SMPTE REGISTERED DISCLOSURE DOCUMENT

LLVC – Low Latency Video Codec for Network Transfer

The attached document is a Registered Disclosure Document prepared by the proponent identified below. It has been examined by the appropriate SMPTE Technology Committee and is believed to contain adequate information to satisfy the objectives defined in the Scope, and to be technically consistent.

This document is NOT a Standard, Recommended Practice or Engineering Guideline, and does NOT imply a finding or representation of the Society.

Errors in this document should be reported to the proponent identified below, with a copy to eng@smpte.org.

All other inquiries in respect of this document, including inquiries as to intellectual property requirements that may be attached to use of the disclosed technology, should be addressed to the proponent identified below.

Proponent contact information:

Toshiaki Kojima
Sony Corporation
4-14-1 Asahi-cho, Atsugi
Kanagawa, 243-0014
Japan

Email: Toshiaki.Kojima@jp.sony.com

Approved
September 1, 2015

Table of Contents

Page

Introduction

The highest speed network environment that is readily available is 10GB Ethernet. Considering the bandwidth required for high resolution HD and UHDTV video, 10GB is not necessarily sufficient and it is desirable to implement some degree of video compression during transfer across the network. The major requirements for such a video compression scheme are low latency, and high picture quality. However, there is a trade-off between low latency, compression ratio, and picture quality. The video compression described in this RDD provides low latency of less than one video frame and is able to offer visually lossless quality.

1 Scope

This RDD describes the ‘Low Latency Video Codec' (LLVC) and related technical information; in particular, a description is given of an example decoder implementation with sub-sampling scheme of 4:2:2 and 4:4:4.

The following items are four main elements described in this RDD:

a) The codec characteristics: brief introduction to the technologies which offer low memory/low latency and high quality simultaneously. Block diagrams of the encoding and decoding processes are also shown.

b) The codestream: a compressed image data representation which includes all necessary data to allow a (full or approximate) reconstruction of the sample values of a digital image. Additional data might be required that define the interpretation of the sample data, such as the spatial dimensions of the samples.

c) A decoder: the decoder takes as input a codestream, and by means of a specified set of procedures generates as output digital reconstructed image data. Sufficient information is provided to enable an expert skilled in the art of wavelet-based image compression to be able to construct a compatible decoder.

d) IP mapping: a simple introduction to the IP mapping of the codestream.

2 Mathematical Operators

2.1 Arithmetic Operators

+	Addition
–	Subtraction (as a binary operator) or negation (as a unary prefix operator)
<<	Left shift
=	Assignment
++	Increment by one
<	Less than
<=	Equal to or less than
&&	Logical AND
==	Equal to
!=	Not equal to

2.2 Mathematical Functions

Ceil(x)	Ceiling of x. Returns the smallest integer that is greater than or equal to x.
Log2(x)	Logarithm of x to the base 2
0^n, 1^n	sequence of n bits of same value (0 or 1)

3 Terms and Definitions

3.1 Codestream

Compressed data followed by IP mapping. Codestream consists of three layers of Picture, Transmission Unit (TU) and Coding Unit (CU). (For details, see Section 5 Structure of Codestream).

3.2 Coefficient

The values that are result of a transformation.

3.3 Component

A two-dimensional array of samples having the same designation in the output or display device. An image typically consists of several components, e.g. red, green and blue.

3.4 CU

Coding unit which represents process unit of the codec. There are two kind of CU, LB (Low Band) CU and HB (CU). The number of CUs in on TU depends on video sub-sampling scheme.

3.5 Decomposition level

A collection of wavelet sub-bands where each coefficient has the same spatial impact or span with respect to the source component samples. The decomposition level is created by recursive multi-wavelet transforms.

3.6 Entropy decoding

A lossless procedure which recovers the sequence of symbols from the sequence of bits produced by the entropy encoder.

3.7 Entropy encoding

A lossless procedure which converts a sequence of input symbols into a sequence of bits such that the average number of bits per symbol approaches the entropy of the input symbols.

3.8 HB CU

High Band Coding Unit. In our implementation, HB CU adopts single wavelet transform and has one decomposition level. 5 HB CUs are included in one TU in the case of 4:2:2 while 9 HB CUs are included in one TU in the case of 4:4:4 video sub-sampling scheme.

3.9 LB CU

Low Band Coding Unit. In our implementation, LB CU adopts three-wavelet transform and has 3 decomposition levels. 3 LB CUs are included in one TU in both 4:2:2 and 4:4:4 video sab-sampling schemes.

3.10 Picture

Segment of codestream for one frame or field.

3.11 Precision

Number of bits allocated to a particular sample, coefficient, or other binary numerical representation.

3.12 Quantization

A method of reducing the precision of the individual coefficients to reduce the number of bits used to entropy code them. This is equivalent to division while compressing and multiplying while decompressing. Quantization can be achieved by an explicit operation with a given quantization value.

3.13 Sub-band

A group of transform coefficients resulting from the same sequence of low-pass and high-pass filtering operations, both vertically and horizontally.

3.14 TU

Transmission Unit of the codec. In our implementation, one TU includes 16 line-video information because three decomposition level is adopted in both 4:2:2 and 4:4:4 video sub-sampling schemes.

3.15 Word

A string of bits of fixed length treated as a unit of processing. 1 word = 128bit.

3.16 VLC

Variable Length Coding. The length of an entropy coded codestream is variable, because it depends on the condition of the input data to the entropy coder.

3.17 VLD

Variable Length Decoding which decodes VLC coded codestream.

4 Overview of Codec Characteristics

4.1 Codec Key Technologies

The codec is designed to offer low memory/low latency and high quality simultaneously. The condition of low latency is a trade-off relation with high quality. If the memory size is large enough, rich rate control in the encoder is possible, which leads to higher picture quality. On the other hand, rate control with smaller memory size buffering leads to lower latency.

- Technologies for Low Latency:
  1. Line-based wavelet transform / Inverse wavelet transform
  2. Lifting operation in wavelet filtering

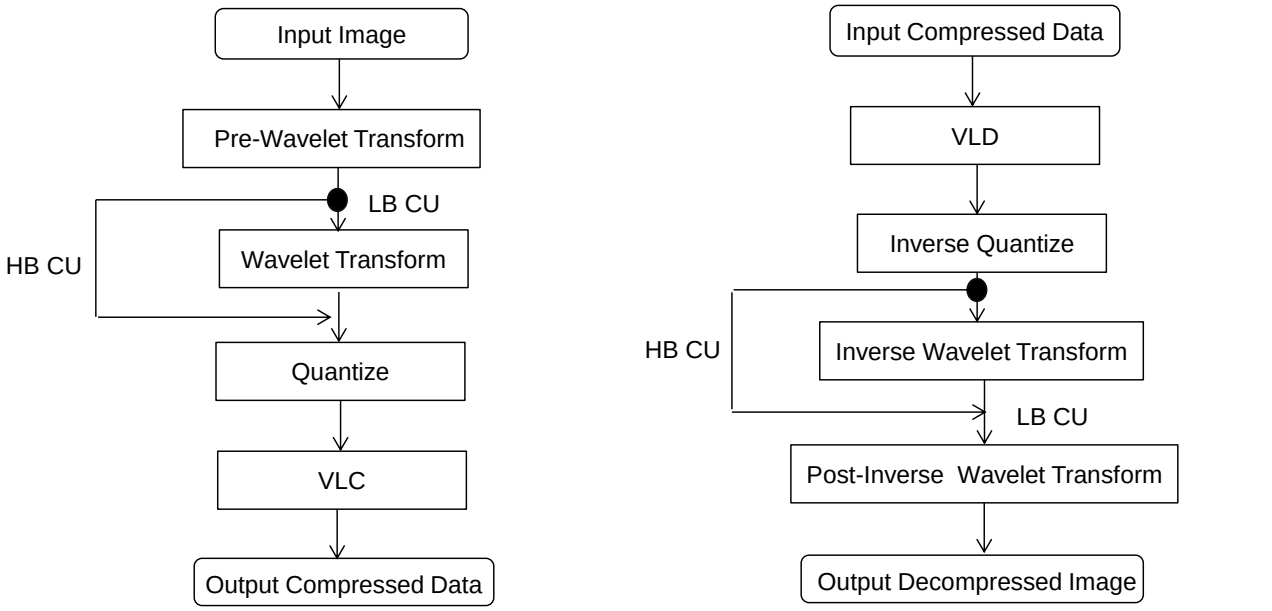
     （Combination of line-based wavelet transform and lifting operation）
  3. Transmission Unit (TU)-based encoding and decoding
  4. Line-based entropy encoding (VLC: Variable Length Coding) and entropy decoding (VLD: Variable Length Decoding)
  5. Parallel processing of encoding and decoding simultaneously

     （Decoding can start whilst encoding a picture）

- Technologies for High Image Quality:
  1. High performance wavelet transform/Inverse wavelet transform (5-3 filter for lower latency)
  2. Highly efficient VLC in encoder and VLD in decoder
  3. Rate control in each TU and sub-band (depends on encoder)
  4. Frequency weighting to wavelet coefficients (optional in encoder)

4.2 Encoding/Decoding Block Diagram

Figure 4.1 shows a block diagram of the encoding and decoding processes. The encoder consists of pre-wavelet transform, wavelet transform, quantization and VLC. The decoder consists of VLD, inverse quantization, inverse wavelet transform and post-inverse wavelet transform.



HB CU: High-Band Coding Unit
LB CU: Low-Band Coding Unit

Figure 4.1 – Encoding / Decoding Block Diagram

5 Structure of Codestream

5.1 Overall

Codestream is a compressed data stream consisting of three layers: Picture, Transmission Unit (TU) and Coding Unit (CU), Picture is for one frame or field.

The data structure of the three elements is shown in Figure 5.1. The Picture layer consists of a Picture_info header followed by a total number of K TUs.

Each TU consists of a TU_info header followed by TU_body which comprises a total number of N CUs. Each CU consists of a CU_info header and CU_body that conveys compressed data as described in Section 5.4, CU (Coding Unit).

Note: Picture_info, TU_info and CU_info are not compressed data.

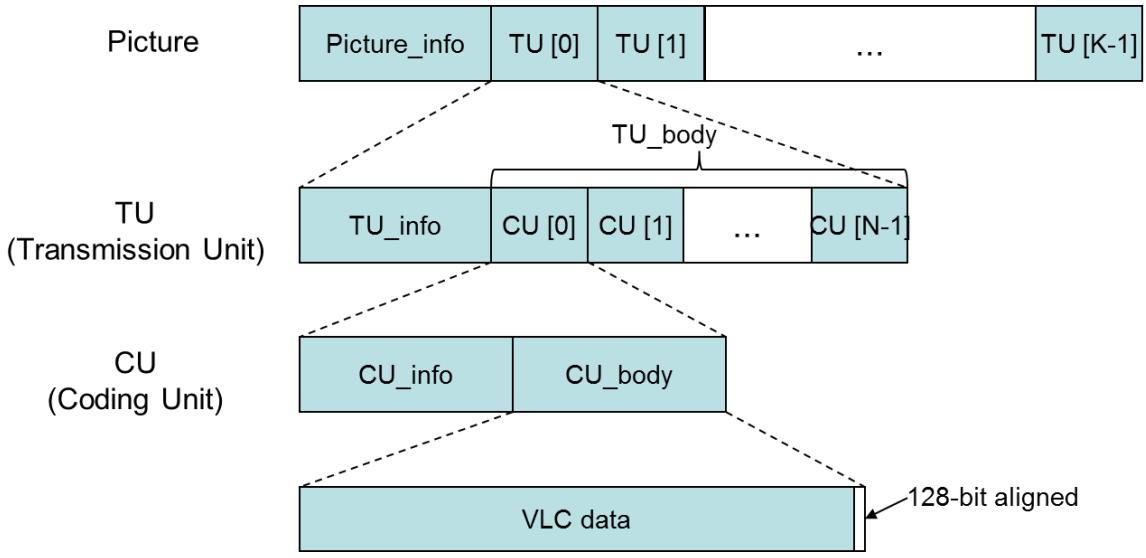


Figure 5.1 – Data Structure of Picture, TU and CU

5.2 Picture

A Picture consists of Picture_info and multiple TUs. The number of TUs (K) depends on the picture resolution and the number of decomposition levels of the wavelet transform.

In our implementation, one TU includes 16 lines of the video image because three decomposition levels are adopted in both 4:2:2 and 4:4:4 video sub-sampling schemes. (The decomposition level is discussed in Section 5.4.) Therefore K shall be the smallest integer not smaller than VS/16. For example the number of TUs (K) in the 1080P video format is 68 (1080/16 = 67.5).

The parameters of Picture_info are shown in Table 5.1.

Table 5.1 – Picture_info Parameters

Parameter	Meaning	Size (bits)
Reserved[1]		64
HS[2]	Horizontal size (e.g. 1920)	16
VS[3]	Vertical size (e.g. 1080)	16
Reserved		6
BBD	Baseband bit-depth (e.g. 10)	6
Reserved		4
NC	Number of components	3
NW	Number of wavelet transform decomposition level (in the case of 4:2:2 or 4:4:4, 3 is adopted in our implementation)	3
NP	Magic code of chrominance sub-sampling (1101010000B: 4:2:2, 1111110000B: 4:4:4)	10

[1] Values in 'Reserved' fields shall be ignored. (Same for all 'Reserved' fields in Table 5.1, 5.2 and 5.3.)
[2] HS = Horizontal size of Component(0). (See Figure 6.16 and Figure 6.17 in Section 6).
[3] VS = Vertical size of Component(0) (See Figure 6.16 and Figure 6.17 in Section 6)
(NW is less than or equal to 3.)

5.3 TU (Transmission Unit)

A TU consists of TU_info and multiple CUs. The number of CUs required (N) depends on the sub-sampling scheme.

Figure 5.2 and Figure 5.3 illustrate CU allocation on 4:2:2 and 4:4:4 video formats respectively. 8 CUs (3 LB CUs and 5 HB CUs) are used in the case of 4:2:2, while 12 CUs (3 LB CUs and 9 HB CUs) are used in the case of 4:4:4 video sub-sampling scheme. (LB CU and HB CU are discussed in Section 5.4.)

In other words, a TU shall contain 8 CUs (N = 8) in the case of 4:2:2, 12 CUs (N = 12) in the case of 4:4:4.

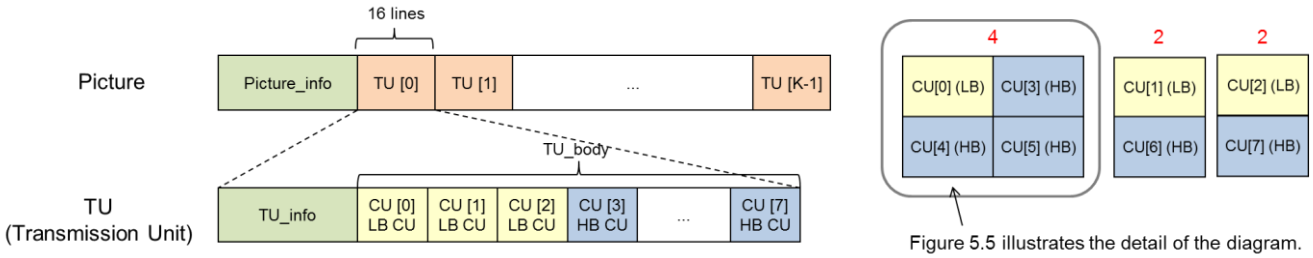


Figure 5.2 – CU Allocation on 4:2:2 Video Sub-sampling Scheme

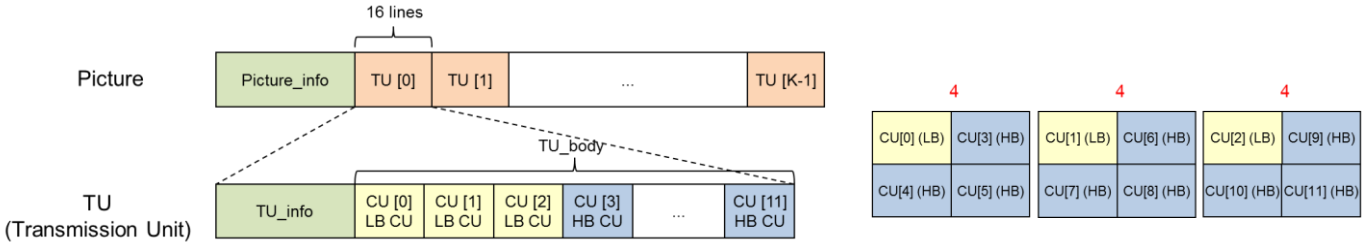


Figure 5.3 – CU Allocation on 4:4:4 Video Sub-sampling Scheme

The parameters of TU_info are shown in Table 5.2.

Table 5.2 – TU_info Parameters

Parameter	Meaning	Size (bits)
Reserved		48
TUI	Transmission unit index (0~(K-1)) (See Figure 6.1 and Figure 7.4)	16
Reserved		48
LG	The number of words of TU_body.	16

5.4 CU (Coding Unit)

The CU consists of CU_info and CU_body. The parameters of CU_info are shown in Table 5.3.

Table 5.3 – CU_info Parameters

Parameter	Meaning	Size (bits)
Reserved		56
CUI	Coding unit index (0 - (N-1)) (See Figure 5.2 and Figure 5.3)	8
Reserved		8
QI[0]	Quantization Information for sub-band S0	4
QI[1]	Quantization Information for sub-band S1	4
QI[2]	Quantization Information for sub-band S2	4
QI[3]	Quantization Information for sub-band S3	4
QI[4]	Quantization Information for sub-band S4	4
QI[5]	Quantization Information for sub-band S5	4
QI[6]	Quantization Information for sub-band S6	4
QI[7]	Quantization Information for sub-band S7	4
QI[8]	Quantization Information for sub-band S8	4
QI[9]	Quantization Information for sub-band S9	4
LG	The number of words of CU_body.	16

Figure 5.4 illustrates two dimensional tree-structured filter bank of our implementation for decoding from wavelet coefficients stored in a TU into decompressed picture of component (0). (4 part in 4:2:2 sub-sampling scheme.) Three recursive inverse wavelet transforms are required for decoding CU[0] (LB CU). The result of the inverse wavelet transforms and HB CUs (CU[3], CU[4] and CU[5]) shall be post-inverse wavelet transformed to generate the final decompressed picture of a TU.

The details of a 4-input/1-output two dimensional inverse wavelet transform comprised of two 2-input/1-output vertical inverse wavelet transforms and a 2-input/1-output horizontal inverse wavelet transform are described in Section 6.5.

1 line of wavelet coefficients is compressed in the decomposition level 3 CUs, 2 lines of wavelet coefficients are compressed in the decomposition level 2 CUs, 4 lines of wavelet coefficients are compressed in the decomposition level 1 CUs and 8 lines of wavelet coefficients are compressed in the decomposition level 0 CUs, finally 16 lines of final decompressed picture stored in a TU can be decoded.

Figure 5.5 shows a passband structure representation of Figure 5.4 in terms of TU[0]. LB CU (CU[0]) has 10 sub-bands (S0 - S9) while HB CU (CU[3], CU[4] and CU[5]) has only one sub-band (S0).

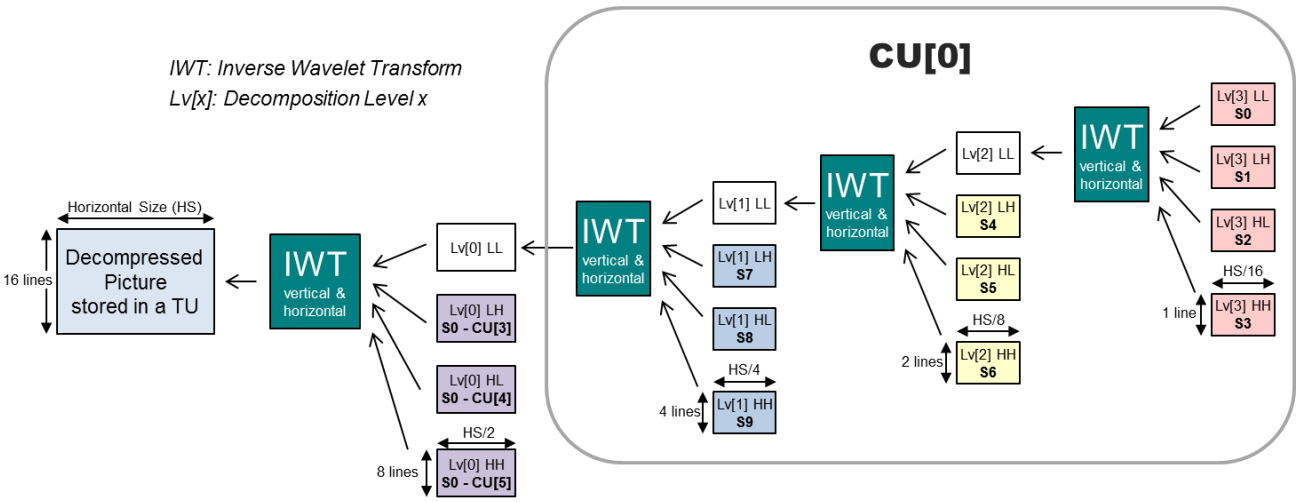


Figure 5.4 – Relationship between TU and CUs (Filter Bank Structure)

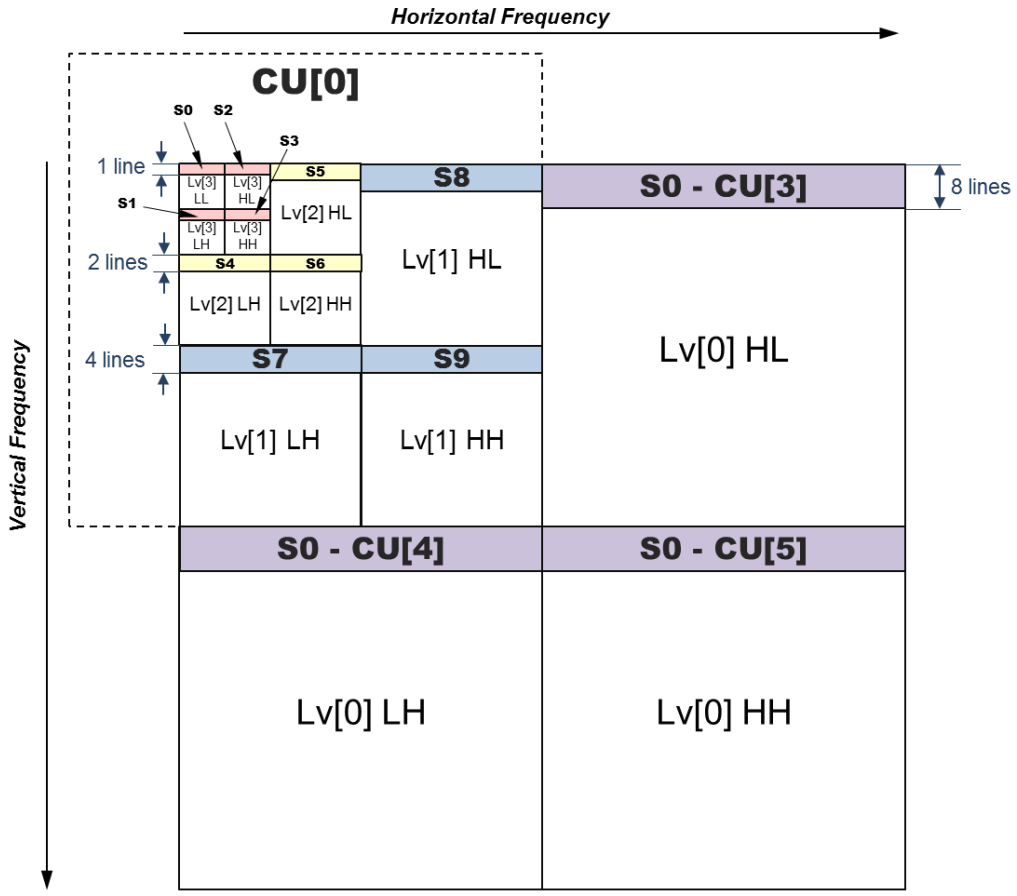


Figure 5.5 – Passband Structure Representation of Figure 5.4

Figure 5.6 shows the mapping of TUs (TU[0], TU[1], TU[2],….) with respect to LB CU and HB CU.

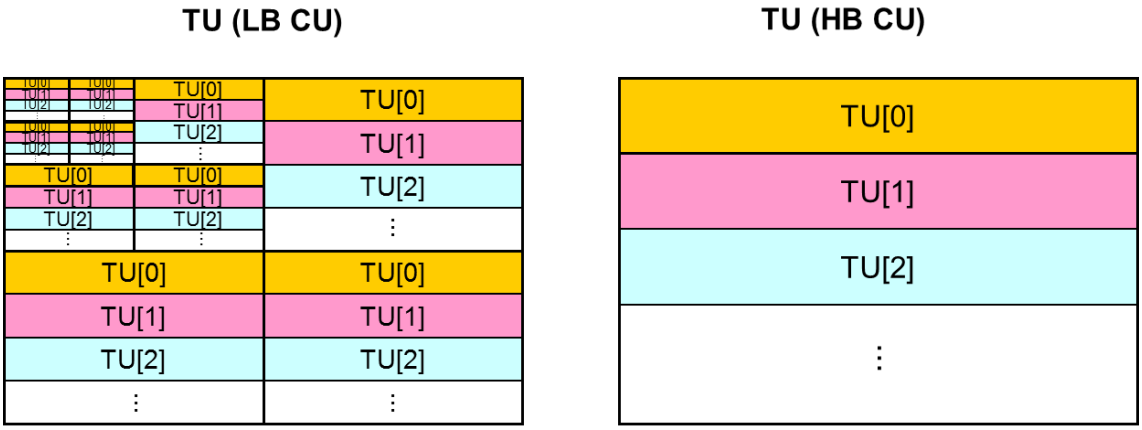


Figure 5.6 – TU Mapping

6 Video Decoder

6.1 Video Decoder Flow

Figure 6.1 is a flow chart for video decoding. Details of video decoder elements are described in the following sections.

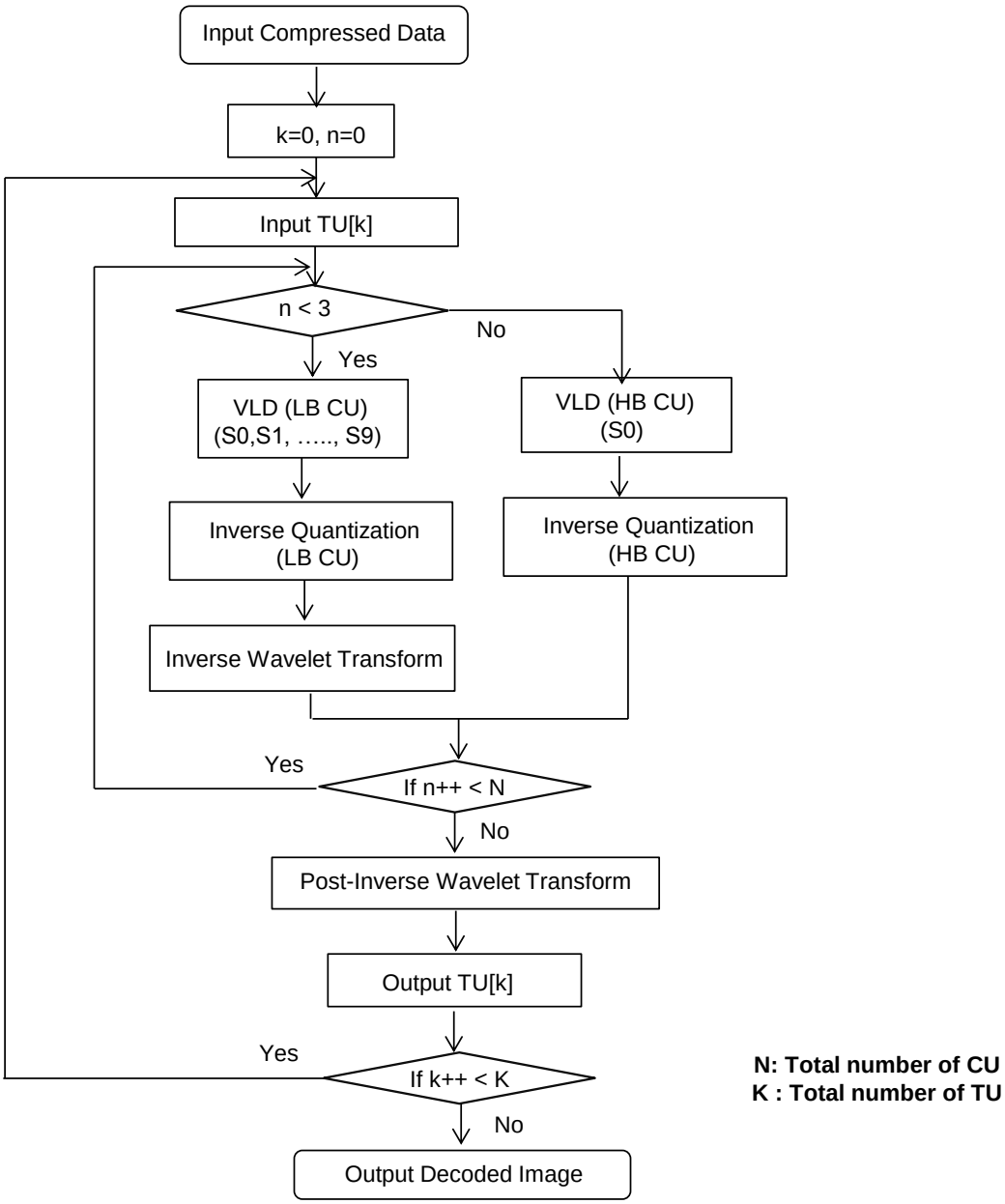


Figure 6.1 – Video Decoder Flow

6.2 Scanning of Coefficients

VLD is employed line by line and each line is segmented in multiples of 4 coefficients.

In Figure 6.2, 4 coefficients (C1, C2, C3 and C4) are decoded with the VLD method. The last set of 4 coefficients shall be decompressed in the same manner and the decoded coefficients out of the image boundary are discarded. In this decoding method, a VLD coefficients set is denoted as CoefSet.

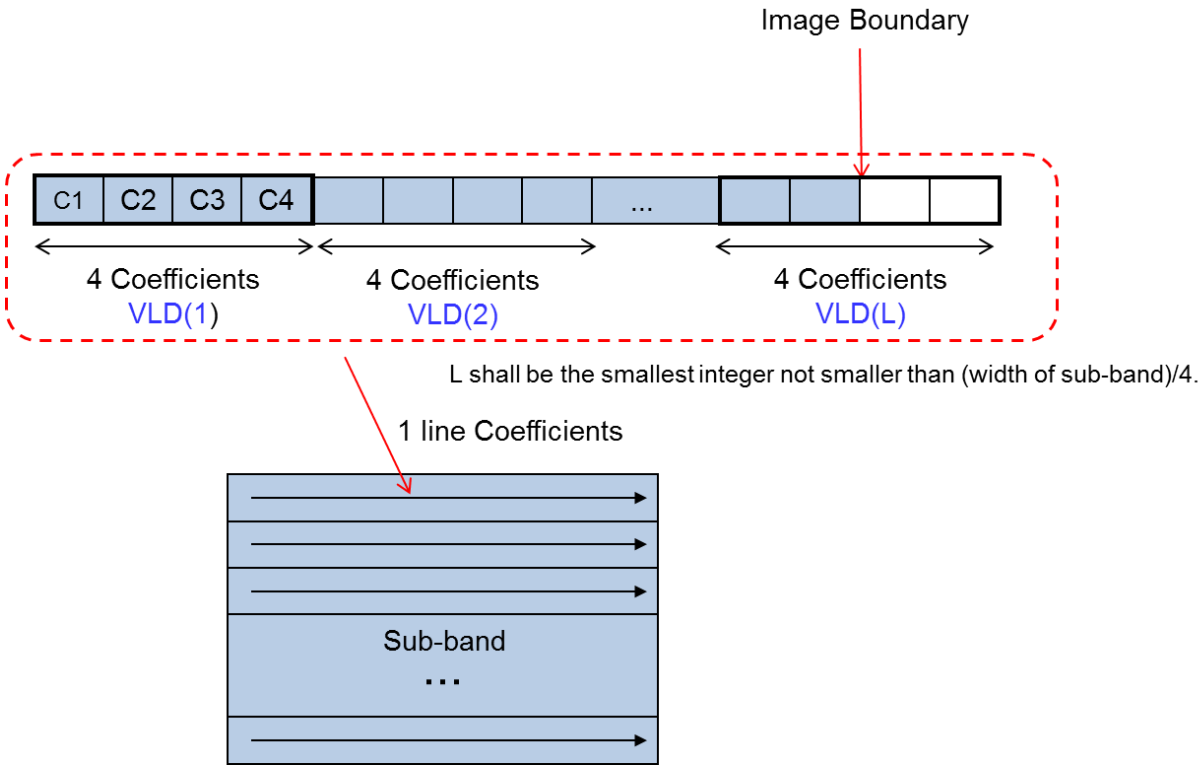


Figure 6.2 – VLD Coefficients Sets (CoefSet) in One Line

6.3 VLD (Variable Length Decoding)

VLD (Variable Length Decoding) is a mechanism to decode compressed data. The output of the VLD is then inverse quantized as described in Section 6.4.

There are three key values that are calculated during VLD as follows:

1. **DPT value**: Bit-depth of absolute values of 4 coefficients (CoefSet)

2. **ABS value**: Absolute value of each coefficient

3. **SGN value**: Sign value of each coefficient

As shown in Figure 6.4, the VLD procedure consists of four stages; “Initialization”, “DPT Value Decoding”, “ABS Value Decoding” and “SGN Value Decoding”. The details of each stage are described in the following sections.

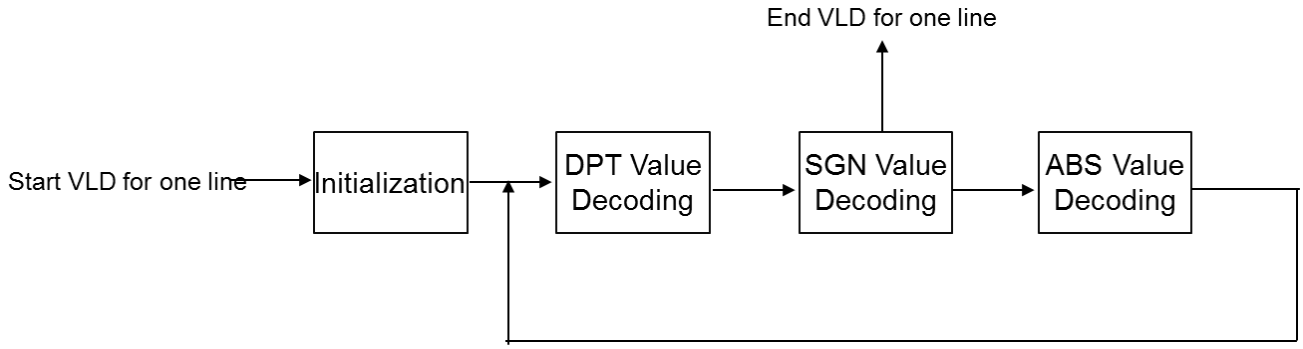


Figure 6.4 – VLD Block Diagram

6.3.1 VLD Parameters

VLD parameters and their meanings are listed in Table 6.1.

Table 6.1 – VLD Parameters

Parameter	Meaning
ABS[i]	Integer array representing ABS value (i = 0, 1, 2, or 3).
DPT	Integer value representing DPT value
SGN	Integer value representing SGN value
Cnt	Integer counter to locate the current horizontal position of processing
DPTs	Integer state value (0, 1, or 2) used for Decoding DPT
x	Horizontal position of output coefficients
ABSv	Boolean valid flag used for ABS Value Decoding
ABSe	Boolean end flag used for ABS Value Decoding
DPTv	Boolean valid flag used for DPT Value Decoding
DPTe	Boolean end flag used for DPT Value Decoding
ZR	Integer Zero-Run counter used for DPT Value Decoding

6.3.2 Initialization

At the Initialization stage in Figure 6.8, the parameters shall be initialized as shown in Figure 6.5.

```
ABS[0] = 0;
ABS[1] = 0;
ABS[2] = 0;
ABS[3] = 0;
DPT = 0;
SGN = 0;
Cnt = number of CoefSet in the current vertical position;
      (L in Figure 6.2)
DPTs = 0;
x  = 0;
ABSv = 0;
ABSe = 0;
DPTv = 1;
DPTe = 1;
ZR = 0;
```

Figure 6.5 – Method of Initialization

6.3.3 DPT Value Decoding

The decoding process is shown in Figure 6.6. There are four conditions to consider in decoding the DPT value, and these are shown in Figure 6.6 in order of precedence. The DPT operation and the state operation are defined by the combination of the condition and bit-pattern of the input compressed data.

If (Cnt == 0) DPTv = 0;
If (Cnt == 1) DPTe = 1;
If (Cnt != 0) Cnt = Cnt – 1;
If (ZR != 0) ZR = ZR – 1;

If (DPTv == 1) {

Condition	Bit-pattern of input compressed data	DPT Operation	State operation
DPTs == 0			
	0	No operation	No operation
	1 0 0^n 1	DPT = DPT+n+1; (0<=n)	No operation
	1 1 0^n 1	DPT = DPT–n-1; (0<=n<=DPT-2)	No operation
	1 1 0^(DPT-1)	DPT = 0;	DPTs = 1;
DPTs == 1			
	1 0^n 1	DPT = n+1;	DPTs = 0;
	0^(Ceil(Log2(Cnt+2)))	ZR = Cnt+2;	DPTs = 2;
	0^m 1	{ ZR = 1; for (i=0; i<m; i++) ZR = (ZR<<1) + get_next_bit(); }	DPTs = 2;
DPTs == 2 && ZR == 1	0^n 1	DPT = n+1;	DPTs = 0;
DPTs == 2 && ZR != 1	No input	No operation	No operation

}

Note: 0 <= n, 0 < m, get_next_bit() is extraction of next bit from input compressed data.

Figure 6.6 – Method of Decoding of DPT Value

6.3.4 SGN Value Decoding

The method of decoding the SGN value is shown in Figure 6.7.

```
If (ABTv == 1) {
  for (i = 0; i < 4; i++)  {
    XQ[b][x][y] = ABS[i]:
    if (ABS[i] != 0) {
       SGN = get_next_bit();
       if (SGN == 1) XQ [b][x][y]  = - XQ [b][x][y];
    }
    x++;
  }
}
If (ABSe == 1) end VLD for one line:
```

Note 1: X_Q [b][x][y] is output of the VLD, where y is vertical position and b is sub-band of current one line.

Note 2: get_next_bit() is extraction of next bit from input compressed data.

Figure 6.7 – Method of Decoding SGN Value

6.3.5 ABS Value Decoding

The method of decoding the ABS value is shown in Figure 6.8.

```
ABSv = DPTv;
ABSe = DPTe;
If (DPTv == 1) {
  for (i = 0; i < 4; i++)  {
     ABS[i] = 0;
     for (j = 0; j < DPT; j++)  {
       ABS[i] = (ABS[i] << 1) + get_next_bit();
     }
  }
}
```

Note: get_next_bit() is extraction of next bit from input compressed data

Figure 6.8 – Method of Decoding ABS Value

6.3.6 Decoding Example

Figure 6.9 shows an example of VLD. In this figure, decoding proceeds from top to end and no update is expected when the field is blank.

Stage	Input compressed data	Output CoefSet	Updated parameters								
			DPT	ABS[i]	DPTs	DPTv	DPSe	ABSv	ABSe	ZR	Cnt
Initialization			0	{0, 0, 0, 0}	0	1	0	0	0	0	8
DPT decoding	100001B		4								7
SGN decoding											
ABS decoding	0110000010000100B			{6, 0, 8, 4}				1	0		
DPT decoding	11000B		0		1						6
SGN decoding	010B	{6, 0, -8, 4}									
ABS decoding				{0, 0, 0, 0}							
DPT decoding	100001B		5		0						5
SGN decoding		{0, 0, 0, 0}									
ABS decoding	0100110100000000000B			{9, 20, 0, 0}							
DPT decoding	110000B		0		1						4
SGN decoding	10B	{-9, 20, 0, 0}									
ABS decoding				{0, 0, 0, 0}							
DPT decoding	010B		0		2					2	3
SGN decoding		{0, 0, 0, 0}									
ABS decoding				{0, 0, 0, 0}							

Stage	Input compressed data	Output CoefSet	Updated parameters								
			DPT	ABS[i]	DPTs	DPTv	DPSe	ABSv	ABSe	ZR	Cnt
DPT decoding	001B		3		0					1	2
SGN decoding		{0, 0, 0, 0}									
ABS decoding	001010011100B			{1, 2, 3, 4}							
DPT decoding	1100B		0		1					0	1
SGN decoding	0000B	{1, 2, 3, 4}									
ABS decoding				{0, 0, 0, 0}							
DPT decoding	0B				2		1				0
SGN decoding		{0, 0, 0, 0}									
ABS decoding				{0, 0, 0, 0}					1		
DPT decoding						0	0				
SGN decoding		{0, 0, 0, 0}									

Figure 6.9 – Example of VLD

6.4 Inverse Quantization

VLD decoded coefficients $X_Q[b][x][y]$ for a CU are left-shifted by $Q_l[b]$ and wavelet coefficients X[b][x][y] are derived using the following equation:

$$X[b][x][y] = X_Q[b][x][y] << Q_l[b]$$

where x, y are the position of coefficients in sub-band “b” and $Q_l[b]$ is the value extracted from CU_info shown in Table 5.3.

6.5 Inverse Wavelet Transform

Figure 6.10 shows a typical two dimensional inverse wavelet transform (IWT) mechanism including both vertical (V) IWTs and horizontal (H) IWT. Lv(n) represents decomposition level n, and Lv(n) LL, Lv(n) LH, Lv(n) HL and Lv(n) HH represent sub-bands at decomposition level n.

Mappings of Lv[n] and X are as follows:

Lv(3)_LL[x][y] = X[0][x][y]
Lv(3)_LH[x][y] = X[1][x][y]
Lv(3)_HL[x][y] = X[2][x][y]
Lv(3)_HH[x][y] = X[3][x][y]

Lv(2)_LL[x][y] = Output of the previous level IWT
Lv(2)_LH[x][y] = X[4][x][y]
Lv(2)_HL[x][y] = X[5][x][y]
Lv(2)_HH[x][y] = X[6][x][y]

Lv(1)_LL[x][y] = Output of the previous level IWT
Lv(1)_LH[x][y] = X[7][x][y]
Lv(1)_HL[x][y] = X[8][x][y]
Lv(1)_HH[x][y] = X[9][x][y]

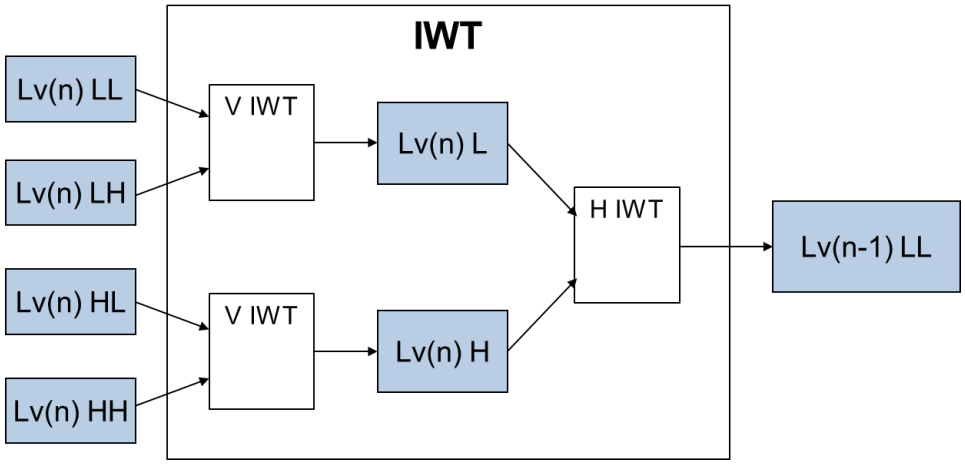


Figure 6.10 – Two Dimensional Inverse Wavelet Transform (IWT)

Figure 6.11 illustrates the vertical inverse wavelet transform mechanism where a, b, c and d represent input coefficient values while a', b' and c' represent output coefficient values. Variables x and y represent horizontal and vertical coordinates of each sub-band respectively.

a = Lv(n)_HL[x][y] , Lv(n)_LL[x][y]
b = Lv(n)_HH[x][y] , Lv(n)_HL[x][y]
c = Lv(n)_HL[x][y+1] ,Lv(n)_LL[x][y+1]
d = Lv(n)_HH[x][y+1] , Lv(n)_HL[x][y+1]

a' = Lv(n)_H[x][y] , Lv(n)_L[x][y]
b' = Lv(n)_H[x][y+1] , Lv(n)_L[x][y+1]
c' = Lv(n)_H[x][y+2] , Lv(n)_L[x][y+2]
d' = Lv(n)_H[x][y+3] , Lv(n)_L[x][y+3]

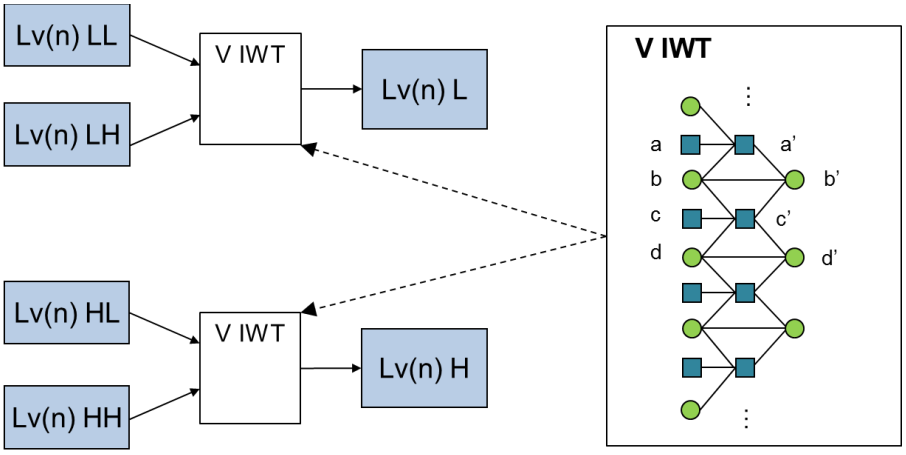


Figure 6.11 – Vertical Inverse Wavelet Transform (V IWT)

Figure 6.12 illustrates the horizontal inverse wavelet transform mechanism.

a = Lv(n)_L[x][y]
b = Lv(n)_H[x][y]
c = Lv(n)_L[x+1][y]
d = Lv(n)_H[x+1][y]

a' = Lv(n-1)_LL[x][y]
b' = Lv(n-1)_LL[x+1][y]
c' = Lv(n-1)_LL[x+2][y]
d' = Lv(n-1)_LL[x+3][y]

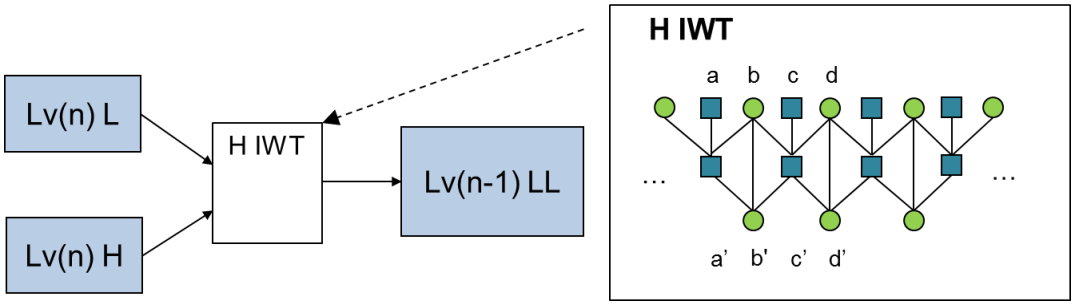


Figure 6.12 – Horizontal Inverse Wavelet Transform (H IWT)

In order to reduce the memory size and latency, the following two-step lifting-based 5-3 inverse wavelet transform is employed:

Step 1: c' = c – (b + d + 2) >> 2

Step 2: b' = b + (a' + c') >> 1

Two levels of the transform are illustrated in Figure 6.13. For example, the Step 1 calculation computes coefficient c', then the Step 2 calculation computes coefficient b'. Out-of-boundary coefficients shall be symmetrically extended as shown in Figure 6.13. The advantage of this method is to be able to directly compute the inverse wavelet transform with a small number of coefficients in memory.

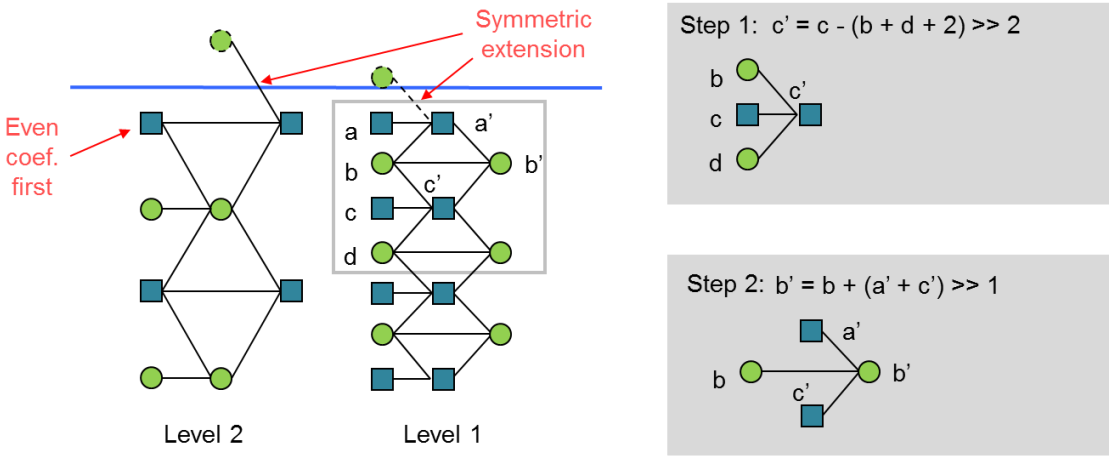


Figure 6.13 – Lifting Operations of 5-3 Inverse Wavelet Transform

6.6 Inverse Wavelet Transform and Post-Inverse Wavelet Transform

As shown in Figure 6.1, the video decoder employs an inverse wavelet transform to LB CU. A post-inverse wavelet transform is employed to generate decoded image correspondents to the TU.

Figure 6.14 shows the overview of the one-dimensional inverse wavelet transform with 3 composition levels, followed by the post-inverse wavelet transform. The lifting operations in Figure 6.14 are the same as shown in Figure 6.13 between level 2 and level 1. The same procedure is used between level 3 and level 2. The post-inverse wavelet transform generates decoded images.

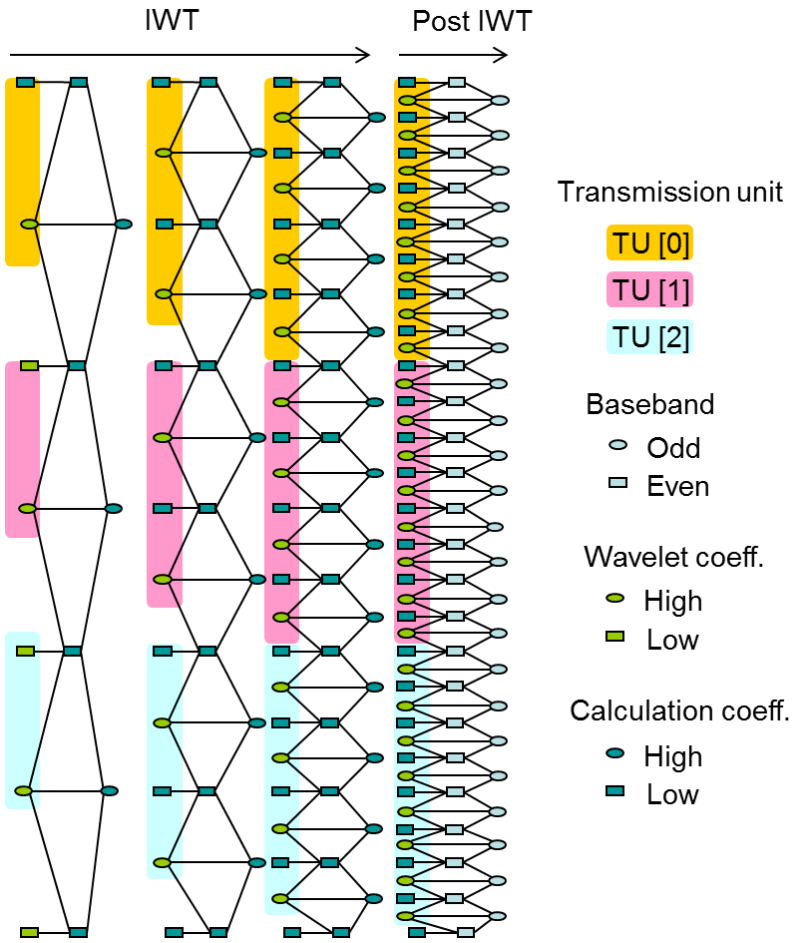


Figure 6.14 – Inverse Wavelet Transform (IWT) and Post-Inverse Wavelet Transform (Post IWT)

6.7 Post Inverse Wavelet Transform and Coding Unit

In the case of the 4:2:2 format, CU[0], CU[3], CU[4] and CU[5] are inverse wavelet transformed as shown in Figure 6.15 so that component(0) is generated. Similarly, CU[1] and CU[6] are inverse wavelet transformed and component(1) is generated. Finally, CU[2] and CU[7] are inverse wavelet transformed and component(2) is generated.

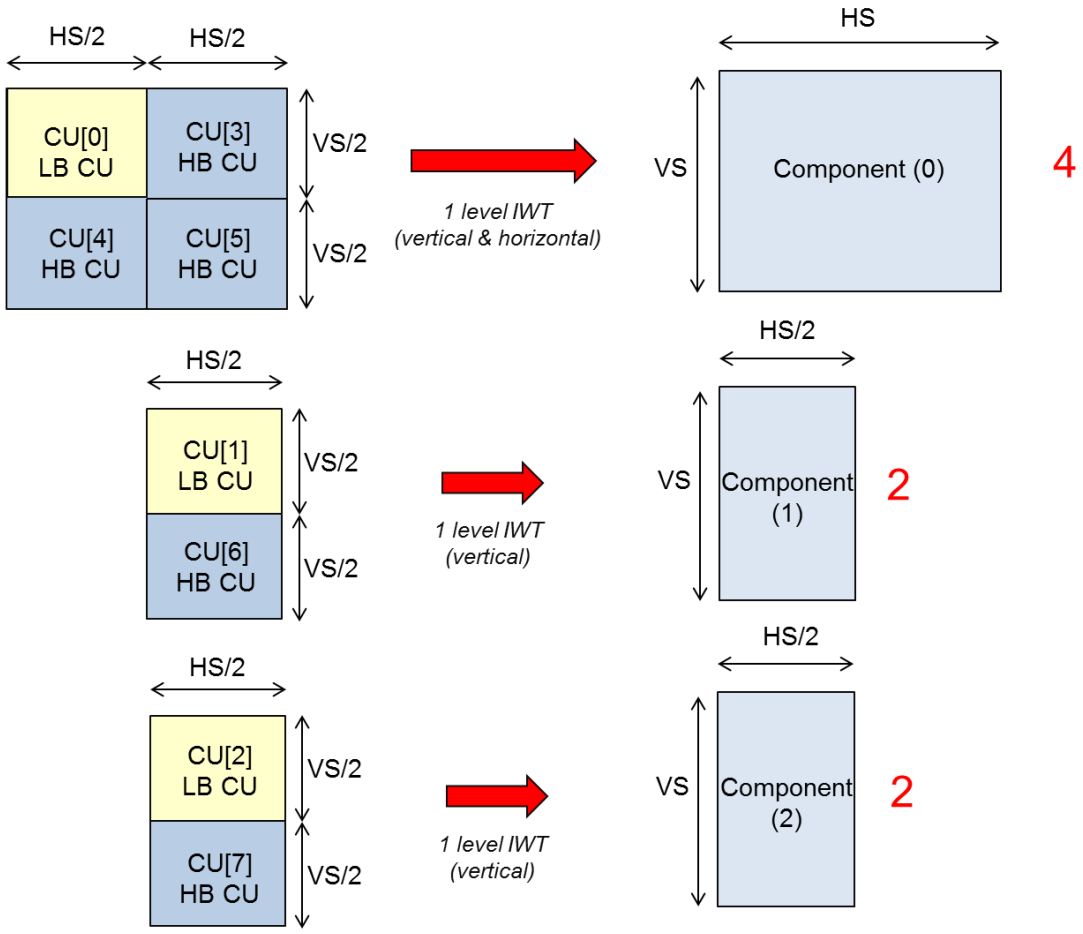


Figure 6.15 – Post-Inverse Wavelet Transform and CU - 4:2:2 Format

In the case of the 4:4:4 format, CU[0], CU[3], CU[4] and CU[5] are inverse wavelet transformed as shown in Figure 7.16 so that component(0) is generated. Similarly, CU[1] , CU[6], CU[7] and CU[8] are inverse wavelet transformed and component(1) is generated. Finally, CU[2], CU[9], CU[10] and CU[11] are inverse wavelet transformed and component(2) is generated.

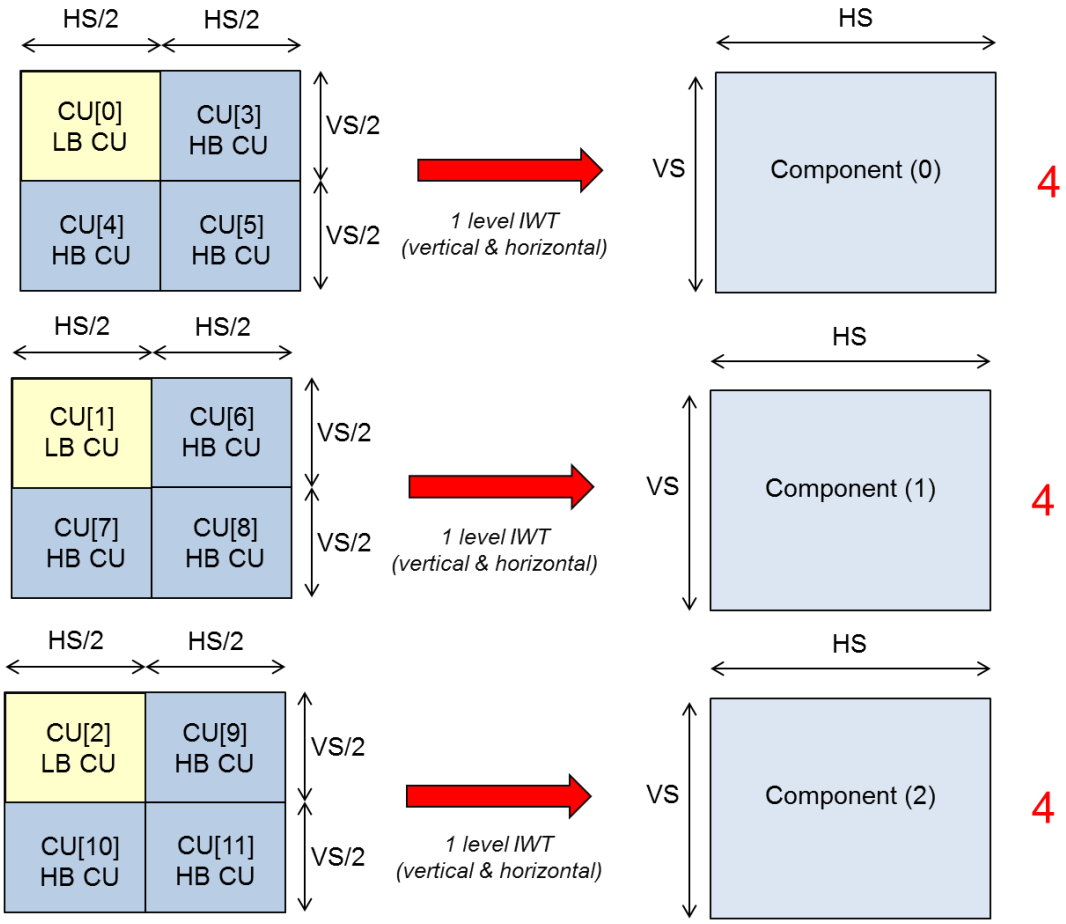


Figure 6.16 – Post Inverse Wavelet Transform and CU - 4:4:4 Format

7 IP Mapping of Codestream

7.1 Overall

In order to achieve correct delivery of the video content from transmitters to receivers, an advanced method is required to split the compressed codestreams into network packets to be sent via an RTP protocol such as multicast.

To this purpose, three main procedures are needed:

1) Preparation of the compressed codestream (low latency video codec)

2) Splitting of codestreams into network packets (such as RTP packets)

3) Adaptation of the IP protocol packet headers to the codestream

7.2 Frame (Field) Support of Media Payload

Each video frame (or field in the case of interlaced video) of consecutive data of the compressed codestream consists of a Picture_info header followed by multiple Transmission Units (TUs), as illustrated in Figure 7.1.

The codestream is divided into a number of fixed-size Media Payload blocks. Zero padding shall be added as required to the final Media Payload at the end of the frame (or field).

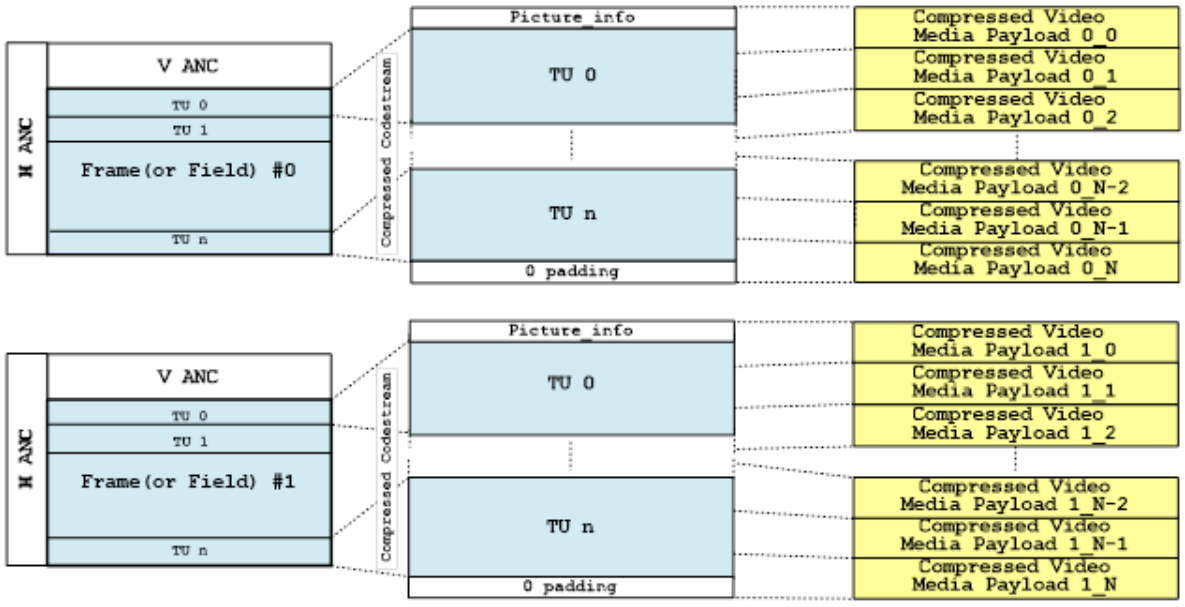


Figure 7.1 – Relation between Video Frame (or Field) and Compressed Video Media Payload

7.3 RTP Packetization of Codestream

Figure 7.2 shows one example of RTP packetization of a compressed codestream. The compressed codestream will be separated into multiple numbers of RTP packets containing the Media Payload.

Each Media Payload is equipped with UDP header, RTP header and Payload Header.

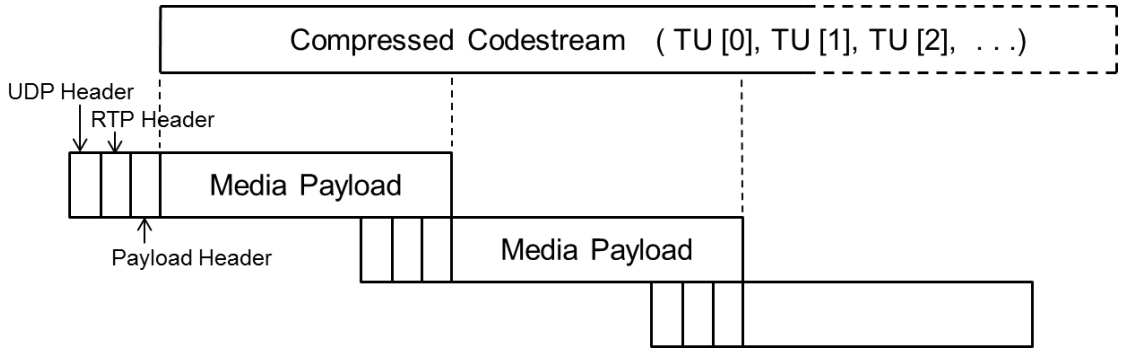


Figure 7.2 – RTP Packetization of Compressed Codestream

Annex A Encoding, Packetizing, De-Packetizing and Decoding (Informative)

The video coding scheme will be used mainly for transmission across the network. Therefore, the codestream will be packetized by a Transmission Unit (TU), transmitted across the network, de-packetized and decompressed in the decoder.

Figure A.1 is an example of this procedure. The different colors show the different TUs. The details of mapping the codestream are described in Section 7 'IP Mapping of Codestream'.

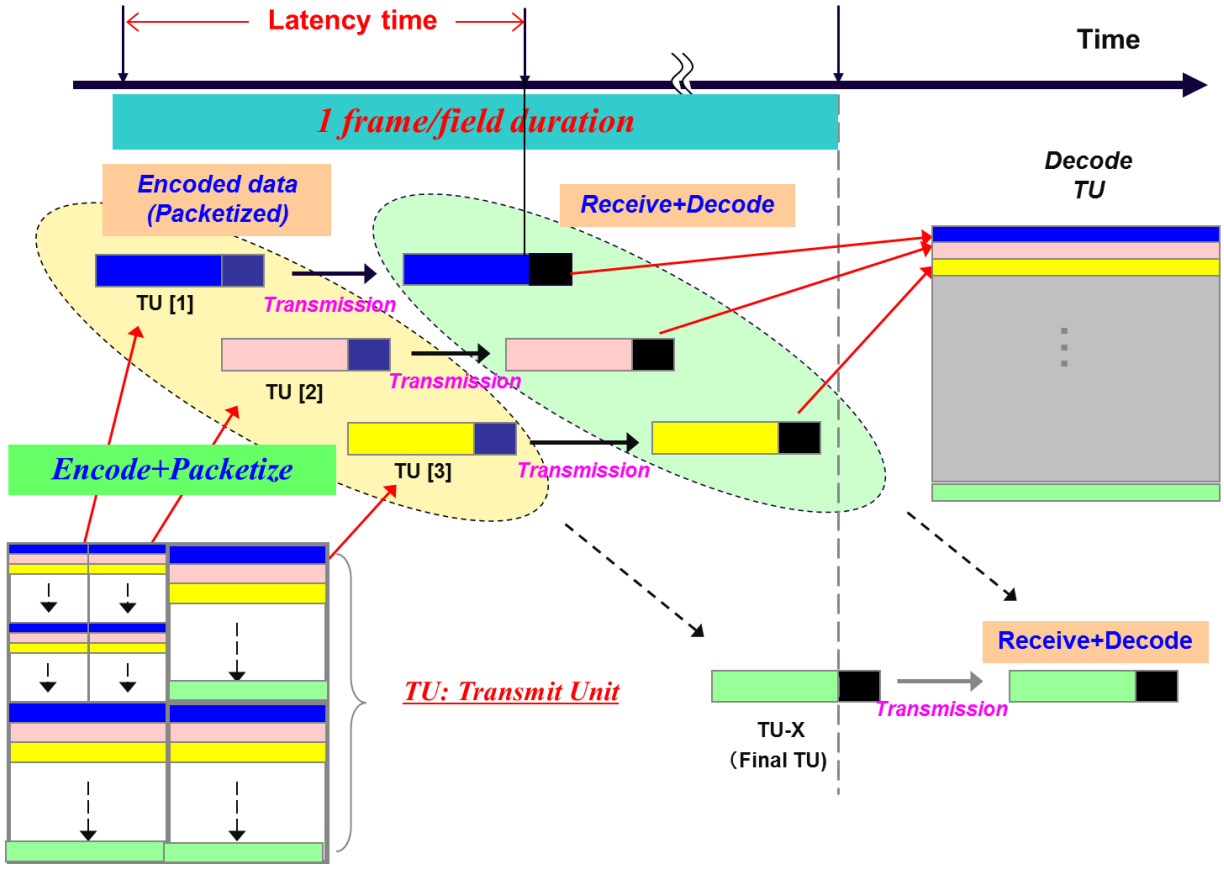


Figure A.1 – Encoding, Packetizing, De-packetizing and Decoding

Annex B Bibliography (Informative)

1. ISO/IEC 15444-1:2004, Information technology -- JPEG 2000 image coding system: Core coding system

2. David Taubman, M.Marcellin, JPEG2000 Image Compression Fundamentals, Standards and Practice, KAP, 2001

3. Stephane G. Mallat, A Theory for Multiresolution Signal Decompsosition : The Wavelet Representation", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.11, No.7, July 1989

4. Ingrid Daubechies, The Wavelet Transform, Time-Frequency Localization and Signal Analysis, IEEE Transactions on Information Theory, Vol.36, No.5, Sep. 1990

5. Marc Antonini, Michel Barlaud, Pierre Mathieu and Ingrid Daubechies, Image Coding Using Wavelet Transform, IEEE Transactions on Image Processing, Vol.1, No.2, April 1992

6. Didier Le Gall, Ali Tabatabai, Sub-band Coding of Digital Images Using Symmetric Short Kernel Filters and Arithmetic Coding Techniques", ICASSP-88, Vol.2, pp.761-764, April. 1988

7. Fang Sheng, Ali Bilgin, Philip J. Sementili, Michael W. Marcellin, Lossy And Lossless Image Compression using Reversible Integer Wavelet Transforms, ICIP, Vol.3, pp.876-880, Oct. 1998

8. A.R.Calderbank, Ingrid Daubechies, Wim Sweldens, Boon-Lock Yeo, Lossless Image Compression using Integer To Integer Wavelet Transforms, ICIP, Vol.1, pp.596-599, Oct. 1997

9. RFC RFC 5371 on RTP Payload Format for JPEG 2000 Video Streams, http://www.ietf.org/mail-archive/web/ietf-announce/current/msg05302.html

10. RFC 5372 on Payload Format for JPEG 2000 Video: Extensions for Scalability and Main Header Recovery, http://www.ietf.org/mail-archive/web/ietf-announce/current/msg05303.html