# Software Implementation Working Group Update

## AOMedia Workshop, ICIP 2024

Abu Dhabi, United Arab Emirates

Hassene Tmar (Meta), Faouzi Kossentini (VLITT)

Hassene Tmar

- Technical Program Manager at Meta
- Supporting the Video Infrastructure team
- 2017-2022 Intel, Open sourcing SVT-HEVC,VP9, AV1
- 2012-2017 eBrisk Video, building HEVC SW implementations
- Software Coordinator at the Software Implementation Working Group AOM
- Maintaining of the SVT-AV1 projects
- Interests: Codec implementations and evaluations frameworks, Open Source development, Quality Metrics and …

# Outline

- SIWG Mission
- Recent SVT-AV1 development focus areas
- SVT-AV1 Performance Updates
- Recent SVT-AV1 Algorithmic Optimizations
- Arm – based Optimizations
- Enabling Further Expansion of AV1 Coverage
- Next Focus Areas
- Acknowledgements

# Software Implementation Working Group Mission

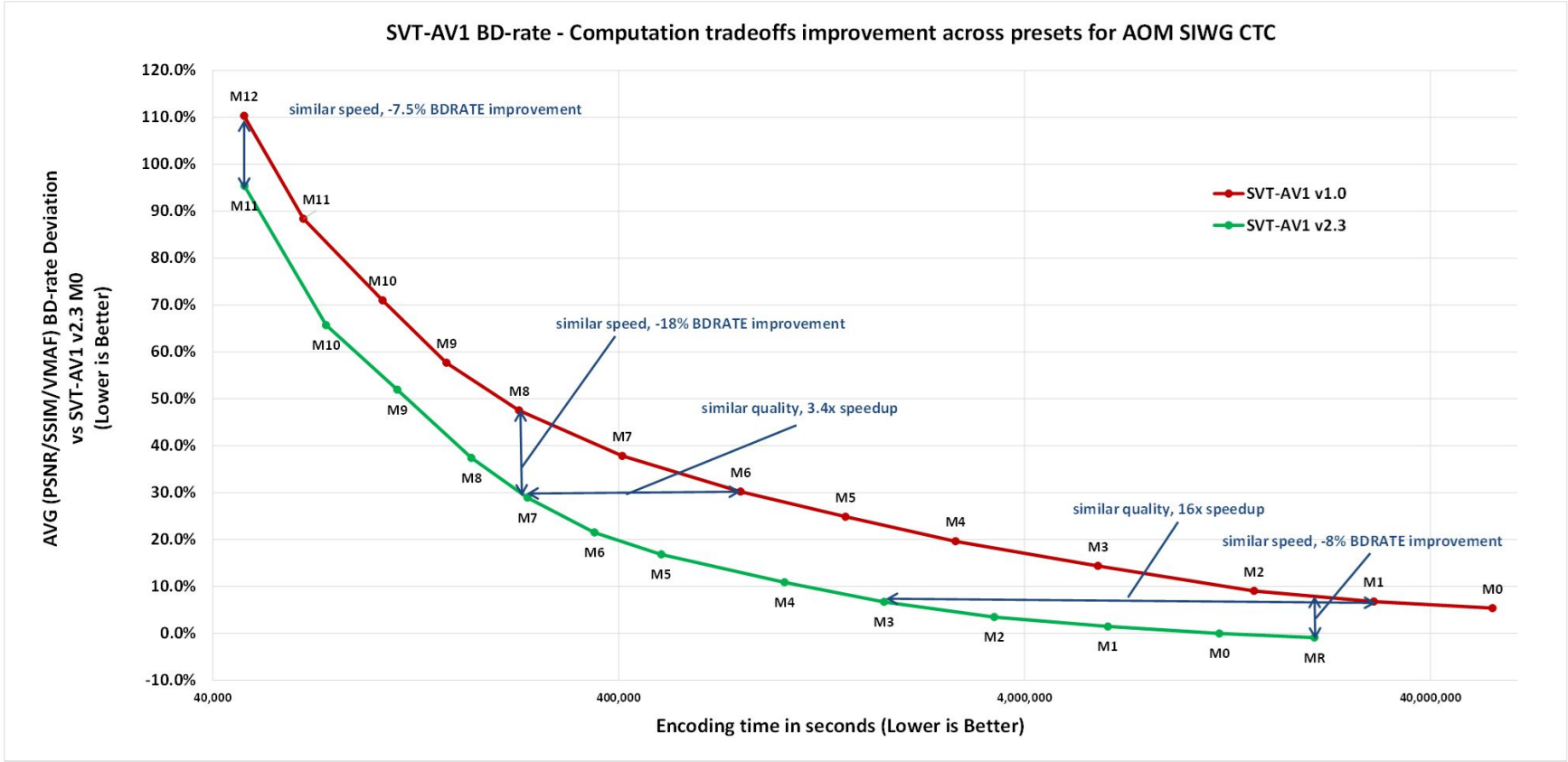
**Identify** areas that are difficult to implement, produce **benchmark** data, and **create production-grade software** implementations that are ready for **deployment** when a corresponding AOMedia specification is published, or soon thereafter.

https://aomedia.org/press%20releases/aomedia-software-implementation-working-group-to-bring-av1-to-more-video-platforms/

# Recent SVT-AV1 development focus areas

- Optimized encoder speed and quality tradeoffs
- Added AVX512 and Arm-Neon optimizations for faster decoding
- Created new encoding modes for faster software decoding
- Improved memory efficiency, reducing consumption by 25% in multithreading scenarios
- Enhanced API to support parameter updates during encoding sessions
- Increased code stability through unit test coverage and bug resolution.

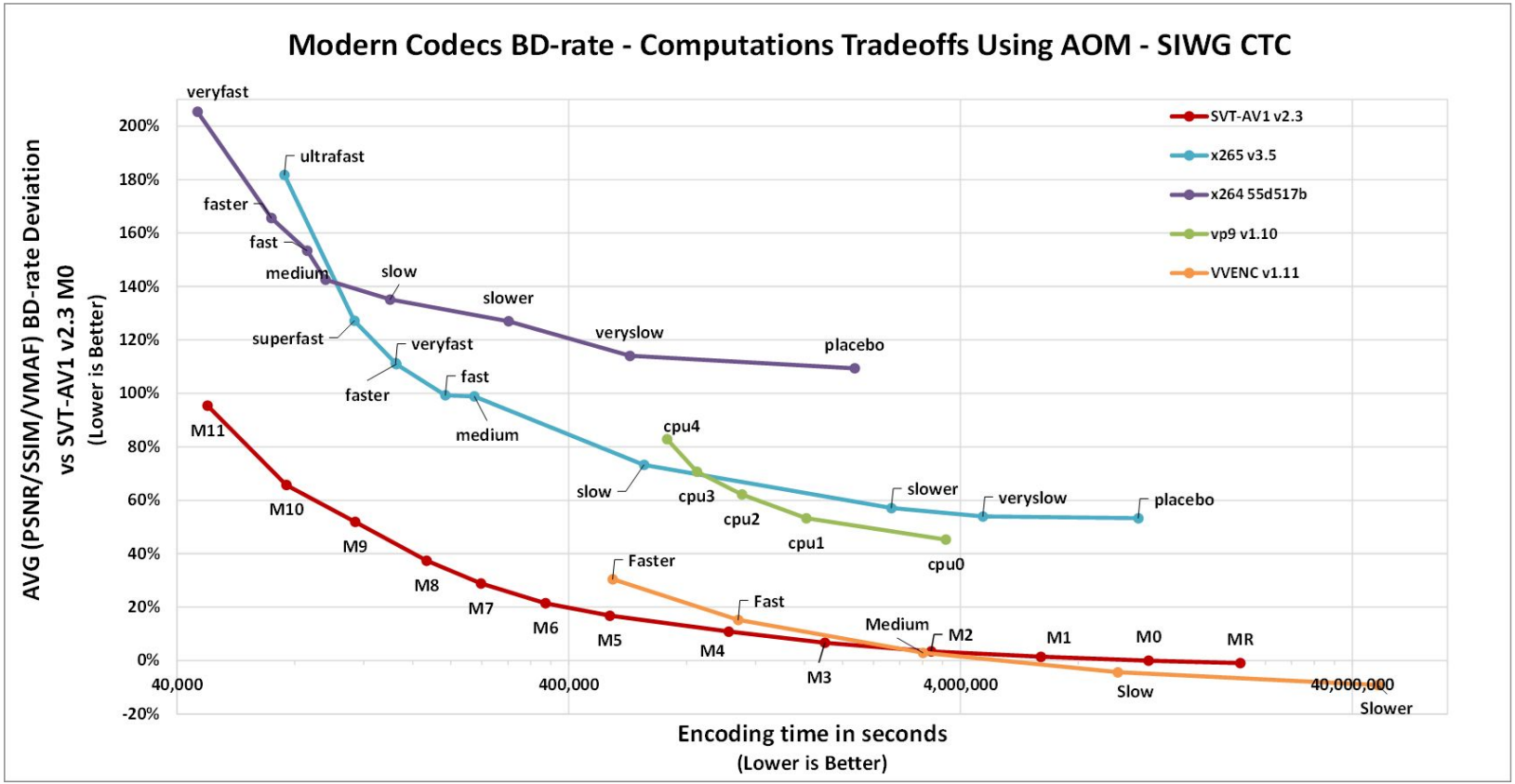
# Performance Improvements since v1.0 on SIWG CTC* – VOD use case




*https://aomedia.org/docs/SIWG-D001o.pdf

Thanks to P'sao Pollard-Flamand - Intel - for the help generating this data

# Comparison Between Encoders Using SIWG CTC-v1




Thanks to P'sao Pollard-Flamand - Intel - for the help generating this data

# Recent SVT-AV1 Algorithmic Optimizations

- Motion Estimation (ME) optimizations
  - Enhanced ME precision by modulating the search area based on Hierarchical Motion Estimation (HME) results
- Non-Square (NSQ) partitioning optimizations
  - Developed fine-grain optimization levels and modulating the ME search area based on the content type
- Depth partitioning optimizations
  - Improved the accuracy of depth prediction within the SuperBlock (SB) using motion estimation results and lookahead data
- Mode Decision (MD) optimizations
  - Improved candidate search and pruning algorithms for INTER, and INTRA candidates, reducing search time and increasing accuracy.
- Filtering optimizations
  - Modulated Deblocking Loop Filter (DLF) and Temporal Filtering (TF) based on coding mode and inter-picture brightness deviations.

# SVT-AV1 Arm SIMD Optimization Summary

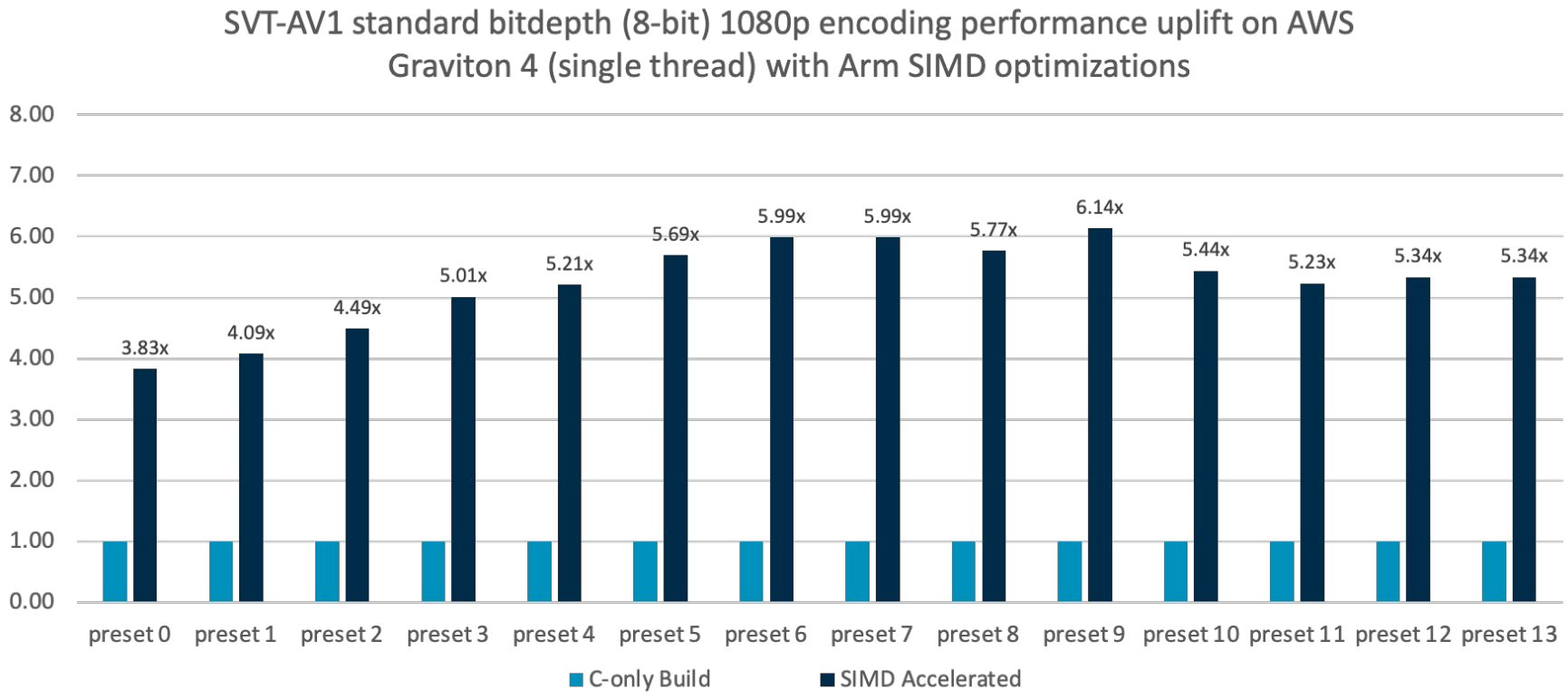
1. Arm SIMD support started in SVT-AV1 release 1.8.0:
   - Ittiam ported lots of Neon code from libaom (originally authored by Aem).
   - Ekumen subsequently translated additional x86 SIMD paths to Neon.
   - Arm contributing to SVT-AV1 directly since release 2.1.1.

2. Other major contributions:
   - Overhauled SVT-AV1 unit test infrastructure and ported many tests from libaom.
   - Added Arm run-time CPU feature detection to make use of Arm v8.x/9.x ISA extensions.

3. Future goals for Arm SIMD acceleration:
   - Port all relevant Arm v8.0 Neon SIMD paths from libaom.
   - Port all Arm v8.4 DotProd and Arm v8.6 I8MM Neon SIMD paths from libaom.
   - Port all Arm v9.0 SVE2 SIMD paths from libaom.
   - Optimize remaining SIMD paths unique to SVT-AV1.

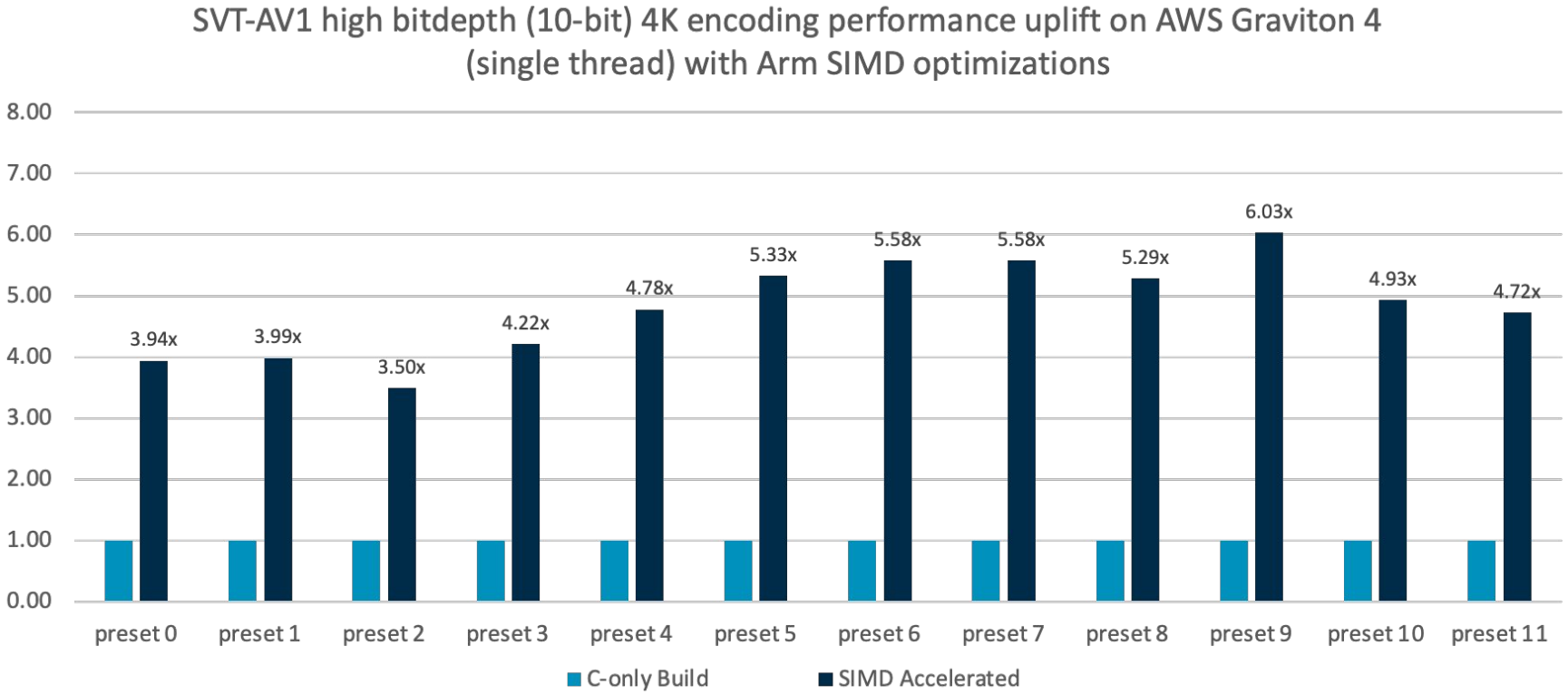
Thanks to Jonathan Wright - Arm - for the help generating this data

# SVT-AV1 SBD Performance Increase on AWS Graviton 4




Thanks to Jonathan Wright - Arm - for the help generating this data

# SVT-AV1 HBD Performance Increase on AWS Graviton 4




Thanks to Jonathan Wright - Arm - for the help generating this data

# Expansion of AV1 Coverage Using SW Decode

- Dav1d open source decoder provided and maintained by VideoLan has been making progress over the past couple of years.

- Dav1d performance optimization was one of the enabling factors that allowed AOM member companies such as Meta to deploy AV1 successfully despite the lack of AV1 HW decoders.

- Earlier this year Google has enabled Dav1D to be the platform decoder on Android 12+.

- Yet, we still needed further improvements to AV1 SW decoding to be deployed at a higher scale!

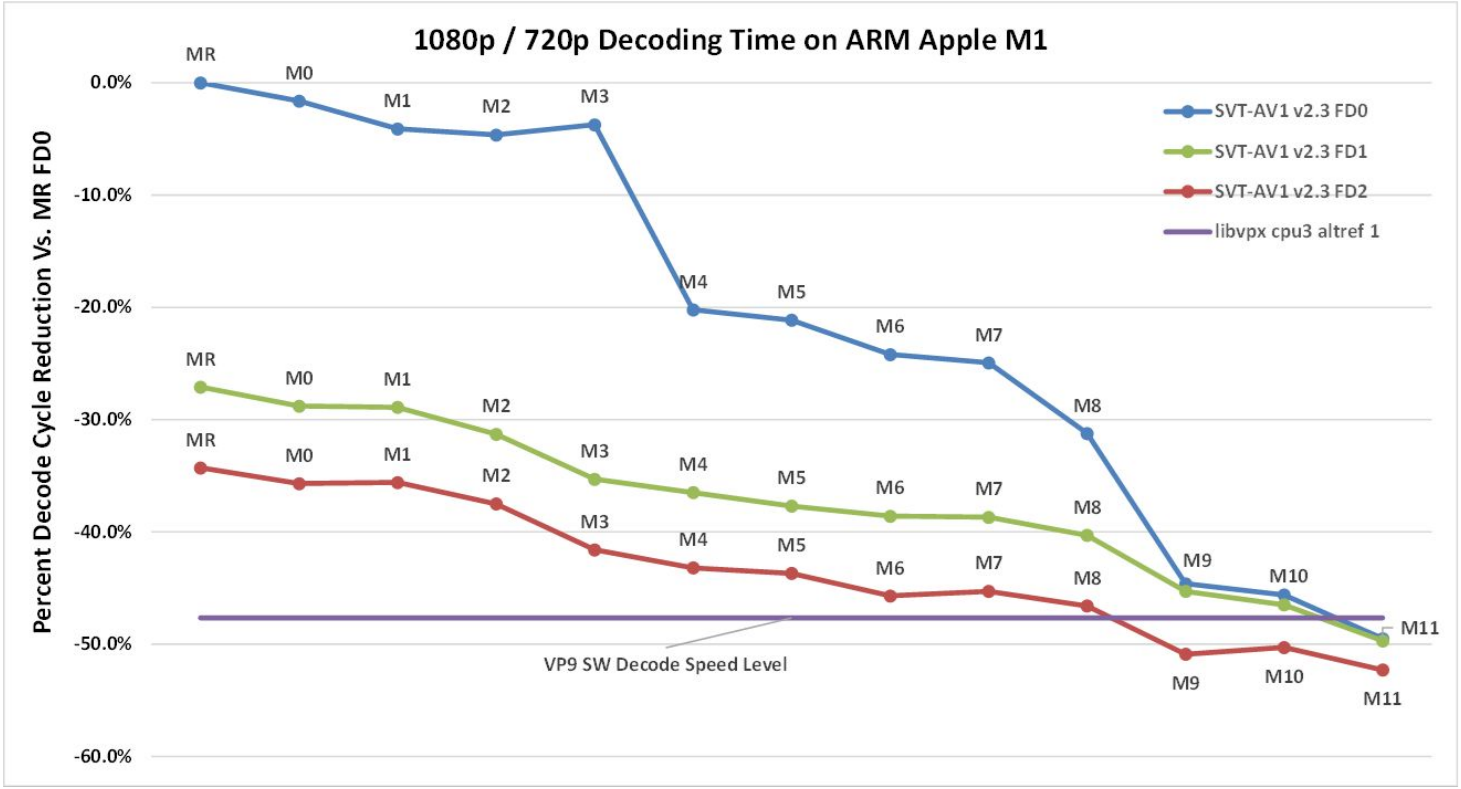
# SVT-AV1 Fast Decode (FD) Encoding Modes

- Since SVT-AV1 v1.0, a fast decoding mode has been developed by the Intel team, producing bitstreams that are easier to decode by SW decoders (10-15% decode time savings).
- SVT-AV1 v2.3 introduces an improvement to the fast decode mode, with the introduction of a second level that allows for even faster decoding

➢ To estimate the impact of these levels, we ran the following tests:

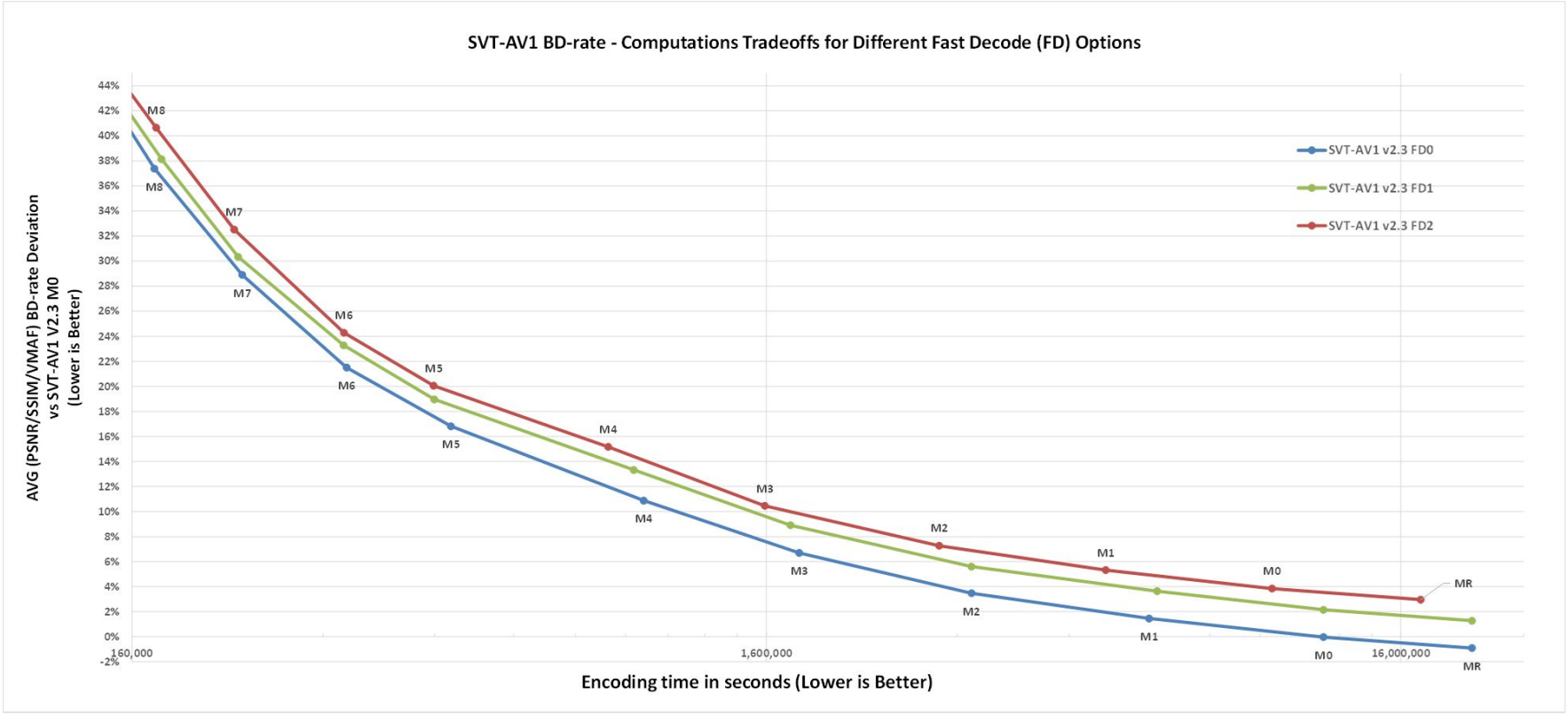
  - Measuring decode complexity on a mac-mini Arm CPU, and
  - Running battery drain tests

# Decode Speed Tests on Arm




Thanks to P'sao Pollard-Flamand - Intel - for the help generating this data

# Impact on Encoder Performance (Zoomed In)



# Battery Drain Tests on mid-end Android Phone

## Phone Spec

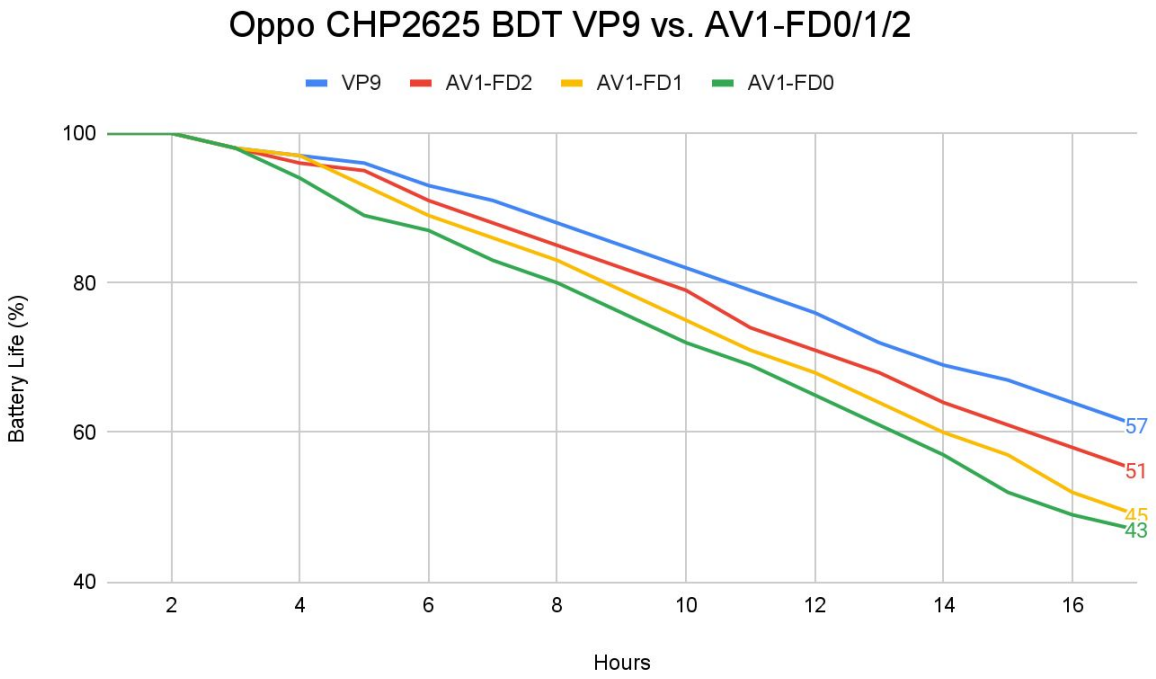
- Oppo Reno 12 5G CPH2625
- Screen resolution 2412x1080
- ~$400 range.

## Test setup

- VLC benchmark suite
- Same content tested offline - bitrates 500-2000 Kbps
- Mix of resolutions 360p-1080p
- Radio off - Display brightness 25%

## Results

- AV1 FD2 has a drain rate that's **6%** higher than that of VP9
- AV1 FD0 has a drain rate that's **14%** higher than that of VP9



# Next Focus Areas For SVT-AV1

- Optimize further the encoders' quality vs complexity tradeoffs based on advanced algorithmic optimizations
- Improve decoding efficiency towards higher decode speeds for the fast-decode mode especially in HBD.
- Continue developing Arm Neon and x86 AVX512 optimizations
- Reduce further the number of presets to at most 10 presets (since the encoder has become much more efficient)

# Acknowledgements


- Thanks to **Foued Ben Amara**, **Hassen Guermazi**, **Phoenix Worth**, and **P'sao Pollard-Flamand** from Intel for their tremendous efforts in the optimization of the SVT-AV1 encoder tradeoffs and the fast-decode encoding modes.
- Thanks to **Ittiam**, **Ekumen**, and **Arm** for their work on developing and integrating Arm Neon code into SVT-AV1
- Thanks to **Christopher Degawa** from **Videolan** for the tireless efforts to keep our CI/CD up and running 24/7.
- Thanks to **Meta** for sponsoring work in different areas of this project.
- Thanks to the SVT-AV1 open source **community** for their contributions, testing, raising issues, and enhancements of the code base.

# Questions

# Benchmark Details Arm

## Standard Bitdepth

Platform: AWS Graviton 4

OS: Ubuntu 24.04

Compiler: Clang 18.1.3

SVT-AV1 hash: e17b8409 (2024-09-19)

Input video:
https://ultravideo.fi/video/Bosphorus_1920x1080_120fps_420_8bit_YUV_Y4M.7z

Benchmark command line:

./SvtAv1EncApp -i ${VIDEO_SBD_FHD} --lp 1 --preset ${PRESET} --input-depth 8 -b out.mkv

## High Bitdepth

Platform: AWS Graviton 4

OS: Ubuntu 24.04

Compiler: Clang 18.1.3

SVT-AV1 hash: e17b8409 (2024-09-19)

Input video:
https://ultravideo.fi/video/Bosphorus_3840x2160_120fps_420_10bit_YUV_Y4M.7z

Benchmark command line:

./SvtAv1EncApp -i ${VIDEO_HBD_4K} --lp 1 --preset ${PRESET} --input-depth 10 -b out.mkv