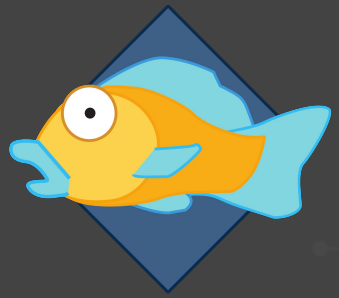


AV1: Nits, Nitpicks and Shortcomings [Things we should fix for AV2]

Nathan Egge <negge@mozilla.com>
AOM Research Symposium - October 21, 2019
Slides: https://xiph.org/~negge/AOM2019.pdf

Who am I?

- Head of Codec Engineering at Mozilla
  - Rust AV1 Encoder (rav1e)
  - Dav1d is an AV1 Decoder (dav1d)
- Co-author on the AV1 format, worked on Daala before that
- Organized and Co-Hosted Big Apple Video Conference with Vimeo
- Member of various non-profits: Xiph.Org, VideoLAN Asso
- Generally advocate for royalty-free media standards

[1] https://bigapple.video

Abstract

The Alliance for Open Media completed the AV1 specification in a record 30 months time. This short development cycle meant that many coding tools were only ever implemented a single time and evaluated under limited test conditions. Since publishing the 1.0.0 specification there are now many independent implementations of AV1 being used across a much broader set of operating points.

This talk will look at a few shortcomings in the AV1 format discovered by implementers and the impact they have on both coding performance and execution time. Some were known during development but for various reasons those experiments did not make it into AV1. Where possible, potential modifications will be provided for use in a future video coding standard.

AV1 Standardization Timeline

AOMedia formed 2015-9-1

VP10 baseline 2016-1-19

AOM F2F #1 2017-2-14

AOM F2F #2 2017-8-22

AV1 launch 2018-3-28

libaom tag 1.0.0 2018-6-25

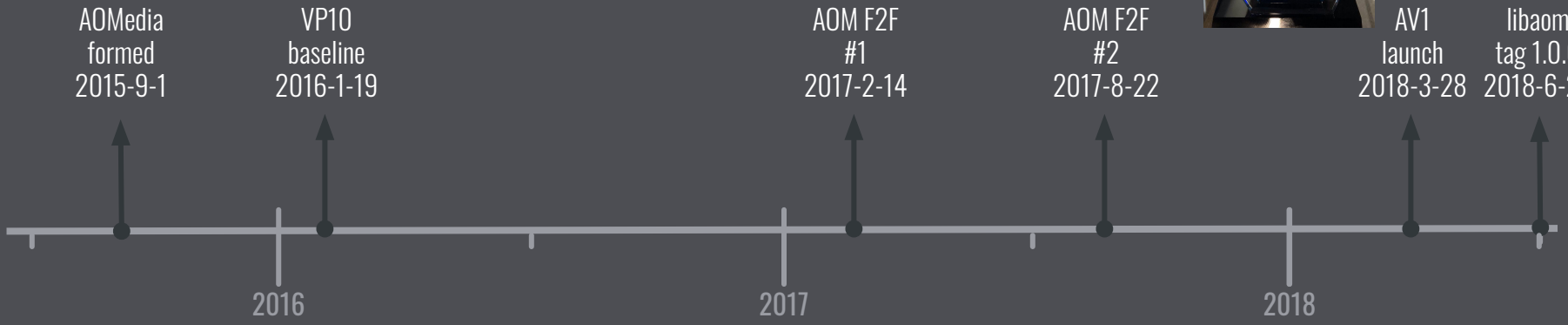
2016

2017

2018

AV1 Format Standardization Effort (~30 months)

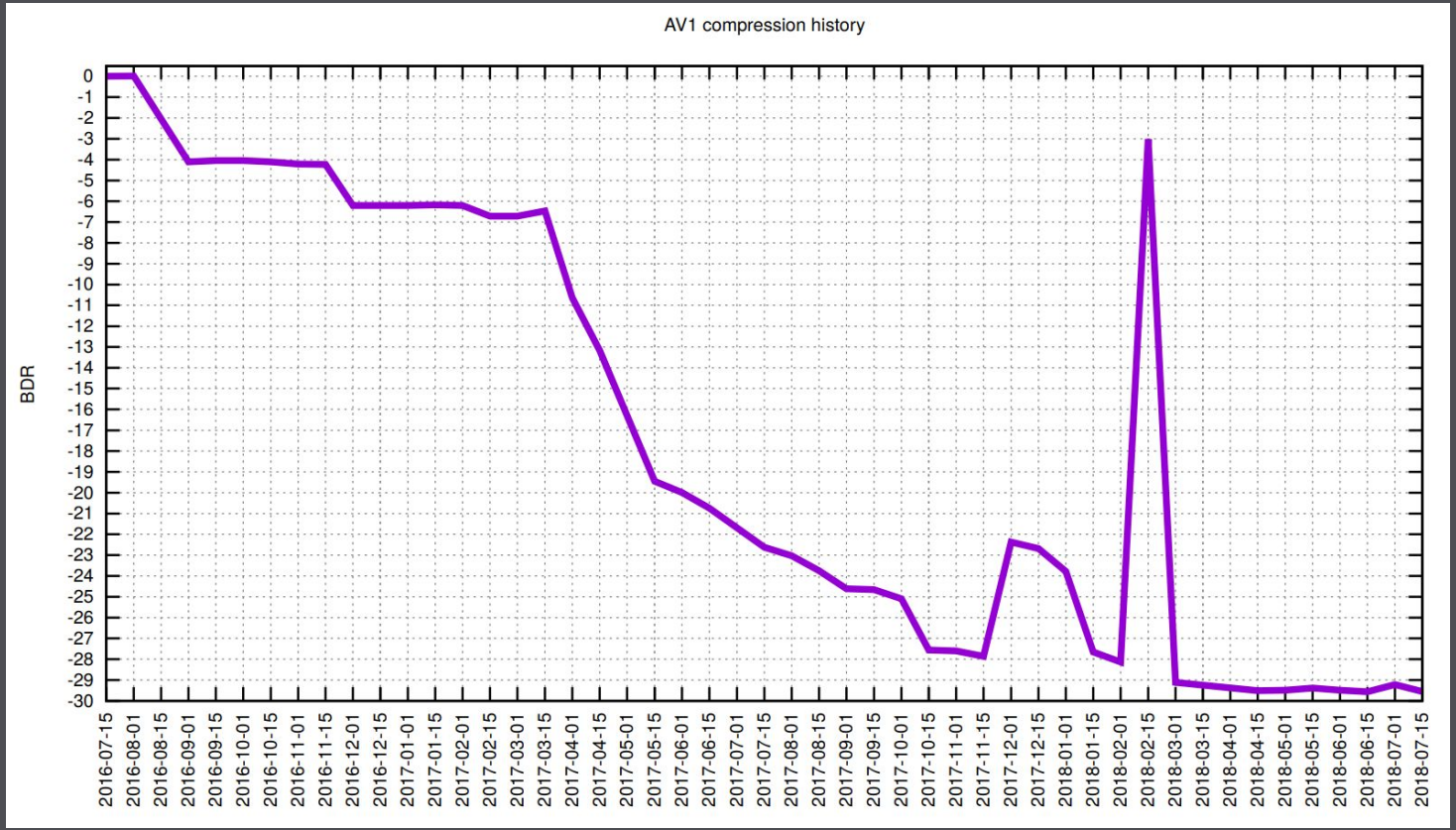
AV1 Standardization Timeline



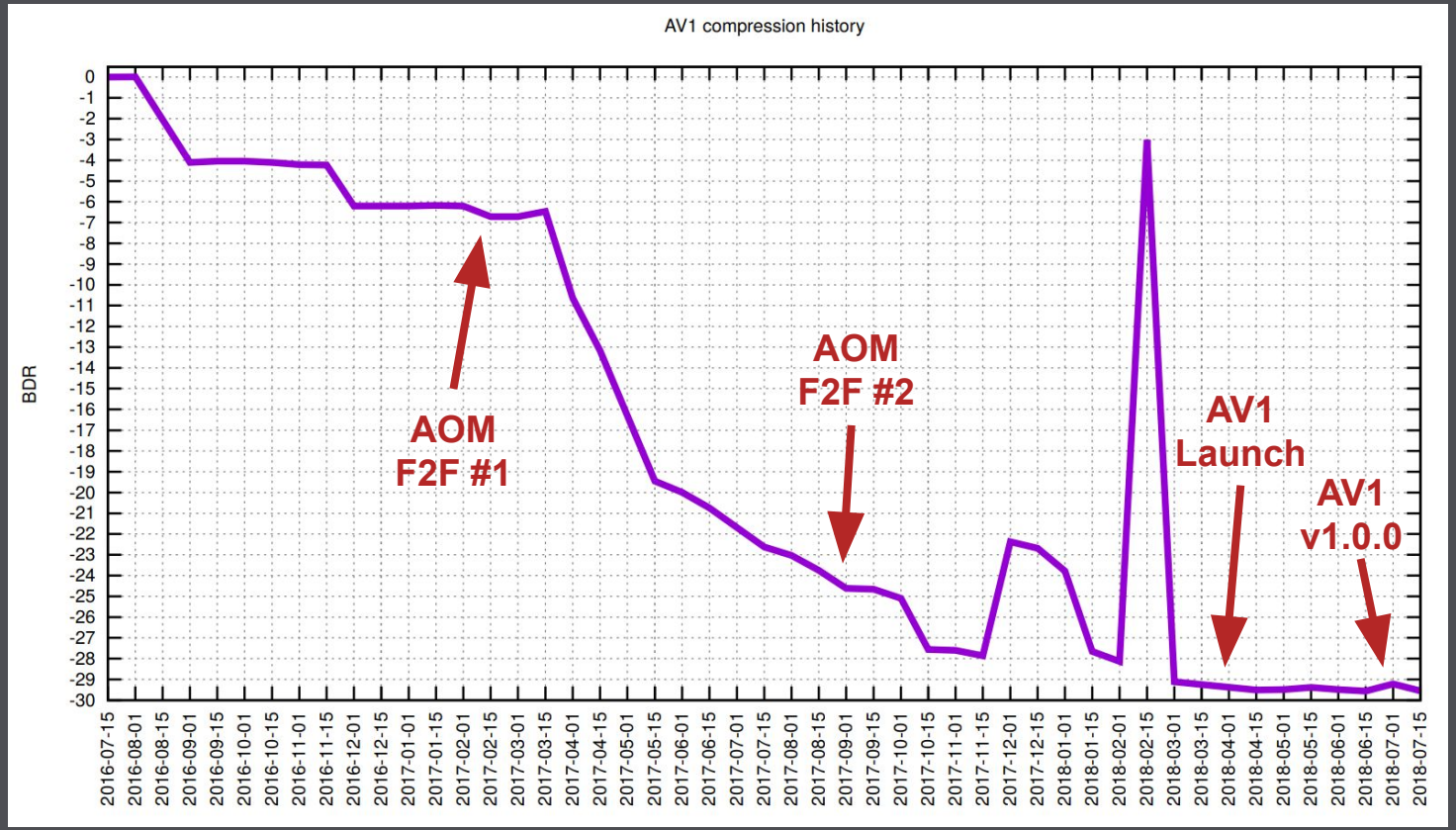
Combination of 3 mature code bases:

- VP10 (nextgenv2 forked 2015-8-25)
- Daala (initial commit 2010-Oct-13)
- Thor (initial commit 2015-7-15)

AV1 (libaom) Compression History



AV1 (libaom) Compression History



Challenges to a short timeline

- Coding tool “success” measured as experiment adoption
  - Based primarily on objective metrics in limited test conditions
- Less relevant evaluation criteria include:
  - Quality of implementation, integration with other tools, peer review
- No incentive to fix or simplify implementation
  - Refactoring hard in the presence of rapidly changing code
  - Required to keep WIP experiments behind flags running properly
- Lots of tools enabled only at the end of the cycle

Result

- Many issues only being found now through in-depth peer review by implementers

Loop Filtering Example #1: The Value is Always 95

- Self-Guided Restoration filter uses prediction to encode its filter coefficients more efficiently
  - Several modes make use of only the second filter coefficient
- AV1 specification (section `5.11.58`) defines the predictor (assigned to v below) as:

```
[i == 1]
min = Sgrproj_Xqd_Min[i]
max = Sgrproj_Xqd_Max[i]
[...]
v = Clip3( min, max, (1 << SGRPROJ_PRJ_BITS) - RefSgrXqd[ plane ][ 0 ] )
```

- Careful inspection of the math involved shows this predictor is useless
  - Values between 224 and 96 clipped to a range -32 and 95 are always equal 95

[1] https://code.videolan.org/videolan/dav1d/commit/48d9c683

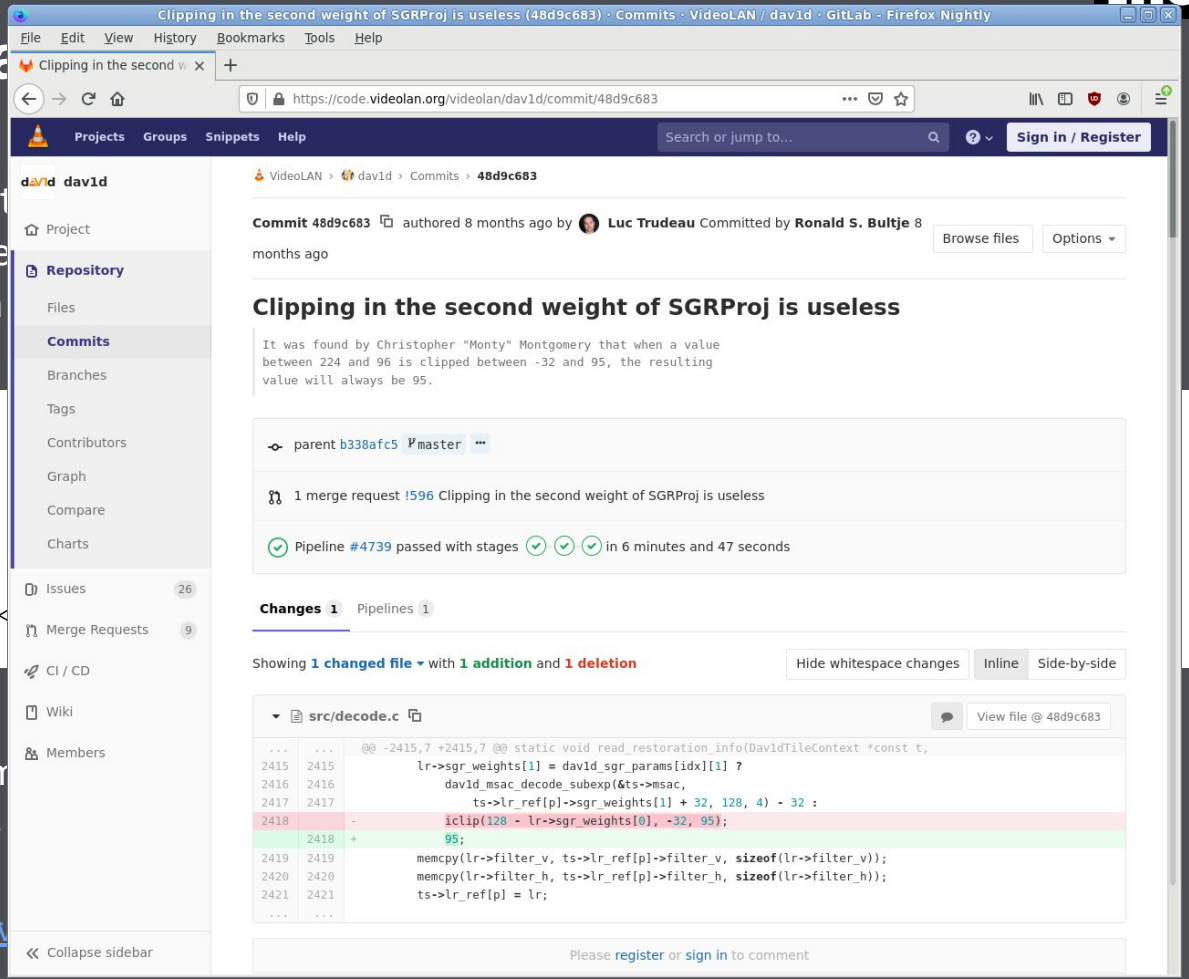
Loop Filtering Exa

- Self-Guided Restoration filt
  - Several modes make
- AV1 specification (section

```
[i == 1]
min = Sgrproj_Xqd_Min[i]
max = Sgrproj_Xqd_Max[i]
[...]
v = Clip3( min, max, (1 <<
```

- Careful inspection of the m
  - Values between 224

[1] https://code.videolan.org/v



Loop Filtering Example #2: Deblocking Simplification

- Deblocking filter in AV1 is a mature contribution from the VPx series of codecs
  - Some form in use for more than a decade (expected it to be solved)
- Analysis showed horizontal + vertical can be optimized individually
  - Separable solution nearly equivalent to “brute force” search

Video	PSNR	PSNR HVS	SSIM	CIEDE 2000	PSNR Cb	PSNR Cr	APSNR	APSNR Cb	APSNR Cr	MS SSIM
Average	-1.83	-0.76	-1.03	-1.51	-1.55	-1.69	-1.83	-1.55	-1.70	-0.74
1080p	-2.89	-1.28	-1.61	-2.19	-2.04	-1.86	-2.87	-2.04	-1.86	-1.31
1080p-screen	-0.60	-0.36	-0.28	-0.47	-0.84	-0.63	-0.60	-0.85	-0.64	-0.22
360p	-1.03	-0.36	-0.77	-1.32	-1.29	-2.15	-1.07	-1.30	-2.23	-0.35
720p	-1.52	-0.48	-0.74	-1.15	-1.36	-1.53	-1.52	-1.36	-1.53	-0.45

- New algorithm gave good gains in rav1e (even without all coding tools)

[1] https://beta.arewecompressedyet.com/?job=deblock-exhaustive%402018-09-18T06%3A43%3A05.711Z&job=deblock-separable%402018-09-18T08%3A22%3A05.399Z
[2] https://beta.arewecompressedyet.com/?job=deblock-baseline%402018-09-18T04%3A59%3A53.700Z&job=deblock-separable%402018-09-18T08%3A22%3A05.399Z

Loop Filtering Example #3: Clipping and Padding

- Padding, cropping and edge extension conventions in a codec can be somewhat arbitrary.
  - There may be a reason to choose one option over another, but pick one and stick with it
  - Exactly what three loop filters did, except each chose a *different* arbitrary convention!

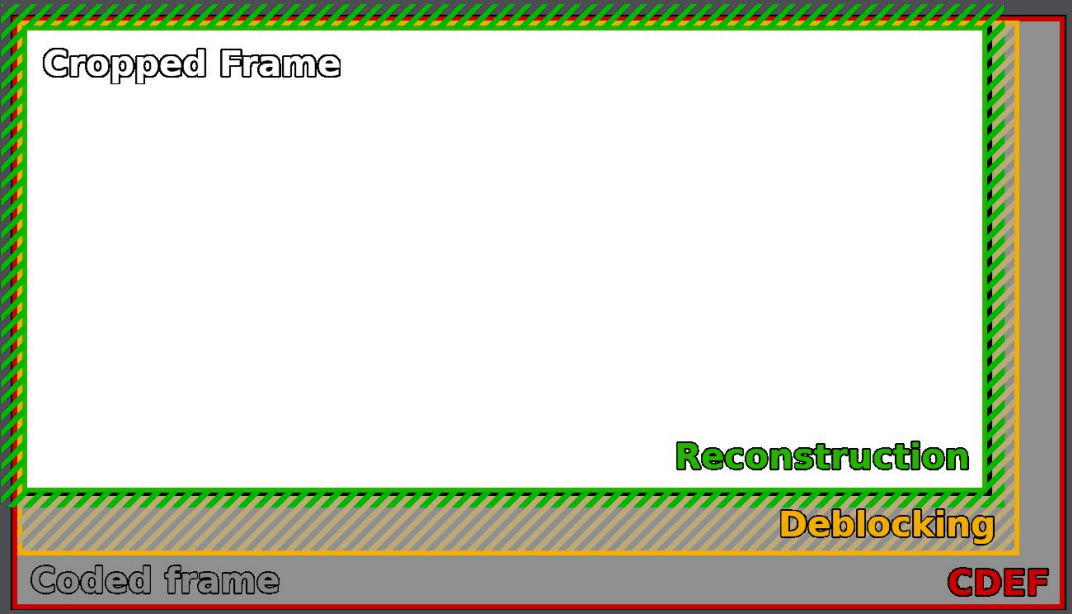
- **Deblocking** is clipped to the luma crop frame rounded up to a multiple of 4 in both directions
  - Deblocking can extend into padding at the right and bottom, but padding itself is not filtered.

- **CDEF** operates across the entire coded frame including all padding.
  - Where the CDEF kernel extends past the coded frame edge, the kernel is clipped.

- **Self-Guided Restoration** filter clips to the unrounded crop frame (plus a few other restrictions).
  - When restoration filter kernel extends past the bounded area, the area is edge-extended.

No attempt at making these consistent during development!

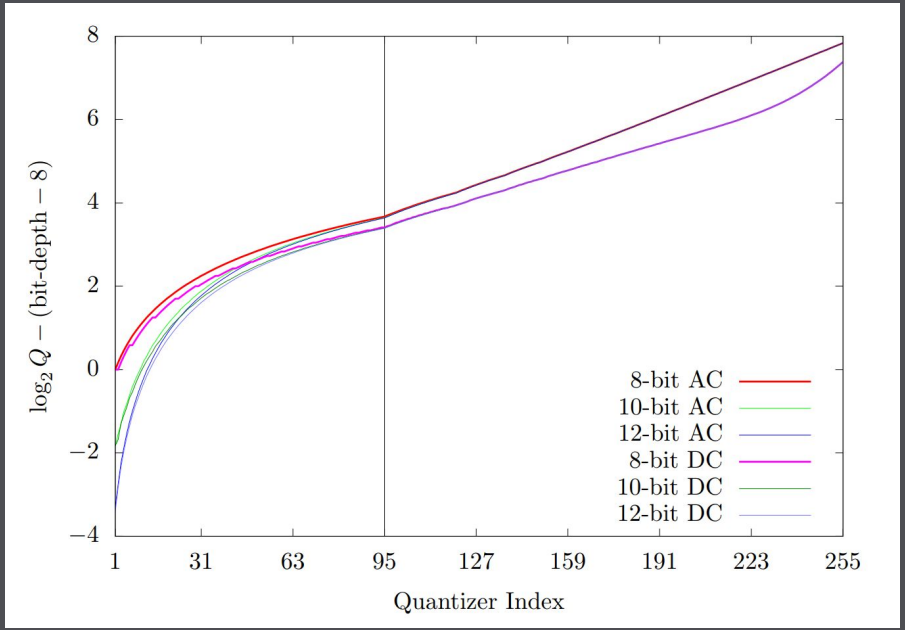
Loop Filtering Example #3: Clipping and Padding (Cont)



- There's no theoretical impact to performance, however...
  - Developers will notice (and often trip over) a lack of consistent convention
  - The three filters are intended to work as a single, pipelined unit

Quantizer Selection Example

- Consider a Frame with six quantizer indices
  - (DC, AC) x (Y, U, V)
- Index maps to a non-linear quantizer table

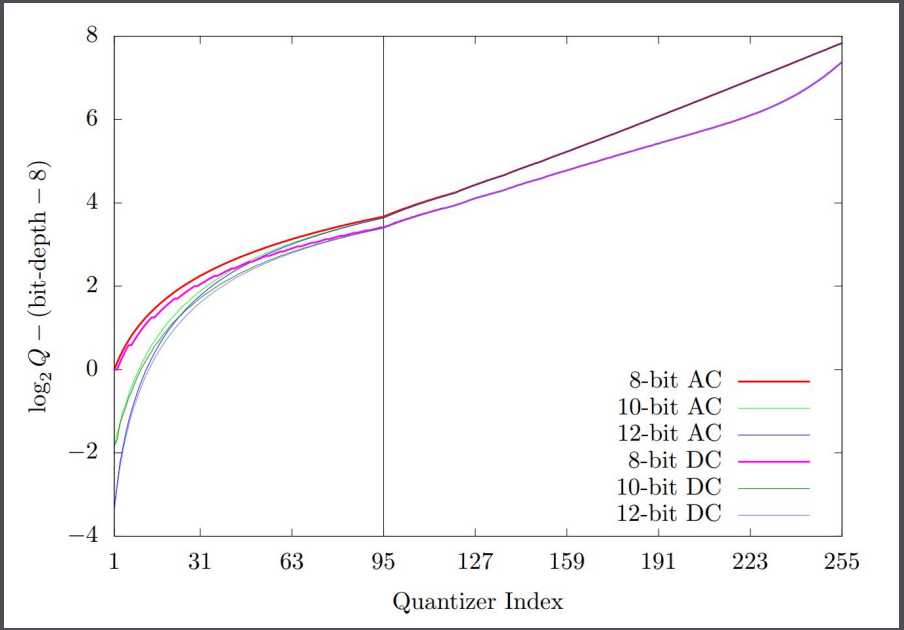


Quantizer Selection Example

- Consider a Frame with six quantizer indices
  - (DC, AC) x (Y, U, V)
- Index maps to a non-linear quantizer table

- Simple example: Flat Quantization
  - Signal indices so quantizer matches

- What happens when we want to boost a block?



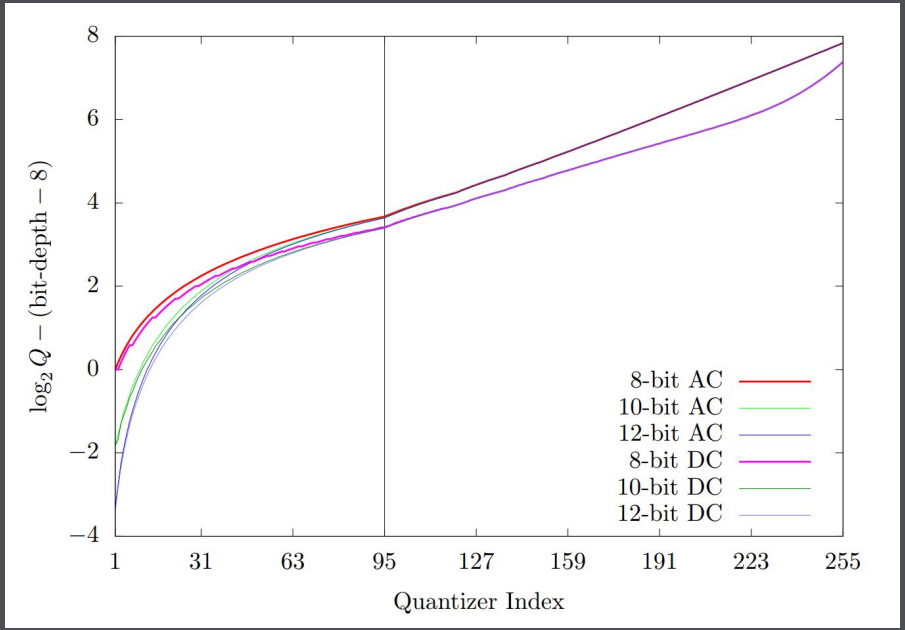
Quantizer Selection Example

- Consider a Frame with six quantizer indices
  - (DC, AC) x (Y, U, V)
- Index maps to a non-linear quantizer table

- Simple example: Flat Quantization
  - Signal indices so quantizer matches

- What happens when we want to boost a block?

- Segmentation lets you signal a delta index, e.g.,
  - $Q(Y_{DC} + \Delta i)$, $Q(U_{DC} + \Delta i)$, etc



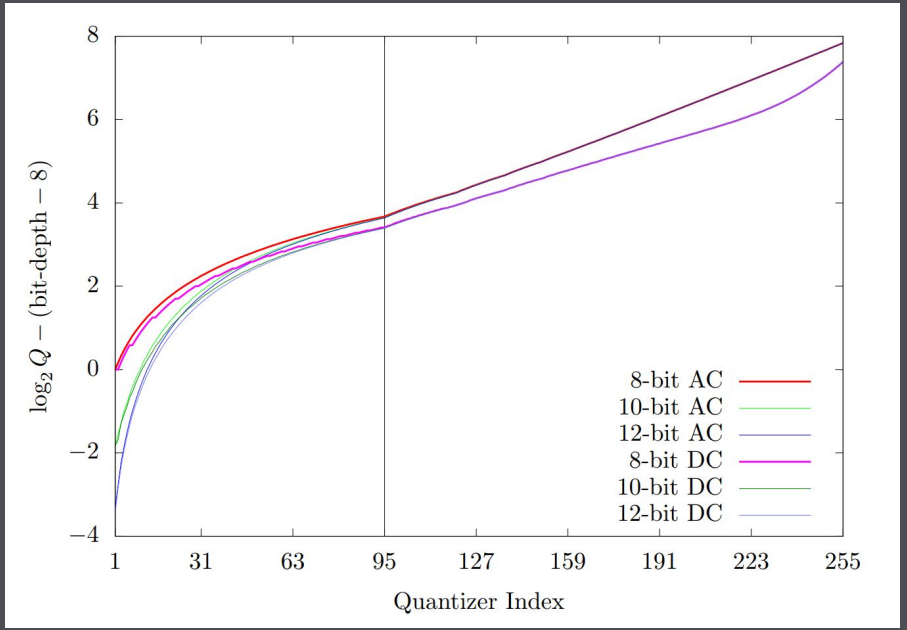
Quantizer Selection Example

- Consider a Frame with six quantizer indices
  - (DC, AC) x (Y, U, V)
- Index maps to a non-linear quantizer table

- Simple example: Flat Quantization
  - Signal indices so quantizer matches

- What happens when we want to boost a block?

- Segmentation lets you signal a delta index, e.g.,
  - $Q(Y_{DC} + \Delta i)$, $Q(U_{DC} + \Delta i)$, etc



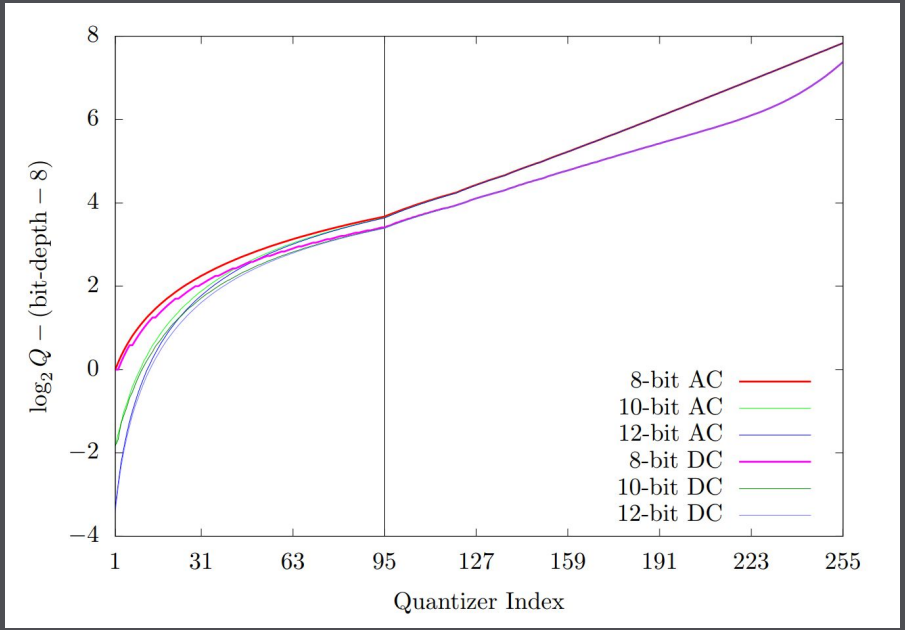
- Problem!
  - Frame may have balanced Chroma v Luma, but non-linear step means balance not maintained

Quantizer Selection Example: Two Solutions

#1 Engineering Solution

- Simply code Δi for each of (DC, AC) x (Y, U, V)
- Segment can set exactly Chroma v Luma balance

- Cost to signal can be made very cheap
  - Only a few bits to keep old behavior
  - Code just like frame indices

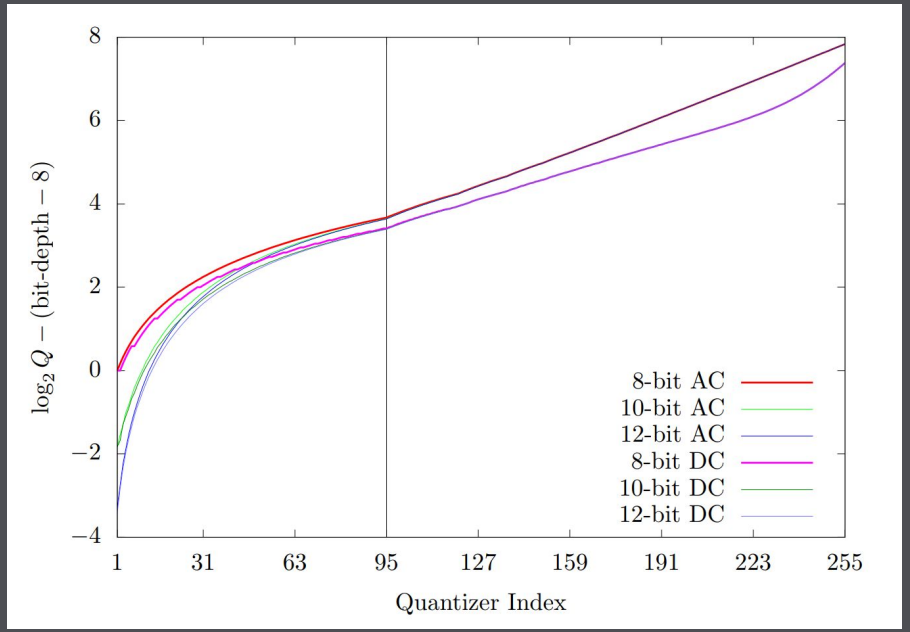


Quantizer Selection Example: Two Solutions

#1 Engineering Solution

- Simply code Δi for each of (DC, AC) x (Y, U, V)
- Segment can set exactly Chroma v Luma balance

- Cost to signal can be made very cheap
  - Only a few bits to keep old behavior
  - Code just like frame indices



#2 Science Solution

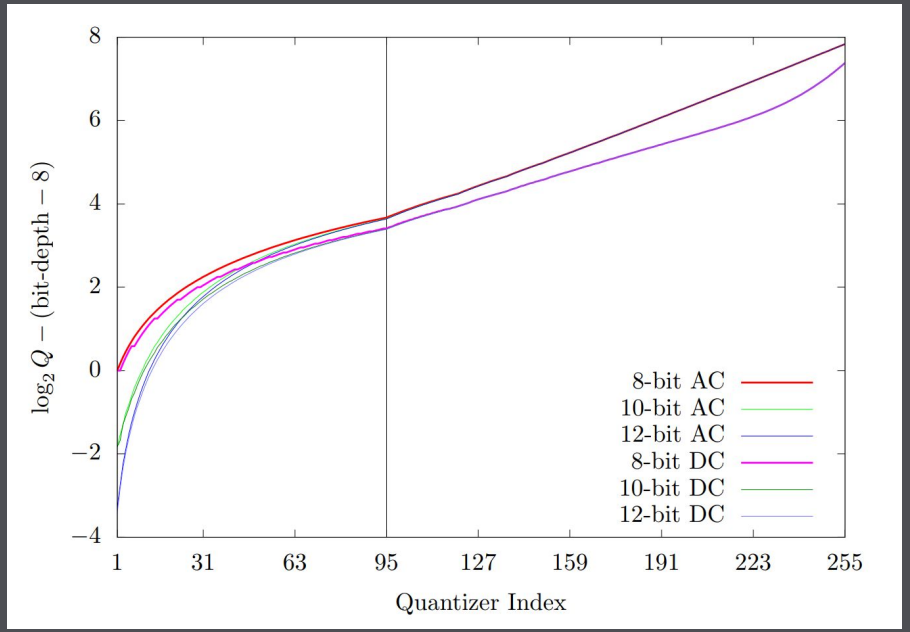
- Do more research to design the right curves (save on coding costs)

Quantizer Selection Example: Two Solutions

#1 Engineering Solution

- Simply code Δi for each of (DC, AC) x (Y, U, V)
- Segment can set exactly Chroma v Luma balance

- Cost to signal can be made very cheap
  - Only a few bits to keep old behavior
  - Code just like frame indices



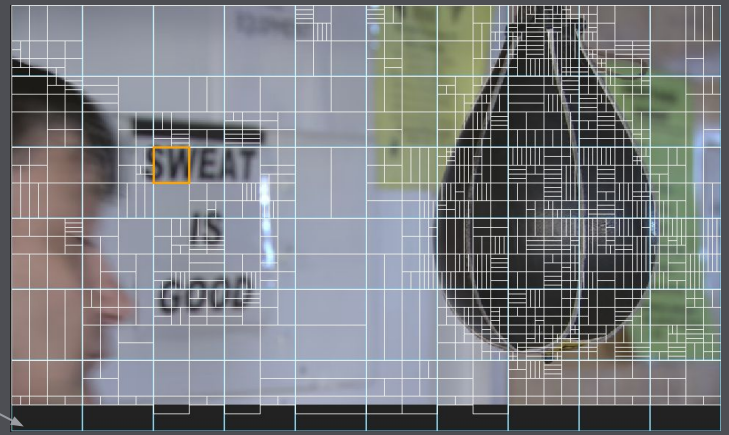
#2 Science Solution

- Do more research to design the right curves (save on coding costs)

  “Nobody looked at this during AV1 development, someone should look at it for AV2.” - Tim

SIMPLE_CROP Experiment (not adopted)

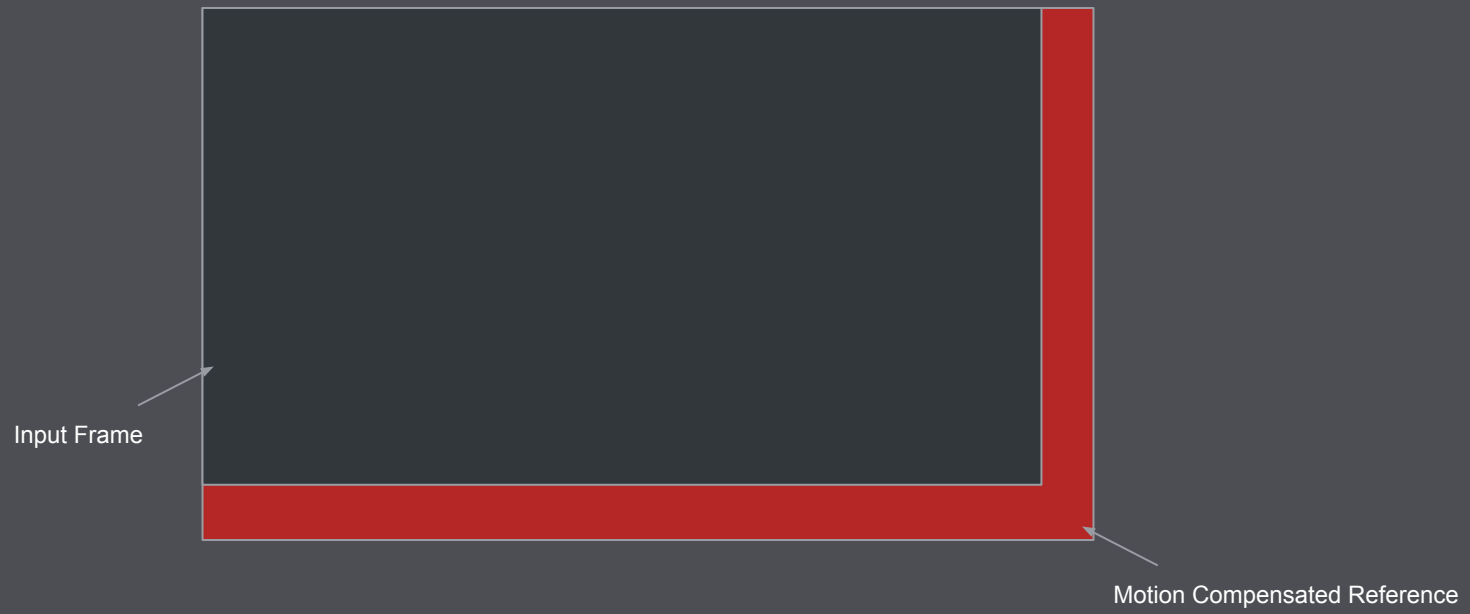
AV1 Split Grid



AV1 Split Grid with Super Block

- In AV1, the decoded width and height are aligned to the nearest multiple of 8 (or 4 for subsampled chroma).
  - Often results in splits being forced at frame boundaries for non-superblock aligned sizes e.g., 144p or 1080p.
- In Daala, the decoded width and height are aligned to the nearest multiple of the superblock size.
  - This avoids having to implement complicated edge case handling for non-superblock aligned sizes.
- To avoid bitrate penalty for the larger area, padded region should be coded as cheaply as possible.
  - Achieved by making the prediction exactly match the reference in padding (coding a residual of zero)
  - When performing RDO, distortion only calculated for visible pixels in these edge blocks.

SIMPLE_CROP Experiment (not adopted)



SIMPLE_CROP Experiment (not adopted)

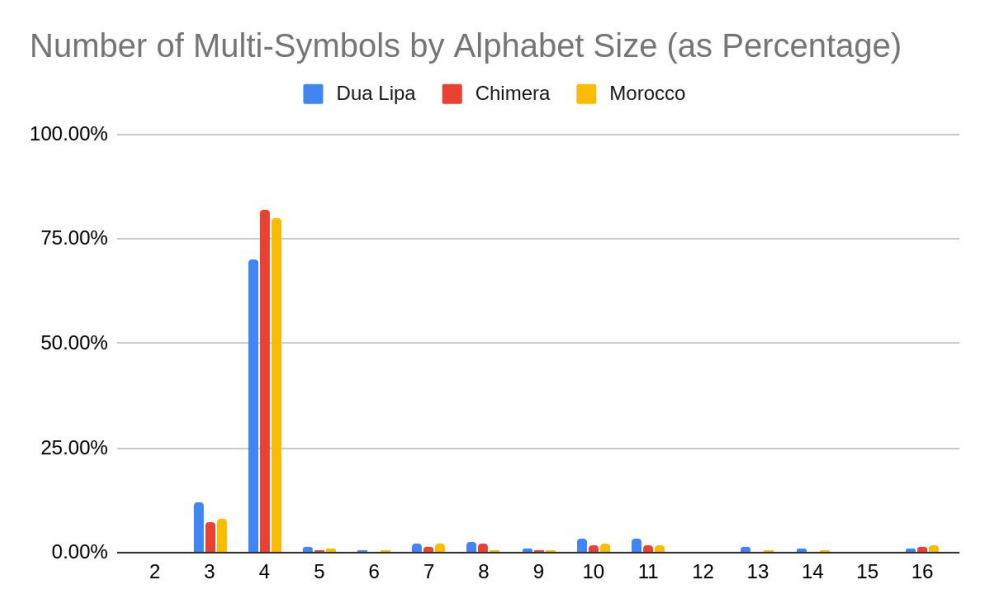
- Steps needed to complete SIMPLE_CROP experiment
  1) Pad MI_BLOCKS to nearest super block
  2) Fix all experiments which make wrong assumptions about MI_GRID size (which this would break)
  3) Make the padded region cheap to code

- Adding (1) was easy, stuck on (2) + (3)
  - Could not keep up with experiments landing
  - Refactor necessary to add prediction block size to function pointers

- Revisiting experiment would mean being able to remove a lot code!!
  - Removes “ragged edge” split logic also present in decoder
  - Special constructions to make probability tables by merging redundant modes

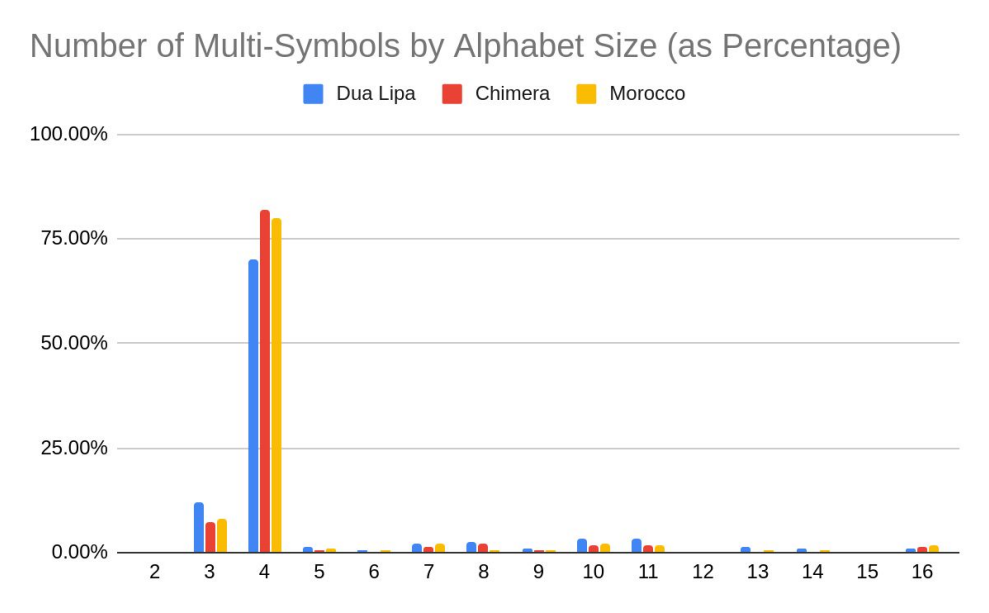
Multi-Symbol Entropy Coder Example

- AV1 added multi-symbol arithmetic coding
  - VP9 boolean trees -> CDFs
  - Maximum alphabet size of 16
- Potential to code up to 4 bits per symbol
  - Format needs to make use of MSAC



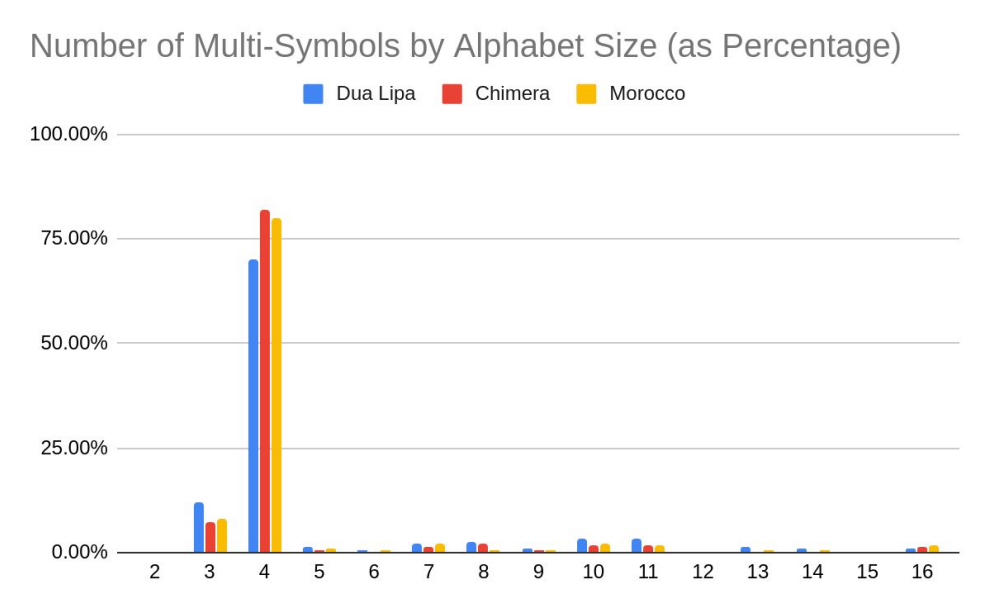
Multi-Symbol Entropy Coder Example

- AV1 added multi-symbol arithmetic coding
  - VP9 boolean trees -> CDFs
  - Maximum alphabet size of 16
- Potential to code up to 4 bits per symbol
  - Format needs to make use of MSAC
- LV_MAP experiment changed coeff coder
  - Added 2017-Feb-24
  - Mostly binary symbols



Multi-Symbol Entropy Coder Example

- AV1 added multi-symbol arithmetic coding
  - VP9 boolean trees -> CDFs
  - Maximum alphabet size of 16
- Potential to code up to 4 bits per symbol
  - Format needs to make use of MSAC
- LV_MAP experiment changed coeff coder
  - Added 2017-Feb-24
  - Mostly binary symbols
- LV_MAP_MULTI experiment uses 4-CDFs
  - Added 2017-Nov-6 [1]



The adapted symbol count is significantly reduced by this experiment.
E.g. for the I-frame of ducks_take_off at cq=12, the number of adapted symbols is reduced from 6.7M to 4.3M.

[1] https://aomedia.googlesource.com/aom/+/1389210

Potential Process Improvements for AV2

- Agree up front on format “launch” criteria, announce it publicly
  - What is a reasonable time between formats?

- Longer engineering review period

- Independent implementations prior to finalization
  - Decoder - based on spec document, not by format authors
  - Encoder - broader set of operating points (interactive, real-time, VOD, etc.)

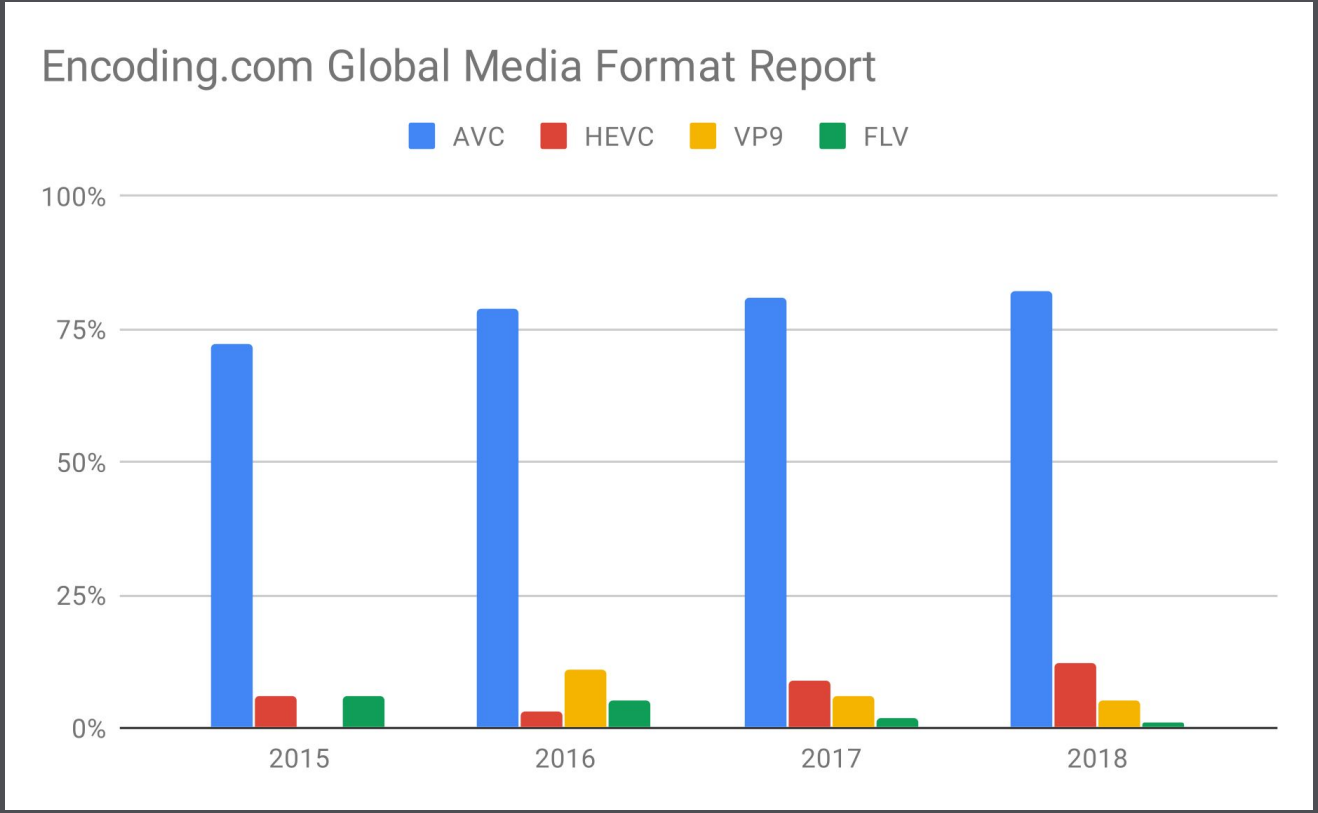
- Improve peer review process
  - Encourage outside experts to participate

- Balance hardware and software implementation concerns

moz://a

Questions?

Significant and Growing H.264 Deployment



	AVC (%)	Ingest (PB)
2014	69	0.95
2015	71	1.45
2016	79	5.9
2017	81	12.3
2018	82	19.3

[1] https://www.encoding.com/resources/