2023 IEEE International Symposium on Workload Characterization (IISWC) | 979-8-3503-0317-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/IISWC59245.2023.00021

Do Video Encoding Workloads Stress the Microarchitecture?

Steffen Jensen[1] Jaekyu Lee[2] Dam Sunwoo[2] Matt Horsnell[2]
Matthew Siggs[3] Jeeho Ryoo[3] Lizy John[1]

[1]*University of Texas at Austin*
[2]*Arm*
[3]*British Columbia Institute of Technology*
sjensen7@utexas.edu, jaekyu.lee@arm.com, dam.sunwoo@arm.com, matt.horsnell@arm.com, msiggs1@my.bcit.ca, jeeho_ryoo@bcit.ca, ljohn@ece.utexas.edu

Abstract

Video encoding/decoding is an extremely relevant workload in our society today. Video accounts for a significant percentage of the world's online traffic, which is expected only to be growing. Thus, it is important to understand these workloads to optimize the hardware to handle them better. There is significant interest in the royalty-free AV1 codec, but we identify that it consumes a significantly higher runtime than other popular codecs, such as H.264/AVC, H.265/HEVC, and VP9. However, the reasons for the slowdown of the AV1 codec are not well-understood by prior work to the best of our knowledge.

This paper explores the reasons for the large runtimes taken by AV1 workloads. We first focus on profiling the microarchitectural characteristics of the SVT-AV1 encoder, which implements the AV1 codec, to identify acceleration opportunities with a wide spectrum of encoding parameters.

We discover that the runtime of AV1 encoders is higher than other encoders because AV1 encoders require a larger number of instructions to encode the same video, rather than any significant microarchitectural inefficiencies. Among microarchitectural components, we observe branch misprediction to be the component with the most significant impact on performance. In light of this, we evaluate the performance of several different branch predictors using Championship Branch Prediction (CBP) frameworks. From this, we find that increasing the size of the branch predictor as well as using a TAGE branch predictor rather than Gshare both had a significant positive impact on branch predictor performance.

We also compare the scaling of SVT-AV1 against other codecs by giving each encoder access to an increasing number of threads. Finally, we observe that SVT-AV1 contains the highest degree of parallelism of the tested encoders. Because of this, increasing concurrently running threads and optimizing branch prediction may help bridge the gap in runtime between SVT-AV1 and encoders implementing other codecs.

1. Introduction

Video is an extremely popular media format in our society. In 2017, video accounted for an estimated 74% of all web traffic [1] and this figure is only growing. According to the Cisco Visual Networking Index Report [2], the video will account for 82% of all internet traffic by the year 2022. Additionally, the recent pandemic has placed a new importance on video in all of our lives as we have adjusted to use video conferencing for work and to stay connected with friends and family. Additionally, many workers are now considering transitioning to a hybrid work model more permanently. A Mckinsey study [3] reports that globally 53% of workers would like to continue to work from home at least three days per week in the future. It is also reported that 86% of businesses used video as a marketing tool in 2020, up from 69% three years ago [4]. The massive volume of video traffic forces video streaming companies such as Youtube, Netflix, and Facebook to build massive infrastructures to stream video at such a large scale. Because of this, these workloads must be optimized to ensure that our web infrastructure will be able to continue to handle the demands placed on it while efficiently using resources.

Unfortunately, raw video files tend to take up an enormous amount of data. For example, streaming a raw 1080p video at just 20 fps would take (1920 x 1080) of internet bandwidth, much more than is available to most consumers. Consequently, video codecs have become a critical part of the video processing flow, as they allow for a highly compressed version of the raw video content to be streamed to and from datacenters with relatively little noticeable degradation in video quality.

Numerous different video codecs exist for video compression, including H.264/AVC [5], H.265/HEVC [6], VP9 [7], offered by MPEG. Unfortunately, their latest offering, HEVC, carries a high cost and a great deal of uncertainty in order to license the patents necessary to implement it. In response, the Alliance for Open Media (AOM) [8] created a royalty-free codec, known as AV1, designed to rival the performance of other state of the art codecs, such as HEVC, while not violating any of their patents.

However, AV1 codecs overall generally consume significantly larger runtime than the other codecs. Fig. 1 shows the runtime of various encoding runs of the video game 1 at various Constant Rate Factor (CRF) values. CRF is an encoding parameter that dictates the quality of the encoded video, and lower CRF values correspond to better quality but higher runtime. From Fig. 1, we can see that at all tested CRFs, SVT-AV1 has by far the highest runtime of all

2835-2238/23/$31.00 ©2023 IEEE
DOI 10.1109/IISWC59245.2023.00021

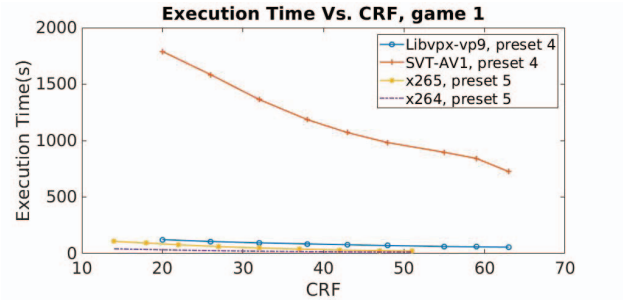


Figure 1: Execution time comparison between various codecs (a smaller CRF value indicates better quality video).

tested encoders. The reasons for the order of magnitude higher runtime are not well-understood. Unfortunately, the microarchitectural bottlenecks in the AV1 codec have yet to be identified in order to accelerate workloads using this codec and maximize their performance. It is critical that this work be done in order to allow video workloads to continue to be processed efficiently while avoiding the significant licensing costs and possible legal fees associated with other popular video codecs.

In this paper, we aim to discover why AV1 encoders take more time to encode than other modern encoders, such as H.264, H.265, or VP9. We perform top-down analysis [9], as well as a scalability study using SVT-AV1 [10].[1] We also found that branch prediction shows interesting behavior and perform detailed analysis using microarchitectural simulations.

This paper makes the following observations.

- Runtime of AV1 encoders such SVT-AV1 is higher than other encoders such as x264 and x265, largely not because they have any significant microarchitectural inefficiencies, but primarily because AV1 encoders need more work and thus require a larger number of instructions to encode the same video.
- The AV1 workloads only achieve 50-60% of the potential throughput achievable by the microarchitecture. Regardless of CRF and input video, the percentage of wasted pipeline slots is roughly 40-50 percent.
- As CRF decreases, the runtime of the encoder increases largely because of increasing instruction count. However, deteriorating microarchitectural inefficiency contributes to a smaller extent.
- Increasing the number of threads from 1 to 8 can improve performance by a factor of roughly 6X, making SVT-AV1 the most parallelizable of the encoders. x265, on the other hand, is the least parallelizable of the encoders, only experiencing a maximum speedup of roughly 1.3X.
- When increasing the number of threads for SVT-AV1, each core seems to perform roughly the same work, whereas when increasing the number of threads with x265, backend performance becomes much worse, which suggests that x265 may spread the workload among its cores unevenly.
- Increasing CRF causes backend performance to deteriorate relative to the performance of the rest of the pipeline. This happens because increasing at lower quality causes requires less computation to fall, but the amount of data movement stays the same, leading to more stalls due to cache misses.
- Increasing the speed preset from 0 to 8 causes a significant decrease in runtime and a significant increase in bitrate but a relatively modest decrease in Peak-Signal-to-Noise Ratio (PSNR).

1. We chose the SVT-AV1 codec for the analysis because this is the main codec used by Netflix and it supports various configurations, unlike rav1e [11].

2. Background and Motivation

2.1. Video Metrics

This section discusses the key video metrics used within this study. *Peak-Signal-to-Noise Ratio (PSNR)* is a widely-used video quality metric. PSNR is usually calculated per frame and measures the ratio between the maximum power of the signal and noise in that frame. Then, typically, the PSNR of each frame is averaged to find the PSNR of an entire video sequence [12] so that the quality of video as a result of using different encoders can be analyzed.

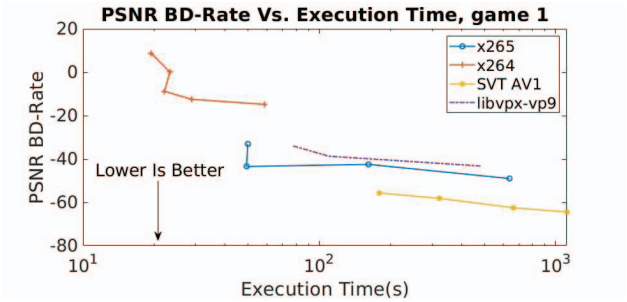
Bitrate is the number of bits per unit of time necessary to transmit an encoded video signal. All else being equal, it is desirable for an encoder to have a lower bitrate, as this means that the video can be transmitted faster, or using a weaker connection, and takes up less storage space. This paper measures bitrate in kilo-bits per second (kbps).

For most encoders, Constant Rate Factor (CRF) is a built in quality control parameter which specifies a certain quality that the encoder then aims to meet. Bitrate is adjusted up or down in order to meet the quality target [13]. Decreasing the value of CRF increases the quality target leading to increased video quality but at the cost of higher bitrate and runtime. Due to these adjustments, in order to get a full picture of the performance of an encoder it must be analyzed using multiple CRF values.

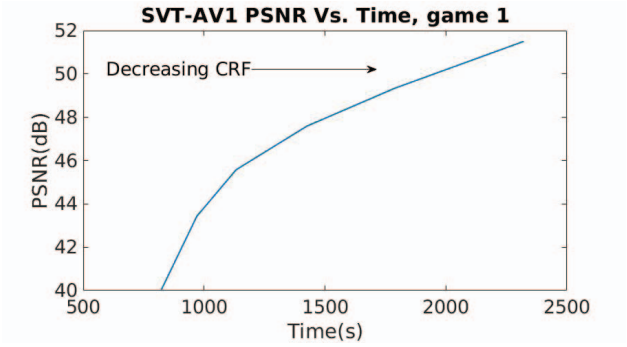
Bjøntegaard delta rate (BD-Rate) [14] measures the percent change in bitrate between two encoders or encoder configurations while maintaining the same video quality. BD-Rate is always calculated relative to some reference encoder or encoder configuration, which is treated as the zero point. In our case for PSNR BD-Rate, a lower PSNR BD-Rate value on the y-axis for some data points means that the encoder configuration corresponding to that data point can encode at the same video quality with a lesser bitrate. A lower PSNR BD-rate with a lower runtime is generally the most desirable, but, as can be seen in Fig. 2a, lower PSNR BD-Rate generally leads to higher runtime.

2.2. Video Encoding Overview

Video files are composed of a huge amount of data, and video streaming is becoming increasingly popular [15] [4], already contributing about 34% of consumer workloads



(a) PSNR BD-Rate vs Execution Time for game 1



(b) PSNR vs Execution Time for game 1. (Speed preset at 4)

Figure 2: PSNR vs Execution Time

[16]. In order to allow this abundance of video data to be easily streamed to the average consumer, this data must be modified to take up less bandwidth, and this is done via video encoding. Video encoding utilizes several techniques that aim to compress the size of a video signal to a smaller, more manageable size. This file compression reduces redundant information within the video data, which results in reduced video quality, so video encoding methods are designed to compress as much as possible while minimizing the loss in quality [17].

For any codec, compared to encoding, video decoding is a fairly straightforward operation because there exists only one valid decoding for each encoding method. Encoding is much more complex because the encoder needs to choose from many options at each encoding step to determine video quality (PSNR) and size (Bitrate). To find an acceptable encoding for meeting user's requirement, the encoder makes a number of passes through the video to search over the possible encoding options. In each new generation of codecs, the encoder is given more choices at each step, allowing for higher quality and/or lower bitrate. However, these improvements come at the cost of significantly increasing algorithmic complexity as increasing the number of options available to the encoder at each step exponentially increases the number of potential combinations of techniques the encoder can use to get to its final result. For example, AV1 allows 10 different ways to partition each block when encoding, whereas it's predecessor VP9 only allows for 4 [18]. Because of this, AV1 must check against 6 more options compared to VP9 at every step of encoding. If both were to investigate every option available to them for 5 steps, VP9 would have to investigate 1024 unique combinations of techniqes to encode, while AV1 would have to check 100000.

Any encoder will have a number of options which allow the user to control the desired Video Quality, Bitrate, and Runtime [19]. Many encoders allow the user to specify an input Constant Rate Factor (CRF). This essentially sets a target quality for the output, which the encoder then attempts to meet. Decreasing CRF tunes the encoder to encode for higher quality video output. This increase in quality lengthens the runtime, and causes the bitrate of the video output by the encoder to increase, requiring stronger connections to stream and more room required to store the output.

2.3. Motivation

AV1 provides an excellent quality and bitrate compared to other codecs [20], as demonstrated by the plot of PSNR BD-Rate in Fig. 2a. In the figure, SVT-AV1 is shown to have the lowest PSNR BD-rate out of all the encoders studied, and is therefore capable of encoding at a lower bitrate than any of the other encoders while retaining the same PSNR. However, it also possesses an order-of-magnitude or worse higher runtime than x265, x264 and VP9. Despite this, its high performance has gained AV1 encoders a great deal of attention, with many studies investigating practical applications for video codecs including AV1 as a main contender [21] [22] [23]. In order to investigate methods to potentially improve the runtime of encoders, while retaining low bitrates and high quality, this paper investigates which components of SVT-AV1 cause its high runtimes.

Additionally within SVT-AV1, encoding at better quality causes further increased runtime [24], as illustrated by the plot of quality metric PSNR plotted against runtime in Figure Fig. 2b. This figure shows that for the tested video, encoding at higher PSNR, and therefore higher quality, leads to significantly higher runtime with diminishing returns on quality. This figure only shows the relationship between PSNR and runtime for encoding the game 1 video, but this trend held for all but a few videos in vbench [25] which had significantly lower amount of uncertainty in their encoded data than their counterparts. Because of this, this paper also studies the causes of SVT-AV1's increase in runtime with video quality with the interest of finding potential areas to improve upon.

3. Methodology

3.1. Hardware

We use Intel Xeon E5 2650 V4 CPU for all of our analyses. This processor has 12 physical (24 logical) cores at 2.8 GHz. Each physical core contains a 32KB L1 I-cache, a 32KB L1 D-cache, and a 256KB L2 cache. The last-level cache (LLC) size is 30 MB shared across all cores.

3.2. Workloads

We use the videos from vbench [25] for our analyses. Vbench is a video benchmarking suite containing a set of 15 5 second long videos of varying resolutions, framerates, and complexities (measured as entropy). The videos in vbench are representative of a set of videos taken from Netflix, Xiph.org, and SPEC2017. The videos available in vbench can be seen in Table 1.

3.3. Codecs

Five codecs were used for the various analyses in this paper; `x264`, `x265`, `libaom`, `SVT-AV1` and `Libvpx-vp9`. The codecs each used one of two ranges of CRF and presets. `libaom`, `SVT-AV1` and `Libvpx-vp9` used a CRF range from 0-63, with higher values corresponding to lower quality and runtimes, along with a preset range from 0-8 that follows the same pattern. `x264` and `x265` on the other hand possess a CRF ranging from 0-51 and a preset range from 0-9. However it is important to note that the preset range of these two are measured in the opposite direction as the rest, with higher values corresponding to higher quality and runtimes.

3.4. Tools

We use the following tools for detailed architectural and microarchitectural analysis of video workloads:

1) Linux perf [26] provides a command line interface to extract CPU's performance event counters. Various microarchitectural statistics, such as the number of retired instructions, branch misses, and cache misses, are used for a top-down analysis [9].
2) We use Intel Pin [27], a binary instrumentation tool, to get instruction mixtures and record instruction traces for branch analyses.
3) The Championship Branch Prediction (CBP) framework [28] is used to evaluate the efficacy of various branch prediction units on our tested workloads.
4) GNU gprof [29] is used for a function level profiling, i.e., find hot functions, which is used for instruction tracing.

TABLE 1: The list of videos from vbench [25]

Video	Resolution	FPS	Entropy
desktop	720p	30	0.2
presentation	1080p	25	0.2
bike	720p	29	0.92
funny	1080p	30	2.5
bike	720p	29	0.92
cricket	720p	30	3.4
game1	1080p	60	4.6
game2	720p	30	4.9
game3	720p	59	6.1
girl	720p	30	5.9
chicken	2160p	30	5.9
cat	480p	29	6.8
holi	480p	30	7
landscape	1080p	29	7.2
hall	1080p	29	7.7

TABLE 2: Instruction mix in % (preset: 8, CRF: 63).

Video	# Insts.	Branch	Load	Store	AVX	SSE	Other
presentation	1.7E+11	6.6	25.8	13.1	33.0	0.9	20.7
landscape	4.4E+11	6.2	26.3	13.6	31.5	0.7	21.7
house	3.4E+11	6.0	26.3	12.9	32.0	0.7	22.1
holi	4.4E+11	3.3	29.4	15.5	33.7	0.2	18.0
hall	3.5E+11	5.7	26.5	13.4	32.6	0.6	21.2
girl	1.2E+11	6.7	25.8	14.3	29.9	0.8	22.5
game2	1.5E+11	6.6	26.2	14.1	29.7	0.8	22.6
game3	2.6E+11	6.6	26.1	13.6	30.4	0.8	22.5
funny	9.5E+11	4.0	28.8	14.8	33.0	0.2	19.1
desktop	9.5E+10	6.9	26.0	14.1	29.6	1.0	22.5
cricket	7.8E+11	3.3	29.2	15.5	34.2	0.2	17.6
cat	7.7E+10	5.9	26.1	13.7	30.3	0.6	23.3
bike	9.9E+10	6.8	26.0	14.3	29.2	0.9	22.8

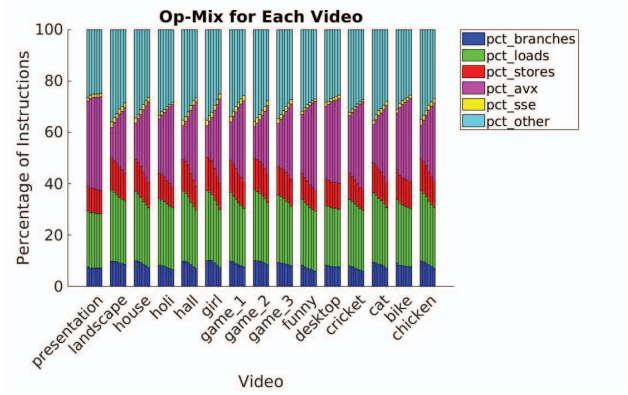


Figure 3: Op-Mix for each video. CRF increases from left to right in each cluster of bar graphs.

4. results

4.1. Instruction Mix

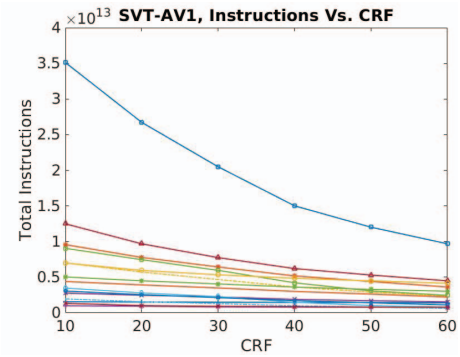
Table 2 shows the instruction mix from a number of runs of the encoder for each video at preset 8 and CRF 63. From this data, we can see that `SVT-AV1` is well vectorized. This suggests that further vectorization is probably not a viable approach to speed up the encoder. Additionally, the encoding workloads contain a relatively small percentage of branch instructions, ranging from 3.3% to 6.9% branches. The percentage of load instructions is relatively higher ranging from 25.8% to 29.4%. The percentage of stores ranges from 12.9% to 15.5%. CRF was increased from 10 to 60 in Fig. 3 to understand any noticeable change, and indeed, the AVX instruction mix shows a growing percentage of instructions, further signifying the important performance implication of vectorizations.

4.2. CRF Sweep Results

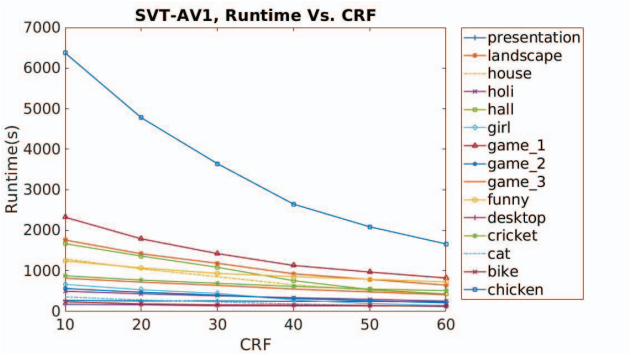
This section examines how runtime and other microarchitectural behaviors change as the CRF value increases.

4.2.1. Runtime. As shown in Fig. 1, `SVT-AV1` has a significantly larger number of instructions for the same CRF than other codecs, including `Libvpx-vp9`, `x264`, and `x265`. This

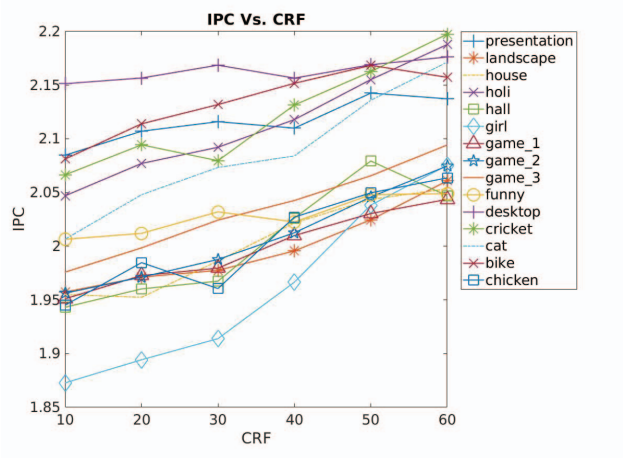
greater number of instructions can be explained by the fact that AV1 allows for many more options at each encoding step. These additional options yield performance increases in terms of PSNR and bitrate, but they exponentially increase runtime as they exponentially increase the design space that the encoder is forced to search to find an acceptable encoding of the video. This observation points towards a viable area to improve the runtime of `SVT-AV1` further.



(a) Instruction count



(b) Runtime



(c) IPC

Figure 4: CRF sweep results (speed preset 4).

Fig. 4 shows changes in the instruction count, execution time, and instruction per cycle (IPC) as we vary CRF from 10 to 60. The following observations can be made from the results:

1) The runtime seems to be directly proportional to the number of instructions.
2) Additionally, Fig. 4c, shows that as CRF is increased from 10 to 60, IPC hovers around 2 for all of the videos and increases by at most 10% depending on the video. This increased IPC, while not insignificant, is not enough to explain the much larger decrease in runtime with larger CRFs and the number of instructions required to perform the encoding fall.

These observations suggest that as CRF increases, the runtime of the encoder decreases not primarily because increasing CRF leads to increased microarchitectural efficiency but because increasing CRF simply decreases the amount of algorithmic work required by the encoder.

4.2.2. Top-down Analysis. Top-down analysis [9] is a useful method to quickly identify performance bottlenecks in a processor using performance counters. Pipeline activities are classified as one for the following four: 1) retiring: micro-ops (uops) are completed without a problem, 2) bad speculation: this category is caused by bad speculations, i.e., branch misprediction, and includes wrong-path instructions that are eventually flushed and pipeline stalls due to pipeline flush, 3) frontend: uops are not sufficiently supplied to the backend, and 4) backend: because of resource constraints, such as data cache misses and busy execution units, uops cannot be issued for execution.

Fig. 5 shows a top-down analysis of videos used in this study. Note that each video is represented by a cluster of stacked bar graphs. One trend observed was that the backend slots account for more wasted pipeline slots than the frontend and bad-speculation for almost all tested videos. In addition, increasing CRF tends to increase the overall proportion of backend-bound slots but decrease the proportion of frontend-bound slots. Interestingly, the sum of the percentages of frontend and backend bound slots stays relatively constant regardless of CRF changes for all videos.

On the other hand, the percentage of bad-speculation-bound slots decreases with increasing CRF, causing the percentage of retiring slots to increase on the whole with increasing CRF. However, the magnitude of this increase is fairly limited as the percentage of bad-speculation-bound slots is relatively small for all videos. Additionally, the percentage of retiring slots is fairly similar across all videos ranging from 0.4 to 0.6. As a result, the overall IPC of the runs was limited to around 2[2], as shown in Fig. 4c. This result makes sense as the IPC of the workload is typically expected to be roughly equivalent to the reciprocal of the percentage of retiring slots. The falling percentage of bad-speculation-bound slots with increasing CRF contributes to IPC improvements. Bad-speculation-bound slots show

2. The maximum IPC of the evaluated machine is 4.

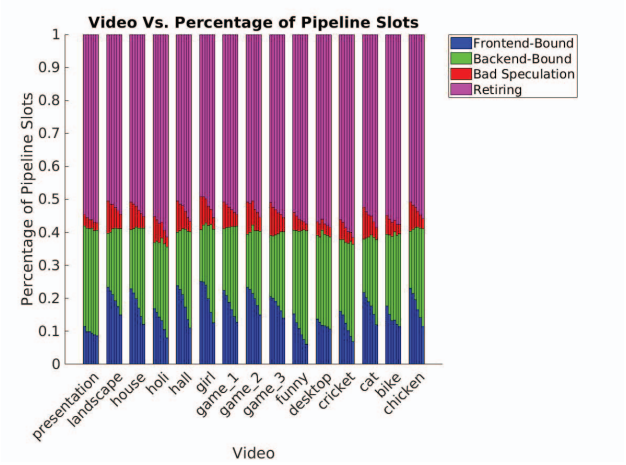


Figure 5: Top-down analysis for each video. CRF increases from left to right in each cluster of bar graphs.

the most interesting results in this section, so further investigation of the branch prediction behavior performed, as shown in Section 4.4.

4.3. Microarchitectural Analysis

Among wasted slots in Section 4.2.2, backend-bound slots can be divided into memory-bound and core-bound slots. Memory-bound slots consist of wasted slots due to cache misses. Core-bound slots are due to the required pipeline resources being unavailable. Frontend-bound slots can typically be divided into latency bound and bandwidth bound. Bad-speculation-bound slots are generally due to branch mispredictions, but both of these had a relatively smaller impact than the backend-bound slots. In order to investigate these wasted slots further, the inefficiency within several microarchitectural resources that may be contributing to these issues was examined. The specific issues investigated were the inefficiency within the branch predictor, the cache hierarchy, and some of the pipeline execution units. Fig. 6 shows all results.

Fig. 6a shows the trend in branch misses per kilo instructions (MPKI) as CRF increases. It displays as CRF increases, the number of branch MPKI decreases significantly. This makes sense as previously we saw that bad-speculation bound slots tend to decrease with increasing CRF. Fig. 6b, Fig. 6c, and Fig. 6d show us the trends in L1D, L2, and LLC MPKI, respectively. From these figures, we can see that generally speaking as CRF increased, cache performance tended to deteriorate. These cache misses account for the majority of the memory bound portion of the workload's backend bound slots. This trend was not mirrored by the LLC, however, the LLC accounted for many fewer misses per kilo instruction than the L1D and L2 Caches. Figures Fig. 6e, Fig. 6f, Fig. 6g, and Fig. 6h show us the trends in inefficiency in the pipeline units as CRF increases. As CRF increases, performance for these units tended to deteriorate except for the reorder buffer which accounted for many fewer stall cycles than the other pipeline units. These are similar to the trends found in [30].

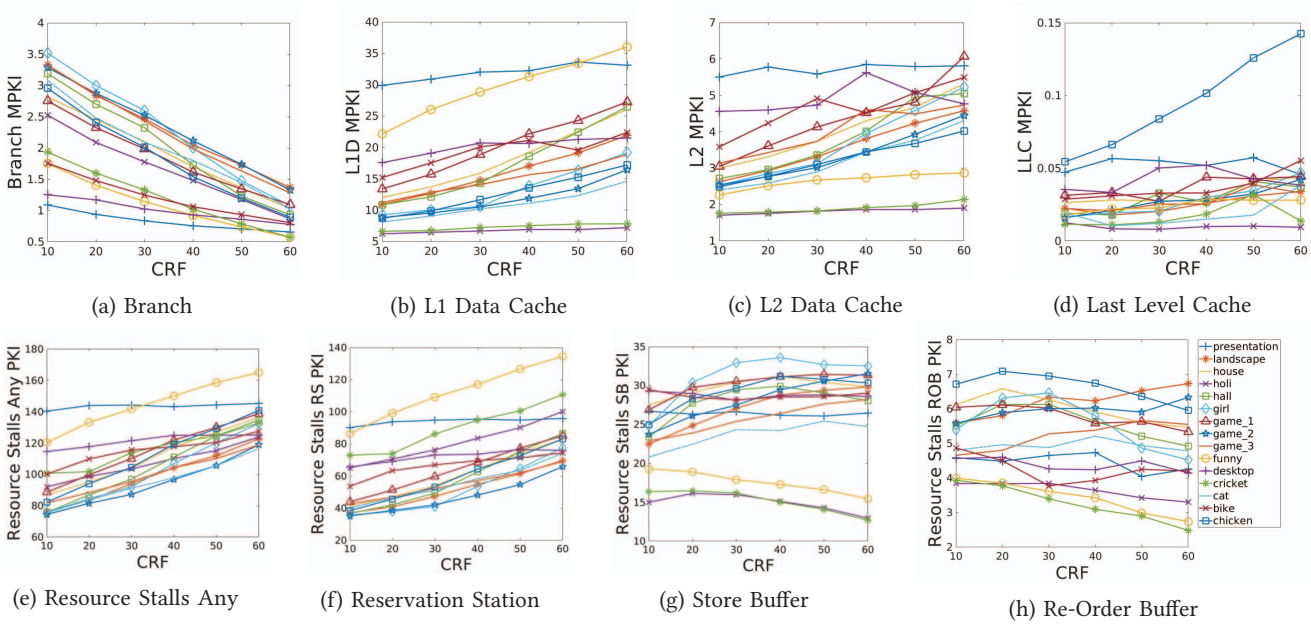
As suggested in [30], the trends in memory boundedness can be explained using the roofline model [31]. The roofline model is a performance model which shows the correlation between performance and operational intensity. The roofline model justifies this by suggesting that if the operational intensity is low enough for the workload to be memory bound, overall performance will increase as operational intensity increases until the workload becomes compute bound. This concept is paralleled here, as when CRF increases, the quality constraint on the encoder is relaxed, which means that less computation is needed to encode the video. On the other hand, the total amount of required data transfer stays the same, which means that operational intensity decreases, and the workload becomes more memory bound.

This trend can also explain the decrease in frontend-boundedness as CRF increases. As Operational intensity decreases due to the increase in CRF, the CPU spends more time waiting for memory traffic which causes it to exhaust pipeline resources. When this happens, the CPU must stop fetching new instructions, causing it to spend fewer cycles stalled due to frontend issues.

4.4. Branch Prediction Analysis

As displayed in Fig. 7, despite the encoder's low branch MPKI, the branch miss rate of the encoder runs is fairly high, around 3.5% for some of the tested examples. Additionally, as displayed in Fig. 6a, and Fig. 4c decreasing branch MPKI seems to have a strong correlation with increasing IPC. This suggests that optimizing branch prediction may provide a significant performance improvement. Because of this, the effectiveness of various branch predictors on branch traces taken from encoding runs of each of the videos is analyzed.

Fig. 8, Fig. 9, and Fig. 10 show the branch miss rate and MPKI of various branch predictors on branch traces taken from encoding runs of SVT-AV1 on each video within vbench. These traces were taken from an interval of 1 billion instructions roughly halfway through the encoding run of each video with CRF set to 63 and preset set to 8. These traces were collected using Pin [27]. The branch traces were evaluated using the CBP 2016 branch prediction simulator [28]. With this simulator, 2 Gshare-based branch predictors were simulated [32], one with a size of 2KB, and one with a size of 32KB, as well as the 8KB and 64KB TAGE-based branch predictors detailed in [33]. From this figure, it can be noticed that increasing the size of the branch predictor for both TAGE and Gshare significantly decreased the simulated branch miss rate. Additionally, the TAGE-based branch predictor performed much better on the trace than the Gshare-based branch predictor. Along with other findings, this suggests that optimizing branch prediction performance by increasing the size of the branch predictor, as well as opting for a more complicated branch prediction scheme such as TAGE rather than Gshare may significantly increase the overall IPC of the workload.



(a) Branch (b) L1 Data Cache (c) L2 Data Cache (d) Last Level Cache

(e) Resource Stalls Any (f) Reservation Station (g) Store Buffer (h) Re-Order Buffer

Figure 6: Microarchitectural analysis with CRF changes.

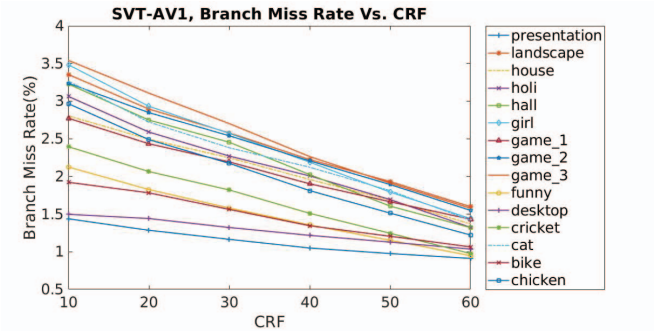


Figure 7: Branch Miss Rate vs CRF

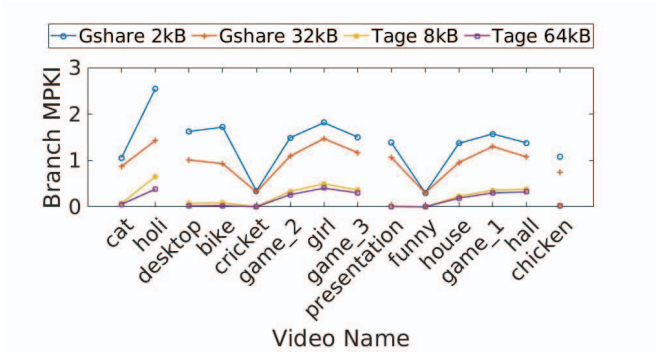


Figure 8: Simulated MPKI for each video. Branch traces were collected using speed preset 8 and CRF 63

4.5. Preset Sweep

Fig. 11b displays the relationship between encoding time, bitrate, and PSNR as speed preset is varied. Fig. 11a shows that as speed preset is increased from 0 to 8 with fixed CRF, runtime fell significantly from over 155k seconds at preset 0 to fewer than 200 seconds at preset 8. Fig. 11b demonstrates that with a fixed CRF, as preset is increased from 0 to 2 bitrate did not significantly increase, however from preset 3 onward, bitrate raised significantly from roughly 2700 kb/s to roughly 3200kb/s. PSNR, on the other hand, showed only a relatively modest decrease as speed preset was increased from 0 to 8, falling from roughly 45.8dB to roughly 45dB. This suggests that with fixed CRF, most of the encoding time reduction is time that was spent reducing the size of the encoded video rather than increasing the quality. In Fig. 11c, Fig. 11d, and Fig. 11e, the trends in percentage of pipeline slot usage, branch/cache MPKI, and resource stalls as preset is increased with fixed CRF. Unlike in [30], no noticeable trends were observed in any of these statistics as preset is increased from 0 to 8 with constant CRF.

4.6. Thread Scalability Study

Most video encoding algorithms have sufficiently large parallelism. For example, multiple threads can process different partitions, i.e., macro-blocks, in parallel to speed up the encoding process. Consequently, thread scalability is an important aspect of video codecs. This section examines how multi-threading affects the performance of different codecs, including `x264`, `x265`, `libaom`, and `SVT-AV1`.

Fig. 12, Fig. 13, Fig. 14, and Fig. 15 show the thread scalability results as the number of concurrently running threads increases from 1 to 8. From one to three threads, `SVT-AV1` shows similar speedup to `libaom` but significantly less speedup than `x264`. However, from four threads and

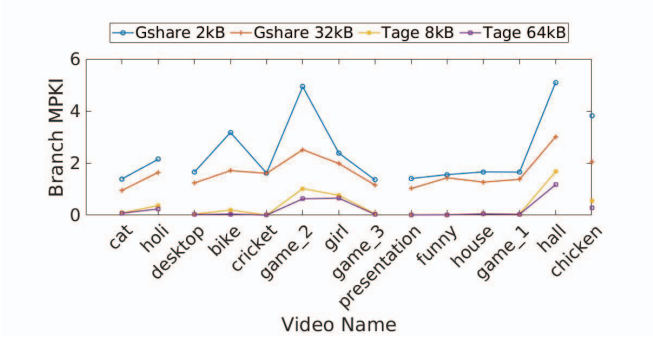


Figure 9: Simulated MPKI for each video. Branch traces were collected using speed preset 4 and CRF 10

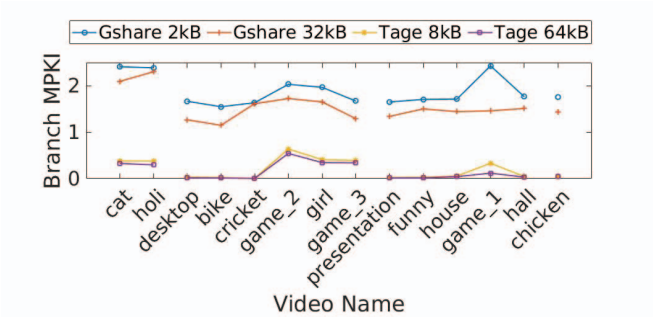


Figure 10: Simulated MPKI for each video. Branch traces were collected using speed preset 4 and CRF 60

onward, SVT-AV1 exhibits by far the best speedup, reaching roughly 6x speedup compared to the single-core runtime with 8 threads. Interestingly, x265 shows by far the poorest speedup of the four encoders, reaching a maximum speedup of about 1.3x. We conduct a top-down analysis to investigate why encoders show different scalability behaviors.

Fig. 16 shows that how much percentage of pipeline slots are wasted by the backend, frontend, and bad-speculation for all four encoders. For libaom, SVT-AV1, and x264, changing the maximum number of concurrent threads does not affect workload characteristics. x265, on the other hand, becomes significantly more backend bound with the increased number of the maximum concurrent threads. This observation suggests that every encoder other than x265 does a fairly good job of dividing the work amongst additional threads as new threads become available, whereas x265 may instead divide it amongst a primary thread which performs most of the work along with some additional helper threads.

5. Related Work

Realizing the significance of video workloads, a significant set of prior works have been done to profile them and examine their performance.

Lottarini et al. [34] formulated vbench, a representative set of video inputs based on their resolution, frame rate, and entropy. It proposes a set of videos to be used as a benchmark for transcoding applications and shows its usefulness by comparing the microarchitectural trends found when executing the new benchmark suite to existing video datasets. We use the videos found in vbench to conduct our analyses. While the paper does investigate some of the microarchitectural trends of the videos within the vbench suite, its primary purpose for doing this is to compare the trends to the those found within the larger dataset of videos in order to argue that the set of videos within vbench is representative of the larger set. The analysis in this paper does not seek to identify any performance bottlenecks or analyze the differences between various codecs. We would like to investigate this in our paper.

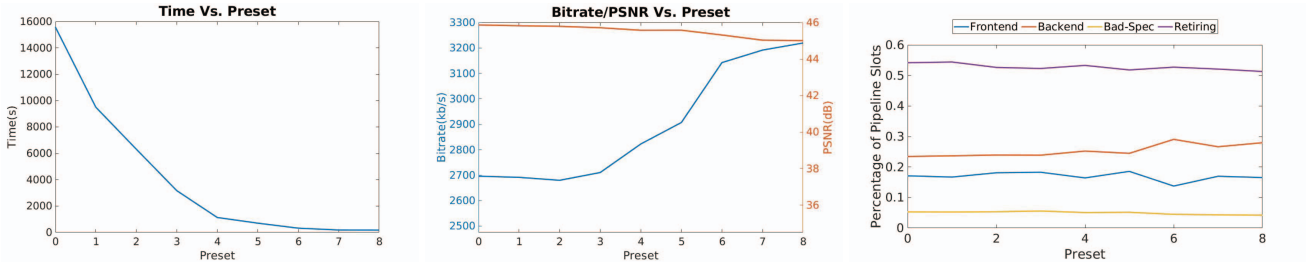
Chen et al. [35] uses the videos from vBench to do microarchitectural performance characterization of video transcoders. They evaluate the performance of the x264 transcoder by varying its parameters like CRF (constant rate factor), refs(reference frame number), and presets. The paper, however, does not mention the AV1 codec which we would like to investigate.

Mansri et al. [20] tried to compare the performance of various state of the art codecs with their predecessors. Additionally, the paper compares the performance of the emerging AV1 codec with some more established codecs. While this paper does discuss AV1 performance, it does not do anything to evaluate any microarchitectural trends found in the AV1 codec or any of the other codecs that it examines.

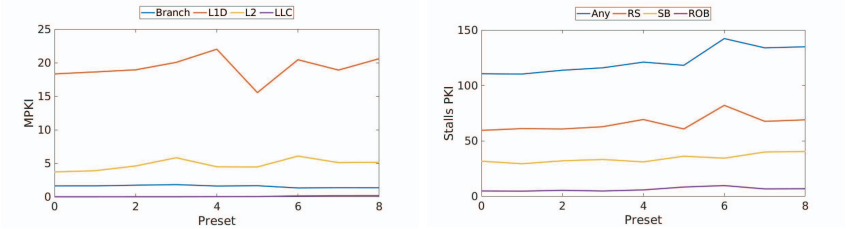
Kossentini et al. [36] discussed various algorithmic features the SVT-AV1 encoder that implements the AV1 codec and investigates its speed and encoding quality tradeoffs. The paper compares the speed vs quality tradeoffs of SVT-AV1 vs. several other encoders implementing various other codecs for various use cases such as premium VOD, and general VOD. The paper finds that SVT-AV1's speed vs quality characteristics put it on the pareto front of the set of tested encoders, but the paper does not focus on the microarchitectural characteristics of any of the tested encoders. Additionally, the paper does not attempt to identify any opportunities to increase the performance of the SVT-AV1 workload.

6. Conclusions

In this paper, we characterize the microarchitectural performance of AV1 video-encoding workloads using the AV1 encoder SVT-AV1. We compare the performance of this encoder against the performance of various other encoders implementing other popular codecs. We find that SVT-AV1's long runtime relative to other popular encoders is not primarily attributable to its microarchitectural performance relative to the other encoders but rather its much greater algorithmic complexity. Similarly, SVT-AV1's increase in runtime as the quality constraint is increased is also largely due to increased algorithmic complexity rather than deteriorating microarchitectural performance. Additionally, we find that improving branch prediction performance



(a) Runtime (b) Bitrate, PSNR (c) % Pipeline Slot Usage



(d) Branch, Cache MPKI (e) Resource Stalls

Figure 11: Preset sweep results for the video game 1.

alone may account for a roughly 10% increase in IPC for all videos over our CRF sweep. We demonstrate that increasing the size and complexity of the branch predictor significantly improves branch-predictor performance on traces of branch instructions taken directly from SVT-AV1 workloads, and therefore may significantly improve SVT-AV1's performance. Finally, we show that despite being the slowest of the tested encoders, SVT-AV1 is also the most parallelizable, achieving a speedup of roughly 6x as the maximum number of concurrent threads was increased from

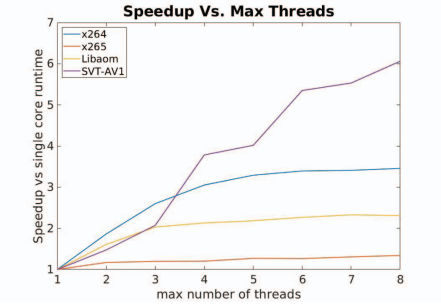


Figure 12: Thread scalability results For the video game 1. highest CRF. X264 preset = 0, CRF = 51

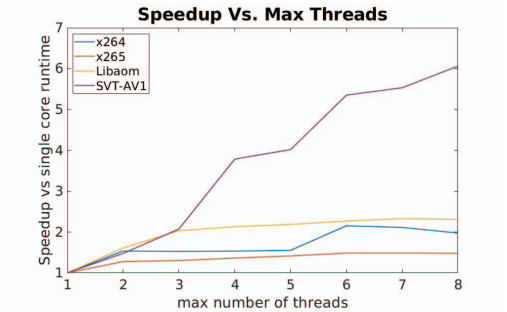


Figure 13: Thread scalability results For the video game 1. highest CRF. X264 preset = 2, CRF = 51

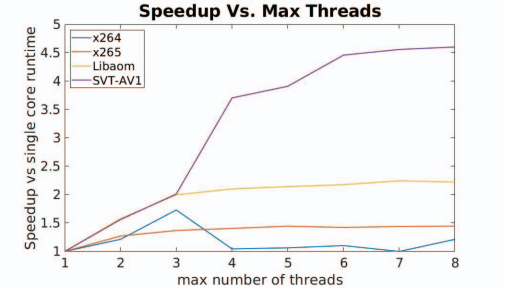


Figure 14: Thread scalability results For the video game 1. X264 preset = 5, CRF = 50.

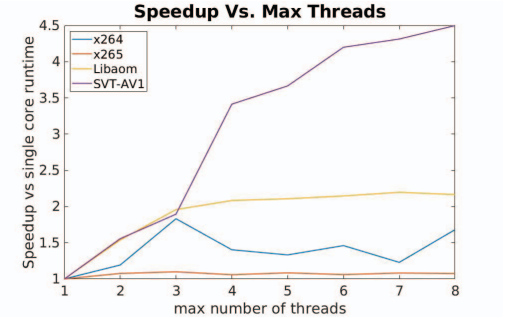
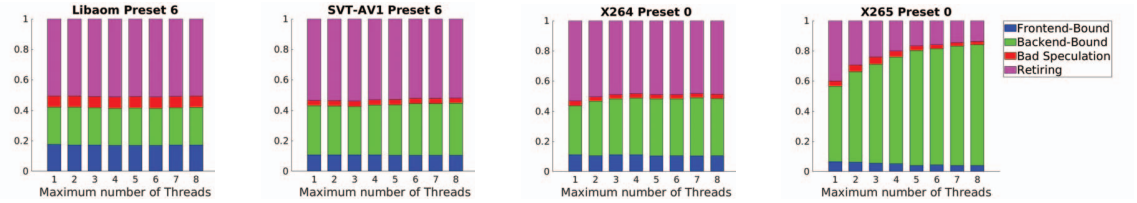


Figure 15: Thread scalability results For the video game 1. X264 preset = 5, CRF = 30.



(a) libaom (b) SVT-AV1 (c) x264 (d) x265

Figure 16: Top-down analysis with different number of threads for the video game 1.

1 to 8. x265, on the other hand, was found to be the least parallelizable, achieving a speedup of only roughly 1.3x. These observations may in the future help to bridge the gap in CPU runtime between SVT-AV1 and other modern encoders.

7. Acknowledgement

We thank anonymous reviewers for their constructive feedback and insightful comments. This research was supported in part by NSF grant number 1763848, a gift from ARM, the iMAGINE Consortium, and the Texas Advanced Computing Center (TACC). The authors would also like to acknowledge the computing servers donated by Ampere and Nvidia. Any opinions, findings, conclusions, or recommendations are those of the authors and not of the National Science Foundation or other sponsors.

References

[1] Lights, camera, engagement: 2017 is the year of video marketing. [Online]. Available: https://www.forbes.com/sites/ajagrawal/2017/02/01/lights-camera-engagement-2017-is-the-year-of-video-marketing/?sh=44e5fa8c2315

[2] Cisco visual networking index forecast 2017-2022. [Online]. Available: https://newsroom.cisco.com/press-release-content?articleId=1955935

[3] What employees are saying about the future of remote work. [Online]. Available: https://www.mckinsey.com/business-functions/people-and-organizational-performance/our-insights/what-employees-are-saying-about-the-future-of-remote-work

[4] InVideo. (2021) 135 video marketing statistics you can't ignore in 2021. https://invideo.io/blog/video-marketing-statistics.

[5] T. Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the h.264/avc video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, 2003.

[6] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (hevc) standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649–1668, 2012.

[7] D. Mukherjee, J. Bankoski, A. Grange, J. Han, J. Koleszar, P. Wilkins, Y. Xu, and R. Bultje, "The latest open-source video codec vp9 - an overview and preliminary results," in *2013 Picture Coding Symposium (PCS)*, 2013, pp. 390–393.

[8] A. for Open Media. (2015) An alliance of global media innovators. https://aomedia.org/.

[9] A. Yasin, "A top-down method for performance analysis and counters architecture," in *2014 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2014, pp. 35–44.

[10] A. Norkin, J. Sole, K. Swanson, M. Afonso, A. Moorthy, and A. Aaron. (2019) Introducing svt-av1: a scalable open-source av1 framework. https://netflixtechblog.com/introducing-svt-av1-a-scalable-open-source-av1-framework-c726cce3103a.

[11] The fastest and safest av1 encoder. [Online]. Available: https://github.com/xiph/rav1e

[12] A. T. Nasrabadi, M. A. Shirsavar, A. Ebrahimi, and M. Ghanbari, "Investigating the psnr calculation methods for video sequences with source and channel distortions," in *2014 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting*, 2014, pp. 1–4.

[13] Saving on h.264 encoding and streaming: Deploy capped crf. [Online]. Available: https://streaminglearningcenter.com/encoding/saving-encoding-streaming-deploy-capped-crf.html

[14] G. Bjontegaard, "Calculation of average psnr differences between rd-curves," *VCEG-M33*, 2001.

[15] S. Lee, S. Lee, H. Joo, and Y. Nam, "Examining factors influencing early paid over-the-top video streaming market growth: A cross-country empirical study," *Sustainability*, vol. 13, no. 10, p. 5702, 2021.

[16] "Cisco global cloud index: Forecast and methodology, 2016-2021," 2018.

[17] V. Sze, M. Budagavi, and G. J. Sullivan, *High Efficiency Video Coding (HEVC) algorithms and Architectures*. Springer International Publishing, 2014.

[18] Y. Chen, D. Murherjee, J. Han, A. Grange, Y. Xu, Z. Liu, S. Parker, C. Chen, H. Su, U. Joshi, C.-H. Chiang, Y. Wang, P. Wilkins, J. Bankoski, L. Trudeau, N. Egge, J.-M. Valin, T. Davies, S. Midtskogen, A. Norkin, and P. de Rivaz, "An overview of core coding tools in the av1 video codec," in *2018 Picture Coding Symposium (PCS)*, 2018, pp. 41–45.

[19] MozDevNet, "Web video codec guide - web media technologies: Mdn," May 2023. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/Media/Formats/Video_codecs

[20] I. Mansri, N. Doghmane, N. Kouadria, S. Harize, and A. Bekhouch, "Comparative evaluation of vvc, hevc, h.264, av1, and vp9 encoders for low-delay video applications," in *2020 Fourth International Conference on Multimedia Computing, Networking and Applications (MCNA)*, 2020, pp. 38–43.

[21] P. Topiwala and W. Dai, "Hdr video coding for aerial videos with vvc and av1," *Applications of Digital Image Processing XLIV*, vol. 11842, Aug 2021.

[22] A. S. Panayides, M. S. Pattichis, M. Pantziaris, A. G. Constantinides, and C. S. Pattichis, "The battle of the video codecs in the healthcare domain - a comparative performance evaluation study leveraging vvc and av1," *IEEE Access*, vol. 8, p. 11469–11481, Jan 2020.

[23] F. Kyslov, M. Paniconi, J. Jiang, Y. Wang, and C. Y. Tsai, "Optimizing av1 encoder for real-time communication," *2022 IEEE International Conference on Image Processing (ICIP)*, Oct 2022.

[24] L. Roux and A. Gouaillard, "Performance of av1 real-time mode," *2020 Principles, Systems and Applications of IP Telecommunications (IPTComm)*, Oct 2020.

[25] Vbench. [Online]. Available: http://arcade.cs.columbia.edu/vbench/

[26] Perf wiki. [Online]. Available: https://perf.wiki.kernel.org/index.php/Main_Page

[27] C.-K. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V. J. Reddi, and K. Hazelwood, "Pin: Building customized program analysis tools with dynamic instrumentation," in *Proceedings of the 2005 ACM SIGPLAN Conference on Programming Language Design and Implementation*, ser. PLDI '05. New York, NY, USA: Association for Computing Machinery, 2005, p. 190–200. [Online]. Available: https://doi.org/10.1145/1065010.1065034

[28] Championship branch prediction 2016 simulator. [Online]. Available: https://jilp.org/cbp2016/framework.html

[29] Gnu gprof. [Online]. Available: https://ftp.gnu.org/old-gnu/Manuals/gprof-2.9.1/html_mono/gprof.html

[30] Y. Chen, J. Zhu, T. A. Khan, and B. Kasikci, "Cpu microarchitectural performance characterization of cloud video transcoding," in *2020 IEEE International Symposium on Workload Characterization (IISWC)*, 2020, pp. 72–82.

[31] S. Williams, A. Waterman, and D. Patterson, "Roofline: An insightful visual performance model for multicore architectures," *Commun. ACM*, vol. 52, no. 4, p. 65–76, Apr. 2009. [Online]. Available: https://doi.org/10.1145/1498765.1498785

[32] S. Mcfarling, "Combining branch predictors," in *Technical Report TN-36, Digital Western Research Laboratory*, June 1993.

[33] A. Seznec, "Tage-sc-l branch predictors again," 2016.

[34] A. Lottarini, A. Ramirez, J. Coburn, M. Kim, P. Ranganathan, D. Stodolsky, and M. Wachsler, "vbench: Benchmarking video transcoding in the cloud," *ACM SIGPLAN Notices*, vol. 53, pp. 797–809, 03 2018.

[35] Y. Chen, J. Zhu, T. A. Khan, and B. Kasikci, "Cpu microarchitectural performance characterization of cloud video transcoding," in *2020 IEEE International Symposium on Workload Characterization (IISWC)*, 2020, pp. 72–82.

[36] F. Kossentini, H. Guermazi, N. Mahdi, C. Nouira, A. Naghdinezhad, H. Tmar, O. Khlif, P. Worth, and F. Ben Amara, "The SVT-AV1 encoder: overview, features and speed-quality tradeoffs," in *Applications of Digital Image Processing XLIII*, ser. Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, vol. 11510, Aug. 2020, p. 1151021.