IN THE UNITED STATES DISTRICT COURT
DISTRICT OF DELAWARE

DOLBY VIDEO COMPRESSION, LLC,	**Civil Action No.**
Plaintiff,	
v.	**JURY TRIAL DEMANDED**
SNAP INC.,	
Defendant.	

ORIGINAL COMPLAINT

Plaintiff Dolby Video Compression LLC ("Dolby," "DVC," or "Plaintiff") files this Original Complaint against Snap Inc. ("Snap," or "Defendant") and alleges as follows:

NATURE OF THE ACTION

1. For nearly 60 years, Dolby has been a leader in innovating and delivering cutting-edge technologies that transform the science of sight and sound into spectacular experiences for billions of people worldwide. Dolby invents and licenses advanced audio and imaging technologies, including video coding technologies, to improve the ways people experience media and communicate with each other. Dolby, along with its parent company, holds over 22,000 patents worldwide, including a large portfolio relating to video encoding and decoding.

2. Snap uses video coding technology as part of its Snapchat product, which allows users to share videos with each other and the public. Snap performs video coding to make these videos available on a wide range of devices. Snap's "Accused Services" are backend processes, including for example transcoding and encoding for the purpose of providing videos to users, wherein such processes infringe one or more claims in Dolby's Asserted Patents (as defined below).

3. Snap's Accused Services rely on Dolby's innovations to efficiently deliver high-quality video to users throughout the United States and the world. Dolby's patented technology is critical to Snap's business, driving the efficiency and quality of the videos that help keep users engaged on the application. While other companies have recognized and validated the strength of Dolby's video patent portfolio by taking licenses, Snap continues to use Dolby's patented technology without a license, thereby obtaining an unfair competitive advantage. Dolby brings this lawsuit to end Snap's unlicensed use of Dolby's patented technology.

PARTIES

4. Plaintiff Dolby Video Compression LLC ("DVC") is a limited liability company organized under the laws of Delaware. DVC is the sole owner by assignment of all right, title, and interest in U.S. Patent Nos. 10,855,990; 9,924,193; 9,596,469; and 10,404,272 (the "Asserted Patents").

5. On information and belief, Snap Inc. is a corporation organized under the laws of the State of Delaware.

6. On information and belief, Snap Inc. has its principal place of business at 2772 Donald Douglas Loop N, Santa Monica, CA 90405, and 3000 31st Street, Santa Monica, CA 90405.

JURISDICTION AND VENUE

7. This Court has exclusive subject matter jurisdiction over the patent infringement claims in this case under 28 U.S.C. §§ 1331 and 1338.

8. The Court also has supplemental jurisdiction over all claims other than the patent infringement claims in this case under 28 U.S.C. § 1367(a), including over Count V (declaratory judgment that Dolby has complied with its RAND commitments). An actual controversy over all counts exists between the parties to this case.

9. This Court has general personal jurisdiction over Snap Inc. by virtue of Snap Inc.'s incorporation in Delaware. Snap has appointed a registered agent for service of process: Corporation

Service Company, 251 Little Falls Drive, Wilmington, Delaware, 19749.

10. This Court has specific personal jurisdiction over Snap because Snap has, directly and/or through agents and/or intermediaries, committed acts and continues to commit acts of patent infringement, including within Delaware, giving rise to this action and has established minimum contacts with Delaware such that the exercise of jurisdiction would not offend traditional notions of fair play and substantial justice.

11. Snap, among other things, uses the infringing Accused Services to decode videos submitted to the Snapchat platform from submitters around the world, including in Delaware; to encode those videos in suitable formats for distribution; and to place one or more encoded video bitstreams into the stream of commerce over the internet with the knowledge and/or understanding that such Accused Services were being utilized for the purpose of offering services and videos in Delaware.

12. On information and belief, Snap regularly conducts and has conducted business in Delaware and purposefully avails itself of the privileges of conducting business in Delaware. On information and belief, Snap, and/or its agents and/or intermediaries, makes, uses, imports, offers for sale, sells, and/or advertises its products and affiliated services in Delaware, sufficient to give rise to jurisdiction.

13. On information and belief, Snap derives and has derived substantial revenue from the Accused Services within Delaware, and/or expects or should reasonably expect its actions to have consequences in Delaware.

14. Snap's infringing activity has led to foreseeable harm and injury to Dolby.

15. Venue is proper under 28 U.S.C. § 1391 and 28 U.S.C. § 1400(b). As set forth above, Snap has committed acts of infringement in this District and is incorporated in this District.

16. This action involves several Delaware companies, including both Parties. In addition, DVC

is a wholly owned subsidiary of Dolby Laboratories, Inc. ("DLI"), which is also organized under the laws of Delaware.

17. Access Advance LLC ("Access Advance") is a patent pool administrator that on information and belief is organized as a Delaware limited liability company with a principal place of business at 100 Cambridge Street, Suite 21400, Boston, MA 02114. DVC licenses its patents bilaterally and through pools administered by Access Advance.

I. <u>DOLBY'S TECHNOLOGY</u>

18. Dolby is a leader in innovating and delivering cutting-edge technologies that enable the creative community and that empower artists and content distributors with new and improving ways to convey rich entertainment experiences to their audiences.

19. From movies and TV shows, to apps, music, sports, and gaming, Dolby transforms the science of sight and sound into spectacular experiences for billions of people worldwide. Dolby partners with creatives, developers, and businesses to revolutionize entertainment and communications with Dolby Atmos, Dolby Vision, Dolby Cinema, and Dolby OptiView. Dolby invents and licenses advanced audio and imaging technologies to improve the ways people experience media and communicate with each other. Each year, billions of devices and services rely on technologies supported through Dolby patent licensing. By aligning standards and simplifying access, these technologies reach the market faster and work more seamlessly across platforms and products worldwide.

20. Dolby, along with its parent company, holds over 22,000 patents worldwide. This includes a large portfolio of patents relating to video coding.

II. <u>SNAP'S USE OF VIDEO CODING</u>

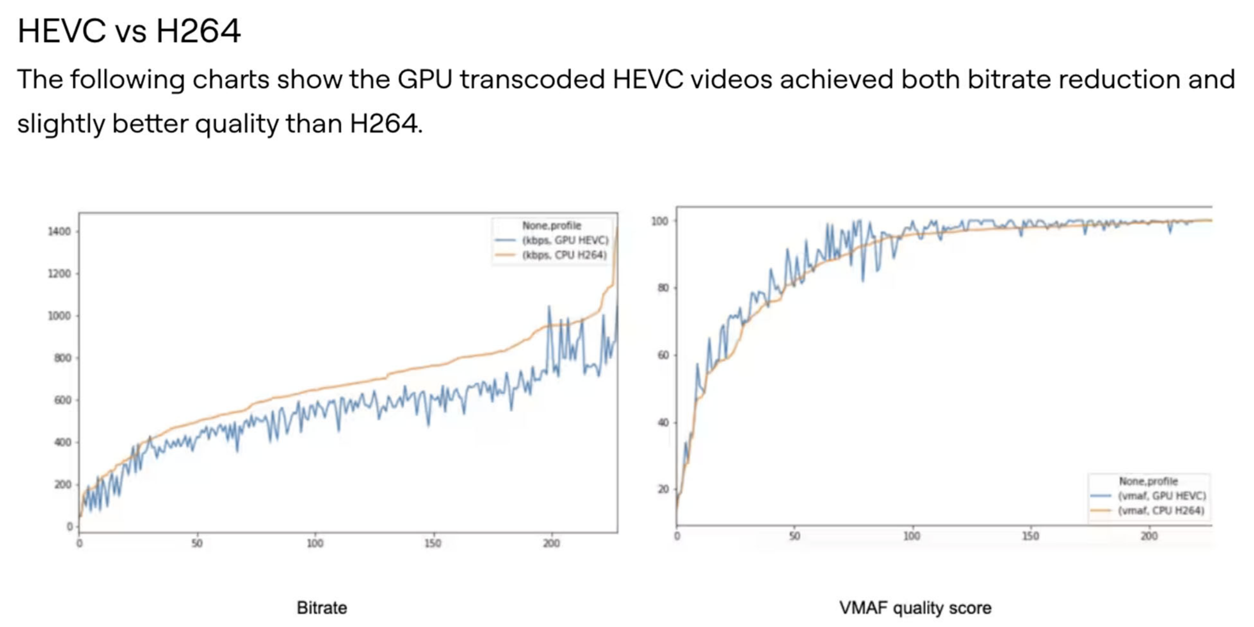
21. Snap describes itself as a camera company focused on visual communication. Its core product, Snapchat, is a mobile application known for its use of videos that can be posted and shared

with friends and with the public. Transmitting high-quality videos across digital networks is a core feature of Snapchat's success.

22. Using the Snapchat app, a user can post a "Story", which is photo or video shared for viewing by the user's friends, followers or public in the Snapchat app for a limited period of time, usually 24 hours. When a user "posts" a Story, the video is first captured in digital form and initially compressed on the user's device by or at the direction of the Snapchat app for upload to Snap's servers. After a user's Story is received by Snap's backend servers, Snap uses one or more codecs (video compression specifications) to transcode (i.e., convert into digitally compressed variants) at different resolutions and bitrate (different versions of the Story optimized for different devices and network conditions). When another user, follower or the public attempts to view the posted Story, the server chooses the optimal variant based on the viewer's device capability and bandwidth. Ex. 5, https://eng.snap.com/gpu_transcoding_at_scale, at 1.

23. On information and belief, Snap decodes videos (including "Stories" and other posts) from the format used by the uploading user device and encodes the video into formats, such as H.265 (High Efficiency Video Coding, or HEVC) and AV1-compliant formats, used by other devices that view the video. In doing so, Snap performs the patented technologies that underlie HEVC and AV1. Snap states: "We want to deliver the best video quality while ensuring smooth playback experience. This means we need to leverage the most advanced codecs where we can. HEVC (H265) allows us to deliver the same video quality with a smaller bitrate." Ex. 5, https://eng.snap.com/gpu_transcoding_at_scale, at 1-2.

24. Snap's testing shows that HEVC-compliant videos achieve both bitrate reduction and better quality over H.264-compliant videos:

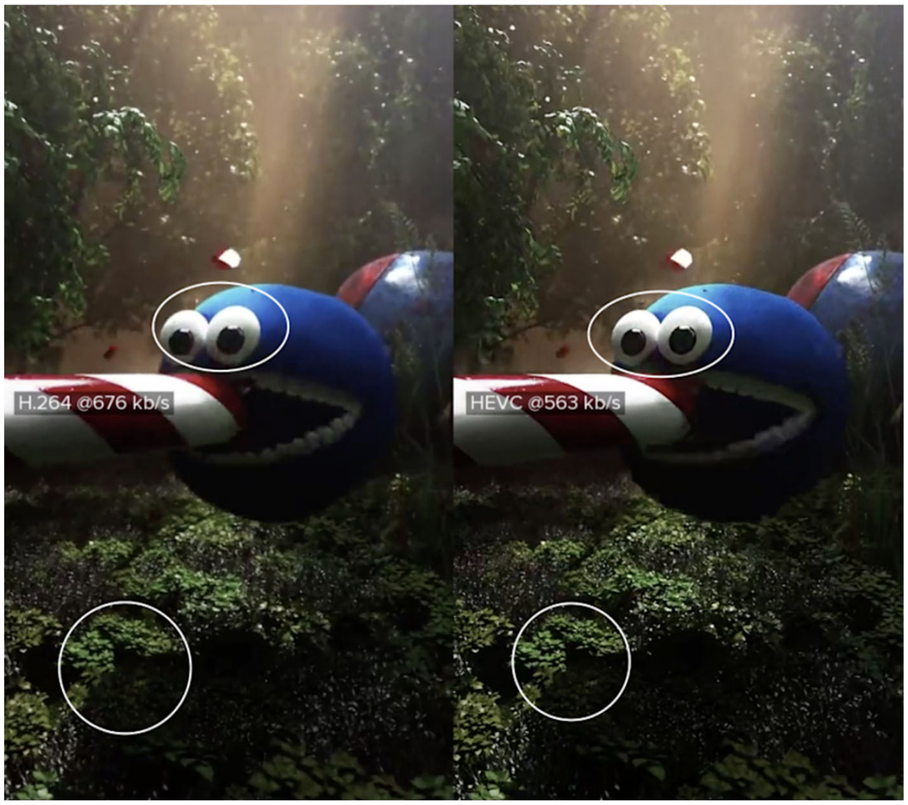


HEVC vs H264

The following charts show the GPU transcoded HEVC videos achieved both bitrate reduction and slightly better quality than H264.

Ex. 5, https://eng.snap.com/gpu_transcoding_at_scale, at 2.

25. Snap's blog shows a visual example of video transcoded into HEVC, demonstrating visual quality at approximately a 20% lower bitrate:



Ex. 5, https://eng.snap.com/gpu_transcoding_at_scale, at 3.

26. AV1, a codec developed after HEVC was released, is also used by Snap. *See* Ex. 6, https://eng.snap.com/snap-video-compression. The AV1 specification was published in 2018 by the Alliance for Open Media (AOM), a consortium including major technology companies and is promoted as a royalty-free video codec. Snap has joined the AOM as a "Promoter" to "support the development and adoption of AV1." Ex. 7, https://aomedia.org/press%20releases/snapinc-joins-the-alliance-for-open-media/.

27. One source describes AV1 as follows: "Since 2013, the High Efficiency Video Coding (HEVC) standard has become the state-of-the-art solution for video compression . . . However, HEVC is expensive for commercial use, especially for video streaming companies, due to royalty distribution policies. Aiming at this issue, the Alliance for Open Media (AOMedia) was founded with the goal of developing [purportedly] royalty-free video codecs, such as the recently launched AOMedia Video 1 (AV1)." Ex. 8 (Borges) at 3571.

28. However, AOM does not own all patents practiced by implementations of the AV1 codec. Rather, the AV1 specification was developed after many foundational video coding patents had already been filed, and AV1 incorporates technologies that are also present in HEVC. Those technologies are subject to existing third-party patent rights and associated licensing obligations. Licensing obligations are not eliminated by the inclusion of patented video coding technology into AV1.

29. As a result, implementations of AV1 practice intellectual property held by multiple licensors, as reflected by the formation of licensing programs such as the Sisvel AV1 Patent Pool and Access Advance's Video Distribution Patent Pool.

30. Dolby is one such licensor. Dolby holds patents that are practiced by AV1 implementations.

31. While Dolby contributed its inventions to the HEVC standard and licenses those

HEVC-essential patent claims through established patent pools on RAND terms, Dolby did not contribute its inventions to be incorporated into the AV1 specification or agree to any of AOM's policies. Accordingly, Dolby has made no contractual RAND commitment with respect to AV1. Dolby seeks injunctive relief against Snap's infringing decoding of AV1-compliant videos and encoding videos into AV1-compliant formats.

32. While today Snap encodes a significant amount of its content into HEVC-compliant formats, Snap has a team that "focuses on optimizing video and audio quality through cutting-edge technologies such as AV1." Ex. 9 (Junhong Nie LinkedIn). Snap's mobile application accepts AV1-compliant videos, and Snap will decode and encode these videos into other formats for delivery and viewing across a range of devices.

33. Snap's software further tracks whether AV1 decoding is supported on a given device to stream AV1 video when appropriate. For example, the source code for Snap's web app contains a flag "AV1_SOFTWARE_DECODING_SUPPORTED." Ex. 14 (Snap web source code) at _.

34. By the time AV1 was developed, HEVC had already been published and deployed in the market for years. Unsurprisingly, AV1 reuses concepts pioneered by HEVC. For example, engineering scholars have noted that both specifications are "based on the same hybrid block-based video-coding flow" and employ nearly the same approach to dividing images into coding units (CUs) and blocks. Ex. 8 (Borges) at 3572.

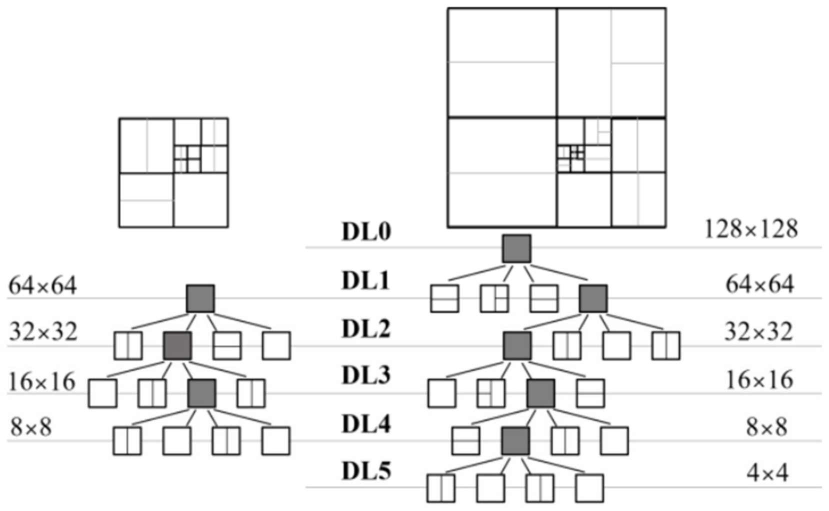


Fig. 2. Example of HEVC CTU (left) and AV1 SB (right) divided into CUs and blocks.

Ex. 8 (Borges) at 3572.

35. Because the AV1 specification incorporates concepts from the HEVC standard, products implementing AV1 implicate patents held by others, including contributors to the HEVC standard. For example, Dolby's '272 Patent asserted in Count IV below covers technology used in both HEVC and AV1.

36. Even though Snap acknowledges that it "need[s] to leverage the most advanced codecs," Ex. 5 at 1, Snap has not obtained the patent licenses required for Snap to lawfully do so. Dolby therefore seeks relief for Snap's continued unlicensed use of Dolby's patented technology.

III. THE ITU'S COMMON PATENT POLICY

37. The International Telecommunication Union (ITU) is a specialized agency of the United Nations responsible for coordinating global issues related to information and communication technologies. It was originally established in 1865 to manage international telegraph networks and has since evolved into a central body for modern telecommunications and digital infrastructure.

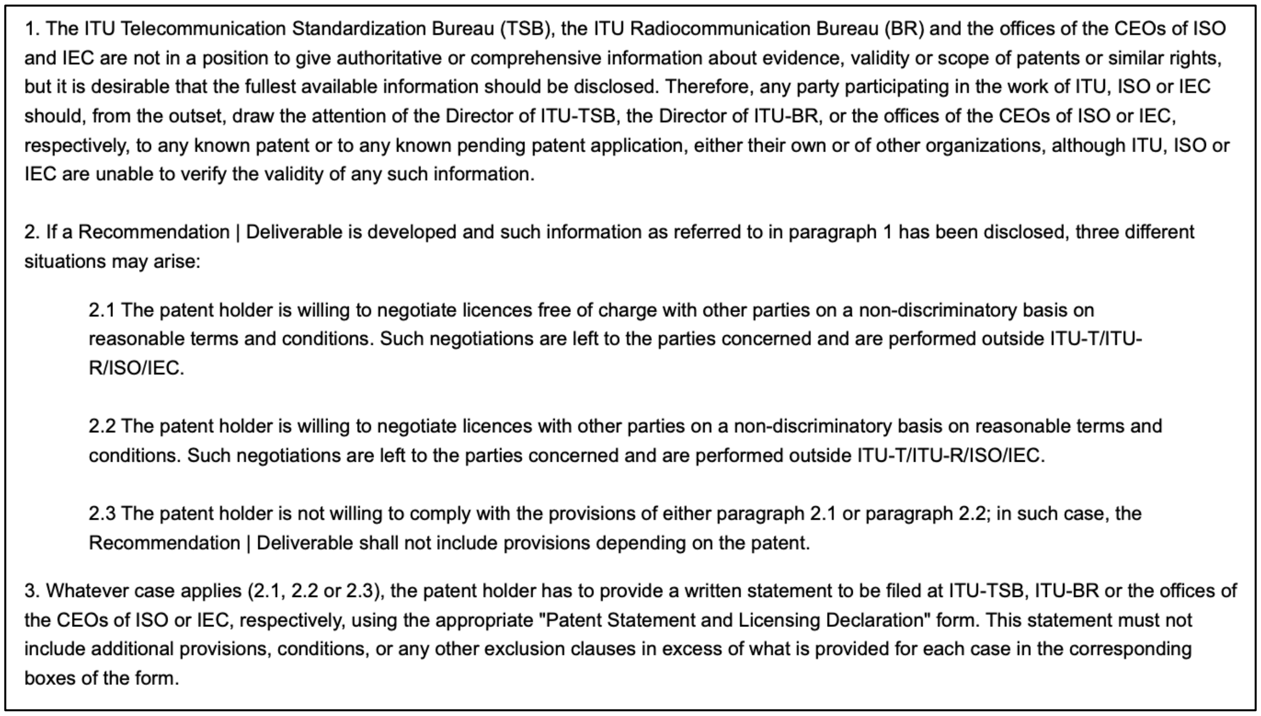
38. The ITU brings together governments, private companies, and technical experts to develop international standards, allocate radio spectrum, and improve global connectivity. One of its sectors,

known as ITU-T, develops technical standards (called "Recommendations") to ensure global interoperability of communication systems.

39. Part of the ITU-T's work involves development of video coding standards. One of its video coding standards, H.264, also known as Advanced Video Coding or AVC, was published in 2003. After overseeing the development of H.264, the ITU-T went on to develop H.265, or "High Efficiency Video Coding" (HEVC). H.265 is a newer standard that provides more efficient coding than H.264.

40. The ITU has a Common Patent Policy regarding the contributions that are ultimately included in a Recommendation and may be covered by one or more patent claims.

41. The ITU summarized its patent policy as follows:



1. The ITU Telecommunication Standardization Bureau (TSB), the ITU Radiocommunication Bureau (BR) and the offices of the CEOs of ISO and IEC are not in a position to give authoritative or comprehensive information about evidence, validity or scope of patents or similar rights, but it is desirable that the fullest available information should be disclosed. Therefore, any party participating in the work of ITU, ISO or IEC should, from the outset, draw the attention of the Director of ITU-TSB, the Director of ITU-BR, or the offices of the CEOs of ISO or IEC, respectively, to any known patent or to any known pending patent application, either their own or of other organizations, although ITU, ISO or IEC are unable to verify the validity of any such information.

2. If a Recommendation | Deliverable is developed and such information as referred to in paragraph 1 has been disclosed, three different situations may arise:

> 2.1 The patent holder is willing to negotiate licences free of charge with other parties on a non-discriminatory basis on reasonable terms and conditions. Such negotiations are left to the parties concerned and are performed outside ITU-T/ITU-R/ISO/IEC.
>
> 2.2 The patent holder is willing to negotiate licences with other parties on a non-discriminatory basis on reasonable terms and conditions. Such negotiations are left to the parties concerned and are performed outside ITU-T/ITU-R/ISO/IEC.
>
> 2.3 The patent holder is not willing to comply with the provisions of either paragraph 2.1 or paragraph 2.2; in such case, the Recommendation | Deliverable shall not include provisions depending on the patent.

3. Whatever case applies (2.1, 2.2 or 2.3), the patent holder has to provide a written statement to be filed at ITU-TSB, ITU-BR or the offices of the CEOs of ISO or IEC, respectively, using the appropriate "Patent Statement and Licensing Declaration" form. This statement must not include additional provisions, conditions, or any other exclusion clauses in excess of what is provided for each case in the corresponding boxes of the form.

Ex. 10 (Common Patent Policy), https://www.itu.int/en/ITU-T/ipr/Pages/policy.aspx.

42. According to the ITU's Guidelines for implementation of its patent policy, a declaration "declares the willingness to license in case part(s) or all of any proposals contained in contributions

submitted by the organization are included in ITU-T Recommendation(s) and the included part(s) contain items that have been patented or for which patent applications have been filed and whose use would be required to implement ITU-T Recommendation(s)." Ex. 11 (Patent Policy Guidelines) at 3.

43. Dolby has committed that it is prepared to grant licenses for implementations of the H.265 Standard to any patent claims essential to the H.265 Standard on reasonable and non-discriminatory (RAND) terms and conditions as set forth in the ITU Common Patent Policy. Consistent with the ITU Common Patent Policy, Dolby and the contributors from which the patents-in-suit originated, including Fraunhofer, submitted Patent Statement and Licensing Declarations to the ITU.

IV. LICENSE OFFER TO SNAP

44. Patent pools are cooperative licensing arrangements where patent holders aggregate their patents and license them collectively — through a single licensing framework — to one another or to third parties. By consolidating access to otherwise fragmented patent portfolios, patent pools reduce transaction costs, eliminate the need for licensees to negotiate separately with each individual patent holder, and allow for a single, aggregate royalty rate to be distributed equitably among pool members. These efficiencies promote the widespread adoption of technical standards, facilitate downstream innovation by lowering barriers to market entry, and encourage the diffusion of technology across industries, ultimately benefiting consumers through greater product interoperability, reduced costs, and increased competition.

45. Consistent with these market benefits, Dolby offers licenses to its patents at reasonable rates through Access Advance's Video Distribution Patent ("VDP") pool, which includes the patents-in-suit as well as other patents from Dolby Video Compression, LLC, Dolby International AB, Dolby Laboratories Licensing Corporation, and others. On August 13, 2025, Access Advance contacted Snap via e-mail and invited it to license Access Advance's Video Distribution Patent ("VDP") pool,

including patents-in-suit. On November 21, 2025, Access Advance sent Snap a patent chart which included HEVC and VVC assets. On March 18, 2026, Access Advance sent Snap a patent chart which included AV1 assets.

46. Access Advance's patent charts identified patents included in the VDP pool, including Dolby patents, and exemplary sections of the HEVC standard (and of the AV1 format) to which those patents relate. Access Advance also directed Snap to information describing the terms and pricing for a VDP license.

47. Dolby has offered a license to the Patents-in-Suit under reasonable and non-discriminatory terms through the VDP pool. Access Advance explained to Snap that the VDP Pool license was based on reasonable and non-discriminatory terms and provided an efficient licensing solution for Snap. Access Advance further explained that, as an alternative, Snap could seek bilateral licenses from individual VDP Pool Licensors.

48. Since the August 13, 2025, e-mail, Access Advance continued negotiating. Snap did not accept the offer to license the VDP Pool and did not seek a bilateral license from Dolby for the patents-in-suit or any other patents practiced by Snap's video distribution systems and functionality.

49. Despite these efforts, Snap remains unlicensed. Snap has continued to use Dolby's patented technology without paying any royalties.

50. Snap's continued unlicensed use of Dolby's patents has prompted Dolby to bring this action.

V. <u>DOLBY'S ASSERTED PATENTS</u>

51. Dolby complied with any applicable marking requirements under 35 U.S.C. § 287(a) at least because the asserted claims do not require marking, and/or there is nothing to mark. Dolby is entitled to all rights to recover for present, past, and future infringement of each of the patents-in-suit. Dolby is entitled to recover to the full extent allowable under 35 U.S.C. §§ 284-287, including up to six years prior to the filing of the complaint. There is no marking issue under 35 U.S.C. § 287

or otherwise which would limit damages, and Dolby has complied with all relevant statutes for maximum recovery under the law.

52. Snap has known about most, if not all, of the Asserted Patents since at least November 24, 2023 as an HEVC Advance pool licensee. *See* Ex. 24, https://web.archive.org/web/20231124024701/https://accessadvance.com/hevc-advance-patent-pool-licensees/ (showing that at least as of November 24, 2023, Snap Inc. was an HEVC Advance pool licensee). The HEVC Advance pool patent list includes the ’990, ’193, and ’469 Patents, as well as all the patents from which the ’272 Patent is a continuation. https://accessadvance.com/hevc-advance-patent-list/.

A. U.S. Patent No. 10,855,990

53. DVC owns by assignment the entire right, title, and interest in and to U.S. Patent No. 10,855,990 (the "’990 Patent"). The ’990 Patent, entitled "Inter-plane prediction," issued on December 1, 2020, to inventors Martin Winken, Heiner Kirchhoffer, Heiko Schwarz, Detlev Marpe, and Thomas Wiegand. DVC owns all rights to the ’990 Patent necessary to bring this action. The ’990 Patent issued from U.S. Patent Application No. 15/195,266, filed on June 28, 2016, and is a continuation of PCT application PCT/EP2010/054840, filed on April 13, 2010. A true and correct copy of the ’990 Patent is attached as Exhibit 1 and incorporated by reference herein.

54. The ’990 Patent is not directed to merely an abstract idea or any patent-ineligible concept. Instead, the ’990 Patent is directed to novel and unconventional improvements to prediction and reconstruction of video data in the field of digital video coding. The ’990 Patent provides improvements over prior prediction and video compression techniques that result in substantial benefits to prediction, video compression, video quality, and video playback.

55. In video coding, pictures are divided into blocks, and coding parameters are provided for individual blocks. ’990 Patent, 1:19-22. Because portions of images often look similar to other parts

of the same image, or to prior frames in a video, prediction techniques including inter-frame (predicting based on a prior frame) or intra-frame (predicting based on other parts of the same frame) are used to reduce the amount of data required to encode a video. *Id*. at 1:33-35. In compressed video, parameters are used to specify how the prediction is to be performed. *Id*. at 1:40-46. For intra-frame prediction, for example, an "intra prediction mode" is used to specify how an intra prediction is to be generated (for example, by indicating which direction to look in to predict the contents of a block). *Id*. at 1:46-56, 41:65-66.

56. Prior to the '990 Patent, video coding systems had the problem that "the bit rate that may be used for transmitting the prediction parameters for the blocks can become significant." *Id*. at 2:34-41. Usually, objects in a video have a unique texture or a unique motion, and would have the same prediction parameters for the whole object. *Id*. at 2:46-52. But because there were multiple blocks describing each object, the same or very similar prediction parameters would need to be coded many times over. *Id*. at 2:58-65. This redundant information became a significant part of the overall bit rate. *Id*. Thus, prior solutions were inefficient in coding video and used significant bandwidth and storage space to store redundant data.

57. One example is for the coding of color planes. Images for compression are often represented using "luma" and "chroma". In conventional image and video coding standards such as H.264, the luma and chroma planes were typically coded together, and the same parameters such as prediction modes would be used for all color components of an image. *Id*. at 39:57-62.

58. Switching from coding color planes together to independently coding color planes had the potential to increase efficiency. "If the color planes are coded together, one set of subdivision and prediction parameters is used for all color components of a block. This ensures that the side information is kept small, but it can result in a reduction of the coding efficiency compared to an

independent coding, since the usage of different block decompositions and prediction parameters for different color components can result in a smaller rate-distortion cost." *Id*. at 40:23-30. On the other hand, "[i]f the color planes are coded independently, the coding parameters such as the block partitioning, the reference indices, and the motion parameters can be selected for each color component separately in order to optimize the coding efficiency for each color component." *Id*. at 40:34-39.

59. However, in conventional systems, it was not feasible to independently code the color planes because it would have resulted in the coding of redundant information that outweighed any performance gain. In conventional systems, when coding color planes independently, it was "not possible, to employ the redundancy between the color components." *Id*. at 40:39-40. "The multiple transmissions of particular coding parameters does result in an increased side information rate (compared to the combined coding) and this increased side information rate can have a negative impact on the overall coding efficiency." *Id*. at 40:40-44. "Also, the support for auxiliary sample arrays" in conventional standards like H.264 was "restricted to the case that the auxiliary sample arrays are coded using their own set of coding parameters." *Id*. at 40:44-48. These technical challenges precluded taking advantage of any gains in efficiency from independently coding color planes.

60. The '990 Patent overcame these technical challenges in the prior systems by inventing a way to efficiently reuse prediction parameters from one color component for use with another. The '990 Patent employs the unconventional solution of using a parameter that specifies for each block whether to simply copy the prediction parameters from another component, or whether the prediction parameters are to be independently specified for the block being decoded. "The encoder can choose, for example based on a rate-distortion criterion, whether all or some of the sample arrays for a

particular block are coded using the same coding parameters or whether different coding parameters are used for different sample arrays." *Id.* at 40:59-63. "This selection can also be achieved by signaling for a particular block of a sample array whether specific coding parameters are inferred from an already coded co-located block of a different sample array." *Id.* at 40:63-67, 63:42-64:22.

61. This allows blocks to share some coding parameters, while separately transmitting others. *Id.* at 41:3-9. "For each block of a secondary plane group, it can be adaptively chosen whether a new set of selected coding parameters is transmitted or whether the selected coding parameters are inferred or predicted from the primary or another secondary plane group." *Id.* at 41:11-16. "The decisions of whether selected coding parameters for a particular block are inferred or predicted are included in the bitstream." *Id.* at 41:16-18.

62. The '990 Patent's unconventional solution allows video coding systems to take advantage of redundancy in some situations but also allow separate prediction parameters to be specified in other situations. "The inter-plane prediction allows a greater freedom in selecting the trade-off between the side information rate and prediction quality relative to the state-of-the-art coding of pictures consisting of multiple sample arrays." *Id.* at 41:18-22. "The advantage is an improved coding efficiency relative to the conventional coding of pictures consisting of multiple sample arrays." *Id.* at 41:22-24.

63. The '990 Patent therefore provides a specific technological improvement to the functionality and capabilities of video encoding technology that results in increased efficiency and improved video playback. For example, the encoder can now decide for each coding block whether to reuse prediction parameters between color planes, resulting in greater efficiency through saved space, or whether a separate prediction parameter should be explicitly signaled, resulting in greater efficiency through better prediction. Conventional technology prior to the '990 Patent did not allow for this

block-by-block flexibility.

64. The ’990 Patent recognizes and solves these specific technological problems that plagued conventional technology at the time. The ’990 Patent’s ability to reuse prediction parameters for some blocks while explicitly specifying prediction parameters for other blocks was a significant advancement over existing technology.

65. The novel solution of the ’990 Patent, which includes adaptively choosing whether a new set of selected coding parameters is transmitted or whether the selected coding parameters are inferred or predicted from another color plane, was not well-understood, routine, or conventional, nor was it simply comprised of well-understood, routine, and conventional activities previously known to the industry. Furthermore, the ordered combination of elements, including coding a first intra coding parameter containing information related to the first color component is determined using one or more parameters including a second intra coding parameter associated with a second coding block; if the first information indicates that a coding parameter of the first coding block is determined using the one or more parameters associated with the second coding block, the second intra coding parameter associated with the second coding block is copied as the first intra coding parameter such that the first and second intra coding parameters are equal to reconstruct the first coding block; and if the first information does not indicate that a coding parameter of the first coding block is determined using the one or more parameters associated with the second coding block, such that the at least one coding parameter is independent of the one or more parameters associated with the second coding block, was not well-understood, routine, or conventional.

66. DVC complied with any applicable marking requirements under 35 U.S.C. § 287 as to the ’990 Patent at least because the asserted claims of the ’990 Patent include method claims that do not require marking and/or there is nothing to mark.

B. U.S. Patent No. 9,924,193

67. DVC owns by assignment the entire right, title, and interest in and to U.S. Patent No. 9,924,193 (the "'193 Patent"). The '193 Patent, entitled "Picture coding supporting block merging and skip mode," issued on March 20, 2018, to inventors Heiko Schwarz, Heiner Kirchhoffer, Philipp Helle, Simon Oudin, Jan Stegemann, Benjamin Bross, Detlev Marpe, and Thomas Wiegand. DVC owns all rights to the '193 Patent necessary to bring this action. The '193 Patent issued from U.S. Patent Application No. 13/875,779, filed on May 2, 2013, and is a continuation of PCT application PCT/EP2010/069408, filed on November 4, 2011. A true and correct copy of the '193 Patent is attached as Exhibit 2 and incorporated by reference herein.

68. The '193 Patent is not directed to merely an abstract idea or any patent-ineligible concept. Instead, the '193 Patent is directed to novel and unconventional improvements to the process of video coding. The '193 Patent provides improvements over prior video coding techniques that result in substantial benefits to video compression, video quality, and video playback.

69. Video codecs divide pictures into blocks of varying sizes in order to compromise between setting prediction parameters set at a high spatial resolution, and lower spatial resolution. '193 Patent at 1:19-30. Providing parameters at a high resolution allows for more accurate predictions but consumes more bits of side information. *Id.* In contrast, parameters set at a lower spatial resolution, which may be coded with fewer bits, but sometimes will provide a less accurate prediction and require more bits of residual information to be encoded to correct these inaccuracies. *Id.*

70. One technique to address this problem is merging of blocks. *Id.* at 1:42-52. "In order to enable a better tradeoff between the amount of side information necessitated in order to signalize the picture subdivision on the one hand and the freedom in subdividing the picture on the other hand, merging of blocks may be used in order to increase the number of possible picture subdivisionings at a reasonable amount of additional data necessitated in order to signalize the merging information."

Id. at 1:42-48. "For blocks being merged, the coding parameters need to be transmitted within the bitstream in full merely once, similarly as if the resulting merged group of blocks was a directly subdivided portion of the picture." *Id.* at 1:48-52.

71. Another technique to increase the efficiency of encoding picture content is called skip mode. *Id.* at 1:53-57. Skip mode "enabl[es] the encoder to refrain from transmitting the residual data of a certain block to the decoder." *Id.* "That is, the skip mode is a possibility to suppress residual data transmission for certain blocks." *Id.* at 1:57-59. "The ability to suppress the transmission of residual data for certain blocks results in a broader granularity interval for encoding the coding/prediction parameters within which an optimum tradeoff between coding quality on the one hand and total bit rate spent on the other hand may be expected: naturally, increasing the spatial resolution of the encoding of the coding/prediction parameters results in an increase of the side information rate while decreasing, however, the residuum thereby lowering the rate necessitated to encode the residual data." *Id.* at 1:59-2:1. "However, due to the availability of the skip mode, it may be favorable to obtain an abrupt coding rate saving by merely moderately further increasing the granularity at which the coding/prediction parameters are transmitted so that the residuum is so small that a separate transmission of the residuum may be left away." *Id.* at 2:1-7.

72. Prior to the '193 Patent, if video coding systems tried to use *both* block merging and skip mode, it would result in "redundancies newly caused by the combination of block merging and skip mode usage." *Id.* at 2:8-10. Prior systems did not address these redundancies, resulting in less efficient coding of video.

73. Providing separate flags to signal block merging and skip mode required extra bits that were in many cases redundant. The '193 Patent, in contrast, allowed for both to be signaled with a single syntax element. "One possible state of one or more syntax elements within the bitstream may

signalize for a current sample set of a picture that the sample set is to be merged and has no prediction residual encoded and inserted into the bitstream." *Id.*, Abstract.

74. The '193 Patent overcame these technical challenges in the prior systems by inventing a common flag to signal, in certain cases, that both block merging and skip mode are to be used. "A common flag may signalize whether the coding parameters associated with a current sample set are to be set according to a merge candidate or to be retrieved from the bitstream, and whether the current sample set of the picture is to be reconstructed based on a prediction signal depending on the coding parameters associated with the current sample set, without any residual data, or to be reconstructed by refining the prediction signal depending on the coding parameters associated with the current sample set by means of residual data within the bitstream." *Id.*, Abstract. The '193 Patent employs the unconventional solution of providing a flag that indicates *both* merging and skip mode are to be used. This allows for more efficient coding of merge and skip information.

75. The '193 Patent therefore provides a specific technological improvement to the functionality and capabilities of video encoding technology that results in increased efficiency and improved video playback. For example, the encoder can now indicate that a block is being merged and uses skip mode by using a single flag.

76. Conventional technology prior to the '193 Patent was not capable of the same efficiency. Conventional video coding systems treated block merging and skip mode separately and did not allow for a single flag to indicate both.

77. The '193 Patent recognizes and solves these specific technological problems that plagued the conventional technology at the time. The '193 Patent's ability to signal the use of both block merging and skip mode with a single flag was a significant advancement over existing technology.

78. The novel solution of the '193 Patent, which includes signaling the use of both block merging

and skip mode with a single flag, was not well-understood, routine, or conventional, nor was it simply comprised of well-understood, routine, and conventional activities previously known to the industry. Furthermore, the ordered combination of elements, including coding first information that indicates both that (1) the coding block is to be reconstructed based on one or more coding parameters of a merge candidate coding block and (2) the coding block is to be reconstructed without residual data; coding, in the first state, the merge candidate coding block and its associated coding parameters; and coding, in the second state, information indicating that the coding block is to be reconstructed using at least one coding parameter coded in the data stream, and a set of coding parameters associated with another merge candidate coding block and residual data for the coding block, was not well-understood, routine, or conventional.

79. DVC complied with any applicable marking requirements under 35 U.S.C. § 287 as to the ’193 Patent at least because the asserted claims of the ’193 Patent include method claims that do not require marking and/or there is nothing to mark.

C. U.S. Patent No. 9,596,469

80. DVC owns by assignment the entire right, title, and interest in and to U.S. Patent No. 9,596,469 (the "’469 Patent"). The ’469 Patent, entitled "Sample array coding for low-delay," issued on March 14, 2017, to inventors Valeri George, Anastasia Henkel, Heiner Kirchhoffer, Detlev Marpe, and Thomas Schierl. DVC owns all rights to the ’469 Patent necessary to bring this action. The ’469 Patent issued from U.S. Patent Application No. 14/141,374, filed on December 26, 2013, and is a continuation of PCT application PCT/EP2012/063929, filed on July 16, 2012. A true and correct copy of the ’469 Patent is attached as Exhibit 3 and incorporated by reference herein.

81. The ’469 Patent is not directed to merely an abstract idea or any patent-ineligible concept. Instead, the ’469 Patent is directed to novel and unconventional improvements to the process of video coding. The ’469 Patent provides improvements over prior video coding techniques that result

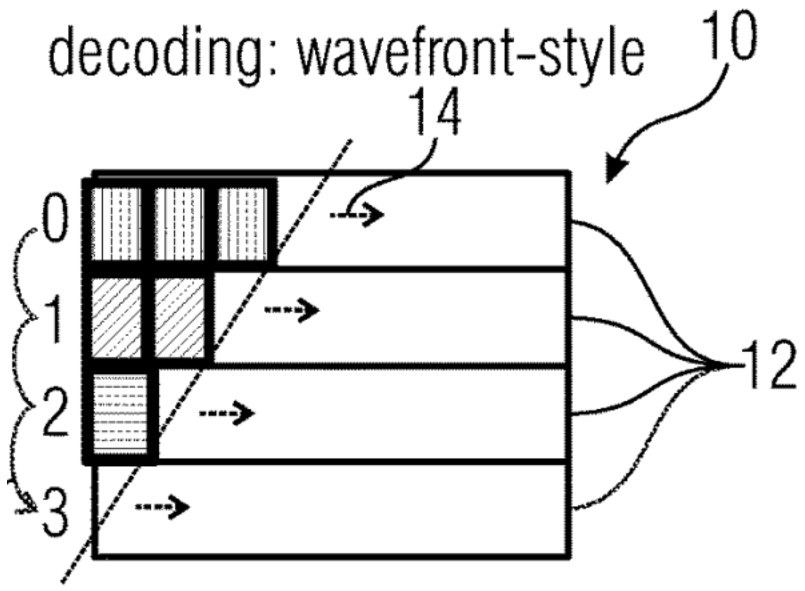
in substantial benefits to video compression, video quality, and video playback.

82. Conventional video coding systems have spatial dependencies. ’469 Patent at 1:26-27. Decoding one part of an image relies on information from previously decoded parts of the image. *Id*. at 1:27-30. Critical concepts in video coding like motion vector prediction, intra prediction, and other parts of video coding systems rely on the fact that another part of the image that has already been decoded can help predict what will be in the part of the image that is currently being decoded. *Id*. The adaptation of CABAC probability between coding units also can have spatial dependencies. *Id*. at 1:24-27.

83. Computer systems, however, often use a multiple-core architecture, where different cores can work in parallel on the same task. *Id*. at 1:20-21. Thus, “[p]arallelization of encoder and decoder is very important due to the increased processing requirements by the HEVC standard as well as by the expected increase of video resolution.” *Id*. at 1:17-20. If different cores of a processor are working on different parts of an image, one core may not be able to encode or decode its part of the image because the other cores have not yet encoded or decoded another part of the image on which the first part relies. Yet for high-resolution videos like 4K videos, and for efficient video coding standards that require significant processing power, the ability to process in parallel can be critical to having sufficient throughput to process the video and to reducing the latency in encoding or decoding videos.

84. An invention of the ’469 Patent, relates to “wavefront parallel processing,” a technique to allow images to be encoded and decoded in parallel. *Id*. at 1:34-35. “Wavefront parallel processing” allowed parallel processing of rows of chunks in “wavefront” style, where each row could be decoded by a separate processor, and decoding of a given chunk in a given row could proceed once the processing of the row above had proceeded to the chunk immediately above and to the right of

the given block.



Id., Fig. 16.

85. An inherent challenge with parallelizing image coding is that processing an image in parallel risks efficiency loss, because encoders and decoders may not be able to take advantage of certain types of prediction and redundancy in pictures. For example, an encoder or decoder cannot update probabilities based on the coding in a previous block unless the coding in that block has been completed. The '469 Patent's invention was designed to "lower the coding efficiency loss and thus reduce the burden on the bitstream for parallelization approaches in encoder and decoder" and to correct the problem that "low-delay processing was not possible with the available techniques." *Id.* at 1:35-43.

86. For example, probability estimations in conventional video coding systems updated probability estimations as decoding proceeded sequentially throughout the entire image. This does not allow for parallel processing, since the probability model must be updated for all previous blocks before any given block can be decoded.

87. The '469 Patent overcame these technical challenges in the prior systems by inventing a new system for decoding video allowing efficient decoding with reduced spatial dependency. The '469

Patent employs an unconventional method of probability estimation to encode and decode each entropy slide. This allows the video coding system to operate in parallel with greatly increased performance.

88. The ’469 Patent therefore provides a specific technological improvement to the functionality and capabilities of video encoding technology that results in increased resiliency and improved video playback. For example, the system can now encode and decode video in parallel without a significant loss in quality, bitrate, or efficiency.

89. Conventional technology prior to the ’469 Patent was not capable of this parallelization. The ’469 Patent recognizes and solves these specific technological problems that plagued the conventional technology at the time. The ’469 Patent’s ability to parallelize video coding without a significant loss in efficiency, was a significant advancement over existing technology.

90. The novel solution of the ’469 Patent, which enables efficient parallelization in video coding, was not well-understood, routine, or conventional, nor was it simply comprised of well-understood, routine, and conventional activities previously known to the industry. Furthermore, the ordered combination of elements, including for example performing entropy encoding along a respective entropy coding path using respective probability estimations; adapting those estimations using a previously encoded part of the respective entropy slice; encoding slices sequentially using an entropy slice order; encoding a slice based on the probability estimations adapted using the previously encoded part of the slice and probability estimations as used in the entropy encoding of a spatially neighboring preceding entropy slice at a neighboring part of the spatially neighboring preceding entropy slice; the arrangement of entropy slices in rows and columns; and initializing the probability estimations, was not well-understood, routine, or conventional.

91. DVC complied with any applicable marking requirements under 35 U.S.C. § 287 as to the

’469 Patent at least because the asserted claims of the ’469 Patent include method claims that do not require marking and/or there is nothing to mark.

D. U.S. Patent No. 10,404,272

92. DVC owns by assignment the entire right, title, and interest in and to U.S. Patent No. 10,404,272 (the “’272 Patent”). The ’272 Patent, entitled “Entropy encoding and decoding scheme,” issued on September 3, 2019, to inventors Detlev Marpe, Tung Nguyen, Heiko Schwarz, and Thomas Wiegand. DVC owns all rights to the ’272 Patent necessary to bring this action. The ’272 Patent issued from U.S. Patent Application No. 16/198,338, filed on November 21, 2018, and is a continuation of a series of applications including PCT application PCT/EP2012/050431, filed on January 12, 2012. The application also claims priority to provisional application No. 61/432,884, filed on January 14, 2011. A true and correct copy of the ’272 Patent is attached as Exhibit 4 and incorporated by reference herein.

93. The ’272 Patent is not directed to merely an abstract idea or any patent-ineligible concept. Instead, the ’272 Patent is directed to novel and unconventional improvements to the process of video coding. The ’272 Patent provides improvements over prior video coding techniques that result in substantial benefits to video compression, video quality, and video playback.

94. In video coding, the pictures of a video sequence are usually decomposed into blocks. ’272 Patent at 63:39-40. These blocks, or the color components of the blocks, are predicted by either motion compensated prediction or intra prediction. *Id.* at 63:40-42. “The difference between the original blocks or the colour components of the original blocks and the corresponding prediction signals, also referred to as the residual signal, is usually transformed and quantized.” *Id.* at 64:5-8. “A two-dimensional transform is applied to the residual signal and the resulting transform coefficients are quantized.” *Id.* at 64:8-10.

95. Prior to the ’272 Patent, entropy coding was traditionally used to transmit the resulting

quantized transform coefficients. *Id*. at 64:23-26. In H.264/AVC, a variety of syntax elements were used to describe the transform coefficients, and "all syntax elements for the transform coefficient levels are coded using a binary probability modelling." *Id*. at 65:42-44. Non-binary syntax elements—those that can have more values than just zero or one—are "first binarized, i.e., . . . mapped onto a sequence of binary decisions (bins), and these bins are sequentially coded." *Id*. at 65:44-47. "Each coded bin (including the binary syntax elements) is associated with a context." *Id*. at 65:50-51. "A context represents a probability model for a class of coded bins." *Id*. at 65:51-52. "A measure related to the probability for one of the two possible bin values is estimated for each context based on the values of the bins that have been already coded with the corresponding context." *Id*. at 65:52-56. "For several bins related to the transform coding, the context that is used for coding is selected based on already transmitted syntax elements or based on the position inside a block." *Id*. at 65:56-59.

96. Because there are a large range of possible values for transform coefficients, however, a large number of bins are required to code data in this way. This not only increases the computation complexity of video coding but also impacts compression efficiency.

97. Other methods for encoding data, such as Exponential-Golomb codes, existed and provided a lower-complexity option. *Id*. at 68:34-39. However, using Exponential-Golomb codes to encode the transform coefficients would not have been feasible because they are less efficient in many circumstances.

98. Existing video coding systems needed the performance benefits of the most compression-efficient entropy codes but suffered a large increase in complexity because of the large range of values that could occur in transform coefficients.

99. The '272 Patent overcame these technical challenges in the prior systems by inventing a

partitioning system. The ’272 Patent employs the unconventional solution of partitioning: using one type of coding to encode smaller values, and if that coding indicates that the value is larger than a threshold, another type of coding is used to indicate how much larger the value is. *Id*. at 68:13-33. With partitioning, different ranges of values can be represented using different coding techniques. This allows for a reduced-complexity system that nonetheless provides efficient compression.

100. The ’272 Patent therefore provides a specific technological improvement to the functionality and capabilities of video encoding technology that results in increased resiliency and improved video playback. For example, the encoder can now use the most efficient coding to represent small transform coefficients, which are the most common, while using other coding techniques to represent larger transform coefficients. This allows the encoder “to achieve a better tradeoff between coding complexity on the one hand and compression efficiency on the other hand,” and a “better compression efficiency per se, at a moderate coding complexity.” ’272 Patent at 2:39-48.

101. Conventional technology prior to the ’272 Patent was not capable of this type of partitioning and instead required a large number of bins to represent transform coefficients. The ’272 Patent recognizes and solves these specific technological problems that plagued the conventional technology at the time. The ’272 Patent’s ability to partition transform coefficients to use different coding schemes was a significant advancement over existing technology.

102. The novel solution of the ’272 Patent, which includes this partitioning, was not well-understood, routine, or conventional, nor was it simply comprised of well-understood, routine, and conventional activities previously known to the industry. Furthermore, the ordered combination of elements, including a sequence of syntax elements having a value range which is sub-divided into a plurality of disjoint portions, related to level values of transform coefficients of a transform coefficient block; decomposing each syntax element into a corresponding set of source symbols

based on a portion of the plurality of disjoint portions associated with the syntax element; receiving and sub-dividing the sequence of source symbols; and encoding source symbols of one sequence using arithmetic context-based entropy coding and source symbols of the other sequence using Exp-Golomb coding, was not well-understood, routine, or conventional.

103. DVC complied with any applicable marking requirements under 35 U.S.C. § 287 as to the ’272 Patent at least because there is nothing to mark.

<u>COUNT I: PATENT INFRINGEMENT OF THE ’990 PATENT</u>

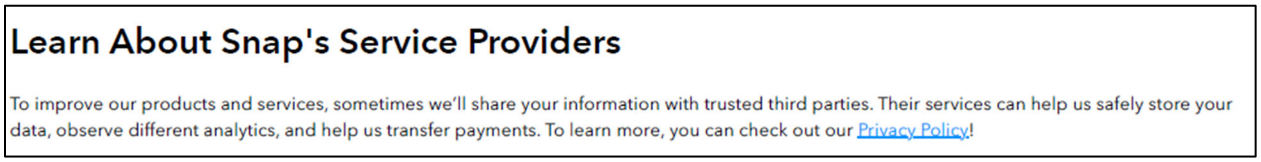
104. Dolby incorporates by reference the preceding paragraphs as though fully set forth herein.

105. Snap had knowledge and notice of the ’990 Patent at least by November 24, 2023, and again through communications on August 13, 2025 and November 21, 2025.

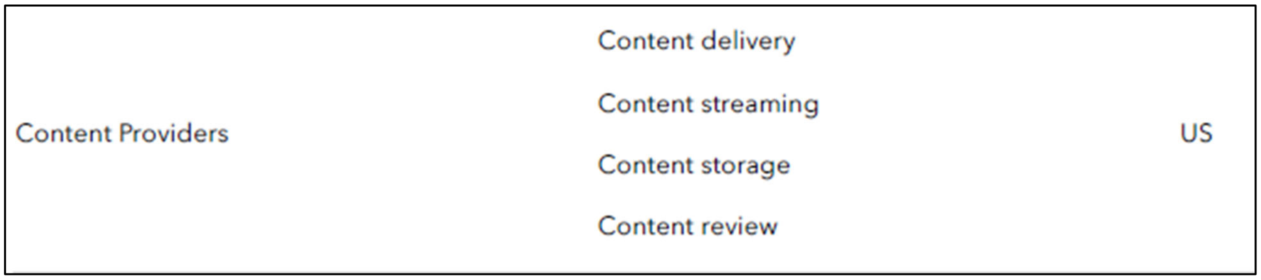
106. Snap infringes the ’990 Patent by making, using, offering to sell, and/or selling the patented invention within the United States, and/or importing the patented invention into the United States. Snap thus directly infringes the ’990 Patent literally and/or under the Doctrine of Equivalents, in violation of 35 U.S.C. § 271. For example, Snap directly infringes the ’990 Patent through its encoding, decoding, and transcoding of H.265-compliant (HEVC) video in the United States.

107. On information and belief, Snap indirectly infringes claims of the ’990 Patent, as provided in 35 U.S.C. § 271(b), by knowingly inducing infringement by others, such as Snap’s partners and vendors, in this District and elsewhere in the United States.

108. For example, Snap uses the infrastructure of its providers to encode and decode video. Snap’s web page indicates that these providers perform content delivery, streaming, storage, and review, including the encoding and decoding of video.


Learn About Snap's Service Providers

To improve our products and services, sometimes we’ll share your information with trusted third parties. Their services can help us safely store your data, observe different analytics, and help us transfer payments. To learn more, you can check out our Privacy Policy!



Content Providers	Content delivery Content streaming Content storage Content review	US

Ex. 12 at 1.

109. As a result of Snap's inducement, Snap's providers perform infringing video coding and directly infringe the '990 Patent. Snap has performed and continues to perform these affirmative acts with knowledge of the '990 Patent and with the intent, or willful blindness, that the induced acts directly infringe the '990 Patent.

110. Upon information and belief, Snap derives revenue, directly and indirectly, from the activities relating to the Accused Services and to infringement of the '990 Patent, including in this District and elsewhere in the United States.

111. Snap's infringement of the '990 Patent is willful and deliberate. As detailed above, Snap had knowledge of the '990 Patent and had knowledge, or was willfully blind, as to its infringement of the '990 Patent. As discussed above, Snap has had actual knowledge of the '990 Patent since before the filing of this suit. Further, as discussed above, Snap knew or should have known that its actions infringe and actively induce infringement of the '990 Patent. As discussed above, Snap specifically intended that both itself and/or its customers and providers infringe the '990 Patent.

112. Snap's infringement of the '990 Patent has damaged and will continue to damage Dolby.

113. Exemplary claim 21 of the '990 Patent recites:[1]

> 21. A method for encoding, into a data stream, a video including pictures of a scene represented by multiple arrays of information samples, comprising:

[1] For each patent in suit, this complaint identifies one exemplary claim to detail in its infringement analysis. Dolby reserves the right to assert all claims, or a subset of all claims, of each asserted patent in its infringement contentions.

inserting, into the data stream, a first set of coding parameters including a first intra coding parameter associated with a first coding block in a first array of information samples that represents a first color component of the video, wherein the first set of coding parameters is to be used for reconstructing the first coding block in an intra coding mode;

inserting, into the data stream, inter-plane interchange information associated with a second coding block in a second array of information samples that represents a second color component of the video, wherein the inter-plane interchange information signals whether a second intra coding parameter of a second set of coding parameters used for reconstructing the second coding block in the intra coding mode is to be derived based on the first intra coding parameter;

responsive to a determination based on the inter-plane interchange information that the second intra coding parameter is not to be derived from the first intra coding parameter, inserting, into the data stream, the second set of coding parameters including the second intra coding parameter;

copying, responsive to a determination based on the inter-plane interchange information that the second intra coding parameter is to be derived from the first intra coding parameter, the first intra coding parameter as the second intra coding parameter such that the first and second intra coding parameters are equal, wherein a spatial resolution of the first array is twice a spatial resolution of the second array, the first intra coding parameter indicates an angle of direction of intra prediction used in the intra coding mode within the first array and the second intra coding parameter indicates an angle of direction of intra prediction used in the intra coding mode within the second array; and

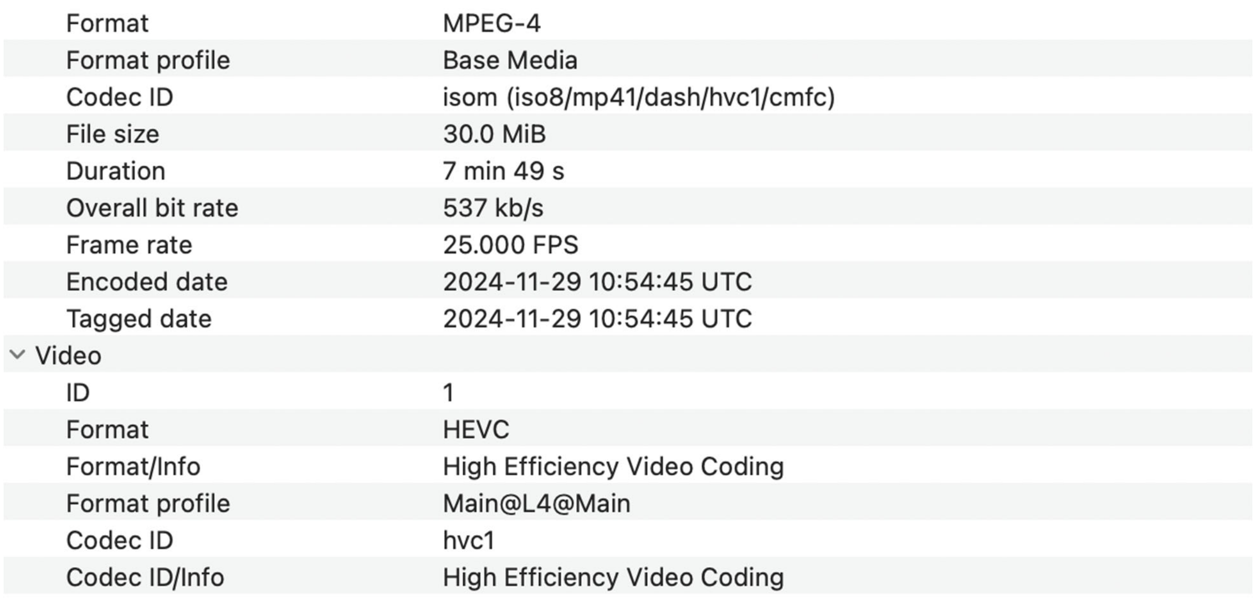
predicting the second coding block based on the second set of coding parameters including the second intra coding parameter to generate a predicted second coding block based on the intra coding mode,

wherein the first and second arrays of information samples represent different types of spatially sampled information of the scene.

114. Snap's Accused Services perform "a method for encoding, into a data stream, a video including pictures of a scene represented by multiple arrays of information samples."

115. For example, Snap's backend servers encode video into H.265-compliant (HEVC) formats for streaming to devices.

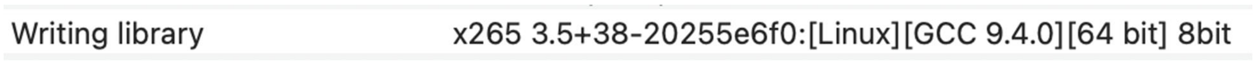
116. Videos downloaded from Snap show that an HEVC-compliant format is used:



Format	MPEG-4
Format profile	Base Media
Codec ID	isom (iso8/mp41/dash/hvc1/cmfc)
File size	30.0 MiB
Duration	7 min 49 s
Overall bit rate	537 kb/s
Frame rate	25.000 FPS
Encoded date	2024-11-29 10:54:45 UTC
Tagged date	2024-11-29 10:54:45 UTC
Video	
ID	1
Format	HEVC
Format/Info	High Efficiency Video Coding
Format profile	Main@L4@Main
Codec ID	hvc1
Codec ID/Info	High Efficiency Video Coding

(Video from Snap as viewed in MediaInfo app.)

117. Snap uses the x265 library, version 3.5, revision 20255e6f0, to encode HEVC-compliant video.



Writing library	x265 3.5+38-20255e6f0:[Linux][GCC 9.4.0][64 bit] 8bit

(Video from Snap as viewed in MediaInfo app.)

118. Snap's Accused Services perform "inserting, into the data stream, a first set of coding parameters including a first intra coding parameter associated with a first coding block in a first array of information samples that represents a first color component of the video, wherein the first set of coding parameters is to be used for reconstructing the first coding block in an intra coding mode."

119. For example, the H.265-compliant bitstream specifies a luma intra prediction mode, IntraPredModeY, associated with a luma coding block.

Table 8-1 specifies the value for the intra prediction mode and the associated names.

Table 8-1 – Specification of intra prediction mode and associated names

Intra prediction mode	Associated name
0	INTRA_PLANAR
1	INTRA_DC
2..34	INTRA_ANGULAR2..INTRA_ANGULAR34

IntraPredModeY[xPb][yPb] labelled 0..34 represents directions of predictions as illustrated in Figure 8-1.

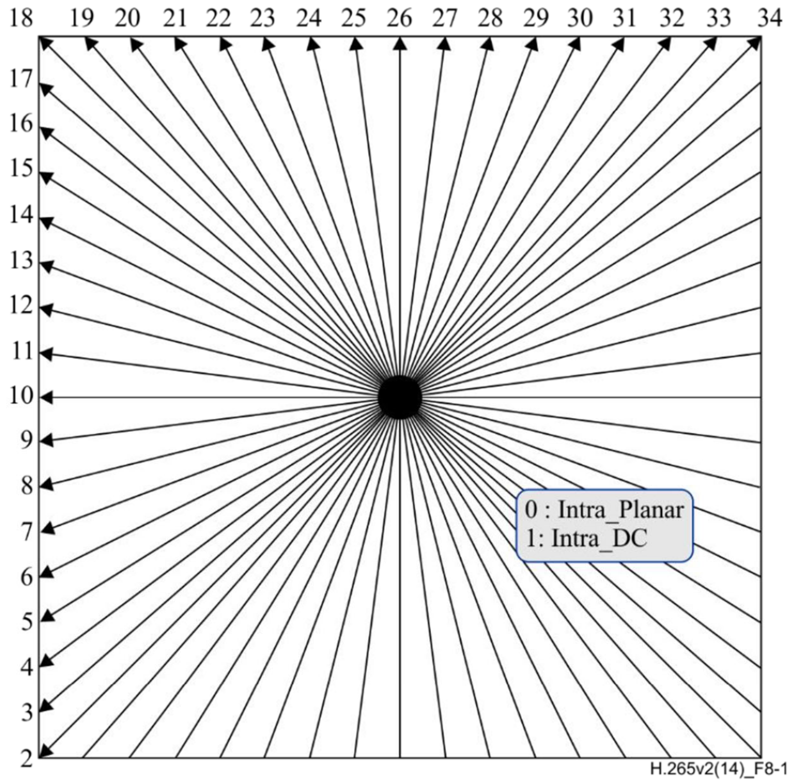


Figure 8-1 – Intra prediction mode directions (informative)

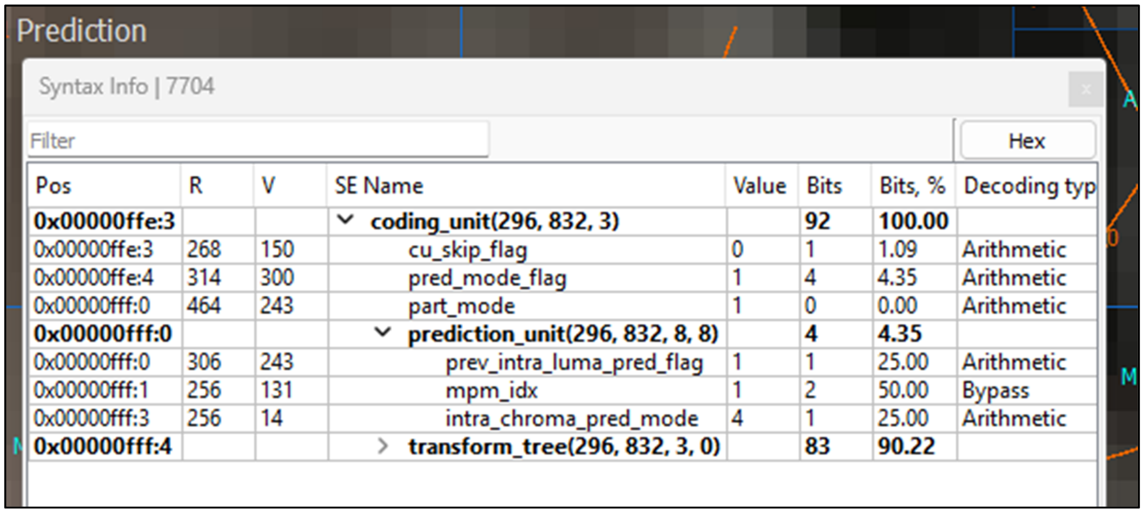
IntraPredModeY[xPb][yPb] is derived by the following ordered steps:

Ex. 13 (H.265 specification) at 118.

The syntax elements **prev_intra_luma_pred_flag**[x0 + i][y0 + j], **mpm_idx**[x0 + i][y0 + j] and **rem_intra_luma_pred_mode**[x0 + i][y0 + j] specify the intra prediction mode for luma samples. The array indices x0 + i, y0 + j specify the location (x0 + i, y0 + j) of the top-left luma sample of the considered prediction block relative to the top-left luma sample of the picture. When prev_intra_luma_pred_flag[x0 + i][y0 + j] is equal to 1, the intra prediction mode is inferred from a neighbouring intra-predicted prediction unit according to clause 8.4.2.

Ex. 13 (H.265 specification) at 101.

120. For example, compressed video data captured from Snap shows the use of prev_intra_luma_pred_flag and mpm_idx to specify the intra prediction mode for a luma coding block.



Pos	R	V	SE Name	Value	Bits	Bits, %	Decoding typ
0x00000ffe:3			**coding_unit(296, 832, 3)**		**92**	**100.00**	
0x00000ffe:3	268	150	cu_skip_flag	0	1	1.09	Arithmetic
0x00000ffe:4	314	300	pred_mode_flag	1	4	4.35	Arithmetic
0x00000fff:0	464	243	part_mode	1	0	0.00	Arithmetic
0x00000fff:0			**prediction_unit(296, 832, 8, 8)**		**4**	**4.35**	
0x00000fff:0	306	243	prev_intra_luma_pred_flag	1	1	25.00	Arithmetic
0x00000fff:1	256	131	mpm_idx	1	2	50.00	Bypass
0x00000fff:3	256	14	intra_chroma_pred_mode	4	1	25.00	Arithmetic
0x00000fff:4			**transform_tree(296, 832, 3, 0)**		**83**	**90.22**	

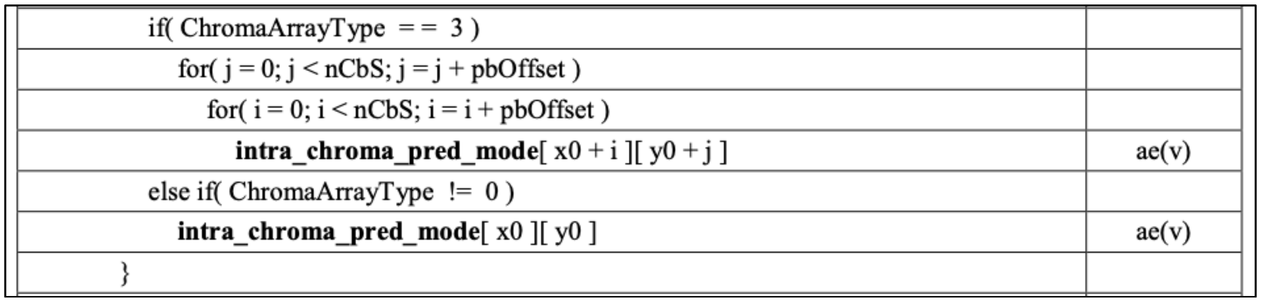
VQ Analyzer screenshot of bitstream collected from Snapchat.

121. The x265 encoder source code also shows encoding of the luma intra mode from which the chroma mode is determined. For example, the function Entropy::codeIntraDirLumaAng encodes the luma prediction mode for intra prediction. Ex. 16 (entropy.cpp) at 29-30.

122. Snap's Accused Services perform "inserting, into the data stream, inter-plane interchange information associated with a second coding block in a second array of information samples that represents a second color component of the video, wherein the inter-plane interchange information signals whether a second intra coding parameter of a second set of coding parameters used for reconstructing the second coding block in the intra coding mode is to be derived based on the first intra coding parameter."

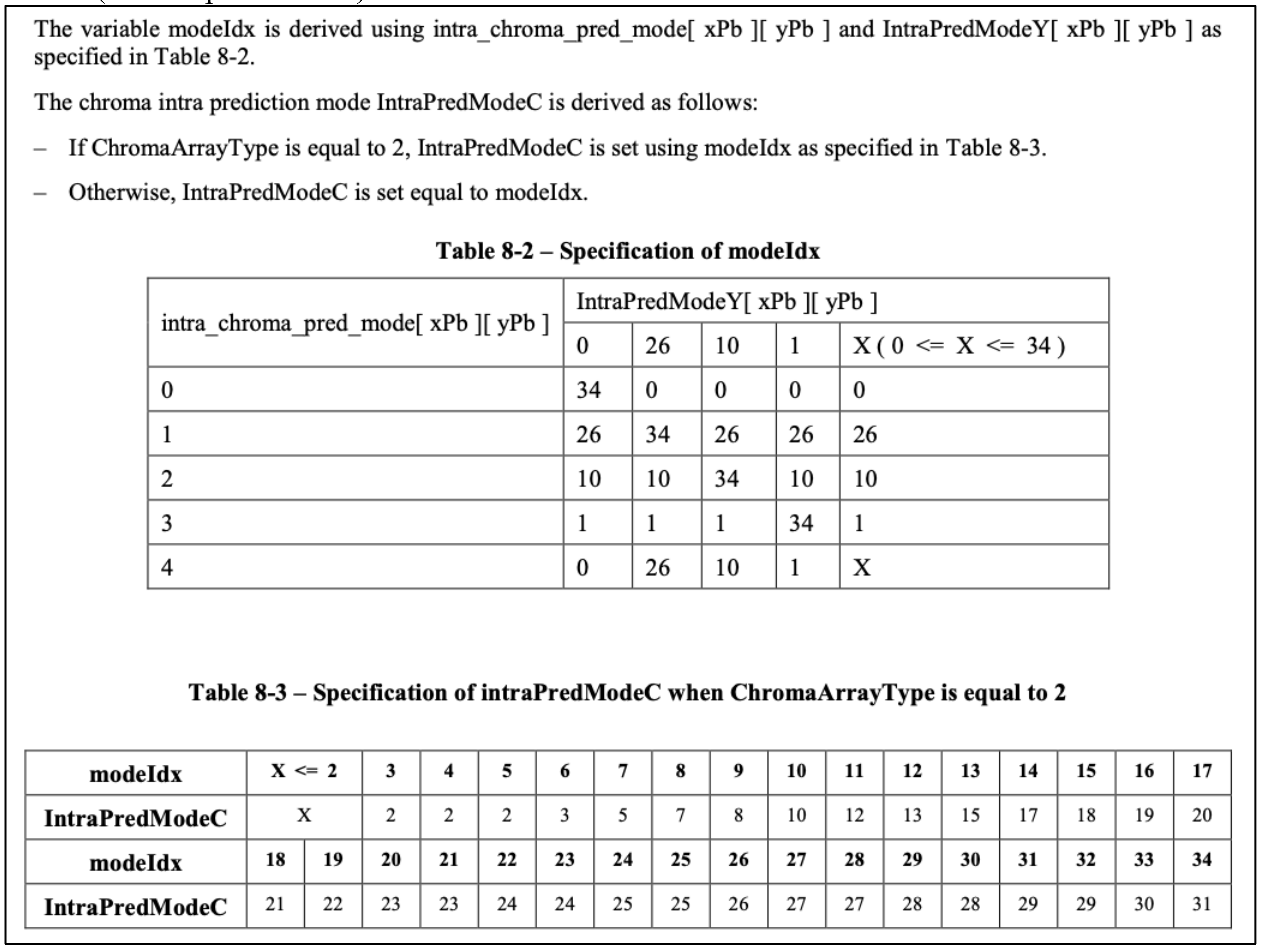
123. For example, the H.265 standard describes a syntax element called intra_chroma_pred_mode. This element, if set to 4, indicates that the chroma prediction mode should be derived from the luma prediction mode signaled in a luma coding block. The first bit of the binarization indicates whether or not intra_chroma_pred_mode is set to 4 and therefore whether

the chroma prediction mode should be derived from the luma prediction mode signaled in a luma coding block.



if(ChromaArrayType == 3)	
for(j = 0; j < nCbS; j = j + pbOffset)	
for(i = 0; i < nCbS; i = i + pbOffset)	
intra_chroma_pred_mode[x0 + i][y0 + j]	ae(v)
else if(ChromaArrayType != 0)	
intra_chroma_pred_mode[x0][y0]	ae(v)
}	

Ex. 13 (H.265 specification) at 52.



The variable modeIdx is derived using intra_chroma_pred_mode[xPb][yPb] and IntraPredModeY[xPb][yPb] as specified in Table 8-2.

The chroma intra prediction mode IntraPredModeC is derived as follows:

– If ChromaArrayType is equal to 2, IntraPredModeC is set using modeIdx as specified in Table 8-3.

– Otherwise, IntraPredModeC is set equal to modeIdx.

Table 8-2 – Specification of modeIdx

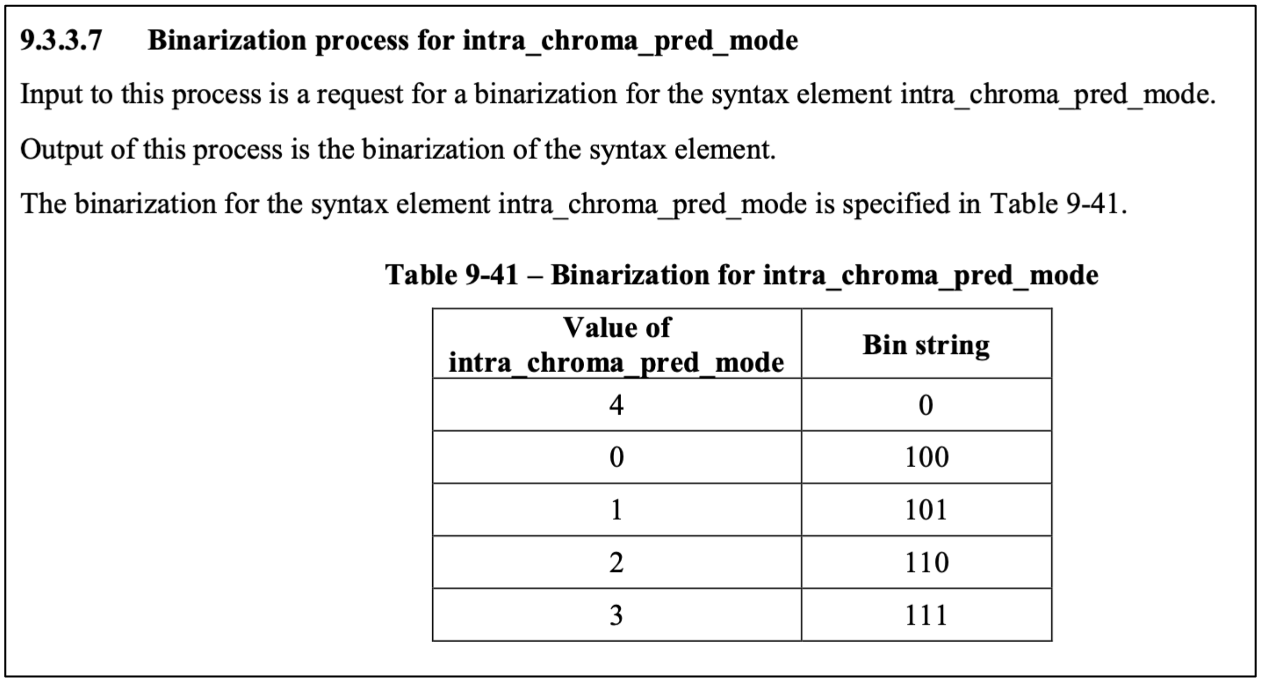
intra_chroma_pred_mode[xPb][yPb]	IntraPredModeY[xPb][yPb]				
	0	26	10	1	X (0 <= X <= 34)
0	34	0	0	0	0
1	26	34	26	26	26
2	10	10	34	10	10
3	1	1	1	34	1
4	0	26	10	1	X

Table 8-3 – Specification of intraPredModeC when ChromaArrayType is equal to 2

modeIdx	**X <= 2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
IntraPredModeC	X	2	2	2	3	5	7	8	10	12	13	15	17	18	19	20

modeIdx	**18**	**19**	**20**	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	**31**	**32**	**33**	**34**
IntraPredModeC	21	22	23	23	24	24	25	25	26	27	27	28	28	29	29	30	31

Ex. 13 (H.265 specification) at 120.



9.3.3.7 Binarization process for intra_chroma_pred_mode

Input to this process is a request for a binarization for the syntax element intra_chroma_pred_mode.

Output of this process is the binarization of the syntax element.

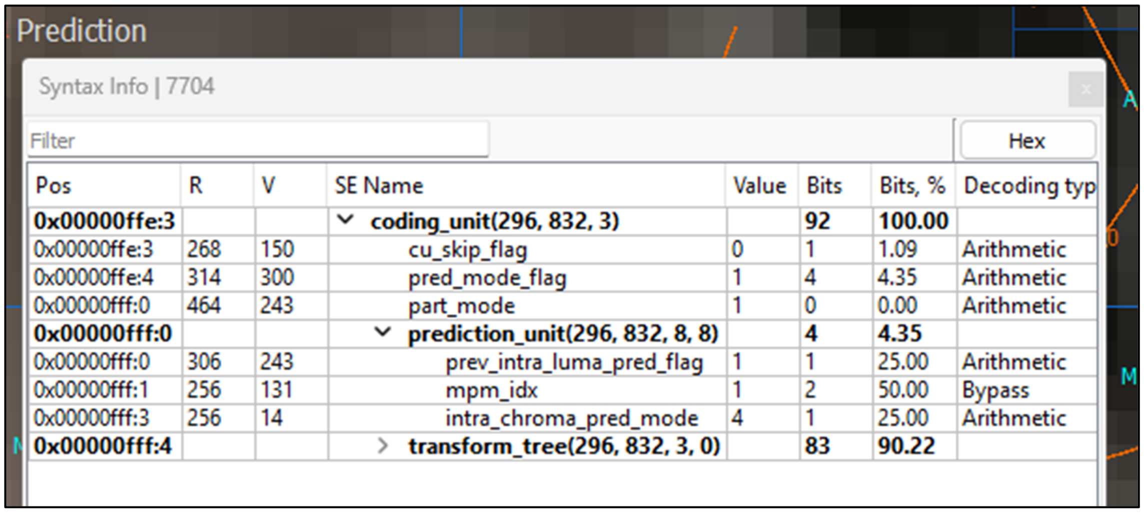
The binarization for the syntax element intra_chroma_pred_mode is specified in Table 9-41.

Table 9-41 – Binarization for intra_chroma_pred_mode

Value of intra_chroma_pred_mode	Bin string
4	0
0	100
1	101
2	110
3	111

Ex. 13 (H.265 specification) at 203.

124. For example, compressed video data captured from Snap shows the use of intra_chroma_pred_mode values, and that intra_chroma_pred_mode is at times set to 4.

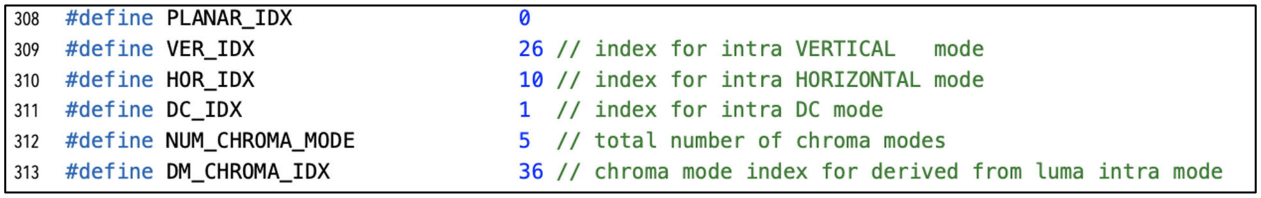


Pos	R	V	SE Name	Value	Bits	Bits, %	Decoding typ
0x00000ffe:3			**coding_unit(296, 832, 3)**		**92**	**100.00**	
0x00000ffe:3	268	150	cu_skip_flag	0	1	1.09	Arithmetic
0x00000ffe:4	314	300	pred_mode_flag	1	4	4.35	Arithmetic
0x00000fff:0	464	243	part_mode	1	0	0.00	Arithmetic
0x00000fff:0			**prediction_unit(296, 832, 8, 8)**		**4**	**4.35**	
0x00000fff:0	306	243	prev_intra_luma_pred_flag	1	1	25.00	Arithmetic
0x00000fff:1	256	131	mpm_idx	1	2	50.00	Bypass
0x00000fff:3	256	14	intra_chroma_pred_mode	4	1	25.00	Arithmetic
0x00000fff:4			**transform_tree(296, 832, 3, 0)**		**83**	**90.22**	

VQ Analyzer screenshot of bitstream collected from Snapchat.

125. For example, the source code for the x265 encoder used by Snap shows the encoding of the chroma intra prediction mode in the Entropy::codeIntraDirChroma function. Ex. 16 (entropy.cpp) at 30-31.

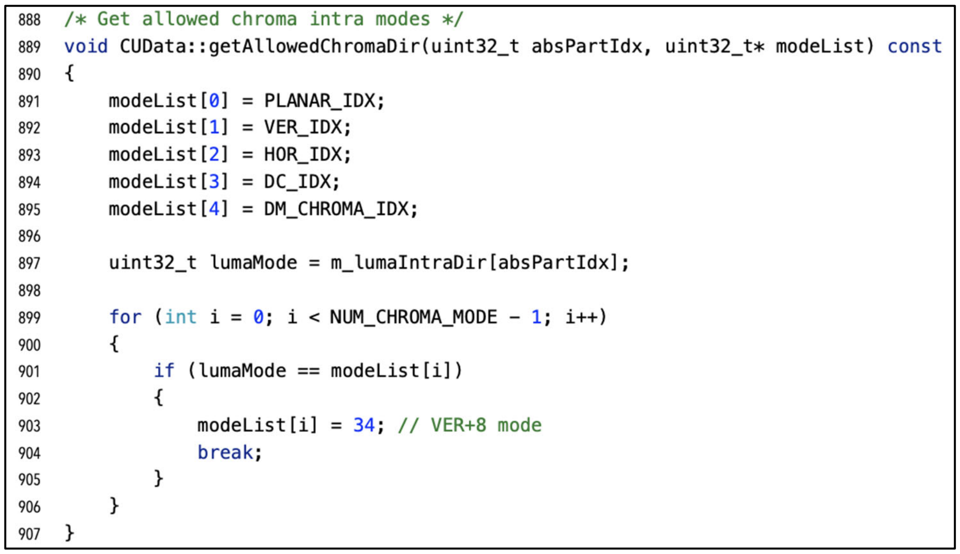
126. The source code defines a value DM_CHROMA_IDX as representing a chroma mode index derived from the luma intra mode, *i.e.*, mode 4 in Tables 8-2 and 9-41 shown above.



```
#define PLANAR_IDX                    0
#define VER_IDX                       26 // index for intra VERTICAL   mode
#define HOR_IDX                       10 // index for intra HORIZONTAL mode
#define DC_IDX                        1  // index for intra DC mode
#define NUM_CHROMA_MODE               5  // total number of chroma modes
#define DM_CHROMA_IDX                 36 // chroma mode index for derived from luma intra mode
```

Ex. 17 (common.h) at 6.

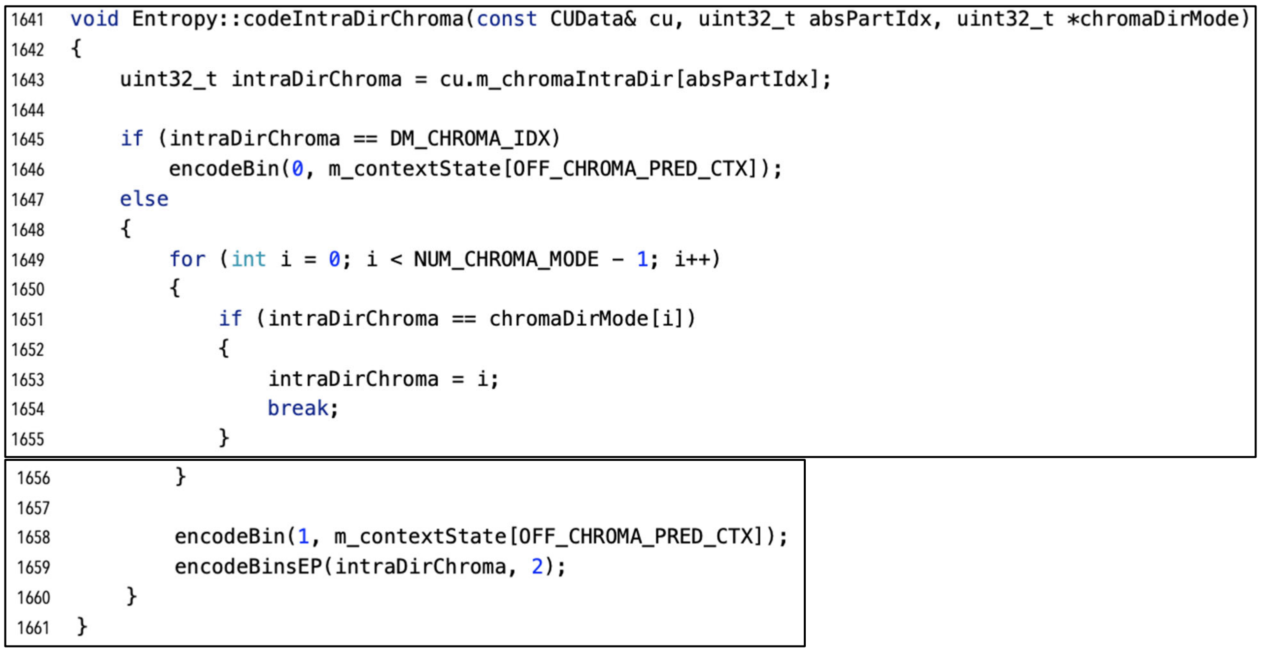
127. The source code shows that the values correspond to each of the modes in Tables 8-2 and 9-41 shown above. At line 895, the DM_CHROMA_INDEX value is assigned position 4 in "modelist," corresponding to mode 4 in Tables 8-2 and 9-41.



```
888  /* Get allowed chroma intra modes */
889  void CUData::getAllowedChromaDir(uint32_t absPartIdx, uint32_t* modeList) const
890  {
891      modeList[0] = PLANAR_IDX;
892      modeList[1] = VER_IDX;
893      modeList[2] = HOR_IDX;
894      modeList[3] = DC_IDX;
895      modeList[4] = DM_CHROMA_IDX;
896
897      uint32_t lumaMode = m_lumaIntraDir[absPartIdx];
898
899      for (int i = 0; i < NUM_CHROMA_MODE - 1; i++)
900      {
901          if (lumaMode == modeList[i])
902          {
903              modeList[i] = 34; // VER+8 mode
904              break;
905          }
906      }
907  }
```

Ex. 18 (cudata.cpp) at 17.

128. The source code shows that in the case where intraDirChroma is equal to DM_CHROMA_IDX, the chroma direction is coded with a single bit of "0" at line 1646, also corresponding to chroma mode index 4 in Table 9-41 shown above; in contrast, a "1" is encoded at line 1658 if the chroma mode is to be explicitly signaled rather than copied from the luma mode.

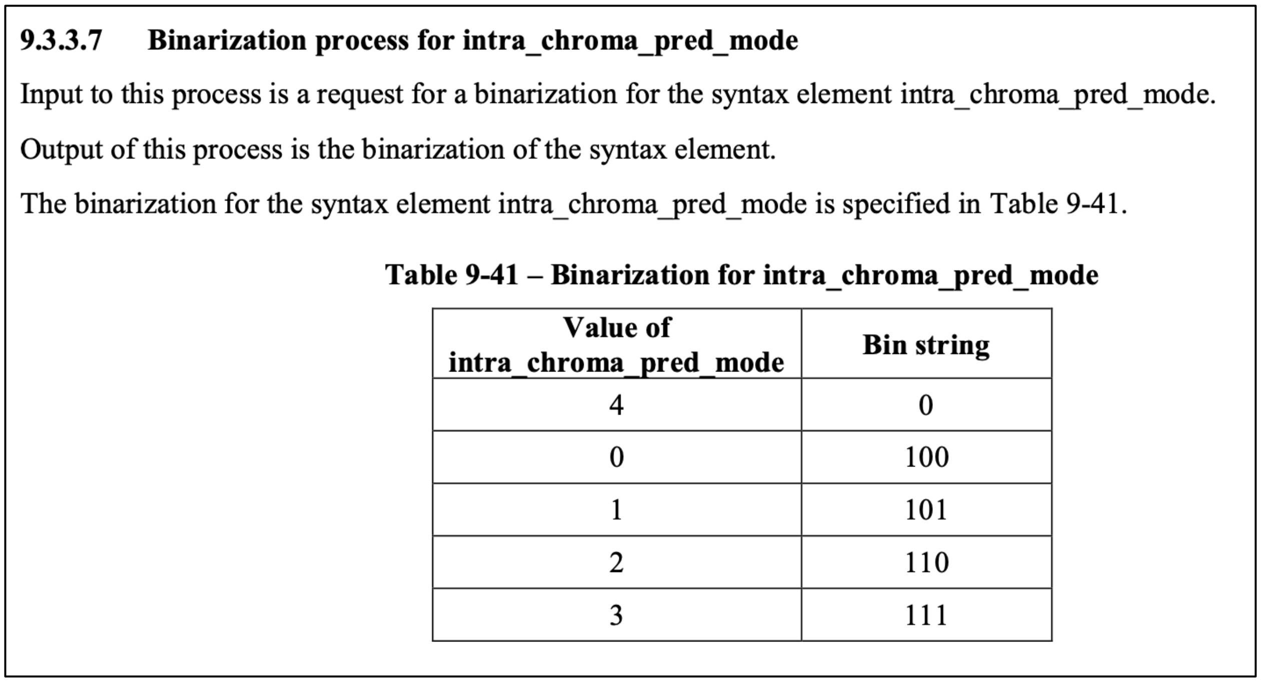


```
void Entropy::codeIntraDirChroma(const CUData& cu, uint32_t absPartIdx, uint32_t *chromaDirMode)
{
    uint32_t intraDirChroma = cu.m_chromaIntraDir[absPartIdx];

    if (intraDirChroma == DM_CHROMA_IDX)
        encodeBin(0, m_contextState[OFF_CHROMA_PRED_CTX]);
    else
    {
        for (int i = 0; i < NUM_CHROMA_MODE - 1; i++)
        {
            if (intraDirChroma == chromaDirMode[i])
            {
                intraDirChroma = i;
                break;
            }
        }

        encodeBin(1, m_contextState[OFF_CHROMA_PRED_CTX]);
        encodeBinsEP(intraDirChroma, 2);
    }
}
```

Ex. 16 (entropy.cpp) at 30-31.

129. Snap's Accused Services perform "responsive to a determination based on the inter-plane interchange information that the second intra coding parameter is not to be derived from the first intra coding parameter, inserting, into the data stream, the second set of coding parameters including the second intra coding parameter."

130. For example, the H.265 standard provides for explicit signaling of the chroma prediction mode when intra_chroma_pred_mode is 0, 1, 2, or 3. In this scenario, the second and third bits of the binarization indicate the coding parameters to be used for the chroma prediction mode.



9.3.3.7 Binarization process for intra_chroma_pred_mode

Input to this process is a request for a binarization for the syntax element intra_chroma_pred_mode.

Output of this process is the binarization of the syntax element.

The binarization for the syntax element intra_chroma_pred_mode is specified in Table 9-41.

Table 9-41 – Binarization for intra_chroma_pred_mode

Value of intra_chroma_pred_mode	Bin string
4	0
0	100
1	101
2	110
3	111

Ex. 13 (H.265 specification) at 203.

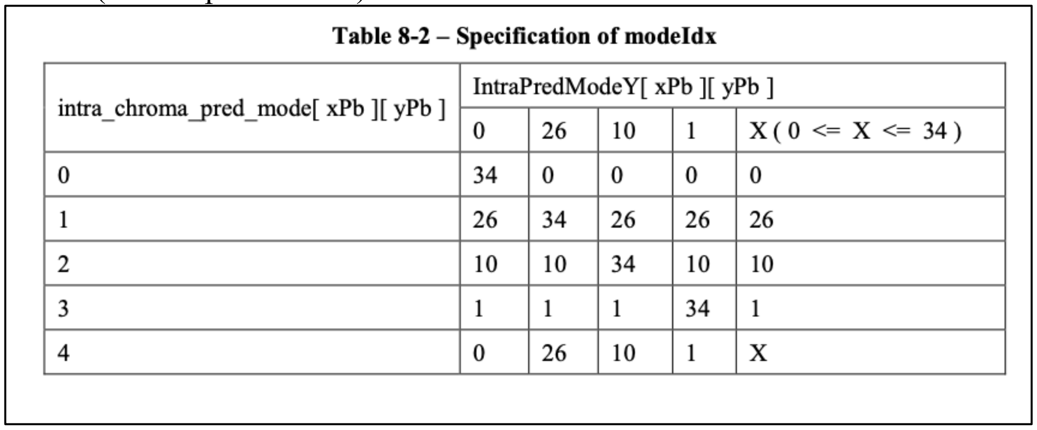
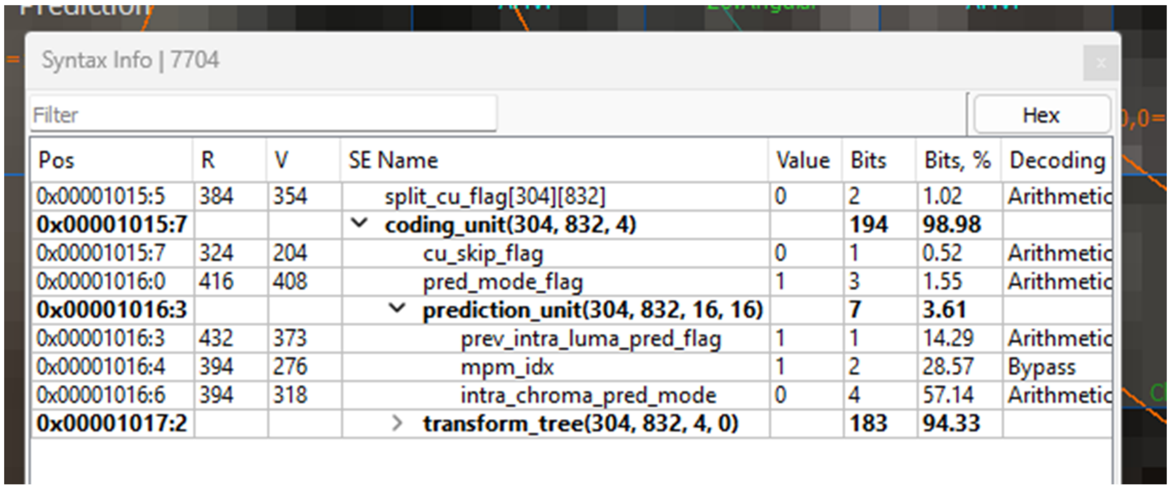


Table 8-2 – Specification of modeIdx

intra_chroma_pred_mode[xPb][yPb]	IntraPredModeY[xPb][yPb]				
	0	26	10	1	X (0 <= X <= 34)
0	34	0	0	0	0
1	26	34	26	26	26
2	10	10	34	10	10
3	1	1	1	34	1
4	0	26	10	1	X

Ex. 13 (H.265 specification) at 120.

131. For example, compressed video data captured from Snap shows cases where intra_chroma_pred_mode has a value other than 4 (0, in the example below), and therefore the chroma prediction mode is explicitly signaled.


Syntax Info | 7704

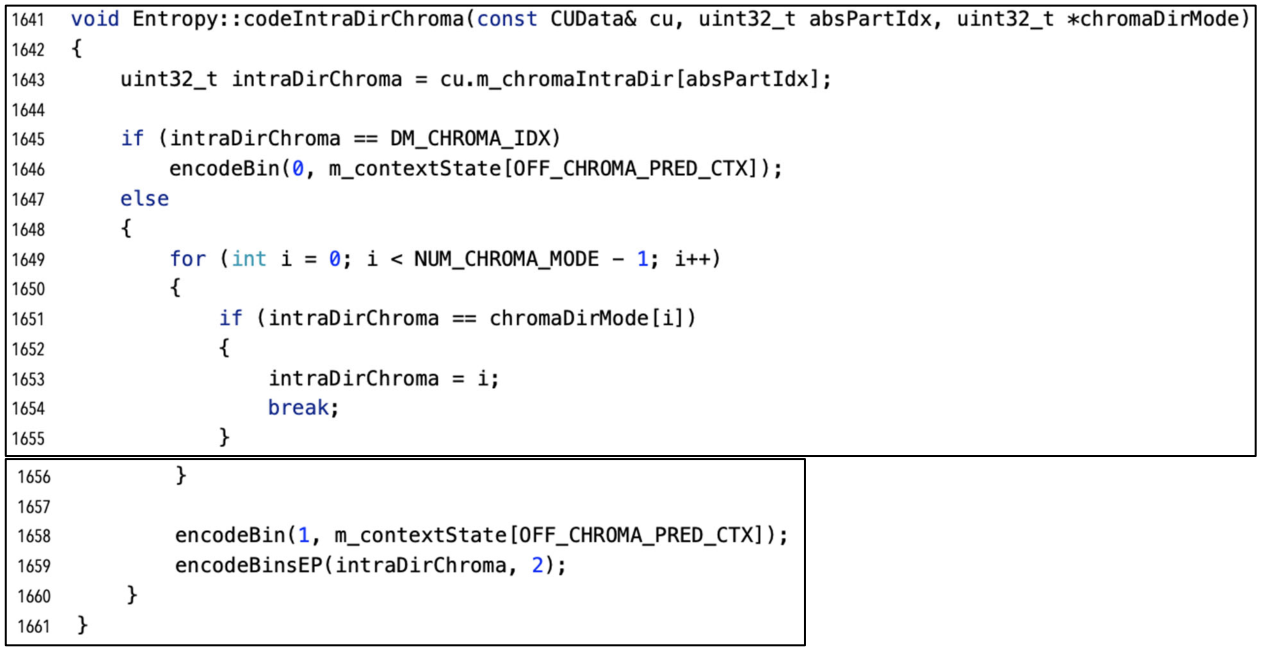
Filter

Hex

Pos	R	V	SE Name	Value	Bits	Bits, %	Decoding
0x00001015:5	384	354	split_cu_flag[304][832]	0	2	1.02	Arithmetic
0x00001015:7			**coding_unit(304, 832, 4)**		**194**	**98.98**	
0x00001015:7	324	204	cu_skip_flag	0	1	0.52	Arithmetic
0x00001016:0	416	408	pred_mode_flag	1	3	1.55	Arithmetic
0x00001016:3			**prediction_unit(304, 832, 16, 16)**		**7**	**3.61**	
0x00001016:3	432	373	prev_intra_luma_pred_flag	1	1	14.29	Arithmetic
0x00001016:4	394	276	mpm_idx	1	2	28.57	Bypass
0x00001016:6	394	318	intra_chroma_pred_mode	0	4	57.14	Arithmetic
0x00001017:2			**transform_tree(304, 832, 4, 0)**		**183**	**94.33**	

VQ Analyzer screenshot of bitstream collected from Snapchat.

132. For example, the source code for the x265 encoder used by Snap shows the encoding of the chroma intra prediction mode in the Entropy::codeIntraDirChroma function. Ex. 16 (entropy.cpp) at 30-31. After a "1" is encoded at line 1658 to indicate the chroma mode is to be explicitly signaled rather than copied from the luma mode, two additional bits are encoded at line 1659 to indicate which intra prediction direction is to be used for chroma.


```
1641 void Entropy::codeIntraDirChroma(const CUData& cu, uint32_t absPartIdx, uint32_t *chromaDirMode)
1642 {
1643     uint32_t intraDirChroma = cu.m_chromaIntraDir[absPartIdx];
1644
1645     if (intraDirChroma == DM_CHROMA_IDX)
1646         encodeBin(0, m_contextState[OFF_CHROMA_PRED_CTX]);
1647     else
1648     {
1649         for (int i = 0; i < NUM_CHROMA_MODE - 1; i++)
1650         {
1651             if (intraDirChroma == chromaDirMode[i])
1652             {
1653                 intraDirChroma = i;
1654                 break;
1655             }
1656         }
1657
1658         encodeBin(1, m_contextState[OFF_CHROMA_PRED_CTX]);
1659         encodeBinsEP(intraDirChroma, 2);
1660     }
1661 }
```
Ex. 16 (entropy.cpp) at 30-31.

133. Snap's Accused Services perform "copying, responsive to a determination based on the inter-

plane interchange information that the second intra coding parameter is to be derived from the first intra coding parameter, the first intra coding parameter as the second intra coding parameter such that the first and second intra coding parameters are equal."

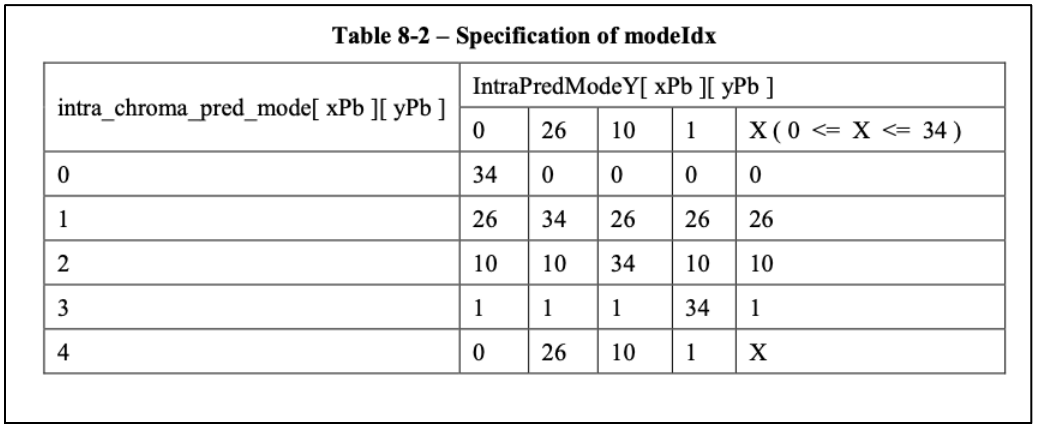


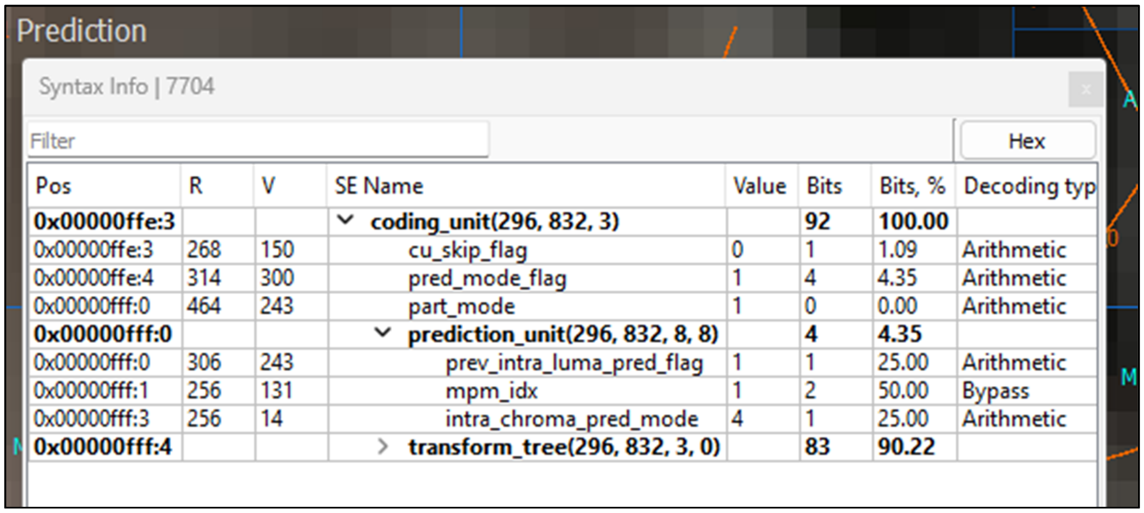
Table 8-2 – Specification of modeIdx

intra_chroma_pred_mode[xPb][yPb]	IntraPredModeY[xPb][yPb]				
	0	26	10	1	X (0 <= X <= 34)
0	34	0	0	0	0
1	26	34	26	26	26
2	10	10	34	10	10
3	1	1	1	34	1
4	0	26	10	1	X

Ex. 13 (H.265 specification) at 120.

134. For example, in order to calculate the predicted block and therefore compute the residual, in the case where intra_chroma_pred_flag is 4, the encoder must copy the luma prediction information and use it to predict the chroma block in order to compute the residual. The H.265 standard explains that where intra_chroma_pred_flag is 4, the luma and chroma parameters intraPredModeY and modeIdx will be equal.

135. For example, compressed video data captured from Snap shows that after an intra_chroma_pred_mode of 4 is signaled, a transform_tree is coded providing residual data. Snap must compute the chroma prediction according to the copied luma prediction mode in order to compute this residual data.



Pos	R	V	SE Name	Value	Bits	Bits, %	Decoding typ
0x00000ffe:3			**coding_unit(296, 832, 3)**		**92**	**100.00**	
0x00000ffe:3	268	150	cu_skip_flag	0	1	1.09	Arithmetic
0x00000ffe:4	314	300	pred_mode_flag	1	4	4.35	Arithmetic
0x00000fff:0	464	243	part_mode	1	0	0.00	Arithmetic
0x00000fff:0			**prediction_unit(296, 832, 8, 8)**		**4**	**4.35**	
0x00000fff:0	306	243	prev_intra_luma_pred_flag	1	1	25.00	Arithmetic
0x00000fff:1	256	131	mpm_idx	1	2	50.00	Bypass
0x00000fff:3	256	14	intra_chroma_pred_mode	4	1	25.00	Arithmetic
0x00000fff:4			**transform_tree(296, 832, 3, 0)**		**83**	**90.22**	

VQ Analyzer screenshot of bitstream collected from Snapchat.

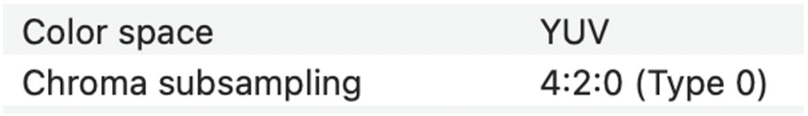
136. Source code from the x265 encoder confirms that the luma parameter is copied for chroma prediction as part of the encoding process. For example:

```
                if (chromaPredMode == DM_CHROMA_IDX)
                    chromaPredMode = cu.m_lumaIntraDir[(m_csp == X265_CSP_I444) ? absPartIdxC : 0];
```

Ex. 19 (search.cpp) at 17; *see also id.* at 18, 22, 32 (showing similar copying).

137. In Snap's Accused Services, "a spatial resolution of the first array is twice a spatial resolution of the second array."

138. For example, Snap uses 4:2:0 chroma subsampling in its videos.



Color space	YUV
Chroma subsampling	4:2:0 (Type 0)

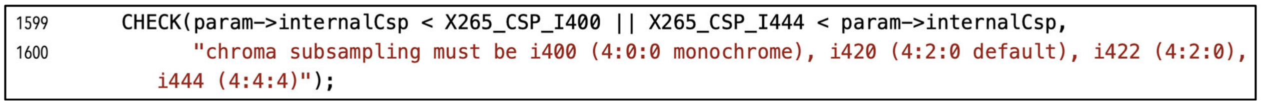
MediaInfo screenshot of bitstream collected from Snapchat.

139. The H.265 specification explains that 4:2:0 sampling results in the luma array having twice the spatial resolution in each dimension compared to the chroma array.

> In monochrome sampling there is only one sample array, which is nominally considered the luma array.
> In 4:2:0 sampling, each of the two chroma arrays has half the height and half the width of the luma array.

Ex. 13 (H.265 specification) at 21.

140. Source code for the x265 encoder indicates that 4:2:0 color subsampling is the default configuration.



```
    CHECK(param->internalCsp < X265_CSP_I400 || X265_CSP_I444 < param->internalCsp,
          "chroma subsampling must be i400 (4:0:0 monochrome), i420 (4:2:0 default), i422 (4:2:0),
       i444 (4:4:4)");
```

Ex. 20 (param.cpp) at 29.

141. In Snap's Accused Services, "the first intra coding parameter indicates an angle of direction of intra prediction used in the intra coding mode within the first array and the second intra coding parameter indicates an angle of direction of intra prediction used in the intra coding mode within the second array."

142. For example, the H.265 specification indicates that both luma and chroma prediction modes indicate an angle of direction of intra prediction.

Table 8-1 specifies the value for the intra prediction mode and the associated names.

Table 8-1 – Specification of intra prediction mode and associated names

Intra prediction mode	Associated name
0	INTRA_PLANAR
1	INTRA_DC
2..34	INTRA_ANGULAR2..INTRA_ANGULAR34

IntraPredModeY[xPb][yPb] labelled 0..34 represents directions of predictions as illustrated in Figure 8-1.

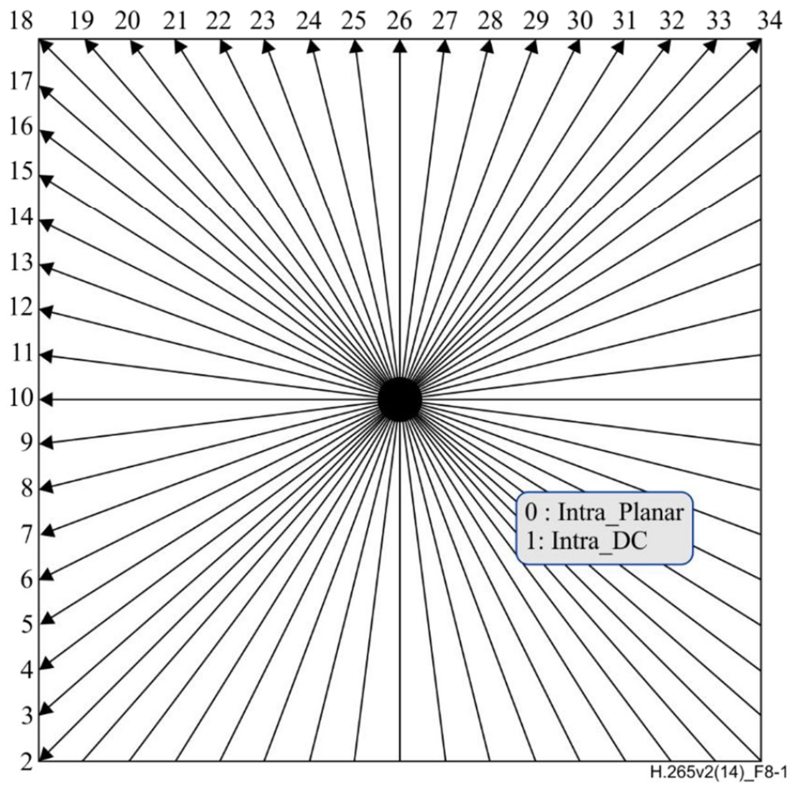


Figure 8-1 – Intra prediction mode directions (informative)

Ex. 13 (H.265 specification) at 118.

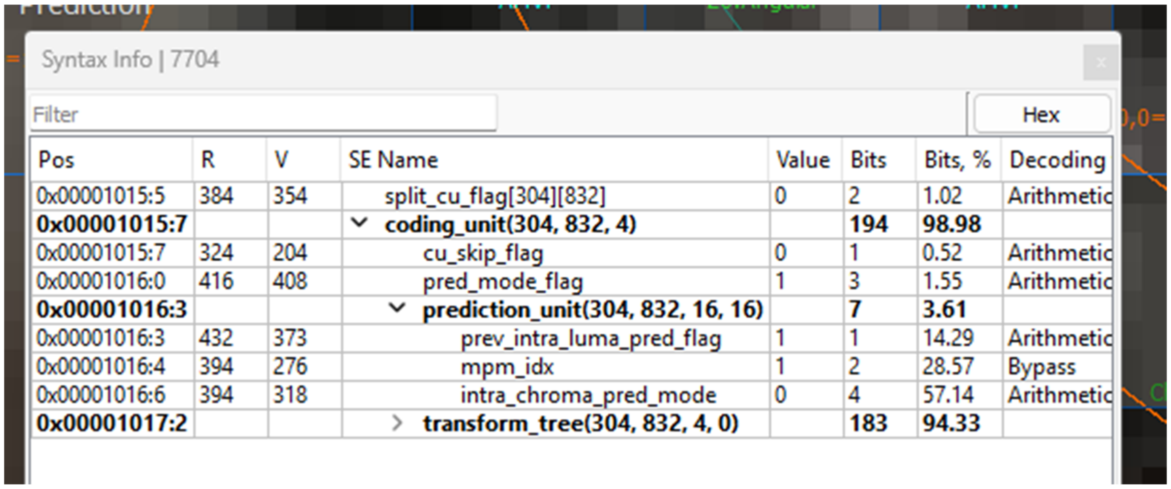
143. The source code similarly refers to the luma and chroma prediction modes at "intraDirLumaAng" and "intraDirChroma," confirming that they represent directions. Ex. 16 (entropy.cpp) at 29-31.

144. Snap's Accused Services perform "predicting the second coding block based on the second

set of coding parameters including the second intra coding parameter to generate a predicted second coding block based on the intra coding mode."

145. For example, the encoder must predict the second coding block in order to calculate the residuals. It must use the second intra coding parameter to accurately do so.

146. For example, compressed video data captured from Snap shows that after an intra_chroma_pred_mode is coded with an explicit intra coding parameter, a transform_tree is also coded providing residual data. Snap must compute the chroma prediction according to the correct prediction mode in order to correctly compute this residual data.


Syntax Info | 7704

Pos	R	V	SE Name	Value	Bits	Bits, %	Decoding
0x00001015:5	384	354	split_cu_flag[304][832]	0	2	1.02	Arithmetic
0x00001015:7			**coding_unit(304, 832, 4)**		**194**	**98.98**	
0x00001015:7	324	204	cu_skip_flag	0	1	0.52	Arithmetic
0x00001016:0	416	408	pred_mode_flag	1	3	1.55	Arithmetic
0x00001016:3			**prediction_unit(304, 832, 16, 16)**		**7**	**3.61**	
0x00001016:3	432	373	prev_intra_luma_pred_flag	1	1	14.29	Arithmetic
0x00001016:4	394	276	mpm_idx	1	2	28.57	Bypass
0x00001016:6	394	318	intra_chroma_pred_mode	0	4	57.14	Arithmetic
0x00001017:2			**transform_tree(304, 832, 4, 0)**		**183**	**94.33**	

VQ Analyzer screenshot of bitstream collected from Snapchat.

147. Source code from the x265 encoder shows that these predictions are used to determine residual data. *See, e.g.,* Ex. 19 (search.cpp) at 13.

```
    // generate chroma prediction, generate residual and recon
    void      codeIntraChromaQt(Mode& mode, const CUGeom& cuGeom, uint32_t tuDepth, uint32_t
  absPartIdx, Cost& outCost);
    void      codeIntraChromaTSkip(Mode& mode, const CUGeom& cuGeom, uint32_t tuDepth, uint32_t
  tuDepthC, uint32_t absPartIdx, Cost& outCost);
    void      extractIntraResultChromaQT(CUData& cu, Yuv& reconYuv, uint32_t absPartIdx, uint32_t
  tuDepth);
```

Ex. 21 (search.h) at 8.

148. In Snap's Accused Services, "the first and second arrays of information samples represent different types of spatially sampled information of the scene."

149. For example, as discussed above, the arrays represent luma and chroma samples, which are different types of information samples.

COUNT II: PATENT INFRINGEMENT OF THE ’193 PATENT

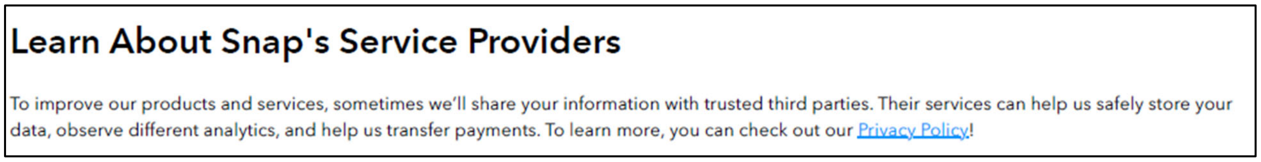
150. Dolby incorporates by reference the preceding paragraphs as though fully set forth herein.

151. Snap had knowledge and notice of the ’193 Patent at least by November 24, 2023, and again through communications on August 13, 2025 and November 21, 2025.

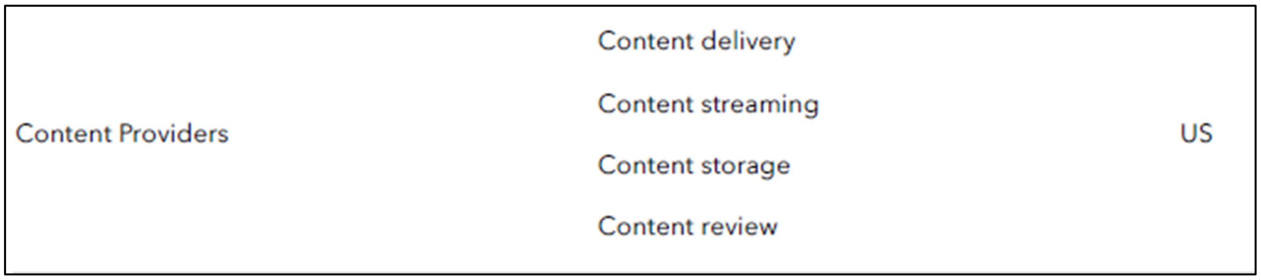
152. Snap infringes the ’193 Patent by making, using, offering to sell, and/or selling the patented invention within the United States, and/or importing the patented invention into the United States. Snap thus directly infringes the ’193 Patent literally and/or under the Doctrine of Equivalents, in violation of 35 U.S.C. § 271. For example, Snap directly infringes the ’193 patent through its encoding, decoding, and transcoding of H.265-compliant (HEVC) video in the United States.

153. On information and belief, Snap also indirectly infringes the ’193 Patent, as provided in 35 U.S.C. § 271(b), by knowingly inducing infringement by others, such as Snap’s partners and vendors, in this District and elsewhere in the United States.

154. For example, Snap uses the infrastructure of its providers to encode and decode video. Snap’s web page indicates that these providers perform content delivery, streaming, storage, and review, including the encoding and decoding of video.


Learn About Snap's Service Providers

To improve our products and services, sometimes we'll share your information with trusted third parties. Their services can help us safely store your data, observe different analytics, and help us transfer payments. To learn more, you can check out our Privacy Policy!


Content Providers	Content delivery Content streaming Content storage Content review	US

Ex. 12 at 1.

155. As a result of Snap's inducement, Snap's providers perform infringing video coding and directly infringe the '193 Patent. Snap has performed and continues to perform these affirmative acts with knowledge of the '193 Patent and with the intent, or willful blindness, that the induced acts directly infringe the '193 Patent.

156. Upon information and belief, Snap derives revenue, directly and indirectly, from the activities relating to the Accused Services and to infringement of the '193 Patent, including in this District and elsewhere in the United States.

157. Snap's infringement of the '193 Patent is willful and deliberate. As detailed above, Snap had knowledge of the '193 Patent and had knowledge, or was willfully blind, as to its infringement of the '193 Patent. As discussed above, Snap has had actual knowledge of the '193 Patent since before this suit. Further, as discussed above, Snap knew or should have known that its actions infringe and actively induce infringement of the '193 Patent. As discussed above, Snap specifically intended that both itself and/or its customers and providers infringe the '193 Patent.

158. Snap's infringement of the '193 Patent has damaged and will continue to damage Dolby.

159. Exemplary claim 13 of the '193 Patent recites:

> 13. A method for encoding, comprising:
>
> coding, into a data stream, first information associated with a coding block, wherein the first information has first or second states, the first state indicates that (1) the coding block is to be reconstructed based on one or more coding parameters of a merge candidate coding block and (2) the coding block is to be reconstructed without residual data;
>
> when the first information is in the first state, coding, into the data stream,
>
> second information which specifies the merge candidate coding block, and
>
> the one or more coding parameters associated with the merge candidate coding block so that the one or more coding parameters of the merge candidate coding block can be used to reconstruct the coding block without residual data when the first information is in the first state; and
>
> when the first information is in the second state, coding, into the data stream,

third information associated with the coding block, the third information having first and second states,

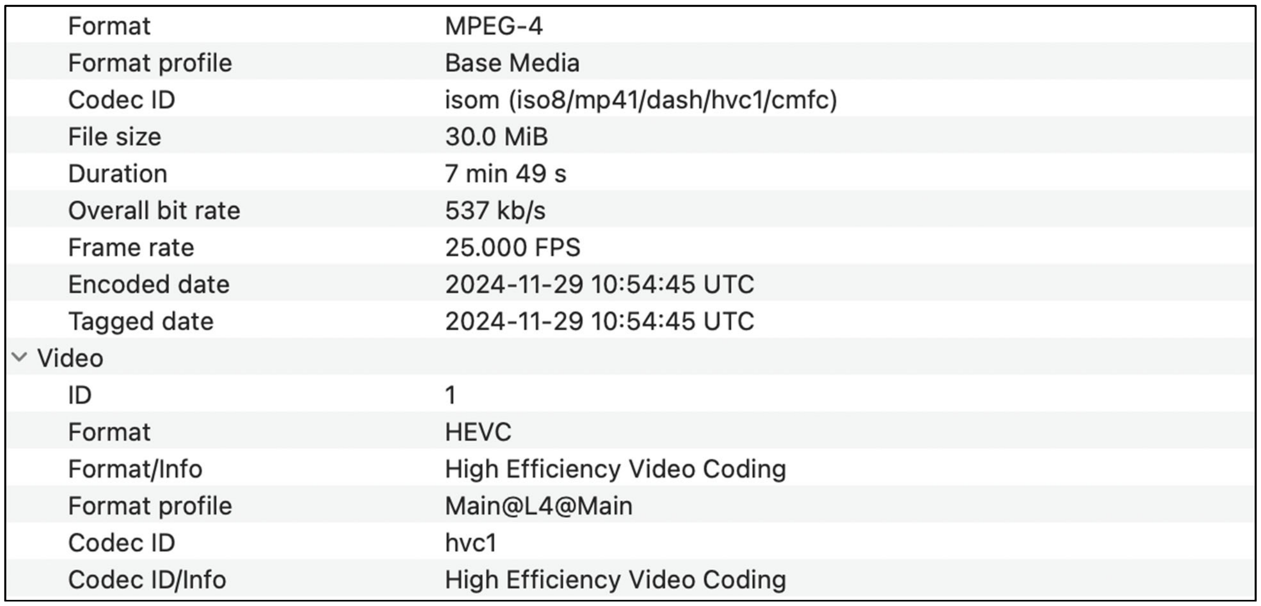
at least one coding parameter for the coding block when the third information is in the first state of the third information indicating that the coding block is to be reconstructed using the at least one coding parameter coded in the data stream, and

a set of coding parameters associated with another merge candidate coding block and residual data for the coding block when the third information is in the second state of the third information, indicating that the coding block is to be reconstructed based on the set of coding parameters and the residual data coded in the data stream.

160. Snap's Accused Services perform "a method for encoding."

161. For example, Snap's backend servers encode H.265-compliant (HEVC) video for streaming to devices.

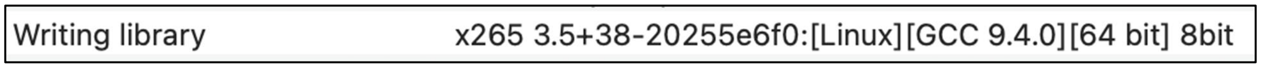
162. Videos downloaded from Snap show that an HEVC format is used:



Format	MPEG-4
Format profile	Base Media
Codec ID	isom (iso8/mp41/dash/hvc1/cmfc)
File size	30.0 MiB
Duration	7 min 49 s
Overall bit rate	537 kb/s
Frame rate	25.000 FPS
Encoded date	2024-11-29 10:54:45 UTC
Tagged date	2024-11-29 10:54:45 UTC
Video	
ID	1
Format	HEVC
Format/Info	High Efficiency Video Coding
Format profile	Main@L4@Main
Codec ID	hvc1
Codec ID/Info	High Efficiency Video Coding

(Video from Snap as viewed in MediaInfo app.)

163. Snap uses the x265 library, version 3.5, revision 20255e6f0, to encode HEVC-compliant video.



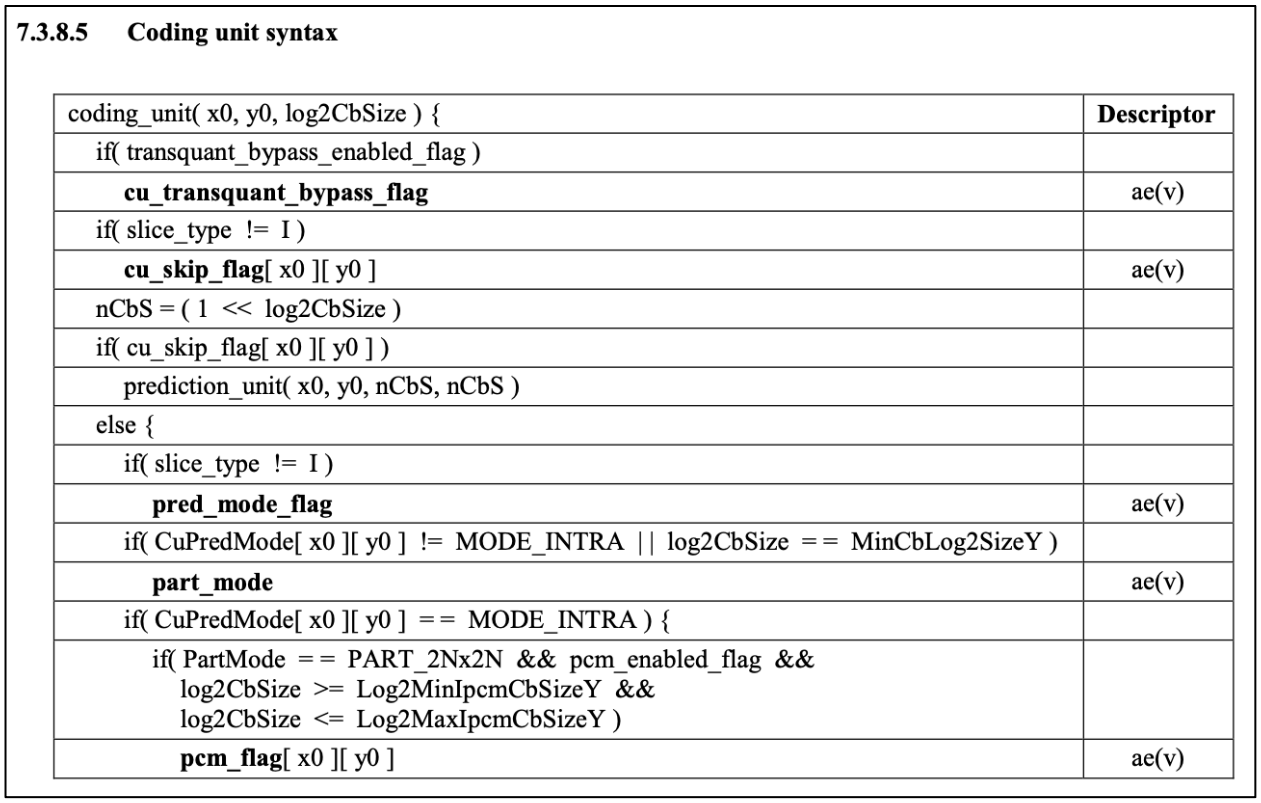
Writing library	x265 3.5+38-20255e6f0:[Linux][GCC 9.4.0][64 bit] 8bit

(Video from Snap as viewed in MediaInfo app.)

164. Snap's Accused Services perform "coding, into a data stream, first information associated

with a coding block, wherein the first information has first or second states, the first state indicates that (1) the coding block is to be reconstructed based on one or more coding parameters of a merge candidate coding block and (2) the coding block is to be reconstructed without residual data."

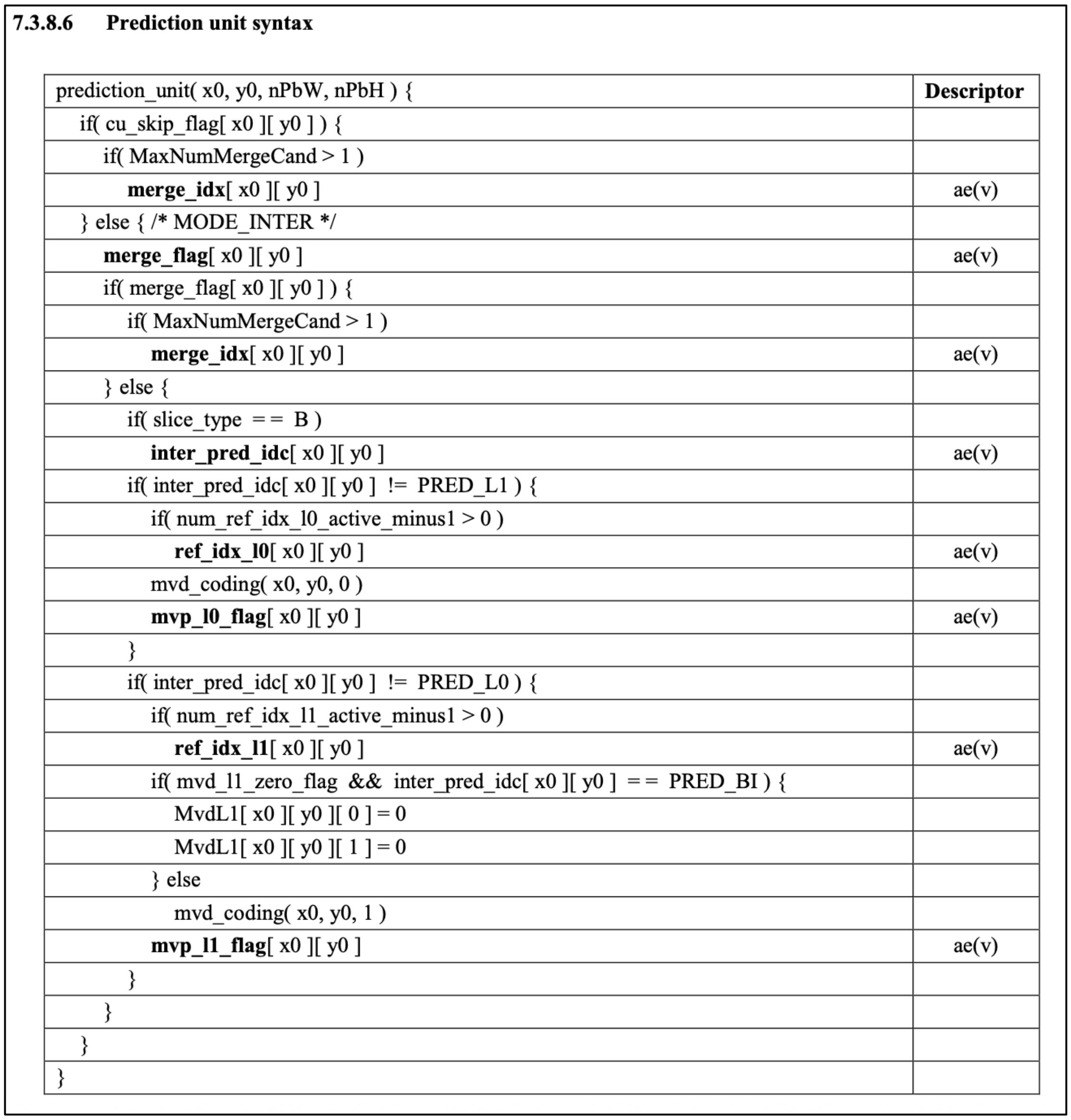
165. For example, sections 7.3.8.5, 7.3.8.6, 7.4.9.5, and 7.4.9.6 of the H.265 specification describe a cu_skip_flag element. If cu_skip_flag[x0][y0] is equal to 1, then a merge candidate indicated by a merge candidate index, merge_idx[x0][y0], is encoded to indicate which merge candidate is to be used. If cu_skip_flag[x0][y0] is equal to 0, then information related to residual data can be separately coded.



7.3.8.5 Coding unit syntax

coding_unit(x0, y0, log2CbSize) {	**Descriptor**
if(transquant_bypass_enabled_flag)	
cu_transquant_bypass_flag	ae(v)
if(slice_type != I)	
cu_skip_flag[x0][y0]	ae(v)
nCbS = (1 << log2CbSize)	
if(cu_skip_flag[x0][y0])	
prediction_unit(x0, y0, nCbS, nCbS)	
else {	
if(slice_type != I)	
pred_mode_flag	ae(v)
if(CuPredMode[x0][y0] != MODE_INTRA || log2CbSize = = MinCbLog2SizeY)	
part_mode	ae(v)
if(CuPredMode[x0][y0] = = MODE_INTRA) {	
if(PartMode = = PART_2Nx2N && pcm_enabled_flag && log2CbSize >= Log2MinIpcmCbSizeY && log2CbSize <= Log2MaxIpcmCbSizeY)	
pcm_flag[x0][y0]	ae(v)

Ex. 13 (H.265 specification) at 51; *see also id*. 52-53.

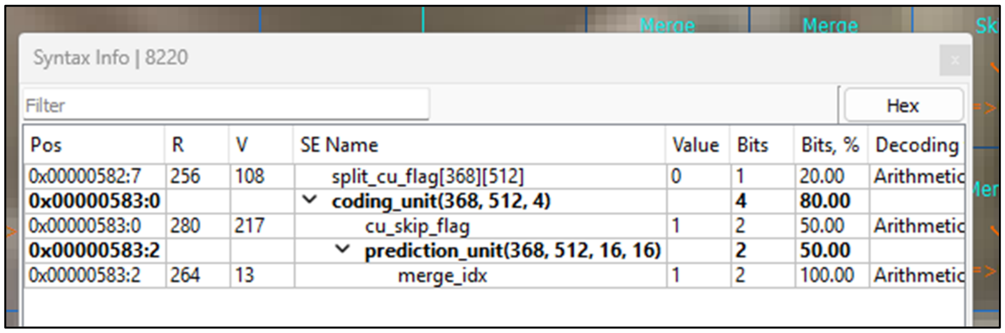


7.3.8.6 Prediction unit syntax

prediction_unit(x0, y0, nPbW, nPbH) {	**Descriptor**
if(cu_skip_flag[x0][y0]) {	
if(MaxNumMergeCand > 1)	
merge_idx[x0][y0]	ae(v)
} else { /* MODE_INTER */	
merge_flag[x0][y0]	ae(v)
if(merge_flag[x0][y0]) {	
if(MaxNumMergeCand > 1)	
merge_idx[x0][y0]	ae(v)
} else {	
if(slice_type == B)	
inter_pred_idc[x0][y0]	ae(v)
if(inter_pred_idc[x0][y0] != PRED_L1) {	
if(num_ref_idx_l0_active_minus1 > 0)	
ref_idx_l0[x0][y0]	ae(v)
mvd_coding(x0, y0, 0)	
mvp_l0_flag[x0][y0]	ae(v)
}	
if(inter_pred_idc[x0][y0] != PRED_L0) {	
if(num_ref_idx_l1_active_minus1 > 0)	
ref_idx_l1[x0][y0]	ae(v)
if(mvd_l1_zero_flag && inter_pred_idc[x0][y0] == PRED_BI) {	
MvdL1[x0][y0][0] = 0	
MvdL1[x0][y0][1] = 0	
} else	
mvd_coding(x0, y0, 1)	
mvp_l1_flag[x0][y0]	ae(v)
}	
}	
}	
}	

Ex. 13 (H.265 specification) at 54.

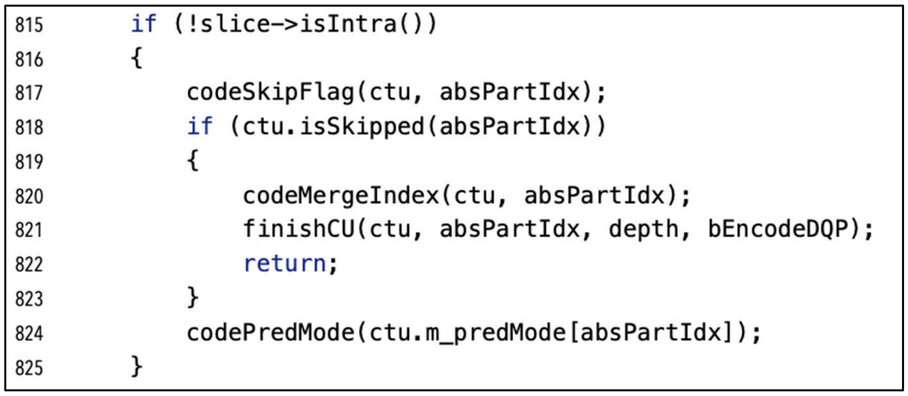
166. For example, compressed video data captured from Snap shows examples where cu_skip_flag is set to 1. In these cases, a merge_idx is sent and no residual data for the block is sent.



Pos	R	V	SE Name	Value	Bits	Bits, %	Decoding
0x00000582:7	256	108	split_cu_flag[368][512]	0	1	20.00	Arithmetic
0x00000583:0			**coding_unit(368, 512, 4)**		**4**	**80.00**	
0x00000583:0	280	217	cu_skip_flag	1	2	50.00	Arithmetic
0x00000583:2			**prediction_unit(368, 512, 16, 16)**		**2**	**50.00**	
0x00000583:2	264	13	merge_idx	1	2	100.00	Arithmetic

VQ Analyzer screenshot of bitstream collected from Snapchat.

167. For example, the source code for the x265 encoder used by Snap shows that a skip flag is coded, and if the CU is skipped, then a merge index is encoded, the CU is finished, and the function is exited without continuing on to encode residual data.



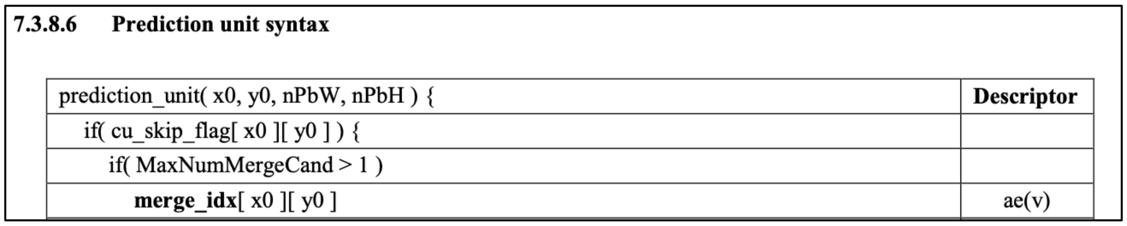
```
    if (!slice->isIntra())
    {
        codeSkipFlag(ctu, absPartIdx);
        if (ctu.isSkipped(absPartIdx))
        {
            codeMergeIndex(ctu, absPartIdx);
            finishCU(ctu, absPartIdx, depth, bEncodeDQP);
            return;
        }
        codePredMode(ctu.m_predMode[absPartIdx]);
    }
```

Ex. 16 (entropy.cpp) at 15.

168. Snap's Accused Services perform "when the first information is in the first state, coding, into the data stream, second information which specifies the merge candidate coding block, and the one or more coding parameters associated with the merge candidate coding block so that the one or more coding parameters of the merge candidate coding block can be used to reconstruct the coding block without residual data when the first information is in the first state."

169. As shown above, when cu_skip_flag is set, and video encoded by Snap's Accused Services include a merge_idx for the CU but no residual data for the CU.

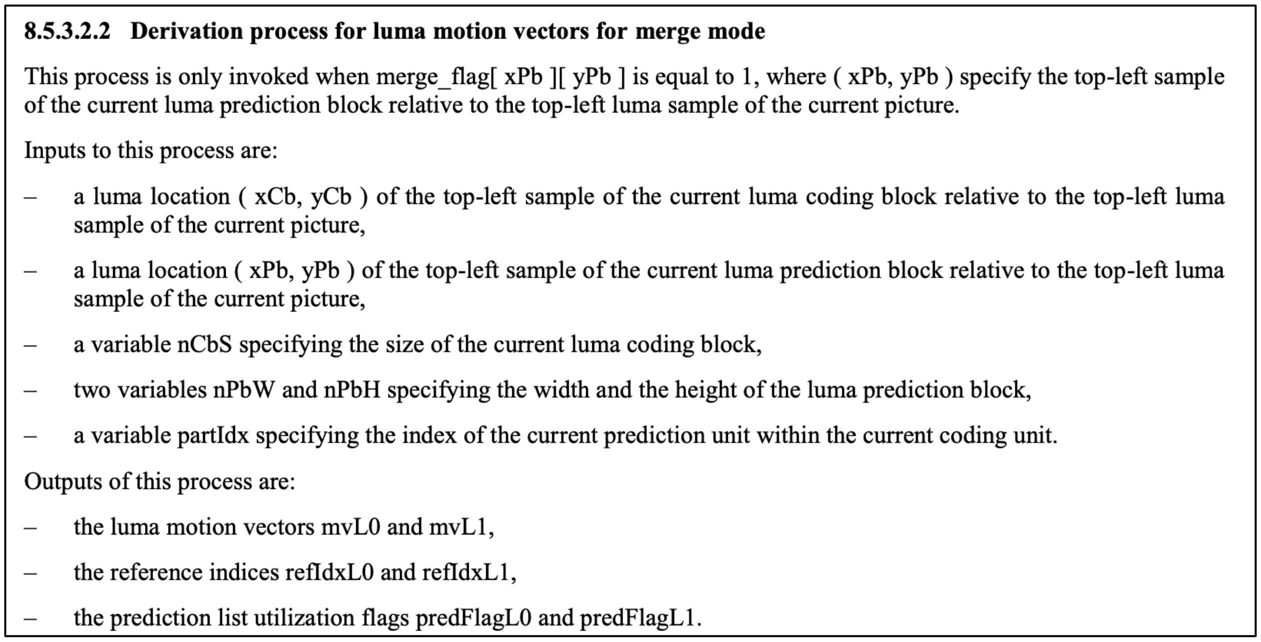
170. For example, the H.265 standard discloses that when flag cu_skip_flag is equal to 1 (or true), the merge candidate index, merge_idx[x0][y0] (i.e., the claimed second information) is determined and encoded into the data stream. Further, the one or more parameters (e.g., motion parameters) associated with the merge candidate block are coded.



7.3.8.6 Prediction unit syntax

prediction_unit(x0, y0, nPbW, nPbH) {	**Descriptor**
if(cu_skip_flag[x0][y0]) {	
if(MaxNumMergeCand > 1)	
merge_idx[x0][y0]	ae(v)

Ex. 13 (H.265 specification) at 54.



8.5.3.2.2 Derivation process for luma motion vectors for merge mode

This process is only invoked when merge_flag[xPb][yPb] is equal to 1, where (xPb, yPb) specify the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture.

Inputs to this process are:

– a luma location (xCb, yCb) of the top-left sample of the current luma coding block relative to the top-left luma sample of the current picture,

– a luma location (xPb, yPb) of the top-left sample of the current luma prediction block relative to the top-left luma sample of the current picture,

– a variable nCbS specifying the size of the current luma coding block,

– two variables nPbW and nPbH specifying the width and the height of the luma prediction block,

– a variable partIdx specifying the index of the current prediction unit within the current coding unit.

Outputs of this process are:

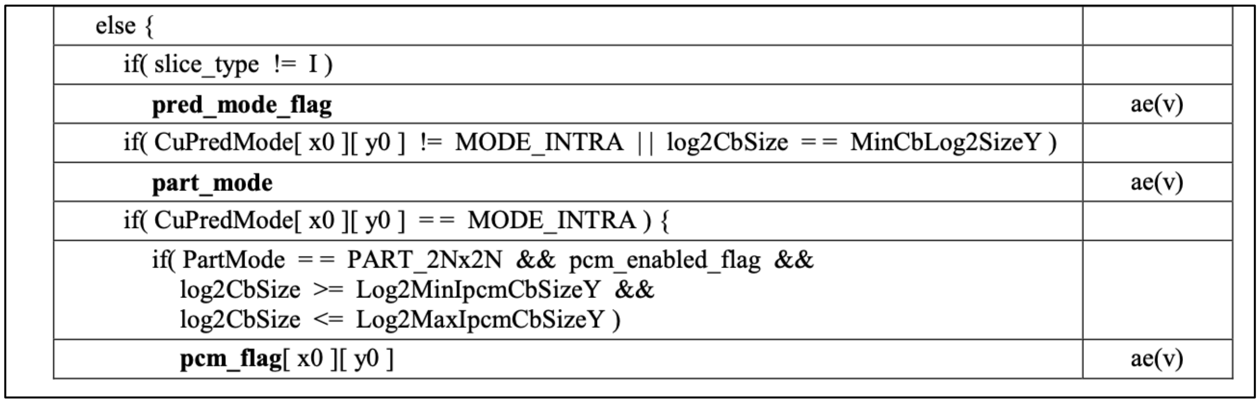
– the luma motion vectors mvL0 and mvL1,

– the reference indices refIdxL0 and refIdxL1,

– the prediction list utilization flags predFlagL0 and predFlagL1.

Ex. 13 (H.265 specification) at 133.

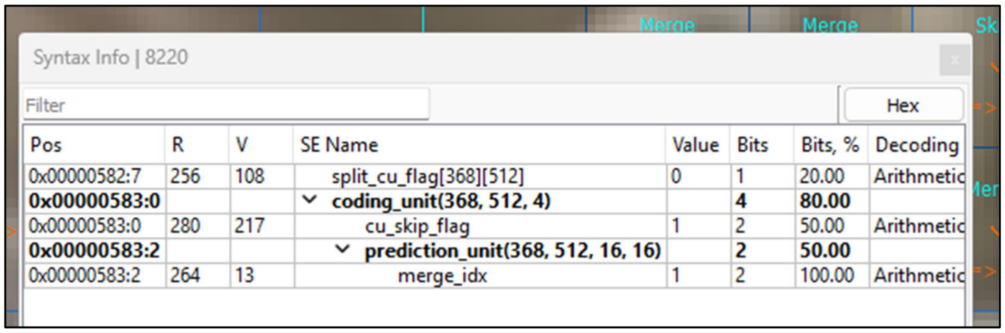
171. For example, parameters for the merge block are encoded through syntax elements such as pred_mode_flag, part_mode, and pcm_flag.



else {	
if(slice_type != I)	
pred_mode_flag	ae(v)
if(CuPredMode[x0][y0] != MODE_INTRA || log2CbSize == MinCbLog2SizeY)	
part_mode	ae(v)
if(CuPredMode[x0][y0] == MODE_INTRA) {	
if(PartMode == PART_2Nx2N && pcm_enabled_flag && log2CbSize >= Log2MinIpcmCbSizeY && log2CbSize <= Log2MaxIpcmCbSizeY)	
pcm_flag[x0][y0]	ae(v)

Ex. 13 (H.265 specification) at 51.

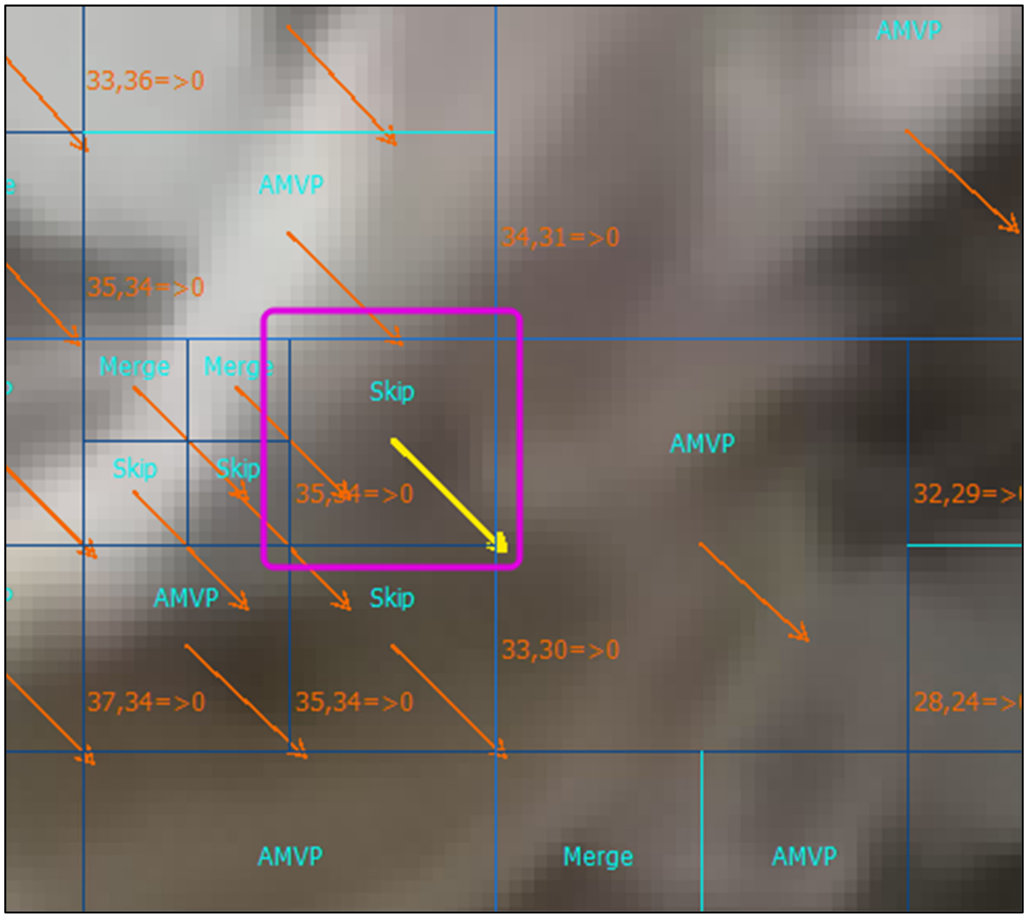
172. For example, compressed video data captured from Snap shows examples where cu_skip_flag is set to 1, and a merge block is selected.


Syntax Info | 8220

Pos	R	V	SE Name	Value	Bits	Bits, %	Decoding
0x00000582:7	256	108	split_cu_flag[368][512]	0	1	20.00	Arithmetic
0x00000583:0			**coding_unit(368, 512, 4)**		**4**	**80.00**	
0x00000583:0	280	217	cu_skip_flag	1	2	50.00	Arithmetic
0x00000583:2			**prediction_unit(368, 512, 16, 16)**		**2**	**50.00**	
0x00000583:2	264	13	merge_idx	1	2	100.00	Arithmetic

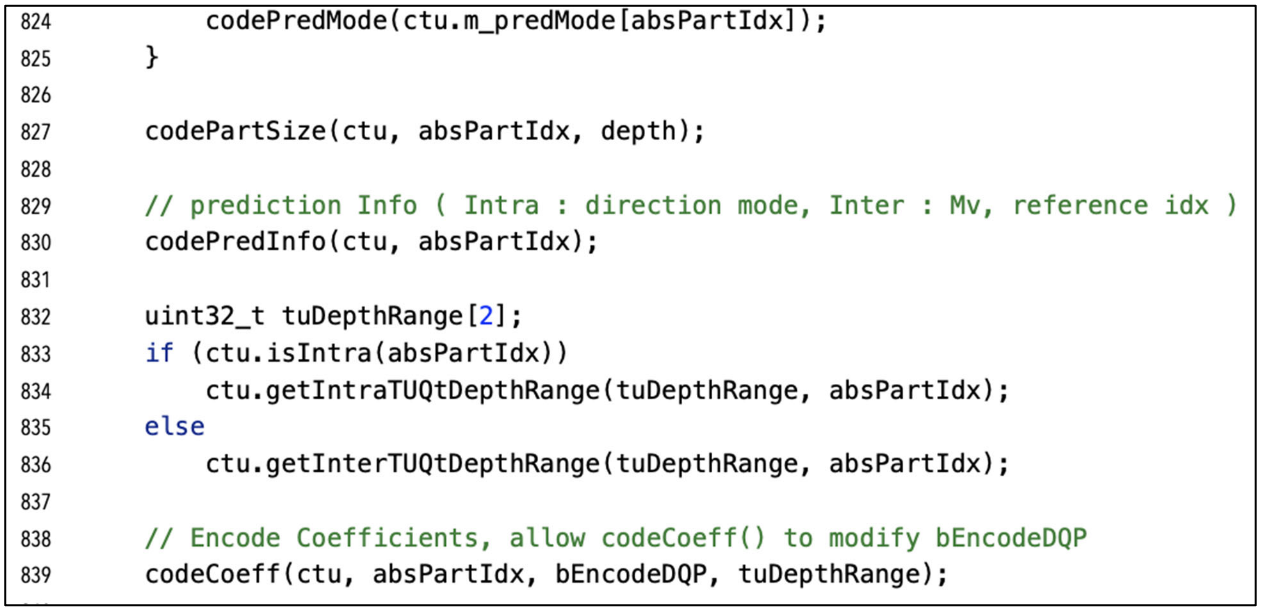
VQ Analyzer screenshot of bitstream collected from Snapchat.

173. VQ Analyzer shows that this block is merged with another block, for which parameters are explicitly coded for Advanced Motion Vector Prediction (AMVP).



VQ Analyzer screenshot of bitstream collected from Snapchat.

174. For example, x265 source code shows a variety of other parameters coded for each non-skipped block, such as a block with which a skipped block is merged.

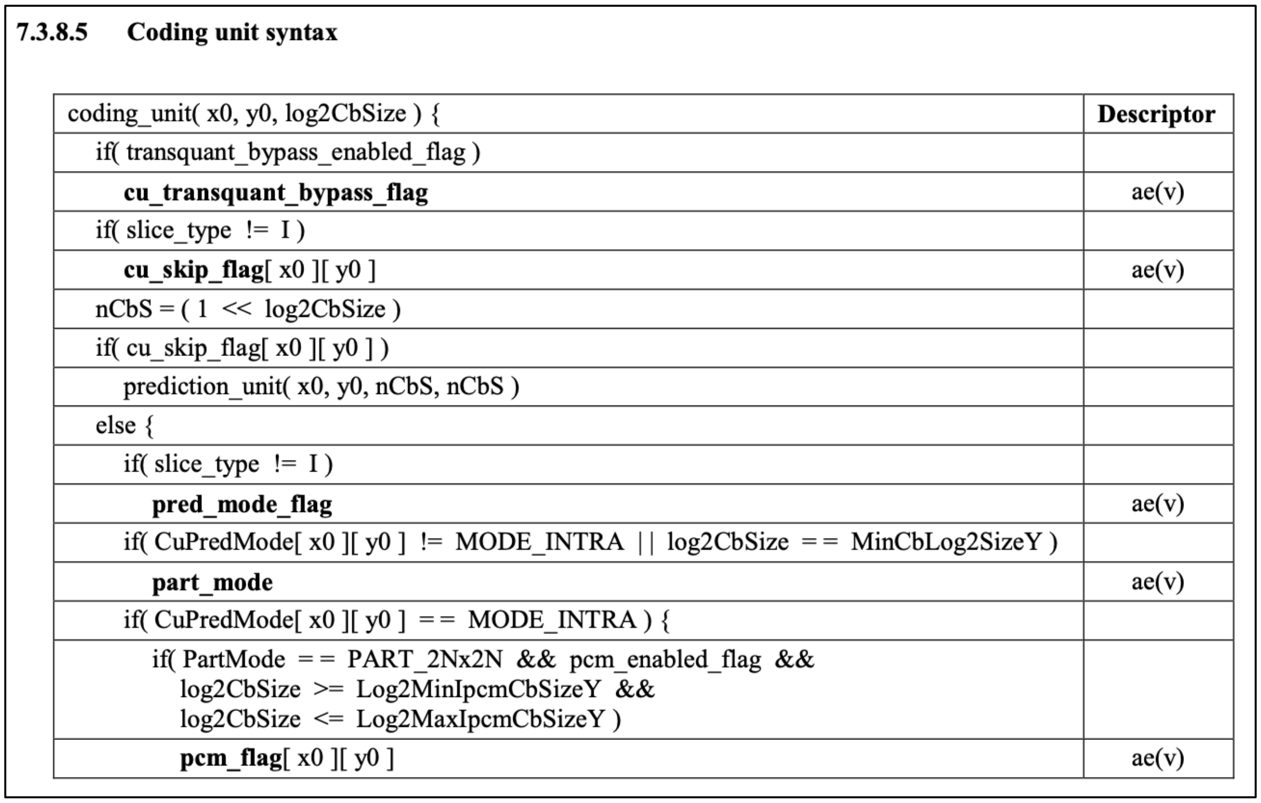


```
            codePredMode(ctu.m_predMode[absPartIdx]);
        }

        codePartSize(ctu, absPartIdx, depth);

        // prediction Info ( Intra : direction mode, Inter : Mv, reference idx )
        codePredInfo(ctu, absPartIdx);

        uint32_t tuDepthRange[2];
        if (ctu.isIntra(absPartIdx))
            ctu.getIntraTUQtDepthRange(tuDepthRange, absPartIdx);
        else
            ctu.getInterTUQtDepthRange(tuDepthRange, absPartIdx);

        // Encode Coefficients, allow codeCoeff() to modify bEncodeDQP
        codeCoeff(ctu, absPartIdx, bEncodeDQP, tuDepthRange);
```

Ex. 16 (entropy.cpp) at 15.

175. Snap's Accused Services perform "when the first information is in the second state, coding, into the data stream, third information associated with the coding block, the third information having first and second states, at least one coding parameter for the coding block when the third information is in the first state of the third information indicating that the coding block is to be reconstructed using the at least one coding parameter coded in the data stream, and a set of coding parameters associated with another merge candidate coding block and residual data for the coding block when the third information is in the second state of the third information, indicating that the coding block is to be reconstructed based on the set of coding parameters and the residual data coded in the data stream."

176. The H.265 standard shows that when cu_skip_flag is not set for a block, pred_mode_flag is encoded for the block. The H.265 standard discloses that pred_mode_flag being equal to 0 indicates Inter mode, and under this mode, residual data and other inter coding-related parameters are encoded into the data stream. Ex. 13 (H.265 specification) at Sec. 7.3.8.5, 7.3.8.8, 7.4.9.5.

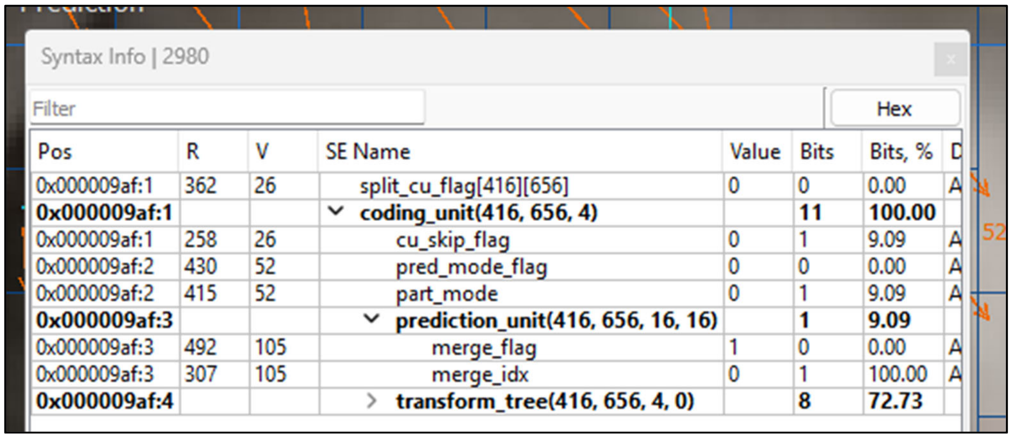


7.3.8.5 Coding unit syntax

coding_unit(x0, y0, log2CbSize) {	**Descriptor**
if(transquant_bypass_enabled_flag)	
cu_transquant_bypass_flag	ae(v)
if(slice_type != I)	
cu_skip_flag[x0][y0]	ae(v)
nCbS = (1 << log2CbSize)	
if(cu_skip_flag[x0][y0])	
prediction_unit(x0, y0, nCbS, nCbS)	
else {	
if(slice_type != I)	
pred_mode_flag	ae(v)
if(CuPredMode[x0][y0] != MODE_INTRA || log2CbSize = = MinCbLog2SizeY)	
part_mode	ae(v)
if(CuPredMode[x0][y0] = = MODE_INTRA) {	
if(PartMode = = PART_2Nx2N && pcm_enabled_flag && log2CbSize >= Log2MinIpcmCbSizeY && log2CbSize <= Log2MaxIpcmCbSizeY)	
pcm_flag[x0][y0]	ae(v)

Ex. 13 (H.265 specification) at 51; *see also id*. 52-53.

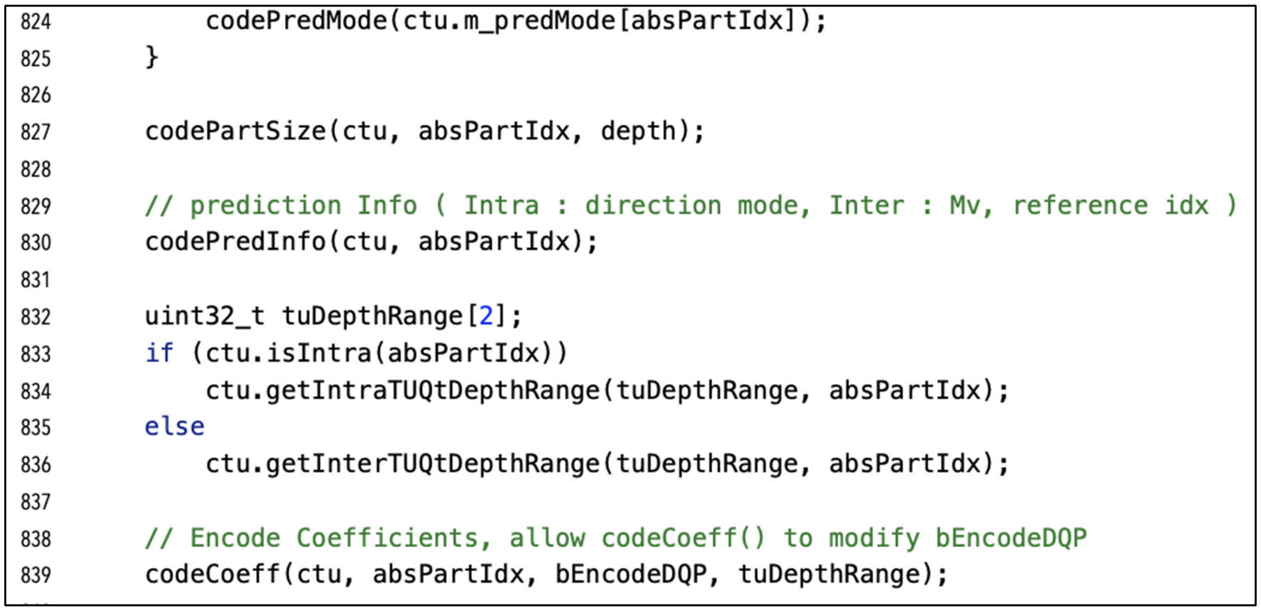
177. For example, compressed video data captured from Snap shows examples where cu_skip_flag is set to 0. In these cases, a pred_mode_flag, part_mode, and other data is encoded.



Pos	R	V	SE Name	Value	Bits	Bits, %	D
0x000009af:1	362	26	split_cu_flag[416][656]	0	0	0.00	A
0x000009af:1			**coding_unit(416, 656, 4)**		**11**	**100.00**	
0x000009af:1	258	26	cu_skip_flag	0	1	9.09	A
0x000009af:2	430	52	pred_mode_flag	0	0	0.00	A
0x000009af:2	415	52	part_mode	0	1	9.09	A
0x000009af:3			**prediction_unit(416, 656, 16, 16)**		**1**	**9.09**	
0x000009af:3	492	105	merge_flag	1	0	0.00	A
0x000009af:3	307	105	merge_idx	0	1	100.00	A
0x000009af:4			**transform_tree(416, 656, 4, 0)**		**8**	**72.73**	

VQ Analyzer screenshot of bitstream collected from Snapchat.

178. For example, x265 source code shows that the prediction mode is coded, as well as other parameters such as prediction info and residual coefficients.



```
            codePredMode(ctu.m_predMode[absPartIdx]);
        }

        codePartSize(ctu, absPartIdx, depth);

        // prediction Info ( Intra : direction mode, Inter : Mv, reference idx )
        codePredInfo(ctu, absPartIdx);

        uint32_t tuDepthRange[2];
        if (ctu.isIntra(absPartIdx))
            ctu.getIntraTUQtDepthRange(tuDepthRange, absPartIdx);
        else
            ctu.getInterTUQtDepthRange(tuDepthRange, absPartIdx);

        // Encode Coefficients, allow codeCoeff() to modify bEncodeDQP
        codeCoeff(ctu, absPartIdx, bEncodeDQP, tuDepthRange);
```

Ex. 16 (entropy.cpp) at 15.

COUNT III: PATENT INFRINGEMENT OF THE ’469 PATENT

179. Dolby incorporates by reference the preceding paragraphs as though fully set forth herein.

180. Snap had knowledge and notice of the ’469 Patent at least by November 24, 2023, and again through communications on at least August 13, 2025 and November 21, 2025.

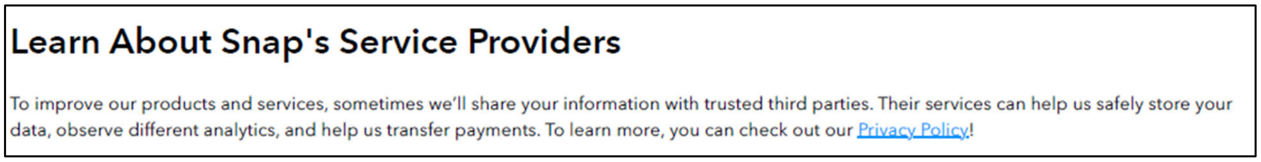
181. Snap infringes the ’469 Patent by making, using, offering to sell, and/or selling the patented invention within the United States, and/or importing the patented invention into the United States. Snap thus directly infringes the ’469 Patent literally and/or under the Doctrine of Equivalents, in violation of 35 U.S.C. § 271. For example, Snap directly infringes the ’469 Patent through its encoding, decoding, and transcoding of H.265-compliant (HEVC) video in the United States.

182. Upon information and belief, Snap derives revenue, directly and indirectly, from the activities relating to the Accused Services and to infringement of the ’469 Patent, including in this District and elsewhere in the United States.

183. On information and belief, Snap indirectly infringes claims of the ’469 Patent, as provided in 35 U.S.C. § 271(b), by knowingly inducing infringement by others, such as Snap’s partners and

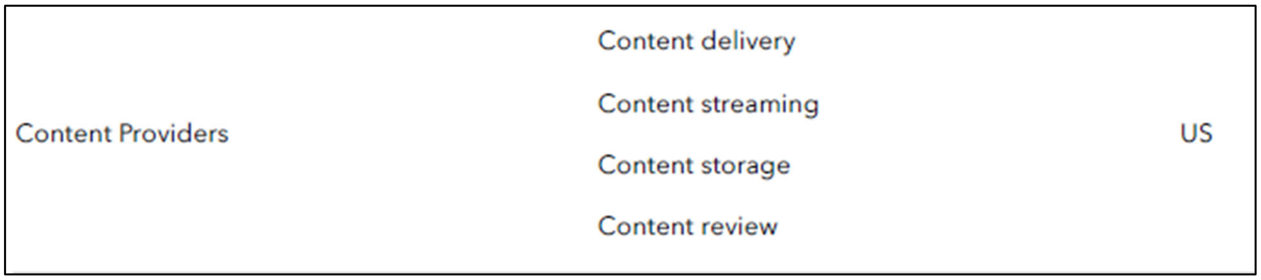
vendors, in this District and elsewhere in the United States.

184. For example, Snap uses the infrastructure of its providers to encode and decode video. Snap's web page indicates that these providers perform content delivery, streaming, storage, and review, including the encoding and decoding of video.



Learn About Snap's Service Providers

To improve our products and services, sometimes we'll share your information with trusted third parties. Their services can help us safely store your data, observe different analytics, and help us transfer payments. To learn more, you can check out our Privacy Policy!



Content Providers	Content delivery Content streaming Content storage Content review	US

Ex. 12 at 1.

185. As a result of Snap's inducement, Snap's providers perform infringing video coding and directly infringe the 469 Patent. Snap has performed and continues to perform these affirmative acts with knowledge of the '469 Patent and with the intent, or willful blindness, that the induced acts directly infringe the '469 Patent.

186. Snap's infringement of the '469 Patent is willful and deliberate. As detailed above, Snap had knowledge of the '469 Patent and had knowledge, or was willfully blind, as to its infringement of the '469 Patent. As discussed above, Snap has had actual knowledge of the '469 Patent since before this suit. Further, as discussed above, Snap knew or should have known that its actions infringe and actively induce infringement of the '469 Patent. As discussed above, Snap specifically intended that both itself and/or its customers and providers infringe the '469 Patent.

187. Snap's infringement of the '469 Patent has damaged and will continue to damage Dolby.

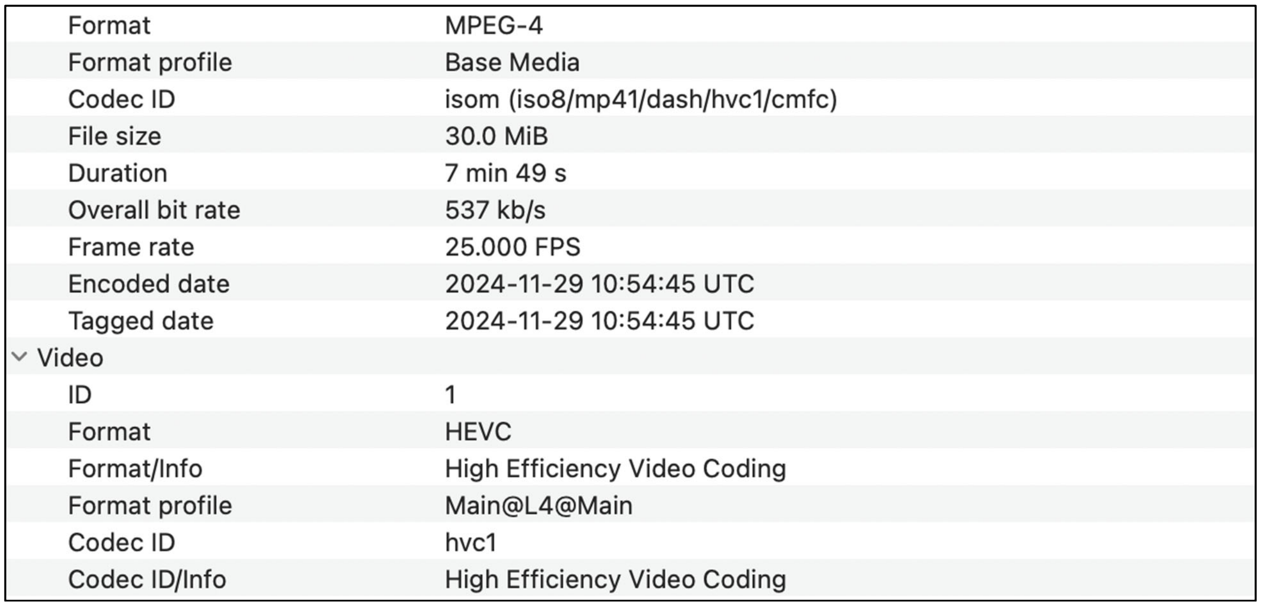
188. Exemplary claim 13 of the '469 Patent recites:

> 13. A method for encoding a sample array into an entropy encoded data stream, comprising
>
> entropy encoding a plurality of entropy slices into the entropy encoded data stream each entropy slice being associated with a different portion of the sample array, respectively, by
>
> performing, for each entropy slice, entropy encoding along a respective entropy coding path using respective probability estimations,
>
> adapting the respective probability estimations along the respective entropy coding path using a previously encoded part of the respective entropy slice,
>
> entropy encoding the plurality of entropy slices sequentially using an entropy slice order, and
>
> performing, in entropy encoding a predetermined entropy slice, entropy encoding a current part of the predetermined entropy slice based on the respective probability estimations of the predetermined entropy slice as adapted using the previously encoded part of the predetermined entropy slice, and probability estimations as used in the entropy encoding of a spatially neighboring, in the entropy slice order, preceding entropy slice at a neighboring part of the spatially neighboring preceding entropy slice,
>
> wherein each entropy slice comprises entropy encoded data for a corresponding portion of the sample array, the different portions forming rows of blocks of the sample array with the blocks being regularly arranged in rows and columns so that portions corresponding to the entropy slices comprise a same number of blocks, and the entropy coding path points in parallel along the rows of the blocks,
>
> the entropy encoding further comprising: initializing, for each respective entropy slice, the probability estimations before encoding a first block of the portion corresponding to the respective entropy slice along the respective coding path with probability estimations manifesting themselves after having entropy encoded a second block of the portion corresponding to, in the entropy slice order, the preceding entropy slice along the respective coding path.

189. Snap's Accused Services perform "a method for encoding a sample array into an entropy encoded data stream."

190. For example, Snap's backend servers encode H.265-compliant (HEVC) video for streaming to devices.

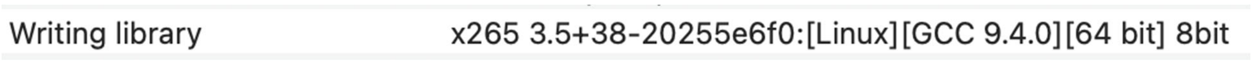
191. Videos downloaded from Snap show that an HEVC-compliant format is used:



Format	MPEG-4
Format profile	Base Media
Codec ID	isom (iso8/mp41/dash/hvc1/cmfc)
File size	30.0 MiB
Duration	7 min 49 s
Overall bit rate	537 kb/s
Frame rate	25.000 FPS
Encoded date	2024-11-29 10:54:45 UTC
Tagged date	2024-11-29 10:54:45 UTC
Video	
ID	1
Format	HEVC
Format/Info	High Efficiency Video Coding
Format profile	Main@L4@Main
Codec ID	hvc1
Codec ID/Info	High Efficiency Video Coding

(Video from Snap as viewed in MediaInfo app.)

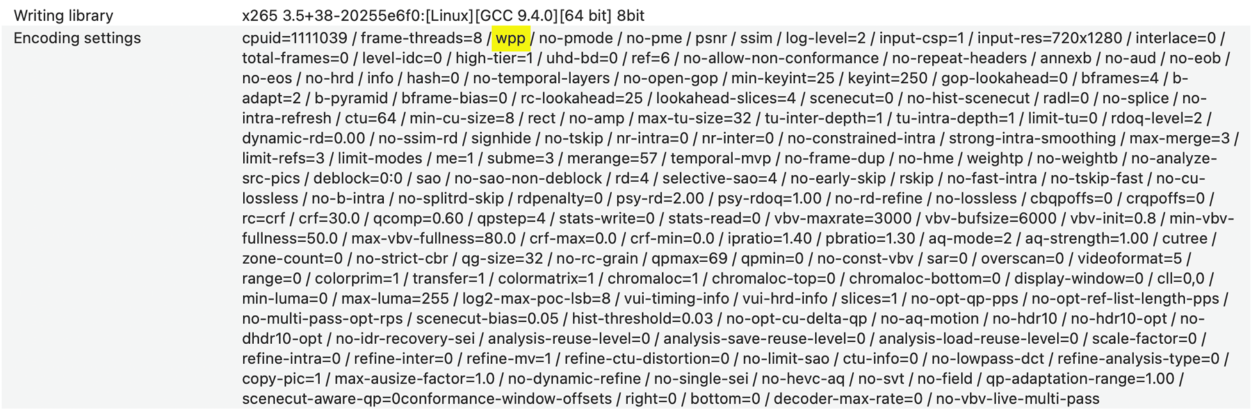
192. Snap uses the x265 library, version 3.5, revision 20255e6f0, to encode HEVC-compliant video.



Writing library	x265 3.5+38-20255e6f0:[Linux][GCC 9.4.0][64 bit] 8bit

(Video from Snap as viewed in MediaInfo app.)

193. The video from Snap was encoded with "wpp," or Wavefront Parallel Processing, enabled.

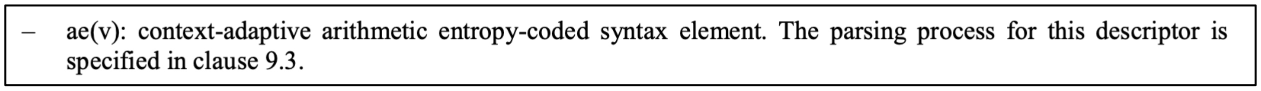


Writing library	x265 3.5+38-20255e6f0:[Linux][GCC 9.4.0][64 bit] 8bit
Encoding settings	cpuid=1111039 / frame-threads=8 / wpp / no-pmode / no-pme / psnr / ssim / log-level=2 / input-csp=1 / input-res=720x1280 / interlace=0 / total-frames=0 / level-idc=0 / high-tier=1 / uhd-bd=0 / ref=6 / no-allow-non-conformance / no-repeat-headers / annexb / no-aud / no-eob / no-eos / no-hrd / info / hash=0 / no-temporal-layers / no-open-gop / min-keyint=25 / keyint=250 / gop-lookahead=0 / bframes=4 / b-adapt=2 / b-pyramid / bframe-bias=0 / rc-lookahead=25 / lookahead-slices=4 / scenecut=0 / no-hist-scenecut / radl=0 / no-splice / no-intra-refresh / ctu=64 / min-cu-size=8 / rect / no-amp / max-tu-size=32 / tu-inter-depth=1 / tu-intra-depth=1 / limit-tu=0 / rdoq-level=2 / dynamic-rd=0.00 / no-ssim-rd / signhide / no-tskip / nr-intra=0 / nr-inter=0 / no-constrained-intra / strong-intra-smoothing / max-merge=3 / limit-refs=3 / limit-modes / me=1 / subme=3 / merange=57 / temporal-mvp / no-frame-dup / no-hme / weightp / no-weightb / no-analyze-src-pics / deblock=0:0 / sao / no-sao-non-deblock / rd=4 / selective-sao=4 / no-early-skip / rskip / no-fast-intra / no-tskip-fast / no-cu-lossless / no-b-intra / no-splitrd-skip / rdpenalty=0 / psy-rd=2.00 / psy-rdoq=1.00 / no-rd-refine / no-lossless / cbqpoffs=0 / crqpoffs=0 / rc=crf / crf=30.0 / qcomp=0.60 / qpstep=4 / stats-write=0 / stats-read=0 / vbv-maxrate=3000 / vbv-bufsize=6000 / vbv-init=0.8 / min-vbv-fullness=50.0 / max-vbv-fullness=80.0 / crf-max=0.0 / crf-min=0.0 / ipratio=1.40 / pbratio=1.30 / aq-mode=2 / aq-strength=1.00 / cutree / zone-count=0 / no-strict-cbr / qg-size=32 / no-rc-grain / qpmax=69 / qpmin=0 / no-const-vbv / sar=0 / overscan=0 / videoformat=5 / range=0 / colorprim=1 / transfer=1 / colormatrix=1 / chromaloc=1 / chromaloc-top=0 / chromaloc-bottom=0 / display-window=0 / cll=0,0 / min-luma=0 / max-luma=255 / log2-max-poc-lsb=8 / vui-timing-info / vui-hrd-info / slices=1 / no-opt-qp-pps / no-opt-ref-list-length-pps / no-multi-pass-opt-rps / scenecut-bias=0.05 / hist-threshold=0.03 / no-opt-cu-delta-qp / no-aq-motion / no-hdr10 / no-hdr10-opt / no-dhdr10-opt / no-idr-recovery-sei / analysis-reuse-level=0 / analysis-save-reuse-level=0 / analysis-load-reuse-level=0 / scale-factor=0 / refine-intra=0 / refine-inter=0 / refine-mv=1 / refine-ctu-distortion=0 / no-limit-sao / ctu-info=0 / no-lowpass-dct / refine-analysis-type=0 / copy-pic=1 / max-ausize-factor=1.0 / no-dynamic-refine / no-single-sei / no-hevc-aq / no-svt / no-field / qp-adaptation-range=1.00 / scenecut-aware-qp=0conformance-window-offsets / right=0 / bottom=0 / decoder-max-rate=0 / no-vbv-live-multi-pass

(Video from Snap as viewed in MediaInfo app.)

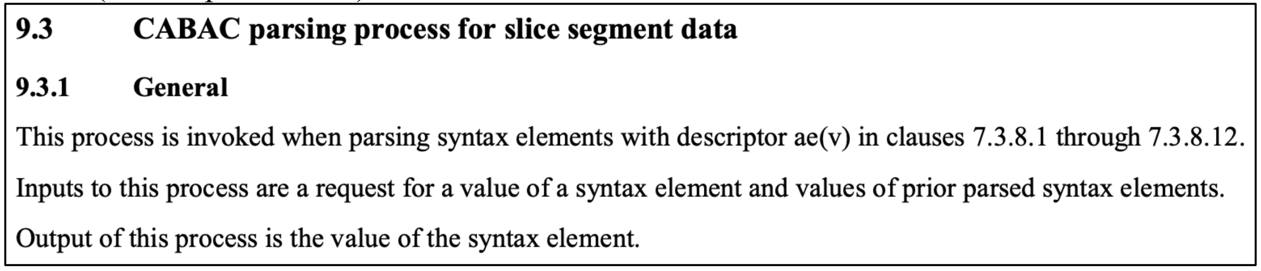
194. Snap's Accused Services perform "entropy encoding a plurality of entropy slices into the entropy encoded data stream each entropy slice being associated with a different portion of the sample array, respectively."

195. For example, the H.265 specification explains that entropy coding techniques such as CABAC are used for the data for each slice.



– ae(v): context-adaptive arithmetic entropy-coded syntax element. The parsing process for this descriptor is specified in clause 9.3.

Ex. 13 (H.265 specification) at 32.



9.3 CABAC parsing process for slice segment data

9.3.1 General

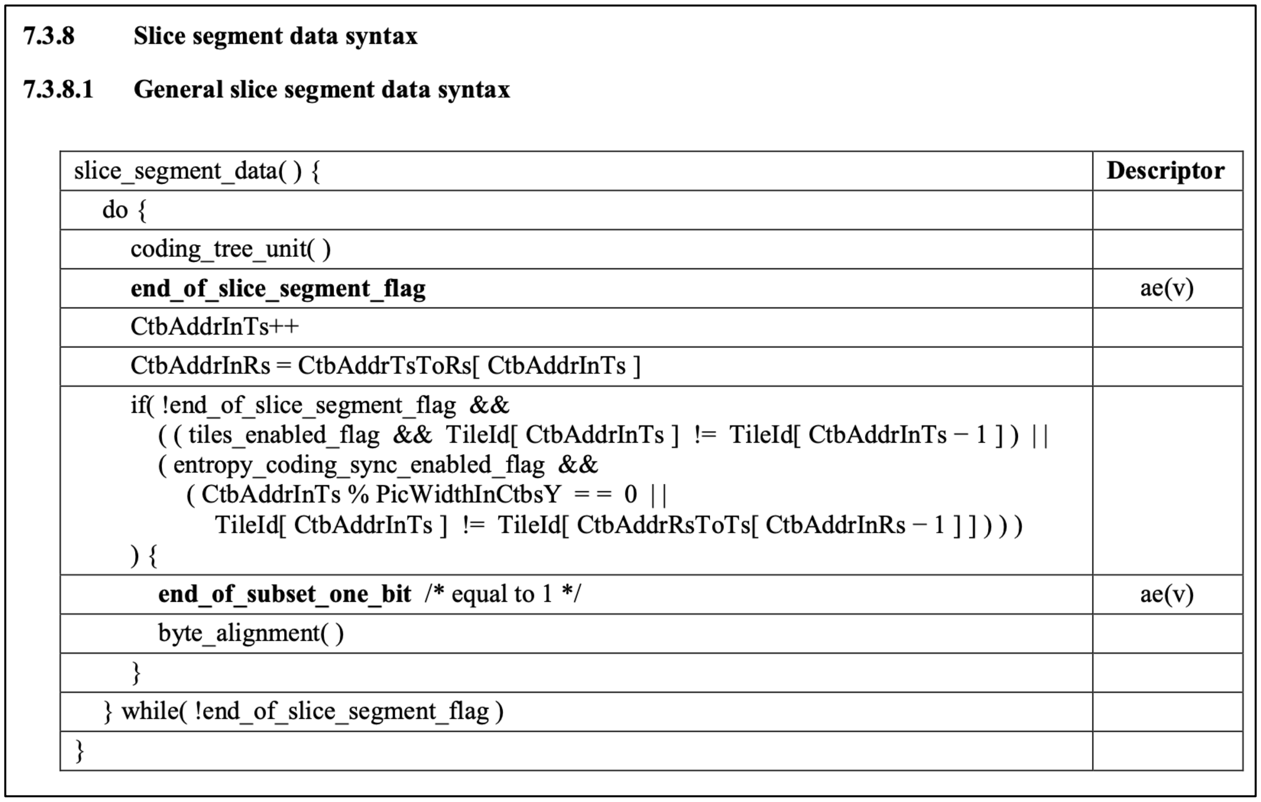
This process is invoked when parsing syntax elements with descriptor ae(v) in clauses 7.3.8.1 through 7.3.8.12.

Inputs to this process are a request for a value of a syntax element and values of prior parsed syntax elements.

Output of this process is the value of the syntax element.

Ex. 13 (H.265 specification) at 184.

196. Further, when WPP is enabled (using entropy_coding_sync_enabled_flag), one slice segment subset of a slice segment may be associated with one picture portion, i.e., one CTU row of the picture. Each slice segment subset is an individual byte aligned CABAC encoded bitstream, which is terminated using the syntax element end_of_subset_one_bit.



7.3.8 Slice segment data syntax

7.3.8.1 General slice segment data syntax

slice_segment_data( ) {	**Descriptor**
do {	
coding_tree_unit( )	
end_of_slice_segment_flag	ae(v)
CtbAddrInTs++	
CtbAddrInRs = CtbAddrTsToRs[CtbAddrInTs]	
if(!end_of_slice_segment_flag && ((tiles_enabled_flag && TileId[CtbAddrInTs] != TileId[CtbAddrInTs − 1]) || (entropy_coding_sync_enabled_flag && (CtbAddrInTs % PicWidthInCtbsY = = 0 || TileId[CtbAddrInTs] != TileId[CtbAddrRsToTs[CtbAddrInRs − 1]])))) {	
end_of_subset_one_bit /* equal to 1 */	ae(v)
byte_alignment( )	
}	
} while(!end_of_slice_segment_flag)	
}	

Ex. 13 (H.265 specification) at 49.

entropy_coding_sync_enabled_flag equal to 1 specifies that a specific synchronization process for context variables is invoked before decoding the coding tree unit which includes the first coding tree block of a row of coding tree blocks in each tile in each picture referring to the PPS, and a specific storage process for context variables is invoked after decoding the coding tree unit which includes the second coding tree block of a row of coding tree blocks in each tile in each picture referring to the PPS. entropy_coding_sync_enabled_flag equal to 0 specifies that no specific synchronization process for context variables is required to be invoked before decoding the coding tree unit which includes the first coding tree block of a row of coding tree blocks in each tile in each picture referring to the PPS, and no specific storage process for context variables is required to be invoked after decoding the coding tree unit which includes the second coding tree block of a row of coding tree blocks in each tile in each picture referring to the PPS.

It is a requirement of bitstream conformance that the value of entropy_coding_sync_enabled_flag shall be the same for all PPSs that are activated within a CVS.

When entropy_coding_sync_enabled_flag is equal to 1 and the first coding tree block in a slice is not the first coding tree block of a row of coding tree blocks in a tile, it is a requirement of bitstream conformance that the last coding tree block in the slice shall belong to the same row of coding tree blocks as the first coding tree block in the slice.

When entropy_coding_sync_enabled_flag is equal to 1 and the first coding tree block in a slice segment is not the first coding tree block of a row of coding tree blocks in a tile, it is a requirement of bitstream conformance that the last coding tree block in the slice segment shall belong to the same row of coding tree blocks as the first coding tree block in the slice segment.

Ex. 13 (H.265 specification) at 79-80.

7.4.9 Slice segment data semantics

7.4.9.1 General slice segment data semantics

end_of_slice_segment_flag equal to 0 specifies that another coding tree unit is following in the slice. end_of_slice_segment_flag equal to 1 specifies the end of the slice segment, i.e., that no further coding tree unit follows in the slice segment.

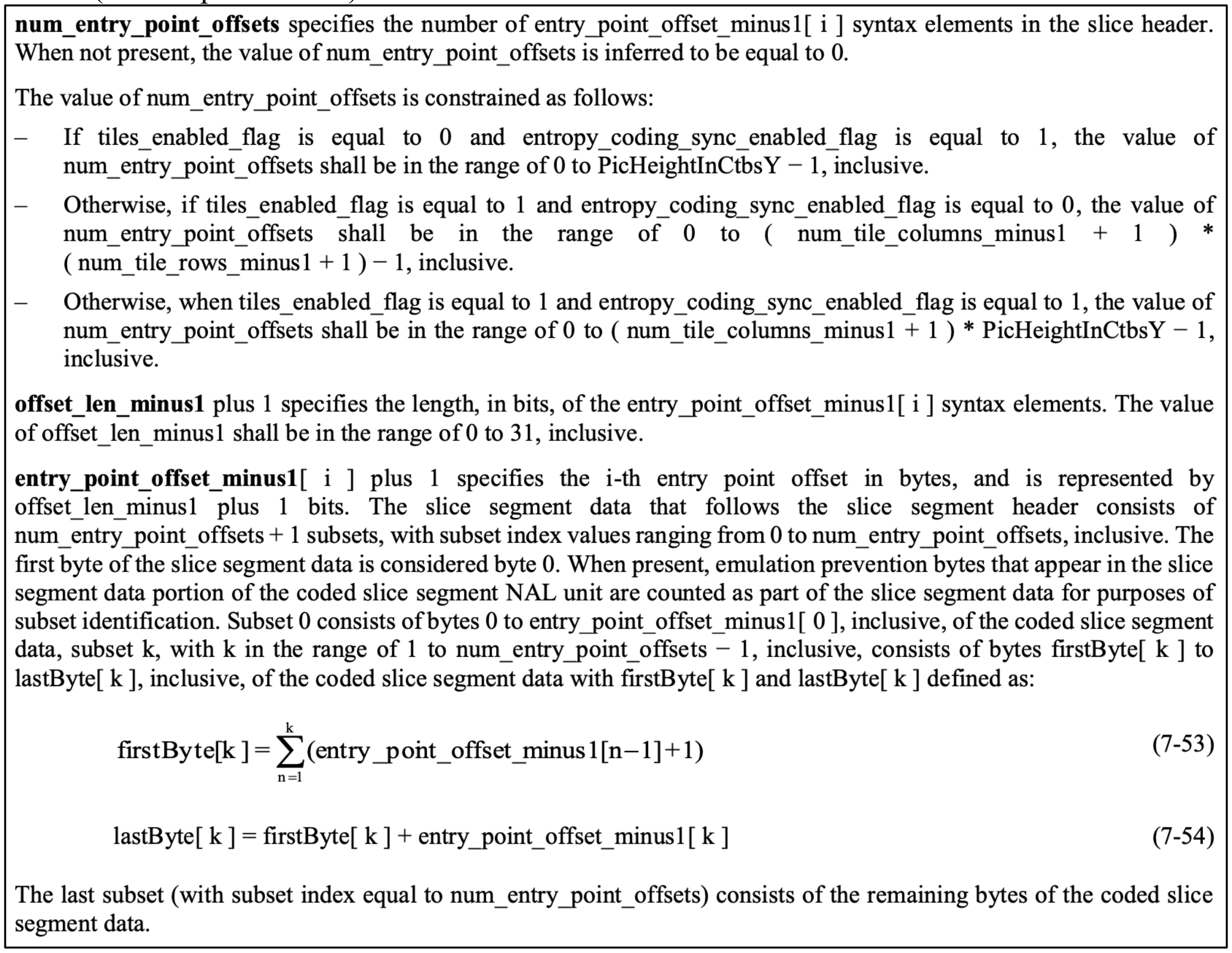
end_of_subset_one_bit shall be equal to 1.

Ex. 13 (H.265 specification) at 98.

197. In the bitstream, for all but the first slice segment subset, the start position for each subset (and therefore, the start position of each CTU row of the picture) is indicated by entry point offsets that are signaled by syntax elements entry_point_offset_minus1 in the encoded bitstream. The total number of such entry point offsets for a picture is specified by the syntax element num_entry_point_offsets, which has the maximum value equal to the total number of rows of the picture (provided in terms of the variable PicHeightInCtbsY in H.265).

if(tiles_enabled_flag || entropy_coding_sync_enabled_flag) {	
num_entry_point_offsets	ue(v)
if(num_entry_point_offsets > 0) {	
offset_len_minus1	ue(v)
for(i = 0; i < num_entry_point_offsets; i++)	
entry_point_offset_minus1[i]	u(v)
}	

Ex. 13 (H.265 specification) at 46.



num_entry_point_offsets specifies the number of entry_point_offset_minus1[i] syntax elements in the slice header. When not present, the value of num_entry_point_offsets is inferred to be equal to 0.

The value of num_entry_point_offsets is constrained as follows:

– If tiles_enabled_flag is equal to 0 and entropy_coding_sync_enabled_flag is equal to 1, the value of num_entry_point_offsets shall be in the range of 0 to PicHeightInCtbsY − 1, inclusive.

– Otherwise, if tiles_enabled_flag is equal to 1 and entropy_coding_sync_enabled_flag is equal to 0, the value of num_entry_point_offsets shall be in the range of 0 to (num_tile_columns_minus1 + 1) * (num_tile_rows_minus1 + 1) − 1, inclusive.

– Otherwise, when tiles_enabled_flag is equal to 1 and entropy_coding_sync_enabled_flag is equal to 1, the value of num_entry_point_offsets shall be in the range of 0 to (num_tile_columns_minus1 + 1) * PicHeightInCtbsY − 1, inclusive.

offset_len_minus1 plus 1 specifies the length, in bits, of the entry_point_offset_minus1[i] syntax elements. The value of offset_len_minus1 shall be in the range of 0 to 31, inclusive.

entry_point_offset_minus1[i] plus 1 specifies the i-th entry point offset in bytes, and is represented by offset_len_minus1 plus 1 bits. The slice segment data that follows the slice segment header consists of num_entry_point_offsets + 1 subsets, with subset index values ranging from 0 to num_entry_point_offsets, inclusive. The first byte of the slice segment data is considered byte 0. When present, emulation prevention bytes that appear in the slice segment data portion of the coded slice segment NAL unit are counted as part of the slice segment data for purposes of subset identification. Subset 0 consists of bytes 0 to entry_point_offset_minus1[0], inclusive, of the coded slice segment data, subset k, with k in the range of 1 to num_entry_point_offsets − 1, inclusive, consists of bytes firstByte[k] to lastByte[k], inclusive, of the coded slice segment data with firstByte[k] and lastByte[k] defined as:

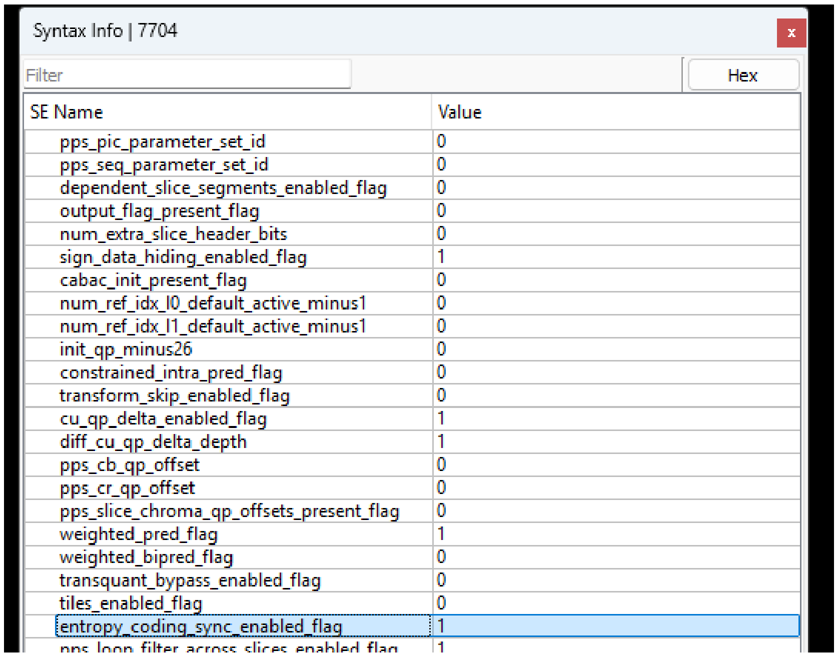
$$\text{firstByte}[k] = \sum_{n=1}^{k} (\text{entry\_point\_offset\_minus1}[n-1]+1) \tag{7-53}$$

$$\text{lastByte}[\,k\,] = \text{firstByte}[\,k\,] + \text{entry\_point\_offset\_minus1}[\,k\,] \tag{7-54}$$

The last subset (with subset index equal to num_entry_point_offsets) consists of the remaining bytes of the coded slice segment data.

Ex. 13 (H.265 specification) at 93.

198. Review of bitstreams collected from Snap indicates that "entropy_coding_sync_enabled_flag" is enabled, that various entry point offsets are encoded, and that arithmetic coding is used throughout Snap bitstreams to encode information about coding units and slices.

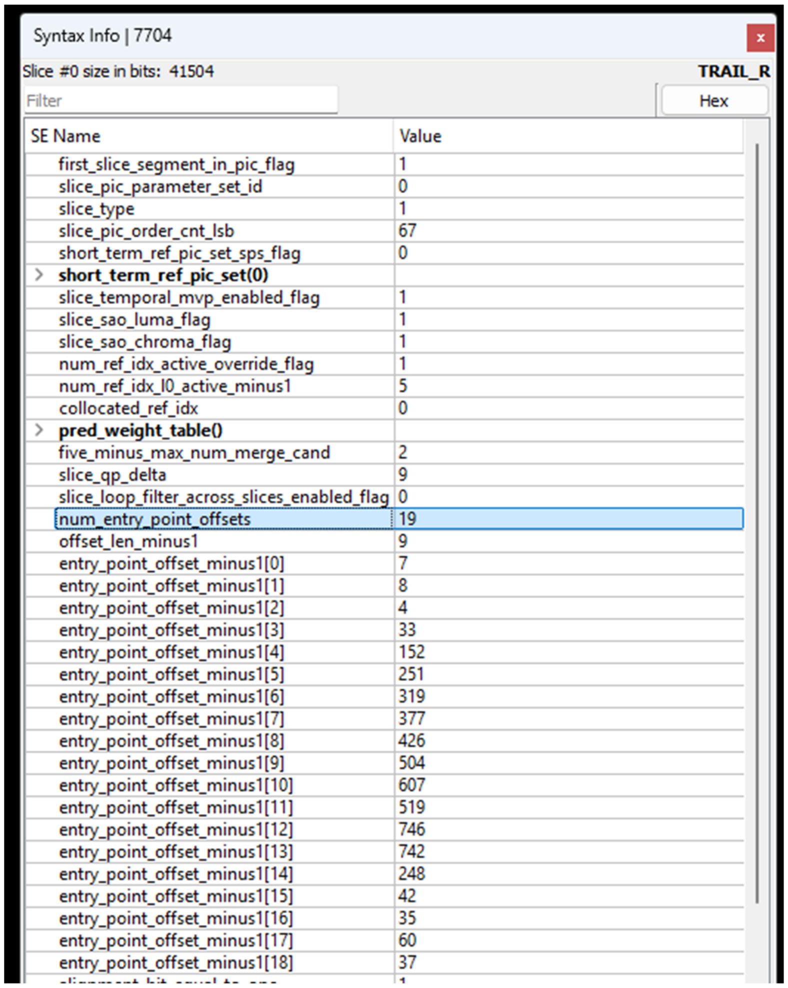


Syntax Info | 7704

Filter

Hex

SE Name	Value
pps_pic_parameter_set_id	0
pps_seq_parameter_set_id	0
dependent_slice_segments_enabled_flag	0
output_flag_present_flag	0
num_extra_slice_header_bits	0
sign_data_hiding_enabled_flag	1
cabac_init_present_flag	0
num_ref_idx_l0_default_active_minus1	0
num_ref_idx_l1_default_active_minus1	0
init_qp_minus26	0
constrained_intra_pred_flag	0
transform_skip_enabled_flag	0
cu_qp_delta_enabled_flag	1
diff_cu_qp_delta_depth	1
pps_cb_qp_offset	0
pps_cr_qp_offset	0
pps_slice_chroma_qp_offsets_present_flag	0
weighted_pred_flag	1
weighted_bipred_flag	0
transquant_bypass_enabled_flag	0
tiles_enabled_flag	0
entropy_coding_sync_enabled_flag	1
pps_loop_filter_across_slices_enabled_flag	1



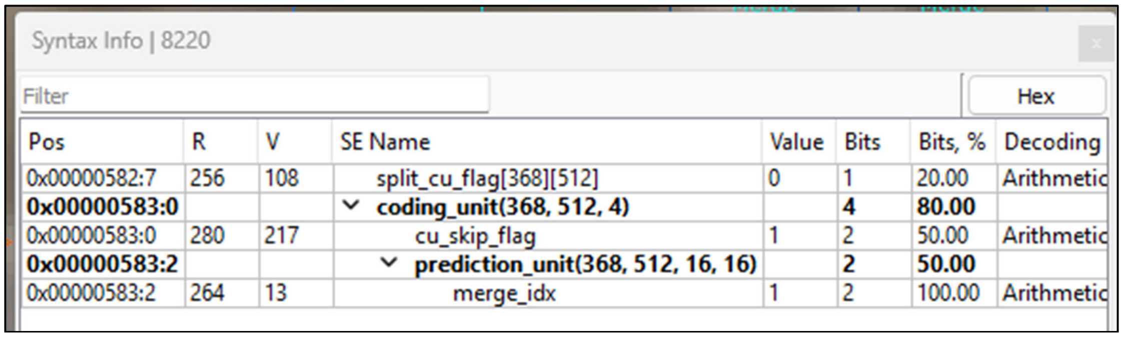
Syntax Info | 7704

Slice #0 size in bits: 41504 TRAIL_R

Filter

Hex

SE Name	Value
first_slice_segment_in_pic_flag	1
slice_pic_parameter_set_id	0
slice_type	1
slice_pic_order_cnt_lsb	67
short_term_ref_pic_set_sps_flag	0
short_term_ref_pic_set(0)	
slice_temporal_mvp_enabled_flag	1
slice_sao_luma_flag	1
slice_sao_chroma_flag	1
num_ref_idx_active_override_flag	1
num_ref_idx_l0_active_minus1	5
collocated_ref_idx	0
pred_weight_table()	
five_minus_max_num_merge_cand	2
slice_qp_delta	9
slice_loop_filter_across_slices_enabled_flag	0
num_entry_point_offsets	19
offset_len_minus1	9
entry_point_offset_minus1[0]	7
entry_point_offset_minus1[1]	8
entry_point_offset_minus1[2]	4
entry_point_offset_minus1[3]	33
entry_point_offset_minus1[4]	152
entry_point_offset_minus1[5]	251
entry_point_offset_minus1[6]	319
entry_point_offset_minus1[7]	377
entry_point_offset_minus1[8]	426
entry_point_offset_minus1[9]	504
entry_point_offset_minus1[10]	607
entry_point_offset_minus1[11]	519
entry_point_offset_minus1[12]	746
entry_point_offset_minus1[13]	742
entry_point_offset_minus1[14]	248
entry_point_offset_minus1[15]	42
entry_point_offset_minus1[16]	35
entry_point_offset_minus1[17]	60
entry_point_offset_minus1[18]	37
alignment_bit_equal_to_one	1



Syntax Info | 8220

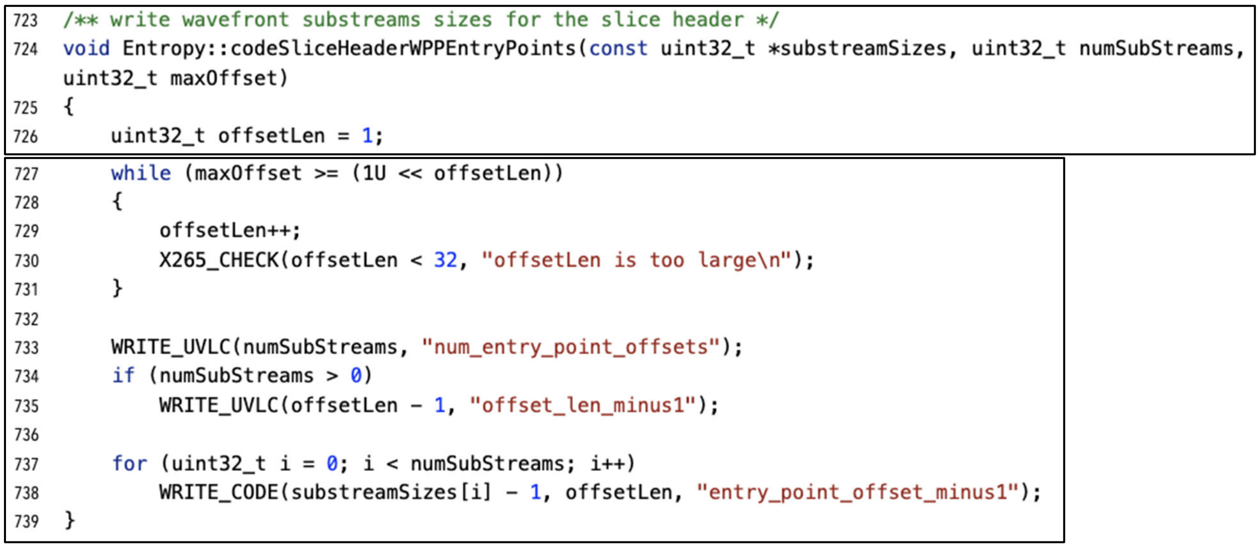
Filter | Hex

Pos	R	V	SE Name	Value	Bits	Bits, %	Decoding
0x00000582:7	256	108	split_cu_flag[368][512]	0	1	20.00	Arithmetic
0x00000583:0			**coding_unit(368, 512, 4)**		**4**	**80.00**	
0x00000583:0	280	217	cu_skip_flag	1	2	50.00	Arithmetic
0x00000583:2			**prediction_unit(368, 512, 16, 16)**		**2**	**50.00**	
0x00000583:2	264	13	merge_idx	1	2	100.00	Arithmetic

Exemplary VQ Analyzer screenshot showing "Arithmetic" decoding in Snap bitstream.

199. Source code for the x265 encoder used by Snap shows the use of entropy coding. For example, the "Entropy" class in entropy.cpp performs entropy coding as part of the coding process. Ex. 16 (entropy.cpp).

200. Source code shows that when WPP is enabled, entry point offsets for each entropy segment are encoded.



```
/** write wavefront substreams sizes for the slice header */
void Entropy::codeSliceHeaderWPPEntryPoints(const uint32_t *substreamSizes, uint32_t numSubStreams,
uint32_t maxOffset)
{
    uint32_t offsetLen = 1;
    while (maxOffset >= (1U << offsetLen))
    {
        offsetLen++;
        X265_CHECK(offsetLen < 32, "offsetLen is too large\n");
    }

    WRITE_UVLC(numSubStreams, "num_entry_point_offsets");
    if (numSubStreams > 0)
        WRITE_UVLC(offsetLen - 1, "offset_len_minus1");

    for (uint32_t i = 0; i < numSubStreams; i++)
        WRITE_CODE(substreamSizes[i] - 1, offsetLen, "entry_point_offset_minus1");
}
```

Ex. 16 (entropy.cpp) at 13-14.

201. Snap's Accused Services perform "performing, for each entropy slice, entropy encoding along a respective entropy coding path using respective probability estimations."

202. For example, the H.265 specification explains that CTUs in a slice segment or a tile are scanned or parsed in the raster scan order. Based on the definition of raster scan in H.265, each row

of the picture is scanned from left to right edge of the picture, going down the rows from top to bottom of the picture.

3.118 **raster scan**: A mapping of a rectangular two-dimensional pattern to a one-dimensional pattern such that the first entries in the one-dimensional pattern are from the first top row of the two-dimensional pattern scanned from left to right, followed similarly by the second, third, etc., rows of the pattern (going down) each scanned from left to right.

Ex. 13 (H.265 specification) at 10.

slice_segment_address specifies the address of the first coding tree block in the slice segment, in coding tree block raster scan of a picture. The length of the slice_segment_address syntax element is Ceil(Log2(PicSizeInCtbsY)) bits. The value of slice_segment_address shall be in the range of 0 to PicSizeInCtbsY − 1, inclusive and the value of slice_segment_address shall not be equal to the value of slice_segment_address of any other coded slice segment NAL unit of the same coded picture. When slice_segment_address is not present, it is inferred to be equal to 0.

The variable CtbAddrInRs, specifying a coding tree block address in coding tree block raster scan of a picture, is set equal to slice_segment_address. The variable CtbAddrInTs, specifying a coding tree block address in tile scan, is set equal to CtbAddrRsToTs[CtbAddrInRs]. The variable CuQpDeltaVal, specifying the difference between a luma quantization

Ex. 13 (H.265 specification) at 89.

entry_point_offset_minus1[i] plus 1 specifies the i-th entry point offset in bytes, and is represented by offset_len_minus1 plus 1 bits. The slice segment data that follows the slice segment header consists of num_entry_point_offsets + 1 subsets, with subset index values ranging from 0 to num_entry_point_offsets, inclusive. The first byte of the slice segment data is considered byte 0. When present, emulation prevention bytes that appear in the slice segment data portion of the coded slice segment NAL unit are counted as part of the slice segment data for purposes of subset identification. Subset 0 consists of bytes 0 to entry_point_offset_minus1[0], inclusive, of the coded slice segment data, subset k, with k in the range of 1 to num_entry_point_offsets − 1, inclusive, consists of bytes firstByte[k] to lastByte[k], inclusive, of the coded slice segment data with firstByte[k] and lastByte[k] defined as:

Ex. 13 (H.265 specification) at 93.

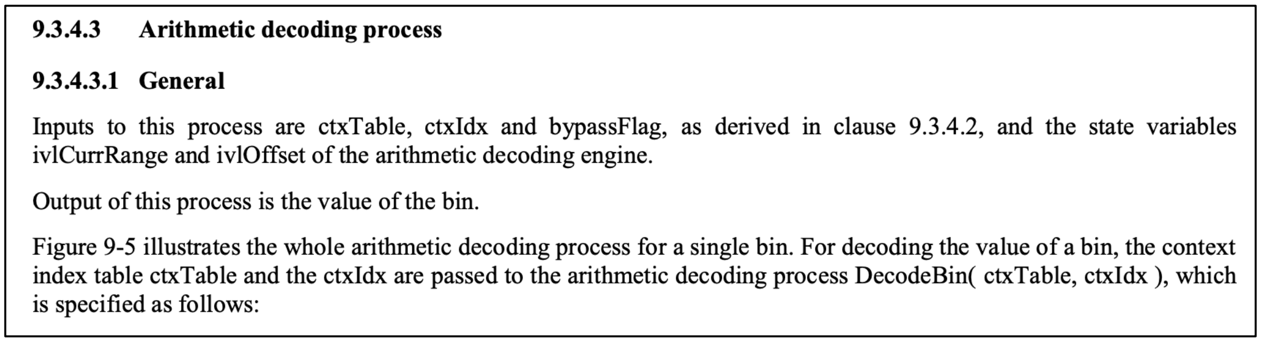
When tiles_enabled_flag is equal to 0 and entropy_coding_sync_enabled_flag is equal to 1, each subset k with k in the range of 0 to num_entry_point_offsets, inclusive, shall consist of all coded bits of all coding tree units in the slice segment that include luma coding tree blocks that are in the same luma coding tree block row of the picture, and the number of

subsets (i.e., the value of num_entry_point_offsets + 1) shall be equal to the number of coding tree block rows of the picture that contain coding tree units that are in the coded slice segment.

NOTE 6 – The last subset (i.e., subset k for k equal to num_entry_point_offsets) may or may not contain all coding tree units that include luma coding tree blocks that are in a luma coding tree block row of the picture.

When tiles_enabled_flag is equal to 1 and entropy_coding_sync_enabled_flag is equal to 1, each subset k with k in the range of 0 to num_entry_point_offsets, inclusive, shall consist of all coded bits of all coding tree units in the slice segment that include luma coding tree blocks that are in the same luma coding tree block row of a tile, and the number of subsets (i.e., the value of num_entry_point_offsets + 1) shall be equal to the number of luma coding tree block rows of a tile that contain coding tree units that are in the coded slice segment.

Ex. 13 (H.265 specification) at 93-94.



9.3.4.3 Arithmetic decoding process

9.3.4.3.1 General

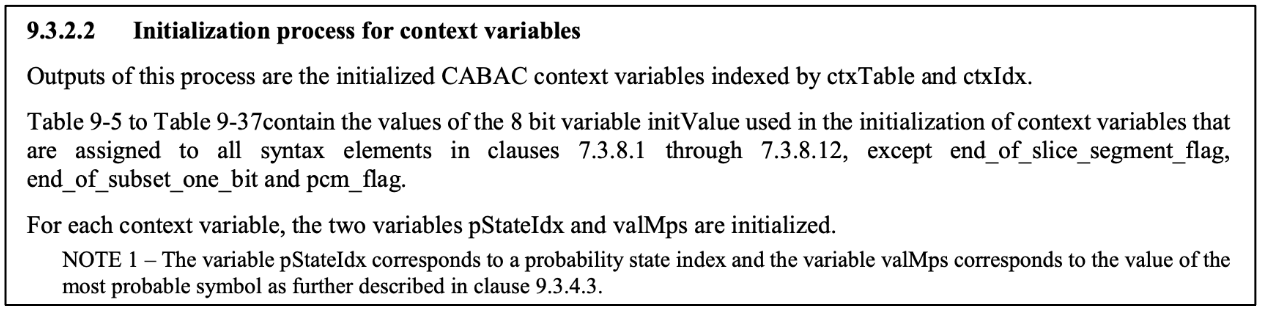
Inputs to this process are ctxTable, ctxIdx and bypassFlag, as derived in clause 9.3.4.2, and the state variables ivlCurrRange and ivlOffset of the arithmetic decoding engine.

Output of this process is the value of the bin.

Figure 9-5 illustrates the whole arithmetic decoding process for a single bin. For decoding the value of a bin, the context index table ctxTable and the ctxIdx are passed to the arithmetic decoding process DecodeBin(ctxTable, ctxIdx), which is specified as follows:

Ex. 13 (H.265 specification) at 212.

203. The H.265 specification further discloses probability estimations of least probable symbol (LPS) and most probable symbol (MPS), represented for example using variables pStateIdx, valMps, ivlCurrRange, etc.



9.3.2.2 Initialization process for context variables

Outputs of this process are the initialized CABAC context variables indexed by ctxTable and ctxIdx.

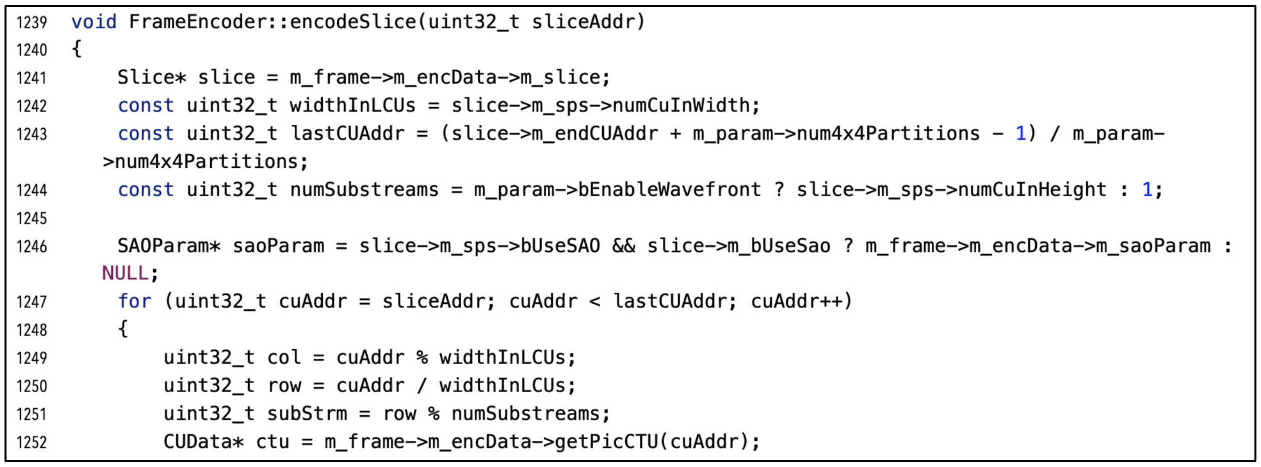
Table 9-5 to Table 9-37contain the values of the 8 bit variable initValue used in the initialization of context variables that are assigned to all syntax elements in clauses 7.3.8.1 through 7.3.8.12, except end_of_slice_segment_flag, end_of_subset_one_bit and pcm_flag.

For each context variable, the two variables pStateIdx and valMps are initialized.

NOTE 1 – The variable pStateIdx corresponds to a probability state index and the variable valMps corresponds to the value of the most probable symbol as further described in clause 9.3.4.3.

Ex. 13 (H.265 specification) at 187.

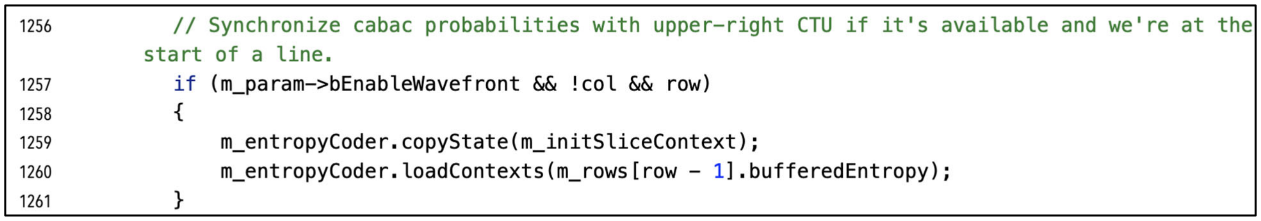
204. Source code for the x265 encoder used by Snap shows the use of entropy coding. For example, the "Entropy" class in entropy.cpp performs entropy coding along a respective entropy coding path using respective probability estimations as part of the coding process. Ex. 16 (entropy.cpp). This code shows that the coding proceeds column by column and row by row:



```
void FrameEncoder::encodeSlice(uint32_t sliceAddr)
{
    Slice* slice = m_frame->m_encData->m_slice;
    const uint32_t widthInLCUs = slice->m_sps->numCuInWidth;
    const uint32_t lastCUAddr = (slice->m_endCUAddr + m_param->num4x4Partitions - 1) / m_param->num4x4Partitions;
    const uint32_t numSubstreams = m_param->bEnableWavefront ? slice->m_sps->numCuInHeight : 1;

    SAOParam* saoParam = slice->m_sps->bUseSAO && slice->m_bUseSao ? m_frame->m_encData->m_saoParam : NULL;
    for (uint32_t cuAddr = sliceAddr; cuAddr < lastCUAddr; cuAddr++)
    {
        uint32_t col = cuAddr % widthInLCUs;
        uint32_t row = cuAddr / widthInLCUs;
        uint32_t subStrm = row % numSubstreams;
        CUData* ctu = m_frame->m_encData->getPicCTU(cuAddr);
```

Ex. 22 (frameencoder.cpp) at 23.

205. The x265 source code used by Snap shows that when WPP is enabled, probabilities are synchronized at the start of each line.

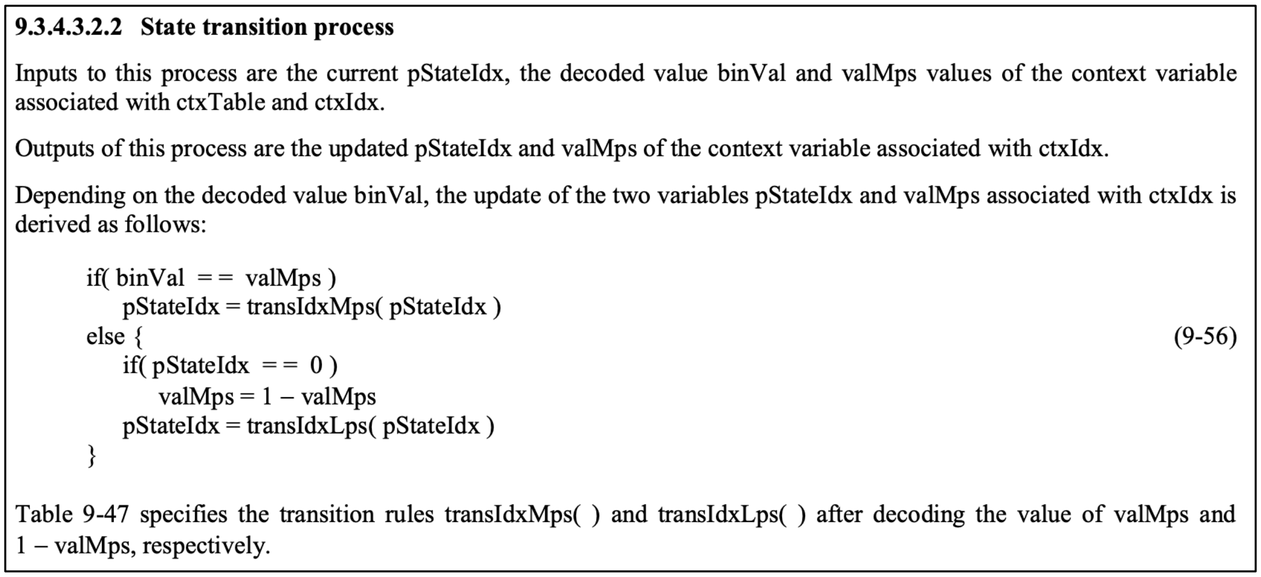


```
        // Synchronize cabac probabilities with upper-right CTU if it's available and we're at the start of a line.
        if (m_param->bEnableWavefront && !col && row)
        {
            m_entropyCoder.copyState(m_initSliceContext);
            m_entropyCoder.loadContexts(m_rows[row - 1].bufferedEntropy);
        }
```

Ex. 22 (frameencoder.cpp) at 24.

206. Snap's Accused Services perform "adapting the respective probability estimations along the respective entropy coding path using a previously encoded part of the respective entropy slice."

207. The H.265 specification discloses CABAC-based entropy coding, which adapts probability estimations based on context throughout the coding.



9.3.4.3.2.2 State transition process

Inputs to this process are the current pStateIdx, the decoded value binVal and valMps values of the context variable associated with ctxTable and ctxIdx.

Outputs of this process are the updated pStateIdx and valMps of the context variable associated with ctxIdx.

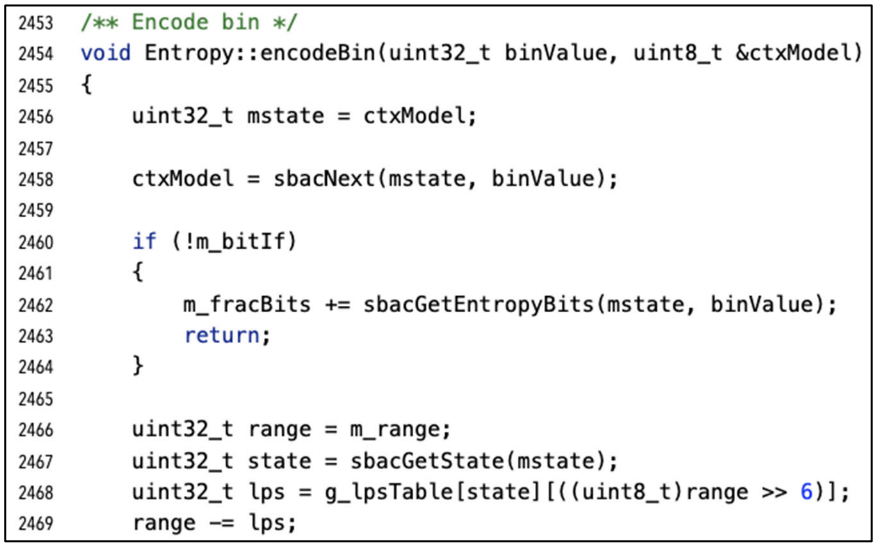
Depending on the decoded value binVal, the update of the two variables pStateIdx and valMps associated with ctxIdx is derived as follows:

```
if( binVal  = =  valMps )
    pStateIdx = transIdxMps( pStateIdx )
else {                                                    (9-56)
    if( pStateIdx  = =  0 )
        valMps = 1 – valMps
    pStateIdx = transIdxLps( pStateIdx )
}
```

Table 9-47 specifies the transition rules transIdxMps( ) and transIdxLps( ) after decoding the value of valMps and 1 – valMps, respectively.

Ex. 13 (H.265 specification) at 214.

208. The x265 source code used by Snap shows that probability estimations are adapted as coding progresses. For example, the Entropy::encodeBin function takes a context model ctxModel and updates it as coding continues.



```
/** Encode bin */
void Entropy::encodeBin(uint32_t binValue, uint8_t &ctxModel)
{
    uint32_t mstate = ctxModel;

    ctxModel = sbacNext(mstate, binValue);

    if (!m_bitIf)
    {
        m_fracBits += sbacGetEntropyBits(mstate, binValue);
        return;
    }

    uint32_t range = m_range;
    uint32_t state = sbacGetState(mstate);
    uint32_t lps = g_lpsTable[state][((uint8_t)range >> 6)];
    range -= lps;
```

Ex. 16 (entropy.cpp) at 45.

209. Snap's Accused Services perform "entropy encoding the plurality of entropy slices sequentially using an entropy slice order."

210. In the H.265 standard, CTUs in a slice segment are scanned or parsed in the raster scan order. Each row of the picture is scanned from left to right edge of the picture, going down the rows from top to bottom of the picture.

> **3.118** **raster scan**: A mapping of a rectangular two-dimensional pattern to a one-dimensional pattern such that the first entries in the one-dimensional pattern are from the first top row of the two-dimensional pattern scanned from left to right, followed similarly by the second, third, etc., rows of the pattern (going down) each scanned from left to right.

Ex. 13 (H.265 specification) at 10.

> **slice_segment_address** specifies the address of the first coding tree block in the slice segment, in coding tree block raster scan of a picture. The length of the slice_segment_address syntax element is Ceil(Log2(PicSizeInCtbsY)) bits. The value of slice_segment_address shall be in the range of 0 to PicSizeInCtbsY − 1, inclusive and the value of slice_segment_address shall not be equal to the value of slice_segment_address of any other coded slice segment NAL unit of the same coded picture. When slice_segment_address is not present, it is inferred to be equal to 0.
>
> The variable CtbAddrInRs, specifying a coding tree block address in coding tree block raster scan of a picture, is set equal to slice_segment_address. The variable CtbAddrInTs, specifying a coding tree block address in tile scan, is set equal to CtbAddrRsToTs[CtbAddrInRs]. The variable CuQpDeltaVal, specifying the difference between a luma quantization

Ex. 13 (H.265 specification) at 89.

> **entry_point_offset_minus1**[i] plus 1 specifies the i-th entry point offset in bytes, and is represented by offset_len_minus1 plus 1 bits. The slice segment data that follows the slice segment header consists of num_entry_point_offsets + 1 subsets, with subset index values ranging from 0 to num_entry_point_offsets, inclusive. The first byte of the slice segment data is considered byte 0. When present, emulation prevention bytes that appear in the slice segment data portion of the coded slice segment NAL unit are counted as part of the slice segment data for purposes of subset identification. Subset 0 consists of bytes 0 to entry_point_offset_minus1[0], inclusive, of the coded slice segment data, subset k, with k in the range of 1 to num_entry_point_offsets − 1, inclusive, consists of bytes firstByte[k] to lastByte[k], inclusive, of the coded slice segment data with firstByte[k] and lastByte[k] defined as:

Ex. 13 (H.265 specification) at 93.

> When tiles_enabled_flag is equal to 0 and entropy_coding_sync_enabled_flag is equal to 1, each subset k with k in the range of 0 to num_entry_point_offsets, inclusive, shall consist of all coded bits of all coding tree units in the slice segment that include luma coding tree blocks that are in the same luma coding tree block row of the picture, and the number of

> subsets (i.e., the value of num_entry_point_offsets + 1) shall be equal to the number of coding tree block rows of the picture that contain coding tree units that are in the coded slice segment.
>
> NOTE 6 – The last subset (i.e., subset k for k equal to num_entry_point_offsets) may or may not contain all coding tree units that include luma coding tree blocks that are in a luma coding tree block row of the picture.
>
> When tiles_enabled_flag is equal to 1 and entropy_coding_sync_enabled_flag is equal to 1, each subset k with k in the range of 0 to num_entry_point_offsets, inclusive, shall consist of all coded bits of all coding tree units in the slice segment that include luma coding tree blocks that are in the same luma coding tree block row of a tile, and the number of subsets (i.e., the value of num_entry_point_offsets + 1) shall be equal to the number of luma coding tree block rows of a tile that contain coding tree units that are in the coded slice segment.

Ex. 13 (H.265 specification) at 93-94.

9.3.4.3 Arithmetic decoding process

9.3.4.3.1 General

Inputs to this process are ctxTable, ctxIdx and bypassFlag, as derived in clause 9.3.4.2, and the state variables ivlCurrRange and ivlOffset of the arithmetic decoding engine.

Output of this process is the value of the bin.

Figure 9-5 illustrates the whole arithmetic decoding process for a single bin. For decoding the value of a bin, the context index table ctxTable and the ctxIdx are passed to the arithmetic decoding process DecodeBin(ctxTable, ctxIdx), which is specified as follows:

Ex. 13 (H.265 specification) at 212.

211. The slices must be encoded sequentially because when entropy_coding_sync_enabled_flag is enabled, for the first CTU block in a row (not including the first row), the context variables are initialized by synchronizing them with the context variables of a block of the second CTU in the row one above the row being decoded.

9.3.2 Initialization process

9.3.2.1 General

Outputs of this process are initialized CABAC internal variables and the initialized Rice parameter initialization states StatCoeff.

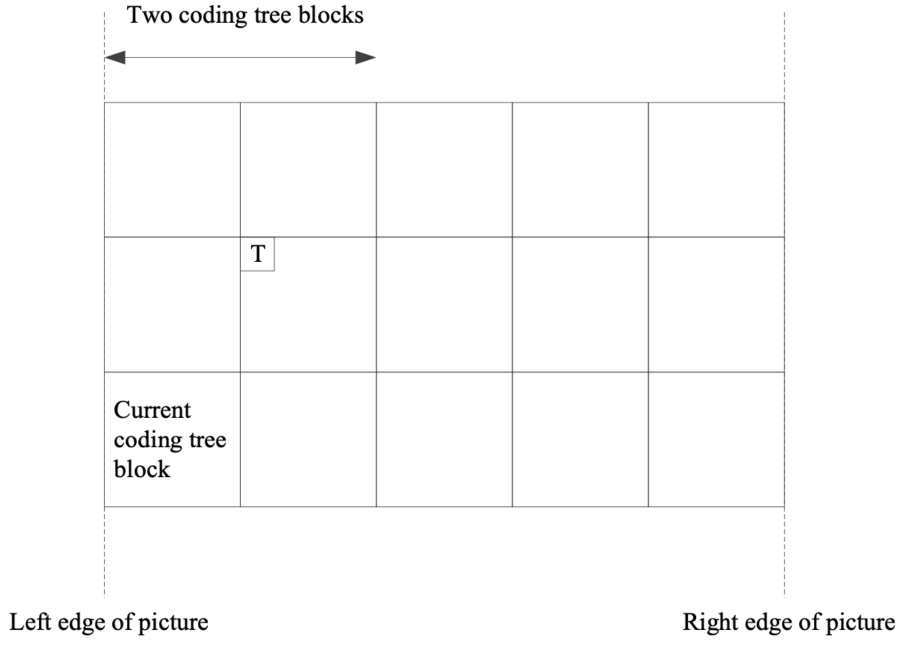
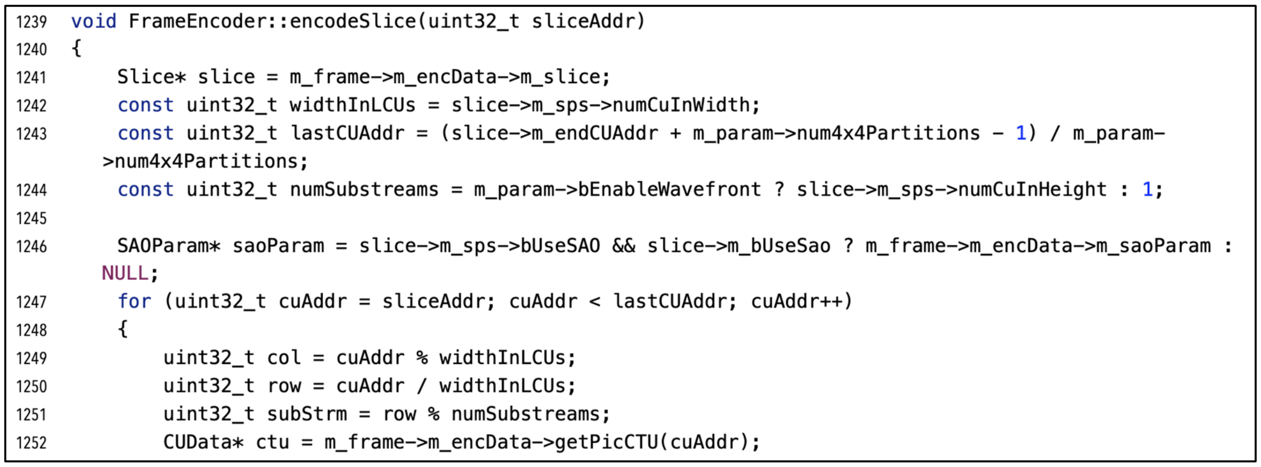


Figure 9-2 – Spatial neighbour T that is used to invoke the coding tree block availability derivation process relative to the current coding tree block (informative)

Ex. 13 (H.265 specification) at 186.

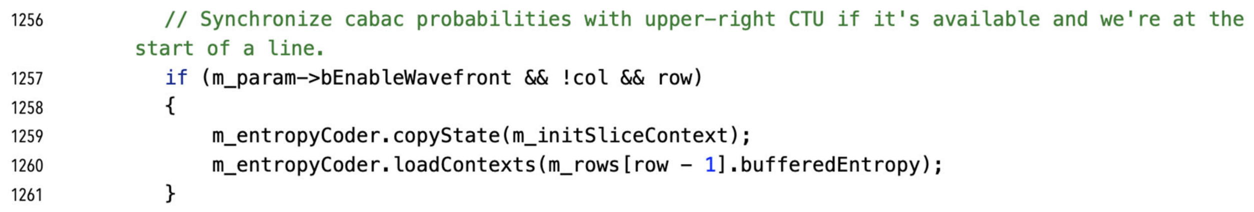
212. Source code for the x265 encoder used by Snap confirms that encoding proceeds sequentially and is synchronized at the start of each line with a block from the previous line.



```
void FrameEncoder::encodeSlice(uint32_t sliceAddr)
{
    Slice* slice = m_frame->m_encData->m_slice;
    const uint32_t widthInLCUs = slice->m_sps->numCuInWidth;
    const uint32_t lastCUAddr = (slice->m_endCUAddr + m_param->num4x4Partitions - 1) / m_param-
  >num4x4Partitions;
    const uint32_t numSubstreams = m_param->bEnableWavefront ? slice->m_sps->numCuInHeight : 1;

    SAOParam* saoParam = slice->m_sps->bUseSAO && slice->m_bUseSao ? m_frame->m_encData->m_saoParam :
  NULL;
    for (uint32_t cuAddr = sliceAddr; cuAddr < lastCUAddr; cuAddr++)
    {
        uint32_t col = cuAddr % widthInLCUs;
        uint32_t row = cuAddr / widthInLCUs;
        uint32_t subStrm = row % numSubstreams;
        CUData* ctu = m_frame->m_encData->getPicCTU(cuAddr);
```

Ex. 22 (frameencoder.cpp) at 23.

213. The x265 source code used by Snap shows that when WPP is enabled, probabilities are synchronized at the start of each line.



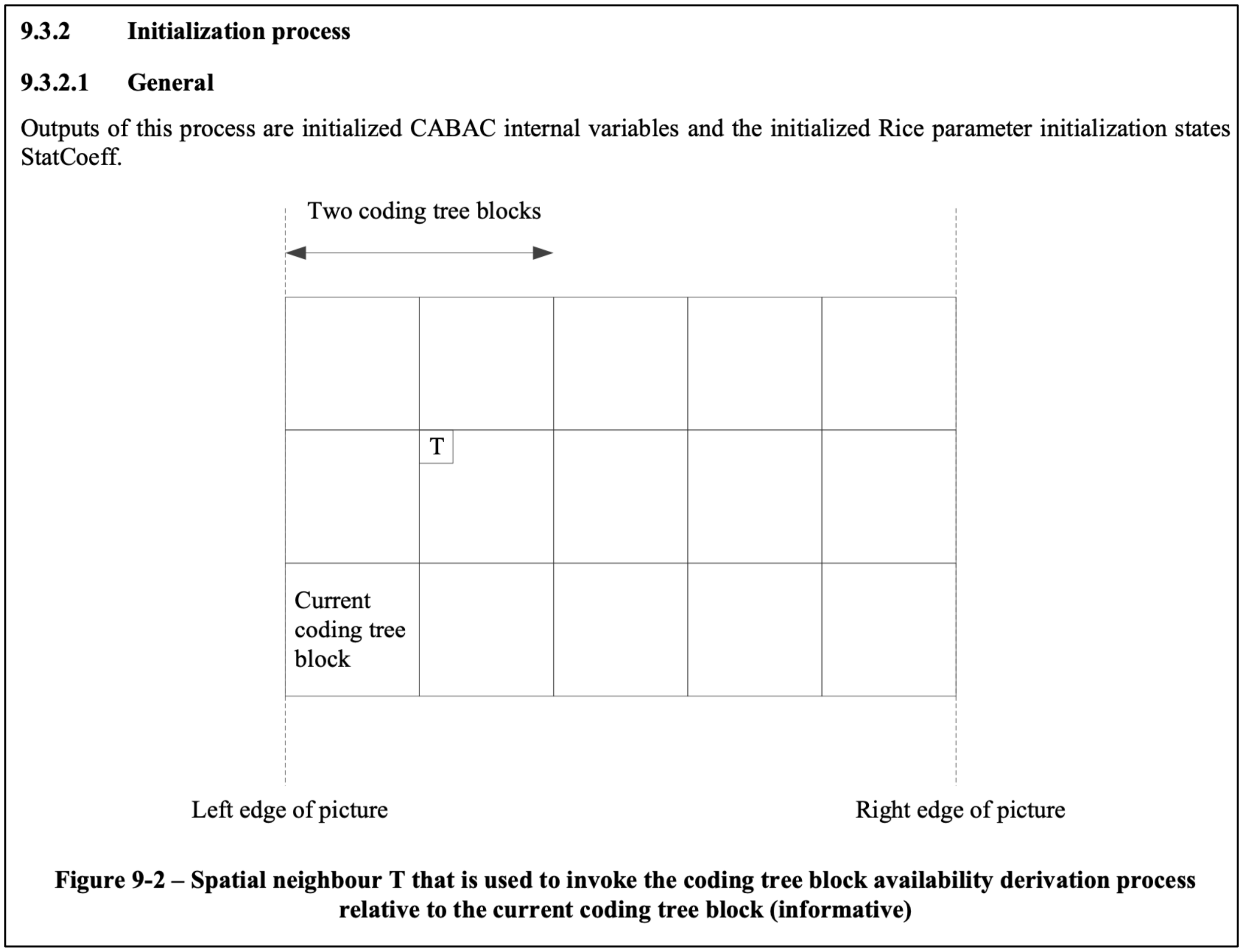
```
        // Synchronize cabac probabilities with upper-right CTU if it's available and we're at the
      start of a line.
        if (m_param->bEnableWavefront && !col && row)
        {
            m_entropyCoder.copyState(m_initSliceContext);
            m_entropyCoder.loadContexts(m_rows[row - 1].bufferedEntropy);
        }
```

Ex. 22 (frameencoder.cpp) at 24.

214. Snap's Accused Services perform "in entropy encoding a predetermined entropy slice, entropy encoding a current part of the predetermined entropy slice based on the respective probability estimations of the predetermined entropy slice as adapted using the previously encoded part of the predetermined entropy slice, and probability estimations as used in the entropy encoding of a spatially neighboring, in the entropy slice order, preceding entropy slice at a neighboring part of the spatially neighboring preceding entropy slice, wherein each entropy slice comprises entropy encoded data for a corresponding portion of the sample array, the different portions forming rows of

blocks of the sample array with the blocks being regularly arranged in rows and columns so that portions corresponding to the entropy slices comprise a same number of blocks, and the entropy coding path points in parallel along the rows of the blocks."

215. For example, the H.265 standard explains that entropy coding proceeds row by row, and for the first CTU block in a row (not including the first row), the context variables are initialized by synchronizing them with the context variables of a neighboring block of the second CTU in the row one above the row being decoded.



9.3.2 Initialization process

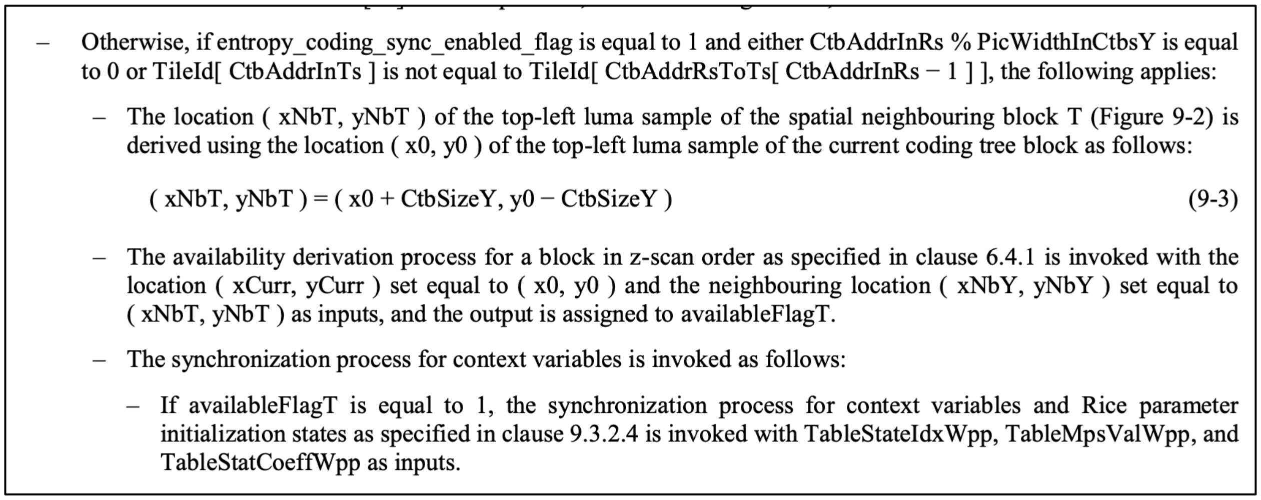
9.3.2.1 General

Outputs of this process are initialized CABAC internal variables and the initialized Rice parameter initialization states StatCoeff.

Two coding tree blocks
T
Current coding tree block
Left edge of picture
Right edge of picture

Figure 9-2 – Spatial neighbour T that is used to invoke the coding tree block availability derivation process relative to the current coding tree block (informative)

Ex. 13 (H.265 specification) at 186.



– Otherwise, if entropy_coding_sync_enabled_flag is equal to 1 and either CtbAddrInRs % PicWidthInCtbsY is equal to 0 or TileId[CtbAddrInTs] is not equal to TileId[CtbAddrRsToTs[CtbAddrInRs − 1]], the following applies:

  – The location (xNbT, yNbT) of the top-left luma sample of the spatial neighbouring block T (Figure 9-2) is derived using the location (x0, y0) of the top-left luma sample of the current coding tree block as follows:

$$(xNbT, yNbT) = (x0 + CtbSizeY, y0 − CtbSizeY) \quad (9\text{-}3)$$

  – The availability derivation process for a block in z-scan order as specified in clause 6.4.1 is invoked with the location (xCurr, yCurr) set equal to (x0, y0) and the neighbouring location (xNbY, yNbY) set equal to (xNbT, yNbT) as inputs, and the output is assigned to availableFlagT.

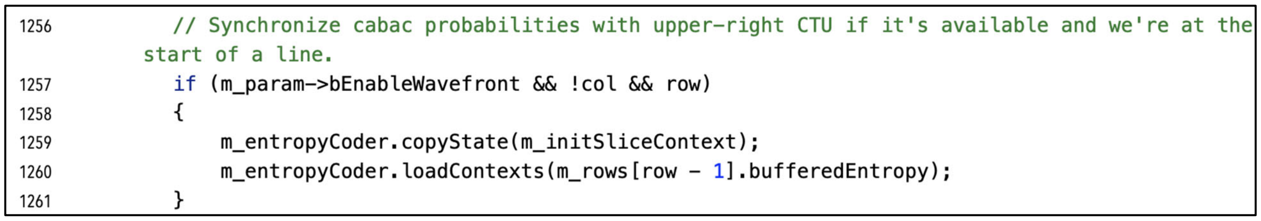
  – The synchronization process for context variables is invoked as follows:

    – If availableFlagT is equal to 1, the synchronization process for context variables and Rice parameter initialization states as specified in clause 9.3.2.4 is invoked with TableStateIdxWpp, TableMpsValWpp, and TableStatCoeffWpp as inputs.

Ex. 13 (H.265 specification) at 186.

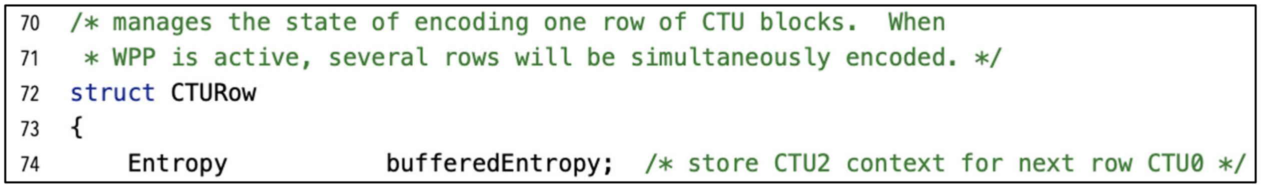
216. This probability estimation is updated as encoding continues.

217. Source code for the x265 encoder used by Snap confirms that when WPP is enabled, probabilities are synchronized with the neighboring "upper-right CTU" at the start of each line. The second CTU in each row ("CTU2") is stored for use with the beginning of the next row ("CTU0").



```
        // Synchronize cabac probabilities with upper-right CTU if it's available and we're at the
    start of a line.
        if (m_param->bEnableWavefront && !col && row)
        {
            m_entropyCoder.copyState(m_initSliceContext);
            m_entropyCoder.loadContexts(m_rows[row - 1].bufferedEntropy);
        }
```

Ex. 22 (frameencoder.cpp) at 24.



```
/* manages the state of encoding one row of CTU blocks.  When
 * WPP is active, several rows will be simultaneously encoded. */
struct CTURow
{
    Entropy           bufferedEntropy;  /* store CTU2 context for next row CTU0 */
```

Ex. 23 (frameencoder.h) at 2.

218. Snap's Accused Services perform "initializing, for each respective entropy slice, the probability estimations before encoding a first block of the portion corresponding to the respective entropy slice along the respective coding path with probability estimations manifesting themselves after having entropy encoded a second block of the portion corresponding to, in the entropy slice

order, the preceding entropy slice along the respective coding path."

219. For example, the H.265 standard explains that entropy coding proceeds row by row, and for the first CTU block in a row (not including the first row), the context variables are initialized by synchronizing them with the context variables of a neighboring block of the second CTU in the row one above the row being decoded.

9.3.2 Initialization process

9.3.2.1 General

Outputs of this process are initialized CABAC internal variables and the initialized Rice parameter initialization states StatCoeff.

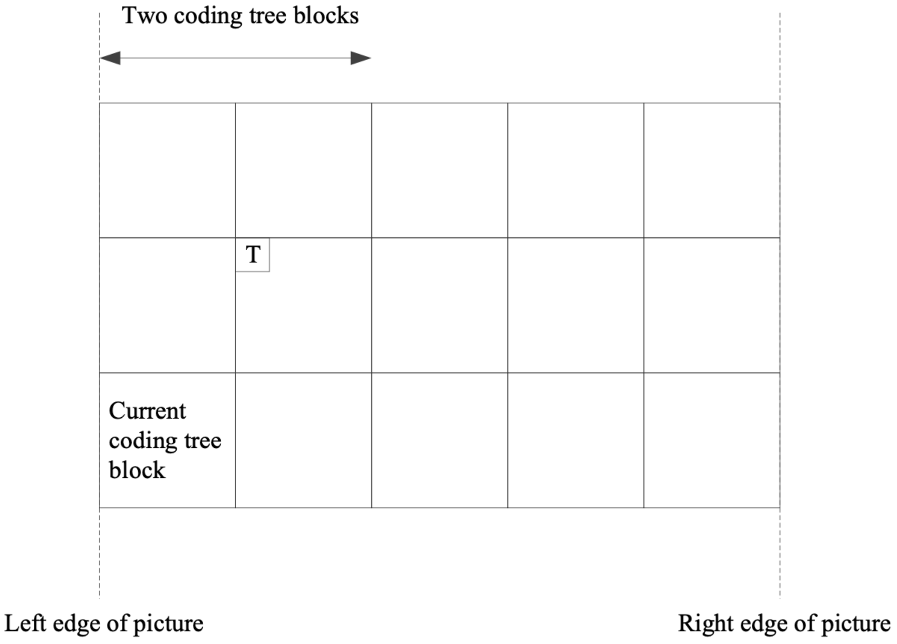
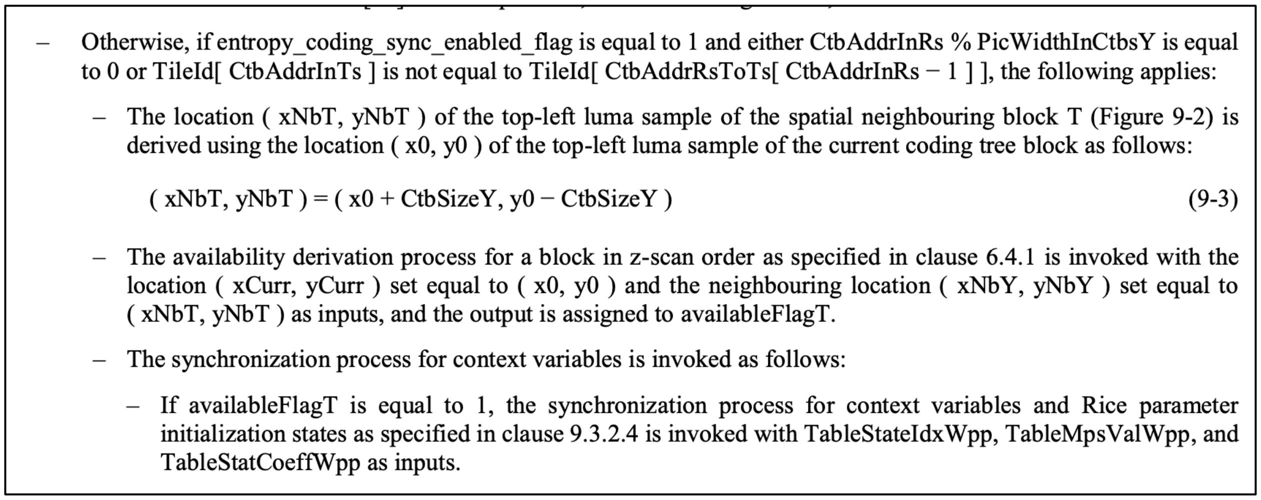


Figure 9-2 – Spatial neighbour T that is used to invoke the coding tree block availability derivation process relative to the current coding tree block (informative)

Ex. 13 (H.265 specification) at 186.



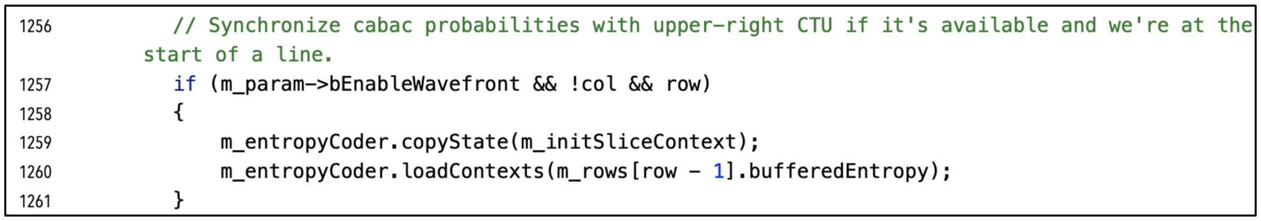
– Otherwise, if entropy_coding_sync_enabled_flag is equal to 1 and either CtbAddrInRs % PicWidthInCtbsY is equal to 0 or TileId[CtbAddrInTs] is not equal to TileId[CtbAddrRsToTs[CtbAddrInRs − 1]], the following applies:
  – The location (xNbT, yNbT) of the top-left luma sample of the spatial neighbouring block T (Figure 9-2) is derived using the location (x0, y0) of the top-left luma sample of the current coding tree block as follows:

$$(xNbT, yNbT) = (x0 + CtbSizeY, y0 − CtbSizeY) \quad (9\text{-}3)$$

  – The availability derivation process for a block in z-scan order as specified in clause 6.4.1 is invoked with the location (xCurr, yCurr) set equal to (x0, y0) and the neighbouring location (xNbY, yNbY) set equal to (xNbT, yNbT) as inputs, and the output is assigned to availableFlagT.
  – The synchronization process for context variables is invoked as follows:
    – If availableFlagT is equal to 1, the synchronization process for context variables and Rice parameter initialization states as specified in clause 9.3.2.4 is invoked with TableStateIdxWpp, TableMpsValWpp, and TableStatCoeffWpp as inputs.

Ex. 13 (H.265 specification) at 186.

220. Source code for the x265 encoder used by Snap confirms that when WPP is enabled, probabilities are initialized with the probability estimations from the "upper-right CTU", i.e., the second block from the prior line, at the start of each line.



```
        // Synchronize cabac probabilities with upper-right CTU if it's available and we're at the
      start of a line.
        if (m_param->bEnableWavefront && !col && row)
        {
            m_entropyCoder.copyState(m_initSliceContext);
            m_entropyCoder.loadContexts(m_rows[row - 1].bufferedEntropy);
        }
```

Ex. 22 (frameencoder.cpp) at 24.

<u>COUNT IV: PATENT INFRINGEMENT OF THE '272 PATENT</u>

221. Dolby incorporates by reference the preceding paragraphs as though fully set forth herein.

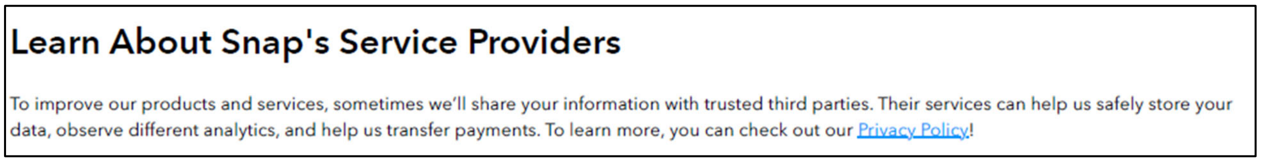
222. Snap had knowledge and notice of the '272 Patent no later than March 18, 2026, and, as discussed above, it has had knowledge and notice of family members of the '272 Patent no later than November 24, 2023 and again through communications on August 13, 2025 and November 21, 2025.

223. Snap infringes the '272 Patent by making, using, offering to sell, and/or selling the patented invention within the United States, and/or importing the patented invention into the United States. Snap thus directly infringes the '272 Patent literally and/or under the Doctrine of Equivalents, in violation of 35 U.S.C. § 271. For example, Snap directly infringes the '272 Patent through its

encoding, decoding, and transcoding of H.265 (HEVC) and AV1-compliant video in the United States.

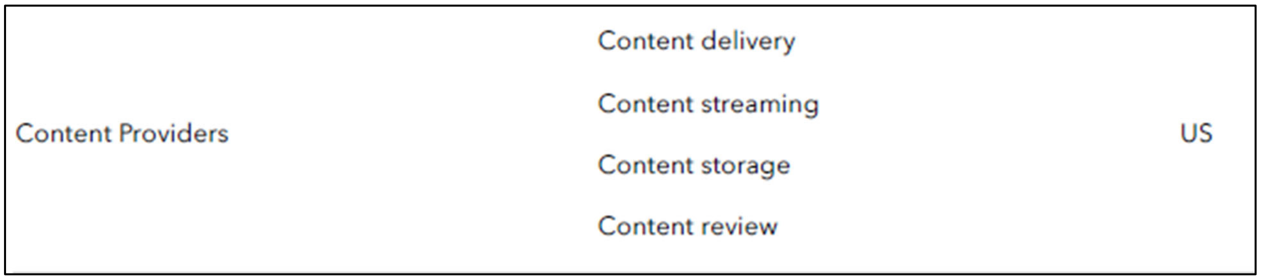
224. On information and belief, Snap indirectly infringes claims of the ’272 Patent, as provided in 35 U.S.C. § 271(b), by knowingly inducing infringement by others, such as Snap’s partners and vendors, in this District and elsewhere in the United States.

225. For example, Snap uses the infrastructure of its providers to encode and decode video. Snap’s web page indicates that these providers perform content delivery, streaming, storage, and review, including the encoding and decoding of video.



Learn About Snap's Service Providers

To improve our products and services, sometimes we'll share your information with trusted third parties. Their services can help us safely store your data, observe different analytics, and help us transfer payments. To learn more, you can check out our Privacy Policy!



Content Providers	Content delivery Content streaming Content storage Content review	US

Ex. 12 at 1.

226. As a result of Snap’s inducement, Snap’s providers perform infringing video coding and directly infringe the ’272 Patent. Snap has performed and continues to perform these affirmative acts with knowledge of the ’272 Patent and with the intent, or willful blindness, that the induced acts directly infringe the ’272 Patent.

227. Upon information and belief, Snap derives revenue, directly and indirectly, from the activities relating to the Accused Services and to infringement of the ’272 Patent, including in this District and elsewhere in the United States.

228. Snap’s infringement of the ’272 Patent is willful and deliberate. As detailed above, Snap had

knowledge of the ’272 Patent and had knowledge, or was willfully blind, as to its infringement of the ’272 Patent. As discussed above, Snap has had actual knowledge of the ’272 Patent since before this suit. Further, as discussed above, Snap knew or should have known that its actions infringe and actively induce infringement of the ’272 Patent. As discussed above, Snap specifically intended that both itself and/or its customers and providers infringe the ’272 Patent.

229. Snap’s infringement of the ’272 Patent has damaged and will continue to damage Dolby.

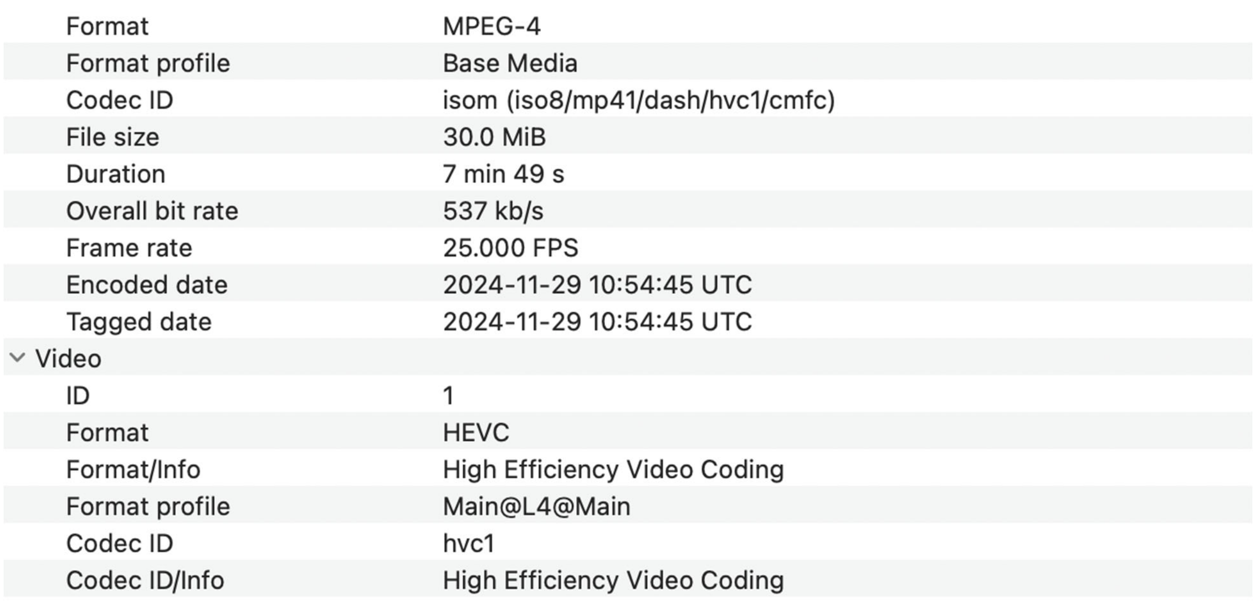
230. Claim 17 of the ’272 Patent recites:

> 17. An apparatus for encoding comprising:
>
> a decomposer configured to:
>
> receive a sequence of syntax elements having a value range which is sub-divided into a plurality of disjoint portions, related to level values of transform coefficients of a transform coefficient block, and
>
> obtain a sequence of source symbols based on the sequence of syntax elements by decomposing each syntax element into a corresponding set of source symbols based on a portion of the plurality of disjoint portions associated with the syntax element, such that a combination of values of the source symbols of the set yields the value of the syntax element; and
>
> a symbol encoder configured to:
>
> receive the sequence of source symbols,
>
> sub-divide the sequence of source symbols into a first sequence of source symbols and a second sequence of source symbols, and
>
> encode a source symbol of the first sequence using arithmetic context-based entropy coding and a source symbol of the second sequence using Exp-Golomb coding.

231. Snap’s Accused Services include “an apparatus for encoding.”

232. For example, Snap’s backend servers encode H.265-compliant (HEVC) video for streaming to devices.

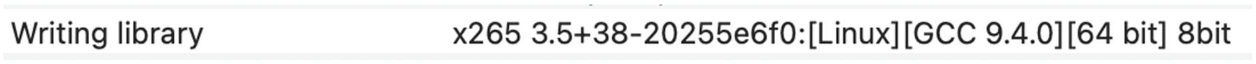
233. Videos downloaded from Snap show that an HEVC format is used:



Format	MPEG-4
Format profile	Base Media
Codec ID	isom (iso8/mp41/dash/hvc1/cmfc)
File size	30.0 MiB
Duration	7 min 49 s
Overall bit rate	537 kb/s
Frame rate	25.000 FPS
Encoded date	2024-11-29 10:54:45 UTC
Tagged date	2024-11-29 10:54:45 UTC
Video	
ID	1
Format	HEVC
Format/Info	High Efficiency Video Coding
Format profile	Main@L4@Main
Codec ID	hvc1
Codec ID/Info	High Efficiency Video Coding

(Video from Snap as viewed in MediaInfo app.)

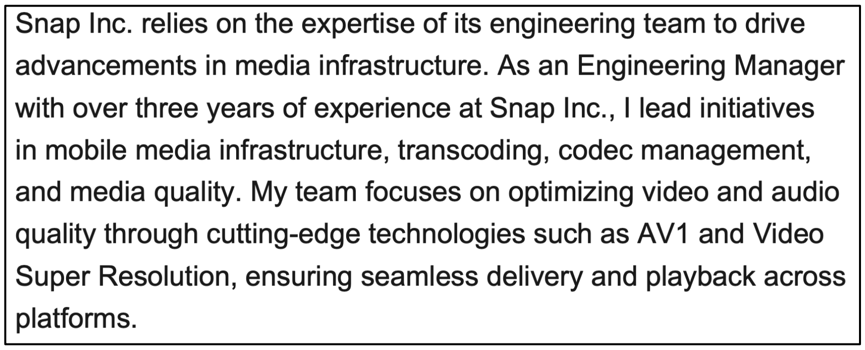
234. Snap uses the x265 library, version 3.5, revision 20255e6f0, to encode HEVC video.



Writing library	x265 3.5+38-20255e6f0:[Linux][GCC 9.4.0][64 bit] 8bit

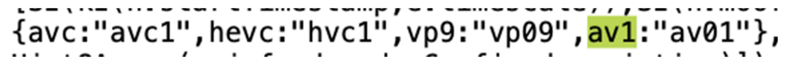
(Video from Snap as viewed in MediaInfo app.)

235. In addition, Snap's backend servers encode AV1-compliant video for streaming to devices. Snap indicates that it uses AV1 to ensure "seamless delivery and playback across devices," and the source code for its web app repeatedly refers to the software and hardware AV1 decoders it uses to decode video on client devices.



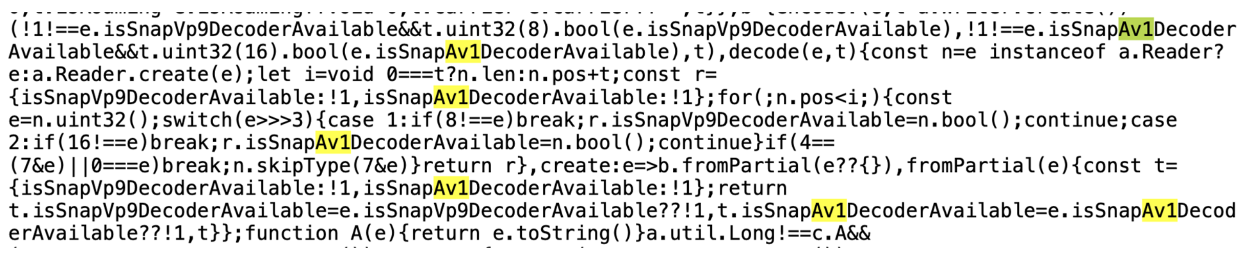
Snap Inc. relies on the expertise of its engineering team to drive advancements in media infrastructure. As an Engineering Manager with over three years of experience at Snap Inc., I lead initiatives in mobile media infrastructure, transcoding, codec management, and media quality. My team focuses on optimizing video and audio quality through cutting-edge technologies such as AV1 and Video Super Resolution, ensuring seamless delivery and playback across platforms.

Ex. 9 (Junhong Nie LinkedIn).



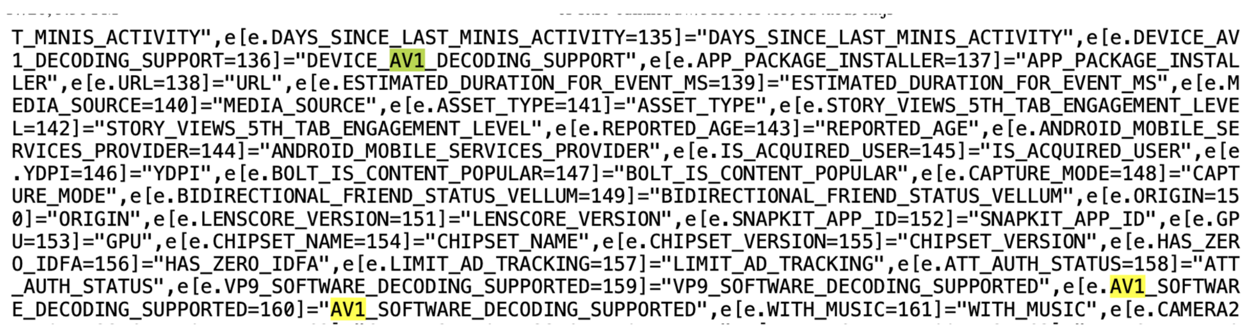
```
{avc:"avc1",hevc:"hvc1",vp9:"vp09",av1:"av01"},
```

Ex. 14 (Snap web source code) at 168.



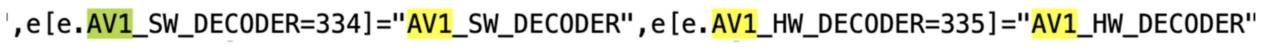
```
(!1!==e.isSnapVp9DecoderAvailable&&t.uint32(8).bool(e.isSnapVp9DecoderAvailable),!1!==e.isSnapAv1Decoder
Available&&t.uint32(16).bool(e.isSnapAv1DecoderAvailable),t),decode(e,t){const n=e instanceof a.Reader?
e:a.Reader.create(e);let i=void 0===t?n.len:n.pos+t;const r=
{isSnapVp9DecoderAvailable:!1,isSnapAv1DecoderAvailable:!1};for(;n.pos<i;){const
e=n.uint32();switch(e>>>3){case 1:if(8!==e)break;r.isSnapVp9DecoderAvailable=n.bool();continue;case
2:if(16!==e)break;r.isSnapAv1DecoderAvailable=n.bool();continue}if(4==
(7&e)||0===e)break;n.skipType(7&e)}return r},create:e=>b.fromPartial(e??{}),fromPartial(e){const t=
{isSnapVp9DecoderAvailable:!1,isSnapAv1DecoderAvailable:!1};return
t.isSnapVp9DecoderAvailable=e.isSnapVp9DecoderAvailable??!1,t.isSnapAv1DecoderAvailable=e.isSnapAv1Decod
erAvailable??!1,t}};function A(e){return e.toString()}a.util.Long!==c.A&&
```

Ex. 14 (Snap web source code) at 524.



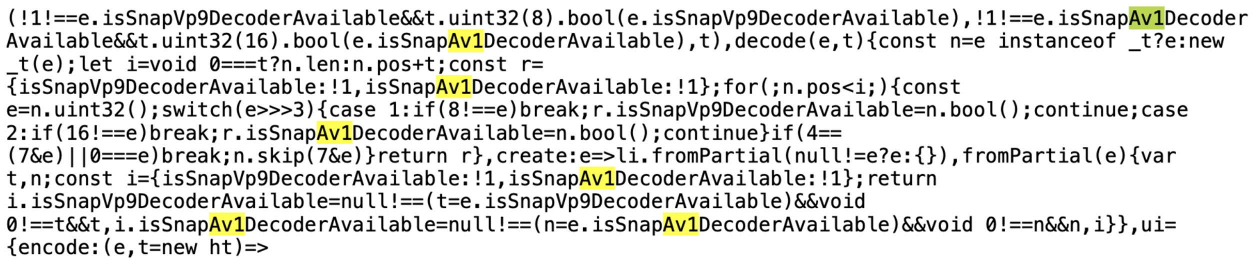
```
T_MINIS_ACTIVITY",e[e.DAYS_SINCE_LAST_MINIS_ACTIVITY=135]="DAYS_SINCE_LAST_MINIS_ACTIVITY",e[e.DEVICE_AV
1_DECODING_SUPPORT=136]="DEVICE_AV1_DECODING_SUPPORT",e[e.APP_PACKAGE_INSTALLER=137]="APP_PACKAGE_INSTAL
LER",e[e.URL=138]="URL",e[e.ESTIMATED_DURATION_FOR_EVENT_MS=139]="ESTIMATED_DURATION_FOR_EVENT_MS",e[e.M
EDIA_SOURCE=140]="MEDIA_SOURCE",e[e.ASSET_TYPE=141]="ASSET_TYPE",e[e.STORY_VIEWS_5TH_TAB_ENGAGEMENT_LEVE
L=142]="STORY_VIEWS_5TH_TAB_ENGAGEMENT_LEVEL",e[e.REPORTED_AGE=143]="REPORTED_AGE",e[e.ANDROID_MOBILE_SE
RVICES_PROVIDER=144]="ANDROID_MOBILE_SERVICES_PROVIDER",e[e.IS_ACQUIRED_USER=145]="IS_ACQUIRED_USER",e[e
.YDPI=146]="YDPI",e[e.BOLT_IS_CONTENT_POPULAR=147]="BOLT_IS_CONTENT_POPULAR",e[e.CAPTURE_MODE=148]="CAPT
URE_MODE",e[e.BIDIRECTIONAL_FRIEND_STATUS_VELLUM=149]="BIDIRECTIONAL_FRIEND_STATUS_VELLUM",e[e.ORIGIN=15
0]="ORIGIN",e[e.LENSCORE_VERSION=151]="LENSCORE_VERSION",e[e.SNAPKIT_APP_ID=152]="SNAPKIT_APP_ID",e[e.GP
U=153]="GPU",e[e.CHIPSET_NAME=154]="CHIPSET_NAME",e[e.CHIPSET_VERSION=155]="CHIPSET_VERSION",e[e.HAS_ZER
O_IDFA=156]="HAS_ZERO_IDFA",e[e.LIMIT_AD_TRACKING=157]="LIMIT_AD_TRACKING",e[e.ATT_AUTH_STATUS=158]="ATT
_AUTH_STATUS",e[e.VP9_SOFTWARE_DECODING_SUPPORTED=159]="VP9_SOFTWARE_DECODING_SUPPORTED",e[e.AV1_SOFTWAR
E_DECODING_SUPPORTED=160]="AV1_SOFTWARE_DECODING_SUPPORTED",e[e.WITH_MUSIC=161]="WITH_MUSIC",e[e.CAMERA2
```

Ex. 14 (Snap web source code) at 539.



```
',e[e.AV1_SW_DECODER=334]="AV1_SW_DECODER",e[e.AV1_HW_DECODER=335]="AV1_HW_DECODER"
```

Ex. 14 (Snap web source code) at 540.



```
(!1!==e.isSnapVp9DecoderAvailable&&t.uint32(8).bool(e.isSnapVp9DecoderAvailable),!1!==e.isSnapAv1Decoder
Available&&t.uint32(16).bool(e.isSnapAv1DecoderAvailable),t),decode(e,t){const n=e instanceof _t?e:new
_t(e);let i=void 0===t?n.len:n.pos+t;const r=
{isSnapVp9DecoderAvailable:!1,isSnapAv1DecoderAvailable:!1};for(;n.pos<i;){const
e=n.uint32();switch(e>>>3){case 1:if(8!==e)break;r.isSnapVp9DecoderAvailable=n.bool();continue;case
2:if(16!==e)break;r.isSnapAv1DecoderAvailable=n.bool();continue}if(4==
(7&e)||0===e)break;n.skip(7&e)}return r},create:e=>li.fromPartial(null!=e?e:{}),fromPartial(e){var
t,n;const i={isSnapVp9DecoderAvailable:!1,isSnapAv1DecoderAvailable:!1};return
i.isSnapVp9DecoderAvailable=null!==(t=e.isSnapVp9DecoderAvailable)&&void
0!==t&&t,i.isSnapAv1DecoderAvailable=null!==(n=e.isSnapAv1DecoderAvailable)&&void 0!==n&&n,i}},ui=
{encode:(e,t=new ht)=>
```

Ex. 14 (Snap web source code) at 695.

See also Ex. 14 (Snap web source code) at 169, 696-98, 831, 1358-1360.

236. Snap's Accused Services include "a decomposer configured to: receive a sequence of syntax elements having a value range which is sub-divided into a plurality of disjoint portions, related to level values of transform coefficients of a transform coefficient block, and obtain a sequence of source symbols based on the sequence of syntax elements by decomposing each syntax element into a corresponding set of source symbols based on a portion of the plurality of disjoint portions

associated with the syntax element, such that a combination of values of the source symbols of the set yields the value of the syntax element."

237. For example, the H.265 specification explains how a sequence of elements, TransCoeffLevel[x0][y0][cIdx][xC][yC], each representing a value range of a transform coefficient residual, is decomposed and signaled.

238. First, in the H.265 specification, sig_coeff_flag, coeff_abs_level_greater1_flag, and coeff_abs_level_greater2_flag indicate whether the absolute value of the transform coefficient is greater than 0, greater than 1, or greater than 2, respectively.

sig_coeff_flag[xC][yC] specifies for the transform coefficient location (xC, yC) within the current transform block whether the corresponding transform coefficient level at the location (xC, yC) is non-zero as follows:

– If sig_coeff_flag[xC][yC] is equal to 0, the transform coefficient level at the location (xC, yC) is set equal to 0.

– Otherwise (sig_coeff_flag[xC][yC] is equal to 1), the transform coefficient level at the location (xC, yC) has a non-zero value.

When sig_coeff_flag[xC][yC] is not present, it is inferred as follows:

– If (xC, yC) is the last significant location (LastSignificantCoeffX, LastSignificantCoeffY) in scan order or all of the following conditions are true, sig_coeff_flag[xC][yC] is inferred to be equal to 1:

  – (xC & 3, yC & 3) is equal to (0, 0)

  – inferSbDcSigCoeffFlag is equal to 1

  – coded_sub_block_flag[xS][yS] is equal to 1

– Otherwise, sig_coeff_flag[xC][yC] is inferred to be equal to 0.

coeff_abs_level_greater1_flag[n] specifies for the scanning position n whether there are absolute values of transform coefficient levels greater than 1.

When coeff_abs_level_greater1_flag[n] is not present, it is inferred to be equal to 0.

coeff_abs_level_greater2_flag[n] specifies for the scanning position n whether there are absolute values of transform coefficient levels greater than 2.

Ex. 13 (H.265 specification) at 106.

239. Second, in the H.265 specification, if the absolute value of the transform coefficient is greater than 2, coeff_abs_level_remaining represents the remaining absolute value. The representation of coeff_abs_level_remaining is further divided into a prefix bin string and suffix bin string.

coeff_abs_level_remaining[n] is the remaining absolute value of a transform coefficient level that is coded with Golomb-Rice code at the scanning position n. When coeff_abs_level_remaining[n] is not present, it is inferred to be equal to 0.

Ex. 13 (H.265 specification) at 107.

9.3.3.10 Binarization process for coeff_abs_level_remaining[ ]

Input to this process is a request for a binarization for the syntax element coeff_abs_level_remaining[n], the current sub-block scan index i, baseLevel, the colour component cIdx and the luma location (x0, y0) specifying the top-left sample of the current luma transform block relative to the top-left luma sample of the picture.

Output of this process is the binarization of the syntax element.

Ex. 13 (H.265 specification) at 204.

The binarization of the syntax element coeff_abs_level_remaining[n] is a concatenation of a prefix bin string and (when present) a suffix bin string.

For the derivation of the prefix bin string, the following applies:

– The prefix value of coeff_abs_level_remaining[n], prefixVal, is derived as follows:

$$\text{prefixVal} = \text{Min}(\ \text{cMax},\ \text{coeff\_abs\_level\_remaining}[\ n\ ]\ ) \quad (9\text{-}23)$$

– The prefix bin string is specified by invoking the TR binarization process as specified in clause 9.3.3.2 for prefixVal with the variables cMax and cRiceParam as inputs.

When the prefix bin string is equal to the bit string of length 4 with all bits equal to 1, the suffix bin string is present and it is derived as follows:

– The suffix value of coeff_abs_level_remaining[n], suffixVal, is derived as follows:

$$\text{suffixVal} = \text{coeff\_abs\_level\_remaining}[\ n\ ] - \text{cMax} \quad (9\text{-}24)$$

– If extended_precision_processing_flag is equal to 0, the suffix bin string is specified by invoking the k-th order EGk binarization process as specified in clause 9.3.3.3 for the binarization of suffixVal with the Exp-Golomb order k set equal to cRiceParam + 1.

– Otherwise (extended_precision_processing_flag is equal to 1), the suffix bin string is specified by invoking the limited k-th order EGk binarization process as specified in clause 9.3.3.4 for the binarization of suffixVal with the variable riceParam set equal to cRiceParam + 1 and the colour component cIdx.

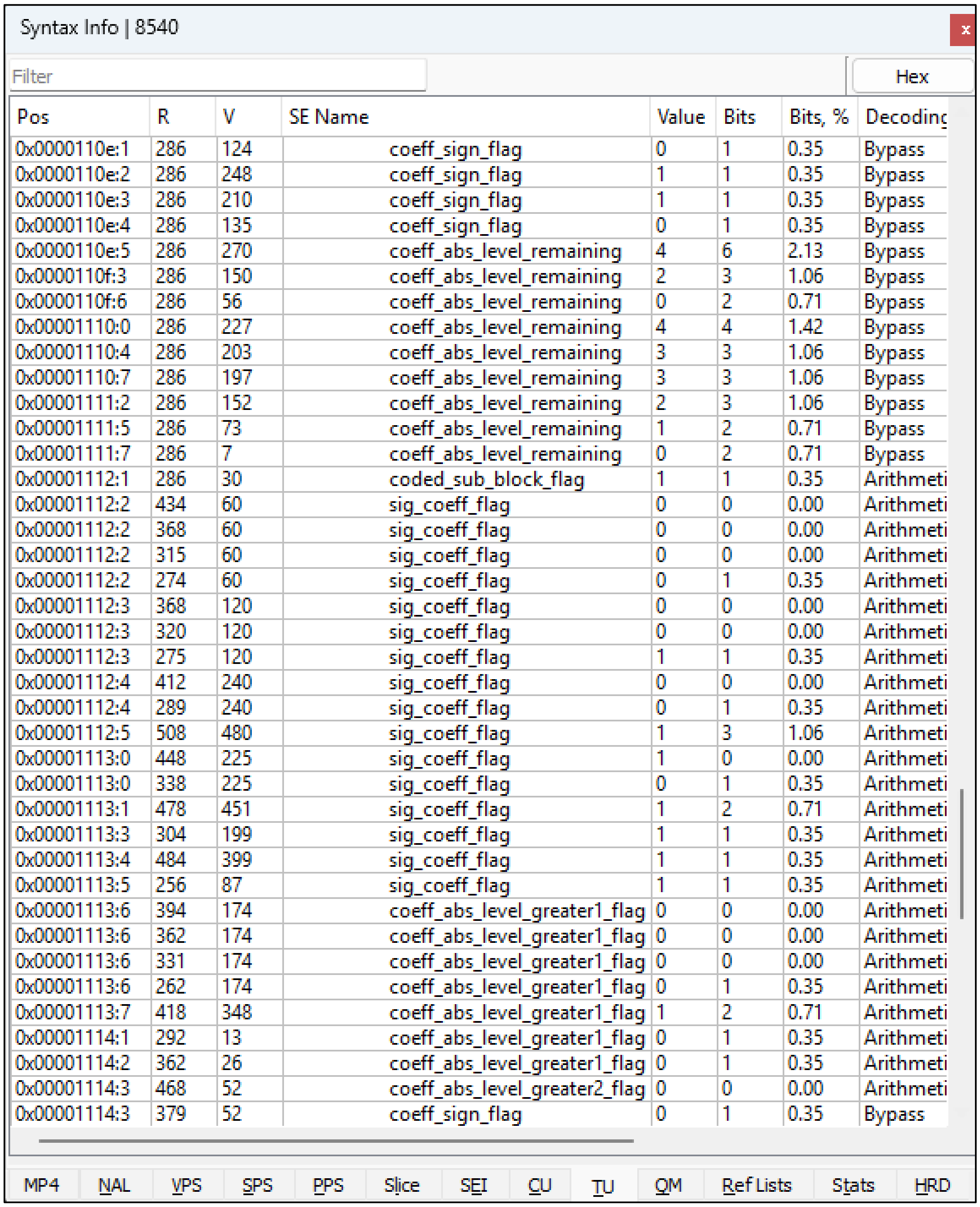
Ex. 13 (H.265 specification) at 205.

240. In the H.265 specification, the combination of the values of the source symbols of the set yields the value of the syntax element:

if(sig_coeff_flag[xC][yC]) {	
baseLevel = 1 + coeff_abs_level_greater1_flag[n] + coeff_abs_level_greater2_flag[n]	
if(baseLevel = = ((numSigCoeff < 8) ? ((n = = lastGreater1ScanPos) ? 3 : 2) : 1))	
coeff_abs_level_remaining[n]	ae(v)
TransCoeffLevel[x0][y0][cIdx][xC][yC] = (coeff_abs_level_remaining[n] + baseLevel) * (1 − 2 * coeff_sign_flag[n])	

Ex. 13 (H.265 specification) at 59.

241. For example, compressed video data captured from Snap shows the use of sig_coeff_flag, coeff_abs_level_greater1_flag, and coeff_abs_level_greater2_flag, as well as encoding of coeff_abs_level_remaining.

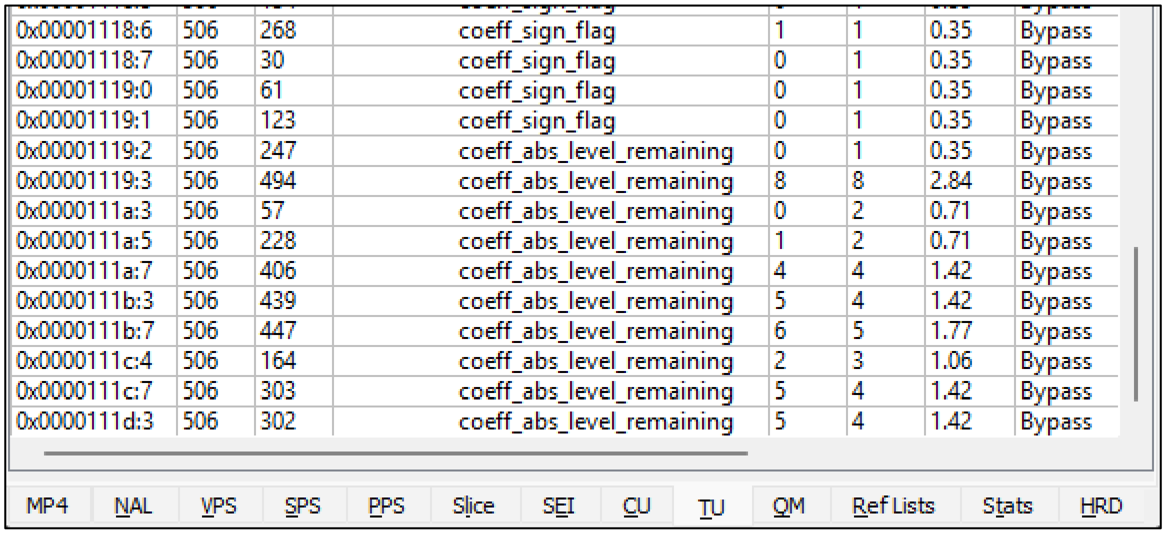

Syntax Info | 8540

Filter

Hex

Pos	R	V	SE Name	Value	Bits	Bits, %	Decoding
0x0000110e:1	286	124	coeff_sign_flag	0	1	0.35	Bypass
0x0000110e:2	286	248	coeff_sign_flag	1	1	0.35	Bypass
0x0000110e:3	286	210	coeff_sign_flag	1	1	0.35	Bypass
0x0000110e:4	286	135	coeff_sign_flag	0	1	0.35	Bypass
0x0000110e:5	286	270	coeff_abs_level_remaining	4	6	2.13	Bypass
0x0000110f:3	286	150	coeff_abs_level_remaining	2	3	1.06	Bypass
0x0000110f:6	286	56	coeff_abs_level_remaining	0	2	0.71	Bypass
0x00001110:0	286	227	coeff_abs_level_remaining	4	4	1.42	Bypass
0x00001110:4	286	203	coeff_abs_level_remaining	3	3	1.06	Bypass
0x00001110:7	286	197	coeff_abs_level_remaining	3	3	1.06	Bypass
0x00001111:2	286	152	coeff_abs_level_remaining	2	3	1.06	Bypass
0x00001111:5	286	73	coeff_abs_level_remaining	1	2	0.71	Bypass
0x00001111:7	286	7	coeff_abs_level_remaining	0	2	0.71	Bypass
0x00001112:1	286	30	coded_sub_block_flag	1	1	0.35	Arithmeti
0x00001112:2	434	60	sig_coeff_flag	0	0	0.00	Arithmeti
0x00001112:2	368	60	sig_coeff_flag	0	0	0.00	Arithmeti
0x00001112:2	315	60	sig_coeff_flag	0	0	0.00	Arithmeti
0x00001112:2	274	60	sig_coeff_flag	0	1	0.35	Arithmeti
0x00001112:3	368	120	sig_coeff_flag	0	0	0.00	Arithmeti
0x00001112:3	320	120	sig_coeff_flag	0	0	0.00	Arithmeti
0x00001112:3	275	120	sig_coeff_flag	1	1	0.35	Arithmeti
0x00001112:4	412	240	sig_coeff_flag	0	0	0.00	Arithmeti
0x00001112:4	289	240	sig_coeff_flag	0	1	0.35	Arithmeti
0x00001112:5	508	480	sig_coeff_flag	1	3	1.06	Arithmeti
0x00001113:0	448	225	sig_coeff_flag	1	0	0.00	Arithmeti
0x00001113:0	338	225	sig_coeff_flag	0	1	0.35	Arithmeti
0x00001113:1	478	451	sig_coeff_flag	1	2	0.71	Arithmeti
0x00001113:3	304	199	sig_coeff_flag	1	1	0.35	Arithmeti
0x00001113:4	484	399	sig_coeff_flag	1	1	0.35	Arithmeti
0x00001113:5	256	87	sig_coeff_flag	1	1	0.35	Arithmeti
0x00001113:6	394	174	coeff_abs_level_greater1_flag	0	0	0.00	Arithmeti
0x00001113:6	362	174	coeff_abs_level_greater1_flag	0	0	0.00	Arithmeti
0x00001113:6	331	174	coeff_abs_level_greater1_flag	0	0	0.00	Arithmeti
0x00001113:6	262	174	coeff_abs_level_greater1_flag	0	1	0.35	Arithmeti
0x00001113:7	418	348	coeff_abs_level_greater1_flag	1	2	0.71	Arithmeti
0x00001114:1	292	13	coeff_abs_level_greater1_flag	0	1	0.35	Arithmeti
0x00001114:2	362	26	coeff_abs_level_greater1_flag	0	1	0.35	Arithmeti
0x00001114:3	468	52	coeff_abs_level_greater2_flag	0	0	0.00	Arithmeti
0x00001114:3	379	52	coeff_sign_flag	0	1	0.35	Bypass

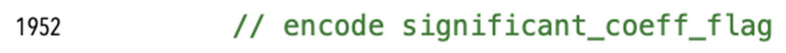
VQ Analyzer screenshot of bitstream collected from Snapchat.



0x00001118:6	506	268	coeff_sign_flag	1	1	0.35	Bypass
0x00001118:7	506	30	coeff_sign_flag	0	1	0.35	Bypass
0x00001119:0	506	61	coeff_sign_flag	0	1	0.35	Bypass
0x00001119:1	506	123	coeff_sign_flag	0	1	0.35	Bypass
0x00001119:2	506	247	coeff_abs_level_remaining	0	1	0.35	Bypass
0x00001119:3	506	494	coeff_abs_level_remaining	8	8	2.84	Bypass
0x0000111a:3	506	57	coeff_abs_level_remaining	0	2	0.71	Bypass
0x0000111a:5	506	228	coeff_abs_level_remaining	1	2	0.71	Bypass
0x0000111a:7	506	406	coeff_abs_level_remaining	4	4	1.42	Bypass
0x0000111b:3	506	439	coeff_abs_level_remaining	5	4	1.42	Bypass
0x0000111b:7	506	447	coeff_abs_level_remaining	6	5	1.77	Bypass
0x0000111c:4	506	164	coeff_abs_level_remaining	2	3	1.06	Bypass
0x0000111c:7	506	303	coeff_abs_level_remaining	5	4	1.42	Bypass
0x0000111d:3	506	302	coeff_abs_level_remaining	5	4	1.42	Bypass

VQ Analyzer screenshot of bitstream collected from Snapchat.

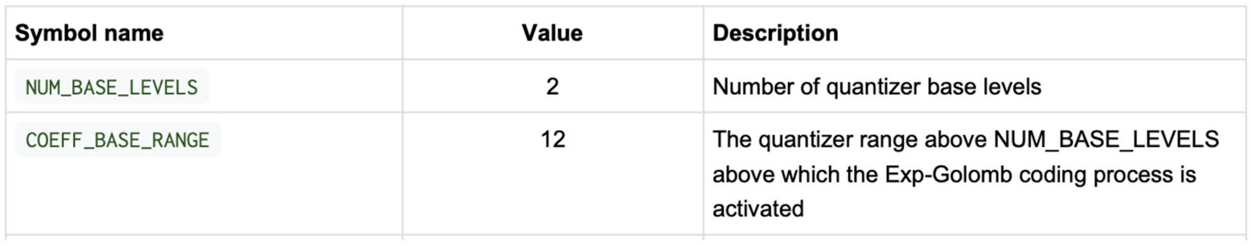
242. For example, the source code for the x265 encoder used by Snap shows the use of sig_coeff_flag, coeff_abs_level_greater1_flag, and coeff_abs_level_greater2_flag.



```
            // encode significant_coeff_flag
```

Ex. 16 (entropy.cpp) at 36; see also *id*. at 34-40 (Entropy::codeCoeffNxN).

243. Similarly, the AV1 specification shows decomposing of transform coefficients into "base levels" and a range above which the Exp-Golomb process is activated.



Symbol name	Value	Description
NUM_BASE_LEVELS	2	Number of quantizer base levels
COEFF_BASE_RANGE	12	The quantizer range above NUM_BASE_LEVELS above which the Exp-Golomb coding process is activated

Ex. 15 (AV1 Spec) at 15.

```
        for ( c = eob - 1; c >= 0; c-- ) {
            pos = scan[ c ]
            if ( c == ( eob - 1 ) ) {
                coeff_base_eob                                          S()
                level = coeff_base_eob + 1
            } else {
                coeff_base                                              S()
                level = coeff_base
            }

            if ( level > NUM_BASE_LEVELS ) {
                for ( idx = 0;
                      idx < COEFF_BASE_RANGE / ( BR_CDF_SIZE - 1 );
                      idx++ ) {
                    coeff_br                                            S()
                    level += coeff_br
                    if ( coeff_br < ( BR_CDF_SIZE - 1 ) )
                        break
                }
            }
            Quant[ pos ] = level
        }
```

Ex. 15 (AV1 Spec) at 91.

```
            if ( Quant[ pos ] >
                ( NUM_BASE_LEVELS + COEFF_BASE_RANGE ) ) {
                length = 0
                do {
                    length++
                    golomb_length_bit                                   L(1)
                } while ( !golomb_length_bit )
```

```
                x = 1
                for ( i = length - 2; i >= 0; i-- ) {
                    golomb_data_bit                                     L(1)
                    x = ( x << 1 ) | golomb_data_bit
                }
               Quant[ pos ] = x + COEFF_BASE_RANGE + NUM_BASE_LEVELS
            }
```

Ex. 15 (AV1 Spec) at 91-92.

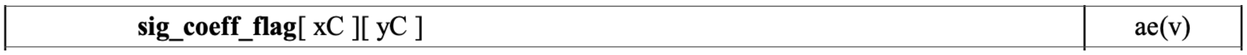
244. In the AV1 specification, the total value can be obtained by adding the individual components:

```
Quant[ pos ] = x + COEFF_BASE_RANGE + NUM_BASE_LEVELS
```

Ex. 15 (AV1 Spec) at 92.

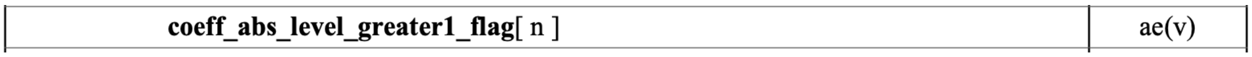
245. Snap's Accused Services include "a symbol encoder configured to: receive the sequence of source symbols, sub-divide the sequence of source symbols into a first sequence of source symbols and a second sequence of source symbols, and encode a source symbol of the first sequence using arithmetic context-based entropy coding and a source symbol of the second sequence using Exp-Golomb coding."

246. For example, in the H.265 specification, portions of the symbols representing the transform coefficients are represented using arithmetic context-based entropy coding.



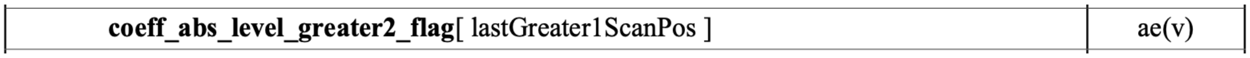
sig_coeff_flag[xC][yC]	ae(v)

Ex. 13 (H.265 specification) at 59.



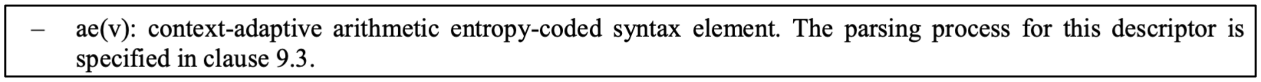
coeff_abs_level_greater1_flag[n]	ae(v)

Ex. 13 (H.265 specification) at 59.



coeff_abs_level_greater2_flag[lastGreater1ScanPos]	ae(v)

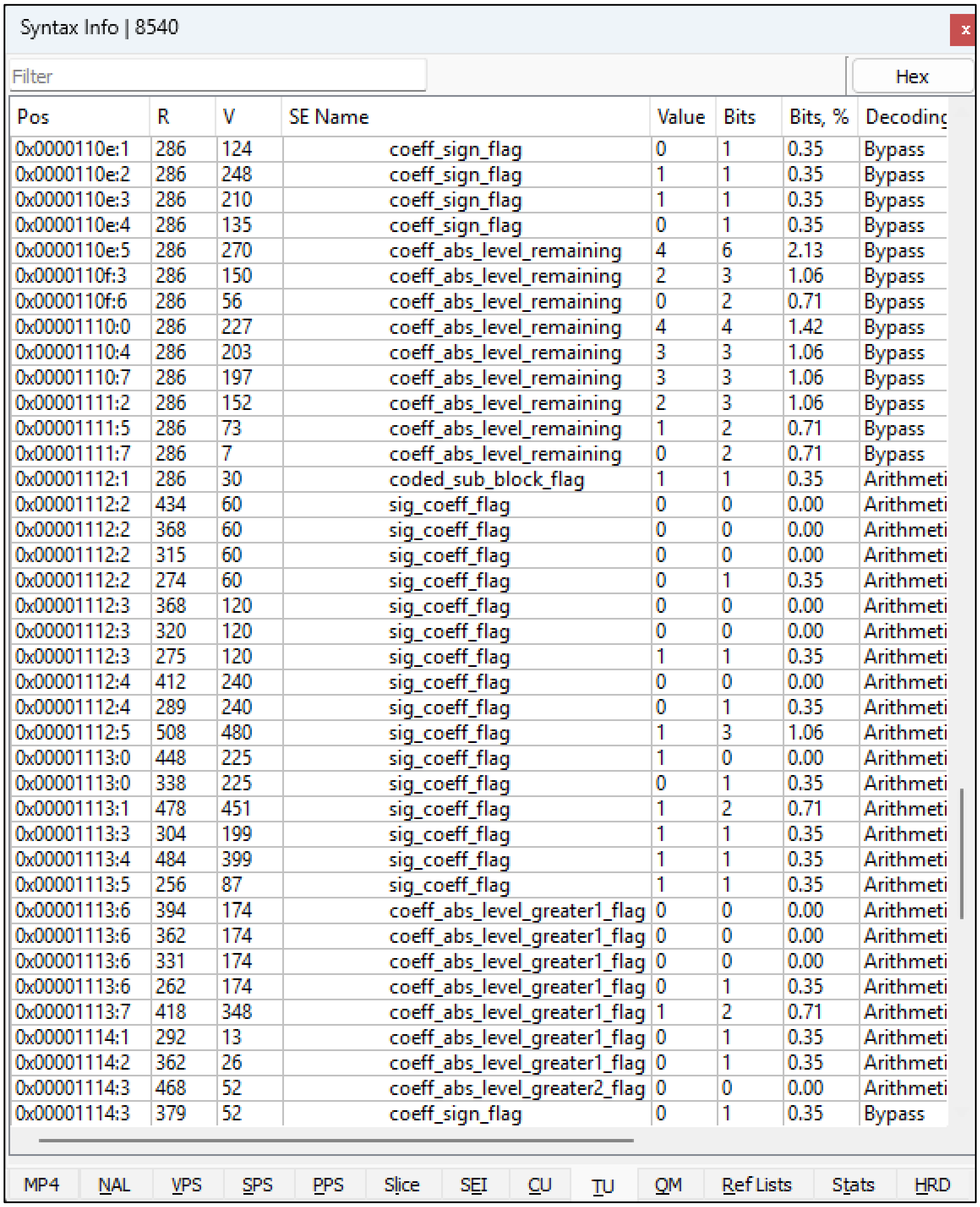
Ex. 13 (H.265 specification) at 60.



– ae(v): context-adaptive arithmetic entropy-coded syntax element. The parsing process for this descriptor is specified in clause 9.3.

Ex. 13 (H.265 specification) at 32.

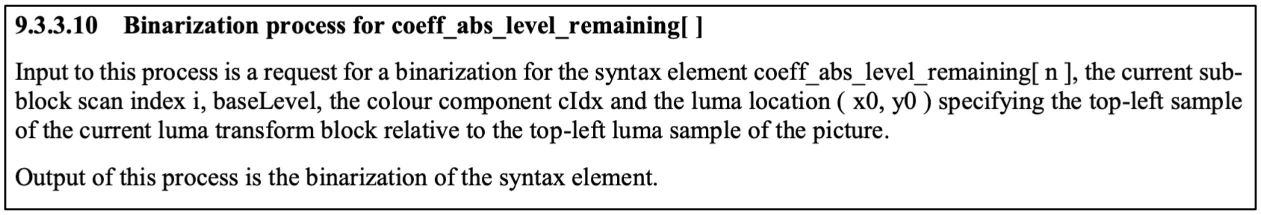
247. For example, compressed video data captured from Snap shows the use of arithmetic coding for sig_coeff_flag, coeff_abs_level_greater1_flag, and coeff_abs_level_greater2_flag, as well as encoding of coeff_abs_level_remaining.


Syntax Info | 8540

Pos	R	V	SE Name	Value	Bits	Bits, %	Decoding
0x0000110e:1	286	124	coeff_sign_flag	0	1	0.35	Bypass
0x0000110e:2	286	248	coeff_sign_flag	1	1	0.35	Bypass
0x0000110e:3	286	210	coeff_sign_flag	1	1	0.35	Bypass
0x0000110e:4	286	135	coeff_sign_flag	0	1	0.35	Bypass
0x0000110e:5	286	270	coeff_abs_level_remaining	4	6	2.13	Bypass
0x0000110f:3	286	150	coeff_abs_level_remaining	2	3	1.06	Bypass
0x0000110f:6	286	56	coeff_abs_level_remaining	0	2	0.71	Bypass
0x00001110:0	286	227	coeff_abs_level_remaining	4	4	1.42	Bypass
0x00001110:4	286	203	coeff_abs_level_remaining	3	3	1.06	Bypass
0x00001110:7	286	197	coeff_abs_level_remaining	3	3	1.06	Bypass
0x00001111:2	286	152	coeff_abs_level_remaining	2	3	1.06	Bypass
0x00001111:5	286	73	coeff_abs_level_remaining	1	2	0.71	Bypass
0x00001111:7	286	7	coeff_abs_level_remaining	0	2	0.71	Bypass
0x00001112:1	286	30	coded_sub_block_flag	1	1	0.35	Arithmeti
0x00001112:2	434	60	sig_coeff_flag	0	0	0.00	Arithmeti
0x00001112:2	368	60	sig_coeff_flag	0	0	0.00	Arithmeti
0x00001112:2	315	60	sig_coeff_flag	0	0	0.00	Arithmeti
0x00001112:2	274	60	sig_coeff_flag	0	1	0.35	Arithmeti
0x00001112:3	368	120	sig_coeff_flag	0	0	0.00	Arithmeti
0x00001112:3	320	120	sig_coeff_flag	0	0	0.00	Arithmeti
0x00001112:3	275	120	sig_coeff_flag	1	1	0.35	Arithmeti
0x00001112:4	412	240	sig_coeff_flag	0	0	0.00	Arithmeti
0x00001112:4	289	240	sig_coeff_flag	0	1	0.35	Arithmeti
0x00001112:5	508	480	sig_coeff_flag	1	3	1.06	Arithmeti
0x00001113:0	448	225	sig_coeff_flag	1	0	0.00	Arithmeti
0x00001113:0	338	225	sig_coeff_flag	0	1	0.35	Arithmeti
0x00001113:1	478	451	sig_coeff_flag	1	2	0.71	Arithmeti
0x00001113:3	304	199	sig_coeff_flag	1	1	0.35	Arithmeti
0x00001113:4	484	399	sig_coeff_flag	1	1	0.35	Arithmeti
0x00001113:5	256	87	sig_coeff_flag	1	1	0.35	Arithmeti
0x00001113:6	394	174	coeff_abs_level_greater1_flag	0	0	0.00	Arithmeti
0x00001113:6	362	174	coeff_abs_level_greater1_flag	0	0	0.00	Arithmeti
0x00001113:6	331	174	coeff_abs_level_greater1_flag	0	0	0.00	Arithmeti
0x00001113:6	262	174	coeff_abs_level_greater1_flag	0	1	0.35	Arithmeti
0x00001113:7	418	348	coeff_abs_level_greater1_flag	1	2	0.71	Arithmeti
0x00001114:1	292	13	coeff_abs_level_greater1_flag	0	1	0.35	Arithmeti
0x00001114:2	362	26	coeff_abs_level_greater1_flag	0	1	0.35	Arithmeti
0x00001114:3	468	52	coeff_abs_level_greater2_flag	0	0	0.00	Arithmeti
0x00001114:3	379	52	coeff_sign_flag	0	1	0.35	Bypass

VQ Analyzer screenshot of bitstream collected from Snapchat.

248. For example, in the H.265 specification, portions of the symbols representing the transform coefficients are represented using Exp-Golomb coding.

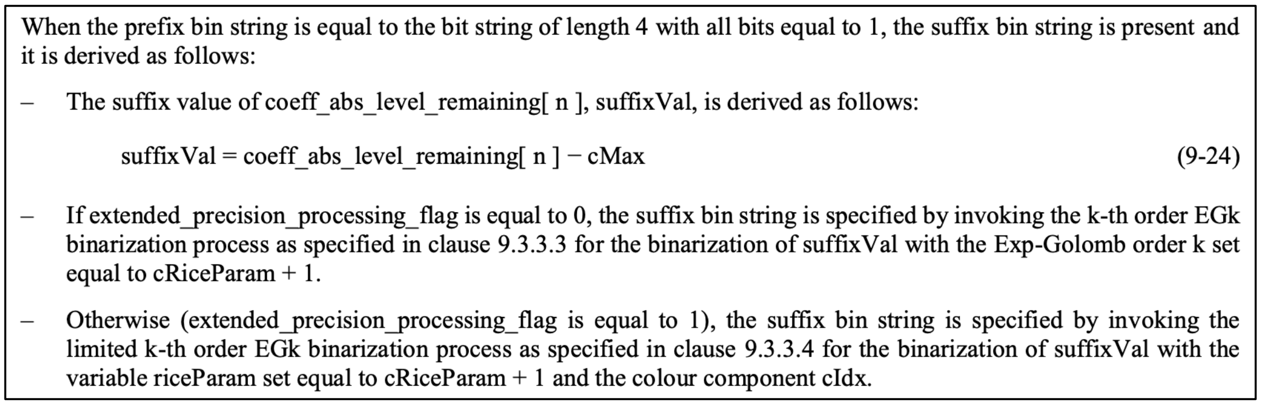


9.3.3.10 Binarization process for coeff_abs_level_remaining[ ]

Input to this process is a request for a binarization for the syntax element coeff_abs_level_remaining[n], the current sub-block scan index i, baseLevel, the colour component cIdx and the luma location (x0, y0) specifying the top-left sample of the current luma transform block relative to the top-left luma sample of the picture.

Output of this process is the binarization of the syntax element.

Ex. 13 (H.265 specification) at 204.



When the prefix bin string is equal to the bit string of length 4 with all bits equal to 1, the suffix bin string is present and it is derived as follows:

– The suffix value of coeff_abs_level_remaining[n], suffixVal, is derived as follows:

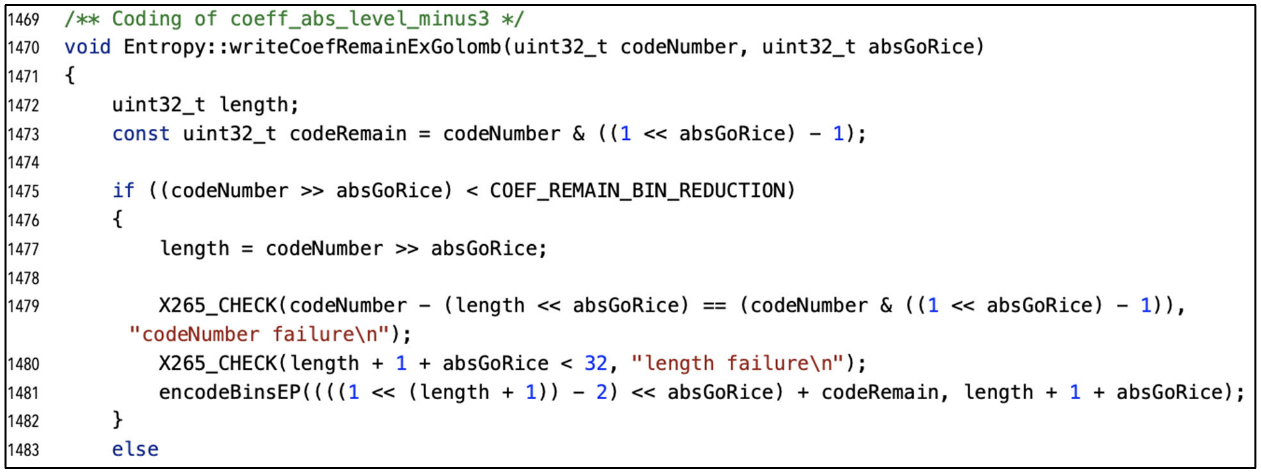
$$\text{suffixVal} = \text{coeff\_abs\_level\_remaining[n]} - \text{cMax} \quad (9\text{-}24)$$

– If extended_precision_processing_flag is equal to 0, the suffix bin string is specified by invoking the k-th order EGk binarization process as specified in clause 9.3.3.3 for the binarization of suffixVal with the Exp-Golomb order k set equal to cRiceParam + 1.

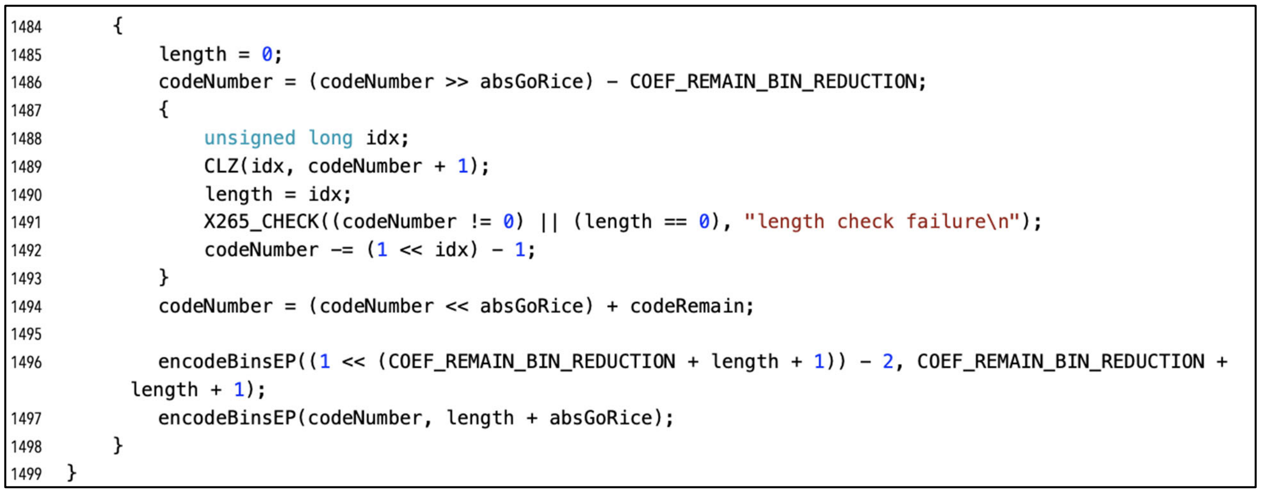
– Otherwise (extended_precision_processing_flag is equal to 1), the suffix bin string is specified by invoking the limited k-th order EGk binarization process as specified in clause 9.3.3.4 for the binarization of suffixVal with the variable riceParam set equal to cRiceParam + 1 and the colour component cIdx.

Ex. 13 (H.265 specification) at 205.

249. For example, the source code for the x265 encoder used by Snap shows the use of Exp-Golomb coding for coeff_abs_level_remaining:



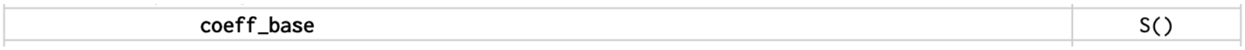
```
/** Coding of coeff_abs_level_minus3 */
void Entropy::writeCoefRemainExGolomb(uint32_t codeNumber, uint32_t absGoRice)
{
    uint32_t length;
    const uint32_t codeRemain = codeNumber & ((1 << absGoRice) - 1);

    if ((codeNumber >> absGoRice) < COEF_REMAIN_BIN_REDUCTION)
    {
        length = codeNumber >> absGoRice;

        X265_CHECK(codeNumber - (length << absGoRice) == (codeNumber & ((1 << absGoRice) - 1)),
      "codeNumber failure\n");
        X265_CHECK(length + 1 + absGoRice < 32, "length failure\n");
        encodeBinsEP((((1 << (length + 1)) - 2) << absGoRice) + codeRemain, length + 1 + absGoRice);
    }
    else
```



```
        {
            length = 0;
            codeNumber = (codeNumber >> absGoRice) - COEF_REMAIN_BIN_REDUCTION;
            {
                unsigned long idx;
                CLZ(idx, codeNumber + 1);
                length = idx;
                X265_CHECK((codeNumber != 0) || (length == 0), "length check failure\n");
                codeNumber -= (1 << idx) - 1;
            }
            codeNumber = (codeNumber << absGoRice) + codeRemain;

            encodeBinsEP((1 << (COEF_REMAIN_BIN_REDUCTION + length + 1)) - 2, COEF_REMAIN_BIN_REDUCTION + length + 1);
            encodeBinsEP(codeNumber, length + absGoRice);
        }
    }
```

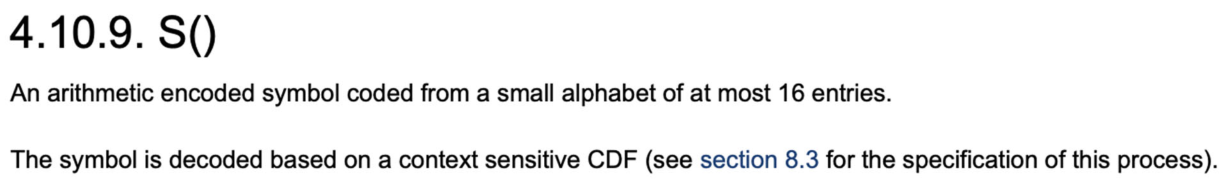
Ex. 16 (entropy.cpp) at 27-28.

250. Similarly, the AV1 specification uses arithmetic coding for certain components, including base levels:



coeff_base	S()

Ex. 15 (AV1 Spec) at 91.



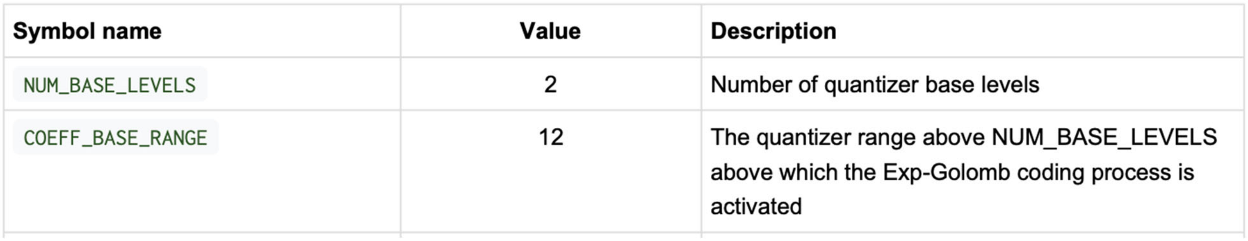
4.10.9. S()

An arithmetic encoded symbol coded from a small alphabet of at most 16 entries.

The symbol is decoded based on a context sensitive CDF (see section 8.3 for the specification of this process).

Ex. 15 (AV1 Spec) at 26.

251. In the AV1 specification, portions of the symbols representing the transform coefficients are also represented using Exp-Golomb coding:



Symbol name	Value	Description
NUM_BASE_LEVELS	2	Number of quantizer base levels
COEFF_BASE_RANGE	12	The quantizer range above NUM_BASE_LEVELS above which the Exp-Golomb coding process is activated

Ex. 15 (AV1 Spec) at 15.

if (Quant[pos] >	
(NUM_BASE_LEVELS + COEFF_BASE_RANGE)) {	
length = 0	
do {	
length++	
golomb_length_bit	L(1)
} while (!golomb_length_bit)	

x = 1	
for (i = length - 2; i >= 0; i--) {	
golomb_data_bit	L(1)
x = (x << 1) | golomb_data_bit	
}	
Quant[pos] = x + COEFF_BASE_RANGE + NUM_BASE_LEVELS	
}	

Ex. 15 (AV1 Spec) at 91-92.

<u>COUNT V: DECLARATORY JUDGMENT THAT DOLBY HAS COMPLIED WITH ITS RAND COMMITMENTS</u>

252. Dolby incorporates by reference the preceding paragraphs as though fully set forth herein.

253. Snap makes and uses products that use and comply with the H.265 Recommendations. Snap requires a license to Dolby’s essential patent claims.

254. Dolby has voluntarily declared to ITU it is prepared to grant licenses to its essential patent claims for implementations of the H.265 Recommendation on a worldwide, non-discriminatory basis and on reasonable terms and conditions, and has at all times been willing to grant such a license to Snap.

255. Dolby has negotiated in good faith with Snap, including for example, through negotiations between Snap and Access Advance.

256. Snap is familiar with patent pools and in fact has previously taken a license to H.265 standard essential patents through Access Advance for certain Snap hardware products sold to customers.

257. However, for its services related to the Snapchat app, Snap has given no indication that it is willing to license Dolby’s essential patent claims on RAND terms and has not accepted the offer referenced above.

258. On information and belief, Snap contends that claims of the patents-in-suit are essential to ITU standards and that Dolby's actions, including the offer previously made to Snap, do not satisfy Dolby's RAND obligations. Dolby, on the other hand, contends that it has complied with any RAND obligations for its essential claims by offering a license under RAND terms. There is a case or controversy of sufficient immediacy, reality, and ripeness to warrant the issuance of a declaratory judgment.

259. Dolby seeks a declaration that it has negotiated in good faith toward a license with Snap for its essential patent claims and complied with any obligations it has under the ITU IPR Policy and Dolby's relevant Patent Statement and Licensing declarations.

ATTORNEYS' FEES

260. Dolby is entitled to recover reasonable and necessary attorneys' fees under applicable law.

DEMAND FOR JURY TRIAL

261. Dolby hereby demands a jury trial for all issues so triable.

PRAYER FOR RELIEF

WHEREFORE, Dolby respectfully requests that this Court enter judgment in its favor as follows and afford Dolby the following relief:

A. Adjudge and declare that Snap infringes claims of the Asserted Patents;

B. Adjudge and declare that Snap's infringement of claims of the Asserted Patents was willful, and that Snap's continued infringement is willful;

C. Award Dolby its actual damages going back six years from the date of the Original Complaint due to Snap's infringement and the fact that the asserted claims do not require marking and/or there is nothing to mark;

D. Award Dolby enhanced damages pursuant to 35 U.S.C. § 284;

E. Award Dolby pre-judgment and post-judgment interest to the full extent allowed under the law, as well as its costs;

F. Adjudge and declare that this is an exceptional case and award Dolby its reasonable attorneys' fees pursuant to 35 U.S.C. § 285;

G. Order an accounting of damages for acts of infringement;

H. Adjudge and declare that to the extent Dolby's patent claims are subject to a RAND obligation, that Dolby has negotiated in good faith and satisfied its RAND obligations;

I. Enjoin Snap from committing further acts of infringement under 35 U.S.C. §271 of any of the claims of the '272 Patent through encoding or decoding of AV1-compliant video formats pursuant to 35 U.S.C. §283.

J. Award Dolby its costs of suit; and

K. Award such other relief, including equitable relief, that the Court deems appropriate.

Dated: March 23, 2026

Respectfully submitted,

FARNAN LLP

/s/ Michael J. Farnan

Brian E. Farnan (Bar No. 4089)
Michael J. Farnan (Bar No. 5165)
919 North Market Street, 12th Floor
Wilmington, DE 19801
Phone: (302) 777-0300
bfarnan@farnanlaw.com
mfarnan@farnanlaw.com

Attorneys for Plaintiff

Of Counsel:

McKool Smith P.C.

Kevin Schubert
kschubert@mckoolsmith.com
Robert Burns
rburns@mckoolsmith.com
McKool Smith P.C.
1301 6th Ave, 32nd Floor
New York, NY 10019

Warren Lipschitz
wlipschitz@mckoolsmith.com
Greg Saltz
gsaltz@mckoolsmith.com
McKool Smith P.C.
300 Crescent Court
Dallas, TX 75201

Mitch Verboncoeur
mverboncoeur@mckoolsmith.com
McKool Smith P.C.
303 Colorado Street, Suite 2100
Austin, TX 78701

Christopher McNett
cmcnett@mckoolsmith.com
McKool Smith P.C.
1717 K Street, NW, Suite 1000
Washington, DC 20006