



KOREA UNIVERSITY

Scalable Video Coding

JVT-T038



Chun-Su Park

Korea Univ.

CMCP (<http://cmcp.korea.ac.kr>)

cspark@dali.korea.ac.kr

Contents

1. Review of EOB shift

2. Num_end_vals

1. Luma
2. Chroma

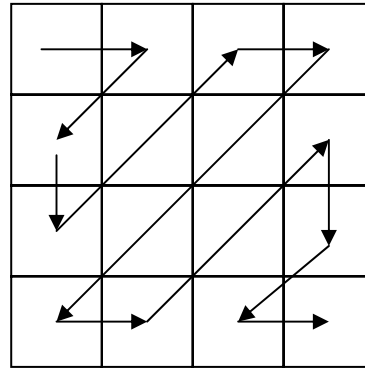
3. StartLevel

4. Index inconsistency

5. Summary

Review of EOB shift

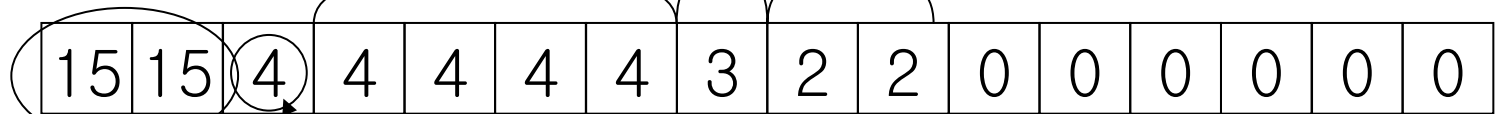
Luma 4x4



eobShiftLuma[x]

X: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

eob_run=4 1 2



```
(num_end_vals + 1) = 2
```

```
eobShiftLuma[num_end_vals]=4
```

Current syntax

pp. 382 in jvt-S201

```
num_end_vals  
num_end_vals = (num_end_vals + 1 ) % 4  
for( i = 0; i < num_end_vals; i++)  
    eobShiftLuma[i] = 15
```

Part 1

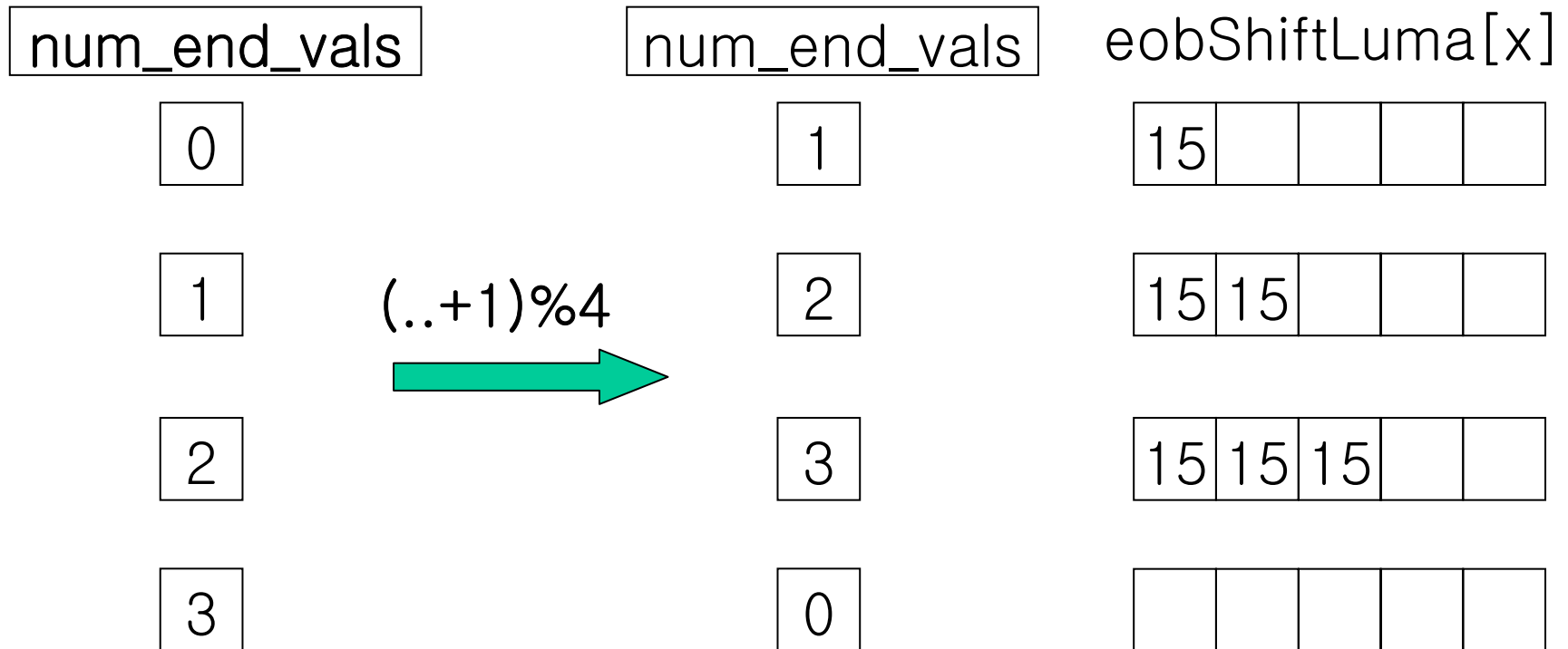
```
eobShiftLuma[num_end_vals]  
startLevel = eobShiftLuma[num_end_vals] - 1  
for( i = 0 ; i < 14 - num_end_vals && startLevel > 0; ) {  
    eob_run  
    for( j = 0 ; j < eob_run ; j++ )  
        eobShiftLuma[num_end_vals+1+i+j] = startLevel  
    startLevel--;  
    i += eob_run
```

Part 2

num_end_vals [1/6]

```
num_end_vals
num_end_vals = (num_end_vals + 1) % 4
for( i = 0; i < num_end_vals; i++)
    eobShiftLuma[i] = 15
```

Restrict the range
of num_end_vals
from 0 to 3



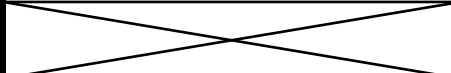
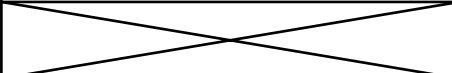
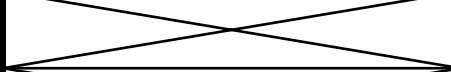
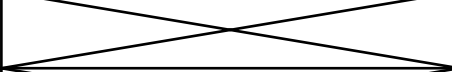
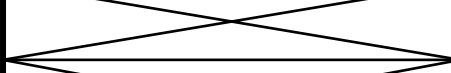
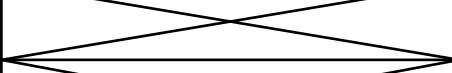
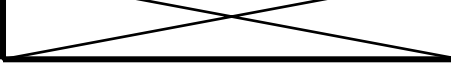
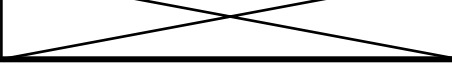
num_end_vals [2/6]

□ num_end_vals is parsed using te(v).

- te(v): If a syntax element is coded as te(v), the range of possible values for the syntax element is determined first. The range of this syntax element may be between 0 and x, with x being greater than or equal to 1 and the range is used in the derivation of the value of the syntax element values as follows
 - If x is greater than 1, codeNum and the value of the syntax element is derived in the same way as for syntax elements coded as ue(x)
 - If x is equal to 1, the parsing process for codeNum which is equal to the value of the syntax element is given by a process equivalent to:
 b = read_bits(1)
 codeNum = !b

num_end_vals [3/6]

te(v)

Max[x] = 1		2 ≤ Max[x]	
x	Bit string	x	Bit string
0	1	0	1
1	0	1	010
		2	011
		3	00100
		4	00101
		5	00110

num_end_vals = (num_end_vals + 1) % 4

→ In case of num_end_vals, Max(x) is always equal to 3!!

num_end_vals [4/6]

□ Change the parsing process for num_end_values to tu(v)

- tu(v)
 - For syntax element values less than cMax, the U binarization process is invoke. For the syntax element value equal to cMax the bin string is a bit string of length cMax with all bins being equal to 1.

syntax

te(v)

x	Bit string
0	1
1	010
2	011
3	00100

Max(x) = 3

change



software

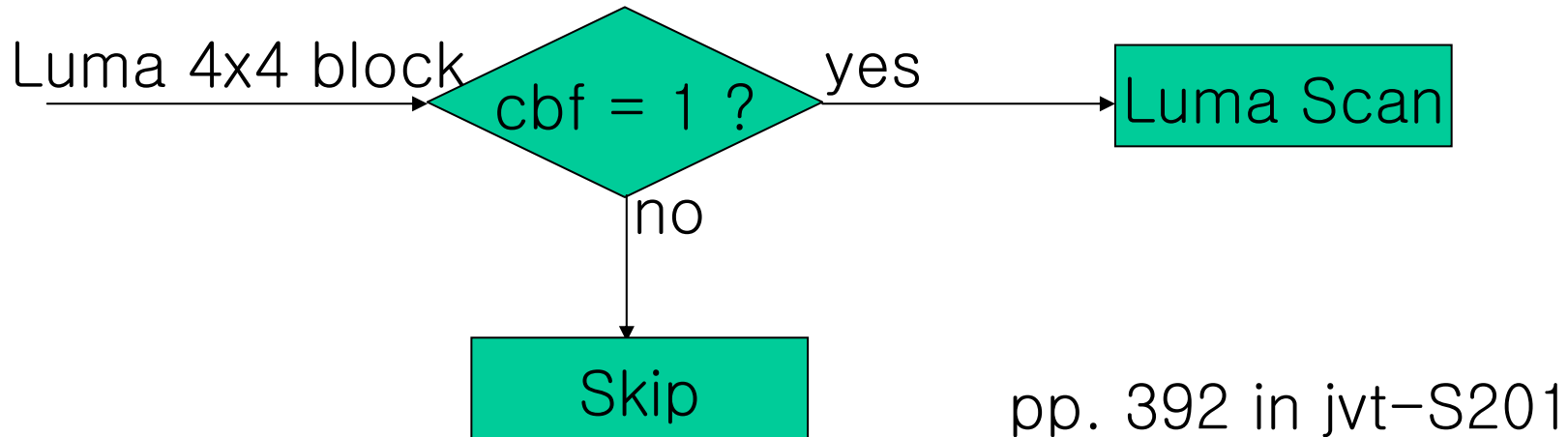
tu(v)

x	Bit string
0	0
1	10
2	110
3	111

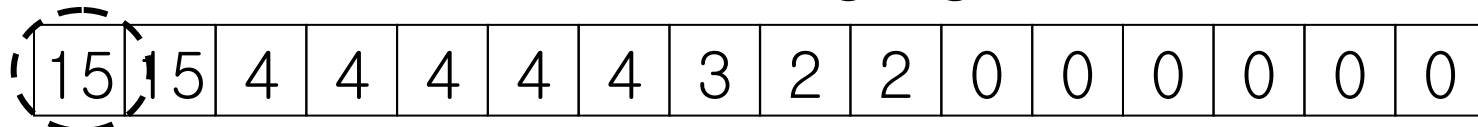
Max(x) = 3

Reference software uses tu(v) instead of te(v)!!

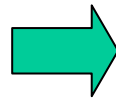
num_end_vals [5/6]



Luma 4x4 block in zigzag scan order



The first coefficient
in luma 4x4 block
can not have EOB
flag!!



eobShiftLuma[0] is always
15. This is also verified by
our experiments.

num_end_vals [6/6]

❑ Don't need to send the first value of eobShiftLuma!!

- Skip coding of the first value of eobShiftLuma.
- Decrease the maximum value of num_end_vals.
 - $\text{Max}(x) = 3 \rightarrow \text{Max}(x) = 2$

syntax

te(v)

x	Bit string
0	1
1	010
2	011
3	00100

$\text{Max}(x) = 3$

change



software

tu(v)

x	Bit string
0	0
1	10
2	110
3	111

$\text{Max}(x) = 3$

change



proposed

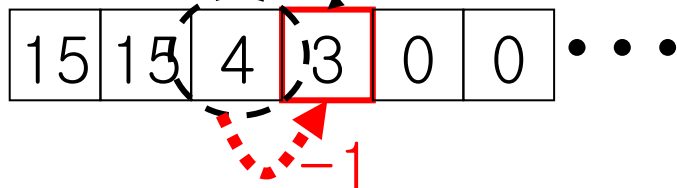
tu(v)

x	Bit string
0	0
1	10
2	11

$\text{Max}(x) = 2$

```
eobShiftLuma[num_end_vals]=4
```

startLevel

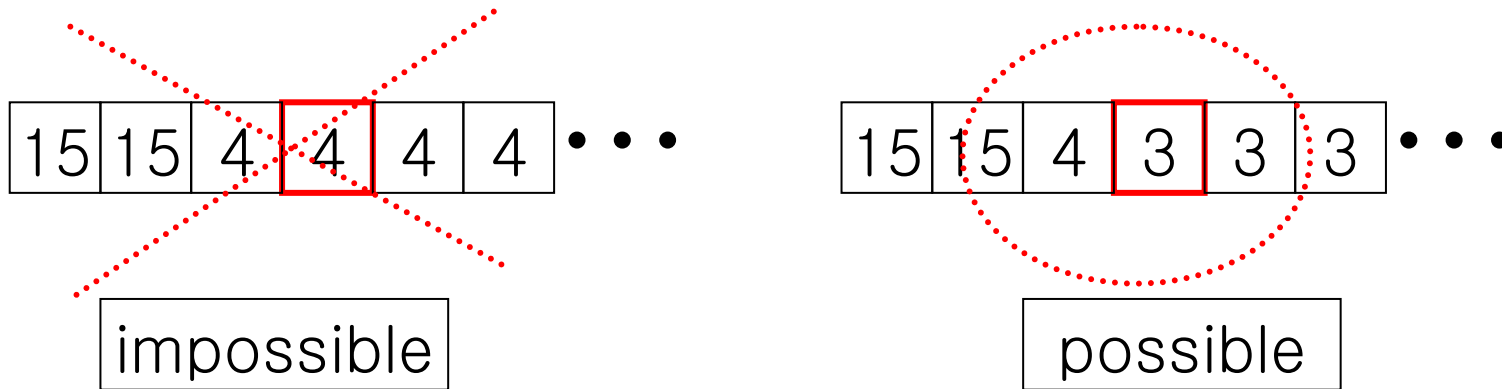


Problem!!

The startLevel can't have the value, eobShiftLuma[num_end_vals].

StartLevel [2/2]

□ Example



□ We propose that the line should be modified as follows:

```
startLevel = eobShiftLuma[num_end_vals] - 1
```

syntax



```
startLevel = eobShiftLuma[num_end_vals]
```

software

Initialization of eobShiftLuma [1/2]

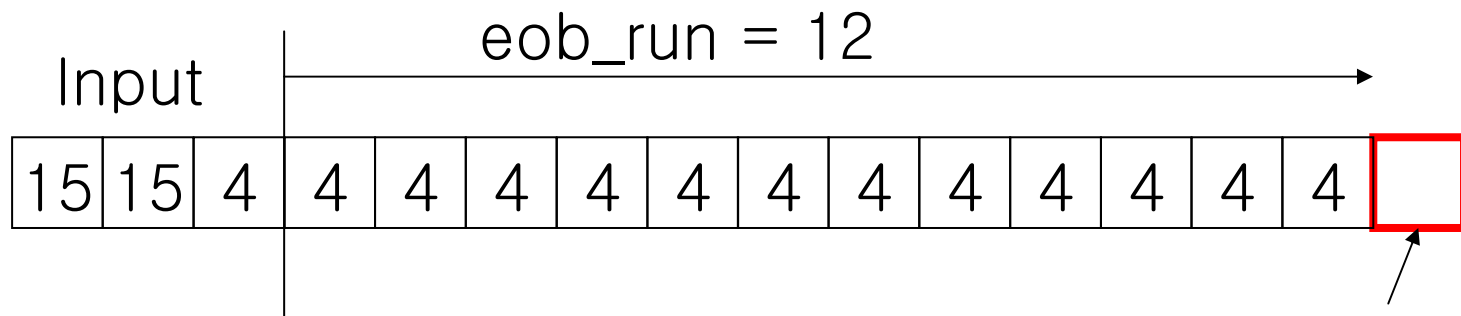
```
for( i = 0 ; i < 14 - num_end_vals && startLevel > 0; ) {  
    eob_run  
    for( j = 0 ; j < eob_run ; j++ )  
        eobShiftLuma[num_end_vals+1+i+j] = startLevel  
    startLevel--;  
    i += eob_run  
}
```

Example)

num_end_vals = 2, eob_run = 12

eobShiftLuma[num_end_vals]=4

End of loop
(i = 12 < 14 - 2)



Does not be initialized!!

Initialization of eobShiftLuma [2/2]

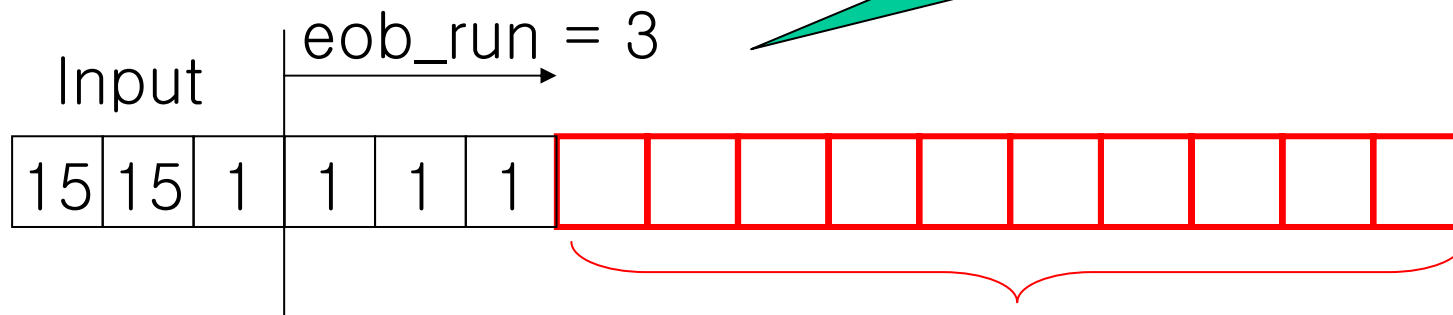
```

for( i = 0 ; i < 14 - num_end_vals && startLevel > 0; ) {
    eob_run
    for( j = 0 ; j < eob_run ; j++ )
        eobShiftLuma[num_end_vals+1+i+j] = startLevel
    startLevel--;
    i += eob_run
}

```

Example)

num_end_vals = 2, eob_run = 3
eobShiftLuma[num_end_vals]=1



Do not be initialized!!

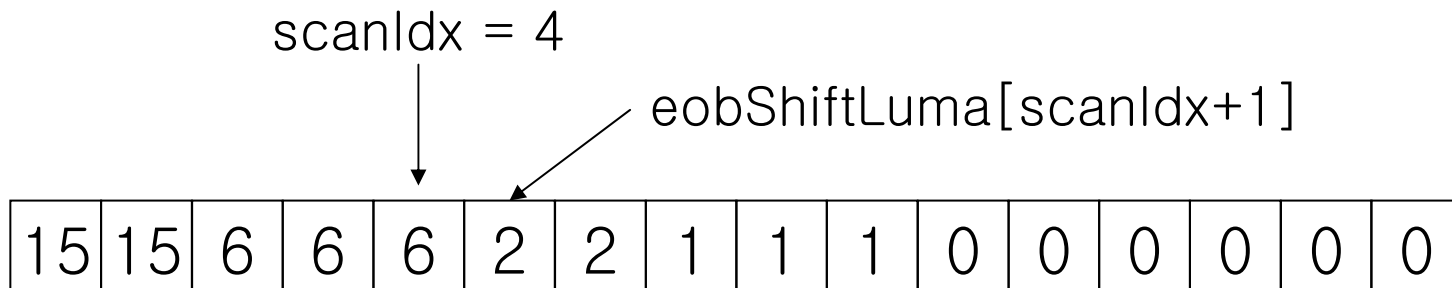
Index mismatching

luma_coefficient()

```
significant_coefficient_cavlc( scanIdx, scanIdxStart, numRemain,  
vlc_selector[ baseLastIdx + 16 * scanIdx ], eobShiftLuma[ scanIdx + 1 ],  
numCoef, trmCoef )
```

scanIdx: current scan index

eobShiftLuma[scanIdx+1]: indicates the EOB offset values for current scan index



Modified syntax for luma 4x4 blocks

```
num_end_vals
```

```
num_end_vals = num_end_vals % 3
```

```
for( i = 0; i < num_end_vals; i++)
```

```
    eobShiftLuma_idxMinus1[i] = 15
```

```
eobShiftLuma_idxMinus1[x]  
    = eobShiftLuma[x+1]
```

```
eobShiftLuma_idxMinus1[num_end_vals]
```

```
startLevel = eobShiftLuma_idxMinus1[num_end_vals]
```

```
for( i = 0 ; i < 14 - num_end_vals && startLevel > 0; ) {
```

```
    eob_run
```

```
    for( j = 0 ; j < eob_run ; j++ )
```

```
        eobShiftLuma_idxMinus1[num_end_vals+i+j+1] = startLevel
```

```
    startLevel--;
```

```
    i += eob_run
```

```
}
```

```
if( startLevel == 0 ) {
```

```
    for( ; i < 14 - num_end_vals ; i++ )
```

```
        eobShiftLuma_idxMinus1[num_end_vals+1+i] = 0
```

```
}
```

Chroma 4x4 blocks (1/3)

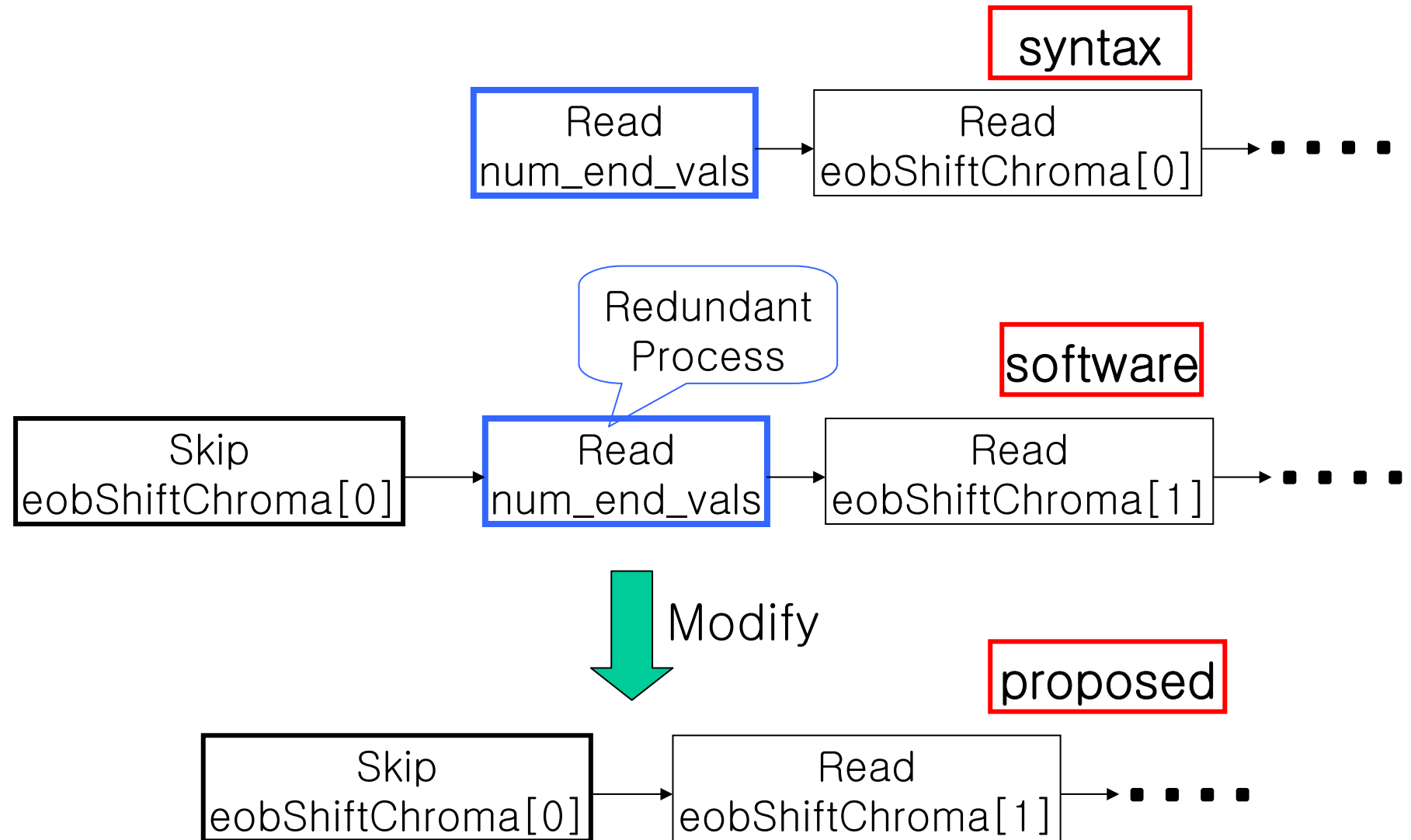
- ❑ When the coding of `eobShiftChroma[0]` is skipped, the `num_end_vals` is always 3.
- ❑ Since the `num_end_vals` is always 3, there is no element which has the maximum offset value.
- ❑ We propose that `num_end_vals` should not be coded in case of the chroma 4x4 block.

Chroma 4x4 blocks (2/3)

	num_end_vals = 0	1	2	3
City	0	0	0	100%
Crew	0	0	0	100%
Harbour	0	0	0	100%
Foreman	0	0	0	100%
Football	0	0	0	100%
Mobile	0	0	0	100%
Bus	0	0	0	100%
Soccer	0	0	0	100%

{QCIF, CIF, 4CIF}, 300 frames, 3 FGS layers

Chroma 4x4 blocks (3/3)



Modified syntax for chroma 4x4 blocks

```
eobShiftChroma_idxMinus1[0]
startLevel = eobShiftChroma_idxMinus1[0]
for( i = 0; i < 14 && startLevel > 0; ) {
    eob_run
    for( j = 0 ; j < eob_run ; j++ )
        eobShiftChroma_idxMinus1[1+i+j] = startLevel
    startLevel--;
    i += eob_run
}
if( startLevel == 0 ) {
    for( ; i < 14 ; i++ )
        eobShiftChroma_idxMinus1[1+i] = 0
}
```

eobShiftChroma_idxMinus1[x]
= eobShiftChroma[x+1]

Summary

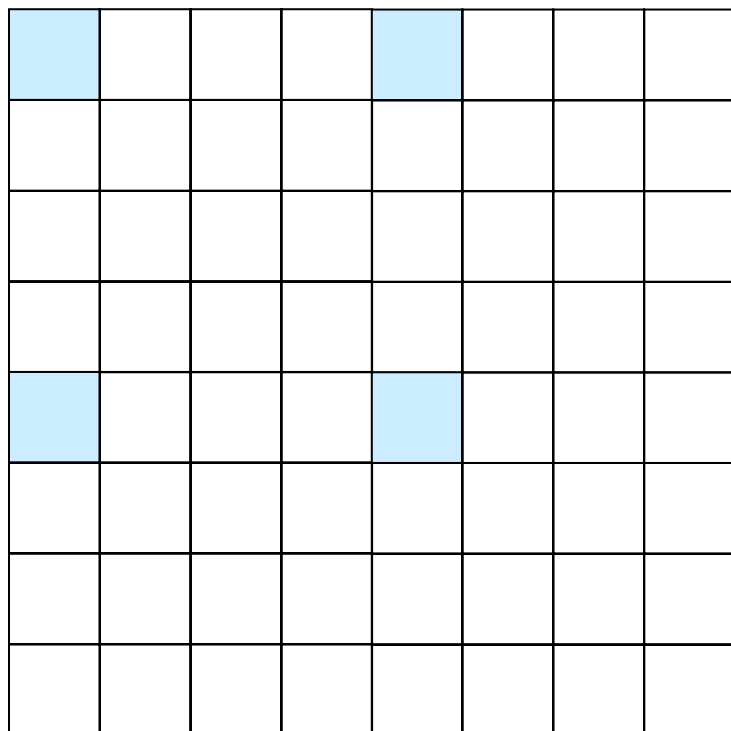
1. Change the entropy coding method for num_end_vals from te(v) to tu(v). (Luma)
2. Skip coding of the first value of eobShiftLuma. (Luma)
3. Change the initial value of startLevel. (Luma, Chroma)
4. Confirm the initialization process. (Luma, Chroma)
5. Skip coding of num_end_vals. (Chroma)
 - Experimental Results (syntax_vs_proposed.xls)
 - Comparison (T036 vs T038)

Bug in jsvm software

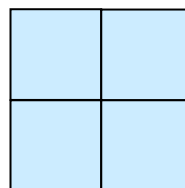
- ❑ In syntax, `eobShiftLuma[num_end_vals]` and `eob_run` are parsed using `ue(v)`. But, in the software, the parsing method of `eobShiftLuma[num_end_vals]` and `eob_run` is not `ue(v)`. We propose to fix the bug in the jsvm software.
 - Experimental result (`ue_vs_not_ue.xls`)

Appendix

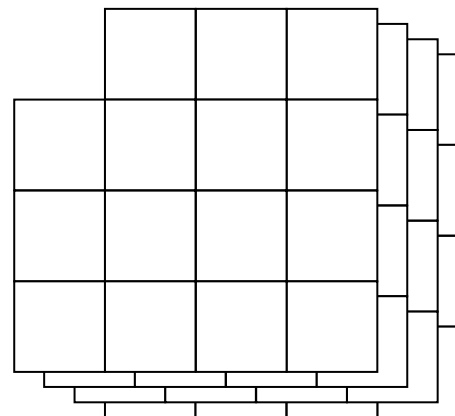
Chroma 8x8 block



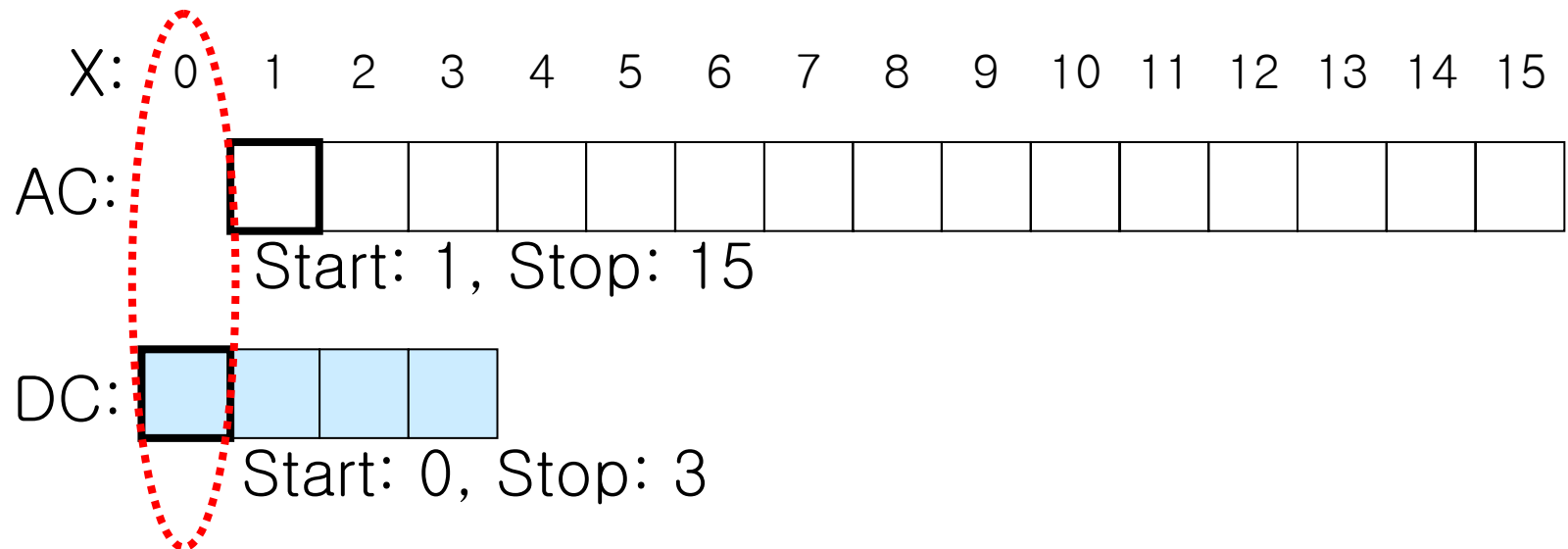
DC 2x2 block



AC 4x4 blocks without DC



eobShiftChroma[x]



`eobShiftChroma[0]` is not required in decoding process.

- We propose to skip the encoding and the decoding of the first value of the `eobShiftChroma`, `eobShiftChroma[0]`.