### Vertical hybrid diagonal scanning array initialisation process

Input to this process is a block size blkSize.

Output of this process is the array VertHybScan[ pos ][ comp ]. The array index pos specify the scan position ranging from 0 to ( blkSize\*blkSize ) − 1. The array index comp equal to 0 specifies the horizontal component and the array index, comp equal to 1 specifies the vertical component.

VertHybScan [ 0 ][ 0 ] = 0  
VertHybScan [ 0 ][ 1 ] = 0  
i = 1  
x = 0  
y = 1  
while( x==0 )   
{

VertHybScan[ i ][ 0 ] = x

VertHybScan[ i ][ 1 ] = y

y++

i=y

if ( y==blkSize )

{

x=1

}  
}

for (x=1; x<blkSize; x++)

{

for (y=0; y<blkSize; y++)

{

diagSum = x + y

if (diagSum < (blkSize-1))

{

VertHybScan[ i ] = DiagScan[ i – (blkSize-1) + diagSum ]

}

else

{

VertHybScan[ i ] = DiagScan[ i ]

}

i++

}

}

## 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, scanIdx equal to 3 specifies an up-right diagonal scan as specified in subclause 6.5.2, scanIdx equal to 4 specifies a horizontal hybrid scan and scanIdx equal to 5 specifies a vertical hybrid scan in subclause 6.5.3. 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

ScanOrder[5] = VertHybScan

i = 0

while (i < blkWidth\*blkWidth)

{

ScanOrder[ 4 ][ i ][ 0 ] = ScanOrder[ 5 ][ i ][ 1 ]

ScanOrder[ 4 ][ i ][ 1 ] = ScanOrder[ 5 ][ i ][ 0 ]

i++

}

### 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.

Table 7‑16 – Specification of ScanType[ log2TrafoSize − 2 ][ IntraPredMode ]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **IntraPredMode** | **log2TrafoSize − 2** | | | |
| **0** | **1** | **2** | **3** |
| 0 | 1 | 1 | 4 | 4 |
| 1 | 2 | 2 | 5 | 5 |
| 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 | 1 | 1 | 0 | 0 |
| 21-22 | 1 | 1 | 4 | 4 |
| 23 | 1 | 1 | 0 | 0 |
| 24-27 | 0 | 0 | 0 | 0 |
| 28 | 2 | 2 | 0 | 0 |
| 29-30 | 2 | 2 | 5 | 5 |
| 31 | 2 | 2 | 0 | 0 |
| 32-33 | 0 | 0 | 0 | 0 |
| 34 | 0 | 0 | 0 | 0 |
| 35 | 0 | 0 | 0 | 0 |

### 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 which scan order is used to scan the transform coefficients (zigzag, horizontal, vertical, diagonal, horizontal hybrid or vertical hybrid), is derived as follows.

* 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).

When entropy\_coding\_mode\_flag is equal to 1 (CABAC) and scanIdx is not equal to 0 to 3, the variable c[ i ][ j ] which is located in the ( i, j ) position in the array c is derived as follows:

– Define position arrays I and J, each of length n (where I(0) corresponds to position i for n=0)

– When n==0:  
 iNew=I(0)=0

jNew=J(0)=0

– Define Boolean processedLastSigCoef, and initialize to false when n==0

– When n>0

if (scanIdx==4)

{

lastSigI = last\_significant\_coefficient\_x

lastSigJ = last\_significant\_coefficient\_y

}

else // transpose data

{

lastSigI = last\_significant\_coefficient\_y

lastSigJ = last\_significant\_coefficient\_x

}

lastSigCoefInFirstRowOrCol = lastSigJ==0 ? true : false

iOld=I(n-1)

jOld=J(n-1)

if (lastSigCoefInFirstRowOrCol)

{

iNew = iOld + 1

jNew = 0

}

else

{

if (processedLastSigCoef)

{

iNew = iOld + jOld + 1

jNew = 0

}

else

{

if ( iOld==0 && jOld<(nS-1) )

{

iNew = jOld + 1

jNew = 0

}

else if ( jOld==(nS-1) )

{

jNew = (iOld + jOld) – (nS-2)

iNew = nS - 1

}

else

{

iNew = iOld – 1

jNew = jOld + 1

}

if ( iNew==lastSigI && jNew==lastSigJ )

{

processedLastSigCoef = true

}

}

}

I(n) = iNew

J(n) = jNew

– c[ i ][ j ] is given as follows:

* When scanIdx== 4:

c[ iNew ][ jNew ] = transCoeffLevel[ xT ][ yT ][ trafoDepth ][ cIdx ][ n ]

* When scanIdx== 5: // transpose the data

c[ jNew ][ iNew ] = transCoeffLevel[ xT ][ yT ][ trafoDepth ][ cIdx ][ n ]

with n = 0.. ( nS \* nS ) – 1

Otherwise, 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 (‑)