|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11**  24th Meeting: Geneva, CH, 26 May – 1 June 2016 | Document: JCTVC-X0079 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **Suggestion for new draft version of Conversion and Coding Practices for HDR/WCG Video** | | |
| *Status:* | Input contribution to JCT-VC | | |
| *Purpose:* | Input contribution | | |
| *Author(s) or Contact(s):* | Jonatan Samuelsson  Alexis Tourapis  Jacob Ström  Chad Fogg | Email: | jonatan.samuelsson@ericsson.com |
| *Source:* |  | | |

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

# Abstract

This document contains a new suggested draft of the “recommended practice” technical report on HDR video. It is based on JCTVC-W1017 with modifications discussed during the teleconference meetings of AHG13 on April 11 and May 9 and some further modifications.Summary

This document provides guidance on processing of high dynamic range (HDR) video. The purpose of this document is to provide a publically referencable recommended practice for the operation of AVC or HEVC systems adapted for compressing HDR video. This document includes a description of processing steps for converting linear light, RGB, 4:4:4 video signals into ST 2084, Non Constant Luminance (NCL) Y′CbCr, 4:2:0 video signals compressed by the AVC or HEVC encoding. This document also includes a description of post-decoding processing steps for converting ST 2084, NCL Y′CbCr, 4:2:0 to linear light, RGB, 4:4:4 after decoding. Some high level recommendations for compression with HEVC and AVC are also included in this document.

**CONTENTS**

| Page |
| --- |
| Abstract i  1 Scope 4  2 References 5  3 Definitions 5  4 Abbreviations and acronyms 6  5 Conventions 7  5.1 General 7  5.2 Arithmetic operators 7  5.3 Bit-wise operators 7  5.4 Assignment operators 8  5.5 Mathematical functions 8  5.6 Order of operations 8  6 Introduction 9  7 Pre-encoding process 10  7.1 Pre-encoding processes 11  7.1.1 Conversion from a linear to a non-linear light representation: RGB to R′G′B′ 11  7.1.2 Colour representation conversion: R′G′B′ to Non Constant Luminance Y′CbCr 12  7.1.3 Chroma down-conversion 12  7.1.4 Floating-point to fixed-point (narrow range) 10 bit conversion 14  7.2 High Precision Pre-encoding 15  7.3 Closed loop pre-encoding conversion – Luma Adjustment 16  7.3.1 Luma adjustment – Bisection search 18  8 Encoding process 20  8.1 HEVC encoding 20  8.2 AVC encoding 21  9 Decoding process 21  10 Post-decoding processes 21  10.1 Post-decoding processes 22  10.1.1 Conversion from a fixed-point to a floating-point representation 22  10.1.2 Chroma up-conversion 22  10.1.3 Colour representation conversion: Non Constant Luminance Y′CbCr to R′G′B ′ 23  10.1.4 Conversion from a non-linear to a linear light representation: R′G′B′ to RGB 23  10.2 High precision post-decoding 24  Annex A. Supplemental enhancement information (SEI) messages. 26  A.1 Mastering display colour volume SEI message 26  A.2 Content Light Level information SEI message 26 |

Conversion and Coding Practices for HDR/WCG Video

# Scope

[Ed. (AT): We should add a small introduction about the importance of HDR.]

This document provides guidance on processing of consumer distribution high dynamic range video including conversions steps for going from a linear light RGB representation with BT.2020 colour primaries to a 10-bit, narrow range, ST 2084, 4:2:0, non-constant luminance Y′CbCr representation. The scope of this document is illustrated in Figure 1‑1.



Figure 1‑1 – Illustration of the scope of this document.

[Ed. (CF): Suggest adding “HDR/WCG encoding parameters” to the “optional metadata” dashed arrow from Content preparation to Encoding process.]

It should be noted that the content preparation step as well as the display adaptation step are considered to be out of scope of this document, however metadata generated during the content preparation may be passed through the encoder-decoder chain that significantly affect display adaptation. The content preparation step may include filtering and image enhancement processing such as de-noising, colour correction, and sharpening filtering among others. Such methods are deliberately not described in this document. The processing steps described in this document are made available for reference only and the document does not contain any elements of normative nature. It is possible to replace one or more of the processing steps described in this document, for example in order to reduce computational complexity or to improve fidelity. This document’s intention is to provide a recommended practice guideline for operating an HDR video system that is constrained to code a 10-bit, ST 2084, 4:2:0, non-constant luminance Y′CbCr signal representation. This configuration is aligned with the HDR10 media profile defined in [6] and the restrictions in [12]. The processing steps in this document are optimized with the intention of providing the best possible result when the same Hypothetical Reference Viewing Environment (HRVE) is used before and after the HDR system. This document does not account for when different viewing environments are used before and after the HDR system.

[Ed. (AT): I somehow do not feel so comfortable with this description. What is the definition of “Same HRVE”? We do not unfortunately define this here (as note 1 mentions), making this too vague and confusing. . To give some examples, ignoring ambient environment, it is quite possible that maybe the grading display was a pro grading display only capable of 1K cd/m^2 supporting P3 primaries. The delivery display is a display that also supports P3 primaries but can go up to 2K in terms of brightness. Would that be defined as the same HRVE or not? I think maybe a better way to describe this is to be a bit more precise and say that the intent of this work is to provide the best experience using a “display referred” system. If there is a SMPTE or other specification that we could use that provides us with the characteristics of such a system, then we should maybe reference that. Then mention that no consideration of display management based on display, environment, and content characteristics is considered. Otherwise this appears maybe confusing and misleading.]

[Ed. (CF): Suggest just referencing BT.2390 and not spend any cycles or text here definining viewing environment. BBC et al have made this point in UHDF]

NOTE 1 – This document does not provide any description of any preferred HRVE but acknowledges the fact that in many applications of HDR video it might be desirable with a well-defined HRVE description to ensure alignment between content preparation and content consumption.

[Ed. (AT): It would be good in this paragraph to mention that the methods described here were the outcome of investigation within the JCTVC as part of the group’s HDR investigation and development activities.]

Recommended practices described in this document are based on investigative testing conducted within JCT and its parent organizations (ISO/IEC JCT1 SC29 WG11 (MPEG) and ITU-T SG 16 Q.6 (VCEG)) of HEVC coding on HDR/WCG content available at the time.

# References

[Ed. (AT): Should we move this to the end and rename this as bibliography? That seems to be the common practice with other specs (see HEVC and CICP). Also numbering is done using actual numbers instead of name abbreviations.]

1. Recommendation ITU-T H.264 (V10) (2015), *Advanced video coding for generic audiovisual services*. Also available as ISO/IEC 14496-10.
2. Recommendation ITU-R BT.709-6 (2015), *Parameter values for the HDTV standards for production and international programme exchange.*
3. Recommendation ITU-R BT.2020-2 (2015), *Parameter values for ultra-high definition television systems for production and international programme exchange*.
4. Recommendation ITU-R BT.1886 (2011), *Reference electro-optical transfer function for flat panel displays used in HDTV studio production*.
5. Draft Recommendation ITU-R BT.[HDR-TV] (2016), *Image parameter values for high dynamic range television for use in production and international programme exchange.*
6. DECE, *Common File Format & Media Formats Specification Version 2.1.*
7. Recommendation ITU-T H.265 (V3) (2015), *High Efficiency Video Coding.* Also available as ISO/IEC 23008-3.
8. [RP 431-2] SMPTE Standard RP 431-2 (2011), *D-cinema Quality - Reference Projector and Environment*.
9. SMPTE Standard ST 2084 (2014), *High Dynamic Range Electro-Optical Transfer Function for Mastering Reference Display*.
10. Charles Poynton, A Technical Introduction to Digital Video (New York: John Wiley & Sons, 1996).
11. Jacob Ström, Jonatan Samuelsson and Kristofer Dovstam, ”Luma Adjustment for High Dynamic Range Video”, Proceedings of the IEEE Data Compression Conference (DCC), Snowbird, March 2016.
12. Blu-Ray Disc Association, “BD-ROM: Audio Visual Application Format Specifications version 3” (July 2015)

# Definitions

This document defines the following terms:

* 1. **Inverse transfer function:** The function used in the pre-encoding process to convert from a linear representation to a non-linear representation. The function is the inverse of the transfer function used in the post-decoding process. In applications where the transfer function is called Electro-Optical Transfer Function (EOTF), the inverse transfer function may be called inverse-EOTF.

NOTE 1 – In this document the pre-encoding process is assumed to operate on HDR video content that has been prepared using a hypothetical reference viewing environment as shown in Figure 1‑1. The content preparation step may contain processing such as applying a system gamma, in which the HDR video is converted from one linear representation (corresponding to the scene) to another linear representation (corresponding to the display). In systems were no such system gamma is applied in the content preparation step, the process of converting from a linear representation (corresponding to the scene) to a non-linear representation is typically called Opto-Electrical Transfer Function.

* 1. **Narrow range:** A range in a fixed-point (integer) representation that does not span the full range of values that could be expressed with that bit depth. In this document the range from 64 (black) to 940 (peak white) is used for Y′, and the range from 64 to 960 is used for Cb and Cr for 10 bit representations, which is aligned with [4]. Narrow range is in some applications called by synonyms such as: “limited range”, “video range”, “legal range”, “SMPTE range” or “standard range.”
  2. **Full range:** a range in a fixed-point (integer) representation that spans the full range of values that could be expressed with that bit depth. For 10-bit signals, black corresponds to code 0 and peak white corresponds to code 1023.
  3. **Random Access Point Access Unit (RAPAU):** An access unit in the bitstream containing an intra coded picture with the property that all pictures following the intra coded picture in output order can be correctly decoded without using any information preceding the Random Access Point Access Unit in the bitstream.
  4. **Transfer function:** The function used in the post-decoding process to convert from a non-linear representation to a linear representation. In applications where no further processing or display adaptation is performed, this function is sometimes called Electro-Optical Transfer Function (EOTF).

# Abbreviations and acronyms

This document uses the following abbreviations and acronyms:

AVC Advanced Video Coding

CL Constant Luminance

EOTF Electro-Optical Transfer Function

FIR Finite Impulse Response

HD High Definition

HDR High Dynamic Range

HDR10 HDR content representation that utilizes the HEVC Main 10 codec, BT.2020 colour primaries, the  
ST 2084 transfer function (PQ), and non-constant luminance Y′CbCr encoding.

[Ed. (JS): I do not think we should make our own definition of “HDR10”. There are already other definitions of the term HDR10 such as in [5] that are not 100% aligned with this definition which could cause confusion. (CF): the term “HDR10” is defined in DECE, UHD-Forum, and UHD-Alliance]

HEVC High Efficiency Video Coding

HRVE Hypothetical Reference Viewing Environment

HVS Human Visual System

NCL Non Constant Luminance

PQ Perceptual Quantizer. Informal naming of the SMPTE ST 2084 transfer function standard.

QP Quantization Parameter

RAPAU Random Access Point Access Unit

RGB Colour System using Red, Green, and Blue components

SDR Standard Dynamic Range

SEI Supplemental Enhancement Information

OETF Opto-Electrical Transfer Function

TF Transfer Function

VUI Video Usability Information

WCG Wide Colour Gamut

XYZ The CIE 1931 colour space. Y corresponds to the luminance signal.

Y′CbCr Colour space representation commonly used for video/image distribution as a away of encoding RGB information. Also commonly expressed as YCbCr or Y′C′BC′R. Relationship between Y′CbCr and RGB is dictated by certain signal parameters, such as colour primaries, transfer characteristics, and matrix coefficients. Unlike the (constant luminance) Y component in the XYZ representation, Y′ in this representation might not be representing the same quantity. Y′ is commonly referred to as “luma.”

# Conventions

## General

NOTE – The mathematical operators used in this document are similar to those used in the C programming language. However, the results of integer division and arithmetic shift operations are defined more precisely, and additional operations are defined, such as exponentiation and real-valued division. Numbering and counting conventions generally begin from 0, e.g., "the first" is equivalent to the 0-th, "the second" is equivalent to the 1-th, etc.

## Arithmetic operators

The following arithmetic operators are defined as follows:

 Addition

− Subtraction (as a two-argument operator) or negation (as a unary prefix operator)

\* Multiplication, including matrix multiplication

xy Exponentiation. Specifies x to the power of y. In other contexts, such notation is used for superscripting not intended for interpretation as exponentiation.

/ Integer division with truncation of the result toward zero. For example, 7 / 4 and (−7) / (−4) are truncated to 1 and (−7) / 4 and 7 / (−4) are truncated to −1.

 Used to denote division in mathematical equations where no truncation or rounding is intended.

Used to denote division in mathematical equations where no truncation or rounding is intended.

 The summation of f( i ) with i taking all integer values from x up to and including y.

x % y Modulus. Remainder of x divided by y, defined only for integers x and y with x >= 0 and y > 0.

## Bit-wise operators

The following bit-wise operators are defined as follows:

& Bit-wise "and". When operating on integer arguments, operates on a two's complement representation of the integer value. When operating on a binary argument that contains fewer bits than another argument, the shorter argument is extended by adding more significant bits equal to 0.

| Bit-wise "or". When operating on integer arguments, operates on a two's complement representation of the integer value. When operating on a binary argument that contains fewer bits than another argument, the shorter argument is extended by adding more significant bits equal to 0.

^ Bit-wise "exclusive or". When operating on integer arguments, operates on a two's complement representation of the integer value. When operating on a binary argument that contains fewer bits than another argument, the shorter argument is extended by adding more significant bits equal to 0.

x >> y Arithmetic right shift of a two's complement integer representation of x by y binary digits. This function is defined only for non-negative integer values of y. Bits shifted into the MSBs as a result of the right shift have a value equal to the MSB of x prior to the shift operation.

x << y Arithmetic left shift of a two's complement integer representation of x by y binary digits. This function is defined only for non-negative integer values of y. Bits shifted into the LSBs as a result of the left shift have a value equal to 0.

## Assignment operators

The following arithmetic operators are defined as follows:

 Assignment operator

  Increment, i.e., *x*  is equivalent to *x*  *x*  1; when used in an array index, evaluates to the value of the variable prior to the increment operation.

− − Decrement, i.e., *x*− − is equivalent to *x*  *x* − 1; when used in an array index, evaluates to the value of the variable prior to the decrement operation.

+= Increment by amount specified, i.e., x += 3 is equivalent to x = x + 3, and x += (−3) is equivalent to x = x + (−3).

−= Decrement by amount specified, i.e., x −= 3 is equivalent to x = x − 3, and x −= (−3) is equivalent to x = x − (−3).

## Mathematical functions

Abs( x ) =

Clip3( x, y, z ) =

Floor( x ) the largest integer less than or equal to x.

Round( x ) = Sign( x ) \* Floor( Abs( x ) + 0.5 )

Sign( x ) =

## Order of operations

When order of precedence in an expression is not indicated explicitly by use of parentheses, the following rules apply:

– Operations of a higher precedence are evaluated before any operation of a lower precedence.

– Operations of the same precedence are evaluated sequentially from left to right.

Table 5‑1 specifies the precedence of operations from highest to lowest; a higher position in the table indicates a higher precedence.

NOTE – For those operators that are also used in the C programming language, the order of precedence used in this Specification is the same as used in the C programming language.

Table 5‑1 – Operation precedence from highest (at top of table) to lowest (at bottom of table)

|  |
| --- |
| **Operations (with operands x, y, and z)** |
| "x++", "x− −" |
| "!x", "−x" (as a unary prefix operator) |
| xy |
| "x \* y", "x / y", "x  y""", "x % y" |
| "x + y", "x − y" (as a two-argument operator), "" |
| "x  <<  y", "x  >>  y" |
| "x < y", "x  <=  y", "x > y", "x  >=  y" |
| "x  = =  y", "x  !=  y" |
| "x & y" |
| "x | y" |
| "x  &&  y" |
| "x  | |  y" |
| "x ? y : z" |
| "x..y" |
| "x = y", "x  +=  y", "x  −=  y" |

# Introduction

The HDR System described in this document consists of four major stages; Pre-encoding processes (Section 7), an Encoding process (Section 8), a Decoding process (Section 9), and Post-decoding processes (Section 10). These four stages are applied sequentially with the output of one stage being used as input to the next stage according to the above-mentioned order.

It is assumed that both the input to and the output of the HDR System are 4:4:4, linear light, floating-point signals, in an RGB colour representation using the same colour primaries. The output signal is targeted to resemble the input video signal as closely as possible. Other video formats can be used as input to the HDR System by first converting them to the above defined input signal representation. The HDR System described in this document is, in practice, a system for both HDR and Wide Colour Gamut (WCG) video since it is assumed that the input video is represented with colour primaries in accordance with BT.2020 [3] and BT.[HDR-TV] [5].

The primary purpose of the pre-encoding process is to convert the video input from its 4:4:4 RGB linear light, floating-point signal representation to a signal that is suitable for a video encoder. The conversion to a non-linear representation is performed in an attempt to exploit the characteristics of the human visual system (HVS) that could allow the re-quantization of the signal at a limited precision.

NOTE 1 – For a fixed-point linear HDR video representation, approximately a 28-bit integer representation would be required to avoid introducing visible quantization/banding errors due to the 28 f-stop linear light dynamic range (0.00005 to 10,000 nits) that is spanned by the 10-bit ST 2084 transfer function. In practice, the input to the HDR System will typically be in a non-linear representation that could either have to be first converted to linear light data or be directly converted to ST 2084.

[Ed. (AT): It may be a good idea that somewhere in the text we clarified that the scope of this is about HDR-TV/Consumer entertainment applications. We seem not to be doing so, which might not be appropriate. (CF): I added “consumer distribution” to the first sentence of the scope section 1. Note 2 below also reminds the reader of this scope]

It is assumed that encoding and decoding is performed in a 4:2:0, 10-bit representation. An encoder is expected to make the best use of the encoding tools available according to a particular specification, profile, and level, given also the characteristics of the content and the limitations of the intended application and implementation. In particular, different encoding algorithms, such as algorithms for motion estimation, mode decision, rate allocation, rate control, and post-filtering control among other aspects, may have to be considered when encoding HDR material, in a given representation, compared to SDR material. The decoding process on the other hand is fully described in the respective HEVC and AVC decoder standards, where a decoder must fully comply to the intended profile and level to output precisely reconstructed video samples from a given input bistream accoding to a deterministic decoding process, nominally over a time window indicated in the bitstream.

NOTE 2 – The focus of this document is on consumer and direct-to-home applications, which are expected to be, at least in the near term future, in a 4:2:0 10-bit format. Processes similar to the ones described in this document can be used for conversion and compression of other formats, such as 4:2:2 and 4:4:4 chroma formats and/or video with bit-depth higher than 10.

The steps in the post-decoding process are aligned with what is commonly referred to as the non-constant luminance representation (NCL) in which colour conversion, to R′G′B′, is performed prior to applying the transfer function to produce linear RGB.

There is no specific or minimum bitdepth required for performing the operations described in the pre-encoding process and the post-encoding process. Using the precision associated with 64 bit floating-point operations will give high accuracy, but it is also possible to use fixed-point arithmetic and/or floating-point operations with precision lower than 64 bits. Using too low precision should be avoided since it could lead to loss of precision in the output video. The input to the encoding step and the output of the decoding step are 10-bit integer representations.

# Pre-encoding process

The Pre-encoding process of this recommendation includes the following components:

1. a conversion component from a linear data representation to a non-linear data representation using the appropriate transfer function,
2. a colour format conversion component that converts data to the Non-Constant Luminance Y′CbCr representation,
3. a chroma down-conversion component that converts data from 4:4:4 to 4:2:0, and
4. a conversion component that converts a floating-point representation to a fixed-point representation (i.e. 10 bits).

NOTE 3 – Scaling may also be a vital component of the pre-encoding process, for example if the target system requires a particular image resolution be delivered to the decoder. It may be desirable, for example, to rescale a source from a 1920x1080 to a 3840x2160 resolution, or vice versa. In this document we will not elaborate on this topic, however, we would suggest that rescaling is performed in either the linear domain, or at least using the R′G′B′ representation.  
[Ed. (JS): If the suggestion should be kept in this note, there needs to be results to back up the suggestion.]

Figure 7‑1 presents a diagram of how these components could potentially be combined together to generate the desirable outcome, in a conventional manner. In this system, all blocks work independently, whereas chroma subsampling is performed using fixed-point arithmetic and at the same precision as the target outcome.



Figure 7‑1 – Conventional pre-encoding process system diagram.

Although this combination could likely be the most appropriate for some implementations, it has several limitations that can affect both its performance and implementation complexity. In this section we first introduce in more detail the pre-encoding process components, and then present several alternative configurations to the one presented in Figure 7‑1 that can provide either performance or complexity benefits. Recommendations on how to best utilize some of the conversion components are also presented.

## Pre-encoding processes

### Conversion from a linear to a non-linear light representation: RGB to R′G′B′

Conversion from a linear to a non-linear light representation is performed using an inverse transfer function, or as is commonly referred to other specifications, an opto-electrical transfer function (OETF). In the context of HDR10, the ST.2084 transfer function, also referred to as PQ, is used.

More specifically, the non-linear light representation V of a linear light intensity signal Lo, that takes values normalized to the range [0, 1], can be computed as:

V = iTF(Lo) = ( ( c1 + c2 \* Lon ) ÷ ( 1 + c3 \* Lon ) )m (7‑1)

where c1, c2, c3, m, and n are constants, which are defined as follows:

c1 = c3 − c2 + 1 = 3424 ÷ 4096 = 0.835 937 5 (7‑2)

c2 = 2413 ÷ 128 = 18.851 562 5 (7‑3)

c3 = 299 ÷ 16 = 18.687 5 (7‑4)

m = 2523 ÷ 32 = 78.843 75 (7‑5)

n = 1305 ÷ 8192 = 0.159 301 757 812 5 (7‑6)

The peak value of 1 for Lo is ordinarily intended to correspond to an intensity level of 10 000 candelas per square metre, while the value of 0 for Lo is ordinarily intended to correspond to an intensity level of 0 candelas per square metre. The behaviour of the inverse ST 2084 [9] transfer function in relationship to the BT.709 [2][3] and the inverse of the BT.1886 [4] transfer functions is shown in Figure 7‑2.

NOTE  – It should be noted that a direct comparison of the ST 2084 transfer function with BT.709 might not be appropriate since BT.709 may assume the use of a system gamma during decoding.



Figure 7‑2 – Behaviour of the inverse of ST 2084 (PQ), the inverse of BT.1886 and BT.709 transfer functions.

This process is applied to all R, G, and B linear light samples, where each component is a number between 0.0 (representing no light) and 1.0 (representing 10 000 candelas per square metre). This results in their non-linear counterparts R′, G′, and B′

as follows.

R′ = iTF( R ) (7‑7)

G′ = iTF( G ) (7‑8)

B′ = iTF( B ) (7‑9)

Although it is, in general, recommended to perform this conversion process using Equation 7‑1 directly, this, however, may not be possible in some implementations given the complexity of the computation. Instead, look-up tables (LUT) may be preferred. Due to the characteristics of the conversion and the desire to achieve high precision for both low/dark and high values, it is highly recommended that, in such scenario, a non-uniformly indexed LUT interpolator is used. Such schemes can achieve relatively high accuracy/minimum approximation error for the conversion, while achieving considerable memory savings.

### Colour representation conversion: R′G′B′ to Non Constant Luminance Y′CbCr

Conversion from the R′G′B′ to the Non Constant Luminance Y′CbCr representation is commonly performed using a 3x3 matrix conversion process of the form:

(7‑10)

where *wYR*, *wYG*, *wYB*, *wCbR*, *wCbG*, *wCbB*, *wCbR*, *wCbG*, and *wCbB* are constants. The values for *wYR*, *wYG*, and *wYB*, are set to exactly the same values used to convert R, G, and B data to the CIE 1931 Y (luminance) signal. For BT.2020 colour primaries these are defined as follows:

(7‑11)

(7‑12)

(7‑13)

NOTE  – These constants are commonly specified using precision limited to 16-bits or less. It may be desirable and beneficial that these are used with a higher precision representation, such as 64-bit float, by computing the appropriate parameters using the primary coordinates.

On the other hand, the values of the constants *wCbR*, *wCbG*, *wCbB*, *wCrR*, *wCrG*, and *wCrB* are computed in a manner that the resulting Cb and Cr components are always within the [-0.5, 0.5] range. This essentially results in the following values:

(7‑14)

(7‑15)

(7‑16)

(7‑17)

(7‑18)

(7‑19)

NOTE  – In some implementations, *wCbR*, *wCbG*, *wCrG*, and *wCrB* may be further restricted to 6 decimal points, and in particular to values -0.13963, -0.36370, -0.459786, and -0.040214 respectively. However, using the above specified higher precision values may be beneficial.

NOTE 7 – The evaluations conducted in MPEG and JCT-VC used 6 decimal point precision.

An alternative method to perform the same conversion process is presented in [3][5], where the computation of the chroma components is computed after the conversion of the luma components as follows:

(7‑20)

(7‑21)

with and .

The inverse process, i.e. converting Y′, Cb, and Cr data back to R′, G′, and B′ data, is specified in Section 10.1.3.

### Chroma down-conversion

Converting the HDR video data from a 4:4:4 representation to a 4:2:0 representation nominally involves filtering and down-converting/subsampling the two chroma planes in both the horizontal and vertical directions, though it is possible to apply more complex methods that preserve edges and thus reduce the impact of interpolated color values that did not exist in the local neighborhood of the original 4:4:4 pixel. It is also a requirement, according to both [3][5], that the resulting chroma samples are co-sited with those of luma at even horizontal and vertical positions (Figure 7‑3) where the first respective sample and line are counted starting from zero.



Figure 7‑3 – Chroma and Luma sample location relationship.

It is anticipated that a considerable amount of consumer electronics conversion systems would use 2-D separable finite impulse response (FIR) linear filters for low-pass filtering the chroma data before subsampling (2:1 decimation step). Such filters would basically be of the form:

(7‑22)

where x[n] is the input chroma signal, y[n] is the filtered output chroma signal, (2 \* N) corresponds to the filter order or, equivalently, (2 \* N + 1) corresponds to the number of taps of the filter, and bi corresponds to the coefficient of the filter at position i.

It has been observed that, especially due to the step transfer slope characteristics and quantization behaviour of ST.2084, special caution needs to be made when selecting the filter coefficients of such a resampling filter, in order to mitigate chroma leakage. Conventional filters used for the down-conversion of SDR chroma signals may result in ringing artefacts that can be rather objectionable when applied to HDR signals, most noticeable when converting the signal back to an RGB representation. That is, errors due to the NCL representation when chroma subsampled tend to be much more objectionable [10] in HDR signals than in SDR signals. It has been suggested that some of these issues could be reduced through the use of non-linear and/or adaptive filters. However, the investigation on the performance of such filters is still ongoing in JCT-VC, though commercial and high end systems have deployed non-linear methods for a number of years.

[Ed. (JS): Further description of the effects of using long filters for downsampling might be desirable.]

Assuming that the use of non-adaptive FIR filters is the only implementation option for a system, it has been found that the following FIR filters may result in less objectionable artefacts. Such filters can be utilized for both vertical and horizontal filtering of the chroma samples:

Table 7‑1 – Suggested filters for chroma downsampling

|  |  |  |  |
| --- | --- | --- | --- |
| **Filter** | **Interpolation filter coefficients** | | |
| b-1 | b0 | b1 |
| 0 | 1/8 | 6/8 | 1/8 |
| 1 | 1/4 | 2/4 | 1/4 |

The characteristics (magnitude and phase) of these filters are shown in Figure 7‑4. Filter 1 is a much sharper filter that is down -6dB by its half-band, whereas the softer rolloff of filter 0 could potentially result in some aliasing caused by significant energy past its half-band. However, filter 0 may also avoid some of the chroma leakage that has been mentioned earlier since edges passed through this filter would output less severe transients.

|  |
| --- |
|  |

Figure 7‑4 – Magnitude characteristics of filters 0 and 1 for chroma downsampling

The upconversion process, i.e. from a 4:2:0 representation back to a 4:4:4 representation, is discussed in section 10.1.2.

### Floating-point to fixed-point (narrow range) 10 bit conversion

A key component of the pre-encoding process is the conversion from a floating-point to a fixed-point, narrow range, 10-bit representation. This process is essentially a quantization step that would introduce some distortion. In general, the conversion process can be expressed as:

(7‑23)

where is the floating-point representation of a particular component and D′ is the resulting quantized value at a precision of b bits. In the case of HDR10, b = 10. The scale and offset constants depend on the target range (narrow versus full range video) and the component type (luma, chroma, or primary components). More specifically, for the narrow range NCL representation, the scale and offset for the luma component are set as:

(7‑24)

(7‑25)

On the other hand, the fixed precision narrow range representation for the two chroma components can be computed as follows:

(7‑26)

(7‑27)

NOTE 8 – If a full range is desired for some applications, the recommendation in [5] is to use for all components, for the chroma components, and offset = 0 in all other cases. An alternative full range representation is used in MPEG-4 AVC/H.264 [1] and HEVC [7] for some other of applications using different transfer functions other than ST 2084 (PQ) or HLG.

Presented in Figure 7‑5: the mapping of non-normalized, gray (R=G=B) linear light values to a non-linear representation according to respective HDR (ST 2084) and SDR (gamma 2.4 for BT.709 / BT.2020, assuming the use of the BT.1886 forward transfer function during display).



Figure 7‑5 – Mapping of “gray” linear light values to quantized 8 (SDR only) and 10 bit (SDR and HDR) values

The inverse conversion process, i.e. converting from a fixed-point representation back to a floating-point representation, is discussed in detail in Section 10.1.1.

## High Precision Pre-encoding

[Ed. (JS): This section was discussed at the telephone conference on May 9, 2016. It has not been concluded whether this should be included or not. If it should be included there needs to be results tpresented that verify the claims.]

Figure 7‑6 presents an alternative configuration where chroma subsampling is performed prior to the conversion from a floating-point representation to a fixed-point representation. This can present an advantage since it can reduce the effect of quantization errors during filtering and subsampling, therefore resulting in a higher quality final representation. This comes, however, at a higher complexity/implementation cost that may not be feasible in some systems.



Figure 7‑6 – Alternative pre-encoding process system with high precision subsampling.

As a compromise to the system presented in Figure 7‑6, one may still utilize fixed-point, but higher precision than the final target, arithmetic operations for certain stages of the conversion process. In Figure 7‑7, for example, the video data are first converted to a fixed-point representation utilizing N-bits, with N > 10 such as N=12 or N=14, immediately after the linear to non-linear representation conversion. Such conversion could also be moved at a later stage of the processing pipe if it is so desired. In the shown example, R′G′B′ data in a fixed-point representation are directly converted to a Y′CbCr fixed-point representation, while chroma subsampling is also performed at this higher fixed precision representation. Finally, an additional conversion step is performed to convert the data to the final desired precision, i.e. in a 10-bit 4:2:0 Y′CbCr, limited/video range video signal.



Figure 7‑7 – Alternative pre-encoding process system utilizing fixed-point high precision processing

## Closed loop pre-encoding conversion – Luma Adjustment

As mentioned in section 7.1.3, chroma leakage may occur in the NCL representation, primarily due to chroma down‑sampling, potentially resulting in objectionable artifacts. This section presents an alternative conversion method, which can considerably alleviate this problem. This method, which is pixel-wise independent, is called the Luma Adjustment method, and it is basically a closed loop conversion process where the impact of chroma down‑sampling, quantization, inverse quantization, and up‑sampling, is accounted for during the luma conversion process. An example schematic diagram of such a system is presented in Figure 7‑8.



Figure 7‑8 – Example schematic diagram of a closed loop pre-encoding conversion system

More specifically, in section 7.1.2 and Equation 7‑10, we presented the conversion process from R′G′B′ to the Y′CbCr NCL representation. If we isolate only the Cb and Cr components from this equation, we then have:

(7‑28)

(7‑29)

If we first convert these two components to their target resolution, i.e. using the steps specified in Section 7.1.3, followed by conversion back to the original representation resolution, as presented in Section 10.1.2, we can have the opportunity to analyze the introduced error into the signal, and potentially compensate for it. More specifically, quantization and downscaling (QD) and subsequently up-scaling and inverse quantization (IQU) onto these components would result into the reconstructed and components, which are defined as follows:

(7‑30)

(7‑31)

Luminance (Y), unlike luma (Y ′), is computed given the linear domain values using a formulation of the form:

(7‑32)

Since , we can write this equation as:

(7‑33)

However, since the reconstructed and values will likely differ from the original values and values, the reconstructed R′\*, G′\*, and B′\* values will also differ from the original R′, G′, and B′ values. Therefore, the reconstructed luminance , which is equal to:

(7‑34)

will also differ from the original luminance Y. Using Equation 10‑2 we can then compute that:

. (7‑35)

For convenience we can define the following Chroma component dependent factors:

(7‑36)

(7‑37)

(7‑38)

resulting in the following formulation for Y*rec*:

(7‑39)

The intent of the luma adjustment method is to try and locate the value of Y′ that would result in the smallest possible distortion for Yrec compared to the original luminance value Y. Unfortunately, due to the non-linear characteristics of the transfer function, solving for Y′ given Y and the values of Crfactor, Gfactor, and Cbfactor is not a straightforward process. However, root-finding numerical methods, such as the bisection method, can be used instead. Performance, and more specifically the convergence speed and accuracy of these methods, is though considerably impacted by the selection of the initial interval as well as the computations performed during the search.

NOTE 9 – Alternative methods that try to approximate the impact on Y and/or methods that evaluate impact on other components have also been suggested and are currently under evaluation.

Assuming optimization of the luminance distortion, the luma adjustment process can basically be seen as the following ordered steps:

1. Calculate the luminance value Y from the original R, G, and B, e.g., using Equation 7‑32. We will refer to this as the value.
2. Convert the R, G, and B data to their R′, G′, and B′ representation.
3. Given the R′, G′, and B′ planes generate the Cb and Cr chroma planes.
4. Downscale and quantize the chroma planes to their target representation.
5. De-quantize and up-convert the chroma planes back to their original representation, i.e. and .
6. Calculate Crfactor, Gfactor, and Cbfactor from and .
7. Given the reconstructed chroma planes, try to find for each luma position an appropriate Y′ value, i.e. bestY′, that would potentially result in a minimum distortion for a particular aspect of the signal, i.e. in this case minimum luminance distortion. The value of bestY′ at each luma position would be the value used for the encoding of the luma signal. In particular, bestY′ would be the solution to the equation:  
    (7‑40)

### Luma adjustment – Bisection search

The Bisection search method is an iterative technique that is commonly used to derive the roots of an equation f(x) = 0. The function f is assumed to be continuous, and defined over an interval [a, b], where f(a) and f(b) need to have opposite signs. For this method to work, the behaviour of f(x) within this interval needs to also be monotonic (i.e. consistently increasing or decreasing). At each iteration of the search the interval is divided by two, i.e. at the midpoint c = (a + b) / 2). Then the value of the function f(c) is computed at this point. Depending on the value of f(c) and its relationship with f(a) and f(b) a new smaller interval is defined that satisfies the opposite sign condition. This search is repeated until a root is found, the interval is sufficiently small, or if a certain maximum number of iterations has been achieved.

This method can be used to find the bestY′ value for luma, as discussed in the previous section, in the following way:

Let be the value that represents the quantized representation of the luma component, as defined in Section 7.1.1. Furthermore, let be the dequantization function (Section 10.1.4), that maps the value of *x* back to its original representation and essentially a value between [0, 1]. For the narrow range, and assuming a 10-bit representation, we will then have:

. (7‑41)

Now let be the function:

(7‑42)

Let us assume that the initial interval is the entire range, i.e. [a, b] = [64, 940]. Given the characteristics of the ST 2084 transfer function it is expected that f(a) ≤ 0 and f(b) ≥ 0, which satisfy the bisection conditions. We can then compute the value of f(*x*) at position *x* = (a + b) / 2 = 502 and adjust the interval accordingly. If, for example, f(502) > 0 then the interval will be adjusted to [64,502]. The next evaluation point will then be the middle of this new interval, i.e. *x* = (64 + 502) / 2 = 283. At this point if now f(283) < 0 then the interval will be adjusted to [283,502]. The process can continue until either a value is found that satisfies f(x) = 0, or the interval is of the form [k, k+1], e.g. [343, 344]. In this case both cases can be evaluated and the one resulting in the smallest distortion for Y or iTF(Y) can be used.

NOTE 10 – Some reduction in complexity could be achieved if instead of trying to determine the optimal *x* as defined in the above equations, we try to determine the optimal value for *z* = *x* – 64.

A critical component of this method is the selection of the initial interval. The brute force approach is to use the entire valid range as the initial range, e.g. for the 10-bit narrow representation the range of [64, 940] as in the previous example. This, though, will require a number of iterations equal to the target bitdepth of the content to reach an interval of size one. However, using information about the original pixel and the chroma values , upper and lower bounds can be found for the value of bestY′, greatly reducing the size of the initial interval. This can have a direct impact on the number of computations performed, and thus the complexity of the process. For example, in the 10-bit case, the number of average iterations may be reduced from around ten to less than two. Three such bounds are preferably used. The first is described in [11], and means that if we define

(7‑43)

(7‑44)

. (7‑45)

we know that is in the interval The proof for this is out of the scope of this document, but is presented in [11]. The second bound uses the fact that all three variables R′bound, G′bound, and B′bound are smaller than 1, we can use a tighter upper bound, i.e. . This makes use of the fact that the transfer function is convex [11]. The third bound relies on the fact that all three of the reconstructed color components R′\*, G′\*, and B′\* cannot simultaneously be smaller (or all simultaneously larger) than the original color components R′, G′, and B′for the computation of bestY′. This means that if we define:

(7‑46)

(7‑47)

(7‑48)

then we know that the value for bestY′ must be in the interval:

(7‑49)

By combining all three bounds together the minimum and maximum bounds for Y′ can be computed as:

(7‑50)

(7‑51)

Finally the initial interval [*a*, *b*] for x is calculated as and where is the inverse of (7‑41).

  


Figure 7‑9  – Tone-mapped examples showing improvments of luma adjustment. Left column: Originals in 4:4:4. Middle column: Using traditional subsampling as described in Section 7.1.3 and keeping luma unaltered. Right column: Using the luma adjustment technique described in this section to change luma.

Figure 7‑9 shows an example of what the difference can be when doing traditional subsampling as described in Section 7.1.3 and using the luma component unchanged (middle column) compared to the luma adjustment method described in this section (right column). These images were processed in the Rec.709 color space to more easily show the effects. Since the printed medium cannot reproduce HDR images, tone-mapped versions are calculated using RSDR = clip(255\*(RHDR\*2c)1/), where clip( ) clips to the interval [0, 255], =2.2 and c is an exposure parameter set to make the SDR image look similar to the HDR image. Note how the artifacts clearly visible with traditional subsampling are avoided when using Luma Adjustment.

# Encoding process

[Ed. (CF): should this block of text in specification style language (“inputs of this process”) be removed as it was in the pre-processing stage]

Inputs of this process are:

– a (PicWidthInSamples)x(PicHeightInSamples) array PicSampleL of integer luma samples in the range of 0 to 1023, inclusive,

– a (PicWidthInHalf)x(PicHeightInHalf) array PicSampleSubCb of integer chroma samples of the component Cb in the range of 0 to 1023, inclusive,

– a (PicWidthInHalf)x(PicHeightInHalf) array PicSampleSubCr of integer chroma samples of the component Cr in the range of 0 to 1023, inclusive.

Output of this process is a bitstream.

This document does not provide a detailed description of the encoding process or the bitstream format. Section 8.1 provides information on suitable settings of syntax elements when creating bitstreams in accordance with HEVC [7]. Section 8.2 provides information on suitable settings of syntax elements when creating bitstreams in accordance with MPEG-4 AVC/H.264 [1].

## HEVC encoding

When creating the HEVC bitstream it is recommended to set the syntax elements listed in Table 8‑1 to the values listed in Table 8‑1 in each Sequence Parameter Set (SPS) in the bitstream. The syntax elements in Table 8-1 below are conveyed in the Video Usability Information syntax branch of the SPS defined in Annex E of the HEVC specification. They may also be duplicated and carried in various application layer headers.

Table 8‑1 – Recommended settings for HEVC encoding

|  |  |
| --- | --- |
| **Syntax element** | **Recommended value** |
| **general\_profile\_space** | 0 |
| **general\_profile\_idc** | 2 |
| **video\_full\_range\_flag** | 0 |
| **colour\_primaries** | 9 |
| **transfer\_characteristics** | 16 |
| **matrix\_coeffs** | 9 |
| **chroma\_sample\_loc\_type\_top\_field** | 2 |
| **chroma\_sample\_loc\_type\_bottom\_field** | 2 |

For HDR content represented with the colour primaries of BT.2020 [3] and the transfer function of ST 2084 [9], the video characteristics is typically different compared to the video characteristics of SDR content represented with BT.709 colour primaries and BT.709 OETF (BT.1886 EOTF) transfer function [2]. It is recommended to adjust the bit-distribution between chroma and luma for example by setting chroma QP offset (controlled by the HEVC syntax elements pps\_cb\_qp\_offset, slice\_cb\_qp\_offset, pps\_cr\_qp\_offset and slice\_cr\_qp\_offset) such that a small negative offset is used for low luma QP values and a large negative offset is used for high luma QP values. It is further recommended to adjust the bit-distribution between dark samples and bright samples for example by setting delta QP (controlled by the syntax elements cu\_qp\_delta\_abs and cu\_qp\_delta\_sign\_flag) such that blocks with a high averge luma value are assigned lower QP than blocks with a low average luma value. It is also recommended to take into account the activity (variance) of a block when setting delta QP for the block.

## AVC encoding

When creating the AVC bitstream it is recommended to set the syntax elements listed in Table 8‑2 to the values listed in Table 8‑2 in each Sequence Parameter Set in the bitstream. The syntax elements in Table 8-2 below are conveyed in the Video Usability Information syntax branch of the SPS defined in Annex E of the AVC specification. They may also be duplicated and carried in various application layer headers.

Table 8‑2 – Recommended settings for AVC encoding

|  |  |
| --- | --- |
| **Syntax element** | **Recommended value** |
| **profile\_idc** | 110 |
| **video\_full\_range\_flag** | 0 |
| **colour\_primaries** | 9 |
| **transfer\_characteristics** | 16 |
| **matrix\_coeffs** | 9 |
| **chroma\_sample\_loc\_type\_top\_field** | 2 |
| **chroma\_sample\_loc\_type\_bottom\_field** | 2 |

For HDR content represented with the colour primaries of BT.2020 [3] and the transfer function of ST 2084 [9], the video characteristics is typically different compared to the video characteristics of SDR content represented with BT.709 colour primaries and BT.709 transfer function [2]. It is recommended to adjust the bit-distribution between chroma and luma for example by setting chroma QP offset (controlled by the syntax elements chroma\_qp\_index\_offset and second\_chroma\_qp\_index\_offset) such that a small negative offset is used for low luma QP values and a large negative offset is used for high luma QP values. It is further recommended to adjust the bit-distribution between dark samples and bright samples for example by setting delta QP (controlled by the syntax element mb\_qp\_delta) such that blocks with a high averge luma value are assigned lower QP than blocks with a low average luma value. It is also recommended to take into account the activity (variance) of a block when setting delta QP for the block.

# Decoding process

[Ed. (CF): should this block of text in specification style language (“inputs of this process”) be removed as it was in the pre-processing stage]

Input to this process is a bitstream.

Outputs of this process are:

– a (PicWidthInSamples)x(PicHeightInSamples) array PicSampleL of integer luma samples in the range of 0 to 1023, inclusive,

– a (PicWidthInHalf)x(PicHeightInHalf) array PicSampleSubCb of integer chroma samples of the component Cb in the range of 0 to 1023, inclusive,

– a (PicWidthInHalf)x(PicHeightInHalf) array PicSampleSubCr of integer chroma samples of the component Cr in the range of 0 to 1023, inclusive.

When the bitstream is an HEVC bitstream the decoding process in the HEVC specification [7] is performed with the sample values of each output picture iteratively assigned to PicSampleL, PicSampleCb, and PicSampleCr, respectively.

When the bitstream is an AVC bitstream the decoding process in the MPEG-4 AVC/H.264 specification [1] is performed with the sample values of each output picture iteratively assigned to PicSampleL, PicSampleCb, and PicSampleCr, respectively.

# Post-decoding processes

The post-decoding stage includes the following components:

1. a conversion component that converts a fixed-point representation, i.e. 10 bits, to a floating-point representation,
2. a chroma up-conversion component that converts data from 4:2:0 to 4:4:4,
3. a colour format conversion component that converts data from the Non-Constant Luminance Y′CbCr representation back to a the non-linear R′G′B′ representation, and
4. a conversion component from the non-linear data representation back to a linear data representation.

NOTE 11 – As was also in the case of pre-encoding, image scaling might also be desirable during this stage. For example, up‑scaling the decoded data from a 1920x1080 to a 3840x2160 resolution is highly likely given the prevalence of 4K HDR displays. Again, we will not elaborate on this topic, however, we would suggest that rescaling is performed in either the linear domain, or at least using the R′G′B′ representation.   
[Ed. (JS): If the suggestion should be kept in this note, there needs to be results to back up the suggestion.]

Figure 10‑1 presents a diagram of how these components could potentially be combined together to generate the desirable outcome, in a conventional manner. In this system, all blocks work independently, whereas chroma up‑sampling is performed using fixed-point arithmetic and its outcome is at the same precision as the input signal.



Figure 10‑1 – Conventional post-decoding process system diagram.

The various components of this stage are better introduced in the Section 10.1. More specifically, sub-section 10.1.1 describes the conversion process from a fixed-point representation back to a floating point representation, sub-section 10.1.2 discusses chroma up-conversion, sub-section 10.1.3 describes the colour representation conversion, i.e. from Y′CbCr back to R′G′B′ and sub-section 10.1.4 describes the conversion steps from a non-linear representation back to a linear one. Section 10.2, on the other hand, also introduces a few alternatives to the configuration of Figure 7‑1, that can provide different performance/complexity trade-offs.

## Post-decoding processes

### Conversion from a fixed-point to a floating-point representation

This process can be seen as the exact inverse of the process presented in Section 7.1.4. In particular, a fixed-point precision value can be converted to a floating-point precision value using the following formula:

(10‑1)

The exact same values for scale and offset as in Section 7.1.4 are used according to the component type, whereas minE and maxE are equal to -0.5 and 0.5 for the chroma components respectively, and equal to 0 and 1.0 for all other colour components.

### Chroma up-conversion

Chroma plane interpolation both vertically and horizontally is performed to convert the 4:2:0 NCL Y′CbCr signal to a 4:4:4 representation. Similar to the down-conversion process in Section 7.1.3, this step needs to again account for the siting of the chroma components compared to those of the luma (Figure 7‑3).

It is quite likely that FIR linear filters would be used by many implementations for this process. Similar to the down-conversion case, it has been reported that non-linear and/or adaptive filters could result in improved performance, but this work is still under investigation. Regardless, it has been observed that the 2-phase resampling filter shown in Table 10‑1 could provide relatively reasonable performance results on highly compressed HDR10 material. The same filter is applied both vertically and horizontally.

NOTE 12 – This is essentially a Lanczos 2 filter. Higher precision and order filters could potentially be used when up‑sampling content of very high quality/no compression.

Table 10‑1 – 2-phase chroma resampling filter

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Phase p** | **Interpolation filter coefficients** | | | |
| fC[ p, -1 ] | fC[ p, 0 ] | fC[ p, 1 ] | fC[ p, 2 ] |
| 0 | 0 | 1 | 0 | 0 |
| 1 | −1/16 | 9/16 | 9/16 | −1/16 |

### Colour representation conversion: Non Constant Luminance Y′CbCr to R′G′B ′

Conversion from the non-constant luminance Y′CbCr representation back to the R′G′B′ representation can be performed using the following formula:

(10‑2)

Since , all the matrix coefficients in **A** can be calculated directly from *wYR*, *wYG*, *wYB*. In particular, given the characteristics and precision of the coefficients used in Section 7.1.2, we can compute that:

, (10‑3)

(10‑4)

(10‑5)

(10‑6)

(10‑7)

(10‑8)

Basically, if high precision is possible, it is recommended that the following matrix should be used:

(10‑9)

For systems with limited precision, a lower precision representation of the above matrix, that retains only 6 digits of precision, could be used instead.

### Conversion from a non-linear to a linear light representation: R′G′B′ to RGB

Conversion from a non-linear to a linear light representation is performed using a forward transfer function, or as is commonly referred to other specifications, an electrical-optical transfer function (EOTF). In the context of HDR10, the exact inverse of the process described in Section 7.1.1 is used.

More specifically, the linear light intensity signal Lo can be computed from the non-linear representation V, which takes values in the range [0, 1], as follows:

(10‑10)

where c1, c2, c3, m, and n were defined in Section 7.1.1. In Figure 10-2 below, notable points on the plot include: non-linear real-value signal value 0.5 (corresponding to 10-bit narrow code level 502) maps to 0.01 real-value signal output (0.01 \* 10 000 = 100 nits), while input value 0.9 (10-bit code level 853) maps to 0.4 (4000 nits).



Figure 10‑2 – Behaviour of the ST 2084 (PQ) forward transfer function.

This process is applied to all R′, G′, and B′ non-linear representations, resulting in their linear light counterparts R, G, and B as follows.

R = TF( R′ ) (10‑11)

G = TF( G′ ) (10‑12)

B = TF( B′ ) (10‑13)

LUTs, as was also the case for the linear to non-linear conversion, could also be used instead of using directly Equation 10‑10. In that scenario, a non-uniform LUT interpolator is also recommended.

## High precision post-decoding

[Ed. (JS): This section was discussed at the telephone conference on May 9, 2016. It has not been concluded whether this should be included or not. If it should be included there needs to be results tpresented that verify the claims.]

Figure 7‑6 presents an alternative configuration where chroma subsampling is performed prior to the conversion from a floating-point representation to a fixed-point representation. This can present an advantage since it can reduce the effect of quantization errors during filtering and subsampling, therefore resulting in a higher quality final representation. This comes, however, at a higher complexity/implementation cost that may not be feasible in some systems.



Figure 10‑3 – Alternative post-decoding process system with high precision (floating-point) upsampling.

As a compromise to the system presented in Figure 7‑6, one may still utilize fixed-point, but higher precision than the final target, arithmetic operations for certain stages of the conversion process. In Figure 7‑7, for example, the video data are first converted to a fixed-point representation utilizing N-bits, with N > 10 such as N=12 or N=14, immediately after the linear to non-linear representation conversion. Such conversion could also be moved at a later stage of the processing pipe if it is so desired. In the shown example, R′G′B′ data in a fixed-point representation are directly converted to a Y′CbCr fixed-point representation, while chroma subsampling is also performed at this higher fixed precision representation. Finally, an additional conversion step is performed to convert the data to the final desired precision, i.e. in a 10-bit 4:2:0 Y′CbCr, limited/video range video signal.



Figure 10‑4 – 2nd alternative post-decoding process system with high precision (fixed-point) upsampling



Figure 10‑5 – 3rd alternative post-decoding process system with efficient fixed precision utilization

Annex A. Supplemental enhancement information (SEI) messages.

This annex provides short descriptions of SEI messages that can be used together with HDR video.

* 1. Mastering display colour volume SEI message

If mastering display colour volume information is included, it is recommended that Mastering display colour volume SEI messages are included at least at each Random Access Point Access Unit (RAPAU). The information provided in the mastering display colour volume information SEI message shall apply until, but not necessarily including, the next RAPAU. If multiple mastering display colour volume SEI messages are included in the bitstream between the start of two RAPAUs then those SEI messages shall have the same content. Table A.1­ shows an example of what values the Master display colour volume SEI message would contain in case the mastering display uses P3 colour primaries [RP 431-2], D65 white point and luminance range of 0 cd/m2 to 2000 cd/m2, inclusive.

Table A.1­ – Example Mastering display colour volume SEI message representing P3 colour primaries [RP 431-2], D65 white point and luminance range of 0 cd/m2 to 2000 cd/m2, inclusive.

|  |  |
| --- | --- |
| **Syntax element** | **Example value** |
| **display\_primaries\_x[0]** | 13250 |
| **display\_primaries\_y[0]** | 34500 |
| **display\_primaries\_x[1]** | 7500 |
| **display\_primaries\_y[1]** | 3000 |
| **display\_primaries\_x[2]** | 34000 |
| **display\_primaries\_y[2]** | 16000 |
| **white\_point\_x** | 15635 |
| **white\_point\_y** | 16450 |
| **max\_display\_mastering\_luminance** | 20000000 |
| **min\_display\_mastering\_luminance** | 0 |

If the decoder contains an interface for output of mastering display colour volume information and the bitstream contains mastering display colour volume SEI messages it is recommended that the decoder outputs mastering display colour volume information synchronously to the first picture for which the SEI message apply.

* 1. Content Light Level information SEI message

[Ed. (JS): Add description and example.]

Blu-Ray Disc Association BD-ROM 3.0 UltraHD specification [12] provides guidelines that MaxFALL not exceed 400 nits, and that only a small percentage of pixels, such as specular highlights, exceed 1000 nits. For the BD-ROM 3.0 application, these measures are taken off-line during a complete pass through the content before the final MaxFALL and MaxCLL calculation is made. Live or broadcast services may set these values to pre-determined levels without actual measurement on content.