|  |  |
| --- | --- |
| **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-H0309 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **CE7: Boundary-Dependent Transform for Inter-Predicted Residue** | | |
| *Status:* | Input Document to JCT-VC | | |
| *Purpose:* | Proposal | | |
|  |  | | |
| *Author(s) or Contact(s):* | \*Jicheng An, \*Xin Zhao, \*Xun Guo, #Shawmin Lei  \*North Building 10F, Raycom Infotech Park Tower C, No. 2 Kexueyuan South Rd., Haidian District, Beijing, China 100190  #No. 1, Dusing Rd. 1, Hsinchu Science Park, Hsinchu, Taiwan 30078 | Email: | {jicheng.an, x.zhao, xun.guo, shawmin.lei}@mediatek.com | |
| *Source:* | MediaTek Inc. | | |

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

# Abstract

This contribution tests the boundary-dependent transform (BDT) proposed in JCTVC-G281 on top of HM5.0. It is reported that the proposed method can achieve around 0.2%~0.6% BD-rate reductions for luma. The run time is reportedly similar to that of HM5.0. It is also reported that the additional transforms needed by the proposed method can be implemented by re-using the existing transform cores in HEVC.

# Introduction for boundary-dependent transform

## Mapping from boundary type to transform

In JCTVC-G281 [1], it was observed that the inter prediction error was larger near the PU boundaries than in the middle of the PU, and a BDT method was proposed to adapt this uneven error distribution in a single PU. The proposed BDT method selects transform according to the boundary locations, which is summarized in Table 1. The proposed method is applied for inter-predicted luma blocks.

Table 1. Mapping from boundary type to transform (4-pt, 8-pt, and 16-pt trans.)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| TU Boundary | | Horizontal Trans. | TU Boundary | | Vertical Trans. |
| Left | Right | Top | Bottom |
| non-PU | PU | **T** | non-PU | PU | **T** |
| PU | non-PU | **Flip-T** | PU | non-PU | **Flip-T** |
| PU | PU | DCT | PU | PU | DCT |
| non-PU | non-PU | DCT | non-PU | non-PU | DCT |

In Table 1, **T** indicates a specific transform, which is defined in Table 2, and **Flip-T** denotes a transform using flipped transform matrix of **T**. The proposed method in Table 1 was only used for 4-point, 8-point and 16-point transforms, and **T** in Table 1 should be Flip-DST-VII or DCT-IV depending on transform sizes as illustrated in Table 2. F-DST-VII indicates a transform using flipped transform matrix of DST-VII. For 32-pt transform, **T** would be always DCT.

Table 2 Additional transform depending on transform sizes

|  |  |  |  |
| --- | --- | --- | --- |
|  | 4-pt trans. | 8-pt trans. | 16-pt trans. |
| **T** | F-DST-VII | DCT-IV | DCT-IV |

Fig. 1 shows two examples of BDT for different PU and TU partitions. The transform used for each TU can be derived according to Table 1 and Table 2.





Fig. 1. Examples of boundary-dependent transform

## Implementation of additional transform

Besides the existing transforms used in current HEVC, the additional transforms used in this scheme are 4-pt F-DST-VII, 8-pt and 16-pt DCT-IV, 8-pt and 16-pt F-DCT-IV. All of these additional transforms can be implemented by re-using the logic of the existing transforms in HEVC.

The forward and inverse F-DST-VII can be implemented by reordering the input and output of DST-VII respectively as shown in Fig. 2.

 

Fig. 2 4-pt F-DST-VII by using 4-pt DST-VII (left: forward, right: inverse)

The 8-pt and 16-pt DCT-IV are actually part of 16-pt and 32-pt DCT respectively. The N-pt DCT can be designed as shown in Fig. 3.

 

Fig. 3 N-pt DCT by using N/2-pt DCT and N/2-pt DCT-IV (left: forward, right: inverse)

Therefore, the 8-pt DCT-IV can share the partial logic of 16-pt DCT, and the 16-pt DCT-IV can share the partial logic of 32-pt DCT.

The forward and inverse F-DCT-IV (8-pt and 16-pt) can be implemented by reordering the input and output of DCT-IV respectively as shown in Fig. 4 (N is 8 or 16).

 

Fig. 4 N-pt F-DCT-IV by using N-pt DCT-IV (left: forward, right: inverse)

# Experimental Results

The proposed BDT method has been implemented on top of the HM5.0. Experiments are conducted according to the common test condition JCTVC-G1200 [2]. Table 3 shows the BD-rate performance. It can be seen that 0.2%~0.6% BD-rate gain for luma is achieved with similar run-time as that of anchor.

The marginal loss for chroma may be caused by the re-sue of luma RQT depth for chroma, which is actually disadvantageous for chroma and is currently being investigated in CE2. The proposed method increases the ratio of larger luma RQT depth, but this may result in further degradation of the chroma coding performance.

Table 3 BD-rate performance of the proposed method

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Random Access HE** | | | **Random Access LC** | | | **Random Access HE-10** | | |
|  | Y | U | V | Y | U | V | Y | U | V |
| Class A (8bit) | -0.5% | -0.1% | -0.3% | -0.6% | 0.2% | 0.1% | -0.2% | 0.0% | 0.1% |
| Class B | -0.2% | 0.2% | 0.2% | -0.3% | 0.5% | 0.3% | -0.2% | 0.3% | 0.2% |
| Class C | -0.2% | 0.0% | 0.0% | -0.3% | 0.2% | 0.2% |  |  |  |
| Class D | -0.3% | -0.1% | -0.2% | -0.3% | 0.4% | 0.3% |  |  |  |
| Class E |  |  |  |  |  |  |  |  |  |
| **Overall** | -0.3% | 0.0% | 0.0% | -0.3% | 0.3% | 0.3% | -0.2% | 0.2% | 0.2% |
|  | -0.3% | 0.0% | 0.0% | -0.3% | 0.3% | 0.2% | -0.2% | 0.1% | 0.2% |
| Class F | -0.1% | 0.0% | -0.1% | -0.1% | 0.1% | 0.1% |  |  |  |
| Enc Time[%] | 101% | | | 101% | | | 101% | | |
| Dec Time[%] | 101% | | | 100% | | | 100% | | |
|  |  |  |  |  |  |  |  |  |  |
|  | **Low delay B HE** | | | **Low delay B LC** | | | **Low delay B HE-10** | | |
|  | Y | U | V | Y | U | V | Y | U | V |
| Class A |  |  |  |  |  |  |  |  |  |
| Class B | -0.5% | 0.1% | 0.2% | -0.6% | 0.4% | 0.9% |  |  |  |
| Class C | -0.4% | -0.3% | -0.1% | -0.7% | 0.3% | 0.4% |  |  |  |
| Class D | -0.4% | -0.2% | 0.5% | -0.6% | 0.8% | 0.6% |  |  |  |
| Class E | -0.6% | 0.1% | -0.2% | -0.6% | 0.2% | 1.2% |  |  |  |
| **Overall** | -0.5% | -0.1% | 0.1% | -0.6% | 0.4% | 0.7% |  |  |  |
|  | -0.5% | -0.1% | 0.1% | -0.6% | 0.5% | 0.7% |  |  |  |
| Class F | -0.5% | -0.1% | -0.3% | -0.9% | -0.9% | -0.4% |  |  |  |
| Enc Time[%] | 101% | | | 102% | | |  | | |
| Dec Time[%] | 101% | | | 100% | | |  | | |

# Conclusion

In this proposal, the BDT method proposed in JCTVC-G281 is tested on top of HM5.0. 0.2%~0.6% BD-rate gain for luma can be achieved with similar runtime compared to HM5.0. The BD-rate gain is similar to that in HM4.0, which shows the stable character of the proposed method. It is suggested to include this boundary-dependent transform into HEVC.

# References

1. J. An, X. Zhao, X. Guo, and S. Lei, “Non-CE7: Boundary-Dependent Transform for Inter-Predicted Residue,” JCTVC-G281, Geneva, CH, Nov., 2011.
2. F. Bossen, “Common test conditions and software reference configurations”, JCTVC-G1200, Geneva, CH, Nov., 2011.

# Patent rights declaration(s)

**MediaTek Inc. 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).**

# Appendix: working draft modification



### Scaling and transformation process

Inputs to this process are:

– a luma location ( xT, yT ) specifying the top-left luma sample of the current transform unit relative to the top‑left luma sample of the current picture,

– a variable trafoDepth specifying the hierarchy depth of the current block relative to the coding unit,

– a variable cIdx specifying the chroma component of the current block,

– a variable nW specifying the width of the current transform block.

– a variable nH specifying the height of the current transform block.

* a lum location ( xB, yB ) specifying the top-left luma sample of the current transform unit relative to the top‑left luma sample of the current coding unit,
* a variable nCS specifying size of current coding unit.

Output of this process is:

– a modified version of the (nW)x(nH) array of residual samples r with elements rij.

The quantization parameter qP is derived as follows.

– If cIdx is equal to 0,

qP = QP’Y (8‑150)

– Otherwise,

qP = QP’C (8‑151)

The (nW)x(nH) array of residual samples r are derived as specified in the following ordered steps.

1. The inverse scanning process for transform coefficients as specified in subclause 8.5.2 is invoked with the width of the current transform unit nW, the height of the current transform unit nH, the list of transform coefficients transCoeffLevel[ xT ][ yT ][ trafoDepth ][ cIdx ] and the variable cIdx as the inputs and the output is a (nW)x(nH) array cij.
2. The scaling process for transform coefficients as specified in subclause 8.5.3 is invoked with the width of the transform unit nW, the height of the transform unit nH, the (nW)x(nH) array c, the chroma component variable cIdx and the quantization parameter qP as the inputs and the output is a scaled transform coefficient (nW)x(nH) array d.
3. The transformation process for scaled transform coefficients as specified in subclause 8.5.4 is invoked with the width of the transform unit nW, the height of the transform unit nH, the scaled transform coefficient (nW)x(nH) array d, the chroma component variable cIdx, the lum location ( xB, yB ), and the coding unit size nCS as the inputs and the output is a residual samples (nW)x(nH) array r.

### Transformation process for scaled transform coefficients

Inputs of 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 (nW)x(nH) array d of scaled transform coefficients with elements dij.

– a variable cIdx specifying the chroma component of the current block,

* a lum location ( xB, yB ) specifying the top-left luma sample of the current transform unit relative to the top‑left luma sample of the current coding unit,
* a variable nCS specifying size of current coding unit.

Output of this process is residual samples as a (nW)x(nH) array r with elements rij.

The variable trafoPrecisionExt is derived as follows:

– If nW is greater than 4 or nH is greater than 4,

* If cIdx is equal to 0 and bit\_depth\_luma\_minus8 is equal to 0, trafoPrecisionExt is set equal to 4.
* Otherwise, if cIdx is not equal to 0 and bit\_depth\_chroma\_minus8 is equal to 0, trafoPrecisionExt is set equal to 4.
* Otherwise, trafoPrecisionExt is set equal to 0.

– Otherwise, trafoPrecisionExt is set equal to 0.

[Ed.: (WJ) wait decision on transform precision extension]

[Ed: (WJ) derivation of trafoPrecisionExt is duplicated to that of the scaling process. Better to remove one of two.]

Depending on PredMode, IntraPredMode, and PartMode the following applies:

– If PredMode is equal to MODE\_INTRA, log2(nW\*nH) is equal to 4, and cIdx is equal to 0, the variables horizTrType and vertTrType are specified as Table 8‑11 with IntraPredMode as input. [Ed. (WJ): DST is applied only for luma 4x4 block]

– Otherwise, if PredMode is not equal to MODE\_INTRA, and cIdx is equal to 0, then the variables horizTrType and vertTrType are derived by invoking the process as specified in subclause 8.5.4.5 with the PartMode, xB,yB, nW, nH, and nCS as the inputs and the output are horizTrType and vertTrType.

– Otherwise, the variables horizTrType and vertTrType are set equal to 0.

Table 8‑14 – Specification of horizTrType and vertTrType

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **IntraPredMode** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| **horizTrType** | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| **vertTrType** | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **IntraPredMode** | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 |
| **horizTrType** | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **vertTrType** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |

The constructed residual samples are derived as specified in the following ordered steps.

1. The variable nS is derived as follows.

* If nW is greater than or equal to nH, nS is set to nW.
* Otherwise (nW is less than nH), nS is set to nH.

1. Each (horizontal) row of scaled transform coefficients dij (i=0..nW−1, j=0..nH−1) is transformed to eij (i=0..nW−1, j=0..nH−1) by invoking the one-dimensional transformation process as specified in subclause 8.5.4.1 to 8.5.4.4 according to the width of the transform unit nW, with the (nW)x(nH) array d and the transform type variable horizTrType as the inputs and the output is the (nW)x(nH) array e.
2. Each (vertical) column of the resulting matrix eij (i=0..nW−1, j=0..nH−1) is transformed to fij (i=0..nW−1, j=0..nH−1) by invoking the one-dimensional transformation process as specified in subclause 8.5.4.1 to 8.5.4.4 according to the height of the transform unit nH, with the (nW)x(nH) array e and the transform type variable vertTrType as the inputs and the output is the (nW)x(nH) array f.
3. The residual sample value rij is derived as follows.

* If nS is equal to 4 or 8,

rij = ( fij + 25+trafoPrecisionExt ) >> ( 6 + trafoPrecisionExt ), with i=0..(nW)−1, j=0..(nH)−1 (8‑159)

* Otherwise (nS is equal to 16 or 32),

rij = ( fij + 29+trafoPrecisionExt ) >> ( 10 + trafoPrecisionExt ), with i=0..(nW)−1, j=0..(nH)−1 (8‑160)

[Ed.: (WJ) dependent on transformation process – should be modified again after transformation process is fixed]

[Ed.: (BB) if modified, the condition on nS for NSQT should be checked]

#### Transformation process for 4 samples

Inputs of this process are an array of 4 samples of the scaled transform coefficients x with elements xi, with i = 0..3 and the transform type variable trType.

Output of this process is an array of 4 samples of the residual samples y with elements yi, with i = 0..3.

Depending on trType, the following applies:

– If tyType is equal to 1 or 3, the following ordered steps apply:

1. A set of intermediate values c0, c1, c2 and c3 is calculated as follows:

c0 = x0 + x2 (8‑159)  
c1 = x2 + x3 (8‑159)  
c2 = x0 – x3 (8‑159)  
c3 = 74 \* x1 (8‑159)

1. The output values yi with i = 0..3 are then specified as follows:

y0 = ( 29\*c0 + 55\*c1 + c3 + 64 ) >> 7 (8‑159)  
y1 = ( 55\*c2 – 29\*c1 + c3 + 64 ) >> 7 (8‑159)  
y2 = ( 74\*( x0 - x2 + x3 ) + 64 ) >> 7 (8‑159)  
y3 = ( 55\*c0 + 29 \* c2 – c3 + 64 ) >> 7 (8‑159)

[Ed. (WJ): proponent’s comment - to share the quantization scaling value, transform precision (7-bit in this case) should be synchronized with DCT]

– Otherwise, if tyType is equal to 2, the following ordered steps apply:

1. A set of intermediate values c0, c1, c2 and c3 is calculated as follows:

c0 = x0 + x2 (8‑159)  
c1 = x2 + x3 (8‑159)  
c2 = x0 – x3 (8‑159)  
c3 = 74 \* x1 (8‑159)

1. The output values yi with i = 0..3 are then specified as follows:

y3 = ( 29\*c0 + 55\*c1 + c3 + 64 ) >> 7 (8‑159)  
y2 = ( 55\*c2 – 29\*c1 + c3 + 64 ) >> 7 (8‑159)  
y1 = ( 74\*( x0 - x2 + x3 ) + 64 ) >> 7 (8‑159)  
y0 = ( 55\*c0 + 29 \* c2 – c3 + 64 ) >> 7 (8‑159)

[Ed. (WJ): proponent’s comment - to share the quantization scaling value, transform precision (7-bit in this case) should be synchronized with DCT]

– Otherwise, the following ordered steps apply: [Ed. (WJ): TBD – 4-point DCT]

#### 8.5.4.5 Transformation type derivation process

Inputs of this process are:

– a luma location ( xB, yB ) specifying the top-left luma sample of the current transform unit relative to the top‑left luma sample of the current coding unit,

– a variable nW specifying the width of the current transform unit,

– a variable nH specifying the height of the current transform unit,

– a variable PartMode specifying prediction partition mode of current coding unit,

– a variable nCS specifying size of current coding unit.

Output of this process are:

Variables horizTrType and vertTrType.

The variable leftBoundaryType is set to 1 if one of the following conditions is true; else the variable leftBoundaryType is set to 0.

* xB is equal to 0
* xB is equal to nCS>>2, and PartMode is equal to PART\_nLx2N
* xB is equal to nCS>>1, and PartMode is equal to PART\_Nx2N or PART\_NxN
* xB is equal to 3\*nCS>>2, and PartMode is equal to PART\_nRx2N

The variable rightBoundaryType is set to 1 if one of the following conditions is true; else the variable rightBoundaryType is set to 0.

* xB+nW-1 is equal to (nCS>>2)-1, and PartMode is equal to PART\_nLx2N
* xB+nW-1 is equal to (nCS>>1)-1, and PartMode is equal to PART\_Nx2N or PART\_NxN
* xB+nW-1 is equal to (3\*nCS>>2)-1, and PartMode is equal to PART\_nRx2N
* xB+nW-1 is equal to nCS-1

If the variable nW is equal to or larger than 32, the variable horizTrType is set to 0, else the variable horizTrType is derived as follow:

* If the leftBoundaryType is equal to 1 and rightBoundaryType is equal to 0, the variable horizTrType is set to 2,
* Else if the leftBoundaryType is equal to 0 and rightBoundaryType is equal to 1, the variable horizTrType is set to 3,
* Else the variable horizTrType is set to 0.

The variable topBoundaryType is set to 1 if one of the following conditions is true; else the variable topBoundaryType is set to 0.

* yB is equal to 0
* yB is equal to nCS>>2, and PartMode is equal to PART\_2NxnU
* yB is equal to nCS>>1, and PartMode is equal to PART\_2NxN or PART\_NxN
* yB is equal to 3\*nCS>>2, and PartMode is equal to PART\_2NxnD

The variable bottomBoundaryType is set to 1 if one of the following conditions is true; else the variable bottomBoundaryType is set to 0.

* yB+nH-1 is equal to (nCS>>2)-1, and PartMode is equal to PART\_2NxnU
* yB+nH-1 is equal to (nCS>>1)-1, and PartMode is equal to PART\_2NxN or PART\_NxN
* yB+nH-1 is equal to (3\*nCS>>2)-1, and PartMode is equal to PART\_2NxnD
* yB+nH-1 is equal to nCS-1

If the variable nH is equal to or larger than 32, the variable vertTrType is set to 0, else the variable vertTrType is derived as follow:

* If the topBoundaryType is equal to 1 and bottomBoundaryType is equal to 0, the variable vertTrType is set to 2,
* Else if the topBoundaryType is equal to 0 and bottomBoundaryType is equal to 1, the variable vertTrType is set to 3,
* Else the variable vertTrType is set to 0.