### 7.3.5 Coding tree syntax

|  |  |
| --- | --- |
| coding\_tree( x0, y0, log2CUSize ) { | Descriptor |
| if( x0 + ( 1 << log2CUSize ) <= PicWidthInSamplesL &&  y0 + ( 1 << log2CUSize ) <= PicHeightInSamplesL &&  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) |
| } |  |
| ... |  |
| } |  |

### 7.3.5 Coding unit syntax

|  |  |
| --- | --- |
| 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 &&  ( cu\_split\_pred\_part\_mode[x0][y0]==4)) |  |
| { |  |
| **inter\_part\_horz\_flag** | u(1) |
| if (log2CUSiz != Log2MinCUSize) |  |
| **rem\_inter\_part\_mode** | u(v) |
| } |  |
| } else if( slice\_type != I | | log2CUSize = = Log2MinCUSize ) |  |
| **pred\_type** | u(v) | ae(v) |
| ... |  |
| } |  |
| ... |  |
| } |  |

### 7.4.5 Coding tree semantics

**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.

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.

### 7.4.5 Coding unit semantics

**inter\_part\_horz\_flag** specifies PartMode of a coding unit in conjunction with cu\_split\_pred\_part\_mode and optionally with rem\_inter\_part\_mode. inter\_part\_horz\_flag equal to 1 specifies that there are horizontally long rectangular partitions in a coding unit. inter\_part\_horz\_flag equal to 0 specifies that there are vertically long rectangular partitions in a coding unit. The variable PredMode is derived from slice\_type, cu\_split\_pred\_part\_mode, inter\_part\_dir\_flag, and rem\_inter\_part\_mode based on Table 7-xx.

**rem\_inter\_part\_mode** specifies PartMode of a coding unit in conjunction with cu\_split\_pred\_part\_mode and inter\_part\_horz\_flag. The value of rem\_inter\_part\_mode indicates one of PartMode from a group of partitioning types. The variable PredMode is derived from slice\_type, cu\_split\_pred\_part\_mode, inter\_part\_dir\_flag, and rem\_inter\_part\_mode based on Table 7-xx. When the rem\_inter\_part\_mode is not present, it shall be infered to be equal to 0.

* Table 7‑xx – Definition of PredMode and PartMode based on slice\_type, cu\_split\_pred\_part\_mode, intra\_part\_mode, inter\_part\_horz\_flag, and rem\_ianter\_part\_mode

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **slice\_type** | **cu\_split\_pred\_part\_mode** | **intra\_part\_mode** | **inter\_part\_horz\_flag** | **rem\_inter\_part\_mode** | **PredMode** | **PartMode** |
|  | 1 | - | - | - | MODE\_SKIP | PART\_2Nx2N |
| P or B | 2 | - | - | - | MODE\_INTER (MERGE) | PART\_2Nx2N |
| 3 | - | - | - | MODE\_INTER | PART\_2Nx2N |
| 4 | - | 1 | 0 | MODE\_INTER | PART\_2NxN |
| - | 1 | MODE\_INTER | PART\_2NxnU |
| - | 2 | MODE\_INTER | PART\_2NxnD |
| - | 0 | 0 | MODE\_INTER | PART\_Nx2N |
| - | 1 | MODE\_INTER | PART\_nLx2N |
| - | 2 | MODE\_INTER | PART\_nRx2N |
| 5 | 0 | - | - | MODE\_INTRA | PART\_2Nx2N |
| 1 | - | - | MODE\_INTRA | PART\_NxN |

### 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 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **codeNum** | | | | | |
| **cuDepth** | **0** | **1** | **2** | **3** | **4** | **5** |
| 0 | 0 | 1 | 2 | 3 | 4 | 5 |
| 1 | 0 | 1 | 2 | 3 | 4 | 5 |
| 2 | 0 | 1 | 2 | 3 | 4 | 5 |
| 3 | 1 | 2 | 3 | 4 | 5 | - |

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.

### 9.2.4.5 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 5 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 4 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] (9‑41)

* 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.

### 9.2.4.x Parsing process for rem\_inter\_part\_mode

This process is invoked when entropy\_coding\_mode\_flag is equal to 0 for parsing syntax element **rem\_inter\_part\_mode** in subclause .

Inputs to this process are bits from slice data.

Output of this process is the syntax element rem\_inter\_part\_mode.

The value of rem\_inter\_part\_mode is derived as follows.

* The parsing process described in subclause is invoked with vlcNum of 14 and cMax of 2 as inputs and the value of rem\_inter\_part\_mode as output.