The VP8 Video Codec

Multimedia Codecs
SS 2011

Thomas Maier <tm061@hdm-stuttgart.de>
Dominik Hübner <dh052@hdm-stuttgart.de>
Sven Pfleiderer <sp055@hdm-stuttgart.de>

Problem Definition

- No standardized codec for web video
- Currently used:
  - H264: patent licensing royalties needed
  - Theora: royalty free, outdated technology
- Heterogenous client hardware
- Bandwidth constraints

History

- On2 Technologies developed VP8
- Announced September 2008 to replace VP7
- Acquisition of On2 by Google early 2010
- Open letter from the Free Software Foundation to Google demanding open sourcing of VP8

History

- Release of VP8 under a BSD-like license
- Launch of the WebM and WebP projects
- Faster VP8 decoder written by x264 developers in July 2010
- RFC draft of bitstream guide submitted to IETF (not as a standard) in January 2011

Patent Situation

- Patent situation unclear
- VP8 affects patents of h264
  - Possible prior art by Nokia in ~2000
- MPEG LA announced a call for patents against VP8

The WebM-Project

- Founded by Google in May 2010
- Royalty free media file format
- Open-sourced under a BSD-style license
- Optimized for the web
  - Low computational complexity
  - Simple container format
  - Click and encode

The WebM-Project

- Container is a subset of Matroska
  - VP8 for video
  - Vorbis for audio
- *.webm extension
- Internet media types
  - video/webm
  - audio/webm

Web Video

- HTML5 video tag < video >
  - Replacement for Flash and Silverlight
  - Customizable video controls with CSS
  - Scriptable with standardized JavaScript APIs
- No standardized video format
  - h264
  - VP8
  - Theora

Application

Browser	Theora	H.264	VP8 WebM
Internet Explorer	Manual Install	9.0	Manual Install
Mozilla Firefox	3.5	No	4.0
Google Chrome	3.0	Yes (removed in future)	6.0
Safari	Manual Install	3.1	Manual Install
Opera	10.50	No	10.60
Konquerer	4.4	Depends on QT	Yes
Epiphany	2.28	Depends on GStreamer	Depends on GStreamer

http://en.wikipedia.org/wiki/HTML5_video#Table

Application

- Youtube successively converts to VP8
- Flash support announced
  - Important for DRM
- Skype 5.0
- Nvidia announced 3D support

Application

Tools and libraries

- GStreamer
- FFmpeg
- libvpx
- ffvp8

Application

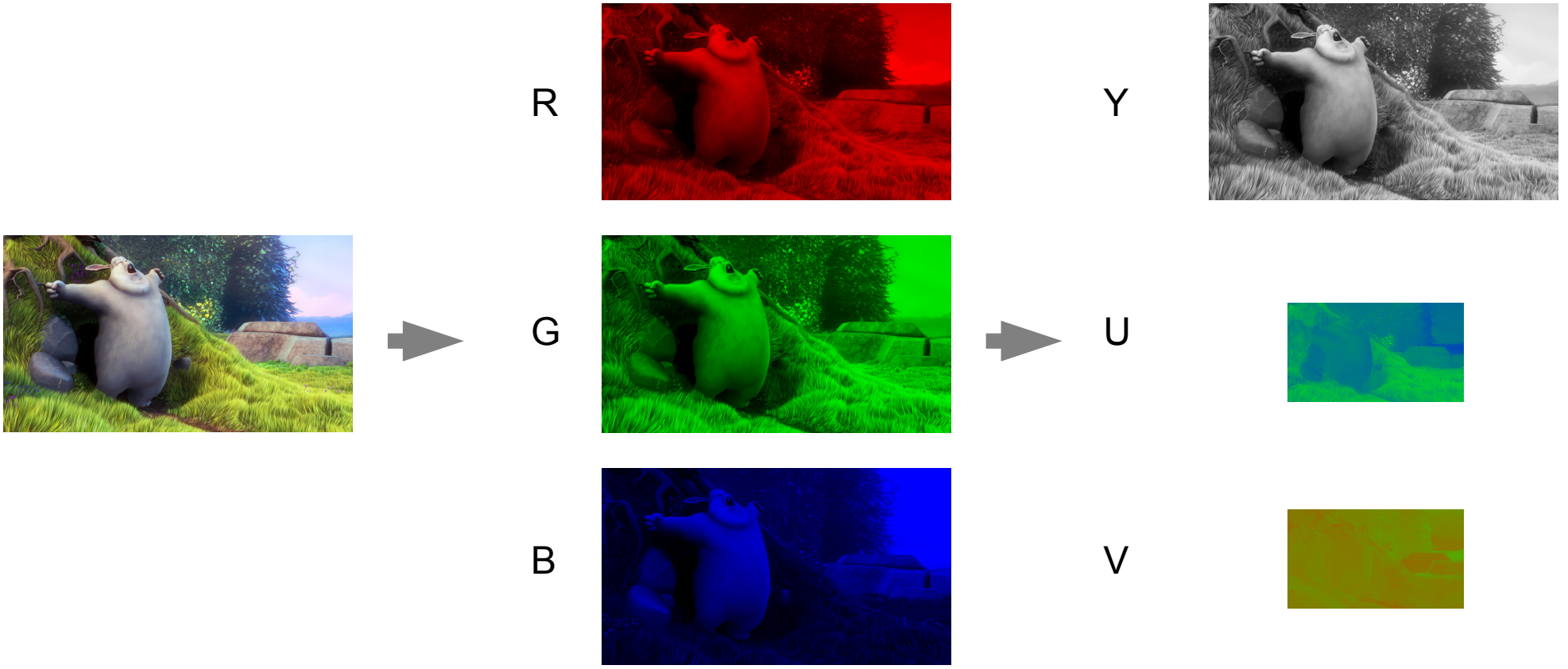
Hardware support

- AMD
- ARM
- Broadcom
- MIPS
- Nvidia
- Texas Instruments
- Open IP for hardware decoders

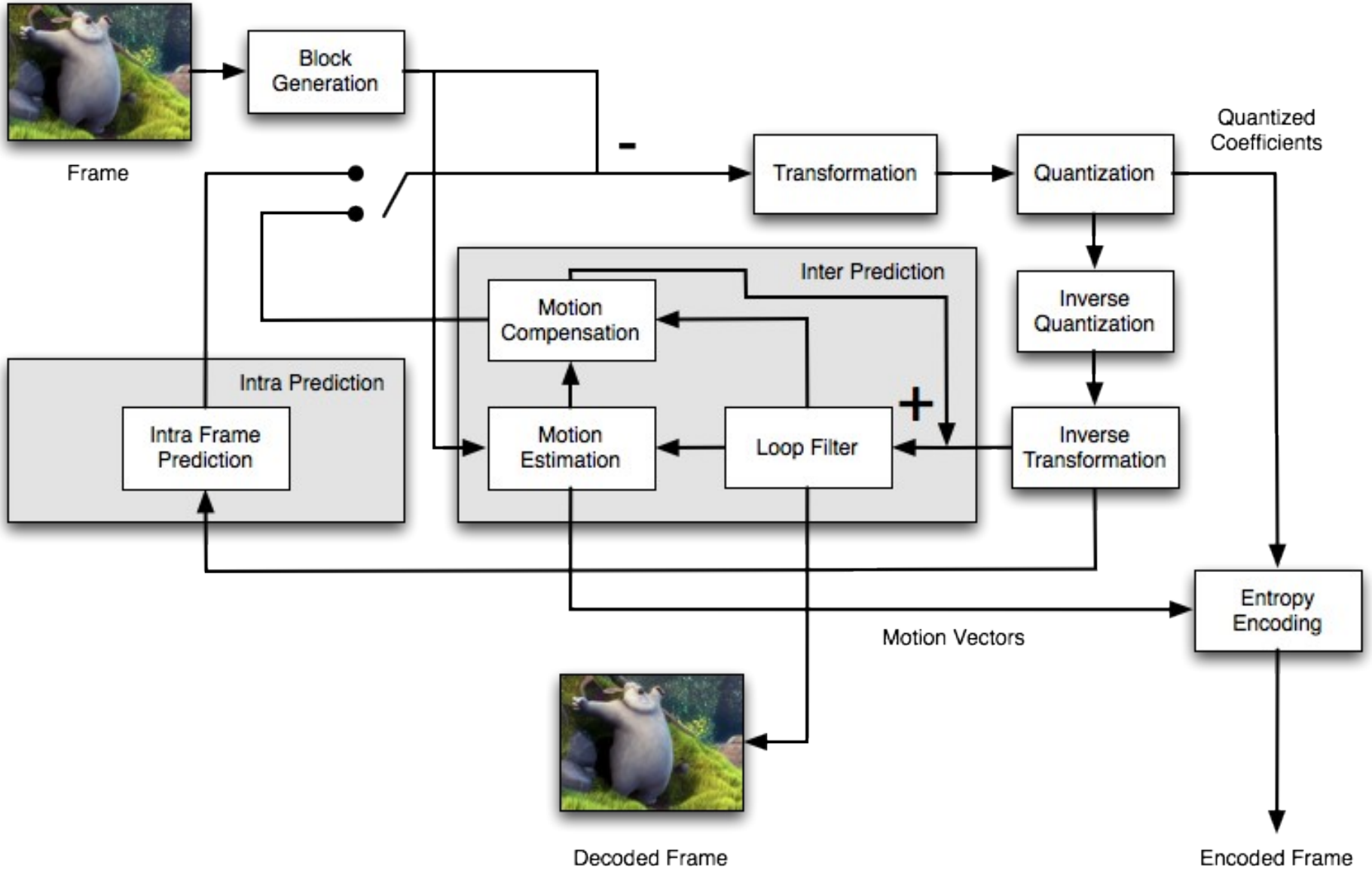
VP8 in-depth

Color Space

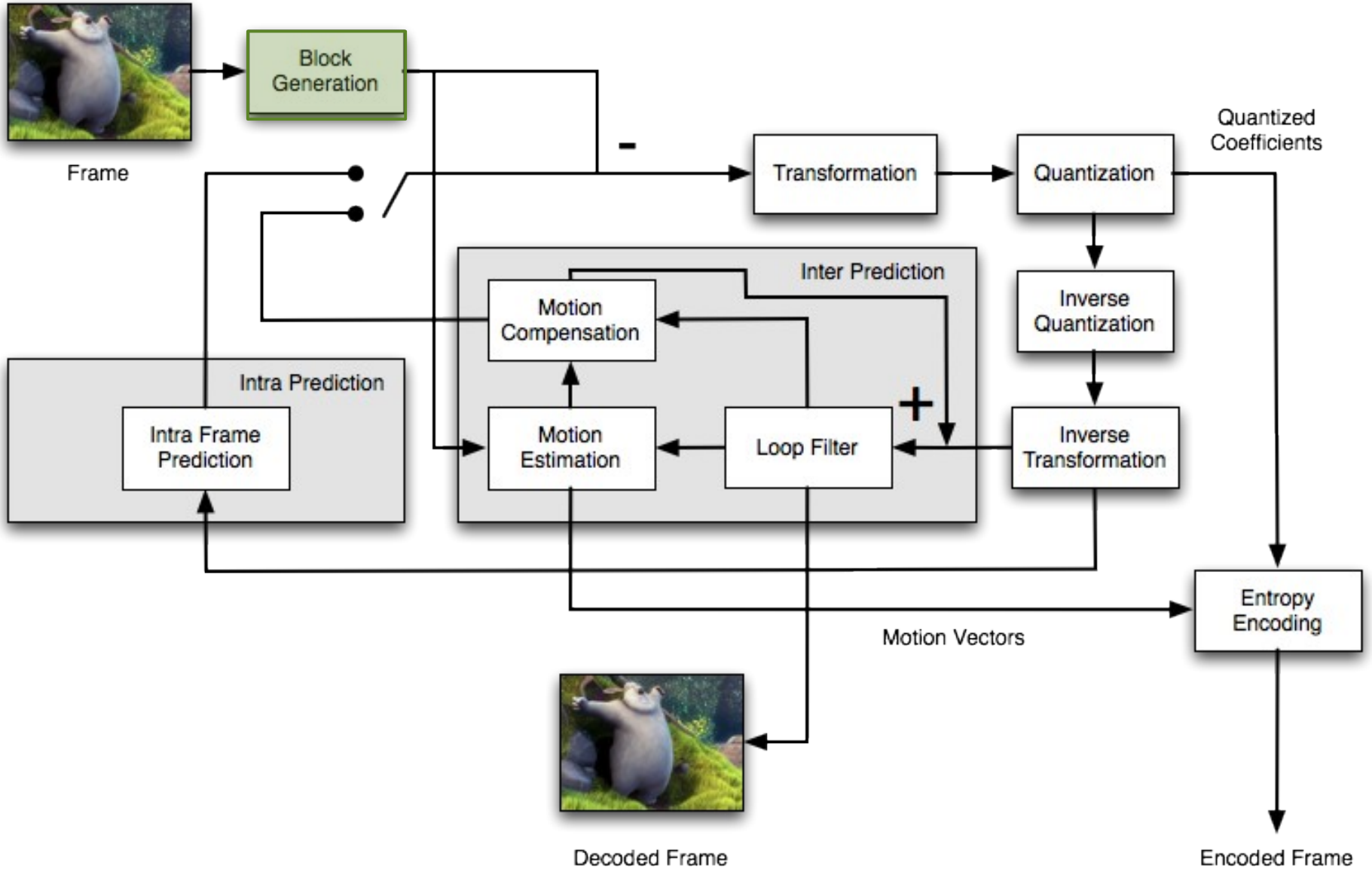
- YUV 4:2:0 sub-sampling



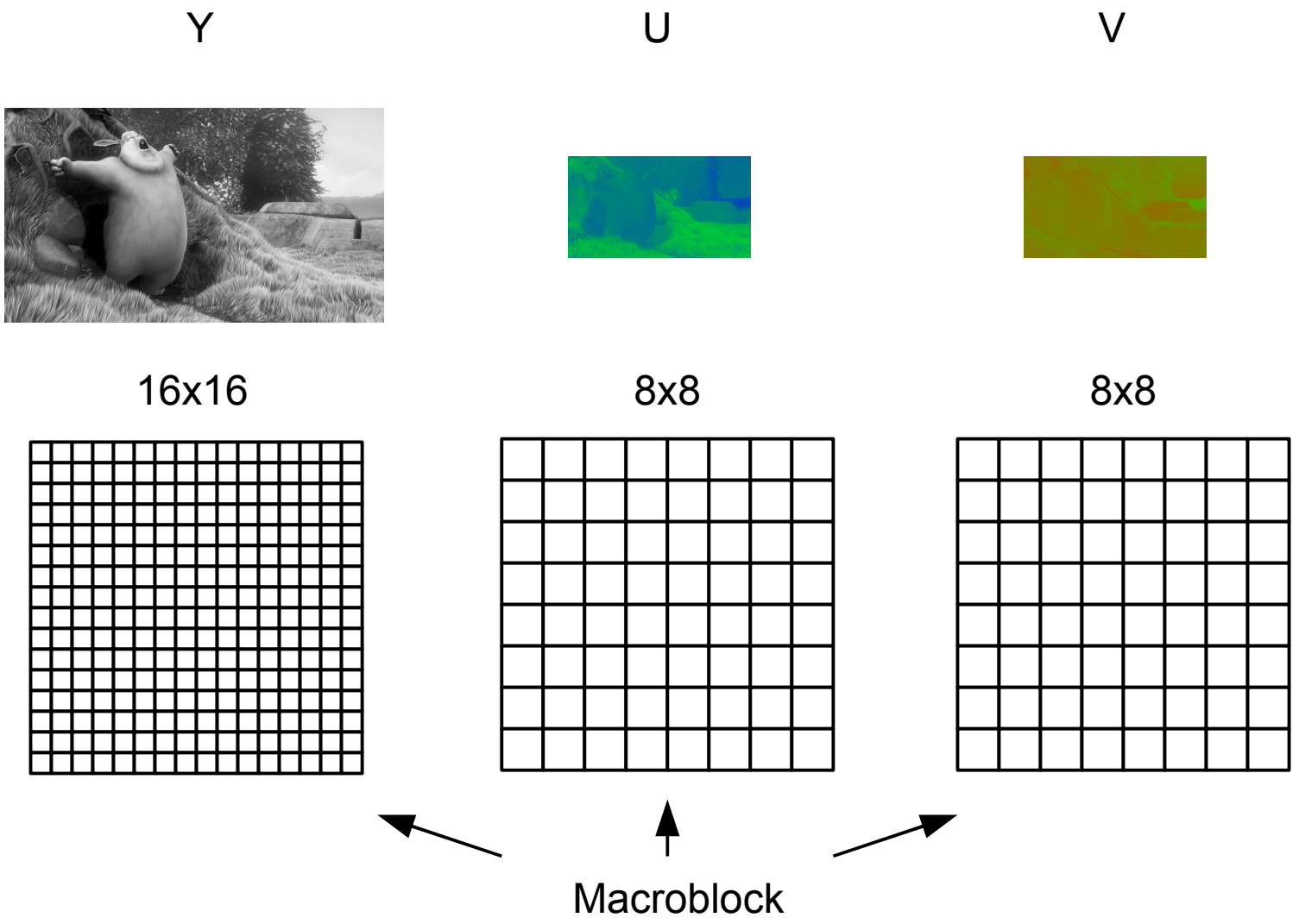
VP8 Encoding Overview



Block Generation

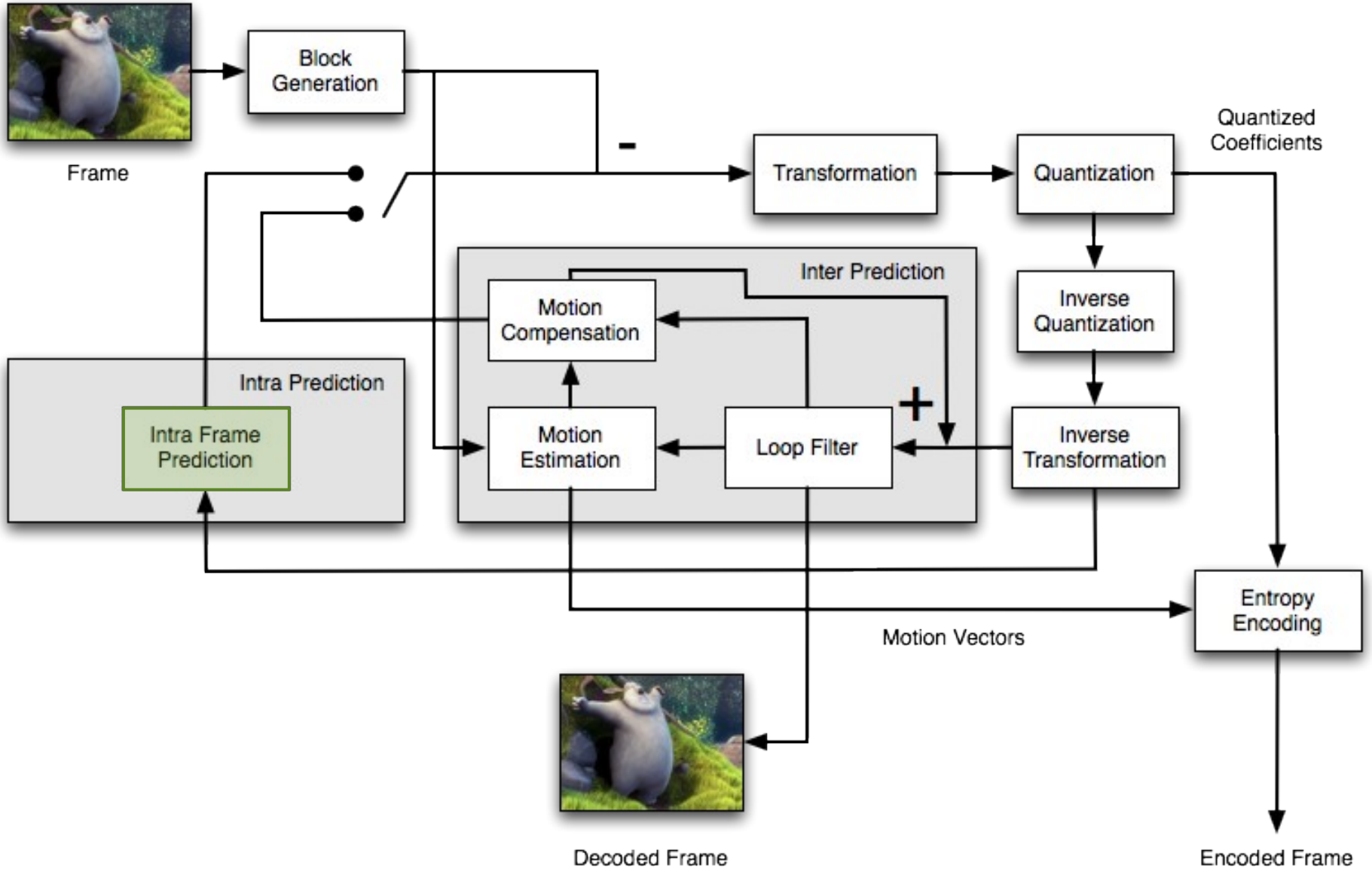


Block Generation



Prediction

Intra Frame Prediction



Intra Frame Prediction

- Exploits spacial coherence of frames
- Uses already coded blocks within current frame
- Applies to macroblocks in an interframe as well as to macroblocks in a key frame
- 16x16 luma and 8x8 chroma components are predicted independently

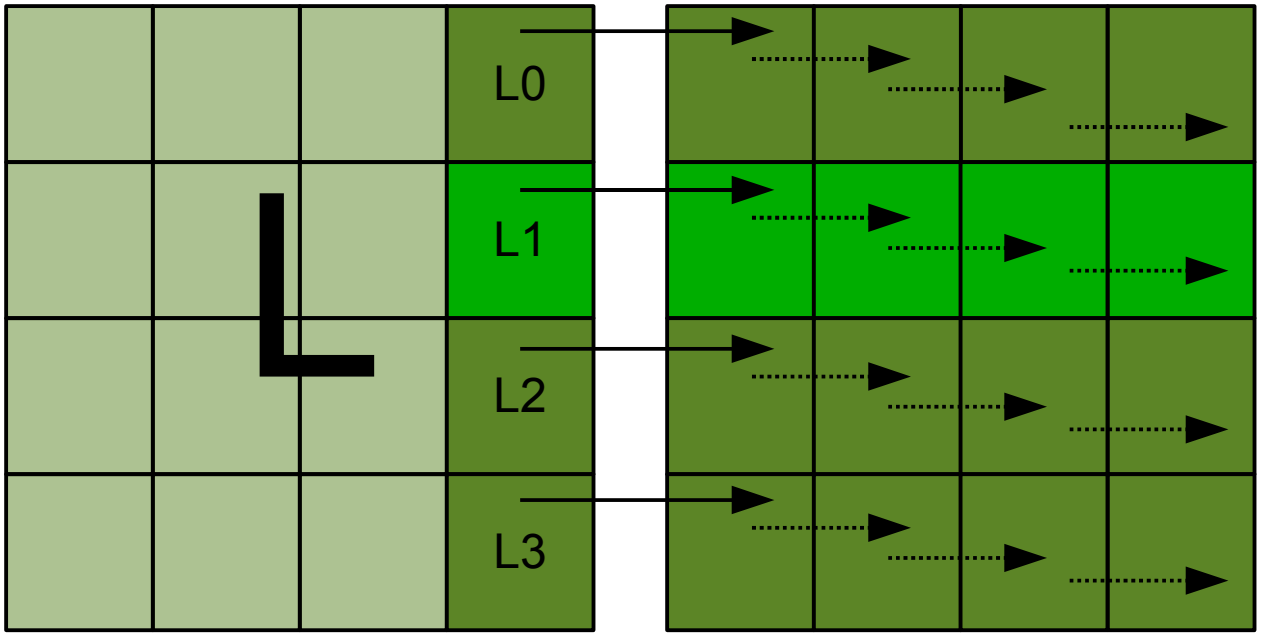
Chroma Prediction Modes

- H_PRED
- V_PRED
- DC_PRED
- TM_PRED

H_PRED

- Horizontal Prediction
- Fills each pixel column with a copy of left neighboring column (L)
- If current macroblock is on the left column, a default value of 129 is assigned

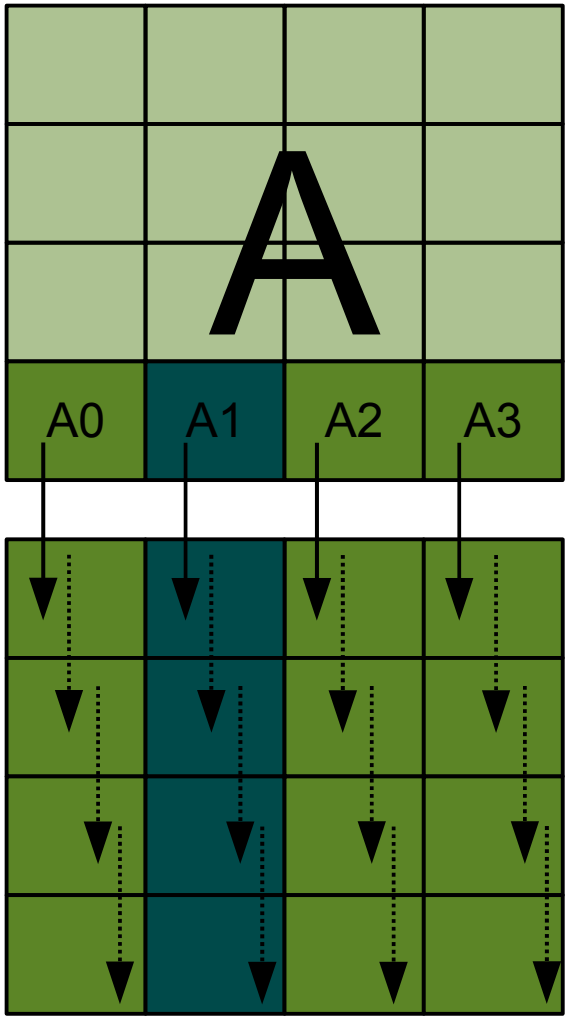
H_PRED



V_PRED

- Vertical Prediction
- Fills each pixel row with a copy of the row above (A)
- If current macroblock is on the top column, a default value of 127 is assigned

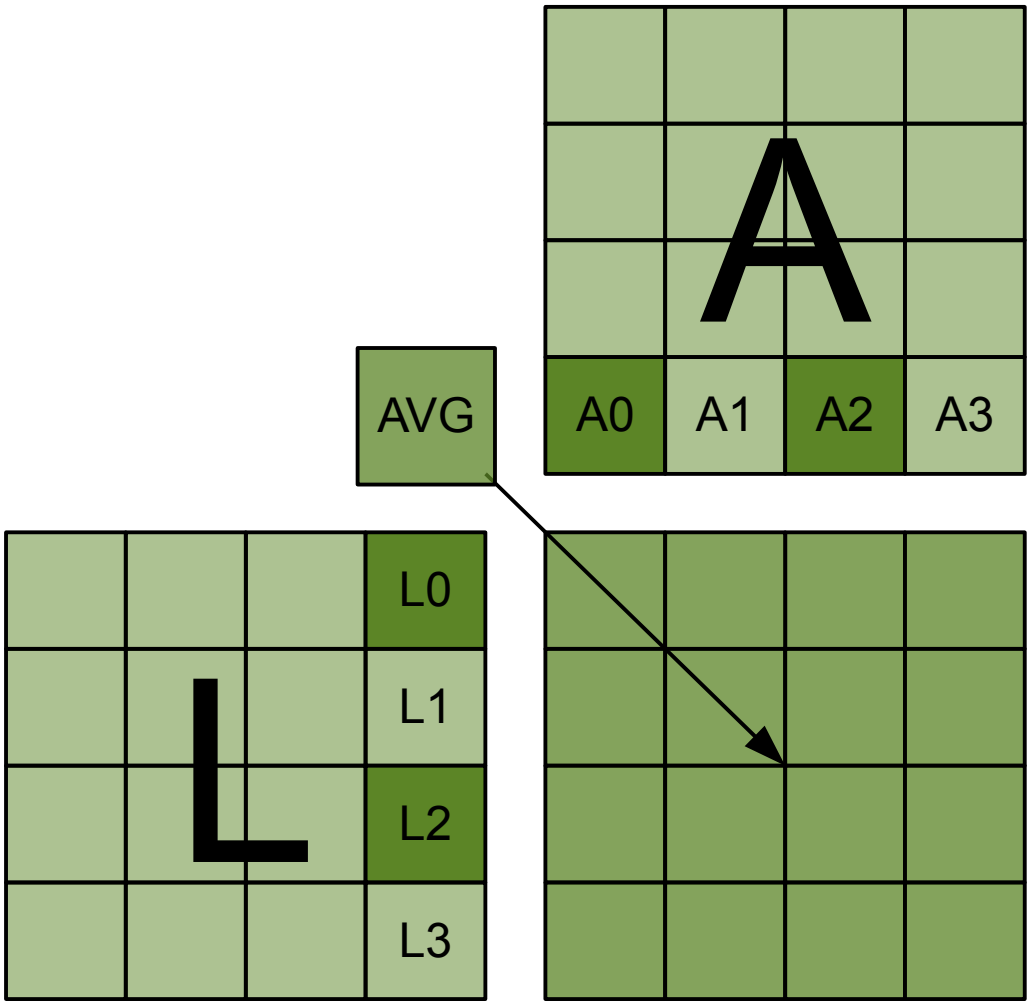
V_PRED



DC_PRED

- Fills each block with a single value
- This value is the average of the pixels left and above of the block
- If block is on the top: The average of the left pixels is used
- If block is on the left: The average of the above pixels is used
- If block is on the left top corner: A constant value of 128 is used

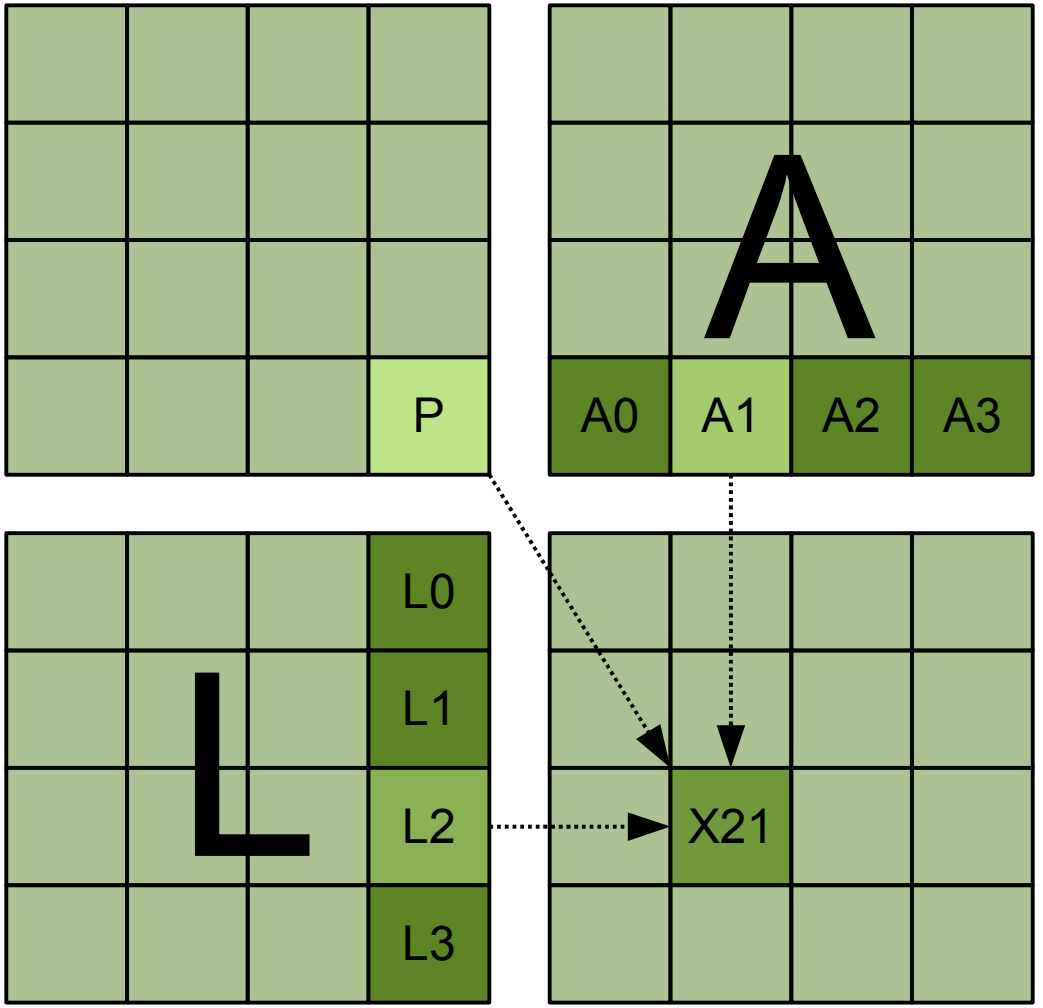
DC_PRED



TM_PRED

- TrueMotion Prediction
- Uses above row A, left column L and a pixel P which is above and left of the block
- Most used intra prediction mode
- $X_{ij} = L_i + A_j - P$

TM_PRED



X21 = L2 + A1 - P

TM_PRED Code

```
void TMpred(
    Pixel b[8][8],       /* prediction block */
    const Pixel A[8],  /* row of above block */
    const Pixel L[8],   /* column of left of block */
    const Pixel P        /* pixel P */
){
    int r = 0;          /* row */
    do {
        int c = 0;      /* column */
        do {
            b[r][c] = clamp255( L[r]+ A[c] - P);
        } while( ++c < 8);
    } while( ++r < 8);
}
```

Luma Prediction Modes

- Basically all chroma prediction modes
- With 16x16 macroblocks
- Additional B_PRED mode

B_PRED

- Splits 16x16 macroblock into 16 4x4 sub-blocks
- Each sub-block is independently predicted
- Ten available prediction modes for sub-blocks

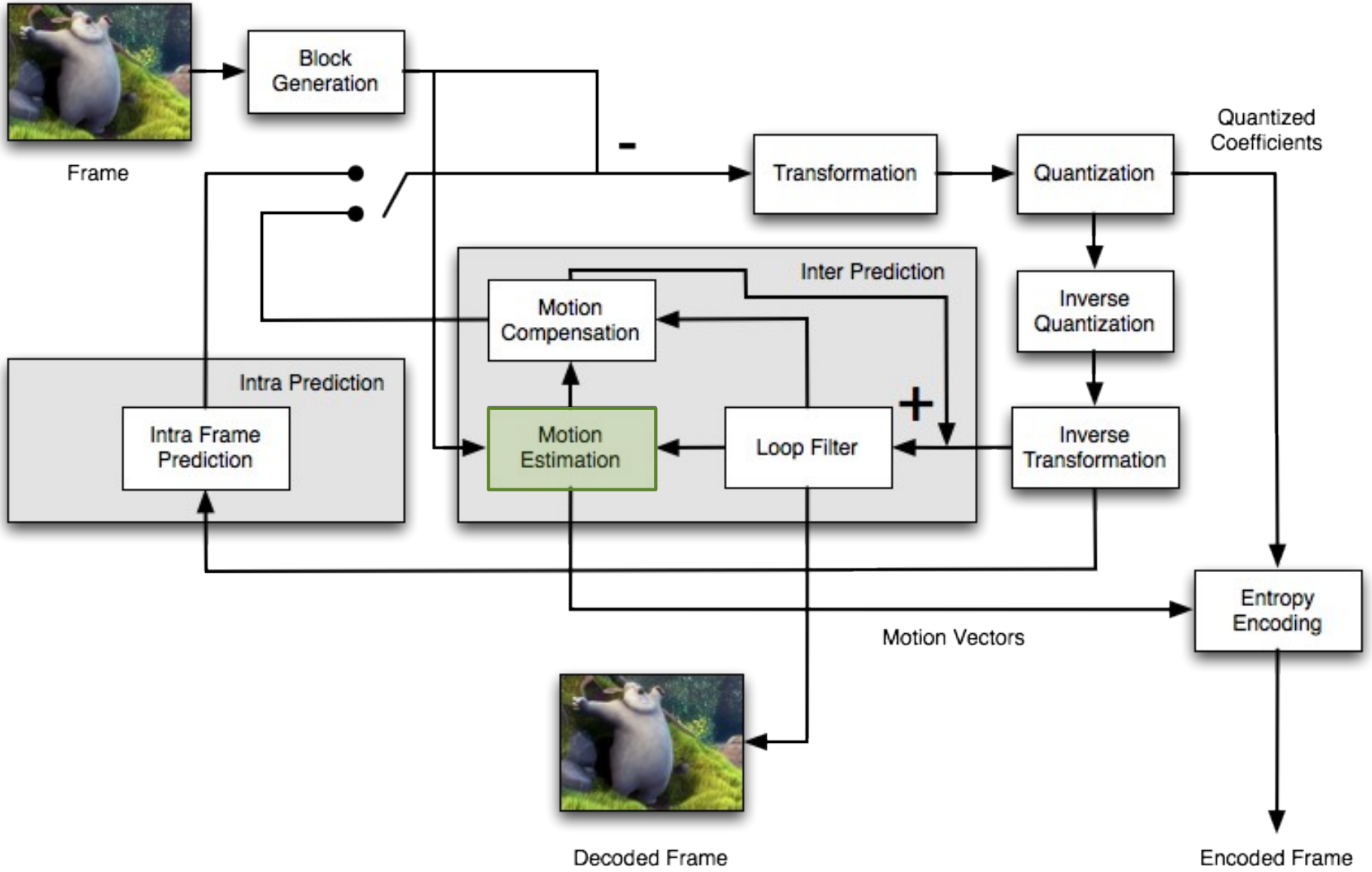
B_PRED Modes

- B_DC_PRED: predict DC using row above and column
- B_TM_PRED: propagate second differences a la TM
- B_VE_PRED: predict rows using row above
- B_HE_PRED: predict columns using column to the left
- B_LD_PRED: southwest (left and down) 45 degree diagonal prediction

B_PRED Modes

- B_RD_PRED: southeast (right and down)
- B_VR_PRED: SSE (vertical right) diagonal
- B_VL_PRED: SSW (vertical left)
- B_HD_PRED: ESE (horizontal down)
- B_HU_PRED: ENE (horizontal up)

Motion Estimation



Motion Estimation

- Determine motion vectors which transform one frame to another
- Uses motion vectors for 16x16, 16x8, 8x16, 8x8 and 4x4 blocks
- Motion vectors from neighboring blocks can be referenced

Motion Estimation

- Motion vector: Horizontal and vertical displacement
- Only luma blocks are predicted, chroma blocks are calculated from luma
- Resolution: 1/4 pixel for luma, 1/8 pixel for chroma
- Chroma vectors are calculated by averaging vectors from luma blocks

Motion Vector Types

- MV_NEAREST
- MV_NEAR
- MV_ZERO
- MV_NEW
- MV_SPLIT

MV_NEAREST

- Re-use non-zero motion vector of last decoded block

MV_NEAR

- Re-use non-zero motion vector of second-to-last decoded block

MV_ZERO

- Block has not moved
- Block is at the same position as in preceding frame

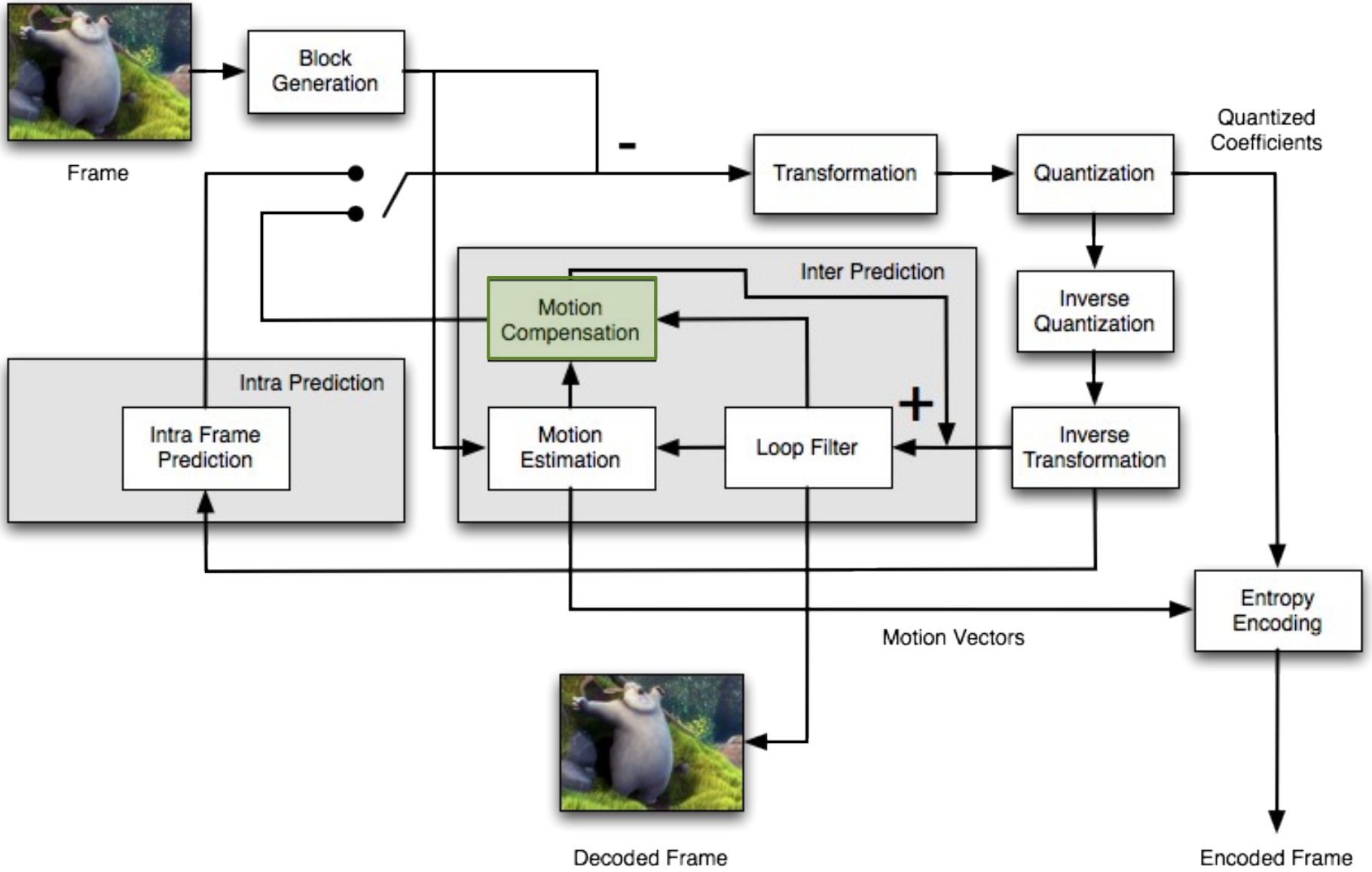
MV_NEW

- New motion vector
- Mode followed by motion vector data
- Data is added to buffer of last encoded blocks

MV_SPLIT

- Use multiple motion vectors for a macroblock
- Macroblock can be split up into sub-blocks
- Each sub-block can have its own motion vector
- Useful when objects within a macroblock have different motion characteristics

Motion Compensation



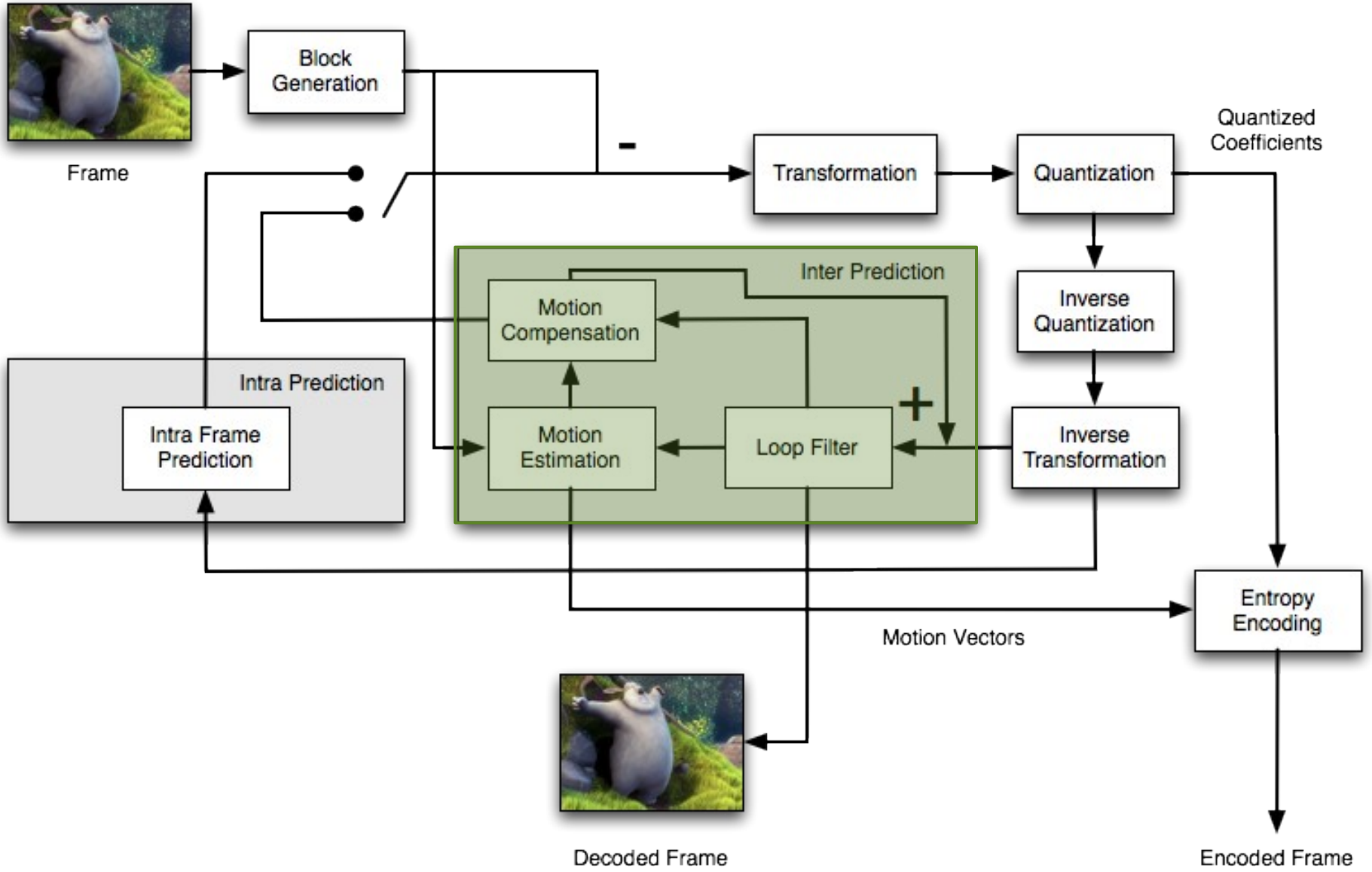
Motion Compensation

- Apply motion vectors to previous frame
- Generate a predicted frame
- Only difference between predicted and actual frame needs to be transmitted

Sub-pixel Interpolation

- If “full pixel” motion vector, block is copied to corresponding piece of the prediction buffer
- If at least one of the displacements affects sub-pixels, missing pixels are synthesized by horizontal and vertical interpolation

Inter Frame Prediction



Inter Frame Prediction

Exploits the temporal coherence between nearby frames

Components:

- Reference Frames
- Motion Vectors

Inter-Frame Types

- Key Frames
  - Decoded without reference to other frames
  - Provide seeking points
- Predicted Frames
  - Decoding depends on all prior frames up to last Key-Frame
- No usage of B-Frames

Prediction Frame Types

- Previous Frame
- Alternate Reference Frame
- Golden Reference Frame
- Each of these three types can be used for prediction

Previous Frame

- Last fully decoded frame
- Updated with every shown frame

Alternate Reference Frame

- Fully decoded frame buffer
- Can be used for noise reduced prediction
- In combination with golden frames: Compensate lack of B-frames

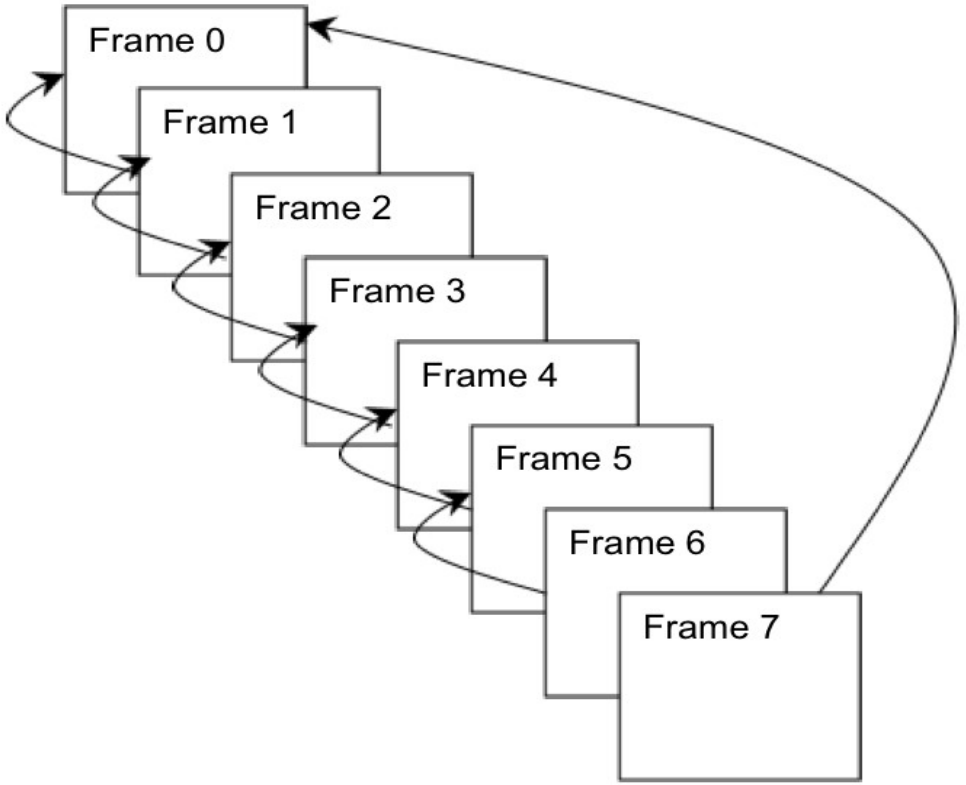
Golden Reference Frame

- Fully decoded image buffer
- Can be partially updated
- Can be used for error recovery
- Can be used to encode a cut between scenes

Updating Frame Buffers

- Key frame: Updates all three buffers
- Predicted frame: Flag for updating alternate or golden frame buffer

Error Recovery



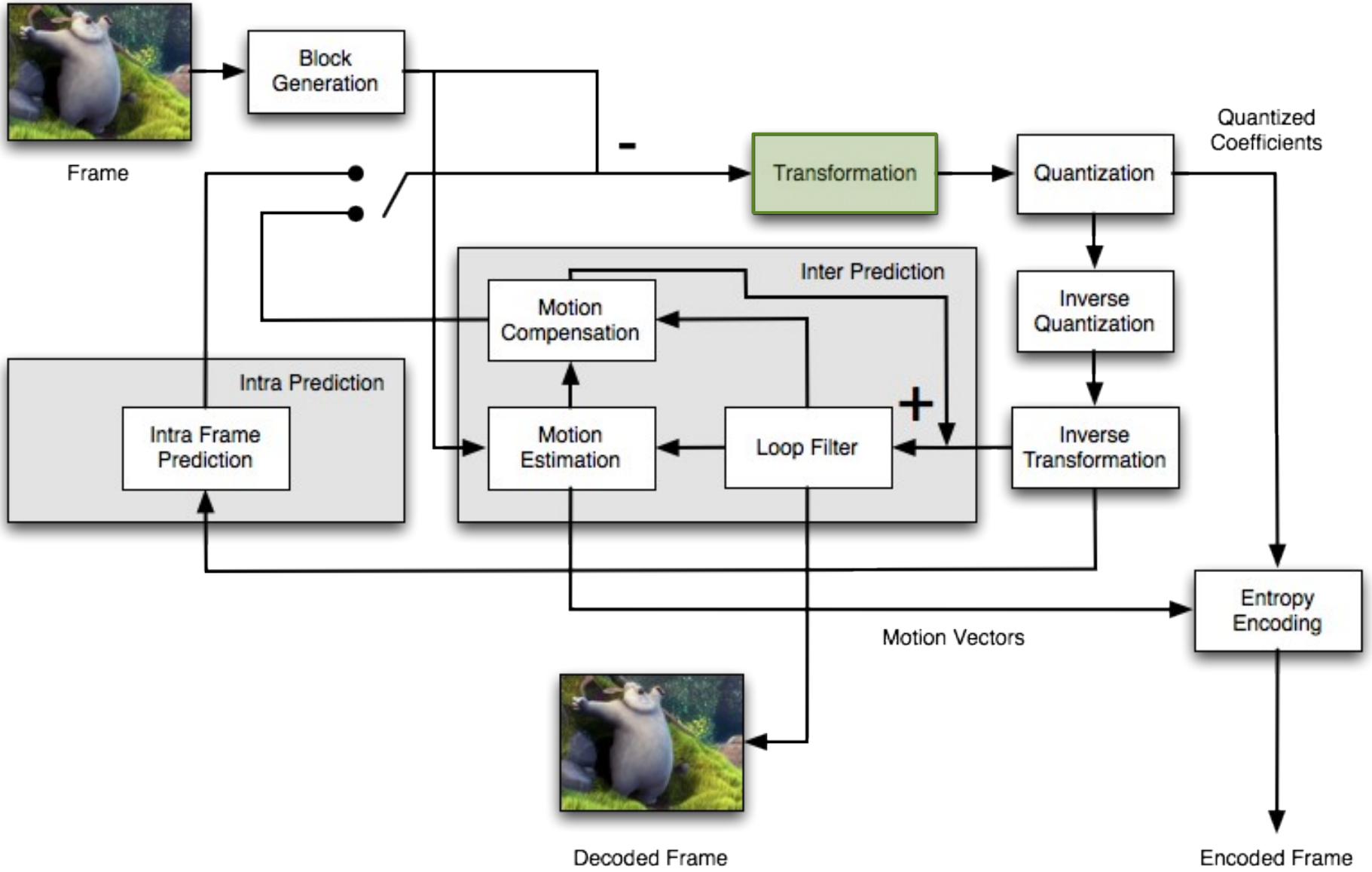
Frame 0 is a key frame / gold frame
Frame 1 through 6 build predictors using the prior frame
Frame 7 uses only frame 0 as a reference.

If any frame between 1 and 6 is lost VP8 can still decode frame 7 as it references only to frame 0.

Source: http://webm.googlecode.com/files/Realtime_VP8_2-9-2011.pdf

Transformation

Transformation

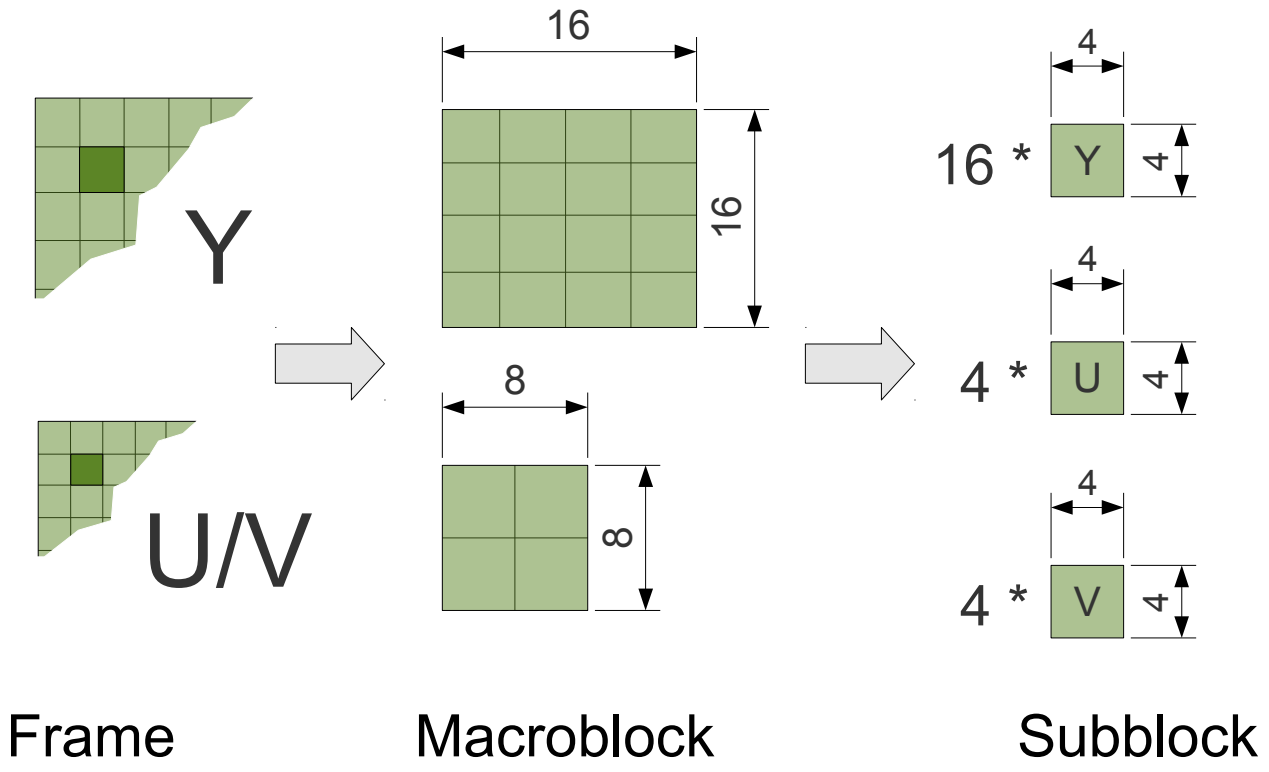


Decorrelation

- Necessary for efficient entropy encoding
- Achieved with hybrid transformation
  - Discrete Cosine Transformation
  - Walsh-Hadamard Transformation

Transformation

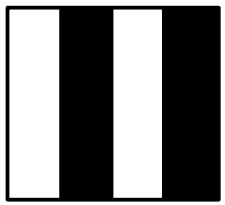
Preparation for transfomation process: Divide Macroblocks into Subblocks



Discrete Cosine Transformation

- 16 luma blocks / 4 + 4 chroma blocks
- Transform each block into spectral components using the 2D - DCT

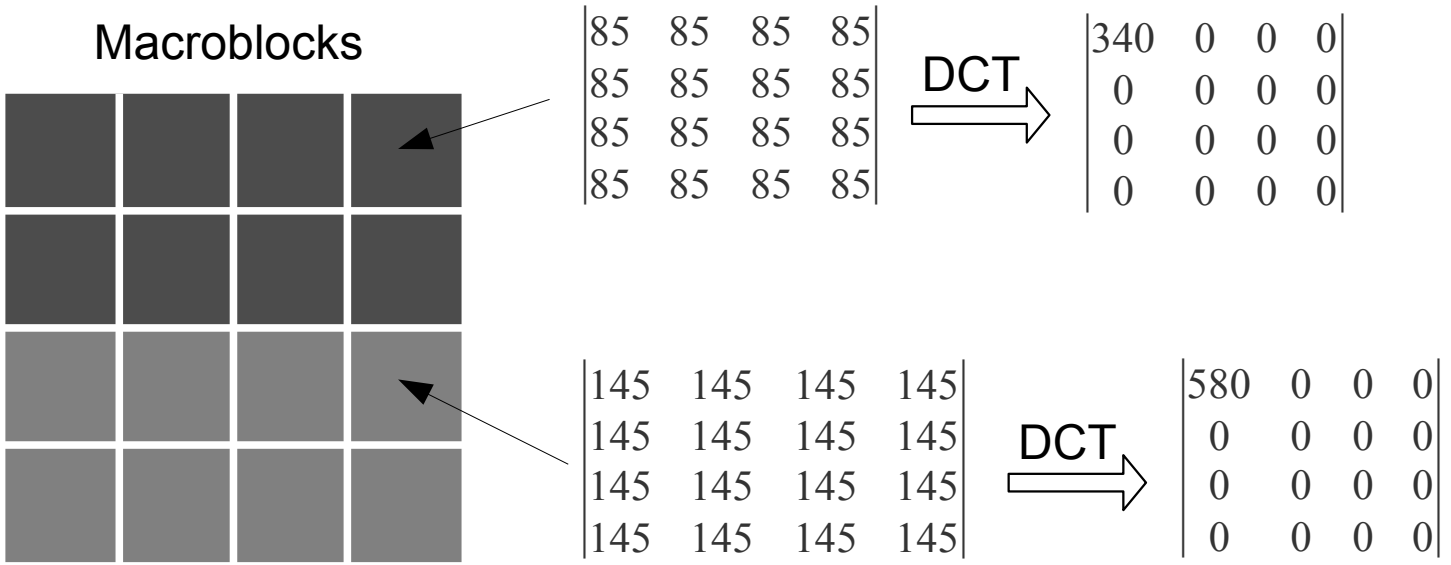
$$\begin{vmatrix} 255 & 0 & 255 & 0 \\ 255 & 0 & 255 & 0 \\ 255 & 0 & 255 & 0 \\ 255 & 0 & 255 & 0 \end{vmatrix} \xrightarrow{\text{DCT}} \begin{vmatrix} 510 & 195.1686 & 0 & 471.1786 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{vmatrix}$$



Values based on dct2() function of Matlab

Transformation

The DC components of all subblocks are often correlated among each other



Values based on dct2() function of Matlab

Walsh-Hadamard Transformation

- Use the correlation of the DC components with a 2nd order transformation
- The WHT works with a simple transformation matrix

  → Transformation is a matrix multiplication

$$H = \begin{vmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{vmatrix} \qquad H = \frac{1}{\sqrt{4}} \begin{vmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{vmatrix}$$

Normalized Walsh-Hadamard matrix

Walsh-Hadamard Transformation

Example

1st order transformation DC components

$$A = \begin{vmatrix} 340 & 340 & 340 & 340 \\ 340 & 340 & 340 & 340 \\ 580 & 580 & 580 & 580 \\ 580 & 580 & 580 & 580 \end{vmatrix}$$

Normalized transformation matrix

$$H = \begin{vmatrix} 1/2 & 1/2 & 1/2 & 1/2 \\ 1/2 & 1/2 & -1/2 & -1/2 \\ 1/2 & -1/2 & 1/2 & -1/2 \\ 1/2 & -1/2 & -1/2 & 1/2 \end{vmatrix}$$

(Re) Transformation

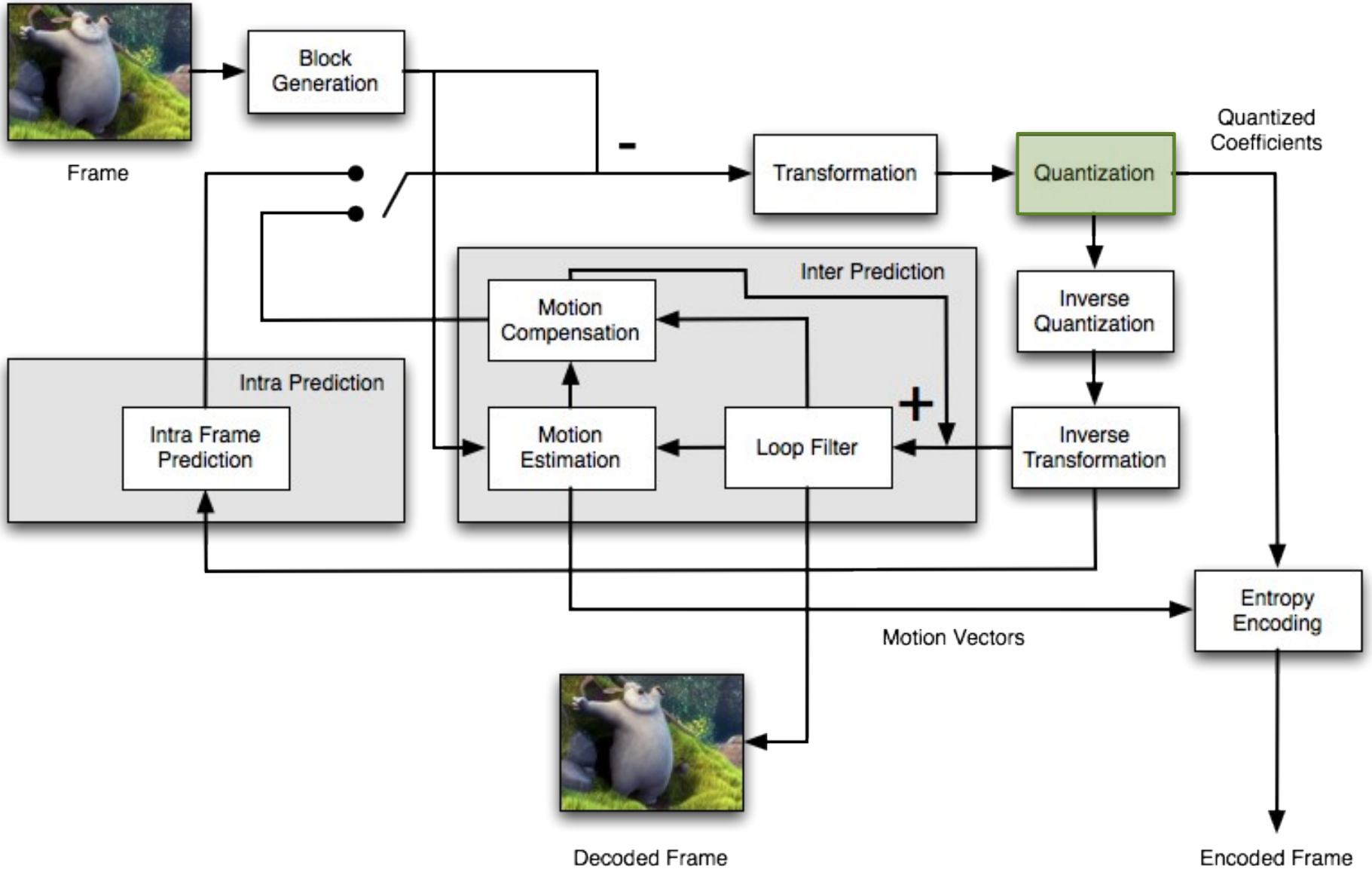
$$H * A * H$$

$$B = H * A = \begin{vmatrix} 1840 & 1840 & 1840 & 1840 \\ -480 & -480 & -480 & -480 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{vmatrix}$$

$$C = B * H = \begin{vmatrix} 1840 & 0 & 0 & 0 \\ -480 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{vmatrix}$$

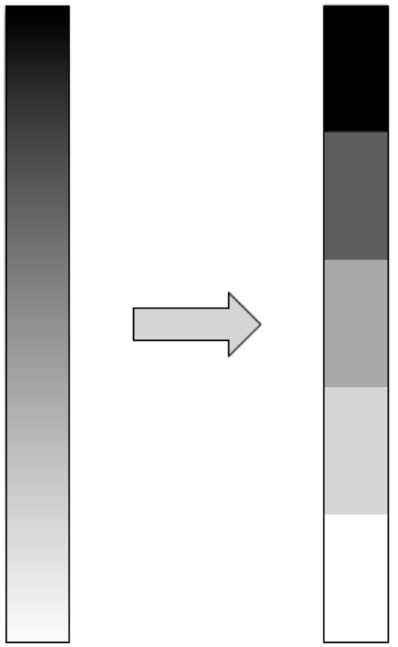
Quantization

Quantization



Quantization

- Quantization of the transformation coefficients:
  - Less data per coefficient
  - More zeros!
- Scalar quantization
- Designed for quality range of
  ~30dB to ~45dB SNR

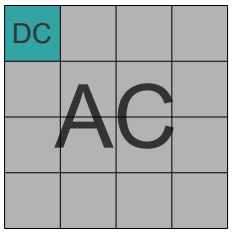


Quantization

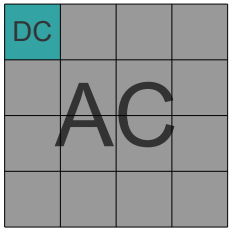
For each frame different factors for:

- 1st order luma DC
- 1st order luma AC
- 2nd order luma DC
- 2nd order luma AC
- Chroma DC
- Chroma AC

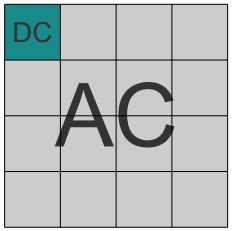
2nd order luma (WHT)



1st order luma (DCT)



Chroma (DCT)



Quantization

- 128 quantization levels with given factors
- Quantization table for DC coefficients in Y1 planes

```
static const int dc_qlookup[QINDEX_RANGE] =
{
     4,   5,   6,   7,   8,   9,  10,  10,   11,  12,  13,  14,  15,
    16,  17,  17,  18,  19,  20,  20,  21,   21,  22,  22,  23,  23,
    24,  25,  25,  26,  27,  28,  29,  30,   31,  32,  33,  34,  35,
    36,  37,  37,  38,  39,  40,  41,  42,   43,  44,  45,  46,  46,
    47,  48,  49,  50,  51,  52,  53,  54,   55,  56,  57,  58,  59,
    60,  61,  62,  63,  64,  65,  66,  67,   68,  69,  70,  71,  72,
    73,  74,  75,  76,  76,  77,  78,  79,   80,  81,  82,  83,  84,
    85,  86,  87,  88,  89,  91,  93,  95,   96,  98, 100, 101, 102,
    104, 106, 108, 110, 112, 114, 116, 118, 122, 124, 126, 128, 130,
    132, 134, 136, 138, 140, 143, 145, 148, 151, 154, 157
};
```

Quantization

Example: 1st order luma AC coefficients

- Quantization level: 3

  → Quantization factor from table: 6
- DC coefficient is ignored here

$$A = \begin{vmatrix} -312 & 7 & 1 & 0 \\ 1 & 12 & -5 & 2 \\ 2 & -3 & 3 & -1 \\ 1 & 0 & -2 & 1 \end{vmatrix} \qquad Q = round\left(\frac{1}{6} * A\right) = \begin{vmatrix} 0 & 1 & 0 & 0 \\ 0 & 2 & -1 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{vmatrix}$$

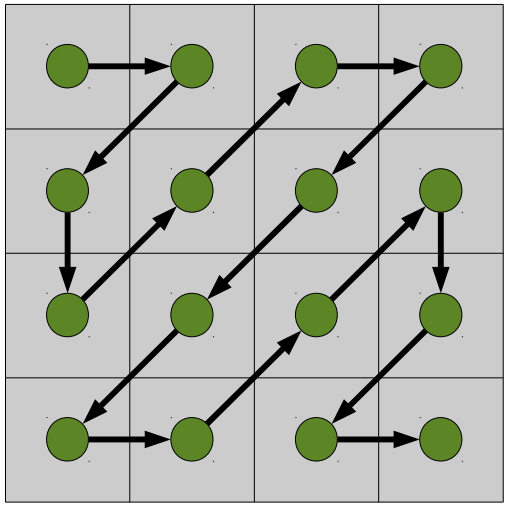
Quantization

Adaptive Quantization

- Up to 4 different segments (q0-q3)
- Each segment with n macroblocks and its own quantization parameter set

Quantization

The quantized coefficients are read in zig-zag order

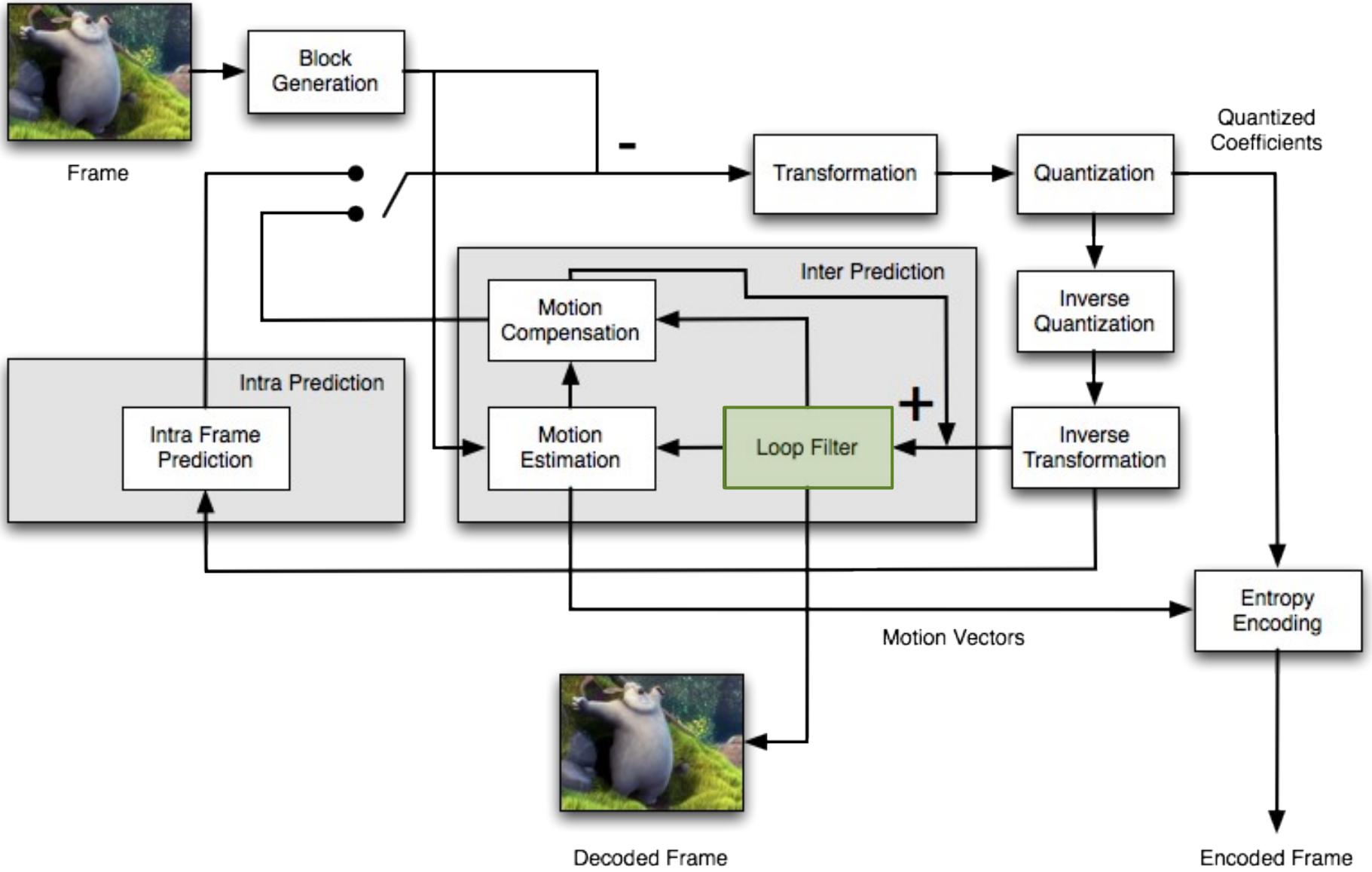


-19	1	0	0
0	2	-1	0
-1	0	0	0
0	0	0	0

$$vals = [-19, 1, 0, -1, 2, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0]$$

Adaptive Loop Filtering

Adaptive Loop Filtering

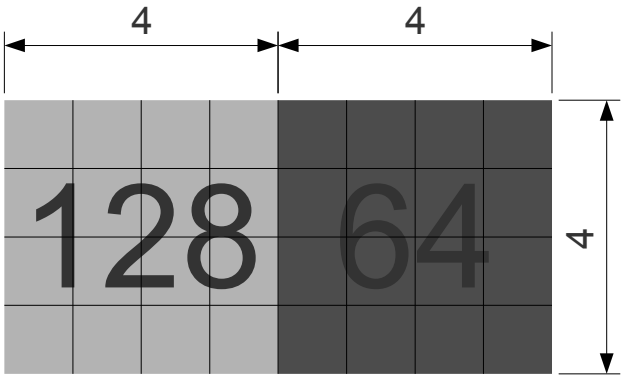


Adaptive Loop Filtering

Problem

- Strong quantization (“worst” case: only DC)
- Many pixels with same values
- Blocking artifacts

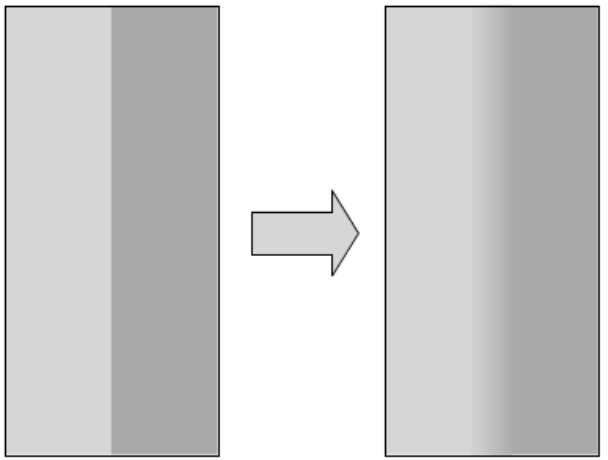
$$A = \begin{vmatrix} 128 & 128 & 128 & 128 \\ 128 & 128 & 128 & 128 \\ 128 & 128 & 128 & 128 \\ 128 & 128 & 128 & 128 \end{vmatrix} \quad B = \begin{vmatrix} 64 & 64 & 64 & 64 \\ 64 & 64 & 64 & 64 \\ 64 & 64 & 64 & 64 \\ 64 & 64 & 64 & 64 \end{vmatrix}$$



Subblocks

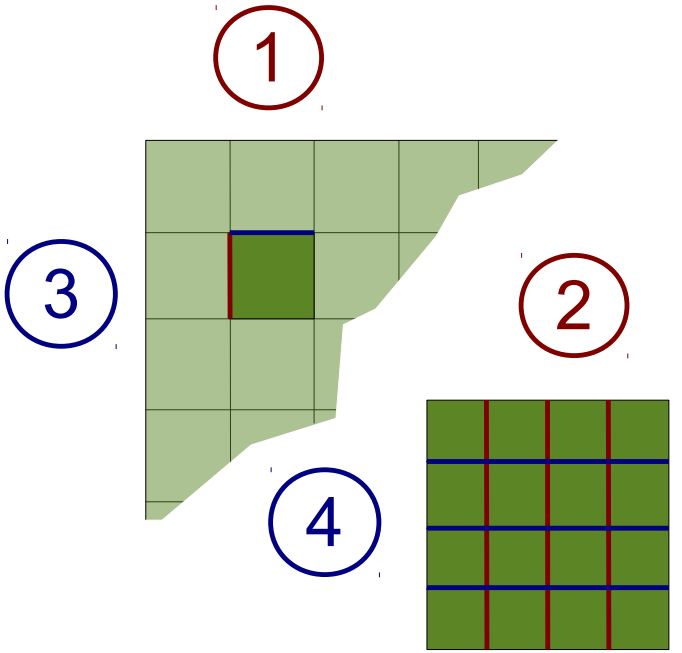
Adaptive Loop Filtering

- VP8 has two filter modes
  - Simple
  - Normal
- Configuration in frame-header
- Two parameters
  - loop_filter_level
  - sharpness_level



Adaptive Loop Filtering

- Filter order per macroblock
  1. Left macroblock edge
  2. Vertical subblock edges
  3. Macroblock edge at the top
  4. Horizontal subblock edges

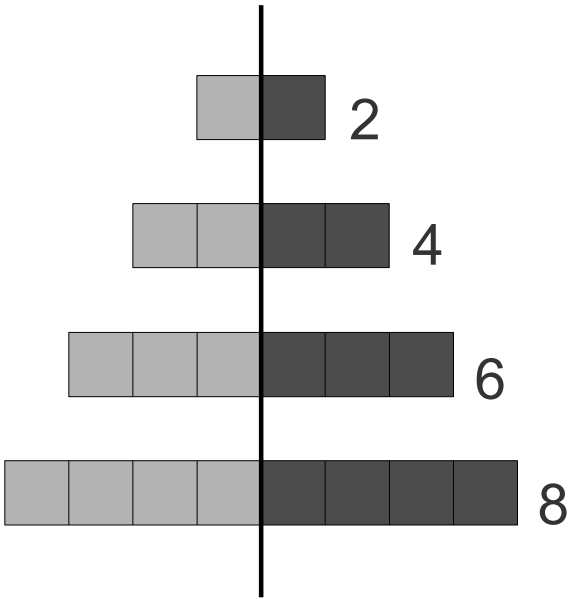


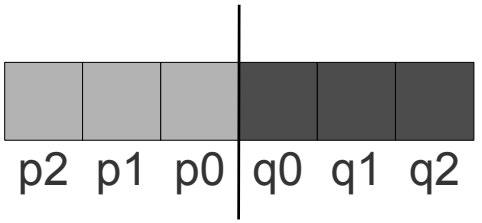
- Macroblock processing in scan line order

Adaptive Loop Filtering

Filter segments

- n segments per edge
  n = blocklength
- 2,4,6 or 8 taps wide
- Pixels before edge: px
- Pixels after edge: qx





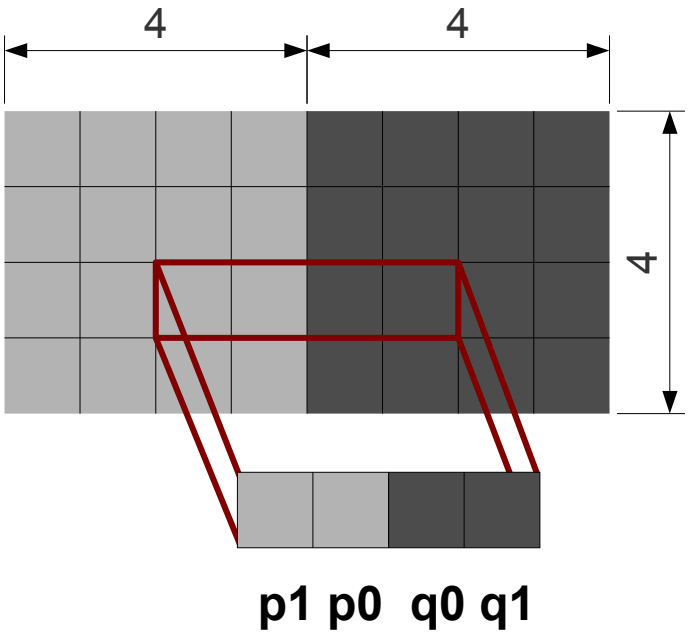
Adaptive Loop Filtering

Simple Mode

- Segments 4 or 6 taps wide
- sharpness_level ignored
- Filter edge if total difference > threshold
- Threshold derived from loop_filter_level, quantization level and other factors

Adaptive Loop Filtering

Simple Mode – Example



p1: 128
p0: 128
q0: 64
q1: 64

$$p = \frac{(3*128)+128}{4} = 128$$

$$q = \frac{(3*64)+64}{4} = 64$$

$$a = \frac{q-p}{4} = \frac{64-128}{4} = -16$$

$$p0 = p0 + a = 128 + (-16) = 122$$

$$q0 = q0 - a = 128 - (-16) = 80$$

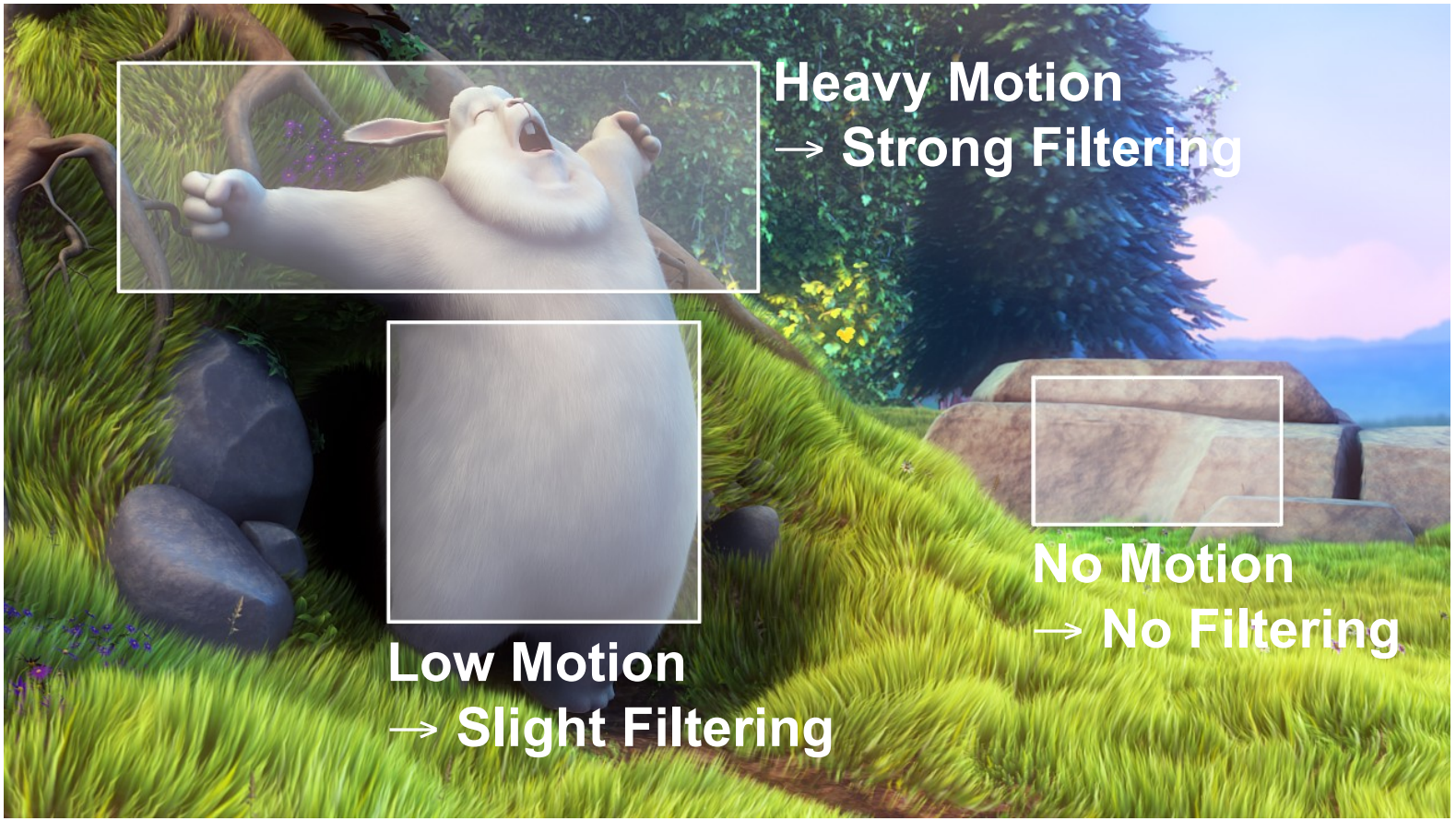
Adaptive Loop Filtering

Normal Mode

- Segments 2,4,6 or 8 taps wide
- Different adjustments for different positions
- Different weightings for inner positions

Adaptive Loop Filtering

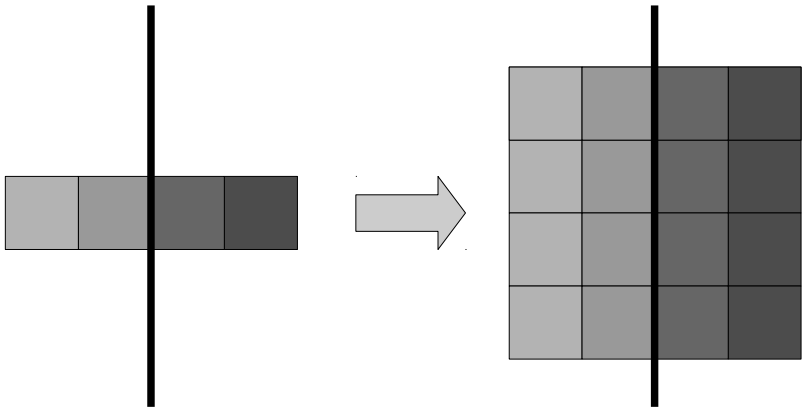
Adaptive?



Adaptive Loop Filtering

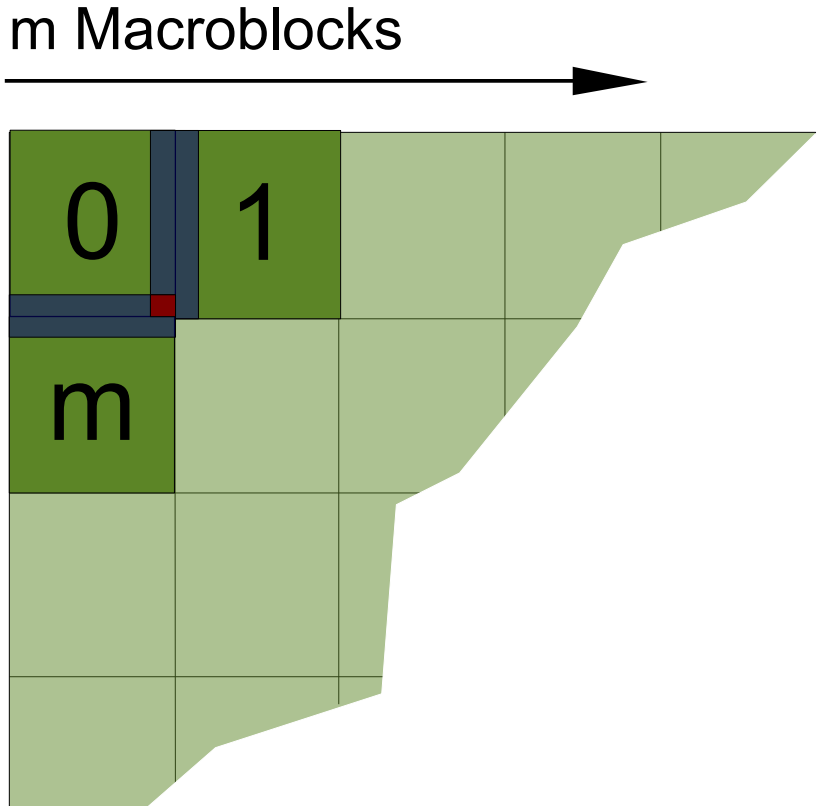
SIMD processors aka Vector CPUs

- Loop filter optimized for SIMD operations
- Sources already implemented



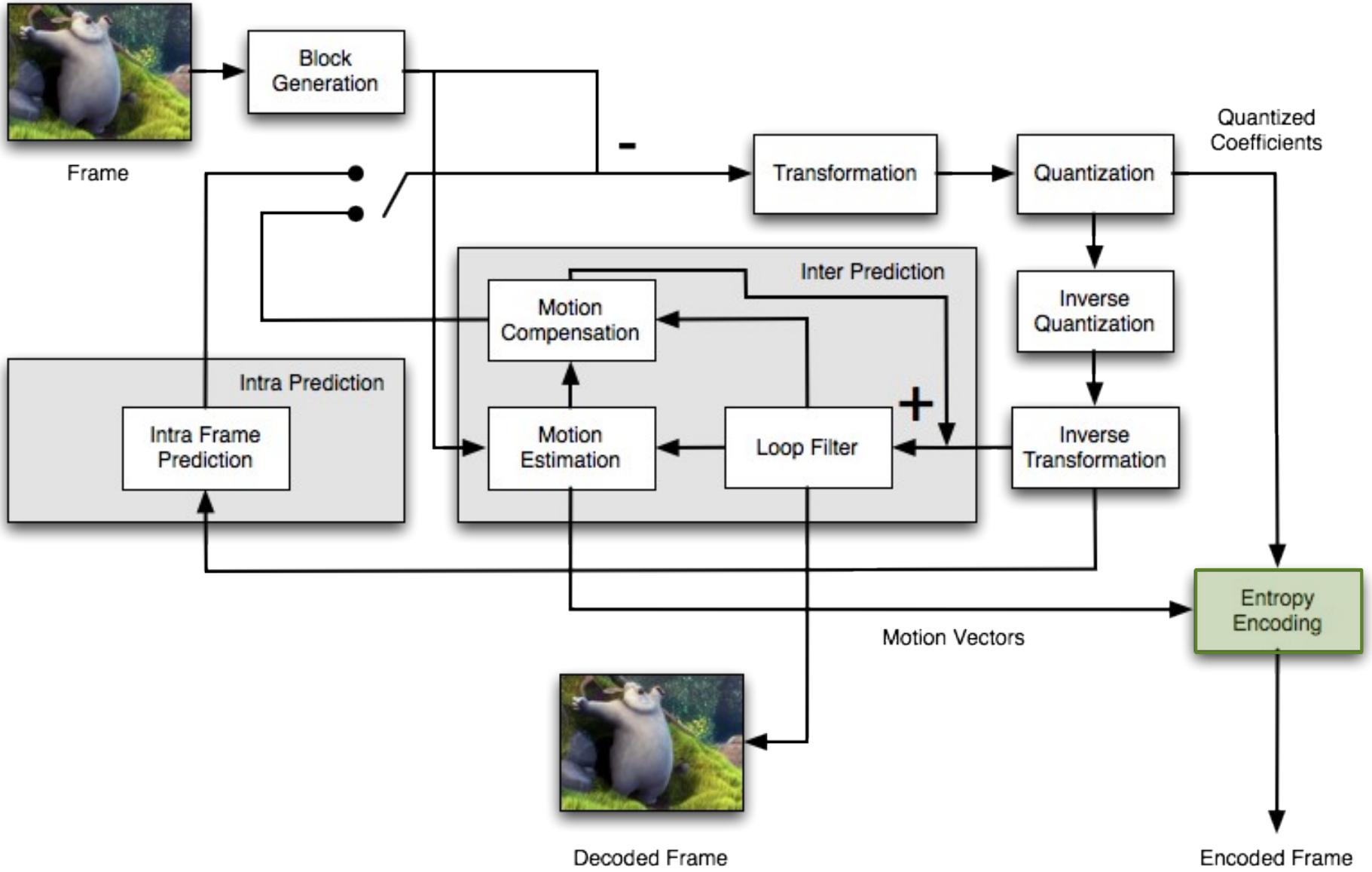
Adaptive Loop Filtering

Problem: Dependencies between macroblocks



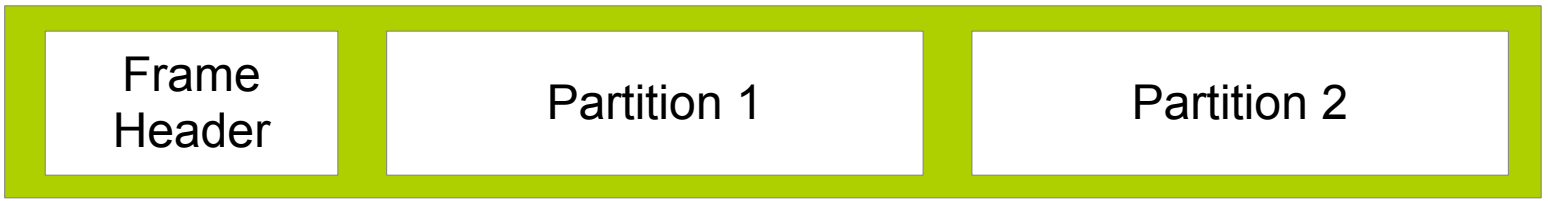
Entropy Coding

Entropy Coding



Frame Format

- Frames are divided in 3 partitions
  - Uncompressed header chunk
  - Macroblock coding modes and motion vectors
  - Quantized transform coefficients



Entropy Encoding

- Entropy coding minifies redundancy
- 2 steps
  - Huffman Tree with a small alphabet
  - Binary arithmetic coding

Entropy Encoding

- DCT and WHT coefficients are precoded to tokens using a predefined tree structure
- Goal
  - Reduce number of reads from raw binary stream
- Solution
  - Create tokens for symbol values
  - Minimize necessary reads for most frequent symbols

Entropy Encoding

- Token types
  - Single numbers
    - Coefficient value
      - 0, 1, 2, 3, 4
  - Number ranges
    - 6 ranges of coefficient values
      - 5-6, 7-10, 11-18, 19-34, 35-66, 67-2048
  - EOB (End Of Block)
    - No more non-zeros values remaining in macroblock

Entropy Encoding

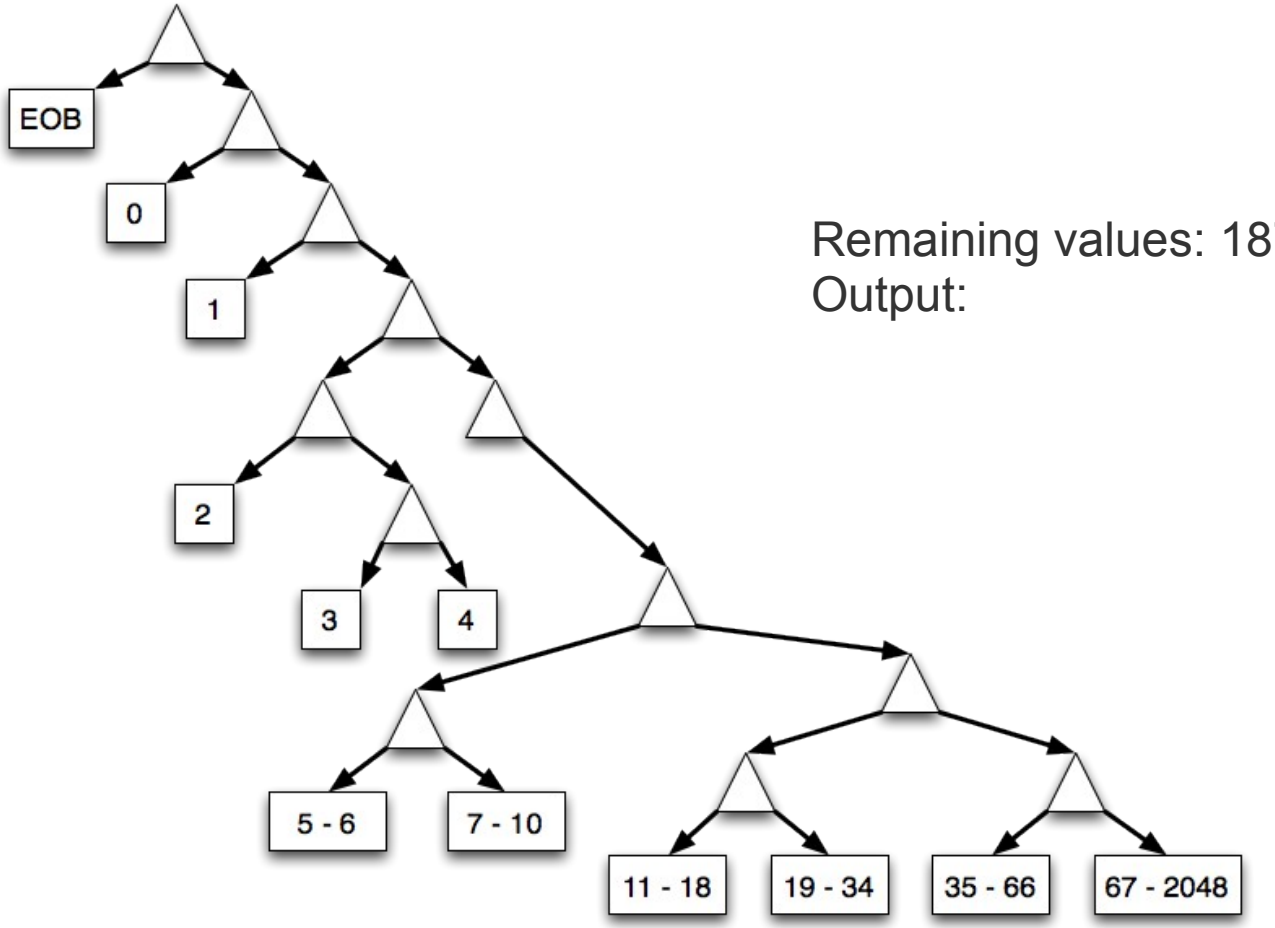
- How are these tokens created?
- Step 1: Read quantized DCT/WHT coefficients from 4x4 sub-blocks

$$\begin{vmatrix} 187 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{vmatrix}$$

187, 0, 2, 1, 0, 0, 0, 0, 0, ...

Entropy Encoding

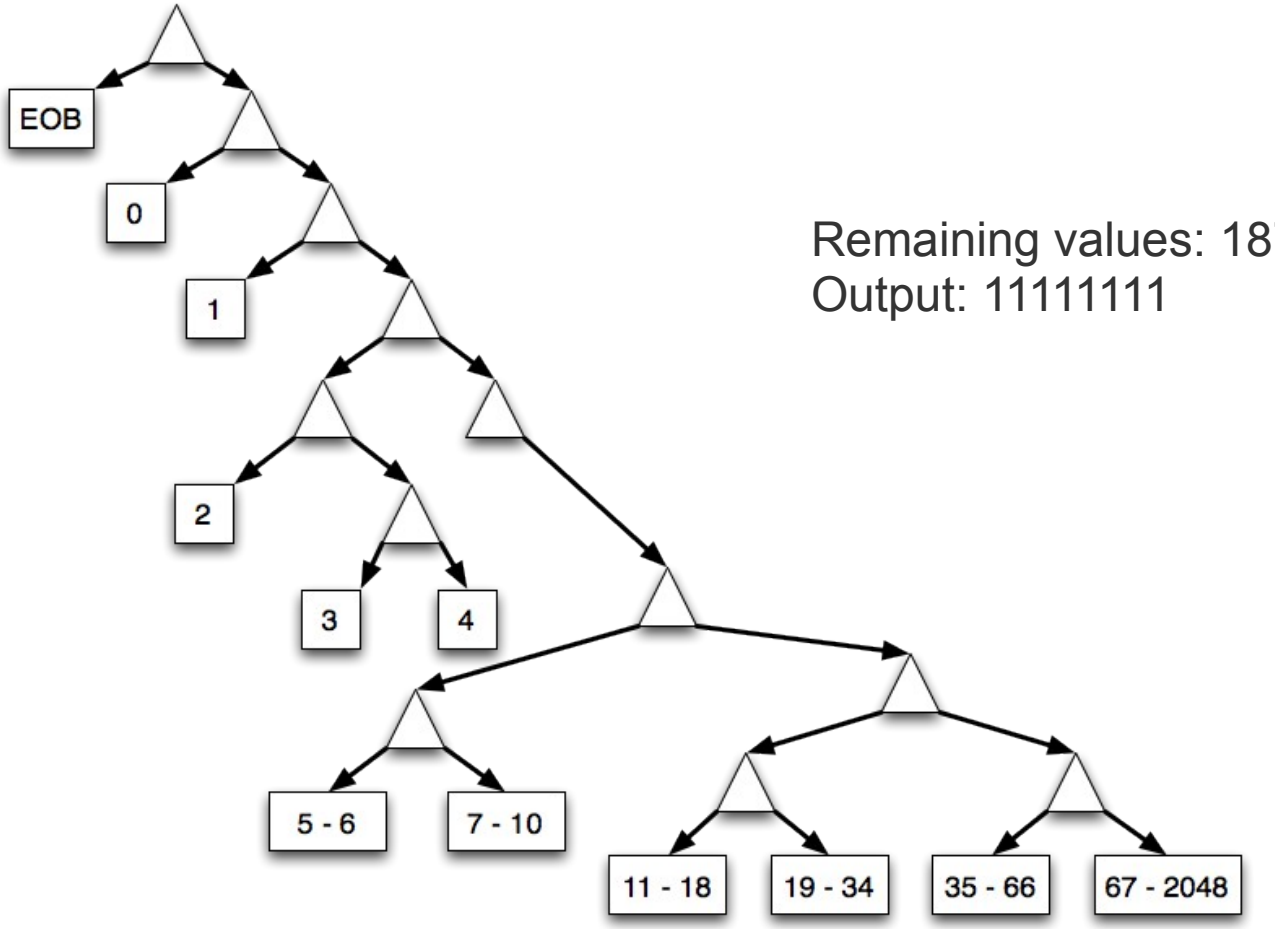
- Step 2: Lookup regarding tokens for each value



Remaining values: 187, 0, 2, 1, 0, 0, 0, 0, 0, ...
Output:

Entropy Encoding

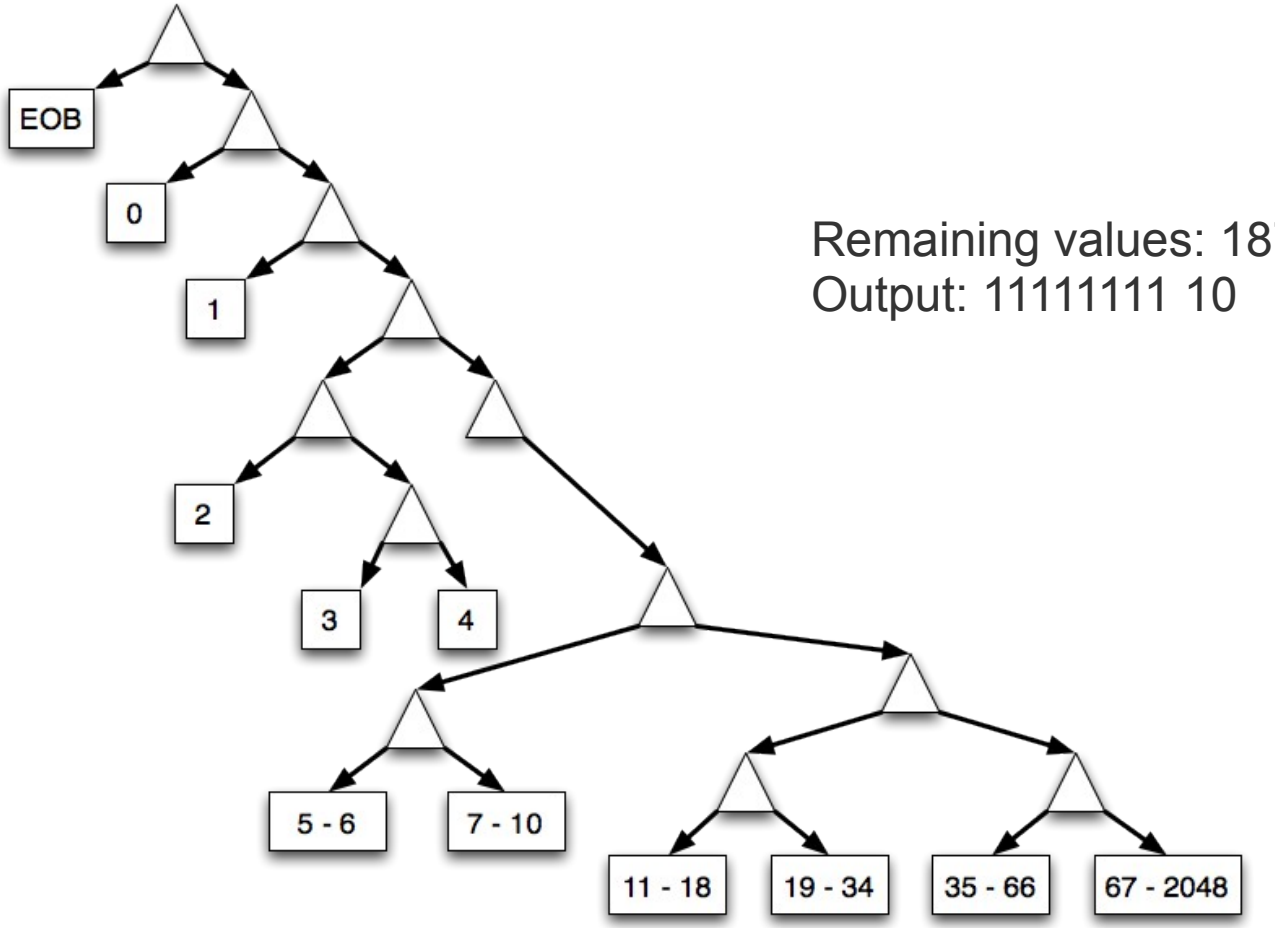
- Step 2: Lookup regarding tokens for each value



Remaining values: 187, 0, 2, 1, 0, 0, 0, 0, 0, ...
Output: 11111111

Entropy Encoding

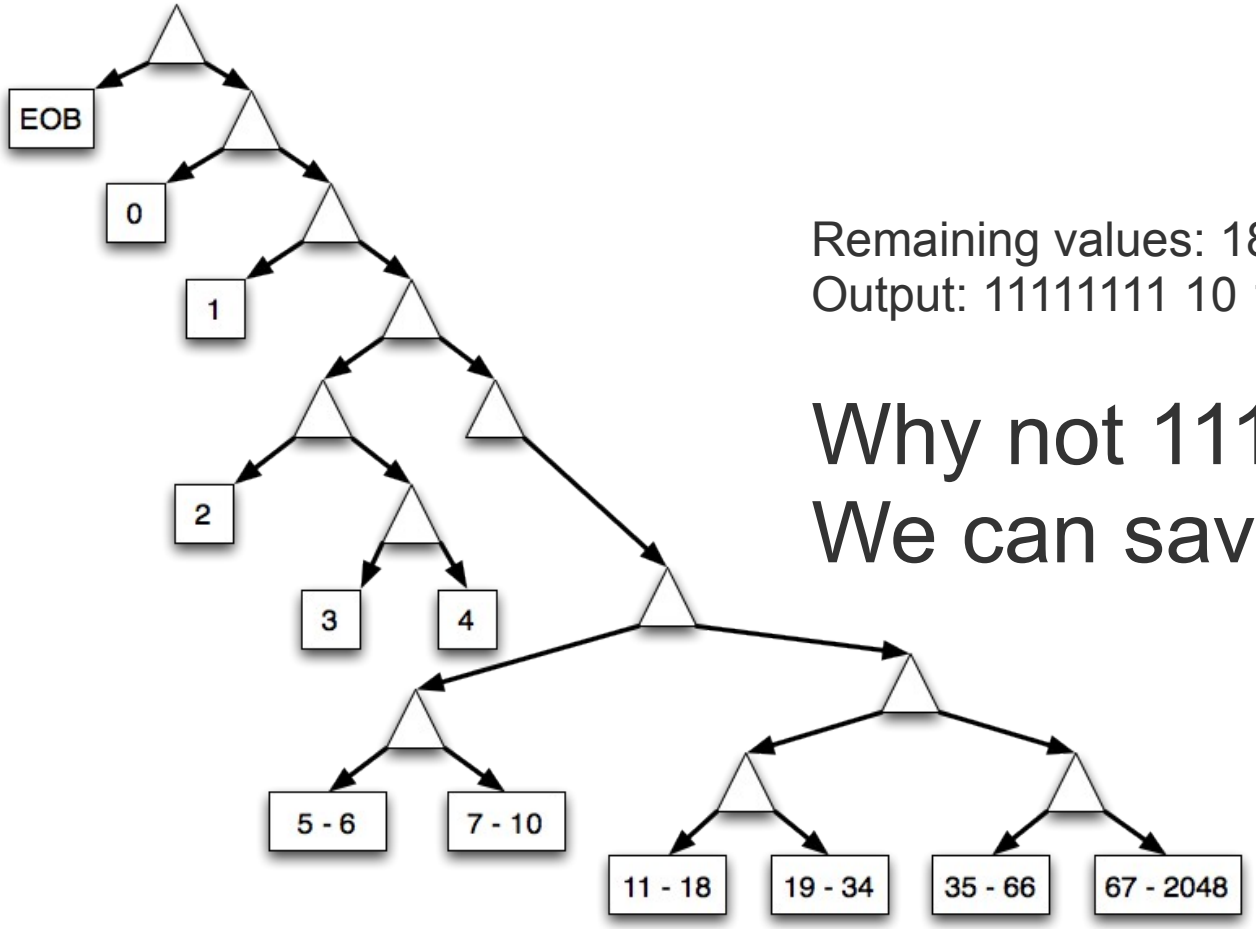
- Step 2: Lookup regarding tokens for each value



Remaining values: 187, 0, 2, 1, 0, 0, 0, 0, 0, ...
Output: 11111111 10

Entropy Encoding

- Step 2: Lookup regarding tokens for each value

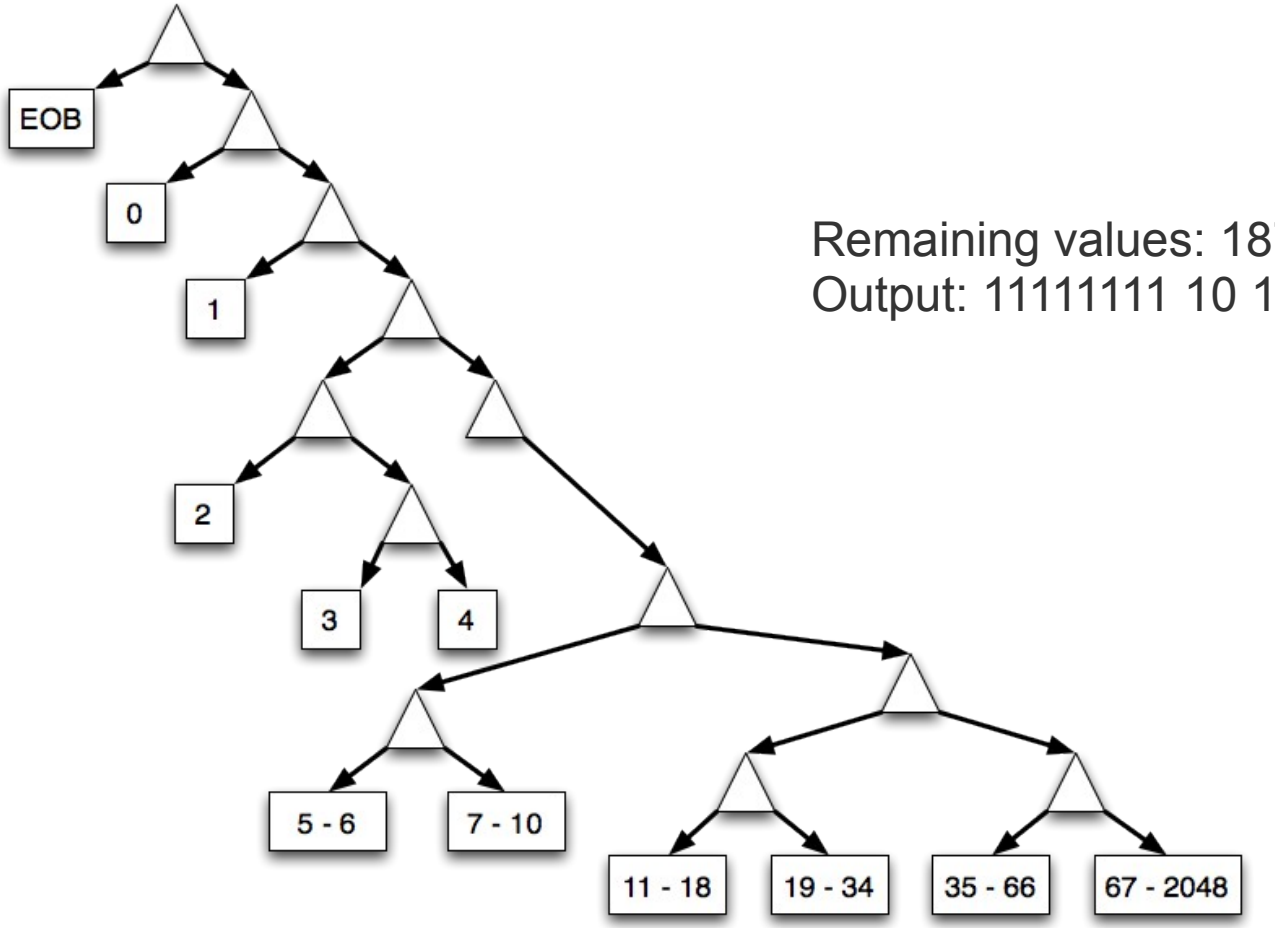


Remaining values: 187, 0, 2, 1, 0, 0, 0, 0, 0, ...
Output: 11111111 10 1100

Why not 11100?
We can save 1 bit!

Entropy Encoding

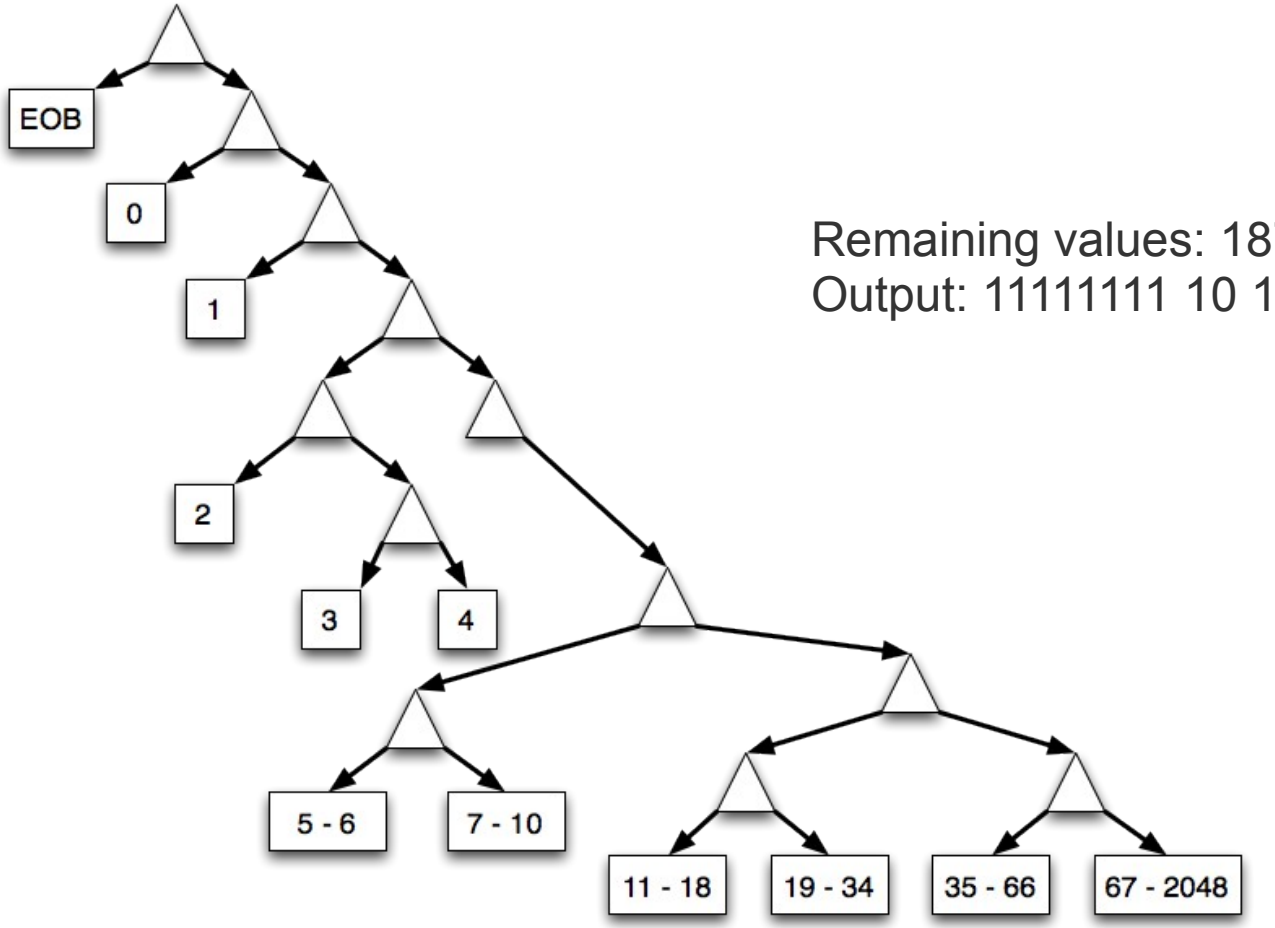
- Step 2: Lookup regarding tokens for each value



Remaining values: 187, 0, 2, 1, 0, 0, 0, 0, 0, ...
Output: 11111111 10 1100 110

Entropy Encoding

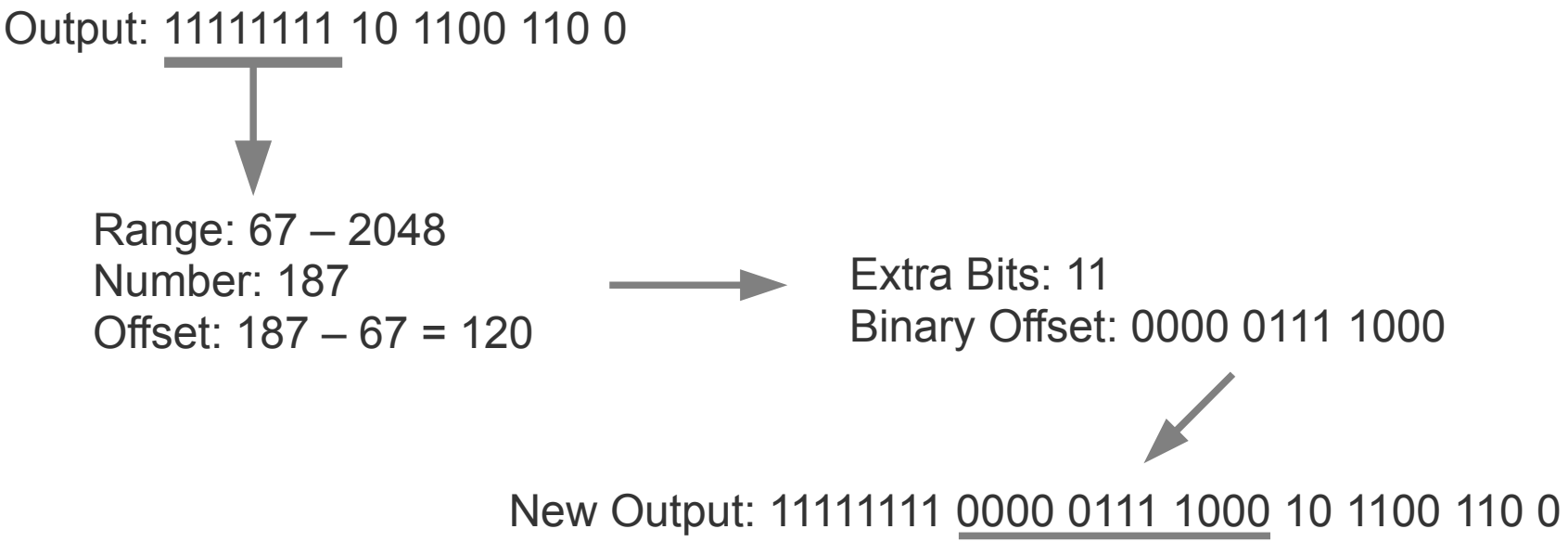
- Step 2: Lookup regarding tokens for each value



Remaining values: 187, 0, 2, 1, 0, 0, 0, 0, 0, ...
Output: 11111111 10 1100 110 0

Entropy Encoding

- Restoring coefficients from value ranges
  - Add some extra bits as offset from base of the current range



Entropy Encoding

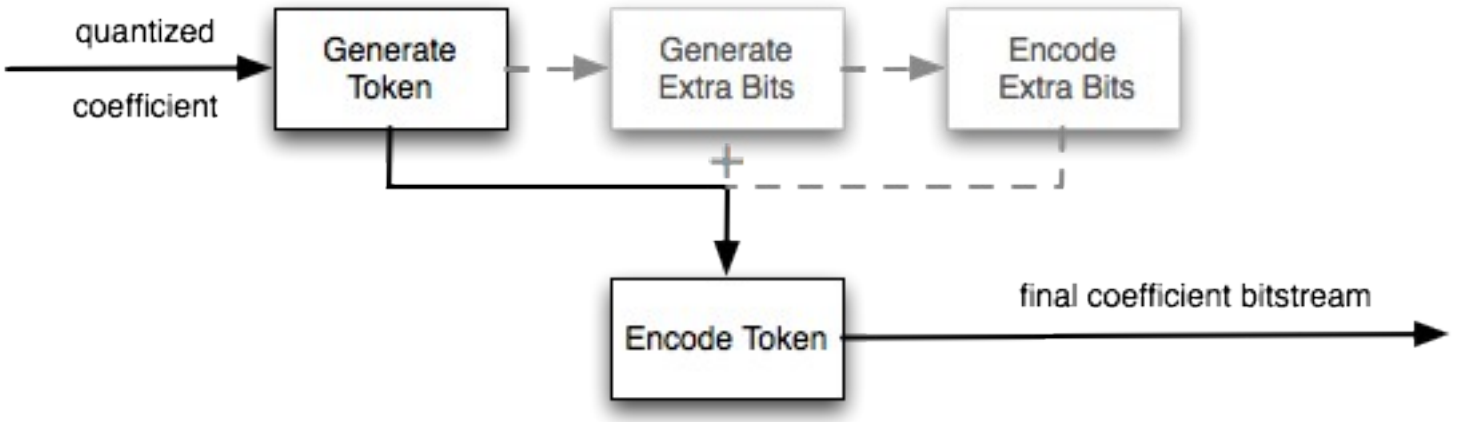
- Binary arithmetic encoding
  - Extra bits are encoded with pre-set, constant probabilities
  - Token probabilities reside in 96 probability tables
  - Token bits are encoded with
    - Default probabilities whenever keyframes are updated
    - Regarding probability tables can be updated with each new frame

Entropy Encoding

- Binary arithmetic encoding
  - Token probability tables are chosen according to 3 contexts
    - Plane (Y, U, V)
    - Band (position of the coefficient)
    - Local complexity (value of the preceding coefficient)

Entropy Encoding

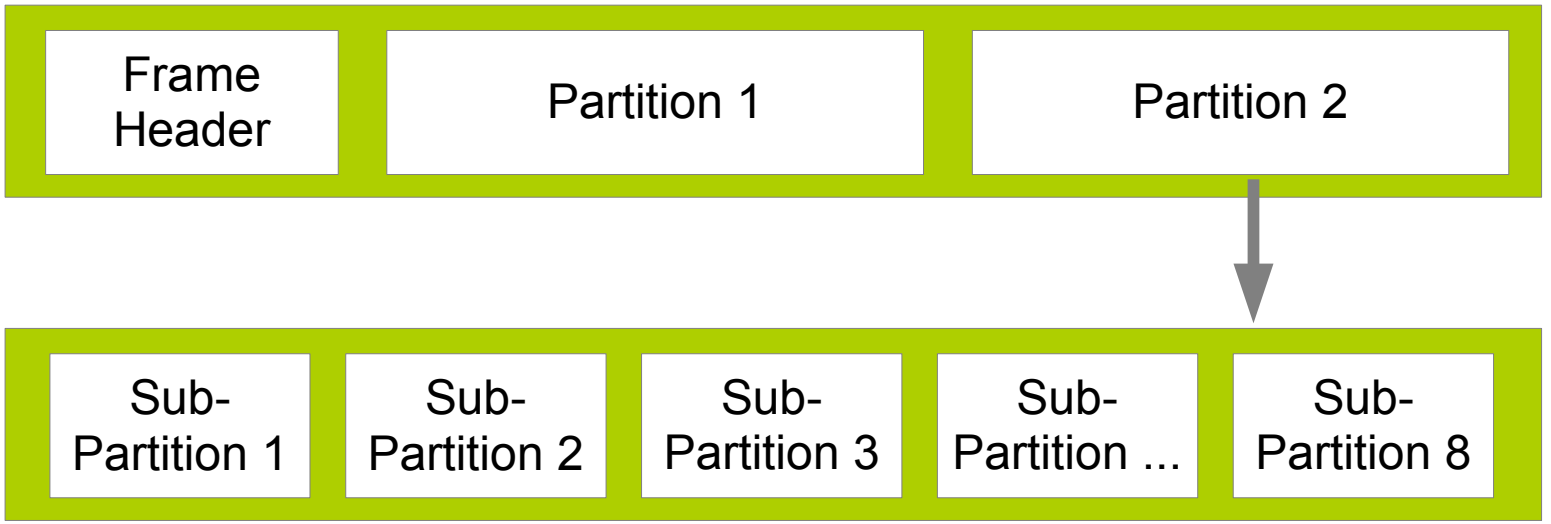
- Binary arithmetic encoding



Parallel Processing

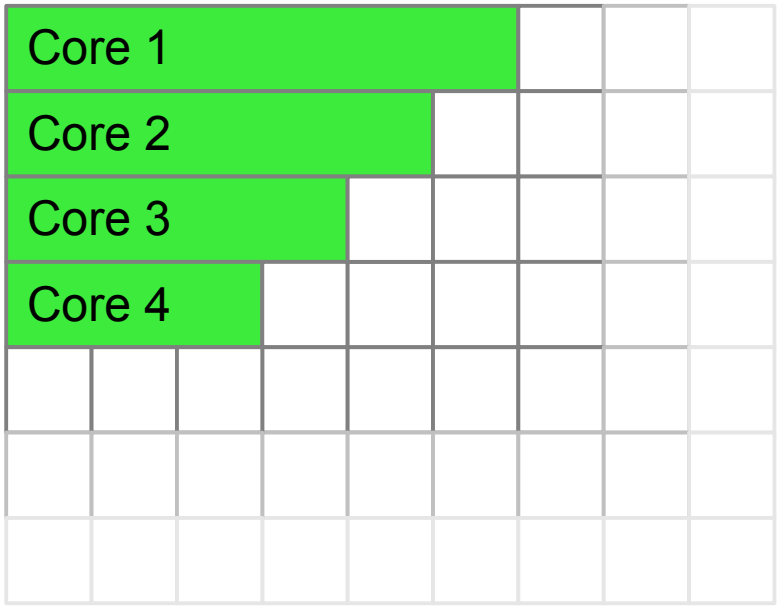
Parallel Processing

- Partition 2 (DCT/WHT coefficients) can be divided in 8 sub-partitions



Parallel Processing

- Partition 2 (DCT/WHT coefficients) can be divided sub-partitions
- Support for up to 8 cores

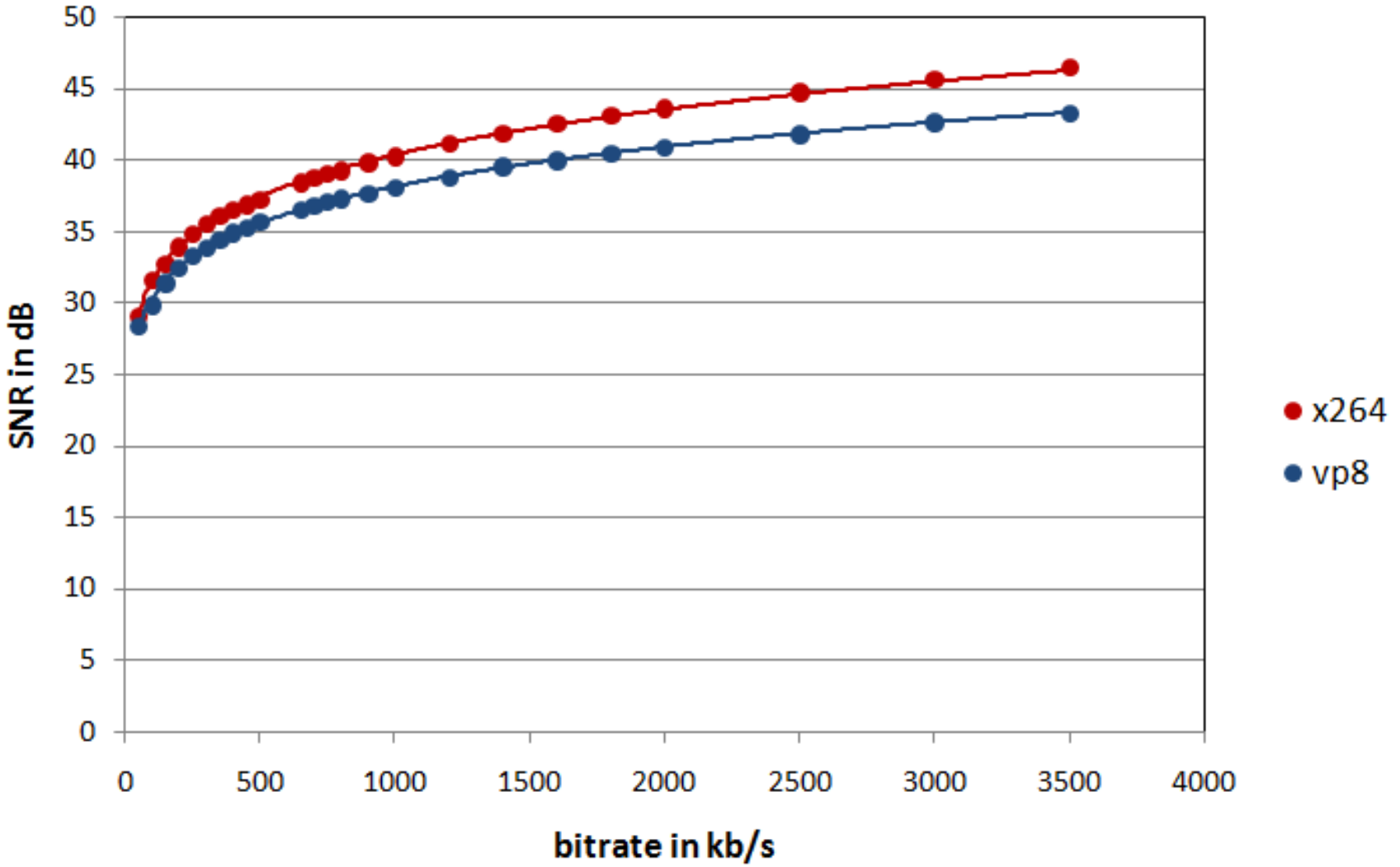


Benchmarks

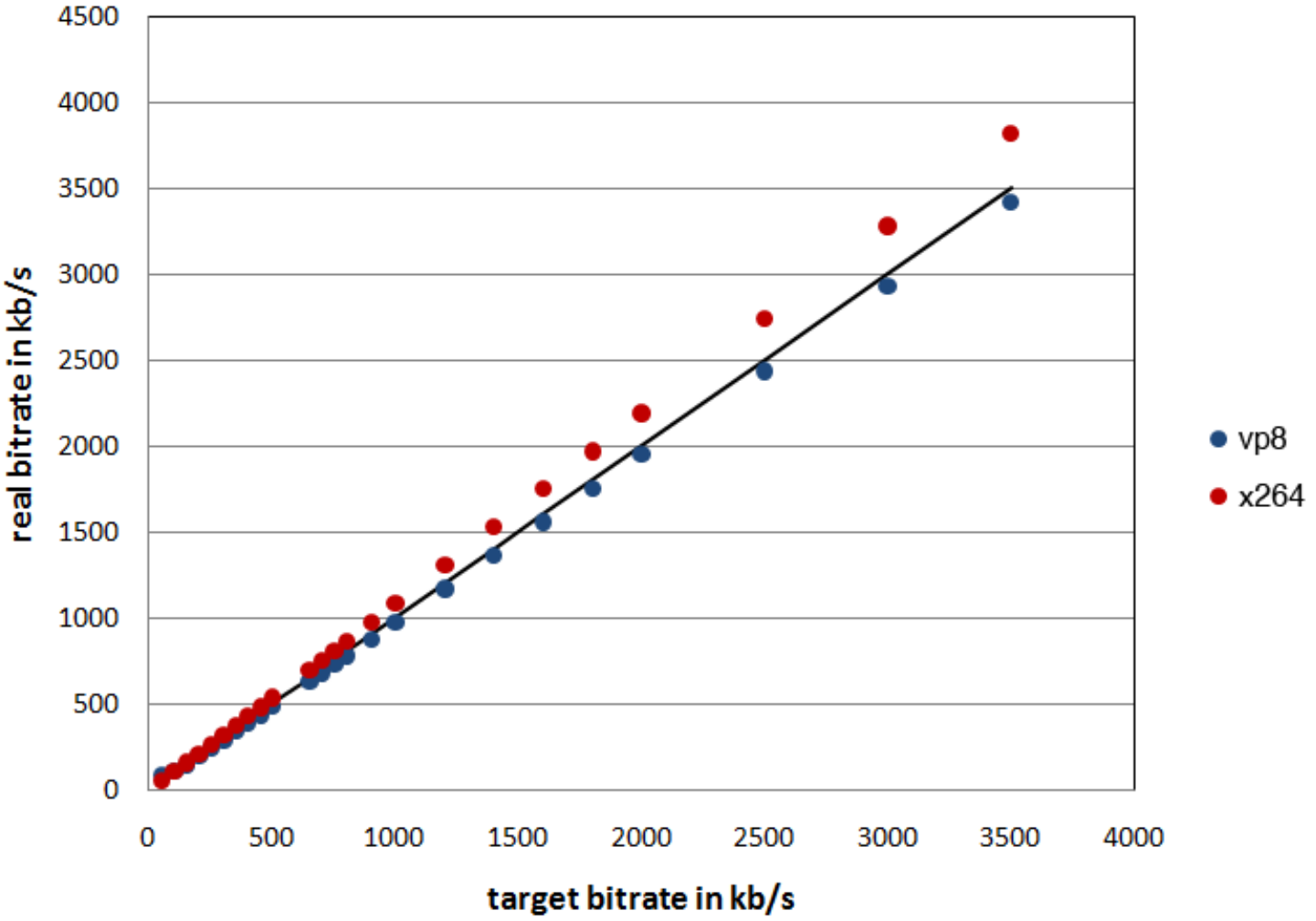
Tools

- ffmpeg
- libvpx
- libx264
- custom scripts
- qpsnr (qpsnr.youlink.org)

Benchmarks


50
45
40
35
30
25
20
15
10
5
0
SNR in dB
0
500
1000
1500
2000
2500
3000
3500
4000
bitrate in kb/s
x264
vp8

Benchmarks


4500
4000
3500
3000
2500
2000
1500
1000
500
0
real bitrate in kb/s
0
500
1000
1500
2000
2500
3000
3500
4000
target bitrate in kb/s
vp8
x264

Demos

Conclusions

- “Good enough” for web video
- Maybe new default choice for web video
- “There’s no way in hell anyone could write a decoder solely with this spec alone.” - x264 developer
- Patent situation still unclear

Conclusions

Resources

- http://x264dev.multimedia.cx
- http://multimedia.cx/eggs
- http://www.slideshare.net/DSPIP/google-vp8
- http://qpsnr.youlink.org/vp8_x264/VP8_vs_x264.html
- http://tools.ietf.org/html/draft-bankoski-vp8-bitstream-01
- Google VP8 Paper

Questions?