|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11**  16th Meeting: San Jose, CA, 9 Jan. – 17 Jan. 2014 | Document: JCTVC-Oxxxx | | Document: JCTVC-P0160 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **Non-RCE4: Palette prediction for palette coding** | | |
| *Status:* | Input Document to JCT-VC | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Guoxin Jin, Ankur Saxena and Felix Fernandes | Email: Tel: | [guoxin.jin@sta.samsung.com](mailto:guoxin.jin@sta.samsung.com),  [asaxena@sta.samsung.com](mailto:asaxena@sta.samsung.com),  [ffernandes@sta.samsung.com](mailto:ffernandes@sta.samsung.com)  1-972-761-7761 |
| *Source:* | Samsung Electronics, Co., Ltd. | | |

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

# Abstract

Palette coding for screen content was proposed in JCTVC-O0218. It was reported in JCTVC-O0218 that palette coding can significantly improve the coding performance for screen content. In JCTVC-O0218, there was no palette prediction considered. In JCTVC-P0198, a palette prediction technique which tries to re-use the palettes from left adjacent CU is presented. In this contribution, two improved palette prediction algorithms are proposed. The first prediction technique uses the palette from CUs to the left; and the second uses the palette from CUs in the current LCU. Simulation results tests show up to 3.0 % average bit-rate gain for lossless configurations and 2.5 % gain for lossy configurations on top of HM12.0+RExt5.1+JCTVC-O0218 .

# Introduction

Palette coding proposed in JCTVC-O0218 [1] achieves significant compression gains for screen content, as screen content is likely to have similar regions locally. In the ongoing RCE4 [2], in JCTVC-P0198, a palette prediction algorithm is used, where the palette of adjacent left CU is re-ordered and re-used for the current CU if available.

However, if the adjacent left CU is not coded by palette coding, there is no palette available. As a result, current CU couldn’t predict palette anymore. In this contribution, we propose two schemes which build upon the work in JCTVC-O0218 and JCTVC-P0198, and overcome this issue.

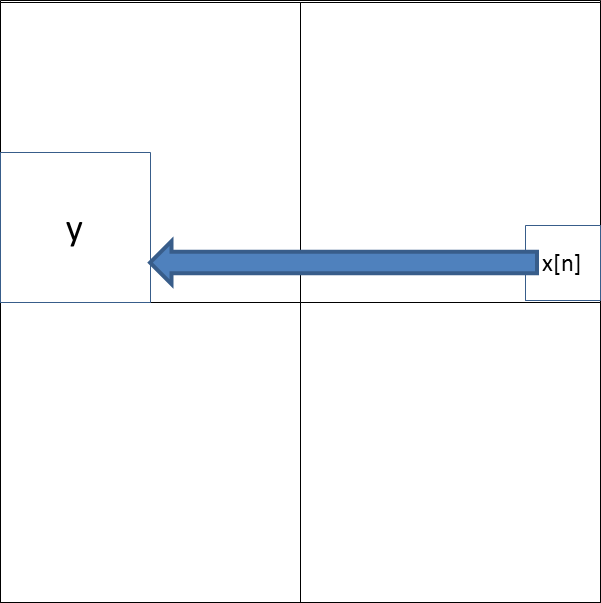
# Scheme A: Predicting Palette by Searching Left CUs

In screen content video, the colors in the neighboring regions are close with high probability. Moreover, for the document type of sequence, such as Word Document, the number of colors in the palettes is small. This implies that the previously encoded palette can be a good candidate for predicting current palette. The bits used in the encoding of the palette are quite large. Let the current CU be represented by x[n] where n is the zig-zag order of encoding. x[n-1] represents the previous encoded CU which may not be located at the left of x[n] spatially. xL[n] is the left neighboring CU and xU[n] is the above neighboring CU of x[n] respectively. Note that the sizes of neighboring CUs may be different to x[n] and depend on the optimal code modes selected by Rate-Distortion Optimizer (RDO).

In this scheme, only the CU’s located to the left of the current CU is considered, since no data is required to signaling the location of the left CU. Depending on the search range, there are two scheme of the prediction: within Large Coding Unit (LCU) (Scheme A.1), within LCU and left LCU (Scheme A.2).

**2.1. SchemeA.1 : Predicting Palette from Left CU which resides in the same LCU**

In current LCU, during encoding of CU x[n], the encoder keeps on searching to the left until it finds the first palette coded CU, y. Then the palette of y will be used directly as the palette of x[n]. This procedure can be seen in Figure 1. If there are no CUs which are palette coded in the LCU, x[n] will compute its own palette. Table 1 shows the scheme in an algorithmic form for both the encoder, and decoder.



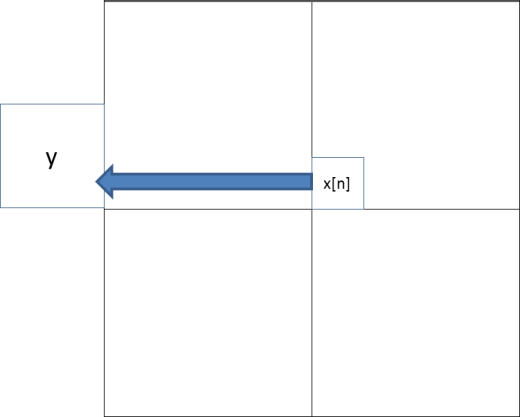
**Figure 1: Re-using left Palette in LCU**

**Table 1: Palette Prediction from Left CU in LCU**

|  |
| --- |
| **Encoder**  Given a coding unit x[n]:  y = getLeftCU( x[n] )  while (y is a valid CU and y is in the same LCU as x[n])  if (y is a palette coded CU)  encode 1 for indicating there is a prediction  copy palette from y to x[n]  do palette coding for x[n] as in [1]  return  else  y = getLeftCU(y)  endif  endwhile  encode 0 for indicating there isn’t a prediction  compute palette of x[n] as in [1]  encode palette of x[n] as in [1]  do palette coding for x[n] as in [1]  return  end |
| **Decoder**  Given a coding unit x[n]  read 1 bit  if bit is 0  decode palette of x[n] as in [1]  decode x[n] as in [1]  return  else  y = getLeftCU( x[n] )  while (y is a valid CU and y is in the same LCU as x[n])  if (y is a palette coded CU)  copy palette from y to x[n]  decode x[n] as in [1]  return  else  y = getLeftCU(y)  endif  endwhile  endif  end |

**2.2 SchemeA.2: Predicting Palette from Left CU which resides in the same LCU and left LCU**

In Scheme A.1, the CUs on the left boundary can never be predicted. In order to fully utilize the benefit of Scheme A, CUs coded as palette coding in left LCU would also be considered. An example of Scheme A.2 is shown in Figure 2. If there is no CU coded by palette coding, x[n] will compute its own palette as the same as it does in Scheme A.1. Table 2 shows the scheme in an algorithmic form for both the encoder, and decoder.



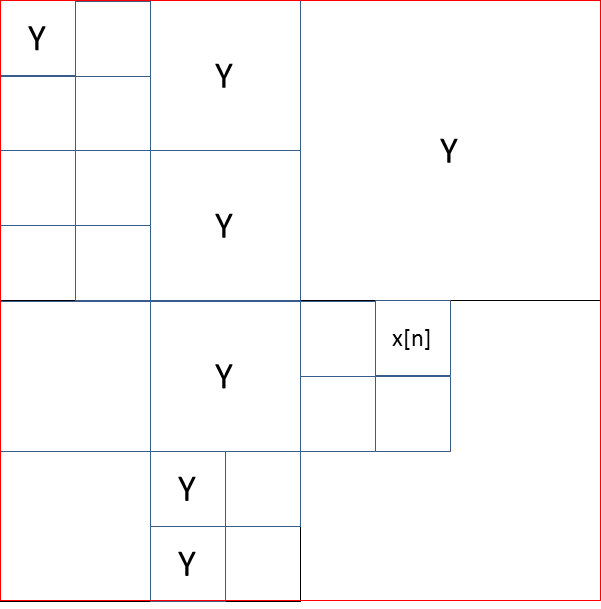
**Figure 2: Reuse left Palette in left LCU**

**Table 2: Palette Prediction from Left CU in LCU and left LCU**

|  |
| --- |
| **Encoder**  Given a coding unit x[n]:  y = getLeftCU( x[n] )  while (y is a valid CU and y is in the same LCU as x[n] or y is in the left LCU of x[n])  if (y is a palette coded CU)  encode 1 for indicating there is a prediction  copy palette from y to x[n]  do palette coding for x[n] as in [3]  return  else  y = getLeftCU(y)  endif  endwhile  encode 0 for indicating there isn’t a prediction  compute palette of x[n] as in [3]  encode palette of x[n] as in [3]  do palette coding for x[n] as in [3]  return  end |
| **Decoder**  Given a coding unit x[n]  read 1 bit  if bit is 0  decode palette of x[n] as in [3]  decode x[n] as in [3]  return  else  y = getLeftCU( x[n] )  while (y is a valid CU and y is in the same LCU as x[n] or y is in the left LCU of x[n])  if (y is a palette coded CU)  copy palette from y to x[n]  decode x[n] as in [3]  return  else  y = getLeftCU(y)  endif  endwhile  endif  end |

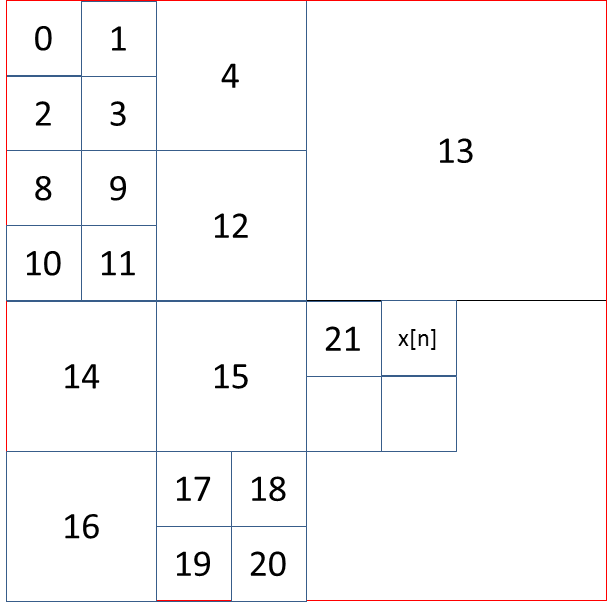
# Scheme B: Predicting Palette by Searching LCU

Scheme A considers predicting palettes from the left region only. However, there may be better palettes encoded in the LCU which are located at other position. Scheme B searches all the palette-coded CUs in current LCU and uses the one which is the best (in the sense of RD cost). An example is shown in Figure 3. For CU x[n], all the previous palette coded CUs (marked as Y) will be examined.



**Figure 3: Example: Predict Palette in LCU**

Scheme B requires signaling which palette has been reused. Since only the CUs in LCU are checked (this can be trivially extended to all palette-coded CUs in a slice, or frame), the zig-zag order index (which is a 1-dimension index) can be used to indicate the previous palette location. The demonstration of zig-zag order index corresponds to the example in Figure 3 is shown in Figure 4.



**Figure 4: Example: zig-zag order for n=22**

Given current CU index as n, the previous CU index as m. The zig-zag index is then differently coded as k = (n-m-1). Since n is always greater than m, this value is always greater than zero. For example, when n=22, if the best palette is the palette in CU x[15], then k = (22-15-1)=6.

For each CU x[n], scheme B searches all the palette coded CUs y[m] in the same LCU. Then the best y[m] (in the sense of Rate-Distortion cost) will be selected and k = (n-m-1) will be signaled.

The algorithm of Palette Prediction in LCU is shown in Table 3.

**Table 3: Scheme B: Palette Prediction in LCU**

|  |
| --- |
| Encoder  Given a coding unit x[n]:  y[m] = getPreviousCUInZorder(x[n])  bestY = n  RDcost = infinity;  while (y[m] is a valid CU in LCU)  if (y[m] is a palette coded CU)  compute RD cost RDnew of :  encode 1 for indicating there is a prediction  encode k = n-m-1  copy palette from y to x[n]  do palette coding for x[n] as in [3]  end compute RDnew  if RDnew < RDcost  bestY = m  RDcost = RDnew  endif  else  y = getPreviousCUInZorder(y)  endif  endwhile  if bestY==n  encode 0 for indicating no prediction  compute palette x[n]  encode palette of x[n] as in [3]  else  encode 1 for indicating there is a prediction  encode k = n-m-1  copy palette from y to x[n]  endif  do palette coding for x[n] as in [3]  return  end |
| Decoder  Given a coding unit x[n]  read 1 bit  if bit is 0  decode palette of x[n] as in [3]  decode x[n] as in [3]  return  else  decode k  m = n-k-1  y = getCU(m)  copy palette from y to x[n]  decode x[n] as in [3]  return  endif  end |

In this contribution, we used Exponential Golomb code of order 3 for coding the index of the CU block used for prediction.

# Rate distortion Mode Selection of Palette Coding with Palette Prediction

The above scheme A.1, A.2 or B can replace the current Palette Coding mode in [1]. However, the benefits of Palette Reusing can add if Palette Coding and Palette Reusing are mixed. So for each CU, both Palette Coding and Palette Reusing are checked by Rate-Distortion Optimizer and the best of which is used in the coding process. This algorithm is shown in Table 4.

**Table 4: Palette Coding and Palette Reusing in RDO**

|  |
| --- |
| **Encoder**  Given a coding unit x[n]:  compute the RDCost of palette coding as in [3], denote by d1  compute the RDCost of Scheme A.1, Scheme A.2 or Scheme B, denote by d2  if ( no palette can be predicted from previous CUs)  encode 1 bit 0 to indicate use palette coding in [1]  else  if (d1<d2)  encode 1 bit 0 to indicate use palette coding in [3]  else  encode 1 bit 1 to indicate use proposed palette prediction  (depends on which scheme been used)  endif  end if  end |
| **Decoder**  Given a coding unit x[n]  decode 1 bit  if bit is 0  decode as in [3]  else  decode as proposed depending on which scheme used in the encoder  endif  end |

If there are other modes except of palette coding mode (such as in HEVC12.1+RExt5.1, there are Angular prediction mode, Intra Block Copy and so on), proposed schemes can be one additional mode. Each node will be tested by RDO and the best mode of them is selected. So the bits used to indicate the proposed mode is:

**Table 5: Palette Coding Mode Bits**

|  |  |
| --- | --- |
| Mode | Bit |
| No Palette Coding | 0 |
| Palette Coding as in [3], encode current palette | 10 |
| Palette Prediction as proposed | 11 |

# Test Results

Here we present results for various schemes. There was a bug in the results in our previous v1 of the contribution, so we delete those results, and present the correct “bug-fix-ed” results. Various results which are presented are as follows:

**Tables 6 and 7:** Lossless and Lossy results for Scheme A.1 on top of RExt 5.1 + JCTVC-O0218 (implemented in RExt 5.1 software) as Anchor

**Tables 8 and 9**: Lossless and Lossy results for Scheme A.2 on top of RExt 5.1 + JCTVC-O0218 (implemented in RExt 5.1 software) as Anchor

**Tables 10 and 11**: Lossless and Lossy results for Scheme A.2 on top of RExt 5.1 + JCTVC-P0198 as Anchor

**Tables 12 and 13**: Lossless and Lossy results for Scheme A.2 (without any restriction of LCU being in a range) on top of RExt 5.1 + JCTVC-P0198 as Anchor

**Table 6: Lossless results for Scheme A.1 on top of RExt 5.1 + JCTVC-O0218 (implemented in RExt 5.1 software) as Anchor**



**Table 7: Lossy results for Scheme A.1 on top of RExt 5.1 + JCTVC-O0218 (implemented in RExt 5.1 software) as Anchor**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **BD-rate Y** | **AI-MT** | **AI-HT** | **AI-SHT** | **RA-MT** | **RA-HT** | **LB-MT** | **LB-HT** |
| Class F | -0.2% | -0.2% | -0.2% | -0.1% | -0.1% | 0.0% | 0.0% |
| Class B | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% |
| SC RGB 444 | -0.8% | -1.2% | -1.6% | -0.8% | -1.1% | -0.6% | -0.7% |
| Animation RGB 444 | 0.0% | 0.0% | 0.0% | 0.0% | 0.1% | 0.0% | 0.0% |
| SC YUV 444 | -0.6% | -1.0% | -1.5% | -0.8% | -1.2% | -0.4% | -0.7% |
| Animation YUV 444 | 0.0% | 0.0% | 0.0% | -0.1% | 0.0% | 0.0% | 0.0% |
| RangeExt | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% |
| SC(444) GBR Optional | -2.9% | -3.6% | -4.2% | -2.3% | -3.0% | -1.6% | -2.4% |
| SC(444) YUV Optional | -2.5% | -3.2% | -3.7% | -1.9% | -2.5% | -1.3% | -1.5% |
| Enc Time[%] | 101% | 101% | 101% | 100% | 100% | 100% | 100% |
| Dec Time[%] | 100% | 100% | 100% | 101% | 100% | 101% | 101% |

**Table 8: Lossless results for Scheme A.1 on top of RExt 5.1 + JCTVC-O0218 (implemented in RExt 5.1 software) as Anchor**



**Table 9: Lossy results for Scheme A.1 on top of RExt 5.1 + JCTVC-O0218 (implemented in RExt 5.1 software) as Anchor**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **BD-rate Y** | **AI-MT** | **AI-HT** | **AI-SHT** | **RA-MT** | **RA-HT** | **LB-MT** | **LB-HT** |
| Class F | -0.4% | -0.4% | -0.4% | -0.3% | -0.2% | -0.1% | 0.0% |
| Class B | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% |
| SC RGB 444 | -3.1% | -4.0% | -4.5% | -3.3% | -3.9% | -1.8% | -2.3% |
| Animation RGB 444 | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% |
| SC YUV 444 | -2.5% | -3.8% | -4.6% | -2.5% | -3.7% | -1.5% | -2.5% |
| Animation YUV 444 | 0.0% | 0.0% | 0.0% | -0.1% | 0.0% | 0.0% | 0.0% |
| RangeExt | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% |
| SC(444) GBR Optional | -7.0% | -8.2% | -9.2% | -5.8% | -7.0% | -3.6% | -5.3% |
| SC(444) YUV Optional | -6.5% | -7.8% | -8.6% | -5.5% | -6.4% | -4.1% | -4.5% |
| Enc Time[%] | 102% | 103% | 103% | 100% | 101% | 101% | 101% |
| Dec Time[%] | 101% | 101% | 101% | 100% | 100% | 100% | 100% |

**Table 10: Lossless results for Scheme A.2 on top of RExt 5.1 + JCTVC-P0198 (implemented in RExt 5.1 software) as Anchor**



**Table 11: Lossy results for Scheme A.2 on top of RExt 5.1 + JCTVC-P0198 (implemented in RExt 5.1 software) as Anchor**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **BD-rate Y** | **AI-MT** | **AI-HT** | **AI-SHT** | **RA-MT** | **RA-HT** | **LB-MT** | **LB-HT** |
| Class F | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% |
| Class B | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% |
| SC RGB 444 | -0.9% | -1.1% | -1.1% | -0.8% | -0.8% | -0.2% | -0.4% |
| Animation RGB 444 | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% |
| SC YUV 444 | -0.8% | -1.0% | -1.1% | -0.8% | -0.9% | -0.2% | -0.3% |
| Animation YUV 444 | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% |
| RangeExt | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% |
| SC(444) GBR Optional | -2.8% | -2.9% | -3.2% | -1.9% | -2.2% | -1.0% | -1.0% |
| SC(444) YUV Optional | -2.6% | -2.9% | -3.1% | -2.0% | -2.1% | -1.0% | -1.6% |
| Enc Time[%] | 100% | 100% | 100% | 100% | 100% | 100% | 100% |
| Dec Time[%] | 100% | 100% | 100% | 99% | 99% | 100% | 100% |

**Table 12: Lossless results for Scheme A.2 (without any restriction of LCU being in a range) on top of RExt 5.1 + JCTVC-P0198 (implemented in RExt 5.1 software) as Anchor**



**Table 13: Lossy results for Scheme A.2 (without any restriction of LCU being in a range) on top of RExt 5.1 + JCTVC-P0198 (implemented in RExt 5.1 software) as Anchor**



# Conclusion

The proposed palette prediction can achieve up to 3.0% average bit-rate gain for lossless configurations and 2.5% gain for lossy configurations compared to JCTVC-O0218 implemented in RExt5.1 software for SC YUV 444 content. We therefore recommend adopting this proposal in committee draft of HEVC range extensions if palette prediction is adopted.

# References

1. L. Guo, M. Karczewicz, J. Sole and R. Joshi, “Evaluation of palette mode coding on HM-12.0+RExt-4.1”, JCTVC-O0218, Geneva, Switzerland, October 2013.
2. L. Guo, X. Guo and A. Saxena, “HEVC Range Extensions Core Experiment 4 (RCE 4): Palette coding for screen content”, JCTVC-O1124, Geneva, Switzerland, October 2013.
3. L. Guo et al., “RCE4: Results of Test 2 on Palette Mode for Screen Content Coding”, JCTVC-P0198, San Jose, USA, Jan 2014.

# Patent rights declaration(s)

**Samsung Electronics Co., Ltd., may have IPR 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)**