

JVET-O0307

Non-CE9/3: Unification of division processing

K. Abe, T. Toma, Y. Kato

Panasonic Corporation

Block level division processing in VVC

Tool	Explanation	Equation	Implementation
CCLM	parameter a calculation (1 time per CU)	$a = \frac{\max C - \min C}{\max Y - \min Y}$	LUT (size 16) method using the difference between numerator and denominator magnitude.
BDOF	v_x and v_y calculation (2 times per 4x4 subblock)	$v_x = -\frac{sG_{xdl}}{sG_{x2}}$ $v_y = \frac{sG_{ydl} - v_x \times sG_{xGy}}{sG_{y2s_2}}$	Round denominator to power of 2, and perform using only bit shift.
DMVR	parametric MV refinement (2 times per CU)	$dMvX = \frac{sad[3] - sad[5]}{sad[3] + sad[5] - sad[4] \times 2}$ $dMvY = \frac{sad[1] - sad[7]}{sad[1] + sad[7] - sad[4] \times 2}$	Repeat the comparison of numerator and denominator, and assign to -8 to +8 values.

Problem

The division processing of each tool is specified using different methods individually.

➡ **Propose a common division method for all tools.**

Definition of common division method

Equation : nume / denom

```
x = Floor( Log2( denom ) );  
normDiff = ( ( denom << 6 ) >> x ) & 63;  
x += ( normDiff != 0 ) ? 6 - frac : 5 - frac;  
a = ( nume * ( divTable64[ normDiff ] | 32 ) + offsetctl * 2x-1 ) >> x;
```

divTable64[]

= { 0, 31, 30, 29, 28, 27, 27, 26, 25, 24, 23, 23, 22, 21, 21, 20, 19, 19, 18, 17, 17, 16, 16, 15, 15, 14, 14,
13, 13, 12, 12, 11, 11, 10, 10, 9, 9, 9, 8, 8, 7, 7, 7, 6, 6, 6, 5, 5, 5, 4, 4, 4, 3, 3, 3, 2, 2, 2, 2, 1, 1, 1, 1, 0 }

frac : control parameter of fractional precision

= 5 // CCLM

= 0 // BDOF, DMVR

offsetctl : control parameter of rounding offset

= 1 // CCLM

= 0 // BDOF, DMVR

Size 64 LUT is proposed in order to simplify division processing and to keep enough accuracy especially for higher bit depth input images.

■ Apply to CCLM division

Equation : $a = \text{diffC} / \text{diffY}$

VTM5.0

```
x = Floor( Log2( diffY ) );
normDiff = ( ( diffY << 4 ) >> x ) & 15;
x += ( normDiff != 0 ) ? 1 : 0;
y = Floor( Log2( Abs ( diffC ) ) ) + 1;
a = ( diffC * ( divTable16[ normDiff ] | 8 ) + 2y-1 ) >> y;
k = ( ( 3 + x - y ) < 1 ) ? 1 : 3 + x - y;
a = ( ( 3 + x - y ) < 1 ) ? Sign(a) * 15 : a;
b = minC - ( ( a * minY ) >> k );
```

```
divTable16[ ]
= { 0, 7, 6, 5, 5, 4, 4, 3, 3, 2, 2, 1, 1, 1, 1, 0 }
```

Proposal

common division

```
x = Floor( Log2( diffY ) );
normDiff = ( ( diffY << 6 ) >> x ) & 63;
x += ( normDiff != 0 ) ? 1 : 0;
a = ( diffC * ( divTable64[normDiff] | 32 ) + 2x-1 ) >> x;
k = 5;
a = Clip3 ( -15 << (k - 1), 15 << (k - 1), a );
b = minC - ( ( a * minY ) >> k );
```

Table size is increased from 16 to 64,
but the processing can be simpler than VTM5.0,
coding performance is slightly better than VTM5.0.

Proposal

■ Apply to BDOF division

Equation : $V_x = sG_{xI} / sG_{x2}$

VTM5.0

$V_x = (sG_{xI} \ll 3) \gg \text{Floor}(\text{Log2}(sG_{x2}));$

Proposal

common division

```
x = Floor( Log2( sGx2 ) );  
normDiff = ( ( sGx2 << 6 ) >> x ) & 63;  
x += ( normDiff != 0 ) ? 6 : 5;  
Vx = ( sGxdI * ( divTable64[normDiff] | 32 )) >> x;
```

Slightly complex than VTM 5.0,
but additional gain is expected.

■ Apply to DMVR division

Equation : $dMvX = ((sad[3]-sad[5]) \ll 3) / (sad[3]+sad[5]-(sad[4] \ll 1))$

VTM5.0

```
nume = ( sad[3] - sad[5] ) << 4;
denom = (( sad[3] + sad[5] ) - ( sad[4] << 1 )) << 3;
if (denom == 0) dMvX = 0;
else {
    if (sad[3] == sad[4] ) dMvC = -8;
    else if (sad[5] == sad[4] ) dMvC = 8;
    else {
        signNume = ( nume < 0 ) ? 1 : 0;
        nume = abs( nume );
        quotient = 0; counter = 3;
        while ( counter > 0 ) {
            counter = counter - 1; quotient = quotient << 1;
            if ( nume >= denom ) {
                nume = nume - denom; quotient = quotient + 1;
            }
            denom = (denom >> 1);
        }
        dMvX = signNume ? - quotient : quotient;
    }
}
```

Proposal

```
nume = ( sad[3] - sad[5] ) << 3;
denom = ( sad[3] + sad[5] ) - ( sad[4] << 1 );
if (denom == 0) dMvX = 0;
else {
```

common division

```
x = Floor( Log2( denom ) );
normDiff = ( ( denom << 6 ) >> x ) & 63;
x += ( normDiff != 0 ) ? 6 : 5;
dMvX = ( nume * ( divTable64[normDiff] | 32 )) >> x;
```

```
}
```

The processing is simpler than VTM5.0,
and the coding loss is negligible.

Simulation results

Anchor : VTM-5.0

Test : Proposal (common division method for all tools)

	All Intra Main10					Random Access Main 10					Low delay B Main10				
	Y	U	V	EncT	DecT	Y	U	V	EncT	DecT	Y	U	V	EncT	DecT
Class A1	-0.04%	-0.30%	-0.02%	99%	102%	-0.06%	-0.65%	0.06%	101%	101%					
Class A2	-0.02%	-0.12%	0.00%	101%	101%	-0.01%	-0.19%	-0.12%	100%	101%					
Class B	0.00%	-0.06%	0.01%	99%	101%	0.00%	-0.06%	-0.19%	100%	101%	-0.01%	0.09%	-0.11%	100%	99%
Class C	0.00%	-0.09%	0.02%	100%	101%	0.00%	-0.11%	-0.21%	100%	101%	0.02%	0.05%	0.07%	100%	100%
Class E	-0.01%	-0.06%	-0.04%	99%	101%						0.00%	-0.03%	1.17%	100%	100%
Overall	-0.01%	-0.12%	0.00%	100%	101%	-0.01%	-0.22%	-0.13%	100%	101%	0.00%	0.04%	0.27%	100%	100%
Class D	-0.01%	0.07%	0.05%	100%	107%	0.15%	-0.12%	0.17%	101%	100%	0.03%	-0.18%	1.82%	100%	103%
Class F	-0.04%	-0.01%	0.13%	100%	100%	0.04%	-0.14%	0.10%	100%	101%	-0.11%	0.17%	-0.16%	99%	101%

Unification can be achieved without any coding loss.

- Propose a common division method for all block level processing.
Target tools : CCLM, BDOF, DMVR
- Size 64 LUT method is used.
- -0.01%, -0.01%, -0.00% for AI, RA, LDB on VTM-5.0.

We propose to adopt this unification into VVC.

Thanks KDDI for cross-checking of the proposal.