|  |  |
| --- | --- |
| sao\_offset\_vlc( rx, ry, cIdx ) { | Descriptor |
| **sao\_on\_switch** [ cIdx ][ rx ][ ry ] | u(1) |
| if (sao\_on\_switch [ cIdx ][ rx ][ ry ]) { |  |
| **sao\_type\_idx**[ cIdx ][ rx ][ ry ] | u(1) |
| if( sao\_type\_idx[ cIdx ][ rx ][ ry ] = =1 ) { |  |
| **sao\_subtype\_idx**[ cIdx ][ rx ][ ry ] | u(5) |
| for( i = 0; i < 4; i++ ) |  |
| **sao\_offset**[ cIdx ][ rx][ ry ][ i ] | se(v) |
| } else { |  |
| **sao\_subtype\_idx**[ cIdx ][ rx ][ ry ] | u(2) |
| for( i = 0; i < 4; i++ ) |  |
| **sao\_offset**[ cIdx ][ rx][ ry ][ i ] | ue(v) |
| **}** |  |
| } |  |
| } |  |

|  |  |
| --- | --- |
| sao\_offset\_cabac( rx, ry, cIdx ) { | Descriptor |
| **sao\_on\_switch** [ cIdx ][ rx ][ ry ] | ae(v) |
| if (sao\_on\_switch [ cIdx ][ rx ][ ry ]) { |  |
| **sao\_type\_idx**[ cIdx ][ rx ][ ry ] | ae(v) |
| if( sao\_type\_idx[ cIdx ][ rx ][ ry ] = =1 ) |  |
| **sao\_subtype\_idx**[ cIdx ][ rx ][ ry ] | u(5) |
| else |  |
| **sao\_subtype\_idx**[ cIdx ][ rx ][ ry ] | u(2) |
| for( i = 0; i < 4; i++ ) |  |
| **sao\_offset**[ cIdx ][ rx][ ry ][ i ] | ae(v) |
| } |  |
| } |  |

#### Sample adaptive offset VLC semantics

**sao\_on\_switch** [ cIdx ][ rx ][ ry ] indicates whether SAO is applied to the current coding treeblock at position rx and ry for the colour component cIdx.

**sao\_type\_idx**[ cIdx ][ rx ][ ry ] indicates the offset type as specified in of current coding treeblock at position rx and ry for the colour component cIdx.

When sao\_type\_idx[ cIdx ][ rx ][ ry ] is not present, it is inferred as follows.

* If sao\_merge\_up\_flag is equal to 1, sao\_type\_idx[ cIdx ][ rx ][ ry ] is set equal to sao\_type\_idx[ cIdx ][ rx ][ ry − 1 ].
* Otherwise, sao\_type\_idx[ cIdx ][ rx ][ ry ] is set equal to sao\_type\_idx[ cIdx ][ rx − 1 ][ ry ].

Table 7‑6 – Specification of the edge type for SAO

|  |  |
| --- | --- |
| **sao\_type\_idx[ cIdx ][ rx ][ ry ]** | **Type** |
| 0 | Edge |
| 1 | Band |

**sao\_subtype\_idx** [ cIdx ][ rx ][ ry ] indicates the direction in case of edge offset and the displacement of the band offset in case of band offset. Edge offset directions are specified in when sao\_type\_idx[ cIdx ][ rx ][ ry ]==0.

|  |  |
| --- | --- |
| **sao\_subtype\_idx[ cIdx ][ rx ][ ry ]** | **Edge type** |
| 0 | 1D 0-degree edge |
| 1 | 1D 90-degree edge |
| 2 | 1D 135-degree edge |
| 3 | 1D 45-degree edge |

When sao\_subtype\_idx [ cIdx ][ rx ][ ry ] is not present it is inferred as follows.

* If sao\_merge\_up\_flag is equal to 1, sao\_subtype\_idx [ cIdx ][ rx ][ ry ] is set equal to sao\_subtype\_idx [ cIdx ][ rx ][ ry − 1 ].
* Otherwise, sao\_subtype\_idx [ cIdx ][ rx ][ ry ] is set equal to sao\_subtype\_idx [ cIdx ][ rx − 1 ][ ry ].

**sao\_offset**[ cIdx ][ rx ][ ry ][ i ] indicates the offset value of i-th category of current coding treeblock at position rx and ry for the colour component cIdx.

The variable bitDepth is derived as follows.

* If cIdx is equal to 0, bitDepth is set equal to BitDepthY..
* Otherwise (cIdx is equal to1 or 2), bitDepth is set equal to BitDepthC.

It is a requirement of bitstream conformance that when sao\_type\_idx[ cIdx ][ rx ][ ry ] is not equal to 5, the values of sao\_offset[ cIdx ][ rx ][ ry ][ i ] shall be in the range of 0 to ( 1<< ( Min( bitDepth, 10 ) − 5 ) ) − 1, inclusive and otherwise, the values of sao\_offset[ cIdx ][ rx ][ ry ][ i ] shall be in the range of −( 1<< ( Min( bitDepth, 10 ) − 5 ) ) to ( 1<< ( Min( bitDepth, 10 ) − 5 ) ) − 1, inclusive.

When sao\_offset[ cIdx ][ rx ][ ry ][ i ] is not present, it is inferred as follows.

* If sao\_merge\_up\_flag is equal to 1, sao\_offset[ cIdx ][ rx ][ ry ][ i ] is set equal to sao\_offset[ cIdx ][ rx ][ ry − 1 ][ i ].
* Otherwise, sao\_offset[ cIdx ][ rx ][ ry ][ i ] is set equal to sao\_offset[ cIdx ][ rx − 1 ][ ry ][ i ].

The variable offsetSign is derived as follows.

* If sao\_type\_idx[ cIdx ][ rx ][ ry ] is less than 5 and i is larger than 1,offsetSign is set to −1.
* Otherwise (sao\_type\_idx[ cIdx ][ rx ][ ry ] is equal to 5 or i is less than 2), offsetSign is set to 1.

The array SaoOffsetVal is derived as follows.

SaoOffsetVal[ cIdx ][ rx ][ ry ][ 0 ] = 0 (7‑36)

SaoOffsetVal[ cIdx ][ rx ][ ry ][ i + 1 ] =   
 offsetSign\*sao\_offset[ cIdx ][ rx ][ ry ][ i ] << ( bitDepth – Min( bitDepth, 10 ) ) (7‑37)