|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG16 WP3 and ISO/IEC JTC1/SC29/WG11**  8th Meeting: San José, CA, USA, 1–10 February, 2012 | Document: JCTVC-H0427 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **Non-CE11: Support modification for parallel calculation of significant map contexts** | | |
| *Status:* | Input Document to JCT-VC | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Vadim Seregin  Joel Sole  Marta Karczewicz  5775 Morehouse Drive San Diego, CA 92121, USA | Email: | [vseregin@qualcomm.com](mailto:vseregin@qualcomm.com)  [joels@qualcomm.com](mailto:joels@qualcomm.com)  [martak@qualcomm.com](mailto:martak@qualcomm.com) |
| *Source:* | Qualcomm | | |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Abstract

In HM5.0, for large transforms, significance map context calculation is dependent on the number of non-zero coefficients in a neighborhood support. Coefficients are removed from the neighbourhood to allow a two parallel context estimation. In this contribution, having 3 and 4 parallel bin processing is studied and proposed. Three methods are analyzed. A recommended approach has an average BD-rate of 0.1% and doesn’t introduce additional conditions.

# Introduction

In current test model (HM), five-point support *S* is used to define a context model to code significance map of the transform coefficients. The example of the five-point support in blue color, relative to target point marked in purple is shown in Fig. 1. Context model *Ctx* is derived from the sum of the significant flags in every point of the support, significance flag is set to *1* if corresponding transform coefficient is nonzero, otherwise significance flag is set to *0*.

(1)



Fig. 1. Significance map support and reverse scan order.

However, this support may not be suitable in case there is a requirement to calculate a context of more than 2 coefficients in parallel, because one or several significance context of transform coefficients are dependent within the support. To illustrate this dependency, let’s consider 4x4 block, shown on Fig. 2.



Fig. 2. Dependency in significance context calculation within support.

To calculate a significance context for the point marked in purple, it is necessary to parse significance flag of the coefficient within the support depicted in red in Fig. 2(a). So, this necessary parsing introduces a delay if there is a requirement to calculate significance contexts of two coefficients in parallel.

To resolve this dependency, it was proposed and adopted in current HM to remove dependent transform coefficient from the support, making the support with so called “*hole*”, depicted on Fig. 2(b).

The term *hole* was borrowed from Semiconductor Physics, used to describe the process in p-n junction in particular and meaning absence of electron, due to its similarity to the absence of coefficient in a support here.

The significance flag in the hole is skipped and not taken into account for the context calculation (i.e., assumed to be zero), so there is no need to know the significance flag in that point. The support shape depends on the position to allow for better parallel processing. However, in current HM, making a hole in the support as shown on Fig. 2(b), only two significance can be computed in parallel.

# Proposal

## Method 1. Extending the holes for parallel context calculation

It is desired to be able to calculate more bins in parallel. In this case, more holes should be conditionally inserted in the support, depending on the desired degree of parallelization.

On Fig. 3 (a), holes are conditionally inserted for three context calculated in parallel. All the significance contexts of the coefficients marked in non-black color are calculated using support with holes. In some cases, more than one hole in the support is necessary as shown on Fig. 3(b),(c),(d). So there are four different supports depending on the current coefficient position. The original support that has no holes, then 2 supports with one hole (beneath and on the right of the coded coefficient), and another support with two holes.

 Fig. 3. Inserting the holes for the three coefficients parallel processing.

In a similar way, shown on Fig. 4, holes are inserted to have 4 bins processed in parallel.



Fig. 4. Inserting the holes for the four coefficients parallel processing.

## Method 2.Grouping coefficients for parallel context calculation

In the method 1, conditions have to be checked in order to apply the right support to each position. And the number of conditions is increasing with the increase of bins processed in parallel, since more holes should be checked and inserted for more positions.

However, number of conditions to be checked can be reduced assuming that every group of three or four significant contexts are calculated in parallel and significant flags are parsed right after that. In this case, only coefficient located in the same group are excluded from context calculation.

Grouping for three bin parallel processing, is shown on Fig. 5.



Fig. 5. Inserting the holes for the parallel processing in groups of three coefficients.

In a similar way, grouping is done for four bins, shown on Fig. 6.



Fig. 6. Inserting the holes for the parallel processing in groups of four coefficients.

It is seen from the Fig. 5 and 6, number of holes and conditions to be checked is much less than for method 1.

## Method 3. Support modification for parallel context calculation

To reduce number of conditions to be checked further, context support is modified. This modified support is applied to calculate significant map context for all coefficient regardless of their positions in the block. New supports are shown on Fig. 7, where support for three bins is shown on Fig. 7(a) and for four bins on Fig. 7(b).



Fig. 7. Modified support for the parallel processing.

With this modified supports, only two holes (conditions) are necessary be inserted to have 3 and 4 bins processed in parallel for significant map. The location of that holes are shown on Fig. 8 and 9 for 3 and 4 bins respectively.



Fig. 8. Modified support and holes for the three bins parallel processing.



Fig. 9. Modified support and holes for the four bins parallel processing.

Note that the 3- and 4-bins modified support uses the same number of conditional checks than HM5.0.

# Simulation results

All three methods were implemented on top of HM5.0. And simulations results based on common test conditions [1] are summarised in the Table 1. More detailed test data can be found in the accompanied excel sheets.

Table 1. Luma BD-rate summary for proposed methods

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | AI-HE | AI-LC | RA-HE | RA-LC | RA-HE10 | HE-LB | LC-LB |
| Method 1 3 bins | 0.09 | 0.07 | 0.06 | 0.05 | 0.14 | 0.06 | 0.02 |
| Method 1 4 bins | 0.11 | 0.09 | 0.08 | 0.07 | 0.16 | 0.05 | 0.05 |
| Method 2 3 bins | 0.01 | 0.01 | 0.00 | 0.02 | 0.04 | 0.01 | 0.00 |
| Method 2 4 bins | 0.12 | 0.09 | 0.09 | 0.08 | 0.16 | 0.07 | 0.02 |
| Method 3 3 bins | 0.11 | 0.06 | 0.08 | 0.05 | 0.16 | 0.10 | 0.04 |
| Method 3 4 bins | 0.21 | 0.12 | 0.17 | 0.11 | 0.25 | 0.22 | 0.07 |

# Conclusion

In this contribution, several schemes are studied allowing 3 and 4 significant map context be calculated in parallel. Among all, last scheme with modified support has less holes and conditions to be checked, so we recommend method 3 for adoption in HM. The 3 and 4 bins case for method 3 has the same number of conditions than HM5.0.

# References

1. F. Bossen, “*Common HM test conditions and software reference configurations*,” JCTVC-G1200, 7th JCT-VC Meeting, Geneva, CH, November 2011.

# Patent rights declaration(s)

**Qualcomm 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).**

# Working Draft text

This text modification for working draft is based on JCTVC-G1103 version d8.

## Method2 3 bins

###### 9.2.3.1.1.5 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 ), the transform block width log2TrafoWidth and the transform block height log2TrafoHeight.

Output of this process is ctxIdxInc.

The variable sigCtx depends on the current position ( xC, yC ), the color component index cIdx, the transform block size and previsously decoded bins of the syntax element significant\_coeff\_flag. For the derivation of sigCtx, the following applies.

* If log2TrafoWidth is equal to log2TrafoHeight and log2TrafoWidth is equal to 2, sigCtx is derived using ctxIdxMap4x4[ ] specified in Table 9‑39 as follows..

sigCtx = ctxIdxMap4x4[ ((cIdx > 0) ? 15 : 0) + (yC << 2) + xC ] (9‑55)

* Otherwise if log2TrafoWidth is equal to log2TrafoHeight and log2TrafoWidth is equal to 3, sigCtx is derived using ctxIdxMap8x8[ ] specified in Table 9‑40 as follows.

sigCtx = ((xC + yC) = = 0) ? 10 : ctxIdxMap8x8[ ((yC >> 1 ) << 2) + (xC >> 1) ] (9‑56)

sigCtx += ( cIdx > 0) ? 6: 9 (9‑56)

* Otherwise if xC + yC is equal to 0, sigCtx is derived as follows.

sigCtx = ( cIdx > 0) ? 17: 20 (9‑57)

* Otherwise (xC + yC is greater than 0), sigCtx is derived using previously decoded bins of the syntax element significant\_coeff\_flag as follows.
* The variable sigCtx is initialized as follows.

sigCtx = 0 (9‑58)

* When xC is less than ( 1 << log2TrafoWidth ) – 1 and xC % 4 is not equal to 3 or yC % 4 is not equal to 2, the following applies.

sigCtx = sigCtx + significant\_coeff\_flag[ xC + 1 ][ yC ] (9‑59)

* When xC is less than ( 1 << log2TrafoWidth ) − 1 and yC is less than ( 1 << log2TrafoHeight ) − 1, the following applies.

sigCtx = sigCtx + significant\_coeff\_flag[ xC + 1 ][ yC + 1 ] (9‑60)

* When xC is less than ( 1 << log2Width ) − 2, the following applies.

sigCtx = sigCtx + significant\_coeff\_flag[ xC + 2 ][ yC ] (9‑61)

* When all of the following conditions are true,
  + yC is less than ( 1 << log2TrafoHeight ) − 1,
  + xC % 4 is not equal to 0 or yC % 4 is not equal to 1,
  + xC % 4 is not equal to 3 or yC % 4 is not equal to 2,

the following applies.

sigCtx = sigCtx + significant\_coeff\_flag[ xC ][ yC + 1 ] (9‑62)

* When yC is less than ( 1 << log2TrafoHeight ) − 2 and sigCtx is less than 4, the following applies.

sigCtx = sigCtx + significant\_coeff\_flag[ xC ][ yC + 2 ] (9‑63)

* The variable sigCtx is modified as follows.
  + If cIdx is equal to 0 and xC + yC are greater than (1 << (max(log2TrafoWidth, log2TrafoHeight) − 2)) − 1, the following applies.

sigCtx = ( (sigCtx + 1) >> 1 ) + 24 (9‑63)

* + Otherwise, the following applies.

sigCtx = ( (sigCtx + 1) >> 1 ) + ( (cIdx > 0) ? 18 : 21 ) (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 = 27 + sigCtx

## Method2 4 bins

###### 9.2.3.1.1.5 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 ), the transform block width log2TrafoWidth and the transform block height log2TrafoHeight.

Output of this process is ctxIdxInc.

The variable sigCtx depends on the current position ( xC, yC ), the color component index cIdx, the transform block size and previsously decoded bins of the syntax element significant\_coeff\_flag. For the derivation of sigCtx, the following applies.

* If log2TrafoWidth is equal to log2TrafoHeight and log2TrafoWidth is equal to 2, sigCtx is derived using ctxIdxMap4x4[ ] specified in Table 9‑39 as follows..

sigCtx = ctxIdxMap4x4[ ((cIdx > 0) ? 15 : 0) + (yC << 2) + xC ] (9‑55)

* Otherwise if log2TrafoWidth is equal to log2TrafoHeight and log2TrafoWidth is equal to 3, sigCtx is derived using ctxIdxMap8x8[ ] specified in Table 9‑40 as follows.

sigCtx = ((xC + yC) = = 0) ? 10 : ctxIdxMap8x8[ ((yC >> 1 ) << 2) + (xC >> 1) ] (9‑56)

sigCtx += ( cIdx > 0) ? 6: 9 (9‑56)

* Otherwise if xC + yC is equal to 0, sigCtx is derived as follows.

sigCtx = ( cIdx > 0) ? 17: 20 (9‑57)

* Otherwise (xC + yC is greater than 0), sigCtx is derived using previously decoded bins of the syntax element significant\_coeff\_flag as follows.
* The variable sigCtx is initialized as follows.

sigCtx = 0 (9‑58)

* When xC is less than ( 1 << log2TrafoWidth ) − 1 and
  + xC % 4 is not equal to 0 or yC % 4 is not equal to 0,
  + xC % 4 is not equal to 3 or yC % 4 is not equal to 2,

the following applies.

sigCtx = sigCtx + significant\_coeff\_flag[ xC + 1 ][ yC ] (9‑59)

* When xC is less than ( 1 << log2TrafoWidth ) − 1 and yC is less than ( 1 << log2TrafoHeight ) − 1, the following applies.

sigCtx = sigCtx + significant\_coeff\_flag[ xC + 1 ][ yC + 1 ] (9‑60)

* When xC is less than ( 1 << log2Width ) − 2, the following applies.

sigCtx = sigCtx + significant\_coeff\_flag[ xC + 2 ][ yC ] (9‑61)

* When all of the following conditions are true,
  + yC is less than ( 1 << log2TrafoHeight ) − 1,
  + xC % 4 is not equal to 0 or yC % 4 is not equal to 0,
  + xC % 4 is not equal to 3 or yC % 4 is not equal to 2,
  + yC % 4 is not equal to 1,

the following applies.

sigCtx = sigCtx + significant\_coeff\_flag[ xC ][ yC + 1 ] (9‑62)

* When yC is less than ( 1 << log2TrafoHeight ) − 2 and
  + xC % 4 is not equal to 0 or yC % 4 is not equal to 0,
  + xC % 4 is not equal to 1 or yC % 4 is not equal to 3,
  + sigCtx is less than 4, the following applies.

sigCtx = sigCtx + significant\_coeff\_flag[ xC ][ yC + 2 ] (9‑63)

* The variable sigCtx is modified as follows.
  + If cIdx is equal to 0 and xC + yC are greater than (1 << (max(log2TrafoWidth, log2TrafoHeight) − 2)) − 1, the following applies.

sigCtx = ( (sigCtx + 1) >> 1 ) + 24 (9‑63)

* + Otherwise, the following applies.

sigCtx = ( (sigCtx + 1) >> 1 ) + ( (cIdx > 0) ? 18 : 21 ) (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 = 27 + sigCtx

## Method3 3 bins

###### 9.2.3.1.1.5 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 ), the transform block width log2TrafoWidth and the transform block height log2TrafoHeight.

Output of this process is ctxIdxInc.

The variable sigCtx depends on the current position ( xC, yC ), the color component index cIdx, the transform block size and previsously decoded bins of the syntax element significant\_coeff\_flag. For the derivation of sigCtx, the following applies.

* If log2TrafoWidth is equal to log2TrafoHeight and log2TrafoWidth is equal to 2, sigCtx is derived using ctxIdxMap4x4[ ] specified in Table 9‑39 as follows..

sigCtx = ctxIdxMap4x4[ ((cIdx > 0) ? 15 : 0) + (yC << 2) + xC ] (9‑55)

* Otherwise if log2TrafoWidth is equal to log2TrafoHeight and log2TrafoWidth is equal to 3, sigCtx is derived using ctxIdxMap8x8[ ] specified in Table 9‑40 as follows.

sigCtx = ((xC + yC) = = 0) ? 10 : ctxIdxMap8x8[ ((yC >> 1 ) << 2) + (xC >> 1) ] (9‑56)

sigCtx += ( cIdx > 0) ? 6: 9 (9‑56)

* Otherwise if xC + yC is equal to 0, sigCtx is derived as follows.

sigCtx = ( cIdx > 0) ? 17: 20 (9‑57)

* Otherwise (xC + yC is greater than 0), sigCtx is derived using previously decoded bins of the syntax element significant\_coeff\_flag as follows.
* The variable sigCtx is initialized as follows.

sigCtx = 0 (9‑58)

* When xC is less than ( 1 << log2TrafoWidth ) − 1, the following applies.

sigCtx = sigCtx + significant\_coeff\_flag[ xC + 1 ][ yC ] (9‑59)

* When xC is less than ( 1 << log2TrafoWidth ) − 1 and yC is less than ( 1 << log2TrafoHeight ) − 1, and ~~the following applies.~~
  + xC % 4 is not equal to 0 or yC % 4 is not equal to 0,
  + xC % 4 is not equal to 2 or yC % 4 is not equal to 3,

the following applies.

sigCtx = sigCtx + significant\_coeff\_flag[ xC + 1 ][ yC + 1 ] (9‑60)

* When xC is less than ( 1 << log2Width ) − 2, the following applies.

sigCtx = sigCtx + significant\_coeff\_flag[ xC + 2 ][ yC ] (9‑61)

* ~~When all of the following conditions are true,~~
  + ~~yC is less than ( 1 << log2TrafoHeight ) − 1,~~
  + ~~xC % 4 is not equal to 0 or yC % 4 is not equal to 0,~~
  + ~~xC % 4 is not equal to 3 or yC % 4 is not equal to 2,~~

~~the following applies.~~

~~sigCtx = sigCtx + significant\_coeff\_flag[ xC ][ yC + 1 ]~~ (9‑62)

* When yC is less than ( 1 << log2TrafoHeight ) − 2 ~~and sigCtx is less than 4, the following applies.~~

sigCtx = sigCtx + significant\_coeff\_flag[ xC ][ yC + 2 ] (9‑63)

and if sigCtx is less than 4 and xC is less than ( 1 << log2Width ) – 2

sigCtx = sigCtx + significant\_coeff\_flag[ xC + 1 ][ yC + 2 ]

* The variable sigCtx is modified as follows.
  + If cIdx is equal to 0 and xC + yC are greater than (1 << (max(log2TrafoWidth, log2TrafoHeight) − 2)) − 1, the following applies.

sigCtx = ( (sigCtx + 1) >> 1 ) + 24 (9‑63)

* + Otherwise, the following applies.

sigCtx = ( (sigCtx + 1) >> 1 ) + ( (cIdx > 0) ? 18 : 21 ) (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 = 27 + sigCtx (9‑65)

## Method3 4 bins

###### 9.2.3.1.1.5 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 ), the transform block width log2TrafoWidth and the transform block height log2TrafoHeight.

Output of this process is ctxIdxInc.

The variable sigCtx depends on the current position ( xC, yC ), the color component index cIdx, the transform block size and previsously decoded bins of the syntax element significant\_coeff\_flag. For the derivation of sigCtx, the following applies.

* If log2TrafoWidth is equal to log2TrafoHeight and log2TrafoWidth is equal to 2, sigCtx is derived using ctxIdxMap4x4[ ] specified in Table 9‑39 as follows..

sigCtx = ctxIdxMap4x4[ ((cIdx > 0) ? 15 : 0) + (yC << 2) + xC ] (9‑55)

* Otherwise if log2TrafoWidth is equal to log2TrafoHeight and log2TrafoWidth is equal to 3, sigCtx is derived using ctxIdxMap8x8[ ] specified in Table 9‑40 as follows.

sigCtx = ((xC + yC) = = 0) ? 10 : ctxIdxMap8x8[ ((yC >> 1 ) << 2) + (xC >> 1) ] (9‑56)

sigCtx += ( cIdx > 0) ? 6: 9 (9‑56)

* Otherwise if xC + yC is equal to 0, sigCtx is derived as follows.

sigCtx = ( cIdx > 0) ? 17: 20 (9‑57)

* Otherwise (xC + yC is greater than 0), sigCtx is derived using previously decoded bins of the syntax element significant\_coeff\_flag as follows.
* The variable sigCtx is initialized as follows.

sigCtx = 0 (9‑58)

* ~~When xC is less than ( 1 << log2TrafoWidth ) − 1, the following applies.~~

~~sigCtx = sigCtx + significant\_coeff\_flag[ xC + 1 ][ yC ]~~ (9‑59)

* When xC is less than ( 1 << log2Width ) − 2, the following applies.

sigCtx = sigCtx + significant\_coeff\_flag[ xC + 2 ][ yC ] (9‑61)

and when yC is less than ( 1 << log2TrafoHeight ) − 1, the following applies

sigCtx = sigCtx + significant\_coeff\_flag[ xC + 2 ][ yC + 1]

* When yC is less than ( 1 << log2TrafoHeight ) − 2 and
  + xC % 4 is not equal to 0 or yC % 4 is not equal to 0,
  + xC % 4 is not equal to 1 or yC % 4 is not equal to 3,

the following applies.

sigCtx = sigCtx + significant\_coeff\_flag[ xC ][ yC + 2 ]

* When xC is less than ( 1 << log2TrafoWidth ) − 1 and yC is less than ( 1 << log2TrafoHeight ) − 1, the following applies.

sigCtx = sigCtx + significant\_coeff\_flag[ xC + 1 ][ yC + 1 ] (9‑60)

and when yC is less than ( 1 << log2TrafoHeight ) – 2 and sigCtx is less than 4, the following applies

sigCtx = sigCtx + significant\_coeff\_flag[ xC + 1 ][ yC + 2]

* ~~When all of the following conditions are true,~~
  + ~~yC is less than ( 1 << log2TrafoHeight ) − 1,~~
  + ~~xC % 4 is not equal to 0 or yC % 4 is not equal to 0,~~
  + ~~xC % 4 is not equal to 3 or yC % 4 is not equal to 2,~~

~~the following applies.~~

~~sigCtx = sigCtx + significant\_coeff\_flag[ xC ][ yC + 1 ]~~ (9‑62)

* ~~When yC is less than ( 1 << log2TrafoHeight ) − 2 and sigCtx is less than 4, the following applies.~~

~~sigCtx = sigCtx + significant\_coeff\_flag[ xC ][ yC + 2 ] (9‑63)~~

* The variable sigCtx is modified as follows.
  + If cIdx is equal to 0 and xC + yC are greater than (1 << (max(log2TrafoWidth, log2TrafoHeight) − 2)) − 1, the following applies.

sigCtx = ( (sigCtx + 1) >> 1 ) + 24 (9‑63)

* + Otherwise, the following applies.

sigCtx = ( (sigCtx + 1) >> 1 ) + ( (cIdx > 0) ? 18 : 21 ) (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 = 27 + sigCtx (9‑65)