dav1d, 1 year later

Jean-Baptiste Kempf
0202-2020
Who am I?

President of VideoLAN

Work/Manage VLC, x264, FFMpeg, dav1d

Other multimedia projects
VP9++?

- VP9 is a semi-failure
- Good format, royalties OK
- Rarely used
  - Have you ever watched an anime rip in VP9?
  - Spec?
- YT, Netflix

AV1

- Different from just VP10
- AOM, Mozilla, Cisco
- Excellent results
AV1 ecosystem

- Numerous encoders
  - libaom, SVT-AV1, rav1e
  - EVE-AV1, Ateme, Harmonic, Bitmovin
  - Ngcodec, FPGA, ...

- Numerous deployments
  - Youtube, Netflix, Facebook
  - Cloud vendors

- Hardware is coming in 2020
  - Intel, nVidia, AMD?
  - Samsung TV, Amlogic, Broadcom
• Competition is coming?
  - VVC in July 2020, EVC in April 2020
  - MPEG-5 LC-EVC
  - AV2???

• Royalties
  - VVC is based on HEVC
    • 5 patent pools? :D
    • Are improvements enough to justify?
    • HEVC semi-failure
  - EVC is not enough
    • Gains?
    • MC-IF
  - LC-EVC is not actually a codec
Dav1d goals

- "AV1 needs a great software decoder"
- Faster decoder everywhere
- Very portable and cross-platform
- Small binary size (ffvp9)

Launched last year

- Announced at VDD 2018
- First release in December 2018
- Last release: 0.5.2, 0.6.0 soon
Historique

- **Oct ‘18** Announce
- **Dec ‘18 0.1** 4x faster than libaom on x64
- **Mar ‘19 0.2** 2x faster than libaom on ARM64, 4x on ARM32, 5x on x64
- **May ‘19 0.3** Focus on SSSE3 (+25%), ARM (+12%)
- **Aug ‘19 0.4** Bugs, MSAC, RAM usage, VSX
- **Oct ‘19 0.5** Finish ARM64, SSSE3
- **Dec ‘19 0.5.2** SSE2, ARM32
Fast on desktop

dav1d vs aomdec multi-thread performance

3x - 5x faster

SSE2
AV1 Decoding (ARMv8 Multi-Thread)

Faster on ARM

2.5x - 4x faster
Complexity of AV1

![Graph showing decoding speed versus CPU cores for different codecs: dav1d, 62fcd0c, ffvp9, ffh264, and ffhevc.]
Dav1d architecture

- Dual Passes
  - Rare inside a decoder
  - First pass to analyze, Second to decode

- Dual Threading model
  - Tile Thread
  - Frame Thread
  - Need to set both to get best decoding
1. C version is faster

AV1 Decode Performance (Single Threaded ARMv8 64-bit)

And more is coming!
2. Threading is better

Why is dav1d faster?

Thread Scaling on x86_64 (2019-Oct-24)

- libgav1-b48796f
- libaom-2b471a3
- dav1d-3b33c52

FPS

Threads
3. low-level development

C (no C++ overhead)
Hand-written asm
No intrinsics
ASM aware code

- MSAC
- Inverse Transform
- Motion Compensation
- Intra Pred
- Loopfilter
- Loop Restoration
- CDEF
- Film Grain

Non-ASM code

- Decode_coef (8%)
- Ref_mv (12%)
- Decode
<table>
<thead>
<tr>
<th></th>
<th>AVX-2</th>
<th>SSSE-3 32 + 64bit</th>
<th>ARM64</th>
<th>ARM32</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSAC</td>
<td>→</td>
<td>Only SSE2</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Inverse Transform</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Motion Compensation</td>
<td>Yes</td>
<td>Yes Warp SSE2</td>
<td>Yes</td>
<td>Yes emu_edge</td>
</tr>
<tr>
<td>Intra Pred</td>
<td>Yes</td>
<td>Yes z1, z2, z3</td>
<td>Yes</td>
<td>Partial</td>
</tr>
<tr>
<td>Loopfilter</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Loop Restoration</td>
<td>Yes</td>
<td>Yes Wiener SSE2</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CDEF</td>
<td>Yes</td>
<td>Yes + SSE2</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Film Grain</td>
<td>Yes</td>
<td>Yes Except 4:4:4</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
X264, libavcodec

- x264
  - 68kLoC C
  - 37kLoC asm (25k x86, 12k ARM)
- libavcodec
  - 540 kLoC C
  - 80 kLoC asm (40k x86, 40k ARM)
- dav1d
  - 25 kLoC C
  - 64 kLoC asm (45k x86, 19k ARM)
GSoC 2019: GPU optimizations

- Vulkan Shaders
- Android only

Done:
- Loop Restoration (SGR, Wiener)
- CDEF
- Film Grain in GLSL

Future:
- Finish?
Future

- 10bit
  - 16bit
  - ARM64/ARM32 ongoing
  - X86 ??

- GPGPU
Thanks!