11	1	0	0	0	0	0
		2	0	0	0	0	0
intra_chroma_pred_mode	Table 9-15	0	0,1,2 (subclause 9.3.3.1.1.1)	3	3	3	na
	Table 9-12	4	0,1,2 (subclause 9.3.3.1.1.1)	3	3	3	na
		8	0,1,2 (subclause 9.3.3.1.1.1)	3	3	3	na
merge_flag	Table 9-16	0	0,1,2 (subclause 9.3.3.1.1.1)	na	na	na	na
	Table 9-13	3	0,1,2 (subclause 9.3.3.1.1.1)	na	na	na	na
merge_idx	Table 9-17	0	0,1,2,3 (subclause 9.3.3.1.1.2)	0,1,2,3 (subclause 9.3.3.1.1.2)	0,2,3 (subclause 9.3.3.1.1.2)	3	na
	Table 9-14	4	0,1,2,3 (subclause 9.3.3.1.1.2)	0,1,2,3 (subclause 9.3.3.1.1.2)	0,2,3 (subclause 9.3.3.1.1.2)	3	na
inter_pred_flag	Table 9-18	0	0,1,2 (subclause 9.3.3.1.1.1)	na	na	na	na

서식 있음: 글꼴: 8 pt

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000459

Table 9-499-499-49 – Assignment of ctxIdxInc to binIdx for all ctxIdxTable and ctxIdxOffset values

Syntax element	ctxIdxTable, ctxIdxOffset		binIdx				
			0	1	2	3	>=4
ref_idx_l0	Table 9-29Table 9-16	0	0,1,2,3 (subclause 9.3.3.1.1.1)	4	4	4	4
ref_idx_l0, ref_idx_l1, ref_idx_lc	Table 9-29Table 9-16	6	0,1,2,3 (subclause 9.3.3.1.1.1)	4	4	4	4
mvd_l0[ ][ ][0]	Table 9-30Table 9-17	0	0,1,2 (subclause 9.3.3.1.1.1)	3	4	5	6
mvd_l0[ ][ ][1]	Table 9-30Table 9-17	7	0,1,2 (subclause 9.3.3.1.1.1)	3	4	5	6
mvd_l0[ ][ ][0], mvd_lc[ ][ ][0], mvd_l1[ ][ ][0]	Table 9-30Table 9-17	15	0,1,2 (subclause 9.3.3.1.1.1)	3	4	5	6
mvd_l0[ ][ ][1], mvd_lc[ ][ ][1], mvd_l1[ ][ ][1]	Table 9-30Table 9-17	21	0,1,2 (subclause 9.3.3.1.1.1)	3	4	5	6
mvp_idx_l0	Table 9-31Table 9-18	0	0	1	1	1	1
mvp_idx_l0, mvp_idx_l1, mvp_idx_lc	Table 9-31Table 9-18	2	0	1	1	1	1
no_residual_data_flag	Table 9-32Table 9-19	0	0,1,2,3 (subclause 9.3.3.1.1.1)	na	na	na	na
		4	0,1,2,3 (subclause 9.3.3.1.1.1)	na	na	na	na
split_transform_flag	Table 9-33Table 9-20	0	cuDepth + trafoDepth	na	na	na	na
		4	cuDepth + trafoDepth	na	na	na	na
		8	cuDepth + trafoDepth	na	na	na	na
cbf_luma	Table 9-34Table 9-21	0	0,1,2,3 (subclause 9.3.3.1.1.1)	na	na	na	na
		4	0,1,2,3 (subclause 9.3.3.1.1.1)	na	na	na	na
		8	0,1,2,3 (subclause 9.3.3.1.1.1)	na	na	na	na
cbf_cb (PredMode == MODE_INTRA)	Table 9-35Table 9-22	0	0,1,2,3 (subclause 9.3.3.1.1.1)	na	na	na	na
		4	0,1,2,3 (subclause 9.3.3.1.1.1)	na	na	na	na
		8	0,1,2,3 (subclause 9.3.3.1.1.1)	na	na	na	na
cbf_cb (PredMode != MODE_INTRA)	Table 9-35Table 9-22	0	trafoDepth	na	na	na	na
		4	trafoDepth	na	na	na	na
		8	trafoDepth	na	na	na	na
cbf_cr (PredMode == MODE_INTRA)	Table 9-36Table 9-23	0	0,1,2,3 (subclause 9.3.3.1.1.1)	na	na	na	na
		4	0,1,2,3 (subclause 9.3.3.1.1.1)	na	na	na	na
		8	0,1,2,3 (subclause 9.3.3.1.1.1)	na	na	na	na

서식 있음: 글꼴: 8 pt

HEVC

Table 9-499-488-39 – Assignment of ctxIdxInc to binIdx for all ctxIdxTable and ctxIdxOffset values

Syntax element	ctxIdxTable, ctxIdxOffset		binIdx				
			0	1	2	3	>=4
cbf_cr (PredMode != MODE_INTRA)	Table 9-36Table 9-23	0	trafoDepth	na	na	na	na
		4	trafoDepth	na	na	na	na
		8	trafoDepth	na	na	na	na

서식 있음: 글꼴: 8 pt

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000461

Table 9-~~499~~ ~~499~~ ~~49~~ – Assignment of ctxIdxInc to binIdx for all ctxIdxTable and ctxIdxOffset values

Syntax element	ctxIdxTable, ctxIdxOffset		binIdx 0	binIdx 1	binIdx 2	binIdx 3	binIdx >=4
last_significant_coeff_x	Table 9-37~~Table 9-24~~	0	0..40 (subclause 9.3.3.1.1.3)				
		41	0..40 (subclause 9.3.3.1.1.3)				
		82	0..40 (subclause 9.3.3.1.1.3)				
last_significant_coeff_y	Table 9-38~~Table 9-25~~	0	0..40 (subclause 9.3.3.1.1.3)				
		41	0..40 (subclause 9.3.3.1.1.3)				
		82	0..40 (subclause 9.3.3.1.1.3)				
significant_coeff_flag	Table 9-39~~Table 9-26~~	0	0..111 (subclause 9.3.3.1.1.4)	na	na	na	na
		0	0..111 (subclause 9.3.3.1.1.4)	na	na	na	na

significant_coeff_flag ctxIdx

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
64	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64
16	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
64	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64
32	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**	**45**	**46**	**47**
-22	-14	-17	-26	5	-2	-32	-19	-7	-24	0	-1	2	0	-15	0
128	96	99	113	31	65	125	72	69	111	25	49	54	64	97	64
48	**49**	**50**	**51**	**52**	**53**	**54**	**55**	**56**	**57**	**58**	**59**	**60**	**61**	**62**	**63**
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
64	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64
64	**65**	**66**	**67**	**68**	**69**	**70**	**71**	**72**	**73**	**74**	**75**	**76**	**77**	**78**	**79**
-4	0	-4	-5	17	2	-8	1	2	-11	6	5	2	0	-8	0
88	68	75	74	7	56	80	36	52	85	12	41	54	65	82	64
80	**81**	**82**	**83**	**84**	**85**	**86**	**87**	**88**	**89**	**90**	**91**	**92**	**93**	**94**	**95**
-8	-7	-7	-9	-18	-7	-11	-29	4	-14	-26	2	5	6	-22	0
95	77	73	74	77	72	81	92	47	83	79	46	50	52	104	64
96	**97**	**98**	**99**	**100**	**101**	**102**	**103**	**104**	**105**	**106**	**107**	**108**	**109**	**110**	**111**
-7	-5	-6	-8	5	3	-18	-10	0	-11	-3	8	4	-1	-7	0
93	79	79	79	36	56	99	59	56	84	31	35	51	66	81	64

Table 9-40

~~Table 9-23~~

Table 9-41~~Table 9-28~~	0	0..111 (subclause 9.3.3.1.1.4)	na	na	na	na

서식 있음: 글꼴: 8 pt

서식 있음: 글꼴: 8 pt

서식 있음: 글꼴: 8 pt

서식 있음: 글꼴: Times New Roman, 8 pt, 영어(영국)

서식 있는 표

서식 있음: 글꼴: Times New Roman, 8 pt, 영어(영국)

서식 있음: 글꼴: 8 pt

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000462

Table 9-499-499-49 – Assignment of ctxIdxInc to binIdx for all ctxIdxTable and ctxIdxOffset values

Syntax element	ctxIdxTable, ctxIdxOffset		binIdx				
			0	1	2	3	>=4
coeff_abs_level_greater1_flag	Table 9-42~~Table 9-29~~	0	0..59 (subclause 9.3.3.1.1.5)	na	na	na	na
		60	0..59 (subclause 9.3.3.1.1.5)	na	na	na	na
		120	0..59 (subclause 9.3.3.1.1.5)	na	na	na	na
coeff_abs_level_greater2_flag	Table 9-43~~Table 9-30~~	0	0..59 (subclause 9.3.3.1.1.6)	na	na	na	na
		60	0..59 (subclause 9.3.3.1.1.6)	na	na	na	na
		120	0..59 (subclause 9.3.3.1.1.6)	na	na	na	na

서식 있음: 글꼴: 8 pt

서식 있음: 글꼴: 8 pt

9.3.3.1.1 Assignment process of ctxIdxInc using neighbouring syntax elements

서식 있음: 제목 5, 들여쓰기: 왼쪽: 0 cm, 탭: 4.26 글자(없음) + 5.67 글자 + 7.09 글자

Subclause 9.3.3.1.1.1 specifies the derivation process of ctxIdxInc for the syntax elements alf_cu_flag, split_coding_unit_flag, skip_flag, merge_flag, intra_chroma_pred_mode, inter_pred_flag, ref_idx_lc, ref_idx_l0, ref_idx_l1, mvd_l0, mvd_l1, mvd_lc, no_residual_data_flag, cbf_luma, cbf_cb and cbf_cr.

9.3.3.1.1.1 Derivation process of ctxIdxInc using left and above syntax elements

Input to this process is the luma location (xP, yP) specifying the top-left luma sample of the current prediction unit relative to the top-left sample of the current picture.

Output of this process is ctxIdxInc.

Let the luma location (xL, yL) specify a location covered by the prediction unit to the left of the top-left luma sample of the current prediction unit with xL = xP − MinPuSize and yL = yP, the variable availableL is derived as follows. [Ed.: (BB) MinPuSize should be defined somewhere]

– If the prediction unit covering location (xL, yL) is available, availableL is set equal to 1.

– Otherwise (the prediction unit covering location (xL, yL) is not available), availableL is set equal to 0.

Let the luma location (xA, yA) specify a location covered by the prediction unit above the top-left luma sample of the current prediction unit with xA = xP and yA = yP − MinPuSize, the variable availableA is derived as follows. [Ed.: (BB) MinPuSize should be defined somewhere]

– If the prediction unit covering location (xA, yA) is available, availableA is set equal to 1.

– Otherwise (the prediction unit covering location (xA, yA) is not available), availableA is set equal to 0

The assignment of ctxIdxInc for the syntax elements alf_cu_flag, split_coding_unit_flag, skip_flag, merge_flag, intra_chroma_pred_mode, inter_pred_flag, ref_idx_lc, ref_idx_l0, ref_idx_l1, mvd_lc, mvd_l0, mvd_l1, no_residual_data_flag, cbf_luma, cbf_cb and cbf_cr is specified in Table 9-50 – Specifcation of ctxIdxInc using left and above syntax elements~~Table 9-38~~.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000463

Table 9-[illegible] – Specifcation of ctxIdxInc using left and above syntax elements

Syntax element	condL	condA	ctxIdxInc
alf_cu_flag	alf_cu_flag[xL][yL]	alf_cu_flag[xA][yA]	(condL && availableL) + (condA && availableA)
split_coding_unit_flag	cuDepth[xL][yL] > cuDepth[xP][yP]	cuDepth[xA][yA] > cuDepth[xP][yP]	(condL && availableL) + (condA && availableA)
skip_flag	skip_flag[xL][yL]	skip_flag[xA][yA]	(condL && availableL) + (condA && availableA)
merge_flag	merge_flag[xL][yL]	merge_flag[xA][yA]	(condL && availableL) + (condA && availableA)
intra_chroma_pred_mode	IntraPredMode[xL][yL] == 4	IntraPredMode [xA][yA] == 4	(condL && availableL) + (condA && availableA)
inter_pred_flag	inter_pred_flag[xL][yL]	inter_pred_flag[xA][yA]	(condL && availableL) + (condA && availableA)
ref_idx_lc	ref_idx_lc[xL][yL] > 0	ref_idx_lc[xA][yA] > 0	(condL && availableL) + (condA && availableA) << 1
ref_idx_l0	ref_idx_l0[xL][yL] > 0	ref_idx_l0[xA][yA] > 0	(condL && availableL) + (condA && availableA) << 1
ref_idx_l1	ref_idx_l1[xL][yL] > 0	ref_idx_l1[xA][yA] > 0	(condL && availableL) + (condA && availableA) << 1
mvd_lc	mvd_lc[xL][yL] > 16	mvd_lc[xA][yA] > 16	(condL && availableL) + (condA && availableA)
mvd_l0	mvd_l0[xL][yL] > 16	mvd_l0[xA][yA] > 16	(condL && availableL) + (condA && availableA)
mvd_l1	mvd_l1[xL][yL] > 16	mvd_l1[xA][yA] > 16	(condL && availableL) + (condA && availableA)
no_residual_data_flag	no_residual_data_flag[xL][yL]	no_residual_data_flag[xA][yA]	(condL && availableL) + (condA && availableA) << 1
cbf_luma	cbf_luma[xL][yL]	cbf_luma[xA][yA]	(condL && availableL) + (condA && availableA) << 1
cbf_cb	cbf_cb[xL][yL]	cbf_cb[xA][yA]	(condL && availableL) + (condA && availableA) << 1
cbf_cr	cbf_cr[xL][yL]	cbf_cr[xA][yA]	(condL && availableL) + (condA && availableA) << 1

9.3.3.1.1.2 Derivation process of ctxIdxInc for the syntax element merge_index

Input to this process is binIdx.

Output of this process is ctxIdxInc.

The variable ctxIdxInc is derived as follows.

- If binIdx is equal to 0, the following applies.
  - If availableLeft is equal to 1and availableAbove is equal to 1, ctxIdxInc is set equal to 0.
  - Otherwise if availableLeft is equal to 1 or availableAbove is equal to 1, ctxIdxInc is set equal to 2.
  - Otherwise (availableLeft is equal to 0 and availableAbove is equal to 0), ctxIdxInc is set equal to 3.
- Otherwise if binIdx is equal to 1 and NumMergeCand is greater than 2, the following applies.
  - If availableAbove is equal to 1, ctxIdxInc is set equal to 2.
  - Otherwise (availableAbove is equal to 0), ctxIdxInc is set equal to 3.
- Otherwise if binIdx is equal to 2 and NumMergeCand is greater than 3, ctxIdxInc is set equal to 3.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000464

– Otherwise, ctxIdxInc is set equal to 0.

When ctxIdxInc is not equal to 0, ctxIdxInc is derived as follows.

ctxIdxInc = ctxIdxInc – availableCollocated (9-516)

[Ed. (BB): The flags availableLeft, availableAbove, availableCollocated need to be defined or a simpler context derivation should be used.]

9.3.3.1.1.3 Derivation process of ctxIdxInc for the syntax elements last_significant_coeff_x and last_significant_coeff_y

Inputs to this process are the binIdx, the color component index cIdx and the transform block size log2TrafoSize.

Output of this process is ctxIdxInc.

Table 9-518-519-88 – Specifcation of lastCtx depending on binIdx

binIdx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lastCtx	0	1	2	3	3	4	4	5	5	5	5	6	6	6	6	7
binIdx	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	
lastCtx	7	7	7	8	8	8	8	9	9	9	9	10	10	10	10	

The variable lastCtx is dervied from binIdx as given by Table 9-51Table 9-39. For the derivation of ctxIdxInc, the following applies.

– If log2TrafoSize is less than or equal to 2, ctxIdxInc is dervied as follows.

ctxIdxInc = lastCtx (9-527)

– Otherwise if log2TrafoSize is equal to 3, ctxIdxInc is dervied as follows.

ctxIdxInc = lastCtx + 3 (9-538)

– Otherwise if log2TrafoSize is equal to 4, ctxIdxInc is dervied as follows.

ctxIdxInc = lastCtx + 8 (9-549)

– Otherwise (log2TrafoSize is greater than 4), ctxIdxInc is dervied as follows.

ctxIdxInc = lastCtx + 15 (9-5540)

When cIdx is greater than 0, ctxIdxInc is modified as follows.

ctxIdxInc = ctxIdxInc + 26 (9-5614)

[Ed. (BB): The context derivation assumes maximum transform sizes less than or equal to 32x32 for luma and 16x16 for chroma and minimum transform sizes greater than or equal to 4x4.]

9.3.3.1.1.4 Derivation process of ctxIdxInc for the syntax element significant_coeff_flag

Inputs to this process are the color component index cIdx, the current coefficient scan position (xC , yC) and the transform block size log2TrafoSize.

Output of this process is ctxIdxInc.

The variable sigCtx depends on the current position (xC, yC), the transform block size and previsously decoded bins of the syntax element significant_coeff_flag. For the derivation of sigCtx, the following applies.

– If log2TrafoSize is less than or equal to 3, sigCtx is derived as follows.

shift = log2TrafoSize == 3 ? 1 : 0
sigCtx = ((yC >> shift) << 2) + (xC >> shift) (9-5742)

– Otherwise if xC is less than 2 and yC is less than 2, sigCtx is derived as follows.

sigCtx = (yC << 1) + xC (9-5843)

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000465

– Otherwise if yC is equal to 0, sigCtx is derived as follows.

sigCtx = 4 + significant_coeff_flag[xC-1][yC] + significant_coeff_flag[xC-2][yC] (9-5944)

– Otherwise if xC is equal to 0, sigCtx is derived as follows.

sigCtx = 4 + significant_coeff_flag[xC][yC-1] + significant_coeff_flag[xC][yC-2] (9-6045)

– Otherwise (xC is greater than 1 and yC is greater than 1) , the following applies.

– The variable sigCtx is initialized using previously decoded bins of the syntax element significant_coeff_flag as follows.

sigCtx = significant_coeff_flag[xC-1][yC] + significant_coeff_flag[xC][yC-1] + significant_coeff_flag[xC-1][yC-1] (9-6146)

– When xC is greater than 1, the following applies.

sigCtx = sigCtx + significant_coeff_flag[xC-2][yC] (9-6247)

– When yC is greater than 1, the following applies.

sigCtx = sigCtx + significant_coeff_flag[xC][yC-2] (9-6348)

– The variable sigCtx is dervied as follows.

sigCtx = 10 + min(4, sigCtx) (9-6449)

The context index increment ctxIdxInc is derived using the color component index cIdx, the transform block size log2TrafoSize and sigCtx as follows.

– If cIdx is equal to 0, ctxIdxInc is derived as follows.

ctxIdxInc = ((5 – log2TrafoSize) * 16) + sigCtx (9-6529)

– Otherwise (cIdx is greater than 0), ctxIdxInc is derived as follows.

ctxIdxInc = 64 + ((4 – log2TrafoSize) * 16) + sigCtx (9-6631)

[Ed. (BB): The context derivation assumes maximum transform sizes less than or equal to 32x32 for luma and 16x16 for chroma and minimum transform sizes greater than or equal to 4x4.]

9.3.3.1.1.5 Derivation process of ctxIdxInc for the syntax element coeff_abs_level_greater1_flag

Inputs to this process are the color component index cIdx, the transform block size log2TrafoSize, the 4x4 subset index i and the current coefficient scan index n within the current subset.

Output of this process is ctxIdxInc.

The variable ctxSet specifies the current context set and for its derivation the following applies.

– If n is equal to 0 or all previous syntex elements coeff_abs_level_greater1_flag[pos] with pos less than n are derived to be equal to 0 instead of beeing explicitly parsed, the following applies.

– If the current transform block size log2TrafoSize is equal to 2, ctxSet is derived as follows.

ctxSet = 0 (9-6722)

– Otherwise if the current 4x4 subset is the first one in scanning order (i is equal to 0), ctxSet is derived as follows.

ctxSet = 5 (9-6823)

– Otherwise (i is not equal to 0), the variable lastGreater2Ctx is set equal to the variable greater2Ctx that has been derived during the last invocation of subclause 9.3.3.1.1.6 for the syntax element coeff_abs_level_greater2_flag for the subset i – 1 and ctxSet is derived as follows.

ctxSet = ((lastGreater2Ctx + 1) >> 2) + 1 (9-6924)

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000466

- The variable greater1Ctx is set equal to 1.

- Otherwise (coeff_abs_level_greater1_flag[n] is not the first to be parsed within the current subset i),.for the derivation of ctxSet and greater1Ctx the following applies.
  - The variable ctxSet is set equal to the variable ctxSet that has been derived during the last invocation of this subclause.
  - The variable greater1Ctx is set equal to the variable greater1Ctx that has been derived during the last invocation of this subclause.
  - When greater1Ctx is equal to 1, the variable lastGreater1Flag is set equal to the syntax element coeff_abs_level_greater1_flag that has been used during the last invocation of this subclause as follows.
    - If lastGreater1Flag is equal to 1, greater1Ctx is set equal to 0.
    - Otherwise (lastGreater1Flag is equal to 0), greater1Ctx is incremented by 1.

The context index increment ctxIdxInc is derived using the current context set ctxSet and the current context greater1Ctx as follows.

$$ctxIdxInc = (ctxSet * 5) + min(4, greater1Ctx) \quad (9\text{-}7025)$$

When cIdx is greater than 0, ctxIdxInc is modified as follows.

$$ctxIdxInc = ctxIdxInc + 30 \quad (9\text{-}7126)$$

9.3.3.1.1.6 Derivation process of ctxIdxInc for the syntax element coeff_abs_level_greater2_flag

Inputs to this process are the color component index cIdx, the transform block size log2TrafoSize, the 4x4 subset index i and the current coefficient scan index n within the current subset.

Output of this process is ctxIdxInc.

The variable ctxSet specifies the current context set and for its derivation the following applies.

- If n is equal to 0 or all previous syntex elements coeff_abs_level_greater1_flag[pos] with pos less than n are derived to be equal to 0 instead of beeing explicitly parsed, the following applies.
  - If the current transform block size log2TrafoSize is equal to 2, ctxSet is derived as follows.

    $$ctxSet = 0 \quad (9\text{-}7227)$$

  - Otherwise if the current 4x4 subset is the first one in scanning order (i is equal to 0), ctxSet is derived as follows.

    $$ctxSet = 5 \quad (9\text{-}7328)$$

  - Otherwise (i is not equal to 0), the variable lastGreater2Ctx is set equal to the variable greater2Ctx that has been derived during the last invocation of this subclause for the subset i – 1 and ctxSet is derived as follows.

    $$ctxSet = (lastGreater2Ctx >> 2) + 1 \quad (9\text{-}7429)$$

  - The variable greater2Ctx is set equal to 1.

- Otherwise (coeff_abs_level_greater1_flag[n] is not the first to be parsed within the current subset i), for the derivation of ctxSet and greater2Ctx the following applies.
  - The variable ctxSet is set equal to the variable ctxSet that has been derived during the last invocation of this subclause.
  - The variable greater2Ctx is set equal to the variable greater2Ctx that has been derived during the last invocation of this subclause incremented by 1.

The context index increment ctxIdxInc is derived using the current context set ctxSet and the current context greater2Ctx as follows.

$$ctxIdxInc = (ctxSet * 5) + min(4, greater2Ctx) \quad (9\text{-}7530)$$

When cIdx is greater than 0, ctxIdxInc is modified as follows.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000467

ctxIdxInc = ctxIdxInc + 30 (9-7634)

9.3.3.2 Arithmetic decoding process

Inputs to this process are the bypassFlag, ctxIdxTable, and ctxIdx as derived in subclause 9.3.3.1, and the state variables codIRange and codIOffset of the arithmetic decoding engine.

Output of this process is the value of the bin.

Figure 9-1Figure 9-1 illustrates the whole arithmetic decoding process for a single bin. For decoding the value of a bin, the context index table ctxIdxTable and the ctxIdx is passed to the arithmetic decoding process DecodeBin(ctxIdxTable, ctxIdx), which is specified as follows.

- If bypassFlag is equal to 1, DecodeBypass( ) as specified in subclause 9.3.3.2.3 is invoked.
- Otherwise, if bypassFlag is equal to 0, ctxIdxTable is equal 0 and ctxIdx is equal to 0, DecodeTerminate( ) as specified in subclause 9.3.3.2.4 is invoked.
- Otherwise (bypassFlag is equal to 0, ctxIdxTable is not equal to 0 and ctxIdx is not equal to 0), DecodeDecision( ) as specified in subclause 9.3.3.2.1 is applied.

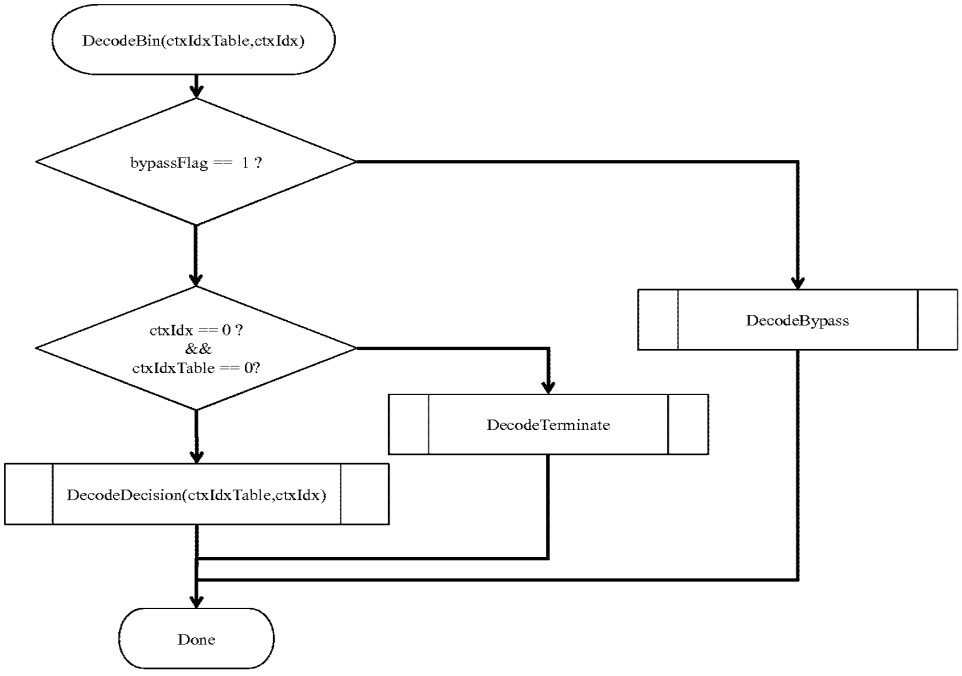


Figure 9-1 – Overview of the arithmetic decoding process for a single bin (informative)

NOTE – Arithmetic coding is based on the principle of recursive interval subdivision. Given a probability estimation p(0) and p(1) = 1 – p(0) of a binary decision (0, 1), an initially given code sub-interval with the range codIRange will be subdivided into two sub-intervals having range p(0) * codIRange and codIRange – p(0) * codIRange, respectively. Depending on the decision, which has been observed, the corresponding sub-interval will be chosen as the new code interval, and a binary code string pointing into that interval will represent the sequence of observed binary decisions. It is useful to distinguish between the most probable symbol (MPS) and the least probable symbol (LPS), so that binary decisions have to be identified as either MPS or LPS, rather than 0 or 1. Given this terminology, each context is specified by the probability p_{LPS} of the LPS and the value of MPS (valMPS), which is either 0 or 1.

The arithmetic core engine in this Recommendation | International Standard has three distinct properties:

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000468

- The probability estimation is performed by means of a finite-state machine with a table-based transition process between 64 different representative probability states $\{ p_{LPS}(pStateIdx) \mid 0 <= pStateIdx < 64 \}$ for the LPS probability p_{LPS}. The numbering of the states is arranged in such a way that the probability state with index pStateIdx = 0 corresponds to an LPS probability value of 0.5, with decreasing LPS probability towards higher state indices.
- The range codIRange representing the state of the coding engine is quantised to a small set $\{Q_1,\ldots,Q_4\}$ of pre-set quantisation values prior to the calculation of the new interval range. Storing a table containing all 64x4 pre-computed product values of $Q_i * p_{LPS}(pStateIdx)$ allows a multiplication-free approximation of the product codIRange * $p_{LPS}(pStateIdx)$.
- For syntax elements or parts thereof for which an approximately uniform probability distribution is assumed to be given a separate simplified encoding and decoding bypass process is used.

9.3.3.2.1 Arithmetic decoding process for a binary decision

Inputs to this process are ctxIdxTable, ctxIdx, codIRange, and codIOffset.

Outputs of this process are the decoded value binVal, and the updated variables codIRange and codIOffset.

Figure 9-2~~Figure 9-3~~ shows the flowchart for decoding a single decision (DecodeDecision):

1. The value of the variable codIRangeLPS is derived as follows.
   - Given the current value of codIRange, the variable qCodIRangeIdx is derived by

     qCodIRangeIdx =(codIRange >> 6) & 3 (9-~~77~~78)

   - Given qCodIRangeIdx and pStateIdx associated with ctxIdxTable and ctxIdx, the value of the variable rangeTabLPS as specified in Table 9-52~~Table 9-40~~ is assigned to codIRangeLPS:

     codIRangeLPS = rangeTabLPS[pStateIdx][qCodIRangeIdx] (9-~~78~~79)

2. The variable codIRange is set equal to codIRange – codIRangeLPS and the following applies.
   - If codIOffset is greater than or equal to codIRange, the variable binVal is set equal to 1 – valMPS, codIOffset is decremented by codIRange, and codIRange is set equal to codIRangeLPS.
   - Otherwise, the variable binVal is set equal to valMPS.

Given the value of binVal, the state transition is performed as specified in subclause 9.3.3.2.1.1. Depending on the current value of codIRange, renormalization is performed as specified in subclause 9.3.3.2.2.

9.3.3.2.1.1 State transition process

Inputs to this process are the current pStateIdx, the decoded value binVal and valMPS values of the context variable associated with ctxIdxTable and ctxIdx.

Outputs of this process are the updated pStateIdx and valMPS of the context variable associated with ctxIdx.

Depending on the decoded value binVal, the update of the two variables pStateIdx and valMPS associated with ctxIdx is derived as specified by the following pseudo-code:

```
if( binVal == valMPS )
    pStateIdx = transIdxMPS( pStateIdx )
else {                                                    (9-~~79~~80)
    if( pStateIdx == 0 )
        valMPS = 1 – valMPS
    pStateIdx = transIdxLPS( pStateIdx )
}
```

Table 9-53~~Table 9-41~~ specifies the transition rules transIdxMPS( ) and transIdxLPS( ) after decoding the value of valMPS and 1 – valMPS, respectively.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000469

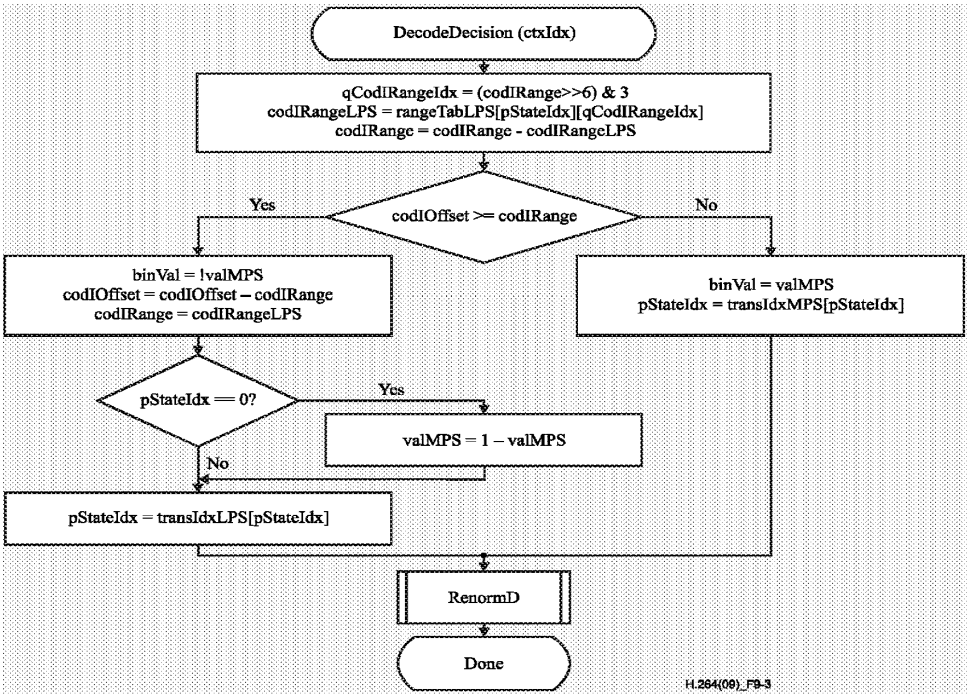


Figure 9-2 – Flowchart for decoding a decision

Table 9-52~~9-S29-S3 – Specification of rangeTabLPS depending on pStateIdx and qCodIRangeIdx

pStateIdx	qCodIRangeIdx				pStateIdx	qCodIRangeIdx			
	0	1	2	3		0	1	2	3
0	128	176	208	240	**32**	27	33	39	45
1	128	167	197	227	**33**	26	31	37	43
2	128	158	187	216	**34**	24	30	35	41
3	123	150	178	205	**35**	23	28	33	39
4	116	142	169	195	**36**	22	27	32	37
5	111	135	160	185	**37**	21	26	30	35
6	105	128	152	175	**38**	20	24	29	33
7	100	122	144	166	**39**	19	23	27	31
8	95	116	137	158	**40**	18	22	26	30
9	90	110	130	150	**41**	17	21	25	28
10	85	104	123	142	**42**	16	20	23	27
11	81	99	117	135	**43**	15	19	22	25
12	77	94	111	128	**44**	14	18	21	24

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000470

13	73	89	105	122	**45**	14	17	20	23
14	69	85	100	116	**46**	13	16	19	22
15	66	80	95	110	**47**	12	15	18	21
16	62	76	90	104	**48**	12	14	17	20
17	59	72	86	99	**49**	11	14	16	19
18	56	69	81	94	**50**	11	13	15	18
19	53	65	77	89	**51**	10	12	15	17
20	51	62	73	85	**52**	10	12	14	16
21	48	59	69	80	**53**	9	11	13	15
22	46	56	66	76	**54**	9	11	12	14
23	43	53	63	72	**55**	8	10	12	14
24	41	50	59	69	**56**	8	9	11	13
25	39	48	56	65	**57**	7	9	11	12
26	37	45	54	62	**58**	7	9	10	12
27	35	43	51	59	**59**	7	8	10	11
28	33	41	48	56	**60**	6	8	9	11
29	32	39	46	53	**61**	6	7	9	10
30	30	37	43	50	**62**	6	7	8	9
31	29	35	41	48	**63**	2	2	2	2

Table 9-538-63[illegible]-53 – State transition table

pStateIdx	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
transIdxLPS	0	0	1	2	2	4	4	5	6	7	8	9	9	11	11	12
transIdxMPS	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
pStateIdx	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
transIdxLPS	13	13	15	15	16	16	18	18	19	19	21	21	22	22	23	24
transIdxMPS	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
pStateIdx	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**	**45**	**46**	**47**
transIdxLPS	24	25	26	26	27	27	28	29	29	30	30	30	31	32	32	33
transIdxMPS	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
pStateIdx	**48**	**49**	**50**	**51**	**52**	**53**	**54**	**55**	**56**	**57**	**58**	**59**	**60**	**61**	**62**	**63**
transIdxLPS	33	33	34	34	35	35	35	36	36	36	37	37	37	38	38	63
transIdxMPS	49	50	51	52	53	54	55	56	57	58	59	60	61	62	62	63

9.3.3.2.2 Renormalization process in the arithmetic decoding engine

Inputs to this process are bits from slice data and the variables codIRange and codIOffset.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000471

Outputs of this process are the updated variables codIRange and codIOffset.

A flowchart of the renormalization is shown in Figure 9-3~~Figure 9-3~~. The current value of codIRange is first compared to 256 and further steps are specified as follows.

– If codIRange is greater than or equal to 256, no renormalization is needed and the RenormD process is finished;

– Otherwise (codIRange is less than 256), the renormalization loop is entered. Within this loop, the value of codIRange is doubled, i.e., left-shifted by 1 and a single bit is shifted into codIOffset by using read_bits(1).

The bitstream shall not contain data that result in a value of codIOffset being greater than or equal to codIRange upon completion of this process.

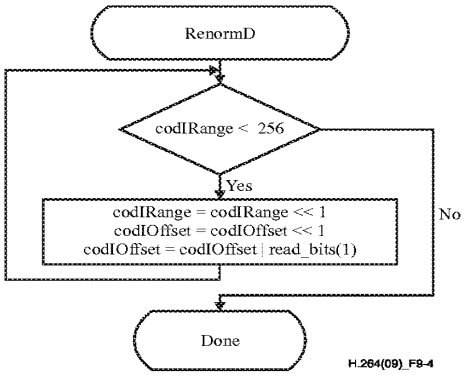


Figure 9-3 – Flowchart of renormalization

9.3.3.2.3 Bypass decoding process for binary decisions

Inputs to this process are bits from slice data and the variables codIRange and codIOffset.

Outputs of this process are the updated variable codIOffset and the decoded value binVal.

The bypass decoding process is invoked when bypassFlag is equal to 1. Figure 9-4~~Figure 9-4~~ shows a flowchart of the corresponding process.

First, the value of codIOffset is doubled, i.e., left-shifted by 1 and a single bit is shifted into codIOffset by using read_bits(1). Then, the value of codIOffset is compared to the value of codIRange and further steps are specified as follows.

– If codIOffset is greater than or equal to codIRange, the variable binVal is set equal to 1 and codIOffset is decremented by codIRange.

– Otherwise (codIOffset is less than codIRange), the variable binVal is set equal to 0.

The bitstream shall not contain data that result in a value of codIOffset being greater than or equal to codIRange upon completion of this process.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000472

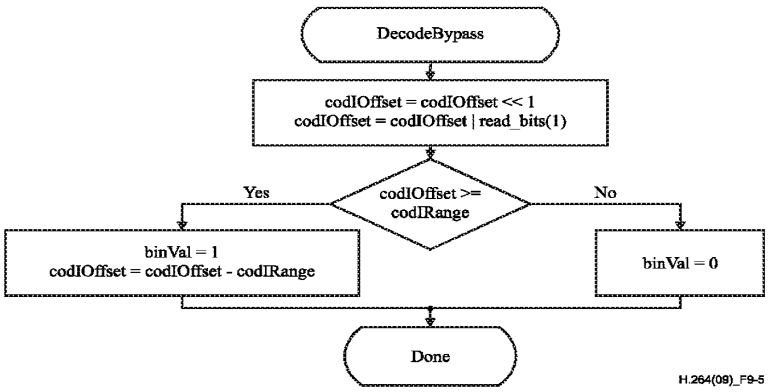


Figure 9-4 – Flowchart of bypass decoding process

9.3.3.2.4 Decoding process for binary decisions before termination

Inputs to this process are bits from slice data and the variables codIRange and codIOffset.

Outputs of this process are the updated variables codIRange and codIOffset, and the decoded value binVal.

This special decoding routine applies to decoding of end_of_slice_flag and pcm_flag corresponding to ctxIdxTable equal to 0 and ctxIdx equal to 0. Figure 9-5~~Figure 9-5~~ shows the flowchart of the corresponding decoding process, which is specified as follows.

First, the value of codIRange is decremented by 2. Then, the value of codIOffset is compared to the value of codIRange and further steps are specified as follows.

– If codIOffset is greater than or equal to codIRange, the variable binVal is set equal to 1, no renormalization is carried out, and CABAC decoding is terminated. The last bit inserted in register codIOffset is equal to 1. When decoding end_of_slice_flag, this last bit inserted in register codIOffset is interpreted as rbsp_stop_one_bit.

– Otherwise (codIOffset is less than codIRange), the variable binVal is set equal to 0 and renormalization is performed as specified in subclause 9.3.3.2.2.

NOTE – This procedure may also be implemented using DecodeDecision(ctxIdxTable, ctxIdx) with ctxIdxTable = 0 and ctxIdx = 0. In the case where the decoded value is equal to 1, seven more bits would be read by DecodeDecision(ctxIdxTable, ctxIdx) and a decoding process would have to adjust its bitstream pointer accordingly to properly decode following syntax elements.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000473

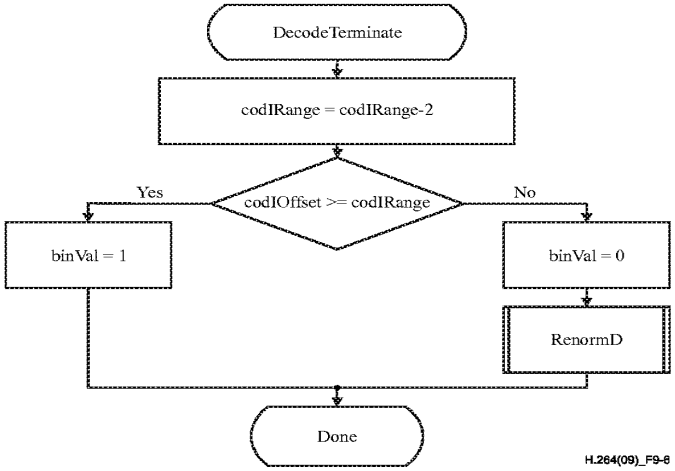


Figure 9-5 – Flowchart of decoding a decision before termination

9.3.4 Arithmetic encoding process (informative)

This subclause does not form an integral part of this Recommendation | International Standard.

Inputs to this process are decisions that are to be encoded and written.

Outputs of this process are bits that are written to the RBSP.

This informative subclause describes an arithmetic encoding engine that matches the arithmetic decoding engine described in subclause 9.3.3.2. The encoding engine is essentially symmetric with the decoding engine, i.e., procedures are called in the same order. The following procedures are described in this section: InitEncoder, EncodeDecision, EncodeBypass, EncodeTerminate, which correspond to InitDecoder, DecodeDecision, DecodeBypass, and DecodeTerminate, respectively. The state of the arithmetic encoding engine is represented by a value of the variable codILow pointing to the lower end of a sub-interval and a value of the variable codIRange specifying the corresponding range of that sub-interval.

9.3.4.1 Initialisation process for the arithmetic encoding engine (informative)

This subclause does not form an integral part of this Recommendation | International Standard.

This process is invoked before encoding the first macroblock of a slice, and after encoding any pcm_alignment_zero_bit and all pcm_sample_luma and pcm_sample_chroma data for a ~~macroblock~~ coding unit of type I_PCM.

Outputs of this process are the values codILow, codIRange, firstBitFlag, bitsOutstanding, and BinCountsInNALunits of the arithmetic encoding engine.

In the initialisation procedure of the encoder, codILow is set equal to 0, and codIRange is set equal to 510. Furthermore, firstBitFlag is set equal to 1 and the counter bitsOutstanding is set equal to 0.

Depending on whether the current slice is the first slice of a coded picture, the following applies.

– If the current slice is the first slice of a coded picture, the counter BinCountsInNALunits is set equal to 0.

– Otherwise (the current slice is not the first slice of a coded picture), the counter BinCountsInNALunits is not modified. The value of BinCountsInNALunits is the result of encoding all the slices of a coded picture that precede the current slice in decoding order. After initialising for the first slice of a coded picture as specified in this subclause, BinCountsInNALunits is incremented as specified in subclauses 9.3.4.2, 9.3.4.4, and 9.3.4.5.

NOTE – The minimum register precision required for storing the values of the variables codILow and codIRange after invocation of any of the arithmetic encoding processes specified in subclauses 9.3.4.2, 9.3.4.4, and 9.3.4.5 is 10 bits and 9 bits, respectively. The encoding process for a binary decision (EncodeDecision) as specified in subclause 9.3.4.2 and the encoding process for a binary decision before termination (EncodeTerminate) as specified in subclause 9.3.4.5 require a minimum register precision of 10 bits for the variable codILow and a minimum register precision of 9 bits for the variable codIRange. The bypass encoding process for binary decisions (EncodeBypass) as specified in subclause 9.3.4.4 requires a minimum register precision of 11 bits for the variable codILow and a minimum register precision of 9 bits for the variable codIRange. The precision required for the

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000474

counters bitsOutstanding and BinCountsInNALunits should be sufficiently large to prevent overflow of the related registers. When maxBinCountInSlice denotes the maximum total number of binary decisions to encode in one slice and maxBinCountInPic denotes the maximum total number of binary decisions to encode a picture, the minimum register precision required for the variables bitsOutstanding and BinCountsInNALunits is given by Ceil(Log2(maxBinCountInSlice + 1)) and Ceil(Log2(maxBinCountInPic + 1)), respectively.

9.3.4.2 Encoding process for a binary decision (informative)

This subclause does not form an integral part of this Recommendation | International Standard.

Inputs to this process are the context index ctxIdx, the value of binVal to be encoded, and the variables codIRange, codILow and BinCountsInNALunits.

Outputs of this process are the variables codIRange, codILow, and BinCountsInNALunits.

Figure 9-6~~Figure 9-6~~ shows the flowchart for encoding a single decision. In a first step, the variable codIRangeLPS is derived as follows.

Given the current value of codIRange, codIRange is mapped to the index qCodIRangeIdx of a quantised value of codIRange by using Equation 9-77~~9-6~~. The value of qCodIRangeIdx and the value of pStateIdx associated with ctxIdx are used to determine the value of the variable rangeTabLPS as specified in Table 9-52~~Table 9-8~~, which is assigned to codIRangeLPS. The value of codIRange − codIRangeLPS is assigned to codIRange.

In a second step, the value of binVal is compared to valMPS associated with ctxIdx. When binVal is different from valMPS, codIRange is added to codILow and codIRange is set equal to the value codIRangeLPS. Given the encoded decision, the state transition is performed as specified in subclause 9.3.3.2.1.1. Depending on the current value of codIRange, renormalization is performed as specified in subclause 9.3.4.3. Finally, the variable BinCountsInNALunits is incremented by 1.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000475

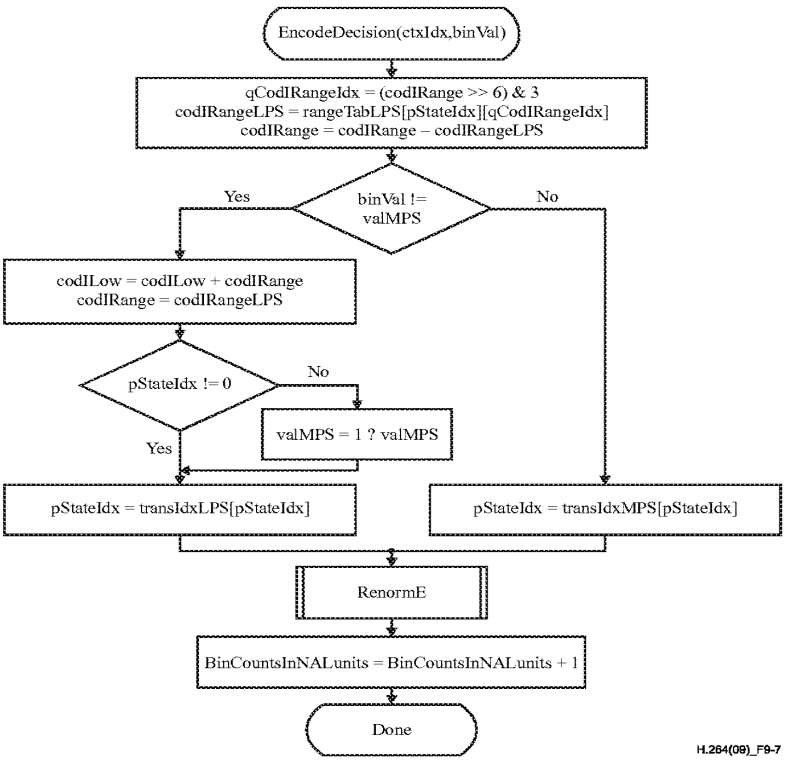


Figure 9-6 – Flowchart for encoding a decision

9.3.4.3 Renormalization process in the arithmetic encoding engine (informative)

This subclause does not form an integral part of this Recommendation | International Standard.

Inputs to this process are the variables codIRange, codILow, firstBitFlag, and bitsOutstanding.

Outputs of this process are zero or more bits written to the RBSP and the updated variables codIRange, codILow, firstBitFlag, and bitsOutstanding.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000476

Renormalization is illustrated in Figure 9-7Figure 9-7.

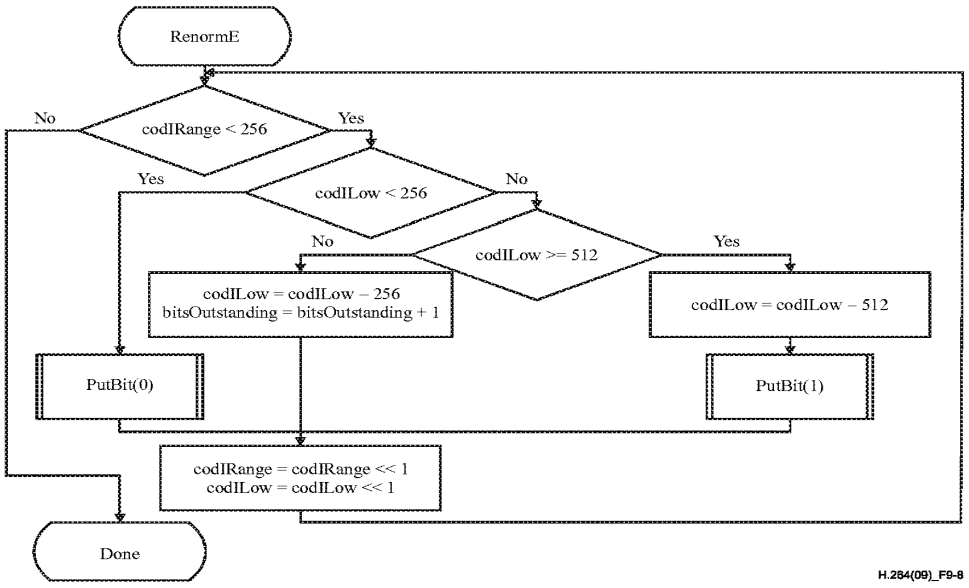


Figure 9-7 – Flowchart of renormalization in the encoder

The PutBit( ) procedure described in Figure 9-8Figure 9-8 provides carry over control. It uses the function WriteBits(B, N) that writes N bits with value B to the bitstream and advances the bitstream pointer by N bit positions. This function assumes the existence of a bitstream pointer with an indication of the position of the next bit to be written to the bitstream by the encoding process.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000477

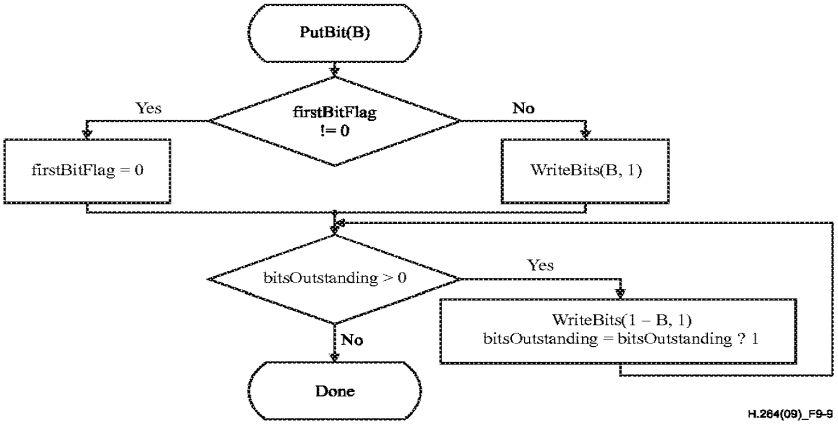


Figure 9-8 – Flowchart of PutBit(B)

9.3.4.4 Bypass encoding process for binary decisions (informative)

This subclause does not form an integral part of this Recommendation | International Standard.

Inputs to this process are the variables binVal, codILow, codIRange, bitsOutstanding, and BinCountsInNALunits.

Output of this process is a bit written to the RBSP and the updated variables codILow, bitsOutstanding, and BinCountsInNALunits.

This encoding process applies to all binary decisions with bypassFlag equal to 1. Renormalization is included in the specification of this process as given in Figure 9-9Figure 9-9.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000478

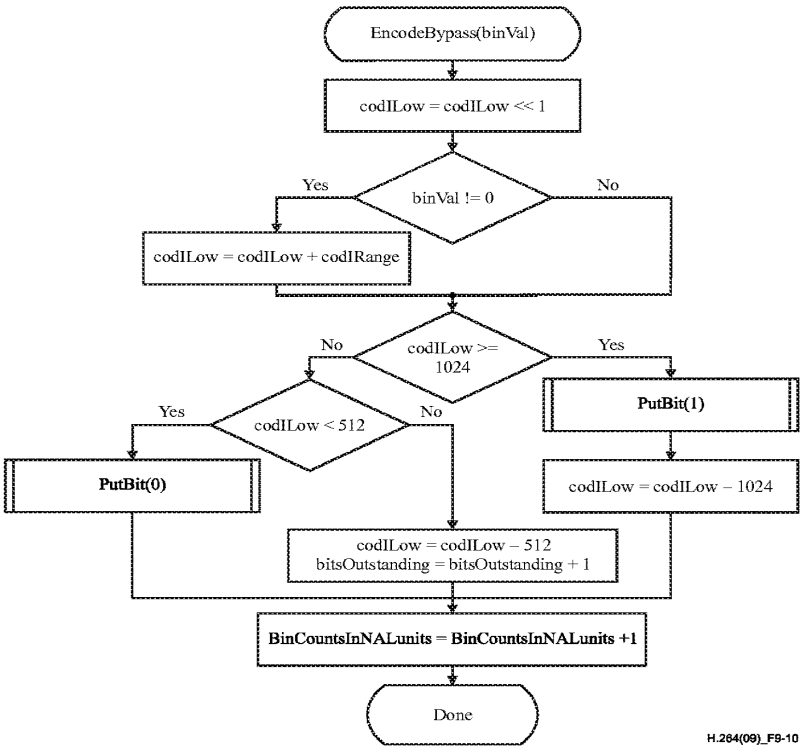


Figure 9-9 – Flowchart of encoding bypass

9.3.4.5 Encoding process for a binary decision before termination (informative)

This subclause does not form an integral part of this Recommendation | International Standard.

Inputs to this process are the variables binVal, codIRange, codILow, bitsOutstanding, and BinCountsInNALunits.

Outputs of this process are zero or more bits written to the RBSP and the updated variables codILow, codIRange, bitsOutstanding, and BinCountsInNALunits.

This encoding routine shown in Figure 9-10~~Figure 9-10~~ applies to encoding of the end_of_slice_flag and pcm_flag ~~of the isia~~ indicating the I_PCM ~~mb_type~~mode both associated with ctxIdx equal to 276.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000479

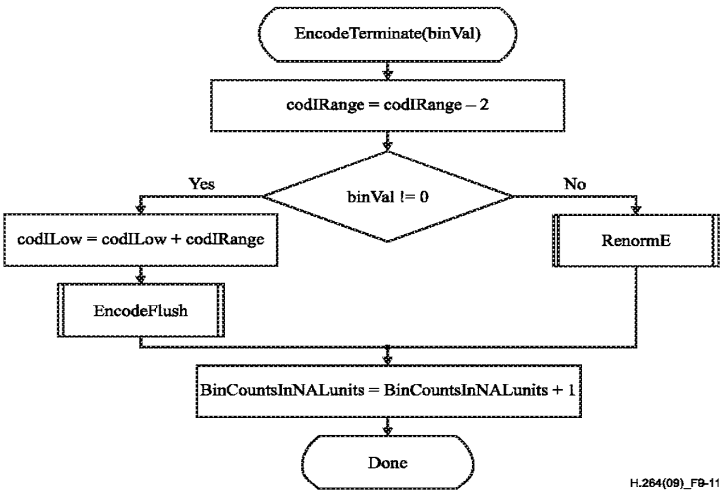


Figure 9-10 – Flowchart of encoding a decision before termination

When the value of binVal to encode is equal to 1, CABAC encoding is terminated and the flushing procedure shown in Figure 9-11~~Figure 9-11~~ is applied. In this flushing procedure, the last bit written by WriteBits(B, N) is equal to 1. When encoding end_of_slice_flag, this last bit is interpreted as the rbsp_stop_one_bit.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000480

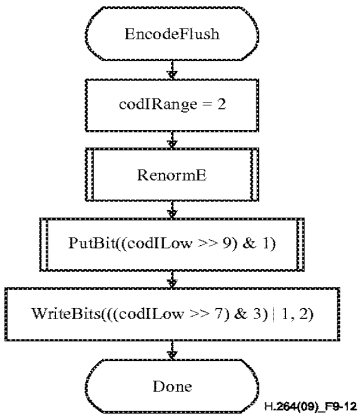


Figure 9-11 – Flowchart of flushing at termination

9.3.4.6 Byte stuffing process (informative)

This subclause does not form an integral part of this Recommendation | International Standard.

This process is invoked after encoding the last macroblock of the last slice of a picture and after encapsulation.

Inputs to this process are the number of bytes NumBytesInVclNALunits of all VCL NAL units of a picture, the number of macroblocks PicSizeInMbs in the picture, and the number of binary symbols BinCountsInNALunits resulting from encoding the contents of all VCL NAL units of the picture.

NOTE – The value of BinCountsInNALunits is the result of encoding all slices of a coded picture. After initialising for the first slice of a coded picture as specified in subclause 9.3.4.1, BinCountsInNALunits is incremented as specified in subclauses 9.3.4.2, 9.3.4.4, and 9.3.4.5.

Outputs of this process are zero or more bytes appended to the NAL unit.

Let the variable k be set equal to Ceil((Ceil(3 * (32 * BinCountsInNALunits − RawMbBits * PicSizeInMbs) ÷ 1024) − NumBytesInVclNALunits) ÷ 3). Depending on the variable k the following applies.

– If k is less than or equal to 0, no cabac_zero_word is appended to the NAL unit.

– Otherwise (k is greater than 0), the 3-byte sequence 0x000003 is appended k times to the NAL unit after encapsulation, where the first two bytes 0x0000 represent a cabac_zero_word and the third byte 0x03 represents an emulation_prevention_three_byte.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000481

Annex A

Profiles and levels

(This annex forms an integral part of this Recommendation | International Standard)

Profiles and levels specify restrictions on bitstreams and hence limits on the capabilities needed to decode the bitstreams. Profiles and levels may also be used to indicate interoperability points between individual decoder implementations.

NOTE 1 – This Recommendation | International Standard does not include individually selectable "options" at the decoder, as this would increase interoperability difficulties.

Each profile specifies a subset of algorithmic features and limits that shall be supported by all decoders conforming to that profile.

NOTE 2 – Encoders are not required to make use of any particular subset of features supported in a profile.

Each level specifies a set of limits on the values that may be taken by the syntax elements of this Recommendation | International Standard. The same set of level definitions is used with all profiles, but individual implementations may support a different level for each supported profile. For any given profile, levels generally correspond to decoder processing load and memory capability.

The profiles that are specified in subclause A.2 are also referred to as the profiles specified in Annex A.

A.1 Requirements on video decoder capability

Capabilities of video decoders conforming to this Recommendation | International Standard are specified in terms of the ability to decode video streams conforming to the constraints of profiles and levels specified in this annex. For each such profile, the level supported for that profile shall also be expressed.

Specific values are specified in this annex for the syntax elements profile_idc and level_idc. All other values of profile_idc and level_idc are reserved for future use by ITU-T | ISO/IEC.

NOTE – Decoders should not infer that when a reserved value of profile_idc or level_idc falls between the values specified in this Recommendation | International Standard that this indicates intermediate capabilities between the specified profiles or levels, as there are no restrictions on the method to be chosen by ITU-T | ISO/IEC for the use of such future reserved values.

A.2 Profiles

All constraints for picture parameter sets that are specified are constraints for picture parameter sets that are activated in the bitstream. All constraints for sequence parameter sets that are specified in subclause A.2.1 are constraints for sequence parameter sets that are activated in the bitstream.

A.2.1 ABC profile

A.3 Levels

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000482

Table A-9-54 – Level limits [Ed. (TW) Kept for convenience]

Level number	Max macroblock processing rate MaxMBPS (MB/s)	Max picture size MaxFS (MBs)	Max decoded picture buffer size MaxDpbMbs (MBs)	Max video bit rate MaxBR (1000 bits/s, 1200 bits/s, cpbBrVclFactor bits/s, or cpbBrNalFactor bits/s)	Max CPB size MaxCPB (1000 bits, 1200 bits, cpbBrVclFactor bits, or cpbBrNalFactor bits)	Vertical MV component range MaxVmvR (luma picture samples)	Min compression ratio MinCR	Max number of motion vectors per two consecutive MBs MaxMvsPer2Mb
1	1 485	99	396	64	175	[−64,+63.75]	2	-
1b	1 485	99	396	128	350	[−64,+63.75]	2	-
1.1	3 000	396	900	192	500	[−128,+127.75]	2	-
1.2	6 000	396	2 376	384	1 000	[−128,+127.75]	2	-
1.3	11 880	396	2 376	768	2 000	[−128,+127.75]	2	-
2	11 880	396	2 376	2 000	2 000	[−128,+127.75]	2	-
2.1	19 800	792	4 752	4 000	4 000	[−256,+255.75]	2	-
2.2	20 250	1 620	8 100	4 000	4 000	[−256,+255.75]	2	-
3	40 500	1 620	8 100	10 000	10 000	[−256,+255.75]	2	32
3.1	108 000	3 600	18 000	14 000	14 000	[−512,+511.75]	4	16
3.2	216 000	5 120	20 480	20 000	20 000	[−512,+511.75]	4	16
4	245 760	8 192	32 768	20 000	25 000	[−512,+511.75]	4	16
4.1	245 760	8 192	32 768	50 000	62 500	[−512,+511.75]	2	16
4.2	522 240	8 704	34 816	50 000	62 500	[−512,+511.75]	2	16
5	589 824	22 080	110 400	135 000	135 000	[−512,+511.75]	2	16
5.1	983 040	36 864	184 320	240 000	240 000	[−512,+511.75]	2	16

Levels with non-integer level numbers in Table A-9-54 are referred to as "intermediate levels".

NOTE 4 – All levels have the same status, but some applications may choose to use only the integer-numbered levels.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000483

Annex B

Byte stream format

(This annex forms an integral part of this Recommendation | International Standard)

This annex specifies syntax and semantics of a byte stream format specified for use by applications that deliver some or all of the NAL unit stream as an ordered stream of bytes or bits within which the locations of NAL unit boundaries need to be identifiable from patterns in the data, such as ITU-T Rec. H.222.0 | ISO/IEC 13818-1 systems or ITU-T Rec. H.320 systems. For bit-oriented delivery, the bit order for the byte stream format is specified to start with the MSB of the first byte, proceed to the LSB of the first byte, followed by the MSB of the second byte, etc.

The byte stream format consists of a sequence of byte stream NAL unit syntax structures. Each byte stream NAL unit syntax structure contains one start code prefix followed by one nal_unit(NumBytesInNALunit) syntax structure. It may (and under some circumstances, it shall) also contain an additional zero_byte syntax element. It may also contain one or more additional trailing_zero_8bits syntax elements. When it is the first byte stream NAL unit in the bitstream, it may also contain one or more additional leading_zero_8bits syntax elements.

B.1 Byte stream NAL unit syntax and semantics

B.1.1 Byte stream NAL unit syntax

byte_stream_nal_unit(NumBytesInNALunit) {	C	Descriptor
while(next_bits(24) != 0x000001 && next_bits(32) != 0x00000001)		
leading_zero_8bits /* equal to 0x00 */		f(8)
if(next_bits(24) != 0x000001)		
zero_byte /* equal to 0x00 */		f(8)
start_code_prefix_one_3bytes /* equal to 0x000001 */		f(24)
nal_unit(NumBytesInNALunit)		
while(more_data_in_byte_stream( ) && next_bits(24) != 0x000001 && next_bits(32) != 0x00000001)		
trailing_zero_8bits /* equal to 0x00 */		f(8)
}		

B.1.2 Byte stream NAL unit semantics

The order of byte stream NAL units in the byte stream shall follow the decoding order of the NAL units contained in the byte stream NAL units (see subclause 7.4.1.2). The content of each byte stream NAL unit is associated with the same access unit as the NAL unit contained in the byte stream NAL unit (see subclause 7.4.1.2.3~~7.4.1.2.3~~).

leading_zero_8bits is a byte equal to 0x00.

NOTE – The leading_zero_8bits syntax element can only be present in the first byte stream NAL unit of the bitstream, because (as shown in the syntax diagram of subclause B.1.1) any bytes equal to 0x00 that follow a NAL unit syntax structure and precede the four-byte sequence 0x00000001 (which is to be interpreted as a zero_byte followed by a start_code_prefix_one_3bytes) will be considered to be trailing_zero_8bits syntax elements that are part of the preceding byte stream NAL unit.

zero_byte is a single byte equal to 0x00.

When any of the following conditions are fulfilled, the zero_byte syntax element shall be present:

- the nal_unit_type within the nal_unit( ) is equal to 7 (sequence parameter set) or 8 (picture parameter set),
- the byte stream NAL unit syntax structure contains the first NAL unit of an access unit in decoding order, as specified by subclause 7.4.1.2.3.

start_code_prefix_one_3bytes is a fixed-value sequence of 3 bytes equal to 0x000001. This syntax element is called a start code prefix.

trailing_zero_8bits is a byte equal to 0x00.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000484

B.2 Byte stream NAL unit decoding process

Input to this process consists of an ordered stream of bytes consisting of a sequence of byte stream NAL unit syntax structures.

Output of this process consists of a sequence of NAL unit syntax structures.

At the beginning of the decoding process, the decoder initialises its current position in the byte stream to the beginning of the byte stream. It then extracts and discards each leading_zero_8bits syntax element (if present), moving the current position in the byte stream forward one byte at a time, until the current position in the byte stream is such that the next four bytes in the bitstream form the four-byte sequence 0x00000001.

The decoder then performs the following step-wise process repeatedly to extract and decode each NAL unit syntax structure in the byte stream until the end of the byte stream has been encountered (as determined by unspecified means) and the last NAL unit in the byte stream has been decoded:

1. When the next four bytes in the bitstream form the four-byte sequence 0x00000001, the next byte in the byte stream (which is a zero_byte syntax element) is extracted and discarded and the current position in the byte stream is set equal to the position of the byte following this discarded byte.
2. The next three-byte sequence in the byte stream (which is a start_code_prefix_one_3bytes) is extracted and discarded and the current position in the byte stream is set equal to the position of the byte following this three-byte sequence.
3. NumBytesInNALunit is set equal to the number of bytes starting with the byte at the current position in the byte stream up to and including the last byte that precedes the location of any of the following conditions:
   - A subsequent byte-aligned three-byte sequence equal to 0x000000,
   - A subsequent byte-aligned three-byte sequence equal to 0x000001,
   - The end of the byte stream, as determined by unspecified means.
4. NumBytesInNALunit bytes are removed from the bitstream and the current position in the byte stream is advanced by NumBytesInNALunit bytes. This sequence of bytes is nal_unit(NumBytesInNALunit) and is decoded using the NAL unit decoding process.
5. When the current position in the byte stream is not at the end of the byte stream (as determined by unspecified means) and the next bytes in the byte stream do not start with a three-byte sequence equal to 0x000001 and the next bytes in the byte stream do not start with a four byte sequence equal to 0x00000001, the decoder extracts and discards each trailing_zero_8bits syntax element, moving the current position in the byte stream forward one byte at a time, until the current position in the byte stream is such that the next bytes in the byte stream form the four-byte sequence 0x00000001 or the end of the byte stream has been encountered (as determined by unspecified means).

B.3 Decoder byte-alignment recovery (informative)

This subclause does not form an integral part of this Recommendation | International Standard.

Many applications provide data to a decoder in a manner that is inherently byte aligned, and thus have no need for the bit-oriented byte alignment detection procedure described in this subclause.

A decoder is said to have byte-alignment with a bitstream when the decoder is able to determine whether or not the positions of data in the bitstream are byte-aligned. When a decoder does not have byte alignment with the encoder's byte stream, the decoder may examine the incoming bitstream for the binary pattern '00000000 00000000 00000000 00000001' (31 consecutive bits equal to 0 followed by a bit equal to 1). The bit immediately following this pattern is the first bit of an aligned byte following a start code prefix. Upon detecting this pattern, the decoder will be byte aligned with the encoder and positioned at the start of a NAL unit in the byte stream.

Once byte aligned with the encoder, the decoder can examine the incoming byte stream for subsequent three-byte sequences 0x000001 and 0x000003.

When the three-byte sequence 0x000001 is detected, this is a start code prefix.

When the three-byte sequence 0x000003 is detected, the third byte (0x03) is an emulation_prevention_three_byte to be discarded as specified in subclause 7.4.1.

When an error in the bitstream syntax is detected (e.g., a non-zero value of the forbidden_zero_bit or one of the three-byte or four-byte sequences that are prohibited in subclause 7.4.1), the decoder may consider the detected condition as an indication that byte alignment may have been lost and may discard all bitstream data until the detection of byte alignment at a later position in the bitstream as described in this subclause.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000485

Annex C

Hypothetical reference decoder

(This annex forms an integral part of this Recommendation | International Standard)

This annex specifies the hypothetical reference decoder (HRD) and its use to check bitstream and decoder conformance.

Two types of bitstreams are subject to HRD conformance checking for this Recommendation | International Standard. The first such type of bitstream, called Type I bitstream, is a NAL unit stream containing only the VCL NAL units and filler data NAL units for all access units in the bitstream. The second type of bitstream, called a Type II bitstream, contains, in addition to the VCL NAL units and filler data NAL units for all access units in the bitstream, at least one of the following:

– additional non-VCL NAL units other than filler data NAL units,

– all leading_zero_8bits, zero_byte, start_code_prefix_one_3bytes, and trailing_zero_8bits syntax elements that form a byte stream from the NAL unit stream (as specified in Annex B).

Figure C-9-12~~Figure C-9-12~~ shows the types of bitstream conformance points checked by the HRD.

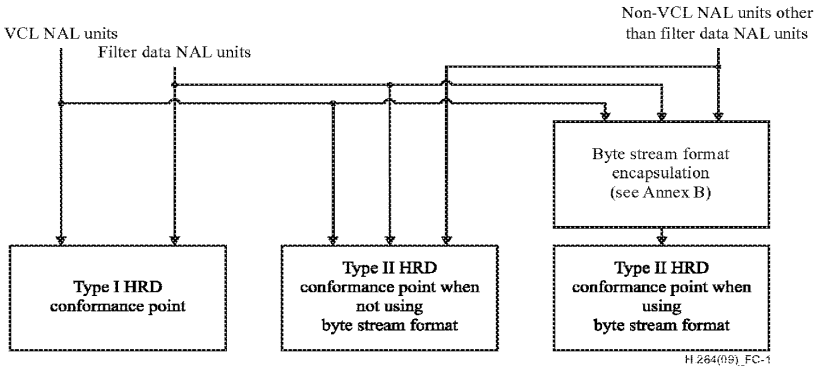


Figure C-9-12 – Structure of byte streams and NAL unit streams for HRD conformance checks

The syntax elements of non-VCL NAL units (or their default values for some of the syntax elements), required for the HRD, are specified in the semantic subclauses of clause 6.67, Annexes D and E.

The HRD contains a coded picture buffer (CPB), an instantaneous decoding process, a decoded picture buffer (DPB), and output cropping as shown in Figure C-9-13~~Figure C-9-13~~.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000486

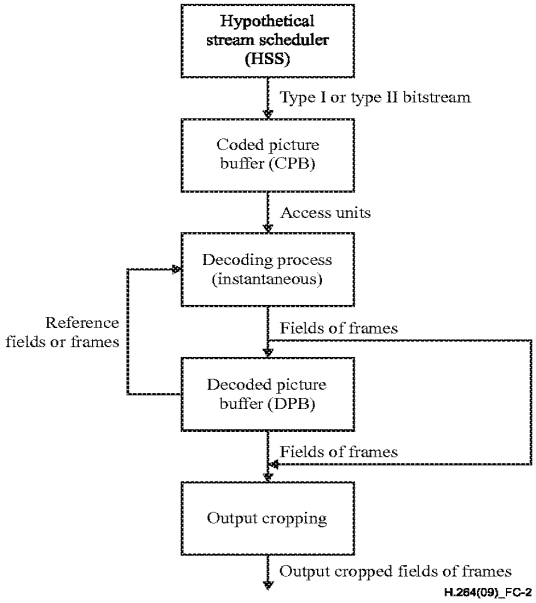


Figure C-9-13 – HRD buffer model

C.1 Operation of coded picture buffer (CPB)

The specifications in this subclause apply independently to each set of CPB parameters that is present and to both the Type I and Type II conformance points shown in Figure C-9-12~~Figure C-9-13~~.

C.1.1 Timing of bitstream arrival

C.1.2 Timing of coded picture removal

C.2 Operation of the decoded picture buffer (DPB)

C.3 Bitstream conformance

C.4 Decoder conformance

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000487

Annex D

Supplemental enhancement information

(This annex forms an integral part of this Recommendation | International Standard)

This annex specifies syntax and semantics for SEI message payloads.

SEI messages assist in processes related to decoding, display or other purposes. However, SEI messages are not required for constructing the luma or chroma samples by the decoding process. Conforming decoders are not required to process this information for output order conformance to this Recommendation | International Standard (see Annex 0 for the specification of conformance). Some SEI message information is required to check bitstream conformance and for output timing decoder conformance.

In Annex D, specification for presence of SEI messages are also satisfied when those messages (or some subset of them) are conveyed to decoders (or to the HRD) by other means not specified by this Recommendation | International Standard. When present in the bitstream, SEI messages shall obey the syntax and semantics specified in subclauses 7.3.2.3 and 0 and this annex. When the content of an SEI message is conveyed for the application by some means other than presence within the bitstream, the representation of the content of the SEI message is not required to use the same syntax specified in this annex. For the purpose of counting bits, only the appropriate bits that are actually present in the bitstream are counted.

D.1 SEI payload syntax

[Ed. (TW): insert table]

D.2 SEI payload semantics

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000488

Annex E

Video usability information

(This annex forms an integral part of this Recommendation | International Standard)

This annex specifies syntax and semantics of the VUI parameters of the sequence parameter sets.

VUI parameters are not required for constructing the luma or chroma samples by the decoding process. Conforming decoders are not required to process this information for output order conformance to this Recommendation | International Standard (see Annex 0 for the specification of conformance). Some VUI parameters are required to check bitstream conformance and for output timing decoder conformance.

In Annex E, specification for presence of VUI parameters is also satisfied when those parameters (or some subset of them) are conveyed to decoders (or to the HRD) by other means not specified by this Recommendation | International Standard. When present in the bitstream, VUI parameters shall follow the syntax and semantics specified in subclauses 7.3.2.1 and 7.4.2.1 and this annex. When the content of VUI parameters is conveyed for the application by some means other than presence within the bitstream, the representation of the content of the VUI parameters is not required to use the same syntax specified in this annex. For the purpose of counting bits, only the appropriate bits that are actually present in the bitstream are counted.

E.1 VUI syntax

E.1.1 VUI parameters syntax

[Ed. (TW): insert table]

E.2 VUI semantics

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000489

페이지 77: [1] 서식 있음 JCTVC-E321 2011-06-22 PM 7:11:00

글꼴: 굵게

페이지 77: [2] 서식 있음 JCTVC-E321 2011-06-22 PM 7:13:00

글꼴: 굵게

페이지 77: [3] 서식 있음 JCTVC-E321 2011-06-24 PM 5:50:00

글꼴: (한글) 맑은 고딕

페이지 77: [4] 서식 있음 JCTVC-E321 2011-06-24 PM 5:52:00

글머리 기호 + 수준:1 + 맞춤 위치: 0 cm + 탭 간격: 0.69 cm + 들여쓰기 위치: 0.69 cm

페이지 77: [5] 서식 있음 JCTVC-E321 2011-06-24 PM 5:52:00

글꼴: (한글) 맑은 고딕

페이지 77: [6] 서식 있음 JCTVC-E321 2011-06-24 PM 5:52:00

들여쓰기: 왼쪽: 0.81 cm, 글머리 기호 + 수준:1 + 맞춤 위치: 0 cm + 탭 간격: 0.69 cm + 들여쓰기 위치: 0.69 cm, 탭: 4.26 글자, 탭 목록 + 1.95 글자(없음)

페이지 77: [7] 서식 있음 JCTVC-E321 2011-06-24 PM 5:57:00

글꼴: (한글) 맑은 고딕, 영어(영국)

페이지 77: [8] 서식 있음 JCTVC-E321 2011-06-24 PM 5:59:00

영어(영국)

페이지 77: [9] 서식 있음 JCTVC-E321 2011-06-24 PM 5:59:00

영어(영국)

페이지 77: [10] 서식 있음 JCTVC-E321 2011-06-24 PM 5:59:00

영어(영국)

페이지 77: [11] 서식 있음 JCTVC-E321 2011-06-24 PM 5:59:00

영어(영국)

페이지 77: [12] 서식 있음 JCTVC-E321 2011-06-24 PM 5:59:00

영어(영국)

페이지 77: [13] 서식 있음 JCTVC-E321 2011-06-24 PM 5:59:00

영어(영국)

페이지 77: [14] 서식 있음 JCTVC-E321 2011-06-24 PM 5:59:00

영어(영국)

페이지 77: [15] 서식 있음 **JCTVC-E321** **2011-06-24 PM 5:59:00**

영어(영국)

페이지 77: [16] 서식 있음 **JCTVC-E321** **2011-06-24 PM 5:59:00**

영어(영국)

페이지 77: [17] 서식 있음 **JCTVC-E321** **2011-06-24 PM 5:59:00**

영어(영국)

페이지 77: [18] 서식 있음 **JCTVC-E321** **2011-06-24 PM 5:59:00**

영어(영국)

페이지 77: [19] 서식 있음 **JCTVC-E321** **2011-06-24 PM 5:59:00**

영어(영국)

페이지 77: [20] 서식 있음 **JCTVC-E321** **2011-06-24 PM 5:59:00**

영어(영국)

페이지 77: [21] 서식 있음 **JCTVC-E321** **2011-06-24 PM 5:59:00**

영어(영국)

페이지 77: [22] 서식 있음 **JCTVC-E321** **2011-06-24 PM 5:59:00**

영어(영국)

페이지 77: [23] 서식 있음 **JCTVC-E321** **2011-06-24 PM 5:59:00**

영어(영국)

페이지 77: [24] 서식 있음 **JCTVC-E321** **2011-06-24 PM 5:59:00**

영어(영국)

페이지 77: [25] 서식 있음 **JCTVC-E321** **2011-06-24 PM 5:59:00**

영어(영국)

페이지 77: [26] 서식 있음 **JCTVC-E321** **2011-06-24 PM 5:59:00**

영어(영국)

페이지 77: [27] 서식 있음 **JCTVC-E321** **2011-06-24 PM 6:00:00**

영어(영국)

탭: 5.67 글자, 왼쪽 + 5.96 글자(없음)

영어(영국)

영어(영국)

영어(영국)

영어(영국)

글꼴: (한글) 바탕, (한글) 한국어

들여쓰기: 왼쪽: 1.31 cm, 글머리 기호 + 수준:1 + 맞춤 위치: 0 cm + 탭 간격: 0.69 cm + 들여쓰기 위치: 0.69 cm, 탭: 5.67 글자, 왼쪽 + 1.95 글자(없음) + 3.97 글자 + 5.96 글자

영어(영국)

영어(영국)

영어(영국)

영어(영국)

영어(영국)

영어(영국)

영어(영국)

페이지 77: [42] 서식 있음	JCTVC-E321	2011-06-24 PM 6:00:00

영어(영국)

페이지 77: [43] 서식 있음	JCTVC-E321	2011-06-24 PM 6:00:00

영어(영국)

페이지 77: [44] 서식 있음	wjhan-Samsung	2011-06-27 AM 11:26:00

탭: 5.67 글자, 왼쪽 + 3.97 글자(없음) + 5.96 글자

페이지 77: [45] 서식 있음	JCTVC-E321	2011-06-24 PM 6:09:00

글꼴: (한글) 맑은 고딕

페이지 77: [46] 서식 있음	JCTVC-E321	2011-06-24 PM 6:09:00

글머리 기호 + 수준:1 + 맞춤 위치: 0 cm + 탭 간격: 0.69 cm + 들여쓰기 위치: 0.69 cm

페이지 104: [47] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

영어(영국)

페이지 104: [47] 서식 있음 JCTVC-E069 2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음 JCTVC-E069 2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음 JCTVC-E069 2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음 JCTVC-E069 2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음 JCTVC-E069 2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음 JCTVC-E069 2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음 JCTVC-E069 2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음 JCTVC-E069 2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음 JCTVC-E069 2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음 JCTVC-E069 2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음 JCTVC-E069 2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [47] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [48] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [48] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [49] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [49] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [49] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [49] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [49] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [49] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [49] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [49] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [49] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [49] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [49] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [49] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [49] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [49] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [49] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [49] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [49] 서식 있음	JCTVC-E069	2011-06-27 PM 1:32:00

영어(영국)

영어(영국)

페이지 104: [49] 서식 있음 JCTVC-E069 2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [49] 서식 있음 JCTVC-E069 2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [49] 서식 있음 JCTVC-E069 2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [49] 서식 있음 JCTVC-E069 2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [49] 서식 있음 JCTVC-E069 2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [49] 서식 있음 JCTVC-E069 2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [49] 서식 있음 JCTVC-E069 2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [49] 서식 있음 JCTVC-E069 2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [49] 서식 있음 JCTVC-E069 2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [49] 서식 있음 JCTVC-E069 2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [49] 서식 있음 JCTVC-E069 2011-06-27 PM 1:32:00

영어(영국)

영어(영국)

페이지 104: [49] 서식 있음 JCTVC-E069 2011-06-27 PM 1:32:00
영어(영국)
페이지 104: [49] 서식 있음 JCTVC-E069 2011-06-27 PM 1:32:00
영어(영국)
페이지 104: [49] 서식 있음 JCTVC-E069 2011-06-27 PM 1:32:00
영어(영국)
페이지 104: [49] 서식 있음 JCTVC-E069 2011-06-27 PM 1:32:00
영어(영국)
페이지 104: [49] 서식 있음 JCTVC-E069 2011-06-27 PM 1:32:00
영어(영국)
페이지 104: [49] 서식 있음 JCTVC-E069 2011-06-27 PM 1:32:00
영어(영국)
페이지 104: [49] 서식 있음 JCTVC-E069 2011-06-27 PM 1:32:00
영어(영국)
페이지 104: [49] 서식 있음 JCTVC-E069 2011-06-27 PM 1:32:00
영어(영국)
페이지 104: [49] 서식 있음 JCTVC-E069 2011-06-27 PM 1:32:00
영어(영국)
페이지 104: [49] 서식 있음 JCTVC-E069 2011-06-27 PM 1:32:00
영어(영국)
페이지 104: [49] 서식 있음 JCTVC-E069 2011-06-27 PM 1:32:00
영어(영국)
페이지 104: [49] 서식 있음 JCTVC-E069 2011-06-27 PM 1:32:00
영어(영국)
페이지 104: [49] 서식 있음 JCTVC-E069 2011-06-27 PM 1:32:00

영어(영국)

페이지 104: [49] 서식 있음 **JCTVC-E069** **2011-06-27 PM 1:32:00**

영어(영국)

페이지 104: [49] 서식 있음 **JCTVC-E069** **2011-06-27 PM 1:32:00**

영어(영국)

페이지 104: [49] 서식 있음 **JCTVC-E069** **2011-06-27 PM 1:32:00**

영어(영국)

페이지 104: [49] 서식 있음 **JCTVC-E069** **2011-06-27 PM 1:32:00**

영어(영국)

페이지 104: [49] 서식 있음 **JCTVC-E069** **2011-06-27 PM 1:32:00**

영어(영국)

페이지 104: [49] 서식 있음 **JCTVC-E069** **2011-06-27 PM 1:32:00**

영어(영국)

페이지 104: [49] 서식 있음 **JCTVC-E069** **2011-06-27 PM 1:32:00**

영어(영국)

페이지 104: [49] 서식 있음 **JCTVC-E069** **2011-06-27 PM 1:32:00**

영어(영국)

페이지 104: [49] 서식 있음 **JCTVC-E069** **2011-06-27 PM 1:32:00**

영어(영국)

페이지 104: [49] 서식 있음 **JCTVC-E069** **2011-06-27 PM 1:32:00**

영어(영국)

페이지 104: [49] 서식 있음 **JCTVC-E069** **2011-06-27 PM 1:32:00**

영어(영국)

페이지 104: [49] 서식 있음 **JCTVC-E069** **2011-06-27 PM 1:32:00**

영어(영국)

영어(영국)

페이지 107: [50] 서식 있음 JCTVC-E321 2011-06-22 PM 8:52:00
영어(영국)

페이지 107: [50] 서식 있음 JCTVC-E321 2011-06-22 PM 8:52:00
영어(영국)

페이지 107: [50] 서식 있음 JCTVC-E321 2011-06-22 PM 8:52:00
영어(영국)

페이지 107: [50] 서식 있음 JCTVC-E321 2011-06-22 PM 8:52:00
영어(영국)

페이지 107: [50] 서식 있음 JCTVC-E321 2011-06-22 PM 8:52:00
영어(영국)

페이지 107: [50] 서식 있음 JCTVC-E321 2011-06-22 PM 8:52:00
영어(영국)

페이지 107: [50] 서식 있음 JCTVC-E321 2011-06-22 PM 8:52:00
영어(영국)

페이지 107: [50] 서식 있음 JCTVC-E321 2011-06-22 PM 8:52:00
영어(영국)

페이지 107: [50] 서식 있음 JCTVC-E321 2011-06-22 PM 8:52:00
영어(영국)

페이지 107: [50] 서식 있음 JCTVC-E321 2011-06-22 PM 8:52:00
영어(영국)

페이지 107: [50] 서식 있음 JCTVC-E321 2011-06-22 PM 8:52:00
영어(영국)

페이지 107: [50] 서식 있음 JCTVC-E321 2011-06-22 PM 8:52:00
영어(영국)

페이지 107: [50] 서식 있음 JCTVC-E321 2011-06-22 PM 8:52:00
영어(영국)

페이지 107: [50] 서식 있음 JCTVC-E321 2011-06-22 PM 8:52:00

영어(영국)

페이지 107: [50] 서식 있음 **JCTVC-E321** **2011-06-22 PM 8:52:00**

영어(영국)

페이지 107: [51] 서식 있음 **JCTVC-E321** **2011-06-22 PM 8:52:00**

영어(영국)

페이지 107: [51] 서식 있음 **JCTVC-E321** **2011-06-22 PM 8:52:00**

영어(영국)

페이지 107: [51] 서식 있음 **JCTVC-E321** **2011-06-22 PM 8:52:00**

영어(영국)

페이지 107: [51] 서식 있음 **JCTVC-E321** **2011-06-22 PM 8:52:00**

영어(영국)

페이지 107: [51] 서식 있음 **JCTVC-E321** **2011-06-22 PM 8:52:00**

영어(영국)

페이지 107: [51] 서식 있음 **JCTVC-E321** **2011-06-22 PM 8:52:00**

영어(영국)

페이지 107: [51] 서식 있음 **JCTVC-E321** **2011-06-22 PM 8:52:00**

영어(영국)

페이지 107: [51] 서식 있음 **JCTVC-E321** **2011-06-22 PM 8:52:00**

영어(영국)

페이지 107: [51] 서식 있음 **JCTVC-E321** **2011-06-22 PM 8:52:00**

영어(영국)

페이지 107: [51] 서식 있음 **JCTVC-E321** **2011-06-22 PM 8:52:00**

영어(영국)

페이지 107: [51] 서식 있음 **JCTVC-E321** **2011-06-22 PM 8:52:00**

영어(영국)

페이지 107: [51] 서식 있음 JCTVC-E321 2011-06-22 PM 8:52:00

영어(영국)

페이지 107: [51] 서식 있음 JCTVC-E321 2011-06-22 PM 8:52:00

영어(영국)

페이지 107: [51] 서식 있음 JCTVC-E321 2011-06-22 PM 8:52:00

영어(영국)

페이지 107: [52] 서식 있음 JCTVC-E321 2011-06-22 PM 8:52:00

영어(영국)

페이지 107: [52] 서식 있음 JCTVC-E321 2011-06-22 PM 8:52:00

영어(영국)

페이지 107: [52] 서식 있음 JCTVC-E321 2011-06-22 PM 8:52:00

영어(영국)

페이지 107: [52] 서식 있음 JCTVC-E321 2011-06-22 PM 8:52:00

영어(영국)

페이지 107: [52] 서식 있음 JCTVC-E321 2011-06-22 PM 8:52:00

영어(영국)

페이지 107: [52] 서식 있음 JCTVC-E321 2011-06-22 PM 8:52:00

영어(영국)

페이지 107: [52] 서식 있음 JCTVC-E321 2011-06-22 PM 8:52:00

영어(영국)

페이지 107: [52] 서식 있음 JCTVC-E321 2011-06-22 PM 8:52:00

영어(영국)

페이지 107: [52] 서식 있음 JCTVC-E321 2011-06-22 PM 8:52:00

영어(영국)

페이지 107: [52] 서식 있음 JCTVC-E321 2011-06-22 PM 8:52:00

영어(영국)

페이지 107: [52] 서식 있음	JCTVC-E321	2011-06-22 PM 8:52:00

영어(영국)

페이지 107: [52] 서식 있음	JCTVC-E321	2011-06-22 PM 8:52:00

영어(영국)

페이지 107: [53] 서식 있음	JCTVC-E321	2011-06-22 PM 8:52:00

영어(영국)

페이지 107: [53] 서식 있음	JCTVC-E321	2011-06-22 PM 8:52:00

영어(영국)

페이지 107: [53] 서식 있음	JCTVC-E321	2011-06-22 PM 8:52:00

영어(영국)

페이지 107: [53] 서식 있음	JCTVC-E321	2011-06-22 PM 8:52:00

영어(영국)

페이지 107: [54] 서식 있음	JCTVC-E266	2011-06-23 AM 10:55:00

아래 첨자

페이지 107: [54] 서식 있음	JCTVC-E266	2011-06-23 AM 10:55:00

아래 첨자

페이지 107: [55] 서식 있음	JCTVC-E266	2011-06-23 AM 10:57:00

아래 첨자

페이지 107: [55] 서식 있음	JCTVC-E266	2011-06-23 AM 10:57:00

아래 첨자

페이지 107: [55] 서식 있음	JCTVC-E266	2011-06-23 AM 10:57:00

아래 첨자

Joint Collaborative Team on Video Coding (JCT-VC)
of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11
6th Meeting: Torino, IT, 14-22 July, 2011

Document: JCTVC-F803_d65

Title:	**WD4: Working Draft 4 of High-Efficiency Video Coding**		
Status:	Output Document of JCT-VC		
Purpose:	Working Draft of HEVC		
Author(s) or Contact(s):	Benjamin Bross Fraunhofer HHI	Email:	benjamin.bross@hhi.fraunhofer.de
	Woo-Jin Han Samsung Electronics	Email:	wjhan.han@samsung.com
	Jens-Rainer Ohm RWTH Aachen	Email:	ohm@ient.rwth-aachen.de
	Gary J. Sullivan Microsoft	Email:	garysull@microsoft.com
	Thomas Wiegand Fraunhofer HHI / TU Berlin	Email:	thomas.wiegand@hhi.fraunhofer.de
Source:	Editor		

Abstract

Working Draft 4 of High-Efficiency Video Coding.

Ed. Notes (WD4):

- Removed inferred merge (JCTVC-F082)
- Incorporated slice header flag to disable 4x4 inter partitions (JCTVC-F744)
- Incorporated modified rounding in MV scaling (JCTVC-F142)
- Removed intermediate amvp spatial candidates redundancy check (JCTVC-F050)
- Incorporated reducing the number of spatial mv scalings to 1 (JCTVC-F088)
- Incorporated spatial merge candidate positions unification (JCTVC-F419)
- Incorporated one reference list check for temporal mvp (JCTVC-F587)
- Incorporated AMVP/merge parsing robustness with simplifications (JCTVC-F470)
- Incorporated unified availability check for intra (JCTVC-F477)
- Incorporated generic interpolation filter (JCTVC-F537)
- Incorporated non-square quadtree transform NSQT (JCTVC-F412)
- Incorporated asymmetric motion partitions AMP (JCTVC-F379)
- Incorporated CBF redundancy reduction (JCTVC-C277)
- Incorporated modified delta QP binarization (JCTVC-F745)
- Incorporated diagonal coefficient scanning in CABAC (JCTVC-F129)
- Incorporated parallel context processing for coefficient levels in CABAC (JCTVC-F130)
- Incorporated context sharing for significant_coeff_flag of 16x16 and 32x32 transforms (JCTVC-F132)
- Incorporated unified scans (JCTVC-F288)
- Incorporated sample adaptive offset (JCTVC-E049)
- Incorporated sample adaptive offset for chroma (JCTVC-F057)
- Incorporated sample adaptive offset offset accuracy (JCTVC-F396)
- Incorporated updated ALF slice padding (JCTVC-D128)
- Incorporated updated ALF slice padding due to ALF filter shape change (JCTVC-F303/F042)
- Incorporated updated ALF slice padding due to unified luma and chroma filter shapes (JCTVC-F157)
- Incorporated ALF filter using subset of pixels (JCTVC-F301)
- Incorporated modified deblocking process for luma (JCTVC-F118)
- Incorporated modified tc_offset in deblocking process (JCTVC-F143)
- Incorporated MDIS and pixel position change of planar (JCTVC-F483)
- Incorporated availability check removal for intra DC filtering (JCTVC-F178)
- Incorporated size-independent intra DC filtering (JCTVC-F252)

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000506

- Incorporated modified MDIS table (JCTVC-F126)
- Incorporated simplified intra_FromLuma prediction (JCTVC-F760)
- Incorporated fixed number of MPM (JCTVC-F765)
- Incorporated SAO boundary processing (JCTVC-F232)
- Minor bug in deriving sample positions in SAO process was fixed
- Bug in coding tree syntax table related to the initialization of variable IsCuQpDeltaCoded was fixed
- Incorporated modified last significant coefficient position coding in CABAC (JCTVC-F375)
- Incorporated modified mvd coding in CABAC (JCTVC-F455)
- Incorporated reduced number of contexts in CABAC (JCTVC-F746)
- Incorporated high-level syntax cleanup (JCTVC-F714)
- Incorporated NAL unit type and CDR (CRA) (JCTVC-F803/463)
- Incorporated adaptation parameter set (APS) (JCTVC-F747)

Ed. Notes (WD3):

- Added Residual coding CABAC syntax and semantics
- Added Zig-zag scanning process
- Added CABAC Binarization processes
- Incorporated MV coding (JCTVC-E481)
- Incorporated Compression of reference indices (JCTVC-E059)
- Incorporated Zero merge candidate (JCTVC-E146)
- Incorporated Intra mode coding (JCTVC-E088/E131) (Inserted by TK 31/3/2011 with notes)
- Fixed the CABAC coefficients syntax, semantics and inverse scanning process
- Incorporated CABAC coeffs (JCTVC-E253)
- Moved the EGk binarization from the UEGk subclause in a separate subclause
- Added text representing CABAC entropy coding context initialisation
- Added text representing CABAC entropy coding context derivation.
- Mode-dependent 3- scan for intra (JCTVC-D393)
- Incorporated CABAC: Context size reduction (JCTVC-E227/E489)
- Incorporated CABAC: significance map coding simplification (JCTVC-E227/E338/E344/E494)
- Incorporated CABAC: Contexts for MVD (JCTVC-E324)
- Incorporated initial draft of CAVLC text
- CAVLC for 16x16 & 32x32 (JCTVC-E383)
- CAVLC table size reduction (JCTVC-E384)
- CAVLC for RQT (JCTVC-E404)
- CAVLC: counters (JCTVC-E143)
- CAVLC: Intra prediction mode coding in LCEC (JCTVC-D366)
- CAVLC: Inter prediction mode coding in LCEC (JCTVC- D370)
- CAVLC: 4x4 and 8x8 transform coefficient coding in LCEC (JCTVC- D374)
- Block-based ALF (JCTVC-E046/E323)
- ALF parameters to PPS (JCTVC-E045)
- Parallel deblocking (JCTVC-E496/E181/E224)
- Clipping for bi-pred averaging (JCTVC-E242)
- Reference sample padding (JCTVC-E488)
- Transformation processes are replaced by [TBD] mark (meeting note, JCTVC-E243)
- Sub-LCU-level dQP (JCTVC-E051/E220)
- Temporal layer switching and reference list management based on temporal_id (JCTVC-E279/D081/D200)
- Improved text of entropy slice (JCTVC-D070) [Ed. (WJ): the term entropy slice is replaced by lightweight slice according to proponent's text]
- Slice independent deblocking and adaptive loop filtering (JCTVC-D128)
- Fine-granularity slices (JCTVC-E483)
- PCM mode (JCTVC-E057 and JCTVC-E192)
- CAVLC: Inter pred coding (JCTVC-E381)
- CAVLC: Combined coding of inter prediction direction and reference frame index (JCTVC-D141)
- 4x4 DST (JCTVC-E125)
- Planar mode (JCTVC-E321)
- Luma-based chroma intra prediction (JCTVC-E266)
- Modification of DC predictor (JCTVC-E069)
- Bug in mapping table and the corresponding text for mostProbableIntra was fixed. (64x64 uses 3-directions, but the table was specified for 5-directions)

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000507

- Non-existing cases of Intra_DC, Intra_Planar and Intra_FromLuma are removed (due to reference sample padding, JCTVC-E488)

Ed. Notes (WD2):
- Incorporated Partial Merging according to JCTVC-D441
  - removed direct mode
  - moved merge to prediction_unit and added candidates
  - added partial merge restrictions
  - inter NxN partitioning only for smallest coding_unit
- Updated transform_tree and transform_coeff syntax
- Added transform_coeff to coding_unit syntax (Fix)
- Incorporated intra NxN partitioning only for smallest coding_unit according to JCTVC-D432
- Incorporated modified temporal motion vector predition according to JCTVC-D164
- Incorporated simplified motion vector prediction according to JCTVC-D231
  - removed median
  - removed pruning process
  - changed the selection manner of left/top predictor
- 8-tap luma interpolation filter according to JCTVC-D344
- 4-tap chroma interpolation filter according to JCTVC-D347
- Improved deblocking filter text according to JCTVC-D395
- IBDI syntax is removed
- Updated syntax and semantics
  - Two tool-enabling flags (adaptive_loop_filter_enabled_flag and cu_qp_delta_enabled_flag) are added in SPS according to software. However, low_delay_coding_enabled_flag is not added – it could be handled by more general reference management scheme. merging_enabled_flag is not added – partial merging (JCTVC-D441) was adopted thus merging cannot be turned off any more. amvp_mode[] is not added since amvp cannot be turned off any more due to absence of median predictor (JCTVC-D231). Note that software has all switches.
  - cu_qp_delta (coding unit layer), syntax and semantics are added. (JCTVC-D258)
  - collocated_from_l0 (slice header), syntax and semantics are added.
- Clean decoding refresh (CDR) (JCTVC-D234).
- Temporal motion vector memory compression (JCTVC-D072)
- Constrained intra prediction (JCTVC-D086)
- Mode-dependent intra smoothing (JCTVC-D282)
- Merging chroma intra prediction process into luma intra prediction process
- Combined reference list (JCTVC-D421)
- Chroma intra prediction mode reordering (JCTVC-D255/D278/D166)
- Adaptive loop filter text is added
- Entropy slice is added (JCTVC-D070)
- High precision bi-directional averaging (JCTVC-D321)
- Reduction of number of intra prediction modes for 64x64 blocks (JCTVC-D100)
- Misc.
  - TPE bits are reduced from 4 to 2
  - Clipping is applied to (temporally) scaled mv – revisit

Ed. Notes (WD1):
- Incorporated the decisions on high-level syntax according to JCTVC-B121
- Incorporated text from JCTVC-B205revision7
- Incorporated text from JCTVC-C319 (as found to be stable)
- Revised coding tree, coding unit and prediction unit syntaxes (coding tree syntax is newly added. needs to be confirmed)
- Initial drafting of decoding process of coding units in intra prediction mode (luma part, JCTVC-B100 and JCTVC-C042)
- Initial drafting of decoding process of coding units in inter prediction mode
- Initial drafting of scaling and transformation process
- Added text, transform 16T and 32T
- Initial drafting of deblocking process
- Improving the text, derivation process for motion vector components and reference indices
- Added text, boundary filtering strength

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000508

Open issues:

- Should support for monochrome, 4:2:2 and 4:4:4 (with and w/o separate color planes) be included from the start? Currently, it has been left in the text as it doesn't seem to affect much text.
- Handling of the term "frame". We are currently leaning towards changing all occurrences of "frame" to "picture" (removed or marked all occurrences of "field")
- Large size table (de-scaling matrices) is not inserted yet.
- Slice-header syntaxes and their semantics are not completed yet. Also possible modifications that maybe necessary because of larger treeblocks (64x64) compared to macroblocks (16x16) are not yet considered.
- Clipping is applied to temporally scaled motion vector in 8.4.2.1.7. Do we need this? (c.f. AVC seems not)
- High-level syntax items are significantly different from software in terms of both meaning and position. Needs to be unified.
- Use of bin string and bit string should be consistent.
- MD5 hash (JCTVC-E490)
- ALF coefficient derivation from syntax elements
- Improve text quality by considering: strict use of terms “unit” and “block” – block = a rectangular 2D array (one component), unit = collective term for specifying information for both luma and chroma. Don’t use ther term ‘unit’ by itself – always use the term ‘unit’ with prefix – coding unit, prediction unit or transform unit.
- Syntax functions cuAddress and granularity_block_boundary (fine granularity slices, JCTVC-E483) are missing
- Do we need SPS flag for disabling PCM mode? Currently PCM mode is turned on for all cases.
- Software seems to have SPS syntax to turn on/off combined reference list (JCTVC-D421), but currently text does not.
- Both variables IntraPredMode and IntraPredModeC are used in the syntax table but the actual derivations are specified in the decoding process. Maybe it’s better to move them to the semantics section.
- Binarization of intra_chroma_pred_mode reflecting codeword switching and luma-based chroma intra prediction (JCTVC-E266) is missing.
- Clarification on the use of intra_chroma_pred_mode and IntraPredModeC is needed. The former specifies the syntax item to indicate how to determine the chroma intra prediction mode (IntraPredModeC) as 0 (to Intra_FromLuma), 1 (to Intra_Vertical), 2 (to Intra_Horizontal), 3 (Intra_DC or Intra_Planar) or 4 (re-use luma mode). The latter specifies chroma intra prediction mode, which is actually mapped to the specific prediction process.
- The reference list construction and the decoded picture buffer management are not fully described. This is closely related to the work of the reference picture buffering and list construction AHG.
- NSQT text does not match the software w.r.t. non-square transforms. In HM, e.g.16x4 and 4x16 blocks are both transformed as 16x4 block and in case of 4x16 block, the transformed 16x4 sample are reordered to 4x16 after transform.
- NSQT mapping of coefficient level after inverse scan from square to non-square always uses zigzag scan and may be harmonized with the different scans used for Intra (horizontal, vertical), CAVLC (zigzag) and CABAC (diagonal).
- Chroma QP derivation process is missing. Can be taken from H.264/AVC and modified accordingly.
- The variable qPL used in deblocking filter process has to be defined by, e.g. setting it to QP_Y.
- The Chroma QP should be considered in chroma edge filtering instad of luma QP.
- The t_c and beta values of the deblocking filter should be explicitly scaled in the text since their values are provided in the table for the 8-bit case.
- There is no CABAC parsing described for the following syntax elements (ae(v)): sao_flag_cb, sao_flag_cr, sao_split_flag, sao_type_idx and sao_offset.
- Deblocking filter text seems incorrect. Some horizontal edges are not filtered mainly due to the parallel deblocking process. Usage of indices xDk and XPOS also has to be cross-verified with the software behaviour.
- Software-text mismatch of JCTVC-F252: SW applies DC filtering to 64x64 intra block but WD does not.
- There are input/output parameter inconsistencies between function call and actual functions of intra prediction. Function call uses sample position, block, size and chroma index while some functions do not use some parameters.
- PPS flag disabling the temporal motion vector predictor missing in both WD and HM (JCTVC-F744)

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000509

CONTENTS

Page

Abstract i

0 Introduction 11
0.1 Prologue 11
0.2 Purpose 11
0.3 Applications 11
0.4 Publication and versions of this Specification 11
0.5 Profiles and levels 11
0.6 Overview of the design characteristics 12
0.7 How to read this Specification 12

1 Scope 13

2 Normative references 13

3 Definitions 13

4 Abbreviations 18

5 Conventions 19
5.1 Arithmetic operators 19
5.2 Logical operators 19
5.3 Relational operators 20
5.4 Bit-wise operators 20
5.5 Assignment operators 20
5.6 Range notation 20
5.7 Mathematical functions 20
5.8 Order of operation precedence 21
5.9 Variables, syntax elements, and tables 22
5.10 Text description of logical operations 23
5.11 Processes 24

6 Source, coded, decoded and output data formats, scanning processes, and neighbouring relationships 24
6.1 Bitstream formats 24
6.2 Source, decoded, and output picture formats 24
6.3 Spatial subdivision of pictures and slices 26
6.4 Inverse scanning processes and derivation processes for neighbours 27
6.4.1 Inverse treeblock scanning process 27
6.5 Zig-zag scanning array initialisation process 28
6.6 Scanning order array initialisation process 28

7 Syntax and semantics 29
7.1 Method of specifying syntax in tabular form 29
7.2 Specification of syntax functions, categories, and descriptors 30
7.3 Syntax in tabular form 32
7.3.1 NAL unit syntax 32
7.3.2 Raw byte sequence payloads and RBSP trailing bits syntax 33
7.3.2.1 Sequence parameter set RBSP syntax 33
7.3.2.2 Picture parameter set RBSP syntax 33
7.3.2.3 Supplemental enhancement information RBSP syntax 34
7.3.2.4 Access unit delimiter RBSP syntax 34
7.3.2.5 Filler data RBSP syntax 34
7.3.2.6 Slice layer RBSP syntax 35
7.3.2.7 RBSP slice trailing bits syntax 35
7.3.2.8 RBSP trailing bits syntax 35
7.3.3 Slice header syntax 36
7.3.3.1 Reference picture list modification syntax 37
7.3.3.2 Reference picture lists combination syntax 37
7.3.3.3 Decoded reference picture marking syntax 38
7.3.3.4 Adaptive loop filter parameter syntax 39
7.3.4 Slice data syntax 40
7.3.5 Coding tree syntax 41

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000510

7.3.6 Coding unit syntax....41
7.3.7 Prediction unit syntax....43
7.3.8 Transform tree syntax....45
7.3.9 Transform coefficient syntax....47
7.3.10 Residual coding CABAC syntax....48
7.3.11 Residual coding CAVLC syntax....48
7.3.12 Level and sign CAVLC syntax....49
7.4 Semantics....50
7.4.1 NAL unit semantics....50
7.4.1.1 Encapsulation of an SODB within an RBSP (informative)....52
7.4.1.2 Order of NAL units and association to coded pictures, access units, and video sequences....53
7.4.2 Raw byte sequence payloads and RBSP trailing bits semantics....57
7.4.2.1 Sequence parameter set RBSP semantics....57
7.4.2.2 Picture parameter set RBSP semantics....58
7.4.2.3 Supplemental enhancement information RBSP semantics....59
7.4.2.4 Access unit delimiter RBSP semantics....59
7.4.2.5 Filler data RBSP semantics....59
7.4.2.6 Slice layer RBSP semantics....59
7.4.2.7 RBSP slice trailing bits semantics....59
7.4.2.8 RBSP trailing bits semantics....60
7.4.3 Slice header semantics....60
7.4.3.1 Reference picture list modification semantics....62
7.4.3.2 Reference picture lists combination semantics....63
7.4.3.3 Decoded reference picture marking semantics....63
7.4.3.4 Adaptive loop filter parameter semantics....66
7.4.4 Slice data semantics....67
7.4.5 Coding tree semantics....67
7.4.6 Coding unit semantics....68
7.4.7 Prediction unit semantics....69
7.4.8 Transform tree semantics....71
7.4.9 Transform coefficient semantics....72
7.4.10 Residual coding CABAC semantics....73
7.4.11 Residual coding CAVLC semantics....74
8 Decoding process....75
8.1 NAL unit decoding process....76
8.2 Slice decoding process....76
8.2.1 Decoding process for picture order count....76
8.2.1.1 Decoding process for picture order count type 0....76
8.2.1.2 Decoding process for picture order count type 1....76
8.2.1.3 Decoding process for picture order count type 2....76
8.2.2 Decoding process for reference picture lists construction....76
8.2.2.1 Decoding process for picture numbers....77
8.2.2.2 Initialisation process for reference picture lists....77
8.2.2.3 Modification process for reference picture lists....78
8.2.2.4 Mapping process for reference picture lists combination in B slices....80
8.2.3 Decoded reference picture marking process....81
8.2.3.1 Sequence of operations for decoded reference picture marking process....81
8.2.3.2 Decoding process for gaps in frame_num....82
8.2.3.3 Sliding window decoded reference picture marking process....82
8.2.3.4 Adaptive memory control decoded reference picture marking process....82
8.3 Decoding process for coding units coded in intra prediction mode....83
8.3.1 Derivation process for luma intra prediction mode....84
8.3.2 Derivation process for chroma intra prediction mode....86
8.3.3 Decoding process for intra blocks....86
8.3.3.1 Intra sample prediction....87
8.4 Decoding process for coding units coded in inter prediction mode....93
8.4.1 Inter prediction process....93
8.4.2 Decoding process for prediction units in inter prediction mode....95
8.4.2.1 Derivation process for motion vector components and reference indices....96
8.4.2.2 Decoding process for inter prediction samples....107
8.4.3 Decoding process for the residual signal of coding units coded in inter prediction mode....114
8.4.3.1 Decoding process for luma residual blocks....114

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000511

8.4.3.2 Decoding process for chroma residual blocks 115
8.5 Scaling, transformation and array construction process prior to deblocking filter process 116
8.5.1 Scaling and transformation process 116
8.5.2 Inverse scanning process for transform coefficients 117
8.5.3 Scaling process for transform coefficients 117
8.5.4 Transformation process for scaled transform coefficients 118
8.5.4.1 Transformation process for 4 samples 118
8.5.4.2 Transformation process for 8 samples 118
8.5.4.3 Transformation process for 16 samples 118
8.5.4.4 Transformation process for 32 samples 121
8.5.5 Array construction process 128
8.6 In-loop filter process 128
8.6.1 Deblocking filter process 128
8.6.1.1 Derivation process of transform unit boundary 129
8.6.1.2 Derivation process of prediction unit boundary 129
8.6.1.3 Derivation process of boundary filtering strength 130
8.6.1.4 Filtering process for coding unit 131
8.6.2 Adaptive loop filter process 136
8.6.2.1 Derivation process for filter coefficients 136
8.6.2.2 Derivation process for filter index array for luma samples 137
8.6.2.3 Filtering process for luma samples 138
8.6.2.4 Filtering process for chroma samples 139
9 Parsing process 139
9.1 Parsing process for Exp-Golomb codes 140
9.1.1 Mapping process for signed Exp-Golomb codes 141
9.2 CAVLC parsing process for slice data 142
9.2.1 Parsing process for VLC codes 142
9.2.2 Initialisation process 144
9.2.2.1 Counter initialisation for codeword index adaptive mapping 144
9.2.2.2 Initialisation process for lastPosVlcNumIndex 145
9.2.2.3 Initialisation process for lastPosTable 145
9.2.2.4 Initialisation process for splitPredPartModeTable 145
9.2.2.5 Initialisation process for intraModeTable 145
9.2.2.6 Initialisation process for cbpSplitTransTable 146
9.2.3 Codeword index mapping update process 146
9.2.4 Parsing process for transform coefficient syntax elements 147
9.2.4.1 Parsing process for last_pos_level_one 147
9.2.4.2 Parsing process for level_minus2_and_sign 148
9.2.4.3 Parsing process for run_level_one 148
9.2.4.4 Parsing process for level 152
9.2.4.5 Parsing process for cu_split_pred_part_mode 152
9.2.4.6 Parsing process for rem_intra_luma_pred_mode 153
9.2.4.7 Parsing process for cbp_and_split_transform 153
9.3 CABAC parsing process for slice data 154
9.3.1 Initialisation process 155
9.3.1.1 Initialisation process for context variables 155
9.3.1.2 Initialisation process for the arithmetic decoding engine 165
9.3.2 Binarization process 166
9.3.2.1 Unary (U) binarization process 169
9.3.2.2 Truncated unary (TU) binarization process 170
9.3.2.3 Truncated Rice (TR) binarization process 170
9.3.2.4 k-th order Exp-Golomb (EGk) binarization process 170
9.3.2.5 Concatenated unary/ k-th order Exp-Golomb (UEGk) binarization process 171
9.3.2.6 Fixed-length (FL) binarization process 171
9.3.2.7 Binarization process for cu_qp_delta 171
9.3.2.8 Binarization process for pred_type 171
9.3.2.9 Binarization process for rem_intra_luma_pred_mode 172
9.3.2.10 Binarization process for coeff_abs_level_minus3 172
9.3.3 Decoding process flow 173
9.3.3.1 Derivation process for ctxIdx 173
9.3.3.2 Arithmetic decoding process 182
9.3.4 Arithmetic encoding process (informative) 189

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000512

9.3.4.1 Initialisation process for the arithmetic encoding engine (informative) 189
9.3.4.2 Encoding process for a binary decision (informative) 190
9.3.4.3 Renormalization process in the arithmetic encoding engine (informative) 191
9.3.4.4 Bypass encoding process for binary decisions (informative) 193
9.3.4.5 Encoding process for a binary decision before termination (informative) 194
9.3.4.6 Byte stuffing process (informative) 196
Annex A Profiles and levels 197
A.1 Requirements on video decoder capability 197
A.2 Profiles 197
A.2.1 ABC profile 197
A.3 Levels 197
Annex B Byte stream format 199
B.1 Byte stream NAL unit syntax and semantics 199
B.1.1 Byte stream NAL unit syntax 199
B.1.2 Byte stream NAL unit semantics 199
B.2 Byte stream NAL unit decoding process 200
B.3 Decoder byte-alignment recovery (informative) 200
Annex C Hypothetical reference decoder 201
C.1 Operation of coded picture buffer (CPB) 202
C.1.1 Timing of bitstream arrival 202
C.1.2 Timing of coded picture removal 202
C.2 Operation of the decoded picture buffer (DPB) 202
C.3 Bitstream conformance 202
C.4 Decoder conformance 202
Annex D Supplemental enhancement information 203
D.1 SEI payload syntax 203
D.2 SEI payload semantics 203
Annex E Video usability information 204
E.1 VUI syntax 204
E.1.1 VUI parameters syntax 204
E.2 VUI semantics 204

LIST OF FIGURES

Figure 6-2 – Nominal vertical and horizontal locations of 4:2:2 luma and chroma samples in a picture [Ed. Re-draw figure] 26
Figure 6-3 – Nominal vertical and horizontal locations of 4:4:4 luma and chroma samples in a picture [Ed. Re-draw figure] 26
Figure 6-4 – A picture with 11 by 9 treeblocks that is partitioned into two slices 27
Figure 7-1 – Structure of an access unit not containing any NAL units with nal_unit_type equal to 0, 7, 8, or in the range of 12 to 18, inclusive, or in the range of 20 to 31, inclusive 56
Figure 8-2 – Intra prediction angle definition (informative) 92
Figure 8-3 – Spatial neighbours that can be used as merging candidates (informative) illustrates the position of the spatial neighbours A, B, C and D relative to the current prediction unit. 99
Figure 8-4 – Spatial neighbours, that can be used to derive reference indices for temporal merging candidates, (informative) illustrates the position of the spatial neighbours A, B, C, D and E relative to the current prediction unit. 101
Figure 8-5 – Spatial motion vector neighbours 103
Figure 8-6 – Integer samples (shaded blocks with upper-case letters) and fractional sample positions (un-shaded blocks with lower-case letters) for quarter sample luma interpolation 109
Figure 8-8 Mapping between geometric position and luma adaptive loop filter index according to alfTap (informative) 139
Figure 9-1 – Overview of the arithmetic decoding process for a single bin (informative) 183

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000513

Figure 9-2 – Flowchart for decoding a decision 185
Figure 9-3 – Flowchart of renormalization 187
Figure 9-4 – Flowchart of bypass decoding process 188
Figure 9-5 – Flowchart of decoding a decision before termination 189
Figure 9-6 – Flowchart for encoding a decision 191
Figure 9-7 – Flowchart of renormalization in the encoder 192
Figure 9-8 – Flowchart of PutBit(B) 193
Figure 9-9 – Flowchart of encoding bypass 194
Figure 9-10 – Flowchart of encoding a decision before termination 195
Figure 9-11 – Flowchart of flushing at termination 196
Figure C-9-12 – Structure of byte streams and NAL unit streams for HRD conformance checks 201
Figure C-9-13 – HRD buffer model 202

LIST OF TABLES

Table 5-1 – Operation precedence from highest (at top of table) to lowest (at bottom of table) 23
Table 6-1 – SubWidthC, and SubHeightC values derived from chroma_format_idc and separate_colour_plane_flag 26
Table 7-1 – NAL unit type codes, syntax element categories, and NAL unit type classes 52
Table 7-2 – Meaning of primary_pic_type 60
Table 7-3 – Name association to slice_type 61
Table 7-4 – modification_of_pic_nums_idc operations for modification of reference picture lists 64
Table 7-5 – Interpretation of adaptive_ref_pic_marking_mode_flag 65
Table 7-6 – Memory management control operation (memory_management_control_operation) values 66
Table 7-7 – Specification of alf_chroma_idc 68
Table 7-8 – Specification of cu_split_pred_part_mode when log2CUSize is greater than Log2MinCUSize 69
Table 7-9 – Specification of cu_split_pred_part_mode when log2CUSize is equal to Log2MinCUSize 69
Table 7-10 - Name association to prediction mode and partitioning type 70
Table 7-11 – Number of bins and number of mode for rem_intra_luma_pred_mode associated to PuSize 70
Table 7-12 – Name association to inter prediction mode 71
Table 7-13 – Specification of cbp_and_split_transform 73
Table 7-14 – Specification of ScanType[log2TrafoSize – 2][IntraPredMode] 74
Table 7-15 – Specification of ScanTypeC[log2TrafoSize – 2][IntraPredModeC] 74
Table 8-1 – Specification of intraPredModeNum 85
Table 8-2 – Specification of mapIntraPredMode5 and mapIntraPredMode9 86
Table 8-3 – Specification of IntraPredModeC according to the values of intra_chroma_pred_mode and IntraPredMode[xB][yB] 87
Table 8-4 – Specification of intraPredMode and associated names 89
Table 8-5 – Specification of intraFilterType[nS][IntraPredMode] for various prediction unit sizes 90
Table 8-6 – Specification of intraPredOrder 92
Table 8-7 – Specification of intraPredAngle 93
Table 8-8 – Specification of invAngle 93

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000514

Table 8-9 – Assignment of the luma prediction sample predSampleLX$_L$[x_L, y_L]....112
Table 8-10 – Assignment of the chroma prediction sample predSampleLX$_C$[x_C, y_C] for (X, Y) being replaced by (1, b), (2, c), (3, d), (4, e), (5, f), (6, g), and (7, h), respectively....114
Table 8-11 – Derivation of threshold variables β and t_C from input Q....136
Table 8-12 – Specification of horPos[i] according to alfTap for adaptive loop filter process of luma samples....139
Table 8-13 – Specification of verPos[i] according to alfTap for adaptive loop filter process of luma samples....139
Table 9-1 – Bit strings with "prefix" and "suffix" bits and assignment to codeNum ranges (informative)....141
Table 9-2 – Exp-Golomb bit strings and codeNum in explicit form and used as ue(v) (informative)....142
Table 9-3 – Assignment of syntax element to codeNum for signed Exp-Golomb coded syntax elements se(v)....143
Table 9-4 – codeNum and codeword with vlcNum equal to 15....145
Table 9-5 – Specification of initial values of lastPosTable[tableNum][codeNum]....146
Table 9-6 – Specification of intraModeTable[k][codeNum]....146
Table 9-7 – Specification of initial values of cbpSplitTransTable[flagPattern-8][n][codeNum]....147
Table 9-8 – Specification of lastPosVlcNumTable[blockType][vlcNumIdx]....149
Table 9-9 – Derivation of vlcNum from tableIdx and maxRunIdx....150
Table 9-10 – Derivation of largeOnePos from tableIdx and maxRunIdx when N is equal to 4....151
Table 9-11 – Derivation of largeOnePos from tableIdx and maxRunIdx when N is greater than 4....151
Table 9-12 – Derivation of runOfZeros from maxRun and codeNum....151
Table 9-13 – Derivation of levelThreshold from vlcIdx....153
Table 9-14 – Specification of cbpVlcNumTable[cbpVlcNumIdx]....155
Table 9-15 – Derivation of cMax from flagPattern and n....155
Table 9-16 – Specification of cbpSplitTransTable and counterNum associated....155
Table 9-17 – Association of ctxIdx and syntax elements for each slice type in the initialisation process....157
Table 9-18 – Values of variables m and n for alf_cu_flag ctxIdx....157
Table 9-19 – Values of variables m and n for split_coding_unit_flag ctxIdx....158
Table 9-20 – Values of variables m and n for skip_flag ctxIdx....158
Table 9-21 – Values of variables m and n for cu_qp_delta ctxIdx....158
Table 9-22 – Values of variables m and n for pred_type....158
Table 9-23 – Values of variable m and n for prev_intra_luma_pred_flag ctxIdx....158
Table 9-24 – Values of variable m and n for rem_intra_luma_pred_mode ctxIdx....158
Table 9-25 – Values of variable m and n for intra_chroma_pred_mode ctxIdx....159
Table 9-26 – Values of variable m and n for merge_flag ctxIdx....159
Table 9-27 – Values of variable m and n for merge_idx ctxIdx....159
Table 9-28 – Values of variable m and n for inter_pred_flag ctxIdx....159
Table 9-29 – Values of variable m and n for ref_idx_lc, ref_idx_l0, ref_idx_l1 ctxIdx....159
Table 9-30 – Values of variable m and n for mvd_lc, mvd_l0, mvd_l1 ctxIdx....159
Table 9-31 – Values of variable m and n for mvp_idx_lc, mvp_idx_l0, mvp_idx_l1 ctxIdx....160
Table 9-32 – Values of variable m and n for no_residual_data_flag ctxIdx....160
Table 9-33 – Values of variable m and n for split_transform_flag ctxIdx....160
Table 9-34 – Values of variable m and n for cbf_luma ctxIdx....160

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000515

Table 9-35 – Values of variable m and n for cbf_cb ctxIdx 160
Table 9-36 – Values of variable m and n for cbf_cr ctxIdx 160
Table 9-37 – Values of variable m and n for last_significant_coefficient_x ctxIdx 161
Table 9-38 – Values of variable m and n for last_significant_coefficient_y ctxIdx 162
Table 9-39 – Values of variable m and n for significant_coeff_flag ctxIdx (A) 163
Table 9-40 – Values of variable m and n for significant_coeff_flag ctxIdx (B) 163
Table 9-41 – Values of variable m and n for significant_coeff_flag ctxIdx (C) 164
Table 9-42 – Values of variable m and n for coeff_abs_level_greater1_flag ctxIdx 165
Table 9-43 – Values of variable m and n for coeff_abs_level_greater2_flag ctxIdx 166
Table 9-44 – Syntax elements and associated types of binarization, maxBinIdxCtx, ctxIdxTable, and ctxIdxOffset 168
Table 9-45 – Bin string of the unary binarization (informative) 171
Table 9-46 – Binarization for pred_type 173
Table 9-47 – Binarization for rem_intra_luma_pred_mode 173
Table 9-48 – Specifcation of cTRMax depending on cRiceParam 174
Table 9-49 – Assignment of ctxIdxInc to binIdx for all ctxIdxTable and ctxIdxOffset values 176
Table 9-50 – Specifcation of ctxIdxInc using left and above syntax elements 179
Table 9-51 – Specifcation of lastCtx depending on binIdx 180
Table 9-52 – Specification of rangeTabLPS depending on pStateIdx and qCodIRangeIdx 186
Table 9-53 – State transition table 187
Table A-9-54 – Level limits [Ed. (TW) Kept for convenience] 199

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000516

Foreword

The International Telecommunication Union (ITU) is the United Nations specialized agency in the field of telecommunications. The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T is responsible for studying technical, operating and tariff questions and issuing Recommendations on them with a view to standardising telecommunications on a world-wide basis. The World Telecommunication Standardization Assembly (WTSA), which meets every four years, establishes the topics for study by the ITU-T study groups that, in turn, produce Recommendations on these topics. The approval of ITU-T Recommendations is covered by the procedure laid down in WTSA Resolution 1. In some areas of information technology that fall within ITU-T's purview, the necessary standards are prepared on a collaborative basis with ISO and IEC.

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialised system for world-wide standardisation. National Bodies that are members of ISO and IEC participate in the development of International Standards through technical committees established by the respective organisation to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organisations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75% of the national bodies casting a vote.

This Recommendation | International Standard was prepared jointly by ITU-T SG 16 Q.6, also known as VCEG (Video Coding Experts Group), and by ISO/IEC JTC 1/SC 29/WG 11, also known as MPEG (Moving Picture Experts Group). VCEG was formed in 1997 to maintain prior ITU-T video coding standards and develop new video coding standard(s) appropriate for a wide range of conversational and non-conversational services. MPEG was formed in 1988 to establish standards for coding of moving pictures and associated audio for various applications such as digital storage media, distribution, and communication.

In this Recommendation | International Standard Annexes A through E contain normative requirements and are an integral part of this Recommendation | International Standard.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000517

INTERNATIONAL STANDARD

ITU-T RECOMMENDATION

High-Efficiency Video Coding

0 Introduction

This clause does not form an integral part of this Recommendation | International Standard.

0.1 Prologue

This subclause does not form an integral part of this Recommendation | International Standard.

As the costs for both processing power and memory have reduced, network support for coded video data has diversified, and advances in video coding technology have progressed, the need has arisen for an industry standard for compressed video representation with substantially increased coding efficiency and enhanced robustness to network environments. Toward these ends the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG) formed a Joint Collaborative Team on Video Coding (JCT-VC) in 2010 for development of a new Recommendation | International Standard.

0.2 Purpose

This subclause does not form an integral part of this Recommendation | International Standard.

[Ed. TW: revise the following] This Recommendation | International Standard was developed in response to the growing need for higher compression of moving pictures for various applications such as videoconferencing, digital storage media, television broadcasting, internet streaming, and communication. It is also designed to enable the use of the coded video representation in a flexible manner for a wide variety of network environments. The use of this Recommendation | International Standard allows motion video to be manipulated as a form of computer data and to be stored on various storage media, transmitted and received over existing and future networks and distributed on existing and future broadcasting channels.

0.3 Applications

This subclause does not form an integral part of this Recommendation | International Standard.

[Ed. TW: revise the following] This Recommendation | International Standard is designed to cover a broad range of applications for video content including but not limited to the following:

CATV	Cable TV on optical networks, copper, etc.
DBS	Direct broadcast satellite video services
DSL	Digital subscriber line video services
DTTB	Digital terrestrial television broadcasting
ISM	Interactive storage media (optical disks, etc.)
MMM	Multimedia mailing
MSPN	Multimedia services over packet networks
RTC	Real-time conversational services (videoconferencing, videophone, etc.)
RVS	Remote video surveillance
SSM	Serial storage media (digital VTR, etc.)

0.4 Publication and versions of this Specification

This subclause does not form an integral part of this Recommendation | International Standard.

This Specification has been jointly developed by ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group. It is published as technically-aligned twin text in both organizations ITU-T and ISO/IEC.

0.5 Profiles and levels

This subclause does not form an integral part of this Recommendation | International Standard.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000518

This Recommendation | International Standard is designed to be generic in the sense that it serves a wide range of applications, bit rates, resolutions, qualities, and services. Applications should cover, among other things, digital storage media, television broadcasting and real-time communications. In the course of creating this Specification, various requirements from typical applications have been considered, necessary algorithmic elements have been developed, and these have been integrated into a single syntax. Hence, this Specification will facilitate video data interchange among different applications.

Considering the practicality of implementing the full syntax of this Specification, however, a limited number of subsets of the syntax are also stipulated by means of "profiles" and "levels". These and other related terms are formally defined in clause 3.

A "profile" is a subset of the entire bitstream syntax that is specified by this Recommendation | International Standard. Within the bounds imposed by the syntax of a given profile it is still possible to require a very large variation in the performance of encoders and decoders depending upon the values taken by syntax elements in the bitstream such as the specified size of the decoded pictures. In many applications, it is currently neither practical nor economic to implement a decoder capable of dealing with all hypothetical uses of the syntax within a particular profile.

In order to deal with this problem, "levels" are specified within each profile. A level is a specified set of constraints imposed on values of the syntax elements in the bitstream. These constraints may be simple limits on values. Alternatively they may take the form of constraints on arithmetic combinations of values (e.g., picture width multiplied by picture height multiplied by number of pictures decoded per second).

Coded video content conforming to this Recommendation | International Standard uses a common syntax. In order to achieve a subset of the complete syntax, flags, parameters, and other syntax elements are included in the bitstream that signal the presence or absence of syntactic elements that occur later in the bitstream.

0.6 Overview of the design characteristics

This subclause does not form an integral part of this Recommendation | International Standard.

[Ed. TW: revise the following] The coded representation specified in the syntax is designed to enable a high compression capability for a desired image or video quality. The algorithm is typically not lossless, as the exact source sample values are typically not preserved through the encoding and decoding processes. A number of techniques may be used to achieve highly efficient compression. Encoding algorithms (not specified in this Recommendation | International Standard) may select between inter and intra coding for block-shaped regions of each picture. Inter coding uses motion vectors for block-based inter prediction to exploit temporal statistical dependencies between different pictures. Intra coding uses various spatial prediction modes to exploit spatial statistical dependencies in the source signal for a single picture. Motion vectors and intra prediction modes may be specified for a variety of block sizes in the picture. The prediction residual is then further compressed using a transform to remove spatial correlation inside the transform block before it is quantised, producing an irreversible process that typically discards less important visual information while forming a close approximation to the source samples. Finally, the motion vectors or intra prediction modes are combined with the quantised transform coefficient information and encoded using either variable length coding or arithmetic coding.

0.7 How to read this Specification

This subclause does not form an integral part of this Recommendation | International Standard.

It is suggested that the reader starts with clause 1 (Scope) and moves on to clause 3 (Definitions). Clause 6 should be read for the geometrical relationship of the source, input, and output of the decoder. Clause 7 (Syntax and semantics) specifies the order to parse syntax elements from the bitstream. See subclauses 7.1-7.3 for syntactical order and see subclause 7.4 for semantics; i.e., the scope, restrictions, and conditions that are imposed on the syntax elements. The actual parsing for most syntax elements is specified in clause 9 (Parsing process). Finally, clause 8 specifies how the syntax elements are mapped into decoded samples. Throughout reading this Specification, the reader should refer to clauses 2 (Normative references), 4 (Abbreviations), and 5 (Conventions) as needed. Annexes A through E also form an integral part of this Recommendation | International Standard.

Annex A specifies profiles each being tailored to certain application domains, and defines the so-called levels of the profiles. Annex B specifies syntax and semantics of a byte stream format for delivery of coded video as an ordered stream of bytes. Annex C specifies the hypothetical reference decoder and its use to check bitstream and decoder conformance. Annex D specifies syntax and semantics for supplemental enhancement information message payloads. Annex E specifies syntax and semantics of the video usability information parameters of the sequence parameter set.

Throughout this Specification, statements appearing with the preamble "NOTE -" are informative and are not an integral part of this Recommendation | International Standard.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000519

1 Scope

This document specifies High-Efficiency Video Coding.

2 Normative references

The following Recommendations and International Standards contain provisions which, through reference in this text, constitute provisions of this Recommendation | International Standard. At the time of publication, the editions indicated were valid. All Recommendations and Standards are subject to revision, and parties to agreements based on this Recommendation | International Standard are encouraged to investigate the possibility of applying the most recent edition of the Recommendations and Standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards. The Telecommunication Standardization Bureau of the ITU maintains a list of currently valid ITU-T Recommendations. [Ed. TW: revise the following]

- ITU-T Recommendation T.35 (2000), *Procedure for the allocation of ITU-T defined codes for non-standard facilities*.
- ISO/IEC 11578:1996, Annex A, *Universal Unique Identifier*.
- ISO/CIE 10527:2007, *Colorimetric Observers*.

3 Definitions

[Ed. (TW) adpated definitions so far. Needs more work including turning them into 1 sentence each.]

For the purposes of this Recommendation | International Standard, the following definitions apply:

3.1 **access unit**: A set of *NAL units* that are consecutive in *decoding order* and contain exactly one *primary coded picture*. In addition to the *primary coded picture* one *auxiliary coded picture*, or other *NAL units* not containing *slices* of a *primary coded picture*. The decoding of an access unit always results in a *decoded picture*.

3.2 **AC transform coefficient**: Any *transform coefficient* for which the *frequency index* in one or both dimensions is non-zero.

3.3 **adaptive binary arithmetic decoding process**: An entropy *decoding process* that derives the values of *bins* from a *bitstream* produced by an *adaptive binary arithmetic encoding process*.

3.4 **adaptive binary arithmetic encoding process**: An entropy *encoding process*, not normatively specified in this Recommendation | International Standard, that codes a sequence of *bins* and produces a *bitstream* that can be decoded using the *adaptive binary arithmetic decoding process*.

3.5 **B slice**: A *slice* that may be decoded using *intra prediction* or *inter prediction* using at most two *motion vectors* and *reference indices* to *predict* the sample values of each *block*.

3.6 **bin**: One bit of a *bin string*.

3.7 **binarization**: A set of *bin strings* for all possible values of a *syntax element*.

3.8 **binarization process**: A unique mapping process of all possible values of a *syntax element* onto a set of *bin strings*.

3.9 **bin string**: A string of *bins*. A bin string is an intermediate binary representation of values of *syntax elements* from the *binarization* of the *syntax element*.

3.10 **bi-predictive slice:** See *B slice*.

3.11 **bitstream**: A sequence of bits that forms the representation of *coded pictures* and associated data forming one or more *coded video sequences*. Bitstream is a collective term used to refer either to a *NAL unit stream* or a *byte stream*.

3.12 **block**: An MxN (M-column by N-row) array of samples, or an MxN array of *transform coefficients*.

3.13 **broken link**: A location in a *bitstream* at which it is indicated that some subsequent *pictures* in *decoding order* may contain serious visual artefacts due to unspecified operations performed in the generation of the *bitstream*.

3.14 **byte**: A sequence of 8 bits, written and read with the most significant bit on the left and the least significant bit on the right. When represented in a sequence of data bits, the most significant bit of a byte is first.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000520

3.15 **byte-aligned**: A position in a *bitstream* is byte-aligned when the position is an integer multiple of 8 bits from the position of the first bit in the *bitstream*. A bit or *byte* or *syntax element* is said to be byte-aligned when the position at which it appears in a *bitstream* is byte-aligned.

3.16 **byte stream**: An encapsulation of a *NAL unit stream* containing *start code prefixes* and *NAL units* as specified in Annex B.

3.17 **can**: A term used to refer to behaviour that is allowed, but not necessarily required.

3.18 **chroma**: An adjective specifying that a sample array or single sample is representing one of the two colour difference signals related to the primary colours. The symbols used for a chroma array or sample are Cb and Cr.

NOTE – The term chroma is used rather than the term chrominance in order to avoid the implication of the use of linear light transfer characteristics that is often associated with the term chrominance.

3.19 **clean ~~decoding refresh~~random access (CRA) access unit**: An *access unit* in which the *primary coded picture* is a *CRA picture*.

3.20 **clean ~~random access~~decoding refresh (CRA) picture**: A *coded picture* containing only ~~*slices with I slice*~~ and for which each *slice* has *nal_unit_type* equal to 4, all *coded pictures* that follow the *CRA picture* both in *decoding order* and *output order* shall not use *inter prediction* from any *picture* that precedes the *CRA picture* either in *decoding order* or *output order*, and any *picture* that precedes the *CRA picture* in *decoding order* also precedes the *CRA* picture in *output order*. *Types* that cause the *decoding process* to mark all *reference pictures* except the current *CRA picture* as "unused for reference" immediately before the decoding of the first picture following the current *CRA* with an *output order* greater than the current *CRA picture*. All *coded pictures* that follow a *CRA picture* in *output order* can be decoded without *inter prediction* from any picture that pre-order the *CRA picture* in *output order*. All *coded pictures* with *output order* smaller than the current *CRA* are not affected by the *deferred marking process*.

3.21 **coded picture**: A *coded representation* of a *picture*.

3.22 **coded picture buffer (CPB)**: A first-in first-out buffer containing *access units* in *decoding order* specified in the *hypothetical reference decoder* in Annex 0.

3.23 **coded representation**: A data element as represented in its coded form.

3.24 **coded slice NAL unit**: A *NAL unit* containing a *slice*.

3.25 **coded video sequence**: A sequence of *access units* that consists, in decoding order, of an *IDR access unit* followed by zero or more non-IDR *access units* including all subsequent *access units* up to but not including any subsequent *IDR access unit*.

3.26 **component**: An array or single sample from one of the three arrays (*luma* and two *chroma*) that make up a *picture* in 4:2:0, 4:2:2, or 4:4:4 colour format or the array or a single sample of the array that make up a *picture* in monochrome format.

3.27 **context variable**: A variable specified for the *adaptive binary arithmetic decoding process* of a *bin* by an equation containing recently decoded *bins*.

3.28 **DC transform coefficient**: A *transform coefficient* for which the *frequency index* is zero in all dimensions.

3.29 **decoded picture**: A *decoded picture* is derived by decoding a *coded picture*.

3.30 **decoded picture buffer (DPB)**: A buffer holding *decoded pictures* for reference, output reordering, or output delay specified for the *hypothetical reference decoder* in Annex 0.

3.31 **decoder**: An embodiment of a *decoding process*.

3.32 **decoder under test (DUT)**: A *decoder* that is tested for conformance to this Recommendation | International Standard by operating the *hypothetical stream scheduler* to deliver a conforming *bitstream* to the *decoder* and to the *hypothetical reference decoder* and comparing the values and timing of the output of the two *decoders*.

3.33 **decoding order**: The order in which *syntax elements* are processed by the *decoding process*.

3.34 **decoding process**: The process specified in this Recommendation | International Standard that reads a *bitstream* and derives *decoded pictures* from it.

3.35 **display process**: A process not specified in this Recommendation | International Standard having, as its input, the cropped decoded *pictures* that are the output of the *decoding process*.

3.36 **emulation prevention byte**: A *byte* equal to 0x03 that may be present within a *NAL unit*. The presence of emulation prevention bytes ensures that no sequence of consecutive *byte-aligned bytes* in the *NAL unit* contains a *start code prefix*.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000521

3.37 **encoder**: An embodiment of an *encoding process*.

3.38 **encoding process**: A process, not specified in this Recommendation | International Standard, that produces a *bitstream* conforming to this Recommendation | International Standard.

3.39 **flag**: A variable that can take one of the two possible values 0 and 1.

3.40 **frequency index**: A one-dimensional or two-dimensional index associated with a *transform coefficient* prior to an *inverse transform* part of the *decoding process*.

3.41 **hypothetical reference decoder (HRD)**: A hypothetical *decoder* model that specifies constraints on the variability of conforming *NAL unit streams* or conforming *byte streams* that an encoding process may produce.

3.42 **hypothetical stream scheduler (HSS)**: A hypothetical delivery mechanism for the timing and data flow of the input of a *bitstream* into the *hypothetical reference decoder*. The HSS is used for checking the conformance of a *bitstream* or a *decoder*.

3.43 **I slice**: A *slice* that is decoded using *intra prediction* only.

3.44 **informative**: A term used to refer to content provided in this Recommendation | International Standard that is not an integral part of this Recommendation | International Standard. Informative content does not establish any mandatory requirements for conformance to this Recommendation | International Standard.

3.45 **instantaneous decoding refresh (IDR) access unit**: An *access unit* in which the *primary coded picture* is an *IDR picture*.

3.46 **instantaneous decoding refresh (IDR) picture**: A *coded picture* for which the variable IdrPicFlag is equal to 1. An IDR picture causes the *decoding process* to mark all *reference pictures* as "unused for reference" immediately after the decoding of the IDR picture. All *coded pictures* that follow an IDR picture in *decoding order* can be decoded without *inter prediction* from any *picture* that precedes the IDR picture in *decoding order*. The first *picture* of each *coded video sequence* in *decoding order* is an IDR picture.

3.47 **inter coding**: Coding of a *block*, *macroblock*, *slice*, or *picture* that uses *inter prediction*.

3.48 **inter prediction**: A *prediction* derived from only data elements (e.g. sample value or motion vector) of decoded samples of *reference pictures* other than the current *decoded picture*.

3.49 **intra coding**: Coding of a *block, macroblock*, *slice*, or *picture* that uses *intra prediction*.

3.50 **intra prediction**: A *prediction* derived from only data elements (e.g. sample value) of the decoded samples of the same decoded *slice*.

3.51 **intra slice**: See *I slice*.

3.52 **inverse transform**: A part of the *decoding process* by which a set of *transform coefficients* are converted into spatial-domain values, or by which a set of *transform coefficients* are converted into *DC transform coefficients*.

3.53 **layer**: One of a set of syntactical structures in a non-branching hierarchical relationship. Higher layers contain lower layers. The coding layers are the *coded video sequence*, *picture*, *slice*, and *treeblock* layers.

3.54 **leaf**: A terminating node of a tree that is a root node of a tree of depth 0.

3.55 **level**: A defined set of constraints on the values that may be taken by the *syntax elements* and variables of this Recommendation | International Standard. The same set of levels is defined for all *profiles*, with most aspects of the definition of each level being in common across different *profiles*. Individual implementations may, within specified constraints, support a different level for each supported *profile*. In a different context, level is the value of a *transform coefficient* prior to *scaling*.

3.56 **list 0 (list 1) motion vector**: A *motion vector* associated with a *reference index* pointing into *reference picture list 0* (*list 1*).

3.57 **list 0 (list 1, list combination) prediction**: *Inter prediction* of the content of a *slice* using a *reference index* pointing into *reference picture list 0* (*list 1, combination*).

3.58 **luma**: An adjective specifying that a sample array or single sample is representing the monochrome signal related to the primary colours. The symbol or subscript used for luma is Y or L.

NOTE – The term luma is used rather than the term luminance in order to avoid the implication of the use of linear light transfer characteristics that is often associated with the term luminance. The symbol L is sometimes used instead of the symbol Y to avoid confusion with the symbol y as used for vertical location.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000522

3.59 **may**: A term used to refer to behaviour that is allowed, but not necessarily required. In some places where the optional nature of the described behaviour is intended to be emphasized, the phrase "may or may not" is used to provide emphasis.

3.60 **memory management control operation**: Seven operations that control *reference picture marking*.

3.61 **motion vector**: A two-dimensional vector used for *inter prediction* that provides an offset from the coordinates in the *decoded picture* to the coordinates in a *reference picture*.

3.62 **must**: A term used in expressing an observation about a requirement or an implication of a requirement that is specified elsewhere in this Recommendation | International Standard. This term is used exclusively in an *informative* context.

3.63 **NAL unit**: A *syntax structure* containing an indication of the type of data to follow and *bytes* containing that data in the form of an *RBSP* interspersed as necessary with *emulation prevention bytes*.

3.64 **NAL unit stream**: A sequence of *NAL units*.

3.65 **non-reference picture**: A *picture* coded with nal_ref_flag equal to 0. A *non-reference picture* is not used for *inter prediction* of any other *pictures*.

3.66 **note**: A term used to prefix *informative* remarks. This term is used exclusively in an *informative* context.

3.67 **output order**: The order in which the *decoded pictures* are output from the *decoded picture buffer*.

3.68 **P slice**: A *slice* that may be decoded using *intra prediction* or *inter prediction* using at most one *motion vector* and *reference index* to *predict* the sample values of each *block*.

3.69 **parameter**: A *syntax element* of a *sequence parameter set* or a *picture parameter set*. Parameter is also used as part of the defined term *quantisation parameter*.

3.70 **partitioning**: The division of a set into subsets such that each element of the set is in exactly one of the subsets.

3.71 **picture**: [Ed. (TW) define]

3.72 **picture parameter set**: A *syntax structure* containing *syntax elements* that apply to zero or more entire *coded pictures* as determined by the pic_parameter_set_id *syntax element* found in each *slice header*.

3.73 **picture order count**: [Ed. (TW) re-check] A variable that is associated with each *coded picture* and has a value that is non-decreasing with increasing *picture* position in *output order* relative to the first output *picture* of the previous *IDR picture* in *decoding order* or relative to the first output *picture* of the previous *picture*, in *decoding order*, that contains a *memory management control operation* that marks all *reference pictures* as "unused for reference".

3.74 **prediction**: An embodiment of the *prediction process*.

3.75 **prediction process**: The use of a *predictor* to provide an estimate of the sample value or data element (e.g. sample value or motion vector) currently being decoded.

3.76 **predictive slice**: See *P slice*.

3.77 **predictor**: A combination of specified values or previously decoded sample values or data elements (e.g. sample value or motion vector) used in the *decoding process* of subsequent sample values or data elements.

3.78 **primary coded picture**: The coded representation of a *picture* to be used by the *decoding process* for a bitstream conforming to this Recommendation | International Standard. The primary coded picture contains all *treeblocks* of the *picture*. The only *pictures* that have a normative effect on the *decoding process* are primary coded pictures.

3.79 **profile**: A specified subset of the syntax of this Recommendation | International Standard.

3.80 **quadtree**: A *tree* in which a parent node can be split into four child nodes. A child node may become parent node for another plit into four child nodes.

3.81 **quantisation parameter**: A variable used by the *decoding process* for *scaling* of *transform coefficient levels*.

3.82 **random access**: The act of starting the decoding process for a *bitstream* at a point other than the beginning of the stream.

3.83 **raster scan**: A mapping of a rectangular two-dimensional pattern to a one-dimensional pattern such that the first entries in the one-dimensional pattern are from the first top row of the two-dimensional pattern scanned from left to right, followed similarly by the second, third, etc., rows of the pattern (going down) each scanned from left to right.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000523

3.84 **raw byte sequence payload (RBSP):** A *syntax structure* containing an integer number of *bytes* that is encapsulated in a *NAL unit*. An RBSP is either empty or has the form of a *string of data bits* containing *syntax elements* followed by an *RBSP stop bit* and followed by zero or more subsequent bits equal to 0.

3.85 **raw byte sequence payload (RBSP) stop bit:** A bit equal to 1 present within a *raw byte sequence payload (RBSP)* after a *string of data bits*. The location of the end of the *string of data bits* within an *RBSP* can be identified by searching from the end of the *RBSP* for the *RBSP stop bit*, which is the last non-zero bit in the *RBSP*.

3.86 **recovery point**: A point in the *bitstream* at which the recovery of an exact or an approximate representation of the *decoded pictures* represented by the *bitstream* is achieved after a *random access* or *broken link*.

3.87 **reference index**: An index into a *reference picture list*.

3.88 **reference picture**: A *picture* with nal_ref_[illegible] equal to [illegible]. A *reference picture* contains samples that may be used for *inter prediction* in the *decoding process* of subsequent *pictures* in *decoding order*.

3.89 **reference picture list**: A list of *reference pictures* that is used for *uni-prediction* of a *P* or *B slice*. For the *decoding process* of a *P slice*, there is one reference picture list. For the *decoding process* of a *B slice*, there are two reference picture lists (list 0 and list 1) and a reference picture lists combination. [Ed. (TW) the latter may not be true anymore]

3.90 **reference picture list 0**: A *reference picture list* used for *inter prediction* of a *P* or *B slice*. All *inter prediction* used for *P slices* uses reference picture list 0. Reference picture list 0 is one of two *reference picture lists* used for *bi-prediction* for a *B slice*, with the other being *reference picture list*. [Ed. (TW) the latter may not be true anymore]

3.91 **reference picture list 1**: A *reference picture list* used for *bi-prediction* of a *B slice*. Reference picture list 1 is one of two *reference picture lists* used for *bi-prediction* for a *B slice*, with the other being *reference picture list 0*.

3.92 **reference picture lists combination**: A reference picture list used for uni-prediction of a B slice. Reference picture lists combination is derived from the entries of the reference picture lists 0 and reference picture list 1.

3.93 **reference picture marking**: Specifies, in the bitstream, how the *decoded pictures* are marked for *inter prediction*.

3.94 **reserved**: The term reserved, when used in the clauses specifying some values of a particular *syntax element*, are for future use by ITU-T | ISO/IEC. These values shall not be used in *bitstreams* conforming to this Recommendation | International Standard, but may be used in future extensions of this Recommendation | International Standard by ITU-T | ISO/IEC.

3.95 **residual**: The decoded difference between a *prediction* of a sample or data element and its decoded value.

3.96 **sample aspect ratio**: Specifies, for assisting the *display process*, which is not specified in this Recommendation | International Standard, the ratio between the intended horizontal distance between the columns and the intended vertical distance between the rows of the *luma* sample array in a *picture*. Sample aspect ratio is expressed as *h*:*v*, where *h* is horizontal width and *v* is vertical height (in arbitrary units of spatial distance).

3.97 **scaling**: The process of multiplying *transform coefficient levels* by a factor, resulting in *transform coefficients*.

3.98 **sequence parameter set**: A *syntax structure* containing *syntax elements* that apply to zero or more entire *coded video sequences* as determined by the content of a seq_parameter_set_id *syntax element* found in the *picture parameter set* referred to by the pic_parameter_set_id *syntax element* found in each *slice header*.

3.99 **shall**: A term used to express mandatory requirements for conformance to this Recommendation | International Standard. When used to express a mandatory constraint on the values of *syntax elements* or on the results obtained by operation of the specified *decoding process*, it is the responsibility of the *encoder* to ensure that the constraint is fulfilled. When used in reference to operations performed by the *decoding process*, any *decoding process* that produces identical results to the *decoding process* described herein conforms to the *decoding process* requirements of this Recommendation | International Standard.

3.100 **should**: A term used to refer to behaviour of an implementation that is encouraged to be followed under anticipated ordinary circumstances, but is not a mandatory requirement for conformance to this Recommendation | International Standard.

3.101 **slice**: An integer number of *treeblocks* ordered consecutively in the *raster scan*. For the *primary coded picture*, the division of each *picture* into slices is a *partitioning*. The *treeblock addresses* are derived from the first *treeblock address* in a slice (as represented in the *slice header*).

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000524

3.102 **slice header**: A part of a coded *slice* containing the data elements pertaining to the first or all *macroblocks* represented in the *slice*.

3.103 **source**: Term used to describe the video material or some of its attributes before encoding.

3.104 **start code prefix**: A unique sequence of three *bytes* equal to 0x000001 embedded in the *byte stream* as a prefix to each *NAL unit*. The location of a start code prefix can be used by a *decoder* to identify the beginning of a new *NAL unit* and the end of a previous *NAL unit*. Emulation of start code prefixes is prevented within *NAL units* by the inclusion of *emulation prevention bytes*.

3.105 **string of data bits (SODB)**: A sequence of some number of bits representing *syntax elements* present within a *raw byte sequence payload* prior to the *raw byte sequence payload stop bit*. Within an SODB, the left-most bit is considered to be the first and most significant bit, and the right-most bit is considered to be the last and least significant bit.

3.106 **syntax element**: An element of data represented in the *bitstream*.

3.107 **syntax structure**: Zero or more *syntax elements* present together in the *bitstream* in a specified order.

3.108 **transform coefficient**: A scalar quantity, considered to be in a frequency domain, that is associated with a particular one-dimensional or two-dimensional *frequency index* in an *inverse transform* part of the *decoding process*.

3.109 **transform coefficient level**: An integer quantity representing the value associated with a particular two-dimensional frequency index in the *decoding process* prior to *scaling* for computation of a *transform coefficient* value.

3.110 **tree**: A tree is a finite ste of nodes with a unique root node. A terminating node is called a *leaf*.

3.111 **treeblock**: A NxN *block* of *luma* samples and two corresponding *blocks* of *chroma* samples of a *picture* that has three sample arrays, or a NxN *block* of samples of a monochrome *picture* or a *picture* that is coded using three separate colour planes. The division of a *slice* into treeblocks is a *partitioning*.

3.112 **treeblock address**: A treeblock address is the index of a *treeblock* in a *treeblock raster scan* of the *picture* starting with zero for the top-left *treeblock* in a *picture*.

3.113 **treeblock location**: The two-dimensional coordinates of a *treeblock* in a *picture* denoted by (x, y). For the top left *treeblock* of the *picture* (x, y) is equal to (0, 0). x is incremented by 1 for each *treeblock* column from left to right. The value of y is incremented by 1 for each *treeblock* row from top to bottom.

3.114 **treeblock partition**: A *block* of *luma* samples and two corresponding *blocks* of *chroma* samples resulting from a *partitioning* of a *treeblock* for *inter prediction* for a *picture* that has three sample arrays or a *block* of *luma* samples resulting from a *partitioning* of a *treeblock* for *inter prediction* for a monochrome *picture* or a *picture* that is coded using three separate colour planes.

3.115 **universal unique identifier (UUID)**: An identifier that is unique with respect to the space of all universal unique identifiers.

3.116 **unspecified:** The term unspecified, when used in the clauses specifying some values of a particular *syntax element*, indicates that the values have no specified meaning in this Recommendation | International Standard and will not have a specified meaning in the future as an integral part of this Recommendation | International Standard.

3.117 **variable length coding (VLC)**: A reversible procedure for entropy coding that assigns shorter bit strings to *symbols* expected to be more frequent and longer bit strings to *symbols* expected to be less frequent.

3.118 **VCL NAL unit**: A collective term for *coded slice NAL units*.

3.119 **zig-zag scan**: A specific sequential ordering of *transform coefficient levels* from (approximately) the lowest spatial frequency to the highest.

4 Abbreviations

For the purposes of this Recommendation | International Standard, the following abbreviations apply:

CABAC Context-based Adaptive Binary Arithmetic Coding

CAVLC Context-based Adaptive Variable Length Coding

CBR Constant Bit Rate

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000525

CRA	Clean Random Access
CPB	Coded Picture Buffer
DPB	Decoded Picture Buffer
DUT	Decoder under test
FIFO	First-In, First-Out
HRD	Hypothetical Reference Decoder
HSS	Hypothetical Stream Scheduler
IDR	Instantaneous Decoding Refresh
LSB	Least Significant Bit
MSB	Most Significant Bit
NAL	Network Abstraction Layer
RBSP	Raw Byte Sequence Payload
SEI	Supplemental Enhancement Information
SODB	String Of Data Bits
TB	Treeblock
UUID	Universal Unique Identifier
VBR	Variable Bit Rate
VCL	Video Coding Layer
VLC	Variable Length Coding
VUI	Video Usability Information

5 Conventions

NOTE – The mathematical operators used in this Specification are similar to those used in the C programming language. However, integer division and arithmetic shift operations are specifically defined. Numbering and counting conventions generally begin from 0.

5.1 Arithmetic operators

The following arithmetic operators are defined as follows:

+	Addition
–	Subtraction (as a two-argument operator) or negation (as a unary prefix operator)
*	Multiplication, including matrix multiplication
x^y	Exponentiation. Specifies x to the power of y. In other contexts, such notation is used for superscripting not intended for interpretation as exponentiation.
/	Integer division with truncation of the result toward zero. For example, 7/4 and –7/–4 are truncated to 1 and –7/4 and 7/–4 are truncated to –1.
÷	Used to denote division in mathematical equations where no truncation or rounding is intended.
$\frac{x}{y}$	Used to denote division in mathematical equations where no truncation or rounding is intended.
$\sum_{i=x}^{y} f(i)$	The summation of f(i) with i taking all integer values from x up to and including y.
x % y	Modulus. Remainder of x divided by y, defined only for integers x and y with x >= 0 and y > 0.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000526

5.2 Logical operators

The following logical operators are defined as follows:

- x && y Boolean logical "and" of x and y.
- x || y Boolean logical "or" of x and y.
- ! Boolean logical "not".
- x ? y : z If x is TRUE or not equal to 0, evaluates to the value of y; otherwise, evaluates to the value of z.

5.3 Relational operators

The following relational operators are defined as follows:

- > Greater than.
- >= Greater than or equal to.
- < Less than.
- <= Less than or equal to.
- == Equal to.
- != Not equal to.

When a relational operator is applied to a syntax element or variable that has been assigned the value "na" (not applicable), the value "na" is treated as a distinct value for the syntax element or variable. The value "na" is considered not to be equal to any other value.

5.4 Bit-wise operators

The following bit-wise operators are defined as follows:

- & Bit-wise "and". When operating on integer arguments, operates on a two's complement representation of the integer value. When operating on a binary argument that contains fewer bits than another argument, the shorter argument is extended by adding more significant bits equal to 0.
- | Bit-wise "or". When operating on integer arguments, operates on a two's complement representation of the integer value. When operating on a binary argument that contains fewer bits than another argument, the shorter argument is extended by adding more significant bits equal to 0.
- ^ Bit-wise "exclusive or". When operating on integer arguments, operates on a two's complement representation of the integer value. When operating on a binary argument that contains fewer bits than another argument, the shorter argument is extended by adding more significant bits equal to 0.
- x >> y Arithmetic right shift of a two's complement integer representation of x by y binary digits. This function is defined only for positive integer values of y. Bits shifted into the MSBs as a result of the right shift have a value equal to the MSB of x prior to the shift operation.
- x << y Arithmetic left shift of a two's complement integer representation of x by y binary digits. This function is defined only for positive integer values of y. Bits shifted into the LSBs as a result of the left shift have a value equal to 0.

5.5 Assignment operators

The following arithmetic operators are defined as follows:

- = Assignment operator.
- ++ Increment, i.e., x++ is equivalent to $x = x + 1$; when used in an array index, evaluates to the value of the variable prior to the increment operation.
- −− Decrement, i.e., x−− is equivalent to $x = x - 1$; when used in an array index, evaluates to the value of the variable prior to the decrement operation.
- += Increment by amount specified, i.e., x += 3 is equivalent to x = x + 3, and x += (−3) is equivalent to x = x + (−3).
- −= Decrement by amount specified, i.e., x −= 3 is equivalent to x = x − 3, and x −= (−3) is equivalent to x = x − (−3).

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000527

5.6 Range notation

The following notation is used to specify a range of values:

x = y..z x takes on integer values starting from y to z, inclusive, with x, y, and z being integer numbers.

5.7 Mathematical functions

The following mathematical functions are defined as follows:

$$\text{Abs}(x) = \begin{cases} x & ; \quad x >= 0 \\ -x & ; \quad x < 0 \end{cases} \tag{5-1}$$

Ceil(x) the smallest integer greater than or equal to x. (5-2)

$$\text{Clip1}_Y(x) = \text{Clip3}(0, (1 << \text{BitDepth}_Y) - 1, x) \tag{5-3}$$

$$\text{Clip1}_C(x) = \text{Clip3}(0, (1 << \text{BitDepth}_C) - 1, x) \tag{5-4}$$

$$\text{Clip3}(x, y, z) = \begin{cases} x & ; \quad z < x \\ y & ; \quad z > y \\ z & ; \quad \text{otherwise} \end{cases} \tag{5-5}$$

Floor(x) the greatest integer less than or equal to x. (5-6)

$$\text{InverseRasterScan}(a, b, c, d, e) = \begin{cases} (a \% (d / b)) * b & ; \quad e == 0 \\ (a / (d / b)) * c & ; \quad e == 1 \end{cases} \tag{5-7}$$

Log2(x) returns the base-2 logarithm of x. (5-8)

Log10(x) returns the base-10 logarithm of x. (5-9)

$$\text{Median}(x, y, z) = x + y + z - \text{Min}(x, \text{Min}(y, z)) - \text{Max}(x, \text{Max}(y, z)) \tag{5-10}$$

$$\text{Min}(x, y) = \begin{cases} x & ; \quad x <= y \\ y & ; \quad x > y \end{cases} \tag{5-11}$$

$$\text{Max}(x, y) = \begin{cases} x & ; \quad x >= y \\ y & ; \quad x < y \end{cases} \tag{5-12}$$

$$\text{Round}(x) = \text{Sign}(x) * \text{Floor}(\text{Abs}(x) + 0.5) \tag{5-13}$$

$$\text{Sign}(x) = \begin{cases} 1 & ; \quad x >= 0 \\ -1 & ; \quad x < 0 \end{cases} \tag{5-14}$$

$$\text{Sqrt}(x) = \sqrt{x} \tag{5-15}$$

5.8 Order of operation precedence

When order of precedence in an expression is not indicated explicitly by use of parentheses, the following rules apply:

– operations of a higher precedence are evaluated before any operation of a lower precedence,

– operations of the same precedence are evaluated sequentially from left to right.

Table 5-1 specifies the precedence of operations from highest to lowest; a higher position in the table indicates a higher precedence.

NOTE – For those operators that are also used in the C programming language, the order of precedence used in this Specification is the same as used in the C programming language.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000528

Table 5-1 – Operation precedence from highest (at top of table) to lowest (at bottom of table)

operations (with operands x, y, and z)
"x++", "x– –"
"!x", "–x" (as a unary prefix operator)
x^y
"x * y", "x / y", "x ÷ y", "$\frac{x}{y}$", "x % y"
"x + y", "x – y" (as a two-argument operator), "$\sum_{i=x}^{y} f(i)$"
"x << y", "x >> y"
"x < y", "x <= y", "x > y", "x >= y"
"x = = y", "x != y"
"x & y"
"x | y"
"x && y"
"x | | y"
"x ? y : z"
"x = y", "x += y", "x –= y"

5.9 Variables, syntax elements, and tables

Syntax elements in the bitstream are represented in **bold** type. Each syntax element is described by its name (all lower case letters with underscore characters), its one or two syntax categories, and one or two descriptors for its method of coded representation. The decoding process behaves according to the value of the syntax element and to the values of previously decoded syntax elements. When a value of a syntax element is used in the syntax tables or the text, it appears in regular (i.e., not bold) type.

In some cases the syntax tables may use the values of other variables derived from syntax elements values. Such variables appear in the syntax tables, or text, named by a mixture of lower case and upper case letter and without any underscore characters. Variables starting with an upper case letter are derived for the decoding of the current syntax structure and all depending syntax structures. Variables starting with an upper case letter may be used in the decoding process for later syntax structures without mentioning the originating syntax structure of the variable. Variables starting with a lower case letter are only used within the subclause in which they are derived.

In some cases, "mnemonic" names for syntax element values or variable values are used interchangeably with their numerical values. Sometimes "mnemonic" names are used without any associated numerical values. The association of values and names is specified in the text. The names are constructed from one or more groups of letters separated by an underscore character. Each group starts with an upper case letter and may contain more upper case letters.

NOTE – The syntax is described in a manner that closely follows the C-language syntactic constructs.

Functions that specify properties of the current position in the bitstream are referred to as syntax functions. These functions are specified in subclause 0 and assume the existence of a bitstream pointer with an indication of the position of the next bit to be read by the decoding process from the bitstream. Syntax functions are described by their names, which are constructed as syntax element names and end with left and right round parentheses including zero or more variable names (for definition) or values (for usage), separated by commas (if more than one variable).

Functions that are not syntax functions (including mathematical functions specified in subclause 5.7) are described by their names, which start with an upper case letter, contain a mixture of lower and upper case letters without any underscore character, and end with left and right parentheses including zero or more variable names (for definition) or values (for usage) separated by commas (if more than one variable).

A one-dimensional array is referred to as a list. A two-dimensional array is referred to as a matrix. Arrays can either be syntax elements or variables. Subscripts or square parentheses are used for the indexing of arrays. In reference to a visual depiction of a matrix, the first subscript is used as a row (vertical) index and the second subscript is used as a column

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000529

(horizontal) index. The indexing order is reversed when using square parentheses rather than subscripts for indexing. Thus, an element of a matrix s at horizontal position x and vertical position y may be denoted either as s[x, y] or as s_{yx}.

Binary notation is indicated by enclosing the string of bit values by single quote marks. For example, '01000001' represents an eight-bit string having only its second and its last bits (counted from the most to the least significant bit) equal to 1.

Hexadecimal notation, indicated by prefixing the hexadecimal number by "0x", may be used instead of binary notation when the number of bits is an integer multiple of 4. For example, 0x41 represents an eight-bit string having only its second and its last bits (counted from the most to the least significant bit) equal to 1.

Numerical values not enclosed in single quotes and not prefixed by "0x" are decimal values.

A value equal to 0 represents a FALSE condition in a test statement. The value TRUE is represented by any value different from zero.

5.10 Text description of logical operations

In the text, a statement of logical operations as would be described in pseudo-code as

```
if( condition 0 )
  statement 0
else if ( condition 1 )
  statement 1
…
else /* informative remark on remaining condition */
  statement n
```

may be described in the following manner:

... as follows / ... the following applies.

- If condition 0, statement 0
- Otherwise, if condition 1, statement 1
- …
- Otherwise (informative remark on remaining condition), statement n

Each "If ... Otherwise, if ... Otherwise, ..." statement in the text is introduced with "... as follows" or "... the following applies" immediately followed by "If ... ". The last condition of the "If ... Otherwise, if ... Otherwise, ..." is always an "Otherwise, ...". Interleaved "If ... Otherwise, if ... Otherwise, ..." statements can be identified by matching "... as follows" or "... the following applies" with the ending "Otherwise, ...".

In the text, a statement of logical operations as would be described in pseudo-code as

```
if( condition 0a  &&  condition 0b )
  statement 0
else if ( condition 1a  ||  condition 1b )
  statement 1
…
else
  statement n
```

may be described in the following manner:

... as follows / ... the following applies.

- If all of the following conditions are true, statement 0
  - condition 0a
  - condition 0b
- Otherwise, if any of the following conditions are true, statement 1
  - condition 1a
  - condition 1b
- …
- Otherwise, statement n

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000530

In the text, a statement of logical operations as would be described in pseudo-code as:

```
if( condition 0 )
  statement 0
if ( condition 1 )
  statement 1
```

may be described in the following manner:

When condition 0, statement 0

When condition 1, statement 1

5.11 Processes

Processes are used to describe the decoding of syntax elements. A process has a separate specification and invoking. All syntax elements and upper case variables that pertain to the current syntax structure and depending syntax structures are available in the process specification and invoking. A process specification may also have a lower case variable explicitly specified as the input. Each process specification has explicitly specified an output. The output is a variable that can either be an upper case variable or a lower case variable.

When invoking a process, the assignment of variables is specified as follows.

- If the variables at the invoking and the process specification do not have the same name, the variables are explicitly assigned to lower case input or output variables of the process specification.
- Otherwise (the variables at the invoking and the process specification have the same name), assignment is implied.

In the specification of a process, a specific macroblock may be referred to by the variable name having a value equal to the address of the specific macroblock.

6 Source, coded, decoded and output data formats, scanning processes, and neighbouring relationships

6.1 Bitstream formats

This subclause specifies the relationship between the NAL unit stream and byte stream, either of which are referred to as the bitstream.

The bitstream can be in one of two formats: the NAL unit stream format or the byte stream format. The NAL unit stream format is conceptually the more "basic" type. It consists of a sequence of syntax structures called NAL units. This sequence is ordered in decoding order. There are constraints imposed on the decoding order (and contents) of the NAL units in the NAL unit stream.

The byte stream format can be constructed from the NAL unit stream format by ordering the NAL units in decoding order and prefixing each NAL unit with a start code prefix and zero or more zero-valued bytes to form a stream of bytes. The NAL unit stream format can be extracted from the byte stream format by searching for the location of the unique start code prefix pattern within this stream of bytes. Methods of framing the NAL units in a manner other than use of the byte stream format are outside the scope of this Recommendation | International Standard. The byte stream format is specified in Annex B.

6.2 Source, decoded, and output picture formats

This subclause specifies the relationship between source and decoded pictures that is given via the bitstream.

The video source that is represented by the bitstream is a sequence of pictures in decoding order.

The source and decoded pictures are each comprised of one or more sample arrays:

- Luma (Y) only (monochrome).
- Luma and two chroma (YCbCr or YCgCo).
- Green, Blue and Red (GBR, also known as RGB).
- Arrays representing other unspecified monochrome or tri-stimulus colour samplings (for example, YZX, also known as XYZ).

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000531

For convenience of notation and terminology in this Specification, the variables and terms associated with these arrays are referred to as luma (or L or Y) and chroma, where the two chroma arrays are referred to as Cb and Cr; regardless of the actual colour representation method in use. The actual colour representation method in use can be indicated in syntax that is specified in Annex E.

The variables SubWidthC, and SubHeightC are specified in Table 6-1, depending on the chroma format sampling structure, which is specified through chroma_format_idc and separate_colour_plane_flag. An entry marked as "-" in Table 6-1 denotes an undefined value for SubWidthC or SubHeightC. Other values of chroma_format_idc, SubWidthC, and SubHeightC may be specified in the future by ITU-T | ISO/IEC.

Table 6-1 – SubWidthC, and SubHeightC values derived from chroma_format_idc and separate_colour_plane_flag

chroma_format_idc	separate_colour_plane_flag	Chroma Format	SubWidthC	SubHeightC
0	0	monochrome	-	-
1	0	4:2:0	2	2
2	0	4:2:2	2	1
3	0	4:4:4	1	1
3	1	4:4:4	-	-

In monochrome sampling there is only one sample array, which is nominally considered the luma array.

In 4:2:0 sampling, each of the two chroma arrays has half the height and half the width of the luma array.

In 4:2:2 sampling, each of the two chroma arrays has the same height and half the width of the luma array.

In 4:4:4 sampling, depending on the value of separate_colour_plane_flag, the following applies.

– If separate_colour_plane_flag is equal to 0, each of the two chroma arrays has the same height and width as the luma array.

– Otherwise (separate_colour_plane_flag is equal to 1), the three colour planes are separately processed as monochrome sampled pictures.

The number of bits necessary for the representation of each of the samples in the luma and chroma arrays in a video sequence is in the range of 8 to 14, and the number of bits used in the luma array may differ from the number of bits used in the chroma arrays.

When the value of chroma_format_idc is equal to 1, the nominal vertical and horizontal relative locations of luma and chroma samples in pictures are shown in Figure 6-1. Alternative chroma sample relative locations may be indicated in video usability information (see Annex E).

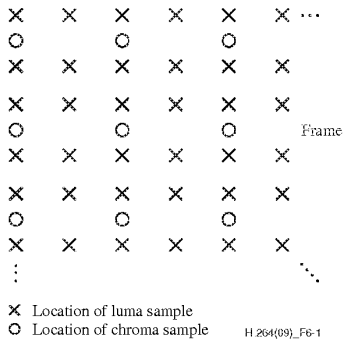


Figure 6-1 – Nominal vertical and horizontal locations of 4:2:0 luma and chroma samples in a picture [Ed. Re-draw figure]

When the value of chroma_format_idc is equal to 2, the chroma samples are co-sited with the corresponding luma samples and the nominal locations in a picture are as shown in Figure 6-2.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000532

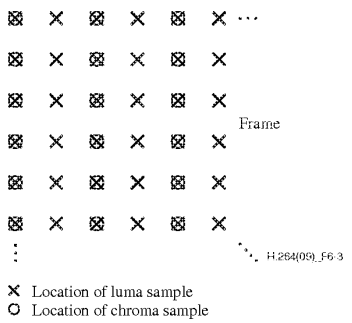


Figure 6-2 – Nominal vertical and horizontal locations of 4:2:2 luma and chroma samples in a picture [Ed. Re-draw figure]

When the value of chroma_format_idc is equal to 3, all array samples are co-sited for all cases of pictures and the nominal locations in a picture are as shown in Figure 6-3.

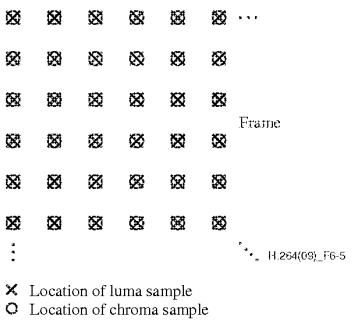


Figure 6-3 – Nominal vertical and horizontal locations of 4:4:4 luma and chroma samples in a picture [Ed. Re-draw figure]

The samples are processed in units of treeblocks. The luma array for each treeblock in samples in both width and height is

$$TbSize = 64 \quad (6\text{-}1)$$
$$Log2TbSize = 6 \quad (6\text{-}2)$$

The variables TbWidthC and TbHeightC, which specify the width and height, respectively, of the chroma arrays for each treeblock, are derived as follows.

– If chroma_format_idc is equal to 0 (monochrome) or separate_colour_plane_flag is equal to 1, TbWidthC and TbHeightC are both equal to 0.

– Otherwise, TbWidthC and TbHeightC are derived as

$$TbWidthC = TbSize / SubWidthC \quad (6\text{-}3)$$
$$TbHeightC = TbSize / SubHeightC \quad (6\text{-}4)$$

6.3 Spatial subdivision of pictures and slices

This subclause specifies how a picture is partitioned into slices and treeblocks. Pictures are divided into slices. A slice is a sequence of treeblocks.

Each treeblock is comprised of one Ntb x Ntb luma array and, when the chroma sampling format is not equal to 4:0:0 and separate_colour_plane_flag is equal to 0, two corresponding chroma sample arrays. When separate_colour_plane_flag is

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000533

equal to 1, each treeblock is comprised of one Ntb x Ntb luma or chroma sample array. For example, a picture may be divided into two slices as shown in Figure 6-4.

When a picture is coded using three separate colour planes (separate_colour_plane_flag is equal to 1), a slice contains only treeblocks of one colour component being identified by the corresponding value of colour_plane_id, and each colour component array of a picture consists of slices having the same colour_plane_id value. Coded slices with different values of colour_plane_id within an access unit can be interleaved with each other under the constraint that for each value of colour_plane_id, the coded slice NAL units with that value colour_plane_id shall be in the order of increasing treeblock address for the first treeblock of each coded slice NAL unit.

NOTE – When separate_colour_plane_flag is equal to 0, each treeblock of a picture is contained in exactly one slice. When separate_colour_plane_flag is equal to 1, each treeblock of a colour component is contained in exactly one slice (i.e., information for each treeblock of a picture is present in exactly three slices and these three slices have different values of colour_plane_id).

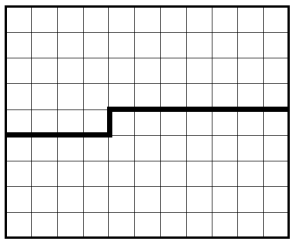


Figure 6-4 – A picture with 11 by 9 treeblocks that is partitioned into two slices

Each treeblock is assigned a partition signalling to identify the block sizes for intra or inter prediction and for transform coding. The partitioning is a recursive quadtree partitioning. The root of the quadtree is associated with the treeblock. The quadtree is split until a leaf is reached, which is referred to as the coding node. The coding node is the root node of two trees, the prediction tree and the transform tree.

The prediction tree specifies the position and size of prediction blocks. The prediction tree and associated prediction data are referred to as prediction unit.

The transform tree specifies the position and size of transform blocks. The transform tree and associated transform data are referred to as transform unit.

The splitting information for luma and chroma is identical for the prediction tree and may or may not be identical for the transform tree.

The coding node and the associated prediction and transform units form together a coding unit.

[Ed.: (WJ) block = a rectangular 2D array (one component), unit = collective term for specifying information for both luma and chroma. Don't use ther term 'unit' by itself – always use the term 'unit' with prefix – coding unit, prediction unit or transform unit.]

6.4 Inverse scanning processes and derivation processes for neighbours

This subclause specifies inverse scanning processes; i.e., the mapping of indices to locations, and derivation processes for neighbours.

6.4.1 Inverse treeblock scanning process

Input to this process is a treeblock address tbAddr.

Output of this process is the location (x, y) of the upper-left luma sample for the treeblock with address tbAddr relative to the upper-left sample of the picture.

The inverse treeblock scanning process is specified as follows.

$$x = \text{InverseRasterScan}(tbAddr, 16, 16, \text{PicWidthInSamples}_L, 0) \quad (6\text{-}5)$$

$$y = \text{InverseRasterScan}(tbAddr, 16, 16, \text{PicWidthInSamples}_L, 1) \quad (6\text{-}6)$$

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000534

[Ed. (TW): The text found in JCVTVC-B205 that relates to the rest of this clause requires substantial editorial and formatting work. It cannot be imported as is.]

6.4.2 Derivation process for the availability of treeblock addresses

Input to this process is a treeblock address tbAddr.

Output of this process is the availability of the treeblock tbAddr.

NOTE – The meaning of availability is determined when this process is invoked.

The treeblock is marked as available, unless one of the following conditions is true in which case the treeblock shall be marked as not available:

- tbAddr < 0
- tbAddr > CurrTbAddr
- the treeblock with address tbAddr belongs to a different slice than the treeblock with address CurrTbAddr and the [illegible]_slice_flag of the slice containing the treeblock with address CurrTbAddr is equal to 0.

[Ed. (WJ): do we need to separate the use of slice and [illegible] slice? E.g. Is [illegible] slice a kind of slice or sub-slice? We may need the concrete definitions]

6.5 Scanning array initialisation process

6.5.1 Zig-zag scanning array initialisation process

Input to this process is a block size width blkWidth and a block height blkHeight.

Output of this process is the array ZigZag[pos][comp]. The array index pos specify the scan position ranging from 0 to (blkWidthSize * blkHeightSize) – 1. The array index comp equal to 0 specifies the horizontal component and the array index comp equal to 1 specifies the vertical component. Depending on the values of blkWidth and blkHeight, the array ZigZag is derived as follows.

- If blkWidth is less than or equal to blkHeight, the following applies.

```
ZigZag[ 0 ][ 0 ] = 0
ZigZag[ 0 ][ 1 ] = 0
i = 1
x = 1
y = 0
stopLoop = ( blkHeight == 1 ) ? true : false
while( !stopLoop ) {
    while( x >= 0 ) {
        if (x >= 0 && x < blkWidth && y >= 0 && y < blkHeight ) {
            ZigZag[ i ][ 0 ] = x
            ZigZag[ i ][ 1 ] = y
            i++
        }
        x– –
        y++
    }
    x = 0
    while( y >= 0 ) {
        if (x >= 0 && x < blkWidth && y >= 0 && y < blkHeight ) {
            ZigZag[ i ][ 0 ] = x
            ZigZag[ i ][ 1 ] = y
            i++
        }
        x++
        y– –
    }
    y = 0
    if ( i >= blkWidth * blkHeight )
        stopLoop = true
}
```

- Otherwise (blkWidth is greater than blkHeight), the following applies.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000535

```
ZigZag[ 0 ][ 0 ] = 0
ZigZag[ 0 ][ 1 ] = 0
i = 1
x = 0
y = 1
stopLoop = false
while( !stopLoop ) {
    while( y >= 0 ) {
        if (x >= 0 && x < blkWidth && y >= 0 && y < blkHeight ) {
            ZigZag[ i ][ 0 ] = x
            ZigZag[ i ][ 1 ] = y
            i++
        }
        x++
        y--
    }
    y = 0
    while( x >= 0 ) {
        if (x >= 0 && x < blkWidth && y >= 0 && y < blkHeight ) {
            ZigZag[ i ][ 0 ] = x
            ZigZag[ i ][ 1 ] = y
            i++
        }
        x--
        y++
    }
    x = 0

    if ( i >= blkWidth * blkHeight )
        stopLoop = true
}
```

6.5.2 Up-right diagonal scanning array initialisation process

Input to this process is a block size width blkWidth and a block height blkHeight.

Output of this process is the array DiagScan[pos][comp]. The array index pos specify the scan position ranging from 0 to (blkWidthSize * blkHeightSize) – 1. The array index comp equal to 0 specifies the horizontal component and the array index comp equal to 1 specifies the vertical component. Depending on the values of blkWidth and blkHeight, the array DiagScan is derived as follows.

```
DiagScan[ 0 ][ 0 ] = 0
DiagScan [ 0 ][ 1 ] = 0
i = 1
x = 0
y = 1
stopLoop = ( blkWidth == 1 ) ? true : false
while( !stopLoop ) {
    while( y >= 0 ) {
        if (x >= 0 && x < blkWidth && y >= 0 && y < blkHeight ) {
            DiagScan [ i ][ 0 ] = x
            DiagScan [ i ][ 1 ] = y
            i++
        }
        y--
        x++
    }
    y = x
    x = 0
    if ( i >= blkWidth * blkHeight )
        stopLoop = true
}
```


Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000536

6.6 Scanning order array initialisation process

Input to this process is a block widthsize blkWidth and a block height blkHeight.

Output of this process is the array ScanOrder[scanIdx][pos][comp]. The array index scanIdx equal to 0 specifies a zig-zag scan as specified in subclause 6.5.1**.** with blkWidth and blkHeight as inputs, scanIdx equal to 1 specifies a horizontal scan, scanIdx equal to 2 specifies a vertical scan, and scanIdx equal to 3 specifies an up-right diagonal scan in subclause 6.5.2. The array index pos specifies the scan position ranging from 0 to (blkWidth * blkHeight) – 1. The array index comp equal to 0 specifies the horizontal component and the array index comp equal to 1 specifies the vertical component. The array ScanOrder is derived as follows.

```
ScanOrder[0] = ZigZag

i = 0
y = 0
while( y < blkHeight ) {
    x = 0
    while( x < blkWidth ) {
        ScanOrder[ 1 ][ i ][ 0 ] = x
        ScanOrder[ 1 ][ i ][ 1 ] = y
        x++
        i++
    }
    y++
}

i = 0
x = 0
while( x < blkWidth ) {
    y = 0
    while( y < blkHeight ) {
        ScanOrder[ 2 ][ i ][ 0 ] = x
        ScanOrder[ 2 ][ i ][ 1 ] = y
        y++
        i++
    }
    x++
}

ScanOrder[3] = DiagScan
```

7 Syntax and semantics

7.1 Method of specifying syntax in tabular form

The syntax tables specify a superset of the syntax of all allowed bitstreams. Additional constraints on the syntax may be specified, either directly or indirectly, in other clauses.

NOTE – An actual decoder should implement means for identifying entry points into the bitstream and means to identify and handle non-conforming bitstreams. The methods for identifying and handling errors and other such situations are not specified here.

The following table lists examples of pseudo code used to describe the syntax. When **syntax_element** appears, it specifies that a syntax element is parsed from the bitstream and the bitstream pointer is advanced to the next position beyond the syntax element in the bitstream parsing process.

	C	Descriptor
/* A statement can be a syntax element with an associated syntax category and descriptor or can be an expression used to specify conditions for the existence, type, and quantity of syntax elements, as in the following two examples */		
syntax_element conditioning statement	3	ue(v)
/* A group of statements enclosed in curly brackets is a compound statement and is treated functionally as a single statement. */		
{		

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000537

statement			
statement			
…			
}			
/* A "while" structure specifies a test of whether a condition is true, and if true, specifies evaluation of a statement (or compound statement) repeatedly until the condition is no longer true */			
while(condition)			
statement			
/* A "do … while" structure specifies evaluation of a statement once, followed by a test of whether a condition is true, and if true, specifies repeated evaluation of the statement until the condition is no longer true */			
do			
statement			
while(condition)			
/* An "if … else" structure specifies a test of whether a condition is true, and if the condition is true, specifies evaluation of a primary statement, otherwise, specifies evaluation of an alternative statement. The "else" part of the structure and the associated alternative statement is omitted if no alternative statement evaluation is needed */			
if(condition)			
primary statement			
else			
alternative statement			
/* A "for" structure specifies evaluation of an initial statement, followed by a test of a condition, and if the condition is true, specifies repeated evaluation of a primary statement followed by a subsequent statement until the condition is no longer true. */			
for(initial statement; condition; subsequent statement)			
primary statement			

7.2 Specification of syntax functions and descriptors

The functions presented here are used in the syntactical description. These functions assume the existence of a bitstream pointer with an indication of the position of the next bit to be read by the decoding process from the bitstream.

byte_aligned( ) is specified as follows.

- If the current position in the bitstream is on a byte boundary, i.e., the next bit in the bitstream is the first bit in a byte, the return value of byte_aligned( ) is equal to TRUE.
- Otherwise, the return value of byte_aligned( ) is equal to FALSE.

more_data_in_byte_stream( ), which is used only in the byte stream NAL unit syntax structure specified in Annex B, is specified as follows.

- If more data follow in the byte stream, the return value of more_data_in_byte_stream( ) is equal to TRUE.
- Otherwise, the return value of more_data_in_byte_stream( ) is equal to FALSE.

more_rbsp_data( ) is specified as follows.

- If there is no more data in the RBSP, the return value of more_rbsp_data( ) is equal to FALSE.
- Otherwise, the RBSP data is searched for the last (least significant, right-most) bit equal to 1 that is present in the RBSP. Given the position of this bit, which is the first bit (rbsp_stop_one_bit) of the rbsp_trailing_bits( ) syntax structure, the following applies.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000538

- If there is more data in an RBSP before the rbsp_trailing_bits( ) syntax structure, the return value of more_rbsp_data( ) is equal to TRUE.
- Otherwise, the return value of more_rbsp_data( ) is equal to FALSE.

The method for enabling determination of whether there is more data in the RBSP is specified by the application (or in Annex B for applications that use the byte stream format).

more_rbsp_trailing_data( ) is specified as follows.

- If there is more data in an RBSP, the return value of more_rbsp_trailing_data( ) is equal to TRUE.
- Otherwise, the return value of more_rbsp_trailing_data( ) is equal to FALSE.

next_bits(n) provides the next bits in the bitstream for comparison purposes, without advancing the bitstream pointer. Provides a look at the next n bits in the bitstream with n being its argument. When used within the byte stream as specified in Annex B, next_bits(n) returns a value of 0 if fewer than n bits remain within the byte stream.

read_bits(n) reads the next n bits from the bitstream and advances the bitstream pointer by n bit positions. When n is equal to 0, read_bits(n) is specified to return a value equal to 0 and to not advance the bitstream pointer.

The following descriptors specify the parsing process of each syntax element. For some syntax elements, two descriptors, separated by a vertical bar, are used. In these cases, the left descriptors apply when entropy_coding_mode_flag is equal to 0 and the right descriptor applies when entropy_coding_mode_flag is equal to 1.

- ae(v): context-adaptive arithmetic entropy-coded syntax element. The parsing process for this descriptor is specified in subclause 9.3.
- b(8): byte having any pattern of bit string (8 bits). The parsing process for this descriptor is specified by the return value of the function read_bits(8).
- ce(v): context-adaptive variable-length entropy-coded syntax element with the left bit first. The parsing process for this descriptor is specified in subclause 9.2.
- f(n): fixed-pattern bit string using n bits written (from left to right) with the left bit first. The parsing process for this descriptor is specified by the return value of the function read_bits(n).
- i(n): signed integer using n bits. When n is "v" in the syntax table, the number of bits varies in a manner dependent on the value of other syntax elements. The parsing process for this descriptor is specified by the return value of the function read_bits(n) interpreted as a two's complement integer representation with most significant bit written first.
- me(v): mapped Exp-Golomb-coded syntax element with the left bit first. The parsing process for this descriptor is specified in subclause 9.1.
- se(v): signed integer Exp-Golomb-coded syntax element with the left bit first. The parsing process for this descriptor is specified in subclause 9.1.
- te(v): truncated Exp-Golomb-coded syntax element with left bit first. The parsing process for this descriptor is specified in subclause 9.1.
- u(n): unsigned integer using n bits. When n is "v" in the syntax table, the number of bits varies in a manner dependent on the value of other syntax elements. The parsing process for this descriptor is specified by the return value of the function read_bits(n) interpreted as a binary representation of an unsigned integer with most significant bit written first.
- ue(v): unsigned integer Exp-Golomb-coded syntax element with the left bit first. The parsing process for this descriptor is specified in subclause 9.1.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000539

7.3 Syntax in tabular form

7.3.1 NAL unit syntax

nal_unit(NumBytesInNALunit) {	Descriptor
forbidden_zero_bit	f(1)
nal_ref_flag	u(1)
nal_unit_type	u(6)
NumBytesInRBSP = 0	
nalUnitHeaderBytes = 1	
if(nal_unit_type == 1 || nal_unit_type == 4 || nal_unit_type == 5) {	
temporal_id	u(3)
output_flag	u(1)
reserved_one_4bits	u(4)
nalUnitHeaderBytes += 1	
}	
for(i = nalUnitHeaderBytes; i < NumBytesInNALunit; i++) {	
if(i + 2 < NumBytesInNALunit && next_bits(24) == 0x000003) {	
rbsp_byte[NumBytesInRBSP++]	b(8)
rbsp_byte[NumBytesInRBSP++]	b(8)
i += 2	
emulation_prevention_three_byte /* equal to 0x03 */	f(8)
} else	
rbsp_byte[NumBytesInRBSP++]	b(8)
}	
}	

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000540

7.3.2 Raw byte sequence payloads and RBSP trailing bits syntax

7.3.2.1 Sequence parameter set RBSP syntax

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000541

seq_parameter_set_rbsp( ) {	Descriptor
profile_idc	u(8)
reserved_zero_8bits /* equal to 0 */	u(8)
level_idc	u(8)
seq_parameter_set_id	ue(v)
max_temporal_layers_minus1	u(3)
pic_width_in_luma_samples	u(16)
pic_height_in_luma_samples	u(16)
[Ed. (BB): x,y padding syntax missing here, present in HM software.]	
bit_depth_luma_minus8	ue(v)
bit_depth_chroma_minus8	ue(v)
[Ed. (BB): chroma bit depth present in HM software but not used further.]	
pcm_bit_depth_luma_minus1	u(4)
pcm_bit_depth_chroma_minus1	u(4)
log2_max_frame_num_minus4	ue(v)
pic_order_cnt_type	ue(v)
if(pic_order_cnt_type == 0)	
log2_max_pic_order_cnt_lsb_minus4	ue(v)
else if(pic_order_cnt_type == 1) {	
delta_pic_order_always_zero_flag	u(1)
offset_for_non_ref_pic	se(v)
num_ref_frames_in_pic_order_cnt_cycle	ue(v)
for(i = 0; i < num_ref_frames_in_pic_order_cnt_cycle; i++)	
offset_for_ref_frame[i]	se(v)
}	
max_num_ref_frames	ue(v)
gaps_in_frame_num_value_allowed_flag	u(1)
[Ed. (BB): not present in HM software, depends on reference list construction decision.]	
log2_min_coding_block_size_minus3	ue(v)
log2_diff_max_min_coding_block_size	ue(v)
log2_min_transform_block_size_minus2	ue(v)
log2_diff_max_min_transform_block_size	ue(v)
log2_min_pcm_coding_block_size_minus3	ue(v)
max_transform_hierarchy_depth_inter	ue(v)
max_transform_hierarchy_depth_intra	ue(v)
chroma_pred_from_luma_enabled_flag	u(1)
loop_filter_across_slice_flag	u(1)
sample_adaptive_offset_enabled_flag	u(1)
adaptive_loop_filter_enabled_flag	u(1)
pcm_loop_filter_disable_flag	u(1)
cu_qp_delta_enabled_flag	u(1)
temporal_id_nesting_flag	u(1)
inter_4x4_enabled_flag	u(1)
rbsp_trailing_bits( )	
}	

Formatted: Font: Not Bold
Formatted: Font: Not Bold
Formatted: Font: Not Bold
Formatted: Highlight
Formatted: Highlight
Formatted: Highlight
Formatted: Highlight
Formatted: Highlight
Formatted: Highlight
Formatted: Highlight
Formatted: Highlight
Formatted: Highlight
Formatted: Highlight
Formatted: Highlight
Formatted: Highlight

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000542

7.3.2.2 Picture parameter set RBSP syntax

pic_parameter_set_rbsp( ) {	Descriptor
pic_parameter_set_id	ue(v)
seq_parameter_set_id	ue(v)
entropy_coding_mode_flag	u(1)
num_temporal_layer_switching_point_flags	ue(v)
for(i = 0; i < num_temporal_layer_switching_point_flags; i++)	
temporal_layer_switching_point_flag[i]	u(1)
num_ref_idx_l0_default_active_minus1	ue(v)
num_ref_idx_l1_default_active_minus1	ue(v)
[Ed. (BB): not present in HM software, depends on reference list construction decision]	
pic_init_qp_minus26 /* relative to 26 */	se(v)
[Ed. (BB): not present in HM software, signaled in slice header as absolute value slice_qp, should be implemented to be used for slice_qp_delta]	
constrained_intra_pred_flag	u(1)
slice_granularity	u(2)
~~**shared_pps_info_enabled_flag**~~	~~u(1)~~
~~if(shared_pps_info_enabled_flag)~~	
~~if(adaptive_loop_filter_enabled_flag)~~	
~~alf_param( )~~	
~~if(cu_qp_delta_enabled_flag)~~	
~~**max_cu_qp_delta_depth**~~	~~ue(v)~~
[Ed. (BB): not present in HM software, should be implemented]	
rbsp_trailing_bits( )	
}	

Formatted: Highlight

Formatted: Highlight

Formatted: Highlight

Formatted: Highlight

Formatted: Highlight

7.3.2.3 Adaptation parameter set RBSP syntax

Formatted: Heading 4,Heading 4 Char1,Heading 4 Char Char, Space Before: 0 pt, After: 0 pt, No bullets or numbering,Tab stops: Not at 0.25" + 0.5" + 0.75" + 1"

aps_rbsp( ) {	Descriptor
aps_id	ue(v)
aps_sample_adaptive_offset_flag	u(1)
aps_adaptive_loop_filter_flag	u(1)
if(aps_sample_adaptive_offset_flag || aps_adaptive_loop_filter_flag) {	
aps_cabac_use_flag	u(1)
if(aps_cabac_use_flag) {	
aps_cabac_init_idc	ue(v)
aps_cabac_init_qp_minus26	se(v)
}	
}	
if(aps_sample_adaptive_offset_flag) {	
sao_data_byte_count	u(8)
byte_align( )	
sao_param( )	
byte_align( )	
}	
if(aps_adaptive_loop_filter_flag) {	

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000543

alf_data_byte_count	u(8)
byte_align( )	
alf_param( )	
}	
rbsp_trailing_bits( )	
}	

7.3.2.47.3.2.4 **Supplemental enhancement information RBSP syntax**

sei_rbsp( ) {	Descriptor
do	
sei_message( )	
while(more_rbsp_data( ))	
rbsp_trailing_bits( )	
}	

7.3.2.47.3.2.4.1 **Supplemental enhancement information message syntax**

sei_message( ) {	Descriptor
payloadType = 0	
while(next_bits(8) == 0xFF) {	
ff_byte /* equal to 0xFF */	f(8)
payloadType += 255	
}	
last_payload_type_byte	u(8)
payloadType += last_payload_type_byte	
payloadSize = 0	
while(next_bits(8) == 0xFF) {	
ff_byte /* equal to 0xFF */	f(8)
payloadSize += 255	
}	
last_payload_size_byte	u(8)
payloadSize += last_payload_size_byte	
sei_payload(payloadType, payloadSize)	
}	

7.3.2.47.3.2.5 **Access unit delimiter RBSP syntax**

access_unit_delimiter_rbsp( ) {	Descriptor
primary_pic_type	u(3)
rbsp_trailing_bits( )	
}	

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000544

7.3.2.67.3.2.6 Filler data RBSP syntax

filler_data_rbsp( ) {	Descriptor
while(next_bits(8) == 0xFF)	
ff_byte /* equal to 0xFF */	f(8)
rbsp_trailing_bits( )	
}	

7.3.2.67.3.2.7 Slice layer RBSP syntax

slice_layer_rbsp( ) {	Descriptor
slice_header( )	
slice_data( )	
rbsp_slice_trailing_bits( )	
}	

7.3.2.77.3.2.8 RBSP slice trailing bits syntax

rbsp_slice_trailing_bits( ) {	Descriptor
rbsp_trailing_bits( )	
if(entropy_coding_mode_flag)	
while(more_rbsp_trailing_data( ))	
cabac_zero_word /* equal to 0x0000 */	f(16)
}	

7.3.2.87.3.2.9 RBSP trailing bits syntax

rbsp_trailing_bits( ) {	Descriptor
rbsp_stop_one_bit /* equal to 1 */	f(1)
while(!byte_aligned( ))	
rbsp_alignment_zero_bit /* equal to 0 */	f(1)
}	

7.3.2.10 RBSP byte alignment syntax

byte_align( ) {	Descriptor
while(!byte_aligned( ))	
bit_equal_to_one	f(1)
}	

Formatted: Heading 4,Heading 4 Char1,Heading 4 Char Char

Formatted: Normal

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000545

7.3.3 Slice header syntax

slice_header( ) {	Descriptor
light_weight_copy_slice_flag	u(1)
if(!light_weight_copy_slice_flag) {	
slice_type	ue(v)
pic_parameter_set_id	ue(v)
if(sample_adaptive_offset_enabled_flag || adaptive_loop_filter_enabled_flag)	
aps_id	ue(v)
frame_num	u(v)
if(IdrPicFlag)	
idr_pic_id	ue(v)
/* (BB): not present in HM software, depends on reference list construction design */	
if(pic_order_cnt_type == 0)	
pic_order_cnt_lsb /*	u(v)
/* (BB): always ignored as pic_order_cnt_type == 1 in HM software, depends on reference list construction design */	
if(slice_type == P || slice_type == B) {	
num_ref_idx_active_override_flag	u(1)
if(num_ref_idx_active_override_flag) {	
num_ref_idx_l0_active_minus1	ue(v)
if(slice_type == B)	
num_ref_idx_l1_active_minus1	ue(v)
}	
}	
ref_pic_list_modification( )	
/* (BB): not present in HM software, depends on reference list construction design */	
ref_pic_list_combination( )	
if(nal_ref_flag != 0)	
dec_ref_pic_marking( )	
/* (BB): not present in HM software, depends on reference list construction design */	
}	
if(entropy_coding_mode_flag && slice_type != I)	
cabac_init_idc	ue(v)
first_slice_in_pic_flag	u(1)
if(first_slice_in_pic_flag == 0)	
slice_address	u(v)
if(!light_weight_copy_slice_flag) {	
slice_qp_delta	se(v)
/* (BB): coded as absolute value here as in HM software, should be implemented similar to mvd_l0 and mvd_l1 in future */	
if(sample_adaptive_offset_enabled_flag)	
sao_param()	
if(deblocking_filter_control_present_flag) {	
disable_deblocking_filter_idc	u(1)
/* (BB): Although it was agreed to use AVC-like deblocking control syntax, WD and HM should be changed to make use of the AVC-like syntax */	
if(disable_deblocking_filter_idc != 1) {	

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000546

slice_alpha_c0_offset_div2	
slice_beta_offset_div2	
}	
}	
if(slice_type = = B)	
collocated_from_l0_flag	u(1)
if(adaptive_loop_filter_enabled_flag && aps_adaptive_loop_filter_flag) {	
if(!shared_pps_info_enabled_flag)byte_align( )	
alf_param( )	
alf_cu_control_param( )	
byte_align( )	
}	
}	
}	

Formatted Table

7.3.3.1 Reference picture list modification syntax

ref_pic_list_modification( ) {	**Descriptor**
if(slice_type % 5 != 2 && slice_type % 5 != 4) {	
ref_pic_list_modification_flag_l0	u(1)
if(ref_pic_list_modification_flag_l0)	
do {	
modification_of_pic_nums_idc	ue(v)
if(modification_of_pic_nums_idc = = 0 || modification_of_pic_nums_idc = = 1)	
abs_diff_pic_num_minus1	ue(v)
else if(modification_of_pic_nums_idc = = 2)	
long_term_pic_num	ue(v)
} while(modification_of_pic_nums_idc != 3)	
}	
if(slice_type % 5 = = 1) {	
ref_pic_list_modification_flag_l1	u(1)
if(ref_pic_list_modification_flag_l1)	
do {	
modification_of_pic_nums_idc	ue(v)
if(modification_of_pic_nums_idc = = 0 || modification_of_pic_nums_idc = = 1)	
abs_diff_pic_num_minus1	ue(v)
else if(modification_of_pic_nums_idc = = 2)	
long_term_pic_num	ue(v)
} while(modification_of_pic_nums_idc != 3)	
}	
}	

7.3.3.2 Reference picture lists combination syntax

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000547

ref_pic_list_combination( ) {	Descriptor
if(slice_type % 5 = = 1) { // b slice	
ref_pic_list_combination_flag	u(1)
if(ref_pic_list_combination_flag) {	
num_ref_idx lc_active_minus1	ue(v)
ref_pic_list_modification_flag_lc	u(1)
if(ref_pic_list_modification_flag_lc)	
for (i =0; i <= num_ref_idx_lc_active_minus1; i++) {	
pic_from_list_0_flag	u(1)
ref_idx_list_curr	ue(v)
}	
}	
}	
}	

7.3.3.3 Decoded reference picture marking syntax

dec_ref_pic_marking( ) {	Descriptor
if(IdrPicFlag) {	
no_output_of_prior_pics_flag	u(1)
long_term_reference_flag	u(1)
} else {	
adaptive_ref_pic_marking_mode_flag	u(1)
if(adaptive_ref_pic_marking_mode_flag)	
do {	
memory_management_control_operation	ue(v)
if(memory_management_control_operation = = 1 || memory_management_control_operation = = 3)	
difference_of_pic_nums_minus1	ue(v)
if(memory_management_control_operation = = 2)	
long_term_pic_num	ue(v)
if(memory_management_control_operation = = 3 || memory_management_control_operation = = 6)	
long_term_frame_idx	ue(v)
if(memory_management_control_operation = = 4)	
max_long_term_frame_idx_plus1	ue(v)
} while(memory_management_control_operation != 0)	
}	
}	

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000548

7.3.3.4 Sample adaptive offset parameter syntax

sao_param( ) {	Descriptor
····sample_adaptive_offset_flag	u(1)
····if(sample_adaptive_offset_flag) {	
····sao_split_param(0, 0, 0, 0)	
····sao_offset_param(0, 0, 0, 0)	
····**sao_flag_cb**	u(1) | ae(v)
····if(sao_flag_cb) {	
···· sao_split_param(0, 0, 0, 1)	
···· sao_split_param(0, 0, 0, 1)	
····}	
····**sao_flag_cr**	u(1) | ae(v)
····if(sao_flag_cr) {	
···· sao_split_param(0, 0, 0, 2)	
···· sao_split_param(0, 0, 0, 2)	
····}	
····}	
}	

sao_split_param(rx, ry, saoDepth , cIdx) {	Descriptor
if(saoDepth < SaoMaxDepth)	
sao_split_flag[cIdx][saoDepth][rx][ry]	u(1) | ae(v)
else	
sao_split_flag[cIdx][saoDepth][rx][ry] = 0	
if(sao_split_flag[cIdx][saoDepth][rx][ry]) {	
sao_split_param(2*rx + 0, 2*ry + 0, saoDepth + 1 , cIdx)	
sao_split_param(2*rx + 1, 2*ry + 0, saoDepth + 1 , cIdx)	
sao_split_param(2*rx + 0, 2*ry + 1, saoDepth + 1 , cIdx)	
sao_split_param(2*rx + 1, 2*ry + 1, saoDepth + 1 , cIdx)	
}	
}	

sao_offset_param(rx, ry, saoDepth , cIdx) {	Descriptor
if(sao_split_flag[cIdx][saoDepth][rx][ry]) {	
sao_offset_param(2*rx + 0, 2*ry + 0, saoDepth + 1 , cIdx)	
sao_offset_param(2*rx + 1, 2*ry + 0, saoDepth + 1 , cIdx)	
sao_offset_param(2*rx + 0, 2*ry + 1, saoDepth + 1 , cIdx)	
sao_offset_param(2*rx + 1, 2*ry + 1, saoDepth + 1 , cIdx)	
} else {	
sao_type_idx[cIdx][saoDepth][rx][ry]	ue(v) | ae(v)
if(sao_type_idx[cIdx][saoDepth][rx][ry] != 0)	
for(i = 0; i < NumSaoClass[sao_type_idx]; i++)	
sao_offset[cIdx][saoDepth][x0][y0][i]	se(v) | ae(v)
}	
}	

[Ed.: (WJ) is it better to assign separate section?]

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000549

7.3.3.5 Adaptive loop filter parameter syntax

alf_param() {	Descriptor
~~adaptive_loop_filter_flag~~	~~ae(1)~~
~~if (adaptive_loop_filter_flag) {~~	
alf_region_adaptation_flag	u(1) | ae(v)
alf_length_luma_minus_5_div2	ue(v) | ae(v)
alf_no_filters_minus1	ue(v) | ae(v)
if (alf_no_filters_minus1 == 1)	
alf_start_second_filter	ue(v) | ae(v)
else if (alf_no_filters_minus1 > 1) {	
for (i=1; i< 16; i++)	
alf_filter_pattern[i]	u(1) | ae(v)
}	
if (AlfNumFilters > 1)	
alf_pred_method	u(1) | ae(v)
alf_min_kstart_minus1	ue(v) | ae(v)
for (i=0; i < AlfMaxDepth; i++)	
alf_golomb_index_bit[i]	u(1) | ae(v)
for (i=0; i< AlfNumFilters; i++)	
for (j=0; j< AlfCodedLengthLuma; j++)	
alf_coeff_luma[i][j]	ge(v) | ae(v)
alf_chroma_idc	ue(v) | ae(v)
if (alf_chroma_idc) {	
alf_length_chroma_minus_5_div2	ue(v) | ae(v)
for(i = 0; i< AlfCodedLengthChroma; i++)	
alf_coeff_chroma[i]	se(v) | ae(v)
}	
~~}~~	
}	

Formatted Table

7.3.3.6 Adaptive loop filter coding unit control parameter syntax

alf_cu_control_param() {	Descriptor
~~if(adaptive_loop_filter_flag) {~~	
alf_cu_control_flag	u(1) | ae(v)
if(alf_cu_control_flag) {	
alf_cu_control_max_depth	ue(v) | ae(v)
alf_length_cu_control_info	se(v) | ae(v)
for(i = 0; i < NumAlfCuFlag; i++)	
alf_cu_flag[i]	u(1) | ae(v)
}	
~~}~~	
}	

Formatted Table

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000550

7.3.4 Slice data syntax

slice_data( ) {	Descriptor
CurrTbAddr = LCUAddress	
moreDataFlag = 1	
if(adaptive_loop_filter_flag && alf_cu_control_flag)	
AlfCuFlagIdx = -1	
do {	
xCU = HorLumaLocation(CurrTbAddr)	
yCU = VerLumaLocation(CurrTbAddr)	
moreDataFlag = coding_tree(xCU, yCU, Log2TbSize, 0)	
CurrTbAddr = NextTbAddress(CurrTbAddr)	
} while(moreDataFlag)	
}	

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000551

7.3.5 Coding tree syntax

coding_tree(x0, y0, log2CUSize, cuDepth) {	Descriptor
if(x0 + (1 << log2CUSize) <= PicWidthInSamples$_L$ && y0 + (1 << log2CUSize) <= PicHeightInSamples$_L$ && cuAddress(x0, y0) >= SliceAddress) {	
if(!entropy_coding_mode_flag && slice_type != I)	
cu_split_pred_part_mode[x0][y0]	ce(v)
else if(log2CUSize > Log2MinCUSize)	
split_coding_unit_flag[x0][y0]	u(1) | ae(v)
}	
if(adaptive_loop_filter_flag && alf_cu_control_flag) {	
if(cuDepth <= alf_cu_control_max_depth)	
if(cuDepth == alf_cu_control_max_depth ||	
split_coding_unit_flag[x0][y0] == 0)	
AlfCuFlagIdx++	
}	
if(cu_qp_delta_enabled_flag && log2CUSize >= log2MinCUDQPSize)	
IsCuQpDeltaCoded = 0	
if(split_coding_unit_flag[x0][y0]) {	
x1 = x0 + ((1 << log2CUSize) >> 1)	
y1 = y0 + ((1 << log2CUSize) >> 1)	
if(cuAddress(x1, y0) > SliceAddress)	
moreDataFlag = coding_tree(x0, y0, log2CUSize – 1, cuDepth + 1)	
if(cuAddress(x0, y1) > SliceAddress && moreDataFlag && x1 < PicWidthInSamples$_L$)	
moreDataFlag = coding_tree(x1, y0, log2CUSize – 1, cuDepth + 1)	
if(cuAddress(x1, y1) > SliceAddress && moreDataFlag && y1 < PicHeightInSamples$_L$)	
moreDataFlag = coding_tree(x0, y1, log2CUSize – 1, cuDepth + 1)	
if(moreDataFlag && x1 < PicWidthInSamples$_L$ && y1 < PicHeightInSamples$_L$)	
moreDataFlag = coding_tree(x1, y1, log2CUSize – 1, cuDepth + 1)	
} else {	
if(adaptive_loop_filter_flag && alf_cu_control_flag)	
AlfCuFlag[x0][y0] = alf_cu_flag[AlfCuFlagIdx]	
coding_unit(x0, y0, log2CUSize)	
if(!entropy_coding_mode_flag)	
moreDataFlag = more_rbsp_data( )	
else {	
if(granularity_block_boundary(x0, y0, log2CUSize)) {	
end_of_slice_flag	ae(v)
moreDataFlag = !end_of_slice_flag	
} else	
moreDataFlag = 1	
}	
}	
return moreDataFlag	
}	

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000552

[Ed. (WJ): cuAddress(x, y) returns the address of the smallest coding unit at (x, y), expressed in global decoding order smallest coding unit coordinates. granularity_block_boundary(x0, y0, log2CUSize) returns true when the coding unit specified by (x0, y0, log2CUSize) is the last coding unit in the slice in global decoding order. These functions are not defined yet. They will be added later.]

7.3.6 Coding unit syntax

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000553

coding_unit(x0, y0, log2CUSize) {	Descriptor
if(entropy_coding_mode_flag && slice_type != I)	
skip_flag[x0][y0]	u(1) | ae(v)
if(skip_flag[x0][y0])	
prediction_unit(x0, y0 , log2CUSize)	
else {	
if(!entropy_coding_mode_flag) {	
if(slice_type == I && log2CUSize == Log2MinCUSize)	
intra_part_mode	u(1)
} else if(slice_type != I || log2CUSize == Log2MinCUSize)	
pred_type	u(v) | ae(v)
x1 = x0 + ((1 << log2CUSize) >> 1)	
y1 = y0 + ((1 << log2CUSize) >> 1)	
x2 = x1 − ((1 << log2CUSize) >> 2)	
y2 = y1 − ((1 << log2CUSize) >> 2)	
x3 = x1 + ((1 << log2CUSize) >> 2)	
y3 = y1 + ((1 << log2CUSize) >> 2)	
if(PartMode == PART_2Nx2N) {	
prediction_unit(x0, y0 , log2CUSize)	
} else if(PartMode == PART_2NxN) {	
prediction_unit(x0, y0 , log2CUSize)	
prediction_unit(x0, y1 , log2CUSize)	
} else if(PartMode == PART_Nx2N) {	
prediction_unit(x0, y0 , log2CUSize)	
prediction_unit(x1, y0 , log2CUSize)	
} else if(PartMode == PART_2NxnU) {	
prediction_unit(x0, y0 , log2CUSize)	
prediction_unit(x0, y2 , log2CUSize)	
} else if(PartMode == PART_2NxnD) {	
prediction_unit(x0, y0 , log2CUSize)	
prediction_unit(x0, y3 , log2CUSize)	
} else if(PartMode == PART_nLx2N) {	
prediction_unit(x0, y0 , log2CUSize)	
prediction_unit(x2, y0 , log2CUSize)	
} else if(PartMode == PART_nRx2N) {	
prediction_unit(x0, y0 , log2CUSize)	
prediction_unit(x3, y0 , log2CUSize)	
} else { /* PART_NxN */	
prediction_unit(x0, y0 , log2CUSize)	
prediction_unit(x1, y0 , log2CUSize)	
prediction_unit(x0, y1 , log2CUSize)	
prediction_unit(x1, y1 , log2CUSize)	
}	
}	
if(!pcm_flag) {	
transform_tree(x0, y0, log2CUSize, log2CUSize, 0, 0)	
transform_coeff(x0, y0, log2CUSize, log2CUSize, 0, 0)	
transform_coeff(x0, y0, log2CUSize, log2CUSize, 0, 1)	
transform_coeff(x0, y0, log2CUSize, log2CUSize, 0, 2)	
}	
}	
}	

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000554

7.3.7 Prediction unit syntax

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000555

prediction_unit(x0, y0, log2CUSize) {	Descriptor
if(skip_flag[x0][y0]) {	
merge_idx[x0][y0]	ue(v) | ae(v)
} else if(PredMode == MODE_INTRA) {	
if(PartMode == PART_2Nx2N && log2CUSize >= Log2MinIPCMCUSize)	
pcm_flag	u(1) | ae(v)
if(pcm_flag) {	
while (!byte_aligned( ))	
pcm_alignment_zero_bit	u(v)
for(i = 0; i < 1 << (log2CUSize << 1); i++)	
pcm_sample_luma[i]	u(v)
for(i = 0; i < (1 << (log2CUSize << 1)) >> 1; i++)	
pcm_sample_chroma[i]	u(v)
} else {	
prev_intra_luma_pred_flag[x0][y0]	u(1) | ae(v)
if(prev_intra_luma_pred_flag[x0][y0])	
mpm_idx[x0][y0]	u(1) | ae(v)
else	
rem_intra_luma_pred_mode[x0][y0]	ce(v) | ae(v)
intra_chroma_pred_mode[x0][y0]	ue(v) | ae(v)
SignaledAsChromaDC = (chroma_pred_from_luma_enabled_flag ? intra_chroma_pred_mode[x0][y0] == 3 : intra_chroma_pred_mode[x0][y0] == 2)	
}	
} else { /* MODE_INTER */	
if(entropy_coding_mode_flag || PartMode != PART_2Nx2N)	
merge_flag[x0][y0]	u(1) | ae(v)
if(merge_flag[x0][y0]) {	
merge_idx[x0][y0]	ue(v) | ae(v)
} else {	
if(slice_type == B) {	
if(!entropy_coding_mode_flag) {	
combined_inter_pred_ref_idx	ue(v)
if(combined_inter_pred_ref_idx == MaxPredRef)	
inter_pred_flag[x0][y0]	ue(v)
} else	
inter_pred_flag[x0][y0]	~~ue(v)~~ | ae(v)
}	
if(inter_pred_flag[x0][y0] == Pred_LC) {	
if(num_ref_idx_lc_active_minus1 > 0) {	
if(!entropy_coding_mode_flag) {	
if(combined_inter_pred_ref_idx == MaxPredRef)	
ref_idx_lc_minus4[x0][y0]	ue(v)
} else	
ref_idx_lc[x0][y0]	ae(v)
}	
if(entropy_coding_mode_flag) {	

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000556

mvd_coding_cabac(mvd_lc[x0][y0][0], mvd_lc[x0][y0][1])	
} else {	
mvd_lc[x0][y0][0]	se(v)
mvd_lc[x0][y0][1]	se(v)
}	
mvp_idx_lc[**x0**][**y0**]	ue(v) | ae(v)
}	
else { /* Pred_L0 or Pred_BI */	
if(num_ref_idx_l0_active_minus1 > 0) {	
if(!entropy_coding_mode_flag) {	
if(combined_inter_pred_ref_idx == MaxPredRef)	
ref_idx_l0_minusX[x0][y0]	ue(v)
} else	
ref_idx_l0_minusX[x0][y0]	~~ue(v)~~ | ae(v)
}	
if(entropy_coding_mode_flag) {	
mvd_coding_cabac(mvd_l0[x0][y0][0], mvd_l0[x0][y0][1])	
} else {	
mvd_l0[x0][y0][0]	se(v)
mvd_l0[x0][y0][1]	se(v)
}	
mvp_idx_l0[**x0**][**y0**]	ue(v) | ae(v)
}	
if(inter_pred_flag[x0][y0] == Pred_BI) {	
if(num_ref_idx_l1_active_minus1 > 0) {	
if(!entropy_coding_mode_flag) {	
if(combined_inter_pred_ref_idx == MaxPredRef)	
ref_idx_l1_minusX[x0][y0]	ue(v)
} else	
ref_idx_l1[x0][y0]	~~ue(v)~~ | ae(v)
}	
if(entropy_coding_mode_flag) {	
mvd_coding_cabac(mvd_l1[x0][y0][0], mvd_l1[x0][y0][1])	
} else {	
mvd_l1[x0][y0][0]	se(v)
mvd_l1[x0][y0][1]	se(v)
}	
mvp_idx_l1[**x0**][**y0**]	ue(v) | ae(v)
}	
}	
}	
}	

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000557

7.3.7.1 Motion vector difference coding CABAC syntax

mvd_coding_cabac(mvd_x, mvd_y) {	Descriptor
abs_mvd_greater0_flag[0]	ae(v)
abs_mvd_greater0_flag[1]	ae(v)
if(abs_mvd_greater0_flag[0])	
abs_mvd_greater1_flag[0]	ae(v)
if(abs_mvd_greater0_flag[1])	
abs_mvd_greater1_flag[1]	ae(v)
if(abs_mvd_greater0_flag[0]) {	
if(abs_mvd_greater1_flag[0])	
abs_mvd_minus2[0]	ae(v)
mvd_sign_flag[0]	ae(v)
}	
if(abs_mvd_greater0_flag[1]) {	
if(abs_mvd_greater1_flag[1])	
abs_mvd_minus2[1]	ae(v)
mvd_sign_flag[1]	ae(v)
}	
mvd_x = abs_mvd_greater0_flag[0] * (abs_mvd_minus2[0] + 2) * (1 – 2 * mvd_sign_flag[0])	
mvd_y = abs_mvd_greater0_flag[1] * (abs_mvd_minus2[1] + 2) * (1 – 2 * mvd_sign_flag[1])	
}	

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000558

7.3.8 Transform tree syntax

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000559

transform_tree(x0, y0, log2TrafoWidth, log2TrafoHeight, trafoDepth, blkIdx) {	Descriptor
if(entropy_coding_mode_flag && trafoDepth == 0 && IntraSplitFlag == 0) {	
if(PredMode != MODE_INTRA)	
no_residual_data_flag	ae(v)
residualDataPresentFlag = !no_residual_data_flag	
} else {	
residualDataPresentFlag = TRUE	
}	
if(residualDataPresentFlag) {	
log2TrafoSize = (log2TrafoWidth + log2TrafoHeight) >> 1	
intraSplitFlag = (IntraSplitFlag && trafoDepth == 0 ? 1 : 0)	
maxDepth = (PredMode == MODE_INTRA ? max_transform_hierarchy_depth_intra + IntraSplitFlag : max_transform_hierarchy_depth_inter)	
xBase = x0 – (x0 & (1 << log2TrafoWidth))	
yBase = y0 – (y0 & (1 << log2TrafoHeight))	
if(!entropy_coding_mode_flag) {	
if(trafoDepth != 0 && !cbf_luma[xBase][yBase][trafoDepth – 1]) {	
if(cbf_cb[xBase][yBase][trafoDepth – 1] && log2TrafoSize > Log2MinTrafoSize)	
cbf_cb[x0][y0] [trafoDepth]	u(1)
if(cbf_cr[xBase][yBase][trafoDepth – 1] && log2TrafoSize > Log2MinTrafoSize)	
cbf_cr[x0][y0] [trafoDepth]	u(1)
if(log2TrafoSize <= Log2MaxTrafoSize && !intraSplitFlag && log2TrafoSize > Log2MinTrafoSize && trafoDepth < maxDepth)	
split_transform_flag[x0][y0] [trafoDepth]	u(1)
} else	
cbp_and_split_transform	vlc(n,v)
}	
if(log2TrafoSize <= Log2MaxTrafoSize && log2TrafoSize > Log2MinTrafoSize && trafoDepth < maxDepth && !intraSplitFlag && entropy_coding_mode_flag)	
split_transform_flag[x0][y0][trafoDepth]	ae(v)

if(PredMode != MODE_INTRA &&
log2TrafoSize <= Log2MaxTrafoSize &&
entropy_coding_mode_flag) {	

firstChromaCbf = (log2TrafoSize == Log2MaxTrafoSize || trafoDepth == 0 ? 1 : 0)	
if(firstChromaCbf || log2TrafoSize > Log2MinTrafoSize) {	
if(firstChromaCbf || cbf_cb[xBase][yBase][trafoDepth – 1]) {	
readCbf = true	
if(blkIdx = = 3 && log2TrafoSize < Log2MaxTrafoSize)	
readCbf = cbf_cb[xBase][yBase][trafoDepth] || cbf_cb[xBase + (1 << log2TrafoWidth)][yBase][trafoDepth] || cbf_cb[xBase][yBase + (1 << log2TrafoHeight)][trafoDepth]	
if (!readCbf)	
cbf_cb[x0][y0][trafoDepth] = 1	
else	
cbf_cb[x0][y0][trafoDepth]	ae(v)
}	
if(firstChromaCbf || cbf_cr[xBase][yBase][trafoDepth – 1]) {	
readCbf = true	

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000560

if(blkIdx = = 3 && log2TrafoSize < Log2MaxTrafoSize)	
readCbf = cbf_cr[xBase][yBase][trafoDepth] || cbf_cr[xBase + (1 << log2TrafoWidth)][yBase][trafoDepth] || cbf_cr[xBase][yBase + (1 << log2TrafoHeight)][trafoDepth]	
if (!readCbf)	
cbf_cr[x0][y0][trafoDepth] = 1	
else	
cbf_cr[x0][y0][trafoDepth]	ae(v)
}	
}	
}	
if(split_transform_flag[x0][y0][trafoDepth]) {	
if(InterTUSplitDirection = = 2) {	
x1 = x0 + ((1 << log2TrafoWidth) >> 1)	
y1 = y0	
x2 = x0	
y2 = y0 + ((1 << log2TrafoHeight) >> 1)	
x3 = x1	
y3 = y2	
} else {	
x1 = x0 + ((1 << log2TrafoWidth) >> 2) * InterTUSplitDirection	
y1 = y0 + ((1 << log2TrafoHeight) >> 2) * (1 − InterTUSplitDirection)	
x2 = x1 + ((1 << log2TrafoWidth) >> 2) * InterTUSplitDirection	
y2 = y1 + ((1 << log2TrafoHeight) >> 2) * (1 − InterTUSplitDirection)	
x3 = x2 + ((1 << log2TrafoWidth) >> 2) * InterTUSplitDirection	
y3 = y2 + ((1 << log2TrafoHeight) >> 2) * (1 − InterTUSplitDirection)	
log2TrafoHeight = log2TrafoHeight + 2 * InterTUSplitDirection − 1	
log2TrafoWidth = log2TrafoWidth − 2 * InterTUSplitDirection + 1	
}	
transform_tree(x0, y0, log2TrafoWidth − 1, log2TrafoHeight − 1, trafoDepth + 1, 0)	
transform_tree(x1, y1, log2TrafoWidth − 1, log2TrafoHeight − 1, trafoDepth + 1, 1)	
transform_tree(x2, y2, log2TrafoWidth − 1, log2TrafoHeight − 1, trafoDepth + 1, 2)	
transform_tree(x3, y3, log2TrafoWidth − 1, log2TrafoHeight − 1, trafoDepth + 1, 3)	
} else if(entropy_coding_mode_flag) {	
if(PredMode = = MODE_INTRA || trafoDepth != 0 || cbf_cb[x0][y0][trafoDepth] || cbf_cr[x0][y0][trafoDepth]) {	
readCbf = true	
if(blkIdx = = 3 && log2TrafoSize < Log2MaxTrafoSize && PredMode != MODE_INTRA)	
readCbf = cbf_luma[xBase][yBase][trafoDepth] || cbf_luma[xBase + (1 << log2TrafoWidth)][yBase][trafoDepth] || cbf_luma[xBase][yBase + (1 << log2TrafoHeight)][trafoDepth] || cbf_cb[xBase][yBase][trafoDepth – 1] || cbf_cr[xBase][yBase][trafoDepth – 1]	
if (!readCbf)	
cbf_luma[x0][y0][trafoDepth] = 1	
else	
cbf_luma[x0][y0][trafoDepth]	ae(v)
}	
if(PredMode = = MODE_INTRA)	

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000561

if(log2TrafoSize > Log2MinTrafoSize) {	
cbf_cb[x0][y0][trafoDepth]	ae(v)
cbf_cr[x0][y0][trafoDepth]	ae(v)
} else if(blkIdx == 0) {	
cbf_cb[x0][y0][trafoDepth − 1]	ae(v)
cbf_cr[x0][y0][trafoDepth − 1]	ae(v)
}	
}	
}	
}	

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000562

7.3.9 Transform coefficient syntax

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000563

transform_coeff(x0, y0, log2TrafoWidth, log2TrafoHeight, trafoDepth, cIdx) {	Descriptor
if((cIdx = = 0 && cbf_luma[x0][y0][trafoDepth]) || (cIdx = = 1 && cbf_cb[x0][y0][trafoDepth]) || (cIdx = = 2 && cbf_cr[x0][y0][trafoDepth]))) {	
if(cu_qp_delta_enabled_flag && !IsCuQpDeltaCoded) {	
cu_qp_delta	se(v) | ae(v)
IsCuQpDeltaCoded = 1	
}	
if(split_transform_flag[x0][y0][trafoDepth] && (cIdx = = 0 || log2TrafoSize > Log2MinTrafoSize + 1)) {	
if(InterTUSplitDirection = = 2) {	
x1 = x0 + ((1 << log2TrafoWidth) >> 1)	
y1 = y0	
x2 = x0	
y2 = y0 + ((1 << log2TrafoHeight) >> 1)	
x3 = x1	
y3 = y2	
} else {	
x1 = x0 + ((1 << log2TrafoWidth) >> 2) * InterTUSplitDirection	
y1 = y0 + ((1 << log2TrafoHeight) >> 2) * (1 − InterTUSplitDirection)	
x2 = x1 + ((1 << log2TrafoWidth) >> 2) * InterTUSplitDirection	
y2 = y1 + ((1 << log2TrafoHeight) >> 2) * (1 − InterTUSplitDirection)	
x3 = x2 + ((1 << log2TrafoWidth) >> 2) * InterTUSplitDirection	
y3 = y2 + ((1 << log2TrafoHeight) >> 2) * (1 − InterTUSplitDirection)	
log2TrafoHeight = log2TrafoHeight + 2 * InterTUSplitDirection − 1	
log2TrafoWidth = log2TrafoWidth − 2 * InterTUSplitDirection + 1	
}	
transform_coeff(x0, y0, log2TrafoWidth − 1, log2TrafoHeight − 1, trafoDepth + 1, cIdx)	
transform_coeff(x1, y1, log2TrafoWidth − 1, log2TrafoHeight − 1, trafoDepth + 1, cIdx)	
transform_coeff(x2, y2, log2TrafoWidth − 1, log2TrafoHeight − 1, trafoDepth + 1, cIdx)	
transform_coeff(x3, y3, log2TrafoWidth − 1, log2TrafoHeight − 1, trafoDepth + 1, cIdx)	
} else {	
log2TrafoSize = ((log2TrafoWidth + log2TrafoHeight) >> 1) − (cIdx > 0 ? 1 : 2)	
if (PredMode = = MODE_INTRA) {	
if (cIdx = = 0)	
scanIdx = ScanType[log2TrafoSize – 2][IntraPredMode]	
else	
scanIdx = ScanType[log2TrafoSize – 2][IntraPredModeC]	
} else {	
scanIdx = 0	
}	
if (entropy_coding_mode_flag)	
if (scanIdx = = 0)	
scanIdx = 3	
residual_coding_cabac(x0, y0, log2TrafoSize, trafoDepth, scanIdx, cIdx)	
else if (!entropy_coding_mode_flag)	
residual_coding_cavlc(x0, y0, log2TrafoSize, trafoDepth, cIdx)	
}	
}	

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000564

}	

[Ed. (BB): the condition (cIdx == 0 || log2TrafoSize > Log2MinTrafoSize + 1) assumes 4:2:0 color sampling and assures that the luma log2TrafoSize is greater than Log2MinTrafoSize for chroma.]

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000565

7.3.10 Residual coding CABAC syntax

residual_coding_cabac(x0, y0, log2TrafoSize, trafoDepth, scanIdx, cIdx) {	Descriptor
last_significant_coeff_x	ae(v)
last_significant_coeff_y	ae(v)
numCoeff = 0	
xC = ScanOrder[log2TrafoSize − 2][log2TrafoSize − 2][scanIdx][numCoeff][0]	
yC = ScanOrder[log2TrafoSize − 2][log2TrafoSize − 2][scanIdx][numCoeff][1]	
while((xC != last_significant_coeff_x) || (yC != last_significant_coeff_y)) {	
numCoeff++	
xC = ScanOrder[log2TrafoSize − 2][log2TrafoSize − 2][scanIdx][numCoeff][0]	
yC = ScanOrder[log2TrafoSize − 2][log2TrafoSize − 2][scanIdx][numCoeff][1]	
}	
for(n = numCoeff − 1; n >= 0; n− −) {	
xC = ScanOrder[log2TrafoSize − 2][log2TrafoSize − 2][scanIdx][n][0]	
yC = ScanOrder[log2TrafoSize − 2][log2TrafoSize − 2][scanIdx][n][1]	
significant_coeff_flag[xC][yC]	ae(v)
}	
numLastSubset = numCoeff >> 4	
for(i = numLastSubset − 1; i >= 0; i− −) {	
offset = i << 4	
for(n = 15; n >= 0; n− −) {	
xC = ScanOrder[log2TrafoSize − 2][log2TrafoSize − 2][scanIdx0][n + offset][0]	
yC = ScanOrder[log2TrafoSize − 2][log2TrafoSize − 2][scanIdx0][n + offset`][1]	
if(significant_coeff_flag[xC][yC])	
coeff_abs_level_greater1_flag[n]	ae(v)
}	
for(n = 15; n >= 0; n− −) {	
if(coeff_abs_level_greater1_flag[n])	
coeff_abs_level_greater2_flag[n]	ae(v)
}	
for(n = 15; n >= 0; n− −) {	
xC = ScanOrder[log2TrafoSize − 2][log2TrafoSize − 2][scanIdx0][n + offset][0]	
yC = ScanOrder[log2TrafoSize − 2][log2TrafoSize − 2][scanIdx0][n + offset`][1]	
if(significant_coeff_flag[xC][yC]) {	
coeff_sign_flag[n]	ae(v)
}	
for(n = 15; n >= 0; n− −) {	
if(coeff_abs_level_greater2_flag[n])	
coeff_abs_level_minus3[n]	ae(v)
xC = ScanOrder[log2TrafoSize − 2][log2TrafoSize − 2][scanIdx0][n + offset][0]	
yC = ScanOrder[log2TrafoSize − 2][log2TrafoSize − 2][scanIdx0][n + offset`][1]	
if(significant_coeff_flag[xC][yC]) {	
transCoeffLevel[x0][y0][trafoDepth][cIdx][n + offset] = (coeff_abs_level_minus3[n] + 3) * (1 − 2 * coeff_sign_flag[n])	
} else	
transCoeffLevel[x0][y0][trafoDepth][cIdx][n + offset] = 0	
}	
}	
}	

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000566

7.3.11 Residual coding CAVLC syntax

residual_coding_cavlc(x0, y0, log2TrafoSize, trafoDepth, cIdx) {	Descriptor
n = 1 << (log2TrafoSize << 1)	
last_pos_level_one	ce(v)
lastPos = last_pos_level_one % n	
levelGreaterThanOneFlag = (last_pos_level_one > n)	
level_and_sign(levelGreaterThanOneFlag, transCoeffLevel, x0, y0, trafoDepth, cIdx, lastPos)	
runModeFlag = TRUE	
n = lastPos – 1	
sumBigCoeff = 0	
switchThres = (PredMode == MODE_INTRA && cIdx == 0)? 0 : 49	
while (runModeFlag && n >= 0) {	
run_level_one	ce(v)
runOfZeros = run_level_one % n	
levelGreaterThanOneFlag = (run_level_one > n)	
trOne = (trOne = = 0 | | levelGreaterThanOneFlag) ? 0 : Max(4, trOne + 1)	
n = n – runOfZeros	
if(n >= 0) {	
level_and_sign(levelGreaterThanOneFlag, transCoeffLevel, x0, y0, trafoDepth, cIdx, n)	
if (levelGreaterThanOneFlag) {	
sumBigCoeff = sumBigCoeff + level	
if (log2TrafoSize==2 | | n > switchThres | | sumBigCoeff > 2)	
runModeFlag = FALSE	
}	
}	
n--	
}	
while (n >= 0) {	
level	ce(v)
if (level > 0) {	
sign_flag	u(1)
sign = 1 – 2 * sign_flag	
transCoeffLevel[x0][y0][trafoDepth][cIdx][n] = level * sign	
}	
n--	
}	
}	

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000567

7.3.12 Level and sign CAVLC syntax

level_and_sign(levelGreaterThanOneFlag, transCoeffLevel, x0, y0, trafoDepth, cIdx, n) {	Descriptor
if (levelGreaterThanOneFlag) {	
level_minus2_and_sign	ce(v)
level = ((level_minus2_and_sign >> 1) + 2	
sign = 1 - 2*(level_minus2_and_sign & 1)	
transCoeffLevel[x0][y0][trafoDepth][cIdx][n] = level * sign	
}	
else {	
sign_flag	u(1)
transCoeffLevel[x0][y0][trafoDepth][cIdx][n] = 1 – 2 *(sign_flag)	
}	
}	

7.4 Semantics

Semantics associated with the syntax structures and with the syntax elements within these structures are specified in this subclause. When the semantics of a syntax element are specified using a table or a set of tables, any values that are not specified in the table(s) shall not be present in the bitstream unless otherwise specified in this Recommendation | International Standard.

7.4.1 NAL unit semantics

NOTE 1 – The VCL is specified to efficiently represent the content of the video data. The NAL is specified to format that data and provide header information in a manner appropriate for conveyance on a variety of communication channels or storage media. All data are contained in NAL units, each of which contains an integer number of bytes. A NAL unit specifies a generic format for use in both packet-oriented and bitstream systems. The format of NAL units for both packet-oriented transport and byte stream is identical except that each NAL unit can be preceded by a start code prefix and extra padding bytes in the byte stream format.

NumBytesInNALunit specifies the size of the NAL unit in bytes. This value is required for decoding of the NAL unit. Some form of demarcation of NAL unit boundaries is necessary to enable inference of NumBytesInNALunit. One such demarcation method is specified in Annex B for the byte stream format. Other methods of demarcation may be specified outside of this Recommendation | International Standard.

forbidden_zero_bit shall be equal to 0.

nal_ref_flag equal to 1 specifies that the content of the NAL unit contains a sequence parameter set, a picture parameter set or a slice of a reference picture.

For coded video sequences conforming to one or more of the profiles specified in Annex A that are decoded using the decoding process specified in clauses 2-9, nal_ref_flag equal to 0 for a NAL unit containing a slice indicates that the slice is part of a non-reference picture.

nal_ref_flag shall not be equal to 0 for sequence parameter set or picture parameter set NAL units. When nal_ref_flag is equal to 0 for one NAL unit with nal_unit_type equal to 1 or 4 of a particular picture, it shall be equal to 0 for all NAL units with nal_unit_type equal to 1 or 4 of the picture.

nal_ref_flag shall not be equal to 0 for NAL units with nal_unit_type equal to 5.

nal_ref_flag shall be equal to 0 for all NAL units having nal_unit_type equal to 6, 9, 10, 11, or 12.

nal_unit_type specifies the type of RBSP data structure contained in the NAL unit as specified in Table 7-1.

NAL units that use nal_unit_type equal to 0 or in the range of 24..63, inclusive, shall not affect the decoding process specified in this Recommendation | International Standard.

NOTE 2 – NAL unit types 0 and 24..63 may be used as determined by the application. No decoding process for these values of nal_unit_type is specified in this Recommendation | International Standard. Since different applications might use NAL unit types 0 and 24..63 for different purposes, particular care must be exercised in the design of encoders that generate NAL units with nal_unit_type equal to 0 or in the range of 24 to 63, inclusive, and in the design of decoders that interpret the content of NAL units with nal_unit_type equal to 0 or in the range of 24 to 63, inclusive.

Decoders shall ignore (remove from the bitstream and discard) the contents of all NAL units that use reserved values of nal_unit_type.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000568

NOTE 3 – This requirement allows future definition of compatible extensions to this Recommendation | International Standard.

Table 7-1 – NAL unit type codes and NAL unit type classes

nal_unit_type	Content of NAL unit and RBSP syntax structure	NAL unit type class
0	Unspecified	non-VCL
1	Coded slice of a non-IDR and non-CRA picture slice_layer_rbsp( )	VCL
2-3	Reserved	n/a
4	Coded slice of a CRA picture slice_layer_rbsp( )	VCL
5	Coded slice of an IDR picture slice_layer_rbsp( )	VCL
6	Supplemental enhancement information (SEI) sei_rbsp( )	non-VCL
7	Sequence parameter set seq_parameter_set_rbsp( )	non-VCL
8	Picture parameter set pic_parameter_set_rbsp( )	non-VCL
9	Access unit delimiter access_unit_delimiter_rbsp( )	non-VCL
10-11	Reserved	n/a
12	Filler data filler_data_rbsp( )	non-VCL
13-23	Reserved	n/a
24..63	Unspecified	non-VCL

Formatted: Centered

In the text, coded slice NAL unit collectively refers to a coded slice of a non-IDR picture NAL unit or to a coded slice of an IDR picture NAL unit. The variable IdrPicFlag is specified as

$$IdrPicFlag = ((nal\_unit\_type == 5) ? 1 : 0) \quad (7\text{-}1)$$

When the value of nal_unit_type is equal to 5 for a NAL unit containing a slice of a particular picture, the picture shall not contain NAL units with nal_unit_type equal to 1 or 4. For coded video sequences conforming to one or more of the profiles specified in Annex A that are decoded using the decoding process specified in clauses 2-9, such a picture is referred to as an IDR picture.

temporal_id specifies a temporal identifier for the NAL unit. The value of temporal_id shall be the same for all NAL units of an access unit. When an access unit contains any NAL unit with nal_unit_type equal to 5, temporal_id shall be equal to 0.

output_flag affects the decoded picture output and removal processes as specified in Annex C.

[Ed. KS: There is no output definition on Annex C yet]

reserved_one_4bits shall be equal to 1. Other values of reserved_one_4bits may be specified in the future by ITU-T | ISO/IEC. Decoders shall ignore the value of reserved_one_4bits.

rbsp_byte[i] is the i-th byte of an RBSP. An RBSP is specified as an ordered sequence of bytes as follows.

The RBSP contains an SODB as follows.

– If the SODB is empty (i.e., zero bits in length), the RBSP is also empty.

– Otherwise, the RBSP contains the SODB as follows:

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000569

1) The first byte of the RBSP contains the (most significant, left-most) eight bits of the SODB; the next byte of the RBSP shall contain the next eight bits of the SODB, etc., until fewer than eight bits of the SODB remain.
2) rbsp_trailing_bits( ) are present after the SODB as follows:
   i) The first (most significant, left-most) bits of the final RBSP byte contains the remaining bits of the SODB (if any).
   ii) The next bit consists of a single rbsp_stop_one_bit equal to 1.
   iii) When the rbsp_stop_one_bit is not the last bit of a byte-aligned byte, one or more rbsp_alignment_zero_bit is present to result in byte alignment.
3) One or more cabac_zero_word 16-bit syntax elements equal to 0x0000 may be present in some RBSPs after the rbsp_trailing_bits( ) at the end of the RBSP.

Syntax structures having these RBSP properties are denoted in the syntax tables using an "_rbsp" suffix. These structures shall be carried within NAL units as the content of the rbsp_byte[i] data bytes. The association of the RBSP syntax structures to the NAL units shall be as specified in Table 7-1.

NOTE 6 – When the boundaries of the RBSP are known, the decoder can extract the SODB from the RBSP by concatenating the bits of the bytes of the RBSP and discarding the rbsp_stop_one_bit, which is the last (least significant, right-most) bit equal to 1, and discarding any following (less significant, farther to the right) bits that follow it, which are equal to 0. The data necessary for the decoding process is contained in the SODB part of the RBSP.

emulation_prevention_three_byte is a byte equal to 0x03. When an emulation_prevention_three_byte is present in the NAL unit, it shall be discarded by the decoding process.

The last byte of the NAL unit shall not be equal to 0x00.

Within the NAL unit, the following three-byte sequences shall not occur at any byte-aligned position:

– 0x000000
– 0x000001
– 0x000002

Within the NAL unit, any four-byte sequence that starts with 0x000003 other than the following sequences shall not occur at any byte-aligned position:

– 0x00000300
– 0x00000301
– 0x00000302
– 0x00000303

NOTE 7 – When nal_unit_type is equal to 0, particular care must be exercised in the design of encoders to avoid the presence of the above-listed three-byte and four-byte patterns at the beginning of the NAL unit syntax structure, as the syntax element emulation_prevention_three_byte cannot be the third byte of a NAL unit.

7.4.1.1 Encapsulation of an SODB within an RBSP (informative)

This subclause does not form an integral part of this Recommendation | International Standard.

The form of encapsulation of an SODB within an RBSP and the use of the emulation_prevention_three_byte for encapsulation of an RBSP within a NAL unit is specified for the following purposes:

– to prevent the emulation of start codes within NAL units while allowing any arbitrary SODB to be represented within a NAL unit,
– to enable identification of the end of the SODB within the NAL unit by searching the RBSP for the rbsp_stop_one_bit starting at the end of the RBSP,
– to enable a NAL unit to have a size larger than that of the SODB under some circumstances (using one or more cabac_zero_word).

The encoder can produce a NAL unit from an RBSP by the following procedure:

1. The RBSP data is searched for byte-aligned bits of the following binary patterns:

   '00000000 00000000 000000xx' (where xx represents any 2 bit pattern: 00, 01, 10, or 11),

   and a byte equal to 0x03 is inserted to replace these bit patterns with the patterns:

   '00000000 00000000 00000011 000000xx',

   and finally, when the last byte of the RBSP data is equal to 0x00 (which can only occur when the RBSP ends in a cabac_zero_word), a final byte equal to 0x03 is appended to the end of the data. The last zero byte of a

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000570

byte-aligned three-byte sequence 0x000000 in the RBSP (which is replaced by the four-byte sequence 0x00000300) is taken into account when searching the RBSP data for the next occurrence of byte-aligned bits with the binary patterns specified above.

2. The resulting sequence of bytes is then prefixed as follows.
   - If nal_unit_type is not equal to 14 or 20, the sequence of bytes is prefixed with the first byte of the NAL unit containing the syntax elements forbidden_zero_bit, ~~nal_ref_idcnal_ref_flag~~, and nal_unit_type, where nal_unit_type indicates the type of RBSP data structure the NAL unit contains.
   - Otherwise (nal_unit_type is equal to 14 or 20), the sequence of bytes is prefixed with the first four bytes of the NAL unit, where the first byte contains the syntax elements forbidden_zero_bit, ~~nal_ref_idcnal_ref_flag~~, and nal_unit_type and the following three bytes contain the syntax structure nal_unit_header_svc_extension( ). The syntax element nal_unit_type in the first byte indicates the presence of the syntax structure nal_unit_header_svc_extension( ) in the following three bytes and the type of RBSP data structure the NAL unit contains.

The process specified above results in the construction of the entire NAL unit.

This process can allow any SODB to be represented in a NAL unit while ensuring that

- no byte-aligned start code prefix is emulated within the NAL unit,
- no sequence of 8 zero-valued bits followed by a start code prefix, regardless of byte-alignment, is emulated within the NAL unit.

7.4.1.2 Order of NAL units and association to coded pictures, access units, and video sequences

This subclause specifies constraints on the order of NAL units in the bitstream.

Any order of NAL units in the bitstream obeying these constraints is referred to in the text as the decoding order of NAL units. Within a NAL unit, the syntax in subclauses 7.3, D.1, and E.1 specifies the decoding order of syntax elements. Decoders shall be capable of receiving NAL units and their syntax elements in decoding order.

7.4.1.2.1 Order of sequence, ~~and~~ picture and adaptation parameter set RBSPs and their activation

This subclause specifies the activation process of picture, ~~and~~ sequence and adaptation parameter sets for coded video sequences that conform to one or more of the profiles specified in Annex A that are decoded using the decoding process specified in clauses 2-9.

NOTE 1 – The sequence, ~~and~~ picture and adaptation parameter set mechanism decouples the transmission of infrequently changing information from the transmission of coded macroblock data. Sequence, ~~and~~ picture and adaptation parameter sets may, in some applications, be conveyed "out-of-band" ~~using a reliable transport mechanism~~.

An adaptation parameter set RBSP includes parameters that can be referred to by the coded slice NAL units of one or more coded pictures when at least one of sample_adaptive_offset_enabled_flag or adaptive_loop_filter_enabled_flag are equal to 1. Each adaptation parameter set RBSP is initially considered not active at the start of the operation of the decoding process. At most one adaptation parameter set RBSP is considered active at any given moment during the operation of the decoding process, and the activation of any particular adaptation parameter set RBSP results in the deactivation of the previously-active adaptation parameter set RBSP (if any).

When an adaptation parameter set RBSP (with a particular value of aps_id) is not active and it is referred to by a coded slice NAL unit (using that value of aps_id), it is activated. This adaptation parameter set RBSP is called the active adaptation parameter set RBSP until it is deactivated by the activation of another adaptation parameter set RBSP. An adaptation parameter set RBSP, with that particular value of aps_id, shall be available to the decoding process prior to its activation.

A picture parameter set RBSP includes parameters that can be referred to by the coded slice NAL units or coded slice data partition A NAL units of one or more coded pictures. Each picture parameter set RBSP is initially considered not active at the start of the operation of the decoding process. At most one picture parameter set RBSP is considered active at any given moment during the operation of the decoding process, and the activation of any particular picture parameter set RBSP results in the deactivation of the previously-active picture parameter set RBSP (if any).

When a picture parameter set RBSP (with a particular value of pic_parameter_set_id) is not active and it is referred to by a coded slice NAL unit or coded slice data partition A NAL unit (using that value of pic_parameter_set_id), it is activated. This picture parameter set RBSP is called the active picture parameter set RBSP until it is deactivated by the activation of another picture parameter set RBSP. A picture parameter set RBSP, with that particular value of pic_parameter_set_id, shall be available to the decoding process prior to its activation.

Any picture parameter set NAL unit containing the value of pic_parameter_set_id for the active picture parameter set RBSP for a coded picture shall have the same content as that of the active picture parameter set RBSP for the coded picture unless it follows the last VCL NAL unit of the coded picture and precedes the first VCL NAL unit of another coded picture.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000571

A sequence parameter set RBSP includes parameters that can be referred to by one or more picture parameter set RBSPs or one or more SEI NAL units containing a buffering period SEI message. Each sequence parameter set RBSP is initially considered not active at the start of the operation of the decoding process. At most one sequence parameter set RBSP is considered active at any given moment during the operation of the decoding process, and the activation of any particular sequence parameter set RBSP results in the deactivation of the previously-active sequence parameter set RBSP (if any).

When a sequence parameter set RBSP (with a particular value of seq_parameter_set_id) is not already active and it is referred to by activation of a picture parameter set RBSP (using that value of seq_parameter_set_id) or is referred to by an SEI NAL unit containing a buffering period SEI message (using that value of seq_parameter_set_id), it is activated. This sequence parameter set RBSP is called the active sequence parameter set RBSP until it is deactivated by the activation of another sequence parameter set RBSP. A sequence parameter set RBSP, with that particular value of seq_parameter_set_id, shall be available to the decoding process prior to its activation. An activated sequence parameter set RBSP shall remain active for the entire coded video sequence.

NOTE 2 – Because an IDR access unit begins a new coded video sequence and an activated sequence parameter set RBSP must remain active for the entire coded video sequence, a sequence parameter set RBSP can only be activated by a buffering period SEI message when the buffering period SEI message is part of an IDR access unit.

Any sequence parameter set NAL unit containing the value of seq_parameter_set_id for the active sequence parameter set RBSP for a coded video sequence shall have the same content as that of the active sequence parameter set RBSP for the coded video sequence unless it follows the last access unit of the coded video sequence and precedes the first VCL NAL unit and the first SEI NAL unit containing a buffering period SEI message (when present) of another coded video sequence.

NOTE 3 – If picture parameter set RBSP, ~~or~~ sequence parameter set RBSP or adaptation parameter set RBSP are conveyed within the bitstream, these constraints impose an order constraint on the NAL units that contain the picture parameter set RBSP or sequence parameter set RBSP, respectively. Otherwise (picture parameter set RBSP, ~~or~~ sequence parameter set RBSP or adaptation parameter set RBSP are conveyed by other means not specified in this Recommendation | International Standard), they must be available to the decoding process in a timely fashion such that these constraints are obeyed.

~~When present, a sequence parameter set extension RBSP includes parameters having a similar function to those of a sequence parameter set RBSP. For purposes of establishing constraints on the syntax elements of the sequence parameter set extension RBSP and for purposes of determining activation of a sequence parameter set extension RBSP, the sequence parameter set extension RBSP shall be considered part of the preceding sequence parameter set RBSP with the same value of seq_parameter_set_id. When a sequence parameter set RBSP is present that is not followed by a sequence parameter set extension RBSP with the same value of seq_parameter_set_id prior to the activation of the sequence parameter set RBSP, the sequence parameter set extension RBSP and its syntax elements shall be considered not present for the active sequence parameter set RBSP.~~

All constraints that are expressed on the relationship between the values of the syntax elements (and the values of variables derived from those syntax elements) in sequence parameter sets, ~~and~~ picture parameter sets and adaptation parameter sets and other syntax elements are expressions of constraints that apply only to the active sequence parameter set, ~~and~~ the active picture parameter set and the active adaptation parameter set. If any sequence parameter set RBSP is present that is not activated in the bitstream, its syntax elements shall have values that would conform to the specified constraints if it were activated by reference in an otherwise-conforming bitstream. If any picture parameter set RBSP is present that is not ever activated in the bitstream, its syntax elements shall have values that would conform to the specified constraints if it were activated by reference in an otherwise-conforming bitstream. If any adaptation parameter set RBSP is present that is not ever activated in the bitstream, its syntax elements shall have values that would conform to the specified constraints if it were activated by reference in an otherwise-conforming bitstream.

During operation of the decoding process (see clause 0), the values of parameters of the active picture parameter set and the active sequence parameter set shall be considered in effect. For interpretation of SEI messages, the values of the parameters of the picture parameter set and sequence parameter set that are active for the operation of the decoding process for the VCL NAL units of the primary coded picture in the same access unit shall be considered in effect unless otherwise specified in the SEI message semantics.

7.4.1.2.2 Order of access units and association to coded video sequences

A bitstream conforming to this Recommendation | International Standard consists of one or more coded video sequences.

A coded video sequence consists of one or more access units. For coded video sequences that conform to one or more of the profiles specified in Annex A that are decoded using the decoding process specified in clauses 2-9, the order of NAL units and coded pictures and their association to access units is described in subclause 7.4.1.2.3.

The first access unit of each coded video sequence is an IDR access unit. All subsequent access units in the coded video sequence are non-IDR access units.

The values of picture order count for the coded pictures in consecutive access units in decoding order containing non-reference pictures shall be non-decreasing.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000572

When present, an access unit following an access unit that contains an end of sequence NAL unit shall be an IDR access unit.

When an SEI NAL unit contains data that pertain to more than one access unit (for example, when the SEI NAL unit has a coded video sequence as its scope), it shall be contained in the first access unit to which it applies.

When an end of stream NAL unit is present in an access unit, this access unit shall be the last access unit in the bitstream and the end of stream NAL unit shall be the last NAL unit in that access unit.

7.4.1.2.3 Order of NAL units and coded pictures and association to access units

This subclause specifies the order of NAL units and coded pictures and association to access unit for coded video sequences that conform to one or more of the profiles specified in Annex A that are decoded using the decoding process specified in clauses 2-9.

An access unit consists of one primary coded picture, zero or more corresponding redundant coded pictures, and zero or more non-VCL NAL units. The association of VCL NAL units to primary or redundant coded pictures is described in subclause 7.4.1.2.5.

The first access unit in the bitstream starts with the first NAL unit of the bitstream.

The first of any of the following NAL units after the last VCL NAL unit of a primary coded picture specifies the start of a new access unit:

- access unit delimiter NAL unit (when present),
- sequence parameter set NAL unit (when present),
- picture parameter set NAL unit (when present),
- SEI NAL unit (when present),
- NAL units with nal_unit_type in the range of 14 to 18, inclusive (when present),
- first VCL NAL unit of a primary coded picture (always present).

The constraints for the detection of the first VCL NAL unit of a primary coded picture are specified in subclause 7.4.1.2.4.

The following constraints shall be obeyed by the order of the coded pictures and non-VCL NAL units within an access unit:

- When an access unit delimiter NAL unit is present, it shall be the first NAL unit. There shall be at most one access unit delimiter NAL unit in any access unit.
- When any SEI NAL units are present, they shall precede the primary coded picture.
- When an SEI NAL unit containing a buffering period SEI message is present, the buffering period SEI message shall be the first SEI message payload of the first SEI NAL unit in the access unit.
- The primary coded picture shall precede the corresponding redundant coded pictures.
- NAL units having nal_unit_type equal to 0, 12, or in the range of 20 to 31, inclusive, shall not precede the first VCL NAL unit of the primary coded picture.

  NOTE 1 – Sequence parameter set NAL units or picture parameter set NAL units may be present in an access unit, but cannot follow the last VCL NAL unit of the primary coded picture within the access unit, as this condition would specify the start of a new access unit.

The structure of access units not containing any NAL units with nal_unit_type equal to 0, 7, 8, or in the range of 12 to 18, inclusive, or in the range of 20 to 31, inclusive, is shown in Figure 7-1. [Ed. (TW):adjust text and figure]

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000573

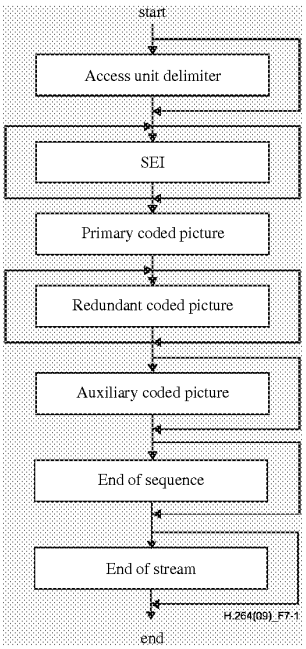


Figure 7-1 – Structure of an access unit not containing any NAL units with nal_unit_type equal to 0, 7, 8, or in the range of 12 to 18, inclusive, or in the range of 20 to 31, inclusive

7.4.1.2.4 Detection of the first VCL NAL unit of a primary coded picture

This subclause specifies constraints on VCL NAL unit syntax that are sufficient to enable the detection of the first VCL NAL unit of each primary coded picture for coded video sequences that conform to one or more of the profiles specified in Annex A that are decoded using the decoding process specified in clauses 2-9.

Any coded slice NAL unit or coded slice data partition A NAL unit of the primary coded picture of the current access unit shall be different from any coded slice NAL unit or coded slice data partition A NAL unit of the primary coded picture of the previous access unit in one or more of the following ways:

– frame_num differs in value. The value of frame_num used to test this condition is the value of frame_num that appears in the syntax of the slice header, regardless of whether that value is inferred to have been equal to 0 for subsequent use in the decoding process due to the presence of memory_management_control_operation equal to 5.

  NOTE 1 – A consequence of the above statement is that a primary coded picture having frame_num equal to 1 cannot contain a memory_management_control_operation equal to 5 unless some other condition listed below is fulfilled for the next primary coded picture that follows after it (if any).

– pic_parameter_set_id differs in value.

– nal_ref_idc differs in value with one of the nal_ref_idc values being equal to 0.

– pic_order_cnt_type is equal to 0 for both and either pic_order_cnt_lsb differs in value, or delta_pic_order_cnt_bottom differs in value.

– pic_order_cnt_type is equal to 1 for both and either delta_pic_order_cnt[0] differs in value, or delta_pic_order_cnt[1] differs in value.

– IdrPicFlag differs in value.

– IdrPicFlag is equal to 1 for both and idr_pic_id differs in value.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000574

7.4.1.2.5 Order of VCL NAL units and association to coded pictures

[Ed. (TW): insert text]

7.4.2 Raw byte sequence payloads and RBSP trailing bits semantics

7.4.2.1 Sequence parameter set RBSP semantics

profile_idc and **level_idc** indicate the profile and level to which the coded video sequence conforms.

reserved_zero_8bits shall be equal to 0. Decoders shall ignore the value of reserved_zero_8bits.

seq_parameter_set_id identifies the sequence parameter set that is referred to by the picture parameter set. The value of seq_parameter_set_id shall be in the range of 0 to 31, inclusive.

max_temporal_layers_minus1 + 1 specifies the maximum number of temporal layers present in the sequence. The value of max_temporal_layers_minus1 shall be in the range of 0 to 7, inclusive.

pic_width_in_luma_samples specifies the width of each decoded picture in luma samples. The value of pic_width_in_luma_samples shall be in the range of 0 to 2^{16}-1, inclusive.

pic_height_in_luma_samples specifies the height of each decoded picture in luma samples. The value of pic_height_in_luma_samples shall be in the range of 0 to 2^{16}-1, inclusive.

bit_depth_luma_minus8 + 8 specifies the bit depth of the samples of the luma array and the value of the luma quantisation parameter range offset QpBdOffset$_Y$, as specified by

$$\text{BitDepth}_Y = 8 + \text{bit\_depth\_luma\_minus8} \quad (7\text{-}2)$$
$$\text{QpBdOffset}_Y = 6 * \text{bit\_depth\_luma\_minus8} \quad (7\text{-}3)$$

bit_depth_luma_minus8 shall be in the range of 0 to 6, inclusive.

bit_depth_chroma_minus8 + 8 specifies the bit depth of the samples of the chroma arrays and the value of the chroma quantisation parameter range offset QpBdOffset$_C$, as specified by

$$\text{BitDepth}_C = 8 + \text{bit\_depth\_chroma\_minus8} \quad (7\text{-}4)$$
$$\text{QpBdOffset}_C = 6 * \text{bit\_depth\_chroma\_minus8} \quad (7\text{-}5)$$

bit_depth_chroma_minus8 shall be in the range of 0 to 6, inclusive.

pcm_sample_bit_depth_luma_minus1 + 1 specifies the number of bits used to represent each of PCM sample values of luma component. The value of pcm_sample_bit_depth_luma_minus1 + 1 shall be smaller than or equal to the value of BitDepth$_Y$.

$$\text{PCMBitDepth}_Y = 1 + \text{pcm\_sample\_bit\_depth\_luma\_minus1} \quad (7\text{-}4)$$

pcm_sample_bit_depth_chroma_minus1 + 1 specifies the number of bits used to represent each of PCM sample values of chroma components. The value of pcm_sample_bit_depth_chroma_minus1 + 1 shall be smaller than or equal to the value of BitDepth$_C$.

$$\text{PCMBitDepth}_C = 1 + \text{pcm\_sample\_bit\_depth\_chroma\_minus1} \quad (7\text{-}4)$$

log2_max_frame_num_minus4 specifies the value of the variable MaxFrameNum that is used in frame_num related derivations as follows:

$$\text{MaxFrameNum} = 2^{(\text{log2\_max\_frame\_num\_minus4} + 4)} \quad (7\text{-}6)$$

The value of log2_max_frame_num_minus4 shall be in the range of 0 to 12, inclusive.

pic_order_cnt_type specifies the method to decode picture order count (as specified in subclause XX). The value of pic_order_cnt_type shall be in the range of 0 to 2, inclusive.

pic_order_cnt_type shall not be equal to 2 in a coded video sequence that contains an access unit containing a non-reference frame followed immediately by an access unit containing a non-reference picture,

log2_max_pic_order_cnt_lsb_minus4 specifies the value of the variable MaxPicOrderCntLsb that is used in the decoding process for picture order count as follows:

$$\text{MaxPicOrderCntLsb} = 2^{(\text{log2\_max\_pic\_order\_cnt\_lsb\_minus4} + 4)} \quad (7\text{-}7)$$

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000575

The value of log2_max_pic_order_cnt_lsb_minus4 shall be in the range of 0 to 12, inclusive.

delta_pic_order_always_zero_flag equal to 1 specifies that delta_pic_order_cnt[0] and delta_pic_order_cnt[1] are not present in the slice headers of the sequence and shall be inferred to be equal to 0. delta_pic_order_always_zero_flag equal to 0 specifies that delta_pic_order_cnt[0] is present in the slice headers of the sequence and delta_pic_order_cnt[1] may be present in the slice headers of the sequence.

offset_for_non_ref_pic is used to calculate the picture order count of a non-reference picture as specified in subclause XX. The value of offset_for_non_ref_pic shall be in the range of $-2^{31}+1$ to $2^{31}-1$, inclusive.

num_ref_frames_in_pic_order_cnt_cycle is used in the decoding process for picture order count as specified in subclause XX. The value of num_ref_frames_in_pic_order_cnt_cycle shall be in the range of 0 to 255, inclusive.

offset_for_ref_frame[i] is an element of a list of num_ref_frames_in_pic_order_cnt_cycle values used in the decoding process for picture order count as specified in subclause XX. The value of offset_for_ref_frame[i] shall be in the range of $-2^{31}+1$ to $2^{31}-1$, inclusive.

When pic_order_cnt_type is equal to 1, the variable ExpectedDeltaPerPicOrderCntCycle is derived by

```
ExpectedDeltaPerPicOrderCntCycle = 0
for( i = 0; i < num_ref_frames_in_pic_order_cnt_cycle; i++ )
    ExpectedDeltaPerPicOrderCntCycle += offset_for_ref_frame[ i ]        (7-8)
```

max_num_ref_frames specifies the maximum number of short-term and long-term reference frames, complementary reference field pairs, and non-paired reference fields that may be used by the decoding process for inter prediction of any picture in the sequence. max_num_ref_frames also determines the size of the sliding window operation. The value of max_num_ref_frames shall be in the range of 0 to MaxDpbFrames, inclusive.

gaps_in_frame_num_value_allowed_flag specifies the allowed values of frame_num and the decoding process in case of an inferred gap between values of frame_num.

log2_min_coding_block_size_minus3 specifies the minimum size of a coding block.

The variable Log2MinCUSize is set equal to log2_min_coding_block_size_minus3 + 3.

log2_diff_max_min_coding_block_size specifies the difference between the maximum and minimum coding block size.

The variable Log2MaxCUSize is set equal to log2_min_coding_block_size_minus 3 + 3 + log2_diff_max_min_coding_block_size.

log2_min_transform_block_size_minus2 specifies the minimum size of a transform block.

The variable Log2MinTrafoSize is set equal to log2_min_transform_block_size_minus2 + 2.

log2_diff_max_min_transform_block_size specifies the difference between the maximum and minimum transform size.

The variable Log2MaxTrafoSize is set equal to log2_min_transform_block_size_minus 2 + 2 + log2_diff_max_min_transform_block_size.

The bitstream shall not contain data that result in Log2MaxTrafoSize greater than Log2MaxCUSize.

log2_min_pcm_coding_block_size_minus3 + 3 specifies the minimum size of I_PCM coding blocks.

The variable Log2MinIPCMCUSize is set equal to log2_min_pcm_coding_block_size_minus3 + 3. The variable Log2MinIPCMCUSize should be equal or less than Log2MaxCUSize.

max_transform_hierarchy_depth_intra specifies the maximum hierarchy depth for transform blocks of coding blocks coded in intra prediction mode.

max_transform_hierarchy_depth_inter specifies the maximum hierarchy depth for transform units of coding units coded in inter prediction mode.

chroma_pred_from_luma_enabled_flag equal to 1 specifies the intra chroma prediction process using the reconstructed luma block is applied according to the intra chroma prediction mode.

loop_filter_across_slice_flag equal to 1 specifies the in-loop filtering operations are performed across slice boundary; otherwise, the in-loop operations are slice-independent and not across slice boundary. The in-loop filtering operations include deblocking filter, sample adaptive offset, and adaptive loop filter.

sample_adaptive_offset_enabled_flag equal to 1 specifies the sample adaptive offset process is applied to the reconstruced picture after the deblocking filter process.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000576

adaptive_loop_filter_enabled_flag equal to 1 specifies the adaptive loop filter process is applied to the reconstructed picture after the deblocking filter process.

pcm_loop_filter_disable_flag specifies whether the loop filter process is disabled on reconstructed pixels of I_PCM blocks. If the pcm_loop_filter_disable_flag value is equal to 1, both deblocking and adaptive loop filter processes on the reconstructed pixels of I_PCM blocks are disabled; otherwise the pcm_loop_filter_disable_flag value is equal to 0, both deblocking and adaptive loop filter processes on the reconstructed pixels of I_PCM blocks are not disabled.

[Ed. (WJ): select one expression – enabled_flag or disable_flag]

cu_qp_delta_enabled_flag equal to 1 specifies cu_qp_delta presents in coding unit layer to further modify the value of QP.

temporal_id_nesting_flag specifies whether inter prediction is additionally restricted for the coded video sequence.

Dependent on temporal_id_nesting_flag, the following applies.

– If temporal_id_nesting_flag is equal to 0, additional constraints may not be obeyed.

– Otherwise (temporal_id_nesting_flag is equal to 1), the following applies.

  – For each access unit auA with temporal_id equal to tIdA, an access unit auB with temporal_id equal to tIdB and tIdB less than or equal to tIdA shall not be referenced by inter prediction when there exists an access unit auC with temporal_id equal to tIdC and tIdC less than tIdB, which follows the access unit auB and precedes the access unit auA in decoding order.

NOTE – The syntax element temporal_id_nesting_flag is used to indicate that temporal up-switching, i.e., switching from decoding of up to a specific temporal_id tIdN to decoding up to a temporal_id tIdM > tIdN, is always possible.

inter_4x4_enabled_flag specifies whether inter prediction can be applied to blocks having the size of 4 by 4 luma samples.

7.4.2.2 Picture parameter set RBSP semantics

pic_parameter_set_id identifies the picture parameter set that is referred to in the slice header. The value of pic_parameter_set_id shall be in the range of 0 to 255, inclusive.

seq_parameter_set_id refers to the active sequence parameter set. The value of seq_parameter_set_id shall be in the range of 0 to 31, inclusive.

entropy_coding_mode_flag selects the entropy decoding method to be applied for the syntax elements for which two descriptors appear in the syntax tables as follows.

– If entropy_coding_mode_flag is equal to 0, the method specified by the left descriptor in the syntax table is applied.

– Otherwise (entropy_coding_mode_flag is equal to 1), the method specified by the right descriptor in the syntax table is applied.

num_temporal_layer_switching_point_flags specifies how many temporal switching point flags are present. If temporal_id_nesting_flag is equal to 1, num_temporal_layer_switching_point_flags shall be equal to 0.

temporal_layer_switching_point_flag[i] specifies if the current access point is a temporal switching point allowing the decoding of higher temporal id layers following this access unit. If temporal_id_nesting_flag is equal to 1, temporal_layer_switching_point_flag[i] shall be inferred to be equal to 1. If temporal_id_nesting_flag is equal to 0 and num_temporal_layer_switching_point_flags is less than i, temporal_layer_switching_point_flag[i] shall be inferred to be equal to 0 .

NOTE – When starting to decode a higher temporal layer i, availability of required reference pictures can be guaranteed immediately following an IDR, or a picture with the temporal_id value j lower than i and temporal_switching_flag[j] equal to 1.

num_ref_idx_l0_default_active_minus1 specifies how num_ref_idx_l0_active_minus1 is inferred for P and B slices with num_ref_idx_active_override_flag equal to 0. The value of num_ref_idx_l0_default_active_minus1 shall be in the range of 0 to 31, inclusive.

num_ref_idx_l1_default_active_minus1 specifies how num_ref_idx_l1_active_minus1 is inferred for B slices with num_ref_idx_active_override_flag equal to 0. The value of num_ref_idx_l1_default_active_minus1 shall be in the range of 0 to 31, inclusive.

pic_init_qp_minus26 specifies the initial value minus 26 of $SliceQP_Y$ for each slice. The initial value is modified at the slice layer when a non-zero value of slice_qp_delta is decoded, and is modified further when a non-zero value of cu_qp_delta is decoded at the coding unit layer. The value of pic_init_qp_minus26 shall be in the range of $-(26 + QpBdOffset_Y)$ to +25, inclusive.

constrained_intra_pred_flag equal to 0 specifies that intra prediction allows usage of residual data and decoded samples of neighbouring macroblocks coded using Inter macroblock prediction modes for the prediction of macroblocks coded

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000577

using Intra macroblock prediction modes. constrained_intra_pred_flag equal to 1 specifies constrained intra prediction, in which case prediction of macroblocks coded using Intra macroblock prediction modes only uses residual data and decoded samples from I macroblock types.

slice_granularity indicates the slice granularity within a picture. The value of slice_granularity shall not be larger than Min(Log2MaxCUSize – 4, log2_diff_max_min_coding_block_size). The variable SliceGranularity is set to the value of (slice_granularity << 1).

shared_pps_info_enabled_flag specifies the shared information in picture parameter set RBSP shall be used for the referred slices. If shared_pps_info_enabled_flag is equal to 1, the alf_param() in picture parameter set RBSP shall be applied for the referred slices; otherwise, the alf_param() in slice header(s) shall be applied.

max_cu_qp_delta_depth specifies the maximum allowed depth that is used for specifying QP_Y values for coding unit. The value of max_cu_qp_delta_depth shall be in the range of 0 to 15, inclusive.

[Ed. (BB): Why in the range of 0 to 15, inclusive? is a Log2MaxCUSize of 16 supported?]

The variable log2MinCUDQPSize specifying the minimum coding unit size that can further modifies the value of QP_Y as follows:

log2MinCUDQPSize = Log2MaxCUSize - max_cu_qp_delta_depth (7-7)

7.4.2.3 Adaptive slice parameter set RBSP semantics

aps_id identifies the adaptation parameter set that is referred to in the slice header. The value of aps_id shall be in the range of 0 to TBD, inclusive.

[Ed. (BB): Proponent suggests that this could be made level/profile dependend.]

aps_sample_adaptive_offset_flag equal to 1 specifies that the SAO is on for slices referred to the current APS; equal to 0 specifies that the SAO is off for slices referred to the current APS. If there is no active APS, the aps_sample_adaptive_offset_flag value is inferred to be 0.

aps_adaptive_loop_filter_flag equal to 1 specifies that the ALF is on for slices referred to the current APS, equal to 0 specifies that the ALF is off for slices referred to the current APS. If there is no active APS, the aps_adaptive_loop_filter_flag value is inferred to be 0.

aps_cabac_use_flag equal to 1 specifies that the CABAC decoding process shall be used for sao_param( ) when present, and alf_param( ) when present; equal to 0 specifies that the CAVLC decoding process shall be used for sao_param( ) when present, and alf_param( ) when present.

aps_cabac_init_idc specifies the index for determining the initialisation table used in the initialisation process for context variables of SAO and ALF. The value of cabac_init_idc shall be in the range of 0 to 2, inclusive.

aps_cabac_init_qp_minus26 specifies a quantization parameter minus 26 wherein the quantization parameter is used in the initialization process for context variables of SAO and ALF.

alf_data_byte_count specifies the number of bytes

sao_data_byte_point specifies the number of bytes

~~7.4.2.37.4.2.4~~ **Supplemental enhancement information RBSP semantics**

Supplemental Enhancement Information (SEI) contains information that is not necessary to decode the samples of coded pictures from VCL NAL units.

~~7.4.2.3.17.4.2.4.1~~ **Supplemental enhancement information message semantics**

An SEI RBSP contains one or more SEI messages. Each SEI message consists of the variables specifying the type payloadType and size payloadSize of the SEI payload. SEI payloads are specified in Annex D. The derived SEI payload size payloadSize is specified in bytes and shall be equal to the number of RBSP bytes in the SEI payload.

NOTE – The NAL unit byte sequence containing the SEI message might include one or more emulation prevention bytes (represented by emulation_prevention_three_byte syntax elements). Since the payload size of an SEI message is specified in RBSP bytes, the quantity of emulation prevention bytes is not included in the size payloadSize of an SEI payload.

ff_byte is a byte equal to 0xFF identifying a need for a longer representation of the syntax structure that it is used within.

last_payload_type_byte is the last byte of the payload type of an SEI message.

last_payload_size_byte is the last byte of the payload size of an SEI message.

Formatted: Heading 4,Heading 4 Char1,Heading 4 Char Char, Space Before: 0 pt, After: 0 pt, No bullets or numbering,Tab stops: Not at 0.25" + 0.5" + 0.75" + 1"

Formatted: Highlight

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000578

7.4.2.47.4.2.5 **Access unit delimiter RBSP semantics**

The access unit delimiter may be used to indicate the type of slices present in a primary coded picture and to simplify the detection of the boundary between access units. There is no normative decoding process associated with the access unit delimiter.

primary_pic_type indicates that the slice_type values for all slices of the primary coded picture are members of the set listed in Table 7-2 for the given value of primary_pic_type.

Table 7-2 – Meaning of primary_pic_type

primary_pic_type	slice_type values that may be present in the primary coded picture
0	2
1	0, 2
2	0, 1, 2

7.4.2.47.4.2.6 **Filler data RBSP semantics**

The filler data RBSP contains bytes whose value shall be equal to 0xFF. No normative decoding process is specified for a filler data RBSP.

ff_byte is a byte equal to 0xFF.

7.4.2.47.4.2.7 **Slice layer RBSP semantics**

The slice layer RBSP consists of a slice header and slice data.

[Ed. (TW): insert text]

7.4.2.47.4.2.8 **RBSP slice trailing bits semantics**

cabac_zero_word is a byte-aligned sequence of two bytes equal to 0x0000.

Let NumBytesInVclNALunits be the sum of the values of NumBytesInNALunit for all VCL NAL units of a coded picture.

Let BinCountsInNALunits be the number of times that the parsing process function DecodeBin( ), specified in subclause 9.3.3.2, is invoked to decode the contents of all VCL NAL units of a coded picture. When entropy_coding_mode_flag is equal to 1, BinCountsInNALunits shall not exceed (32 ÷ 3) * NumBytesInVclNALunits + (RawMbBits * PicSizeInMbs) ÷ 32.

NOTE – The constraint on the maximum number of bins resulting from decoding the contents of the slice layer NAL units can be met by inserting a number of cabac_zero_word syntax elements to increase the value of NumBytesInVclNALunits. Each cabac_zero_word is represented in a NAL unit by the three-byte sequence 0x000003 (as a result of the constraints on NAL unit contents that result in requiring inclusion of an emulation_prevention_three_byte for each cabac_zero_word).

7.4.2.47.4.2.9 **RBSP trailing bits semantics**

rbsp_stop_one_bit shall be equal to 1.

rbsp_alignment_zero_bit shall be equal to 0.

7.4.2.10 RBSP byte alignment semantics

bit_equal_to_one specifies a bit equal to 1.

7.4.3 Slice header semantics

When present, the value of the slice header syntax elements pic_parameter_set_id, frame_num, field_pic_flag, bottom_field_flag, idr_pic_id, pic_order_cnt_lsb, delta_pic_order_cnt[0], delta_pic_order_cnt[1], and slice_group_change_cycle shall be the same in all slice headers of a coded picture.

lightweightentropy_**slice_flag** equal to 1 specifies that the value of slice header syntax elements not present shall be inferred to be equal to the value of slice header syntax elements in a proceeding slice, where a proceeding slice is defined as the slice containing treeblock with location (LCUAddress – 1). lightweightentropy_slice_flag shall be equal to 0 when LCUAddress equal to 0.

slice_type specifies the coding type of the slice according to Table 7-3.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000579

Table 7-3 – Name association to slice_type

slice_type	Name of slice_type
0	P (P slice)
1	B (B slice)
2	I (I slice)

When nal_unit_type is equal to 5 (IDR picture), slice_type shall be equal to 2.

When max_num_ref_frames is equal to 0, slice_type shall be equal to 2.

pic_parameter_set_id specifies the picture parameter set in use. The value of pic_parameter_set_id shall be in the range of 0 to 255, inclusive.

frame_num is used as an identifier for pictures and shall be represented by log2_max_frame_num_minus4 + 4 bits in the bitstream. frame_num is constrained as follows:

The variable PrevRefFrameNum is derived as follows.

- If the current picture is an IDR picture, PrevRefFrameNum is set equal to 0.
- Otherwise (the current picture is not an IDR picture), PrevRefFrameNum is set as follows.
  - If the decoding process for gaps in frame_num was invoked by the decoding process for an access unit that contained a non-reference picture that followed the previous access unit in decoding order that contained a reference picture, PrevRefFrameNum is set equal to the value of frame_num for the last of the "non-existing" reference frames inferred by the decoding process for gaps in frame_num.
  - Otherwise, PrevRefFrameNum is set equal to the value of frame_num for the previous access unit in decoding order that contained a reference picture.

The value of frame_num is constrained as follows.

- If the current picture is an IDR picture, frame_num shall be equal to 0.
- Otherwise (the current picture is not an IDR picture), referring to the primary coded picture in the previous access unit in decoding order that contains a reference picture as the preceding reference picture, the value of frame_num for the current picture shall not be equal to PrevRefFrameNum unless both of the following conditions are true.
  - the current picture and the preceding reference picture belong to consecutive access units in decoding order
  - one or more of the following conditions is true
    - the preceding reference picture is an IDR picture
    - the preceding reference picture includes a memory_management_control_operation syntax element equal to 5

      NOTE 1 – When the preceding reference picture includes a memory_management_control_operation syntax element equal to 5, PrevRefFrameNum is equal to 0.
    - there is a primary coded picture that precedes the preceding reference picture and the primary coded picture that precedes the preceding reference picture does not have frame_num equal to PrevRefFrameNum
    - there is a primary coded picture that precedes the preceding reference picture and the primary coded picture that precedes the preceding reference picture is not a reference picture

When the value of frame_num is not equal to PrevRefFrameNum, the following applies.

- There shall not be any previous frame in decoding order that is currently marked as "used for short-term reference" that has a value of frame_num equal to any value taken on by the variable UnusedShortTermFrameNum in the following:

```
UnusedShortTermFrameNum = ( PrevRefFrameNum + 1 ) % MaxFrameNum
while( UnusedShortTermFrameNum != frame_num )
    UnusedShortTermFrameNum = ( UnusedShortTermFrameNum + 1 ) % MaxFrameNum
```

- The value of frame_num is constrained as follows.
  - If gaps_in_frame_num_value_allowed_flag is equal to 0, the value of frame_num for the current picture shall be equal to (PrevRefFrameNum + 1) % MaxFrameNum.

Formatted: Highlight

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000580

- Otherwise (gaps_in_frame_num_value_allowed_flag is equal to 1), the following applies.
  - If frame_num is greater than PrevRefFrameNum, there shall not be any non-reference pictures in the bitstream that follow the previous reference picture and precede the current picture in decoding order in which either of the following conditions is true.
    - The value of frame_num for the non-reference picture is less than PrevRefFrameNum.
    - The value of frame_num for the non-reference picture is greater than the value of frame_num for the current picture.
  - Otherwise (frame_num is less than PrevRefFrameNum), there shall not be any non-reference pictures in the bitstream that follow the previous reference picture and precede the current picture in decoding order in which both of the following conditions are true.
    - The value of frame_num for the non-reference picture is less than PrevRefFrameNum.
    - The value of frame_num for the non-reference picture is greater than the value of frame_num for the current picture.

A picture including a memory_management_control_operation equal to 5 shall have frame_num constraints as described above and, after the decoding of the current picture and the processing of the memory management control operations, the picture shall be inferred to have had frame_num equal to 0 for all subsequent use in the decoding process.

idr_pic_id identifies an IDR picture. The values of idr_pic_id in all the slices of an IDR picture shall remain unchanged. When two consecutive access units in decoding order are both IDR access units, the value of idr_pic_id in the slices of the first such IDR access unit shall differ from the idr_pic_id in the second such IDR access unit. The value of idr_pic_id shall be in the range of 0 to 65535, inclusive.

pic_order_cnt_lsb specifies the picture order count modulo MaxPicOrderCntLsb for the top field of a coded frame or for a coded field. The length of the pic_order_cnt_lsb syntax element is log2_max_pic_order_cnt_lsb_minus4 + 4 bits. The value of the pic_order_cnt_lsb shall be in the range of 0 to MaxPicOrderCntLsb – 1, inclusive.

num_ref_idx_active_override_flag equal to 1 specifies that the syntax element num_ref_idx_l0_active_minus1 is present for P and B slices and that the syntax element num_ref_idx_l1_active_minus1 is present for B slices. num_ref_idx_active_override_flag equal to 0 specifies that the syntax elements num_ref_idx_l0_active_minus1 and num_ref_idx_l1_active_minus1 are not present.

When the current slice is a P or B slice and field_pic_flag is equal to 0 and the value of num_ref_idx_l0_default_active_minus1 in the picture parameter set exceeds 15, num_ref_idx_active_override_flag shall be equal to 1.

When the current slice is a B slice and field_pic_flag is equal to 0 and the value of num_ref_idx_l1_default_active_minus1 in the picture parameter set exceeds 15, num_ref_idx_active_override_flag shall be equal to 1.

num_ref_idx_l0_active_minus1 specifies the maximum reference index for reference picture list 0 that shall be used to decode the slice.

When the current slice is a P or B slice and num_ref_idx_l0_active_minus1 is not present, num_ref_idx_l0_active_minus1 shall be inferred to be equal to num_ref_idx_l0_default_active_minus1.

The range of num_ref_idx_l0_active_minus1 is specified as follows.p

- If field_pic_flag is equal to 0, num_ref_idx_l0_active_minus1 shall be in the range of 0 to 15, inclusive. When MbaffFrameFlag is equal to 1, num_ref_idx_l0_active_minus1 is the maximum index value for the decoding of frame macroblocks and 2 * num_ref_idx_l0_active_minus1 + 1 is the maximum index value for the decoding of field macroblocks.
- Otherwise (field_pic_flag is equal to 1), num_ref_idx_l0_active_minus1 shall be in the range of 0 to 31, inclusive.

num_ref_idx_l1_active_minus1 specifies the maximum reference index for reference picture list 1 that shall be used to decode the slice.

When the current slice is a B slice and num_ref_idx_l1_active_minus1 is not present, num_ref_idx_l1_active_minus1 shall be inferred to be equal to num_ref_idx_l1_default_active_minus1.

The range of num_ref_idx_l1_active_minus1 is constrained as specified in the semantics for num_ref_idx_l0_active_minus1 with l0 and list 0 replaced by l1 and list 1, respectively.

~~cabac_init_idx specifies the index for determining the initialization table used in the initialization process for context variables. The value of cabac_init_idx shall be in the range of 0 to 2, inclusive.~~

first_slice_in_pic_flag indicates whether the slice is the first slice of the picture. If first_slice_in_pic_flag is equal to 1, the variables SliceAddress and LCUAddress are both set to 0 and the decoding starts with the first LCU in the picture.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000581

slice_address specifies the address in slice granularity resolution in which the slice starts and shall be represented by (Ceil(Log2(NumLCUsInPicture)) + SliceGranularity) bits in the bitstream where NumLCUsInPicture is the number of LCUs in a picture.

The variable LCUAddress is set to (slice_address >> SliceGranularity) and represents the LCU part of the slice address in raster scan order. The variable GranularityAddress is set to (slice_address - (LCUAddress << SliceGranularity)) and represents the sub-LCU part of the slice address expressed in z-scan order.

The variable SliceAddress is then set to (LCUAddress << (log2_diff_max_min_coding_block_size << 1)) + (GranularityAddress << ((log2_diff_max_min_coding_block_size << 1) – SliceGranularity) and the slice decoding starts with the largest coding unit possible at the slice starting coordinate.

slice_qp_delta specifies the initial value of QP_Y to be used for all the macroblocks in the slice until modified by the value of cu_qp_delta in the coding unit layer. The initial QP_Y quantisation parameter for the slice is computed as

$$SliceQP_Y = 26 + pic\_init\_qp\_minus26 + slice\_qp\_delta \quad (7\text{-}9)$$

The value of slice_qp_delta shall be limited such that $SliceQP_Y$ is in the range of $-QpBdOffset_Y$ to +51, inclusive.

disable_deblocking_filter_flag equal to 1 specifies that the operation of the deblocking filter shall be disabled and slice_deblocking_filter_flag equal to 0 specifies that the operation of the deblocking filter shall be enabled.

collocated_from_l0_flag equal to 1 specifies the picture that contains the co-located partition shall be derived from list 0, otherwise the picture shall be derived from list 1.

7.4.3.1 Reference picture list modification semantics

The syntax elements modification_of_pic_nums_idc, abs_diff_pic_num_minus1, and long_term_pic_num specify the change from the initial reference picture lists to the reference picture lists to be used for decoding the slice.

ref_pic_list_modification_flag_l0 equal to 1 specifies that the syntax element modification_of_pic_nums_idc is present for specifying reference picture list 0. ref_pic_list_modification_flag_l0 equal to 0 specifies that this syntax element is not present.

When ref_pic_list_modification_flag_l0 is equal to 1, the number of times that modification_of_pic_nums_idc is not equal to 3 following ref_pic_list_modification_flag_l0 shall not exceed num_ref_idx_l0_active_minus1 + 1.

When RefPicList0[num_ref_idx_l0_active_minus1] in the initial reference picture list produced as specified in subclause 8.2.2.2 is equal to "no reference picture", ref_pic_list_modification_flag_l0 shall be equal to 1 and modification_of_pic_nums_idc shall not be equal to 3 until RefPicList0[num_ref_idx_l0_active_minus1] in the modified list produced as specified in subclause 8.2.2.3 is not equal to "no reference picture".

ref_pic_list_modification_flag_l1 equal to 1 specifies that the syntax element modification_of_pic_nums_idc is present for specifying reference picture list 1. ref_pic_list_modification_flag_l1 equal to 0 specifies that this syntax element is not present.

When ref_pic_list_modification_flag_l1 is equal to 1, the number of times that modification_of_pic_nums_idc is not equal to 3 following ref_pic_list_modification_flag_l1 shall not exceed num_ref_idx_l1_active_minus1 + 1.

When decoding a slice with slice_type equal to 1 or 6 and RefPicList1[num_ref_idx_l1_active_minus1] in the initial reference picture list produced as specified in subclause 8.2.2.2 is equal to "no reference picture", ref_pic_list_modification_flag_l1 shall be equal to 1 and modification_of_pic_nums_idc shall not be equal to 3 until RefPicList1[num_ref_idx_l1_active_minus1] in the modified list produced as specified in subclause 8.2.2.3 is not equal to "no reference picture".

modification_of_pic_nums_idc together with abs_diff_pic_num_minus1 or long_term_pic_num specifies which of the reference pictures are re-mapped. The values of modification_of_pic_nums_idc are specified in Table 7-4. The value of the first modification_of_pic_nums_idc that follows immediately after ref_pic_list_modification_flag_l0 or ref_pic_list_modification_flag_l1 shall not be equal to 3.

Formatted: Font: Not Bold

Formatted: Font: Not Bold

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000582

Table 7-4 – modification_of_pic_nums_idc operations for modification of reference picture lists

modification_of_pic_nums_idc	modification specified
0	abs_diff_pic_num_minus1 is present and corresponds to a difference to subtract from a picture number prediction value
1	abs_diff_pic_num_minus1 is present and corresponds to a difference to add to a picture number prediction value
2	long_term_pic_num is present and specifies the long-term picture number for a reference picture
3	End loop for modification of the initial reference picture list

abs_diff_pic_num_minus1 plus 1 specifies the absolute difference between the picture number of the picture being moved to the current index in the list and the picture number prediction value. abs_diff_pic_num_minus1 shall be in the range of 0 to MaxPicNum − 1. The allowed values of abs_diff_pic_num_minus1 are further restricted as specified in subclause 8.2.2.3.1.

[Ed. (TW): clarify the following paragraph]

long_term_pic_num specifies the long-term picture number of the picture being moved to the current index in the list. When decoding a coded frame, long_term_pic_num shall be equal to a LongTermPicNum assigned to one of the reference frames or complementary reference field pairs marked as "used for long-term reference". When decoding a coded field, long_term_pic_num shall be equal to a LongTermPicNum assigned to one of the reference fields marked as "used for long-term reference".

7.4.3.2 Reference picture lists combination semantics

ref_pic_list_combination_flag equal to 1 indicates that the reference picture list 0 and the reference picture list 1 are combined to be an additional reference picture lists combination used for the prediction units being uni-directional predicted. This flag equal to 0 indicates that the reference picture list 0 and reference picture list 1 are identical thus reference picture list 0 is used as the reference picture lists combination. The reference picture lists combination is set to be empty at the start of the loop defined in this table.

num_ref_idx lc_active_minus1+1 specifies the number of reference pictures selected from reference picture list 0 or reference picture list 1 in the reference picture lists combination.

ref_pic_list_modification_flag_lc equal to 1 specifies that the syntax element pic_from_list_0_flag and ref_idx_list_curr are present for specifying the mapping for the entries of the reference picture lists combination to the entries of reference picture list 0 and reference picture list 1.

ref_pic_list_modification_flag_lc equal to 0 specifies that these syntax elements are not present. The reference picture lists combination is initialized as specified in subsclause X.X.X.X.

pic_from_list_0_flag indicates the current reference picture added into the reference picture lists combination is from reference picture list 0 or reference picture list 1. When this flag is equal to 1, the picture is from the reference picture list 0, and the CurrRefPicList is reference picture list 0; when this flag is equal to 0, the picture is from the reference picture list 1, and the CurrRefPicList is reference picture list 1;

ref_idx_list_curr indicates the reference index of the picture in the CurrRefPicList to be appended at the end of the reference picture lists combination.

7.4.3.3 Decoded reference picture marking semantics

[Ed. (TW): clarify the following paragraphs wrt frame/field/picture]

The syntax elements no_output_of_prior_pics_flag, long_term_reference_flag, adaptive_ref_pic_marking_mode_flag, memory_management_control_operation, difference_of_pic_nums_minus1, long_term_frame_idx, long_term_pic_num, and max_long_term_frame_idx_plus1 specify marking of the reference pictures.

The marking of a reference picture can be "unused for reference", "used for short-term reference", or "used for long-term reference", but only one among these three. When a reference picture is referred to as being marked as "used for reference", this collectively refers to the picture being marked as "used for short-term reference" or "used for long-term reference" (but not both). A reference picture that is marked as "used for short-term reference" is referred to as a short-term reference picture. A reference picture that is marked as "used for long-term reference" is referred to as a long-term reference picture.

The content of the decoded reference picture marking syntax structure shall be the same in all slice headers of the primary coded picture.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000583

The syntax category of the decoded reference picture marking syntax structure shall be inferred as follows.

– If the decoded reference picture marking syntax structure is in a slice header, the syntax category of the decoded reference picture marking syntax structure is inferred to be equal to 2.

– Otherwise (the decoded reference picture marking syntax structure is in a decoded reference picture marking repetition SEI message as specified in Annex D), the syntax category of the decoded reference picture marking syntax structure is inferred to be equal to 5.

no_output_of_prior_pics_flag specifies how the previously-decoded pictures in the decoded picture buffer are treated after decoding of an IDR picture. When the IDR picture is the first IDR picture in the bitstream, the value of no_output_of_prior_pics_flag has no effect on the decoding process. When the IDR picture is not the first IDR picture in the bitstream and the value of PicWidthInMbs, FrameHeightInMbs, or max_dec_frame_buffering derived from the active sequence parameter set is different from the value of PicWidthInMbs, FrameHeightInMbs, or max_dec_frame_buffering derived from the sequence parameter set active for the preceding picture, no_output_of_prior_pics_flag equal to 1 may (but should not) be inferred by the decoder, regardless of the actual value of no_output_of_prior_pics_flag.

long_term_reference_flag equal to 0 specifies that the MaxLongTermFrameIdx variable is set equal to "no long-term picture indices" and that the IDR picture is marked as "used for short-term reference". long_term_reference_flag equal to 1 specifies that the MaxLongTermFrameIdx variable is set equal to 0 and that the current IDR picture is marked "used for long-term reference" and is assigned LongTermFrameIdx equal to 0. When max_num_ref_frames is equal to 0, long_term_reference_flag shall be equal to 0.

adaptive_ref_pic_marking_mode_flag selects the reference picture marking mode of the currently decoded picture as specified in Table 7-5. adaptive_ref_pic_marking_mode_flag shall be equal to 1 when the number of pictures that are currently marked as "used for long-term reference" is equal to Max(max_num_ref_frames, 1).

Table 7-5 – Interpretation of adaptive_ref_pic_marking_mode_flag

adaptive_ref_pic_marking_mode_flag	Reference picture marking mode specified
0	Sliding window reference picture marking mode: A marking mode providing a first-in first-out mechanism for short-term reference pictures.
1	Adaptive reference picture marking mode: A reference picture marking mode providing syntax elements to specify marking of reference pictures as "unused for reference" and to assign long-term frame indices.

memory_management_control_operation specifies a control operation to be applied to affect the reference picture marking. The memory_management_control_operation syntax element is followed by data necessary for the operation specified by the value of memory_management_control_operation. The values and control operations associated with memory_management_control_operation are specified in Table 7-6. The memory_management_control_operation syntax elements are processed by the decoding process in the order in which they appear in the slice header, and the semantics constraints expressed for each memory_management_control_operation apply at the specific position in that order at which that individual memory_management_control_operation is processed.

memory_management_control_operation shall not be equal to 1 in a slice header unless the specified reference picture is marked as "used for short-term reference" when the memory_management_control_operation is processed by the decoding process.

memory_management_control_operation shall not be equal to 2 in a slice header unless the specified long-term picture number refers to a reference picture that is marked as "used for long-term reference" when the memory_management_control_operation is processed by the decoding process.

memory_management_control_operation shall not be equal to 3 in a slice header unless the specified reference picture is marked as "used for short-term reference" when the memory_management_control_operation is processed by the decoding process.

memory_management_control_operation shall not be equal to 3 or 6 if the value of the variable MaxLongTermFrameIdx is equal to "no long-term picture indices" when the memory_management_control_operation is processed by the decoding process.

Not more than one memory_management_control_operation equal to 4 shall be present in a slice header.

Not more than one memory_management_control_operation equal to 5 shall be present in a slice header.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000584

Not more than one memory_management_control_operation equal to 6 shall be present in a slice header.

memory_management_control_operation shall not be equal to 5 in a slice header unless no memory_management_control_operation in the range of 1 to 3 is present in the same decoded reference picture marking syntax structure.

A memory_management_control_operation equal to 5 shall not follow a memory_management_control_operation equal to 6 in the same slice header.

When a memory_management_control_operation equal to 6 is present, any memory_management_control_operation equal to 2, 3, or 4 that follows the memory_management_control_operation equal to 6 within the same slice header shall not specify the current picture to be marked as "unused for reference".

NOTE 2 – These constraints prohibit any combination of multiple memory_management_control_operation syntax elements that would specify the current picture to be marked as "unused for reference". However, some other combinations of memory_management_control_operation syntax elements are permitted that may affect the marking status of other reference pictures more than once in the same slice header. In particular, it is permitted for a memory_management_control_operation equal to 3 that specifies a long-term frame index to be assigned to a particular short-term reference picture to be followed in the same slice header by a memory_management_control_operation equal to 2, 3, 4 or 6 that specifies the same reference picture to subsequently be marked as "unused for reference".

Table 7-6 – Memory management control operation (memory_management_control_operation) values

memory_management_control_operation	Memory Management Control Operation
0	End memory_management_control_operation syntax element loop
1	Mark a short-term reference picture as "unused for reference"
2	Mark a long-term reference picture as "unused for reference"
3	Mark a short-term reference picture as "used for long-term reference" and assign a long-term frame index to it
4	Specify the maximum long-term frame index and mark all long-term reference pictures having long-term frame indices greater than the maximum value as "unused for reference"
5	Mark all reference pictures as "unused for reference" and set the MaxLongTermFrameIdx variable to "no long-term frame indices"
6	Mark the current picture as "used for long-term reference" and assign a long-term frame index to it

difference_of_pic_nums_minus1 is used (with memory_management_control_operation equal to 3 or 1) to assign a long-term frame index to a short-term reference picture or to mark a short-term reference picture as "unused for reference". When the associated memory_management_control_operation is processed by the decoding process, the resulting picture number derived from difference_of_pic_nums_minus1 shall be a picture number assigned to one of the reference pictures marked as "used for reference" and not previously assigned to a long-term frame index.

The resulting picture number shall be one of the set of picture numbers assigned to reference pictures.

long_term_pic_num is used (with memory_management_control_operation equal to 2) to mark a long-term reference picture as "unused for reference". When the associated memory_management_control_operation is processed by the decoding process, long_term_pic_num shall be equal to a long-term picture number assigned to one of the reference pictures that is currently marked as "used for long-term reference".

The resulting long-term picture number shall be one of the set of long-term picture numbers assigned to reference pictures.

long_term_frame_idx is used (with memory_management_control_operation equal to 3 or 6) to assign a long-term picture index to a picture. When the associated memory_management_control_operation is processed by the decoding process, the value of long_term_frame_idx shall be in the range of 0 to MaxLongTermFrameIdx, inclusive.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000585

max_long_term_frame_idx_plus1 minus 1 specifies the maximum value of long-term picture index allowed for long-term reference pictures (until receipt of another value of max_long_term_frame_idx_plus1). The value of max_long_term_frame_idx_plus1 shall be in the range of 0 to max_num_ref_frames, inclusive.

7.4.3.4 Sample adaptive offset parameter semantics

~~sample_adaptive_offset_flag specifies whether sample adaptive offset applies or not for the current picture.~~

sao_flag_cb equal to 1 denotes sample adaptive offset process for Cb shall be applied to the current picture.

sao_flag_cr equal to 1 denotes sample adaptive offset process for Cr shall be applied to the current picture.

sao_split_flag[cIdx][saoDepth][rx][ry] specifies whether a region is split into four sub regions with half horizontal and vertical number of LCU for the color component cIdx. The array indices rx and ry specify the region index and saoDepth specifies the split depth of the region. When sao_split_flag[cIdx][saoDepth][rx][ry] is not present, it shall be inferred to be equal to 0.

The maximum allowed depth for sample adaptive offset process SaoMaxDepth is derived as follows:

SaoMaxDepth = Min(4, Min(Floor(Log2(PicWidthInLCUs)), Floor(Log2(PicHeightInLCUs)))) (7-10)

where

PicWidthInLCUs = Ceil($PicWidthInSamples_L$ ÷ (1 << Log2MaxCUSize)) (7-10)
PicHeightInLCUs = Ceil($PicHeightInSamples_L$ ÷ (1 << Log2MaxCUSize)) (7-10)

[Ed.: (WJ) PicWidthInLCUs and PicHeightInLCUs could be defined elsewhere]

sao_type_idx[cIdx][saoDepth][rx][ry] indicates the offset type for the color component cIdx of the region specified by saoDepth, rx and ry.

sao_offset[cIdx][saoDepth][rx][ry][i] indicates the offset value of i-th category for the color component cIdx of the region specified by saoDepth, rx and ry.

The variable bitDepth is derived as follows.

– If cIdx is equal to 0, bitDepth is set equal to $BitDepth_Y$.

– Otherwise (cIdx is equal to1 or 2), bitDepth is set equal to $BitDepth_C$.

The offset value shall be in the range of [-(1 << (SaoBitsRange – 1)), (1<< (SaoBitsRange – 1)) – 1] where

SaoBitRange = Min(bitDepth, 10) – 4 (7-10)

An array SaoOffsetVal is specified as

SaoOffsetVal[cIdx][saoDepth][rx][ry][0] = 0 (7-10)

SaoOffsetVal[cIdx][saoDepth][rx][ry][i + 1] =

sao_offset[cIdx][saoDepth][rx][ry][i] << (bitDepth – Min(bitDepth, 10))

with i = 0..NumSaoCategory – 1 (7-10)

The number of categories, NumSaoClass, is specified in Table 7-7.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000586

Table 7-7 – Specification of NumSaoClass

sao_type_idx[cIdx][saoDepth][rx][ry]	NumSaoCategory	Edge type (informative)
0	0	Not applied
1	4	1D 0-degree edge
2	4	1D 90-degree edge
3	4	1D 135-degree edge
4	4	1D 45-degree edge
5	16	Central band
6	16	Side band

7.4.3.5 Adaptive loop filter parameter semantics

~~adaptive_loop_filter_flag specifies whether adaptive loop filter applies or not for the current slice.~~

alf_region_adaptation_flag specifies the filter adaptation method shall be applied for the current slice. If alf_region_adaptation_flag is equal to 1, the region-based filter adaptation shall be applied, otherwise, the block-based directional-activity filter adaptation shall be applied.

alf_length_luma_minus5_div2 specifies the filter length in horizontal direction for the luma component alfTap used in the adaptive loop filter process as specified by

$$alfTap = (alf\_length\_luma\_minus5\_div2 << 1) + 5 \quad (7\text{-}11)$$

The number of coded luma filter coefficients AlfCodedLengthLuma and the number of luma filter coefficients AlfLengthLuma are derived as follows:

– If alfTap is equal to 9, [Ed.: (WJ) truncated diamond shape]

$$AlfCodedLengthLuma = ((alfTap * alfTap) >> 2) + 1 \quad (7\text{-}11)$$

$$AlfLengthLuma = ((alfTap * alfTap) >> 1) \quad (7\text{-}12)$$

– Otherwise (alfTap is less than 9), [Ed.: (WJ) diamond shape]

$$AlfCodedLengthLuma = ((alfTap * alfTap) >> 2) + 2 \quad (7\text{-}13)$$

$$AlfLengthLuma = ((alfTap * alfTap) >> 1) + 2 \quad (7\text{-}14)$$

alf_no_filters_minus1 plus 1 specifies the number of filter sets for the current slice.

alf_start_second_filter specifies the variance index of luma samples where the second filter is applied, when alf_no_filters_minus1 is equal to 1.

alf_filter_pattern[i] specifies the filter index array corresponding to i-th variance index of luma samples, when alf_no_filters_minus1 is greater than 1. The number of filter sets AlfNumFilters is derived as follows:

– If alf_no_filters_minus1 is less than 2,

$$AlfNumFilters = alf\_no\_filters\_minus1 + 1 \quad (7\text{-}15)$$

– Otherwise (alf_no_filters_minus1 is greater than 2)

$$AlfNumFilters = \Sigma_i\ alf\_filter\_pattern[i] \text{ with } i = 0..15 \quad (7\text{-}16)$$

alf_pred_method specifies whether the filter coefficients are coded in a predictive way.

alf_min_kstart_minus1 plus 1 specifies the minimum order k of k-th order exponential golomb code for the luma filter coefficients for the adaptive loop filter.

alf_golomb_index_bit specifies the difference in order k of k-th order exponential golomb code for the different groups of the luma filter coefficients. Note that there are several groups of the luma filter coefficients where each group may have different order k.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000587

alf_coeff_luma[i **][** j **]** specifies the j-th filter coefficient of i-th filter for the adaptive loop filtering process for luma samples.

alf_chroma_idc specifies which chroma components are to be filtered.

Table 7-8 – Specification of alf_chroma_idc

alf_chroma_idc	chroma component to be filtered
0	None
1	Cr
2	Cb
3	Cb and Cr

alf_length_chroma_minus5_div2 specifies the filter length in horizontal direction for the luma component alfTapC used in the adaptive loop filter process as specified by

$$alfTapC = (alf\_length\_chroma\_minus5\_div2 << 1) + 5 \quad (7\text{-}17)$$

The number of coded chroma filter coefficients AlfCodedLengthChroma and the number of chroma filter coefficients AlfLengthChroma are derived as follows:

$$AlfCodedLengthChroma = ((alfTapC * alfTapC - 1) >> 1) + 2 \quad (7\text{-}18)$$

$$AlfLengthChroma = (alfTapC * alfTapC) + 1 \quad (7\text{-}19)$$

alf_coeff_chroma[i **]** specifies the i-th filter coefficient for the adaptive loop filtering process for chroma samples.

7.4.3.6 Adaptive loop filter coding unit control parameter semantics

alf_cu_control_flag specifies whether the adaptive loop filter process for luma component shall be applied adaptively according to the coding unit. If alf_cu_control_flag is equal to 1, the filtering process shall be applied only when alf_cu_flag is equal to 1, otherwise, the filtering process shall be applied to all luma samples in the current slice.

alf_cu_control_max_depth specifies the maximum split depth from treeblock for transmitting the flag indicating the application of the adaptive loop filter process.

alf_length_cu_control_info specifies the information of the number of alf_cu_flag succeeding this information. When the variable specifying the number of alf_cu_flag shall be computed as

$$NumAlfCuFlag = alf\_length\_cu\_control\_info + NumTBsInPicture \quad (7\text{-}20)$$

alf_cu_flag[i**]** specifies the information whether a coding unit is filtered when the adaptive loop filter is applied.

7.4.4 Slice data semantics

end_of_slice_flag equal to 0 specifies that another macroblock is following in the slice. end_of_slice_flag equal to 1 specifies the end of the slice and that no further macroblock follows.

7.4.5 Coding tree semantics

split_coding_unit_flag[x0 **][** y0 **]** specifies whether a coding unit is split into coding units with half horizontal and vertical size. The array indices x0, y0 specify the location (x0, y0) of the top-left luma sample of the considered coding block relative to the top-left luma sample of the picture.

When split_coding_unit_flag[x0][y0] is not present, the following applies:

– If log2CUSize is greater than Log2MinCUSize, the value of split_coding_unit_flag[x0][y0] is inferred to be equal to 1.

– Otherwise (log2CUSize is equal to Log2MinCUSize), the value of split_coding_unit_flag[x0][y0] is inferred to be equal to 0.

cu_split_pred_part_mode[x0 **][** y0 **]** specifies split_coding_unit_flag and when the coding unit is not split the skip_flag[x0][y0], the merge_flag[x0][y0], PredMode and PartMode of a coding unit. The array indices x0 and y0 specify the location (x0, y0) of the top-left luma sample of the coding unit relative to the top-left luma sample of the picture.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000588

The variable allowInterNxN is specified as follows.

– If log2CUSize is greater than 3 or inter_4x4_enabled_flag is equal to 1, allowInterNxN is set to true.

– Otherwise (log2CUSize is equal to 3 and inter_4x4_enabled_flag is equal to 0), allowInterNxN is set to false.

Table 7-9 – Specification of cu_split_pred_part_mode when log2CUSize is greater than Log2MinCUSize

cu_split_pred_part_mode	split_coding_unit_flag	skip_flag	merge_flag	PredMode	PartMode
0	1	-	-	-	-
1	0	1	-	MODE_SKIP	PART_2Nx2N
2	0	0	1	MODE_INTER	PART_2Nx2N
3	0	0	0	MODE_INTER	PART_2Nx2N
4	0	-	-	MODE_INTER	PART_2NxN
5	0	-	-	MODE_INTER	PART_Nx2N
6	0	-	-	MODE_INTRA	PART_2Nx2N
7	0	-	-	MODE_INTER	PART_2NxnU
8	0	-	-	MODE_INTER	PART_2NxnD
9	0	-	-	MODE_INTER	PART_nLx2N
10	0	-	-	MODE_INTER	PART_nRx2N

Table 7-10 – Specification of cu_split_pred_part_mode when log2CUSize is equal to Log2MinCUSize and allowInterNxN is true

cu_split_pred_part_mode	split_coding_unit_flag	skip_flag	merge_flag	PredMode	PartMode
0	0	1	-	MODE_SKIP	PART_2Nx2N
1	0	0	1	MODE_INTER	PART_2Nx2N
2	0	0	0	MODE_INTER	PART_2Nx2N
3	0	-	-	MODE_INTER	PART_2NxN
4	0	-	-	MODE_INTER	PART_Nx2N
5 (escape symbol)	0	-	-	MODE_INTRA	PART_2Nx2N
				MODE_INTRA	PART_NxN
				MODE_INTER	PART_NXN

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000589

Table 7-11 – Specification of cu_split_pred_part_mode when log2CUSize is equal to Log2MinCUSize and allowInterNxN is equal to false

cu_split_pred_part_mode	split_coding_unit_flag	skip_flag	merge_flag	PredMode	PartMode
0	0	1	-	MODE_SKIP	PART_2Nx2N
1	0	0	1	MODE_INTER	PART_2Nx2N
2	0	0	0	MODE_INTER	PART_2Nx2N
3	0	-	-	MODE_INTER	PART_2NxN
4	0	-	-	MODE_INTER	PART_Nx2N
5 (escape symbol)	0	-	-	MODE_INTRA	PART_2Nx2N
				MODE_INTRA	PART_NxN

7.4.6 Coding unit semantics

skip_flag[x0][y0] equal to 1 specifies that for the current coding unit, when decoding a P or B slice, no more syntax elements except the motion vector predictor indices are parsed after skip_flag[x0][y0]. skip_flag[x0][y0] equal to 0 specifies that the coding unit is not skipped. The array indices x0, y0 specify the location (x0, y0) of the top-left luma sample of the considered coding block relative to the top-left luma sample of the picture.

When skip_flag[x0][y0] is not present, it shall be inferred to be equal to 0.

intra_part_mode specifies whether the PartMode for the current coding unit is PART_2Nx2N or PART_NxN, when the slice type is I and log2CUSize is equal to Log2MinCUSize. If intra_part_mode is equal to 1, the PartMode is PART_2Nx2N. Otherwise if intra_part_mode is equal to 0, the PartMode is PART_NxN.

pred_type specifies prediction mode and partitioning type of the current coding unit. The semantics of pred_type depend on the slice type. The variables PredMode, PartMode and IntraSplitFlag are derived from the value of pred_type as defined in Table 7-8.

When log2CUSize is greater than Log2MinCUSize, pred_type shall not be equal to 3 or 9.

When log2CUSize is equal to 3 and inter_4x4_enabled_flag is equal to 0, pred_type shall not be equal to 3.

When pred_type is not present, the variables PredMode and PartMode are derived as follows.

– If slice_type is equal to I and log2CUSize is greater than Log2MinCUSize,

  -PredMode is inferred to be equal to MODE_INTRA

  -PartMode is inferred to be equal to PART_2Nx2N

  -IntraSplitFlag is inferred to be equal to 0

– Otherwise (slice_type is equal to P or B), if skip_flag[x0][y0] is equal to 1,

  -PredMode is inferred to be equal to MODE_SKIP

  -PartMode is inferred to be equal to PART_2Nx2N

pcm_flag specifies whether the associated coding unit with PART_2Nx2N is coded by I_PCM: If the pcm_flag is equal to 1, the associated coding unit with PART_2Nx2N is coded by I_PCM. When the pcm_flag is not present, it shall be infered to be equal to 0.

pcm_alignment_zero_bit is a bit equal to 0.

pcm_sample_luma[i] represents a coded luma sample value in the raster scan within the coding unit. The number of bits used to represent each of these samples is $PCMBitDepth_Y$.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000590

pcm_sample_chroma[i] represents a coded chroma sample value in the raster scan within the coding unit. The first half of the values represent coded Cb samples and the remaining half of the values represent coded Cr samples. The number of bits used to represent each of these samples is PCMBitDepthc.

Table 7-12 - Name association to prediction mode and partitioning type

slice_type	pred_type	IntraSplitFlag	PredMode	PartMode
I	0	0	MODE_INTRA	PART_2Nx2N
	1	1	MODE_INTRA	PART_NxN
P or B	0	-	MODE_INTER	PART_2Nx2N
	1	-	MODE_INTER	PART_2NxN
	2	-	MODE_INTER	PART_Nx2N
	3	-	MODE_INTER	PART_NxN
	4	-	MODE_INTER	PART_2NxnU
	5	-	MODE_INTER	PART_2NxnD
	6	-	MODE_INTER	PART_nLx2N
	7	-	MODE_INTER	PART_nRx2N
	8	0	MODE_INTRA	PART_2Nx2N
	9	1	MODE_INTRA	PART_NxN
	inferred	-	MODE_SKIP	PART_2Nx2N

7.4.7 Prediction unit semantics

mvp_idx_l0[x0][y0] specifies the motion vector predictor index of list 0 where x0, y0 specify the location (x0, y0) of the top-left luma sample of the considered prediction block relative to the top-left luma sample of the picture.

When mvp_idx_l0[x0][y0] is not present, it shall be inferred to be equal to 0.

mvp_idx_l1[x0][y0] has the same semantics as mvp_idx_l0, with l0 and list 0 replaced by l1 and list 1, respectively.

mvp_idx_lc[x0][y0] has the same semantics as mvp_idx_l0, with l0 and list 0 replaced by lc and list combination, respectively.

prev_intra_luma_pred_flag[x0][y0], **mpm_idx**[x0][y0] and **rem_intra_luma_pred_mode**[x0][y0] specify the intra prediction mode for luma samples. The array indices x0, y0 specify the location (x0, y0) of the top-left luma sample of the considered prediction block relative to the top-left luma sample of the picture. When prev_intra_luma_pred_flag[x0][y0] is equal to 1, the intra prediction mode is inferred from a neighbouring intra-predicted prediction unit according to subclause 8.3.1.

~~planar_flag_luma[x0][y0] specifies whether the luma block shall be derived by the planar prediction when IntraPredMode[x0][y0] is equal to 2. If planar_flag_luma[x0][y0] is not present, planar_flag_luma[x0][y0] shall be inferred to be equal to 0.~~

intra_chroma_pred_mode[x0][y0] specifies the intra prediction mode for chroma samples. The array indices x0, y0 specify the location (x0, y0) of the top-left luma sample of the considered prediction block relative to the top-left luma sample of the picture.

~~planar_flag_chroma[x0][y0] specifies whether the chroma block shall be derived by the planar prediction when IntraPredModeC is equal to 2.~~

~~If planar_flag_chroma[x0][y0] is not present (SignaledAsChromaDC is not equal to 1 or IntraPredMode[x0][y0] is equal to Intra_DC or IntraPredMode[x0][y0] is equal to Intra_Planar), planar_flag_chroma[x0][y0] shall be inferred as follows:~~

~~If SignaledAsChromaDC is not equal to 1, planar_flag_chroma[x0][y0] shall be inferred to be equal to 0.~~

~~Otherwise, if IntraPredMode[x0][y0] is equal to Intra_DC or Intra_Planar, the following applies.~~

~~If chroma_pred_from_luma_enabled_flag is equal to 1, the following applies.~~

~~If intra_chroma_pred_mode[x0][y0] is equal to 3, planar_flag_chroma[x0][y0] is set equal to planar_flag_luma[x0][y0]. [Ed. (WJ): possible conflict with codeword switch]~~

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000591

~~Otherwise (intra_chroma_pred_mode[x0][y0] is equal to 4), planar_flag_chroma[x0][y0] is set equal to planar_flag_luma[x0][y0].~~

~~Otherwise (chroma_pred_from_luma_enabled_flag is equal to 0), the following applies.~~

~~If intra_chroma_pred_mode[x0][y0] is equal to 2, planar_flag_chroma[x0][y0] is set equal to 1 planar_flag_luma[x0][y0]. [Ed. (WJ): possible conflict with codeword switch]~~

~~Otherwise (intra_chroma_pred_mode[x0][y0] is equal to 3), planar_flag_chroma[x0][y0] is set equal to planar_flag_luma[x0][y0].~~

~~[Ed. (WJ): suggested to be noted from proponents under the specific value of intra_chroma_pred_mode, the planar_flag_chroma may be inferred based on planar_flag_luma. The binarization of intra_chroma_pred_mode should take account of this property and the codeword switching as listed in the open issue]~~

merge_flag[x0][y0] specifies whether the inter prediction parameters for the current prediction unit are inferred from a neighbouring inter-predicted partition. The array indices x0, y0 specify the location (x0, y0) of the top-left luma sample of the considered prediction block relative to the top-left luma sample of the picture.

merge_idx[x0][y0] specifies the merging candidate index of the merging candidate list where x0, y0 specify the location (x0, y0) of the top-left luma sample of the considered prediction block relative to the top-left luma sample of the picture.

When merge_idx[x0][y0] is not present, it is inferred to be equal to 0.

combined_inter_pred_ref_idx specifies the values of inter_pred_flag and ref_idx_lX with X being 0, 1 or c. When it is not present, it shall be inferred to be equal to 0. [Ed. (WJ): needs to be improved]

The variables NumPredRefLC, NumPredRefL0 and NumPredRefL1 are specified as

NumPredRefLC = Min(4, num_ref_idx_lc_active_minus1 + 1) (7-18)
NumPredRefL0 = Min(2, num_ref_idx_l0_active_minus1 + 1) (7-18)
NumPredRefL1 = Min(2, num_ref_idx_l1_active_minus1 + 1) (7-18)

The variable MaxPredRef is specified as

– If num_ref_idx_lc_active_minus1 is greater than 0, the following applies.

MaxPredRef = NumPredRefLC + NumPredRefL0 * NumPredRefL1 (7-18)

– Otherwise (combined list is not used), the following applies.

MaxPredRef = NumPredRefL0 + NumPredRefL0 * NumPredRefL1 (7-18)

inter_pred_flag[x0][y0] specifies whether uni-prediction, or bi-prediction is used for the current prediction unit according to Table 7-11. The array indices x0, y0 specify the location (x0, y0) of the top-left luma sample of the considered prediction block relative to the top-left luma sample of the picture.

Table 7-14 – Name association to inter prediction mode

slice_type	inter_pred_flag	Name of inter_pred_flag
P	inferred	Pred_L0
B	0	Pred_LC
	1	Pred_BI

When entropy_coding_mode_flag is equal to 1 and inter_pred_flag[x0][y0] is not present, it shall be inferred to be equal to Pred_L0 when slice_type is equal to P and Pred_BI when slice_type is equal to B.

When entropy_coding_mode_flag is equal to 0, inter_pred_flag[x0][y0] is not present and combined_inter_pred_ref_idx is present, it shall be inferred as follows:

– If num_ref_idx_lc_active_minus1 is greater than 0 and combined_inter_pred_ref_idx < NumPredRefLC, inter_pred_flag[x0][y0] is set equal to Pred_LC,

– Otherwise, if num_ref_idx_lc_active_minus1 is equal to 0 and combined_inter_pred_ref_idx < NumPredRefL0, inter_pred_flag[x0][y0] is set equal to Pred_L0,

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000592

– Otherwise, inter_pred_flag[x0][y0] is set equal to Pred_BI.

ref_idx_l0[x0][y0] specifies the list 0 reference picture index for the current prediction unit. The array indices x0, y0 specify the location (x0, y0) of the top-left luma sample of the considered prediction block relative to the top-left luma sample of the picture.

When ref_idx_l0[x0][y0] is not present and combined_inter_pred_ref_idx is present, ref_idx_l0[x0][y0] shall be inferred to be equal to ((combined_inter_pred_ref_idx – NumPredRefLC) / NumPredRefL1).

ref_idx_l0_minusX[x0][y0] + X specifies the value of ref_idx_l0[x0][y0] when combined_inter_pred_ref_idx is present and it is equal to MaxPredRef.

The constant X is derived as follows:

– If inter_pred_flag[x0][y0] is Pred_L0, X is set equal to 2,

– Otherwise, X is set equal to 0.

ref_idx_l1[x0][y0] has the same semantics as ref_idx_l0, with l0 and list 0 replaced by l1 and list 1, respectively.

When ref_idx_l1[x0][y0] is not present and combined_inter_pred_ref_idx is present, ref_idx_l1[x0][y0] shall be inferred to be equal to ((combined_inter_pred_ref_idx – NumPredRefLC) % NumPredRefL1).

ref_idx_l1_minusX[x0][y0] + X specifies the value of ref_idx_l1[x0][y0] when combined_inter_pred_ref_idx is present and it is equal to MaxPredRef.

The constant X is derived as follows:

– If ref_idx_l0[x0][y0] is less than 2, X is set equal to 2,

– Otherwise, X is set equal to 0.

ref_idx_lc[x0][y0] has the same semantics as ref_idx_l0, with l0 and list 0 replaced by lc and list combination, respectively.

When ref_idx_lc[x0][y0] is not present and combined_inter_pred_ref_idx is present, ref_idx_lc[x0][y0] shall be inferred to be equal to combined_inter_pred_ref_idx.

ref_idx_lc_minus4[x0][y0] + 4 specifies the value of ref_idx_lc [x0][y0] when combined_inter_pred_ref_idx is present and it is equal to MaxPredRef.

mvd_l0[x0][y0][compIdx], specifies the difference between a list 0 vector component to be used and its prediction. The array indices x0, y0 specify the location (x0, y0) of the top-left luma sample of the considered prediction block relative to the top-left luma sample of the picture. The horizontal motion vector component difference is assigned compIdx = 0 and the vertical motion vector component is assigned compIdx = 1.

When any of the two components is not present, the inferred value is 0.

mvd_l1[x0][y0][compIdx] has the same semantics as mvd_l0, with l0 and list 0 replaced by l1 and list 1, respectively.

mvd_lc[x0][y0][compIdx] has the same semantics as mvd_l0, with l0 and list 0 replaced by lc and list combination, respectively.

7.4.7.1 Motion vector difference coding CABAC semantics

abs_mvd_greater0_flag[compIdx] specifies whether the absolute value of a motion vector component difference is greater than 0. The horizontal motion vector component difference is assigned compIdx = 0 and the vertical motion vector component is assigned compIdx = 1.

abs_mvd_greater1_flag[compIdx] specifies whether the absolute value of a motion vector component difference is greater than 1. The horizontal motion vector component difference is assigned compIdx = 0 and the vertical motion vector component is assigned compIdx = 1.

When abs_mvd_greater1_flag[compIdx] is not present, it is inferred to be equal to 0.

abs_mvd_minus2[compIdx] is the absolute value of a motion vector component difference minus 2. The horizontal motion vector component difference is assigned compIdx = 0 and the vertical motion vector component is assigned compIdx = 1.

When abs_mvd_minus2[compIdx] is not present, it is inferred as follows.

– If abs_mvd_greater1_flag[compIdx] is equal to 0, abs_mvd_minus2[compIdx] is inferred to be equal to −1.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000593

– Otherwise (abs_mvd_greater1_flag[compIdx] is equal to 1), abs_mvd_minus2[compIdx] is inferred to be equal to 0.

mvd_sign_flag[compIdx] specifies the sign of a motion vector component difference as follows.

– If mvd_sign_flag[compIdx] is equal to 0, the corresponding motion vector component difference has a positive value.

– Otherwise (mvd_sign_flag[compIdx] is equal to 1), the corresponding motion vector component difference has a negative value.

The horizontal motion vector component difference is assigned compIdx = 0 and the vertical motion vector component is assigned compIdx = 1.

When mvd_sign_flag[compIdx] is not present, it is inferred to be equal to 0.

7.4.8 Transform tree semantics

The variable InterTUSplitDirection defines how a transform block is split into four blocks with smaller horizontal or vertical size for the purpose of transform coding. The blocks are half horizontal and vertical size when PredMode is equal to MODE_INTRA or InterTUSplitDirection is equal to 2, full horizontal and quarter vertical size when InterTUSplitDirection is equal to 0, quarter horizontal and full vertical size when InterTUSplitDirection is equal to 1. InterTUSplitDirection is specified as follows.

```
if( ( ( log2TrafoSize == Log2MaxTrafoSize
            || ( log2TrafoSize < Log2MaxTrafoSize && trafoDepth == 0 ) )
        && log2TrafoSize > ( Log2MinTrafoSize + 1 )
        && ( PartMode == PART_2NxN
            || PartMode == PART_2NxnU || PartMode == PART_2NxnD ) )
    || ( log2TrafoSize == ( Log2MinTrafoSize + 1 )
        && log2TrafoWidth < log2TrafoHeight ) )
{
    InterTUSplitDirection = 0
}
else if( ( ( log2TrafoSize == Log2MaxTrafoSize
            || ( log2TrafoSize < Log2MaxTrafoSize && trafoDepth == 0 ) )
        && log2TrafoSize > ( Log2MinTrafoSize + 1 )
        && ( PartMode == PART_Nx2N
            || PartMode == PART_nLx2N || PartMode == PART_nRx2N ) )
    || ( log2TrafoSize == ( Log2MinTrafoSize + 1 )
        && log2TrafoWidth > log2TrafoHeight ) )
{
    InterTUSplitDirection = 1
}
else
{
    InterTUSplitDirection = 2
}
```

no_residual_data_flag equal to 1 specifies that no residual data are present for the current coding unit. no_residual_data_flag equal to 0 specifies that residual data are present for the current coding unit.

When no_residual_data_flag is not present, its value shall be inferred to be equal to 0.

split_transform_flag[x0][y0][trafoDepth] specifies whether a block is split into four blocks with smaller horizontal or vertical size for the purpose of transform coding. The array indices x0, y0 specify the location (x0, y0) of the top-left luma sample of the considered block relative to the top-left luma sample of the picture. The array index trafoDepth specifies the current subdivision level of a coding unit into blocks for the purpose of transform coding. trafoDepth is equal to 0 for blocks that correspond to coding units.

When split_transform_flag[x0][y0][trafoDepth] is not present, it is inferred as follows:

– If log2TrafoSize is greater than Log2MaxTrafoSize or intraSplitFlag is equal to 1, the value of split_transform_flag[x0][y0][trafoDepth] is inferred to be equal to 1.

– Otherwise (log2TrafoSize is less than or equal to Log2MaxTrafoSize and intraSplitFlag is equal to 0), the value of split_transform_flag[x0][y0][trafoDepth] is inferred to be equal to 0.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000594

cbf_luma[x0][y0][trafoDepth] equal to 1 specifies that the luma transform block contains one or more transform coefficient levels not equal to 0. The array indices x0, y0 specify the location (x0, y0) of the top-left luma sample of the considered transform block relative to the top-left luma sample of the picture. The array index trafoDepth specifies the current subdivision level of a coding unit into blocks for the purpose of transform coding. trafoDepth is equal to 0 for blocks that correspond to coding units.

When cbf_luma[x0][y0][trafoDepth] is not present, it is inferred to be equal to 1.

cbf_cb[x0][y0][trafoDepth] equal to 1 specifies that the Cb transform block contains one or more transform coefficient levels not equal to 0. The array indices x0, y0 specify the location (x0, y0) of the top-left luma sample of the considered transform block relative to the top-left luma sample of the picture. The array index trafoDepth specifies the current subdivision level of a coding unit into blocks for the purpose of transform coding. trafoDepth is equal to 0 for blocks that correspond to coding units.

When cbf_cb[x0][y0][trafoDepth] is not present and PredMode is not equal to MODE_INTRA, the value of cbf_cb[x0][y0][trafoDepth] is inferred to be equal to 0.

cbf_cr[x0][y0][trafoDepth] equal to 1 specifies that the Cr transform block contains one or more transform coefficient levels not equal to 0. The array indices x0, y0 specify the location (x0, y0) of the top-left luma sample of the considered transform block relative to the top-left luma sample of the picture. The array index trafoDepth specifies the current subdivision level of a coding unit into blocks for the purpose of transform coding. trafoDepth is equal to 0 for blocks that correspond to coding units.

When cbf_cr[x0][y0][trafoDepth] is not present and PredMode is not equal to MODE_INTRA, the value of cbf_cr[x0][y0][trafoDepth] is inferred to be equal to 0.

cbp_and_split_transform contains a number of coded block flags and/or a split_transform_flag. Depending on a 4-bit value flagPattern defined as follows, the value of cbp_and_split_transform is defined according to Table 7-13.

```
codeLuma = trafoDepth == 0 || cbf_luma[ xB ][ yB ][ trafoDepth − 1 ]
codeCb =   trafoDepth == 0 || (cbf_cb[ xB ][ yB ][ trafoDepth − 1 ] &&
           log2TrafoSize > Log2MinTrafoSize )
codeCr =   trafoDepth == 0 || (cbf_cr[ xB ][ yB ][ trafoDepth − 1 ] &&
           log2TrafoSize > Log2MinTrafoSize )                                  (7-23)
codeSplitTrans = log2TrafoSize <= Log2MaxTrafoSize &&
                 log2TrafoSize > Log2MinTrafoSize &&
                 trafoDepth < maxDepth &&
                 !intraSplitFlag
flagPattern = (codeLuma << 3) + (codeCb << 2) + (codeCr << 1) + codeSplitTrans
```

The values of codeLuma, codeCb, codeCr and codeSplitTrans specify respectivley if an associated flag of cbf_luma, cbf_cb, cbf_cr, and split_transform_flag is included in the syntax element cbp_and_split_transform.

In Table 7-13, Luma, Cb, Cr and Split denote the values of cbf_luma[x0][y0][trafoDepth], cbf_cb[x0][y0][trafoDepth], cbf_cr[x0][y0][trafoDepth] and split_transform_flag[x0][y0][trafoDepth] respectively. Luma1, Luma2 and Luma3 denote respectively the values of cbf_luma[x1][y1][trafoDepth], cbf_luma[x2][y2][trafoDepth] and cbf_luma[x3][y3][trafoDepth]. The values of x1, y1, x2, y2, x3 and y3 are derived as follows.

If InterTUSplitDirection is equal to 2,

$$x1 = x0 + ((1 << log2TrafoWidth) >> 1) \quad (7\text{-}31)$$

$$y1 = y0 \quad (7\text{-}32)$$

$$x2 = x0 \quad (7\text{-}33)$$

$$y2 = y0 + ((1 << log2TrafoHeight) >> 1) \quad (7\text{-}34)$$

$$x3 = x1 \quad (7\text{-}35)$$

$$y3 = y2 \quad (7\text{-}36)$$

Otherwise (InterTUSplitDirection is equal to 0 or 1),

$$x1 = x0 + ((1 << log2TrafoWidth) >> 2) * InterTUSplitDirection \quad (7\text{-}37)$$

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000595

y1 = y0 + ((1 << log2TrafoHeight) >> 2) * (1 – InterTUSplitDirection) (7-38)

x2 = x1 + ((1 << log2TrafoWidth) >> 2) * InterTUSplitDirection (7-39)

y2 = y1 + ((1 << log2TrafoHeight) >> 2) * (1 – InterTUSplitDirection) (7-40)

x3 = x2 + ((1 << log2TrafoWidth) >> 2) * InterTUSplitDirection (7-41)

y3 = y2 + ((1 << log2TrafoHeight) >> 2) * (1 – InterTUSplitDirection) (7-42)

Table 7-15 – Specification of cbp_and_split_transform

flagPattern	cbp_and_split_transform
8	(Luma<<3) + (Luma1<<2) + (Luma2<<1) + Luma3 - 1
9	(Luma<<1) + Split
10	(Luma<<1) + Cr
11	(Luma<<2) + (Cr<<1) + Split
12	(Luma<<1) + Cb
13	(Luma<<2) + (Cb<<1) + Split
14	(Luma<<2) + (Cb<<1) + Cr
15	(Luma<<2) + ((Cb|Cr)<<1) + Split

7.4.9 Transform coefficient semantics

The transform coefficient levels are parsed into the arrays transCoeffLevel[x0][y0][trafoDepth][cIdx][n]. The array indices x0, y0 specify the location (x0, y0) of the top-left luma sample of the considered transform block relative to the top-left luma sample of the picture. The array index trafoDepth specifies the current subdivision level of a coding unit into blocks for the purpose of transform coding. trafoDepth is equal to 0 for blocks that correspond to coding units. The array index cIdx specifies an indicator for the colour component; it is equal to 0 for luma, equal to 1 for Cb, and equal to 2 for Cr. The array index n specifies the scanning position of the transform coefficient levels.

When PredMode is equal to MODE_INTRA, different scanning orders are used. The array ScanType[log2TrafoSize – 2][IntraPredMode], specifying the scanning order for various luma transform block sizes and intra prediction modes, is derived as specified in Table 7-16.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000596

Table 7-16 – Specification of ScanType[log2TrafoSize – 2][IntraPredMode]

IntraPredMode	log2TrafoSize – 2			
	0	**1**	**2**	**3**
0	1	1	0	0
1	2	2	0	0
2-3	0	0	0	0
4-5	1	1	0	0
6	0	0	0	0
7-8	2	2	0	0
9-10	0	0	0	0
11-12	1	1	0	0
13-14	0	0	0	0
15-16	2	2	0	0
17-19	0	0	0	0
20-23	1	1	0	0
24-27	0	0	0	0
28-31	2	2	0	0
32-33	0	0	0	0
34	0	0	0	0
35	0	0	0	0

cu_qp_delta can change the value of QP_Y for a quantization group of coding units where the quantization group of coding units is specified as follows:

– If a coding unit with the split_coding_unit_flag[x0][y0] equal to 0 and the log2CUSize is greater than or equal to log2MinCUDQPSize, the quantization group includes this coding unit only.

– Otherwise, if a coding unit with the split_coding_unit_flag[x0][y0] equal to 1 and the log2CUSize is equal to log2MinCUDQPSize, the quantization group includes all coding units split from this coding unit.

The decoded value of cu_qp_delta shall be in the range of –(26+ $QpBdOffset_Y$ / 2) to +(25+ $QpBdOffset_Y$ / 2), inclusive. cu_qp_delta shall be inferred to be equal to 0 when it is not present for any quantization group of coding units.

The value of QP_Y is derived as

$$QP_Y = (((QP_{Y,PREV} + cu\_qp\_delta + 52 + 2*QpBdOffset_Y) \% (52 + QpBdOffset_Y)) - QpBdOffset_Y \quad (7\text{-}21)$$

where $QP_{Y,PREV}$ is the luma quantization parameter, QP_Y, of the left neighbor quantization group of coding units in the current slice. If the left neighbor quantization group in the current slice is not available, $QP_{Y,PREV}$ is the luma quantization parameter, of the previous quantization group in decoding order in the current slice. For the first quantization group of coding units in the slice $QP_{Y,PREV}$ is initially set equal to $SliceQP_Y$ at the start of each slice.

The value of QP'_Y is derived as

$$QP'_Y = QP_Y + QpBdOffset_Y \quad (7\text{-}22)$$

7.4.10 Residual coding CABAC semantics

The array ScanOrder[log2BlockWidth – 2][log2BlockHeight – 2][scanIdx][pos][comp] specifies the mapping of the scan position pos, ranging from 0 to ((1 << log2BlockWidth) * (1 << log2BlockHeight)) – 1, to horizontal and vertical components of the scan-order matrix. The array index scanIdx equal to 0 specifies a zig-zag scan, scanIdx equal to 1 specifies a horizontal scan, scanIdx equal to 2 specifies a vertical scan, and scanIdx equal to 3 specifies an up-right diagonal scan. The array index comp equal to 0 specifies the horizontal component and the array index comp equal to 1 specifies the vertical component. The array ScanOrder is derived as follows.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000597

The scanning order array initialisation process as specified in 6.6 is invoked with 1 << log2TrafoWidth and 1 << log2TrafoHeight as input and the output is assigned to ScanOrder[log2TrafoWidth – 2][log2TrafoHeight – 2].

last_significant_coeff_x, last_significant_coeff_y specify the column and the row position of the last significant coefficient in scanning order within a transform block. The values of last_significant_coeff_x and last_significant_coeff_y shall be in the range from 0 to (1 << log2TrafoSize) – 1, inclusive.

When scanIdx is equal to 2, the coordinates are swapped as follows.

```
temp                   = last_significant_coeff_x
last_significant_coeff_x = last_significant_coeff_y          (7-23)
last_significant_coeff_y = temp
```

significant_coeff_flag[xC][yC] specifies for the transform coefficient position (xC, yC) within the current transform block whether the corresponding transform coefficient level at location (xC, yC) is non-zero as follows.

– If significant_coeff_flag[xC][yC] is equal to 0, the transform coefficient level at location (xC, yC) is set equal to 0;

– Otherwise (significant_coeff_flag[xC][yC] is equal to 1), the transform coefficient level at location (xC, yC) has a non-zero value.

The significant_coeff_flag[last_significant_coeff_x][last_significant_coeff_y] at the last significant location (last_significant_coeff_x, last_significant_coeff_y) in scan order is inferred to be equal to 1.

When significant_coeff_flag[xC][yC] is not present, it is inferred to be equal to 0.

coeff_abs_level_greater1_flag[n] specifies for the scanning position n whether there are transform coefficient levels greater than 1.

When coeff_abs_level_greater1_flag[n] is not present, it is inferred to be equal to 0.

coeff_abs_level_greater2_flag[n] specifies for the scanning position n whether there are transform coefficient levels greater than 2.

When coeff_abs_level_greater2_flag[n] is not present, it is inferred to be equal to 0.

coeff_abs_level_minus3[n] is the absolute value of a transform coefficient level minus 3 at the scanning position n. The value of coeff_abs_level_minus3 is constrained by the limits in subclause XX.

When coeff_abs_level_minus3[n] is not present, it is inferred as follows.

– If coeff_abs_level_greater1_flag[n] is equal to 0, coeff_abs_level_minus3[n] is inferred to be equal to –2.

– Otherwise (coeff_abs_level_greater1_flag[n] is equal to 1), coeff_abs_level_minus3[n] is inferred to be equal to –1.

coeff_sign_flag[n] specifies the sign of a transform coefficient level for the scanning position n as follows.

– If coeff_sign_flag[n] is equal to 0, the corresponding transform coefficient level has a positive value.

– Otherwise (coeff_sign_flag[n] is equal to 1), the corresponding transform coefficient level has a negative value.

When coeff_sign_flag[n] is not present, it is inferred to be equal to 0.

7.4.11 Residual coding CAVLC semantics

The following CAVLC syntax elements are used to decode a block of nMax transform coefficients.

last_pos_level_one specifies the position of the last non-zero valued transform coefficient in the scan and whether the absolute value of the transform coefficient is larger than 1.

– The position of the last non-zero transform coefficient in scan order is set equal to last_pos_level_one%nMax.

– If last_pos_level_one is less than nMax, the absolute value of the transform coefficient is equal to 1.

– Otherwise (last_pos_level_one is not less than nMax), the absolute value of the transform coeficient is larger than 1.

level_minus2_and_sign specifies the value of a transform coefficient that has an absolute value that is larger than one as follows.

– The absolute value of the transform coefficient is equal to (level_minus2_and_sign>>1) + 2.

– If (level_minus2_and_sign & 1) is equal to 0, the transform coefficient has a positive value.

– Otherwise ((level_minus2_and_sign & 1) is equal to 1), the transform coefficient has a negative value.

sign_flag specifies the sign of a non-zero valued transform coefficient as follows.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000598

– If sign_flag is equal to 0, the transform coefficient has a positive value.

– Otherwise (sign_flag is equal to 1), the transform coefficient has a negative value.

run_level_one specifies the number of consecutive transform coefficients in the scan with zero value before a non-zero valued transform coefficients and whether the absolute value of the non-zero valued transform coefficient is larger than 1.

– Given the scan position n of the current non-zero valued transform coefficient, the number of consecutive transform coefficients with zero value before the next non-zero valued transform coeficient is set equal to run_level_one%n.

– If run_level_one is less than n, the absolute value of the transform coefficient is equal to 1.

– Otherwise (run_level_one is not less than n), the absolute value of the next non-zero valued transform coefficient is larger than 1.

level specifies the absolute value of a non-zero valued transform coefficient.

8 Decoding process

Outputs of this process are decoded samples of the current picture (sometimes referred to by the variable CurrPic).

Depending on the value of chroma_format_idc, the number of sample arrays of the current picture is as follows.

– If chroma_format_idc is equal to 0, the current picture consists of 1 sample array S_L.

– Otherwise (chroma_format_idc is not equal to 0), the current picture consists of 3 sample arrays S_L, S_{Cb}, S_{Cr}.

This clause describes the decoding process, given syntax elements and upper-case variables from clause 7.

The decoding process is specified such that all decoders shall produce numerically identical results. Any decoding process that produces identical results to the process described here conforms to the decoding process requirements of this Recommendation | International Standard.

Each picture referred to in this clause is a complete primary coded picture or part of a primary coded picture. Each slice referred to in this clause is a slice of a primary coded picture. Each slice data partition referred to in this clause is a slice data partition of a primary coded picture.

Depending on the value of separate_colour_plane_flag, the decoding process is structured as follows.

– If separate_colour_plane_flag is equal to 0, the decoding process is invoked a single time with the current picture being the output.

– Otherwise (separate_colour_plane_flag is equal to 1), the decoding process is invoked three times. Inputs to the decoding process are all NAL units of the primary coded picture with identical value of colour_plane_id. The decoding process of NAL units with a particular value of colour_plane_id is specified as if only a coded video sequence with monochrome colour format with that particular value of colour_plane_id would be present in the bitstream. The output of each of the three decoding processes is assigned to the 3 sample arrays of the current picture with the NAL units with colour_plane_id equal to 0 being assigned to S_L, the NAL units with colour_plane_id equal to 1 being assigned to S_{Cb}, and the NAL units with colour_plane_id equal to 2 being assigned to S_{Cr}.

NOTE – The variable ChromaArrayType is derived as 0 when separate_colour_plane_flag is equal to 1 and chroma_format_idc is equal to 3. In the decoding process, the value of this variable is evaluated resulting in operations identical to that of monochrome pictures with chroma_format_idc being equal to 0.

An overview of the decoding process is given as follows:

1. The decoding of NAL units is specified in subclause 8.1.
2. The processes in subclause 8.2 specify decoding processes using syntax elements in the slice layer and above:
   – Variables and functions relating to picture order count are derived in subclause 8.2.1. (only needed to be invoked for one slice of a picture)
   – [Ed. (TW): clarify the following paragraph] When the frame_num of the current picture is not equal to PrevRefFrameNum and is not equal to (PrevRefFrameNum + 1) % MaxFrameNum, the decoding process for gaps in frame_num is performed according to subclause 8.2.3.2 prior to the decoding of any slices of the current picture.
   – At the beginning of the decoding process for each P, SP, or B slice, the decoding process for reference picture lists construction specified in subclause 8.2.2 is invoked for derivation of reference picture list 0 (RefPicList0), and when decoding a B slice, reference picture list 1 (RefPicList1).

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000599

– When the current picture is a reference picture and after all slices of the current picture have been decoded, the decoded reference picture marking process in subclause 8.2.2.4 specifies how the current picture is used in the decoding process of inter prediction in later decoded pictures.

3. The processes in subclauses 8.3, ABC specify decoding processes using syntax elements in the treeblock layer and above.

[Ed. (TW): insert corresponding text]

8.1 NAL unit decoding process

Inputs to this process are NAL units.

Outputs of this process are the RBSP syntax structures encapsulated within the NAL units.

The decoding process for each NAL unit extracts the RBSP syntax structure from the NAL unit and then operates the decoding processes specified for the RBSP syntax structure in the NAL unit as follows.

Subclause 8.2 describes the decoding process for NAL units with nal_unit_type equal to 1 and 5.

NAL units with nal_unit_type equal to 7 and 8 contain sequence parameter sets and picture parameter sets, respectively. Picture parameter sets are used in the decoding processes of other NAL units as determined by reference to a picture parameter set within the slice headers of each picture. Sequence parameter sets are used in the decoding processes of other NAL units as determined by reference to a sequence parameter set within the picture parameter sets of each sequence.

8.2 Slice decoding process

8.2.1 Decoding process for picture order count

[Ed. (TW): clarify the following paragraphs]

8.2.1.1 Decoding process for picture order count type 0

8.2.1.2 Decoding process for picture order count type 1

8.2.1.3 Decoding process for picture order count type 2

8.2.2 Decoding process for reference picture lists construction

[Ed. (TW): clarify the following paragraphs]

This process is invoked at the beginning of the decoding process for each P, SP, or B slice.

Decoded reference pictures are marked as "used for short-term reference" or "used for long-term reference" as specified by the bitstream and specified in subclause 8.2.2.4. Short-term reference pictures are identified by the value of frame_num. Long-term reference pictures are assigned a long-term picture index as specified by the bitstream and specified in subclause 8.2.2.4.

Subclause 8.2.2.1 is invoked to specify

– the assignment of variables FrameNum, FrameNumWrap, and PicNum to each of the short-term reference pictures, and

– the assignment of variable LongTermPicNum to each of the long-term reference pictures.

Reference pictures are addressed through reference indices as specified in subclause **Error! Reference source not found.**. A reference index is an index into a reference picture list. When decoding a P or SP slice, there is a single reference picture list RefPicList0. When decoding a B slice, there is a second independent reference picture list RefPicList1 in addition to RefPicList0.

At the beginning of the decoding process for each slice, reference picture list RefPicList0, and for B slices RefPicList1, are derived as specified by the following ordered steps:

1. An initial reference picture list RefPicList0 and for B slices RefPicList1 are derived as specified in subclause 8.2.2.2.

2. When ref_pic_list_modification_flag_l0 is equal to 1 or, when decoding a B slice, ref_pic_list_modification_flag_l1 is equal to 1, the initial reference picture list RefPicList0 and, for B slices, RefPicList1 are modified as specified in subclause 8.2.2.3.

   NOTE – The modification process for reference picture lists specified in subclause 8.2.2.3 allows the contents of RefPicList0 and for B slices RefPicList1 to be modified in a flexible fashion. In particular, it is possible for a picture

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000600

that is currently marked "used for reference" to be inserted into RefPicList0 and for B slices RefPicList1 even when the picture is not in the initial reference picture list derived as specified in subclause 8.2.2.2.

The number of entries in the modified reference picture list RefPicList0 is num_ref_idx_l0_active_minus1 + 1, and for B slices the number of entries in the modified reference picture list RefPicList1 is num_ref_idx_l1_active_minus1 + 1. A reference picture may appear at more than one index in the modified reference picture lists RefPicList0 or RefPicList1.

8.2.2.1 Decoding process for picture numbers

This process is invoked when the decoding process for reference picture lists construction specified in subclause 8.2.2, the decoded reference picture marking process specified in subclause 8.2.2.4, or the decoding process for gaps in frame_num specified in subclause 8.2.3.2 is invoked.

The variables FrameNum, FrameNumWrap, PicNum, LongTermFrameIdx, and LongTermPicNum are used for the initialisation process for reference picture lists in subclause 8.2.2.2, the modification process for reference picture lists in subclause 8.2.2.3, the decoded reference picture marking process in subclause 8.2.2.4, and the decoding process for gaps in frame_num in subclause 8.2.3.2.

To each short-term reference picture the variables FrameNum and FrameNumWrap are assigned as follows. First, FrameNum is set equal to the syntax element frame_num that has been decoded in the slice header(s) of the corresponding short-term reference picture. Then the variable FrameNumWrap is derived as

```
if( FrameNum > frame_num )
    FrameNumWrap = FrameNum – MaxFrameNum                    (8-1)
else
    FrameNumWrap = FrameNum
```

where the value of frame_num used in Equation 8-1 is the frame_num in the slice header(s) for the current picture.

Each long-term reference picture has an associated value of LongTermFrameIdx (that was assigned to it as specified in subclause 8.2.2.4).

To each short-term reference picture a variable PicNum is assigned, and to each long-term reference picture a variable LongTermPicNum is assigned.

$$PicNum = FrameNumWrap \tag{8-2}$$

$$LongTermPicNum = LongTermFrameIdx \tag{8-3}$$

8.2.2.2 Initialisation process for reference picture lists

This initialisation process is invoked when decoding a P or B slice header.

RefPicList0 and RefPicList1 have initial entries as specified in subclauses 8.2.2.2.1 through 8.2.2.2.2.

When the number of entries in the initial RefPicList0 or RefPicList1 produced as specified in subclauses 8.2.2.2.1 through 8.2.2.2.2 is greater than num_ref_idx_l0_active_minus1 + 1 or num_ref_idx_l1_active_minus1 + 1, respectively, the extra entries past position num_ref_idx_l0_active_minus1 or num_ref_idx_l1_active_minus1 are discarded from the initial reference picture list.

When the number of entries in the initial RefPicList0 or RefPicList1 produced as specified in subclauses 8.2.2.2.1 through 8.2.2.2.2 is less than num_ref_idx_l0_active_minus1 + 1 or num_ref_idx_l1_active_minus1 + 1, respectively, the remaining entries in the initial reference picture list are set equal to "no reference picture".

8.2.2.2.1 Initialisation process for the reference picture list for P slices

This initialisation process is invoked when decoding a P slice in a coded picture.

When this process is invoked, there shall be at least one reference picture that is currently marked as "used for reference" (i.e., as "used for short-term reference" or "used for long-term reference") and is not marked as "non-existing". Pictures with higher values of temporal_id than the current picture cannot be used for reference, and are not included in the reference picture list.

The reference picture list RefPicList0 is ordered so that short-term reference pictures have lower indices than long-term reference pictures.

The short-term reference pictures are ordered starting with the picture with the highest PicNum value and proceeding through in descending order to the picture with the lowest PicNum value, excluding any picture with a temporal_id value higher than that of the current picture.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000601

The long-term reference pictures are ordered starting with the picture with the lowest LongTermPicNum value and proceeding through in ascending order to the picture with the highest LongTermPicNum value, excluding any picture with a temporal_id value higher than that of the current picture.

For example, when three reference pictures are marked as "used for short-term reference" with PicNum equal to 300, 302, and 303 and two reference pictures are marked as "used for long-term reference" with LongTermPicNum equal to 0 and 3, the initial index order is:

– RefPicList0[0] is set equal to the short-term reference picture with PicNum = 303,

– RefPicList0[1] is set equal to the short-term reference picture with PicNum = 302,

– RefPicList0[2] is set equal to the short-term reference picture with PicNum = 300,

– RefPicList0[3] is set equal to the long-term reference picture with LongTermPicNum = 0,

– RefPicList0[4] is set equal to the long-term reference picture with LongTermPicNum = 3.

8.2.2.2.2 Initialisation process for reference picture lists for B slices

This initialisation process is invoked when decoding a B slice in a coded picture.

For purposes of the formation of the reference picture lists RefPicList0 and RefPicList1 the term reference entry refers in the following to decoded reference pictures.

When this process is invoked, there shall be at least one reference entry that is currently marked as "used for reference" (i.e., as "used for short-term reference" or "used for long-term reference") and is not marked as "non-existing". Pictures with higher values of temporal_id than the current picture cannot be used for reference, and are not included in either RefPicList0 or RefPicList1.

For B slices, the order of short-term reference entries in the reference picture lists RefPicList0 and RefPicList1 depends on output order, as given by PicOrderCnt( ). When pic_order_cnt_type is equal to 0, reference pictures that are marked as "non-existing" as specified in subclause 8.2.3.2 are not included in either RefPicList0 or RefPicList1.

NOTE 1 – When gaps_in_frame_num_value_allowed_flag is equal to 1, encoders should use reference picture list modification to ensure proper operation of the decoding process (particularly when pic_order_cnt_type is equal to 0, in which case PicOrderCnt( ) is not inferred for "non-existing" pictures).

The reference picture list RefPicList0 is ordered such that short-term reference entries have lower indices than long-term reference entries. It is ordered as follows:

1. Let entryShortTerm be a variable ranging over all reference entries that are currently marked as "used for short-term reference" and which have a value of temporal_id equal to or lower than the temporal_id of the current picture. When some values of entryShortTerm are present having PicOrderCnt(entryShortTerm) less than PicOrderCnt(CurrPic), these values of entryShortTerm are placed at the beginning of refPicList0 in descending order of PicOrderCnt(entryShortTerm). All of the remaining values of entryShortTerm (when present) are then appended to refPicList0 in ascending order of PicOrderCnt(entryShortTerm).
2. The long-term reference entries which have a value of temporal_id equal to or lower than the temporal_id of the current picture are ordered starting with the long-term reference entry that has the lowest LongTermPicNum value and proceeding through in ascending order to the long-term reference entry that has the highest LongTermPicNum value.

The reference picture list RefPicList1 is ordered so that short-term reference entries have lower indices than long-term reference entries. It is ordered as follows:

1. Let entryShortTerm be a variable ranging over all reference entries that are currently marked as "used for short-term reference" and which have a value of temporal_id equal to or lower than the temporal_id of the current picture. When some values of entryShortTerm are present having PicOrderCnt(entryShortTerm) greater than PicOrderCnt(CurrPic), these values of entryShortTerm are placed at the beginning of refPicList1 in ascending order of PicOrderCnt(entryShortTerm). All of the remaining values of entryShortTerm (when present) are then appended to refPicList1 in descending order of PicOrderCnt(entryShortTerm).
2. Long-term reference entries which have a value of temporal_id equal to or lower than the temporal_id of the current picture are ordered starting with the long-term reference picture that has the lowest LongTermPicNum value and proceeding through in ascending order to the long-term reference entry that has the highest LongTermPicNum value.
3. When the reference picture list RefPicList1 has more than one entry and RefPicList1 is identical to the reference picture list RefPicList0, the first two entries RefPicList1[0] and RefPicList1[1] are switched.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000602

8.2.2.3 Modification process for reference picture lists

After the invocation of this process, there shall be no reference pictures with greater temporal_id than the current slice included in the output RefPicList0 or RefPicList1.

When ref_pic_list_modification_flag_l0 is equal to 1, the following applies:

1. Let refIdxL0 be an index into the reference picture list RefPicList0. It is initially set equal to 0.
2. The corresponding syntax elements modification_of_pic_nums_idc are processed in the order they occur in the bitstream. For each of these syntax elements, the following applies.
   - If modification_of_pic_nums_idc is equal to 0 or equal to 1, the process specified in subclause 8.2.2.3.1 is invoked with refIdxL0 as input, and the output is assigned to refIdxL0.
   - Otherwise, if modification_of_pic_nums_idc is equal to 2, the process specified in subclause 8.2.2.3.2 is invoked with refIdxL0 as input, and the output is assigned to refIdxL0.
   - Otherwise (modification_of_pic_nums_idc is equal to 3), the modification process for reference picture list RefPicList0 is finished.

When the current slice is a B slice and ref_pic_list_modification_flag_l1 is equal to 1, the following applies:

1. Let refIdxL1 be an index into the reference picture list RefPicList1. It is initially set equal to 0.
2. The corresponding syntax elements modification_of_pic_nums_idc are processed in the order they occur in the bitstream. For each of these syntax elements, the following applies.
   - If modification_of_pic_nums_idc is equal to 0 or equal to 1, the process specified in subclause 8.2.2.3.1 is invoked with refIdxL1 as input, and the output is assigned to refIdxL1.
   - Otherwise, if modification_of_pic_nums_idc is equal to 2, the process specified in subclause 8.2.2.3.2 is invoked with refIdxL1 as input, and the output is assigned to refIdxL1.
   - Otherwise (modification_of_pic_nums_idc is equal to 3), the modification process for reference picture list RefPicList1 is finished.

8.2.2.3.1 Modification process of reference picture lists for short-term reference pictures

Input to this process is an index refIdxLX (with X being 0 or 1).

Output of this process is an incremented index refIdxLX.

The variable picNumLXNoWrap is derived as follows.

- If modification_of_pic_nums_idc is equal to 0,

```
if( picNumLXPred – ( abs_diff_pic_num_minus1 + 1 ) < 0 )
    picNumLXNoWrap = picNumLXPred – ( abs_diff_pic_num_minus1 + 1 ) + MaxPicNum        (8-4)
else
    picNumLXNoWrap = picNumLXPred – ( abs_diff_pic_num_minus1 + 1 )
```

- Otherwise (modification_of_pic_nums_idc is equal to 1),

```
if( picNumLXPred + ( abs_diff_pic_num_minus1 + 1 ) >= MaxPicNum )
    picNumLXNoWrap = picNumLXPred + ( abs_diff_pic_num_minus1 + 1 ) – MaxPicNum        (8-5)
else
    picNumLXNoWrap = picNumLXPred + ( abs_diff_pic_num_minus1 + 1 )
```

picNumLXPred is the prediction value for the variable picNumLXNoWrap. When the process specified in this subclause is invoked the first time for a slice (that is, for the first occurrence of modification_of_pic_nums_idc equal to 0 or 1 in the ref_pic_list_modification( ) syntax), picNumL0Pred and picNumL1Pred are initially set equal to CurrPicNum. After each assignment of picNumLXNoWrap, the value of picNumLXNoWrap is assigned to picNumLXPred.

The variable picNumLX is derived as specified by the following pseudo-code:

```
if( picNumLXNoWrap > CurrPicNum )
    picNumLX = picNumLXNoWrap – MaxPicNum                                             (8-6)
else
    picNumLX = picNumLXNoWrap
```


Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000603

picNumLX shall be equal to the PicNum of a reference picture that is marked as "used for short-term reference" and shall not be equal to the PicNum of a short-term reference picture that is marked as "non-existing". The short-term reference picture with PicNum equal to picNumLX shall not have greater temporal_id than the current slice.

The following procedure is conducted to place the picture with short-term picture number picNumLX into the index position refIdxLX, shift the position of any other remaining pictures to later in the list, and increment the value of refIdxLX.

```
for( cIdx = num_ref_idx_lX_active_minus1 + 1; cIdx > refIdxLX; cIdx− − )
    RefPicListX[ cIdx ] = RefPicListX[ cIdx − 1]
RefPicListX[ refIdxLX++ ] = short-term reference picture with PicNum equal to picNumLX
nIdx = refIdxLX
for( cIdx = refIdxLX; cIdx <= num_ref_idx_lX_active_minus1 + 1; cIdx++ )                    (8-7)
    if( PicNumF( RefPicListX[ cIdx ] ) != picNumLX )
        RefPicListX[ nIdx++ ] = RefPicListX[ cIdx ]
```

where the function PicNumF(RefPicListX[cIdx]) is derived as follows.

– If the picture RefPicListX[cIdx] is marked as "used for short-term reference", PicNumF(RefPicListX[cIdx]) is the PicNum of the picture RefPicListX[cIdx].

– Otherwise (the picture RefPicListX[cIdx] is not marked as "used for short-term reference"), PicNumF(RefPicListX[cIdx]) is equal to MaxPicNum.

NOTE 1 – A value of MaxPicNum can never be equal to picNumLX.

NOTE 2 – Within this pseudo-code procedure, the length of the list RefPicListX is temporarily made one element longer than the length needed for the final list. After the execution of this procedure, only elements 0 through num_ref_idx_lX_active_minus1 of the list need to be retained.

8.2.2.3.2 Modification process of reference picture lists for long-term reference pictures

Input to this process is an index refIdxLX (with X being 0 or 1).

Output of this process is an incremented index refIdxLX.

The following procedure is conducted to place the picture with long-term picture number long_term_pic_num into the index position refIdxLX, shift the position of any other remaining pictures to later in the list, and increment the value of refIdxLX.

```
for( cIdx = num_ref_idx_lX_active_minus1 + 1; cIdx > refIdxLX; cIdx− − )
    RefPicListX[ cIdx ] = RefPicListX[ cIdx − 1]
RefPicListX[ refIdxLX++ ] = long-term reference picture with LongTermPicNum equal to long_term_pic_num
nIdx = refIdxLX
for( cIdx = refIdxLX; cIdx <= num_ref_idx_lX_active_minus1 + 1; cIdx++ )                    (8-8)
    if( LongTermPicNumF( RefPicListX[ cIdx ] ) != long_term_pic_num )
        RefPicListX[ nIdx++ ] = RefPicListX[ cIdx ]
```

where the long-term reference picture with LongTermPicNum equal to long_term_pic_num shall not have greater temporal_id than the current slice, and the function LongTermPicNumF(RefPicListX[cIdx]) is derived as follows.

– If the picture RefPicListX[cIdx] is marked as "used for long-term reference", LongTermPicNumF(RefPicListX[cIdx]) is the LongTermPicNum of the picture RefPicListX[cIdx].

– Otherwise (the picture RefPicListX[cIdx] is not marked as "used for long-term reference"), LongTermPicNumF(RefPicListX[cIdx]) is equal to 2 * (MaxLongTermFrameIdx + 1).

NOTE 1 – A value of 2 * (MaxLongTermFrameIdx + 1) can never be equal to long_term_pic_num.

NOTE 2 – Within this pseudo-code procedure, the length of the list RefPicListX is temporarily made one element longer than the length needed for the final list. After the execution of this procedure, only elements 0 through num_ref_idx_lX_active_minus1 of the list need to be retained.

8.2.2.4 Mapping process for reference picture lists combination in B slices

[Ed.: (WJ) needs to be checked once again. Try to find better way to represent]

This initialisation process is invoked when decoding a B slice header.

Input to this process are the reference picture list RefPicListX and num_ref_idx_lX_active_minus1 with X being 0 or 1.

Outputs of this process are arrays PredLCToPredLx and RefIdxLCToRefIdxLx.

When the current slice is a B slice and ref_pic_list_modification_flag_lc is equal to 0, the following ordered steps apply:

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000604

1. Let refIdxL0 and refIdxL1 be indices into the reference picture lists RefPicListL0 and RefPicListL1. They are initially set equal to 0.
2. Let refIdxLC be an index into PredLCToPredLx and RefIdxLCToRefIdxLx. It is initially set equal to 0.
3. The following process is repeated until refIdxL0 and refIdxL1 are both greater than num_ref_idx_l0_active_minus1 and num_ref_idx_l1_active_minus1, respectively:
   - If refIdxL0 is less than or equal to num_ref_idx_l0_active_minus1,
     - If the entry RefPicListL0[refIdxL0] is the first occurance of the reference picture,

       PredLCToPredLx[refIdxLC] = Pred_L0, (8-9)
       RefIdxLCToRefIdxLx[refIdxLC++] = refIdxL0.

     - refIdxL0++.
   - If refIdxL1 is less than or equal to num_ref_idx_l1_active_minus1 and ref_pic_list_combination_flag equal to 1,
     - If the entry RefPicListL1[refIdxL1] is the first occurance of the reference picture,

       PredLCToPredLx[refIdxLC] = Pred_L1, (8-10)
       RefIdxLCToRefIdxLx[refIdxLC++] = refIdxL1.

     - refIdxL1++.

When the current slice is a B slice and ref_pic_list_modification_flag_lc is equal to 1, the following ordered steps apply:

1. Let refIdxLC be an index into the reference picture list PredLCToPredLx and RefIdxLCToRefIdxLx. It is initially set equal to 0.
2. The corresponding syntax elements pic_from_list_0_flag and ref_idx_list_curr are processed in the order they occur in the bitstream. For each of these syntax elements pairs, the following applies.
   - If pic_from_list_0_flag is equal to 1,

     PredLCToPredLx[refIdxLC] = Pred_L0, (8-11)

   - Otherwise,

     PredLCToPredLx[refIdxLC] = Pred_L1 (8-12)

   - RefIdxLCToRefIdxLx[refIdxLC++] = ref_idx_list_curr

When refIdxLC is greater than num_com_ref_list_active_minus1+ 1, the extra entries past position num_com_ref_list_active_minus1 are discarded from PredLCToPredLx and RefIdxLCToRefIdxLx.

When refIdxLC is less than num_com_ref_list_active_minus1 + 1, the remaining entries in PredLCToPredLx and RefIdxLCToRefIdxLx are set equal to Pred_L0 and 0, respectively.

8.2.3 Decoded reference picture marking process

This process is invoked for decoded pictures when nal_ref_idcnal_ref_flag is not equal to 0.

NOTE 1 – The decoding process for gaps in frame_num that is specified in subclause 8.2.3.2 may also be invoked when nal_ref_idcnal_ref_flag is equal to 0, as specified in clause 0.

A decoded picture with nal_ref_idcnal_ref_flag not equal to 0, referred to as a reference picture, is marked as "used for short-term reference" or "used for long-term reference". A picture that is marked as "used for short-term reference" is identified by its FrameNum. A picture that is marked as "used for long-term reference" is identified by its LongTermFrameIdx.

Pictures marked as "used for short-term reference" or as "used for long-term reference" can be used as a reference for inter prediction when decoding a picture until the picture is marked as "unused for reference".

A picture can be marked as "unused for reference" by the sliding window reference picture marking process, a first-in, first-out mechanism specified in subclause 8.2.3.3 or by the adaptive memory control reference picture marking process, a customised adaptive marking operation specified in subclause 8.2.3.4.

A short-term reference picture is identified for use in the decoding process by its variables FrameNum and FrameNumWrap and its picture number PicNum, and a long-term reference picture is identified for use in the decoding process by its long-

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000605

term picture number LongTermPicNum. When the current picture is not an IDR picture, subclause 8.2.2.1 is invoked to specify the assignment of the variables FrameNum, FrameNumWrap, PicNum and LongTermPicNum.

8.2.3.1 Sequence of operations for decoded reference picture marking process

Decoded reference picture marking proceeds in the following ordered steps:

4. All slices of the current picture are decoded.
5. Depending on whether the current picture is an IDR picture, the following applies.
   - If the current picture is an IDR picture, the following ordered steps are specified:
     a. All reference pictures are marked as "unused for reference"
     b. Depending on long_term_reference_flag, the following applies.
        - If long_term_reference_flag is equal to 0, the IDR picture is marked as "used for short-term reference" and MaxLongTermFrameIdx is set equal to "no long-term picture indices".
        - Otherwise (long_term_reference_flag is equal to 1), the IDR picture is marked as "used for long-term reference", the LongTermFrameIdx for the IDR picture is set equal to 0, and MaxLongTermFrameIdx is set equal to 0.
   - Otherwise (the current picture is not an IDR picture), the following applies.
     - If adaptive_ref_pic_marking_mode_flag is equal to 0, the process specified in subclause 8.2.3.3 is invoked.
     - Otherwise (adaptive_ref_pic_marking_mode_flag is equal to 1), the process specified in subclause 8.2.3.4 is invoked.
6. When the current picture is not an IDR picture and it was not marked as "used for long-term reference" by memory_management_control_operation equal to 6, it is marked as "used for short-term reference".

It is a requirement of bitstream conformance that, after marking the current decoded reference picture, the total number of pictures marked as "used for reference" shall not be greater than Max(max_num_ref_frames, 1).

8.2.3.2 Decoding process for gaps in frame_num

[Ed. (TW): insert text]

8.2.3.3 Sliding window decoded reference picture marking process

This process is invoked when adaptive_ref_pic_marking_mode_flag is equal to 0.

Depending on the properties of the current picture as specified below, the following applies.

1. Let numShortTerm be the total number of reference pictures that are marked as "used for short-term reference". Let numLongTerm be the total number of reference pictures that are marked as "used for long-term reference".
2. When numShortTerm + numLongTerm is equal to Max(max_num_ref_frames, 1), the condition that numShortTerm is greater than 0 shall be fulfilled, and the short-term reference picture that has the smallest value of FrameNumWrap is marked as "unused for reference".

8.2.3.4 Adaptive memory control decoded reference picture marking process

This process is invoked when adaptive_ref_pic_marking_mode_flag is equal to 1.

The memory_management_control_operation commands with values of 1 to 6 are processed in the order they occur in the bitstream after the current picture has been decoded. For each of these memory_management_control_operation commands, one of the processes specified in subclauses 8.2.3.4.1 to 8.2.3.4.6 is invoked depending on the value of memory_management_control_operation. The memory_management_control_operation command with value of 0 specifies the end of memory_management_control_operation commands.

8.2.3.4.1 Marking process of a short-term reference picture as "unused for reference"

This process is invoked when memory_management_control_operation is equal to 1.

Let picNumX be specified by

$$picNumX = CurrPicNum - (difference\_of\_pic\_nums\_minus1 + 1). \quad (8\text{-}13)$$

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000606

The value of picNumX is used to mark the corresponding short-term reference picture as "unused for reference".

8.2.3.4.2 Marking process of a long-term reference picture as "unused for reference"

This process is invoked when memory_management_control_operation is equal to 2.

The value of LongTermPicNum is used to mark the corresponding long-term reference picture as "unused for reference".

8.2.3.4.3 Assignment process of a LongTermFrameIdx to a short-term reference picture

This process is invoked when memory_management_control_operation is equal to 3.

Given the syntax element difference_of_pic_nums_minus1, the variable picNumX is obtained as specified in subclause 8.2.3.4.1. picNumX shall refer to a picture marked as "used for short-term reference" and not marked as "non-existing".

When LongTermFrameIdx equal to long_term_frame_idx is already assigned to a long-term reference picture, that picture is marked as "unused for reference".

The value of LongTermFrameIdx is used to mark the corresponding picture from "used for short-term reference" to "used for long-term reference".

8.2.3.4.4 Decoding process for MaxLongTermFrameIdx

This process is invoked when memory_management_control_operation is equal to 4.

All pictures for which LongTermFrameIdx is greater than max_long_term_frame_idx_plus1 – 1 and that are marked as "used for long-term reference" are marked as "unused for reference".

The variable MaxLongTermFrameIdx is derived as follows.

– If max_long_term_frame_idx_plus1 is equal to 0, MaxLongTermFrameIdx is set equal to "no long-term picture indices".

– Otherwise (max_long_term_frame_idx_plus1 is greater than 0), MaxLongTermFrameIdx is set equal to max_long_term_frame_idx_plus1 – 1.

NOTE – The memory_management_control_operation command equal to 4 can be used to mark long-term reference pictures as "unused for reference". The frequency of transmitting max_long_term_frame_idx_plus1 is not specified by this Recommendation | International Standard. However, the encoder should send a memory_management_control_operation command equal to 4 upon receiving an error message, such as an intra refresh request message.

8.2.3.4.5 Marking process of all reference pictures as "unused for reference" and setting MaxLongTermFrameIdx to "no long-term picture indices"

This process is invoked when memory_management_control_operation is equal to 5.

All reference pictures are marked as "unused for reference" and the variable MaxLongTermFrameIdx is set equal to "no long-term picture indices".

8.2.3.4.6 Process for assigning a long-term picture index to the current picture

This process is invoked when memory_management_control_operation is equal to 6.

When a variable LongTermFrameIdx equal to long_term_frame_idx is already assigned to a long-term reference picture, that picture is marked as "unused for reference".

The current picture is marked as "used for long-term reference" and assigned LongTermFrameIdx equal to long_term_frame_idx.

After marking the current decoded reference picture, the total number of pictures that are marked as "used for reference" shall not be greater than Max(max_num_ref_frames, 1).

NOTE – Under some circumstances, the above statement may impose a constraint on the order in which a memory_management_control_operation syntax element equal to 6 can appear in the decoded reference picture marking syntax relative to a memory_management_control_operation syntax element equal to 1, 2, 3, or 4.

8.3 Decoding process for coding units coded in intra prediction mode

Inputs to this process are:

– a luma location (xB, yB) specifying the top-left luma sample of the current coding unit relative to the top-left luma sample of the current picture,

– a variable log2CUSize specifying the size of the current coding unit.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000607

Output of this process is:

– a modified reconstructed picture before deblocking filtering.

A variable nS is set equal to (1 << log2CUSize).

Depending on pcm_flag and IntraSplitFlag, the decoding process for luma samples is specified as follows.

– If pcm_flag is equal to 1, the reconstucted picture is modified as follows:

$$recSamples_L[xB + i, yB + j] = pcm\_sample\_luma[(nS * j) + i] << (BitDepth_Y - PCMBitDepth_Y), \text{ with } i, j = 0..nS-1 \quad (8\text{-}13)$$

– Otherwise (pcm_flag is equal to 0), if IntraSplitFlag is equal to 0, the following ordered steps apply:

1. The derivation process for the intra prediction mode as specified in subclause 8.3.1 is invoked with the luma location (xB, yB) and the variable log2PUSize set equal to log2CUSize as well as IntraPredMode that is previously (in decoding order) derived for adjacent coding units as the input and the output is the variable IntraPredMode[xB][yB].
2. The decoding process for intra blocks as specified in subclause 8.3.3 is invoked with the luma location (xB, yB), the variable log2TrafoSize set equal to log2CUSize, the variable trafoDepth set equal to 0, the luma intra prediction mode IntraPredMode[xB][yB] and the variable cIdx set equal to 0 as the inputs and the output is a modified reconstructed picture before deblocking filtering.

– Otherwise (pcm_flag is equal to 0 and IntraSplitFlag is equal to 1), for the variable blkIdx proceeding over the values 0..3, the following ordered steps apply:

1. The variable xBS is set equal to xB + (nS >> 1) * (blkIdx % 2).
2. The variable yBS is set equal to yB + (nS >> 1) * (blkIdx / 2).
3. The derivation process for the intra prediction mode as specified in subclause 8.3.1 is invoked with the luma location (xBS, yBS) and the variable log2PUSize set equal to log2CUSize – 1 as well as IntraPredMode that is previously (in decoding order) derived for adjacent coding units as the input and the output is the variable IntraPredMode[xBS][yBS].
4. The decoding process for intra blocks as specified in subclause 8.3.3 is invoked with the luma location (xBS, yBS), the variable log2TrafoSize set equal to log2CUSize – 1, the variable trafoDepth set equal to 1, the luma intra prediction mode IntraPredMode[xBS][yBS] and the variable cIdx set equal to 0 as the inputs and the output is a modified reconstructed picture before deblocking filtering.

Depending on pcm_flag, the decoding process for chroma samples is specified as follows:

– If pcm_flag is equal to 1, the reconstucted picture is modified as follows:

$$recSamples_{Cb}[xB/2 + i, yB/2 + j] = pcm\_sample\_chroma[(nS/2 * j) + i] << (BitDepth_C - PCMBitDepth_C) \text{ with } i, j = 0..nS/2-1 \quad (8\text{-}13)$$

$$recSamples_{Cr}[xB/2 + i, yB/2 + j] = pcm\_sample\_chroma[(nS/2 * (j + nS)) + i] << (BitDepth_C - PCMBitDepth_C) \text{ with } i, j = 0..nS/2-1 \quad (8\text{-}13)$$

– Otherwise (pcm_flag is equal to 0), the following ordered steps apply:

1. The derivation process for the chroma intra prediction mode as specified in 8.3.2 is invoked with the luma location (xB, yB) as input and the output is the variable IntraPredModeC.
2. The decoding process for intra blocks as specified in subclause 8.3.3 is invoked with the chroma location (xB/2, yB/2), the variable log2TrafoSize set equal to log2CUSize-1, the variable trafoDepth set equal to 0, the chroma intra prediction mode IntraPredModeC, and the variable cIdx set equal to 1 as the inputs and the output is a modified reconstructed picture before deblocking filtering.
3. The decoding process for intra blocks as specified in subclause 8.3.3 is invoked with the chroma location (xB/2, yB/2), the variable log2TrafoSize set equal to log2CUSize-1, the variable trafoDepth set equal to 0, the chroma intra prediction mode IntraPredModeC, and the variable cIdx set equal to 2 as the inputs and the output is a modified reconstructed picture before deblocking filtering.

8.3.1 Derivation process for luma intra prediction mode

Inputs to this process are:

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000608

– a luma location (xB, yB) specifying the top-left luma sample of the current block relative to the top-left luma sample of the current picture,

– a variable log2PUSize specifying the size of the current prediction unit,

– variable arrays IntraPredMode (If available) that are previously (in decoding order) derived for adjacent coding units.

Output of this process is the variable IntraPredMode[xB][yB].

Table 8-5 specifies the value for the intra prediction mode and the associated names.

Table 8-1 – Specification of intra prediction mode and associated names

Intra prediction mode	Associated names
0	Intra_Planar
1	Intra_Vertical
2	Intra_Horizontal
3	Intra_DC
Otherwise (4..34)	Intra_Angular
35	Intra_FromLuma (used only for chroma)

Table 8-1 specifies the number of luma intra prediction modes intraPredModeNum depending on log2PUSize.

Table 8-2 – Specification of intraPredModeNum

log2PUSize	intraPredModeNum
2	18
3	35
4	35
5	35
6	4

Table 8-2 specifies the mapping table used for converting the number of intra prediction modes.

Table 8-3 – Specification of mapIntraPredMode4 and mapIntraPredMode10

value	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	
mapIntraPredMode4[value]	0	1	2	3	3	3	1	3	2	2	3	3	1	1	3	3	2	
mapIntraPredMode10[value]	0	1	2	3	4	5	6	7	8	9	3	3	3	3	3	3	3	
value	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**	**32**	**33**	**34**
mapIntraPredMode4[value]	2	3	3	3	3	1	1	1	3	3	3	3	2	2	2	2	3	3
mapIntraPredMode10[value]	3	3	4	5	5	1	1	6	6	7	4	8	8	2	2	9	9	3

IntraPredMode[xB][yB] labelled 0, 1, 2, .., 34 represents directions of predictions as illustrated in Figure 8-1.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000609

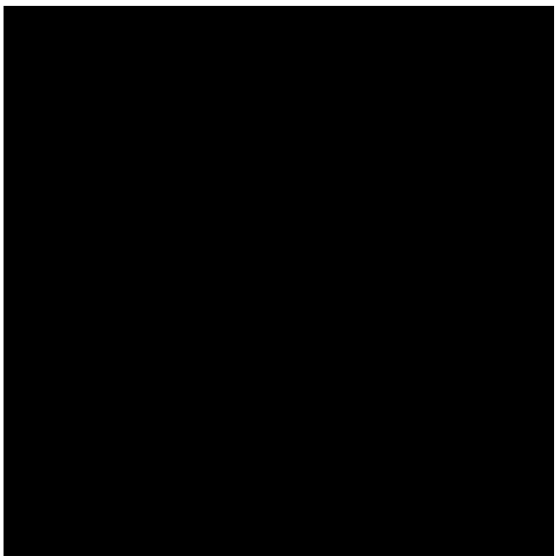


Figure 8-1 – Intra prediction mode directions (informative)

IntraPredMode[xB][yB] is derived as the following ordered steps. [Ed. (WJ): proponent suggests to move this part to the syntax since the other syntax elements utilize IntraPredMode. But it seems too complex to move all the following process to the syntax table. Maybe it's better to move this part to the semantics section or simply avoid the use of IntraPredMode to parse the syntax item]

1. The derivation process for neighbouring treeblocks specified in subclause XXX with (xB, yB) given as input and the output is assigned to tbAddrA and tbAddrB specifying the treeblock addresses of treeblocks covering (xBA, yBA) and (xBB, yBB) respectively where (xBA, yBA) is set equal to (xB-1, yB) and (xBB, yBB) is set equal to (xB, yB-1).
2. For N being either replaced A or B, the variables intraPredModeN are derived as follows.
   - If the treeblock with address tbAddrN is not available, intraPredModeN is set equal to -1
   - Otherwise, if the coding unit covering (xBN, yBN) is not coded as intra mode, intraPredModeN is set equal to Intra_DC,
   - Otherwise, intraPredModeN is set equal to IntraPredMode[xBN][yBN], where IntraPredMode is the variable array assigned to the coding unit covering the luma location (xBN, yBN).
3. For N being either replaced A or B, the variables candIntraPredModeN are derived as follows.
   - If intraPredModeN is greater than or equal to intraPredModeNum
     - If intraPredModeNum is equal to 4 then candIntraPredModeN is set equal to mapIntraPredMode4[intraPredModeN]
     - Otherwise candIntraPredModeN is set equal to mapIntraPredMode10[intraPredModeN].
   - Otherwise, candIntraPredModeN is set equal to intraPredModeN
4. The candModeList[x] is derived as follows:
   - If both candIntraPredModeN are not available, then the value 3 is assigned to candModeList[0] and the value 0 is assigned to candModeList[1]

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000610

- Otherwise, if only one candIntraPredModeN is available or if both candIntraPredModeN are the same, candModeList[x] is derived by applying the following procedure:
  1. candIntraPredModeN is assigned to candModeList[0]
  2. If candIntraPredModeN is not equal to 0, the value 0 is assigned to candModeList[1]
  3. Otherwise, the value 3 is assigned to candModeList[1]
- Otherwise, both candIntraPredModeN are assigned to the candidate modes list with the smaller of the two candidates at candModeList[0] and the larger at candModeList[1].

5. IntraPredMode[xB][yB] is derived by applying the following procedure:
   - If prev_intra_pred_flag[xB][yB] is true, the IntraPredMode[xB][yB] is set equal to candModeList[mpm_idx][xB][yB]]
   - Otherwise IntraPredMode[xB][yB] is derived by applying the following ordered steps:
     1. IntraPredMode[xB][yB] = rem_intra_luma_pred_mode
     2. If candModeList[0] is greater than candModeList[1], then two values are swapped. [Ed. (WJ): may be better expression needed]
     3. When IntraPredMode[xB][yB] is equal or greater than candModeList[0], the value of IntraPredMode[xB][yB] is increased by one
     4. When IntraPredMode[xB][yB] is equal or greater than candModeList[1], the value of IntraPredMode[xB][yB] is increased by one

8.3.2 Derivation process for chroma intra prediction mode

[Ed.: (WJ) this subclause may be moved to the semantics of intra_chroma_pred_mode syntax]

Input to this process is a luma location (xB, yB) specifying the top-left luma sample of the current block relative to the top-left luma sample of the current picture.

Output of this process is the variable IntraPredModeC.

The chroma intra prediction mode IntraPredModeC is derived as specifed in Table 8-3 or Table 8-4 with intra_chroma_pred_mode, IntraPredMode[xB][yB] and chroma_pred_from_luma_enabled_flag as inputs.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000611

Table 8-4 – Specification of IntraPredModeC according to the values of intra_chroma_pred_mode and IntraPredMode[xB][yB] when chroma_pred_from_luma_enabled_flag is equal to 1

intra_chroma_pred_mode	IntraPredMode[xB][yB]				
	0	1	2	3	X (0 <= X < 35)
0	7	0	0	0	0
1	1	7	0	0	1
2	2	2	7	1	2
3	3	3	3	7	3
4	LM	LM	LM	LM	LM
5	0	1	2	3	X

Table 8-5 – Specification of IntraPredModeC according to the values of intra_chroma_pred_mode and IntraPredMode[xB][yB] when chroma_pred_from_luma_enabled_flag is equal to 0

intra_chroma_pred_mode	IntraPredMode[xB][yB]				
	0	1	2	3	X (0 <= X < 35)
0	7	0	0	0	0
1	1	7	1	1	1
2	2	2	7	2	2
3	3	3	3	7	3
4	0	1	2	3	X

8.3.3 Decoding process for intra blocks

Inputs to this process are:

– a sample location (xB, yB) specifying the top-left sample of the current block relative to the top-left sample of the current picture,

– a variable log2TrafoSize specifying the size of the current block,

– a variable trafoDepth specifying the hierarchy depth of the current block relative to the coding unit,

– a variable intraPredMode specifying the intra prediction mode.

– a variable cIdx specifying the chroma component of the current block,

Output of this process is:

– a modified reconstructed picture before deblocking filtering.

Depending split_transform_flag[xB][yB][trafoDepth], the following applies:

– If split_transform_flag[xB][yB][trafoDepth] is equal to 1, the following ordered steps apply:

1. The variable xB1 is set equal to xB + ((1 << log2TrafoSize) >> 1).
2. The variable yB1 is set equal to yB + ((1 << log2TrafoSize) >> 1).
3. The decoding process for intra blocks as specified in this subclause is invoked with the location (xB, yB), the variable log2TrafoSize set equal to log2TrafoSize – 1, the variable trafoDepth set equal to trafoDepth + 1, the intra prediction mode intraPredMode, and the variable cIdx as the inputs and the output is a modified reconstructed picture before deblocking filtering.
4. The decoding process for intra blocks as specified in this subclause is invoked with the location (xB1, yB), the variable log2TrafoSize set equal to log2TrafoSize – 1, the variable trafoDepth set equal to trafoDepth + 1, the intra prediction mode intraPredMode, and the variable cIdx as the inputs and the output is a modified reconstructed picture before deblocking filtering.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000612

5. The decoding process for intra blocks as specified in this subclause is invoked with the location (xB, yB1), the variable log2TrafoSize set equal to log2TrafoSize – 1, the variable trafoDepth set equal to trafoDepth + 1, the intra prediction mode intraPredMode, and the variable cIdx as the inputs and the output is a modified reconstructed picture before deblocking filtering.
6. The decoding process for intra blocks as specified in this subclause is invoked with the location (xB1, yB1), the variable log2TrafoSize set equal to log2TrafoSize – 1, the variable trafoDepth set equal to trafoDepth + 1, the intra prediction mode intraPredMode, and the variable cIdx as the inputs and the output is a modified reconstructed picture before deblocking filtering.

– Otherwise (split_transform_flag[xB][yB][trafoDepth] is equal to 0), the following ordered steps apply:

1. The variable nS is set equal to 1 << log2TrafoSize.
2. The intra sample prediction process as specified in subclause 8.3.3.1 is invoked with the location (xB, yB), the intra prediction mode intraPredMode, the prediction size nS and the variable cIdx as the inputs and the output is a (nS)x(nS) array predSamples.
3. The scaling and transformation process as specified in subclause 8.5.1 is invoked with the location (xB, yB), the variable trafoDepth, the variable cIdx, and the transform size trafoSize set equal to nS as the inputs and the output is a (nS)x(nS) array resSamples.
4. The residual signal accumulation process as specified in subclause XXX is invoked with the variable arraySize set equal to nS, the (nS)x(nS) array predSamples, and the (nS)x(nS) array resSamples as the inputs and the output is a (nS)x(nS) array recSamples.
5. The picture reconstruction process for a component before deblocking filtering as specified in subclause XXX is invoked with the location (xB, yB), the variable arraySize set equal to nS, the variable cIdx set equal to 0, and the (nS)x(nS) array recSamples as the inputs and the output is a modified reconstructed picture before deblocking filtering.

8.3.3.1 Intra sample prediction

Inputs to this process are:

– a sample location (xB, yB) specifying the top-left sample of the current block relative to the top-left sample of the current picture,
– a variable intraPredMode specifying the luma intra prediction mode,
– a variable nS specifying the prediction size.
– a variable cIdx specifying the chroma component of the current block,

Output of this process is:

– the predicted samples predSamples[x, y], with x, y =0..nS-1.

The nS*4+1 neighbouring samples p[x, y] that are constructed luma samples prior to the deblocking filter process, with x = -1, y = -1..nS*2-1 and x = 0..nS*2-1, y=-1, are derived as follows.

– The luma location (xBN, yBN) is specified by

$$xBN = xB + x \quad (8\text{-}14)$$

$$yBN = yB + y \quad (8\text{-}15)$$

– Each sample p[x, y] with x = -1, y= -1..nS*2-1 and x = 0, y = -1 is derived as follows
  – If any of the following condition is true, the sample p[x, y] is marked as “not available for intra prediction”
    1. the coding unit covering (xBN, yBN) is not available
    2. the coding unit covering (xBN, yBN) is not coded as intra mode and constrained_intra_pred_flag is equal to 1
  – Otherwise, the sample p[x, y] is marked as “available for intra prediction” and the sample at the location (xBN, yBN) inside the treeblock tbAddrN is assigned to p[x, y].

When at least one sample p[x, y] with x = 1, y = 1..nS*2 1 and x = 0..nS*2 1, y = 1 is marked as “not available for intra prediction,” the reference sample substitution process for intra sample prediction in subclause 8.3.3.1.1 is invoked with the

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000613

samples p[x, y] with x = 1, y = 1..nS*2 1 and x = 0..nS*2 1, y = 1 as input and the modified samples p[x, y] with x = 1, y = 1..nS*2 1 and x = 0..nS*2 1, y = 1 as output.

Depending on intraPredMode, the following ordered steps apply:

1. When cIdx is equal to 0, filtering process of neighbouring samples specified in 8.3.3.1.2 is invoked with the sample array p and the prediction size nS as the inputs and the output is reassigned to the sample array p.
2. Intra sample prediction process according to intraPredMode applies as follows:
    1. One of the intra prediction modes specified in subclause 8.3.3.1.3 to 8.3.3.1.8 is invoked with the sample location (xB, yB), the sample array p, the prediction size nS and the chroma component index cIdx as the inputs and the output is the predicted sample array predSamples according to intraPredMode.

       [Ed. (WJ): some functions do not use some input parameters. Above sentence should be improved]

8.3.3.1.1 Reference sample substitution process for intra sample prediction

Inputs to this process are the reference samples p[x, y] with x = -1, y = -1..nS*2-1 and x = 0..nS*2-1, y = -1 for intra sample prediction.

Outputs of this process are the modified reference samples p[x, y] with x = -1, y = -1..nS*2-1 and x = 0..nS*2-1, y = -1 for intra sample prediction.

The values of the samples p[x, y] with x = -1, y = -1..nS*2-1 and x = 0..nS*2-1, y = -1 are modified as follows:

– If all samples p[x, y] with x = -1, y = -1..nS*2-1 and x = 0..nS*2-1, y = -1 are marked as “not available for intra prediction,” the value (1 << ($BitDepth_Y$ - 1)) is substituted for the values of all samples p[x, y].

– Otherwise (at least one but not all samples p[x, y] are marked as “not available for intra prediction”), the following steps are performed for each sample p[x0, y0] with x0 = -1, y0 = -1..nS*2-1 and x0 = 0..nS*2-1, y0 = -1 marked as "not available for intra prediction":
    – Let the variables q and r be initially set to -1.
    – The variable q is modified as follows:
        – If x0 is equal to -1, a reference sample p[x, y] marked as “available for intra prediction” is identified by searching sequentially among p[x, y] starting from x = -1, y = y0+1 to x = -1, y = nS*2-1. As soon as a sample marked as “available for intra prediction” is found, the search is terminated and the value of p[x, y] is assigned to q. [Ed.: (WJ) is there any elegant way rather than searching?]
        – Otherwise (x0 is not equal to -1), a sample marked as “available for intra prediction” is identified among the samples p[x, y] by searching sequentially starting from x = x0-1, y = -1 to x = -1, y = -1. When a sample marked as “available for intra prediction” is not found among p[x, y] with x = x0-1..-1, y = -1, the search is continued among p[x, y] sequentially starting from x = -1, y = 0 to x = -1, y = nS*2-1. As soon as a sample marked as “available for intra prediction” is found, the search is terminated and the value of p[x, y] is assigned to q. [Ed.: (WJ) is there any elegant way rather than searching?]
    – The variable r is modified as follows:
        – If x0 is equal to -1, a sample marked as “available for intra prediction” is identified among the samples p[x, y] by searching sequentially starting from x = -1, y = y0-1 to x = -1, y = -1. When a sample marked as “available for intra prediction” is not found among p[x, y] with x = -1, y = y0-1..-1, the search is continued among p[x, y] sequentially starting from x = 0, y = -1 to x = nS*2-1, y = -1. As soon as a sample marked as “available for intra prediction” is found, the search is terminated and the value of p[x, y] is assigned to r. [Ed.: (WJ) is there any elegant way rather than searching?]
        – Otherwise (x0 is not equal to -1), a sample marked as “available for intra prediction” is identified among the samples p[x, y] by searching sequentially starting from x = x0+1, y = -1 to x = nS*2-1, y = -1. As soon as a sample marked as “available for intra prediction” is found, the search is terminated and the value of p[x, y] is assigned to r. [Ed.: (WJ) is there any elegant way rather than searching?]
    – The value of the sample p[x0, y0] is modified as follows:
        – If q is not equal to -1 and r is not equal to -1, the value ((q + r + 1) >> 1) is substituted for the value of p[x0, y0].
        – Otherwise, if q is not equal to -1, the value of q is substituted for the value of p[x0, y0].
        – Otherwise (q is equal to -1 and r is not equal to -1), the value of r is substituted for the value of p[x0, y0].

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000614

All samples p[x, y] with x = -1, y = -1..nS*2-1 and x = 0..nS*2-1, y = -1 are marked as "available for intra prediction."

8.3.3.1.2 Filtering process of neighbouring samples

Inputs to this process are:

– neighbouring samples p[x, y], with x, y = -1..2*nS-1,

– a variable nS specifying the prediction size.

Output of this process is:

– filtered samples pF[x, y],. with x, y = -1..2*nS-1.

Table 8-6 – Specification of intraFilterType[nS][IntraPredMode] for various prediction unit sizes

IntraPredMode	intraFilterType for nS = 4	intraFilterType for nS = 8	intraFilterType for nS = 16	intraFilterType for nS = 32	intraFilterType for nS = 64
0-2	0	0	0	0	0
3	0	1	1	1	0
4, 5	0	0	1	1	0
6	0	1	1	1	0
7, 8	0	0	1	1	0
9	0	1	1	1	0
10-20	0	0	1	1	0
21, 22	0	0	0	1	0
23-28	0	0	1	1	0
29, 30	0	0	0	1	0
31-33	0	0	1	1	0
34	0	1	1	1	0
35	n/a	n/a	n/a	n/a	n/a

Filtered sample array pF[x, y] with x = -1..nS*2-1 and y = -1..nS*2-1 are derived as follows:

– When intraFilterType[nS][IntraPredMode] is equal to 1, the following applies:

$$pF[-1, nS*2-1] = p[-1, nS*2-1] \quad (8\text{-}17)$$

$$pF[nS*2-1, -1] = p[nS*2-1, -1] \quad (8\text{-}18)$$

$$pF[-1, y] = (p[-1, y+1] + 2*p[-1, y] + p[-1, y-1] + 2) >> 2 \text{ for } y = nS*2-2..0 \quad (8\text{-}19)$$

$$pF[-1, -1] = (p[-1, 0] + 2*p[-1, -1] + p[0, -1] + 2) >> 2 \quad (8\text{-}20)$$

$$pF[x, -1] = (p[x-1, -1] + 2*p[x, -1] + p[x+1, -1] + 2) >> 2 \text{ for } x = 0..nS*2-2 \quad (8\text{-}21)$$

8.3.3.1.3 Specification of Intra_Vertical prediction mode

Inputs to this process are:

– neighbouring samples p[x, y], with x, y = -1..2*nS-1,

– a variable nS specifying the prediction size

Output of this process is:

– predicted samples predSamples[x, y], with x, y =0..nS-1.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000615

This intra prediction mode is invoked when intraPredMode is equal to 0.

The values of the prediction samples predSamples[x, y], with x, y = 0..nS-1, are derived by

predSamples[x, y] = p[x, -1], with x, y = 0..nS-1 (8-30)

8.3.3.1.4 Specification of Intra_Horizontal prediction mode

Inputs to this process are:

– neighbouring samples p[x, y], with x, y = -1..2*nS-1,

– a variable nS specifying the prediction size.Output of this process is:

– predicted samples predSamples[x, y], with x, y =0..nS-1.

This intra prediction mode is invoked when intraPredMode is equal to 1.

The values of the prediction samples predSamples[x, y], with x, y = 0..nS-1, are derived by

predSamples[x, y] = p[-1, y], with x, y = 0..nS-1 (8-31)

8.3.3.1.5 Specification of Intra_DC prediction mode

Inputs to this process are:

– neighbouring samples p[x, y], with x, y = -1..2*nS-1,

– a variable nS specifying the prediction block size.

– a variable cIdx specifying the chroma component of the current block.

Output of this process is:

– predicted samples predSamples[x, y], with x, y =0..nS-1.

This intra prediction mode is invoked when intraPredMode is equal to 2.

The values of the prediction samples predSamples[x, y], with x, y = 0..nS-1, are derived as the following ordered steps:

1. A variable DCVal is derived as:

$$DCVal = \left(\sum_{x'=0}^{nS-1} p[x', -1] + \sum_{y'=0}^{nS-1} p[-1, y'] + nS \right) >> (k+1), \text{ with x, y = 0..nS-1} \quad (8\text{-}32)$$

where k=log2(nS)

2. Depending on the prediction block size nS and the chroma component index cIdx, the following applies.

– If cIdx is equal to 0 and nS is less than 64, the following applies.

predSamples[0, 0] = (1*p[-1, 0] + 2*DCVal + 1*p[0, -1] + 2) >> 2 (8-32)
predSamples[x, 0] = (1*p[x, -1] + 3*DCVal + 2) >> 2, with x = 1..nS-1 (8-32)
predSamples[0, y] = (1*p[-1, y] + 3*DCVal + 2) >> 2, with y = 1..nS-1 (8-32)
predSamples[x, y] = DCVal, with x, y = 1..nS-1 (8-32)

– Otherwise, the prediction samples predSamples[x, y] are derived as

predSamples[x, y] = DCVal, with x, y = 0..nS-1 (8-32)

8.3.3.1.6 Specification of Intra_Angular prediction mode

Inputs to this process are:

– neighbouring samples p[x, y], with x, y = -1..2*nS-1,

– a variable nS specifying the prediction size.

Output of this process is:

– predicted samples predSamples[x, y], with x, y =0..nS-1.

This intra prediction mode is invoked when intraPredMode is in the range of 3..33.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000616

Table 8-6 specifies the mapping table between intraPredMode and the rearranged intra prediction order intraPredOrder.

Table 8-7 – Specification of intraPredOrder

intraPredMode	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
intraPredOrder	-	-	-	-	1	5	13	17	21	29	33	3	7	11	15	19	23	27
intraPredMode	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	
intraPredOrder	31	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	

Figure 8-2 illustrates the total 34 intra angles and Table 8-7 specifies the mapping table between intraPredOrder and the angle parameter intraPredAngle.

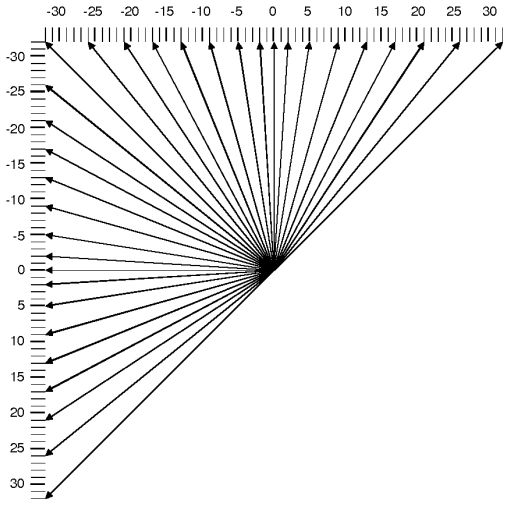


Figure 8-2 – Intra prediction angle definition (informative)

Table 8-8 – Specification of intraPredAngle

intraPredOrder	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
intraPredAngle	-	-32	-26	-21	-17	-13	-9	-5	-2	-	2	5	9	13	17	21	26
intraPredOrder	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
intraPredAngle	32	-26	-21	-17	-13	-9	-5	-2	-	2	5	9	13	17	21	26	32

Table 8-8 further specifies the mapping table between intraPredOrder and the inverse angle parameter invAngle.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000617

Table 8-9 – Specification of invAngle

intraPredOrder	1	2	3	4	5	6	7	8
invAngle	-256	-315	-390	-482	-630	-910	-1638	-4096
intraPredOrder	18	19	20	21	22	23	24	25
invAngle	-315	-390	-482	-630	-910	-1638	-4096	-

The reference pixel array refMain[x], with x=-nS..2*nS is specified as follows.

– If intraPredOrder is less than 18,

refMain[x] = p[-1+x, -1], with x=0..nS (8-36)

– If intraPredAngle is less than 0,

refMain[x] = p[-1, -1+((x*invAngle+128)>>8)], with x=(nS*intraPredAngle) >>5..-1 (8-37)

– Otherwise,

refMain[x] = p[-1+x, -1], with x=nS+1..2*nS (8-38)

Otherwise,

refMain[x] = p[-1, -1+x], with x=0..nS (8-39)

– If intraPredAngle is less than 0,

refMain[x] = p[-1+((x*invAngle+128)>>8), -1], with x=(nS*intraPredAngle) >>5..-1 (8-40)

– Otherwise,

refMain[x] = p[-1, -1+x], with x=nS+1..2*nS (8-41)

The values of the prediction samples predSamples[x, y], with x, y = 0..nS-1 are derived by the following procedures.

– The index variable iIdx and the multiplication factor iFact are derived by

iIdx = ((y + 1)*intraPredAngle) >> 5 (8-42)

iFact = ((y + 1)*intraPredAngle) && 31 (8-43)

– Depending on the value of iFact, the following applies.

– If iFact is not equal to 0, the value of the prediction samples predSamples[x, y] is derived by

predSamples[x, y] = ((32 – iFact)*refMain[x+iIdx+1] + iFact*refMain[x+iIdx+2] + 16) >> 5 (8-44)

– Otherwise, the value of the prediction samples predSamples[x, y] is derived by

predSamples[x, y] = refMain[x+iIdx+1] (8-45)

8.3.3.1.7 Specification of Intra_Planar prediction mode

Inputs to this process are:

– neighbouring samples p[x, y], with x, y = -1..2*nS-1,

– a variable nS specifying the prediction size,

Output of this process is:

– predicted samples predSamples[x, y], with x, y =0..nS-1.

This intra prediction mode is invoked when intraPredMode is equal to 34.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000618

The values of the prediction samples predSamples[x, y], with x, y = 0..nS-1, are derived by

```
predSamples[ x, y ] = (
        ( nS – 1 – x ) * p[ -1, y ] + ( x + 1 ) * p[ nS, -1 ] +
        ( nS – 1 – y ) * p[ x ,-1 ] + ( y + 1 ) * p[ -1, nS ] + nS ) >> ( k + 1 )          (8-45)
with x, y = 0..nS-1 where k = log2( nS )
```

8.3.3.1.8 Specification of Intra_FromLuma prediction mode

Inputs to this process are:

– a sample location (xB, yB) specifying the top-left sample of the current block relative to the top-left sample of the current picture,

– neighbouring samples p[x, y], with x, y = -1..2*nS-1,

– a variable nS specifying the prediction size.

Output of this process is:

– predicted samples predSamples[x, y], with x, y =0..nS-1.

This intra prediction mode is invoked when intraPredMode is equal to 35.

The values of the prediction samples predSamples[x, y], with x, y = 0..nS-1, are derived as the following ordered steps:

1. Variable k3 and the sample array p_Y' are derived as:

   $k3 = \mathrm{Max}(0, \mathrm{BitDepth_C} + \log_2(nS) – 14)$ (8-30)

   $p_Y'[x, y] = (\mathrm{recSamples_L}[2x-1, 2y+1] + 2*\mathrm{recSamples_L}[2x, 2y+1] + \mathrm{recSamples_L}[2x+1, 2y+1] + 2) >> 2$, with x=0..nS-1, y = -1 (8-30)

   $p_Y'[x, y] = (\mathrm{recSamples_L}[2x, 2y] + \mathrm{recSamples_L}[2x, 2y+1]) >> 1$, with x=-1..nS-1, y = 0..nS-1 (8-30)

2. Variables L, C, LL, LC and k2 are derived as follows:

   $$L = \left(\sum_{y=0}^{nS-1} p_Y'[-1, y] + \sum_{x=0}^{nS-1} p_Y'[x, -1] \right) >> k3 \quad (8\text{-}30)$$

   $$C = \left(\sum_{y=0}^{nS-1} p[-1, y] + \sum_{x=0}^{nS-1} p[x, -1] \right) >> k3 \quad (8\text{-}30)$$

   $$LL = \left(\sum_{y=0}^{nS-1} p_Y'[-1, y]^2 + \sum_{x=0}^{nS-1} p_Y'[x, -1]^2 \right) >> k3 \quad (8\text{-}30)$$

   $$LC = \left(\sum_{y=0}^{nS-1} p_Y'[-1, y] * p[-1, y] + \sum_{y=0}^{nS-1} p_Y'[x, -1] * p[x,-1] \right) >> k3 \quad (8\text{-}30)$$

   $k2 = \log_2((2*nS) >> k3)$ (8-30)

3. Variables a, b and k are derived as:

   $a1 = (LC << k2) – L*C$ (8-30)
   $a2 = (LL << k2) – L*L$ (8-30)
   $k1 = \mathrm{Max}(0, \log_2(\mathrm{abs}(a2)) – 5) – \mathrm{Max}(0, \log_2(\mathrm{abs}(a1)) – 14) + 2$ (8-30)
   $a1s = a1 >> \mathrm{Max}(0, \log_2(\mathrm{abs}(a1)) – 14)$ (8-30)
   $a2s = \mathrm{abs}(a2 >> \mathrm{Max}(0, \log_2(\mathrm{abs}(a2)) – 5))$ (8-30)
   $a3 = a2s < 1 \ ? \ 0 : \mathrm{Clip3}(-2^{15}, 2^{15}-1, a1s*\mathrm{lmDiv} + (1 << (k1 – 1)) >> k1)$ (8-30)

   $a = a3 >> \mathrm{Max}(0, \log_2(\mathrm{abs}(a3)) – 6)$ (8-30)
   $k = 13 – \mathrm{Max}(0, \log_2(\mathrm{abs}(a)) – 6)$ (8-30)

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000619

$$b = (L - ((a*C) >> k1) + (1 << (k2 - 1))) >> k2 \quad (8\text{-}30)$$

where lmDiv is specified in Table 8-10 with the input a2s.

4. The values of the prediction samples predSamples[x, y] are derived as:

$$predSamples[x, y] = Clip1_C(((py'[x, y] * a) >> k) + b), \text{ with } x, y = 0..nS-1 \quad (8\text{-}30)$$

Table 8-10 – Specification of lmDiv

a2s	1	2	3	4	5	6	7	8	9	10	11	12	13
lmDiv	32768	16384	10923	8192	6554	5461	4681	4096	3641	3277	2979	2731	2521
a2s	14	15	16	17	18	19	20	21	22	23	24	25	26
lmDiv	2341	2185	2048	1928	1820	1725	1638	1560	1489	1425	1365	1311	1260
a2s	27	28	29	30	31	32	33	34	35	36	37	38	39
lmDiv	1214	1170	1130	1092	1057	1024	993	964	936	910	886	862	840
a2s	40	41	42	43	44	45	46	47	48	49	50	51	52
lmDiv	819	799	780	762	745	728	712	697	683	669	655	643	630
a2s	53	54	55	56	57	58	59	60	61	62	63	64	
lmDiv	618	607	596	585	575	565	555	546	537	529	520	512	

8.4 Decoding process for coding units coded in inter prediction mode

Inputs to this process are:

– a luma location (xC, yC) specifying the top-left luma sample of the current coding unit relative to the top left luma sample of the current picture,

– a variable log2CUSize specifying the size of the current coding unit.

Output of this process is a modified reconstructed picture before deblocking filtering.

The variable nCS_L is set equal to 1 << log2CUSize and the variable nCS_C is set equal to (1 << log2CUSize) >> 1.

Decoding process for coding units coded in inter prediction mode consists of following ordered steps:

1. The inter prediction process as specified in subclause 8.4.1 is invoked with the luma location (xC, yC), coding unit size log2CUSize, inter prediction mode PredMode, and prediction partition mode PartMode as the inputs and the outputs are 3 arrays $predSamples_L$, $predSamples_{Cb}$, $predSamples_{Cr}$.
2. The decoding process for the residual signal of coding units coded in inter prediction mode specified in subclause 8.4.3 is invoked with the luma location (xC, yC), the size of the current coding unit log2CUSize as inputs and the outputs are 3 arrays $resSamples_L$, $resSamples_{Cb}$, $resSamples_{Cr}$.
3. The residual signal accumulation process as specified in subclause XXXX is invoked with arrays $predSamples_L$, $predSamples_{Cb}$, $predSamples_{Cr}$, and the arrays $resSamples_L$, $resSamples_{Cb}$, $resSamples_{Cr}$ as the inputs and the outputs are 3 arrays $recSamples_L$, $recSamples_{Cb}$, $recSamples_{Cr}$. [Ed: (WJ) it may be split per each color component. Revisit after writing residual signal accumulation process]
4. The picture reconstruction process for a component before deblocking filtering as specified in subclause XXXX is invoked with the luma location (xB, yB), and 3 arrays recSamplesL, recSamplesCb, recSamplesCr as the inputs and the output is a modified reconstructed picture before deblocking filtering. [Ed: (WJ) it may be split per each color component. Revisit after writing picture reconstruction process]

8.4.1 Inter prediction process

This process is invoked when decoding coding unit whose PredMode is not equal to MODE_INTRA.

Inputs to this process are:

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000620

– a luma location (xC, yC) specifying the top-left luma sample of the current coding unit relative to the top left luma sample of the current picture,

– a variable log2CUSize specifying the size of the current coding unit,

– a variable PredMode specifying prediction mode of current coding unit,

– a variable PartMode specifying prediction partition mode of current coding unit.

Outputs of this process are:

– a (nCS_L)x(nCS_L) array $predSamples_L$ of luma prediction samples, where nCS_L is derived as specified below,

– a (nCS_C)x(nCS_C) array $predSamples_{Cb}$ of chroma prediction samples for the component Cb, where nCS_C is derived as specified below,

– a (nCS_C)x(nCS_C) array $predSamples_{Cr}$ of chroma prediction samples for the component Cr, where nCS_C is derived as specified below.

The variable nCS_L is set equal to 1 << log2CUSize and the variable nCS_C is set equal to (1 << log2CUSize) >> 1. [Ed: (WJ) revisit for supporting other chroma formats]

The variable $nCS1_L$ is set equal to nCS_L >> 1.

Depending on PartMode, the following applies: [Ed: (WJ) is PUType better?]

– If PartMode is equal to PART_2Nx2N, the following ordered steps apply:

1. The decoding process for prediction units in inter prediction mode as specified in subclause 8.4.2 is invoked with the luma location (xC, yC), the luma location (xB, yB) set equal to (0, 0), the size of the coding unit nCS_L, the width of the luma prediction samples nPSW set equal to nCS_L, the height of the luma prediction samples nPSH set equal to nCS_L and a partition index PartIdx set equal to 0 as inputs, and the outputs are a (nCS_L)x(nCS_L) array $predSamples_L$ and two (nCS_C)x(nCS_C) arrays $predSamples_{Cb}$ and $predSamples_{Cr}$.

– Otherwise, if PartMode is equal to PART_2NxN, the following ordered steps apply:

1. The decoding process for prediction units in inter prediction mode as specified in subclause 8.4.2 is invoked with the luma location (xC, yC), the luma location (xB, yB) set equal to (0, 0), the size of the coding unit nCS_L, the width of the luma prediction samples nPSW set equal to nCS_L, the height of the luma prediction samples nPSH set equal to $nCS1_L$ and a partition index PartIdx set equal to 0 as inputs, and the outputs are a (nCS_L)x(nCS_L) array $predSamples_L$ and two (nCS_C)x(nCS_C) arrays $predSamples_{Cb}$ and $predSamples_{Cr}$.

2. The decoding process for prediction units in inter prediction mode as specified in subclause 8.4.2 is invoked with the luma location (xC, yC), the luma location (xB, yB) set equal to (0, $nCS1_L$), the size of the coding unit nCS_L, the width of the luma prediction samples nPSW set equal to nCS_L, the height of the luma prediction samples nPSH set equal to $nCS1_L$ and a partition index PartIdx set equal to 1 as inputs, and the outputs are a (nCS_L)x(nCS_L) array $predSamples_L$ and two (nCS_C)x(nCS_C) arrays $predSamples_{Cb}$ and $predSamples_{Cr}$.

Otherwise, if PartMode is equal to PART_Nx2N, the following ordered steps apply:

1. The decoding process for prediction units in inter prediction mode as specified in subclause 8.4.2 is invoked with the luma location (xC, yC), the luma location (xB, yB) set equal to (0, 0), the size of the coding unit nCS_L, the width of the luma prediction samples nPSW set equal to $nCS1_L$, the height of the luma prediction samples nPSH set equal to nCS_L and a partition index PartIdx set equal to 0 as inputs, and the outputs are a (nCS_L)x(nCS_L) array $predSamples_L$ and two (nCS_C)x(nCS_C) arrays $predSamples_{Cb}$ and $predSamples_{Cr}$.

2. The decoding process for prediction units in inter prediction mode as specified in subclause 8.4.2 is invoked with the luma location (xC, yC), the luma location (xB, yB) set equal to ($nCS1_L$, 0), the size of the coding unit nCS_L, the width of the luma prediction samples nPSW set equal to $nCS1_L$, the height of the luma prediction samples nPSH set equal to nCS_L and a partition index PartIdx set equal to 1 as inputs, and the outputs are a (nCS_L)x(nCS_L) array $predSamples_L$ and two (nCS_C)x(nCS_C) arrays $predSamples_{Cb}$ and $predSamples_{Cr}$.

Otherwise, if PartMode is equal to PART_NxN, the following ordered steps apply:

1. The decoding process for prediction units in inter prediction mode as specified in subclause 8.4.2 is invoked with the luma location (xC, yC), the luma location (xB, yB) set equal to (0, 0), the size of the coding unit nCS_L, the width of the luma prediction samples nPSW set equal to $nCS1_L$, the height of the luma prediction samples nPSH set equal to $nCS1_L$ as and a partition index PartIdx set equal to 0 inputs, and the outputs are a (nCS_L)x(nCS_L) array $predSamples_L$ and two (nCS_C)x(nCS_C) arrays $predSamples_{Cb}$ and $predSamples_{Cr}$.

2. The decoding process for prediction units in inter prediction mode as specified in subclause 8.4.2 is invoked with the luma location (xC, yC), the luma location (xB, yB) set equal to ($nCS1_L$, 0), the size of the coding unit

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000621

nCS_L, the width of the luma prediction samples nPSW set equal to $nCS1_L$, the height of the luma prediction samples nPSH set equal to $nCS1_L$ and a partition index PartIdx set equal to 1 as inputs, and the outputs are a $(nCS_L)x(nCS_L)$ array $predSamples_L$ and two $(nCS_C)x(nCS_C)$ arrays $predSamples_{Cb}$ and $predSamples_{Cr}$.

3. The decoding process for prediction units in inter prediction mode as specified in subclause 8.4.2 is invoked with the luma location (xC, yC), the luma location (xB, yB) set equal to (0, $nCS1_L$), the size of the coding unit nCS_L, the width of the luma prediction samples nPSW set equal to $nCS1_L$, the height of the luma prediction samples nPSH set equal to $nCS1_L$ and a partition index PartIdx set equal to 2 as inputs, and the outputs are a $(nCS_L)x(nCS_L)$ array $predSamples_L$ and two $(nCS_C)x(nCS_C)$ arrays $predSamples_{Cb}$ and $predSamples_{Cr}$.

4. The decoding process for inter prediction units as specified in subclause 8.4.2 is invoked with the luma location (xC, yC), the luma location (xB, yB) set equal to ($nCS1_L$, $nCS1_L$), the size of the coding unit nCS_L, the width of the luma prediction samples nPSW set equal to $nCS1_L$, the height of the luma prediction samples nPSH set equal to $nCS1_L$ and a partition index PartIdx set equal to 3 as inputs, and the outputs are a $(nCS_L)x(nCS_L)$ array $predSamples_L$ and two $(nCS_C)x(nCS_C)$ arrays $predSamples_{Cb}$ and $predSamples_{Cr}$.

8.4.2 Decoding process for prediction units in inter prediction mode

Inputs to this process are:

– a luma location (xC, yC) specifying the top-left luma sample of the current coding unit relative to the top left luma sample of the current picture,

– a luma location (xB, yB) specifying the top-left luma sample of the current prediction unit relative to the top left luma sample of the current coding unit,

– a variable nCS specifying the size of the current coding unit,

– a variable nPSW specifying the width of the current prediction unit,

– a variable nPSH specifying the width of the current prediction unit,

– a variable PartIdx specifying the index of the current prediction unit within the current coding unit.

Outputs of this process are:

– a $(nCS_L)x(nCS_L)$ array $predSamples_L$ of luma prediction samples, where nCS_L is derived as specified below,

– a $(nCS_C)x(nCS_C)$ array $predSamples_{Cb}$ of chroma prediction samples for the component Cb, where nCS_C is derived as specified below,

– a $(nCS_C)x(nCS_C)$ array $predSamples_{Cr}$ of chroma prediction samples for the component Cr, where nCS_C is derived as specified below.

The variable nCS_L is set equal to nCS and the variable nCS_C is set equal to nCS >> 1. [Ed: (WJ) revisit for supporting other chroma formats]

The decoding process for prediction units in inter prediction mode consists of the following ordered steps:

1. Derivation process for motion vector components and reference indices as specified in subclause 8.4.2.1.

   Inputs to this process are

   – a luma location (xC, yC) of the top-left luma sample of the current coding unit relative to the top-left luma sample of the current picture,

   – a luma location (xB, yB) of the top-left luma sample of the current prediction unit relative to the top-left luma sample of the current coding unit,

   – variables specifying the width and the height of the prediction unit for luma, nPSW and nPSH,

   – a variable PartIdx specifying the index of the current prediction unit within the current coding unit.

   Outputs of this process are

   – luma motion vectors mvL0 and mvL1, and chroma motion vectors mvCL0 and mvCL1,

   – reference indices refIdxL0 and refIdxL1,

   – prediction list utilization flags predFlagL0 and predFlagL1.

2. Decoding process for inter sample prediction as specified in subclause 8.4.2.2.

   Inputs to this process are

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000622

– a luma location (xC, yC) of the top-left luma sample of the current coding unit relative to the top-left luma sample of the current picture,
– a luma location (xB, yB) of the top-left luma sample of the current prediction unit relative to the top-left luma sample of the current coding unit,
– a variable nCS specifying the size of the current coding unit,
– variables specifying the width and the height of the prediction unit for luma, nPSW and nPSH.
– luma motion vectors mvL0 and mvL1, and chroma motion vectors mvCL0 and mvCL1,
– reference indices refIdxL0 and refIdxL1,
– prediction list utilization flags predFlagL0 and prefFlagL1.

Outputs of this process are

– inter prediction samples (predSamples); which are a (nCS_L)x(nCS_L) array $predSamples_L$ of prediction luma samples and two (nCS_C)x(nCS_C) arrays $predSamples_{Cr}$, and $predSamples_{Cr}$ of prediction chroma samples, one for each of the chroma components Cb and Cr.

For use in derivation processes of variables invoked later in the decoding process, the following assignments are made:

MvL0[xB, yB] = mvL0 (8-46)
MvL1[xB, yB] = mvL1 (8-47)

RefIdxL0[xB, yB] = refIdxL0 (8-48)
RefIdxL1[xB, yB] = refIdxL1 (8-49)

PredFlagL0[xB, yB] = predFlagL0 (8-50)
PredFlagL1[xB, yB] = predFlagL1 (8-51)

8.4.2.1 Derivation process for motion vector components and reference indices

Input to this process are

– a luma location (xC, yC) of the top-left luma sample of the current coding unit relative to the top-left luma sample of the current picture,
– a luma location (xB, yB) of the top-left luma sample of the current prediction unit relative to the top-left luma sample of the current coding unit,
– variables specifying the width and the height of the prediction unit for luma, nPSW and nPSH,
– a variable PartIdx specifying the index of the current prediction unit within the current coding unit.

Outputs of this process are

– luma motion vectors mvL0 and mvL1 and chroma motion vectors mvCL0 and mvCL1,
– reference indices refIdxL0 and refIdxL1,
– prediction list utilization flags predFlagL0 and predFlagL1.

Let (xP, yP) specify the top-left luma sample of the current prediction unit relative to the top-left luma sample of the current picture where xP = xC + xB and yP = yC + yB.

For the derivation of the variables mvL0 and mvL1, refIdxL0 and refIdxL1 as well as PredFlagL0 and PredFlagL1, the following applies.

– If PredMode is equal to MODE_SKIP, the derivation process for luma motion vectors for merge mode as specified in subclause 8.4.2.1.3 is invoked with the luma location (xP, yP), variables nPSW, nPSH and the partition index PartIdx as inputs and the output being the luma motion vectors mvL0, mvL1, the reference indices refIdxL0, refIdxL1, and the prediction list utilization flags predFlagL0 and predFlagL1.
– Otherwise, if PredMode is equal to MODE_INTER and merge_flag[xP][yP] is equal to 1,, the derivation process for luma motion vectors for merge mode as specified in subclause 8.4.2.1.3 is invoked with the luma location (xP, yP), variables nPSW and nPSH and the partition index PartIdx as inputs and the outputs being the luma motion vectors mvL0 and mvL1, the reference indices refIdxL0 and refIdxL1, the prediction utilization flags predFlagL0 and predFlagL1.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000623

– Otherwise, for X being replaced by either 0 or 1 in the variables predFlagLX, mvLX, refIdxLX and in Pred_LX and in the syntax elements ref_idx_lX and mvd_lX, the following applies.

1. The variables LcToLx, refIdxLX and predFlagLX are derived as follows.

   – If inter_pred_flag[xP][yP] is equal to Pred_LC and PredLCToPredLx[ref_idx_lc[xP][yP]] is equal to Pred_LX,

   refIdxLX = RefIdxLCToRefIdxLx[ref_idx_lc[xP][yP]] (8-52)
   predFlagLX = 1 (8-53)
   mvd_lX[xP][yP][0] = mvd_lc[xP][yP][0] (8-54)
   mvd_lX[xP][yP][1] = mvd_lc[xP][yP][1] (8-55)
   mvp_idx_lX[xP][yP][0] = mvp_idx_lc[xP][yP] (8-56)
   LcToLx = LX (8-57)

   – Otherwise, if inter_pred_flag[xP][yP] is equal to Pred_LX or Pred_BI,

   refIdxLX = ref_idx_lX[xP][yP] (8-58)
   predFlagLX = 1 (8-59)

   – Otherwise, the variables refIdxLX and predFlagLX are specified by

   refIdxLX = -1 (8-60)
   predFlagLX = 0 (8-61)

2. The variable mvdLX is derived as follows.

   mvdLX[0] = mvd_lX[xP][yP][0] (8-62)
   mvdLX[1] = mvd_lX[xP][yP][1] (8-63)

3. When predFlagLX is equal to 1, the variable mvpLX is derived as follows.

   – The derivation process for luma motion vector prediction in subclause 8.4.2.1.5 is invoked with the luma location (xP, yP), variables nPSW and nPSH and refIdxLX as the inputs and the output being mvpLX.

4. When predFlagLX is equal to 1, the luma motion vector mvLX is derived as

   mvLX[0] = mvpLX[0] + mvdLX[0] (8-64)
   mvLX[1] = mvpLX[1] + mvdLX[1] (8-65)

When ChromaArrayType is not equal to 0 and predFlagLX (with X being either 0 or 1) is equal to 1, the derivation process for chroma motion vectors in subclause 8.4.2.1.8 is invoked with mvLX and refIdxLX as inputs and the output being mvCLX.

8.4.2.1.1 Derivation process for luma motion vectors for merge mode

This process is only invoked when PredMode is equal to MODE_SKIP or PredMode is equal to MODE_ INTER and merge_flag [xP][yP] is equal to 1, where (xP, yP) specify the top-left luma sample of the current prediction unit relative to the top-left luma sample of the current picture.

Inputs of this process are

– a luma location (xP, yP) of the top-left luma sample of the current prediction unit relative to the top-left luma sample of the current picture,

– variables specifying the width and the height of the prediction unit for luma, nPSW and nPSH,

– a variable PartIdx specifying the index of the current prediction unit within the current coding unit.

Outputs of this process are

– the luma motion vectors mvL0 and mvL1,

– the reference indices refIdxL0 and refIdxL1,

– the prediction list utilization flags predFlagL0 and predFlagL1.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000624

The motion vectors mvL0 and mvL1, the reference indices refIdxL0 and refIdxL1, and the prediction utilization flags predFlagL0 and predFlagL1 are derived as specified by the following ordered steps:

1. The derivation process for merging candidates from neighboring prediction unit partitions in subclause 8.4.2.1.2 is invoked with luma location (xP, yP), the width and the height of the prediction unit nPSW and nPSH and the partition index PartIdx as inputs and the output is assigned to the availability flags availableFlagN, the reference indices refIdxL0N and refIdxL1N, the prediction list utilization flags predFlagL0N and predFlagL1N and the motion vectors mvL0N and mvL1N with N being replaced by A_0, A_1, B_0, B_1 or B_2.
2. The derivation process of reference indices for temporal merging candidate in subclause 8.4.2.1.3 is invoked with luma location (xP, yP), nPSW, nPSH as the inputs and the output is directly assigned to refIdxLX.
3. The derivation process for temporal luma motion vector prediction in subclause 8.4.2.1.9 is invoked with luma location (xP, yP), refIdxLX as the inputs and with the output being the availability flag availableFlagLXCol and the temporal motion vector mvLXCol. The variables availableFlagCol and predFlagLXCol (with X being 0 or 1, respectively) are derived as specified below.

   availableFlagCol = availableFlagL0Col || availableFlagL1Col (8-71)

   predFlagLXCol = availableFlagLXCol (8-72)

4. The merging candidate list, mergeCandList, is constructed as follows.
   1. A_1, if availableFlagA_1 is equal to 1
   2. B_1, if availableFlagB_1 is equal to 1
   3. B_0, if availableFlagB_0 is equal to 1
   4. A_0, if availableFlagA_0 is equal to 1
   5. B_2, if availableFlagB_2 is equal to 1
   6. Col, if availableFlagCol is equal to 1
5. When merging candidates have the same motion vectors and the same reference indices, the merging candidates are removed from the list except the merging candidate which has the smallest order in the mergeCandList.
6. The variable numMergeCand and numOrigMergeCand are set to the number of merging candidates in the mergeCandList and the variable maxNumMergeCand is set to 5.
7. When slice_type is equal to B, the following applies.
   - The derivation process for combined bi-predictive merging candidates specified in subclause 8.4.2.1.4 is invoked with mergeCandList, the reference indices refIdxL0N and refIdxL1N, the prediction list utilization flags predFlagL0N and predFlagL1N, the motion vectors mvL0N and mvL1N of every candidate N being in mergeCandList, numMergeCand, numOrigMergeCand and maxNumMergeCand given as input and the output is assigned to mergeCandList, numMergeCand, the reference indices refIdxL0combCand$_k$ and refIdxL1combCand$_k$, the prediction list utilization flags predFlagL0combCand$_k$ and predFlagL1combCand$_k$ and the motion vectors mvL0combCand$_k$ and mvL1combCand$_k$ of every new candidate combCand$_k$ being added in mergeCandList. The number of candidates being added numCombMergeCand is set equal to (numMergeCand – numOrigMergeCand). When numCombMergeCand is greater than 0, k ranges from 0 to numCombMergeCand − 1, inclusive.
   - The derivation process for non-scaled bi-predictive merging candidates specified in subclause 8.4.2.1.5 is invoked with mergeCandList, the reference indices refIdxL0N and refIdxL1N, the prediction list utilization flags predFlagL0N and predFlagL1N, the motion vectors mvL0N and mvL1N of every candidate N being in mergeCandList, numMergeCand, numOrigMergeCand and maxNumMergeCand given as input and the output is assigned to mergeCandList, numMergeCand, the reference indices refIdxL0nscaleCand$_l$ and refIdxL1nscaleCand$_l$, the prediction list utilization flags predFlagL0nscaleCand$_l$ and predFlagL1nscaleCand$_l$, the motion vectors mvL0nscaleCand$_l$ and mvL1nscaleCand$_l$ of every new candidate nscaleCand$_l$ being added in mergeCandList. The number of candidates being added numNscaleMergeCand is set equal to (numMergeCand – numOrigMergeCand – numCombMergeCand). When numNscaleMergeCand is greater than 0, l ranges from 0 to numNscaleMergeCand – 1, inclusive.
8. The derivation process for zero motion vector merging candidates specified in subclause 8.4.2.1.6 is invoked with the mergeCandList, the reference indices refIdxL0N and refIdxL1N, the prediction list utilization flags predFlagL0N and predFlagL1N, the motion vectors mvL0N and mvL1N of every candidate N being in mergeCandList, the NumMergeCand and the MaxNumMergeCand as the inputs and the output is assigned to mergeCandList, numMergeCand, the reference indices refIdxL0zeroCand$_m$ and refIdxL1zeroCand$_m$, the

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000625

prediction list utilization flags predFlagL0zeroCand$_m$ and predFlagL1zeroCand$_m$, the motion vectors mvL0zeroCand$_m$ and mvL1zeroCand$_m$ of every new candidate zeroCand$_m$ being added in mergeCandList. The number of candidates being added numZeroMergeCand is set equal to (numMergeCand – numOrigMergeCand – numCombMergeCand – numNscaleMergeCand). When numZeroMergeCand is greater than 0, m ranges from 0 to numZeroMergeCand – 1, inclusive.

9. The following assignments are made with N being the candidate at position merge_idx[xP][yP] in the merging candidate list mergeCandList (N = mergeCandList[merge_idx[xP][yP]]) and X being replaced by 0 or 1:

mvLX[0] = mvLXN[0] (8-88)

mvLX[1] = mvLXN[1] (8-89)

refIdxLX = refIdxLXN (8-90)

predFlagLX = predFlagLXN (8-91)

8.4.2.1.2 Derivation process for spatial merging candidates

Inputs to this process are

– a luma location (xP, yP) specifying the top-left luma sample of the current prediction unit relative to the top-left sample of the current picture,

– variables specifying the width and the height of the prediction unit for luma, nPSW and nPSH,

– a variable PartIdx specifying the index of the current prediction unit within the current coding unit.

Outputs of this process are (with N being replaced by A_0, A_1, B_0, B_1 or B_2 and with X being replaced by 0 or 1)

– the availability flags availableFlagN of the neighbouring prediction units,

– the reference indices refIdxLXN of the neighbouring prediction units,

– the prediction list utilization flags predFlagLXN of the neighbouring prediction units,

– the motion vectors mvLXN of the neighbouring prediction units.

For the derivation of availableFlagN, with N being A_0, A_1, B_0, B_1 or B_2 and (xN, yN) being (xP – 1, yP + nPSH), (xP – 1, yP + nPSH – 1), (xP + nPSW, yP – 1), (xP + nPSW – 1, yP – 1) or (xP – 1, yP – 1), the following applies.

– If one of the following conditions is true, the availableFlagN is set equal to 0, both components mvLXN are set equal to 0, refIdxLXN and predFlagLX[xN, yN] of the prediction unit covering luma location (xN, yN) are assigned respectively to mvLXN, refIdxLXN and predFlagLXN.
  – N is equal to B_2 and availableFlagA_0 + availableFlagA_1 + availableFlagB_0 + availableFlagB_1 is equal to 4.
  – The prediction unit covering luma location (xN, yN) is not available or PredMode is MODE_INTRA.
  – PartMode of the current prediction unit is PART_2NxN or PART_2NxnU or PART_2NxnD and PartIdx is equal to 1 and the prediction units covering luma location (xP, yP – 1) (PartIdx = 0) and luma location (xN, yN) (Cand. N) have identical motion parameters:
    – mvLX[xP, yP – 1] = = mvLX[xN, yN]
    – refIdxLX[xP, yP – 1] = = refIdxLX[xN, yN]
    1. predFlagLX[xP, yP – 1] = = predFlagLX[xN, yN]
  – PartMode of the current prediction unit is PART_Nx2N or PART_nLx2N or PART_nRx2N and PartIdx is equal to 1 and the prediction units covering luma location (xP – 1, yP) (PartIdx = 0) and luma location (xN, yN) (Cand. N) have identical motion parameters:
    – mvLX[xP – 1, yP] = = mvLX[xN, yN]
    – refIdxLX[xP – 1, yP] = = refIdxLX[xN, yN]
    – predFlagLX[xP – 1, yP] = = predFlagLX[xN, yN]
  – PartMode of the current prediction unit is PART_NxN and PartIdx is equal to 3 and the prediction units covering luma location (xP – 1, yP) (PartIdx = 2) and luma location (xP – 1, yP – 1) (PartIdx = 0) have identical motion parameters:
    – mvLX[xP – 1, yP] = = mvLX[xP – 1, yP – 1]

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000626

- refIdxLX[xP – 1, yP] == refIdxLX[xP – 1, yP – 1]
- predFlagLX[xP – 1, yP] == predFlagLX[xP – 1, yP – 1]

and the prediction units covering luma location (xP, yP – 1) (PartIdx = 1) and luma location (xN, yN) (Cand. N) have identical motion parameters:

- mvLX[xP, yP – 1] == mvLX[xN, yN]
- refIdxLX[xP, yP – 1] == refIdxLX[xN, yN]
- predFlagLX[xP, yP – 1] == predFlagLX[xN, yN]

– PartMode of the current prediction unit is PART_NxN and PartIdx is equal to 3 and the prediction units covering luma location (xP, yP – 1) (PartIdx = 1) and luma location (xP – 1, yP – 1) (PartIdx = 0) have identical motion parameters:

- mvLX[xP, yP – 1] == mvLX[xP – 1, yP – 1]
- refIdxLX[xP, yP – 1] == refIdxLX[xP – 1, yP – 1]
- predFlagLX[xP, yP – 1] == predFlagLX[xP – 1, yP – 1]

and the prediction units covering luma location (xP – 1, yP) (PartIdx = 2) and luma location (xN, yN) (Cand. N) have identical motion parameters:

- mvLX[xP – 1, yP] == mvLX[xN, yN]
- refIdxLX[xP – 1, yP] == refIdxLX[xN, yN]
- predFlagLX[xP – 1, yP] == predFlagLX[xN, yN]

– Otherwise, availableFlagN is set equal to 1 and the variables mvLX[xN, yN], refIdxLX[xN, yN] and predFlagLX[xN, yN] of the prediction unit covering luma location (xN, yN) are assigned respectively to mvLXN, refIdxLXN and predFlagLXN.

8.4.2.1.3 Derivation process of reference indices for temporal merging candidate

Inputs to this process are

– a luma location (xP, yP) specifying the top-left luma sample of the current prediction unit relative to the top-left sample of the current picture,

– variables specifying the width and the height of the prediction unit for luma, nPSW and nPSH.

Outputs of this process are (with X being replaced by 0 or 1)

– the reference indices refIdxLX of the current prediction units.

For the derivation of refIdxLXA, the following applies.

– If the prediction unit covering luma location (xP-1, yP + nPSH – 1) is available and PredMode is not MODE_INTRA, refIdxLXA is assigned to refIdxLX[xP-1, yP + nPSH – 1].

– Otherwise, refIdxLXA is assigned to -1.

For the derivation of refIdxLXB, the following applies.

– If the prediction unit covering luma location (xP + nPSW – 1, yP-1) is available and PredMode is not MODE_INTRA, refIdxLXB is assigned to refIdxLX[xP + nPSW – 1, yP-1].

– Otherwise, refIdxLXB is assigned to -1.

For the derivation of refIdxLXC, the following applies.

– If the prediction unit covering luma location (xP+nPSW, yP-1) is available and PredMode is not MODE_INTRA, refIdxLXC is assigned to refIdxLX[xP+nPSW, yP-1].

– Otherwise if the prediction unit covering luma location (xP-1, yP+nPSH) is available and PredMode is not MODE_INTRA, refIdxLXC is assigned to refIdxLX[xP-1, yP+nPSH].

– Otherwise if the prediction unit covering luma location (xP-1, yP-1) is available and PredMode is not MODE_INTRA refIdxLXC is assigned to refIdxLX[xP-1, yP-1].

– Otherwise, refIdxLXC is assigned to -1.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000627

The variable refIdxLX is derived as follows.

- If refIdxLXA is equal to refIdxLXB and refIdxLXB is equal to refIdxLXC, the following applies.
  - If refIdxLXA is equal to -1 then refIdxLX = 0
  - Otherwise refIdxLX = refIdxLXA
- Otherwise if refIdxLXA is equal to refIdxLXB, the following applies.
  - If refIdxLXA is equal to -1 then refIdxLX = refIdxLXC
  - Otherwise refIdxLX = refIdxLXA
- Otherwise if refIdxLXB is equal to refIdxLXC, the following applies.
  - If refIdxLXB is equal to -1 then refIdxLX = refIdxLXA
  - Otherwise refIdxLX = refIdxLXB
- Otherwise if refIdxLXA is equal to refIdxLXC, the following applies.
  - If refIdxLXA is equal to -1 then refIdxLX = refIdxLXB
  - Otherwise refIdxLX = refIdxLXA
- Otherwise if refIdxLXA is equal to -1, the following applies.
  - refIdxLX = min(refIdxLXB, refIdxLXC)
- Otherwise if refIdxLXB is equal to -1, the following applies.
  - refIdxLX = Min(refIdxLXA, refIdxLXC)
- Otherwise if refIdxLXC is equal to -1, the following applies.
  - refIdxLX = Min(refIdxLXA, refIdxLXB)
- Otherwise, the following applies.
  - refIdxLX = Min(Min(refIdxLXA, refIdxLXB), refIdxLXC)

8.4.2.1.4 Derivation process for combined bi-predictive merging candidates

Inputs of this process are

- a merging candidate list mergeCandList,
- reference indices refIdxL0N and refIdxL1N of every candidate N being in mergeCandList,
- prediction list utilization flags predFlagL0N and predFlagL1N of every candidate N being in mergeCandList,
- motion vectors mvL0N and mvL1N of every candidate N being in mergeCandList,
- the number of elements numMergeCand within mergeCandList,
- the number of elements numOrigMergeCand within the mergeCandList after the spatial and temporal merge candidate derivation process,
- the maximum number of elements maxNumMergeCand within the mergeCandList.

Outputs of this process are

- the merging candidate list mergeCandList,
- the number of elements numMergeCand within mergeCandList.
- reference indices refIdxL0combCand$_k$ and refIdxL1combCand$_k$ of every new candidate combCand$_k$ being added in mergeCandList during the invokation of this process,
- prediction list utilization flags predFlagL0combCand$_k$ and predFlagL1combCand$_k$ of every new candidate combCand$_k$ being added in mergeCandList during the invokation of this process,
- motion vectors mvL0combCand$_k$ and mvL1combCand$_k$ of every new candidate combCand$_k$ being added in mergeCandList during the invokation of this process,

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000628

The function RefPicOrderCnt(refidx, LX) specifies the value of PicOrderCnt of the picture that is the reference picture RefPicListX[refidx] of the current picture with X being 0 or 1. PicOrderCnt of the reference picture shall be maintained until the picture is marked as "non-existing". The variable PicOrderCnt specifies the PicOrderCnt of the current picture.

[Ed. (BB): PicOrderCnt(.) needs to be defined. Currently the Decoding process for picture order count is not described]

When numOrigMergeCand is greater than 1 and less than maxNumMergeCand, the variable numInputMergeCand is set to numMergeCand, the variables combIdx and combCnt are set to 0, the variable combStop is set to false and the following steps are repeated until combStop is equal to true.

1. The variables l0CandIdx and l1CandIdx are derived using combIdx as specified in Table 8-11.
2. The following assignments are made with l0Cand being the candidate at position l0CandIdx and l1Cand being the candidate at position l1CandIdx in the merging candidate list mergeCandList (l0Cand = mergeCandList[l0CandIdx] , l1Cand = mergeCandList[l1CandIdx]).
3. When all of the following conditions are true,
    1. predFlagL0l0Cand = = 1
    2. predFlagL1l1Cand = = 1
    3. RefPicOrderCnt(refIdxL0l0Cand, L0) != RefPicOrderCnt(refIdxL1l1Cand, L1) | | mvL0l0Cand != mvL1l1Cand

    the following applies.

    4. When no candidate in mergeCandList has the same list0 motion vector and reference index as l0Cand and the same list1 motion vector and reference index as l1Cand, the candidate $combCand_k$ with k equal to (numMergeCand – numInputMergeCand) is added at the end of mergeCandList (mergeCandList[numMergeCand] = $combCand_k$) and the reference indices, the prediction list utilization flags and the motion vectors of $combCand_k$ are dervied as follows and numMergeCand is incremented by 1.

        $refIdxL0combCand_k$ = refIdxL0l0Cand (8-92)

        $refIdxL1combCand_k$ = refIdxL1l1Cand (8-93)

        $predFlagL0combCand_k$ = 1 (8-94)

        $predFlagL1combCand_k$ = 1 (8-95)

        $mvL0combCand_k$[0] = mvL0l0Cand[0] (8-96)

        $mvL0combCand_k$[1] = mvL0l0Cand[1] (8-97)

        $mvL1combCand_k$[0] = mvL1l1Cand[0] (8-98)

        $mvL1combCand_k$[1] = mvL1l1Cand[1] (8-99)

        numMergeCand = numMergeCand + 1 (8-100)

    5. The variable combCnt is incremented by 1.
4. The variable combIdx is incremented by 1.
5. When combIdx is equal to (numOrigMergeCand * (numOrigMergeCand – 1)) or numMergeCand is equal to maxNumMergeCand or combCnt is equal to 5, combStop is set to true.

Table 8-11 – Specification of l0CandIdx and l1CandIdx

combIdx	0	1	2	3	4	5	6	7	8	9	10	11
l0CandIdx	0	1	0	2	1	2	0	3	1	3	2	3
l1CandIdx	1	0	2	0	2	1	3	0	3	1	3	2

8.4.2.1.5 Derivation process for non-scaled bi-predictive merging candidates

Inputs of this process are

– a merging candidate list mergeCandList,

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000629

– reference indices refIdxL0N and refIdxL1N of every candidate N being in mergeCandList,
– prediction list utilization flags predFlagL0N and predFlagL1N of every candidate N being in mergeCandList,
– motion vectors mvL0N and mvL1N of every candidate N being in mergeCandList,
– the number of elements numMergeCand within mergeCandList,
– the number of elements numOrigMergeCand within the mergeCandList after the spatial and temporal merge candidate derivation process,
– the maximum number of elements maxNumMergeCand within the mergeCandList.

Outputs of this process are

– the merging candidate list mergeCandList,
– the number of elements numMergeCand within mergeCandList.
– reference indices refIdxL0nscaleCand$_l$ and refIdxL10nscaleCand$_l$ of every new candidate nscaleCand$_l$ being added in mergeCandList during the invokation of this process,
– prediction list utilization flags predFlagL0nscaleCand$_l$ and predFlagL10nscaleCand$_l$ of every new candidate nscaleCand$_l$ being added in mergeCandList during the invokation of this process,
– motion vectors mvL0nscaleCand$_l$ and mvL10nscaleCand$_l$ of every new candidate nscaleCand$_l$ being added in mergeCandList during the invokation of this process,

The function RefPicOrderCnt(refidx, LX) specifies the value of PicOrderCnt of the picture that is the reference picture RefPicListX[refidx] of the current picture with X being 0 or 1. PicOrderCnt of the reference picture shall be maintained until the picture is marked as "non-existing". The variable PicOrderCnt specifies the PicOrderCnt of the current picture.

[Ed. (BB): PicOrderCnt(.) needs to be defined. Currently the Decoding process for picture order count is not described]

When numOrigMergeCand is greater than 0 and numMergeCand is less than maxNumMergeCand, the variable numInputMergeCand is set to numMergeCand, the variables origIdx and nscaleCnt are set to 0, the variable nscaleStop is set to false and the following steps are repeated until nscaleStop is equal to true.

1. The following assignments are made with origCand being the candidate at position origIdx in the merging candidate list mergeCandList (origCand = mergeCandList[origIdx]).
2. The following is applied one time with X being 0 and Y being 1. When nscaleCnt is less than 1 and numMergeCand is less than maxNumMergeCand, it is applied a second time with X being 1 and Y being 0.
    1. When all of the following conditions are true,
        – refIdxLXorigCand >= 0
        – RefPicOrderCnt(refIdxLXorigCand, LX) != RefPicOrderCnt(refIdxorigCandLX, LY)
        – abs(PicOrderCnt – RefPicOrderCnt(refIdxLXorigCand, LX)) = = abs(PicOrderCnt – RefPicOrderCnt(refIdxLXorigCand, LY))
        – refIdxLXorigCand < num_ref_idx_lY_active_minus1 + 1

        the following applies.

        – When no candidate in mergeCandList has a list0 motion vector equal to listX motion vector of origCand, a list1 motion vector equal to the listX motion vector of origCand multiplied by 1 and list0 and list1 reference indices equal to the listX reference index of origCand, the candidate nscaleCand$_l$ with l equal to (numMergeCand – numInputMergeCand)is added at the end of mergeCandList (mergeCandList[numMergeCand] = nscaleCand$_l$) and the reference indices, the prediction list utilization flags and the motion vectors of nscaleCand$_l$ are dervied as follows and numMergeCand is incremented by 1.

            refIdxL0nscaleCand$_l$ = refIdxLXorigCand (8-92)

            refIdxL1nscaleCand$_l$ = refIdxLXorigCand (8-93)

            predFlagL0nscaleCand$_l$ = 1 (8-94)

            predFlagL1nscaleCand$_l$ = 1 (8-95)

            mvL0nscaleCand$_l$[0] = mvLXorigCand[0] (8-96)

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000630

$mvL0nscaleCand_l[\ 1\ ] = mvLXorigCand[\ 1\ ]$ (8-97)

$mvL1nscaleCand_l[\ 0\ ] = -\ mvLXorigCand[\ 0\ ]$ (8-98)

$mvL1nscaleCand_l[\ 1\ ] = -\ mvLXorigCand[\ 1\ ]$ (8-99)

$numMergeCand = numMergeCand + 1$ (8-100)

– The variable nscaleCnt is incremented by 1.

3. The variable origIdx is incremented by 1.
4. When origIdx is equal to numOrigMergeCand or numMergeCand is equal to maxNumMergeCand or nscaleCnt is equal to 1, nscaleStop is set to true.

8.4.2.1.6 Derivation process for zero motion vector merging candidates

Inputs of this process are

– a merging candidate list mergeCandList,
– reference indices refIdxL0N and refIdxL1N of every candidate N being in mergeCandList,
– prediction list utilization flags predFlagL0N and predFlagL1N of every candidate N being in mergeCandList,
– motion vectors mvL0N and mvL1N of every candidate N being in mergeCandList,
– the number of elements numMergeCand within mergeCandList,
– the maximum number of elements maxNumMergeCand within the mergeCandList.

Outputs of this process are

– the merging candidate list mergeCandList,
– the number of elements numMergeCand within mergeCandList.
– reference indices $refIdxL0zeroCand_m$ and $refIdxL10zeroCand_m$ of every new candidate $zeroCand_m$ being added in mergeCandList during the invokation of this process,
– prediction list utilization flags $predFlagL0zeroCand_m$ and $predFlagL10zeroCand_m$ of every new candidate $zeroCand_m$ being added in mergeCandList during the invokation of this process,
– motion vectors $mvL0zeroCand_m$ and $mvL10zeroCand_m$ of every new candidate $zeroCand_m$ being added in mergeCandList during the invokation of this process,

The variable numRefIdx is derived as follows.

– If slice_type is equal to P, numRefIdx is set to num_ref_idx_l0_active_minus1 + 1
– Otherwise (slice_type is equal to B), numRefIdx is set to min(num_ref_idx_l0_active_minus1 + 1, num_ref_idx_l1_active_minus1 + 1)

When numMergeCand is less than maxNumMergeCand, the variable numInputMergeCand is set to numMergeCand, the variables zeroIdx and zeroCnt are set to 0, the variable zeroStop is set to false and the following steps are repeated until zeroStop is equal to true.

1. For the derivation of the reference indices, the prediction list utilization flags and the motion vectors of the zero motion vector merging candidate, the following applies.
   1. If all of the following conditions are true,
      – slice_type is equal to P
      – no candidate in mergeCandList has a list0 motion vector equal to (0,0) and reference index equal to zeroIdx or (zeroCnt is equal to 2 and zeroIdx is equal to 0),

      the candidate $zeroCand_m$ with m equal to (numMergeCand − numInputMergeCand) is added at the end of mergeCandList ($mergeCandList[\ numMergeCand\ ] = zeroCand_m$) and the reference indices, the prediction list utilization flags and the motion vectors of $zeroCand_m$ are dervied as follows and numMergeCand is incremented by 1.

      $refIdxL0zeroCand_m = zeroIdx$ (8-110)

      $refIdxL1zeroCand_m = -1$ (8-111)

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000631

$predFlagL0zeroCand_m = 1$ (8-112)

$predFlagL1zeroCand_m = 0$ (8-113)

$mvL0zeroCand_m[0] = 0$ (8-114)

$mvL0zeroCand_m[1] = 0$ (8-115)

$mvL1zeroCand_m[0] = 0$ (8-116)

$mvL1zeroCand_m[1] = 0$ (8-117)

numMergeCand = numMergeCand + 1 (8-118)

2. Otherwise if all of the following conditions are true,
   - slice_type is equal to B
   - no candidate in mergeCandList has a list0 motion vector equal to (0,0), a list0 reference index equal to zeroIdx, a list1 motion vector equal to (0,0) and a list1 reference index equal to zeroIdx or (zeroCnt is equal to 2 and zeroIdx is equal to 0)

   the candidate $zeroCand_m$ with m equal to (numMergeCand – numInputMergeCand) is added at the end of mergeCandList (mergeCandList[numMergeCand] = $zeroCand_m$) and the reference indices, the prediction list utilization flags and the motion vectors of $zeroCand_m$ are dervied as follows and numMergeCand is incremented by 1.

$refIdxL0zeroCand_m = zeroIdx$ (8-119)

$refIdxL1zeroCand_m = zeroIdx$ (8-120)

$predFlagL0zeroCand_m = 1$ (8-121)

$predFlagL1zeroCand_m = 1$ (8-122)

$mvL0zeroCand_m[0] = 0$ (8-123)

$mvL0zeroCand_m[1] = 0$ (8-124)

$mvL1zeroCand_m[0] = 0$ (8-125)

$mvL1zeroCand_m[1] = 0$ (8-126)

numMergeCand = numMergeCand + 1 (8-127)

2. The variables zeroCnt and zeroIdx are modifed as follows.
   3. If zeroIdx is equal to numRefIdx or zeroCnt is equal to 2, zeroIdx is set to 0.
   4. Otherwise (zeroIdx is less than and zeroCnt is less than 2), zeroIdx and zeroCnt are incremented by 1.
3. When numMergeCand is equal to maxNumMergeCand, zeroStop is set to true.

8.4.2.1.7 Derivation process for luma motion vector prediction

Inputs to this process are

- a luma location (xP, yP) specifying the top-left luma sample of the current prediction unit relative to the top-left sample of the current picture,
- variables specifying the width and the height of the prediction unit for luma, nPSW and nPSH.
- the reference index of the current prediction unit partition refIdxLX (with X being 0 or 1).

Output of this process is

- the prediction mvpLX of the motion vector mvLX (with X being 0 or 1).

The motion vector predictor mvpLX is derived in the following ordered steps.

1. The derivation process for motion vector predictor candidates from neighboring prediction unit partitions in subclause 8.4.2.1.6 is invoked with luma location (xP, yP), the width and the height of the prediction unit nPSW and nPSH, and refIdxLX (with X being 0 or 1, respectively) as inputs and the availability flags availableFlagLXN and the motion vectors mvLXN with N being replaced by A, B as the output.
2. The derivation process for temporal luma motion vector prediction in subclause 8.4.2.1.7 is invoked with luma location (xP, yP) , the width and the height of the prediction unit nPSW and nPSH, and refIdxLX (with X being

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000632

0 or 1, respectively) as the inputs and with the output being the availability flag availableFlagLXCol and the temporal motion vector predictor mvLXCol.

3. The motion vector predictor candidate list, mvpListLX, is constructed as follows.
    1. mvLXA, if availableFlagLXA is equal to 1
    2. mvLXB, if availableFlagLXB is equal to 1
    3. mvLXCol, if availableFlagLXCol is equal to 1
4. When motion vectors have the same value, the motion vectors are removed from the list except the motion vector which has the smallest order in the mvpListLX.
5. When mvpListLX is empty, a zero motion vector is added as follows.

$$mvpListLX[\ 0\ ][0] = 0 \tag{8-128}$$

$$mvpListLX[\ 0\ ][1] = 0 \tag{8-129}$$

6. The variable numMVPCandLX is set to the number of elements within the mvpListLX and maxNumMVPCand is set to 2.
7. The motion vector predictor list is modifed to contain exactly maxNumMVPCand motion vector predictor candidates as follows.
    1. If numMVPCandLX is less than maxNumMVPCand, the derivation process for zero motion vector predictor candidates specified in subclause 8.4.2.1.10 is invoked with mvpListLX and numMVPCandLX given as input and the output is assigned to mvpListLX and numMVPCandLX.
    2. Otherwise (numMVPCandLX is equal to or greater than maxNumMVPCand), all motion vector predictor candidates mvpListLX[idx] with idx greater than maxNumMVPCand – 1 are removed from the list.
8. The motion vector of mvpListLX[mvp_idx_lX[xP, yP]] is assigned to mvpLX.

8.4.2.1.8 Derivation process for motion vector predictor candidates

Inputs to this process are

– a luma location (xP, yP) specifying the top-left luma sample of the current prediction unit relative to the top-left sample of the current picture,

– variables specifying the width and the height of the prediction unit for luma, nPSW and nPSH,

– the reference index of the current prediction unit partition refIdxLX (with X being 0 or 1).

Outputs of this process are (with N being replaced by A, or B)

– the motion vectors mvLXN of the neighbouring prediction units,

– the availability flags availableFlagLXN of the neighbouring prediction units.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000633

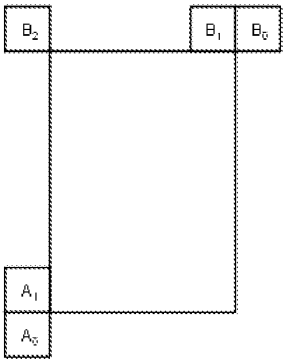


Figure 8-5 – Spatial motion vector neighbours

The function RefPicOrderCnt(pic, refidx, LX) is specified by the value of PicOrderCnt of the picture that is the reference picture RefPicListX[refidx] of pic with X being 0 or 1. PicOrderCnt of the reference picture shall be maintained until the picture is marked as "non-existing"

[Ed. (BB): PicOrderCnt(.) needs to be defined. Currently the Decoding process for picture order count is not described]

The variable isScaledFlagLX with X being 0 or 1 is set equal to 0.

The motion vector mvLXA and the availability flag availableFlagLXA are derived in the following ordered steps:

1. Let a set of two sample locations be (xA_k, yA_k), with k = 0, 1, specifies sample locations with $xA_k = xP - 1$, $yA_0 = yP + nPSH$ and $yA_1 = yA_0$ - MinPuSize. The set of sample locations (xA_k, yA_k) represent the sample locations immediately to the left side of the left partition boundary and it's extended line. [Ed.: (WJ) MinPuSize should be defined somewhere]

2. Let the availability flag availableFlagLXA be initially set equal to 0 and the both components of mvLXA are set equal to 0.

3. For (xA_k, yA_k) from (xA_0, yA_0) to (xA_1, yA_1) where $yA_1 = yA_0$ - MinPuSize, the following applies repeatedly until availableFlagLXA is equal to 1:

   – If the prediction unit covering luma location (xA_k, yA_k) is available, PredMode is not MODE_INTRA, predFlagLX[xA_k][yA_k] is equal to 1 and the reference index refIdxLX[xA_k][yA_k] is equal to the reference index of the current prediction unit refIdxLX, availableFlagLXA is set equal to 1 and the motion vector mvLXA is set equal to the motion vector mvLX[xA_k][yA_k], refIdxA is set equal to refIdxLX[xA_k][yA_k] and ListA is set equal to LX.

   – Otherwise if the prediction unit covering luma location (xAk, yAk) is available, PredMode is not MODE_INTRA, predFlagLY[xA_k][yA_k] (with Y = !X) is equal to 1 and RefPicOrderCnt(currPic, refIdxLY[xA_k][yA_k], LY) is equal to RefPicOrderCnt(currPic, refIdxLX, LX), availableFlagLXA is set equal to 1, the motion vector mvLXA is set equal to the motion vector mvLY[xA_k][yA_k], refIdxA is set equal to refIdxLY[xA_k][yA_k] , ListA is set equal to LY and mvLXA is set equal to mvLXA.

4. When availableFlagLXA is equal to 0, for (xA_k, yA_k) from (xA_0, yA_0) to (xA_1, yA_1) where $yA_1 = yA_0$ - MinPuSize, the following applies repeatedly until availableFlagLXA is equal to 1:

   – If the prediction unit covering luma location (xA_k, yA_k) is available, PredMode is not MODE_INTRA, predFlagLX[xA_k][yA_k] is equal to 1, availableFlagLXA is set equal to 1, the motion vector mvLXA is set equal to the motion vector mvLX[xA_k][yA_k], refIdxA is set equal to refIdxLX[xA_k][yA_k], ListA is set equal to LX.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000634

– Otherwise if the prediction unit covering luma location (xA_k, yA_k) is available, PredMode is not MODE_INTRA, predFlagLY[xA_k][yA_k] (with Y = !X) is equal to 1, availableFlagLXA is set equal to 1, the motion vector mvLXA is set equal to the motion vector mvLY[xA_k][yA_k], refIdxA is set equal to refIdxLY[xA_k][yA_k], ListA is set equal to LY.

– When availableFlagLXA is equal to 1, isScaledFlagLX is set equal to 1 and mvLXA is derived as specified below.

$$tx = (16384 + Abs(td / 2)) / td \quad (8\text{-}130)$$

$$DistScaleFactor = Clip3(-1024, 1023, (tb * tx + 32) >> 6) \quad (8\text{-}131)$$

$$mvLXA = ClipMv(Sign(DistScaleFactor * mvLXA) * ((Abs(DistScaleFactor * mvLXA) + 127) >> 8)) \quad (8\text{-}132)$$

[Ed. (WJ): software has a clip function for scaled motion vector. Do we need this? AVC does not have it]

where td and tb are derived as

$$td = Clip3(-128, 127, PicOrderCnt(currPic) – RefPicOrderCnt(currPic , refIdxA, ListA)) \quad (8\text{-}133)$$

$$tb = Clip3(-128, 127, PicOrderCnt(currPic) – RefPicOrderCnt(currPic, refIdxLX, LX)) \quad (8\text{-}134)$$

The motion vector mvLXB and the availability flag availableFlagLXB are derived in the following ordered steps:

1. Let a set of three sample location (xB_k, yB_k), with k = 0,1,2, specifies sample locations with $xB_0 = xP + nPSW$, $xB_1 = xB_0$ MinPuSize , $xB_2 = xP$ - MinPuSize and $yB_k = yP – 1$. The set of sample locations (xB_k, yB_k) represent the sample locations immediately to the upper side of the above partition boundary and it's extended line. [Ed.: (WJ) MinPuSize should be defined somewhere]

2. Let the availability flag availableFlagLXB be initially set equal to 0 and the both components of mvLXB are set equal to 0.

3. For (xB_k, yB_k) from (xB_0, yB_0) to (xB_2, yB_2) where $xB_0 = xP +nPSW$, $xB_1 = xB_0$ - MinPuSize , and $xB_2 =$ xP - MinPuSize, the following applies repeatedly until availableFlagLXB is equal to 1:

   – If the prediction unit covering luma location (xB_k, yB_k) is available, PredMode is not MODE_INTRA, predFlagLX[xB_k][yB_k] is equal to 1, and the reference index refIdxLX[xB_k][yB_k] is equal to the reference index of the current prediction unit refIdxLX, availableFlagLXB is set equal to 1 and the motion vector mvLXB is set equal to the motion vector mvLX[xB_k][yB_k], refIdxB is set equal to refIdxLX[xB_k][yB_k] and ListB is set equal to LX.

   – Otherwise if the prediction unit covering luma location (xB_k, yB_k) is available, PredMode is not MODE_INTRA, predFlagLY[xB_k][yB_k] (with Y = !X) is equal to 1 and RefPicOrderCnt(currPic , refIdxLY[xB_k][yB_k], LY) is equal to RefPicOrderCnt(currPic, refIdxLX, LX), availableFlagLXB is set equal to 1, the motion vector mvLXB is set equal to the motion vector mvLY[xB_k][yB_k], refIdxB is set equal to refIdxLY[xB_k][yB_k], and ListB is set equal to LY.

4. When availableFlagLXA or availableFlagLXB is equal to 0 and isScaledFlagLX is equal to 0, availableFlagLXB is set equal to 0 and for (xB_k, yB_k) from (xB_0, yB_0) to (xB_2, yB_2) where $xB_0 = xP +nPSW$, $xB_1 = xB_0$ - MinPuSize , and $xB_2 =$ xP - MinPuSize, the following applies repeatedly until availableFlagLXB is equal to 1:

   – If the prediction unit covering luma location (xB_k, yB_k) is available, PredMode is not MODE_INTRA, predFlagLX[xB_k][yB_k] is equal to 1, availableFlagLXB is set equal to 1, the motion vector mvLXB is set equal to the motion vector mvLX[xB_k][yB_k], refIdxB is set equal to refIdxLX[xB_k][yB_k], ListB is set equal to LX.

   – Otherwise if the prediction unit covering luma location (xB_k, yB_k) is available, PredMode is not MODE_INTRA, predFlagLY[xB_k][yB_k] (with Y = !X) is equal to 1, availableFlagLXB is set equal to 1, the motion vector mvLXB is set equal to the motion vector mvLY[xB_k][yB_k], refIdxB is set equal to refIdxLY[xB_k][yB_k], ListB is set equal to LY.

   – When availableFlagLXB is equal to 1 and RefPicOrderCnt(currPic , refIdxB, ListB) is not equal to RefPicOrderCnt(currPic, refIdxLX, LX),, mvLXB is derived as specified below.

$$tx = (16384 + Abs(td / 2)) / td \quad (8\text{-}135)$$

$$DistScaleFactor = Clip3(-1024, 1023, (tb * tx + 32) >> 6) \quad (8\text{-}136)$$

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000635

mvLXB = ClipMv(Sign(DistScaleFactor * mvLXA) *
((Abs(DistScaleFactor * mvLXA) + 127) >> 8)) (8-137)

[Ed. (WJ): software has a clip function for scaled motion vector. Do we need this? AVC does not have it]

where td and tb are derived as

td = Clip3(-128, 127, PicOrderCnt(currPic) – RefPicOrderCnt(currPic , refIdxB, ListB)) (8-138)

tb = Clip3(-128, 127, PicOrderCnt(currPic) – RefPicOrderCnt(currPic, refIdxLX, LX)) (8-139)

8.4.2.1.9 Derivation process for temporal luma motion vector prediction

Inputs to this process are

– a luma location (xP, yP) specifying the top-left luma sample of the current prediction unit relative to the top-left sample of the current picture,

– variables specifying the width and the height of the prediction unit for luma, nPSW and nPSH,

– the reference index of the current prediction unit partition refIdxLX (with X being 0 or 1).

Outputs of this process are

– the motion vector prediction mvLXCol,

– the availability flag availableFlagLXCol.

The function RefPicOrderCnt(pic, refidx, LX) is specified by the value of PicOrderCnt of the picture that is the reference picture RefPicListX[refidx] of pic with X being 0 or 1. PicOrderCnt of the reference picture shall be maintained until the picture is marked as “non-existing”.

[Ed. (BB): PicOrderCnt(.) needs to be defined. Currently the Decoding process for picture order count is not described]

Depending on the values of slice_type and collocated_from_l0_flag, the variable colPic, specifying the picture that contains the co-located partition, is derived as follows.

– If slice_type is equal to B and collocated_from_l0_flag is equal to 0, the variable colPic specifies the picture that contains the co-located partition as specified by RefPicList1[0].

– Otherwise (slice_type is equal to B and collocated_from_l0_flag is equal to 1 or slice_type is equal to P) , the variable colPic specifies the picture that contains the co-located partition as specified by RefPicList0[0].

Variable colPu and its position (xPCol, yPCol) are derived in the following ordered steps:

1. Right-bottom luma position of the current prediction unit is defined by

   xPRb = xP + nPSW (8-140)

   yPRb = yP + nPSH (8-141)

2. The variable colPu is set as the prediction unit covering the modified position given by ((xPRb >> 4) << 4, (yPRb >> 4) << 4) inside the colPic.

3. If colPu is coded in an intra prediction mode or colPu is unavailable, the following applies.

   – Central luma position of the current prediction unit is defined by

     xPCtr = (xP + (nPSW >> 1) – 1 (8-142)

     yPCtr = (yP + (nPSH >> 1) – 1 (8-143)

   – The variable colPu is set as the prediction unit covering the modified position given by ((xPCtr >> 4) << 4, (yPCtr >> 4) << 4) inside the colPic.

4. (xPCol, yPCol) is set equal to the top-left luma sample of the colPu relative to the top-left luma sample of the colPic.

The variables mvLXCol and availableFlagLXCol are derived as follows.

– If colPu is coded in an intra prediction mode or colPu is unavailable, both components of mvLXCol are set equal to 0 and availableFlagLXCol is set equal to 0.

– Otherwise (colPu is not coded in an intra prediction mode and colPu is available), the variables mvCol and refIdxCol are derived as follows,

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000636

– If PredFlagL0[xPCol][yPCol] is equal to 0, the motion vector mvCol and the reference index refIdxCol are set equal to MvL1[xPCol][yPCol] and RefIdxL1[xPCol][yPCol], respectively.

– Otherwise (PredFlagL0[xPCol][yPCol] is equal to 1), the following applies.

  – If PredFlagL1[xPCol][yPCol] is equal to 0, the motion vector mvCol and the reference index refIdxCol are set equal to MvL0[xPCol][yPCol] and RefIdxL0[xPCol][yPCol], respectively.

  – Otherwise (PredFlagL1[xPCol][yPCol] is equal to 1), the following assignments are made.

    – If all PicOrderCnts of pictures in all reference picture lists are smaller than PicOrderCnt(currPic), the motion vector mvCol, the reference index refIdxCol and ListCol are set equal to MvLN[xPCol][yPCol], RefIdxColLN and LN, respectively, with N being value of RefIdxLX.

    – Otherwise (one of PicOrderCnt of pictures in all reference picture lists are larger than PicOrderCnt(currPic)), the motion vector mvCol, the reference index refIdxCol and ListCol are set equal to MvLN[xPCol][yPCol], RefIdxColLN and LN, respectively, with N being value of collocated_from_l0_flag.

and the variable availableFlagLXCol is set equal to 1 and the following applies.

– If PicOrderCnt(colPic) – RefPicOrderCnt(colPic, refIdxCol, ListCol) is equal to PicOrderCnt(currPic) - RefPicOrderCnt(currPic, refIdxLX, LX),

$$mvLXCol = mvCol \tag{8-144}$$

– Otherwise, mvLXCol is derived as scaled version of the motion vector mvCol as specified below

$$tx = (16384 + Abs(td / 2)) / td \tag{8-145}$$

$$DistScaleFactor = Clip3(-1024, 1023, (tb * tx + 32) >> 6) \tag{8-146}$$

$$mvLXCol = ClipMv(Sign(DistScaleFactor * mvLXA) * ((Abs(DistScaleFactor * mvLXA) + 127) >> 8)) \tag{8-147}$$

[Ed. (WJ): software has a clip function for scaled motion vector. Do we need this? AVC does not have it]

where td and tb are derived as

$$td = Clip3(-128, 127, PicOrderCnt(colPic) – RefPicOrderCnt(colPic, refIdxCol, ListCol)) \tag{8-148}$$

$$tb = Clip3(-128, 127, PicOrderCnt(currPic) – RefPicOrderCnt(currPic, refIdxLX, LX)) \tag{8-149}$$

8.4.2.1.10 Derivation process for zero motion vector predictor candidates

Inputs of this process are

– a motion vector predictor list mvpListLX,

– the number of elements numMVPCandLX within mvpListLX.

Outputs of this process are

– the motion vector predictor list mvpListLX,

– the number of elements numMVPCandLX within mvpListLX.

When no motion vector in mvpListLX is equal to (0,0), the zero motion vector predictor candidate is added at the end of mvpListLX and numMVPCandLX is incremented by 1 as follows.

$$mvpListLX[numMVPCandLX][0] = 0 \tag{8-150}$$

$$mvpListLX[numMVPCandLX][1] = 0 \tag{8-151}$$

$$numMVPCandLX = numMVPCandLX + 1 \tag{8-152}$$

8.4.2.1.11 Derivation process for chroma motion vectors

[Ed.: (WJ) 4:2:0 assumption yet]

Inputs to this process are a luma motion vector mvLX and a reference index refIdLX.

Output of this process is a chroma motion vector mvCLX.

A chroma motion vector is derived from the corresponding luma motion vector.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000637

For the derivation of the chroma motion vector mvCLX, the following applies.

mvCLX[0] = mvLX[0] (8-153)

mvCLX[1] = mvLX[1] (8-154)

8.4.2.2 Decoding process for inter prediction samples

Inputs to this process are:

- a luma location (xC, yC) specifying the top-left luma sample of the current coding unit relative to the top left luma sample of the current picture,
- a luma location (xB, yB) specifying the top-left luma sample of the current prediction unit relative to the top-left luma sample of the current coding unit,
- a variable nCS specifying the size of the current coding unit,
- variables specifying the width and the height of the prediction unit, nPSW and nPSH,
- luma motion vectors mvL0 and mvL1, and chroma motion vectors mvCL0 and mvCL1,
- reference indices refIdxL0 and refIdxL1,
- prediction list utilization flags, predFlagL0 and predFlagL1.

Outputs of this process are:

- a (nCS_L)x(nCS_L) array $predSamples_L$ of luma prediction samples, where nCS_L is derived as specified below,
- a (nCS_C)x(nCS_C) array $preSamples_{Cb}$ of chroma prediction samples for the component Cb, where nCS_C is derived as specified below,
- a (nCS_C)x(nCS_C) array $predSamples_{Cr}$ of chroma residual samples for the component Cr, where nCS_C is derived as specified below.

The variable nCS_L is set equal to nCS and the variable nCS_C is set equal to nCS >> 1. [Ed: (WJ) revisit for supporting other chroma formats]

Let $predSamplesL0_L$ and $predSamplesL1_L$ be (nPSW)x(nPSH) arrays of predicted luma sample values and $predSampleL0_{Cb}$, $predSampleL1_{Cb}$, $predSampleL0_{Cr}$, and $predSampleL1_{Cr}$ be (nPSW/2)x(nPSH/2) arrays of predicted chroma sample values.

For LX being replaced by either L0 or L1 in the variables predFlagLX, RefPicListX, refIdxLX, refPicLX, and predPartLX, the following is specified.

When predFlagLX is equal to 1, the following applies.

- The reference picture consisting of an ordered two-dimensional array $refPicLX_L$ of luma samples and two ordered two-dimensional arrays $refPicLX_{Cb}$ and $refPicLX_{Cr}$ of chroma samples is derived by invoking the process specified in subclause 8.4.2.2.1 with refIdxLX and RefPicListX given as input.
- The arrays $predSamplesLX_L$, $predSamplesLX_{Cb}$, and $predSamplesLX_{Cr}$ are derived by invoking the fractional sample interpolation process specified in subclause 8.4.2.2.2 with the luma locations (xC, yC), (xB, yB), the width an the height of the current prediction unit nPSW, nPSH, the motion vectors mvLX, mvCLX, and the reference arrays with $refPicLX_L$, $refPicLX_{Cb}$ and $refPicLX_{Cr}$ given as input.

The array $predSample_L$ of the prediction samples of luma component is derived by invoking the weighted sample prediction process specified in subclause 8.4.2.2.3 with the luma location (xB, yB), the width an the height of the current prediction unit nPSW, nPSH, and the sample arrays $predSamplesL0_L$ and $predSamplesL1_L$ as well as predFlagL0, predFlagL1 and $BitDepth_Y$ given as input.

For C being replaced by Cb, or Cr, the array $predSample_C$ of the prediction samples of component C is derived by invoking the weighted sample prediction process specified in subclause 8.4.2.2.3 with the chroma location (xB/2, yB/2), the width an the height of the current prediction unit $nPSW_C$ set equal to nPSW/2, $nPSH_C$ set equal to nPSH/2, and the sample arrays $predSamplesL0_C$ and $predSamplesL1_C$ as well as predFlagL0, predFlagL1 and $BitDepth_C$ given as input.

8.4.2.2.1 Reference picture selection process

[Ed: (WJ) same as AVC]

8.4.2.2.2 Fractional sample interpolation process

Inputs to this process are:

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000638

– a luma location (xC, yC) specifying the top-left luma sample of the current coding unit relative to the top left luma sample of the current picture,

– a luma location (xB, yB) specifying the top-left luma sample of the current prediction unit relative to the top left luma sample of the current coding unit,

– the width and height of this prediction unit, nPSW and nPSH, in luma-sample units,

– a luma motion vector mvLX given in quarter-luma-sample units,

– a chroma motion vector mvCLX given in eighth-chroma-sample units,

– the selected reference picture sample arrays $refPicLX_L$, $refPicLX_{Cb}$, and $refPicLX_{Cr}$.

Outputs of this process are:

– a (nPSW)x(nPSH) array $predSampleLX_L$ of prediction luma sample values,

– two (nPSW/2)x(nPSH/2) arrays $predSampleLX_{Cb}$, and $predSampleLX_{Cr}$ of prediction chroma sample values.

The location (xP, yP) given in full-sample units of the upper-left luma samples of the current prediction unit relative to the upper-left luma sample location of the given reference sample arrays is derived by

$$xP = xC + xB \quad (8\text{-}87)$$
$$yP = yC + yB \quad (8\text{-}88)$$

Let ($xInt_L$, $yInt_L$) be a luma location given in full-sample units and ($xFrac_L$, $yFrac_L$) be an offset given in quarter-sample units. These variables are used only inside this subclause for specifying general fractional-sample locations inside the reference sample arrays $refPicLX_L$, $refPicLX_{Cb}$, and $refPicLX_{Cr}$.

For each luma sample location ($0 <= x_L < nPSW, 0 <= y_L < nPSH$) inside the prediction luma sample array $predSampleLX_L$, the corresponding prediction luma sample value $predSampleLX_L[x_L, y_L]$ is derived as follows:

– The variables $xInt_L$, $yInt_L$, $xFrac_L$, and $yFrac_L$ are derived by

$$xInt_L = xP + (mvLX[0] >> 2) + x_L \quad (8\text{-}89)$$
$$yInt_L = yP + (mvLX[1] >> 2) + y_L \quad (8\text{-}90)$$

$$xFrac_L = mvLX[0] \& 3 \quad (8\text{-}91)$$
$$yFrac_L = mvLX[1] \& 3 \quad (8\text{-}92)$$

– The prediction luma sample value $predSampleLX_L[x_L, y_L]$ is derived by invoking the process specified in subclause 8.4.2.2.2.1 with ($xInt_L$, $yInt_L$), ($xFrac_L$, $yFrac_L$) and $refPicLX_L$ given as input.

Let ($xInt_C$, $yInt_C$) be a chroma location given in full-sample units and ($xFrac_C$, $yFrac_C$) be an offset given in one-eighth sample units. These variables are used only inside this subclause for specifying general fractional-sample locations inside the reference sample arrays $refPicLX_{Cb}$ and $refPicLX_{Cr}$.

For each chroma sample location ($0 <= x_C < nPSW/2, 0 <= y_C < nPSH/2$) inside the prediction chroma sample arrays $predSampleLX_{Cb}$ and $predSampleLX_{Cr}$, the corresponding prediction chroma sample values $predSampleLX_{Cb}[x_C, y_C]$ and $predSampleLX_{Cr}[x_C, y_C]$ are derived asfollows:

– The variables $xInt_C$, $yInt_C$, $xFrac_C$, and $yFrac_C$ are derived by

$$xInt_C = (xP / 2) + (mvCLX[0] >> 3) + x_C \quad (8\text{-}93)$$
$$yInt_C = (yP / 2) + (mvCLX[1] >> 3) + y_C \quad (8\text{-}94)$$

$$xFrac_C = mvLX[0] \& 7 \quad (8\text{-}95)$$
$$yFrac_C = mvLX[1] \& 7 \quad (8\text{-}96)$$

– The prediction sample value $predSampleLX_{Cb}[x_C, y_C]$ is derived by invoking the process specified in subclause 8.4.2.2.2.2 with ($xInt_C$, $yInt_C$), ($xFrac_C$, $yFrac_C$) and $refPicLX_{Cb}$ given as input.

– The prediction sample value $predSampleLX_{Cr}[x_C, y_C]$ is derived by invoking the process specified in subclause 8.4.2.2.2.2 with ($xInt_C$, $yInt_C$), ($xFrac_C$, $yFrac_C$) and $refPicLX_{Cr}$ given as input.

8.4.2.2.2.1 Luma sample interpolation process

Inputs to this process are:

– a luma location in full-sample units ($xInt_L$, $yInt_L$),

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000639

– a luma location in fractional-sample units ($xFrac_L$, $yFrac_L$),

– the luma reference sample array $refPicLX_L$.

Output of this process is a predicted luma sample value $predSampleLX_L[x_L, y_L]$

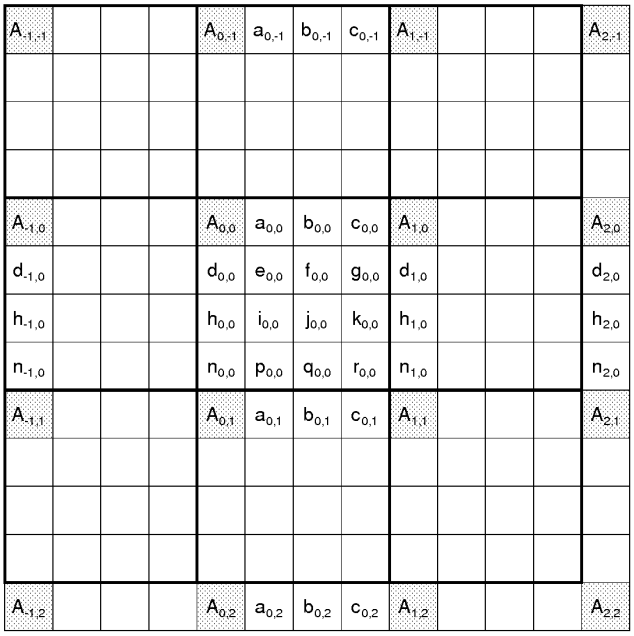


Figure 8-6 – Integer samples (shaded blocks with upper-case letters) and fractional sample positions (un-shaded blocks with lower-case letters) for quarter sample luma interpolation

In Figure 8-6, the positions labelled with upper-case letters $A_{i,j}$ within shaded blocks represent luma samples at full-sample locations inside the given two-dimensional array $refPicLX_L$ of luma samples. These samples may be used for generating the predicted luma sample value $predSampleLX_L[x_L, y_L]$. The locations ($xA_{i,j}$, $yA_{i,j}$) for each of the corresponding luma samples $A_{i,j}$ inside the given array $refPicLX_L$ of luma samples are derived as follows:

$$xA_{i,j} = Clip3(0, PicWidthInSamples_L - 1, xInt_L + i) \quad (8\text{-}97)$$
$$yA_{i,j} = Clip3(0, PicHeightInSamples_L - 1, yInt_L + j) \quad (8\text{-}98)$$

Variables shift1, shift2 and shift3 are derived as follows.

– The variable shift1 is set equal to $BitDepth_Y - 8$, the variable shift2 is set equal to 6, and the variable shift3 is set equal to $14 - BitDepth_Y$.

Given the luma samples $A_{i,j}$ at full-sample locations ($xA_{i,j}$, $yA_{i,j}$), the luma samples '$a_{0,0}$' to '$r_{0,0}$' at fractional sample positions are derived by the following rules.

– The samples labelled $a_{0,0}$, $b_{0,0}$, $c_{0,0}$, $d_{0,0}$, $h_{0,0}$, and $n_{0,0}$ shall be derived by applying the 8-tap filter to the nearest integer position samples:

$$a_{0,0} = (-A_{-3,0} + 4*A_{-2,0} - 10*A_{-1,0} + 57*A_{0,0} + 19*A_{1,0} - 7*A_{2,0} + 3*A_{3,0} - A_{4,0}) >> shift1 \quad (8\text{-}99)$$

$$b_{0,0} = (-A_{-3,0} + 4*A_{-2,0} - 11*A_{-1,0} + 40*A_{0,0} + 40*A_{1,0} - 11*A_{2,0} + 4*A_{3,0} - A_{4,0}) >> shift1 \quad (8\text{-}100)$$

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000640

$$c_{0,0} = (-A_{-3,0} + 3*A_{-2,0} - 7*A_{-1,0} + 19*A_{0,0} + 57*A_{1,0} - 10*A_{2,0} + 4*A_{3,0} - A_{4,0}) >> shift1 \quad (8\text{-}101)$$

$$d_{0,0} = (-A_{0,-3} + 4*A_{0,-2} - 10*A_{0,-1} + 57*A_{0,0} + 19*A_{0,1} - 7*A_{0,2} + 3*A_{0,3} - A_{0,4}) >> shift1 \quad (8\text{-}102)$$

$$h_{0,0} = (-A_{0,-3} + 4*A_{0,-2} - 11*A_{0,-1} + 40*A_{0,0} + 40*A_{0,1} - 11*A_{0,2} + 4*A_{0,3} - A_{0,4}) >> shift1 \quad (8\text{-}103)$$

$$n_{0,0} = (-A_{0,-3} + 3*A_{0,-2} - 7*A_{0,-1} + 19*A_{0,0} + 57*A_{0,1} - 10*A_{0,2} + 4*A_{0,3} - A_{0,4}) >> shift1 \quad (8\text{-}104)$$

– The samples labelled $e_{0,0}$, $f_{0,0}$, $g_{0,0}$, $i_{0,0}$, $j_{0,0}$, $k_{0,0}$, $p_{0,0}$, $q_{0,0}$ and $r_{0,0}$ shall be derived by applying the 8-tap filter to the samples $a_{0,i}$, $b_{0,i}$ and $c_{0,i}$ where $i = -3..4$ in vertical direction:

$$e_{0,0} = (-a_{0,-3} + 4*a_{0,-2} - 10*a_{0,-1} + 57*a_{0,0} + 19*a_{0,1} - 7*a_{0,2} + 3*a_{0,3} - a_{0,4}) >> shift2 \quad (8\text{-}108)$$

$$f_{0,0} = (-a_{0,-3} + 4*a_{0,-2} - 11*a_{0,-1} + 40*a_{0,0} + 40*a_{0,1} - 11*a_{0,2} + 4*a_{0,3} - a_{0,4}) >> shift2 \quad (8\text{-}109)$$

$$g_{0,0} = (-a_{0,-3} + 3*a_{0,-2} - 7*a_{0,-1} + 19*a_{0,0} + 57*a_{0,1} - 10*a_{0,2} + 4*a_{0,3} - a_{0,4}) >> shift2 \quad (8\text{-}110)$$

$$i_{0,0} = (-b_{0,-3} + 4*b_{0,-2} - 10*b_{0,-1} + 57*b_{0,0} + 19*b_{0,1} - 7*b_{0,2} + 3*b_{0,3} - b_{0,4}) >> shift2 \quad (8\text{-}111)$$

$$j_{0,0} = (-b_{0,-3} + 4*b_{0,-2} - 11*b_{0,-1} + 40*b_{0,0} + 40*b_{0,1} - 11*b_{0,2} + 4*b_{0,3} - b_{0,4}) >> shift2 \quad (8\text{-}112)$$

$$k_{0,0} = (-b_{0,-3} + 3*b_{0,-2} - 7*b_{0,-1} + 19*b_{0,0} + 57*b_{0,1} - 10*b_{0,2} + 4*b_{0,3} - b_{0,4}) >> shift2 \quad (8\text{-}113)$$

$$p_{0,0} = (-c_{0,-3} + 4*c_{0,-2} - 10*c_{0,-1} + 57*c_{0,0} + 19*c_{0,1} - 7*c_{0,2} + 3*c_{0,3} - c_{0,4}) >> shift2 \quad (8\text{-}114)$$

$$q_{0,0} = (-c_{0,-3} + 4*c_{0,-2} - 11*c_{0,-1} + 40*c_{0,0} + 40*c_{0,1} - 11*c_{0,2} + 4*c_{0,3} - c_{0,4}) >> shift2 \quad (8\text{-}115)$$

$$r_{0,0} = (-c_{0,-3} + 3*c_{0,-2} - 7*c_{0,-1} + 19*c_{0,0} + 57*c_{0,1} - 10*c_{0,2} + 4*c_{0,3} - c_{0,4}) >> shift2 \quad (8\text{-}116)$$

The positions labelled with lower-case letters within un-shaded blocks represent luma samples at quarter-pel sample fractional locations. The luma location offset in fractional-sample units ($xFrac_L$, $yFrac_L$) specifies which of the generated luma samples at full-sample and fractional-sample locations is assigned to the predicted luma sample value $predSampleLX_L[x_L, y_L]$. This assignment is done according to Table 8-9. The value of $predSampleLX_L[x_L, y_L]$ shall be the output.

Table 8-12 – Assignment of the luma prediction sample $predSampleLX_L[x_L, y_L]$

xFracL	0	0	0	0	1	1	1	1	2	2	2	2	3	3	3	3
yFracL	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3
$predSampleLX_L[x_L, y_L]$	A << shift3	d	h	n	a	e	i	p	b	f	j	q	c	g	k	r

8.4.2.2.2.2 Chroma sample interpolation process

Inputs to this process are:

– a chroma location in full-sample units ($xInt_C$, $yInt_C$),

– a chroma location in fractional-sample units ($xFrac_C$, $yFrac_C$),

– the chroma reference sample array $refPicLX_C$.

Output of this process is a predicted chroma sample value $predSampleLX_C[x_C, y_C]$

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000641

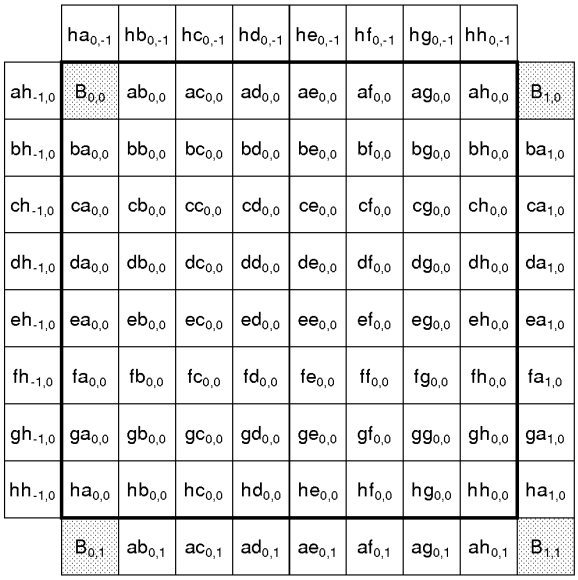


Figure 8-7 – Integer samples (shaded blocks with upper-case letters) and fractional sample positions (un-shaded blocks with lower-case letters) for eighth sample chroma interpolation

In Figure 8-5, the positions labelled with upper-case letters $B_{i,j}$ within shaded blocks represent chroma samples at full-sample locations inside the given two-dimensional array refPicLX$_C$ of chroma samples. These samples may be used for generating the predicted chroma sample value predSampleLX$_C$[x_C, y_C]. The locations ($xB_{i,j}$, $yB_{i,j}$) for each of the corresponding chroma samples $B_{i,j}$ inside the given array refPicLX$_C$ of chroma samples are derived as follows:

$$xB_{i,j} = \text{Clip3}(0, \text{PicWidthInSamples}_C - 1, \text{xInt}_C + i) \quad (8\text{-}117)$$
$$yB_{i,j} = \text{Clip3}(0, \text{PicHeightInSamples}_C - 1, \text{yInt}_C + j) \quad (8\text{-}118)$$

Variables shift1, shift2 and shift3 are derived as follows.

– The variable shift1 is set equal to BitDepth$_C$ – 8, the variable shift2 is set equal to BitDepth$_C$ – 2, and the variable shift3 is set equal to 14 – BitDepth$_C$.

Given the chroma samples $B_{i,j}$ at full-sample locations ($xB_{i,j}$, $yB_{i,j}$), the chroma samples '$ab_{0,0}$' to '$hh_{0,0}$' at fractional sample positions are derived by the following rules.

– The samples labelled $ab_{0,0}$, $ac_{0,0}$, $ad_{0,0}$, $ae_{0,0}$, $af_{0,0}$, $ag_{0,0}$, and $ah_{0,0}$ shall be derived by applying the 4-tap filter to the nearest integer position samples and clipping the filtered value:

$$ab_{0,0} = (-3*B_{-1,0} + 60*B_{0,0} + 8*B_{1,0} - B_{2,0}) >> \text{shift1} \quad (8\text{-}119)$$

$$ac_{0,0} = (-4*B_{-1,0} + 54*B_{0,0} + 16*B_{1,0} - 2*B_{2,0}) >> \text{shift1} \quad (8\text{-}120)$$

$$ad_{0,0} = (-5*B_{-1,0} + 46*B_{0,0} + 27*B_{1,0} - 4*B_{2,0}) >> \text{shift1} \quad (8\text{-}121)$$

$$ae_{0,0} = (-4*B_{-1,0} + 36*B_{0,0} + 36*B_{1,0} - 4*B_{2,0}) >> \text{shift1} \quad (8\text{-}122)$$

$$af_{0,0} = (-4*B_{-1,0} + 27*B_{0,0} + 46*B_{1,0} - 5*B_{2,0}) >> \text{shift1} \quad (8\text{-}123)$$

$$ag_{0,0} = (-2*B_{-1,0} + 16*B_{0,0} + 54*B_{1,0} - 4*B_{2,0}) >> \text{shift1} \quad (8\text{-}124)$$

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000642

$$ah_{0,0} = (-B_{-1,0} + 8*B_{0,0} + 60*B_{1,0} - 3*B_{2,0}) >> shift1 \quad (8\text{-}125)$$

– The samples labelled $ba_{0,0}$, $ca_{0,0}$, $da_{0,0}$, $ea_{0,0}$, $fa_{0,0}$, $ga_{0,0}$, and $ha_{0,0}$ shall be derived by applying the 4-tap filter to the nearest integer position samples and clipping the filtered value:

$$ba_{0,0} = (-3*B_{0,-1} + 60*B_{0,0} + 8*B_{0,1} - B_{0,2}) >> shift1 \quad (8\text{-}126)$$

$$ca_{0,0} = (-4*B_{0,-1} + 54*B_{0,0} + 16*B_{0,1} - 2*B_{0,2}) >> shift1 \quad (8\text{-}127)$$

$$da_{0,0} = (-5*B_{0,-1} + 46*B_{0,0} + 27*B_{0,1} - 4*B_{0,2}) >> shift1 \quad (8\text{-}128)$$

$$ea_{0,0} = (-4*B_{0,-1} + 36*B_{0,0} + 36*B_{0,1} - 4*B_{0,2}) >> shift1 \quad (8\text{-}129)$$

$$fa_{0,0} = (-4*B_{0,-1} + 27*B_{0,0} + 46*B_{0,1} - 5*B_{0,2}) >> shift1 \quad (8\text{-}130)$$

$$ga_{0,0} = (-2*B_{0,-1} + 16*B_{0,0} + 54*B_{0,1} - 4*B_{0,2}) >> shift1 \quad (8\text{-}131)$$

$$ha_{0,0} = (-B_{0,-1} + 8*B_{0,0} + 60*B_{0,1} - 3*B_{0,2}) >> shift1 \quad (8\text{-}132)$$

– The samples labelled $bX_{0,0}$, $cX_{0,0}$, $dX_{0,0}$, $eX_{0,0}$, $fX_{0,0}$, $gX_{0,0}$ and $hX_{0,0}$ for X being replaced by b, c, d, e, f, g and h, respectively, shall be derived by applying the 4-tap filter to the intermediate values $aX_{0,i}$ where $i = -1..2$ in vertical direction:

$$bX_{0,0} = (-3*aX_{0,-1} + 60*aX_{0,0} + 8*aX_{0,1} - aX_{0,2}) >> shift2 \quad (8\text{-}140)$$

$$cX_{0,0} = (-4*aX_{0,-1} + 54*aX_{0,0} + 16*aX_{0,1} - 2*aX_{0,2}) >> shift2 \quad (8\text{-}141)$$

$$dX_{0,0} = (-5*aX_{0,-1} + 46*aX_{0,0} + 27*aX_{0,1} - 4*aX_{0,2}) >> shift2 \quad (8\text{-}142)$$

$$eX_{0,0} = (-4*aX_{0,-1} + 36*aX_{0,0} + 36*aX_{0,1} - 4*aX_{0,2}) >> shift2 \quad (8\text{-}143)$$

$$fX_{0,0} = (-4*aX_{0,-1} + 27*aX_{0,0} + 46*aX_{0,1} - 5*aX_{0,2}) >> shift2 \quad (8\text{-}144)$$

$$gX_{0,0} = (-2*aX_{0,-1} + 16*aX_{0,0} + 54*aX_{0,1} - 4*aX_{0,2}) >> shift2 \quad (8\text{-}145)$$

$$hX_{0,0} = (-aX_{0,-1} + 8*aX_{0,0} + 60*aX_{0,1} - 3*aX_{0,2}) >> shift2 \quad (8\text{-}146)$$

The positions labelled with lower-case letters within un-shaded blocks represent chroma samples at eighth-pel sample fractional locations. The chroma location offset in fractional-sample units (xFracc, yFracc) specifies which of the generated chroma samples at full-sample and fractional-sample locations is assigned to the predicted chroma sample value $predSampleLX_C[x_C, y_C]$. This assignment is done according to Table 8-10. The value of $predSampleLX_C[x_C, y_C]$ shall be the output.

Table 8-13 – Assignment of the chroma prediction sample $predSampleLX_C[x_C, y_C]$ for (X, Y) being replaced by (1, b), (2, c), (3, d), (4, e), (5, f), (6, g), and (7, h), respectively

xFracC	0	0	0	0	0	0	0	0	X	X	X	X	X	X	X	X
yFracC	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
predSampleLXC[xC, yC]	B << shift 3	ba	ca	da	ea	fa	ga	ha	aY	bY	cY	dY	eY	fY	gY	hY

8.4.2.2.3 Weighted sample prediction process

[Ed.: (KU) only the default weighted prediction part is added for bi-directional averaging]

Inputs to this process are:

– a location (xB, yB) specifying the top-left sample of the current prediction unit relative to the top left sample of the current coding unit,

– the width and height of this prediction unit, nPSW and nPSH,

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000643

– two (nPSW)x(nPSH) arrays predSamplesL0 and predSamplesL1,

– prediction list utilization flags, predFlagL0 and predFlagL1,

– the bit-depth of the chroma component, bitDepth.

Outputs of this process are:

– the (nPSW)x(nPSH) array predSamples of prediction sample values.

Variables shift1, shift2, offset1 and offset2 are derived as follows.

– The variable shift1 is set equal to 14 – bitDepth and the variable shift2 is set equal to 15 – bitDepth,

– The variable offset1 is set equal to 1 << (shift1 – 1) and the variable offset2 is set equal to 1 << (shift2 – 1).

Depending on the value of predFlagL0 and predFlagL1, the prediction samples predSamples[x, y] with x = 0..(nPSW)-1 and y = 0..(nPSH)-1 are derived as follows.

– If predFlagL0 is equal to 1 and predFlagL1 is equal to 0,

$$predSamples[x, y] = (predSamplesL0[x, y] + offset1) >> shift1 \quad (8\text{-}147)$$

– Otherwise, if predFlagL0 is equal to 0 and predFlagL1 is equal to 1,

$$predSamples[x, y] = (predSamplesL1[x, y] + offset1) >> shift1 \quad (8\text{-}148)$$

– Otherwise (both predFlagL0 and predFlagL1 are equal to 1),

$$predSamples[x, y] = Clip3(0, (1 << bitDepth) – 1 , predSamplesL0[x, y] + predSamplesL1[x, y] + offset2) >> shift2) \quad (8\text{-}149)$$

8.4.3 Decoding process for the residual signal of coding units coded in inter prediction mode

Inputs to this process are:

– a luma location (xC, yC) specifying the top-left luma sample of the current coding unit relative to the top-left luma sample of the current picture,

– a variable log2CUSize specifying the size of the current coding unit.

Outputs of this process are:

– a (nCS_L)x(nCS_L) array $resSamples_L$ of luma residual samples, where nCS_L is derived as specified below,

– a (nCS_C)x(nCS_C) array $resSamples_{Cb}$ of chroma residual samples for the component Cb, where nCS_C is derived as specified below,

– a (nCS_C)x(nCS_C) array $resSamples_{Cr}$ of chroma residual samples for the component Cr, where nCS_C is derived as specified below.

The variable nCS_L is set equal to 1 << log2CUSize and the variable nCS_C is set equal to (1 << log2CUSize) >> 1.

Let $resSamples_L$ be a (nCS_L)x(nCS_L) array of luma residual samples and let $resSamples_{Cb}$ and $resSamples_{Cr}$ be two (nCS_C)x(nCS_C) arrays of chroma residual samples.

Depending on no_residual_data_flag, the following applies:

– If no_residual_data_flag is equal to 1, all samples of the (nCS_L)x(nCS_L) array $resSamples_L$ and all samples of the two (nCS_C)x(nCS_C) arrays $resSamples_{Cb}$ and $resSamples_{Cr}$ are set equal to 0.

– Otherwise (no_residual_data_flag is equal to 0), the following ordered steps apply:

1. The decoding process for luma residual blocks as specified in subclause 8.4.3.1 below is invoked with the luma location (xC, yC), the luma location (xB, yB) set equal to (0, 0), the variable log2TrafoWidth set equal to log2CUSize, the variable log2TrafoHeight set equal to log2CUSize, the variable trafoDepth set equal to 0, the variable nCS set equal to nCSL, and the (nCSL)x(nCSL) array resSamplesL as the inputs and the output is a modified version of the (nCSL)x(nCSL) array resSamplesL.

2. The decoding process for chroma residual blocks as specified in subclause 8.4.3.2 below is invoked with the luma location (xC, yC), the luma location (xB, yB) set equal to (0, 0), the variable log2TrafoWidth set equal to log2CUSize, the variable log2TrafoHeight set equal to log2CUSize, the variable trafoDepth set equal to 0, the

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000644

variable cIdx set equal to 1, the variable nCS set equal to nCS_C, and the (nCS_C)x(nCS_C) array $resSamples_{Cb}$ as the inputs and the output is a modified version of the (nCS_C)x(nCS_C) array $resSamples_{Cb}$.

3. The decoding process for chroma residual blocks as specified in subclause 8.4.3.2 below is invoked with the luma location (xC, yC), the luma location (xB, yB) set equal to (0, 0), the variable log2TrafoWidth set equal to log2CUSize, the variable log2TrafoHeight set equal to log2CUSize, the variable trafoDepth set equal to 0, the variable cIdx set equal to 2, the variable nCS set equal to nCS_C, and the (nCS_C)x(nCS_C) array $resSamples_{Cr}$ as the inputs and the output is a modified version of the (nCS_C)x(nCS_C) array $resSamples_{Cr}$.

8.4.3.1 Decoding process for luma residual blocks

Inputs to this process are:

- a luma location (xC, yC) specifying the top-left luma sample of the current coding unit relative to the top-left luma sample of the current picture,
- a luma location (xB, yB) specifying the top-left luma sample of the current block relative to the top-left luma sample of the current coding unit,
- a variable log2TrafoWidth specifying the width of the current block,
- a variable log2TrafoHeight specifying the height of the current block,
- a variable trafoDepth specifying the hierarchy depth of the current block relative to the coding unit,
- a variable nCS specifying the size, in luma samples, of the current coding unit,
- a (nCS)x(nCS) array resSamples of luma residual samples.

Output of this process is:

- a modified version of the (nCS)x(nCS) array of luma residual samples.

Depending split_transform_flag[xB][yB][trafoDepth], the following applies:

- If split_transform_flag[xB][yB][trafoDepth] is equal to 1, the following ordered steps apply:
  1. The variables xB1, yB1, xB2, yB2, xB3 and yB3 are derived as follows.
     - If InterTUSplitDirection is equal to 2, the following applies.
       - The variable xB1 is set equal to xB + ((1 << log2TrafoWidth) >> 1).
       - The variable yB2 is set equal to yB + ((1 << log2TrafoHeight) >> 1).
       - The variable xB3 is set equal to xB1.
       - The variable yB3 is set equal to yB2.
       - The variable log2TrafoWidth1 is set equal to log2TrafoWidth − 1.
       - The variable log2TrafoHeight1 is set equal to log2TrafoHeight − 1.
     - Otherwise (InterTUSplitDirection is equal to 0 or 1), the following applies.
       - The variable xB1 is set equal to xB + ((1 << (log2TrafoWidth)) >> 2) * InterTUSplitDirection.
       - The variable yB1 is set equal to yB + ((1 << (log2TrafoHeight)) >> 2) * (1 − InterTUSplitDirection).
       - The variable xB2 is set equal to xB1 + ((1 << (log2TrafoWidth)) >> 2) * InterTUSplitDirection.
       - The variable yB2 is set equal to yB1 + ((1 << (log2TrafoHeight)) >> 2) * (1 − InterTUSplitDirection).
       - The variable xB3 is set equal to xB2 + ((1 << (log2TrafoWidth)) >> 2) * InterTUSplitDirection.
       - The variable yB3 is set equal to yB2 + ((1 << (log2TrafoHeight)) >> 2) * (1 − InterTUSplitDirection).
       - The variable log2TrafoWidth1 is set equal to (log2TrafoWidth – 2) * InterTUSplitDirection.
       - The variable log2TrafoHeight1 is set equal to (log2TrafoHeight − 2) * (1 − InterTUSplitDirection).
  2. The decoding process for luma residual blocks as specified in this subclause is invoked with the luma location (xC, yC), the luma location (xB, yB), the variable log2TrafoWidth set equal to log2TrafoWidth1, the variable log2TrafoHeight set equal to log2TrafoHeight1, the variable trafoDepth set equal to trafoDepth + 1, the variable nCS, and the (nCS)x(nCS) array resSamples as the inputs and the output is a modified version of the (nCS)x(nCS) array resSamples.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000645

3. The decoding process for luma residual blocks as specified in this subclause is invoked with the luma location (xC, yC), the luma location (xB1, yB1), the variable log2TrafoWidth set equal to log2TrafoWidth1, the variable log2TrafoHeight set equal to log2TrafoHeight1, the variable trafoDepth set equal to trafoDepth + 1, the variable nCS, and the (nCS)x(nCS) array resSamples as the inputs and the output is a modified version of the (nCS)x(nCS) array resSamples.
4. The decoding process for luma residual blocks as specified in this subclause is invoked with the luma location (xC, yC), the luma location (xB2, yB2), the variable log2TrafoWidth set equal to log2TrafoWidth1, the variable log2TrafoHeight set equal to log2TrafoHeight1, the variable trafoDepth set equal to trafoDepth + 1, the variable nCS, and the (nCS)x(nCS) array resSamples as the inputs and the output is a modified version of the (nCS)x(nCS) array resSamples.
5. The decoding process for luma residual blocks as specified in this subclause is invoked with the luma location (xC, yC), the luma location (xB3, yB3), the variable log2TrafoWidth set equal to log2TrafoWidth1, the variable log2TrafoHeight set equal to log2TrafoHeight1, the variable trafoDepth set equal to trafoDepth + 1, the variable nCS, and the (nCS)x(nCS) array resSamples as the inputs and the output is a modified version of the (nCS)x(nCS) array resSamples.

– Otherwise (split_transform_flag[xB][yB][trafoDepth] is equal to 0), the following ordered steps apply:

1. The variable nW is set equal to 1 << log2TrafoWidth.
2. The variable nH is set equal to 1 << log2TrafoHeight.
3. The scaling and transformation process as specified in subclause 8.5.1 is invoked with the luma location (xC + xB, yC +yB), the variable trafoDepth, the variable cIdx set equal to 0, the transform width trafoWidth set equal to nW, and the transform height trafoHeight set equal to nH as the inputs and the output is a (nW)x(nH) array resSamplesBlock.
4. The array construction process as specified in subclause 8.5.5 is invoked with the luma location (xB, yB), the variable cIdx set equal to 0, the variable inputArrayWidth set equal to nW, the variable inputArrayHeight set equal to nH, the variable outputArraySize set equal to nCS, the (nW)x(nH) array resSamplesBlock, and the (nCS)x(nCS) array resSamples as the inputs and the output is a modified version of the (nCS)x(nCS) array resSamples.

8.4.3.2 Decoding process for chroma residual blocks

Inputs to this process are:

– a luma location (xC, yC) specifying the top-left luma sample of the current coding unit relative to the top-left luma sample of the current picture,
– a luma location (xB, yB) specifying the top-left luma sample of the current block relative to the top-left luma sample of the current coding unit,
– a variable log2TrafoWidth specifying the width of the current block,
– a variable log2TrafoHeight specifying the height of the current block,
– a variable trafoDepth specifying the hierarchy depth of the current block relative to the coding unit,
– a variable cIdx specifying the chroma component of the current block,
– a variable nCS specifying the size, in chroma samples, of the current coding unit,
– a (nCS)x(nCS) array resSamples of chroma residual samples.

Output of this process is:

– a modified version of the (nCS)x(nCS) array of chroma residual samples.

The variable splitChromaFlag is derived as follows:

– If split_transform_flag[xB][yB][trafoDepth] is equal to 1 and log2TrafoSize is greater than Log2MinTrafoSize + 1, splitChromaFlag is set equal to 1.
– Otherwise (split_transform_flag[xB][yB][trafoDepth] is equal to 0 or log2TrafoSize is equal to Log2MinTrafoSize + 1), splitChromaFlag is set equal to 0.

Depending splitChromaFlag, the following applies:

– If splitChromaFlag is equal to 1, the following ordered steps apply:

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000646

1. The variables xB1, yB1, xB2, yB2, xB3 and yB3 are derived as follows.
   - If InterTUSplitDirection is equal to 2, the following applies.
     - The variable xB1 is set equal to xB + ((1 << log2TrafoWidth) >> 1).
     - The variable yB1 is set equal to yB.
     - The variable xB2 is set equal to xB.
     - The variable yB2 is set equal to yB + ((1 << log2TrafoHeight) >> 1).
     - The variable xB3 is set equal to xB1.
     - The variable yB3 is set equal to yB2.
     - The variable log2TrafoWidth1 is set equal to log2TrafoWidth − 1.
     - The variable log2TrafoHeight1 is set equal to log2TrafoHeight − 1.
   - Otherwise (InterTUSplitDirection is equal to 0 or 1), the following applies.
     - The variable xB1 is set equal to xB + ((1 << (log2TrafoWidth)) >> 2) * InterTUSplitDirection.
     - The variable yB1 is set equal to yB + ((1 << (log2TrafoHeight)) >> 2) * (1 − InterTUSplitDirection).
     - The variable xB2 is set equal to xB1 + ((1 << (log2TrafoWidth)) >> 2) * InterTUSplitDirection.
     - The variable yB2 is set equal to yB1 + ((1 << (log2TrafoHeight)) >> 2) * (1 − InterTUSplitDirection).
     - The variable xB3 is set equal to xB2 + ((1 << (log2TrafoWidth)) >> 2) * InterTUSplitDirection.
     - The variable yB3 is set equal to yB2 + ((1 << (log2TrafoHeight)) >> 2) * (1 − InterTUSplitDirection).
     - The variable log2TrafoWidth1 is set equal to (log2TrafoWidth − 2) * InterTUSplitDirection.
     - The variable log2TrafoHeight1 is set equal to (log2TrafoHeight − 2) * (1 − InterTUSplitDirection).
2. The decoding process for residual chroma blocks as specified in this subclause is invoked with the luma location (xC, yC), the luma location (xB, yB), the variable log2TrafoWidth set equal to log2TrafoWidth1, the variable log2TrafoHeight set equal to log2TrafoHeight1, the variable trafoDepth set equal to trafoDepth + 1, the variable cIdx, the variable nCS, and the (nCS)x(nCS) array resSamples as the inputs and the output is a modified version of the (nCS)x(nCS) array resSamples.
3. The decoding process for residual chroma blocks as specified in this subclause is invoked with the luma location (xC, yC), the luma location (xB1, yB1), the variable log2TrafoWidth set equal to log2TrafoWidth1, the variable log2TrafoHeight set equal to log2TrafoHeight1, the variable trafoDepth set equal to trafoDepth + 1, the variable cIdx, the variable nCS, and the (nCS)x(nCS) array resSamples as the inputs and the output is a modified version of the (nCS)x(nCS) array resSamples.
4. The decoding process for residual chroma blocks as specified in this subclause is invoked with the luma location (xC, yC), the luma location (xB2, yB2), the variable log2TrafoWidth set equal to log2TrafoWidth1, the variable log2TrafoHeight set equal to log2TrafoHeight1, the variable trafoDepth set equal to trafoDepth + 1, the variable cIdx, the variable nCS, and the (nCS)x(nCS) array resSamples as the inputs and the output is a modified version of the (nCS)x(nCS) array resSamples.
5. The decoding process for residual chroma blocks as specified in this subclause is invoked with the luma location (xC, yC), the luma location (xB3, yB3), the variable log2TrafoWidth set equal to log2TrafoWidth1, the variable log2TrafoHeight set equal to log2TrafoHeight1, the variable trafoDepth set equal to trafoDepth + 1, the variable cIdx, the variable nCS, and the (nCS)x(nCS) array resSamples as the inputs and the output is a modified version of the (nCS)x(nCS) array resSamples.

- Otherwise (splitChromaFlag is equal to 0), the following ordered steps apply:
  1. The variable nW is set equal to (1 << log2TrafoWidth) >> 1.
  2. The variable nH is set equal to (1 << log2TrafoHeight) >> 1.
  3. The scaling and transformation process as specified in subclause 8.5.1 is invoked with the luma location (xC + xB, yC +yB), the variable trafoDepth, the variable cIdx, the transform width trafoWidth set equal to nW, and the transform height trafoHeight set equal to nH as the inputs and the output is a (nW)x(nH) array resSamplesBlock.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000647

4. The array construction process as specified in subclause 8.5.5 is invoked with the luma location (xB, yB), the variable cIdx, the variable inputArrayWidth set equal to nW, the variable inputArrayHeight set equal to nH, the variable outputArraySize set equal to nCS, the (nW)x(nH) array resSamplesBlock, and the (nCS)x(nCS) array resSamples as the inputs and the output is a modified version of the (nCS)x(nCS) array resSamples.

8.5 Scaling, transformation and array construction process prior to deblocking filter process

8.5.1 Scaling and transformation process

Inputs to this process are:

– a luma location (xT, yT) specifying the top-left luma sample of the current transform unit relative to the top-left luma sample of the current picture,

– a variable trafoDepth specifying the hierarchy depth of the current block relative to the coding unit,

– a variable cIdx specifying the chroma component of the current block,

– a variable nW specifying the width of the current transform block.

– a variable nH specifying the height of the current transform block.

Output of this process is:

– a modified version of the (nW)x(nH) array of residual samples r with elements r_{ij}.

The quantization parameter qP is derived as follows.

– If cIdx is equal to 0,

$$qP = QP'_Y \tag{8-150}$$

– Otherwise,

$$qP = QP'_C \tag{8-151}$$

The (nW)x(nH) array of residual samples r are derived as specified in the following ordered steps.

1. The inverse scanning process for transform coefficients as specified in subclause 8.5.2 is invoked with the width of the current transform unit nW, the height of the current transform unit nH, the list of transform coefficients transCoeffLevel[xT][yT][trafoDepth][cIdx] and the variable cIdx as the inputs and the output is a (nW)x(nH) array c_{ij}.
2. The scaling process for transform coefficients as specified in subclause 8.5.3 is invoked with the width of the transform unit nW, the height of the transform unit nH, the (nW)x(nH) array c, the chroma component variable cIdx and the quantization parameter qP as the inputs and the output is a scaled transform coefficient (nW)x(nH) array d.
3. The transformation process for scaled transform coefficients as specified in subclause 8.5.4 is invoked with the width of the transform unit nW, the height of the transform unit nH, the scaled transform coefficient (nW)x(nH) array d and the chroma component variable cIdx as the inputs and the output is a residual samples (nW)x(nH) array r.

8.5.2 Inverse scanning process for transform coefficients

Inputs to this process are:

– a variable nW specifying the width of the current transform unit,

– a variable nH specifying the height of the current transform unit.

– a list of (nW)x(nH) values transCoeffLevel[xT][yT][trafoDepth][cIdx].

– a variable cIdx specifying the chroma component of the current block.

Output of this process is a variable c containing a two-dimensional array of (nW)x(nH) values.

The variable nS is set equal to Sqrt(nW*nH).

The variable scanIdx, indicating wich scan is used to scan the transform coefficients (zigzag, horizontal or vertical), is derived as follows.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000648

– If PredMode is equal to MODE_INTRA, the following applies.
  – If cIdx is equal to 0, scanIdx is set to ScanType[Log2(nS >> 2)][IntraPredMode].
  – Otherwise (cIdx is equal to 1 or 2), scanIdx is set to ScanType[Log2(nS >> 2)][IntraPredModeC]
– Otherwise (PredMode is not equal to MODE_INTRA), scanIdx is set to 0.

When entropy_coding_mode_flag is equal to 1 (CABAC) and scanIdx is equal to 0 (zigzag), scanIdx is set to 3 (diagonal).

The variable c[i][j] which is located in the (i, j) position in the array c is derived as follows.

– The transform coefficient level at scanning position n is mapped to the position (i, j) in the array c using the scan specified by scanIdx as follows.

```
i = ScanOrder[ Log2( nS >> 2 ) ][ Log2( nS >> 2 ) ][ scanIdx ][ n ][ 0 ]
j = ScanOrder[ Log2( nS >> 2 ) ][ Log2( nS >> 2 ) ][ scanIdx ][ n ][ 1 ]
c[ i ][ j ] = transCoeffLevel[ xT ][ yT ][ trafoDepth ][ cIdx ][ n ],
  with n = 0.. ( nS * nS ) – 1                                        (8-152)
```

– When nW is not equal to nH, the coefficients in the array c are reordered as follows.
  – The array c0 is the copy of the (nS)x(nS) array c. For scanning position n, a coefficient at position (i0, j0) of c0 is mapped to the new position (i, j) of the reordered (nW)x(nH) array c using zigzag scan.

```
i0 = ScanOrder[ Log2( nS >> 2 ) ][ Log2( nS >> 2 ) ][ 0 ][ n ][ 0 ]
j0 = ScanOrder[ Log2( nS >> 2 ) ][ Log2( nS >> 2 ) ][ 0 ][ n ][ 1 ]
i = ScanOrder[ Log2( nW >> 2 ) ][ Log2( nH >> 2 ) ][ 0 ][ n ][ 0 ]
j = ScanOrder[ Log2( nW >> 2 ) ][ Log2( nH >> 2 ) ][ 0 ][ n ][ 1 ]
c[ i ][ j ] = c0[ i0 ][ j0 ]
  with n = 0…( nS * nS ) – 1                                          (8-152)
```

[Ed. (BB): Despite different scans NSQT reordering always uses zigzag scan. Needs to be reviewed and may be harmonized with the different scans.]

8.5.3 Scaling process for transform coefficients

Inputs of this process are:

– a variable nW specifying the width of the current transform unit,
– a variable nH specifying the height of the current transform unit,
– a (nW)x(nH) array c of transform coefficients with elements c_{ij},
– a variable cIdx specifying the chroma component of the current block,
– a variable qP specifying the quantization parameter.

Output of this process is scaled transform coefficients as a (nW)x(nH) array of d with elements d_{ij}.

The variable trafoPrecisionExt is derived as follows:

– If nW is greater than 4 or nH is greater than 4,
  – If cIdx is equal to 0 and bit_depth_luma_minus8 is equal to 0, trafoPrecisionExt is set equal to 2.
  – Otherwise, if cIdx is not equal to 0 and bit_depth_chroma_minus8 is equal to 0, trafoPrecisionExt is set equal to 2.
  – Otherwise, trafoPrecisionExt is set equal to 0.
– Otherwise, trafoPrecisionExt is set equal to 0.

[Ed.: (WJ) wait decision on transform precision extension]

The scaled transform coefficient array dij is derived as follows.

$$d_{ij} = (c_{ij} * \text{LevelScale}_{(nW)x(nH)}[qP\%6][i][j]) << (qP/6 + \text{trafoPrecisionExt}), \quad \text{with } i = 0..nW - 1, j = 0..nH - 1 \qquad (8\text{-}158)$$

[Ed: (WJ) $\text{LevelScale}_{(nS)x(nS)}$ should be very large table up to 32x32. Recent contributions (JCTVC-C209 and JCTVC-C255) shown that this matrix could be replaced by one constants per qP without coding efficiency penalty. Maybe it is better to wait.]

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000649

[Ed.: (WJ) dependent on transformation process – should be modified again after transformation process is fixed]

8.5.4 Transformation process for scaled transform coefficients

Inputs of this process are:

- a variable nW specifying the width of the current transform unit,
- a variable nH specifying the height of the current transform unit,
- a (nW)x(nH) array d of scaled transform coefficients with elements d_{ij}.
- a variable cIdx specifying the chroma component of the current block,

Output of this process is residual samples as a (nW)x(nH) array r with elements r_{ij}.

The variable trafoPrecisionExt is derived as follows:

- If nW is greater than 4 or nH is greater than 4,
  - If cIdx is equal to 0 and bit_depth_luma_minus8 is equal to 0, trafoPrecisionExt is set equal to 4.
  - Otherwise, if cIdx is not equal to 0 and bit_depth_chroma_minus8 is equal to 0, trafoPrecisionExt is set equal to 4.
  - Otherwise, trafoPrecisionExt is set equal to 0.
- Otherwise, trafoPrecisionExt is set equal to 0.

[Ed.: (WJ) wait decision on transform precision extension]

[Ed: (WJ) derivation of trafoPrecisionExt is duplicated to that of the scaling process. Better to remove one of two.]

Depending on PredMode and IntraPredMode, the following applies:

- If PredMode is equal to MODE_INTRA, log2(nW*nH) is equal to 4, and cIdx is equal to 0, the variables horizTrType and vertTrType are specified as Table 8-11 with IntraPredMode as input. [Ed. (WJ): DST is applied only for luma 4x4 block]
- Otherwise, the variables horizTrType and vertTrType are set equal to 0.

Table 8-14 – Specification of horizTrType and vertTrType

IntraPredMode	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
horizTrType	0	1	0	1	1	0	0	1	1	1	1	1	0	0	1	1	1	1
vertTrType	1	0	0	1	1	1	1	1	0	0	1	1	1	1	1	1	0	0

IntraPredMode	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
horizTrType	1	1	1	1	0	0	0	0	1	1	1	1	1	1	1	1	1
vertTrType	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	1

The constructed residual samples are derived as specified in the following ordered steps.

1. The variable nS is derived as follows.
   - If nW is greater than or equal to nH, nS is set to nW.
   - Otherwise (nW is less than nH), nS is set to nH.
2. Each (horizontal) row of scaled transform coefficients d_{ij} (i=0..nW−1, j=0..nH−1) is transformed to e_{ij} (i=0..nW−1, j=0..nH−1) by invoking the one-dimensional transformation process as specified in subclause 8.5.4.1 to 8.5.4.4 according to the width of the transform unit nW, with the (nW)x(nH) array d and the transform type variable horizTrType as the inputs and the output is the (nW)x(nH) array e.
3. Each (vertical) column of the resulting matrix e_{ij} (i=0..nW−1, j=0..nH−1) is transformed to f_{ij} (i=0..nW−1, j=0..nH−1) by invoking the one-dimensional transformation process as specified in subclause 8.5.4.1 to 8.5.4.4

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000650

according to the height of the transform unit nH, with the (nW)x(nH) array e and the transform type variable vertTrType as the inputs and the output is the (nW)x(nH) array f.

4. The residual sample value r_{ij} is derived as follows.
   - If nS is equal to 4 or 8,

     $$r_{ij} = (f_{ij} + 2^{5+trafoPrecisionExt}) >> (6 + trafoPrecisionExt), \text{ with } i=0..(nW)-1, j=0..(nH)-1 \quad (8\text{-}159)$$

   - Otherwise (nS is equal to 16 or 32),

     $$r_{ij} = (f_{ij} + 2^{9+trafoPrecisionExt}) >> (10 + trafoPrecisionExt), \text{ with } i=0..(nW)-1, j=0..(nH)-1 \quad (8\text{-}160)$$

[Ed.: (WJ) dependent on transformation process – should be modified again after transformation process is fixed]

[Ed.: (BB) if modified, the condition on nS for NSQT should be checked]

8.5.4.1 Transformation process for 4 samples

Inputs of this process are an array of 4 samples of the scaled transform coefficients x with elements x_i, with i = 0..3 and the transform type variable trType.

Output of this process is an array of 4 samples of the residual samples y with elements y_i, with i = 0..3.

Depending on trType, the following applies:

– If tyType is equal to 1, the following ordered steps apply:

1. A set of intermediate values c_0, c_1, c_2 and c_3 is calculated as follows:

   $$c_0 = x_0 + x_2 \quad (8\text{-}159)$$
   $$c_1 = x_2 + x_3 \quad (8\text{-}159)$$
   $$c_2 = x_0 - x_3 \quad (8\text{-}159)$$
   $$c_3 = 74 * x_1 \quad (8\text{-}159)$$

2. The output values y_i with i = 0..3 are then specified as follows:

   $$y_0 = (29*c_0 + 55*c_1 + c_3 + 64) >> 7 \quad (8\text{-}159)$$
   $$y_1 = (55*c_2 - 29*c_1 + c_3 + 64) >> 7 \quad (8\text{-}159)$$
   $$y_2 = (74*(x_0 - x_2 + x_3) + 64) >> 7 \quad (8\text{-}159)$$
   $$y_3 = (55*c_0 + 29 * c_2 - c_3 + 64) >> 7 \quad (8\text{-}159)$$

   [Ed. (WJ): proponent's comment - to share the quantization scaling value, transform precision (7-bit in this case) should be synchronized with DCT]

– Otherwise, the following ordered steps apply: [Ed. (WJ): TBD – 4-point DCT]

8.5.4.2 Transformation process for 8 samples

Inputs of this process are an array of 8 samples of the scaled transform coefficients x with elements x_i, with i = 0..7 and the transform type variable trType.

Output of this process is an array of 8 samples of the residual samples y with elements y_i, with i = 0..7.

[Ed: (WJ) TBD]

8.5.4.3 Transformation process for 16 samples

Inputs of this process are an array of 16 samples of the scaled transform coefficients x with elements x_i, with i = 0..15 and the transform type variable trType.

Output of this process is an array of 16 samples of the residual samples y with elements y_i, with i = 0..15.

[Ed: (WJ) TBD]

8.5.4.4 Transformation process for 32 samples

Inputs of this process are an array of 32 samples of the scaled transform coefficients x with elements x_i, with i=0..31 and the transform type variable trType.

Output of this process is an array of 32 samples of the residual samples y with elements y_i, with i=0..31.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000651

[Ed: (WJ) TBD]

8.5.5 Array construction process

Inputs of this process are:

- a luma location (xB, yB) specifying the top-left luma sample of the current block relative to the top-left luma sample of the current coding unit,
- a variable cIdx specifying the chroma component of the current block,
- a variable nW specifying the width of the current block,
- a variable nH specifying the height of the current block,
- a variable nCS specifying the size of the current coding unit,
- a (nW)x(nH) array resSampleBlock specifying the current block samples,
- a (nCS)x(nCS) array resSample specifying the samples of the current coding unit.

Output of this process is a modified (nCS)x(nCS) array resSample specifying the samples of the current coding unit.

The modified array resSample is derived as follows.

resSample[xB+i, yB+j] += resSampleBlock[i, j], with i = 0..nW − 1, j = 0..nH − 1 (8-427)

8.6 In-loop filter process

[Ed: (WJ) overall process description need to be inserted]

8.6.1 Deblocking filter process

A conditional filtering process shall be performed on a treeblock basis after the completion of the picture construction process prior to deblocking filter process for the entire decoded picture (as specified in subclauses XXX and YYY) [Ed.: (WJ) those subclauses seem not defined yet], with all treeblocks in a picture processed in order of increasing treeblock addresses.

Each treeblock is processed on a coding unit basis with the same order as decoding process. For each coding unit, vertical edges are filtered first, starting with the edge on the left-hand side of the coding unit proceeding through the edges towards the right-hand side of the coding unit in their geometrical order. Then, the horizontal edges are filtered starting with the edge on the top of the coding unit proceeding through the edges towards the bottom of the coding unit in their geometrical order.

Sample values above of the current coding unit that may have already been modified by the filtering of horizontal edges of deblocking filter process operation on previous coding unit shall be used as inputs to the deblocking filter process on the current coding unit and may be further modified during the filtering of the current coding unit. Sample values to the left of the current coding unit shall be used as inputs to the deblocking filter process on the current coding unit and may be further modified during the filtering of the current coding unit. Sample values to the left of the current coding unit may be modified by the filtering of vertical edge and may be further modified by the filtering of horizontal edges.

Sample values modified during filtering of vertical edges are used as input for the filtering of the horizontal edges. For sample values modified by both filtering of horizontal edges and filtering of vertical edges, filtering of horizontal edges is applied after filtering of vertical edges.

The deblocking filter process shall be applied to all prediction unit edges and transform unit edges of a picture, except edges at the boundary of the picture, any edges for which the deblocking filter process is disabled by disable_deblocking_filter_slice_flag and any edges coinside with slice boundaries when loop_filter_across_slice_flag is equal to 0. For the transform units and prediction units with edges smaller than 8 samples in either vertical or horizontal direction, only the edges lying on the 8x8 sample grid are filtered.

When disable_deblocking_filter_slice_flag is not equal to 1, the deblocking filter process is invoked as the following ordered steps for each coding unit with the same order as decoding process.

1. The coding unit size nS is set equal to 1 << log2CUSize.
2. The variables FilterInternalEdgesFlag, FilterLeftCuEdgeFlag and FilterTopCuEdgeFlag are derived as follows.
   - The variable FilterInternalEdges is set equal to 1.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000652

- If the left boundary of current coding unit is the left boundary of the picture or if the left boundary of current coding unit is the left boundary of the slice and loop_filter_across_slice_flag is equal to 0, the variable FilterLeftCuEdgeFlag is set equal to 0, otherwise set equal to 1.
- If the top boundary of current coding unit is the top boundary of the picture or if the top boundary of current coding unit is the top boundary of the slice and loop_filter_across_slice_flag is equal to 0, the variable FilterTopCuEdgeFlag is set equal to 0, otherwise set equal to 1.

3. All elements of two-dimensional array of size (nS)x(nS), horEdgeFlags and verEdgeFlags are initialized to zero.
4. The derivation process of transform unit boundary specified in subclause 8.6.1.1 are invoked with the luma location (xB, yB) set equal to (0, 0), the transform unit width log2TrafoWidth set equal to log2CUSize, the transform unit height log2TrafoHeight set equal to log2CUSize and the variable trafoDepth set equal to 0 as the inputs and the modified horEdgeFlags and verEdgeFlags as outputs.
5. The derivation process of prediction unit boundary specified in subclause 8.6.1.2 are invoked with the coding unit size log2CUSize and the prediction partition mode PartMode as inputs, and the modified horEdgeFlags and verEdgeFlags as outputs.
6. The derivation process of the boundary filtering strength specified in subclause 8.6.1.3 is invoked with the luma location (xC, yC), the coding unit size log2CUSize, horEdgeFlags and verEdgeFlags as inputs and an array of size (2)x(nS)x(nS), bS as output.
7. The filtering process for coding unit specified in subclause 8.6.1.4 are invoked with the luma location (xC, yC) specifying the top-left luma sample of the current coding unit relative to the top left luma sample of the current picture, the coding unit size log2CUSize and the array bS as inputs and the modified reconstructued picture as output.

8.6.1.1 Derivation process of transform unit boundary

Inputs of this process are:

- a luma location (xB, yB) specifying the top-left luma sample of the current block relative to the top-left luma sample of the current coding unit,
- a variable log2TrafoWidth specifying the width of the current block,
- a variable log2TrafoHeight specifying the height of the current block,
- a variable trafoDepth.

Outputs of this process are:

- two-dimensional arrays of (nS)x(nS), horEdgeFlags and verEdgeFlags.

Depending on split_transform_flag[xB][yB][trafoDepth], the following applies:

- If split_transform_flag[xB][yB][trafoDepth] is equal to 1, the following ordered steps apply:
  1. The variables xB1, yB1, xB2, yB2, xB3 and yB3 are derived as follows.
     - If InterTUSplitDirection is equal to 2, the following applies.
       - The variable xB1 is set equal to xB + ((1 << log2TrafoWidth) >> 1).
       - The variable yB1 is set equal to yB.
       - The variable xB2 is set equal to xB.
       - The variable yB1 is set equal to yB + ((1 << log2TrafoHeight) >> 1).
       - The variable xB3 is set equal to xB1.
       - The variable yB3 is set equal to yB2.
       - The variable log2TrafoWidth1 is set equal to log2TrafoWidth – 1.
       - The variable log2TrafoHeight1 is set equal to log2TrafoHeight – 1.
     - Otherwise (InterTUSplitDirection is equal to 0 or 1), the following applies.
       - The variable xB1 is set equal to xB + ((1 << (log2TrafoWidth)) >> 2) * InterTUSplitDirection.
       - The variable yB1 is set equal to yB + ((1 << (log2TrafoHeight)) >> 2) * (1 – InterTUSplitDirection).

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000653

- The variable xB2 is set equal to xB1 + ((1 << (log2TrafoWidth)) >> 2) * InterTUSplitDirection.
- The variable yB2 is set equal to yB1 + ((1 << (log2TrafoHeight)) >> 2) * (1 − InterTUSplitDirection).
- The variable xB3 is set equal to xB2 + ((1 << (log2TrafoWidth)) >> 2) * InterTUSplitDirection.
- The variable yB3 is set equal to yB2 + ((1 << (log2TrafoHeight)) >> 2) * (1 − InterTUSplitDirection).
- The variable log2TrafoWidth1 is set equal to (log2TrafoWidth − 2) * InterTUSplitDirection.
- The variable log2TrafoHeight1 is set equal to (log2TrafoHeight − 2) * (1 − InterTUSplitDirection).

2. The deriviation process of transform unit boundary as specified in this subclause is invoked with the luma location (xB, yB), the variable log2TrafoWidth set equal to log2TrafoWidth1, the variable log2TrafoHeight set equal to log2TrafoHeight1 and the variable trafoDepth1 set equal to trafoDepth + 1 as inputs and the outputs are the modified versions of two arrays, horEdgeFlags and verEdgeFlags.
3. The deriviation process of transform unit boundary as specified in this subclause is invoked with the luma location (xB1, yB1), the variable log2TrafoWidth set equal to log2TrafoSizeWidth1, the variable log2TrafoHeight set equal to log2TrafoHeight1 and the variable trafoDepth1 set equal to trafoDepth + 1 as inputs and the outputs are the modified versions of two arrays, horEdgeFlags and verEdgeFlags.
4. The deriviation process of transform unit boundary as specified in this subclause is invoked with the luma location (xB2, yB2), the variable log2TrafoWidth set equal to log2TrafoSizeWidth1, the variable log2TrafoHeight set equal to log2TrafoHeight1 and the variable trafoDepth1 set equal to trafoDepth + 1 as inputs and the outputs are the modified versions of two arrays, horEdgeFlags and verEdgeFlags.
5. The deriviation process of transform unit boundary as specified in this subclause is invoked with the luma location (xB3, yB3), the variable log2TrafoWidth1 set equal to log2TrafoSizeWidth1, the variable log2TrafoHeight set equal to log2TrafoHeight1 and the variable trafoDepth1 set equal to trafoDepth + 1 as inputs and the outputs are the modified versions of two arrays, horEdgeFlags and verEdgeFlags.

– Otherwise (split_transform_flag[xB][yB][trafoDepth] is equal to 0), the following applies:
  - If yB is equal to zero, horEdgeFlags[xB + k][yB] is set equal to FilterTopCuEdgeFlag, otherwise horEdgeFlags[xB + k][yB] is set equal to FilterInternalEdgesFlag for k = 0.. (1 << log2TrafoWidth) − 1.
  - If xB is equal to zero, verEdgeFlags[xB][yB + k] is set equal to FilterLeftCuEdgeFlag, otherwise verEdgeFlags[xB][yB + k] is set equal to FilterInternalEdgesFlag for k = 0.. (1 << log2TrafoHeight) − 1.

8.6.1.2 Derivation process of prediction unit boundary

Inputs of this process are:

- a variable log2CUSize specifying the coding unit size,
- a prediction partition mode PartMode.

Outputs of this process are:

- two-dimensional arrays of (nS)x(nS), horEdgeFlags and verEdgeFlags.

Depending on PartMode, the following applies:

- If PartMode is equal to PART_2NxN or PART_NxN, horEdgeFlags[k][1 << (log2CUSize − 1)] is set equal to FilterInternalEdgesFlag for k = 0.. (1 << log2CUSize) − 1.
- If PartMode is equal to PART_Nx2N or PART_NxN, verEdgeFlags[1 << (log2CUSize − 1)][k] is set equal to FilterInternalEdgesFlag for k = 0.. (1 << log2CUSize) − 1.
- If PartMode is equal to PART_2NxnU, horEdgeFlags[k][(1 << (log2CUSize − 1)) − (1 << (log2CUSize − 2))] is set equal to FilterInternalEdgesFlag for k = 0.. (1 << log2CUSize) − 1.
- If PartMode is equal to PART_2NxnD, horEdgeFlags[k][(1 << (log2CUSize − 1)) + (1 << (log2CUSize − 2))] is set equal to FilterInternalEdgesFlag for k = 0.. (1 << log2CUSize) − 1.
- If PartMode is equal to PART_nLx2N, verEdgeFlags[(1 << (log2CUSize − 1)) − (1 << (log2CUSize − 2))][k] is set equal to FilterInternalEdgesFlag for k = 0.. (1 << log2CUSize) − 1.
- If PartMode is equal to PART_nRx2N, verEdgeFlags[(1 << (log2CUSize − 1)) + (1 << (log2CUSize − 2))][k] is set equal to FilterInternalEdgesFlag for k = 0.. (1 << log2CUSize) − 1.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000654

8.6.1.3 Derivation process of boundary filtering strength

Inputs of this process are:

- a luma location (xC, yC) specifying the top-left luma sample of the current coding unit relative to the top-left luma sample of the current picture,
- a variable log2CUSize specifying the size of the current coding unit,
- a two-dimensional arrays of size (nS)x(nS), horEdgeFlags and verEdgeFlags.

Output of this process is an array of size (2)x(nS)x(nS), bS specifying the boundary filtering strength.

Let (xE_k, yE_j) with k = 0..nE-1 and j = 0..nE-1 specify a set of edge sample locations where nE is set equal to ((1 << log2CUSize) >> 2), $xE_0 = 0$, $yE_0 = 0$, $xE_{k+1} = xE_k + 4$ and $yE_{j+1} = yE_j + 4$.

For (xE_k, yE_j) with k = 0..nE-1 and j = 0..nE-1, the following applies.

- If horEdgeFlags[xE_k][yE_j] is equal to 1,
  - Set sample p_0 = recPicture[xC + xE_k][yC + yE_j – 1] and q_0 = recPicture[xC + xE_k][yC + yE_j].
  - The variable filterDir is set equal to 1.
- Otherwise, if verEdgeFlags[xE_k][yE_j] is equal to 1,
  - Set sample p_0 = recPicture[xC + xE_k – 1][yC + yE_j] and q_0 = recPicture[xC + xE_k][yC + yE_j].
  - The variable filterDir is set equal to 0.
- Depending on the value of filterDir, the variable bS[filterDir][xE_k][yE_j] is derived as follows.
  - If the block edge is also a coding unit edge and the following condition is true, the variable bS[0][xE_k][yE_j] is set equal to 4.
    - The sample p_0 or q_0 is in a coding unit coded with intra prediction mode
  - Otherwise, if the following condition is true, the variable bS[filterDir][xE_k][yE_j] is set equal to 3.
    - The sample p_0 or q_0 is in a coding unit coded with intra prediction mode
  - Otherwise, if the block edge is also a transform unit edge and the following condition is true, the variable bS[filterDir][xE_k][yE_j] is set equal to 2.
    - The sample p_0 or q_0 is in a transform unit which contains non-zero transform coefficient level.
  - Otherwise, if any of the following conditions are true, the variable bS[filterDir][xE_k][yE_j] is set equal to 1.
    - The prediction unit containing sample p_0 has different reference pictures or a different number of motion vectors with the prediction unit containing the sample q_0.

      NOTE – The determination of whether the reference pictures used for the two prediction are the same or different is based on which pictures are referenced, without regard to whether a prediction is formed using an index into list 0 or an index into list 1, and also without regard to whether or not the index position within a reference picture list is different or not.
    - One motion vector is used to predict the prediction unit containing sample p_0, one motion vector is used to predict the prediction unit containing sample q_0, and the absolute difference between the horizontal or vertical component of the motion vector used is greater than or equal to 4 in units of quarter luma samples.
    - [Ed.: (WJ) needs to be checked again whether this condition covers all 2-motion cases] Two motion vectors are used to predict the prediction unit containing sample p_0, two motion vectors are used to predict the prediction unit containing sample q_0, and at least one of the motion vector pairs corresponding the same reference pictures and the different boundary samples p_0 and q_0 satisfies the following condition:

      1. The absolute difference between the horizontal or vertical component of a motion vector used in the prediction of the two prediction units is greater than or equal to 4 in units of quarter luma samples.
  - Otherwise, the variable bS[filterDir][xE_k][yE_j] is set equal to 0.

8.6.1.4 Filtering process for coding unit

Inputs of this process are:

- a luma location (xC, yC) specifying the top-left luma sample of the current coding unit relative to the top left luma sample of the current picture,

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000655

– a variable log2CUSize specifying the coding unit size,

– an array bS specifying the boundary filtering strength.

Output of this process is:

– modified reconstruction of the picture.

The filtering process for luma edges in the current coding unit consists of the following ordered steps:

1. The variable nD is set equal to 1 << (log2CUSize – 3).
2. All elements of the three-dimensional array of size (2)x(nD)x(nD), dEdge are initialized to zero.
3. All elements of the three-dimensional array of size (2)x(nD)x(1<<log2CUSize), dSample are initialized to zero.
4. All elements of the three-dimensional array of size (2)x(nD)x(nD), bStrength ae initialized to zero.
5. For xD_k set equal to xC+(k << 3), k=0..nD – 1, the following applies:
   – For yD_m set equal to yC+(m << 3), m=0..nD – 1, the following ordered steps apply:
     a. Boundary filtering strength bSVer is derived as follows:

        bSVer = Max(bS[0][xD_k][yD_m + i]) for i = 0..7 (8-428)

     b. bStrength[1][k][m] is set equal to bSVer.
     c. Decision process for luma block edge in subclause 8.6.1.4.1 is invoked with the luma location of the coding unit (xC, yC), the luma location of the block (xD_k, yD_m), a variable verticalEdgeFlag set equal to 1, and the boundary filtering strength bSVer as inputs and the decision dEdge[1][k][m] and an array dS of size (8) as outputs.
     d. dSample[1][k][(m<<3)+i] is set equal to dS[i] for i=0..7.
     e. Boundary filtering strength bSHor is derived as follows:

        bSHor = Max(bS[1][xD_k + i][yD_m]) for i = 0..7 (8-428)

     f. bStrength[0][k][m] is set equal to bSHor.
     g. Decision process for luma block edge in subclause 8.6.1.4.1 is invoked with the luma location of the coding unit (xC, yC), the luma location of the block (xD_k, yD_m), a variable verticalEdgeFlag set equal to 0, the boundary filtering strength bSHor as inputs, the decision dEdge[0][k][m] and an array dS of size (8) as outputs.
     h. dSample[0][m][(k<<3)+i] is set equal to dS[i] for i=0..7.
6. For xD_k set equal to xC+(k << 3), k=0..nD - 1, the following applies:
   – For yD_m set equal to yB+(m << 3), m=0..nD – 1, the following ordered steps apply:
     a. dS[i] is set equal to dSample[1][k][(m<<3)+i] for i=0..7.
     b. Filtering process for luma block edge in subclause 8.6.1.4.2 is invoked with the luma location of the coding unit (xC, yC), the luma location of the block (xD_k, yD_m), a variable verticalEdgeFlag set equal to 1, the boundary filtering strength bStrength[1][k][m], the decision dEdge[1][k][m], and the array of size (8), dS as inputs and the modified luma picture buffer as outputs.
7. For yD_m set equal to yC+(m << 3), m=0..nD - 1, the following applies:
   – For xD_k set equal to xC+(k << 3), k=0..nD – 1, the following ordered steps apply:
     a. If xD_k is equal to 0, the parameter xPOS is set equal to 1. If xD_k is equal to xB+(nD – 1) << 3) xPOS is set equal to 2. Otherwise xPOS is set to 0.
     b. dS[i] is set equal to dSample[0][m][(k << 3) + i] for i = 0..7.
     c. Filtering process for luma block edge in subclause 8.6.1.4.2 is invoked with the luma location of the coding unit (xC, yC), the luma location of the block (xD_k, yD_m), a variable verticalEdgeFlag set equal to 0, the boundary filtering strength bStrength[0][k][m], the decision dEdge[0][k][m], and the array of size (8), dS, xPOS, dS_L[m][], dE_L[m], bS_L[m], and tc_L[m], as inputs and the modified luma picture buffer as output.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000656

- The elements of the two dimensional array of size (3)x(nD), dS_L are set as follows. $dS_L[m][0]$, $dS_L[m][1]$, and $dS_L[m][2]$, are set equal to dS[5], dS[6] and dS[7].
- The elements of the array of size (nD), dE_L are set as follows. $dE_L[m]$ is set equal to dEdge[0][k][m].
- The elements of the array of size (nD), bS_L are set as follows. $bS_L[m]$ is set equal to bStrength[0][k][m].
- The elements of the array of size (nD), t_C are set as follows. $t_{CL}[m]$ is set equal to t_c.

The filtering process for chroma edges in the current coding unit consists of the following ordered steps:

1. The variable nD is set equal to 1 << (Max(log2CUSize, 4) – 4).
2. For xD_k set equal to (xC / 2)+(k << 3), k=0..nD – 1, the following applies:
   - For yD_m set equal to (yC / 2)+(m << 2), m=0..nD*2 – 1, the following ordered steps apply:
     a. Boundary filtering strength bSVer is derived as follows:

        $$bSVer = bS[\ 0\ ][\ xD_k*2\ ][\ yD_m*2\ ] \quad (8\text{-}430)$$

     b. Filtering process for chroma block edge in subclause 8.6.1.4.3 is invoked with the chroma location of the coding unit (xC/2, yC/2), the chroma location of the block (xD_k, yD_m), a variable verticalEdgeFlag set equal to 1, a chroma component index cIdx set equal to 1 and the boundary filtering strength bSVer as inputs and the modified chroma picture buffer as output.

     c. Filtering process for chroma block edge in subclause 8.6.1.4.3 is invoked with the chroma location of the coding unit (xC/2, yC/2), the chroma location of the block (xD_k, yD_m), a variable verticalEdgeFlag set equal to 1, a chroma component index cIdx set equal to 2 and the boundary filtering strength bSVer as inputs and the modified chroma picture buffer as output.
3. For yD_m set equal to (yC / 2)+(m << 3), m=0..nD – 1, the following applies:
   - For xD_k set equal to (xC / 2)+(k << 2), k=0..nD*2 – 1, the following ordered steps apply:
     a. If xD_k is equal to 0, the parameter xPOS is set equal to 1. If xD_k is equal to xB+((nD*2 – 1) << 2) xPOS is set equal to 2. Otherwise xPOS is set to 0.

     b. Boundary filtering strength bSHor is derived as follows:

        $$bSHor = bS[\ 1\ ][\ xD_k*2\ ][\ yD_m*2\ ] \quad (8\text{-}431)$$

     c. Filtering process for chroma block edge in subclause 8.6.1.4.3 is invoked with the chroma location of the coding unit (xC/2, yC/2), the chroma location of the block (xD_k, yD_m), a variable verticalEdgeFlag set equal to 0, a chroma component index cIdx set equal to 1 and the boundary filtering strength bSHor, xPOS, $bS_L[m]$ and $t_{CL}[m]$ as inputs and the modified chroma picture buffer as output.

     d. Filtering process for chroma block edge in subclause 8.6.1.4.3 is invoked with the chroma location of the coding unit (xC/2, yC/2), the chroma location of the block (xD_k, yD_m), a variable verticalEdgeFlag set equal to 0, a chroma component index cIdx set equal to 2 and the boundary filtering strength bSHor, xPOS, $bS_L[m]$ and $t_{CL}[m]$ as inputs and the modified chroma picture buffer as output.

8.6.1.4.1 Decision process for luma block edge

Inputs of this process are:

- a luma location (xC, yC) specifying the top-left luma sample of the current coding unit relative to the top left luma sample of the current picture,
- a luma location (xB, yB) specifying the top-left luma sample of the current block relative to the top left luma sample of the current coding unit,
- a variable verticalEdgeFlag,
- a variable bS specifying the boundary filtering strength,

Output of this process is:

- variables dE, dEp1 and dEq1 containing decisions,
- one-dimensional array of size (8), dS containing decisions.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000657

Let s' represent the luma sample array $recPicture_L$ of the current picture.

A variables β is specified as Table 8-13 with luma quantization parameter qP_L as input.

A variable t_C is specified as follows:

– If bS is greater than 2, the variable t_C is specified specified as Table 8-13 with luma quantization parameter Clip3(0, 55, qP_L + 2) as input,

– Otherwise (bS is equal or less than 2), the variable t_C is specified specified as Table 8-13 with luma quantization parameter qP_L as input.

Depending on verticalEdgeFlag, the following applies:

– If verticalEdgeFlag is equal to 1, the following ordered steps apply:

1. The sample values $p_{i,k}$ and $q_{i,k}$ with i = 0..3 and k = 2,5 are derived as follows:

$$q_{i,k} = s'[\ xC + xB + i,\ yC + yB + k\ ] \quad (8\text{-}447)$$

$$p_{i,k} = s'[\ xC + xB - i - 1,\ yC + yB + k\ ] \quad (8\text{-}448)$$

2. The variables dp, dq and d are derived as follows:

$$dp = |\ p_{2,2} - 2*p_{1,2} + p_{0,2}\ | + |\ p_{2,5} - 2*p_{1,5} + p_{0,5}\ | \quad (8\text{-}449)$$

$$dq = |\ q_{2,2} - 2*q_{1,2} + q_{0,2}\ | + |\ q_{2,5} - 2*q_{1,5} + q_{0,5}\ | \quad (8\text{-}449)$$

$$d = dp + dq \quad (8\text{-}449)$$

3. The variables dE, dEp1 and dEq1 are set equal to 0.

4. If bS is not equal to 0 and d is less than β, the following ordered steps apply:

   a. for each sample location (xC + xB, yC + yB + k), k = 0..7, the following ordered steps apply:

      a. The decision process for a luma sample specified in subclause 8.6.1.4.4 is invoked with sample values $p_{i,k}$, $q_{i,k}$ with i = 0..3, the boundary filtering strength bS and the variables d, β and t_C as inputs and a decision dSam as output.

      b. The variable dS[k] is set equal to dSam

   b. The variable dE is set equal to 1.

   c. If dp is less than (β + (β >> 1)) >> 3, the variable dEp1 is set equal to 1.

   d. If dq is less than (β + (β >> 1)) >> 3, the variable dEq1 is set equal to 1.

– Otherwise (verticalEdgeFlag is equal to 0), the following ordered steps apply:

1. The sample values $p_{i,k}$ and $q_{i,k}$ with i = 0..3 and k = 2,5 are derived as follows:

$$q_{i,k} = s'[\ xC + xB + k,\ yC + yB + i\ ] \quad (8\text{-}435)$$

$$p_{i,k} = s'[\ xC + xB + k,\ yC + yB - i - 1\ ] \quad (8\text{-}436)$$

2. The variables dp, dq and d are derived as follows:

$$dp = |\ p_{2,2} - 2*p_{1,2} + p_{0,2}\ | + |\ p_{2,5} - 2*p_{1,5} + p_{0,5}\ | \quad (8\text{-}437)$$

$$dq = |\ q_{2,2} - 2*q_{1,2} + q_{0,2}\ | + |\ q_{2,5} - 2*q_{1,5} + q_{0,5}\ | \quad (8\text{-}437)$$

$$d = dp + dq \quad (8\text{-}437)$$

3. The variables dE, dEp1 and dEq1 are set equal to 0.

4. If bS is not equal to 0 and d is less than β, the following ordered steps apply:

   a. For each sample location (xC + xB + k, yC + yB), k = 0..7, the following ordered steps apply:

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000658

a. The decision process for a luma sample specified in subclause 8.6.1.4.4 is invoked with sample values $p_{i,k}$, $q_{i,k}$ with i = 0..3, the boundary filtering strength bS and the variables d, β and t_C as inputs and a decision dSam as output.

b. The variable dS[k] is set equal to dSam.

b. The variable dE is set equal to 1.

c. If dp is less than (β + (β >> 1)) >> 3, the variable dEp1 is set equal to 1.

d. If dq is less than (β + (β >> 1)) >> 3, the variable dEq1 is set equal to 1.

8.6.1.4.2 Filtering process for luma block edge

Inputs of this process are:

– a luma location (xC, yC) specifying the top-left luma sample of the current coding unit relative to the top left luma sample of the current picture,

– a luma location (xB, yB) specifying the top-left luma sample of the current block relative to the top left luma sample of the current coding unit,

– a variable verticalEdgeFlag,

– a variable bS specifying the boundary filtering strength,

– variables dE, dEp1 and dEq1 containing decisions,

– one-dimensional array of size (8), dS containing decisions,

– a variable bS_L,

– a variable t_{CL},

Output of this process is:

– modified reconstruction of the picture.

Let s' represent the luma sample array $recPicture_L$ of the current picture.

Depending on pcm_flag, a variable β is specified as follows:

– If pcm_flag is equal to 1, the variables β is specified as Table 8-11 with luma quantization parameter 0 as input.

– Otherwise, the variables β is specified as Table 8-11 with luma quantization parameter qP_L as input.

A variable t_C is specified as follows:

– If bS is greater than 2, the variable t_C is specified as Table 8-11 with luma quantization parameter Clip3(0, 55, qP_L + 2) as input,

– Otherwise (bS is equal or less than 2), the variable t_C is specified as Table 8-11 with luma quantization parameter qP_L as input.

Depending on verticalEdgeFlag, the following applies:

– If verticalEdgeFlag is equal to 1, the following ordered steps apply:

1. The sample values $p_{i,k}$ and $q_{i,k}$ with i = 0..3 and k = 0..7 are derived as follows:

$$q_{i,k} = s'[\ xC + xB + i,\ yC + yB + k\ ] \quad (8\text{-}432)$$

$$p_{i,k} = s'[\ xC + xB - i - 1,\ yC + yB + k\ ] \quad (8\text{-}433)$$

2. If dE is not equal to 0, for each sample location (xC + xB, yC + yB + k), k = 0..7, the following ordered steps apply:

a. The filtering process for a luma sample specified in subclause 8.6.1.4.5 is invoked with sample values $p_{i,k}$, $q_{i,k}$ with i = 0..3, the decision dS[k], variables dEp1 and dEq1, the boundary filtering strength bS and the variable t_C as inputs and the number of filtered samples nDp and nDq from each side of the block boundary, and the filtered sample values p_i' and q_j' as outputs.

b. The filtered sample values p_i' and q_j' with i = 0..nDp – 1, j = 0..nDq – 1 replace the corresponding samples inside the sample array s' as follows:

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000659

$$s'[\ xC + xB + j, yC + yB + k\ ] = q_j' \quad (8\text{-}435)$$

$$s'[\ xC + xB - i - 1, yC + yB + k\ ] = p_i' \quad (8\text{-}436)$$

– Otherwise (verticalEdgeFlag is equal to 0), the following ordered steps apply:

1. If xPOS is equal to 1, the parameters k_s and k_e are set to -3 and 4 respectively. If xD is equal to 2, the parameters k_s and k_e are set to 0 and 4 respectively. Otherwise k_s and k_e are set to 0 and 7 respectively.
2. The sample values $p_{i,k}$ and $q_{i,k}$ with i = 0..3 and k = k_s..k_e are derived as follows:

$$q_{i,k} = s'[\ xC + xB + k, yC + yB + i\ ] \quad (8\text{-}437)$$

$$p_{i,k} = s'[\ xC + xB + k, yC + yB - i - 1\ ] \quad (8\text{-}438)$$

3. If xPOS is less than 0 and dE_L is not equal to 0, for each sample location (xC + xB + k, yC + yB), k = -3..-1, the following ordered steps apply:
   a. The filtering process for a luma sample specified in subclause 8.6.1.4.5 is invoked with sample values $p_{i,k}$, $q_{i,k}$ with i = 0..3, decision dS_L[k+3], variables dEp1 and dEq1, the boundary filtering strength bS_L and the variable t_{CL} as inputs and the number of filtered samples nDp and nDq from each side of the block boundary and the filtered sample values p_i' and q_j' as outputs.
   b. The filtered sample values p_i' and q_j' with i = 0..nDp – 1, j = 0..nDq – 1 replace the corresponding samples inside the sample array s' as follows:

$$s'[\ xC + xB + k, yC + yB + j\ ] = q_j' \quad (8\text{-}459)$$

$$s'[\ xC + xB + k, yC + yB - i - 1\ ] = p_i' \quad (8\text{-}460)$$

5. If dE is not equal to 0, for each sample location (xC + xB + k, yC + yB), k = 0.. k_e , the following ordered steps apply:
   a. The filtering process for a luma sample specified in subclause 8.6.1.4.5 is invoked with sample values $p_{i,k}$, $q_{i,k}$ with i = 0..3, decision dS[k], variables dEp1 and dEq1, the boundary filtering strength bS and the variable t_C as inputs and the number of filtered samples nDp and nDq from each side of the block boundary and the filtered sample values p_i' and q_j' as outputs.
   b. The filtered sample values p_i' and q_j' with i = 0..nDp – 1, j = 0..nDq – 1 replace the corresponding samples inside the sample array s' as follows:

$$s'[\ xC + xB + k, yC + yB + j\ ] = q_j' \quad (8\text{-}461)$$

$$s'[\ xC + xB + k, yC + yB - i - 1\ ] = p_i' \quad (8\text{-}462)$$

8.6.1.4.3 Filtering process for chroma block edge

[Ed.: (WJ) cIdx cannot be 0 here]

Inputs of this process are:

– a luma location (xC, yC) specifying the top-left chroma sample of the current coding unit relative to the top left chroma sample of the current picture,
– a luma location (xB, yB) specifying the top-left chroma sample of the current block relative to the top left chroma sample of the current coding unit,
– a variable verticalEdgeFlag,
– a variable bS specifying the boundary filtering strength,
– a variable cIdx specifying the chroma component index.
– a variable xPOS,
– a variable bS_L,
– a variable t_{CL}

Output of this process is:

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000660

– modified reconstruction of the picture.

Let s' be a variable specifying chroma sample array which is derived as follows.

– If cIdx is equal to 1, s' represents the chroma sample array $recPicture_{Cb}$ of the current picture.

– Otherwise (cIdx is equal to 2), s' represents the chroma sample array $recPicture_{Cr}$ of the current picture.

A variable t_C is specified as follows:

– If bS is greater than 2, the variable t_C is specified as Table 8-13 with luma quantization parameter Clip3(0, 55, qP_L + 2) as input,

– Otherwise (bS is equal or less than 2), the variable t_C is specified as Table 8-13 with luma quantization parameter qP_L as input.

Depending on verticalEdgeFlag, the following applies:

– If verticalEdgeFlag is equal to 1, for each sample location (xC + xB, yC + yB + k), k = 0..3, the following ordered steps apply:

1. The sample values p_i and q_i with i = 0..1 are derived as follows:

$$q_i = s'[\ xC + xB + i,\ yC + yB + k\ ] \quad (8\text{-}442)$$

$$p_i = s'[\ xC + xB - i - 1,\ yC + yB + k\ ] \quad (8\text{-}443)$$

2. If bS is greater than 2, the following ordered steps apply:

   a. The filtering process for a sample specified in subclause 8.6.1.4.6 is invoked with sample values p_i, q_i, with i = 0..1, the boundary filtering strength bS and the variable t_C as inputs and the filtered sample values p_0' and q_0' as outputs.

   b. The filtered sample values p_0' and q_0' replace the corresponding samples inside the sample array s' as follows:

$$s'[\ xC + xB\ ,\ yC + yB + k\ ] = q_0' \quad (8\text{-}444)$$

$$s'[\ xC + xB - 1,\ yC + yB + k\ ] = p_0' \quad (8\text{-}445)$$

– Otherwise (verticalEdgeFlag is equal to 0), the following ordered steps apply:

1. If xPOS is equal to 1, the parameters k_s and k_e are set to -1 and 2 respectively. If xD is equal to 2, the parameters k_s and k_e are set to 0 and 2 respectively. Otherwise k_s and k_e are set to 0 and 3 respectively.

2. The sample values p_i and q_i with i = 0..1 and k = k_s..k_e are derived as follows:

$$q_i = s'[\ xC + xB + k,\ yC + yB + i\ ] \quad (8\text{-}446)$$

$$p_i = s'[\ xC + xB + k,\ yC + yB - i - 1\ ] \quad (8\text{-}447)$$

3. If xPOS is less than 0, and if bS_L, is greater than 2, for each sample location (xC + xB - 1, yC + yB), the following ordered steps apply:

   a. The filtering process for a sample specified in subclause 8.6.1.4.6 is invoked with sample values p_i, q_i, with i = 0..1, the boundary filtering strength bS_L and the variable t_{CL} as inputs and the filtered sample values p_0' and q_0' as outputs.

   b. The filtered sample values p_0' and q_0' replace the corresponding samples inside the sample array s' as follows:

$$s'[\ xC + xB + k,\ yC + yB\ ] = q_0' \quad (8\text{-}452)$$

$$s'[\ xC + xB + k,\ yC + yB - i - 1\ ] = p_i' \quad (8\text{-}445)$$

4. If bS is greater than 2, for each sample location (xC + xB + k, yC + yB), k = 0.. k_e, the following ordered steps apply:

   c. The filtering process for a sample specified in subclause 8.6.1.4.6 is invoked with sample values p_i, q_i, with i = 0..1, the boundary filtering strength bS and the variable t_C as inputs and the filtered sample values p_0' and q_0' as outputs.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000661

d. The filtered sample values p_0' and q_0' replace the corresponding samples inside the sample array s' as follows:

$$s'[xC + xB + k, yC + yB] = q_0' \quad (8\text{-}448)$$

$$s'[xC + xB + k, yC + yB - 1] = p_0' \quad (8\text{-}449)$$

8.6.1.4.4 Decision process for a luma sample

[Ed: (WJ) no filtering when bS is equal to 0]

Inputs of this process are:

- sample values, p_i and q_i with i = 0..2,
- a variable bS specifying the boundary filtering strength,
- variables d, β and t_C.

Output of this process is:

- a variable dSam containing a decision

When the variable bS is not equal to 0, the following applies:

- If d is less than (β >> 2), | $p_3 - p_0$ | + | $q_0 - q_3$ | is less than (β >> 3) and | $p_0 - q_0$ | is less than (5*t_C + 1) >> 1, dSam is set equal to 1.
- Otherwise, dSam is set equal to 0.

8.6.1.4.5 Filtering process for a luma sample

Inputs of this process are:

- sample values, p_i and q_i with i = 0..3,
- a variable dSam containing a decision,
- variables dEp1 and dEq1 containing decisions to filter pixels p1 and q1 respectively,
- a variable t_C.

Output of this process is:

- number of filtered samples nDp and nDq,
- filtered sample values, p_i' and q_j' with i = 0..nDp – 1, j = 0..nDq – 1

Depending on dSam, the following applies:

- When the variable dSam is equal to 1, the following strong filtering applies while nDp and nDq are set equal to 3:

$$p_0' = Clip1_Y((p_2 + 2*p_1 + 2*p_0 + 2*q_0 + q_1 + 4) >> 3) \quad (8\text{-}450)$$

$$p_1' = Clip1_Y((p_2 + p_1 + p_0 + q_0 + 2) >> 2) \quad (8\text{-}451)$$

$$p_2' = Clip1_Y((2*p_3 + 3*p_2 + p_1 + p_0 + q_0 + 4) >> 3) \quad (8\text{-}452)$$

$$q_0' = Clip1_Y((p_1 + 2*p_0 + 2*q_0 + 2*q_1 + q_2 + 4) >> 3) \quad (8\text{-}453)$$

$$q_1' = Clip1_Y((p_0 + q_0 + q_1 + q_2 + 2) >> 2) \quad (8\text{-}454)$$

$$q_2' = Clip1_Y((p_0 + q_0 + q_1 + 3*q_2 + 2*q_3 + 4) >> 3) \quad (8\text{-}455)$$

- Otherwise, the following weak filtering applies while nDp and nDq are set equal to 0:

$$\Delta = (9 * (q_0 - p_0) - 3 * (q_1 - p_1) + 8) >> 4 \quad (8\text{-}456)$$

  - When abs(Δ) is less than tc*10, the following ordered steps apply:

    1. The filtered sample values p0' and q0' are specified as follows:

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000662

$$\Delta = Clip3(-tc, tc, \Delta) \quad (8\text{-}456)$$

$$p_0' = Clip1_Y(p_0 + \Delta) \quad (8\text{-}457)$$

$$q_0' = Clip1_Y(q_0 - \Delta) \quad (8\text{-}458)$$

2. If dEp1 is equal to 1, the filtered sample value p_i' is specified as follows:

$$\Delta p = Clip3(-(tc >> 1), tc >> 1, (((p_2 + p_0 + 1) >> 1) - p_1 + \Delta) >> 1) \quad (8\text{-}458)$$

$$p_i' = Clip1_Y(p_1 + \Delta p) \quad (8\text{-}458)$$

3. If dEq1 is equal to 1, the filtered sample value q_i' is specified as follows:

$$\Delta q = Clip3(-(tc >> 1), tc >> 1, (((q_2 + q_0 + 1) >> 1) - q_1 - \Delta) >> 1) \quad (8\text{-}458)$$

$$q_i' = Clip1_Y(q_1 + \Delta q) \quad (8\text{-}458)$$

4. nDp is set equal to dEp1+1 and nDq is set equal to dEq1+1.

Each of the filtered sample values, p_i' with i = 0..nDp-1, is substituted by the corresponding input sample value p_i if all of the following conditions are true.

- p_i is a sample of an I_PCM block.
- pcm_loop_filter_disable_flag value is equal to 1.

Similarly, each of the filtered sample values, q_j' with j = 0..nDq-1, is substituted by the corresponding input sample value q_j if all of the following conditions are true.

- q_j is a sample of an I_PCM block.
- pcm_loop_filter_disable_flag value is equal to 1.

[Ed. (WJ): for PCM case, deblocking filter applies first and the filtered pixels are restored. Rather than this, it's better to skip the filtering itself for PCM samples since first filtering is actually not needed.]

Table 8-15 – Derivation of threshold variables β and t_C from input Q

Q	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
β	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	7	8
t_C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
Q	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
β	9	10	11	12	13	14	15	16	17	18	20	22	24	26	28	30	32	34	36
t_C	1	1	1	1	1	1	1	1	2	2	2	2	3	3	3	3	4	4	4
Q	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	
β	38	40	42	44	46	48	50	52	54	56	58	60	62	64	64	64	64	64	
t_C	5	5	6	6	7	8	9	9	10	10	11	11	12	12	13	13	14	14	

8.6.1.4.6 Filtering process for a chroma sample

[Ed: (WJ) no filtering when bS is equal or less than 2]

Inputs of this process are:

- sample values, p_i and q_i with i = 0..1,
- a variable bS specifying the boundary filtering strength.
- a variable t_C.

Output of this process is:

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000663

– The filtered sample values, p_0' and q_0'.

When the variable bS is greater than 2, the filtered sample values p_0' and q_0' are derived by

$$\Delta = \text{Clip3}(-t_C, t_C, ((((q_0 - p_0) << 2) + p_1 - q_1 + 4) >> 3)) \quad (8\text{-}461)$$

$$p_0' = \text{Clip1}_C(p_0 + \Delta) \quad (8\text{-}462)$$

$$q_0' = \text{Clip1}_C(q_0 - \Delta) \quad (8\text{-}463)$$

The filtered sample value, p_0' is substituted by the corresponding input sample value p_0 if all of the following conditions are true.

– p_0 is a sample of an I_PCM block.

– pcm_loop_filter_disable_flag value is equal to 1.

Similary, the filtered sample value, q_0' is substituted by the corresponding input sample value q_0 if all of the following conditions are true.

– q_0 is a sample of an I_PCM block.

– pcm_loop_filter_disable_flag value is equal to 1.

[Ed. (WJ): for PCM case, deblocking filter applies first and the filtered pixels are restored. Rather than this, it's better to skip the filtering itself for PCM samples since first filtering is actually not needed.]

8.6.2 Sample adaptive offset process

A sample adaptive offset process shall be conditionally performed after the completion of the deblocking filter process for the decoded picture. This process is invoked when both sample_adaptive_offset_enabled_flag and ~~sample_adaptive_offset_flag~~aps_sample_adaptive_offset_flag are equal to 1.

This process is performed on a region basis which is aligned with the LCU boundaries after the completion of the picture construction process prior to sample adaptive offset process for the entire decoded process.

The sample adaptive offset process is invoked with the region equal to entire picture here.

Depending on ~~sample_adaptive_offset_flag~~aps_sample_adaptive_offset_flag, sao_flag_cb and sao_flag_cr, the following applies:

– When ~~sample_adaptive_offset_flag~~aps_sample_adaptive_offset_flag is equal to 1, the following applies:

  – The sample adaptive offset process for luma region as specified in subclause 8.6.2.1 is invoked with the component index set equal to 0, the region index (0, 0), the largest CU location (0, 0), the variable saoDepth set equal to 0, the variable saoWidthInLCUs set equal to PicWidthInLCUs, and the variable saoHeightInLCUs set equal to PicHeightInLCUs, and the modified luma picture buffer of current region as output.

  – When sao_flag_cb is equal to 1, the sample adaptive offset process for color component Cb as specified in subclause 8.6.2.1 is invoked with the component index set equal to 1, the region index (0, 0), the largest CU location (0, 0), the variable saoDepth set equal to 0, the variable saoWidthInLCUs set equal to PicWidthInLCUs, and the variable saoHeightInLCUs set equal to PicHeightInLCUs, and the modified chroma picture buffer for Cb of current region as output.

  – If sao_flag_cr is equal to 1, the sample adaptive offset process for color component Cr as specified in subclause 8.6.2.1 is invoked with the component index set equal to 2, the region index (0, 0), the largest CU location (0, 0), the variable saoDepth set equal to 0, the variable saoWidthInLCUs set equal to PicWidthInLCUs, and the variable saoHeightInLCUs set equal to PicHeightInLCUs, and the modified chroma picture buffer for Cr of current region as output.

8.6.2.1 Modification process for sample adaptive offset

Inputs to this process are:

– a variable cIdx specifying chroma component index,

– a pair of variables (rx, ry) specifying the region index,

– a sample location (x0, y0) specifying the top-left sample of the current largest coding unit,

– a variable saoDepth specifying the split depth of the region,

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000664

– a variable saoWidthInLCUs specifying the number of largest CUs in width of the current region

– a variable saoHeightInLCUs specifying the number of largest CUs in height of the current region.

Output of this process is a modified picture buffer for the chroma component cIdx.

Depending on sao_split_flag[cIdx][saoDepth][rx][ry], the following applies:

– If sao_split_flag[cIdx][saoDepth][rx][ry] is equal to 1 and saoDepth is smaller than SaoMaxDepth, the following ordered stesps apply:

1. Variable nS is set equal to (1 << log2MaxCUSize).
2. Variables saoWidthInLCU1, saoWidthInLCU2, saoHeightInLCU1, saoHeightInLCU2, x1, y1, x2, and y2 are derived as follows:

$$saoWidthInLCU1 = saoWidthInLCUs >> 1 \quad (8\text{-}463)$$
$$saoWidthInLCU2 = saoWidthInLCUs - saoWidthInLCU1 \quad (8\text{-}463)$$
$$saoHeightInLCU1 = saoHeightInLCUs >> 1 \quad (8\text{-}463)$$
$$saoHeightInLCU2 = saoHeightInLCUs - saoHeightInLCU1 \quad (8\text{-}463)$$

$$x1 = x0 \quad (8\text{-}463)$$
$$y1 = y0 \quad (8\text{-}463)$$
$$x2 = x1 + saoWidthInLCU1 * nS \quad (8\text{-}463)$$
$$y2 = y1 + saoHeightInLCU1 * nS \quad (8\text{-}463)$$

3. The sample adaptive offset process for region as specified in this subclause is invoked with the component index cIdx, the region index (2*rx, 2*ry), the top-left sample position of largest CU (x0, y0), the variable saoDepth set equal to (saoDepth+1), the variable saoWidthInLCUs set equal to saoWidthInLCU1, and the variable saoHeightInLCUs set equal to saoHeightInLCU1.
4. The sample adaptive offset process for region as specified in this subclause is invoked with the component index cIdx, the region index (2*rx+1, 2*ry), the top-left sample position of largest CU (x1, y0), the variable saoDepth set equal to (saoDepth+1), the variable saoWidthInLCUs set equal to saoWidthInLCU2, and the variable saoHeightInLCUs set equal to saoHeightInLCU1.
5. The sample adaptive offset process for region as specified in this subclause is invoked with the component index cIdx, the region index (2*rx, 2*ry+1), the top-left sample position of largest CU (x0, y1), the variable saoDepth set equal to (saoDepth+1), the variable saoWidthInLCUs set equal to saoWidthInLCU1, and the variable saoHeightInLCUs set equal to saoHeightInLCU2.
6. The sample adaptive offset process for region as specified in this subclause is invoked with the component index cIdx, the region index (2*rx+1, 2*ry+1), the top-left sample position of largest CU (x1, y1), the variable saoDepth set equal to (saoDepth+1), the variable saoWidthInLCUs set equal to saoWidthInLCU2, and the variable saoHeightInLCUs set equal to saoHeightInLCU2.

– Otherwise, (sao_split_flag[cIdx][saoDepth][rx][ry] is equal to 0), the following ordered steps apply:

1. Variable nS is set equal to (1 << log2MaxCUSize).
2. Modification process for samples in a largest coding unit covering (iC, jC) with iC = x0 + k*nS for k = 0..saoWidthInLCU – 1 and jC = y0 + j*nS for j = 0..saoHeightInLCU – 1 specified in subclause 8.6.2.1.1 is invoked with the component index cIdx, the top-left sample of current largest coding unit (iC, jC), the region index (rx, ry), the split depth saoDepth, the offset array saoValueArray set equal to SaoOffsetVal[cIdx][saoDepth][rx][ry] and the largest coding unit size nS as inputs and the output is the modified picture buffer of chroma component cIdx.

8.6.2.1.1 Modification process for luma and chroma samples

Inputs to this process are:

– a variable cIdx specifying chroma component index,

– a sample position (xC, yC),

– a pair of variables (rx, ry) specifying the region index,

– a variable saoDepth specifying the split depth of the region,

– an array saoValueArray specifying offset values,

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000665

– a block size nS.

Output of this process is a modified picture buffer for the chroma component cIdx.

Let recSaoPicture represents the processed sample array of the current picture of chroma component cIdx and saoTypeIdx is set equal to sao_type_idx[cIdx][saoDepth][rx][ry].

Variable bitDepth is set equal to $BitDepth_Y$ if cIdx is equal to 0, otherwise, set equal to $BitDepth_C$.

Depending on the value of saoTypeIdx, the following applies:

– If saoTypeIdx is equal to one of the values of 1, 2, 3 or 4, the following ordered steps apply:

1. Arrays hPos[2] and vPos[2] are specified in Table 8-16.
2. A variable edgeIdx is specified as

$$edgeIdx = 2 + \sum_k(Sign(recPicture[xC + i, yC + j] - recPicture[xC + i + hPos[k], yC + j + vPos[k]])) \text{ with } k = 0..1 \quad (8\text{-}463)$$

3. The reconstructed picture buffer is modified as

$$recSaoPicture[xC + i, yC + j] = recPicture[xC + i, yC + j] + saoValueArray[edgeTable[edgeIdx]] \quad (8\text{-}463)$$

with i = 0..nS-1 and j = 0..nS-1 where edgeTable[5] = { 1, 2, 0, 3, 4}.

– Otherwise, if saoTypeIdx is equal to one of the values of 5 or 6, the following ordered steps applies:

1. A variable bandShift is set equal to $BitDepth_Y - 5$ if cIdx is equal to 0, otherwise, set equal to $BitDepth_C - 5$.
2. The reconstructed picture buffer is modified as

$$recSaoPicture[xC + i, yC + j] = recPicture[xC + i, yC + j] + saoValueArray[bandTable[saoTypeIdx - 5][bandIdx]] \quad (8\text{-}463)$$

with i = 0..nS-1 and j = 0..nS-1 where bandIdx is set equal to (recPicture[xC + i, yC + j] >> bandShift) and bandTable is specified in Table 8-17.

– Otherwise (sao_type_idx[cIdx][saoDepth][rx][ry] is equal to 0), the following applies:

$$recSaoPicture[xC + i, yC + j] = recPicture[xC + i, yC + j] \text{ with } i = 0..nS-1 \text{ and } j = 0..nS-1 \quad (8\text{-}463)$$

[Ed. (WJ): copy operation is necessary to use recSaoPicture later.]

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000666

Table 8-16 – Specification of hPos[2] and vPos[2] according to the type of sample adaptive offset process

sao_type_idx[cIdx][saoDepth][rx][ry]	1	2	3	4
hPos[0]	-1	0	-1	1
hPos[1]	1	0	1	-1
vPos[0]	0	-1	-1	1
vPos[1]	0	1	1	-1

Table 8-17 – Specification of array bandTable according to bandIdx

bandIdx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
bandTable[0][bandIdx]	0	0	0	0	0	0	0	0	1	2	3	4	5	6	7	8
bandTable[1][bandIdx]	1	2	3	4	5	6	7	8	0	0	0	0	0	0	0	0
bandIdx	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
bandTable[0][bandIdx]	9	10	11	12	13	14	15	16	0	0	0	0	0	0	0	0
bandTable[1][bandIdx]	0	0	0	0	0	0	0	0	9	10	11	12	13	14	15	16

8.6.3 Adaptive loop filter process

An adaptive loop filtering process shall be conditionally performed on a treeblock basis after the completion of the sample adaptive offset process for the entire decoded picture, with all treeblocks in a picture processed in order of increasing treeblock addresses.

Each treeblock is processed on a coding unit basis with the same order as decoding process.

This process is invoked when both adaptive_loop_filter_enabled_flag and ~~adaptive_loop_filter_flag~~aps_adaptive_loop_filter_flag are equal to 1.

This process is performed on a coding unit basis after the completion of the slice construction process prior to adaptive loop filter process for the entire decoded slice, with all coding units in a slice processed in order of coding unit scan order.

Filter coefficients c_L for luma samples and c_C for chroma samples are derived by invoking the process specified in subclause 8.6.2.1.Filter index array fIdx for luma samples are derived by invoking the process specified in subclause 8.6.2.2 with the luma location (xC, yC) specifying the top-left luma sample of the current coding unit relative to the top left luma sample of the current picture and the coding unit size log2CUSize as inputs.

When interpreting the luma samples of the coding unit as to be filtered, depending on pcm_flag, pcm_loop_filter_disable_flag and alf_cu_control_flag, the following applies.

– If pcm_loop_filter_disable_flag is equal to 1 and pcm_flag is equal to 1, the luma samples are not filtered.

– Otherwise, if alf_cu_control_flag is equal to 0, the luma samples are filtered,

– Otherwise (alf_cu_control_flag is equal to 1), if AlfCuFlag[xC][yC] is equal to 1 where xC and yC are the top-left luma sample of the current coding unit relative to the top left luma sample of the current picture, the luma samples are filtered.

– Otherwise, the luma samples are not filtered.

When interpreting the chroma samples of the coding unit as to be filtered, depending on pcm_flag, pcm_loop_filter_disable_flag and alf_chroma_idc, the following applies.

– If pcm_loop_filter_disable_flag is equal to 1 and pcm_flag is equal to 1, both chroma samples are not filtered.

– Otherwise, if alf_chroma_idc is equal to 1, the Cr samples are filtered.

– Otherwise, if alf_chroma_idc is equal to 2, the Cb samples are filtered.

– Otherwise, if alf_chroma_idc is equal to 3, both chroma samples are filtered.

– Otherwise (alf_chroma_idc is equal to 0), both chroma samples are not filtered.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000667

For the luma samples of the coding unit interpreted as to be filtered, the filtering process for luma samples specified in subclause 8.6.2.3 is invoked with the luma location (xC, yC) specifying the top-left luma sample of the current coding unit relative to the top left luma sample of the current picture, the coding unit size log2CUSize, and the filter index array fIdx as inputs and the output is the modified filtered picture, $recFiltPicture_L$.

For the chroma samples of the coding unit interpreted as to be filtered, the filtering process for chroma samples specified in subclause 8.6.2.4 is invoked with the chroma location (xC/2, yC/2) specifying the top-left chroma sample of the current coding unit relative to the top left chroma sample of the current picture, the coding unit size log2CUSize-1 and the chroma component index cIdx equal to 1 as inputs and the output is the modified filtered picture, $recFiltPicture_{Cb}$.

For the chroma samples of the coding unit interpreted as to be filtered, the filtering process for chroma samples specified in subclause 8.6.2.4 is invoked with the chroma location (xC/2, yC/2) specifying the top-left chroma sample of the current coding unit relative to the top left chroma sample of the current picture, the coding unit size log2CUSize-1 and the chroma component index cIdx equal to 2 as inputs and the output is the modified filtered picture, $recFiltPicture_{Cr}$.

[Ed.: (WJ) recPicture: deblocked/output picture and recFiltPicture: ALFed picture]

[Ed.: (WJ) depending ~~adaptive_loop_filter_flag~~aps_adaptive_loop_filter_flag, recFiltPicture should be copied to recPicture in subclause 8.6]

8.6.3.1 Boundary padding process

Inputs of this process are:

- a location (xC, yC) specifying the top-left sample of the current coding unit relative to the top left sample of the current picture,
- a variable log2CUSize specifying the size of the current coding unit.
- a variable cIdx specifying the chroma component index.

Output of this process is the padded luma sample array s''.

Depending on alf_filter_shape, the following applies:

- If alf_filter_shape is equal to 0, a variable alfTapX is set equal to 5, a variable alfTapY is set equal to 5, a variable nExtPixelsX is set equal to (alfTapX >> 1) and a variable nExtPixelsY is set equal to (alfTapY >> 1).
- Otherwise, if alf_filter_shape is equal to 1, a variable alfTapX is set equal to 11, a variable alfTapY is set equal to 5, a variable nExtPixelsX is set equal to (alfTapX >> 1) and a variable nExtPixelsY is set equal to (alfTapY >> 1).

Depending on cIdx, the following applies:

- If cIdx is equal to 0, a luma sample array s' is set equal to $recPicture_L$ and a variable nS is set equal to (1 << log2CUSize).
- Otherwise, if cIdx is equal to 1, a chroma sample array s' is set equal to $recPicture_{Cb}$ and a variable nS is set equal to (1 << (log2CUSize – 1)).
- Otherwise (cIdx is equal to 2), a chroma sample array s' is set equal to $recPicture_{Cr}$ and a variable nS is set equal to (1 << (log2CUSize – 1)).

Depending on loop_filter_across_slice_flag, a varaible s'' specifying the padded luma sample array is derived as:

- If loop_filter_across_slice_flag is equal to 1, s''[xC+x][yC+y] is set equal to s'[xC+x][yC+y] for x = -nExtPixelsX..nS+nExtPixelsX-1 and y = -nExtPixelsY..nS+nExtPixelsY-1.
- Otherwise (loop_filter_across_slice_flag is equal to 0), s''[xC+x][yC+y] is derived by the following ordered steps:

  1. s''[xC+x][yC+y] is set equal to s'[xC+x][yC+y] for x=0..nS – 1 and y=0..nS – 1.
  2. If the left boundary of current block coincides with the slice boundary,

     s''[xC+x][yC+y] = s'[xC][yC+y] for x = -nExtPixelsX..-1 and y = 0..nS-1 (8-462)

  3. If the right boundary of current block coincides with the slice boundary,

     s''[xC+nS+x][yC+y] = s'[xC+nS-1][yC+y] for x = 0..nExtPixelsX-1 and y = 0..nS-1 (8-462)

  4. If the top boundary of current block coincides with the slice boundary,

     s''[xC+x][yC+y] = s'[xC+x][yC] for x = 0..(nP-nExtPixelsX-1) and y = -nExtPixelsY..-1 (8-462)

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000668

where nP is set equal to the size of top boundary that needs padding. [Ed. (WJ): needs improvements]

– If current block and s'[xC + nP][yC – 1] are not at the same slice,

s''[xC+x][yC+y] = s'[xC+x][yC] for x = (nP-nExtPixelsX)..(nP-1) and y = -nExtPixelsY..-1 (8-462)

– Otherwise,

s''[xC+x][yC+y] = s'[xC+nP][yC+y] for x = (nP-nExtPixelsX)..(nP-1) and y = -nExtPixelsY..-1 (8-462)

5. If the bottom boundary of current block coincides with the slice boundary,

s''[xC+ x][yC+nS+y] = s'[xC+x][yC+nS-1]
for x = 0..(nS-nP+nExtPixelsX) and y = 0..nExtPixelsY-1 (8-462)

where nP is set equal to the size of bottom boundary that needs padding. [Ed. (WJ): needs improvements]

– If current block and s'[xC+nS-nP-1][yC+nS] are not at the same slice,

s''[xC+x][yC+nS+y] = s'[xC+x][yC+nS-1]
where x = (nS-nP)..(nS-nP+nExtPixelsX-1) and y = 0..nExtPixelsY-1 (8-462)

– Otherwise,

s''[xC+x][yC+nS+y] = s'[xC+nS-nP-1][yC+nS+y]
where x = (nS-nP)..(nS-nP+nExtPixelsX-1) and y = 0..nExtPixelsY-1 (8-462)

6. If the current coding unit and its upper-left coding unit are not at the same slice,

– If the left boundary of current block coincides with the slice boundary but the top boundary of current block does not,

s''[xC+x][yC+y] = s'[xC][yC+y] where x = -nExtPixelsX..-1 and y = -nExtPixelsY..-1 (8-462)

– Otherwise, if the top boundary of current block coincides with the slice boundary but the left boundary of current block does not,

s''[xC+x][yC+y] = s'[xC+x][yC] where x = -nExtPixelsX..-1 and y = -nExtPixelsY..-1 (8-462)

– Otherwise

s''[xC+x][yC+y] = s'[xC][yC] where x = -nExtPixelsX..-1 and y = -nExtPixelsY..-1 (8-462)

7. If the current coding unit and its upper-right coding unit are not at the same slice,

– If the right boundary of current block coincides with the slice boundary but the top boundary of current block does not,

s''[xC+nS+x][yC+y] = s'[xC+nS-1][yC+y] where x = 0..nExtPixelsX-1 and y = -nExtPixelsY..-1 (8-462)

– Otherwise, if the top boundary of current block coincides with the slice boundary but the right boundary of current block does not,

s''[xC+nS+x][yC+y] = s'[xC+nS+x][yC] where x = 0..nExtPixelsX-1 and y = -nExtPixelsY..-1 (8-462)

– Otherwise

s''[xC+nS+x][yC+y] = s'[xC+nS-1][yC] where x = 0..nExtPixelsX-1 and y = -nExtPixelsY..-1 (8-462)

8. If the current coding unit and its bottom-left coding unit are not at the same slice,

– If the left boundary of current block coincides with the slice boundary but the bottom boundary of current block does not,

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000669

s''[xC+x][yC+nS+y] = s'[xC][yC+nS+y] where x = -nExtPixelsX..-1 and y = 0..nExtPixelsY-1 (8-462)

– Otherwise, if the bottom boundary of current block coincides with the slice boundary but the left boundary of current block does not,

s''[xC+x][yC+nS+y] = s'[xC+x][yC+nS-1] where x = -nExtPixelsX..-1 and y = 0..nExtPixelsY-1 (8-462)

– Otherwise

s''[xC+ x][yC+nS+y] = s'[xC][yC+nS-1] where x = -nExtPixelsX..-1 and y = 0..nExtPixelsY-1 (8-462)

9. If the current coding unit and its bottom-right coding unit are not at the same slice,

– If the right boundary of current block coincides with the slice boundary but the bottom boundary of current block does not,

s''[xC+nS+x][yC+nS+y] = s'[xC+nS-1][yC+nS+y]
where x = 0..nExtPixelsX-1 and y = 0..nExtPixelsY-1 (8-462)

– Otherwise, if the bottom boundary of current block coincides with the slice boundary but the right boundary of current block does not,

s''[xC+nS+x][yC+nS+y] = s'[xC+nS+x][yC+nS-1]
where x = 0..nExtPixelsX-1 and y = 0..nExtPixelsY-1 (8-462)

– Otherwise

s''[xC+ nS+x][yC+nS+y] = s'[xC+nS-1][yC+nS-1]
where x = 0..nExtPixelsX-1 and y = 0..nExtPixelsY-1 (8-462)

8.6.3.2 Derivation process for filter coefficients

Outputs of this process are filter coefficients c_L for the luma samples and filter coefficients c_C for the chroma samples.

The luma filter coefficients c_L with elements $c_L[i][j]$, i = 0..AlfNumFilters – 1, j = 0..AlfCodedLengthLuma – 1 is derived as follows:

– If alf_pred_method is equal to 0 or the value of i is equal to 0,

$c_L[i][j]$ = alf_coeff_luma[i][j] (8-464)

– Otherwise (alf_pred_method is equal to 1 and the value of i is greater than 1),

$c_L[i][j] = alf\_coeff\_luma[i][j] + c_L[i - 1][j]$ (8-465)

Considering the symmetry of the filter, the luma filter coefficients c_L with elements $c_L[i][j]$, i = 0..AlfNumFilters – 1 is derived as follows:

$c_L[i][AlfLengthLuma - 1] = c_L[i][AlfCodedLengthLuma - 1]$ (8-466)

$c_L[i][AlfLengthLuma - 2 - j] = c_L[i][j]$ for j = 0..AlfCodedLengthLuma – 2 (8-467)

The chroma filter coefficients c_C with elements $c_C[i]$, i = 0..AlfCodedLengthChroma – 1 is derived as follows:

– If i is equal to AlfCodedLengthChroma – 1, the coefficient $c_C[i]$ is derived as

$c_C[i] = 255 - sum - alf\_coeff\_chroma[i]$ (8-468)

where

$sum = alf\_coeff\_chroma[AlfCodedLengthChroma - 2] + \Sigma_j(alf\_coeff\_chroma[j] << 1)$ (8-469)

with j = 0..AlfCodedLengthChroma – 3

– Otherwise (i is less than AlfCodedLengthChroma – 1),

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000670

$$cc[i] = alf\_coeff\_chroma[i] \quad (8\text{-}470)$$

Considering the symmetry of the filter, the chroma filter coefficients c_C with elements $c_C[i]$, i = 0..AlfLengthChroma – 1 is derived as follows:

$$cc[i][AlfLengthChroma - 1] = cc[i][AlfCodedLengthChroma - 1] \quad (8\text{-}471)$$

$$cc[i][AlfLengthChroma - 2 - j] = cc[i][j] \text{ for } j = 0..AlfCodedLengthChroma - 2 \quad (8\text{-}472)$$

8.6.3.3 Derivation process for filter index array for luma samples

Inputs of this process are:

– a luma location (xC, yC) specifying the top-left luma sample of the current coding unit relative to the top left luma sample of the current picture,

– a variable log2CUSize specifying the size of the current coding unit.

Output of this process is the two-dimensional filter index array of (nS)x(nS), fIdx.

A variable nS is set equal to (1 << log2CUSize).

The boundary padding process specified in subclause 8.6.3.1 is invoked with the luma location (xC, yC), the size of coding unit log2CUSize and the chroma component index cIdx set equal to 0, and the output is assigned to the luma sample array s''. [Ed. (WJ): s'' is now a picture-size array, but actually CU size + appropriate margin is enough]

The filter index array fIdx is specified in the follows:

– When alf_region_adaptation_flag is equal to 1, the following ordered steps apply.

1. The variables xIdx and yIdx are derived as

$$regionTab[16] = \{ 0, 1, 4, 5, 15, 2, 3, 6, 14, 11, 10, 7, 13, 12, 9, 8 \} \quad (8\text{-}474)$$
$$offset = 1 << (Log2MaxCUSize - 1) \quad (8\text{-}474)$$
$$xInterval = (((PicWidthInSamples_L + offset) >> Log2MaxCUSize) + 1) >> 2 \quad (8\text{-}474)$$
$$yInterval = (((PicHeightInSamples_L + offset) >> Log2MaxCUSize) + 1) >> 2 \quad (8\text{-}474)$$
$$xIdx = Min(3, Floor((xC + x) / (xInterval << Log2MaxCUSize))) \quad (8\text{-}474)$$
$$yIdx = Min(3, Floor((yC + y) / (yInterval << Log2MaxCUSize))) \quad (8\text{-}474)$$

2. The filter index fIdx[x, y] with x, y = 0..(nS)-1 is derived as

$$fIdx[x][y] = regionTab[(yIdx << 2) + xIdx] \quad (8\text{-}474)$$

– Otherwise (alf_region_flag is equal to 0), the following ordered steps apply.

1. The variables varTempH[x][y], varTempV[x][y] and varTemp1[x][y] with x, y = 0, 2, 4,...(nS) is derived as

$$varTempH[x][y] = | (s''[xC+x, yC+y] << 1) - s''[xC+x-1, yC+y] - s''[xC+x+1, yC+y] | \quad (8\text{-}474)$$
$$varTempV[x][y] = | (s''[xC+x, yC+y] << 1) - s''[xC+x, yC+y-1] - s''[xC+x, yC+y+1] | \quad (8\text{-}474)$$

2. The variables varTempH1[x, y], varTempV1[x, y], and varTemp3[x, y] with x, y = 0..((nS) – 1)>>2 are derived as

$$varTempH1[x][y] = \Sigma_i\Sigma_j\, varTempH[(x << 2) + i][(y << 2) + j] \text{ with } i, j = 0, 2 \quad (8\text{-}474)$$
$$varTempV1[x][y] = \Sigma_i\Sigma_j\, varTempV[(x << 2) + i][(y << 2) + j] \text{ with } i, j = 0, 2 \quad (8\text{-}474)$$
$$varTemp3[x][y] = (varTempH1[x][y] + varTempV1[x][y]) >> 2 \quad (8\text{-}474)$$

3. The variable direction is derived as

– If varTempV1[x >> 2][y >> 2] is greater than varTempH1[x >> 2][y >> 2] << 1,

direction = 1

– Otherwise, if varTempH1[x >> 2][y >> 2] is greater than varTempV1[x >> 2][y >> 2] << 1,

direction = 2

– Otherwise,

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000671

direction = 0

4. The variable avgvar is derived as

varTab[16] = { 0, 1, 2, 2, 2, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 4 } (8-475)
avgVar = Clip3(0, 15, (varTemp3[x >> 2][y >> 2] * 1024) >> (3 + BitDepth$_Y$)) (8-475)

5. The filter index fIdx[x, y] with x, y = 0..(nS)-1 is derived as

fIdx[x][y] = Clip3(0, 4, varTab[avgVar]) + 5 * direction (8-504)

8.6.3.4 Filtering process for luma samples

Inputs of this process are:

– a luma location (xC, yC) specifying the top-left luma sample of the current coding unit relative to the top left luma sample of the current picture,

– a variable log2CUSize specifying the size of the current coding unit,

– a filter index array of (nS)x(nS), fIdx.

Output of this process is the filtered reconstruction of luma picture.

The boundary padding process specified in subclause 8.6.3.1 is invoked with the luma location (xC, yC), the size of coding unit log2CUSize and the chroma component index cIdx set equal to 0, and the output is assigned to the luma sample array s''. [Ed. (WJ): s'' is now a picture-size array, but actually CU size + appropriate margin is enough]

A variable nS is set equal to (1 << log2CUSize) and a variable alfTap is set equal to (alf_length_luma_minus_5_div2 << 1) + 5.

Each sample of luma picture recFiltPicture$_L$[xC + x][yC + y] with x, y = 0..(nS)-1, is derived as follows.

$$recFiltPicture_L[xC+x][yC+y] = c_L[fIdx[x][y]][N] + \sum_{i=0}^{N-1}\left(s''[xC+x+horPos[i],\ yC+y+verPos[i]] * c_L[fIdx[x][y]][i]\right) \quad (8\text{-}476)$$

where N is set equal to AlfNumCoeffLuma-1 and horPos[i] and verPos[i] are specified in Table 8-14 and Table 8-15, respectively.

Table 8-18 – Specification of horPos[i] according to alfTap for adaptive loop filter process of luma samples

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
alfTap = 5	0	-1	0	1	-2	-1	0	1	2	-1	0	1	0	-	-	-	-	-	-	-
alfTap = 7	0	-1	0	1	-2	-1	0	1	2	-3	-2	-1	0	1	2	3	-2	-1	0	1
alfTap = 9	-1	0	1	-2	-1	0	1	2	-3	-2	-1	0	1	2	3	-4	-3	-2	-1	0

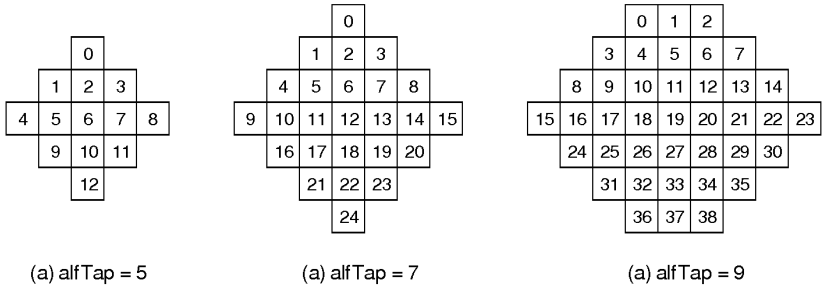
i	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38
alfTap = 5	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
alfTap = 7	2	-1	0	1	0	-	-	-	-	-	-	-	-	-	-	-	-	-	-
alfTap = 9	1	2	3	4	-3	-2	-1	0	1	2	3	-2	-1	0	1	2	-1	0	1

Table 8-19 – Specification of verPos[i] according to alfTap for adaptive loop filter process of luma samples

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
alfTap = 5	-2	-1	-1	-1	0	0	0	0	0	1	1	1	2							
alfTap = 7	-3	-2	-2	-2	-1	-1	-1	-1	-1	0	0	0	0	0	0	0	1	1	1	1
alfTap = 9	-3	-3	-3	-2	-2	-2	-2	-2	-1	-1	-1	-1	-1	-1	-1	0	0	0	0	0

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000672

i	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38
alfTap = 5	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
alfTap = 7	1	2	2	2	3	-	-	-	-	-	-	-	-	-	-	-	-	-	-
alfTap = 9	0	0	0	0	1	1	1	1	1	1	1	2	2	2	2	2	3	3	3



(a) alfTap = 5 (a) alfTap = 7 (a) alfTap = 9

Figure 8-8 Mapping between geometric position and luma adaptive loop filter index according to alfTap (informative)

8.6.3.5 Filtering process for chroma samples

Inputs of this process are:

– a chroma location (xC, yC) specifying the top-left luma sample of the current coding unit relative to the top left chroma sample of the current picture,

– a variable log2CUSize specifying the size of the current coding unit.

– a variable cIdx specifying the chroma component index.

Output of this process is the filtered reconstruction of chroma picture.

The boundary padding process specified in subclause 8.6.3.1 is invoked with the chroma location (xC, yC), the size of coding unit log2CUSize and the chroma component index cIdx, and the output is assigned to the luma sample array s''. [Ed. (WJ): s'' is now a picture-size array, but actually CU size + appropriate margin is enough]

A variable nS is set equal to (1 << log2CUSize) and a variable alfTapChroma is set equal to (alf_length_chroma_minus_5_div2 << 1) + 5.

Filtered samples of chroma picture recFiltPicture[xC + x][yC + y] with x, y = 0..(nS)-1, are derived as follows.

$$recFiltPicture[xC+x][yC+y] = c_C[N] + \sum_{i=0}^{N-1} \left(s''[\, xC+x+horPos[i],\, yC+y+verPos[i]\,] * c_C[i] \right) \quad (8\text{-}477)$$

where

$$N = AlfNumCoeffChroma - 1, \quad (8\text{-}478)$$

horPos[i] = (i % alfTapChroma) – (alfTapChroma >> 1), and (8-479)

verPos[i] = (i / alfTapChroma) – (alfTapChroma >> 1) (8-480)

9 Parsing process

Inputs to this process are bits from the RBSP.

Outputs of this process are syntax element values.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000673

This process is invoked when the descriptor of a syntax element in the syntax tables in subclause 7.3 is equal to ue(v), me(v), se(v), te(v) (see subclause 9.1), ce(v) (see subclause 9.2), or ae(v) (see subclause 9.3).

9.1 Parsing process for Exp-Golomb codes

This process is invoked when the descriptor of a syntax element in the syntax tables in subclause 7.3 is equal to ue(v), me(v), se(v), or te(v). For syntax elements in subclauses 7.3.4 and 7.3.5, this process is invoked only when entropy_coding_mode_flag is equal to 0.

Inputs to this process are bits from the RBSP.

Outputs of this process are syntax element values.

Syntax elements coded as ue(v), me(v), or se(v) are Exp-Golomb-coded. Syntax elements coded as te(v) are truncated Exp-Golomb-coded. The parsing process for these syntax elements begins with reading the bits starting at the current location in the bitstream up to and including the first non-zero bit, and counting the number of leading bits that are equal to 0. This process is specified as follows:

```
leadingZeroBits = −1
for( b = 0; !b; leadingZeroBits++ )                                    (9-1)
    b = read_bits( 1 )
```

The variable codeNum is then assigned as follows:

$$codeNum = 2^{leadingZeroBits} - 1 + read\_bits(leadingZeroBits) \quad (9\text{-}2)$$

where the value returned from read_bits(leadingZeroBits) is interpreted as a binary representation of an unsigned integer with most significant bit written first.

Table 9-1 illustrates the structure of the Exp-Golomb code by separating the bit string into "prefix" and "suffix" bits. The "prefix" bits are those bits that are parsed in the above pseudo-code for the computation of leadingZeroBits, and are shown as either 0 or 1 in the bit string column of Table 9-1. The "suffix" bits are those bits that are parsed in the computation of codeNum and are shown as x_i in Table 9-1, with i being in the range 0 to leadingZeroBits − 1, inclusive. Each x_i can take on values 0 or 1.

Table 9-1 – Bit strings with "prefix" and "suffix" bits and assignment to codeNum ranges (informative)

Bit string form	Range of codeNum
1	0
0 1 x_0	1..2
0 0 1 x_1 x_0	3..6
0 0 0 1 x_2 x_1 x_0	7..14
0 0 0 0 1 x_3 x_2 x_1 x_0	15..30
0 0 0 0 0 1 x_4 x_3 x_2 x_1 x_0	31..62
...	...

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000674

Table 9-2 illustrates explicitly the assignment of bit strings to codeNum values.

Table 9-2 – Exp-Golomb bit strings and codeNum in explicit form and used as ue(v) (informative)

Bit string	codeNum
1	0
0 1 0	1
0 1 1	2
0 0 1 0 0	3
0 0 1 0 1	4
0 0 1 1 0	5
0 0 1 1 1	6
0 0 0 1 0 0 0	7
0 0 0 1 0 0 1	8
0 0 0 1 0 1 0	9
...	…

Depending on the descriptor, the value of a syntax element is derived as follows.

- If the syntax element is coded as ue(v), the value of the syntax element is equal to codeNum.
- Otherwise, if the syntax element is coded as se(v), the value of the syntax element is derived by invoking the mapping process for signed Exp-Golomb codes as specified in subclause 9.1.1 with codeNum as the input.
- Otherwise, if the syntax element is coded as me(v) [Ed. (TW): insert text]
- Otherwise (the syntax element is coded as te(v)), the range of possible values for the syntax element is determined first. The range of this syntax element may be between 0 and x, with x being greater than or equal to 1 and the range is used in the derivation of the value of the syntax element value as follows.
  - If x is greater than 1, codeNum and the value of the syntax element is derived in the same way as for syntax elements coded as ue(v).
  - Otherwise (x is equal to 1), the parsing process for codeNum which is equal to the value of the syntax element is given by a process equivalent to:

```
b = read_bits( 1 )                                    (9-3)
codeNum = !b
```

9.1.1 Mapping process for signed Exp-Golomb codes

Input to this process is codeNum as specified in subclause 9.1.

Output of this process is a value of a syntax element coded as se(v).

The syntax element is assigned to the codeNum by ordering the syntax element by its absolute value in increasing order and representing the positive value for a given absolute value with the lower codeNum. Table 9-3 provides the assignment rule.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000675

Table 9-3 – Assignment of syntax element to codeNum for signed Exp-Golomb coded syntax elements se(v)

codeNum	syntax element value
0	0
1	1
2	−1
3	2
4	−2
5	3
6	−3
k	$(-1)^{k+1}$ Ceil(k÷2)

9.2 CAVLC parsing process for slice data

This process is invoked when the descriptor of a syntax element in the syntax tables in subclause 7.3 is equal to ce(v).

Inputs to this process are bits from the RBSP.

Outputs of this process are syntax element values.

9.2.1 Parsing process for VLC codes

Inputs to this process are bits from the RBSP and the parameter vlcNum specifying the VLC code. If vlcNum is equal to 14, inputs to this process additionally include a parameter cMax.

Output of this process is the variable codeNum specifying an integer value to be used when deriving syntax elements.

If vlcNum is not equal to 8 or 11, and is less than 14, the parsing process for the VLC codes begins with reading the bits starting at the current location in the bitstream up to and including the first non-zero bit, and counting the number of leading bits that are equal to 0. This process is specified as follows.

```
leadingZeroBits = −1
for( b = 0; !b; leadingZeroBits++ )                                    (9-4)
    b = read_bits( 1 )
```

Depending on the value of vlcNum, the value of codeNum is assigned as follows.

– If vlcNum is equal to 0,
  – If leadingZeroBits is less than or equal to 6,

```
codeNum = leadingZeroBits                                              (9-5)
```

  – Otherwise (leadingZeroBits is greater than 6),

```
numBits =  leadingZeroBits – 6                                         (9-6)
codeNum = 5 + (1<< numBits) + read_bits( numBits )
```

– Otherwise, if vlcNum is less than 5,
  – If leadingZeroBits is less than or equal to 6,

```
codeNum = (leadingZeroBits << vlcNum) +  read_bits( vlcNum )           (9-7)
```

  – Otherwise (leadingZeroBits is greater than 6),

```
numBits =  leadingZeroBits – 6 + vlcNum                                (9-8)
codeNum = 5*(1<<vlcNum) + (1<< numBits) + read_bits( numBits)
```

– Otherwise, if vlcNum is less than 8,

```
codeNum = (leadingZeroBits << (vlcNum - 4)) +  read_bits( vlcNum - 4)  (9-9)
```

– Otherwise, if vlcNum is equal to 8,

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000676

```
b =  read_bits(1)                                                    (9-10)
codeNum = b ? 2 - read_bits(1) : 0
```

– Otherwise, if vlcNum is equal to 9,

  – If leadingZeroBits is equal to 0,

```
b =  read_bits(1)
if (b) {
    b =  read_bits(1)
    if (b) {
        codeNum = 3 + read_bits(3)                                   (9-11)
    }
    else{
        b =  read_bits(1)
        codeNum = b ? 1 + read_bits(1) : 0
    }
}
```

  – Otherwise (leadingZeroBits is greater than 0),

```
codeNum = (leadingZeroBits << 4) +  read_bits( 4) + 11               (9-12)
```

– Otherwise, if vlcNum is equal to 10,

```
codeNum = (1<< leadingZeroBits) – 1 + read_bits( leadingZeroBits )   (9-13)
```

– Otherwise, if vlcNum is equal to 11,

```
codeNum = read_bits(3)
if( codeNum ) {
    b = read_bits(1)                                                 (9-14)
    codeNum = (codeNum << 1) + b -1
}
```

– Otherwise, if vlcNum is equal to 12,

```
codeNum = (leadingZeroBits << 6) + read_bits( 6 )                    (9-15)
```

– Otherwise, if vlcNum is equal to 13,

```
codeNum = (leadingZeroBits << 4) + read_bits( 4 )                    (9-16)
```

– Otherwise, if vlcNum is equal to 14,

```
leadingZeroBits = –1
for( b = 0; !b && leadingZeroBits < cMax; leadingZeroBits++ )        (9-17)
    b = read_bits( 1 )
codeNum = leadingZeroBits
```

– Otherwise, if vlcNum is equal to 15, codeNum is determined according to Table 9-4.

– Otherwise, if vlcNum is equal to 16

```
codeNum = read_bits(3)
if( codeNum >= 6)
    codeNum = 7 - codeNum
else if (codeNum > 0) {                                              (9-18)
    b = read_bits( 1 )
    codeNum = 13 – (codeNum << 1) - b
}
else {
    b = read_bits( 2 )
    codeNum = 15 – b
}
```

– Otherwise, if vlcNum is equal to 17

```
codeNum = read_bits(3)
if( codeNum ) {                                                      (9-19)
    b = read_bits(1)
```


Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000677

```
        codeNum = 16 -  (codeNum << 1) - b
    }
```

– Otherwise, if vlcNum is equal to 18

```
    codeNum = read_bits(3)
    if( codeNum && codeNum < 4) {
        b = read_bits(1)
        codeNum = 8 -  (codeNum << 1) - b
    } else if (codeNum < 6) {
        b = read_bits(2)
        codeNum = 30 -  (codeNum << 2) - b                    (9-20)
    } else {
        b = read_bits(3)
        codeNum =  (codeNum << 3) + b
        if (codeNum < 62)
            codeNum = 76 – codeNum
        else {
            b = read_bits(1)
            codeNum = 156 - (codeNum << 1) – b
        }
    }
```

– Otherwise, if vlcNum is equal to 19

```
    codeNum = read_bits(3)
    if( codeNum && codeNum < 3) {
        b = read_bits(1)
        codeNum = 6 -  (codeNum << 1) - b
    } else {                                                  (9-21)
        b = read_bits(2)
        codeNum =  (codeNum << 2) + b
        if (codeNum < 25)
            codeNum = 29 – codeNum
        else {
            b = read_bits(1)
            codeNum = 81 - (codeNum << 1) – b
        }
    }
```

– Otherwise, if vlcNum is equal to 20, codeNum is determined according to Table 9-5.

– Otherwise, if vlcNum is equal to 21, codeNum is determined according to Table 9-6.

Table 9-4 – codeNum and codeword with vlcNum equal to 15

codeNum	Codeword
0	0
1	11
2	1001
3	1011
4	1000
5	10100
6	101010
7	101011

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000678

Table 9-5 – codeNum and codeword with vlcNum equal to 20

codeNum	Codeword
0	0000
1	01110
2	01101
3	01100
4	01011
5	01010
6	01001
7	01000
8	00111
9	00110
10	00101
11	00100
12	00011
13	00010
14	011111
15	011110

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000679

Table 9-6 – codeNum and codeword with vlcNum equal to 21

codeNum	Codeword
0	0000
1	00101
2	00100
3	00011
4	00010
5	011000
6	010111
7	010110
8	010101
9	010100
10	010011
11	010010
12	010001
13	010000
14	001111
15	001110
16	001101
17	001100
18	0111110
19	0111101
20	0111100
21	0111011
22	0111010
23	0111001
24	0111000
25	0110111
26	0110110
27	0110101
28	0110100
29	0110011
30	0110010
31	01111111
32	01111110

9.2.2 Initialisation process

Outputs of this process are initialised CAVLC internal variables.

The processes in subclauses 9.2.2.2 through 9.2.2.6 are invoked when starting the parsing of the slice data of a slice in subclause 7.3.4.

9.2.2.1 Counter initialisation for codeword index adaptive mapping

Outputs of this process are initial values of counters that are used for parsing a syntax element.

If the number of counters used for a syntax element is N (N>0), values of the N counters are all initialized to 0. For each set of counters used for parsing a syntax element, there is an associated value, counterSum, to store the number of counter increments since its previous reset. The value of counterSum is also initialized to 0.

9.2.2.2 Initialisation process for lastPosVlcNumIndex

Outputs of this process are initial values of the variable array lastPosVlcNumIndex.

The variable array lastPosVlcIndex is initialized as follows.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000680

lastPosVlcNumIndex [blockType] = 0, for blockType = 0...7 (9-21)

9.2.2.3 Initialisation process for lastPosTable

Outputs of this process are initial values of the variable array lastPosTable.

The variable array lastPosTable is initialized as specified in Table 9-5.

Table 9-7 – Specification of initial values of lastPosTable[tableNum][codeNum]

	codeNum															
tableNum	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	0	1	2	3	5	4	7	6	11	8	14	9	16	15	10	13
1	0	1	2	5	7	4	6	3	9	12	11	8	10	14	13	15
2	0	1	2	5	7	4	6	3	9	12	11	8	10	14	13	15

	codeNum															
tableNum	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
0	12	17	18	19	22	31	23	21	27	20	25	30	24	26	28	29
1	16	17	21	22	25	18	28	23	30	26	20	19	27	24	29	31
2	16	17	21	22	25	18	28	23	30	26	20	19	27	24	29	31

9.2.2.4 Initialisation process for splitPredPartModeTable

Outputs of this process are initial values of the variable array splitPredPartModeTable and associated counters.

The variable array splitPredPartModeTable is initialized as specified in Table 9-8.

Table 9-8 – Specification of splitPredPartModeTable[cuDepth][codeNum]

	codeNum										
cuDepth	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
0	0	1	2	3	4	5	6	7	8	9	10
1	0	1	2	3	4	5	6	7	8	9	10
2	0	1	2	3	4	5	6	7	8	9	10
3	1	2	3	4	5	0	6	7	8	9	10

For each cuDepth with a value from 0 to (Log2MaxCUSize – Log2MinCUSize), there are four counters used for codeword index adaptive mapping. These counters are initialized according to subclause 9.2.2.1.

9.2.2.5 Initialisation process for intraModeTable

Outputs of this process are initial values of the variable array intraModeTable.

The variable array intraModeTable is initialized as specified in Table 9-6.

Table 9-7 – Specification of intraModeTable[k][codeNum]

	codeNum																
k	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
0	0	3	2	15	8	11	1	10	7	4	14	9	6	5	13	12	
1	0	2	3	29	1	8	30	21	28	16	7	15	20	31	9	11	6

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000681

	codeNum																
k	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**	**32**	**33**
1	4	5	12	10	14	22	19	17	27	13	18	23	26	32	24	25	

9.2.2.6 Initialisation process for cbpSplitTransTable

Outputs of this process are initial values of the variable array cbpSplitTransTable and associated counters.

The values of the variable array cbpSplitTransTable[k][n] are initialized as specified in Table 9-7 where the values of k = 0...5 and n = 0,1. The number of counters used in each case for codeword index adaptive mapping is defined in Table 9-16. These counters are initialized according to subclause 9.2.2.1.

Table 9-8 – Specification of initial values of cbpSplitTransTable[flagPattern-8][n][codeNum]

	k (n = 0)						k (n = 1)					
codeNum	**0**	**1**	**2**	**3**	**4**	**5**	**0**	**1**	**2**	**3**	**4**	**5**
0	14	2	3	6	4	4	0	2	0	0	0	0
1	13	3	1	4	7	7	1	3	2	5	4	4
2	12	1	0	5	0	0	3	0	1	4	5	5
3	10	0	2	7	5	5	7		3	7	6	6
4	6			0	6	6	2			6	1	1
5	11			1	1	1	4			2	7	7
6	9			2	2	2	5			3	2	2
7	8			3	3	3	8			1	3	3
8	5						9					
9	4						11					
10	2						6					
11	7						10					
12	3						12					
13	1						13					
14	0						14					

9.2.3 Codeword index mapping update process

Inputs to this process are a VLC codeword index mapping table indexMappingTable that provides the mapping between VLC codeword indexes and syntax element values, a codeword index value codeNum, and a value counterNum that specifies the number of counters used. If counterNum is not equal to 0, inputs additionally include an array counterArray indexed from 0 to (counterNum-1) that saves the value of counters, a value counterSum that saves the number of all counter increments since its previous reset.

Outputs of this process are indexMappingTable, and if counterNum is not equal to 0 counterArray and counterSum with updated values.

The process is specified in the following ordered steps:

– Syntax element value syntaxVal is set equal to indexMappingTable [codeNum].

– If codeNum is not less than 1, codeNumPre is set equal to (codeNum-1). Otherwise, codeNumPre is set equal to codeNum. Syntax element value syntaxValPre is set equal to indexMappingTable [codeNumPre].

– If codeNum is greater or equal to counterNum, the values of indexMappingTable[codeNumPre] and indexMappingTable[codeNum] are exchanged with each other. No further step is carried out. Otherwise

  – The value of counterArray[codeNum] is increased by 1.

  – If counterArray[codeNum] >= counterArray[codeNumPre], the counter values of counterArray[codeNum] and counterArray[codeNumPre] are swapped, and the values of indexMappingTable [codeNumPre] and indexMappingTable [codeNum] are also swapped.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000682

– If counterSum is not less than 15, the value of counterSum is reset to 0. For i = 0…(counterNum-1), counterArray[i] is set equal to (counterArray[i]>>1).

– Otherwise, the value of counterSum is increased by 1.

9.2.4 Parsing process for transform coefficient syntax elements

This process is invoked when parsing syntax elements with descriptor equal to ce(v) in subclause 7.3.11 and 7.3.12 and when entropy_coding_mode_flag is equal to 0.

Inputs to this process are bits from RBSP, and the variables log2TrafoSize and cIdx, and the variable arrays lastPosVlcNumIndex and lastPosTable.

Outputs of this process are CAVLC syntax elements last_pos_level_one, level_minus2_and_sign, run_level_one, and level, and the variable arrays lastPosTable and lastPosVlcNumIndex.

9.2.4.1 Parsing process for last_pos_level_one

Inputs to this process are bits from RBSP and the variable arrays lastPosVlcNumIndex and lastPosTable.

Output of this process is the syntax element last_pos_level_one, the variables trOne and vlcNumLevel, and the variable arrays lastPosVlcNumIndex and lastPosTable.

The parsing process for last_pos_level_one is specified as follows.

– The variable N is set equal to (1 << log2TrafoSize) >> (cIdx > 0 ? 2 : 0).

– If N is greater than 4, last_pos_level_one is derived in the following ordered steps:

1. The variable blockType is derived as

$$blockType = (cIdx == 0 \ ? (PredMode==MODE\_INTRA \ ? 0 : slice\_type + 1) + (N > 8 \ ? 5 : 2) : cIdx - 1) \quad (9\text{-}23)$$

2. The variable vlcNumIndex is derived as

$$vlcNumIdx = Min(16, lastPosVlcNumIndex[blockType]) \quad (9\text{-}24)$$

3. The variable vlcNum is derived as

$$vlcNum = lastPosVlcNumTable[blockType][vlcNumIdx] \quad (9\text{-}25)$$

   The values of the array lastPosVlcNumTable are shown in Table 9-8.

4. The parsing process described in subclause 9.2.1 is invoked with vlcNum as input and the variable codeNum as output.

5. The variable lastPos is derived as follows

```
if (codeNum <= N+N*N)
    if (codeNum != 0 && (codeNum & (N-1)) = = 0)
        lastPos = (codeNum >> Log2(N)) – 1                    (9-26)
    else
        lastPos = codeNum – (codeNum >> Log2(N))
else
    lastPos = codeNum – N*N
```

6. The variable levelGreaterThanOneFlag is derived as

$$levelGreaterThanOneFlag = (codeNum > (N + N*N)) \ || \ (codeNum != 0 \ \&\& \ (codeNum \ \& \ (N-1))) \quad (9\text{-}27)$$

7. The syntax element last_pos_level_one is derived as

$$last\_pos\_level\_one = levelGreaterThanOneFlag \ ? \ codeNum + N : codeNum \quad (9\text{-}28)$$

8. The variable array lastPosVlcNumIndex[blockType] is updated as follows.

```
if ((N = = 8 ? codeNum : codeNum>>2) < lastPosVlcNumIndex[blockType])
    lastPosVlcNumIndex[blockType] -= 1                        (9-29)
else if ((N = = 8 ? codeNum : codeNum>>2) > lastPosVlcNumIndex[blockType])
    lastPosVlcNumIndex[blockType] += 1
```

– Otherwise, (N equal to 4), last_pos_level_one is derived in the following ordered steps:

1. The parsing process described in subclause 9.2.1 is invoked with 2 as input and the variable codeNum as output.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000683

2. The variable tableNum is derived as

$$tableNum = (PredMode==MODE\_INTRA \,||\, cIdx > 0) \,?\, 0 : slice\_type + 1 \quad (9\text{-}30)$$

3. The syntax element last_pos_level_one is derived as

$$last\_pos\_level\_one = lastPosTable[tableNum][codeNum] \quad (9\text{-}31)$$

4. The variable array lastPosTable[tableNum] is updated by invoking process 9.2.3 with lastPosTable[tableNum], codeNum and 0 as inputs.

- The variable trOne is derived as

$$trOne = !\, levelGreaterThanOneFlag \quad (9\text{-}32)$$

- The variable vlcNumLevel is set equal to 0

Table 9-9 – Specification of lastPosVlcNumTable[blockType][vlcNumIdx]

	vlcNumIdx																
blockType	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
0	10	10	10	10	2	2	2	7	9	9	9	9	9	4	4	4	4
1	10	10	10	10	10	2	9	9	9	9	9	9	9	4	4	4	4
2	2	2	2	2	2	7	7	7	7	7	7	7	7	7	4	4	13
3	2	2	2	2	7	7	7	7	7	7	7	7	7	7	7	7	13
4	2	2	2	2	7	7	7	7	7	7	7	7	7	7	7	7	13
5	10	10	10	4	4	4	4	12	12	12	12	12	12	12	12	12	12
6	10	10	10	10	4	4	12	12	12	12	12	12	12	12	12	12	12
7	10	10	10	10	4	4	12	12	12	12	12	12	12	12	12	12	12

9.2.4.2 Parsing process for level_minus2_and_sign

Input to this process are bits from slice data.

Output of this process is level_minus2_and_sign.

The syntax element level_minus2_and sign is derived by invoking the parsing process of subclause 9.2.1 with 0 as input and level_minus2_and sign as output.

9.2.4.3 Parsing process for run_level_one

Inputs to this process are bits from slice data, the scan position n of the previous non-zero transform coefficient in inverse scan order, and the variable TrOne.

Outputs of this process are the syntax element run_level_one and the variable TrOne.

The value of run_level_one is derived as follows.

- The variable N is set equal to (1 << log2TrafoSize) >> (cIdx > 0 ? 2 : 0).
- The variable blockType is derived as

$$blockType = (cIdx == 0 \,?\, (PredMode==MODE\_INTRA \,?\, 0 : slice\_type + 1) + (N > 8 \,?\, 5 : 2) : cIdx - 1) \quad (9\text{-}33)$$

- The variable tableIdx is derived as

$$tableIdx = (blockType == 2 \,||\, blockType == 5) \,?\, (N <= 8 \,?\, 0 : 1) : (N <= 8 \,?\, 2 : 3) \quad (9\text{-}34)$$

- The variable maxRun is set equal to n.
- The variable maxRunIdx is set equal to Min(28, maxRun).
- The variable vlcNum is derived from tableIdx and maxRunIdx as shown in Table 9-9.
- The parsing process described in subclause 9.2.1 is invoked with vlcNum as input and the variable codeNum as output.If blockType is equal to 2 or blockType is equal to 5, the value of run_level_one is derived as follows.
  - If N is equal to 4, the variable largeOnePos is derived from trOne and maxRunIdx as shown in Table 9-10.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000684

- Otherwise (N is greater than 4), the variable largeOnePos is derived from trOne and maxRunIdx as shown in Table 9-11.
- The variables levelGreaterThanOneFlag and runOfZeros are derived as follows.

```
if ( largeOnePos > 0 ) {
    if( codeNum < min(largeOnePos, maxRunIdx + 2 ) ) {
        levelGreaterThanOneFlag = 0
        runOfZeros = codeNum
    } else if( codeNum < (maxRunIdx << 1 ) + 4 – largeOnePos ) {
        if(( codeNum + largeOnePos ) & 1 ) {
            levelGreaterThanOneFlag = 0
            runOfZeros = (codeNum + largeOnePos - 1) >> 1
        } else {
            levelGreaterThanOneFlag = 1
            runOfZeros = (codeNum - largeOnePos)>>1                    (9-35)
        }
    } else {
        levelGreaterThanOneFlag = 1
        runOfZeros = codeNum - maxRunIdx – 2
    }
} else {
    if( codeNum & 1) {
        levelGreaterThanOneFlag = 0
        runOfZeros = (codeNum - 1)>>1
    } else {
        runOfZeros = codeNum >> 1
        levelGreaterThanOneFlag = ((codeNum >> 1) <= maxRunIdx)?1:0
    }
}
```

- The syntax element run_level_one is derived as

```
run_level_one = levelGreaterThanOneFlag ? runOfZeros + maxRun : runOfZeros    (9-36)
```

- Otherwise (blockType is not equal to 2 and blockType is not equal to 5), the value of run_level_one is derived as follows.
  - The variable levelGreaterThanOneFlag is derived as

```
levelGreaterThanOneFlag  = codeNum <= maxRun + 1 ? 0 : 1                   (9-37)
```

  - If maxRun is less than 28, the variable runOfZeros is derived from maxRun and codeNum as shown in Table 9-12.
  - Otherwise (maxRun is not less than 28), the variable runOfZeros is derived as

```
runOfZeros = codeNum <= maxRun + 1 ? codeNum : codeNum – maxRun – 2        (9-38)
```

  - The syntax element run_level_one is derived as

```
run_level_one = levelGreaterThanOneFlag ? runOfZeros + n : runOfZeros      (9-39)
```

- The variable trOne is derived as

```
trOne = (trOne = = 0 | | levelGreaterThanOneFlag) ? 0 : Max(4, trOne + 1)  (9-40)
```

Table 9-10 – Derivation of vlcNum from tableIdx and maxRunIdx

	maxRunIdx														
tableIdx	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
0	8	0	0	5	5	5	5	5	5	5	6	6	6	6	6
1	8	0	0	5	5	5	5	5	5	5	6	6	6	6	6
2	8	0	0	0	0	5	5	5	5	5	5	5	5	5	5
3	8	0	1	1	1	1	2	2	2	2	2	2	2	6	6

	maxRunIdx

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000685

tableIdx	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
0	6	6	6	6	6	6	6	6	6	6	6	6	6	3	6
1	6	6	6	6	6	6	6	6	6	6	6	6	6	3	6
2	5	5	5	1	1	1	1	1	1	1	1	1	1	2	5
3	6	6	6	6	3	3	3	3	3	3	3	3	3	3	6

Table 9-11 – Derivation of largeOnePos from tableIdx and maxRunIdx when N is equal to 4

	maxRunIdx														
trOne	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	0	0	1	1	1	1	1	1	1	1	1	1	0	0
1	2	3	4	5	6	5	6	7	7	7	7	7	6	4	2
2	2	2	2	3	3	3	3	4	4	4	3	3	3	2	NA
3	2	1	1	2	2	2	2	2	2	2	2	2	1	NA	NA
4	1	1	1	1	1	1	1	1	1	1	1	1	NA	NA	NA

Table 9-12 – Derivation of largeOnePos from tableIdx and maxRunIdx when N is greater than 4

	maxRunIdx														
trOne	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	1	2	2	2	2	2	4	5	5	6	6	6	6	6
1	2	4	4	6	6	8	8	10	11	13	15	13	14	15	16
2	2	3	4	5	4	5	6	6	7	8	8	9	9	10	10
3	2	1	2	2	3	3	4	4	5	5	5	6	6	7	7
4	2	1	2	2	2	2	3	3	3	3	4	4	4	4	4

	maxRunIdx													
trOne	15	16	17	18	19	20	21	22	23	24	25	26	27	28
0	6	7	8	7	8	8	9	10	10	10	12	10	9	8
1	18	18	21	20	21	22	23	25	25	26	27	28	29	27
2	11	11	12	13	14	15	16	16	17	18	19	20	19	19
3	7	8	9	9	10	10	11	11	12	12	13	13	13	13
4	5	5	5	5	5	6	6	6	6	6	6	7	7	8

Table 9-13 – Derivation of runOfZeros from maxRun and codeNum

	codeNum% (maxRun+1)														
maxRun	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	1	0	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
1	2	1	0	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
2	3	0	2	1	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
3	4	1	0	2	3	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
4	5	0	3	2	4	1	NA	NA	NA	NA	NA	NA	NA	NA	NA
5	6	0	1	4	5	3	2	NA	NA	NA	NA	NA	NA	NA	NA
6	7	1	0	2	4	3	6	5	NA	NA	NA	NA	NA	NA	NA
7	8	0	3	4	2	1	5	7	6	NA	NA	NA	NA	NA	NA

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000686

8	9	0	5	1	6	4	8	3	2	7	NA	NA	NA	NA	NA
9	10	0	1	6	7	2	9	5	4	3	8	NA	NA	NA	NA
10	11	1	0	2	3	6	7	4	5	8	10	9	NA	NA	NA
11	12	0	3	2	1	4	6	5	7	11	8	9	10	NA	NA
12	13	0	1	5	6	4	2	3	7	8	12	10	11	9	NA
13	14	0	1	7	2	8	6	5	3	4	13	12	9	11	10
14	15	0	1	2	8	3	9	7	6	4	5	14	10	13	12
15	16	0	1	2	3	4	8	7	5	9	6	10	15	11	13
16	17	0	3	1	2	4	5	8	9	7	6	10	11	12	16
17	18	0	1	5	2	6	4	3	7	8	9	10	17	13	14
18	19	0	1	7	8	2	6	3	9	4	5	10	18	15	16
19	20	0	9	1	10	2	8	3	7	4	16	17	6	19	5
20	21	0	1	10	2	11	3	9	5	4	17	20	18	8	7
21	22	1	0	2	3	4	10	11	5	6	9	7	12	8	13
22	23	0	3	4	2	1	5	7	6	10	11	8	9	12	13
23	24	0	5	6	1	4	3	7	2	8	10	11	9	12	15
24	25	0	7	1	8	6	5	2	9	10	4	11	3	12	17
25	26	0	1	9	10	2	8	11	7	3	6	5	12	4	19
26	27	0	1	11	2	10	12	3	9	7	8	4	6	20	13
27	28	0	1	2	12	3	4	11	13	9	5	8	10	7	6

	codeNum% (maxRun+1)													
maxRun	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**
0	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
1	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
2	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
3	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
4	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
5	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
6	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
7	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
8	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
9	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
10	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
11	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
12	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
13	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
14	11	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
15	14	12	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
16	13	14	15	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
17	12	15	11	16	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
18	14	11	13	17	12	NA	NA	NA	NA	NA	NA	NA	NA	NA
19	11	15	12	14	13	18	NA	NA	NA	NA	NA	NA	NA	NA
20	6	12	16	13	15	19	14	NA	NA	NA	NA	NA	NA	NA
21	18	21	17	19	14	20	16	15	NA	NA	NA	NA	NA	NA
22	14	22	18	15	19	20	17	16	21	NA	NA	NA	NA	NA
23	14	13	23	16	19	17	18	21	20	22	NA	NA	NA	NA

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000687

24	16	13	24	18	19	15	20	23	14	21	22	NA	NA	NA
25	13	18	20	14	17	25	21	16	23	24	15	22	NA	NA
26	5	14	26	21	19	18	25	15	16	17	24	22	23	NA
27	21	14	27	15	22	19	16	20	17	26	18	25	23	24

9.2.4.4 Parsing process for level

Inputs to this process are bits from slice data and the variable vlcNumLevel.

Outputs of this process are the syntax element level and the variable vlcNumLevel.

– The syntax element level is determined by invoking the parsing process described in subclause 9.2.1 with vlcNumLevel as input and level as output.

– The variable vlcNumLevel derived in the following ordered steps updated as follows.

1. The variable vlcIdx is set equal to vlcIdx = Min(3,vlcNumLevel)
2. The variable levelThreshold is derived from vlcIdx as shown in Table 9-13.
3. The variable vlcNumLevel is derived as

$$vlcNumLevel = vlcNumLevel + (level > levelThreshold \;\&\&\; vlcNumLevel < 4 \;?\; 1 : 0) \quad (9\text{-}41)$$

Table 9-14 – Derivation of levelThreshold from vlcIdx

vlcIdx	levelThreshold
0	4
1	6
2	14
3	28

9.2.4.5 Parsing process for cu_split_pred_part_mode

This process is invoked when entropy_coding_mode_flag is equal to 0 for parsing syntax element **cu_split_pred_part_mode** in subclause 7.3.5.

Inputs to this process are bits from slice data, splitPredPartModeTable.

Output of this process is the syntax element cu_split_pred_part_mode.

The value of cu_split_pred_part_mode is derived as follows.

– If log2CUSize is greater than Log2MinCUSize, the parsing process described in subclause 9.2.1 is invoked with vlcNum of 14 and cMax of 10 as inputs and the variable codeNum as output.

– Otherwise, the parsing process described in subclause 9.2.1 is invoked with vlcNum of 14 and cMax of 5 as inputs and the variable codeNum as output.

– The syntax element cu_split_pred_part_mode is derived as

$$cu\_split\_pred\_part\_mode = splitPredPartModeTable[Log2MaxCUSize - log2CUSize][codeNum] \quad (9\text{-}41)$$

– If log2CUSize is equal to Log2MinCUSize and cu_split_part_mode has a value of 5, PredMode and PartMode are determined as follows

1. Read one bit and assign its value to a variable b
2. If b is equal to 1, set PredMode to MODE_INTRA and PartMode to PART_2Nx2N
3. Otherwise if inter_4x4_enabled_flag is equal to 0 and log2CUSize is equal to 3, set PredMode to MODE_INTRA and PartMode to PART_NxN
4. Otherwise, read the next one bit and assign its value to the variable b.
    1. If b is equal to 1, set PredMode to MODE_INTRA and PartMode to PART_NxN
    2. Otherwise, set PredMode to MODE_INTER and PartMode to PART_NxN

– The variable array splitPredPartModeTable[Log2MaxCUSize – log2CUSize] is updated by invoking process 9.2.3 with splitPredPartModeTable[Log2MaxCUSize – log2CUSize], codeNum and a value of 4 as inputs.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000688

9.2.4.6 Parsing process for rem_intra_luma_pred_mode

This process is invoked when entropy_coding_mode_flag is equal to 0 for parsing syntax element rem_intra_luma_pred_mode in subclause 7.3.7.

Inputs to this process are bits from slice data, a variable log2CUSize specifying the size of the current coding unit and a variable array intraModeTable.

Outputs of this process are the syntax element rem_intra_luma_pred_mode and value-updated intraModeTable.

The variable log2PUSize is derived as follows.

– If PartMode is equal to PART_NxN, log2PUSize is set equal to log2CUSize – 1,

– Otherwise, log2PUSize is set equal to log2CUSize.

The value of rem_intra_luma_pred_mode is derived as follows.

– Based on log2PUSize, the value of a variable intraPredModeNum indicating the number of intra prediction modes for the given size of prediction unit is obtained according to Table 8-2.

– If intraPredModeNum is equal to 4, read one bit and assign its value to rem_intra_luma_pred_mode. No further step is carried out. Othewise,

– The values of variable k and vlcNum are obtained as follows

  – If intraPredModeNum is equal to 18, set vlcNum and k equal to 20 and 0, respectively.

  – If intraPredModeNum is equal to 35, set vlcNum and k equal to 21 and 1, respectively.

  – The parsing process described in subclause 9.2.1 is invoked with vlcNum as input and the variable codeNum as output.

  – The value of rem_intra_luma_pred_mode is set equal to intraModeTable[k][codeNum].

  – The value of a variable counterNum is set to 0. The variable array intraModeTable[k] is updated by invoking process in subclause 9.2.3 with intraModeTable[k], codeNum and counterNum as inputs.

9.2.4.7 Parsing process for cbp_and_split_transform

This process is invoked when entropy_coding_mode_flag is equal to 0 for parsing syntax element cbp_and_split_transform in subclause 7.3.8.

Inputs to this process are bits from slice data, a variable cbpVlcNumIdx, trafoDepth and a variable array cbpSplitTransTable.

Outputs of this process are the syntax element cbp_and_split_transform and value-updated cbpVlcNumIndex and cbpSplitTransTable.

The value of cbp_and_split_transform is derived as follows.

– Obtain a 4-bit value flagPattern accoding to subclause 7.4.8.

– Derive the value of a variable k according to Table 9-16.

– Set the value of a variable n as

$$n = (PredMode == MODE\_INTRA\ ?\ 0 :\ 1) \tag{9-43}$$

– Obtain a value of vlcNum as follows

  – If flagPattern is equal to 8,

$$vlcNum = (n == 0\ ?\ cbpVlcNumTable[cbpVlcNumIdx]:11) \tag{9-44}$$

  – Otherwise, if flagPattern is equal to 11, 13 or 15 and n is equal to 0, set vlcNum to 15;

  – Otherwise, set vlcNum to 14.

– If vlcNum is euqal to 14, obtain the value of a variable cMax according to Table 9-15.

– The parsing process described in subclause 9.2.1 is invoked with vlcNum and cMax as inputs and the variable codeNum as output.

– Set the value of cbp_and_split_transform equal to cbpSplitTransTable[k][n][codeNum]

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000689

– If flagPattern is not equal to 10 or 12, a variable counterNum is obtained according to Table 9-16 and the variable array cbpSplitTransTable[k][n] is updated by invoking process 9.2.3 with cbpSplitTransTable[k][n], codeNum and counterNum as inputs.

– If flagPattern is euqal to 8 and n is equal to 0, the value of cbpVlcNumIdx is updated as follows

```
if(codeNum < cbpVlcNumIdx)
    cbpVlcNumIdx -= 1                                   (9-45)
else if (codeNum > cbpVlcNumIdx)
    cbpVlcNumIdx += 1
```

– If flagPattern is euqal to 15, the exact values of Cb and Cr in Table 7-13 are determined as follows.

  – The parsing process described in subclause 9.2.1 is invoked with vlcNum of 14 and cMax of 2 as inputs and the variable codeNum as output.

  – Set the value of variable b as

$$b = n? (codeNum + 1) : (3 - codeNum) \quad (9\text{-}46)$$

  – The values of Cb and Cr are set equal to

$$Cb = (b >> 1) \quad (9\text{-}47)$$

$$Cr = (b \,\&\, 0x01) \quad (9\text{-}48)$$

Table 9-15 – Specification of cbpVlcNumTable[cbpVlcNumIdx]

cbpVlcNumIdx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
vlcNum	1	2	2	2	11	11	11	11	11	11	11	11	11	11	11

Table 9-16 – Derivation of cMax from flagPattern and n

	flagPattern			
n	9	10, 12	11,13,15	14
0	3	3	-	7
1	2	3	6	7

Table 9-17 – Specification of cbpSplitTransTable and counterNum associated

flagPattern	8	9	10, 12	11, 13, 15	14 (trafoDepth=0)	14 (trafoDepth>0)
k	0	1	2	3	4	5
counterNum	2	3	0	4	4	4

9.3 CABAC parsing process for slice data

[Ed. (TW) needs to be extended]

This process is invoked when parsing syntax elements with descriptor ae(v) in subclauses 7.3.4 and 7.3.5 when entropy_coding_mode_flag is equal to 1.

Inputs to this process are a request for a value of a syntax element and values of prior parsed syntax elements.

Output of this process is the value of the syntax element.

When starting the parsing of the slice data of a slice in subclause 7.3.4, the initialisation process of the CABAC parsing process is invoked as specified in subclause 9.3.1.

The parsing of syntax elements proceeds as follows:

For each requested value of a syntax element a binarization is derived as described in subclause 9.3.2.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000690

The binarization for the syntax element and the sequence of parsed bins determines the decoding process flow as described in subclause 9.3.3.

For each bin of the binarization of the syntax element, which is indexed by the variable binIdx, a context index ctxIdx is derived as specified in subclause 9.3.3.1.

For each ctxIdx the arithmetic decoding process is invoked as specified in subclause 9.3.3.2.

The resulting sequence ($b_0..b_{binIdx}$) of parsed bins is compared to the set of bin strings given by the binarization process after decoding of each bin. When the sequence matches a bin string in the given set, the corresponding value is assigned to the syntax element.

In case the request for a value of a syntax element is processed for the syntax element mb_type and the decoded value of mb_type is equal to I_PCM, the decoding engine is initialised after the decoding of any pcm_alignment_zero_bit and all pcm_sample_luma and pcm_sample_chroma data as specified in subclause 9.3.1.2.

9.3.1 Initialisation process

Outputs of this process are initialised CABAC internal variables.

The processes in subclauses 9.3.1.1 and 9.3.1.2 are invoked when starting the parsing of the slice data of a slice in subclause 7.3.4.

9.3.1.1 Initialisation process for context variables

Formatted: Heading 5

Outputs of this process are the initialised CABAC context variables indexed by ctxIdxTable and ctxIdx.

Table 9-21 toTable 9-45 contain the values of the variables n and m used in the initialisation of context variables that are assigned to all syntax elements in subclauses 7.3.4 to 7.3.10 except for the end-of-slice flag and those related to CAVLC.

For each context variable, the two variables pStateIdx and valMPS are initialised.

NOTE 1 – The variable pStateIdx corresponds to a probability state index and the variable valMPS corresponds to the value of the most probable symbol as further described in subclause 9.3.3.2.

The two values assigned to pStateIdx and valMPS for the initialisation are derived from $SliceQP_Y$, which is derived in Equation 7-9. Given the two table entries (m, n), the initialisation is specified by the following pseudo-code process:

```
preCtxState = Clip3( 1, 126, ( ( m * Clip3( 0, 51, SliceQPY ) ) >> 4 ) + n )
if( preCtxState <= 63 ) {
    pStateIdx = 63 − preCtxState
    valMPS = 0                                                    (9-4)
} else {
    pStateIdx = preCtxState − 64
    valMPS = 1
}
```

In Table 9-20, the ctxIdxTable and the ctxIdx for which initialisation is needed for each of the slice types are listed. The referenced context index tables ctxIdxTable include the values of m and n needed for the initialisation.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000691

Table 9-20 – Association of ctxIdx and syntax elements for each slice type in the initialisation process

	Syntax element	ctxIdxTable	Slice Type		
			I	P	B
slice_header()	alf_cu_flag	Table 9-21	0	1	2
coding_tree()	split_coding_unit_flag	Table 9-22	0..2	3..5	6..8
coding_unit()	skip_flag	Table 9-23		0..2	3..5
	cu_qp_delta	Table 9-24	0..3	4..7	8..11
	pred_type	Table 9-25	0	1..4	5..9
prediction_unit()	prev_intra_luma_pred_flag	Table 9-26	0	1	2
	rem_intra_luma_pred_mode	Table 9-27	0	1	2
	intra_chroma_pred_mode	Table 9-28	0..1	2..3	4..5
	merge_flag	Table 9-29		0	1
	merge_idx	Table 9-30		0..3	4..7
	inter_pred_flag	Table 9-31			0..3
	ref_idx_lc, ref_idx_l0, ref_idx_l1	Table 9-32		0..2	3..5
	abs_mvd_greater0_flag	Table 9-33		0	1
	abs_mvd_greater1_flag	Table 9-33		2	3
	mvp_idx_lc, mvp_idx_l0, mvp_idx_l1	Table 9-34		0..1	2..3
transform_tree()	no_residual_data_flag	Table 9-35		0	1
	split_transform_flag	Table 9-36	0..3	4..7	8..11
	cbf_luma	Table 9-37	0..1	2..3	4..5
	cbf_cb	Table 9-38	0..3	4..7	8..11
	cbf_cr	Table 9-39	0..3	4..7	8..11
residual_coding()	last_significant_coeff_x, last_significant_coeff_y	Table 9-40	0..30	31..61	62..92
	significant_coeff_flag		Table 9-41	Table 9-42	Table 9-43
	coeff_abs_level_greater1_flag	Table 9-44	0..79	80..159	160..239
	coeff_abs_level_greater2_flag	Table 9-45	0..79	80..159	160..239

NOTE 2 – ctxIdxTable equal to 0 and ctxIdx equal to 0 are associated with the end_of_slice_flag. The decoding process specified in subclause 9.3.3.2.4 applies to ctxIdxTable equal to 0 and ctxIdx equal to 0. This decoding process, however, may also be implemented by using the decoding process specified in subclause 9.3.3.2.1. In this case, the initial values associated with ctxIdxTable equal to 0 and ctxIdx equal to 0 are specified to be pStateIdx = 63 and valMPS = 0, where pStateIdx = 63 represents a non-adapting probability state.

Table 9-21 – Values of variables m and n for alf_cu_flag ctxIdx

Initialisation variables	alf_cu_flag ctxIdx		
	0	1	2
m	0	0	0
n	64	64	64

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000692

Table 9-22 – Values of variables m and n for split_coding_unit_flag ctxIdx

Initialisation variables	split_coding_unit_flag ctxIdx								
	0	1	2	3	4	5	6	7	8
m	-7	-10	-10	-14	-6	-6	−14	−7	−10
n	68	87	105	71	73	91	71	74	92

Table 9-23 – Values of variables m and n for skip_flag ctxIdx

Initialisation variables	skip_flag ctxIdx					
	0	1	2	3	4	5
m	0	0	0	0	0	0
n	64	64	64	64	64	64

Table 9-24 – Values of variables m and n for cu_qp_delta ctxIdx

Initialisation variables	cu_qp_delta ctxIdx								
	0	1	2	3	4	5	6	7	8
m	0	0	0	0	0	0	0	0	0
n	64	64	64	64	64	64	64	64	64

Table 9-25 – Values of variables m and n for pred_type

Initialisation variables	pred_type ctxIdx									
	0	1	2	3	4	5	6	7	8	9
m	0	-25	-1	-3	6	6	-1	13	-11	-11
n	73	89	64	63	78	50	56	53	76	70

Table 9-26 – Values of variable m and n for prev_intra_luma_pred_flag ctxIdx

Initialisation variables	prev_intra_luma_pred_flag ctxIdx		
	0	1	2
m	2	0	0
n	54	50	51

Table 9-27 – Values of variable m and n for rem_intra_luma_pred_mode ctxIdx

Initialisation variables	rem_intra_luma_pred_mode ctxIdx		
	0	1	2
m	-3	-2	1
n	65	61	55

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000693

Table 9-28 – Values of variable m and n for intra_chroma_pred_mode ctxIdx

Initialisation variables	intra_chroma_pred_mode ctxIdx					
	0	1	2	3	4	5
m	0	0	0	0	0	0
n	64	64	64	64	64	64

Table 9-29 – Values of variable m and n for merge_flag ctxIdx

Initialisation variables	merge_flag ctxIdx
	0
m	0
n	64

Table 9-30 – Values of variable m and n for merge_idx ctxIdx

Initialisation variables	merge_idx ctxIdx							
	0	1	2	3	4	5	6	7
m	0	0	0	0	1	6	-7	−4
n	64	64	64	64	65	42	75	72

Table 9-31 – Values of variable m and n for inter_pred_flag ctxIdx

Initialisation variables	inter_pred_flag ctxIdx			
	0	1	2	3
m	-2	-5	-9	1
n	58	70	85	61

Table 9-32 – Values of variable m and n for ref_idx_lc, ref_idx_l0, ref_idx_l1 ctxIdx

Initialisation variables	ref_idx_lc, ref_idx_l0, ref_idx_l1 ctxIdx					
	0	1	2	3	4	5
m	-6	-17	1	-9	-12	-18
n	59	96	59	55	86	55

Table 9-33 – Values of variable m and n for abs_mvd_greater0_flag and abs_mvd_greater1_flag ctxIdx

Initialisation variables	abs_mvd_greater0_flag, abs_mvd_greater1_flag ctxIdx			
	0	1	2	3
m	5	-2	-7	14
n	61	79	89	39

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000694

Table 9-34 – Values of variable m and n for mvp_idx_lc, mvp_idx_l0, mvp_idx_l1 ctxIdx

Initialisation variables	mvp_idx_lc, mvp_idx_l0, mvp_idx_l1 ctxIdx			
	0	1	2	3
m	0	0	0	0
n	64	64	64	64

Table 9-35 – Values of variable m and n for no_residual_data_flag ctxIdx

Initialisation variables	no_residual_data_flag ctxIdx	
	0	1
m	−22	−36
n	85	103

Table 9-36 – Values of variable m and n for split_transform_flag ctxIdx

Initialisation variables	split_transform_flag ctxIdx											
	0	1	2	3	4	5	6	7	8	9	10	11
m	0	12	22	-2	0	-28	-30	-34	-11	-31	-42	-47
n	64	12	4	49	13	89	99	106	38	88	118	130

Table 9-37 – Values of variable m and n for cbf_luma ctxIdx

Initialisation variables	cbf_luma ctxIdx					
	0	1	2	3	4	5
m	-22	-5	-18	-41	-11	-32
n	116	75	98	120	80	83

Table 9-38 – Values of variable m and n for cbf_cb ctxIdx

Initialisation variables	cbf_cb ctxIdx											
	0	1	2	3	4	5	6	7	8	9	10	11
m	-35	-12	-9	-10	-46	-42	-11	-19	-22	-48	-7	-37
n	116	61	73	75	114	119	74	90	52	123	68	121

Table 9-39 – Values of variable m and n for cbf_cr ctxIdx

Initialisation variables	cbf_cr ctxIdx											
	0	1	2	3	4	5	6	7	8	9	10	11
m	-29	-12	-5	-6	-43	-41	-17	-25	-19	-48	-21	-9
n	104	59	65	67	107	118	86	101	45	123	94	73

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000695

Table 9-40 – Values of variable m and n for last_significant_coefficient_x and last_significant_coefficient_y ctxIdx

Initialisation variables	last_significant_coefficient_x and last_significant_coeff_y ctxIdx															
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	
m	14	10	12	22	10	9	17	15	16	17	18	16	23	20	24	
n	25	38	41	25	37	40	19	21	18	16	13	18	-2	-2	-9	
	15	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	
m	24	18	9	22	37	35	53	31	13	21	35	22	18	9	14	
n	-7	10	24	-7	9	14	-5	21	39	28	2	17	23	40	32	
	30	**31**	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**	
m	7	12	12	13	21	24	23	13	22	29	29	27	20	25	30	
n	44	35	40	47	16	10	15	34	10	10	-8	-5	7	13	-8	
	45	**46**	**47**	**48**	**49**	**50**	**51**	**52**	**53**	**54**	**55**	**56**	**57**	**58**	**59**	
m	35	37	42	42	43	21	22	35	11	22	27	-3	14	23	27	
n	-20	-29	-41	-45	-46	36	37	27	53	27	18	83	35	18	3	
	60	**61**	**62**	**63**	**64**	**65**	**66**	**67**	**68**	**69**	**70**	**71**	**72**	**73**	**74**	
m	32	17	12	12	13	21	24	23	13	22	29	29	27	20	25	
n	-11	11	35	40	47	16	10	15	34	10	10	-8	-5	7	13	
	75	**76**	**77**	**78**	**79**	**80**	**81**	**82**	**83**	**84**	**85**	**86**	**87**	**88**	**89**	
m	30	35	37	42	42	43	21	22	35	11	22	27	-3	14	23	
n	-8	-20	-29	-41	-45	-46	36	37	27	53	27	18	83	35	18	
	90	**91**	**92**	**93**	**94**	**95**	**96**	**97**	**98**	**99**	**100**	**101**	**102**	**103**	**104**	
m	27	32	17													
n	3	-11	11													

Table 9-41 – Values of variable m and n for significant_coeff_flag ctxIdx (A)

Initialisation variables	significant_coeff_flag ctxIdx															
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
m	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
n	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64
	16	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
m	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
n	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64
	32	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**	**45**	**46**	**47**
m	-22	-14	-17	-26	5	-2	-32	-19	-7	-24	0	-1	2	0	-15	0
n	128	96	99	113	31	65	125	72	69	111	25	49	54	64	97	64
	48	**49**	**50**	**51**	**52**	**53**	**54**	**55**	**56**	**57**	**58**	**59**	**60**	**61**	**62**	**63**
m	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
n	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64
	64	**65**	**66**	**67**	**68**	**69**	**70**	**71**	**72**	**73**	**74**	**75**	**76**	**77**	**78**	**79**
m	-4	0	-4	-5	17	2	-8	1	2	-11	6	5	2	0	-8	0
n	88	68	75	74	7	56	80	36	52	85	12	41	54	65	82	64
	80	**81**	**82**	**83**	**84**	**85**	**86**	**87**								
m	-8	-7	-7	-9	-18	-7	-11	-29								
n	95	77	73	74	77	72	81	92								

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000696

Table 9-42 – Values of variable m and n for significant_coeff_flag ctxIdx (B)

Initialisation variables	significant_coeff_flag ctxIdx															
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
m	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
n	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64
	16	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
m	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
n	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64
	32	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**	**45**	**46**	**47**
m	-22	-14	-17	-26	5	-2	-32	-19	-7	-24	0	-1	2	0	-15	0
n	128	96	99	113	31	65	125	72	69	111	25	49	54	64	97	64
	48	**49**	**50**	**51**	**52**	**53**	**54**	**55**	**56**	**57**	**58**	**59**	**60**	**61**	**62**	**63**
m	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
n	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64
	64	**65**	**66**	**67**	**68**	**69**	**70**	**71**	**72**	**73**	**74**	**75**	**76**	**77**	**78**	**79**
m	-4	0	-4	-5	17	2	-8	1	2	-11	6	5	2	0	-8	0
n	88	68	75	74	7	56	80	36	52	85	12	41	54	65	82	64
	80	**81**	**82**	**83**	**84**	**85**	**86**	**87**								
m	-8	-7	-7	-9	-18	-7	-11	-29								
n	95	77	73	74	77	72	81	92								

Table 9-43 – Values of variable m and n for significant_coeff_flag ctxIdx (C)

Initialisation variables	significant_coeff_flag ctxIdx															
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
m	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
n	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64
	16	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
m	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
n	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64
	32	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**	**45**	**46**	**47**
m	-34	-20	-23	38	3	-4	-17	-9	-4	-16	-1	0	-1	-12	-14	0
n	146	108	111	41	34	69	103	50	66	97	24	50	61	90	97	64
	48	**49**	**50**	**51**	**52**	**53**	**54**	**55**	**56**	**57**	**58**	**59**	**60**	**61**	**62**	**63**
m	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
n	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64	64
	64	**65**	**66**	**67**	**68**	**69**	**70**	**71**	**72**	**73**	**74**	**75**	**76**	**77**	**78**	**79**
m	-12	-4	-8	-7	15	2	-10	4	-10	-9	-9	6	3	-1	-14	0
n	99	75	81	80	13	57	85	27	75	81	46	41	56	69	96	64
	80	**81**	**82**	**83**	**84**	**85**	**86**	**87**								
m	-7	-22	-12	-35	-5	-16	-9	-26								
n	85	102	79	121	55	88	78	73								

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000697

Table 9-44 – Values of variable m and n for coeff_abs_level_greater1_flag ctxIdx

Initialisation variables	coeff_abs_level_greater1_flag ctxIdx														
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
m	-11	-20	-16	-13	-10	-5	-8	-8	-3	-9	0	-5	-12	-9	-1
n	87	64	68	71	73	67	26	37	36	56	63	39	56	57	52
	15	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**
m	-4	-19	-28	-23	-3	-2	-27	-22	-14	-1	-10	-22	-13	-6	-5
n	72	73	88	85	59	72	97	89	77	58	86	74	63	57	63
	30	**31**	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**
m	-4	16	9	5	-7	-6	-13	-12	-18	-14	-5	-8	-22	-35	-3
n	70	-2	23	41	67	63	-5	42	53	59	65	36	67	89	47
	45	**46**	**47**	**48**	**49**	**50**	**51**	**52**	**53**	**54**	**55**	**56**	**57**	**58**	**59**
m	-12	-20	-51	-44	-23	-1	-58	-71	-67	-91	-1	20	13	2	5
n	77	66	113	109	84	64	127	143	134	187	64	-3	21	46	48
	60	**61**	**62**	**63**	**64**	**65**	**66**	**67**	**68**	**69**	**70**	**71**	**72**	**73**	**74**
m	-4	29	1	-6	-9	-6	23	3	-5	5	-12	-2	-15	-11	-3
n	71	-5	45	58	67	66	-7	26	42	31	79	45	67	62	54
	75	**76**	**77**	**78**	**79**	**80**	**81**	**82**	**83**	**84**	**85**	**86**	**87**	**88**	**89**
m	-4	-13	-3	-26	-18	-9	-30	-24	-43	-6	4	34	21	14	5
n	70	68	100	91	84	83	103	90	122	66	59	-11	10	25	44
	90	**91**	**92**	**93**	**94**	**95**	**96**	**97**	**98**	**99**	**100**	**101**	**102**	**103**	**104**
m	-4	40	19	-11	-6	-11	12	17	-6	11	-11	1	-26	-28	-2
n	69	-33	8	65	59	68	-2	-10	34	14	70	27	71	79	47
	105	**106**	**107**	**108**	**109**	**110**	**111**	**112**	**113**	**114**	**115**	**116**	**117**	**118**	**119**
m	6	-23	-47	-55	-21	1	-38	-34	-45	10	9	41	32	14	17
n	47	67	107	117	83	59	82	77	95	25	46	-31	-17	19	18
	120	**129**	**130**	**131**	**132**	**133**	**134**	**135**	**136**	**137**	**138**	**139**	**140**	**141**	**142**
m	-2	18	10	-2	-7	-6	19	-5	-7	-4	-23	-3	2	-32	-16
n	65	22	33	55	64	67	11	50	53	54	99	51	41	102	79
	135	**136**	**137**	**138**	**139**	**140**	**141**	**142**	**143**	**144**	**145**	**146**	**147**	**148**	**149**
m	-8	-21	-26	-33	-4	-31	-34	-25	-43	-6	3	23	12	11	8
n	77	84	91	104	61	122	110	96	124	70	60	12	30	33	40
	150	**151**	**152**	**153**	**154**	**155**	**156**	**157**	**158**	**159**	**160**	**161**	**162**	**163**	**164**
m	-2	40	0	-5	7	-39	31	1	-35	-32	-15	16	-43	-75	-8
n	66	-20	46	54	37	116	-27	22	82	85	72	0	102	152	55
	165	**166**	**167**	**168**	**169**	**170**	**171**	**172**	**173**	**174**	**175**	**176**	**177**	**178**	**179**
m	-9	-12	-84	-93	25	-104	-40	-51	110	3	17	55	28	-5	29
n	68	54	171	186	12	222	92	93	-169	52	33	-45	1	55	-4

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000698

Table 9-45 – Values of variable m and n for coeff_abs_level_greater2_flag ctxIdx

Initialisation variables	coeff_abs_level_greater2_flag ctxIdx														
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
m	-12	-10	-11	-14	-35	-10	-1	-17	-5	-22	-13	-2	-13	-21	-3
n	72	79	87	94	136	58	54	86	70	105	70	59	81	96	73
	15	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**
m	-24	-19	-1	1	-17	-7	-20	-11	-27	-12	-9	-8	-4	-3	-9
n	90	88	63	60	97	69	95	84	110	96	72	76	75	76	94
	30	**31**	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**
m	-11	-4	-9	-24	-24	-18	-10	-19	-28	6	-12	-17	-8	-44	-17
n	65	66	82	107	111	58	61	82	97	47	50	73	66	115	86
	45	**46**	**47**	**48**	**49**	**50**	**51**	**52**	**53**	**54**	**55**	**56**	**57**	**58**	**59**
m	-26	-4	-43	-58	-51	-51	-85	-47	-95	-65	-2	7	-2	3	4
n	74	48	115	141	137	117	176	120	202	159	53	43	67	62	67
	60	**61**	**62**	**63**	**64**	**65**	**66**	**67**	**68**	**69**	**70**	**71**	**72**	**73**	**74**
m	-2	-13	-11	-13	-24	-4	-17	-16	-19	-28	-12	-11	-18	-6	-19
n	49	81	82	88	112	44	77	82	89	110	64	70	88	67	97
	75	**76**	**77**	**78**	**79**	**80**	**81**	**82**	**83**	**84**	**85**	**86**	**87**	**88**	**89**
m	-17	-27	-18	-13	-17	-11	-15	-10	-10	-12	-8	-7	-6	-8	-9
n	76	100	88	84	94	73	83	77	80	91	63	71	73	80	90
	90	**91**	**92**	**93**	**94**	**95**	**96**	**97**	**98**	**99**	**100**	**101**	**102**	**103**	**104**
m	14	-7	-4	-8	-50	2	11	-4	-41	-60	-12	-6	1	-5	-12
n	20	68	66	76	148	22	15	49	117	149	49	55	44	58	73
	105	**106**	**107**	**108**	**109**	**110**	**111**	**112**	**113**	**114**	**115**	**116**	**117**	**118**	**119**
m	13	5	-29	0	-23	-12	6	35	42	22	24	11	2	-5	4
n	9	26	101	46	92	57	30	-4	-24	24	0	34	61	75	62
	120	**129**	**130**	**131**	**132**	**133**	**134**	**135**	**136**	**137**	**138**	**139**	**140**	**141**	**142**
m	0	-6	-7	-31	-25	-20	-14	-20	-21	-6	-19	-34	-27	-7	-7
n	43	65	71	117	109	76	73	88	92	71	73	108	101	69	75
	135	**136**	**137**	**138**	**139**	**140**	**141**	**142**	**143**	**144**	**145**	**146**	**147**	**148**	**149**
m	-18	-7	-20	-9	-7	-26	-13	-6	-12	-2	-3	-6	-7	-6	-11
n	77	64	91	75	78	98	81	69	83	70	50	66	73	75	91
	150	**151**	**152**	**153**	**154**	**155**	**156**	**157**	**158**	**159**	**160**	**161**	**162**	**163**	**164**
m	19	2	4	-4	-32	13	-29	-64	-90	26	-24	8	-10	-16	-27
n	10	49	52	69	114	-1	85	143	186	4	68	30	61	78	96
	165	**166**	**167**	**168**	**169**	**170**	**171**	**172**	**173**	**174**	**175**	**176**	**177**	**178**	**179**
m	31	-12	-15	-68	31	-8	-32	-36	-53	36	26	19	-22	5	8
n	-16	54	70	158	12	44	63	81	106	12	-5	17	101	54	53

9.3.1.2 Initialisation process for the arithmetic decoding engine

This process is invoked before decoding the first treeblock of a slice.

Outputs of this process are the initialised decoding engine registers codIRange and codIOffset both in 16 bit register precision.

The status of the arithmetic decoding engine is represented by the variables codIRange and codIOffset. In the initialisation procedure of the arithmetic decoding process, codIRange is set equal to 510 and codIOffset is set equal to the value returned from read_bits(9) interpreted as a 9 bit binary representation of an unsigned integer with most significant bit written first.

The bitstream shall not contain data that result in a value of codIOffset being equal to 510 or 511.

NOTE – The description of the arithmetic decoding engine in this Recommendation | International Standard utilizes 16 bit register precision. However, a minimum register precision of 9 bits is required for storing the values of the variables codIRange and codIOffset after invocation of the arithmetic decoding process (DecodeBin) as specified in subclause 9.3.3.2. The arithmetic

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000699

decoding process for a binary decision (DecodeDecision) as specified in subclause 9.3.3.2.1 and the decoding process for a binary decision before termination (DecodeTerminate) as specified in subclause 9.3.3.2.4 require a minimum register precision of 9 bits for the variables codIRange and codIOffset. The bypass decoding process for binary decisions (DecodeBypass) as specified in subclause 9.3.3.2.3 requires a minimum register precision of 10 bits for the variable codIOffset and a minimum register precision of 9 bits for the variable codIRange.

9.3.2 Binarization process

Input to this process is a request for a syntax element.

Output of this process is the binarization of the syntax element, maxBinIdxCtx, ctxIdxOffset, and bypassFlag.

Table 9-46 specifies the type of binarization process, maxBinIdxCtx, ctxIdxTable, and ctxIdxOffset associated with each syntax element.

The specification of the unary (U) binarization process, the truncated unary (TU) binarization process, the truncated Rice (TR), the k-th order Exp-Golomb (EGk) binarization process, and the fixed-length (FL) binarization process are given in subclauses 0 to 9.3.2.5, respectively. Other binarizations are specified in subclauses 9.3.2.6 to 9.3.2.10.

The binarizations for the syntax element coeff_abs_level_minus3 as specified in subclause 9.3.2.9 consist of bin strings given by a concatenation of prefix and suffix bit strings. For these binarization processes, the prefix and the suffix bit string are separately indexed using the binIdx variable as specified further in subclause 9.3.3. The two sets of prefix bit strings and suffix bit strings are referred to as the binarization prefix part and the binarization suffix part, respectively.

Associated with each binarization or binarization part of a syntax element is a specific value of the context index table (ctxIdxTable) variable and the related context index offset (ctxIdxOffset) variable and a specific value of the maxBinIdxCtx variable as given in Table 9-46. When two values for each of these variables are specified for one syntax element in Table 9-46, the value in the upper row is related to the prefix part while the value in the lower row is related to the suffix part of the binarization of the corresponding syntax element.

The use of the DecodeBypass process and the variable bypassFlag is derived as follows.

– If no value is assigned to ctxIdxOffset for the corresponding binarization or binarization part in Table 9-46 labelled as "na", all bins of the bit strings of the corresponding binarization or of the binarization prefix/suffix part are decoded by invoking the DecodeBypass process as specified in subclause 9.3.3.2.3. In such a case, bypassFlag is set equal to 1, where bypassFlag is used to indicate that for parsing the value of the bin from the bitstream the DecodeBypass process is applied.

– Otherwise, for each possible value of binIdx up to the specified value of maxBinIdxCtx given in Table 9-46, a specific value of the variable ctxIdx is further specified in subclause 9.3.3. bypassFlag is set equal to 0.

The possible values of the context index ctxIdx vary depending on the value of ctxIdxTable. The value assigned to ctxIdxOffset specifies the lower value of the range of ctxIdx assigned to the corresponding binarization or binarization part of a syntax element.

ctxIdxTable = 0 and ctxIdx = ctxIdxOffset = 0 are assigned to the syntax element end_of_slice_flag as further specified in subclause 9.3.3.1. For parsing the value of the corresponding bin from the bitstream, the arithmetic decoding process for decisions before termination (DecodeTerminate) as specified in subclause 9.3.3.2.4 is applied.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000700

Table 9-46 – Syntax elements and associated types of binarization, maxBinIdxCtx, ctxIdxTable, and ctxIdxOffset

Syntax element		Type of binarization	maxBinIdx Ctx	ctxIdxTable	ctxIdxOffset
alf_cu_flag	I	FL, cMax = 1	0	Table 9-21	0
	P		0	Table 9-21	1
	B		0	Table 9-21	2
end_of_slice_flag	all	FL, cMax = 1	0	0	0
split_coding_unit_flag	I	FL, cMax = 1	0	Table 9-22	0
	P		0	Table 9-22	3
	B		0	Table 9-22	6
skip_flag	P	FL, cMax = 1	0	Table 9-23	0
	B		0	Table 9-23	3
cu_qp_delta	I	as specified in subclause 9.3.2.6	2	Table 9-24	0
	P		2	Table 9-24	3
	B		2	Table 9-24	6
pred_type	I	as specified in subclause 9.3.2.7	0	Table 9-25	0
	P		3	Table 9-25	1
	B		4	Table 9-25	5
pcm_flag	all	FL, cMax = 1	0	0	0
prev_intra_luma_pred_flag	I	FL, cMax = 1	0	Table 9-26	0
	P		0	Table 9-26	1
	B		0	Table 9-26	2
rem_intra_luma_pred_mode	I	as specified in subclause 9.3.2.8	0	Table 9-27	0
	P		0	Table 9-27	1
	B		0	Table 9-27	2
intra_chroma_pred_mode (IntraPredMode < 4)	I	TU, cMax = 3	1	Table 9-28	0
	P		1	Table 9-28	2
	B		1	Table 9-28	4
intra_chroma_pred_mode (4 <= IntraPredMode < 34)	I	TU, cMax = 4	1	Table 9-28	0
	P		1	Table 9-28	2
	B		1	Table 9-28	4
merge_flag	P	FL, cMax = 1	0	Table 9-29	0
	B		0	Table 9-29	1
merge_idx	P	TU, cMax = 5	3	Table 9-30	0
	B		3	Table 9-30	4
inter_pred_flag	B	FL, cMax = 1	0	Table 9-31	0

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000701

Table 9-46 – Syntax elements and associated types of binarization, maxBinIdxCtx, ctxIdxTable, and ctxIdxOffset

Syntax element		Type of binarization	maxBinIdxCtx	ctxIdxTable	ctxIdxOffset
ref_idx_lc	B	TU, cMax = num_ref_idx_lC_active_minus1	2	Table 9-32	3
ref_idx_l0	P	TU, cMax = num_ref_idx_l0_active_minus1	2	Table 9-32	0
	B		2	Table 9-32	3
ref_idx_l1	B	TU, cMax = num_ref_idx_l1_active_minus1	2	Table 9-32	3
abs_mvd_greater0_flag[ ]	P	FL, cMax = 1	0	Table 9-33	0
	B		0	Table 9-33	1
abs_mvd_greater1_flag[ ]	P	FL, cMax = 1	0	Table 9-33	2
	B		0	Table 9-33	3
abs_mvd_minus2[ ]	P/B	EG1	na	na	na, (uses Decode Bypass)
mvd_sign_flag[ ]	P/B	FL, cMax = 1	na	na	na, (uses Decode Bypass)
mvp_idx_lc	B	TU, cMax = 2	1	Table 9-34	2
mvp_idx_l0	P	TU, cMax = 2	1	Table 9-34	0
	B		1	Table 9-34	2
mvp_idx_l1	B	TU, cMax = 2	1	Table 9-34	2
no_residual_data_flag	P	FL, cMax = 1	0	Table 9-35	0
	B		0	Table 9-35	1
split_transform_flag	I	FL, cMax = 1	0	Table 9-36	0
	P		0	Table 9-36	4
	B		0	Table 9-36	8
cbf_luma	I	FL, cMax = 1	0	Table 9-37	0
	P		0	Table 9-37	2
	B		0	Table 9-37	4
cbf_cb	I	FL, cMax = 1	0	Table 9-38	0
	P		0	Table 9-38	4
	B		0	Table 9-38	8
cbf_cr	I	FL, cMax = 1	0	Table 9-39	0
	P		0	Table 9-39	4
	B		0	Table 9-39	8
last_significant_coeff_x, last_significant_coeff_y	I	prefix and suffix as specified in subclause 9.3.2.9	30	Table 9-40	0
	P		30	Table 9-40	31
	B		30	Table 9-40	62

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000702

Table 9-46 – Syntax elements and associated types of binarization, maxBinIdxCtx, ctxIdxTable, and ctxIdxOffset

Syntax element		Type of binarization	maxBinIdx Ctx	ctxIdxTable	ctxIdxOffset
significant_coeff_flag	I	FL, cMax = 1	0	Table 9-41	0
	P		0	Table 9-42	0
	B		0	Table 9-43	0
coeff_abs_level_greater1_flag	I	FL, cMax = 1	0	Table 9-44	0
	P		0	Table 9-44	60
	B		0	Table 9-44	120
coeff_abs_level_greater2_flag	I	FL, cMax = 1	0	Table 9-45	0
	P		0	Table 9-45	60
	B		0	Table 9-45	120
coeff_abs_level_minus3	all	prefix and suffix as specified in subclause 9.3.2.9	prefix: na suffix: na	prefix: na suffix: na	prefix: na, (uses Decode Bypass) suffix: na, (uses Decode Bypass)
coeff_sign_flag	all	FL, cMax = 1	na	na	na, (uses Decode Bypass)

[Ed. (BB): The TU binarization for last_significant_coeff_x and last_significant_coeff_y is using 0s and a terminating 1 instead of using 1s and a terminating 0.]

9.3.2.1 Unary (U) binarization process

Input to this process is a request for a U binarization for a syntax element.

Output of this process is the U binarization of the syntax element.

The bin string of a syntax element having (unsigned integer) value synElVal is a bit string of length synElVal + 1 indexed by binIdx. The bins for binIdx less than synElVal are equal to 1. The bin with binIdx equal to synElVal is equal to 0.

Table 9-47 illustrates the bin strings of the unary binarization for a syntax element.

Table 9-47 – Bin string of the unary binarization (informative)

Value of syntax element	Bin string					
0 (I_NxN)	0					
1	1	0				
2	1	1	0			
3	1	1	1	0		
4	1	1	1	1	0	
5	1	1	1	1	1	0
…						
binIdx	0	1	2	3	4	5

9.3.2.2 Truncated unary (TU) binarization process

Input to this process is a request for a TU binarization for a syntax element and cMax.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000703

Output of this process is the TU binarization of the syntax element.

For syntax element (unsigned integer) values less than cMax, the U binarization process as specified in subclause 9.3.2.1 is invoked. For the syntax element value equal to cMax the bin string is a bit string of length cMax with all bins being equal to 1.

NOTE – TU binarization is always invoked with a cMax value equal to the largest possible value of the syntax element being decoded.

9.3.2.3 Truncated Rice (TR) binarization process

Input to this process is a request for a TR binarization for a syntax element, cRiceParam and cTRMax.

Output of this process is the TR binarization of the syntax element.

A TR bin string is a concatenation of a prefix bit string and a suffix bit string. The prefix of the binarization is specified by invoking the TU binarization process for the prefix part of the value specified by synElVal >> cRiceParam with cMax = cTRMax >> cRiceParam. The suffix of the TR bin string is the binary representation of synElVal – ((synElVal >> cRiceParam) << cRiceParam).

NOTE – For the input parameter cRiceParam = 0 the TR binarization is exactly the TU binarization.

9.3.2.4 k-th order Exp-Golomb (EGk) binarization process

Input to this process is a request for an EGk binarization for a syntax element.

Output of this process is the EGk binarization of the syntax element.

The bin string of the EGk binarization process of a syntax element synVal is specified by a process equivalent to the following pseudo-code:

```
absV = Abs( synVal )
stopLoop = 0
do {
    if( absV >= ( 1 << k ) ) {
        put( 1 )
        absV = absV – ( 1 << k )
        k++
    } else {
        put( 0 )                                                    (9-5)
        while( k– – )
            put( ( absV >> k ) & 1 )
        stopLoop = 1
    }
} while( !stopLoop )
```

NOTE 1 – The specification for the k-th order Exp-Golomb (EGk) code uses 1's and 0's in reverse meaning for the unary part of the Exp-Golomb code of 0-th order as specified in subclause 9.1.

9.3.2.5 Fixed-length (FL) binarization process

Input to this process is a request for a FL binarization for a syntax element and cMax.

Output of this process is the FL binarization of the syntax element.

FL binarization is constructed by using a fixedLength-bit unsigned integer bin string of the syntax element value, where fixedLength = Ceil(Log2(cMax + 1)). The indexing of bins for the FL binarization is such that the binIdx = 0 relates to the least significant bit with increasing values of binIdx towards the most significant bit.

9.3.2.6 Binarization process for cu_qp_delta

Input to this process is a request for a binarization for the syntax element cu_qp_delta.

Output of this process is the binarization of the syntax element.

The bin string of this binarization process of a syntax element value synVal is specified by a process equivalent to the following pseudo-code.

```
absV = Abs( synVal )
if( absV = = 0 ){
    put( 0 )
} else {
    put( 1 )
```


Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000704

```
        signV = ( synVal > 0 ) ? 0 : 1
        put( signV )
        cMax = 24 + ( QpBdOffsetY >> 1 ) + signV
        cNum = absV − 1
        absVGreaterThan1Flag = ( absV = = 1 ) ? 0 : 1
        put( absVGreaterThan1Flag )
        if( absVGreaterThan1Flag ){
            while( cNum−− )
                put( 1 )
            if( cMax > absV − 1 )
                put( 0 )
        }
    }
```

9.3.2.7 Binarization process for pred_type

Input to this process is a request for a binarization for the syntax element pred_type and a variable cLog2CUSize specifying the current CU size.

Output of this process is the binarization of the syntax element.

The binarization for pred_type is given by Table 9-48 depending on slice type and the size of the coding unit.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000705

Table 9-48 – Binarization for pred_type

Slice type	Value of pred_type	PredMode	PartMode	Bin string		
				cLog2CUSize > Log2MinCUSize	cLog2CUSize = = Log2MinCUSize	
					cLog2CUSize = = 3 && !inter_4x4_enabled_flag	cLog2CUSize > 3 || inter_4x4_enabled_flag
I	0	MODE_INTRA	PART_2Nx2N	-	1	1
	1	MODE_INTRA	PART_NxN	-	0	0
P	0	MODE_INTER	PART_2Nx2N	0 1	0 1	0 1
	1	MODE_INTER	PART_2NxN	0 011	0 01	0 01
	2	MODE_INTER	PART_Nx2N	0 001	0 00	0 001
	4	MODE_INTER	PART_2NxnU	0 0100	-	-
	5	MODE_INTER	PART_2NxnD	0 0101	-	-
	6	MODE_INTER	PART_nLx2N	0 0000	-	-
	7	MODE_INTER	PART_nRx2N	0 0001	-	-
	3	MODE_INTER	PART_NxN	-	-	0 000
	4	MODE_INTRA	PART_2Nx2N	1	11	11
	5	MODE_INTRA	PART_NxN	-	10	10
B	0	MODE_INTER	PART_2Nx2N	1	1	1
	1	MODE_INTER	PART_2NxN	011	01	01
	2	MODE_INTER	PART_Nx2N	001	001	001
	4	MODE_INTER	PART_2NxnU	0100	-	-
	5	MODE_INTER	PART_2NxnD	0101	-	-
	6	MODE_INTER	PART_nLx2N	0000	-	-
	7	MODE_INTER	PART_nRx2N	0001	-	-
	3	MODE_INTER	PART_NxN	-	-	0001
	4	MODE_INTRA	PART_2Nx2N	000	000 0	0000 0
	5	MODE_INTRA	PART_NxN	-	000 1	0000 1

9.3.2.8 Binarization process for rem_intra_luma_pred_mode

Input to this process is a request for the syntax element rem_intra_luma_pred_mode and cNumBins.

Output of this process is the binarization of the syntax element.

The binarization for rem_intra_luma_pred_mode is given by Table 9-49.

Table 9-49 – Binarization for rem_intra_luma_pred_mode

Value of rem_intra_luma_pred_mode	Bin string
less than 32	FL, cMax = cNumBins
32	111110
33	111111

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000706

9.3.2.9 Binarization process for last_significant_coefficient_x and last_significant_coefficient_y

Input to this process is a request for a binarization for the syntax element last_significant_coefficient_pos with pos being x or y and log2TrafoSize.

Output of this process is the binarization of the syntax element.

The variable halfTrafoSize is set equal to 1 << (log2TrafoSize – 1).

- If log2TrafoSize is smaller than 3, the TU binarization process as specified in subclause 9.3.2.2 is invoked with the value of the syntax element last_significant_coeff_position and cMax = halfTrafoSize + 1 as input and the bin string as output.
- Otherwise if last_significant_coefficient_pos is smaller than halfTrafoSize, the TU binarization process as specified in subclause 9.3.2.2 is invoked with the value of the syntax element last_significant_coefficient_pos and cMax = halfTrafoSize as input and the bin string as output.
- Otherwise (log2TrafoSize is greater than 2 and last_significant_coefficient_pos is greater than halfTrafoSize – 1), the bin string of the syntax element consists of a prefix bin string and suffix bin string as follows.
  - The prefix bin string is derived by the TU binarization process as specified in subclause 9.3.2.2 with the symbol value of halfTrafoSize and cMax = halfTrafoSize as input and the bin string as output.
  - The suffix bin string is derived by the FL binarization process as specified in subclause 9.3.2.5 with the symbol value of last_significant_coefficient_pos – halfTrafoSize and cMax = halfTrafoSize – 1 as input and the bin string as output.

9.3.2.10 Binarization process for coeff_abs_level_minus3

Input to this process is a request for a binarization for the syntax element coeff_abs_level_minus3[n].

Output of this process is the binarization of the syntax element.

The variables cLastSE and cLastRiceParam are derived as follows.

- If n is equal to 0 or all previous syntex elements coeff_abs_level_minus3[pos] with pos less than n are derived to be equal to 0 instead of being explicitly parsed, cLastSE and cLastRiceParam are set equal to 0.
- Otherwise (n is not equal to 0), cLastSE is set equal to coeff_abs_level_minus3[n – 1] and cLastRiceParam is set equal to the value of cRiceParam that has been derived during the invocation of this subclause for the syntax element coeff_abs_level_minus2[n – 1] of the same transform block.

The variable cRiceParam is derived as follows.

- If cLastSE is equal to 0 or 1 and cLastRiceParam is equal to 0, cRiceParam is set equal to 0.
- Otherwise if cLastSE is equal to 2 or 3 and cLastRiceParam is equal to 0 or 1, cRiceParam is set equal to 1.
- Otherwise if one of the following conditions is true, cRiceParam is set equal to 2.
  - cLastSE is in the range of 4 to 12 and cLastRiceParam is equal to 0.
  - cLastSE is in the range of 4 to 10 and cLastRiceParam is equal to 1.
  - cLastSE is in the range of 4 to 9 and cLastRiceParam is equal to 2.
- Otherwise, cRiceParam is set equal to 3.

The variable cTRMax is derived from cRiceParam as given in Table 9-50.

Table 9-50 – Specifcation of cTRMax depending on cRiceParam

cRiceParam	cTRMax
0	7
1	20
2	42
3	70

The binarization of coeff_abs_level_minus3 consists of a prefix part and (when present) a suffix part. The prefix part of the binarization is derived by invoking the TR binarization process as specified in subclause 9.3.2.3 for the syntax element coeff_abs_level_minus3[n] with the variables cRiceParam and cTRMax as the inputs.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000707

When the prefix bit string is not equal to the bin string that consists of (cTRMax + 1 >> cRiceParam) + cRiceParam ones, the bin string consists of a prefix bin string and a suffix bin string. The suffix bin string is derived using the EG0 binarization as specified in subclause 9.3.2.4.

9.3.3 Decoding process flow

Input to this process is a binarization of the requested syntax element, maxBinIdxCtx, bypassFlag, ctxIdxTable and ctxIdxOffset as specified in subclause 9.3.2.

Output of this process is the value of the syntax element.

This process specifies how each bit of a bit string is parsed for each syntax element.

After parsing each bit, the resulting bit string is compared to all bin strings of the binarization of the syntax element and the following applies.

– If the bit string is equal to one of the bin strings, the corresponding value of the syntax element is the output.

– Otherwise (the bit string is not equal to one of the bin strings), the next bit is parsed.

While parsing each bin, the variable binIdx is incremented by 1 starting with binIdx being set equal to 0 for the first bin.

When the binarization of the corresponding syntax element consists of a prefix and a suffix binarization part, the variable binIdx is set equal to 0 for the first bin of each part of the bin string (prefix part or suffix part). In this case, after parsing the prefix bit string, the parsing process of the suffix bit string related to the binarizations specified in subclauses 9.3.2.5 9.3.2.3 and 9.3.2.10 9.3.2.10 is invoked depending on the resulting prefix bit string as specified in subclauses 9.3.2.39.3.2.5 and 9.3.2.109.3.2.10.

Depending on the variable bypassFlag, the following applies.

– If bypassFlag is equal to 1, the bypass decoding process as specified in subclause 9.3.3.2.3 is applied for parsing the value of the bins from the bitstream.

– Otherwise (bypassFlag is equal to 0), the parsing of each bin is specified by the following two ordered steps:

1. Given binIdx, maxBinIdxCtx, ctxIdxTable, and ctxIdxOffset, ctxIdx is derived as specified in subclause 9.3.3.1.
2. Given ctxIdxTable and ctxIdx, the value of the bin from the bitstream as specified in subclause 9.3.3.2 is decoded.

9.3.3.1 Derivation process for ctxIdx

Inputs to this process are binIdx, maxBinIdxCtx, ctxIdxTable, and ctxIdxOffset.

Output of this process is ctxIdx.

Table 9-51 shows the assignment of ctxIdx increments (ctxIdxInc) to binIdx for all ctxIdxTable and ctxIdxOffset.

The ctxIdx to be used with a specific binIdx is specified by first determining the ctxIdxTable and ctxIdxOffset associated with the given bin string or part thereof. The ctxIdxTable and the ctxIdxOffset ~~is~~ are listed in Table 9-51, the ctxIdx for a binIdx is the sum of ctxIdxOffset and ctxIdxInc, which is found in Table 9-51. When more than one value is listed in Table 9-51 for a binIdx, the assignment process for ctxIdxInc for that binIdx is further specified in the subclauses given in parenthesis of the corresponding table entry.

All bins with binIdx greater than maxBinIdxCtx are parsed using the value of ctxIdx being assigned to binIdx equal to maxBinIdxCtx.

All entries in Table 9-51 labelled with "na" correspond to values of binIdx that do not occur for the corresponding ctxIdxOffset.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000708

Table 9-51 – Assignment of ctxIdxInc to binIdx for all ctxIdxTable and ctxIdxOffset values

Syntax element	ctxIdxTable, ctxIdxOffset		binIdx				
			0	1	2	3	>=4
alf_cu_flag	Table 9-21	0	0	na	na	na	na
		1	0	na	na	na	na
		2	0	na	na	na	na
split_coding_unit_flag	Table 9-22	0	0,1,2 (subclause 9.3.3.1.1.1)	na	na	na	na
		3	0,1,2 (subclause 9.3.3.1.1.1)	na	na	na	na
		6	0,1,2 (subclause 9.3.3.1.1.1)	na	na	na	na
skip_flag	Table 9-23	0	0,1,2 (subclause 9.3.3.1.1.1)	na	na	na	na
		3	0,1,2 (subclause 9.3.3.1.1.1)	na	na	na	na
cu_qp_delta	Table 9-24	0	0	na (uses Decode Bypass)	1	2	2
		3	0	na (uses Decode Bypass)	1	2	2
		6	0	na (uses Decode Bypass)	1	2	2
pred_type	Table 9-25	0	0	na	na	na	na
		1	0	1	2	3	na
		5	0	1	2	3	4
prev_intra_luma_pred_flag	Table 9-26	0	0	na	na	na	na
		1	0	na	na	na	na
		2	0	na	na	na	na
rem_intra_luma_pred_mode	Table 9-27	0	0	0	0	0	0
		1	0	0	0	0	0
		2	0	0	0	0	0
intra_chroma_pred_mode	Table 9-28	0	0	1	1	1	na
		2	0	1	1	1	na
		4	0	1	1	1	na
merge_flag	Table 9-29	0	0	na	na	na	na
		1	0	na	na	na	na
merge_idx	Table 9-30	0	0,1,2,3 (subclause 9.3.3.1.1.2)	0,1,2,3 (subclause 9.3.3.1.1.2)	0,2,3 (subclause 9.3.3.1.1.2)	3	na
		4	0,1,2,3 (subclause 9.3.3.1.1.2)	0,1,2,3 (subclause 9.3.3.1.1.2)	0,2,3 (subclause 9.3.3.1.1.2)	3	na
inter_pred_flag	Table 9-31	0	cuDepth	na	na	na	na

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000709

Table 9-51 – Assignment of ctxIdxInc to binIdx for all ctxIdxTable and ctxIdxOffset values

Syntax element	ctxIdxTable, ctxIdxOffset		binIdx				
			0	1	2	3	>=4
ref_idx_l0	Table 9-32	0	0	1	2	2	2
ref_idx_l0, ref_idx_l1, ref_idx_lc	Table 9-32	3	0	1	2	2	2
abs_mvd_greater0_flag[ ]	Table 9-33	0	0	na	na	na	na
		1	0	na	na	na	na
abs_mvd_greater1_flag[ ]	Table 9-33	2	0	na	na	na	na
		3	0	na	na	na	na
mvp_idx_l0	Table 9-34	0	0	1	1	1	1
mvp_idx_l0, mvp_idx_l1, mvp_idx_lc	Table 9-34	2	0	1	1	1	1
no_residual_data_flag	Table 9-35	0	0	na	na	na	na
		1	0	na	na	na	na
split_transform_flag	Table 9-36	0	cuDepth + trafoDepth	na	na	na	na
		4	cuDepth + trafoDepth	na	na	na	na
		8	cuDepth + trafoDepth	na	na	na	na
cbf_luma	Table 9-37	0	(trafoDepth = = 0) || (log2TrafoSize = = Log2MaxTrafoSize) ? 1 : 0	na	na	na	na
		2	(trafoDepth = = 0) || (log2TrafoSize = = Log2MaxTrafoSize) ? 1 : 0	na	na	na	na
		4	(trafoDepth = = 0) || (log2TrafoSize = = Log2MaxTrafoSize) ? 1 : 0	na	na	na	na
cbf_cb	Table 9-38	0	trafoDepth	na	na	na	na
		4	trafoDepth	na	na	na	na
		8	trafoDepth	na	na	na	na
cbf_cr	Table 9-39	0	trafoDepth	na	na	na	na
		4	trafoDepth	na	na	na	na
		8	trafoDepth	na	na	na	na

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000710

Table 9-51 – Assignment of ctxIdxInc to binIdx for all ctxIdxTable and ctxIdxOffset values

Syntax element	ctxIdxTable, ctxIdxOffset		binIdx				
			0	1	2	3	>=4
last_significant_coeff_x, last_significant_coeff_y	Table 9-40	0	0..30 (subclause 9.3.3.1.1.3)				
		31	0..30 (subclause 9.3.3.1.1.3)				
		62	0..30 (subclause 9.3.3.1.1.3)				
significant_coeff_flag	Table 9-41	0	0..87 (subclause 9.3.3.1.1.4)	na	na	na	na
	Table 9-42	0	0..87 (subclause 9.3.3.1.1.4)	na	na	na	na
	Table 9-43	0	0..87 (subclause 9.3.3.1.1.4)	na	na	na	na
coeff_abs_level_greater1_flag	Table 9-44	0	0..59 (subclause 9.3.3.1.1.5)	na	na	na	na
		60	0..59 (subclause 9.3.3.1.1.5)	na	na	na	na
		120	0..59 (subclause 9.3.3.1.1.5)	na	na	na	na
coeff_abs_level_greater2_flag	Table 9-45	0	0..59 (subclause 9.3.3.1.1.6)	na	na	na	na
		60	0..59 (subclause 9.3.3.1.1.6)	na	na	na	na
		120	0..59 (subclause 9.3.3.1.1.6)	na	na	na	na

9.3.3.1.1 Assignment process of ctxIdxInc using neighbouring syntax elements

Subclause 9.3.3.1.1.1 specifies the derivation process of ctxIdxInc for the syntax elements split_coding_unit_flag and skip_flag.

9.3.3.1.1.1 Derivation process of ctxIdxInc using left and above syntax elements

Input to this process is the luma location (xP, yP) specifying the top-left luma sample of the current prediction unit relative to the top-left sample of the current picture.

Output of this process is ctxIdxInc.

Let the luma location (xL, yL) specify a location covered by the prediction unit to the left of the top-left luma sample of the current prediction unit with xL = xP − MinPuSize and yL = yP, the variable availableL is derived as follows. [Ed.: (BB) MinPuSize should be defined somewhere]

– If the prediction unit covering location (xL, yL) is available, availableL is set equal to 1.

– Otherwise (the prediction unit covering location (xL, yL) is not available), availableL is set equal to 0.

Let the luma location (xA, yA) specify a location covered by the prediction unit above the top-left luma sample of the current prediction unit with xA = xP and yA = yP − MinPuSize, the variable availableA is derived as follows. [Ed.: (BB) MinPuSize should be defined somewhere]

– If the prediction unit covering location (xA, yA) is available, availableA is set equal to 1.

– Otherwise (the prediction unit covering location (xA, yA) is not available), availableA is set equal to 0

The assignment of ctxIdxInc for the syntax elements alf_cu_flag and skip_flag is specified in Table 9-52.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000711

Table 9-52 – Specifcation of ctxIdxInc using left and above syntax elements

Syntax element	condL	condA	ctxIdxInc
split_coding_unit_flag	cuDepth[xL][yL] > cuDepth[xP][yP]	cuDepth[xA][yA] > cuDepth[xP][yP]	(condL && availableL) + (condA && availableA)
skip_flag	skip_flag[xL][yL]	skip_flag[xA][yA]	(condL && availableL) + (condA && availableA)

9.3.3.1.1.2 Derivation process of ctxIdxInc for the syntax element merge_index

Input to this process is binIdx.

Output of this process is ctxIdxInc.

The variable ctxIdxInc is derived as follows.

– If binIdx is equal to 0, the following applies.
  – If availableLeft is equal to 1and availableAbove is equal to 1, ctxIdxInc is set equal to 0.
  – Otherwise if availableLeft is equal to 1 or availableAbove is equal to 1, ctxIdxInc is set equal to 2.
  – Otherwise (availableLeft is equal to 0 and availableAbove is equal to 0), ctxIdxInc is set equal to 3.
– Otherwise if binIdx is equal to 1 and NumMergeCand is greater than 2, the following applies.
  – If availableAbove is equal to 1, ctxIdxInc is set equal to 2.
  – Otherwise (availableAbove is equal to 0), ctxIdxInc is set equal to 3.
– Otherwise if binIdx is equal to 2 and NumMergeCand is greater than 3, ctxIdxInc is set equal to 3.
– Otherwise, ctxIdxInc is set equal to 0.

When ctxIdxInc is not equal to 0, ctxIdxInc is derived as follows.

$$ctxIdxInc = ctxIdxInc - availableCollocated \quad (9\text{-}51)$$

[Ed. (BB): The flags availableLeft, availableAbove, availableCollocated need to be defined or a simpler context derivation should be used.]

9.3.3.1.1.3 Derivation process of ctxIdxInc for the syntax elements last_significant_coeff_x and last_significant_coeff_y

Inputs to this process are the binIdx, the color component index cIdx and the transform block size log2TrafoSize.

Output of this process is ctxIdxInc.

Table 9-53 – Specifcation of lastCtx[i]

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
lastCtx	0	1	2	3	3	4	4	5	5	5	5	6	6	6	6	7
i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	
lastCtx	7	7	7	8	8	8	8	9	9	9	9	10	10	10	10	

The vector lastCtx[i] is specified in Table 9-53. For the derivation of ctxIdxInc, the following applies.

– If log2TrafoSize is less than or equal to 2, ctxIdxInc is derived as follows.

$$ctxIdxInc = lastCtx[\ binIdx\ ] \quad (9\text{-}52)$$

– Otherwise (log2TrafoSize is greater than 2), ctxIdxInc is derived as follows.

$$ctxIdxInc = lastCtx[\ (\ 1 << (\ log2TrafoSize - 1\ )\ ) + binIdx\ ] \quad (9\text{-}53)$$

When cIdx is greater than 0, ctxIdxInc is modified as follows.

$$ctxIdxInc = ctxIdxInc + 19 \quad (9\text{-}54)$$

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000712

[Ed. (BB): The context derivation assumes maximum transform sizes less than or equal to 32x32 for luma and 16x16 for chroma and minimum transform sizes greater than or equal to 4x4.]

9.3.3.1.1.4 Derivation process of ctxIdxInc for the syntax element significant_coeff_flag

Inputs to this process are the color component index cIdx, the current coefficient scan position (xC , yC) and the transform block size log2TrafoSize.

Output of this process is ctxIdxInc.

The variable sigCtx depends on the current position (xC, yC), the transform block size and previsously decoded bins of the syntax element significant_coeff_flag. For the derivation of sigCtx, the following applies.

– If log2TrafoSize is less than or equal to 3, sigCtx is derived as follows.

```
shift  = log2TrafoSize == 3 ? 1 : 0
sigCtx = ( shift * 15 ) + ( ( yC >> shift ) << 2 ) + ( xC >> shift )        (9-55)
```

– Otherwise if xC + yC is less than 2, sigCtx is derived as follows.

```
sigCtx = 31 + ( yC << 1 ) + xC                                              (9-56)
```

– Otherwise if xC + yC is less than 5, sigCtx is derived as follows.

```
temp = significant_coeff_flag[ xC + 1 ][ yC ] + significant_coeff_flag[ xC + 2 ][ yC ] +
       significant_coeff_flag[ xC ][ yC + 1 ] + significant_coeff_flag[ xC + 1 ][ yC + 1 ] +
       significant_coeff_flag[ xC ][ yC + 2 ]                               (9-57)
sigCtx = 34 + Min( 4, temp )
```

– Otherwise (xC + yC is greater than 4), sigCtx is derived using previously decoded bins of the syntax element significant_coeff_flag as follows.

  – The variable sigCtx is initialized as follows.

```
sigCtx = 39                                                                 (9-58)
```

  – When xC is less than (1 << log2TrafoSize) – 1, the following applies.

```
sigCtx = sigCtx + significant_coeff_flag[ xC + 1 ][ yC ]                    (9-59)
```

  – When xC and yC are less than (1 << log2TrafoSize) – 1, the following applies.

```
sigCtx = sigCtx + significant_coeff_flag[ xC + 1 ][ yC + 1 ]                (9-60)
```

  – When xC is less than (1 << log2TrafoSize) – 2, the following applies.

```
sigCtx = sigCtx + significant_coeff_flag[ xC + 2 ][ yC ]                    (9-61)
```

  – When yC is less than (1 << log2TrafoSize) – 1, the following applies.

```
sigCtx = sigCtx + significant_coeff_flag[ xC ][ yC + 1 ]                    (9-62)
```

  – When yC is less than (1 << log2TrafoSize) – 2 and sigCtx is less than 43, the following applies.

```
sigCtx = sigCtx + significant_coeff_flag[ xC ][ yC + 2 ]                    (9-63)
```

The context index increment ctxIdxInc is derived using the color component index cIdx and sigCtx as follows.

– If cIdx is equal to 0, ctxIdxInc is derived as follows.

```
ctxIdxInc = sigCtx                                                          (9-64)
```

– Otherwise (cIdx is greater than 0), ctxIdxInc is derived as follows.

```
ctxIdxInc = 44 + sigCtx                                                     (9-65)
```


Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000713

[Ed. (BB): The context derivation assumes maximum transform sizes less than or equal to 32x32 for luma and 16x16 for chroma and minimum transform sizes greater than or equal to 4x4.]

9.3.3.1.1.5 Derivation process of ctxIdxInc for the syntax element coeff_abs_level_greater1_flag

Inputs to this process are the color component index cIdx, the 16 coefficient subset index i and the current coefficient scan index n within the current subset.

Output of this process is ctxIdxInc.

The variable ctxSet specifies the current context set and for its derivation the following applies.

- If n is equal to 15 or all previous syntax elements coeff_abs_level_greater1_flag[pos] with pos greater than n are derived to be equal to 0 instead of being explicitly parsed, the following applies.
  - The variable ctxSet is initialized as follows.
    - If the current subset index i is equal to 0, the following applies.

      $$ctxSet = 0 \quad (9\text{-}66)$$

    - Otherwise (i is greater than 0), the following applies.

      $$ctxSet = 3 \quad (9\text{-}67)$$

  - When the subset i is not the first one to be processed in this subclause, the following applies.
    - The variable numGreater1 is set equal to the variable numGreater1 that has been derived during the last invocation of subclause 9.3.3.1.1.6 for the syntax element coeff_abs_level_greater2_flag for the subset i + 1.
    - When (numGreater1 >> 1) is greater than 0, ctxSet is incremented by one as follows.

      $$ctxSet = ctxSet + 1 \quad (9\text{-}68)$$

    - When (numGreater1 >> 1) is greater than 3, ctxSet is incremented by one as follows.

      $$ctxSet = ctxSet + 1 \quad (9\text{-}69)$$

  - The variable greater1Ctx is set equal to 1.
- Otherwise (coeff_abs_level_greater1_flag[n] is not the first to be parsed within the current subset i),.for the derivation of ctxSet and greater1Ctx the following applies.
  - The variable ctxSet is set equal to the variable ctxSet that has been derived during the last invocation of this subclause.
  - The variable greater1Ctx is set equal to the variable greater1Ctx that has been derived during the last invocation of this subclause.
  - When greater1Ctx is greater than 0, the variable lastGreater1Flag is set equal to the syntax element coeff_abs_level_greater1_flag that has been used during the last invocation of this subclause and greater1Ctx is modifed as follows.
    - If lastGreater1Flag is equal to 1, greater1Ctx is set equal to 0.
    - Otherwise (lastGreater1Flag is equal to 0), greater1Ctx is incremented by 1.

The context index increment ctxIdxInc is derived using the current context set ctxSet and the current context greater1Ctx as follows.

$$ctxIdxInc = (ctxSet * 5) + Min(4, greater1Ctx) \quad (9\text{-}70)$$

When cIdx is greater than 0, ctxIdxInc is modified as follows.

$$ctxIdxInc = ctxIdxInc + 30 \quad (9\text{-}71)$$

9.3.3.1.1.6 Derivation process of ctxIdxInc for the syntax element coeff_abs_level_greater2_flag

Inputs to this process are the color component index cIdx, the 16 coeffcient subset index i and the current coefficient scan index n within the current subset.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000714

Output of this process is ctxIdxInc.

The variable ctxSet specifies the current context set and for its derivation the following applies.

– If n is equal to 15 or all previous syntex elements coeff_abs_level_greater2_flag[pos] with pos greater than n are derived to be equal to 0 instead of being explicitly parsed, the following applies.
  – The variable ctxSet is initialized as follows.
    – If the current subset index i is equal to 0, the following applies.

      ctxSet = 0 (9-72)

    – Otherwise (i is greater than 0), the following applies.

      ctxSet = 3 (9-73)

  – The variable numGreater1 is set equal to 0.
  – When the subset i is not the first one to be processed in this subclause, the following applies.
    – The variable numGreater1 is set equal to the variable numGreater1 that has been derived during the last invocation of this subclause for the subset i + 1 and is modifed as follows.

      numGreater1 = numGreater1 >> 1 (9-74)

    – When numGreater1 is greater than 0, ctxSet is incremented by one as follows.

      ctxSet = ctxSet + 1 (9-75)

    – When numGreater1 is greater than 3, ctxSet is incremented by one as follows.

      ctxSet = ctxSet + 1 (9-76)

  – The variable greater2Ctx is set equal to 0.
– Otherwise (coeff_abs_level_greater2_flag[n] is not the first to be parsed within the current subset i), for the derivation of ctxSet and greater2Ctx the following applies.
  – The variable ctxSet is set equal to the variable ctxSet that has been derived during the last invocation of this subclause.
  – The variable greater2Ctx is set equal to the variable greater2Ctx that has been derived during the last invocation of this subclause incremented by 1.
  – The variable numGreater1 is set equal to the variable numGreater1 that has been derived during the last invocation of this subclause incremented by 1.

The context index increment ctxIdxInc is derived using the current context set ctxSet and the current context greater2Ctx as follows.

ctxIdxInc = (ctxSet * 5) + Min(4, greater2Ctx) (9-77)

When cIdx is greater than 0, ctxIdxInc is modified as follows.

ctxIdxInc = ctxIdxInc + 30 (9-78)

9.3.3.2 Arithmetic decoding process

Inputs to this process are the bypassFlag, ctxIdxTable, and ctxIdx as derived in subclause 9.3.3.1, and the state variables codIRange and codIOffset of the arithmetic decoding engine.

Output of this process is the value of the bin.

Figure 9-1 illustrates the whole arithmetic decoding process for a single bin. For decoding the value of a bin, the context index table ctxIdxTable and the ctxIdx is passed to the arithmetic decoding process DecodeBin(ctxIdxTable, ctxIdx), which is specified as follows.

– If bypassFlag is equal to 1, DecodeBypass( ) as specified in subclause 9.3.3.2.3 is invoked.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000715

– Otherwise, if bypassFlag is equal to 0, ctxIdxTable is equal 0 and ctxIdx is equal to 0, DecodeTerminate( ) as specified in subclause 9.3.3.2.4 is invoked.

– Otherwise (bypassFlag is equal to 0, ctxIdxTable is not equal to 0 and ctxIdx is not equal to 0), DecodeDecision( ) as specified in subclause 9.3.3.2.1 is applied.

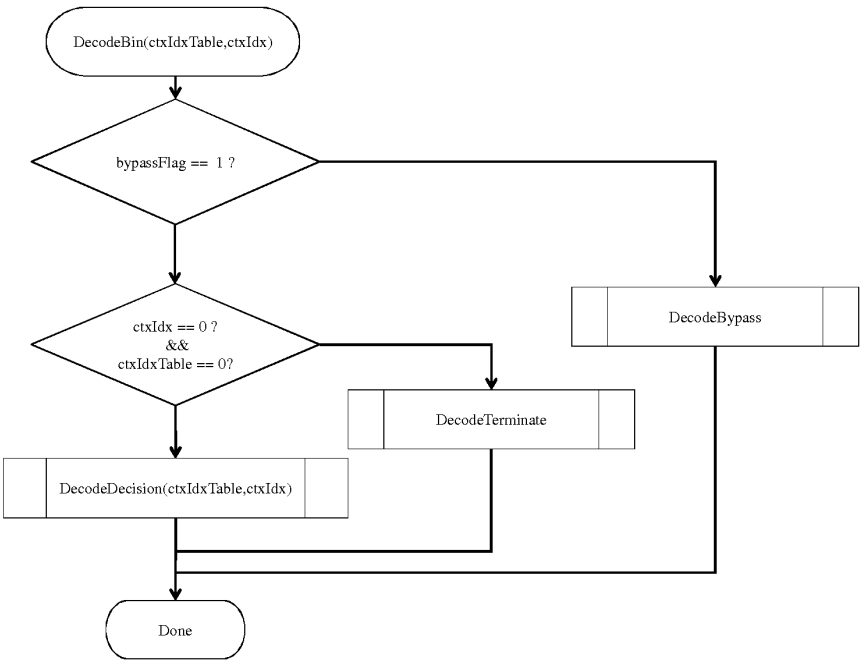


Figure 9-1 – Overview of the arithmetic decoding process for a single bin (informative)

NOTE – Arithmetic coding is based on the principle of recursive interval subdivision. Given a probability estimation p(0) and p(1) = 1 – p(0) of a binary decision (0, 1), an initially given code sub-interval with the range codIRange will be subdivided into two sub-intervals having range p(0) * codIRange and codIRange – p(0) * codIRange, respectively. Depending on the decision, which has been observed, the corresponding sub-interval will be chosen as the new code interval, and a binary code string pointing into that interval will represent the sequence of observed binary decisions. It is useful to distinguish between the most probable symbol (MPS) and the least probable symbol (LPS), so that binary decisions have to be identified as either MPS or LPS, rather than 0 or 1. Given this terminology, each context is specified by the probability p_{LPS} of the LPS and the value of MPS (valMPS), which is either 0 or 1.

The arithmetic core engine in this Recommendation | International Standard has three distinct properties:

– The probability estimation is performed by means of a finite-state machine with a table-based transition process between 64 different representative probability states { p_{LPS}(pStateIdx) | 0 <= pStateIdx < 64 } for the LPS probability p_{LPS}. The numbering of the states is arranged in such a way that the probability state with index pStateIdx = 0 corresponds to an LPS probability value of 0.5, with decreasing LPS probability towards higher state indices.

– The range codIRange representing the state of the coding engine is quantised to a small set $\{Q_1,\ldots,Q_4\}$ of pre-set quantisation values prior to the calculation of the new interval range. Storing a table containing all 64x4 pre-computed product values of Q_i * p_{LPS}(pStateIdx) allows a multiplication-free approximation of the product codIRange * p_{LPS}(pStateIdx).

– For syntax elements or parts thereof for which an approximately uniform probability distribution is assumed to be given a separate simplified encoding and decoding bypass process is used.

9.3.3.2.1 Arithmetic decoding process for a binary decision

Inputs to this process are ctxIdxTable, ctxIdx, codIRange, and codIOffset.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000716

Outputs of this process are the decoded value binVal, and the updated variables codIRange and codIOffset.

Figure 9-2 shows the flowchart for decoding a single decision (DecodeDecision):

1. The value of the variable codIRangeLPS is derived as follows.
   - Given the current value of codIRange, the variable qCodIRangeIdx is derived by

     qCodIRangeIdx =(codIRange >> 6) & 3 (9-29)

   - Given qCodIRangeIdx and pStateIdx associated with ctxIdxTable and ctxIdx, the value of the variable rangeTabLPS as specified in Table 9-40 is assigned to codIRangeLPS:

     codIRangeLPS = rangeTabLPS[pStateIdx][qCodIRangeIdx] (9-30)

2. The variable codIRange is set equal to codIRange – codIRangeLPS and the following applies.
   - If codIOffset is greater than or equal to codIRange, the variable binVal is set equal to 1 – valMPS, codIOffset is decremented by codIRange, and codIRange is set equal to codIRangeLPS.
   - Otherwise, the variable binVal is set equal to valMPS.

Given the value of binVal, the state transition is performed as specified in subclause 9.3.3.2.1.1. Depending on the current value of codIRange, renormalization is performed as specified in subclause 9.3.3.2.2.

9.3.3.2.1.1 State transition process

Inputs to this process are the current pStateIdx, the decoded value binVal and valMPS values of the context variable associated with ctxIdxTable and ctxIdx.

Outputs of this process are the updated pStateIdx and valMPS of the context variable associated with ctxIdx.

Depending on the decoded value binVal, the update of the two variables pStateIdx and valMPS associated with ctxIdx is derived as specified by the following pseudo-code:

```
if( binVal == valMPS )
    pStateIdx = transIdxMPS( pStateIdx )
else {                                                    (9-31)
    if( pStateIdx == 0 )
        valMPS = 1 – valMPS
    pStateIdx = transIdxLPS( pStateIdx )
}
```

Table 9-41 specifies the transition rules transIdxMPS( ) and transIdxLPS( ) after decoding the value of valMPS and 1 – valMPS, respectively.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000717

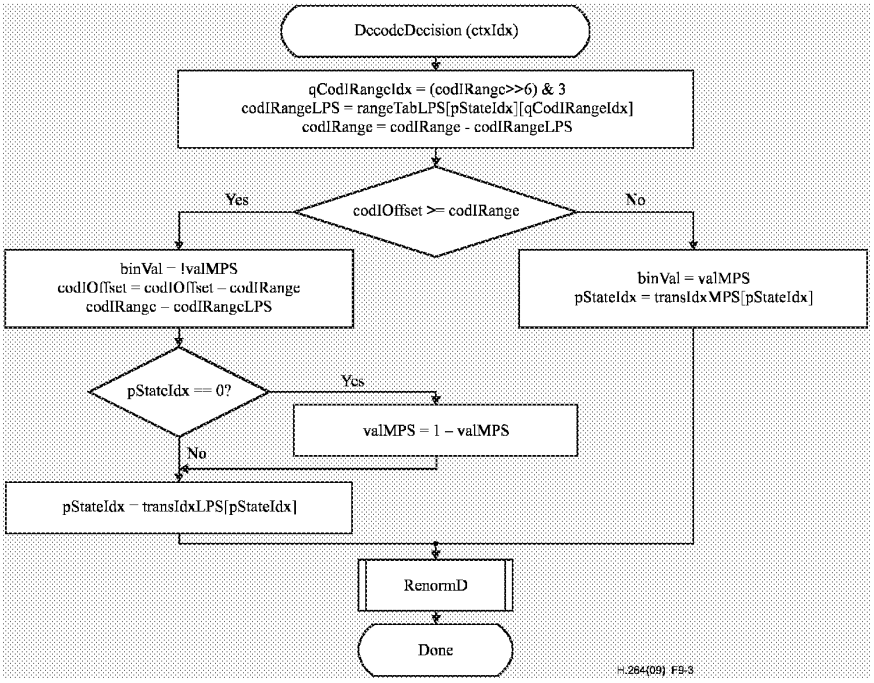


Figure 9-2 – Flowchart for decoding a decision

Table 9-53 – Specification of rangeTabLPS depending on pStateIdx and qCodIRangeIdx

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000718

pStateIdx	qCodIRangeIdx				pStateIdx	qCodIRangeIdx			
	0	1	2	3		0	1	2	3
0	128	176	208	240	**32**	27	33	39	45
1	128	167	197	227	**33**	26	31	37	43
2	128	158	187	216	**34**	24	30	35	41
3	123	150	178	205	**35**	23	28	33	39
4	116	142	169	195	**36**	22	27	32	37
5	111	135	160	185	**37**	21	26	30	35
6	105	128	152	175	**38**	20	24	29	33
7	100	122	144	166	**39**	19	23	27	31
8	95	116	137	158	**40**	18	22	26	30
9	90	110	130	150	**41**	17	21	25	28
10	85	104	123	142	**42**	16	20	23	27
11	81	99	117	135	**43**	15	19	22	25
12	77	94	111	128	**44**	14	18	21	24
13	73	89	105	122	**45**	14	17	20	23
14	69	85	100	116	**46**	13	16	19	22
15	66	80	95	110	**47**	12	15	18	21
16	62	76	90	104	**48**	12	14	17	20
17	59	72	86	99	**49**	11	14	16	19
18	56	69	81	94	**50**	11	13	15	18
19	53	65	77	89	**51**	10	12	15	17
20	51	62	73	85	**52**	10	12	14	16
21	48	59	69	80	**53**	9	11	13	15
22	46	56	66	76	**54**	9	11	12	14
23	43	53	63	72	**55**	8	10	12	14
24	41	50	59	69	**56**	8	9	11	13
25	39	48	56	65	**57**	7	9	11	12
26	37	45	54	62	**58**	7	9	10	12
27	35	43	51	59	**59**	7	8	10	11
28	33	41	48	56	**60**	6	8	9	11
29	32	39	46	53	**61**	6	7	9	10
30	30	37	43	50	**62**	6	7	8	9
31	29	35	41	48	**63**	2	2	2	2

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000719

Table 9-54 – State transition table

pStateIdx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
transIdxLPS	0	0	1	2	2	4	4	5	6	7	8	9	9	11	11	12
transIdxMPS	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
pStateIdx	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**
transIdxLPS	13	13	15	15	16	16	18	18	19	19	21	21	22	22	23	24
transIdxMPS	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
pStateIdx	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**	**45**	**46**	**47**
transIdxLPS	24	25	26	26	27	27	28	29	29	30	30	30	31	32	32	33
transIdxMPS	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
pStateIdx	**48**	**49**	**50**	**51**	**52**	**53**	**54**	**55**	**56**	**57**	**58**	**59**	**60**	**61**	**62**	**63**
transIdxLPS	33	33	34	34	35	35	35	36	36	36	37	37	37	38	38	63
transIdxMPS	49	50	51	52	53	54	55	56	57	58	59	60	61	62	62	63

9.3.3.2.2 Renormalization process in the arithmetic decoding engine

Inputs to this process are bits from slice data and the variables codIRange and codIOffset.

Outputs of this process are the updated variables codIRange and codIOffset.

A flowchart of the renormalization is shown in Figure 9-3. The current value of codIRange is first compared to 256 and further steps are specified as follows.

– If codIRange is greater than or equal to 256, no renormalization is needed and the RenormD process is finished;

– Otherwise (codIRange is less than 256), the renormalization loop is entered. Within this loop, the value of codIRange is doubled, i.e., left-shifted by 1 and a single bit is shifted into codIOffset by using read_bits(1).

The bitstream shall not contain data that result in a value of codIOffset being greater than or equal to codIRange upon completion of this process.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000720

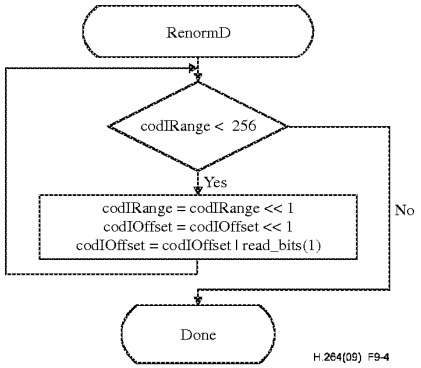


Figure 9-3 – Flowchart of renormalization

9.3.3.2.3 Bypass decoding process for binary decisions

Inputs to this process are bits from slice data and the variables codIRange and codIOffset.

Outputs of this process are the updated variable codIOffset and the decoded value binVal.

The bypass decoding process is invoked when bypassFlag is equal to 1. Figure 9-4 shows a flowchart of the corresponding process.

First, the value of codIOffset is doubled, i.e., left-shifted by 1 and a single bit is shifted into codIOffset by using read_bits(1). Then, the value of codIOffset is compared to the value of codIRange and further steps are specified as follows.

– If codIOffset is greater than or equal to codIRange, the variable binVal is set equal to 1 and codIOffset is decremented by codIRange.

– Otherwise (codIOffset is less than codIRange), the variable binVal is set equal to 0.

The bitstream shall not contain data that result in a value of codIOffset being greater than or equal to codIRange upon completion of this process.

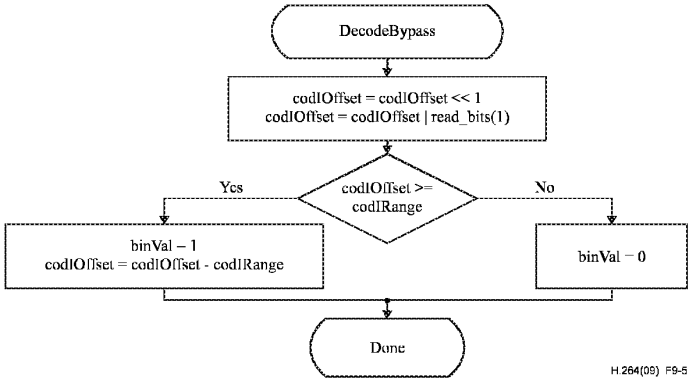


Figure 9-4 – Flowchart of bypass decoding process

9.3.3.2.4 Decoding process for binary decisions before termination

Inputs to this process are bits from slice data and the variables codIRange and codIOffset.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000721

Outputs of this process are the updated variables codIRange and codIOffset, and the decoded value binVal.

This special decoding routine applies to decoding of end_of_slice_flag and pcm_flag corresponding to ctxIdxTable equal to 0 and ctxIdx equal to 0. Figure 9-5 shows the flowchart of the corresponding decoding process, which is specified as follows.

First, the value of codIRange is decremented by 2. Then, the value of codIOffset is compared to the value of codIRange and further steps are specified as follows.

– If codIOffset is greater than or equal to codIRange, the variable binVal is set equal to 1, no renormalization is carried out, and CABAC decoding is terminated. The last bit inserted in register codIOffset is equal to 1. When decoding end_of_slice_flag, this last bit inserted in register codIOffset is interpreted as rbsp_stop_one_bit.

– Otherwise (codIOffset is less than codIRange), the variable binVal is set equal to 0 and renormalization is performed as specified in subclause 9.3.3.2.2.

NOTE – This procedure may also be implemented using DecodeDecision(ctxIdxTable, ctxIdx) with ctxIdxTable = 0 and ctxIdx = 0. In the case where the decoded value is equal to 1, seven more bits would be read by DecodeDecision(ctxIdxTable, ctxIdx) and a decoding process would have to adjust its bitstream pointer accordingly to properly decode following syntax elements.

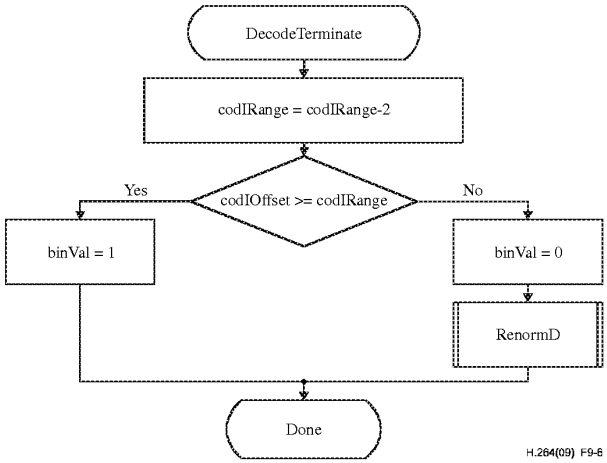


Figure 9-5 – Flowchart of decoding a decision before termination

9.3.4 Arithmetic encoding process (informative)

This subclause does not form an integral part of this Recommendation | International Standard.

Inputs to this process are decisions that are to be encoded and written.

Outputs of this process are bits that are written to the RBSP.

This informative subclause describes an arithmetic encoding engine that matches the arithmetic decoding engine described in subclause 9.3.3.2. The encoding engine is essentially symmetric with the decoding engine, i.e., procedures are called in the same order. The following procedures are described in this section: InitEncoder, EncodeDecision, EncodeBypass, EncodeTerminate, which correspond to InitDecoder, DecodeDecision, DecodeBypass, and DecodeTerminate, respectively. The state of the arithmetic encoding engine is represented by a value of the variable codILow pointing to the lower end of a sub-interval and a value of the variable codIRange specifying the corresponding range of that sub-interval.

9.3.4.1 Initialisation process for the arithmetic encoding engine (informative)

This subclause does not form an integral part of this Recommendation | International Standard.

This process is invoked before encoding the first macroblock of a slice, and after encoding any pcm_alignment_zero_bit and all pcm_sample_luma and pcm_sample_chroma data for a coding unit of type I_PCM.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000722

Outputs of this process are the values codILow, codIRange, firstBitFlag, bitsOutstanding, and BinCountsInNALunits of the arithmetic encoding engine.

In the initialisation procedure of the encoder, codILow is set equal to 0, and codIRange is set equal to 510. Furthermore, firstBitFlag is set equal to 1 and the counter bitsOutstanding is set equal to 0.

Depending on whether the current slice is the first slice of a coded picture, the following applies.

– If the current slice is the first slice of a coded picture, the counter BinCountsInNALunits is set equal to 0.

– Otherwise (the current slice is not the first slice of a coded picture), the counter BinCountsInNALunits is not modified. The value of BinCountsInNALunits is the result of encoding all the slices of a coded picture that precede the current slice in decoding order. After initialising for the first slice of a coded picture as specified in this subclause, BinCountsInNALunits is incremented as specified in subclauses 9.3.4.2, 9.3.4.4, and 9.3.4.5.

NOTE – The minimum register precision required for storing the values of the variables codILow and codIRange after invocation of any of the arithmetic encoding processes specified in subclauses 9.3.4.2, 9.3.4.4, and 9.3.4.5 is 10 bits and 9 bits, respectively. The encoding process for a binary decision (EncodeDecision) as specified in subclause 9.3.4.2 and the encoding process for a binary decision before termination (EncodeTerminate) as specified in subclause 9.3.4.5 require a minimum register precision of 10 bits for the variable codILow and a minimum register precision of 9 bits for the variable codIRange. The bypass encoding process for binary decisions (EncodeBypass) as specified in subclause 9.3.4.4 requires a minimum register precision of 11 bits for the variable codILow and a minimum register precision of 9 bits for the variable codIRange. The precision required for the counters bitsOutstanding and BinCountsInNALunits should be sufficiently large to prevent overflow of the related registers. When maxBinCountInSlice denotes the maximum total number of binary decisions to encode in one slice and maxBinCountInPic denotes the maximum total number of binary decisions to encode a picture, the minimum register precision required for the variables bitsOutstanding and BinCountsInNALunits is given by Ceil(Log2(maxBinCountInSlice + 1)) and Ceil(Log2(maxBinCountInPic + 1)), respectively.

9.3.4.2 Encoding process for a binary decision (informative)

This subclause does not form an integral part of this Recommendation | International Standard.

Inputs to this process are the context index ctxIdx, the value of binVal to be encoded, and the variables codIRange, codILow and BinCountsInNALunits.

Outputs of this process are the variables codIRange, codILow, and BinCountsInNALunits.

Figure 9-6 shows the flowchart for encoding a single decision. In a first step, the variable codIRangeLPS is derived as follows.

Given the current value of codIRange, codIRange is mapped to the index qCodIRangeIdx of a quantised value of codIRange by using Equation 9-6. The value of qCodIRangeIdx and the value of pStateIdx associated with ctxIdx are used to determine the value of the variable rangeTabLPS as specified in Table 9-8, which is assigned to codIRangeLPS. The value of codIRange – codIRangeLPS is assigned to codIRange.

In a second step, the value of binVal is compared to valMPS associated with ctxIdx. When binVal is different from valMPS, codIRange is added to codILow and codIRange is set equal to the value codIRangeLPS. Given the encoded decision, the state transition is performed as specified in subclause 9.3.3.2.1.1. Depending on the current value of codIRange, renormalization is performed as specified in subclause 9.3.4.3. Finally, the variable BinCountsInNALunits is incremented by 1.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000723

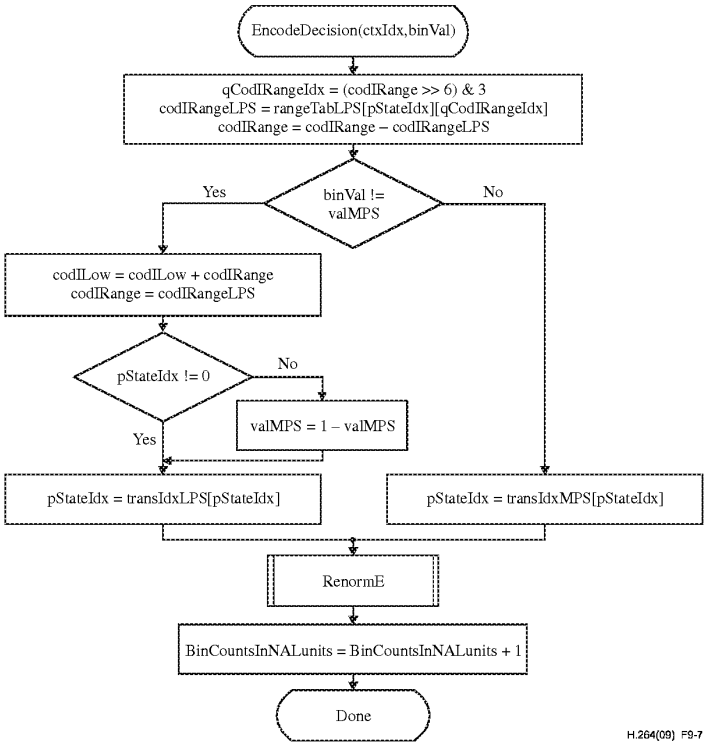


Figure 9-6 – Flowchart for encoding a decision

9.3.4.3 Renormalization process in the arithmetic encoding engine (informative)

This subclause does not form an integral part of this Recommendation | International Standard.

Inputs to this process are the variables codIRange, codILow, firstBitFlag, and bitsOutstanding.

Outputs of this process are zero or more bits written to the RBSP and the updated variables codIRange, codILow, firstBitFlag, and bitsOutstanding.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000724

Renormalization is illustrated in Figure 9-7.

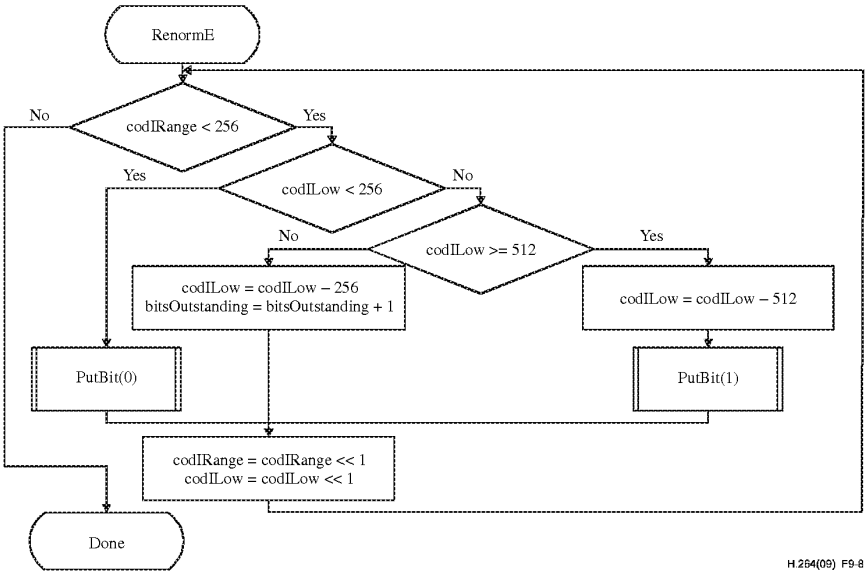


Figure 9-7 – Flowchart of renormalization in the encoder

The PutBit( ) procedure described in Figure 9-8 provides carry over control. It uses the function WriteBits(B, N) that writes N bits with value B to the bitstream and advances the bitstream pointer by N bit positions. This function assumes the existence of a bitstream pointer with an indication of the position of the next bit to be written to the bitstream by the encoding process.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000725

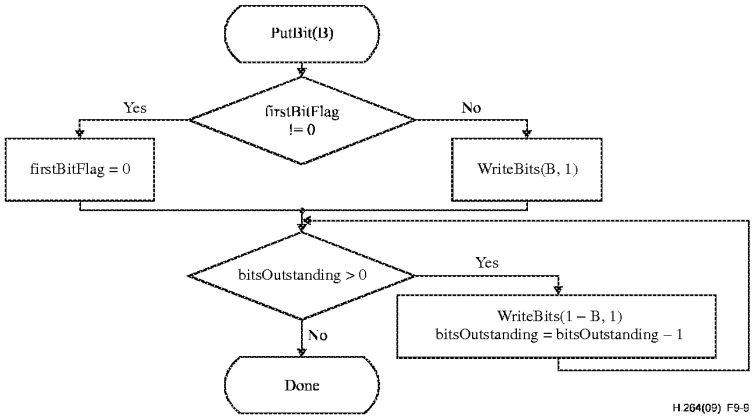


Figure 9-8 – Flowchart of PutBit(B)

9.3.4.4 Bypass encoding process for binary decisions (informative)

This subclause does not form an integral part of this Recommendation | International Standard.

Inputs to this process are the variables binVal, codILow, codIRange, bitsOutstanding, and BinCountsInNALunits.

Output of this process is a bit written to the RBSP and the updated variables codILow, bitsOutstanding, and BinCountsInNALunits.

This encoding process applies to all binary decisions with bypassFlag equal to 1. Renormalization is included in the specification of this process as given in Figure 9-9.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000726

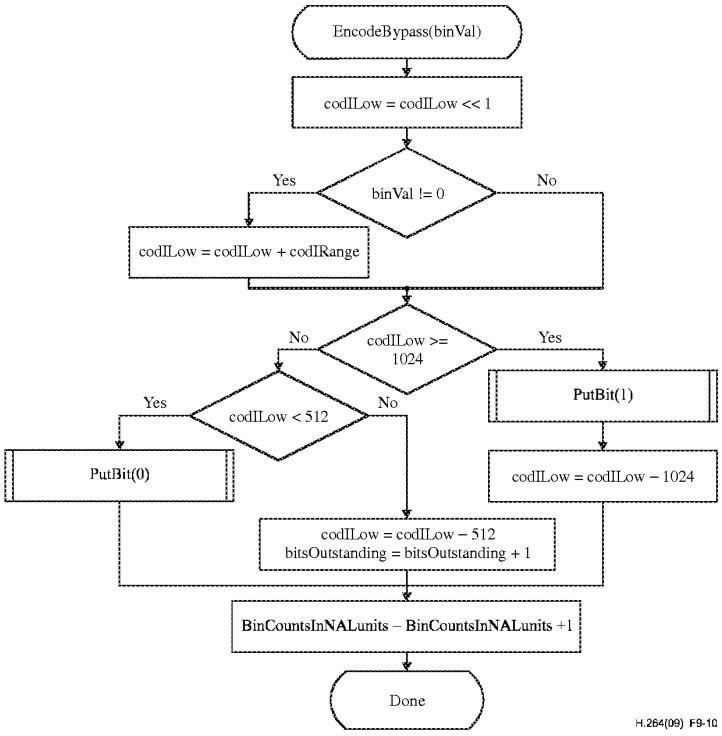


Figure 9-9 – Flowchart of encoding bypass

9.3.4.5 Encoding process for a binary decision before termination (informative)

This subclause does not form an integral part of this Recommendation | International Standard.

Inputs to this process are the variables binVal, codIRange, codILow, bitsOutstanding, and BinCountsInNALunits.

Outputs of this process are zero or more bits written to the RBSP and the updated variables codILow, codIRange, bitsOutstanding, and BinCountsInNALunits.

This encoding routine shown in Figure 9-10 applies to encoding of the end_of_slice_flag and pcm_flag indicating the I_PCM mode both associated with ctxIdx equal to 276.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000727

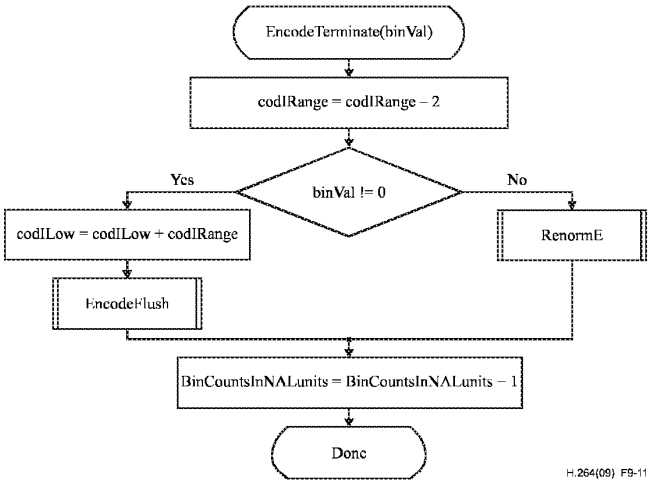


Figure 9-10 – Flowchart of encoding a decision before termination

When the value of binVal to encode is equal to 1, CABAC encoding is terminated and the flushing procedure shown in Figure 9-11 is applied. In this flushing procedure, the last bit written by WriteBits(B, N) is equal to 1. When encoding end_of_slice_flag, this last bit is interpreted as the rbsp_stop_one_bit.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000728

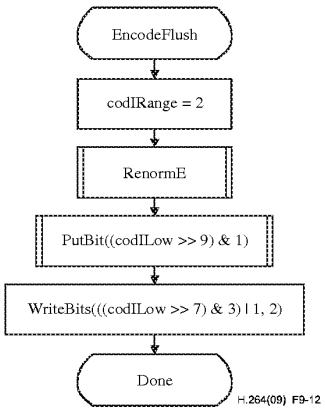


Figure 9-11 – Flowchart of flushing at termination

9.3.4.6 Byte stuffing process (informative)

This subclause does not form an integral part of this Recommendation | International Standard.

This process is invoked after encoding the last macroblock of the last slice of a picture and after encapsulation.

Inputs to this process are the number of bytes NumBytesInVclNALunits of all VCL NAL units of a picture, the number of macroblocks PicSizeInMbs in the picture, and the number of binary symbols BinCountsInNALunits resulting from encoding the contents of all VCL NAL units of the picture.

NOTE – The value of BinCountsInNALunits is the result of encoding all slices of a coded picture. After initialising for the first slice of a coded picture as specified in subclause 9.3.4.1, BinCountsInNALunits is incremented as specified in subclauses 9.3.4.2, 9.3.4.4, and 9.3.4.5.

Outputs of this process are zero or more bytes appended to the NAL unit.

Let the variable k be set equal to Ceil((Ceil(3 * (32 * BinCountsInNALunits – RawMbBits * PicSizeInMbs) ÷ 1024) – NumBytesInVclNALunits) ÷ 3). Depending on the variable k the following applies.

– If k is less than or equal to 0, no cabac_zero_word is appended to the NAL unit.

– Otherwise (k is greater than 0), the 3-byte sequence 0x000003 is appended k times to the NAL unit after encapsulation, where the first two bytes 0x0000 represent a cabac_zero_word and the third byte 0x03 represents an emulation_prevention_three_byte.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000729

Annex A

Profiles and levels

(This annex forms an integral part of this Recommendation | International Standard)

Profiles and levels specify restrictions on bitstreams and hence limits on the capabilities needed to decode the bitstreams. Profiles and levels may also be used to indicate interoperability points between individual decoder implementations.

NOTE 1 – This Recommendation | International Standard does not include individually selectable "options" at the decoder, as this would increase interoperability difficulties.

Each profile specifies a subset of algorithmic features and limits that shall be supported by all decoders conforming to that profile.

NOTE 2 – Encoders are not required to make use of any particular subset of features supported in a profile.

Each level specifies a set of limits on the values that may be taken by the syntax elements of this Recommendation | International Standard. The same set of level definitions is used with all profiles, but individual implementations may support a different level for each supported profile. For any given profile, levels generally correspond to decoder processing load and memory capability.

The profiles that are specified in subclause A.2 are also referred to as the profiles specified in Annex A.

A.1 Requirements on video decoder capability

Capabilities of video decoders conforming to this Recommendation | International Standard are specified in terms of the ability to decode video streams conforming to the constraints of profiles and levels specified in this annex. For each such profile, the level supported for that profile shall also be expressed.

Specific values are specified in this annex for the syntax elements profile_idc and level_idc. All other values of profile_idc and level_idc are reserved for future use by ITU-T | ISO/IEC.

NOTE – Decoders should not infer that when a reserved value of profile_idc or level_idc falls between the values specified in this Recommendation | International Standard that this indicates intermediate capabilities between the specified profiles or levels, as there are no restrictions on the method to be chosen by ITU-T | ISO/IEC for the use of such future reserved values.

A.2 Profiles

All constraints for picture parameter sets that are specified are constraints for picture parameter sets that are activated in the bitstream. All constraints for sequence parameter sets that are specified in subclause A.2.1 are constraints for sequence parameter sets that are activated in the bitstream.

A.2.1 ABC profile

A.3 Levels

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000730

Table A-9-55 – Level limits [Ed. (TW) Kept for convenience]

Level number	Max macroblock processing rate MaxMBPS (MB/s)	Max picture size MaxFS (MBs)	Max decoded picture buffer size MaxDpbMbs (MBs)	Max video bit rate MaxBR (1000 bits/s, 1200 bits/s, cpbBrVclFactor bits/s, or cpbBrNalFactor bits/s)	Max CPB size MaxCPB (1000 bits, 1200 bits, cpbBrVclFactor bits, or cpbBrNalFactor bits)	Vertical MV component range MaxVmvR (luma picture samples)	Min compression ratio MinCR	Max number of motion vectors per two consecutive MBs MaxMvsPer2Mb
1	1 485	99	396	64	175	[−64,+63.75]	2	-
1b	1 485	99	396	128	350	[−64,+63.75]	2	-
1.1	3 000	396	900	192	500	[−128,+127.75]	2	-
1.2	6 000	396	2 376	384	1 000	[−128,+127.75]	2	-
1.3	11 880	396	2 376	768	2 000	[−128,+127.75]	2	-
2	11 880	396	2 376	2 000	2 000	[−128,+127.75]	2	-
2.1	19 800	792	4 752	4 000	4 000	[−256,+255.75]	2	-
2.2	20 250	1 620	8 100	4 000	4 000	[−256,+255.75]	2	-
3	40 500	1 620	8 100	10 000	10 000	[−256,+255.75]	2	32
3.1	108 000	3 600	18 000	14 000	14 000	[−512,+511.75]	4	16
3.2	216 000	5 120	20 480	20 000	20 000	[−512,+511.75]	4	16
4	245 760	8 192	32 768	20 000	25 000	[−512,+511.75]	4	16
4.1	245 760	8 192	32 768	50 000	62 500	[−512,+511.75]	2	16
4.2	522 240	8 704	34 816	50 000	62 500	[−512,+511.75]	2	16
5	589 824	22 080	110 400	135 000	135 000	[−512,+511.75]	2	16
5.1	983 040	36 864	184 320	240 000	240 000	[−512,+511.75]	2	16

Levels with non-integer level numbers in Table A-9-8 are referred to as "intermediate levels".

NOTE 4 – All levels have the same status, but some applications may choose to use only the integer-numbered levels.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000731

Annex B

Byte stream format

(This annex forms an integral part of this Recommendation | International Standard)

This annex specifies syntax and semantics of a byte stream format specified for use by applications that deliver some or all of the NAL unit stream as an ordered stream of bytes or bits within which the locations of NAL unit boundaries need to be identifiable from patterns in the data, such as ITU-T Rec. H.222.0 | ISO/IEC 13818-1 systems or ITU-T Rec. H.320 systems. For bit-oriented delivery, the bit order for the byte stream format is specified to start with the MSB of the first byte, proceed to the LSB of the first byte, followed by the MSB of the second byte, etc.

The byte stream format consists of a sequence of byte stream NAL unit syntax structures. Each byte stream NAL unit syntax structure contains one start code prefix followed by one nal_unit(NumBytesInNALunit) syntax structure. It may (and under some circumstances, it shall) also contain an additional zero_byte syntax element. It may also contain one or more additional trailing_zero_8bits syntax elements. When it is the first byte stream NAL unit in the bitstream, it may also contain one or more additional leading_zero_8bits syntax elements.

B.1 Byte stream NAL unit syntax and semantics

B.1.1 Byte stream NAL unit syntax

byte_stream_nal_unit(NumBytesInNALunit) {	C	Descriptor
while(next_bits(24) != 0x000001 && next_bits(32) != 0x00000001)		
leading_zero_8bits /* equal to 0x00 */		f(8)
if(next_bits(24) != 0x000001)		
zero_byte /* equal to 0x00 */		f(8)
start_code_prefix_one_3bytes /* equal to 0x000001 */		f(24)
nal_unit(NumBytesInNALunit)		
while(more_data_in_byte_stream( ) && next_bits(24) != 0x000001 && next_bits(32) != 0x00000001)		
trailing_zero_8bits /* equal to 0x00 */		f(8)
}		

B.1.2 Byte stream NAL unit semantics

The order of byte stream NAL units in the byte stream shall follow the decoding order of the NAL units contained in the byte stream NAL units (see subclause 7.4.1.2). The content of each byte stream NAL unit is associated with the same access unit as the NAL unit contained in the byte stream NAL unit (see subclause 7.4.1.2.3).

leading_zero_8bits is a byte equal to 0x00.

NOTE – The leading_zero_8bits syntax element can only be present in the first byte stream NAL unit of the bitstream, because (as shown in the syntax diagram of subclause B.1.1) any bytes equal to 0x00 that follow a NAL unit syntax structure and precede the four-byte sequence 0x00000001 (which is to be interpreted as a zero_byte followed by a start_code_prefix_one_3bytes) will be considered to be trailing_zero_8bits syntax elements that are part of the preceding byte stream NAL unit.

zero_byte is a single byte equal to 0x00.

When any of the following conditions are fulfilled, the zero_byte syntax element shall be present:

– the nal_unit_type within the nal_unit( ) is equal to 7 (sequence parameter set) or 8 (picture parameter set),

– the byte stream NAL unit syntax structure contains the first NAL unit of an access unit in decoding order, as specified by subclause 7.4.1.2.3.

start_code_prefix_one_3bytes is a fixed-value sequence of 3 bytes equal to 0x000001. This syntax element is called a start code prefix.

trailing_zero_8bits is a byte equal to 0x00.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000732

B.2 Byte stream NAL unit decoding process

Input to this process consists of an ordered stream of bytes consisting of a sequence of byte stream NAL unit syntax structures.

Output of this process consists of a sequence of NAL unit syntax structures.

At the beginning of the decoding process, the decoder initialises its current position in the byte stream to the beginning of the byte stream. It then extracts and discards each leading_zero_8bits syntax element (if present), moving the current position in the byte stream forward one byte at a time, until the current position in the byte stream is such that the next four bytes in the bitstream form the four-byte sequence 0x00000001.

The decoder then performs the following step-wise process repeatedly to extract and decode each NAL unit syntax structure in the byte stream until the end of the byte stream has been encountered (as determined by unspecified means) and the last NAL unit in the byte stream has been decoded:

1. When the next four bytes in the bitstream form the four-byte sequence 0x00000001, the next byte in the byte stream (which is a zero_byte syntax element) is extracted and discarded and the current position in the byte stream is set equal to the position of the byte following this discarded byte.
2. The next three-byte sequence in the byte stream (which is a start_code_prefix_one_3bytes) is extracted and discarded and the current position in the byte stream is set equal to the position of the byte following this three-byte sequence.
3. NumBytesInNALunit is set equal to the number of bytes starting with the byte at the current position in the byte stream up to and including the last byte that precedes the location of any of the following conditions:
   - A subsequent byte-aligned three-byte sequence equal to 0x000000,
   - A subsequent byte-aligned three-byte sequence equal to 0x000001,
   - The end of the byte stream, as determined by unspecified means.
4. NumBytesInNALunit bytes are removed from the bitstream and the current position in the byte stream is advanced by NumBytesInNALunit bytes. This sequence of bytes is nal_unit(NumBytesInNALunit) and is decoded using the NAL unit decoding process.
5. When the current position in the byte stream is not at the end of the byte stream (as determined by unspecified means) and the next bytes in the byte stream do not start with a three-byte sequence equal to 0x000001 and the next bytes in the byte stream do not start with a four byte sequence equal to 0x00000001, the decoder extracts and discards each trailing_zero_8bits syntax element, moving the current position in the byte stream forward one byte at a time, until the current position in the byte stream is such that the next bytes in the byte stream form the four-byte sequence 0x00000001 or the end of the byte stream has been encountered (as determined by unspecified means).

B.3 Decoder byte-alignment recovery (informative)

This subclause does not form an integral part of this Recommendation | International Standard.

Many applications provide data to a decoder in a manner that is inherently byte aligned, and thus have no need for the bit-oriented byte alignment detection procedure described in this subclause.

A decoder is said to have byte-alignment with a bitstream when the decoder is able to determine whether or not the positions of data in the bitstream are byte-aligned. When a decoder does not have byte alignment with the encoder's byte stream, the decoder may examine the incoming bitstream for the binary pattern '00000000 00000000 00000000 00000001' (31 consecutive bits equal to 0 followed by a bit equal to 1). The bit immediately following this pattern is the first bit of an aligned byte following a start code prefix. Upon detecting this pattern, the decoder will be byte aligned with the encoder and positioned at the start of a NAL unit in the byte stream.

Once byte aligned with the encoder, the decoder can examine the incoming byte stream for subsequent three-byte sequences 0x000001 and 0x000003.

When the three-byte sequence 0x000001 is detected, this is a start code prefix.

When the three-byte sequence 0x000003 is detected, the third byte (0x03) is an emulation_prevention_three_byte to be discarded as specified in subclause 7.4.1.

When an error in the bitstream syntax is detected (e.g., a non-zero value of the forbidden_zero_bit or one of the three-byte or four-byte sequences that are prohibited in subclause 7.4.1), the decoder may consider the detected condition as an indication that byte alignment may have been lost and may discard all bitstream data until the detection of byte alignment at a later position in the bitstream as described in this subclause.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000733

Annex C

Hypothetical reference decoder

(This annex forms an integral part of this Recommendation | International Standard)

This annex specifies the hypothetical reference decoder (HRD) and its use to check bitstream and decoder conformance.

Two types of bitstreams are subject to HRD conformance checking for this Recommendation | International Standard. The first such type of bitstream, called Type I bitstream, is a NAL unit stream containing only the VCL NAL units and filler data NAL units for all access units in the bitstream. The second type of bitstream, called a Type II bitstream, contains, in addition to the VCL NAL units and filler data NAL units for all access units in the bitstream, at least one of the following:

– additional non-VCL NAL units other than filler data NAL units,

– all leading_zero_8bits, zero_byte, start_code_prefix_one_3bytes, and trailing_zero_8bits syntax elements that form a byte stream from the NAL unit stream (as specified in Annex B).

Figure C-9-12 shows the types of bitstream conformance points checked by the HRD.

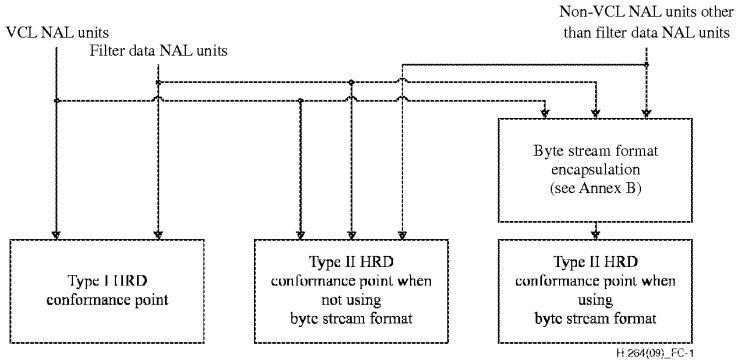


Figure C-9-12 – Structure of byte streams and NAL unit streams for HRD conformance checks

The syntax elements of non-VCL NAL units (or their default values for some of the syntax elements), required for the HRD, are specified in the semantic subclauses of clause 7, Annexes D and E.

The HRD contains a coded picture buffer (CPB), an instantaneous decoding process, a decoded picture buffer (DPB), and output cropping as shown in Figure C-9-13.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000734

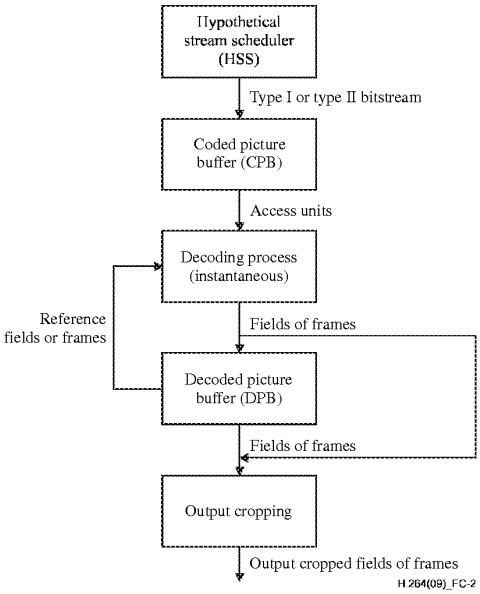


Figure C-9-13 – HRD buffer model

C.1 Operation of coded picture buffer (CPB)

The specifications in this subclause apply independently to each set of CPB parameters that is present and to both the Type I and Type II conformance points shown in Figure C-9-12.

C.1.1 Timing of bitstream arrival

C.1.2 Timing of coded picture removal

C.2 Operation of the decoded picture buffer (DPB)

C.3 Bitstream conformance

C.4 Decoder conformance

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000735

Annex D

Supplemental enhancement information

(This annex forms an integral part of this Recommendation | International Standard)

This annex specifies syntax and semantics for SEI message payloads.

SEI messages assist in processes related to decoding, display or other purposes. However, SEI messages are not required for constructing the luma or chroma samples by the decoding process. Conforming decoders are not required to process this information for output order conformance to this Recommendation | International Standard (see Annex 0 for the specification of conformance). Some SEI message information is required to check bitstream conformance and for output timing decoder conformance.

In Annex D, specification for presence of SEI messages are also satisfied when those messages (or some subset of them) are conveyed to decoders (or to the HRD) by other means not specified by this Recommendation | International Standard. When present in the bitstream, SEI messages shall obey the syntax and semantics specified in subclauses 7.3.2.3 and 0 and this annex. When the content of an SEI message is conveyed for the application by some means other than presence within the bitstream, the representation of the content of the SEI message is not required to use the same syntax specified in this annex. For the purpose of counting bits, only the appropriate bits that are actually present in the bitstream are counted.

D.1 SEI payload syntax

[Ed. (TW): insert table]

D.2 SEI payload semantics

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000736

Annex E

Video usability information

(This annex forms an integral part of this Recommendation | International Standard)

This annex specifies syntax and semantics of the VUI parameters of the sequence parameter sets.

VUI parameters are not required for constructing the luma or chroma samples by the decoding process. Conforming decoders are not required to process this information for output order conformance to this Recommendation | International Standard (see Annex 0 for the specification of conformance). Some VUI parameters are required to check bitstream conformance and for output timing decoder conformance.

In Annex E, specification for presence of VUI parameters is also satisfied when those parameters (or some subset of them) are conveyed to decoders (or to the HRD) by other means not specified by this Recommendation | International Standard. When present in the bitstream, VUI parameters shall follow the syntax and semantics specified in subclauses 7.3.2.1 and 7.4.2.1 and this annex. When the content of VUI parameters is conveyed for the application by some means other than presence within the bitstream, the representation of the content of the VUI parameters is not required to use the same syntax specified in this annex. For the purpose of counting bits, only the appropriate bits that are actually present in the bitstream are counted.

E.1 VUI syntax

E.1.1 VUI parameters syntax

[Ed. (TW): insert table]

E.2 VUI semantics

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000737

Joint Collaborative Team on Video Coding (JCT-VC)
of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11
6th Meeting: Torino, IT, 14-22 July, 2011

Document: JCTVC-F052

Title: **CE9: Results of Experiment ROB04**

Status: Input Document to JCT-VC

Purpose: Proposal

Author(s) or Contact(s): Jian-Liang Lin, Yi-Wen Chen, Yu-Wen Huang, and Shawmin Lei
No. 1, Dusing Rd. 1, Hsinchu Science Park, Hsinchu, Taiwan 30078

Shawmin Lei
Tel: +886-3-5670766 ext. 25555
Email: {jl.lin, yiwen.chen, yuwen.huang, shawmin.lei}@mediatek.com

Source: MediaTek Inc.

Abstract

This contribution reported the performance results of CE9 ROB04 experiment on parsing robustness. In this experiment, a basic scheme with parsing robustness by skipping the process of removing redundant motion vector predictors (MVPs) and forcing the availability of the temporal MVP was firstly provided. The experiment results reported that this basic scheme resulted in average 1.3%, 1.3%, 1.9%, and 2.0% bit rate increases for high HE-RA, LC-RA, HE-LD, and LC-LD, respectively. To reduce the bit rate increases, two approaches were tested in ROB04. The first approach was replacing the redundant MVPs by other substitutes, and the second approach was reducing redundant MVPs without comparing motion vector values. It was reported that combing both approaches caused 0.3%, 0.4%, 0.3%, and 0.5% bit rate increases for HE-RA, LC-RA, HE-LD, and LC-LD, respectively, while providing the ability to prevent all parsing errors caused by any data loss of reference pictures.

1 Introduction

The ROB04 experiment was approved at the 5th JCT-VC meeting in CE9 [1]. The ROB04 experiment is to evaluate the coding efficiency of the parsing robustness solution for MVPs. In HM-3.0 [2] and WD-3.0 [3], to reduce the bits used to encode the index of motion vector predictor (MVP), redundant MVPs are removed from the MVP candidate set. However, this can cause parsing errors leading to severe error propagation. When a motion vector (MV) of a previous picture cannot be decoded correctly, a mismatch between the candidate set on the encoder side and that on the decoder side may occur, resulting in parsing error of the index of the best MVP. Then, the rest of the current picture cannot be parsed or decoded properly. What is even worse, this parsing error can affect subsequent inter pictures that also use temporal MVPs. ROB04 was proposed to achieve the parsing robustness. The evaluation was based on the HM-3.0 with a fix of bug #146. The rest of this contribution is organized as follows. The experiment of ROB04 will be described in Section 2, simulation results will be shown in Section 3, and finally, conclusions will be given in Section 4.

2 Experiment description

To achieve parsing robustness, two constraints are applied for the basic approach of parsing robustness:

a. Avoid the process of removing redundant MVPs
b. Force the availability of the temporal MVP as long as it is inside the picture boundaries

In order to avoid the parsing problem caused by incorrect MV data, the process of removing redundant MVPs is avoided. Moreover, the temporal MVP is forced to be available as long as the temporal block is

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000738

inside the boundaries of that co-located picture. Even if the temporal block in the co-located picture is an Intra block, the parsing robustness system still force the temporal MVP to be available to prevent the case that the mode type (Inter or Intra) of the block in the co-located picture is lost or wrong. When the temporal MVP is not available, a zero MV is simply used to represent the temporal MVP. Two tools are applied to improve the coding efficiency of the parsing robustness solution. The first tool replaces redundant MVPs by other substitutes, and the second tool reduces redundant MVPs without comparing MV values.

2.1 Tool 1 – Replacing redundant MVPs

Since the process of removing redundant MVPs is avoided for parsing robustness, the redundant MVPs are replaced by other substitutes to reduce the redundancy.

In HM-3.0, the top MVP in Inter mode is defined as the first available and non-redundant one searching within B_0, B_1, B_2 as shown in Figure 1 based on the following order:

- B_0' MV with the same list and the same reference picture
- B_1' MV with the same list and the same reference picture
- B_2' MV with the same list and the same reference picture
- B_0' MV with different list or/and different reference picture
- B_1' MV with different list or/and different reference picture
- B_2' MV with different list or/and different reference picture

Please note that "the same list," "the same reference picture," "different list," and "different reference picture" mean the "the same list as the current list," "the same reference picture as the current picture," "different list from the current list," and "different reference picture from the current reference picture." Moreover, the description "MV with different list or/and different reference picture" actually means three sequential items with the order as: "MV with different list and the same reference picture," "MV with the same list and different reference picture," and "MV with different list and different reference picture."

In our approach, the top MVP is only searched within

- B_0' MV with the same list and the same reference picture
- B_1' MV with the same list and the same reference picture

Therefore, the top MVP exists only when B_0 or B_1 has a MV with the same list and the same reference picture. If the top MVP exists and all the possible MVPs are identical to the left MVP, the substitute is searched based on the following order:

- B_2' MV with the same list and the same reference picture
- B_0' MV with different list or/and different reference picture
- B_1' MV with different list or/and different reference picture
- B_2' MV with different list or/and different reference picture

After all these substitutes are searched and the top MVP is still redundant, the left MVP plus 1/4 pixel in both horizontal and vertical directions is used.

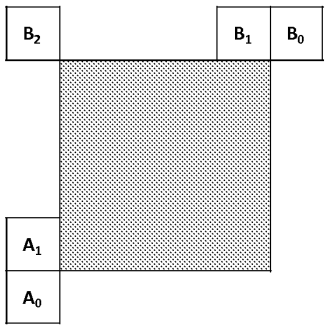


Figure 1. Spatial blocks.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000739

For the spatial MVPs in Merge/Merge-Skip modes, if one spatial MVP exists and is identical to one of the previously derived MVPs, we simply use a zero MV to replace the redundant MVP.

For the temporal MVP in Inter and Merge/Merge-Skip modes, as an example shown in Figure 2 where the gray area denotes the current prediction unit (PU), if the temporal MVP derived from H is redundant, a substitute is searched within T_c, T_o, F, and G until the first non-redundant one is found. If after these blocks are searched and the temporal MVP is still redundant, the zero MV is used.

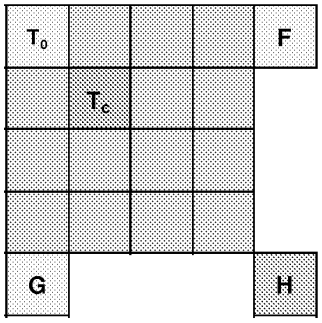


Figure 2. Temporal blocks.

2.2 Tool 2 – Reducing redundant MVPs without comparing MV values

Tool 2 tries to reduce redundant MVPs without using MV values before finding other substitutes for the redundancy reduction. In tool 2, motion identification (ID) is proposed to describe the inheritance process of motion information passed from a coded PU to the current PU to follow the trail of the MV. A unique motion ID is assigned for each new MV created in the Inter mode and can be inherited through the Merge and Merge-Skip modes. By comparing the motion ID, the MVP having the same motion ID as a previous MVP in the candidate set can thus be removed.

3 Simulation results

Several experiments are conducted to evaluate the coding efficiency and parsing robustness. HM-3.0 software [2] is used for the evaluation and the common test conditions specified in JCTVC-E700 [4] are followed. The anchor is HM-3.0 with the fix of bug #146 as described in CE9 [1].

Since the parsing problem is caused by including the temporal MVP in the candidate set, in experiment 1, we simply disable the temporal MVP for all pictures to avoid the parsing problem. The results are shown in Table 1. It shows that disabling temporal MVPs causes 2.3%, 2.2%, 2.4%, and 2.4% bit rate increases for high efficiency random access (HE-RA), low complexity random access (LC-RA), high efficiency low delay (HE-LD), and low complexity low delay (LC-LD), respectively.

In experiment 2, the basic approach for parsing robustness is implemented to avoid the parsing error by skipping the process of removing redundant MVPs and forcing the availability of the temporal MVP. The results are shown in Table 2. The basic approach for parsing robustness causes 1.3%, 1.3%, 1.9%, and 2.0% bit rate increases for HE-RA, LC-RA, HE-LD, and LC-LD while providing the ability to prevent parsing errors caused by data loss of reference pictures.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000740

Table 1. Results of experiment 1: disabling temporal MVPs for all pictures.

	HE-RA			LC-RA		
	Y BD-rate	U BD-rate	V BD-rate	Y BD-rate	U BD-rate	V BD-rate
Class A	3.1	2.6	2.8	3.3	2.8	2.8
Class B	2.0	1.4	1.4	1.7	1.1	1.1
Class C	2.1	2.1	2.2	2.0	2.0	2.1
Class D	2.1	2.0	2.0	2.0	1.8	1.8
Class E						
All	**2.3**	**2.0**	**2.0**	**2.2**	**1.9**	**1.9**
Enc Time[%]	92%			93%		
Dec Time[%]	97%			96%		

	HE-LD			LC-LD		
	Y BD-rate	U BD-rate	V BD-rate	Y BD-rate	U BD-rate	V BD-rate
Class A						
Class B	2.1	1.8	1.2	2.3	1.6	1.5
Class C	2.5	2.6	2.7	2.6	2.4	2.4
Class D	2.2	1.8	2.1	2.2	1.2	1.3
Class E	2.9	2.7	2.1	2.6	1.4	2.0
All	**2.4**	**2.2**	**2.0**	**2.4**	**1.7**	**1.8**
Enc Time[%]	93%			93%		
Dec Time[%]	98%			94%		

Table 2. Results of experiment 2: basic approach – skipping the process of removing redundant MVPs and forcing the availability of the temporal MVP.

	HE-RA			LC-RA		
	Y BD-rate	U BD-rate	V BD-rate	Y BD-rate	U BD-rate	V BD-rate
Class A	1.0	0.9	1.0	1.2	1.0	1.0
Class B	1.3	0.8	0.8	1.4	1.0	0.9
Class C	1.4	1.3	1.3	1.4	1.3	1.4
Class D	1.3	1.1	1.1	1.3	0.9	1.1
Class E						
All	**1.3**	**1.0**	**1.0**	**1.3**	**1.0**	**1.1**
Enc Time[%]	108%			107%		
Dec Time[%]	100%			97%		

	HE-LD			LC-LD		
	Y BD-rate	U BD-rate	V BD-rate	Y BD-rate	U BD-rate	V BD-rate
Class A						
Class B	1.6	1.2	0.7	1.7	1.2	1.2
Class C	1.5	1.3	1.2	1.4	1.0	1.2
Class D	1.5	1.0	1.6	1.4	1.0	0.9
Class E	3.6	3.4	2.7	4.1	2.9	3.5
All	**1.9**	**1.6**	**1.4**	**2.0**	**1.4**	**1.6**
Enc Time[%]	107%			106%		
Dec Time[%]	98%			99%		

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000741

In experiment 3, the tool 1 described in section 2.1 is implemented to find other substitutes for redundant MVPs. According to the results shown in Table 3, tool 1 causes 0.9%, 0.9%, 1.4%, and 1.3% bit rate increases for HE-RA, LC-RA, HE-LD, and LC-LD, respectively.

In experiment 4, the tool 2 described in Section 2.2 is implemented to reduce redundant MVPs without using MV values. According to the results shown in Table 4, Tool 2 causes 0.5%, 0.7%, 0.6%, and 0.9% bit rate increases for HE-RA, LC-RA, HE-LD, and LC-LD, respectively.

In experiment 5, we combine tool 1 and tool 2. The motion ID is firstly applied to reduce redundant MVPs. For the remaining redundant MVPs that cannot be removed by using motion IDs, tool 1 is then applied to find the replacement. According to the results shown in Table 5, experiment 5 achieves 0.3%, 0.4%, 0.3%, and 0.5% bit rate increases for HE-RA, LC-RA, HE-LD, and LC-LD, respectively, while providing the ability of parsing robustness.

Table 3. Results of experiment 3: proposed tool 1 – replacing redundant MVPs by the substitutes described in Section 2.1.

	HE-RA			LC-RA		
	Y BD-rate	U BD-rate	V BD-rate	Y BD-rate	U BD-rate	V BD-rate
Class A	0.7	0.9	1.1	0.7	0.8	0.8
Class B	0.9	0.6	0.5	0.9	0.7	0.7
Class C	1.0	1.0	1.0	1.0	0.9	1.0
Class D	0.9	0.8	0.9	0.8	0.6	0.8
Class E						
All	**0.9**	**0.8**	**0.8**	**0.9**	**0.8**	**0.8**
Enc Time[%]	109%			108%		
Dec Time[%]	98%			97%		

	HE-LD			LC-LD		
	Y BD-rate	U BD-rate	V BD-rate	Y BD-rate	U BD-rate	V BD-rate
Class A						
Class B	1.1	0.6	0.4	1.0	0.6	0.7
Class C	1.0	0.6	0.7	0.8	0.6	0.5
Class D	1.0	0.8	0.9	0.9	0.5	0.5
Class E	3.0	2.8	3.1	3.0	2.4	3.2
All	**1.4**	**1.1**	**1.1**	**1.3**	**0.9**	**1.1**
Enc Time[%]	108%			107%		
Dec Time[%]	99%			105%		

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000742

Table 4. Results of experiment 4: proposed tool 2 – reducing redundant MVPs by comparing motion IDs instead of MV values as described in Section 2.2.

	HE-RA			LC-RA		
	Y BD-rate	U BD-rate	V BD-rate	Y BD-rate	U BD-rate	V BD-rate
Class A	0.6	0.9	0.8	0.7	0.8	0.8
Class B	0.5	0.3	0.3	0.6	0.5	0.5
Class C	0.6	0.6	0.7	0.9	0.9	1.1
Class D	0.5	0.5	0.5	0.6	0.5	0.6
Class E						
All	**0.5**	**0.6**	**0.6**	**0.7**	**0.7**	**0.7**
Enc Time[%]	101%			102%		
Dec Time[%]	100%			102%		

	HE-LD			LC-LD		
	Y BD-rate	U BD-rate	V BD-rate	Y BD-rate	U BD-rate	V BD-rate
Class A						
Class B	0.5	0.3	0.1	0.7	0.4	0.1
Class C	0.6	0.5	0.5	0.7	0.7	0.5
Class D	0.4	0.0	0.4	0.6	0.2	0.0
Class E	1.3	1.2	0.4	1.8	1.1	1.6
All	**0.6**	**0.5**	**0.3**	**0.9**	**0.5**	**0.5**
Enc Time[%]	102%			104%		
Dec Time[%]	100%			101%		

Table 5. Results of experiment 5: proposed tool 1 and tool 2 – first reducing redundant MVPs by comparing motion IDs instead of MV values as described in Section 2.2 and then replacing redundant MVPs by the substitutes described in Section 2.1.

	HE-RA			LC-RA		
	Y BD-rate	U BD-rate	V BD-rate	Y BD-rate	U BD-rate	V BD-rate
Class A	0.3	0.6	0.6	0.4	0.6	0.6
Class B	0.2	0.2	0.2	0.3	0.3	0.3
Class C	0.4	0.4	0.6	0.6	0.7	0.7
Class D	0.4	0.4	0.3	0.4	0.4	0.3
Class E						
All	**0.3**	**0.4**	**0.4**	**0.4**	**0.5**	**0.5**
Enc Time[%]	102%			102%		
Dec Time[%]	101%			102%		

	HE-LD			LC-LD		
	Y BD-rate	U BD-rate	V BD-rate	Y BD-rate	U BD-rate	V BD-rate
Class A						
Class B	0.2	-0.1	-0.2	0.3	0.1	0.1
Class C	0.3	0.1	0.1	0.4	0.4	0.3
Class D	0.2	-0.1	0.8	0.3	0.3	-0.2
Class E	0.8	1.0	-0.2	1.3	1.2	1.2
All	**0.3**	**0.2**	**0.1**	**0.5**	**0.4**	**0.3**
Enc Time[%]	103%			104%		
Dec Time[%]	102%			102%		

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000743

4 Conclusions

ROB04 was tested in CE9. Experiment 2 could solve the parsing problem caused by data loss of reference pictures and two tools are proposed to improve the coding efficiency of the parsing robustness solution. The first tool is replacing redundant MVPs by substitutes, and the second tool is reducing redundant MVPs by comparing motion IDs instead of MV values. The experimental results reportedly showed that the parsing robustness solution combining the proposed two tools caused 0.3%, 0.4%, 0.3%, and 0.5% bit rate increases for HE-RA, LC-RA, HE-LD, and LC-LD, respectively, while providing the ability to prevent all parsing errors caused by any data loss of reference pictures.

5 Patent rights declaration(s)

MediaTek Inc. may have current or pending patent rights relating to the technology described in this contribution and, conditioned on reciprocity, is prepared to grant licenses under reasonable and non-discriminatory terms as necessary for implementation of the resulting ITU-T Recommendation | ISO/IEC International Standard (per box 2 of the ITU-T/ITU-R/ISO/IEC patent statement and licensing declaration form).

6 References

[1] Y.-W. Huang, B. Bross, M. Zhou, W.-J. Chien, I.-K. Kim, "Description of Core Experiment 9 (CE9): MV Coding and Skip/Merge Operations," Document of Joint Collaborative Team on Video Coding, JCTVC-E709, Mar. 2011.

[2] K. McCann, S. Sekiguci, B. Bross, W.-J. Han, "HEVC Test Model 3 (HM 3) Encoder Description," Document of Joint Collaborative Team on Video Coding, JCTVC-E602, Mar. 2011.

[3] T. Wiegand, B. Bross, W.-J. Han, J.-R. Ohm, G. J. Sullivan, "WD3: Working Draft 3 of High-Efficiency Video Coding," Document of Joint Collaborative Team on Video Coding, JCTVC-E603, Mar. 2011.

[4] F. Bossen, "Common test conditions and software reference configurations," Document of Joint Collaborative Team on Video Coding, JCTVC-E700, Mar. 2011.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000744

OA Office Action, Due 21.10.2015, "OA Date 21.08.2015"
Korean Patent Appln. No. 2014-7015093

• Type of Action:	Notice of Preliminary Rejection
• Due Date:	October 21, 2015
• Pending Claim(s):	Claims 1 to 22
• Rejected Claim(s):	Claims 1, 2, 5 to 7, 10 to 12, 15 to 17 and 20 to 22
• Allowable Claim(s):	Claims 3, 4, 8, 9, 13, 14, 18 and 19
• Rejection Ground(s):	(i) Lack of inventiveness [Claims 1, 2, 5 to 7, 10 to 12, 15 to 17 and 20 to 22] (ii) Failure to meet the description requirement for the claims [Claims 5 and 10]
• Cited reference(s):	T. Wiegand et al., "WD3: Working Draft 3 of High Efficiency Video Coding", 5th Meeting: Geneva, CH, 16-23 March, 2011

Summary of the rejection reasons

1. The Examiner rejected claims 1, 2, 5 to 7, 10 to 12, 15 to 17 and 20 to 22 under Article 29(2) of the Korean Patent Law, for the following reasons:

(1) As for claim 1

Claim 1, which is directed to a method of encoding a video image, recites, inter alia:
[…]
selecting a first spatial motion vector prediction candidate from the set of spatial

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000745

motion vector prediction candidates as a potential spatial motion vector prediction candidate to be included in a merge list for the prediction unit ("feature a");

determining a subset of spatial motion vector predictions based on the location of the block associated with the first spatial motion vector prediction candidate ("feature b");

[…]

if at least one of the comparisons indicates that the motion vector information of the spatial motion vector prediction candidates correspond with each other, excluding the first spatial motion vector prediction candidate from the merge list ("feature c").

In this connection, cited reference 1 provides teachings regarding, e.g., deriving merging candidates from neighboring prediction unit partitions and constructing a merging candidate list by using flags of neighboring blocks (corresponding to feature a), using the positions of spatial neighbors A, B, C, and D relative to a current prediction unit (corresponding to feature b), and determining whether to include the candidate of the neighboring prediction unit by using available flags (corresponding to feature c).

Although cited reference 1 does not expressively disclose a portion of feature a, i.e., "[the] subset of spatial motion vector," such portion is inherent in the aforementioned teachings of cited reference 1.

Accordingly, claim 1 is obvious to those skilled in the art over the cited reference.

(2) As for claims 2 and 5

Claims 2 and 5 recite features regarding "at most one other spatial motion vector prediction candidate" and "selecting one motion vector prediction candidate," respectively. However, such features can be easily conceived from those skilled in the art from cited reference 1 which discloses that a motion vector is selected among motion vectors of candidate blocks which are collocated with neighboring blocks.

(3) As for claims 6, 11, 16, 21 and 22

Claims 6, 11, 16, 21 and 22 are substantially identical to claim 1, except their categories, and, therefore, are rejected for the same reason as set forth above with respect to claim 1.

(4) As for claims 7, 12 and 17

Claims 7, 12 and 17 are substantially identical to claim 2, except their categories, and, therefore, are rejected for the same reason as set forth above with respect to claim 2.

(5) As for claims 10, 15 and 20

Claims 10, 15 and 20 are substantially identical to claim 5, except their categories, and, therefore, are rejected for the same reason as set forth above with respect to claim 5.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000746

2. The Examiner rejected claims 5 and 10 under Article 42(4)(ii) of the Korean Patent Law, for the following reason:

According to the Examiner's interpretation of the expression "The method […] to represent …" recited claims 5 and 10, these claims are unclear since their subject-matters are considered different from those of their base claims.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000747

발송번호: 9-5-2015-056594865
발송일자: 2015.08.21.
제출기일: 2015.10.21.

수신 서울특별시 서초구 마방로 60 (양재동, 트러스트 타워)(제일특허법인)
제일특허법인[이혜원]

06775

YOUR INVENTION PARTNER

특 허 청

의견제출통지서

출원인	성명	노키아 코포레이션 (출원인코드: 520000368439)	
	주소	핀랜드 02610 에스푸 카라카리 7	
대리인	명칭	제일특허법인	
	주소	서울특별시 서초구 마방로 60 (양재동, 트러스트 타워)(제일특허법인)	
	지정된변리사	이혜원 외 2명	
발명자	성명	비치 메메트 오구즈	
	주소	핀랜드 에프아이-33500 탐페르 탐멜란 푸이스토카투 1-3 디 46	
발명자	성명	라이네마 자니	
	주소	핀랜드 에프아이-33230 탐페르 키사켄탄카투 12 비 6	
발명자	성명	우구르 케말	
	주소	핀랜드 에프아이-33100 탐페르 라핀티에 6디 25	

출원번호	10-2014-7015093
출원일자	2014.06.03.
발명의 명칭	비디오 인코딩 방법 및 장치

1. 이 출원에 대한 심사결과 다음과 같은 거절이유가 있어 특허법 제63조에 따라 이를 통지하오니 의견이 있거나 보정이 필요할 경우에는 상기 제출기일(2015.10.21.)까지 의견(답변, 소명)서[특허법시행규칙 별지 제24호서식] 또는/및 보정서[특허법시행규칙 별지 제9호서식]를 제출하여 주시기 바랍니다.

2. 상기 제출기일(2015.10.21.)을 연장하려는 경우에는 지정기간연장신청을 통해 그 제출기일을 4개월까지 연장할 수 있습니다. 이 경우 연장신청은 1개월 단위로 해야 하며, 필요 시 4개월을 초과하지 않는 범위에서 2개월 이상을 일괄하여 연장신청할 수 있습니다. 불가피한 사유의 발생(하단의 안내참조)으로 4개월을 초과하여 지정기간을 연장받고자 하는 때에는 그 사유를 기재한 소명서를 추가로 첨부해서 연장신청을 해야 합니다.

[심사결과]

□ 심사 대상 청구항 : 제1-22항

□ 이 출원의 거절이유가 있는 부분과 관련 법조항

순번	거절이유가 있는 부분	관련 법조항
1	청구항 제1항, 제2항, 제5항, 제6항, 제7항, 제10항, 제11항, 제12항, 제15항, 제16항, 제17항, 제20항 내지 제22항	특허법 제29조제2항
2	청구항 제5항, 제10항	특허법 제42조제4항제2호

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000748

□ 특허 가능한 청구항 : 제3-4, 8-9, 13-14, 18-19항

※ 위 특허 가능한 청구항은 의견제출통지시점에서의 심사의견이며 추후 변경될 수 있습니다. 이 출원이 특허결정을 받기 위해서는 이 출원에 대해 지적된 거절이유가 모두 해소되어야 합니다.

[구체적인 거절이유]

1. 이 출원의 특허청구범위의 청구항 제1항, 제2항, 제5항, 제6항, 제7항, 제10항, 제11항, 제12항, 제15항, 제16항, 제17항, 제20항 내지 제22항에 기재된 발명은 아래와 같이 그 출원 전에 이 발명이 속하는 기술분야에서 통상의 지식을 가진 자가 용이하게 발명할 수 있는 것이므로 특허법 제29조제2항에 따라 특허를 받을 수 없습니다.

- 아 래 -

본원 발명과 인용발명1을 비교하면 인용발명1(T. Wiegand et al., "WD3: Working Draft 3 of High Efficiency Video Coding", 5th Meeting: Geneva, CH, 16-23 March, 2011, 공개일:2011년06월27일, 첨부1 참조, 이하 '인용발명1'이라고 함)과 발명의 목적에서 비교해 보면 본원 발명은 "공간적 움직임 벡터 예측 후보들을 이용하여 병합리스트를 생성하는 것"으로 인용발명1의 "현재 블록과 이웃하는 블록들의 움직임 벡터를 머지 리스트로 생성하는 것"과 발명의 목적에서 동일하며 효과면에 현저성이 없습니다.

1-1. 독립 청구항 제1항을 인용발명1과 비교해 보면, 주요 특징부에서 청구항 제1항은 "a)제1공간적 움직임 벡터 예측 후보를 예측 유닛에 대한 병합리스트에 포함될 잠재적인 공간적 움직임 벡터 예측 후보로서 선택, b)연관된 블록의 우치를 기초로하여 공간적 움직임 벡터 예측들의 서브세트를 결정, c)공간적 움직임 벡터 예측 후보들의 움직임 벡터 정보가 서로 부합한다는 표시하는 경우 병합리스트에서 제외"가 기재되어 있고, 인용발명1에는 "A)주변의 예측 유닛 분할로부터 병합 후보들을 생성과 주변 블록의 플래그 정보를 이용해서 병합후보리스트를 생성, B)현재 블록의 주변위치인 A,B,C,D를 이용하여 , C)이용가능플래그를 이용해서 주변 예측 유닛의 후보를 포함할지 여부를 결정"이 기재되어 있어 각각 대응됩니다. 다만 차이점이 있다면, 청구항 제1항에서는 "움직임 벡터 예측의 서브세트"를 기재하고 있고, 인용발명1에서는 단지 현재 블록에 인접한 후보들의 위치만 기재하고 있어 차이가 있으나, 후보들의 위치에 따라서 서브세트를 당연히 구성할 수 있는 것이므로 양자는 통상의 기술자 수준에서 단순한 표현상의 차이일 뿐입니다. 따라서 청구항 제1항은 인용발명1로 용이하게 발명할 수 있습니다.

1-2. 종속 청구항 제2, 5항과 인용발명1을 비교해 보면, 청구항들의 특징부는 "최대한 하나의 다른 공간적 움직임 벡터 예측 후보, 하나의 움직임 벡터 예측 후보를 선택"이 기재되어 있고, 인용발명1에도 "콜로케이트된 후보 블록의 움직임 벡터, 주변 블록들과 콜로케이트된 후보 중 움직임 벡터 선택"이라는 기재가 있어 차이가 있으나 양자는 하다는 점들으로 보아 실질적으로 동일한 특징부에 해당됩니다. 따라서 청구항 제2,5항은 인용발명1로 용이하게 발명할 수 있습니다.

1-3. 청구항 제6,7,10항과 인용발명1을 비교해 보면, 상기 청구항들의 특징부는 디코딩하는

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000749

방법 청구항으로 인코딩하는 방법 청구항인 제1,2,5항의 특징부와 실질적으로 동일하며 단지 그 발명의 청구대상만 달리하여 청구한 것에 불과하므로 인용발명1로 용이하게 발명할 수 있습니다.

1-4. 청구항 제11,12,15항과 제16,17,20항과 인용발명1을 비교해 보면, 상기 청구항들의 특징부는 장치 청구항으로 방법 청구항인 제1,2,5항과 제6,7,10항의 특징부와 실질적으로 동일하며 단지 그 발명의 청구대상만 달리하여 청구한 것에 불과하므로 인용발명1로 용이하게 발명할 수 있습니다.

1-5. 청구항 제21,22항과 인용발명1을 비교해 보면, 상기 청구항들의 특징부는 기록 매체 청구항으로 방법 청구항인 제1항과 제6항의 특징부와 실질적으로 동일하며 단지 그 발명의 청구대상만 달리하여 청구한 것에 불과하므로 인용발명1로 용이하게 발명할 수 있습니다.

2. 이 출원은 특허청구범위의 청구항 제5항, 제10항의 기재가 아래에 지적한 바와 같이 불비하여 특허법 제42조제4항제2호에 따른 요건을 충족하지 못하므로 특허를 받을 수 없습니다.

- 아 래 -

2-1. 청구항 제5,10항은 “~나타내는 방법”으로 청구항 말미를 기재하고 있으나 상기 청구항이 인용하는 항에는 포함하는 방법으로 기재되어 있어 청구대상이 불분명하므로 상기 청구항들의 발명이 간결하고 명확하게 기재되어 있다고 볼 수 없습니다.

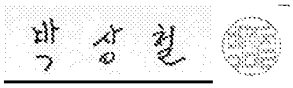
[첨 부]

첨부1 T. Wiegand et al., "WD3: Working Draft 3 of High Efficiency Video Coding", 5th Meeting: Geneva, CH, 16-23 March, 2011 끝.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000750

2015.08.21.

특허청 특허심사3국

멀티미디어방송심사팀 심사관 박상철 

<< 안내 >>

1. 특허법 제47조제2항에 따라 특허출원서에 최초로 첨부된 명세서 또는 도면에 기재된 사항의 범위 안에서 명세서 또는 도면을 보정할 수 있음을 알려드립니다. (참고 : 최후거절이유통지 후 및 특허거절결정에 대한 재심사 청구시의 보정에 대해서는 특허법 제47조제3항 규정도 적용됨을 알려드립니다.)

2. 보정료 납부안내 o 명세서 또는 도면을 보정하기 위하여 명세서등 보정서를 전자문서로 제출할 경우 매건 4,000원, 서면으로 제출할 경우 매건 14,000원의 보정료를 납부하여야 합니다. o 보정료는 접수번호를 부여받아 이를 납부자번호로 "특허료등의 징수규칙" 별지 제1호서식에 기재하여, 접수번호를 부여받은 날의 다음 날까지 납부하여야 합니다. 다만, 납부일이 공휴일(토요휴무일을 포함한다)에 해당하는 경우에는 그날 이후의 첫 번째 근무일까지 납부하여야 합니다. o 보정료는 국고수납은행(대부분의 시중은행)에 납부하거나, 인터넷지로(www.giro.or.kr)로 납부할 수 있습니다. 다만, 보정서를 우편으로 제출하는 경우에는 보정료에 상응하는 통상환을 동봉하여 제출하시면 특허청에서 납부해드립니다.

3. 지정기간연장 안내 : 연장가능기간(4개월)을 초과하여 지정기간을 연장하고자 소명서를 첨부하여 지정기간연장신청서를 제출한 경우 심사관은 아래의 사유에 해당되는지를 판단하여 지정기간연장의 승인 여부 및 연장할 수 있는 기간을 정하여 통지합니다. 【4개월을 초과하여 지정기간연장을 할 수 있는 사유】 ① 기간만료 전 1개월 이내에 최초로 대리인을 선임하거나 선임된 대리인 모두를 해임·변경한 경우 ② 기간만료 전 1개월 이내에 출원인변경신고서를 제출한 경우 ③ 기간만료 전 2개월 이내에 외국특허청의 심사결과를 받은 경우로서 동 심사결과를 보정서에 반영하고자 하는 경우(이 경우 신청서 제출 시 해당 심사결과 사본 및 그 기초가 된 청구범위 사본도 같이 제출해야 함) ④ 의견제출통지서의 송달이 1개월 이상 지연된 경우(1개월 추가 연장 가능) ⑤ 원출원 또는 분할출원이 심판이나 소송에 계류 중인 경우 ⑥ 거절이유와 관련된 시험 및 결과측정에 기간이 더 필요한 경우 ⑦ 기타 불가피하게 기간연장이 필요하다고 인정되는 경우 단, 제3자가 심사청구한 때에는 ①~⑤의 경우라도 불승인

4. 보정안 리뷰(사전검토) 서비스 안내 o 보정서 제출기간 만료일 1개월 이전에 보정안을 제출하고 보정안 리뷰를 신청할 경우 면담을 통해 거절이유 해소 여부 등 보정안에 대한 심사관의 사전검토 서비스를 받을 수 있습니다. ※ 보정안은 보정서 또는 의견서의 형태로 보정안 리뷰 신청 전에 제출하여야 합니다. o 보정안 리뷰 서비스는 특허로 홈페이지(www.patent.go.kr)의 '출원신청 - 보정안리뷰 신청' 메뉴를 통해 신청할 수 있습니다.※ 이 통지서의 내용에 대해 문의사항이 있으시면 특허청 ☎ 042-481-8372(담당심사관 박상철)로, 서식 또는 절차에 대하여는 특허고객상담센터 ☎1544-8080로 문의하시기 바랍니다.

※ 우 35208 대전광역시 서구 청사로 189, 4동 (둔산동, 정부대전청사)

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000751

Electronic Patent Application Fee Transmittal

Application Number:	13666680
Filing Date:	01-Nov-2012
Title of Invention:	METHOD FOR CODING AND AN APPARATUS
First Named Inventor/Applicant Name:	Mehmet Oguz BICI
Filer:	Jonathan Abbott Thomas/Lisa Rone
Attorney Docket Number:	042933/467264

Filed as Large Entity

Filing Fees for Utility under 35 USC 111(a)

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Basic Filing:				
Pages:				
Claims:				
Miscellaneous-Filing:				
Petition:				
Patent-Appeals-and-Interference:				
Post-Allowance-and-Post-Issuance:				
Extension-of-Time:				

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Miscellaneous:				
Submission- Information Disclosure Stmt	1806	1	180	180
		Total in USD ($)		**180**

Electronic Acknowledgement Receipt

EFS ID:	23601419
Application Number:	13666680
International Application Number:	
Confirmation Number:	4782
Title of Invention:	METHOD FOR CODING AND AN APPARATUS
First Named Inventor/Applicant Name:	Mehmet Oguz BICI
Customer Number:	10949
Filer:	Jonathan Abbott Thomas/Lisa Rone
Filer Authorized By:	Jonathan Abbott Thomas
Attorney Docket Number:	042933/467264
Receipt Date:	25-SEP-2015
Filing Date:	01-NOV-2012
Time Stamp:	10:31:38
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	yes
Payment Type	Deposit Account
Payment was successfully received in RAM	$180
RAM confirmation Number	15369
Deposit Account	160605
Authorized User	ALSTON & BIRD, LLP
The Director of the USPTO is hereby authorized to charge indicated fees and credit any overpayment as follows:	

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1		467264-IDS.pdf	235156 65432fbc59050c9e93188186e9c6073aacfd8cea	yes	2
	Multipart Description/PDF files in .zip description				
	Document Description		**Start**	**End**	
	Transmittal Letter		1	1	
	Information Disclosure Statement (IDS) Form (SB08)		2	2	
Warnings:					
Information:					
2	Non Patent Literature	467264-Wiegand_WD3_Working_Draft_NPL.pdf	2552498 6e5d2c619ddf428cb37523ca491d58e77199e47d	no	239
Warnings:					
Information:					
3	Non Patent Literature	467264-Bross_WD4_Working_Draft_NPL.PDF	1821768 ff0766439239504726 48d8d18c05a12d1d410332	no	232
Warnings:					
Information:					
4	Non Patent Literature	467264-CE9_Results_NPL.pdf	102529 bd54227d992df8178fbbff95920c624d3e147acd	no	7
Warnings:					
Information:					
5	Non Patent Literature	467264-korean_office_action.pdf	1383331 9634634c15decfe289d2e6919da9563a3890d373	no	7
Warnings:					
Information:					
6	Fee Worksheet (SB06)	fee-info.pdf	30676 ed0745d0ae47b4e1a977cd1ce56cf34b1f05e758	no	2
Warnings:					
Information:					

Total Files Size (in bytes):	6125958

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

Attorney's Docket No. 042933/467264

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re:	Mehmet Oguz Bici	Confirmation No.:	4782
Appl. No.:	13/666,680	Group Art Unit:	2488
Filed:	11/01/2012	Examiner:	Clifford Hilaire
For:	METHOD FOR CODING AND AN APPARATUS		

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

INFORMATION DISCLOSURE STATEMENT
UNDER 37 C.F.R. § 1.97(c)

Attached is a list of documents on form PTO-1449 along with a copy of any cited foreign patent documents and non-patent literature document in accordance with 37 CFR 1.98(a)(2). Also enclosed is a translation or a concise explanation of each non-English language document.

It is requested that the Examiner consider these documents and officially make them of record in accordance with the provisions of 37 C.F.R. § 1.97 and Section 609 of the MPEP. By identifying the listed documents, Applicant in no way makes any admission as to the prior art status of the listed documents, but is instead identifying the listed documents for the sake of full disclosure.

This Information Disclosure Statement is submitted in accordance with 37 C.F.R. § 1.97(c), before final Office Action or Allowance, whichever is earlier.

The $180.00 fee specified in 37 C.F.R. § 1.17(p) is being paid at the time of e-filing. The Commissioner is authorized to charge any additional fee, or credit any refund, to our Deposit Account No. 16-0605.

Respectfully submitted,

/Douglas E. McKay/

Douglas E. McKay
Registration No. 70,175

CUSTOMER NO. 10949
ALSTON & BIRD LLP
Bank of America Plaza
101 South Tryon Street, Suite 4000
Charlotte, NC 28280-4000
Tel Charlotte Office (704) 444-1000
Fax Charlotte Office (704) 444-1111

Attorney's Docket No. 042933/467264

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re:	Bici et al.	Confirmation No.:	4782
Appl. No.:	13/666,680	Group Art Unit:	2488
Filed:	November 1, 2012		
For:	METHOD FOR CODING AND AN APPARATUS		

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

REQUEST FOR CORRECTED FILING RECEIPT

In reviewing the Filing Receipt for the above-referenced application, Applicant notes that errors appear as noted on the enclosed copy of the Filing Receipt. Applicant requests that a corrected Filing Receipt be issued.

Respectfully submitted,

/Douglas E. McKay/

Douglas E. McKay
Registration No. 70,175

Customer No. 10949
ALSTON & BIRD LLP
Bank of America Plaza
101 South Tryon Street, Suite 4000
Charlotte, NC 28280-4000
Tel Charlotte Office (704) 444-1000
Fax Charlotte Office (704) 444-1111

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING or 371(c) DATE	GRP ART UNIT	FIL FEE REC'D	ATTY.DOCKET.NO	TOT CLAIMS	IND CLAIMS
13/666,680	11/01/2012	2488	2640	042933/467264	20	8

CONFIRMATION NO. 4782

CORRECTED FILING RECEIPT

10949
Nokia Corporation and Alston & Bird LLP
c/o Alston & Bird LLP
Bank of America Plaza, 101 South Tryon Street
Suite 4000
Charlotte, NC 28280-4000

OC000000077598891

Date Mailed: 09/23/2015

Receipt is acknowledged of this non-provisional patent application. The application will be taken up for examination in due course. Applicant will be notified as to the results of the examination. Any correspondence concerning the application must include the following identification information: the U.S. APPLICATION NUMBER, FILING DATE, NAME OF APPLICANT, and TITLE OF INVENTION. Fees transmitted by check or draft are subject to collection. Please verify the accuracy of the data presented on this receipt. **If an error is noted on this Filing Receipt, please submit a written request for a Filing Receipt Correction. Please provide a copy of this Filing Receipt with the changes noted thereon. If you received a "Notice to File Missing Parts" for this application, please submit any corrections to this Filing Receipt with your reply to the Notice. When the USPTO processes the reply to the Notice, the USPTO will generate another Filing Receipt incorporating the requested corrections**

Inventor(s)

Mehmet Oguz BICI, Tampere, FINLAND;
Jani LAINEMA, Tampere, FINLAND;
Kemal UGUR, Tampere, FINLAND;

Applicant(s)

Nokia Technologies Oy, Espoo, FINLAND;

Assignment For Published Patent Application

~~NOKIA CORPORATION, Espoo, FINLAND~~ **Nokia Technologies Oy, Espoo, FINLAND;**

Power of Attorney: The patent practitioners associated with Customer Number 10949

Domestic Priority data as claimed by applicant

This appln claims benefit of 61/555,703 11/04/2011

Foreign Applications for which priority is claimed (You may be eligible to benefit from the **Patent Prosecution Highway** program at the USPTO. Please see http://www.uspto.gov for more information.) - None.
Foreign application information must be provided in an Application Data Sheet in order to constitute a claim to foreign priority. See 37 CFR 1.55 and 1.76.

If Required, Foreign Filing License Granted: 11/20/2012

The country code and number of your priority application, to be used for filing abroad under the Paris Convention, is **US 13/666,680**

Projected Publication Date: Not Applicable

Non-Publication Request: No

Early Publication Request: No

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000759

Title

METHOD FOR CODING AND AN APPARATUS

Preliminary Class

375

Statement under 37 CFR 1.55 or 1.78 for AIA (First Inventor to File) Transition Applications:

PROTECTING YOUR INVENTION OUTSIDE THE UNITED STATES

Since the rights granted by a U.S. patent extend only throughout the territory of the United States and have no effect in a foreign country, an inventor who wishes patent protection in another country must apply for a patent in a specific country or in regional patent offices. Applicants may wish to consider the filing of an international application under the Patent Cooperation Treaty (PCT). An international (PCT) application generally has the same effect as a regular national patent application in each PCT-member country. The PCT process **simplifies** the filing of patent applications on the same invention in member countries, but **does not result** in a grant of "an international patent" and does not eliminate the need of applicants to file additional documents and fees in countries where patent protection is desired.

Almost every country has its own patent law, and a person desiring a patent in a particular country must make an application for patent in that country in accordance with its particular laws. Since the laws of many countries differ in various respects from the patent law of the United States, applicants are advised to seek guidance from specific foreign countries to ensure that patent rights are not lost prematurely.

Applicants also are advised that in the case of inventions made in the United States, the Director of the USPTO must issue a license before applicants can apply for a patent in a foreign country. The filing of a U.S. patent application serves as a request for a foreign filing license. The application's filing receipt contains further information and guidance as to the status of applicant's license for foreign filing.

Applicants may wish to consult the USPTO booklet, "General Information Concerning Patents" (specifically, the section entitled "Treaties and Foreign Patents") for more information on timeframes and deadlines for filing foreign patent applications. The guide is available either by contacting the USPTO Contact Center at 800-786-9199, or it can be viewed on the USPTO website at http://www.uspto.gov/web/offices/pac/doc/general/index.html.

For information on preventing theft of your intellectual property (patents, trademarks and copyrights), you may wish to consult the U.S. Government website, http://www.stopfakes.gov. Part of a Department of Commerce initiative, this website includes self-help "toolkits" giving innovators guidance on how to protect intellectual property in specific countries such as China, Korea and Mexico. For questions regarding patent enforcement issues, applicants may call the U.S. Government hotline at 1-866-999-HALT (1-866-999-4258).

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000760

LICENSE FOR FOREIGN FILING UNDER

Title 35, United States Code, Section 184

Title 37, Code of Federal Regulations, 5.11 & 5.15

GRANTED

The applicant has been granted a license under 35 U.S.C. 184, if the phrase "IF REQUIRED, FOREIGN FILING LICENSE GRANTED" followed by a date appears on this form. Such licenses are issued in all applications where the conditions for issuance of a license have been met, regardless of whether or not a license may be required as set forth in 37 CFR 5.15. The scope and limitations of this license are set forth in 37 CFR 5.15(a) unless an earlier license has been issued under 37 CFR 5.15(b). The license is subject to revocation upon written notification. The date indicated is the effective date of the license, unless an earlier license of similar scope has been granted under 37 CFR 5.13 or 5.14.

This license is to be retained by the licensee and may be used at any time on or after the effective date thereof unless it is revoked. This license is automatically transferred to any related applications(s) filed under 37 CFR 1.53(d). This license is not retroactive.

The grant of a license does not in any way lessen the responsibility of a licensee for the security of the subject matter as imposed by any Government contract or the provisions of existing laws relating to espionage and the national security or the export of technical data. Licensees should apprise themselves of current regulations especially with respect to certain countries, of other agencies, particularly the Office of Defense Trade Controls, Department of State (with respect to Arms, Munitions and Implements of War (22 CFR 121-128)); the Bureau of Industry and Security, Department of Commerce (15 CFR parts 730-774); the Office of Foreign AssetsControl, Department of Treasury (31 CFR Parts 500+) and the Department of Energy.

NOT GRANTED

No license under 35 U.S.C. 184 has been granted at this time, if the phrase "IF REQUIRED, FOREIGN FILING LICENSE GRANTED" DOES NOT appear on this form. Applicant may still petition for a license under 37 CFR 5.12, if a license is desired before the expiration of 6 months from the filing date of the application. If 6 months has lapsed from the filing date of this application and the licensee has not received any indication of a secrecy order under 35 U.S.C. 181, the licensee may foreign file the application pursuant to 37 CFR 5.15(b).

SelectUSA

The United States represents the largest, most dynamic marketplace in the world and is an unparalleled location for business investment, innovation, and commercialization of new technologies. The U.S. offers tremendous resources and advantages for those who invest and manufacture goods here. Through SelectUSA, our nation works to promote and facilitate business investment. SelectUSA provides information assistance to the international investor community; serves as an ombudsman for existing and potential investors; advocates on behalf of U.S. cities, states, and regions competing for global investment; and counsels U.S. economic development organizations on investment attraction best practices. To learn more about why the United States is the best country in the world to develop technology, manufacture products, deliver services, and grow your business, visit http://www.SelectUSA.gov or call +1-202-482-6800.

Electronic Acknowledgement Receipt

EFS ID:	24423480
Application Number:	13666680
International Application Number:	
Confirmation Number:	4782
Title of Invention:	METHOD FOR CODING AND AN APPARATUS
First Named Inventor/Applicant Name:	Mehmet Oguz BICI
Customer Number:	10949
Filer:	Jason P. Cooper/Dennis Avery
Filer Authorized By:	Jason P. Cooper
Attorney Docket Number:	042933/467264
Receipt Date:	21-DEC-2015
Filing Date:	01-NOV-2012
Time Stamp:	12:26:50
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Request for Corrected Filing Receipt	467264_CFR_Request_12-21-2015.pdf	403255 af9d84ebd189ca2ae5ddcd1155dffc6e99a180b8	no	4

Warnings:	
Information:	

Total Files Size (in bytes):	403255

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NUMBER	FILING OR 371(C) DATE	FIRST NAMED APPLICANT	ATTY. DOCKET NO./TITLE
13/666,680	11/01/2012	Mehmet Oguz BICI	042933/467264

CONFIRMATION NO. 4782

10949
Nokia Corporation and Alston & Bird LLP
c/o Alston & Bird LLP
Bank of America Plaza, 101 South Tryon Street
Suite 4000
Charlotte, NC 28280-4000

IMPROPER CFR REQUEST

OC000000079594886

Date Mailed: 12/30/2015

RESPONSE TO REQUEST FOR CORRECTED FILING RECEIPT

Power of Attorney, Claims, Fees, System Limitations, and Miscellaneous

In response to your request for a corrected Filing Receipt, the Office is unable to comply with your request because:

- An Application Data Sheet is needed to make this change.

Questions about the contents of this notice and the requirements it sets forth should be directed to the Office of Data Management, Application Assistance Unit, at **(571) 272-4000** or **(571) 272-4200** or **1-888-786-0101**.

/sstephanos/

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000764

Substitute for form SB08 (Revised 07/09)				Complete if Known	
				Application Number	13/666,680
				Filing Date	11/01/2012
INFORMATION DISCLOSURE STATEMENT BY APPLICANT				First Named Inventor	Mehmet Oguz Bici
				Art Unit	2488
(Use as many sheets as necessary)				Examiner Name	Clifford Hilaire
Sheet	*1*	of	1	Attorney Docket Number	042933/467264

U. S. PATENT DOCUMENTS

Examiner Initials*	Cite No.	Document Number Number - Kind Code (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages of Relevant Figures Appear
	1	US-2004/0234144 A1	11-25-2004	Sugimoto et al.	
	2	US-2008/0159401 A1	07-03-2008	Lee et al.	
	3	US-2011/0182362 A1	07-28-2011	Kim et al.	

OTHER DOCUMENTS

Examiner Initials*	Cite No.	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s) , volume-issue number(s), publisher, city and/or country where published.	English Language Translation Attached
	4	Taiwanese Office Action and Search Report from Taiwanese Patent Application No. 101140777 dated December 2, 2015	

Examiner Signature		Date Considered	

**Examiner: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

CLT#36095683v1

Submitted December 29, 2015

正本

檔　　號：
保存年限：

經濟部智慧財產局　審查意見通知函

機關地址：臺北市大安區辛亥路2段185號3樓
聯 絡 人：王紋星
聯絡電話：(02)23764143
電子郵件：
傳　　真：(02)23779875

受文者：諾基亞股份有限公司（代理人：惲軼群　專利師、陳文郎　專利代理人）

發文日期：中華民國104年11月30日
發文字號：（104）智專二（三）04372字第10421639210號
速　　別：
密等及解密條件或保密期限：
附　　件：如文

10421639210

主旨：第101140777號專利申請案經審查後認有如說明一所述情事，台端（貴公司）請於文到次日起3個月內提出申復說明(一式2份)或修正至局。逾期未復者，本局將依現有資料續行審查，請查照。

說明：

一、本案經審查認為：

（一）本案「編碼方法與裝置」依申請人申請時所送資料內容審查，本案申請專利範圍共20項，其中第1、9、15~20項為獨立項，其餘為附屬項。

（二）　本案申請專利範圍第1、9、19、20項，不符專利法第26條第2項之規定；本案申請專利範圍第4~8、12~14項，不符專利法第26條第4項暨其施行細則第18條之規定。

1、 本案申請專利範圍第1、9、19、20項所載標的「一

經濟部智慧財產局電子公文
簽收時間 104/12/02(2015/12/02) 09:10:40

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000766

種方法」、「一種設備」未敘明係使用於何種技術領域，請查明修正。

2、 本案申請專利範圍第4~8、12~14項係多項附屬項，為直接或間接依附於多項附屬項，請查明修正。

（三）本案除上述指出請求項以外之請求項發明，於現在時點並未發現不予專利理由，如有發現新的不予專利理由時，會再通知不予專利理由。

二、本案如有修正、訂正，應依專利法第43條、第44條第3項、專利法施行細則第36條至第38條之規定辦理，並請依專利修正申請書、專利誤譯訂正申請書之規定撰寫。

三、若希望來局當面示範或說明，請於申復說明書內註明「申請面詢」並繳交規費新臺幣1千元正，本局認為有必要時，將另行通知面詢地點及時間。

四、檢送本案檢索報告1份。

正本：諾基亞股份有限公司（代理人：惲軼群　專利師、陳文郎　專利代理人）
副本：

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000767

附件

第101140777號專利申請案檢索報告

1.申 請 日： 101年11月2日
2.優先權日： 2011年11月4日
3.本案國際專利分類號(IPC)： ***H04N7/28*** (2006.01) ， ***H04N7/26*** (2006.01)
4.檢索國際專利分類號(IPC)範圍： H04N11/02 (2006.01),G06K9/36 (2006.01),H04N7/26 (2006.01)
5.檢索使用資料庫名稱(關鍵詞)： TIPO 國內外專利資料庫

關聯性代 碼	引用文獻資料與相關段落處	相關聯請求項
A	1. US 2011/0170602 A1 2011/07/14 全文	1~20
A	2. US 2011/0182362 A1 2011/07/28 全文	1~20
A	3. US 2004/0234144 A1 2004/11/25 全文	1~20
A	4. US 2008/0159401 A1 2008/07/03 全文	1~20

關聯性代碼說明：

X：單獨引用即足以否定發明新穎性或進步性之特別相關的文獻。

Y：結合一或多篇其他文獻後足以否定發明進步性之特別相關的文獻。

A：一般技術水準之參考文獻。

D：說明書已記載之文獻。

E：申請在前、公開／公告在後之專利文獻。

O：公開使用、販賣或展覽陳列之文件。

P：申請日與優先權日間公開之文獻。

L：其他理由引用之文獻。

完成日： 104年11月26日

簽收時間 104/12/02(2015/12/02) 09:10:40

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000768

Electronic Acknowledgement Receipt

EFS ID:	24475845
Application Number:	13666680
International Application Number:	
Confirmation Number:	4782
Title of Invention:	METHOD FOR CODING AND AN APPARATUS
First Named Inventor/Applicant Name:	Mehmet Oguz BICI
Customer Number:	10949
Filer:	Jonathan Abbott Thomas/Lisa Rone
Filer Authorized By:	Jonathan Abbott Thomas
Attorney Docket Number:	042933/467264
Receipt Date:	29-DEC-2015
Filing Date:	01-NOV-2012
Time Stamp:	11:45:02
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1		467264-IDS.pdf	249813 115b2e7eaac9960db50101b5741ce29b28c6f2c5	yes	2

	Multipart Description/PDF files in .zip description		
	Document Description	**Start**	**End**
	Transmittal Letter	1	1
	Information Disclosure Statement (IDS) Form (SB08)	2	2

Warnings:					
Information:					
2	Non Patent Literature	467264-TW_OA_and_Search_Report.pdf	475313 30a566fc9522ecae69e884a5094097fca66cbb94	no	3
Warnings:					
Information:					
		Total Files Size (in bytes):	725126		

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

Attorney's Docket No. 042933/467264

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re:	Mehmet Oguz Bici	Confirmation No.:	4782
Appl. No.:	13/666,680	Group Art Unit:	2488
Filed:	11/01/2012	Examiner:	Clifford Hilaire
For:	METHOD FOR CODING AND AN APPARATUS		

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

INFORMATION DISCLOSURE STATEMENT
UNDER 37 C.F.R. § 1.97(c)

Attached is a list of documents on form PTO-1449 along with a copy of any cited foreign patent documents and non-patent literature document in accordance with 37 CFR 1.98(a)(2). Also enclosed is a translation or a concise explanation of each non-English language document.

It is requested that the Examiner consider these documents and officially make them of record in accordance with the provisions of 37 C.F.R. § 1.97 and Section 609 of the MPEP. By identifying the listed documents, Applicant in no way makes any admission as to the prior art status of the listed documents, but is instead identifying the listed documents for the sake of full disclosure.

This Information Disclosure Statement is submitted in accordance with 37 C.F.R. § 1.97(c), before final Office Action or Allowance, whichever is earlier.

In accordance with the requirements of 37 C.F.R. § 1.97(c), the following statement as specified in 37 C.F.R. § 1.97(e) is made:

Each item of information contained in this statement was first cited in a communication from a foreign patent office in a counterpart foreign application not more than three (3) months prior to the filing of this statement. In this regard, Applicants note that the communication from the foreign patent office was not received by any individual designated by 37 CFR 1.56(c) more than thirty (30) days prior to the filing of this Information Disclosure Statement.

Respectfully submitted,

/Douglas E. McKay/

Douglas E. McKay
Registration No. 70,175

CUSTOMER NO. 10949
ALSTON & BIRD LLP
Bank of America Plaza
101 South Tryon Street, Suite 4000
Charlotte, NC 28280-4000
Tel Charlotte Office (704) 444-1000
Fax Charlotte Office (704) 444-1111

To: usptomail@alston.com,,
From: PAIR_eOfficeAction@uspto.gov
Cc: PAIR_eOfficeAction@uspto.gov
Subject: Private PAIR Correspondence Notification for Customer Number 10949

Dec 30, 2015 05:24:40 AM

Dear PAIR Customer:

Nokia Corporation and Alston & Bird LLP
c/o Alston & Bird LLP
Bank of America Plaza, 101 South Tryon Street
Suite 4000
Charlotte, NC 28280-4000
UNITED STATES

The following USPTO patent application(s) associated with your Customer Number, 10949 , have new outgoing correspondence. This correspondence is now available for viewing in Private PAIR.

The official date of notification of the outgoing correspondence will be indicated on the form PTOL-90 accompanying the correspondence.

Disclaimer:
The list of documents shown below is provided as a courtesy and is not part of the official file wrapper. The content of the images shown in PAIR is the official record.

Application	Document	Mailroom Date	Attorney Docket No.
13666680	M327	12/30/2015	042933/467264

To view your correspondence online or update your email addresses, please visit us anytime at https://sportal.uspto.gov/secure/myportal/privatepair.

If you have any questions, please email the Electronic Business Center (EBC) at EBC@uspto.gov with 'e-Office Action' on the subject line or call 1-866-217-9197 during the following hours:

Monday - Friday 6:00 a.m. to 12:00 a.m.

Thank you for prompt attention to this notice,

UNITED STATES PATENT AND TRADEMARK OFFICE

PATENT APPLICATION INFORMATION RETRIEVAL SYSTEM

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Appl. No.:	13/666,680	Confirmation No.: 4782
Applicant(s):	Bici et al.	
Filed:	November 1, 2012	
Art Unit:	2488	
Examiner:	Clifford Hilaire	
Title:	METHOD FOR CODING AND AN APPARATUS	

Docket No.: 042933/467264
Customer No.: 10949

Mail Stop Amendment
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

AMENDMENT UNDER 37 C.F.R. § 1.111

In response to the Office Action dated July 6, 2015, please amend the above-identified application as follows:

Amendments to the Claims are reflected in the listing of claims beginning on page 2 of this paper.

Remarks begin on page 13 of this paper.

Amendments to the Claims:

1. (Currently Amended) A method comprising:

receiving a block of pixels including a prediction unit;

determining a set of spatial motion vector prediction candidates for the block of pixels; the spatial motion vector prediction candidates being provided with motion information;

selecting a first spatial motion vector prediction candidate from the set of spatial motion vector prediction candidates as a potential spatial motion vector prediction candidate to be included in a merge list for the prediction unit, where the merge list is selected based on the motion information;

determining a subset of spatial motion vector ~~predictions~~ prediction candidates based on the location of the block associated with the first spatial motion vector prediction candidate;

comparing motion information of the first spatial motion vector prediction candidate with motion information of ~~the~~ spatial motion vector prediction ~~candidate~~ candidates in the determined subset of spatial motion vector prediction candidates; and

if at least one of the comparisons indicates that the motion vector information of the spatial motion vector prediction candidates correspond with each other, excluding the first spatial motion vector prediction candidate from the merge list.

2. (Original) The method according to claim 1 comprising selecting spatial motion vector prediction candidates from the set of spatial motion vector prediction candidates as the potential spatial motion vector prediction candidate in a predetermined order.

3. (Original) The method according to claim 1 , comprising comparing motion information of the potential spatial motion vector prediction candidate with motion information of at most one other spatial motion vector prediction candidate of the set of spatial motion vector prediction candidates.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000774

4. (Original) The method according to claim 1 comprising examining whether the received block of pixels is divided into a first prediction unit and a second prediction unit; and if so, excluding the potential spatial motion vector prediction candidate from the merge list if the prediction unit is the second prediction unit.

5. (Original) The method according to claim 1, further comprising

determining a maximum number of spatial motion vector prediction candidates to be included in a merge list; and

limiting the number of spatial motion vector prediction candidates in the merge list smaller or equal to the maximum number .

6. (Currently Amended) The method according to claim ~~1~~ 5 comprising:

examining, if the number of spatial motion vector prediction candidates in the merge list smaller than the maximum number;

if so, examining whether a prediction unit to which the potential spatial motion vector prediction candidate belongs is available for motion prediction;

if so, performing at least one of the following:

for the potential spatial motion vector prediction candidate on the left side of the prediction unit, excluding the potential spatial motion vector prediction candidate from the merge list if any of the following conditions are fulfilled:

- the received block of pixels is vertically divided into a first prediction unit and a second prediction unit, and the prediction unit is the second prediction unit;
- the received block of pixels is horizontally divided into a first prediction unit and a second prediction unit, and if the prediction unit is the second prediction unit, and the potential spatial motion vector prediction candidate has essentially similar motion information than the spatial motion vector prediction candidate above the prediction unit;

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000775

for the potential spatial motion vector prediction candidate above the prediction unit, excluding the potential spatial motion vector prediction candidate from the merge list if any of the following conditions are fulfilled:

- the received block of pixels is horizontally divided into a first prediction unit and a second prediction unit, and the prediction unit is the second prediction unit;
- the potential spatial motion vector prediction candidate has essentially similar motion information than the spatial motion vector prediction candidate on the left side of the prediction unit;

for the potential spatial motion vector prediction candidate, which is on the right side of the potential spatial motion vector prediction candidate above the prediction unit, excluding the potential spatial motion vector prediction candidate from the merge list if the potential spatial motion vector prediction candidate has essentially similar motion information than the spatial motion vector prediction candidate above the prediction unit;

for the potential spatial motion vector prediction candidate, which is below the potential spatial motion vector prediction candidate on the left side of the prediction unit, excluding the potential spatial motion vector prediction candidate from the merge list if the potential spatial motion vector prediction candidate has essentially similar motion information than the spatial motion vector prediction candidate on the left side of the prediction unit;

for the potential spatial motion vector prediction candidate cornerwise neighbouring the prediction unit, excluding the potential spatial motion vector prediction candidate from the merge list if any of the following conditions are fulfilled:

- all the other potential spatial motion vector prediction candidates have been included in the merge list;
- the potential spatial motion vector prediction candidate has essentially similar motion information than the spatial motion vector prediction candidate above the prediction unit;

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000776

– the potential spatial motion vector prediction candidate has essentially similar motion information than the spatial motion vector prediction candidate on the left side of the prediction unit.

7. (Original) The method according to claim 1 further comprising including a temporal motion prediction candidate into the merge list.

8. (Original) The method according to claim 1 comprising selecting one motion vector prediction candidate from the merge list to represent a motion vector prediction for the block of pixels.

9. (Currently Amended) A method comprising:

receiving an encoded block of pixels including a prediction unit;

determining a set of spatial motion vector prediction candidates for the encoded block of pixels; the spatial motion vector prediction candidates being provided with motion information;

selecting a first spatial motion vector prediction candidate from the set of spatial motion vector prediction candidates as a potential spatial motion vector prediction candidate to be included in a merge list for the prediction unit, where the merge list is selected based on the motion information;

determining a subset of spatial motion vector ~~predictions~~ prediction candidates based on the location of the block associated with the first spatial motion vector prediction candidate;

comparing motion information of the first spatial motion vector prediction candidate with motion information of another spatial motion vector prediction candidate of the set of spatial motion vector prediction candidates; and

if at least one of the comparisons indicates that the motion vector information of the spatial motion vector prediction candidates correspond with each other, excluding the first spatial motion vector prediction candidate from the merge list.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000777

10. (Original) The method according to claim 9 comprising comparing motion information of the potential spatial motion vector prediction candidate with motion information of at most one other spatial motion vector prediction candidate of the set of spatial motion vector prediction candidates.

11. (Original) The method according to claim 9 comprising examining whether the received encoded block of pixels is divided into a first prediction unit and a second prediction unit; and if so, excluding the potential spatial motion vector prediction candidate from the merge list if the prediction unit is the second prediction unit.

12. (Original) The method according to claim 9 further comprising

determining a maximum number of spatial motion vector prediction candidates to be included in a merge list; and

limiting the number of spatial motion vector prediction candidates in the merge list smaller or equal to the maximum number.

13. (Currently Amended) The method according to claim ~~9~~ 12 comprising:

examining, if the number of spatial motion vector prediction candidates in the merge list smaller than the maximum number;

if so, examining whether a prediction unit to which the potential spatial motion vector prediction candidate belongs is available for motion prediction;

if so, performing at least one of the following:

for the potential spatial motion vector prediction candidate on the left side of the prediction unit, excluding the potential spatial motion vector prediction candidate from the merge list if any of the following conditions are fulfilled:

– the received encoded block of pixels is vertically divided into a first prediction unit and a second prediction unit, and the prediction unit is the second prediction unit;

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000778

– the received encoded block of pixels is horizontally divided into a first prediction unit and a second prediction unit, and if the prediction unit is the second prediction unit, and the potential spatial motion vector prediction candidate has essentially similar motion information than the spatial motion vector prediction candidate above the prediction unit;

for the potential spatial motion vector prediction candidate above the prediction unit, excluding the potential spatial motion vector prediction candidate from the merge list if any of the following conditions are fulfilled:

– the received encoded block of pixels is horizontally divided into a first prediction unit and a second prediction unit, and the prediction unit is the second prediction unit;

– the potential spatial motion vector prediction candidate has essentially similar motion information than the spatial motion vector prediction candidate on the left side of the prediction unit;

for the potential spatial motion vector prediction candidate, which is on the right side of the potential spatial motion vector prediction candidate above the prediction unit, excluding the potential spatial motion vector prediction candidate from the merge list if the potential spatial motion vector prediction candidate has essentially similar motion information than the spatial motion vector prediction candidate above the prediction unit;

for the potential spatial motion vector prediction candidate, which is below the potential spatial motion vector prediction candidate on the left side of the prediction unit, excluding the potential spatial motion vector prediction candidate from the merge list if the potential spatial motion vector prediction candidate has essentially similar motion information than the spatial motion vector prediction candidate on the left side of the prediction unit; and

for the potential spatial motion vector prediction candidate cornerwise neighbouring the prediction unit, excluding the potential spatial motion vector prediction candidate from the merge list if any of the following conditions are fulfilled:

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000779

– all the other potential spatial motion vector prediction candidates have been included in the merge list;

– the potential spatial motion vector prediction candidate has essentially similar motion information than the spatial motion vector prediction candidate above the prediction unit;

– the potential spatial motion vector prediction candidate has essentially similar motion information than the spatial motion vector prediction candidate on the left side of the prediction unit.

14. (Original) The method according to claim 9 comprising selecting one motion vector prediction candidate from the merge list to represent a motion vector prediction for the received encoded block of pixels.

15. (Currently Amended) An apparatus comprising a processor and a memory including computer program code, the memory and the computer program code configured to, with the processor, cause the apparatus to:

~~receiving~~ receive a block of pixels including a prediction unit;

~~determining~~ determine a set of spatial motion vector prediction candidates for the block of pixels; the spatial motion vector prediction candidates being provided with motion information;

~~selecting~~ select a first spatial motion vector prediction candidate from the set of spatial motion vector prediction candidates as a potential spatial motion vector prediction candidate to be included in a merge list for the prediction unit, where the merge list is selected based on the motion information;

~~determining~~ determine a subset of spatial motion vector ~~predictions~~ prediction candidates based on the location of the block associated with the first spatial motion vector prediction candidate;

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000780

~~comparing~~ compare motion information of the first spatial motion vector prediction candidate with motion information of the spatial motion vector prediction candidate in the determined subset of spatial motion vector prediction candidates; and

if at least one the comparisons indicates that the motion vector information of the spatial motion vector prediction candidates correspond with each other, excluding the first spatial motion vector prediction candidate from the merge list.

16. (Currently Amended) An apparatus comprising a processor and a memory including computer program code, the memory and the computer program code configured to, with the processor, cause the apparatus to:

receive an encoded block of pixels including a prediction unit;

determine a set of spatial motion vector prediction candidates for the encoded block of pixels; the spatial motion vector prediction candidates being provided with motion information;

select a first spatial motion vector prediction candidate from the set of spatial motion vector prediction candidates as a potential spatial motion vector prediction candidate to be included in a merge list for the prediction unit, where the merge list is selected based on the motion information;

determine a subset of spatial motion vector ~~predictions~~ prediction candidates based on the location of the block associated with the first spatial motion vector prediction candidate;

compare motion information of the first spatial motion vector prediction candidate with motion information of the spatial motion vector prediction candidate in the determined subset of spatial motion vector prediction candidates; and

exclude the first spatial motion vector prediction candidate from the merge list, if at least one of the comparisons indicates that the motion vector information of the spatial motion vector prediction candidates correspond with each other

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000781

17. (Currently Amended) A ~~storage~~ non-transitory computer readable medium having stored thereon a computer executable program code for use by an encoder, said program codes comprise instructions for use by an encoder, said program code comprises instructions for:

receiving a block of pixels including a prediction unit;

determining a set of spatial motion vector prediction candidates for the block of pixels; the spatial motion vector prediction candidates being provided with motion information;

selecting a first spatial motion vector prediction candidate from the set of spatial motion vector prediction candidates as a potential spatial motion vector prediction candidate to be included in a merge list for the prediction unit, where the merge list is selected based on the motion information;

determining a subset of spatial motion vector ~~predictions~~ prediction candidates based on the location of the block associated with the first spatial motion vector prediction candidate;

comparing motion information of the first spatial motion vector prediction candidate with motion information of the spatial motion vector prediction candidate in the determined subset of spatial motion vector prediction candidates; and

if at least one of the comparisons indicates that the motion vector information of the spatial motion vector prediction candidates correspond with each other, excluding the first spatial motion vector prediction candidate from the merge list.

18. (Currently Amended) A ~~storage~~ non-transitory computer readable medium having stored thereon a computer executable program code for use by an encoder, said program codes comprise instructions for use by an encoder, said program code comprises instructions for:

receiving an encoded block of pixels including a prediction unit;

determining a set of spatial motion vector prediction candidates for the encoded block of pixels; the spatial motion vector prediction candidates being provided with motion information;

selecting a first spatial motion vector prediction candidate from the set of spatial motion vector prediction candidates as a potential spatial motion vector prediction candidate to be

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000782

included in a merge list for the prediction unit, where the merge list is selected based on the motion information;

determining a subset of spatial motion vector ~~predictions~~ prediction candidates based on the location of the block associated with the first spatial motion vector prediction candidate;

comparing motion information of the first spatial motion vector prediction candidate with motion information of the spatial motion vector prediction candidate in the determined subset of spatial motion vector prediction candidates; and

if at least one of the comparisons indicates that the motion vector information of the spatial motion vector prediction candidates correspond with each other, excluding the first spatial motion vector prediction candidate from the merge list.

19. (Currently Amended) An apparatus comprising:

means for receiving a block of pixels including a prediction unit;

means for determining a set of spatial motion vector prediction candidates for the block of pixels; the spatial motion vector prediction candidates being provided with motion information;

means for selecting a first spatial motion vector prediction candidate from the set of spatial motion vector prediction candidates as a potential spatial motion vector prediction candidate to be included in a merge list for the prediction unit, where the merge list is selected based on the motion information;

means for determining a subset of spatial motion vector ~~predictions~~ prediction candidates based on the location of the block associated with the first spatial motion vector prediction candidate;

means for comparing motion information of the first spatial motion vector prediction candidate with motion information of the spatial motion vector prediction candidate in the determined subset of spatial motion vector prediction candidates; and

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000783

means for excluding the first spatial motion vector prediction candidate from the merge list, if at least one of the comparisons indicates that the motion vector information of the spatial motion vector prediction candidates correspond with each other.

20. (Currently Amended) An apparatus comprising:

means for receiving an encoded block of pixels including a prediction unit;

means for determining a set of spatial motion vector prediction candidates for the encoded block of pixels; the spatial motion vector prediction candidates being provided with motion information;

means for selecting a first spatial motion vector prediction candidate from the set of spatial motion vector prediction candidates as a potential spatial motion vector prediction candidate to be included in a merge list for the prediction unit, where the merge list is selected based on the motion information;

means for determining a subset of spatial motion vector ~~predictions~~ prediction candidates based on the location of the block associated with the first spatial motion vector prediction candidate;

means for comparing motion information of the first spatial motion vector prediction candidate with motion information of the spatial motion vector prediction candidate in the determined subset of spatial motion vector prediction candidates; and

means for excluding the first spatial motion vector prediction candidate from the merge list, if at least one of the comparisons indicates that the motion vector information of the spatial motion vector prediction candidates correspond with each other.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000784

REMARKS

Claim Rejections - 35 USC § 112

Claims 1, 6, 9,13, 15, 16, 17, 18, 19, and 20 were rejected under 35 USC § 112(b) or 35 USC § 112 (pre-AIA), second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which the inventor or a joint inventor, or for pre-AIA the applicant regards as the invention. These claims have been amended. No new matter is added. Reconsideration of this rejection is respectfully requested.

Claim Rejections - 35 USC § 101

Claims 17 and 18 were rejected under 35 USC § 101 as not falling within one of the four statutory categories of invention. Claims 17 and 18 have been amended so as to no longer recite "A storage medium." Instead, the independent Claims 17 and 18 has been amended to recite "A ~~storage~~ non-transitory computer readable medium" as suggested by the Office Action. As such, it is submitted that the rejection under 35 USC § 101 of independent Claims 17 and 18, as amended, is overcome.

Claim Rejections - 35 USC § 103

Claims 1, 2, 3, 4, 5, 6, 7, 9, 10, 11, 12, 14, 15, 16, 17, 18, 19, and 20 were rejected under pre-AIA 35 USC § 103(a) as being unpatentable over Han et al. (*Improved Video Compression Efficiency Through Flexible Unit Representation and Corresponding Extension of Coding Tools*) in view of Tai et al. (*A Multi-Pass True Motion Estimation Scheme With Motion Vector Propagation for Frame Rate Up-Conversion Applications.*).

The Office Action agrees that the Han does not teach or suggest the feature of "selecting a first spatial motion vector prediction candidate from the set of spatial motion vector prediction

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000785

candidates as a potential spatial motion vector prediction candidate to be included in a merge list for the prediction unit; determining a subset of spatial motion vector prediction candidates based on the location of the block associated with the first spatial motion vector prediction candidate; comparing motion information of the first spatial motion vector prediction candidate with motion information of ~~the~~ spatial motion vector prediction candidates in the determined subset of spatial motion vector prediction candidates; if at least one of the comparisons indicates that the motion vector information of the spatial motion vector prediction candidates correspond with each other, excluding the first spatial motion vector prediction candidate from the merge list," and cite Tai for these missing teaching from Han. (See the Office Action, page 5-6) Applicant respectfully asserts that Tai fails to cure the deficiencies of Han. While Applicants respectfully disagree with the above-noted rejections, Applicants have amended the independent claims to clarify and further distinguish the claims from the cited art. In this regard, the independent claims have been amended to recite that the merge list is selected based on the motion information and if at least one of the comparisons indicates that the motion vector information of the spatial motion vector prediction candidates correspond with each other, excluding the first spatial motion vector prediction candidate from the merge list. Support for the amendment is provided at least by paragraphs [0120]-[0122] of the published application.

Applicant respectfully asserts that Tai fails to cure the deficiencies of Han. Specifically, the Office Action cites Col 1, ¶ 4-5 of Tai with respect to the rejection of these recitations of the independent claims. However, neither the cited portion and, indeed, no other portion of Tai teaches or suggests these elements. Instead, the cited portion describes selecting minimum distortion in the current pass to refer in the future pass. However, Tai fails to teach or suggest the "selecting a first spatial motion vector prediction candidate from the set of spatial motion vector prediction candidates as a potential spatial motion vector prediction candidate to be included in a merge list for the prediction unit, where the merge list is selected based on the motion information," as recited in independent Claims 1, 9, and 15-20. Specifically, Tai doesn't disclose various merge lists and select certain merge list based on the motion information to store spatial motion vector prediction candidate. As such, Tai fails to teach excluding the spatial

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000786

motion vector prediction candidate from the selected merge list. In addition, Tai does not teach such exclusion as Tai merely select PMV pattern with minimal distortion. Consequently, Tai does not teach or suggest "selecting a first spatial motion vector prediction candidate from the set of spatial motion vector prediction candidates as a potential spatial motion vector prediction candidate to be included in a merge list for the prediction unit, where the merge list is selected based on the motion information...if at least one of the comparisons indicates that the motion vector information of the spatial motion vector prediction candidates correspond with each other, excluding the first spatial motion vector prediction candidate from the merge list," as recited in independent Claims 1, 9, and 15-20.

Applicants therefore assert that none of the cited references, taken alone or in combination, teach or suggest: "selecting a first spatial motion vector prediction candidate from the set of spatial motion vector prediction candidates as a potential spatial motion vector prediction candidate to be included in a merge list for the prediction unit, where the merge list is selected based on the motion information; determining a subset of spatial motion vector prediction candidates based on the location of the block associated with the first spatial motion vector prediction candidate; comparing motion information of the first spatial motion vector prediction candidate with motion information of ~~the~~ spatial motion vector prediction candidates in the determined subset of spatial motion vector prediction candidates; if at least one of the comparisons indicates that the motion vector information of the spatial motion vector prediction candidates correspond with each other, excluding the first spatial motion vector prediction candidate from the merge list," as recited by independent Claims 1, 9, and 15-20. For each of the foregoing reasons, it is submitted that the rejection under pre-AIA 35 USC § 103(a) of the independent claims, as amended, as well as the claims which depend therefrom, is overcome.

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000787

CONCLUSION

In view of the amendments to the claims and the remarks presented above, it is respectfully submitted that all of the claims of the present application are in condition for immediate allowance. It is therefore respectfully requested that a Notice of Allowance be issued. The Examiner is encouraged to contact Applicant's undersigned attorney to resolve any remaining issues in order to expedite examination of the present application.

The patentability of the independent claims has been argued as set forth above and thus Applicants will not take this opportunity to argue the merits of the rejection with regard to specific dependent claims. However, Applicants do not concede that the dependent claims are not independently patentable and reserve the right to argue the patentability of dependent claims at a later date if necessary.

It is not believed that extensions of time or fees for net addition of claims are required, beyond those that may otherwise be provided for in documents accompanying this paper. However, in the event that additional extensions of time are necessary to allow consideration of this paper, such extensions are hereby petitioned under 37 CFR § 1.136(a), and any fee required therefor (including fees for net addition of claims) is hereby authorized to be charged to Deposit Account No. 16-0605.

Respectfully submitted,

/Guy R. Gosnell/

Guy R. Gosnell
Registration No. 34,610
Customer No. Error! Reference source not found.
ALSTON & BIRD LLP
Bank of America Plaza
101 South Tryon Street, Suite 4000
Charlotte, NC 28280-4000
Tel Charlotte Office (704) 444-1000
Fax Charlotte Office (704) 444-1111

ELECTRONICALLY FILED USING THE EFS-WEB ELECTRONIC FILING SYSTEM OF THE UNITED STATES PATENT & TRADEMARK OFFICE ON January 6, 2016

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000788

Electronic Patent Application Fee Transmittal

Application Number:	13666680
Filing Date:	01-Nov-2012
Title of Invention:	METHOD FOR CODING AND AN APPARATUS
First Named Inventor/Applicant Name:	Mehmet Oguz BICI
Filer:	Jonathan Abbott Thomas/Patricia Forbes
Attorney Docket Number:	042933/467264

Filed as Large Entity

Filing Fees for Utility under 35 USC 111(a)

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Basic Filing:				
Pages:				
Claims:				
Miscellaneous-Filing:				
Petition:				
Patent-Appeals-and-Interference:				
Post-Allowance-and-Post-Issuance:				
Extension-of-Time:				

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Extension - 3 months with $0 paid	1253	1	1400	1400
Miscellaneous:				
		Total in USD ($)		**1400**

Electronic Acknowledgement Receipt

EFS ID:	24547373
Application Number:	13666680
International Application Number:	
Confirmation Number:	4782
Title of Invention:	METHOD FOR CODING AND AN APPARATUS
First Named Inventor/Applicant Name:	Mehmet Oguz BICI
Customer Number:	10949
Filer:	Jonathan Abbott Thomas/Patricia Forbes
Filer Authorized By:	Jonathan Abbott Thomas
Attorney Docket Number:	042933/467264
Receipt Date:	06-JAN-2016
Filing Date:	01-NOV-2012
Time Stamp:	17:03:36
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	yes
Payment Type	Deposit Account
Payment was successfully received in RAM	$1400
RAM confirmation Number	3991
Deposit Account	160605
Authorized User	ALSTON & BIRD, LLP
The Director of the USPTO is hereby authorized to charge indicated fees and credit any overpayment as follows: Charge any Additional Fees required under 37 CFR 1.16 (National application filing, search, and examination fees) Charge any Additional Fees required under 37 CFR 1.17 (Patent application and reexamination processing fees)	

Charge any Additional Fees required under 37 CFR 1.21 (Miscellaneous fees and charges)

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1		467264_AmendmentEoT.PDF	9560286 60e084100ccf40d17eeeaab52ca9571509ddf617	yes	17

Multipart Description/PDF files in .zip description		
Document Description	**Start**	**End**
Extension of Time	1	1
Amendment/Req. Reconsideration-After Non-Final Reject	2	2
Claims	3	13
Applicant Arguments/Remarks Made in an Amendment	14	17

Warnings:

Information:

2	Fee Worksheet (SB06)	fee-info.pdf	31097 b07533a5c61db03227ce80bafcfb6a3fa14f3328	no	2

Warnings:

Information:

Total Files Size (in bytes):	9591383

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

PTO/AIA/22 (03-13)
Approved for use through 7/31/2016. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PETITION FOR EXTENSION OF TIME UNDER 37 CFR 1.136(a)

Docket Number (Optional): 042933/467264

Application Number: 13/666,680

Filed: November 1, 2012

For: METHOD FOR CODING AND AN APPARATUS

Art Unit: 2488

Examiner: Clifford Hilaire

This is a request under the provisions of 37 CFR 1.136(a) to extend the period for filing a reply in the above-identified application.

The requested extension and fee are as follows (check time period desired and enter the appropriate fee below):

	Fee	Small Entity Fee	Micro Entity Fee	
☐ One month (37 CFR 1.17(a)(1))	$200	$100	$50	$
☐ Two months (37 CFR 1.17(a)(2))	$600	$300	$150	$
☑ Three months (37 CFR 1.17(a)(3))	$1,400	$700	$350	$ 1,400.00
☐ Four months (37 CFR 1.17(a)(4))	$2,200	$1,100	$550	$
☐ Five months (37 CFR 1.17(a)(5))	$3,000	$1,500	$750	$

☐ Applicant asserts small entity status. See 37 CFR 1.27.

☐ Applicant certifies micro entity status. See 37 CFR 1.29.
Form PTO/SB/15A or B or equivalent must either be enclosed or have been submitted previously.

☐ A check in the amount of the fee is enclosed.

☐ Payment by credit card. Form PTO-2038 is attached.

☐ The Director has already been authorized to charge fees in this application to a Deposit Account.

☑ The Director is hereby authorized to charge any fees which may be required, or credit any overpayment, to Deposit Account Number 16-0605.

☐ Payment made via EFS-Web.

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

I am the

☐ applicant.

☑ attorney or agent of record. Registration number 34,610.

☐ attorney or agent acting under 37 CFR 1.34. Registration number _____.

/Guy R. Gosnell/ — Signature

January 6, 2016 — Date

Guy R. Gosnell — Typed or printed name

704-444-1000 — Telephone Number

NOTE: This form must be signed in accordance with 37 CFR 1.33. See 37 CFR 1.4 for signature requirements and certifications. Submit multiple forms if more than one signature is required, see below*.

☐ * Total of _____ forms are submitted.

This collection of information is required by 37 CFR 1.136(a). The information is required to obtain or retain a benefit by the public, which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection is estimated to take 6 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Mail Stop PCT, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

PTO/SB/06 (09-11)
Approved for use through 1/31/2014. OMB 0651-0032
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

PATENT APPLICATION FEE DETERMINATION RECORD Substitute for Form PTO-875	Application or Docket Number 13/666,680	Filing Date 11/01/2012	☐ To be Mailed

ENTITY: ☒ LARGE ☐ SMALL ☐ MICRO

APPLICATION AS FILED – PART I

	(Column 1)	(Column 2)		
FOR	NUMBER FILED	NUMBER EXTRA	RATE ($)	FEE ($)
☐ BASIC FEE (37 CFR 1.16(a), (b), or (c))	N/A	N/A	N/A	
☐ SEARCH FEE (37 CFR 1.16(k), (i), or (m))	N/A	N/A	N/A	
☐ EXAMINATION FEE (37 CFR 1.16(o), (p), or (q))	N/A	N/A	N/A	
TOTAL CLAIMS (37 CFR 1.16(i))	minus 20 =	*	X $ =	
INDEPENDENT CLAIMS (37 CFR 1.16(h))	minus 3 =	*	X $ =	
☐ APPLICATION SIZE FEE (37 CFR 1.16(s))	If the specification and drawings exceed 100 sheets of paper, the application size fee due is $310 ($155 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).			
☐ MULTIPLE DEPENDENT CLAIM PRESENT (37 CFR 1.16(j))				
* If the difference in column 1 is less than zero, enter "0" in column 2.			TOTAL	

APPLICATION AS AMENDED – PART II

		(Column 1)		(Column 2)	(Column 3)		
	01/06/2016	CLAIMS REMAINING AFTER AMENDMENT		HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA	RATE ($)	ADDITIONAL FEE ($)
AMENDMENT	Total (37 CFR 1.16(i))	* 20	Minus	** 20	=	X $ =	
	Independent (37 CFR 1.16(h))	* 8	Minus	***8	=	X $ =	
	☐ Application Size Fee (37 CFR 1.16(s))						
	☐ FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))						
						TOTAL ADD'L FEE	

		(Column 1)		(Column 2)	(Column 3)		
		CLAIMS REMAINING AFTER AMENDMENT		HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA	RATE ($)	ADDITIONAL FEE ($)
AMENDMENT	Total (37 CFR 1.16(i))	*	Minus	**	=	X $ =	
	Independent (37 CFR 1.16(h))	*	Minus	***	=	X $ =	
	☐ Application Size Fee (37 CFR 1.16(s))						
	☐ FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))						
						TOTAL ADD'L FEE	

* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.
** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20".
*** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3".
The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in the appropriate box in column 1.

LIE
/KATRINA . TURNER/

This collection of information is required by 37 CFR 1.16. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
13/666,680	11/01/2012	Mehmet Oguz BICI	042933/467264	4782

10949 7590 03/01/2016

Nokia Corporation and Alston & Bird LLP
c/o Alston & Bird LLP
Bank of America Plaza, 101 South Tryon Street
Suite 4000
Charlotte, NC 28280-4000

EXAMINER
HILAIRE, CLIFFORD

ART UNIT	PAPER NUMBER
2488	

NOTIFICATION DATE	DELIVERY MODE
03/01/2016	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

usptomail@alston.com

Amazon.com, Inc., Amazon.com Services LLC - Ex. 1002, Page 000795

Office Action Summary	Application No. 13/666,680	Applicant(s) BICI ET AL.	
	Examiner CLIFFORD HILAIRE	Art Unit 2488	AIA (First Inventor to File) Status No

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTHS FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *01/06/2016*.

☐ A declaration(s)/affidavit(s) under **37 CFR 1.130(b)** was/were filed on _____.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ An election was made by the applicant in response to a restriction requirement set forth during the interview on _____; the restriction requirement and election have been incorporated into this action.

4)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims*

5)☒ Claim(s) *1-20* is/are pending in the application.

5a) Of the above claim(s) _____ is/are withdrawn from consideration.

6)☐ Claim(s) _____ is/are allowed.

7)☒ Claim(s) *1-20* is/are rejected.

8)☐ Claim(s) _____ is/are objected to.

9)☐ Claim(s) _____ are subject to restriction and/or election requirement.

* If any claims have been determined allowable, you may be eligible to benefit from the **Patent Prosecution Highway** program at a participating intellectual property office for the corresponding application. For more information, please see http://www.uspto.gov/patents/init_events/pph/index.jsp or send an inquiry to PPHfeedback@uspto.gov.

Application Papers

10)☐ The specification is objected to by the Examiner.

11)☒ The drawing(s) filed on *11/01/2012* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

Priority under 35 U.S.C. § 119

12)☒ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

Certified copies:

a)☒ All b)☐ Some** c)☐ None of the:

1.☒ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

** See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)

2) ☒ Information Disclosure Statement(s) (PTO/SB/08a and/or PTO/SB/08b) Paper No(s)/Mail Date *09/25/2015;12/29/2015*.

3) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .

4) ☐ Other: _____.

DETAILED ACTION

Notice of Pre-AIA or AIA Status

1. The present application is being examined under the pre-AIA first to invent provisions.

Applicant(s) Response to Official Action

2. The response filed on 01/06/2016 has been entered and made of record.

Response to Arguments/Amendments

3. Presented arguments have been fully considered, but are rendered moot in view of the new ground(s) of rejection necessitated by amendment(s) initiated by the applicant(s).

Claim Rejections - 35 USC § 112

The following is a quotation of 35 U.S.C. 112(b):

(b) CONCLUSION.—The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the inventor or a joint inventor regards as the invention.

The following is a quotation of 35 U.S.C. 112 (pre-AIA), second paragraph:

The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

4. Claims 6 & 13 rejected under 35 U.S.C. 112(b) or 35 U.S.C. 112 (pre-AIA), second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which the inventor or a joint inventor, or for pre-AIA the applicant regards as the invention.

5. Claims 6 & 13 recites the limitations:

"**the** potential spatial motion vector prediction candidate **on the left side of** the prediction unit",

"**the** spatial motion vector prediction candidate **above** the prediction unit",

"**the** potential spatial motion vector prediction candidate **above** the prediction unit",

"**the** spatial motion vector prediction candidate on **the left side** of the prediction unit",

"**the** potential spatial motion vector prediction candidate, **which is on the right side of** the potential spatial motion vector prediction candidate above the prediction unit",

"**the** potential spatial motion vector prediction candidate, **which is below** the potential spatial motion vector prediction candidate on the left side of the prediction unit", and

"**the** potential spatial motion vector prediction candidate **cornerwise** neighboring the prediction unit" in lines 7, 16, 18, 26, 28, 33 and 39 respectively.

There are insufficient antecedent bases for these limitation in the claims.

6. Claim 6 recites the limitations:

"- **the received block of pixels** is vertically divided into a first prediction unit and a second prediction unit, and **the prediction unit** is the second prediction unit" *(line 10),*

"- **the received block of pixels** is horizontally divided into a first prediction unit and a second prediction unit, and if **the prediction unit** is the second prediction unit" *(line 13)* and

"- **the received block of pixels** is horizontally divided into a first prediction unit and a second prediction unit, and **the prediction unit** is the second prediction unit" *(line 21)*.

The original antecedent basis for "**received block of pixels**" is traced back to claim 1 "receiving a **block of pixels** including a **prediction unit**" *(line 2)*. From the latter citation, "**received block of pixels**" can be interpreted as a set of prediction units (RBP_N) and "a **prediction unit**" can be identified as (PU_i), where $RBP_N = \{PU_i | \ i \in [1, N]\}$ and N is a non-zero number of prediction unit included in RBP_N.

Then in claim 6, "**a prediction unit**" *(line 4)* is introduced anew to the one in claim 1, as underlined above, without mentioning any correspondence to (RBP_N). If "**the received block of pixels** is vertically divided into a first prediction unit and a second prediction unit **"** or "**the received block of pixels** is horizontally divided into a first prediction unit and a second prediction unit", said set of prediction units if further limited to $RBP_2 = \{PU_1, PU_2\}$ where any of PU_1 or PU_2 is "**the prediction unit**" for which the antecedent basis resides in claim 1.