ITU - Telecommunications Standardization Sector
STUDY GROUP 16 Question 6
Video Coding Experts Group (VCEG)
35th Meeting: Berlin, Germany, 16-18 July, 2008

Document VCEG-AI33
Filename: VCEG-AI33.doc

Question: Q.6/SG16 (VCEG)

Source: Yan Ye, Peisong Chen, Marta Karczewicz
5775 Morehouse Dr
San Diego, CA 92121
USA

Tel: +1-858-8455080
+1-848-6585673
Email: martak@qualcomm.com

Title: **High Precision Interpolation and Prediction**

Purpose: Proposal

Abstract

The proposed method increases precision of interpolation and bi-directional prediction during motion compensated prediction. The average improvement obtained for 720p sequences is 7.7% for IPP configuration, 4.8% for IBBP configuration and 4.9% for hierarchical B configuration.

Introduction

In the interpolation process defined in H.264, pixels at 1/4-pixel positions are interpolated by using pixels at half pixel positions after they have been rounded and clipped. Moreover, during bilinear filtering, which is used to obtain the 1/4-pixel values, upward rounding is always used. Such frequent and biased rounding effectively reduces precision of the interpolation filtering. In [1] it was proposed to postpone rounding and clipping until the last step not to compromise the final precision. In this contribution we demonstrate that after this modification the filtering can still be performed using 16 bit arithmetic. In addition we address rounding accumulation in case of bidirectional prediction.

High Precision Interpolation

When deriving pixels at positions pixels at 1/4-pixel positions labeled as a, c, d, n, f, i, k, and q (Figure 1) we postpone the shift, rounding and clipping till the last step:

```
a = (G<<5+b1+32)>>6
c = (H<<5+b1+32)>>6
d = (G<<5+h1+32)>>6
n = (M<<5+h1+32)>>6
f = (j1>>5+b1+32)>>6
i = (j1>>5+h1+32)>>6
k = (j1>>5+m1+32)>>6
l = (j1>>5+s1+32)>>6
```

where the intermediated values denoted as b1, h1, m1 and s1 are derived by applying the 6-tap filter vertically and horizontally, e.g.:

```
b1 =E-5*F+20*G+20*H-5*I+J.
h1 =A-5*C+20*G+20*M-5*R+T.
```

The pixels at 1/4-pixel positions labeled as e, g, p and r are calculated by averaging with upward rounding of the two nearest pixels at half pixel positions in the diagonal direction by using

```
e = (b1+h1+32)>>6
g = (b1+m1+32)>>6
p = (h1+s1+32)>>6
r = (m1+s1+32)>>6
```

Nokia Exhibit 2010, p. 1
Amazon.com v. Nokia
IPR2024-00627

The final values of those 1/4-pixels will be clipped to values in the range (0,255). The implementation using 16 bit precision is given in Appendix.

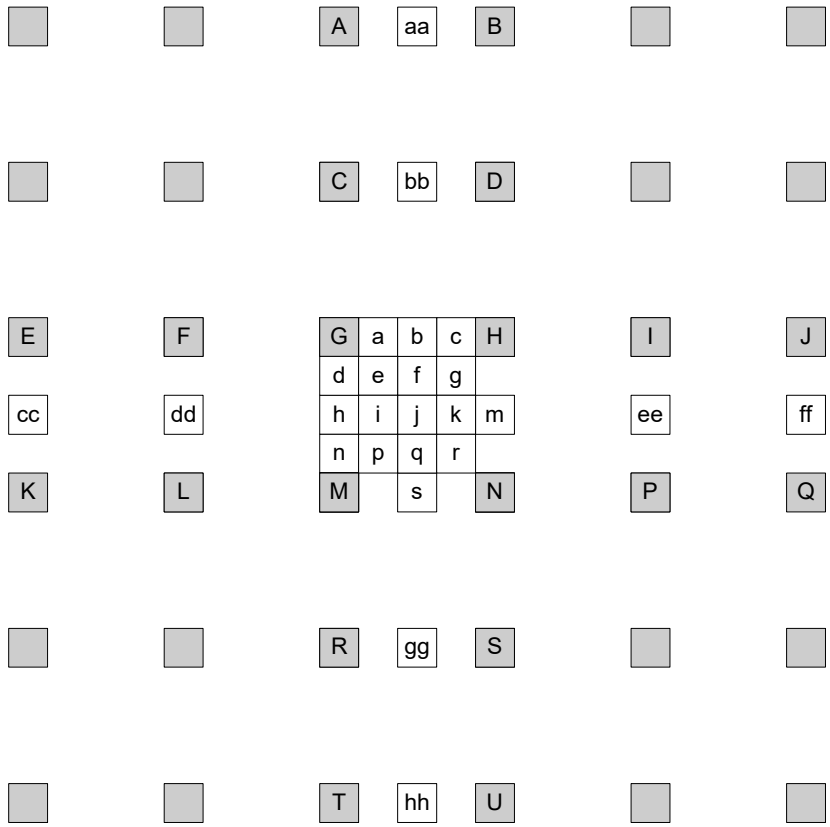

Figure 1: Integer pixels (shaded blocks with upper-case letters) and fractional pixel positions (white blocks with lower-case letters).

Bidirectional Averaging

Bidirectional prediction averaging in H.264 is performed using bilinear interpolation with upward rounding. Especially in case of hierarchical B-frames the rounding error may accumulate. To reduce the effect of error accumulation we propose to signal to the decoder should upward on downward round be used for a given slice.

Simulation Results

The results are obtained when High Precision Interpolation is enabled (Table 1) and when both High Precision Interpolation and bidirectional averaging rounding modifications are enabled.

Table 1: High Precision Interpolation results.

		IPP	IBBP	IbBb
QCIF	foreman	-1.19	-0.91	-1.20
	container	-3.38	-2.18	-1.45
	silent	-0.42	-0.11	0.06
QCIF ave		**-1.66**	**-1.04**	**-0.86**
CIF	foreman	-2.19	-0.92	-1.25
	mobile	-0.85	-0.14	-0.33

Nokia Exhibit 2010, p. 2
Amazon.com v. Nokia
IPR2024-00627

	paris	-1.32	0.34	0.24
	tempete	-3.17	-1.08	-1.28
CIF ave		**-1.88**	**-0.45**	**-0.66**
	bigships	-7.23	-5.64	-5.24
	city	-7.99	-6.02	-3.80
	crew	-6.93	-4.44	-1.87
	night	-1.75	-0.56	-0.50
720p	shuttlestart	-14.60	-6.13	-5.18
720p ave		**-7.70**	**-4.56**	**-3.32**
AVERAGE		**-4.25**	**-2.32**	**-1.82**

Table 1: High Precision Interpolation and bidirectional averaging rounding improvements.

		IBBP	IbBb
	foreman	-0.67	-1.14
	container	-2.11	-2.13
QCIF	silent	0.07	-0.10
QCIF ave		-0.90	-1.12
	foreman	-1.08	-1.99
	mobile	-0.11	-0.67
	paris	0.59	0.29
CIF	tempete	-1.06	-1.63
CIF ave		-0.42	-1.00
	bigships	-5.77	-6.74
	city	-6.01	-5.88
	crew	-4.92	-3.67
	night	-0.69	-0.87
720p	shuttlestart	-6.46	-7.15
720p ave		-4.77	-4.86
AVERAGE		-2.35	-2.64

Appendix

16-bit register implementation

The interpolation of a, c, d, n, e, g, p and r uses intermediate values b1, h1, s1 and m1, which are within range -2550 to 10710 and can be represented using 15 bits plus sign (15s). Hence their calculation does not pose a problem for 16 bit arithmetic. During calculation of subpel values f, i, k and q intermediate value j1, requiring 20 bits representation, is used. We apply the method introduced in **Error! Reference source not found.**] to make an intermediate value j1 fit 16 bit registers. Shaded rows contain operations done at the end only (not used for interpolation of other sub-pixel positions).

{j}

Operation	Comment	Min val	Max val	Reg size
r1 = y0	y0 is1-D half-pixel aa1 before shifting down	-2550	10710	15s
r2 = y5	y5 is1-D half-pixel hh1 before shifting down	-2550	10710	15s

Nokia Exhibit 2010, p. 3
Amazon.com v. Nokia
IPR2024-00627

r1 = r1+r2	r1 is y0+y5	-5100	21420	16s
r2 = y1	y1 is1-D half-pixel bb1 before shifting down	-2550	10710	15s
r3 = y4	y4 is1-D half-pixel gg1 before shifting down	-2550	10710	15s
r2 = r2+r3	r2 is y1+y4	-5100	21420	16s
r1 = r1-r2	r1 is (y0+y5)-(y1+y4)	-26520	26520	16s
r1 = r1 >> #2	r1 is ((y0+y5)-(y1+y4))>>2	-6630	6630	14s
r3 = y2	y2 is 1-D half-pixel b1 before shifting down	-2550	10710	15s
r4 = y3	y3 is 1-D half-pixel s1 before shifting down	-2550	10710	15s
r3 = r3+r4	r3 is y2+y3	-5100	21420	16s
r3 = max(# -2372,r3)	clip r3 to avoid overflow (low side)	-2372	21420	16s
r3 = min(#18640,r3)	clip r3 to avoid overflow (high side)	-2372	18640	16s
r2 = r3-r2	r2 is (y2+y3)-(y1+y4)	-23792	23740	16s
r1 = r1 + r2	r1 is ((y0+y5)-5*(y1+y4)+4*(y2+y3))>>2	-30422	30370	16s
r1 = r1 >> #2	r1 is ((y0+y5)-5*(y1+y4)+4*(y2+y3))>>4	-7605	7592	14s
r1 = r1 + r3	r1 is ((y0+y5)-5*(y1+y4)+20*(y2+y3))>>4	-9977	26232	16s
j1 = r1	**store result**			
r1 = r1+32	r1 is ((y0+y5)-5*(y1+y4)+20*(y2+y3)+512)>>4	-9945	26264	16s
r1 = r1 >> #6	r1 is ((y0+y5)-5*(y1+y4)+20*(y2+y3)+512)>>10	-155	410	10s
r1 = max(#0,r1)	clip r1 on the low side	0	410	9u
r1 = min(#255,r1)	clip r1 on the high side	0	255	8u

The code below describes the 16-bit implementation of interpolating {f,i,k,q}:

{f,i,k,q}

Operation	Comment	Min val	Max val	Reg size
r1 = y0	r1 is y0 (1-D half-pixel such as b1, h1, m1 and s1 before shifting down)	-2550	10710	15s
r2 = j1	r2 is j1 (2-D half-pixel j1 before shifting down)	-9914	26232	16s
r2 = r2>>1	r2 is j1>>1	-4957	13116	15s
r1 = r1+r2	r1 is y0+(j1>>1)	-7507	23826	16s
r1 = r1+32	r1 is y0+(j1>>1)+32	-7491	23842	16s
r1 = r1>>#6	r1 is (y0+(j1>>1)+32)>>6	-235	745	11s
r1 = max(#0,r1)	clip r1 on the low side	0	745	10u
r1 = min(#255,r1)	clip r1 on the high side	0	255	8u

References

[1] TK Frank Bossen, “Full 16-bit implementation of 1/4 pel motion compensation”, ISO/IEC MPEG & ITU-T VCEG, JVT-C037, Fairfax, Virginia, USA, May, 200.

[2] “Switched Interpolation Filter with Offset”, ITU-T Q.6/SG16 VCEG, COM 16 – C 463 – E, Geneva, April 2008.

IPR Notification: Qualcomm may have IPR relating to the technology described 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 (per box 2 of the ITU-T/ITU-R/ISO/IEC patent statement and licensing declaration form).

Nokia Exhibit 2010, p. 4
Amazon.com v. Nokia
IPR2024-00627