Mpeg WebVC

ISO/IEC 14496-29 Web video coding
Open Source Hardware

Contents

Mpeg WebVC

- WebVC === H.264 constrained baseline
- Purpose: a royalty free web codec
  - a clean spec simplifies IP sign-off
  - Editors ☺ kept the exact structure of AVC.
- A WebVC encoder is compatible with all AVC decoders!
- WVC Open source hardware encoder
  - Lower complexity
  - Low latency (¼ mSec)

"This International Standard specifies Web Video Coding, a technology that is compatible with the Constrained Baseline Profile of ISO/IEC 14996-10. Only the subset that is specified in Annex A for the Constrained Baseline Profile is a normative specification, while all remaining aspects are informative. This text is derived from ISO/IEC 14996-10, with which the section numbers in this specification are aligned, and that specification may additionally be consulted if desired, as an aid to understanding this Specification."

WebVC vs AVC

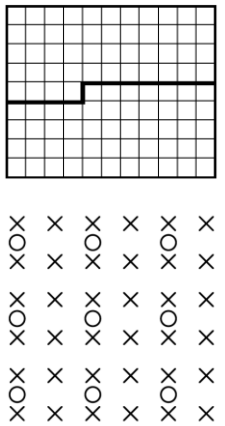
- 4:2:0 8-bit IP coded frame slices with Cavlc
  - Macroblocks: I4x4, I16x16, P16, P8x8, P8x8ref0, PCM
- Does not support avc tools
  - B-slices, 10-bit, 4:2:2, cabac, interlace, fields, weighted pred, 8x8 transforms, scaling lists, slice groups, arbitrary slice order, redundant pics, Mvc, Svc
- Profile Limits
  - PCM pels > 0
  - Level_prefix <= 15 → |coeffs| <= [-2063…2063],
    - Be cautious with $Qp_y < 10$, or $Qp_c < 4$
  - Profile Level <= 5.1→ 2160p30, 1080p120
  - Bitrate <= 288 Mbits

WebVC

Detailed Summary

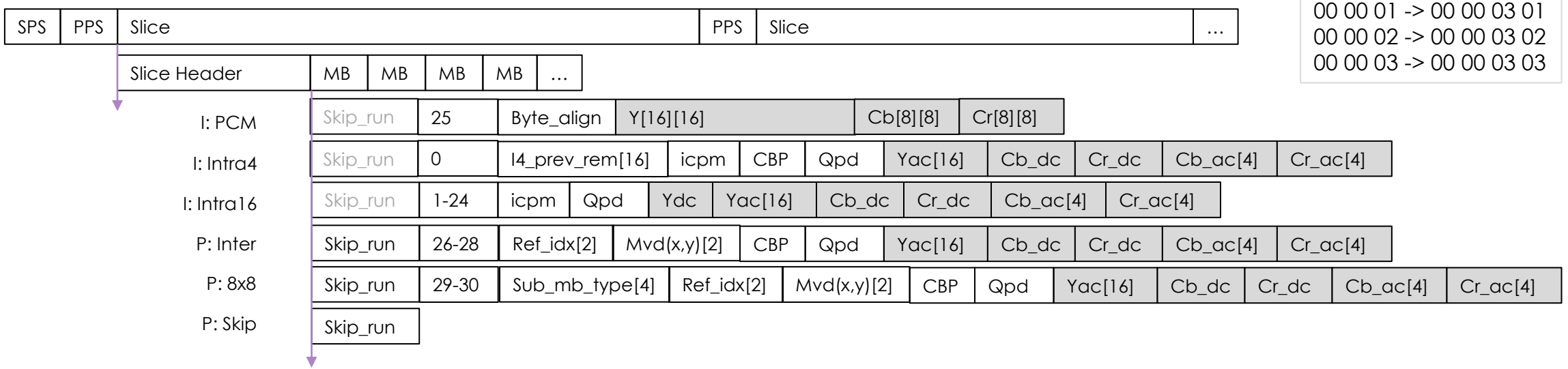
1. Pictures & Streams

- Pictures are composed of a raster of 16x16 macroblocks which may be divided up into slices. A raw 4:2:0 macroblock is 384 bytes
- Frames of video are encoded as a bit stream
- Bitstreams are rate limited via an HRD model parameterized by a byte rate and buffer size and assuming instantaneous decode, without buffer under/overflow.
- Bitstreams are segmented into nals by start codes with nal type signaled by 1st byte in nal.
  - SPS describes stream info (width, height, …), PPS describes picture info (Qp, …)
  - Emulation prevention 03 byte inserted to prevent accidental start code detection
- Slices contain a slice header followed by the encoded macroblocks



SLC: 00 00 01 x1
IDR: 00 00 01 x5
PPS: 00 00 00 01 x8
SPS: 00 00 00 01 x9

00 00 00 -> 00 00 03 00
00 00 01 -> 00 00 03 01
00 00 02 -> 00 00 03 02
00 00 03 -> 00 00 03 03



2. VLC Coding

- Bytes are parsed msb first (Big endian) into bits. The stream remains byte aligned at Nal boundaries using rbsp_tailing)bits(), as well as during PCM nals by using a byte_align() process.
- Variable length coding is employed, like Huffman, to use fewer bits for high probability smaller values. Exp-Golumb coded ue(v) variable length values are used extensively.
- Also binary coded unsigned integers u(n) with flat distribution are used
- Additional signed versions se(v), i(n) and mapping via table lookup me(v) is done with codec_block_pattern in order to optimize expected patters for Intra4x4 vs Inter.

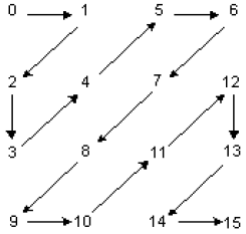
u(3)	i(3)	Bitstring
0	0	000
1	1	001
2	2	010
3	3	011
4	-4	100
5	-3	101
6	-2	110
7	-1	111

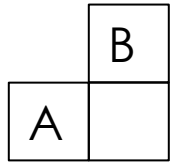
ue(v)	se(v)	Bitstring
0	0	1
1	1	010
2	-1	011
3	2	00100
4	-2	00101
5	3	00110
6	-3	00111
7	4	0001000
8	-4	0001001
9	5	0001010

pic_parameter_set_rbsp( ) {	Descriptor
pic_parameter_set_id	ue(v)
seq_parameter_set_id	ue(v)
entropy_coding_mode_flag /*equal to zero*/	u(1)
bottom_field_pic_order_in_frame_present_flag	u(1)
num_slice_groups_minus1 /*equal to zero*/	ue(v)
num_ref_idx_l0_default_active_minus1	ue(v)
num_ref_idx_l1_default_active_minus1	ue(v)
weighted_pred_flag /* = 0 */	u(1)
weighted_bipred_idc /* = 0 */	u(2)
pic_init_qp_minus26 /* relative to 26 */	se(v)
pic_init_qs_minus26 /* relative to 26 */	se(v)
chroma_qp_index_offset	se(v)
deblocking_filter_control_present_flag	u(1)
constrained_intra_pred_flag	u(1)
redundant_pic_cnt_present_flag /* equal to zero*/	u(1)
rbsp_trailing_bits( )	
}	

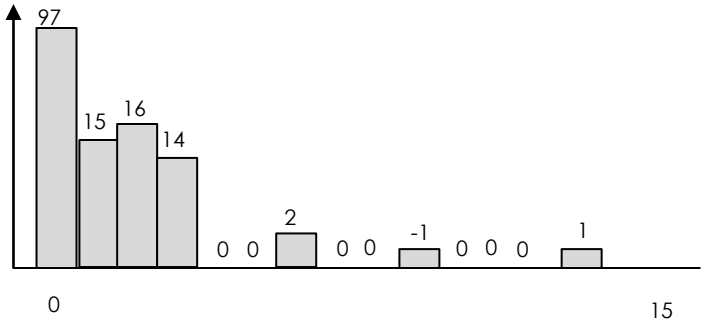
3. CAVLC Coding

- Context Adaptive coding optimizes bitcost by changing the VLC coding table to statistically minimize the bitcost given context information.
- Used to encode transform blocks, assuming DC is largest, and with the zig zag order the coefficients decrease in magnitude with sparse 1's at the tail.
  - coeff_level: transform block is coded from high freq coeff -> dc. A state variable suffix_length is init to 0, and incremented each time a significant coeff is coded. The coding consists of a leading prefix code with 0 to 15 zeros (0) followed by a one (1). The suffix consists of 0 to 12 binary bits
  - Coeff_token: merged TotalCoeffs, as well as Trailing 1's. VLC used is modified based on the sum of the similar neighbours (A, B) non-zero coefficients.
  - Total_zeros: coded modified based on num of zero's possible, given the TotalCoeffs
  - Run_before: run length coding of the zero's between coeffs.





Residual_block() :	Coeff_token	T1s_sign[0-3]	Coeff_Level[0-16]	Total_zeros	Run_before[0-7]
Example :	(7, 2)	0, 1	0, 13, 15, 14, 96	7	0, 0, 0, 2, 2



Suffix_length = 0

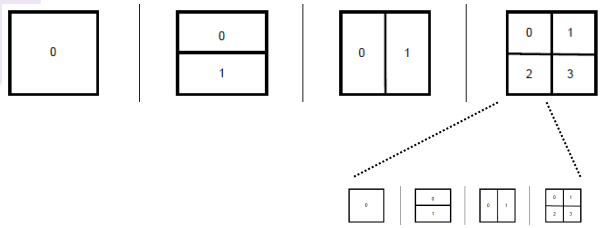
0	1
	01
	001
	0001
	00001
	000001
	0000001
	00000001
	000000001
	0000000001
	00000000001
	000000000001
	0000000000001
	00000000000001
	000000000000001xxxx
15	000000000000001xxxxxxxxxxxx

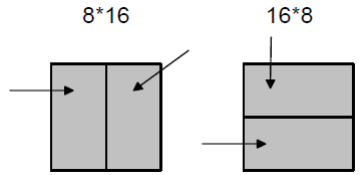
Suffix_length = 6

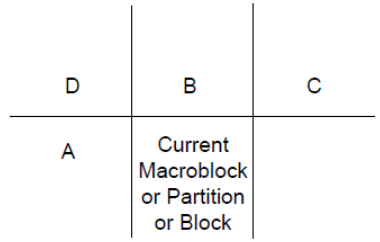
1xxxxxx
01xxxxxx
001xxxxxx
0001xxxxxx
00001xxxxxx
000001xxxxxx
0000001xxxxxx
00000001xxxxxx
000000001xxxxxx
0000000001xxxxxx
00000000001xxxxxx
000000000001xxxxxx
0000000000001xxxxxx
00000000000001xxxxxx
000000000000001xxxxxx
0000000000000001xxxxxxxxxxxx

4. Inter Prediction / Motion Vectors

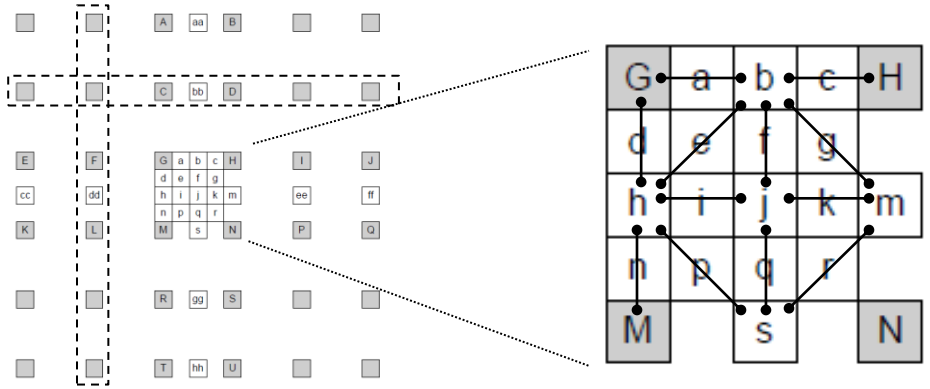
- Macroblock Inter prediction from previously frames is available on P-frames.
  - Maximum of 16 reference frames (if not DPB limited)
- The macroblock is optionally partitioned, and in the case of 8x8 further sub-partitioned.
- Each differential MV has dx, dy and each partition a refIdx. The resolution is ¼ luma pel, with dx = (-2048.00 to 2047.75), and dy = (-512.00 to 511.75)
- The MVd differentials in the stream are predicted for each partition based on neighboring MVs
  - For 8x16 and 16x8 specific predictors are used.
  - All other cases use Median prediction
- A skipped macroblock uses the 16x16 predicted motion vector (
- Luma ¼ pel interpolation is a 2 step process, first half pel interpolation uses a 6 tap filter, both row and column. ¼ pel values are then linear interpolated from neighbours.
- For chroma 1/8th pel a basic Bi-linear interpolation is then used.

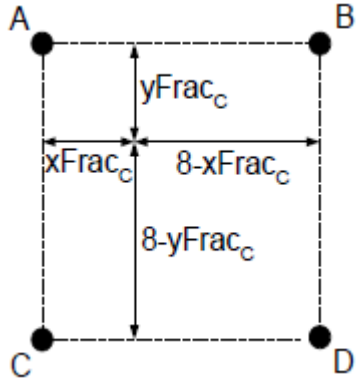






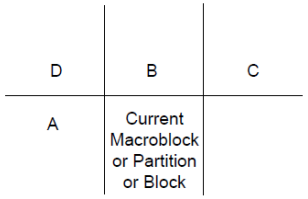
mvp = Median(A, B, C)

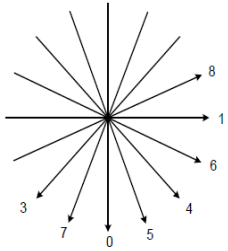




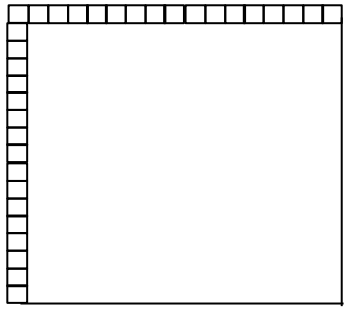
5. Intra Prediction

- PCM: raw 384 pels/bytes, no prediction, no transform.
- Intra4 Luma has 9 modes: 8 directional + Dc
  - The 16 intra4 modes are coded each with a prev flag, and optional 3 bit rem code. If prev set, then the min(A, B) neighbor intra4 mode is used
  - Reconstructed pels are used for internal 4x4 blocks. The order in which the blocks are processed is specified. This constrains the order of processing verses other mb types.
  - Each 4x4 block is predicted by 13 pels, as available, including pels from the upper right macroblock
- Intra16 Luma has 4 modes: H, V, Dc, Planar
  - Predicted as a 16x16 block from reconstructed 33 neighbor pels
- Intra Chroma also has 4 modes: H, V, Dc, Planar
  - predicted as an 8x8 block for each of Cb and Cr. Predicted from 17 neighbor pels each.

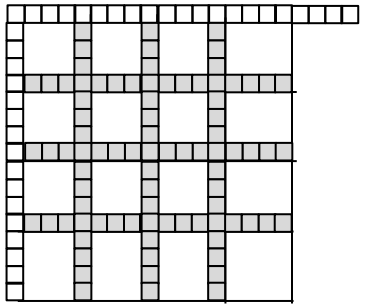




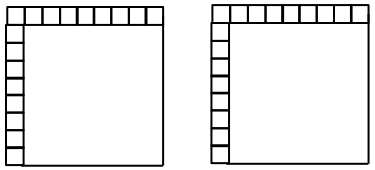
0	1	4	5
2	3	6	7
8	9	12	13
10	11	14	15



Intra16x16



Intra4x4



Cb

Cr

Intra Chroma

6. Quantization & Transform

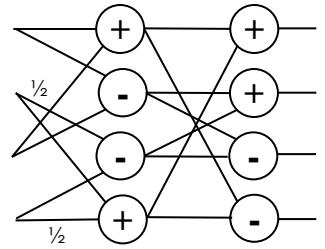
- inverse quantization is controlled by qP, doubling in scale every 6 qP steps. A non uniform LevelScale Matrix is selected based on qP%6. This also serves as part of the DCT operation which is completed by the integer transform.
- A 4x4 transform defined as an integer butterfly 2 stage (with shifts). It is applied on the 4 rows and then the 4 columns. Only 16-bit math need be used for the intermediate and final values.
- For intra16x16 and chroma, a transform is applied to the DC coefficients, which involves only adds or subtracts (+/-1 coeffs).
- A final rounding and right shift gives us the residual.

$$d_{ij} = (c_{ij} * \text{LevelScale4x4}(qP\%6,i,j))<<(qP / 6 - 4)$$

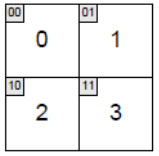
$$v = \begin{bmatrix} 10 & 16 & 13 \\ 11 & 18 & 14 \\ 13 & 20 & 16 \\ 14 & 23 & 18 \\ 16 & 25 & 20 \\ 18 & 29 & 23 \end{bmatrix}.$$

LevelScale4x4(0,i,j)

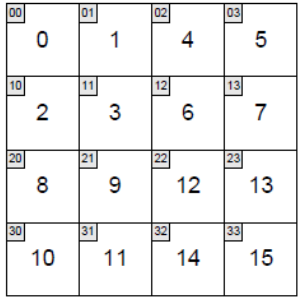
10	13	10	13
13	16	13	16
10	13	10	13
13	16	13	16



$$r_{ij} = (h_{ij} + 2^5) >> 6 \quad \text{with} \quad i, j = 0..3$$



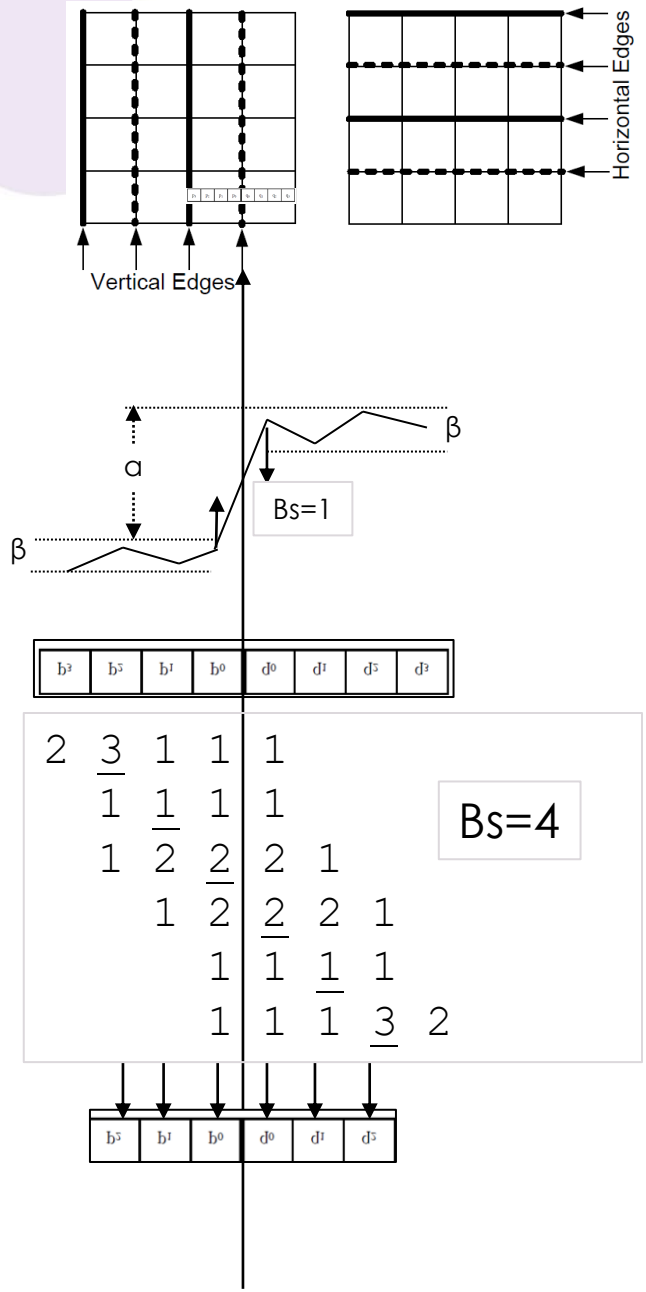
$$f = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} * \begin{bmatrix} c_{00} & c_{01} \\ c_{10} & c_{11} \end{bmatrix} * \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$



$$f = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} * \begin{bmatrix} c_{00} & c_{01} & c_{02} & c_{03} \\ c_{10} & c_{11} & c_{12} & c_{13} \\ c_{20} & c_{21} & c_{22} & c_{23} \\ c_{30} & c_{31} & c_{32} & c_{33} \end{bmatrix} * \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix}.$$

7. in-loop Deblocking

- Deblocking of reconstructed pels helps to hide reconstruction artifacts. Deblocking is defined as being done after decoding a frame, before it is referenced. It is done traversing the frame on a macroblock basis.
- Deblocking is defined as traversal of 4x4 block edges. Chroma is only filtered on 4x4 edges with strengths inherited from luma. PPS and slice level deblocking controls (slice edges), as well as PCM data is never filtered.
- Block strength Bs=4 for intra MB edge, Bs=3 for intra internal edge, Bs=2 if either side has coeffs, Bs=1 if different refs or more than slightly different MVs, otherwise Bs=0 (none).
- Filtering parameters (a,β) are derived from qPavg and slice parameters. For each pel filter the inputs are examined to determine if these pels are consistent with an edge. No deblocking for Qp < 16 (unless slice offset).
- Filtering is recursive in that the pels are modified in place and will be used as inputs to following filters. Filter functions take Qp into account.
- After deblocking a MB, the right and bottom edges are still fully or partially filtered so some care is needed



Hardware

Open Source

Open Source Hardware

- Open source hardware has been historically disappointing to me. Consisting of random libraries of RTL code, sample code, published schematics, or open specifications. It does not give me free and complete useful solutions like open source software.
- Cloud FPGA (field programmable gate arrays) introduced in the last couple of years by multiple vendors have changed that.
  - Low cost of entry: tools can be rented for 25¢/hr and boards with huge $$$$ fpga's for $1.65/hr (billing per second).
  - This enable open source hardware to be built (synthesize, place, route) and run for under $5.
- A sub 1mS latency WebVC hardware 1080p video encoder would be a useful open source project that takes real advantage of FPGAs.
- Support accessible cloud targets and build environments to enable experimenters to modify, build, and run the project without investment.

Latency for 1080p60

- Macroblock time: 2 uSec
- Camera Latency: rolling shutter to minimize latency
- Raster to Macroblock: delay to start of MB processing
  - Some cameras have special output modes, so can be eliminated
- Processing latency:
  - 0.24ms (120 MBs) down to 0.024ms (12 MBs)
- Communication latency:
  - Additive to latency, not under control, assume 0.
- Decode Latency:
  - HRD model uses zero, and a real decoder can output immediately.
  - Practically <= 0.24ms to decode slice row.
- Rasterization:
  - Double counting if camera has a raster output?
  - 0.24ms to raster a MB row.

Camera	Raster To Macroblock	Macroblock Encode Pipeline	Comm Delay	Decode	Macroblock To Raster
					
→ 0	.24 ms	0.024ms to 0.240 ms	→ 0	.24 ms	.24 ms

Core Architecture Requirements

- 288 Mbit, ¼ mSec, 1080p60, 4:2:0, 8-bit, WebVC encoder
- Programable bitrate and pic width, height: (720p, 1080p, 4K, …)
- Macroblock Picture Data Input: 512 bit MB stream (6 cycles/mb)
- Macroblock Rate: > 500 Khz
- Macroblock Pipeline Depth: 24 – 256 macroblocks
- Codec type: P frame, Dual pass encoder, intra col refresh, row slice
- Per Macroblock Quality *Target* Input: 0 to 51, rate control built-in
- Output bitstream: Byte stream / DMA ring buffer.
- Multi-port memory interface, 40-bit address, 512-bit read/write
- 32 bit control register read/write bus.
- Single 250 Mhz clock input.
- Programmable Interrupt
- Copyright/left licensing terms: tbd