|  |  |
| --- | --- |
| **ITU – Telecommunications Standardization Sector**  STUDY GROUP 16 Question 6  **Video Coding Experts Group (VCEG)**  56th Meeting: 17–21 July 2017, Turin, IT | Document VCEG-BD01 |

|  |  |  |  |
| --- | --- | --- | --- |
| Question: | Q.6/SG16 (VCEG) | | |
| Source: | **Gary Sullivan**  Microsoft Corp.  1 Microsoft Way  Redmond, WA 98052 USA  **Tony Tang**  Nvidia Co., Ltd.  No. 5709 Shenjiang Road  Shanghai, China 201210  **Alexis Tourapis**  1 Infinite Loop  Cupertino, CA 95014  USA | Tel:  Email:  Tel:  Email:  Tel:  Email: | +1 (425) 703-5308  garysull@microsoft.com  +86 (21) 61049012  totang@nvidia.com  +1-408-228-7983  atourapis@apple.com |
| Title: | **Error in the AVC text specification for motion vector deltas in levels 6, 6.1, and 6.2** | | |
| Purpose: | Erratum report | | |

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

**Abstract**

This contribution reports that there is an error in the AVC text specification for the recently added levels 6, 6.1, and 6.2 (which were added to support very large picture sizes such as “8K” pictures). It is reported that the current specification requires the decoder to support a 17-bit range of motion vector deltas, which is suggested to be an oversight and is reportedly not supported in the JM reference software. It is proposed that corrective action be taken to limit the range to 16 bits when feasible.

1. **Problem description and proposed action**

On 2017-06-15, Tony Tang of Nvidia reported this problem in the AVC specification to Gary Sullivan and Karsten Sühring, which was confirmed in correspondence with Alexis Tourapis.

The AVC specification was recently revised (11th edition of Rec. ITU-T H.264 approved 2016-10-14 and published 2017-01-31, and ISO/IEC 14496-10:2014/FDAM 2 approved 2016-08-16 and consolidated into the upcoming 9th edition of ISO/IEC 14496-10) to add the specification of levels 6, 6.1, and 6.2, which support larger picture sizes than were previously supported in the standard.

Motion vectors (MVs) in these new levels are specified to be constrained to a 16-bit range, to enable practical implementation. However, when specifying these levels, we apparently failed to notice that the MV *deltas* that are used to calculate the MV values are not constrained, except indirectly by the limits on the value of the resulting MV that is derived from the delta. As a result, the MV deltas could span a 17-bit range.

We believe it is clear that there was no intent to support such a large range of MV deltas in practice, that there is no benefit to supporting this range, and that encoder designers would not ever want to use such large MV deltas in any implementations (except perhaps for creating deliberately “evil” conformance-testing bitstreams). Moreover, the JM reference software only supports a 16-bit range for the MV deltas.

We therefore suggest correcting the error by imposing a 16-bit limit on the range of MV deltas in the text specification when feasible.

1. **Proposed draft text**

For the sake of simplicity of the structure of the text, we suggest to express the proposed limit in clause 7 so that it applies across all profiles and levels (although it is really only necessary in levels 6, 6.1, and 6.2).

The semantics of mvd\_l0[ mbPartIdx ][ 0 ][ compIdx ] in 7.4.5.1 currently end with the following sentence:

The range of the components of mvd\_l0[ mbPartIdx ][ 0 ][ compIdx ] is specified by constraints on the motion vector variable values derived from it as specified in Annex ‎A.

We suggest that this sentence also has another small problem, since the phrase “components of mvd\_l0[ mbPartIdx ][ 0 ][ compIdx ]” doesn’t really make sense, because mvd\_l0[ mbPartIdx ][ 0 ][ compIdx ] is a scalar quantity (and therefore there are no “components of” it). We suggest to replace that sentence with this:

The value of mvd\_l0[ mbPartIdx ][ 0 ][ compIdx ] shall be in the range of −8192 to 8191.75, inclusive. The range of mvd\_l0[ mbPartIdx ][ 0 ][ compIdx ] is also constrained indirectly by constraints on the motion vector variable values derived from it as specified in Annex ‎A.

A similar change will also be needed in G.7.4.5.1, which contains the following sentence:

The range of the components of mvd\_l0[ mbPartIdx ][ 0 ][ compIdx ] and mvd\_l1[ mbPartIdx ][ 0 ][ compIdx ] is specified by constraints on the motion vector variable values derived from it as specified in clause G.10.

We suggest replacing that sentence (similarly) with this:

The value of mvd\_l0[ mbPartIdx ][ 0 ][ compIdx ] and mvd\_l1[ mbPartIdx ][ 0 ][ compIdx ] shall be in the range of −8192 to 8191.75, inclusive. The range of mvd\_l0[ mbPartIdx ][ 0 ][ compIdx ] and mvd\_l1[ mbPartIdx ][ 0 ][ compIdx ] is also constrained indirectly by constraints on the motion vector variable values derived from it as specified in clause G.10.

A similar change will also be needed in G.7.4.5.2, which contains the following sentence:

The range of the components of mvd\_l0[ mbPartIdx ][ subMbPartIdx ][ compIdx ] and mvd\_l1[ mbPartIdx ][ subMbPartIdx ][ compIdx ] is specified by constraints on the motion vector variable values derived from it as specified in clause G.10.

We suggest replacing that sentence (similarly) with this:

The value of mvd\_l0[ mbPartIdx ][ subMbPartIdx ][ compIdx ] and mvd\_l1[ mbPartIdx ][ subMbPartIdx ][ compIdx ] shall be in the range of −8192 to 8191.75, inclusive. The range of mvd\_l0[ mbPartIdx ][ subMbPartIdx ][ compIdx ] and mvd\_l1[ mbPartIdx ][ subMbPartIdx ][ compIdx ] is also constrained indirectly by constraints on the motion vector variable values derived from it as specified in clause G.10.

# Remarks on absMvdCompN

Alexis Tourapis remarked that there is also a potential dynamic range issue that may be worth thinking about in regard to the computation of absMvdCompN in 9.3.3.1.1.7, especially in the MBAFF case when the current macroblock is a frame macroblock, and the macroblock mbAddrN is a field macroblock, in which case the MVD value is doubled, with a result that could exceed a 16 bit dynamic range of the MVD value is maximally negative prior to the absolute value. There is also the addition absMvdCompA + absMvdCompB, which can also add another bit of dynamic range. This would only be a problem if a decoder designer did not realize that there was an issue; it is easy to deal with if implementers are aware of the potential problem. No normative action seems needed for this (and the MBAFF issue is not a problem since MBAFF is not supported in levels 6, 6.1, and 6.2). However, two potential actions may be worth considering:

* Adding a NOTE to point out the potential dynamic range problem, or
* Rewriting the way that the calculations are performed so that the dynamic range problem would be avoided, e.g., as in:
* If absMvdCompA is greater than 32 or absMvdCompA is greater than 32, ctxIdxInc is set equal to 2.
* Otherwise, if absMvdCompA + absMvdCompB is greater than 32, ctxIdxInc is set equal to 2.
* Otherwise, absMvdCompA + absMvdCompB is greater than 2, ctxIdxInc is set equal to 1.
* Otherwise (absMvdCompA + absMvdCompB is less than or equal to 2), ctxIdxInc is set equal to 0.

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