### Residual coding syntax

|  |  |
| --- | --- |
| numSigCoeff = 0 |  |
| sumAbs = 0 |  |
| cParam = 0 |  |
| for( n = 15; n >= 0; n− − ) { |  |
| xC = ScanOrder[ log2TrafoWidth ][ log2TrafoHeight ][ scanIdx ][ n + offset ][ 0 ] |  |
| yC = ScanOrder[ log2TrafoWidth ][ log2TrafoHeight ][ scanIdx ][ n + offset ][ 1 ] |  |
| if( significant\_coeff\_flag[ xC ][ yC ] ) { |  |
| baseLevel[ n ] = 1 + coeff\_abs\_level\_greater1\_flag[ n ] + coeff\_abs\_level\_greater2\_flag[ n ] |  |
| if( baseLevel[ n ] = = ( ( numSigCoeff < 8 ) ? ( (n = = firstGreater1CoeffIdx) ? 3 : 2 ) : 1 ) ) |  |
| **coeff\_abs\_level\_remaining[** n **]** | ae(v) |
| if(coeff\_abs\_level\_remaining[ n ] + baseLevel[ n ] > 3\*(1<< cParam) |  |
| cParam = Min(cParam+1,4) |  |
| transCoeffLevel[ x0 ][ y0 ][ cIdx ][ xC ][ yC ] =   ( coeff\_abs\_level\_remaining[ n ] + baseLevel[ n ] ) \* ( 1 − 2 \* coeff\_sign\_flag[ n ] ) |  |
| if( sign\_data\_hiding\_flag && signHidden ) { |  |
| sumAbs += ( coeff\_abs\_level\_remaining[ n ] + baseLevel[ n ] ) |  |
| if( n = = firstNZPosInCG && (sumAbs%2 = = 1) ) |  |
| transCoeffLevel[x0][y0][cIdx][xC][yC] = −  transCoeffLevel[x0][y0][cIdx][xC][yC] |  |
| } |  |
| numSigCoeff++ |  |
| } else |  |
| transCoeffLevel[ x0 ][ y0 ][ cIdx ][ xC ][ yC ] = 0 |  |
| } |  |

#### Binarization process for coeff\_abs\_level\_remaining

Input to this process is a request for a binarization for the syntax element coeff\_abs\_level\_remaining[ n ].

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

The variables cLastAbsCoeff and cLastParam are derived as follows– If n is equal to 0 or all previous syntex elements coeff\_abs\_level\_remaining[ pos ] with pos less than n are derived to be equal to 0 instead of being explicitly parsedcLastAbsCoeff is set to baseLevel[ 0 ] and cLastParam is set equal to 0 individually.

– Otherwise ( n is not equal to 0 ), cLastAbsCoeff is set equal to sum of coeff\_abs\_level\_remaining[ n − 1 ] and baseLevel[ n-1 ]. cLastParam is set equal to the value of cParam that has been derived during the invocation of this subclause for the syntax element coeff\_abs\_level\_remaining[ n − 1 ] of the same transform block.

The variable cParam is derived from cLastAbsCoeff and cLastParam equivalent to the following pseudo-code

If cLastAbsCoeff > 3\*(1<<cLastParam)

cParam = Min(cLastParam +1, 4)



The binarization of coeff\_abs\_level\_remaining consists of a prefix part and (when present) a suffix part. The prefix part of the binarization is derived by invoking the -unary binarization process as specified in subclause 9.2.2.1 - with input synElVal derived as follows:.

– cLength is set equal to cParam and cVal is set equal to coeff\_abs\_level\_remaining[ n ].

– If coeff\_abs\_level\_remaining[ n ] is less than (8<<cParam), synElVal is set equal to

(coeff\_abs\_level\_remaining[ n ]>> cParam)

– Otherwise, the value of synElVal is specified by a process equivalent to the following pseudo-code:

cVal = cVal - (8<<cParam )

while (cVal >= (1<<cLength))

{

cVal = cVal - (1<<cLength)

cLength = cLength+1

}

synElVal = 8+ cLength – cParam

The suffix bin string is derived by invoking the FL binarization process as specified in subclause 9.2.2.5 with cMax equal to (1<<cLength)-1 and input equal to cVal % (1<<cLength). The value of cLength and cVal is specified by the process equivalent to the above pseudo-code.