Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 19, 2016

A. Fuldseth
G. Bjontegaard
S. Midtskogen
T. Davies
M. Zanaty
Cisco

March 18, 2016

# Thor Video Codec
# draft-fuldseth-netvc-thor-02

## Abstract

This document provides a high-level description of the Thor video codec. Thor is designed to achieve high compression efficiency with moderate complexity, using the well-known hybrid video coding approach of motion-compensated prediction and transform coding.

## Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 19, 2016.

## Copyright Notice

Table of Contents

1. Introduction . . . . . 3
2. Definitions . . . . . 5
2.1. Requirements Language . . . . . 5
2.2. Terminology . . . . . 6
3. Block Structure . . . . . 6
3.1. Super Blocks and Coding Blocks . . . . . 6
3.2. Special Processing at Frame Boundaries . . . . . 6
3.3. Transform Blocks . . . . . 8
3.4. Prediction Blocks . . . . . 8
4. Intra Prediction . . . . . 8
5. Inter Prediction . . . . . 9
5.1. Multiple Reference Frames . . . . . 9
5.2. Bi-Prediction . . . . . 10
5.3. Reordered Frames . . . . . 10
5.4. Interpolated Reference Frames . . . . . 10
5.5. Sub-Pixel Interpolation . . . . . 10
5.5.1. Luma Poly-phase Filter . . . . . 10
5.5.2. Luma Special Filter Position . . . . . 12
5.5.3. Chroma Poly-phase Filter . . . . . 13
5.6. Motion Vector Coding . . . . . 13
5.6.1. Inter0 and Inter1 Modes . . . . . 13
5.6.2. Inter2 and Bi-Prediction Modes . . . . . 15
5.6.3. Motion Vector Direction . . . . . 16
6. Transforms . . . . . 16
7. Quantization . . . . . 16
7.1. Quantization matrices . . . . . 17
7.1.1. Quantization matrix selection . . . . . 17
7.1.2. Quantization matrix design . . . . . 18
8. Loop Filtering . . . . . 18
8.1. Deblocking . . . . . 18
8.1.1. Luma deblocking . . . . . 18
8.1.2. Chroma Deblocking . . . . . 19
8.2. Constrained Low Pass Filter (CLPF) . . . . . 20
9. Entropy coding . . . . . 20
9.1. Overview . . . . . 20
9.2. Low Level Syntax . . . . . 21
9.2.1. CB Level . . . . . 21
9.2.2. PB Level . . . . . 21
9.2.3. TB Level . . . . . 22
9.2.4. Super Mode . . . . . 22
9.2.5. CBP . . . . . 23
9.2.6. Transform Coefficients . . . . . 23
10. High Level Syntax . . . . . 25

10.1. Sequence Header . . . . . 25
10.2. Frame Header . . . . . 26
11. IANA Considerations . . . . . 26
12. Security Considerations . . . . . 26
13. Normative References . . . . . 27
Authors' Addresses . . . . . 27

## 1. Introduction

This document provides a high-level description of the Thor video codec. Thor is designed to achieve high compression efficiency with moderate complexity, using the well-known hybrid video coding approach of motion-compensated prediction and transform coding.

The Thor video codec is a block-based hybrid video codec similar in structure to widespread standards. The high level encoder and decoder structures are illustrated in Figure 1 and Figure 2 respectively.

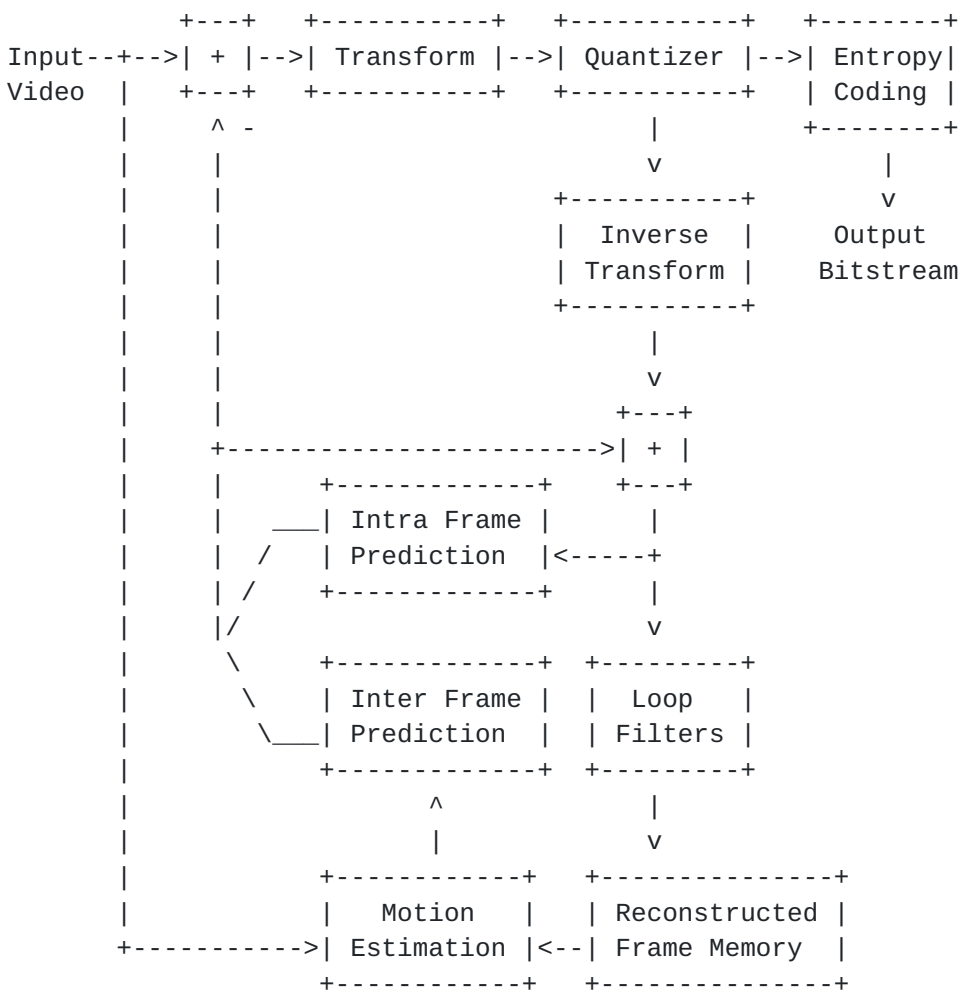


Figure 1: Encoder Structure

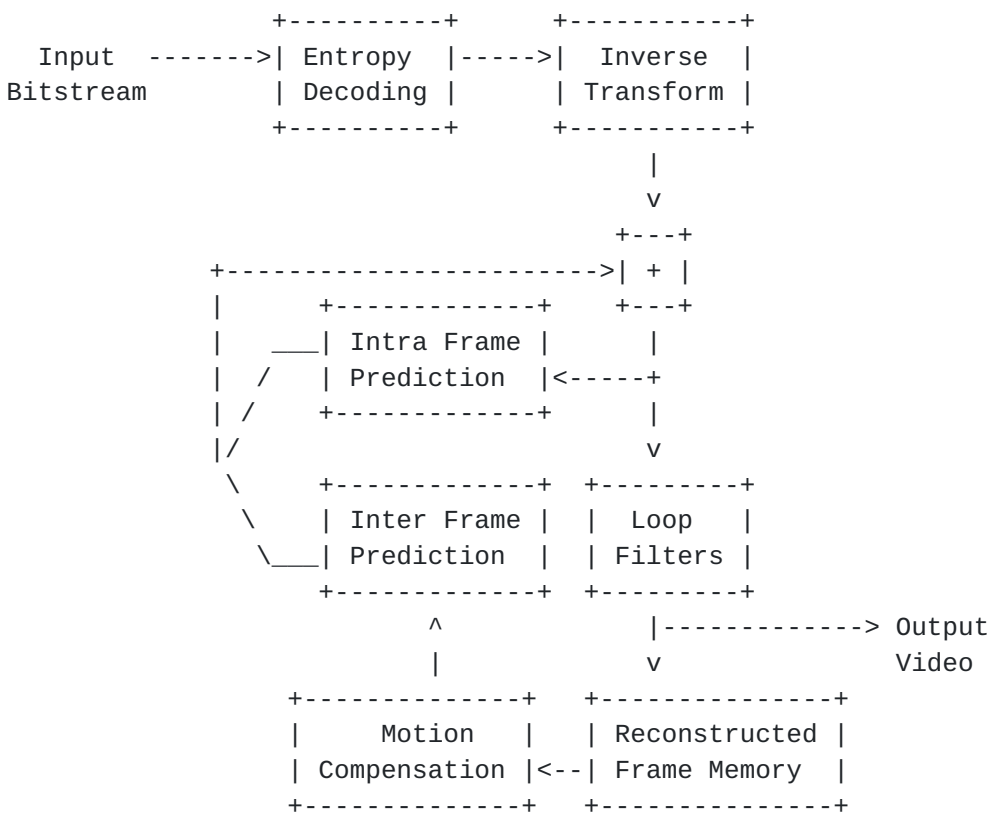


Figure 2: Decoder Structure

The remainder of this document is organized as follows. First, some requirements language and terms are defined. Block structures are described in detail, followed by intra-frame prediction techniques, inter-frame prediction techniques, transforms, quantization, loop filters, entropy coding, and finally high level syntax.

An open source reference implementation is available at github.com/cisco/thor.

## 2. Definitions

### 2.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

## 2.2. Terminology

This document frequently uses the following terms.

SB: Super Block - 64x64 or 128x128 block (luma pixels) which can be divided into CBs.

CB: Coding Block - Subdivision of a SB, down to 8x8 (luma pixels).

PB: Prediction Block - Subdivision of a CB, into 1, 2 or 4 equal blocks.

TB: Transform Block - Subdivision of a CB, into 1 or 4 equal blocks.

# 3. Block Structure

## 3.1. Super Blocks and Coding Blocks

Each frame is divided into 64x64 or 128x128 Super Blocks (SB) which are processed in raster-scan order. The SB size is signaled in the sequence header. Each SB can be divided into Coding Blocks (CB) using a quad-tree structure. The smallest allowed CB size is 8x8 luma pixels. The four CBs of a larger block are coded/signaled in the following order; upleft, downleft, upright, and downright.

The following modes are signaled at the CB level:

- Intra
- Inter0 (skip): MV index, no residual information
- Inter1 (merge): MV index, residual information
- Inter2 (uni-pred): explicit motion information, residual information
- Inter3 (ni-pred): explicit motion information, residual information

## 3.2. Special Processing at Frame Boundaries

At frame boundaries some square blocks might not be complete. For example, for 1920x1080 resolutions, the bottom row would consist of rectangular blocks of size 64x56. Rectangular blocks at frame boundaries are handled as follows. For each rectangular block, send one bit to choose between:

- A rectangular inter0 block and
- Further split.

For the bottom part of a 1920x1080 frame, this implies the following:

- For each 64x56 block, transmit one bit to signal a 64x56 inter0 block or a split into two 32x32 blocks and two 32x24 blocks.
- For each 32x24 block, transmit one bit to signal a 32x24 inter0 block or a split into two 16x16 blocks and two 16x8 blocks.
- For each 16x8 block, transmit one bit to signal a 16x8 inter0 block or a split into two 8x8 blocks.

Two examples of handling 64x56 blocks at the bottom row of a 1920x1080 frame are shown in Figure 3 and Figure 4 respectively.

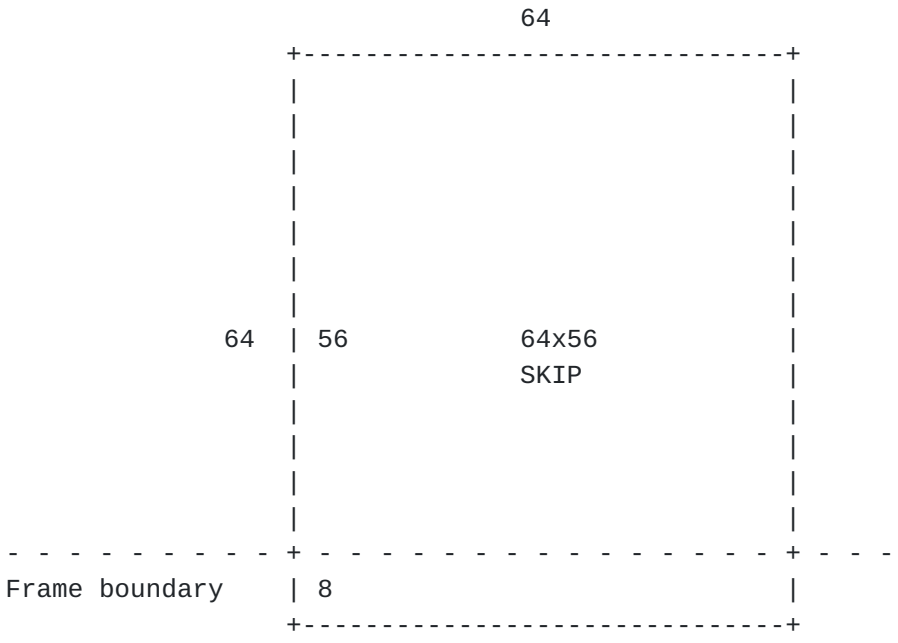


Figure 3: Super block at frame boundary

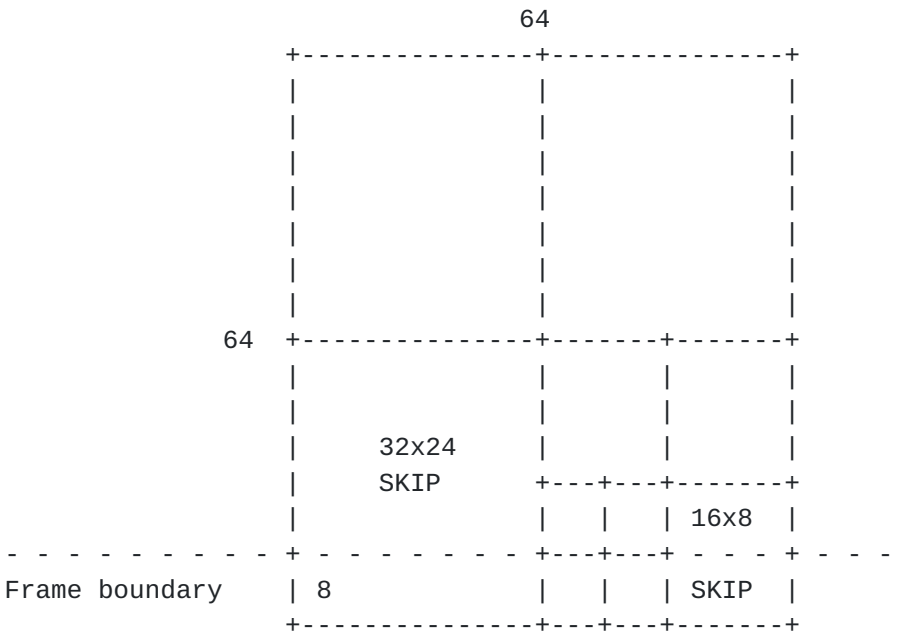


Figure 4: Coding block at frame boundary

## 3.3. Transform Blocks

A coding block (CB) can be divided into four smaller transform blocks (TBs).

## 3.4. Prediction Blocks

A coding block (CB) can also be divided into smaller prediction blocks (PBs) for the purpose of motion-compensated prediction. Horizontal, vertical and quad split are used.

# 4. Intra Prediction

8 intra prediction modes are used:

1. DC
2. Vertical (V)
3. Horizontal (H)
4. Upupright (north-northeast)
5. Upupleft (north-northwest)
6. Upleft (northwest)

7. Upleftleft (west-northwest)

8. Downleftleft (west-southwest)

The definition of DC, vertical, and horizontal modes are straightforward.

The upleft direction is exactly 45 degrees.

The upupright, upupleft, and upleftleft directions are equal to arctan(1/2) from the horizontal or vertical direction, since they are defined by going one pixel horizontally and two pixels vertically (or vice versa).

For the 5 angular intra modes (i.e. angle different from 90 degrees), the pixels of the neighbor blocks are filtered before they are used for prediction:

$$y(n) = (x(n-1) + 2*x(n) + x(n+1) + 2)/4$$

For the angular intra modes that are not 45 degrees, the prediction sometimes requires sample values at a half-pixel position. These sample values are determined by an additional filter:

$$z(n + 1/2) = (y(n) + y(n+1))/2$$

## 5. Inter Prediction

### 5.1. Multiple Reference Frames

Multiple reference frames are currently implemented as follows.

- Use a sliding-window process to keep the N most recent reconstructed frames in memory. The value of N is signaled in the sequence header.

- In the frame header, signal which of these frames shall be active for the current frame.

- For each CB, signal which of the active frames to be used for MC.

Combined with re-ordering, this allows for MPEG-1 style B frames.

A desirable future extension is to allow long-term reference frames in addition to the short-term reference frames defined by the sliding-window process.

### 5.2. Bi-Prediction

In case of bi-prediction, two reference indices and two motion vectors are signaled per CB. In the current version, PB-split is not allowed in bi-prediction mode. Sub-pixel interpolation is performed for each motion vector/reference index separately before doing an average between the two predicted blocks:

p(x,y) = (p0(x,y) + p1(x,y))/2

### 5.3. Reordered Frames

Frames may be transmitted out of order. Reference frames are selected from the sliding window buffer as normal.

### 5.4. Interpolated Reference Frames

A flag is sent in the sequence header indicating that interpolated reference frames may be used.

If a frame is using an interpolated reference frame, it will be the first reference in the reference list, and will be interpolated from the second and third reference in the list. It is indicated by a reference index of -1 and has a frame number equal to that of the current frame.

The interpolated reference is created by a deterministic process common to the encoder and decoder, and described in the separate IRFVC draft [I-D.davies-netvc-irfvc].

### 5.5. Sub-Pixel Interpolation

#### 5.5.1. Luma Poly-phase Filter

Inter prediction uses traditional block-based motion compensated prediction with quarter pixel resolution. A separable 6-tap poly-phase filter is the basis method for doing MC with sub-pixel accuracy. The luma filter coefficients are as follows:

When bi-prediction is enabled in the sequence header:

1/4 phase: [2,-10,59,17,-5,1]/64

2/4 phase: [1,-8,39,39,-8,1]/64

3/4 phase: [1,-5,17,59,-10,2]/64

When bi-prediction is disabled in the sequence header:

1/4 phase: [1,-7,55,19,-5,1]/64

2/4 phase: [1,-7,38,38,-7,1]/64

3/4 phase: [1,-5,19,55,-7,1]/64

With reference to Figure 5, a fractional sample value, e.g. i0,0 which has a phase of 1/4 in the horizontal dimension and a phase of 1/2 in the vertical dimension is calculated as follows:

a0,j = 2*A-2,i - 10*A-1,i + 59*A0,i + 17*A1,i - 5*A2,i + 1*A3,i

where j = -2,...,3

i0,0 = (1*a0,-2 - 8*a0,-1 + 39*a0,0 + 39*a0,1 - 8*a0,2 + 1*a0,3 + 2048)/4096

The minimum sub-block size is 8x8.

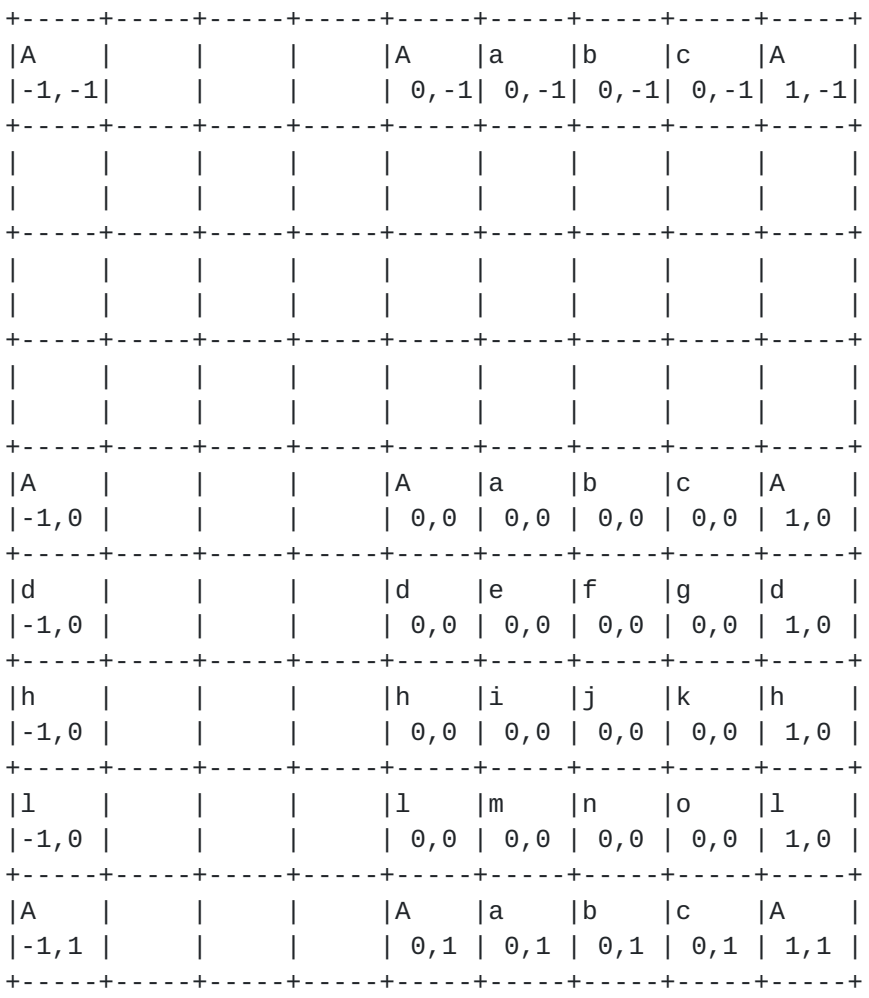


Figure 5: Sub-pixel positions

### 5.5.2. Luma Special Filter Position

For the fractional pixel position having exactly 2 quarter pixel offsets in each dimension, a non-separable filter is used to calculate the interpolated value. With reference to Figure 5, the center position j0,0 is calculated as follows:

j0,0 =

[0*A-1,-1 + 1*A0,-1 + 1*A1,-1 + 0*A2,-1 +

1*A-1,0 + 2*A0,0 + 2*A1,0 + 1*A2,0 +

1*A-1,1 + 2*A0,1 + 2*A1,1 + 1*A2,1 +

0*A-1,2 + 1*A0,2 + 1*A1,2 + 0*A2,2 + 8]/16

### 5.5.3. Chroma Poly-phase Filter

Chroma interpolation is performed with 1/8 pixel resolution using the following poly-phase filter.

1/8 phase: [-2, 58, 10, -2]/64

2/8 phase: [-4, 54, 16, -2]/64

3/8 phase: [-4, 44, 28, -4]/64

4/8 phase: [-4, 36, 36, -4]/64

5/8 phase: [-4, 28, 44, -4]/64

6/8 phase: [-2, 16, 54, -4]/64

7/8 phase: [-2, 10, 58, -2]/64

## 5.6. Motion Vector Coding

### 5.6.1. Inter0 and Inter1 Modes

Inter0 and inter1 modes imply signaling of a motion vector index to choose a motion vector from a list of candidate motion vectors with associated reference frame index. A list of motion vector candidates are derived from at most two different neighbor blocks, each having a unique motion vector/reference frame index. Signaling of the motion vector index uses 0 or 1 bit, dependent on the number of unique motion vector candidates. If the chosen neighbor block is coded in bi-prediction mode, the inter0 or inter1 block inherits both motion vectors, both reference indices and the bi-prediction property of the neighbor block.

For block sizes less than 64x64, inter0 has only one motion vector candidate, and its value is always zero.

Which neighbor blocks to use for motion vector candidates depends on the availability of the neighbor blocks (i.e. whether the neighbor blocks have already been coded, belong to the same slice and are not outside the frame boundaries). Four different availabilities, U, UR, L, and LL, are defined as illustrated in Figure 6. If the neighbor block is intra it is considered to be available but with a zero motion vector.

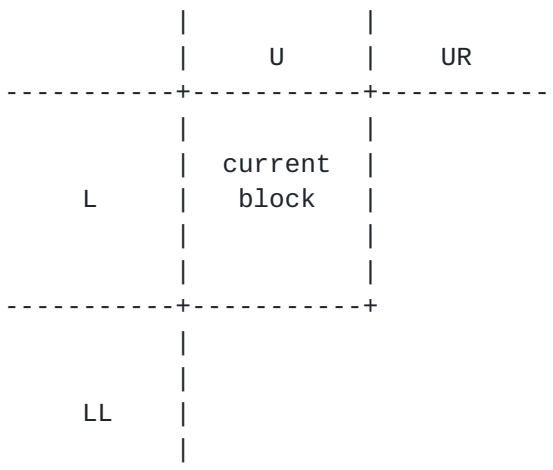


Figure 6: Availability of neighbor blocks

Based on the four availabilities defined above, each of the motion vector candidates is derived from one of the possible neighbor blocks defined in Figure 7.

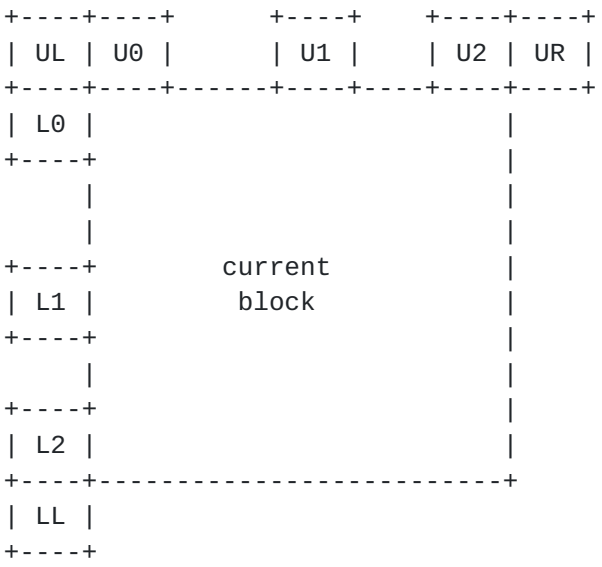


Figure 7: Motion vector candidates

The choice of motion vector candidates depends on the availability of neighbor blocks as shown in Table 1.

| U | UR | L | LL | Motion vector candidates |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | zero vector |
| 1 | 0 | 0 | 0 | U2, zero vector |
| 0 | 1 | 0 | 0 | NA |
| 1 | 1 | 0 | 0 | U2,zero vector |
| 0 | 0 | 1 | 0 | L2, zero vector |
| 1 | 0 | 1 | 0 | U2,L2 |
| 0 | 1 | 1 | 0 | NA |
| 1 | 1 | 1 | 0 | U2,L2 |
| 0 | 0 | 0 | 1 | NA |
| 1 | 0 | 0 | 1 | NA |
| 0 | 1 | 0 | 1 | NA |
| 1 | 1 | 0 | 1 | NA |
| 0 | 0 | 1 | 1 | L2, zero vector |
| 1 | 0 | 1 | 1 | U2,L2 |
| 0 | 1 | 1 | 1 | NA |
| 1 | 1 | 1 | 1 | U2,L2 |

Table 1: Motion vector candidates for different availability of neighbor blocks

### 5.6.2. Inter2 and Bi-Prediction Modes

Motion vectors are coded using motion vector prediction. The motion vector predictor is defined as the median of the motion vectors from three neighbor blocks. Definition of the motion vector predictor uses the same definition of availability and neighbors as in Figure 6 and Figure 7 respectively. The three vectors used for median filtering depends on the availability of neighbor blocks as shown in Table 2. If the neighbor block is coded in bi-prediction mode, only the first motion vector (in transmission order), MV0, is used as input to the median operator.

| U | UR | L | LL | Motion vectors for median filtering |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 3 x zero vector |
| 1 | 0 | 0 | 0 | U0,U1,U2 |
| 0 | 1 | 0 | 0 | NA |
| 1 | 1 | 0 | 0 | U0,U2,UR |
| 0 | 0 | 1 | 0 | L0,L1,L2 |
| 1 | 0 | 1 | 0 | UL,U2,L2 |
| 0 | 1 | 1 | 0 | NA |
| 1 | 1 | 1 | 0 | U0,UR,L2,L0 |
| 0 | 0 | 0 | 1 | NA |
| 1 | 0 | 0 | 1 | NA |
| 0 | 1 | 0 | 1 | NA |
| 1 | 1 | 0 | 1 | NA |
| 0 | 0 | 1 | 1 | L0,L2,LL |
| 1 | 0 | 1 | 1 | U2,L0,LL |
| 0 | 1 | 1 | 1 | NA |
| 1 | 1 | 1 | 1 | U0,UR,L0 |

Table 2: Neighbor blocks used to define motion vector predictor through median filtering

### 5.6.3. Motion Vector Direction

Motion vectors referring to reference frames later in time than the current frame are stored with their sign reversed, and these reversed values are used for coding and motion vector prediction.

## 6. Transforms

Transforms are applied at the TB or CB level, implying that transform sizes range from 4x4 to 128x128. The transforms form an embedded structure meaning the transform matrix elements of the smaller transforms can be extracted from the larger transforms.

## 7. Quantization

For the 32x32, 64x64 and 128x128 transform sizes, only the 16x16 low frequency coefficients are quantized and transmitted.

The 64x64 inverse transform is defined as a 32x32 transform followed by duplicating each output sample into a 2x2 block. The 128x128 inverse transform is defined as a 32x32 transform followed by duplicating each output sample into a 4x4 block.

### 7.1. Quantization matrices

A flag is transmitted in the sequence header to indicate whether quantization matrices are used. If this flag is true, a 6 bit value qmtx_offset is transmitted in the sequence header to indicate matrix strength.

If used, then in dequantization a separate scaling factor is applied to each coefficient, so that the dequantized value of a coefficient ci at position i is:

```
(ci * d(q) * IW(i,c,s,t,q) + 2^(k + 5)) >> (k + 6)
```

Figure 8: Equation 1

where IW is the scale factor for coefficient position i with size s, frame type (inter/inter) t, component (Y, Cb or Cr) c and quantizer q; and k=k(s,q) is the dequantization shift. IW has scale 64, that is, a weight value of 64 is no different to unweighted dequantization.

#### 7.1.1. Quantization matrix selection

The current luma qp value qpY and the offset value qmtx_offset determine a quantisation matrix set by the formula:

```
qmlevel = max(0,min(11,((qpY + qmtx_offset) * 12) / 44))
```

Figure 9: Equation 2

This selects one of the 12 different sets of default quantization matrix, with increasing qmlevel indicating increasing flatness.

For a given value of qmlevel, different weighting matrices are provided for all combinations of transform block size, type (intra/inter), and component (Y, Cb, Cr). Matrices at low qmlevel are flat (constant value 64). Matrices for inter frames have unity DC gain (i.e. value 64 at position 0), whereas those for intra frames are designed such that the inverse weighting matrix has unity energy gain (i.e. normalized sum-squared of the scaling factors is 1).

### 7.1.2. Quantization matrix design

Further details on the quantization matrix and implementation can be found in the separate QMTX draft [I-D.davies-netvc-qmtx].

## 8. Loop Filtering

### 8.1. Deblocking

#### 8.1.1. Luma deblocking

Luma deblocking is performed on an 8x8 grid as follows:

1. For each vertical edge between two 8x8 blocks, calculate the following for each of line 2 and line 5 respectively:

   d = abs(a-b) + abs(c-d),

   where a and b, are on the left hand side of the block edge and c and d are on the right hand side of the block edge:

   a b | c d

2. For each line crossing the vertical edge, perform deblocking if and only if all of the following conditions are true:

   * d2+d5 < beta(QP)

   * The edge is also a transform block edge

   * abs(mvx(left)) > 2, or abs(mvx(right)) > 2, or

     abs(mvy(left)) > 2, or abs(mvy(right)) > 2, or

     One of the transform blocks on each side of the edge has non-zero coefficients, or

     One of the transform blocks on each side of the edge is coded using intra mode.

3. If deblocking is performed, calculate a delta value as follows:

   delta = clip((18*(c-b) - 6*(d-a) + 16)/32,tc,-tc),

   where tc is a QP-dependent value.

4. Next, modify two pixels on each side of the block edge as follows:

a' = a + delta/2

b' = b + delta

c' = c + delta

d' = d + delta/2

5. The same procedure is followed for horizontal block edges.

The relative positions of the samples, a, b, c, d and the motion vectors, MV, are illustrated in Figure 10.

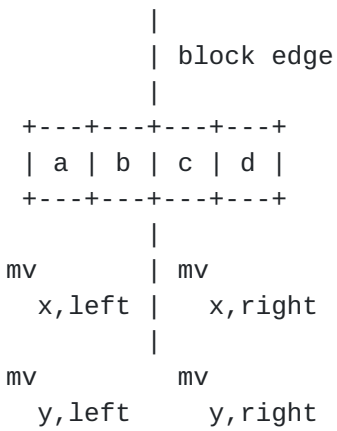


Figure 10: Deblocking filter pixel positions

### 8.1.2. Chroma Deblocking

Chroma deblocking is performed on a 4x4 grid as follows:

1. Delocking of the edge between two 4x4 blocks is performed if and only if:

   * The pixels on either side of the block edge belongs to an intra block.

   * The block edge is also an edge between two transform blocks.

2. If deblocking is performed, calculate a delta value as follows:

   delta = clip((4*(c-b) + (d-a) + 4)/8,tc,-tc),

   where tc is a QP-dependent value.

3. Next, modify one pixel on each side of the block edge as follows:

b' = b + delta

c' = c + delta

## 8.2. Constrained Low Pass Filter (CLPF)

A low-pass filter is applied after the deblocking filter if signaled in the sequence header. It can still be switched off for individual frames in the frame header. Also signaled in the frame header is whether to apply the filter for all qualified 128x128 blocks or to transmit a flag for each such block. A super block does not qualify if it only contains Inter0 (skip) coding block and no signal is transmitted for these blocks.

The filter is described in the separate CLPF draft [I-D.midtskogen-netvc-clpf].

# 9. Entropy coding

## 9.1. Overview

The following information is signaled at the sequence level:

- Sequence header

The following information is signaled at the frame level:

- Frame header

The following information is signaled at the CB level:

- Super-mode (mode, split, reference index for uni-prediction)
- Intra prediction mode
- PB-split (none, hor, ver, quad)
- TB-split (none or quad)
- Reference frame indices for bi-prediction
- Motion vector candidate index
- Transform coefficients if TB-split=0

The following information is signaled at the TB level:

- CBP (8 combinations of CBPY, CBPU, and CBPV)
- Transform coefficients

The following information is signaled at the PB level:

- Motion vector differences

## 9.2. Low Level Syntax

### 9.2.1. CB Level

```
    super-mode          (inter0/split/inter1/inter2-ref0/intra/inter2-ref1/
inter2-ref2/inter2-ref3,..)

    if (mode == inter0 || mode == inter1)

      mv_idx                    (one of up to 2 motion vector candidates)

    else if (mode == INTRA)

      intra_mode                (one of up to 8 intra modes)

      tb_split                  (NONE or QUAD, coded jointly with CBP for
tb_split=NONE)

    else if (mode == INTER)

      pb_split          (NONE,VER,HOR,QUAD)

      tb_split_and_cbp  (NONE or QUAD and CBP)

    else if (mode == BIPRED)

      mvd_x0, mvd_y0    (motion vector difference for first vector)

      mvd_x1, mvd_y1    (motion vector difference for second vector)

      ref_idx0, ref_idx1        (two reference indices)
```

### 9.2.2. PB Level

```
       if (mode == INTER2 || mode == BIPRED)

         mvd_x, mvd_y              (motion vector differences)
```

### 9.2.3. TB Level

```
if (mode != INTER0 and tb_split == 1)
  cbp                      (8 possibilities for CBPY/CBPU/CBPV)
if (mode != INTER0)
  transform coefficients
```

### 9.2.4. Super Mode

For each block of size NxN (64>=N>8), the following mutually exclusive events are jointly encoded using a single VLC code as follows (example using 4 reference frames):

If there is no interpolated reference frame:

```
INTER0      1
SPLIT       01
INTER1      001
INTER2-REF0 0001
BIPRED      00001
INTRA       000001
INTER2-REF1     0000001
INTER2-REF2     00000001
INTER2-REF3     00000000
```

If there is an interpolated reference frame:

```
INTER0      1
SPLIT       01
INTER1      001
BIPRED      0001
INTRA       00001
INTER2-REF1     000001
INTER2-REF2     0000001
INTER2-REF3     00000001
INTER2-REF0 00000000
```

If less than 4 reference frames is used, a shorter VLC table is used. If bi-pred is not possible, or split is not possible, they are omitted from the table and shorter codes are used for subsequent elements.

Additionally, depending on information from the blocks to the left and above (meta data and CBP), a different sorting of the events can be used, e.g.:

```
SPLIT       1
INTER1      01
INTER2-REF0 001
INTER0      0001
INTRA       00001
INTER2-REF1 000001
INTER2-REF2 0000001
INTER2-REF3 00000001
BIPRED      00000000
```

### 9.2.5. CBP

Calculate code as follows:

```
if (tb-split == 0)

  N = 4*CBPV + 2*CBPU + CBPY

else

  N = 8
```

Map the value of N to code through a table lookup:

```
code = table[N]
```

where the purpose of the table lookup is the sort the different values of code according to decreasing probability (typically CBPY=1, CBPU=0, CBPV=0 having the highest probability).

Use a different table depending on the values of CBPY in neighbor blocks (left and above).

Encode the value of code using a systematic VLC code.

### 9.2.6. Transform Coefficients

Transform coefficient coding uses a traditional zig-zag scan pattern to convert a 2D array of quantized transform coefficients, coeff, to a 1D array of samples. VLC coding of quantized transform coefficients starts from the low frequency end of the 1D array using two different modes; level-mode and run-mode, starting in level-mode:

- Level-mode
  - Encode each coefficient, coeff, separately
  - Each coefficient is encoded by:

+ The absolute value, level=abs(coeff), using a VLC code and

+ If level > 0, the sign bit (sign=0 or sign=1 for coeff>0 and coeff<0 respectively).

* If coefficient N is zero, switch to run-mode, starting from coefficient N+1.

o Run-mode

* For each non-zero coefficient, encode the combined event of:

1. Length of the zero-run, i.e. the number of zeros since the last non-zero coefficient.

2. Whether or not level=abs(coeff) is greater than 1.

3. End of block (EOB) indicating that there are no more non-zero coefficients.

* Additionally, if level = 1, code the sign bit.

* Additionally, if level > 1 define code = 2*(level-2)+sign,

* If the absolute value of coefficient N is larger than 1, switch to level-mode, starting from coefficient N+1.

Example

Figure 11 illustrates an example where 16 quantized transform coefficients are encoded.

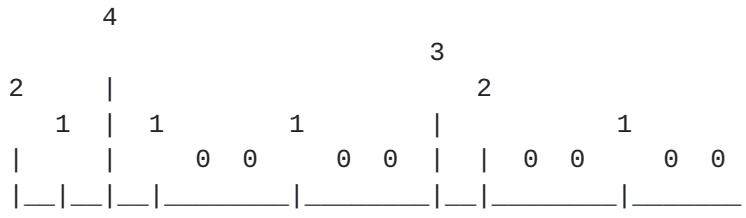


Figure 11: Coefficients to encode

Table 3 shows the mode, VLC number and symbols to be coded for each coefficient.

| Index | abs(coeff) | Mode | Encoded symbols |
|---|---|---|---|
| 0 | 2 | level-mode | level=2,sign |
| 1 | 1 | level-mode | level=1,sign |
| 2 | 4 | level-mode | level=4,sign |
| 3 | 1 | level-mode | level=1,sign |
| 4 | 0 | level-mode | level=0 |
| 5 | 0 | run-mode | |
| 6 | 1 | run-mode | (run=1,level=1) |
| 7 | 0 | run-mode | |
| 8 | 0 | run-mode | |
| 9 | 3 | run-mode | (run=1,level>1), 2*(3-2)+sign |
| 10 | 2 | level-mode | level=2, sign |
| 11 | 0 | level-mode | level=0 |
| 12 | 0 | run-mode | |
| 13 | 1 | run-mode | (run=1,level=1) |
| 14 | 0 | run-mode | EOB |
| 15 | 0 | run-mode | |

Table 3: Transform coefficient encoding for the example above.

## 10. High Level Syntax

High level syntax is currently very simple and rudimentary as the primary focus so far has been on compression performance. It is expected to evolve as functionality is added.

### 10.1. Sequence Header

- Width - 16 bits
- Height - 16 bits
- Enable/disable PB-split - 1 bit
- SB size - 3 bits
- Enable/disable TB-split - 1 bit
- Number of active reference frames (may go into frame header) - 2 bits (max 4)
- Enable/disable interpolated reference frames - 1 bit
- Enable/disable delta qp - 1 bit

- Enable/disable deblocking - 1 bit
- Constrained low-pass filter (CLPF) enable/disable - 1 bit
- Enable/disable block context coding - 1 bit
- Enable/disable bi-prediction - 1 bit
- Enable/disable quantization matrices - 1 bit
- If quantization matrices enabled: quantization matrix offset - 6 bit

### 10.2. Frame Header

- Frame type - 1 bit
- QP - 8 bits
- Identification of active reference frames - num_ref*4 bits
- Number of intra modes - 4 bits
- Number of active reference frames - 2 bits
- Active reference frames - number of active reference frames * 6 bits
- Frame number - 16 bits
- If CLPF is enabled in the sequence header: Constrained low-pass filter (CLPF) strength - 2 bits (00 = off, 01 = strength 1, 10 = strength 2, 11 = strength 4)
- IF CLPF is enabled in the sequence header: Enable/disable CLPF signal for each qualified filter block

## 11. IANA Considerations

This document has no IANA considerations yet. TBD

## 12. Security Considerations

This document has no security considerations yet. TBD

## 13. Normative References

[I-D.davies-netvc-irfvc]
Davies, T., "Interpolated reference frames for video coding", draft-davies-netvc-irfvc-00 (work in progress), October 2015.

[I-D.davies-netvc-qmtx]
Davies, T., "Quantisation matrices for Thor video coding", draft-davies-netvc-qmtx-00 (work in progress), March 2016.

[I-D.midtskogen-netvc-clpf]
Midtskogen, S., Fuldseth, A., and M. Zanaty, "Constrained Low Pass Filter", draft-midtskogen-netvc-clpf-01 (work in progress), March 2016.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

## Authors' Addresses

Arild Fuldseth
Cisco
Lysaker
Norway

Email: arilfuld@cisco.com

Gisle Bjontegaard
Cisco
Lysaker
Norway

Email: gbjonteg@cisco.com

Steinar Midtskogen
Cisco
Lysaker
Norway

Email: stemidts@cisco.com

Thomas Davies
Cisco
London
UK

Email: thdavies@cisco.com

Mo Zanaty
Cisco
RTP,NC
USA

Email: mzanaty@cisco.com