

---

**JVET-L0274**

---

**CE7: TRANSFORM COEFFICIENT  
CODING WITH REDUCED NUMBER OF  
REGULAR-CODED BINS**

---

# Overview

## Goals: Complexity reduction of transform coefficient coding

- Reduce maximum number of context-coded bins
- Reduction of bin-to-bit ratio (avoid padding)

## Main aspects

- Specify maximum number of bins (sig, par, gt1) for first pass over a subblock
- Specify maximum number of bins (gt2) for second pass over a subblock
- Simplify Rice parameter derivation (similar to HEVC) for remainder (third pass)
- Add 4th pass: Code complete absolute levels in bypass mode (modified Rice-Golomb)  
(for scan positions not included in first pass)
- Local neighbourhood is accessed only once
- 2 version: (A) Use same binarization as in VTM-2  
(B) Modify binarization (swap par\_flag and gt1\_flag) in first pass

## VTM-2:

```
for( k ... ) { // pass 1
    sig_flag[ k ]
    if( sig_flag[ k ] ) {
        par_flag[ k ]
        gt1_flag[ k ]
    }
}
```

```
for( k ... ) { // pass 2
    if( gt1_flag[ k ] )
        gt2_flag[ k ]
}
```

```
for( k ... ) { // pass 3 (EP)
    if( gt2_flag[ k ] )
        remainder[ k ]
}
```

## test 7.1.3a:

```
for( k ... ) { // max 32 bins
    sig_flag[ k ]
    if( sig_flag[ k ] ) {
        par_flag[ k ]
        gt1_flag[ k ]
    }
}
```

```
for( k ... ) { // max 2 bins
    if( gt1_flag[ k ] )
        gt2_flag[ k ]
}
```

```
for( k ... ) { // EP bins
    remainder[ k ]
}
```

```
for( k ... ) { // NEW: EP
    abs_level[ k ]
}
```

## test 7.1.3b:

```
for( k ... ) { // max 28 bins
    sig_flag[ k ]
    if( sig_flag[ k ] ) {
        gt1_flag[ k ]
        if( gt1_flag[ k ] )
            par_flag[ k ]
    }
}
```

```
for( k ... ) { // max 4 bins
    if( gt1_flag[ k ] )
        gt2_flag[ k ]
}
```

```
for( k ... ) { // EP bins
    remainder[ k ]
}
```

```
for( k ... ) { // NEW: EP
    abs_level[ k ]
}
```

New last pass: Bypass coded absolute levels (no data coded in first pass)

# Maximum number of bins in first pass (Version A)

```
regBins = 34 //maximum number of regular coded bins in first pass

for( k = startScanIdx; k <= endScanIdx && regBins >= 3; k++ ) {
    coeff[ k ] = 0
    if( sig_flag cannot be inferred to be equal to 1 ) {
        decode sig_flag[ k ]
        regBins = regBins - 1
    }
    if( sig_flag != 0 ) {
        decode par_flag[ k ]
        decode gt1_flag[ k ]
        regBins = regBins - 2
        coeff[ k ] = 1 + par_flag[ k ] + 2 * gt1_flag[ k ]
    }
}
```

# Maximum number of bins in first pass (Version B)

```
regBins = 28 //maximum number of regular coded bins in first pass

for( k = startScanIdx; k <= endScanIdx && regBins >= 3; k++ ) {
    coeff[ k ] = 0
    if( sig_flag cannot be inferred to be equal to 1 ) {
        decode sig_flag[ k ]
        regBins = regBins - 1
    }
    if( sig_flag != 0 ) {
        decode gt1_flag[ k ]
        regBins = regBins - 1
        coeff[ k ] = 1 + gt1_flag[ k ]
        if( gt1_flag[ k ] != 0 ) {
            decode par_flag[ k ]
            regBins = regBins - 1
            coeff[ k ] = coeff[ k ] + par_flag[ k ]
        }
    }
}
```

# Second and Third Pass

## Second and third pass

- Second pass: Up to 2 / 4 gt2\_flags for version A / B
- Third pass: Rice-Golomb code of remainder (for scan positions coded in first pass)

## Context modelling and Rice parameter derivation

- Same context index derivation as in VTM-2 for context-coded bins
- Rice parameter is updated similar to HEVC

```
ricePar = 0
for( k ... ) {
    decode remainder[ k ]
    if( ricePar < 3 && remainder[ k ] > ( 3 << ricePar ) - 1 )
        ricePar++
}
```

# Fourth Pass: Bypass Coding of Absolute Levels

## Modified Rice-Golomb code

- Same Rice-Golomb code as for remainder, but zero is moved to a different position
- Determination of Rice parameter `ricePar` and zero position `posZero`
  - Determine sum of absolute levels `sumAbs` in local template (same as for sig, ...)
  - Derive `ricePar` and `posZero` according to

```
ricePar = RiceParTable[ max(31, sumAbs) ]  
posZero = PosZeroTable[ max(0, state-1) ][ max(31, sumAbs) ]
```

- Decode temporary value `codeValue` using Rice-Golomb code with `ricePar`
- Derive value of `absLevel[ k ]` according to

```
if( codeValue == posZero )      absLevel[k] = 0  
else if( codeValue < posZero )  absLevel[k] = codeValue + 1  
else                           absLevel[k] = codeValue
```

# Update after Core Experiment

- Dedicated restriction for chroma 2x2 subblocks
  - Exactly same algorithm with modified maximum numbers
  - Maximum number of bins for first pass = 6
  - Maximum number of bins for second pass = 2
- Update of CABAC initialization tables for version B
  - Due to modified binarization, statistics are different
  - Updated initialization for sig\_flag, par\_flag, gtx\_flag's

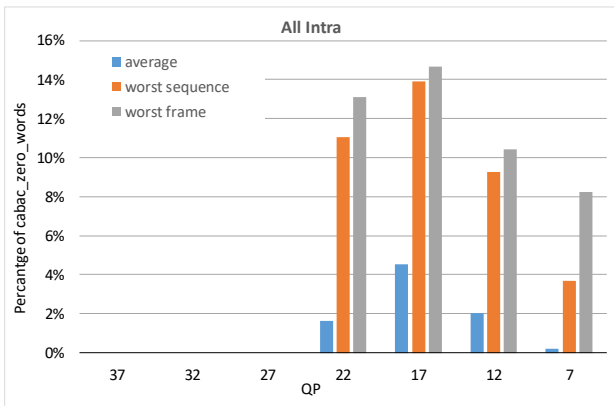


# Simulation Results

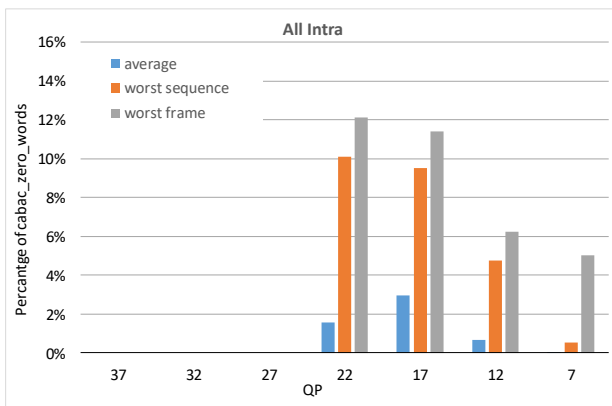
		high complexity (CTC)					low complexity (DQ off, RDOQ off, SDH on)				
		Y	U	V	encT	decT	Y	U	V	encT	decT
AI	7.1.3a	0.03%	0.13%	0.15%	110%	100%	0.02%	0.05%	0.05%	101%	100%
	7.1.3a (update)	0.05%	0.13%	0.21%	113%	104%	0.03%	0.12%	0.07%	100%	102%
	7.1.3b	-0.18%	-0.06%	0.00%	109%	101%	0.04%	0.08%	0.11%	101%	101%
	7.1.3b (update)	-0.20%	-0.11%	-0.03%	109%	102%	0.01%	0.05%	0.05%	101%	101%
RA	7.1.3a	0.03%	0.10%	0.16%	105%	100%	0.02%	0.09%	-0.03%	100%	100%
	7.1.3a (update)	0.05%	0.08%	0.13%	106%	101%	0.02%	0.12%	-0.05%	104%	101%
	7.1.3b	-0.07%	-0.02%	0.09%	104%	100%	0.08%	0.08%	-0.06%	100%	101%
	7.1.3b (update)	-0.14%	-0.03%	-0.06%	105%	104%	0.03%	0.04%	-0.07%	101%	102%
LB	7.1.3a	0.06%	0.14%	-0.16%	105%	105%	0.01%	0.08%	-0.24%	101%	103%
	7.1.3a (update)	0.08%	0.34%	0.05%	105%	101%	0.01%	-0.06%	-0.22%	100%	100%
	7.1.3b	-0.15%	-0.27%	-0.36%	104%	104%	0.00%	-0.03%	0.12%	100%	103%
	7.1.3b (update)	-0.26%	-0.16%	-0.31%	104%	102%	-0.07%	-0.14%	0.05%	102%	104%

# Bin-to-Bit Ratio and Padding

## VTM-2.0.1



## Version A



## Version B

