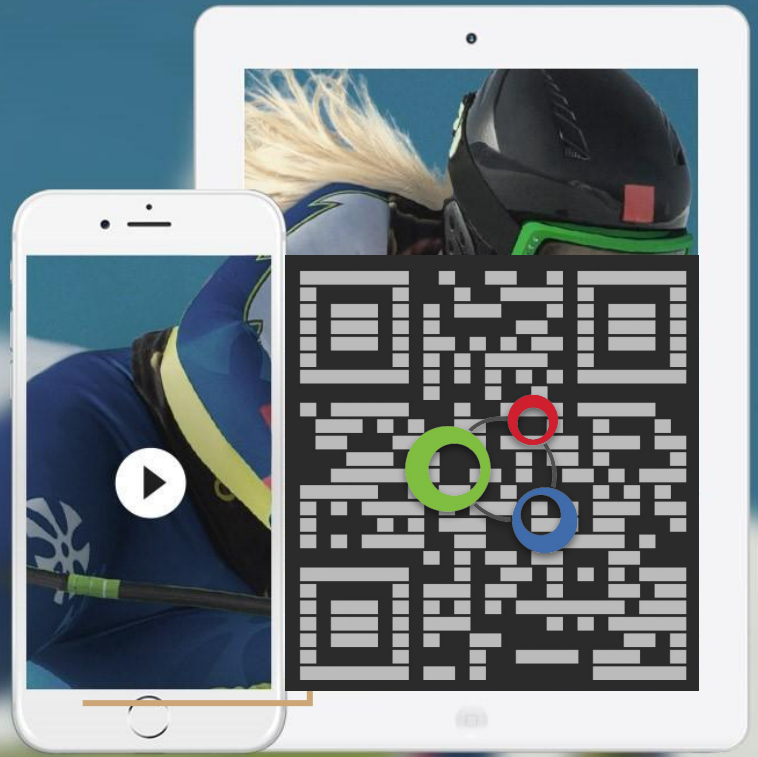
Working with the AV1 Codec

Kevin Staunton-Lambert
Solutions Architect
R&D
@kevleyski



switchmedia
www.switch.tv

Okay, explain video “**codec**” like I’m five…

Motion video is a flick book of photos

A movie length flick book would be tricky to share around and takes up heaps of room. There are also lots and lots of similarities between the photos you could cut out with scissors and discard - just write down what's needed to glue things back

Determining what can be removed and is changing between the pages and when it’s better to just include another photo instead - is the art of the ‘*en**co**der*’

Figuring out how to trick you to having the same flick book back again, without being a huge burden to have done so - is the art of the ‘***dec**oder*’

Grown-up analogy… compressing the photos

Raw video footage is captured digitally as **frames** of picture elements (pixels)

Image frames usually contain large sections of similar brightness - **luma** - and colour information - **chroma**. Chroma split different ways if know blue and red, green can be deduced. Humans can only perceive larger differences in brightness and less so do we notice minor variations in the colour

Mathematical functions called **transforms** - such as DCT Fourier - are used to determine within the luma/chroma data what is similar enough to be removed, aka **quantized** away, and which values are most common and can be referenced instead to use less bits, **entropy encoded** - combined together this removes the **spatial** redundancy.

Grown-up analogy… compressing the flick book

More maths functions called **predictors** are applied to determine what changed between frames over time removing things that didn't move, or, applying **motion vectors** to capture where're moving to, thus removing the **temporal** redundancy

More functions, **filters**, tidy up the resulting mess from all the above and can be used to feedback into the process as to what else to look for, textures etc

Combinations of the above determine the strengths and weakness of a codec

- Colour space and resolution - how rich is the compressed vs original footage
- Visual quality at lower bitrates
- How much processing/battery power/latency needed to encode and decode
- But also, how much you might get screwed over for the best of the above

So, what's all the fuss about **AV1 codec**?

A raw 4K video is big, *bloody big*, like 33 GiBs per minute big *

The **bandwidth** needed to transmit 4K and the **storage** required to hold the media files gets linearly worse as the number of assets and the length of them increases. Also multiple ABR bitrates and quality representations.

4K coders are patent encumbered, companies are squabbling with each other a lot, and no one really knows who owes whom the royalties - some mongrels about

AOMedia Video 1 makes things **smaller** **than all the others** and provides a big getting along shirt by releasing all **intellectual property rights, so royalty-free**!

** YUV4Mpeg raw 24 fps UHD 3840x2160*

But, couldn't we just *pay* for MPEG HEVC/H.265?

You could, but you take a risk and you'd be on your own pretty soon…

The founder and chairman of the MPEG acknowledged the **Alliance** to be the biggest threat to their business model, furthermore stating that:

"*Alliance for Open Media has occupied the void created by MPEG's outdated video compression standard (AVC), absence of competitive [royalty free] standards (IVC) and unusable modern standard (HEVC)… Everybody realises that the old MPEG business model is now broke."*

Vs. "each Licensor, on behalf of itself and successors in interest and assigns, grants Licensee a non-sublicensable, perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable…

No worries, what's **AOMedia**?

After some success with free codecs such as VP9, Daala and Thor the sponsors of those projects realised their split efforts were hindering development and ultimately their market adoption and so together kicked off the initiative for the **A**lliance for **O**pen **Media**

Their goal is to bring together work from anyone interested. Particular noteworthy contributions include the Mozilla/Xipf Daala codec with WebM and VP10 and Cisco Thor (conferencing)

The founding members in September 2015 were *Amazon, Cisco, Google, Intel, Microsoft, Mozilla, and Netflix*. *Apple* only joined Jan 2018

What's been promised

The alliance focuses on delivering a next-generation video format that is:

- Interoperable and open
- Optimized for the web
- Scalable to any modern device at any bandwidth
- Designed with a low computational footprint and optimized for hardware
- Capable of consistent, highest-quality, real-time video delivery
- Flexible for both commercial and non-commercial content, including user-generated content
- Ultimate goal of being 30%+ more bandwidth efficient compared to H.265.

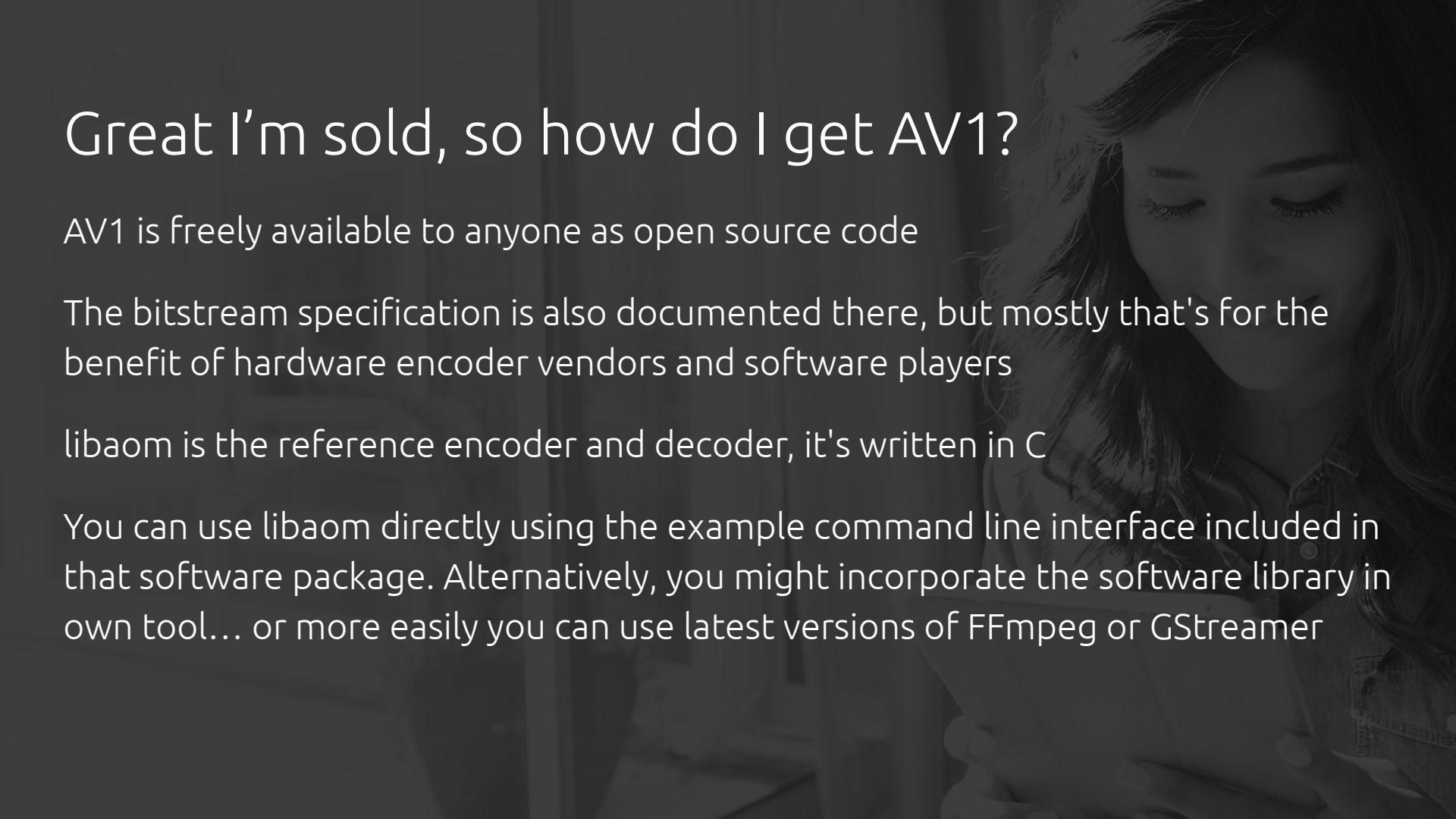
Great I’m sold, so how do I get AV1?

AV1 is freely available to anyone as open source code

The bitstream specification is also documented there, but mostly that's for the benefit of hardware encoder vendors and software players

libaom is the reference encoder and decoder, it's written in C

You can use libaom directly using the example command line interface included in that software package. Alternatively, you might incorporate the software library in own tool… or more easily you can use latest versions of FFmpeg or GStreamer



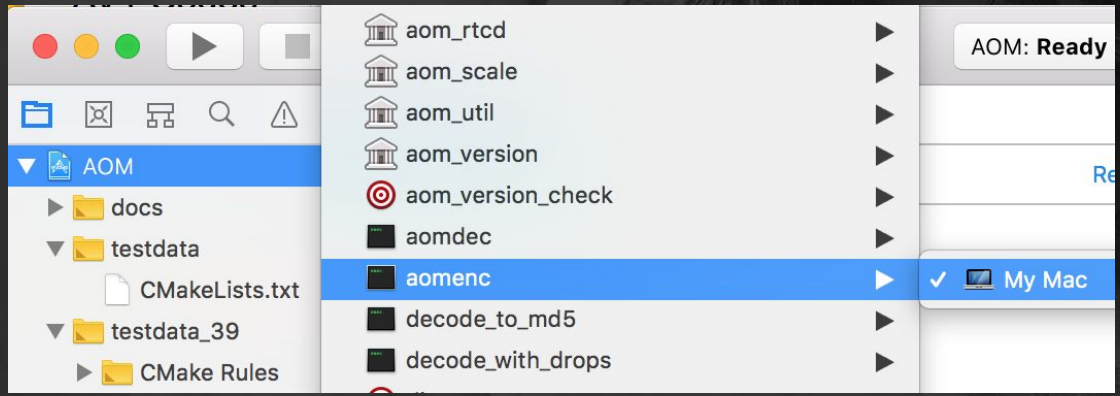
libaom

libaom is effectively a set of tools - referred to ‘experiments’ - which you can find freely available here: https://aomedia.googlesource.com/aom
Bugs (Monorail): https://bugs.chromium.org/p/aomedia/issues/list

Builds for any platform using CMake (>= 3.5) - example: `cmake ../aom -DCONFIG_DEBUG=1`

Cross compile gcc/clang/lvmm
Various IDE integrations
e.g. Xcode

```
cmake -G Xcode
open AOM.xcodeproj
```



Encoding: aomenc (libaom command line encoder tool)

Encoder works on raw **yuv4mpeg** (e.g. `ffmpeg -i in.mp4 --pix_fmt yuv422p raw.y4m`)

```
aomenc -v -w 3840 -h 2160 --cpu-used=0 --target-bitrate=1500 -t 16 -u 0  -b 10  --fps=60 -o av1.webm raw.y4m
```

(tool output) Pass 1/2 frame 4/3 552B 11382 ms 21.08 fpm [ETA 48:16:11] or maybe a *lot* longer ;-)

Rate Control options	`--target-bitrate` (kbps)
Bitstream profile	`-u` (`-u 0` =**main**, `-u 1` =**high**, `-u 2` =**professional**)
Bit depth	`-b` (`-b 8`, `-b 10` or `-b 12`)
Aspect, frame size	`-w -h` (e.g. 4K `-w 3840 -h 2160`)
Frame rate	`--fps` (e.g. `--fps=60`)
Keyframe Placement	`--kf-min-dist` (e.g. `--kf-min-dist=360` key frame every 6 seconds)
Multi-threaded op	`-t` (`-t 16` use up to 16 threads) `--cpu-used=0`

+ heaps of other options (*see* `aomenc --help`)

Encoding: AV1 Bitstream **profiles** and **levels**

Profiles and levels specify restrictions on the capabilities needed to decode.

“**main**” - supports YUV 4:2:0 or monochrome streams with bit-depths 8 or 10.
“**high**” - adds support for YUV 4:4:4 streams with the same bit-depth
“**professional**” - bit-depth 12 + support for the YUV 4:2:2 video format.

Various levels, here are some examples...

1080p	Level **4.0**	32 tiles	< 12Mbps (main)	< 30Mbps (high)	1920x1080@30fps
4K	Level **5.0**	64	< 30Mbps	< 100Mbps	3840x2160@30fps
8K	Level **6.0**	128	< 60Mbps	< 240Mbps	7680x4320@30fps

Encoding: AV1 Digital Rights Management

AV1 is support's by **ISOBMFF** and **WebM** (Matroska) and so implies **common encryption** (**CENC**) / AES-128 based encryption of the payload

Both **cenc** and **cbcs** scheme types are permitted.

All **Open Bitstream Unit** (**OBU**) headers are *unencrypted* which makes manipulating encrypted AV1 streams easy.
Similarly Temporal Delimiter, Sequence Header, Metadata, except for those requiring protection are also unencrypted allowing easy processing and transport of streams containing encrypted AV1 payloads.

Tile groups are individually encrypted (so encryption can be parallel/concurrent)

Encoding: **FFmpeg** 4.0 (**libaom-av1** encoder)

First, you'll need to build latest ffmpeg yourself with `--enable-aom`

```
ffmpeg -i input.mp4 -c:v libaom-av1 -strict -2 -y av1.mkv
```

note, this is still considered experimental, hence `-strict -2`

Some parameters that get passed through to libaom (note there are many that are not today passed on)

Quality/Speed ratio modifier	`-cpu-used` from -8 to 8 (default 1)
Alternate reference frames	`-auto-alt-ref` enable use of alternate frames 2-pass only, from -1 to 2 (default -1)
Number of frames to look ahead	`-lag-in-frames` at for alternate reference frame selection from -1 to INT_MAX (def -1)
Constant quality mode	`-crf` select the quality from -1 to 63 (default -1)
Change threshold to skip	`-static-thresh` when blocks should be skipped by the encoder, 0 to INT_MAX (def 0)
Frame drop threshold	`-drop-threshold` from INT_MIN to INT_MAX (default 0)
Noise sensitivity	`-noise-sensitivity` noise sensitivity (from 0 to 4) (default 0)
Error resilience configuration	`-error-resilience` (default) losses of whole frames (partitions) frame partitions

Encoding: **GStreamer** 1.14 (**av1enc** element)

Libaom is wrapped in the “aom” bad plug-in right now, it’s based on the libvpx plugin element which is why currently it sits in that category until reviewed

Again you need to build this yourself from sources

```
gst-launch-1.0 filesrc location=input.mp4 !
video/x-raw,width=3840,height=2160 ! av1enc ! queue !
filesink location=av1.mp4 -v -e
```

Encoding: others **rav1e SVT-AV1**

rav1e - (Xipf) encoder, improves on speed of aomenc (Rust codebase)

SVT-AV1 - (Intel) Scalable Video Technology runs as encoder and decoder plugin to both FFmpeg and GStreamer (C code) - NetFlix is using this today

Like most things AV1 is work in progress

TODO: add graphs showing comparisons as similar quality

Transport: AV1 over **MPEG-DASH**

ISOBMFF (MP4) containers already support AV1 payloads

WebM files can also play under Firefox Nightly

```
<Period id="0" start="PT0S" duration="PT175.014S" >
     <AdaptationSet id="0" mimeType="video/mp4" codecs="av1.experimental.[version]"
                    width="3840" height="2160" subsegmentAlignment="true" >
          <Representation id="0" bandwidth="1500000">
               <BaseURL>av1.mp4</BaseURL>
               <SegmentBase indexRange="6978450-6978478">
                    <Initialization range="0-1116" />
               </SegmentBase>
          </Representation>
     </AdaptationSet>
</Period>
```

Transport: AV1 over **HLS/FPS** (CMAF)

Whilst Apple devices *do not* yet support playback of AV1 samples, we should expect this to work in the future.
HLS specification - since v5 - includes support for **fragmented MPEG-4 (fMP4)**
AV1 **HLS** segments would consist of a *Movie Fragment Box* (*moof*) containing a subset of the sample table and a *Media Data Box* (*mdat*) containing the AV1

```
#EXTM3U
#EXT-X-TARGETDURATION:175
#EXT-X-VERSION:5
#EXT-X-MEDIA-SEQUENCE:1
#EXT-X-PLAYLIST-TYPE:VOD
#EXT-X-MAP:URI="av1.mp4",BYTERANGE="1116@0"
#EXTINF:175.014,
#EXT-X-BYTERANGE:6978478@6978450
av1.mp4
```

Transport: AV1 over **WebRTC**

AV1 similar to VPx is to be included as an **RTP payload format** (which describes how the encoded AV1 bitstream is encapsulated in RTP packing)

This means AV1 encodings can be **peer2peer shared** via WebRTC, and in the future, low bitrate/**low latency** AV1 encodings for live* sports and conferencing will become possible.
Media delivery would be **WebM** contained AV1/**Opus**** over secure WebSockets

** Problem right now is live software encoding is just way too slow to benefit from low latency integrations, but we should expect this to be the norm in the future with dedicated hardware live encoders*

*** WebRTC requires **Opus** audio codec, based on (Skype) SILK and (Xipf) CELT which is also royalty-free!*

Playback: AV1 decoders

libaom - aomdec tool will decode AV1 to raw yuv4mpeg frames, or you can include the library into your own native player.

FFmpeg - ffplay tool includes libaom and so will can play AV1 4K directly

```
Stream #0:0: Video: av1 (Main), yuv420p(tv), 3840x2160, SAR 1:1 DAR 16:9, 29.97 fps, 29.97 tbr, 1k tbn, 1k tbc (default)
Metadata:
  HANDLER_NAME    : Core Media Video
  ENCODER         : Lavc58.19.102 libaom-av1
  DURATION        : 00:01:30.026000000
```

Mozilla Firefox Nightly (WebM)

Google Chrome Canary (WebM)

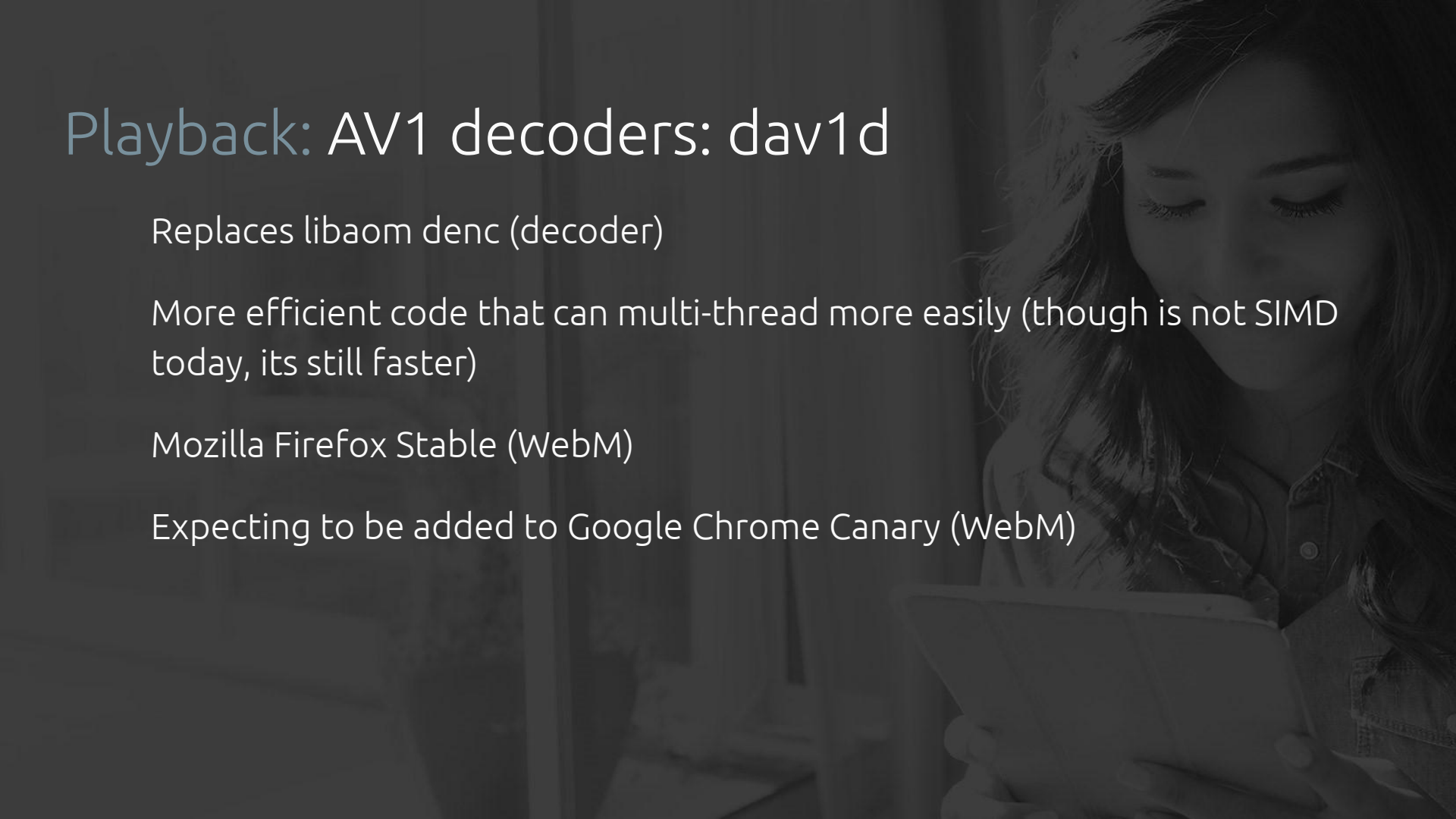
Playback: AV1 decoders: dav1d

Replaces libaom denc (decoder)

More efficient code that can multi-thread more easily (though is not SIMD today, its still faster)

Mozilla Firefox Stable (WebM)

Expecting to be added to Google Chrome Canary (WebM)



How does AV1 compress **30%** better than HEVC?

Larger and more dynamic block sizes and transforms have less compression overhead and have been proven to compress better overall.

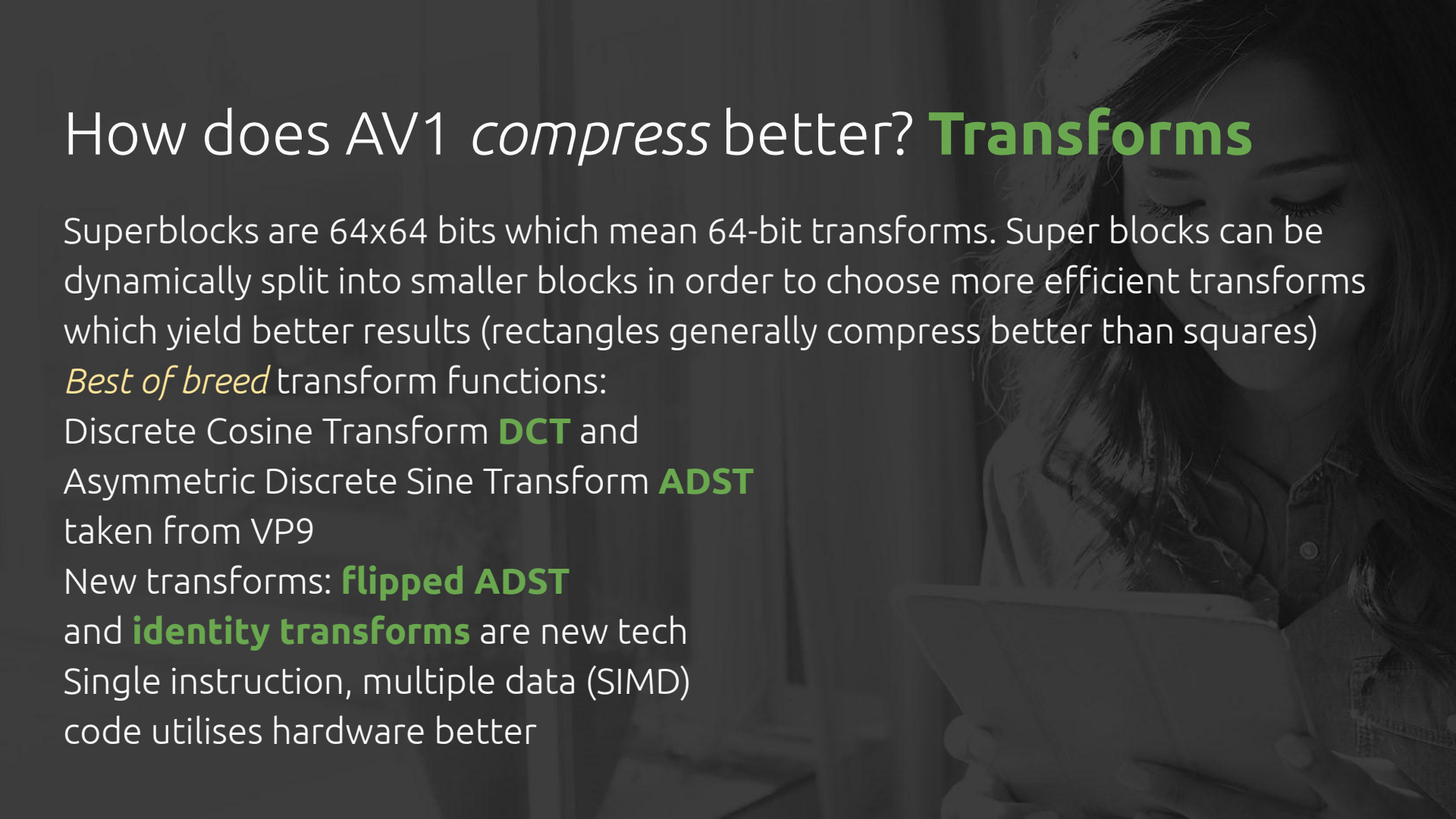
For live, tile parallelism means we can more highly compress more sets of tiles by sharing the same compute time space.

A larger suite of prediction algorithms and motion vectors. Algorithms from Daala and Thor such as **Chroma-To-Luma** predictions and other coefficient predictions are all included as part of AV1.

Smarter use of reference inter-frames onto golden intra-frames.
New techs like texture segmentation and things like warp/global motion sampling

How does AV1 *compress* better? **Transforms**

Superblocks are 64x64 bits which mean 64-bit transforms. Super blocks can be dynamically split into smaller blocks in order to choose more efficient transforms which yield better results (rectangles generally compress better than squares)
Best of breed transform functions:
Discrete Cosine Transform **DCT** and
Asymmetric Discrete Sine Transform **ADST**
taken from VP9
New transforms: **flipped ADST**
and **identity transforms** are new tech
Single instruction, multiple data (SIMD)
code utilises hardware better



Why does AV1 *look* better? More References

Similar to MPEG, AV1 maintains concept of **Inter** and **Intra** frames

Intra frames make references to Inter frames in similar fashion, but new to AV1 is concept of the **golden frame**, which is a key frame **encoded at the highest quality**, allowing reference frames to link through to that without wasting space

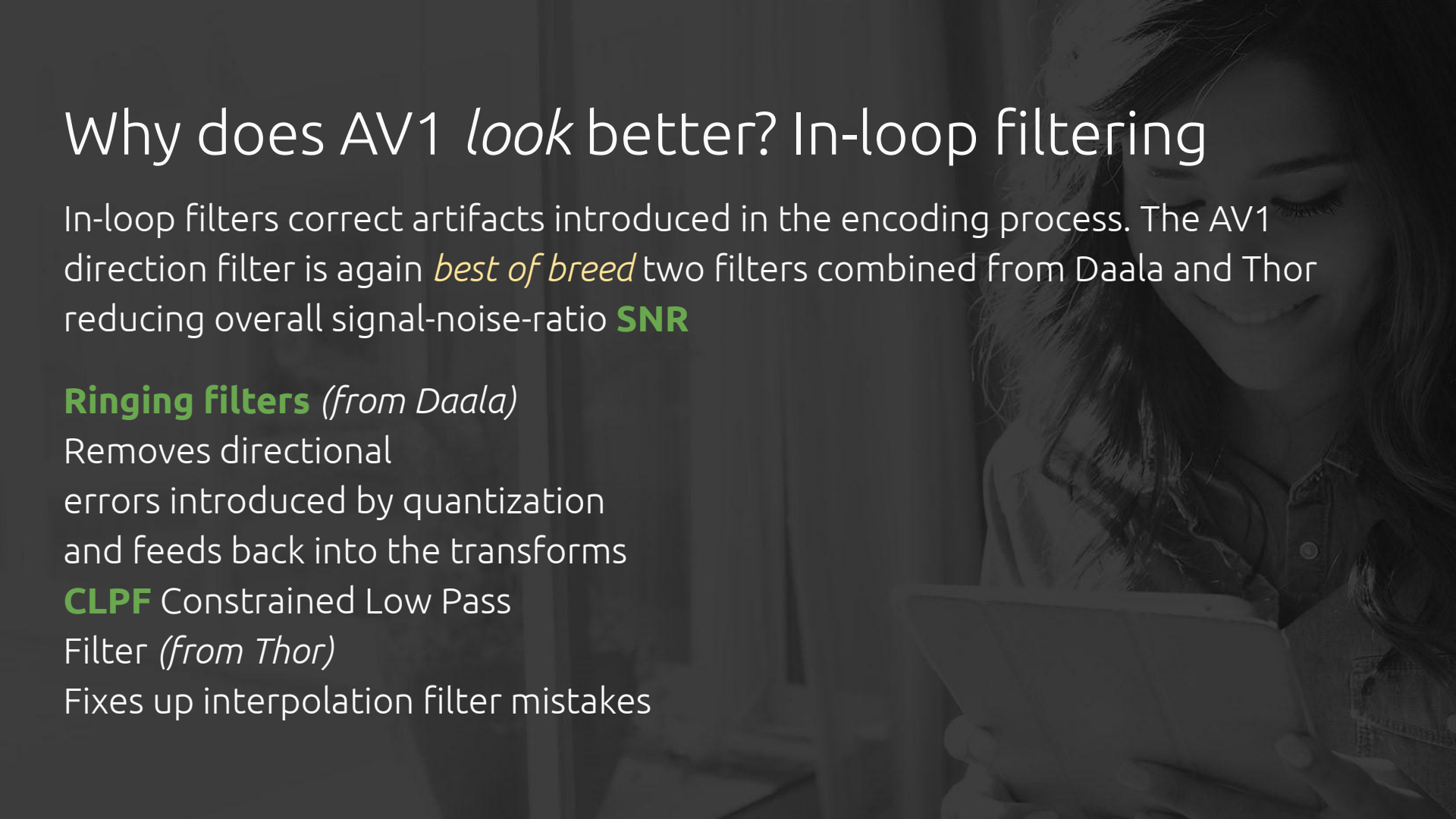
As with MPEG Inter-frames also includes a compound mode - i.e. bi-prediction with other frames

The players also keep 8 references frames enough for 3 temporal or 3 spatial layers at same time off a golden frame allowing run-time dynamic choices

Why does AV1 *look* better? In-loop filtering

In-loop filters correct artifacts introduced in the encoding process. The AV1 direction filter is again *best of breed* two filters combined from Daala and Thor reducing overall signal-noise-ratio **SNR**

Ringing filters *(from Daala)*
Removes directional
errors introduced by quantization
and feeds back into the transforms
CLPF Constrained Low Pass
Filter *(from Thor)*
Fixes up interpolation filter mistakes



Machine learning potential (smarter encoders)

Predictors take previous frames to determine where blocks will end up/look like

Whilst having more predictors increases encoding complexity - and so time to compress - there are more choices available which can improve overall quality of the playback.
With machine learning selecting the best predictors per asset to improve overall quality vs compression time becomes possible

Many Intra and Inter frame predictors to choose proven from codecs, e.g. Directional, Paeth, Smooth, Palette, Blend neighbour, Edge extension, Chroma from Luma, Dynamic Reference Motion Vector Prediction (REFMV)

Low latency, how AV1 benefits **live sport**

AV1 introduces the concept of **tiles** which can be grouped as sets of blocks and encoded and decoded independently from other tiles.

What this means is with sports in particular playback of the action can carry on even if part of the image gets lost, for example the action is more likely to continue and be watchable

Processing power can be dedicated for the tiles with the most motion

(right now though thats a lot of processing power, but it’ll get better)

Some other interesting stuff, kinda

Texture segmentation methods uses a deep learning based approach to detect the texture regions in a frame that is perceptually insignificant to the human visual system

High Dynamic Range (**HDR**) + Wide Colour Gamut (**WCG**) are support to a point where colour spaces, matrices and transfer functions can be encoded directly in the bitstream itself

Less interesting… CABAC replacement (Adaptive Multi-symbol Entropy Coding), Coefficient Coding, Perceptual Vector Quantization, Golumb codes… so let's move on...

How do we know/prove all this *is* actually better?

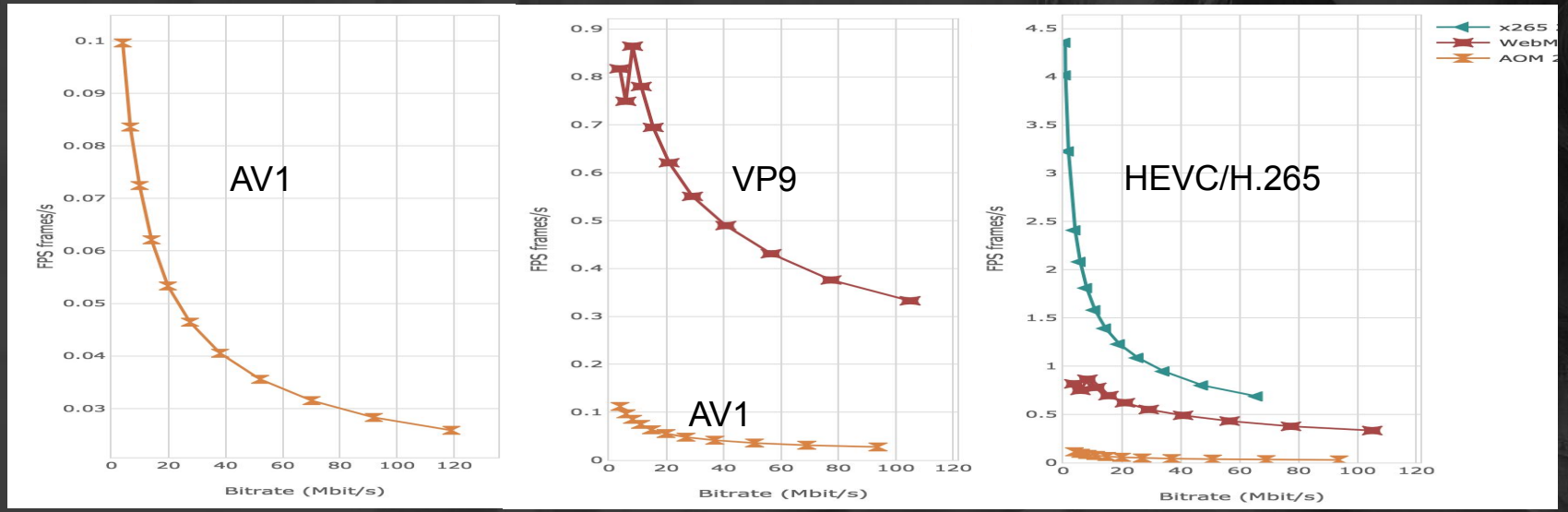
arewecompressedyet.com - automation first decides if an experiment compresses better, then *humans* crowd source using a web browser decide if it actually looks better, same, or worse using the WebAssember (Wasm) based **AOMAnalyser** tool. This tool also shows how frames have been divvied up into blocks, what predictions were applied etc

MSU VQMT Moscow State University (Video Quality Measurement Tool) is also used to look at peak-to-peak signal-to-noise ratio (PSNR), mean absolute difference (MAD), MSU Blurring/Blocking luminance/contrast similarity (SSIM), etc

Some others, e.g. Elecard benchmark https://www.elecard.com/benchmarks

Where are things at today? The Elephant in the room

30% smaller! but... encoding is really (really slow) many (2500 to 3000) times slower than VP9 which is also many factors slower than HEVC.... today.
My example, several days to do several minutes on 16-core IvyBridge :-(



What’s being done about that?

Basically the AV1 ‘bitstream’ format is not quite frozen whilst it is moving hardware vendors were unable to work with it, that’s bad. Soon though.
(update Sep 2019, v1.1 *is* frozen)

We can expect to seem some FPGAs shortly, in fact there are some already, Socionext over cloud encoding service is a recent new comer.
Typically we should expect a few years until mainstream hardware is available.

But.. some really big players are in that mix including Broadcom, ARM and Apple
(update Sep 2019 Broadcom came good with this, see BCM7218X)

It’s all going to be ok!

Final thought...

Encoding is slow, and likely remain expensive in the short term

Just maybe Studios, such as Disney could provide the AV1 encodings at some standard ABR rates?

There is a climate concern element to consider too, how many hours and energy is wasted re-encoding, why not get the encoding from the source instead?

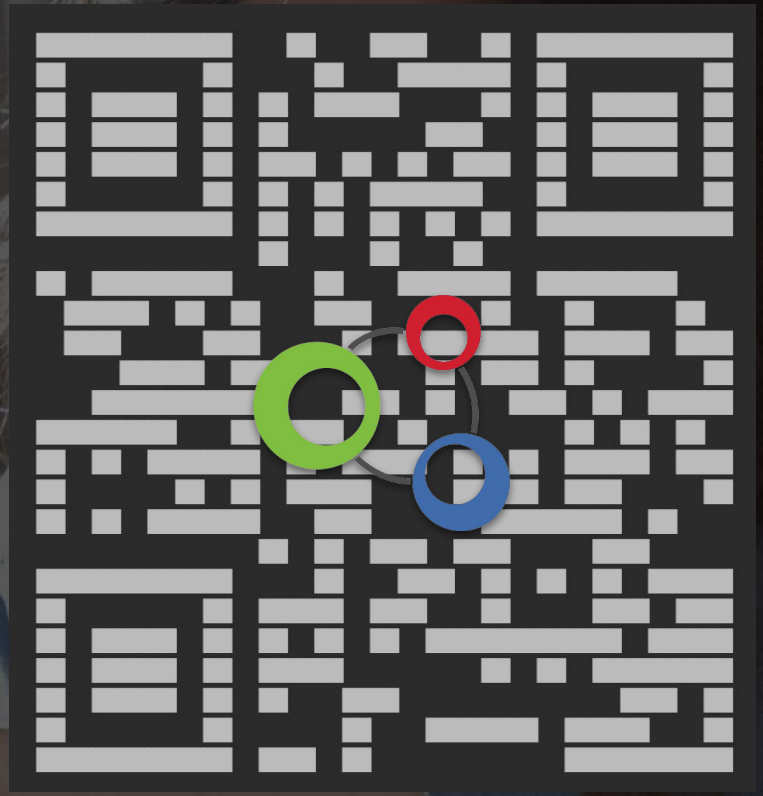
Thanks!

Slides are here:

https://goo.gl/**pGnNgJ**

Apple Low-Latency https://tinyurl.com/**yyr2rz8m**
WebAssembly https://goo.gl/**2ahsEY**

Also thanks the the Reddit r/av1 community for review:
Balance- ScopeB flashmozzg



Twitter/LinkedIn: kevleyski

Phone: +61 2 8096 8277
Email: klambert@switch.tv
Website: www.switch.tv
Address: Suite 121, Jones Bay Wharf,
26-32 Pirrama Rd, Pyrmont, Sydney, NSW, Australia

switchmedia