

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

SNAP INC. and HISENSE USA CORPORATION,

Petitioner

v.

NOKIA TECHNOLOGIES OY,

Patent Owner

Case: IPR2025-01345

U.S. Patent No. 9,036,701

PATENT OWNER'S PRELIMINARY RESPONSE

Mail Stop "PATENT BOARD"

Patent Trial and Appeal Board

U.S. Patent and Trademark Office

P.O. Box 1450

Alexandria, VA 22313-1450

CONTENTS

I. INTRODUCTION	5
II. BACKGROUND	5
A. THE '701 PATENT	5
B. THE ASSERTED PRIOR ART	12
1. Murashita (Ex-1007)	12
2. Marpe (Ex-1006)	15
3. Yu (Ex-1018).....	16
C. POSITA	18
D. CLAIM CONSTRUCTION	18
E. CLAIM LISTING	19
III. ARGUMENT.....	19
A. PETITIONER HAS NOT ESTABLISHED A REASONABLE LIKELIHOOD THAT THE CHALLENGED CLAIMS ARE UNPATENTABLE IN LIGHT OF GROUND 1- MURASHITA	19
1. Petitioner fails to demonstrate that limitations 1[a], 8[a], and 15[a] are obvious over Murashita	19
B. PETITIONER HAS NOT ESTABLISHED THAT THE CHALLENGED CLAIMS ARE UNPATENTABLE IN LIGHT OF GROUND 2—MURASHITA IN COMBINATION WITH MARPE.	28
1. Petitioner Fails to Demonstrate that Limitations 1[a], 8[a], and 15[a] are Obvious Over Murashita	28
2. Marpe Does Not Categorize Syntax Elements Based on “Frequency of Occurrence”	29
3. Murashita in view of Marpe Cannot Be Combined Because They Would Produce an Unworkable Result	31
4. No Reasonable Expectation of Success in Combining Murashita with Marpe.....	34
C. PETITIONER HAS NOT ESTABLISHED THAT THE CHALLENGED CLAIMS ARE UNPATENTABLE IN LIGHT OF GROUND 3—YU	35
1. Yu’s Syntax Element Categories Do Not Determine Whether They Are Context-coded or Bypass-coded.....	35

2. Yu’s Syntax Element Categories are Unrelated to the Comparison of <i>Offset</i> to an Alleged Threshold.....	36
3. Yu’s Use of Bypass Is Not Related to the “Frequency of Occurrence of Syntax Elements”	36
4. Yu’s Probability Monitoring Is Not “Expected Frequency of Occurrence of Syntax Elements”	38
5. Petitioner’s Reliance on Yu’s Threshold is Misplaced	41
6. As to Dependent Claims 6, 13, and 20, Yu’s Threshold is Not “Predefined”	43
IV. CONCLUSION	44

TABLE OF AUTHORITIES

	Page(s)
STATUTES	
35 U.S.C. § 103	21

TABLE OF EXHIBITS

Ex. No.	Title	Previously Submitted
2001	Order No. 23-Granting Joint Unopposed Motion by the Private Parties to Modify the Procedural Schedule to Extend Fact and Expert Discovery in Investigation No. 337-TA-1448, dated September 30, 2025	X
2002	Investigation Initiation Letter to The Honorable Lisa R. Barton in Investigation No. 337-TA-1448, dated April 11, 2025	X
2003	Respondents Initial Invalidity Contentions in Investigation No. 337-TA-1448, dated July 1, 2025	X
2004	Exhibit A-10 Claim Chart Regarding U. S. Patent No. 9,036,701, in Investigation No. 337-TA-1448	X
2005	Exhibit A-5 Claim Chart Regarding U. S. Patent No. 9,036,701, in Investigation No. 337-TA-1448	X
2006	Exhibit A-9 Claim Chart Regarding U. S. Patent No. 9,036,701, in Investigation No. 337-TA-1448	X
2007	Original Complaint in <i>Nokia Technologies Oy v. Hisense Co. Ltd. et al.</i> , Case No. 1:25-cv-01871-TWT (N.D. Cal.), dated April 7, 2025	X
2008	USPTO Proposed Revision to Rules of Practice before the Patent Trial and Appeal Board, 37 CFR Part 42, RIN 0651-AD89, Docket No. PTO-P-2025-0025, dated Oct. 17, 2025	X
2009	Declaration of Dr. Iain Richardson	
2010	CV of Dr. Iain Richardson	

I. INTRODUCTION

Patent Owner Nokia Technologies Oy (“Nokia”) sets forth the substantive errors, omissions, and shortcomings underlying the alleged grounds for invalidity argued in the Petition for *Inter Partes* Review of U.S. Patent No. 9,036,701 (the “’701 Patent”) (“Petition” or “Pet.”). Both Petitioner’s Grounds are flawed. Institution should be denied.

As to Grounds 1 to 3, Petitioner fails to show that Murashita, Marpe, and/or Yu disclose the ’701 claims as obvious. In support of this Patent Owner’s Response, Nokia provides a declaration from its expert, Dr. Iain Richardson (Ex-2009). Dr. Richardson’s CV is attached as Ex-2010.

Nokia respectfully requests that the Board reject Petitioner’s arguments and find that the ’701 Patent’s challenged claims not unpatentable under any of Petitioner’s Grounds. Because none of the grounds establish a reasonable likelihood that the challenged claims are unpatentable, institution should be denied.

II. BACKGROUND

A. The ’701 Patent

U.S. Patent No. 9,036,701 (the “’701 Patent”), titled “Method and Apparatus for Providing Complexity Balanced Entropy Coding” issued to inventors Jani Lainema, Kemal Ugur, and Antti Hallapuro on May 19, 2015. The ’701 Patent claims priority to U.S. Provisional Application No. 61/368,316, which was filed on July 28, 2010.

As Dr. Richardson explains, the '701 Patent is directed to a method and apparatus for “complexity balanced entropy coding ... in order to provide a high level of capability with respect to video coding and decoding while keeping the costs in terms of complexity relatively low.” ’701 Patent at 1:54-60. The H.264 video coding standard developed prior to the '701 Patent employed two distinct coding engines, 1) “a low complexity Context Adaptive Variable Length Coding based approach (CAVLC)” and 2) “the Context Adaptive Binary Arithmetic Coding approach (CABAC).” ’701 Patent at 4:23-42. While CABAC provided improved efficiency in entropy coding, the '701 Patent notes that “the computational requirements of the CABAC method increase substantially as a function of the utilized bitrate” and “the CABAC approach may cause problems in real life implementations as the computational requirements of that operation increases.” *Id.* In other words, H.264’s CABAC offered superior compression efficiency over CAVLC, but CAVLC was too computationally complex to be utilized on certain consumer devices. This split between computational complexity led to CAVLC being the default entropy coding engine for low-complexity devices at the cost of increasing bitstream sizes. Ex-2009 at ¶ 44.

During the development of H.265, there was a desire to “achieve a balance between coding efficiency and practicality” through adjusting “the design of transform coefficient coding in HEVC.” One solution implemented into the H.265 Standard was to “[i]ncrease the percentage of bypass bins and group bypass bins together for higher

throughput.” *Transform Coefficient Coding in HEVC*, Sole et al., IEEE Transactions on Circuits and Systems for Video Technology, Vol 22, No. 12, December 2012 at 1766. Ex-2009 at ¶ 45.

The inventions of the ’701 Patent achieve this balance “by categorizing syntax elements differently, different categories of symbols may be operated on differently in order to achieve relatively higher efficiency in a lower complexity implementation.” ’701 Patent at 4:43-56. As one example, this process can use a CABAC coding engine, wherein “if encoding is performed using CABAC of Pipe/V2V related approaches, the corresponding decoding approaches for each may be implemented for the first category symbols. Meanwhile, syntax elements belonging to the second category may employ the same coding engine (e.g., the entropy coding engine 80) as first category symbols except that static fifty percent bin probabilities may be employed in relation to decoding of these symbols (e.g., by employing a bypass coding mode of a CABAC engine).” ’701 Patent at 10:25-36; Ex-2009 at ¶ 46.

According to Dr. Richardson and the ’701 Patent, this is advantageous because “first category encoding may be structured to provide relatively high efficiency while second category encoding may be structured to provide lower complexity (e.g., by bypassing context updating).” ’701 Patent at 10:37-54. While this is beneficial for coding syntax elements, it need not be used for every syntax element within a bitstream, allowing for more flexibility in implementing the inventions of the ’701 Patent. ’701

Patent at 10:55-65 (“the method and apparatus of on[e] example embodiment may encode some of the syntax elements in accordance with complexity balanced entropy coding and others of the syntax elements in accordance with other coding schemes, such as in accordance with arithmetic coding schemes.”). The ’701 Patent explains that “the complexity balanced entropy coding of an example embodiment of the present invention may be utilized jointly or in combination with arithmetic or other coding schemes.” *Id.*; Ex-2009 at ¶ 47.

The ’701 Patent discloses that the inventions of the ’701 Patent may also be used in both encoding and decoding scenarios. ’701 Patent at 10:66-11:4 (“an instance of the categorizer 82 and the entropy coding engine 80 may be provided at the video encoder/decoder 48.”). Dr. Richardson explains that this means “the categorization may be performed at both the encoder and decoder to align with the entropy coding mechanisms employed.” Ex-2009 at ¶ 48. Such categorization can also take place at the “block, slice, picture, or sequence levels.” ’701 Patent at 11:41-61.

One embodiment described in the ’701 Patent’s specification focuses on the categorization of Discrete Cosine Transform (DCT) syntax elements:

“As indicated in FIG. 4, by employing an example embodiment that categorizes syntax elements and then encodes each category differently to reduce complexity, the computational complexity as indicated by the number of DCT coefficients, is kept below the desired maximum level

by utilizing context adaptive arithmetic coding while bypassing probability estimation for DCT coefficient data for some symbols.”

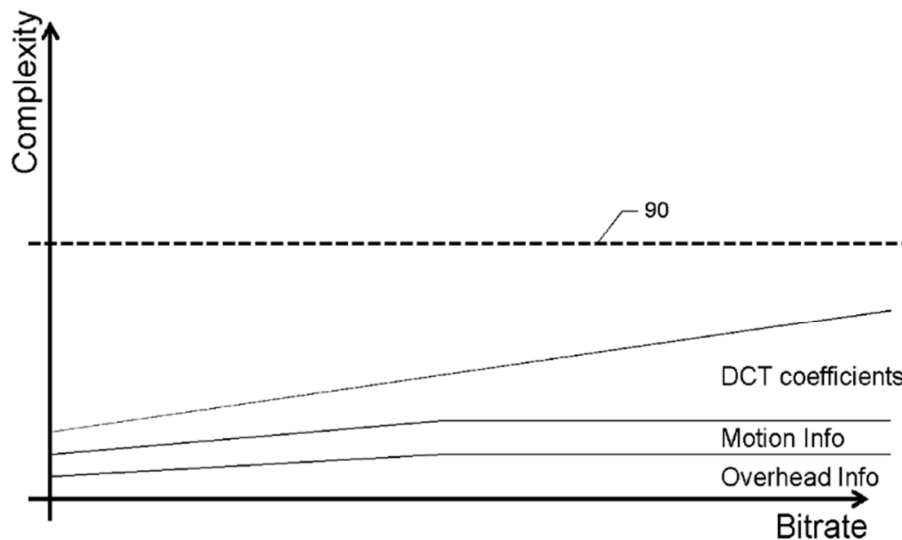


FIG. 4.

'701 Patent at 11:23-40; Fig. 4. This means that the frequency of occurrence of DCT coefficient syntax elements increases as more are coded, causing complexity challenges as detail and motion information of a video raises. This in turn leads to a rise in the number of DCT coefficients, which “may cause problems in real life implementations”, '701 Patent at 4:37-42. By considering the frequency of occurrence of syntax elements, the '701 Patent provides a solution to balance the increase in complexity with compression efficiency. The '701 Patent teaches to adjust entropy coding modes according to the frequency of occurrence which allows for improved management of

the computational complexity required to encode and decode data such as DCT coefficients. '701 Patent at 11:23-40, Fig. 4; Ex-2009 at ¶ 49.

The '701 Patent further illustrates an embodiment with a flowchart “of a method and program product according to an example embodiment of the invention” noting that “each block of the flowchart, and combinations of blocks in the flowchart, may be implemented by various means, such as hardware, firmware, processor, circuitry and/or other device associated with execution of software including one or more computer program instructions.” '701 Patent at 12:4-34, Fig. 5; Ex-2009 at ¶ 50.

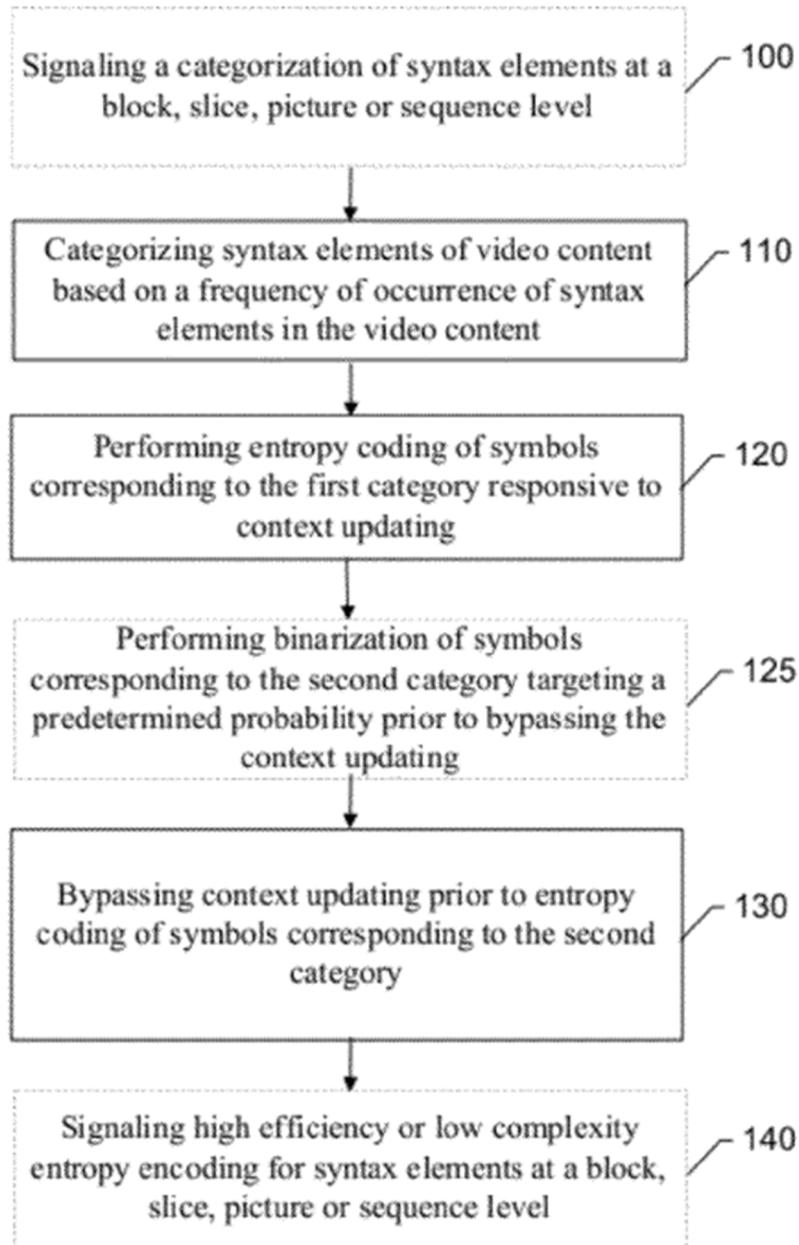


FIG. 5.

In describing Figure 5, the '701 Patent states “may include categorizing syntax elements of video content based on a frequency of occurrence of syntax elements in the video content at operation 110. The categorizing may include defining at least a first category and a second category of symbols” and “may further include performing entropy coding of symbols corresponding to the first category responsive to context updating at operation 120, and bypassing context updating prior to entropy coding of symbols corresponding to the second category at operation 130 ... additional optional operations may also be included (examples of which are shown in dashed lines in FIG. 5).” ’701 Patent at 12:35-13:15; Ex-2009 at ¶ 51.

B. The Asserted Prior Art

1. Murashita (Ex-1007)

U.S. Patent No. 5,650,783 (“Murashita”), titled “Data Coding/Decoding Device and Method” was filed on Sep. 8, 1995. As Dr. Richardson explains, “Murashita relates to coding of what it calls ‘symbols’ based on a multi-level arithmetic coding scheme. Petition at 10. A POSITA would be aware that multi-level arithmetic coding is distinct and different from binary arithmetic coding.” Ex-2009 at ¶ 54.

Dr. Richardson explains how a POSITA would understand the disclosures of Murashita and specifically, Murashita’s “symbols.” Ex-2009 at ¶ 55 (“Murashita explains that what it calls “symbols” are “events of characters,” which are symbol values. Ex-1007 at 1:63-2:4. Murashita’s Figure 1a further confirms this, showing the

frequency (probability) of certain symbol values a, b, c, d and e occurring and equating each to a fractional range. Murashita provides further examples that confirm this, such as the example of encoding a character string “abe” (Ex 1007 at 2:5-20) where each of the symbol values “a”, then “b”, then “e” are described as “symbols”. *See also* Ex 1007 at 3:11-24.”

The disclosures of Murashita relate to the process of updating a context and considering how many times a given context should be updated. For example, Murashita explains that “controlling unit 13 controls the context holding/updating unit 12 to update the context data. In doing so, the controlling unit compares a predetermined threshold value with a cumulative frequency of occurrence for all the symbols registered under a given context. Then, only when the cumulative frequency is smaller than the threshold value, the controlling unit 13 controls the context holding/updating unit 12 to update the context data for the given context.” Ex-1007 at 6:40-47. As Dr. Richardson explains, Murashita suggests pausing the updating of context data when certain conditions are met for a given symbol. Ex-2009 at ¶ 56. Dr. Richardson shows that the “symbols” referenced in Murashita are the symbol values used in entropy coding, as seen in Murashita’s description of the symbols and Figures 1A and 1C. Ex-2009 at ¶ 56. For example, the symbol values a, b, c, d, and e represent different values of a “character string” and have a distinct frequency and range associated with each. Ex-1007 at 2:5-20, Fig. 1A. Murashita describes the use of a string of symbol values in arithmetic

coding in describing Fig. 1A, as explained by Dr. Richardson. *See* Ex-1007 at 2:5-20 (describing an example of encoding a character string “abe” where each of the symbol values “a”, then “b”, then “e” are described as “symbols”); *See also* Ex-1007 at 3:11-24; Ex-2009 at ¶ 56.

Dr. Richardson explains a POSITA would understand that Each time a symbol value (such as the value of a text character in the character string) registered to a context is coded, that context is updated with the new frequency and may reorder the context accordingly. Ex-1007 at 3:57-4:15; 7:24-39; Ex-2009 at ¶ 57. This process is repeated for each symbol registered to the context, tracking the cumulative frequency of occurrence of all symbol values within the context up to the symbol being coded. When Murashita detects the cumulative frequency exceeds a threshold, the process is not performed. Ex-1007 11:25-12:12; Ex-2009 at ¶ 57.

In this way, Murashita’s process looks to the number of times a context tree has been updated and halts context updating once a certain number of symbol values have been coded. As Dr. Richardson points out, Murashita does not discuss or even mention syntax elements Further, Murashita does not disclose or discuss a frequency of occurrence of syntax elements. or bypassing context updating as the word “bypass” does not appear in Murashita. Ex-2009 at ¶ 58.

2. Marpe (Ex-1006)

U.S. Patent Application Publication No. 2005/0253740 (“Marpe”), titled “Method and Apparatus for Binarization and Arithmetic Coding of a Data Value” was filed on May 3, 2005. As Petitioner points out, Marpe’s approach is “based on binarization and binary arithmetic entropy coding”. Dr. Richardson explains that Binary arithmetic coding (BAC) is a distinct and different coding approach from multi-level arithmetic coding. Ex-2009 at ¶ 59. Marpe discusses the use of bins related to syntax elements or symbol values. Dr. Richardson explains that “A bin is a binary value of 0 or 1 that represents the smallest addressable unit in entropy coding. A series of bins (a bin string) can represent a single, non-binary symbol value, one or more of which may be used to represent a syntax element in the bitstream.” Ex-2009 at ¶ 59. A POSITA would understand that “Marpe’s disclosure of ‘the most frequently observed bins’ refers to bins that correspond to smaller absolute values, which would typically occur first in the order of binarization of a non-binary value.” Ex-2009 at ¶ 59. This leads to the distinction that Marpe’s frequency of observation of bins is actually related to the frequency of occurrence of specific syntax element values, not the frequency of occurrence of syntax elements as required by ’701 Patent. This is clear from Marpe’s figure 5, which provides an illustration of binarizations of a syntax element with a range of possible absolute values (‘Abs. value’), or multiple distinct values for a single syntax element. In this example, Bin 1 is included in every bin string representing an abs. value,

while bin 2 is included for bin strings representing an abs. value greater than 1. This pattern continues for further abs. values. Dr. Richardson explains that a POSITA would understand “bin 1 is the most frequently observed, followed by bin 2 and so on.” Ex-2009 at ¶ 59. Marpe’s bin strings are described with the assumption that small absolute values are more frequently observed than large absolute values, meaning the frequency of observation of a particular bin (such as bin 2) relates to the expected frequency of occurrence of syntax element values. Ex-2009 at ¶ 59.

3. Yu (Ex-1018)

“A High Performance CABAC Decoding Architecture,” authored by Wie Yu and Yun He, was published in the November 2005 Issue of IEEE Transactions on Consumer Electronics. Yu discloses an approach for implementing CABAC decoding in H.264/AVC systems with a redesigned hardware implementation. As Dr. Richardson explains, the H.264 CABAC engine was a known prior art method and is referenced in the specification of the ’701 Patent. See ’701 Patent at 4:23-42; Ex-2009 at ¶ 60. Yu does not disclose any changes to H.264’s CABAC process, and instead focuses on implementing it in a different hardware system that Yu describes as efficient. Dr. Richardson points out that Yu notes “that CABAC significantly increases computational complexity compared with H.264’s CAVLC entropy coding technique, because H.264 CABAC encodes video information as a sequence of binary symbols—called “bins”—each representing a 0 or 1, and an H.264 decoder must maintain and continually update

399 probability models that estimate whether each upcoming bin is more likely to be a 0 or a 1.” Ex-2009 at ¶ 60 (citing Ex-1018 at 1352–53). Yu attributes these complexities to two primary operations in H.264 CABAC: (1) the arithmetic decoding of these binary symbols, and (2) the loading, updating, and writing back of the associated context (probability) models for each symbol. *Id.* According to Yu, when these operations are performed in software, “about 4500 cycles are needed” for decoding the binary symbols of a single macroblock. *Id.* Yu proposes a hardware structure to address these fallbacks. Yu’s hardware structure is organized around two decoding trees—one for regular binary symbols and one for bypass-mode symbols. *Id.* Yu’s hardware attempts to decode two regular binary symbols (“bins”) per cycle and, for specific syntax elements attempts to decode a bypass sign bit “within the same cycle of the last regular bin of the prefix.” *Id.* at 1355; Ex-2009 at ¶ 60.

Yu explains that in binary arithmetic decoding, the decoder maintains a range that is divided into two sub-intervals based on the probability estimation for the least probable symbol (LPS) and most probable symbol (MPS). *Id.* at 1353. The decoder uses these intervals by comparing an incoming offset value to the subintervals (rLPS, rMPS). The result of this comparison informs the decoder as to whether the next symbol (i.e. the next decoded bin) is the most probable symbol or the least probable symbol, updating the range and offset registers accordingly. *Id.* As Dr. Richardson explains, “Yu divides H.264 CABAC’s 399 probability models into 18 groups that are loaded from

memory and written back group-by-group rather than on a per-symbol basis.” Ex-2009 at ¶ 61 (citing Ex-1018. at 1356, 1358).

C. POSITA

Petitioners state a POSITA “would have had a (1) Bachelor’s degree in electrical engineering, computer engineering, computer science, or a comparable field of study such as physics, and (2) approximately two to three years of practical experience with video encoding/decoding . . . [and] [m]ore education could substitute for less experience, and vice-versa.” Patent Owner uses this same definition only for purposes of this preliminary response. Asserted Grounds

Petitioner asserts the following grounds, none of which have a reasonable likelihood of success:

Ground		Claims	Prior Art
1	§ 103	1, 6, 8, 13, 15, 20	Murashita
2	§ 103	1-20	Murashita, Marpe
3	§ 103	1, 3, 6-8, 10, 13-15, 17, 20	Yu

D. Claim Construction

The Petition states claim terms are construed based on the plain and ordinary meaning as a POSITA would have understood it at the time of the invention has been applied in accordance with the principles. Petition at 10. Patent Owner adopted the same approach for purposes of this preliminary response.

E. Claim Listing

Patent Owner will use Petitioner's limitation labels provided in the Petition for purposes of this preliminary response.

III. ARGUMENT

A. Petitioner Has Not Established a Reasonable Likelihood That The Challenged Claims Are Unpatentable in Light of Ground 1-Murashita

1. Petitioner fails to demonstrate that limitations 1[a], 8[a], and 15[a] are obvious over Murashita

Petitioner has failed to show that Murashita teaches or suggests this limitation. Limitations 1[a], 8[a], and 15[a] each, in substance, require “determining a frequency of occurrence threshold based on an expected frequency of occurrence of syntax elements in a bit stream.”

Petitioner equates “data symbols” with the claimed “syntax elements.” According to Petitioner, syntax elements are simply elements of data represented in the bitstream and therefore constitute data symbols. Petition at 13.

a. Murashita's Data Symbols Are Not the Claimed “Syntax Elements”

Murashita does not teach limitation 1[a]. Murashita contains no teaching, discussion, or suggestion of the “frequency of occurrence of syntax elements” required by the challenged claims. As Dr. Richardson explains, Murashita is directed to an adaptive arithmetic coder that tracks only the frequency of occurrence of symbols—a different concept. Ex-2009 at ¶ 66. Each of the symbols disclosed in Murashita is a

symbol value (i.e. one of many possibilities a symbol may take). *See supra* § II.B.2. For example, Murashita explains that “frequency of occurrence is counted for each symbol” (Ex-1007 at 3:21-24) and repeatedly emphasizes that the system maintains a dictionary in which “frequency and cumulative frequency are assigned to each symbol” (Ex-1007 at 3:66-67). Notably, Murashita describes a device that “codes the symbols based on frequency of occurrence of the symbols,” not syntax elements. *Id.* at Abstract. Murashita also recites “[i]n the dictionary of FIG. 4A, frequency and cumulative frequency ***are assigned to each symbol***. Symbols are arranged from the top of the list in descending order of frequency. The cumulative frequency ***of a given symbol*** is a sum of the frequency from the rarest symbol at the bottom of the list to the given symbol, and is used when ranges are divided.” Ex-1007, 3:66-4:5, *emphasis added*. These disclosures make clear that Murashita is exclusively concerned with the frequencies of given symbols—not the frequency of occurrence of syntax elements as required by the challenged claims.

According to Dr. Richardson, Murashita “does not disclose or suggest syntax elements or any mapping from syntax elements to symbols.” Ex-2009 ¶ 67. Murashita instead defines its update procedure entirely in terms of symbol occurrences and symbol statistics. For example, Murashita explains that when a symbol is processed, the system “increments the frequency of the symbol K by 1” and “increments the cumulative frequency of all symbols listed above the symbol K.” Ex-1007 at 8:24–28. Murashita

elsewhere reiterates that its encoder counts the frequency of occurrence for each symbol (Ex-1007 at 5:57–59), maintains a dictionary including symbols, frequency, cumulative frequency (Ex-1007 at 3:61–66), and updates the frequency of the symbol that appears after each coding operation (Ex-1007 at 4:6–24).

These disclosures confirm that Murashita is focused exclusively on symbol-level frequency tracking—i.e., counting how often particular symbol values appear and updating a symbol dictionary accordingly. Nothing in Murashita teaches or implies any frequency analysis of syntax elements, any association between syntax elements and symbols, or any measurement or use of “frequency of occurrence of syntax elements” as required by the challenged claims. Ex-2009 ¶ 67. A POSITA would understand that a syntax element is an element of data represented in the bitstream, whereas a symbol is an element used to represent syntax elements—such as a value, a decision, or a bin. This distinction is confirmed by Petitioner’s own prior art. Marpe expressly states that “[e]ach syntax element is a data value having a certain meaning in the video signal bit stream that is passed to the entropy coder.” Ex-1006 ¶38. Likewise, Yu explains that “the syntax element is an element of data represented in the bit stream to be encoded,” and that syntax elements are mapped to bin strings, where “each bin of the bin string is arithmetically encoded into the bit stream.” Ex-1018 at 1353. Yu further clarifies that a “bin is a binary value.” *Id.* at 1352. Thus, according to Dr. Richardson, syntax elements are meaningful constructs of the bitstream, while symbols (including bins) are the unit

of information passed to or from the entropy coder. Ex-2009 at ¶ 68. The occurrence rate of specific symbol values (e.g., the number of times a character equals “a,” “b,” or “e”) is not equivalent to the occurrence rate of a syntax element; the former concerns symbol values, while the latter concerns the appearance of the syntax element itself. Thus, Murashita addresses only symbol values—not syntax elements. Ex-2009 at ¶ 68. According to Dr. Richardson, in video coding, a single syntax element may be mapped into multiple symbols, or a symbol may represent only a portion of a syntax element. Ex-2009 at ¶ 69. Thus, although syntax elements may ultimately be encoded as symbols, the concepts remain distinct; a syntax element does not transform into a “symbol” simply because it is entropy-coded. Ex-2009 at ¶ 69.

Accordingly, Murashita does not render independent claims 1, 8, or 15 obvious because Murashita operates exclusively on symbol values, not syntax elements, and does not operate on or even consider the frequency of occurrence of syntax elements. Ex-2009 at ¶ 70. Because dependent claims 6, 13, and 20 incorporate the limitations of their respective independent claims, Murashita likewise cannot render those dependent claims obvious.

b. Murashita Does Not Teach The ‘701 Patent’s Frequency-Based Categorization of Syntax Elements and Its Dual-Mode Entropy-Coding Behavior

The '701 Patent recites that a coding apparatus “determin[es] a frequency of occurrence threshold” and “categorize[s] a plurality of syntax elements ... based on the frequency of occurrence of the syntax elements.” ‘701 Patent at 14:5–16. The ‘701 Patent then uses this categorization to select between the use of two distinct entropy coding behaviors: (1) context-updated entropy coding symbols corresponding to the first category of syntax elements and (2) bypass-mode entropy coding—using a fixed 50% probability—for symbols corresponding to the second category of syntax elements. *Id.* at 10:7–39. However, Murashita neither discloses categorizing syntax elements nor applying different entropy-coding schemes based on such categories. As Dr. Richardson explains, the '701 Patent fundamentally departs from Murashita because it expressly requires categorizing syntax elements according to their frequency of occurrence and then selecting different entropy-coding modes based on the assigned category—concepts that Murashita does not teach or even suggest. Ex-2009 ¶¶ 71, 72. Notably, Murashita does not compute or use any frequency-of-occurrence statistic for syntax elements, does not categorize syntax elements, and does not provide any dual-mode entropy-coding framework. Ex-2009 at ¶ 72.

Murashita is concerned solely with updating and reordering symbol-value frequency tables used in multi-level arithmetic coding, and it contains no disclosure of syntax elements or any category-specific entropy-coding operations. *Id.* For example, Murashita explains: “[a]t a step S101, **a symbol *K*** is coded by the coding unit 102. At a

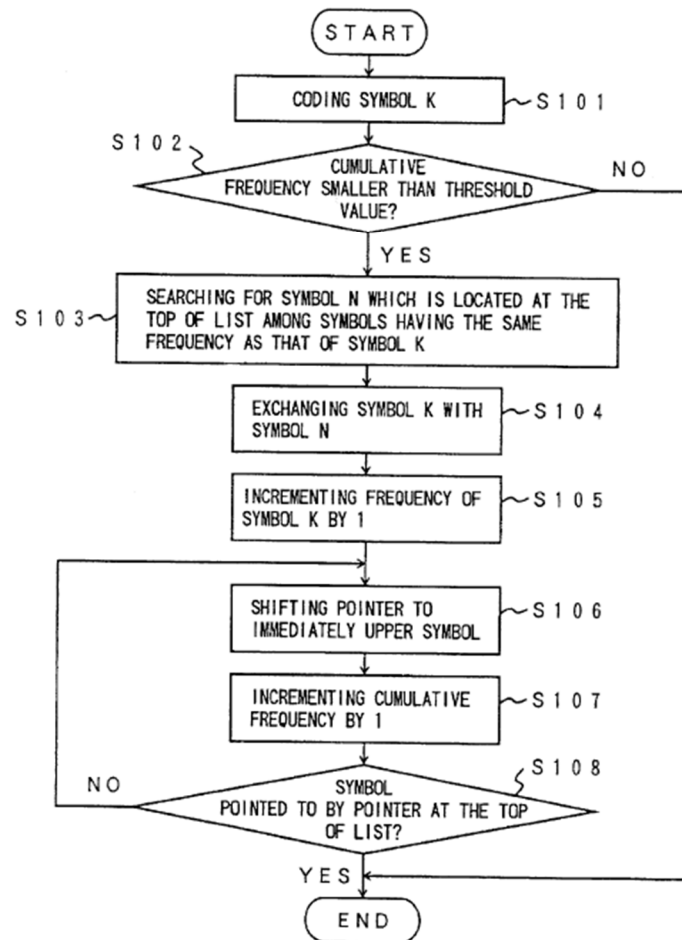
step S102, a check is made ... whether the cumulative frequency is less than the threshold value.... At a step S103, the sorting unit 105 *searches for a symbol N* which is located at the top of the list among symbols having the same frequency as that of the symbol K. At a step S104, *the symbol K is exchanged with the symbol N*. At a step S105, the frequency updating unit 107 *increments the frequency of the symbol K* by 1.” Ex-1007 at 8:34–39, 8:49–54, *emphasis added*. These operations all relate to symbol frequencies and symbol ordering within a generic arithmetic-coding dictionary. The ’701 Patent’s central concepts—categorizing syntax elements based on their frequency of occurrence and then applying context-based or bypass entropy-coding modes depending on that category—are entirely absent from Murashita. Ex-2009 ¶ 72.

c. Murashita Does Not Teach Bypassed Context Updating

The Petition contends that Murashita’s cessation of its context-update procedure is equivalent to “bypassed context updating.” Petition at 43-44. That is not the case. Murashita stores “context data” indicating frequency of occurrence, cumulative frequency of occurrence and an order. Ex-2009 at ¶ 73; Ex-1007 at 6:32-35. Further, as shown in Murashita’s Figure 10, there are only two operational possibilities when a symbol is received: either (1) continue performing the update operations on the symbol-frequency data or (2) discontinue further updates when a threshold is reached. Murashita’s “discontinue updating” branch merely halts additional updates to its

symbol-frequency tables; it does not trigger a second, alternative coding mode. Ex-2009 at ¶ 73.

FIG. 10



In contrast, independent claims 1, 8, and 15 of the '701 Patent expressly require a dual-mode entropy-coding framework: (1) symbols that are coded *after* undergoing context updating, and (2) symbols that are coded by *bypassing* context updating. Simply stopping further updates does not constitute “bypassing context updating,” because the symbols in Murashita have already been subjected to context updating in

earlier cycles. As Dr. Richardson explains, Murashita’s single-mode arithmetic coder always relies on stored, updated context data—meaning it always codes symbols that have undergone a context update. Ex-2009 ¶ 74. Murashita provides no disclosure of any “bypass mode”—that is, a mode in which symbols are coded without context data or with context updating deliberately avoided. Because such a mode is entirely absent, Murashita cannot fulfill the claims’ dual-mode entropy-coding requirements, either individually or in combination. Ex-2009 ¶ 74.

d. Murashita Does Not Disclose Categorizing Syntax Elements based on the Frequency of Occurrence of Syntax Elements

As discussed above, Murashita evaluates characteristics of symbol values rather than syntax elements. *See Supra* § II.B.2. But even setting that deficiency aside, Murashita does not disclose any categorization based on frequency of occurrence—whether of syntax elements or of symbol values. Ex-2009 at ¶ 75.

Petitioner mischaracterizes Murashita in asserting that cumulative frequency “represents the expected individual frequency of occurrence of a symbol.” Petition at 28. Murashita’s Figure 4A (below) demonstrates the opposite: it shows exemplary frequency and cumulative frequency values used in a symbol dictionary for arithmetic-coding range calculations, not any expected-frequency estimation.

FIG. 4A

ORDER OF FREQUENCY	SYMBOL	FREQUENCY	CUMULATIVE FREQUENCY
0	-----	0 0	3 2 7
1	0 0	1 0	3 1 7
2	0 1	7	3 1 0
3	0 2	4	3 0 6
⋮	⋮	⋮	⋮
n - 1	'M'	2	m 2
n	'N'	1	m 1
⋮	⋮	⋮	⋮
X	'A'	1	m 0
⋮	⋮	⋮	⋮
2 5 4	F D	1	3
2 5 5	F E	1	2
2 5 6	F F	1	1
2 5 7	E O F	1	0

Ex-1007, Figure 4A

Dr. Richardson explains that in this example, symbol 01 has a frequency of occurrence of 7 and a cumulative frequency of 310. Further, symbols FD and FF have identical frequencies of occurrence (1) but different cumulative frequencies (3 and 1). The cumulative frequency provides no information as to the “individual frequency of occurrence” of a symbol. Ex-2009 at ¶ 77. Further, symbol 01 has a frequency of occurrence seven times larger than the frequency of occurrence of symbol FF and a cumulative frequency of occurrence 310 times larger than the cumulative frequency of symbol FF. As such, Murashita’s cumulative frequency of occurrence does not even correlate with the frequency of occurrence of a symbol. *Id.*

B. Petitioner Has Not Established That the Challenged Claims Are Unpatentable in Light of Ground 2—Murashita in Combination With Marpe.

As explained in further detail below, there are multiple reasons why Murashita does not teach or suggest to a POSITA the limitations of independent claims 1, 8, or 15.

1. Petitioner Fails to Demonstrate that Limitations 1[a], 8[a], and 15[a] are Obvious Over Murashita

Limitations 1[a], 8[a], and 15[a] each require “determining a frequency of occurrence of syntax elements in a bit stream.” ’701 Patent, 14:7–9. For the reasons discussed above, Petitioner has not shown that Murashita teaches or suggests these limitations.

2. Marpe Does Not Categorize Syntax Elements Based on “Frequency of Occurrence”

The '701 Patent requires that syntax elements be classified according to how frequently they occur in the bitstream. Claim 1 recites determining “a frequency of occurrence threshold based on an expected frequency of occurrence of syntax elements in a bit stream” and “categorizing a plurality of syntax elements... wherein syntax elements which occur greater than the frequency of occurrence threshold are categorized into the first category and syntax elements which occur less than the frequency of occurrence threshold are categorized into the second category.” ‘701 Patent, 14:7-16. The ‘701 Patent also requires applying different entropy-coding treatments to each category—coding with context updating for one category, coding with bypassed context updating for the other. *Id.* at 14:17-22.

However, Marpe does not teach these limitations. Marpe describes CABAC (Context-based Adaptive Binary Arithmetic Coding), where a bin enters “the context modeling stage, where a probability model is selected such that the corresponding choice may depend on previously encoded syntax elements or bins.” Ex-1006, ¶ 0119. Marpe further recites that in bypass mode, “the bypass coding engine... works preferably with a fixed probability model without any context adaption.” Ex-1006, ¶ 0120. These disclosures confirm that the distinction between regular and bypass modes is predetermined by the structure of the syntax element and bin type—not by any “frequency of occurrence threshold.” According to Dr. Richardson, Marpe nowhere

indicates that the selection of coding mode depends on how often a syntax element appears in the bitstream. Ex-2009 ¶ 83. Equally important, Dr. Richardson contends that Marpe’s context-modeling mechanisms are structurally based, not frequency-based. Ex-2009 at ¶ 84. Marpe states that “[f]our basic design types of context models can be distinguished,” including models that use “up to two neighboring syntax elements,” models where “values of prior coded bins” are used, and models that depend on block type or scanning position. *Id.* at ¶¶ 0130–0132.

According to Dr. Richardson, none of these mechanisms involve determining an expected frequency of occurrence, computing a threshold, or dividing syntax elements based on whether they occur more or less frequently than such a threshold. Ex-2009 at ¶ 85. Marpe’s modeling is driven by local structure, bin location, and neighborhood context—not statistical occurrence frequency. *Id.*

Marpe further reinforces that bypass mode is selected according to bin type—not based on any syntax element occurrence statistics. Marpe recites that bypass mode “is especially effective when coding the bins of the primary suffix of those syntax elements concerning components of differences of motion vectors and transform coefficient levels.” *Id.* at ¶0120. This is a fixed rule tied to binarization structure—the suffix portion—rather than any statistic about how frequently a syntax element occurs. Dr. Richardson contends nothing in Marpe suggests that bypass mode is invoked because a syntax element is rare or common. Ex-2009 ¶ 86.

For these reasons, Marpe lacks any teaching of a “frequency of occurrence threshold” or of categorizing syntax elements according to how often they are expected to occur. Marpe’s CABAC operation is wholly structural rather than frequency-driven. Ex-2009 ¶ 87. Because the ’701 Patent’s frequency-based categorization requirement is missing from Marpe, Petitioner cannot satisfy this limitation under obviousness. *Id.*

3. Murashita in view of Marpe Cannot Be Combined Because They Would Produce an Unworkable Result

A person of ordinary skill in the art could not feasibly merge the architectures disclosed in Murashita and Marpe, as doing so would dismantle the essential operating principles of both systems. Ex-2009 ¶ 90. According to Dr. Richardson, the two references approach entropy coding from fundamentally different premises: Murashita relies on a single, global, multi-level arithmetic coding model, while Marpe employs a distributed, context-driven, binary arithmetic coding framework. Ex-2009 ¶¶ 91–92. These approaches are technically incompatible, and attempting to graft one onto the other would produce a nonfunctional system.

Murashita’s encoder is built around a centralized dictionary—a global table of symbol frequencies—that governs every probability assignment during coding. Ex-2009 ¶ 91. The reference expressly states that “ranges are divided according to frequency of occurrence” and that this requires a “dictionary as shown in FIG. 4A” to record and maintain symbol counts. Ex-1007, 3:57–60. In this paradigm, arithmetic subdivision proceeds by allocating a subrange for each possible symbol value, as

illustrated in Figure 1A (showing five distinct subranges). Because the cumulative-frequency data in the dictionary determines the partitioning of the coding interval, maintaining and updating this single table is indispensable to Murashita's method.

Marpe works in an entirely different manner. Rather than tracking global symbol frequencies, Marpe utilizes Context-based Adaptive Binary Arithmetic Coding (CABAC), where each syntax element is first binarized into a sequence of binary "bins," and each bin is assigned a probability model based on local structural cues—neighboring syntax elements, block type, scanning position, or prior bin values. Ex-2009 ¶ 92; Ex-1006 ¶¶ 0114–0115, 0130–0132. Marpe emphasizes that in CABAC "a given non-binary valued syntax element is uniquely mapped to a binary sequence" (Ex-1006, ¶ 0115) and that in regular mode "a probability model is selected" based on previously encoded bins (Ex-1006, ¶ 0119). Unlike Murashita's global table, these context models evolve independently as the encoder processes different parts of a video frame. Ex-2009 ¶ 92.

Because the two systems rely on opposing update mechanisms, they cannot be combined without destroying their inherent functionality. Murashita's algorithm requires updating one cumulative-frequency table for all symbols, whereas Marpe's CABAC engine depends on multiple evolving context states whose updates diverge based on the specific binarization path taken during video encoding. Ex-2009 ¶ 93. Attempting to inject Marpe's context updates into Murashita's global table would break

the arithmetic-subdivision structure that the table must preserve. Conversely, forcing Murashita's cumulative-frequency updates into CABAC would disrupt the local-context modeling on which CABAC depends. Ex-2009 ¶¶ 94–95. Either approach would render the resulting coder inoperable.

The gulf between the two systems is not limited to coding mechanics—their applications and design motivations also diverge sharply. Murashita targets general-purpose compression of files such as text. Ex-1007, 1:7–13, 1:33–37. Marpe, on the other hand, is specific to video compression, where CABAC's binary arithmetic coding is used because it can be effective for “motion vector differences or transform coefficient level values.” Ex-1006 ¶ 14. A POSITA would understand that m-ary arithmetic coding can model multi-valued symbol probabilities more directly than binary arithmetic coding, which must approximate them through binarization. Ex-1006 ¶ 16; Ex-2009 ¶ 96. According to Dr. Richardson, the rationale supporting Marpe's design choices simply does not translate to Murashita. Ex-2009 ¶ 96.

Murashita itself identifies the problem it aims to solve: long processing times in text compression. Ex-1007, 5:4–10. Its solution is to selectively update a cumulative-frequency table—not to redesign its coding architecture. A POSITA would thus see no reason to replace Murashita's m-ary arithmetic coder with Marpe's CABAC engine, which is not only architecturally incompatible but also optimized for a different purpose. Petition at 54; Ex-2009 ¶ 97.

Petitioner nevertheless attempts to splice together fragments from both references, stitching Marpe's CABAC decision tree onto Murashita's global frequency framework. Petition at 57. In Marpe's Figure 2, the choice between regular and bypass modes turns on the binarization and value of the syntax element. *See* Ex-1006, ¶¶ 81–83. Yet Petitioner proposes that this decision be controlled instead by Murashita's cumulative symbol frequencies. Petition at 58. According to Dr. Richardson, the Petition never explains how cumulative-frequency data could drive per-bin CABAC mode selection, what effect such a change would have on coding performance, or how the modified system would maintain correctness in either coding framework. Ex-2009 ¶ 98. The theory rests on hindsight and uses the '701 Patent's solution as a template rather than deriving a motivation from the prior art itself. *Id.* For these reasons, a POSITA would have neither the motivation nor the ability to combine Murashita and Marpe into a single functional system.

4. No Reasonable Expectation of Success in Combining Murashita with Marpe

For the reasons described in § III.B.3., a POSITA would not have been motivated to modify Murashita's system/process in view Marpe in the manner proposed by the Petition, nor would a POSITA have had a reasonable expectation of success regarding the outcome. Ex-2009 at ¶ 99.

Even if a POSITA were to successfully combine Murashita with Marpe, the

combination would not satisfy the Challenged Claims at least because neither reference discloses categorizing syntax elements based on their frequency of occurrence, nor entropy coding symbols based on such a categorization, as explained above. Ex-2009 at ¶ 100.

C. Petitioner Has Not Established That The Challenged Claims Are Unpatentable in Light of Ground 3—Yu

Yu fails to disclose any changes to the entropy coding process as implemented in H.264's CABAC and merely suggests an allegedly improved hardware implementation capable of improved processing. In other words, Yu discloses improvement to hardware rather than processing. Ex-2009 at ¶ 101.

1. Yu's Syntax Element Categories Do Not Determine Whether They Are Context-coded or Bypass-coded

Yu discloses “two classes of syntax elements,” which are classified based on bin length. Ex-1018 at 1355. This classification is not dispositive as to how symbols corresponding to the syntax elements are entropy coded. Yu's first class of syntax element contains syntax elements that “contain a few bins and occur only once or a few times in one MB.” *Id.* Yu's second class includes syntax elements that “contain several bins and occur tens or hundreds of times in one MB.” *Id.* Yu's hardware processes the bins at different speed cycles based on this classification: the first group has “*one bin* [that] is processed per cycle” and the second group has *three bins*, “two regular [context updated] bins together with one bypass bin” processed per cycle. *Id., emphasis added.*

Notbaly, in Yu, there is no distinction as to whether the first group's bin is bypass or regular ("context") coded. Ex-2009 at ¶ 102.

2. Yu's Syntax Element Categories are Unrelated to the Comparison of *Offset* to an Alleged Threshold.

As Dr. Richardson points out, Petitioner mischaracterizes Yu in an attempt to link Yu's two "classes" with the determination of LPS or MPS. *See* Ex-1018 at 1355; Ex-2009 at ¶ 103. Petition is incorrect that syntax elements that contain a few bins and occur only once or a few times in one MB, correspond or relate to "bins where the offset value is below the threshold, indicating an LPS." Petition at 89; Ex-2009 at ¶ 103. Petitioner is likewise incorrect that elements that contain several bins and occur tens or hundreds of times in one MB, correspond or relate to "bins where the offset value is above the threshold, indicating an MPS." Petition at 89; Ex-2009 at ¶ 103. The fundamental process of binary arithmetic coding requires that every bin coded in "regular mode" has the possibility of being an LPS (least probable symbol value, 0 or 1) or an MPS (most probable symbol value, 0 or 1). Without this possibility, there would be no need to code any value. Yu discloses this. Ex-1018 at 1353-1354. This means that there is no link between Yu's first and second "classes" and the comparison of *offset* to the subintervals of rLPS and rMPS. Ex-2009 at ¶ 103.

3. Yu's Use of Bypass Is Not Related to the "Frequency of Occurrence of Syntax Elements"

Yu's disclosure of bypass coding is wholly unrelated to the "frequency of

occurrence of syntax elements.” Instead, the decision of whether to use bypass mode is determined based on the type of the syntax element, according to H.264’s CABAC coding rules. Ex-2009 at ¶ 104. For example, Yu states that “bins encoded by bypass mode only appear in two cases, either as sign or as Exp-Golomb (EG) [suffix] code.” Ex-1018 at 1355. Yu also explains that the EG suffix occurs only when certain values (e.g., `abs_mvd` greater than 8 or `coeff_abs_level_minus1` greater than 13) are present, which, according to Yu, are “rare occasions” because “at least 90%” of these syntax elements have small values that do not trigger bypass suffixes. Ex-1018 at 1355; Ex-2009 at ¶ 104. As Dr. Richardson explains, this means that “in Yu (as in H.264), bypass coding is determined by whether a value is represented as a sign bit or EG suffix and not by the rate or frequency with which a syntax element arises.” EX-2009 at ¶ 104. Yu’s disclosure stands in contrast to Petitioner’s incorrect statement that “high-occurrence [syntax] elements are decoded in bypass mode, where context is not needed.” Petition at 84; Ex-2009 at ¶ 104. Instead, Yu’s bypass coding is applied to bins that have been determined to have approximately 50% probability of being 0 or 1, found in two scenarios: (i) sign, which is approximately equally likely to be positive or negative, and (ii) the aforementioned infrequently-occurring Exp-Golomb suffix bins. *Id.*

Yu’s discussion of syntax element classes (frequent versus infrequent syntax elements) relates only to hardware scheduling—i.e., whether a syntax element is placed

in the “first class” (processed one bin per cycle) or “second class” (processed two regular bins, and when applicable, one bypass sign bit per cycle). Ex-2009 at ¶ 105. This is not a basis for selecting bypass versus context-coded arithmetic decoding. Ex-1018 at 1355. Yu fails to provide any connection between bypass coding and frequency of occurrence. Instead, bypass mode is strictly determined by the internal coding structure of the syntax element, not how often that element appears. Ex-2009 at ¶ 105. Further, Yu’s discussion of the “first class” and “second class” of syntax elements does not relate to a categorization of syntax elements based on a comparison with a frequency of occurrence threshold as required by the claims of the ‘701 Patent. *Id.* Yu’s “first class” and “second class” are categories of syntax elements types that have already been defined in the H.264 Standard, e.g. “[t]hose syntax elements in the second class include `abs_mvd`, `significant_coeff_flag`, `last_significant_coeff_flag` and `coeff_abs_level_minusl`.” Ex-1018 at 1355; Ex-2009 at ¶ 105.

4. Yu’s Probability Monitoring Is Not “Expected Frequency of Occurrence of Syntax Elements”

Petitioner contends that Yu’s CABAC framework discloses the ‘701 Patent’s frequency-based categorization and threshold. Petitioner claims that by modeling probabilities for bins and splitting range into rLPS and rMPS such that, in “regular mode,” “the LPS probability `pLPS` is estimated by the corresponding context model,” while in “bypass mode,” “`pLPS` is estimated as 0.5 and no context model is needed,” establishes a threshold between LPS and MPS decisions based on expected occurrence

frequency. *See* Petition at 82–84, 88–90; Ex-2009 at ¶ 106. Petitioner incorrectly summarizes Yu, claiming that Yu “classifies SEs [syntax elements] into two classes based on frequency of occurrence: a first class with SEs that ‘occur only once or a few times in one MB [macroblock],’ and a second class with SEs that ‘occur tens or hundreds of times in one MB [macroblock],’” and uses a fixed decision rule—“range/2 to determine *the bin* is MPS or LPS”—in bypass mode. *Id.* at 83–84, 88–89. The Petition follows this argument to assert that “the decision point between decoding *a bin* as LPS or MPS (the threshold) thus indicates the frequency of occurrence of symbols at which context updating occurs”. *Id.* at 90. Dr. Richardson explains the faults with Petitioner’s arguments in detail. *See* Ex-2009 at ¶¶ 108-112.

First, Dr. Richardson explains that “Yu processes and determines probabilities one bin at a time, not for any batch or set of bins which make up a syntax element.” Ex-2009 at ¶¶ 108-109. Yu states that a “bin is a binary value and the basic arithmetic decoding unit in CABAC” meaning each bin is just a single binary symbol (0 or 1). Ex-1018 at 1352. Yu makes clear that the associated processes (probability lookup, the decision of LPS versus MPS, and the context model) occur per bin, not per group of bins. Ex-2009 at ¶¶ 108-109; Ex-1018 at 1352 (“[b]efore arithmetically encoding one bin, the corresponding context model is selected...and the model is updated after the bin is encoded.”). Even when Yu’s hardware decodes more than one bin in a cycle, the probability estimation and decision processes are bin-by-bin operations and rely on the

model's probability estimation for the given bin. Ex-2009 at ¶¶ 108-109.

Second, Dr. Richardson explains that “Petitioner’s reliance on Yu’s ‘probability modeling’ to suggest disclosure of an “expected frequency of occurrence of syntax elements in a bitstream” is misplaced. Yu’s probability modeling concerns only the likelihood that a decoded bin (i.e., a single binary symbol) is a 0 or 1 and is not tied to how often any syntax element appears in the bitstream.” Ex-2009 at ¶ 110. Yu defines probability modeling as the process by which “each context model... corresponds to a certain bin,” and where each model stores (1) “a 6-bit state value... as the probability index” and (2) “a binary value of the most probable symbol (MPS).” Ex-1018 at 1353; Ex-2009 at ¶ 110. The model outputs an estimate of *pLPS*, “the estimated probability value of the least probable symbol”, and then subdivides the arithmetic-coding interval into two (*rLPS* and *rMPS*) based on that per-bin probability. Ex-1018 at 1353; Ex-2009 at ¶ 110. The probability modeling of Yu is not tied to the frequency of occurrence of syntax elements. Ex-2009 at ¶ 111 Yu makes this clear in stating that syntax elements are separate constructs that “are uniquely mapped to a bin string,” meaning syntax elements consist of individual bins, and the probability model operates at the level of each bin within that string. Ex-1018 at 1353; Ex-2009 at ¶ 111. This is distinct from operating at the level of the syntax element itself. Ex-1018 at 1353; Ex-2009 at ¶ 111. at 1353. As such, Yu’s probability modeling relates only to the probability that a bin is 0 or 1, not any expected frequency of occurrence of syntax elements. Ex-2009 at ¶ 112.

Additionally, when Yu decodes “two regular bins together with one bypass bin in one cycle,” the hardware is not applying a single probability model to all three bins. Instead, each regular (i.e. context coded) bin uses a unique context state and rLPS/rMPS values. As Dr. Richardson explains, Bypass bins, by definition, **do not use a probability model at all** and rely on a fixed assumption of a probability of 0 or 1 being 0.5. *Id.* at 1353–55; Ex-2009 at ¶ 112.

5. Petitioner’s Reliance on Yu’s Threshold is Misplaced

Petitioner mischaracterizes Yu’s “threshold” in an attempt to link the “threshold” to a frequency of occurrence. *See* Ex-3001 at 90. Ex-2009 at ¶113. Yu’s “threshold” is not a measure of how frequently a syntax element appears in a bitstream, and is unrelated to any occurrence-based criterion. *Id.* As Dr. Richardson points out, “Yu expressly explains that the arithmetic decoder divides the current interval into two probability-based subintervals—rLPS and rMPS—where ‘rLPS correspond[s] to the estimated probability interval of the LPS’ and rMPS correspond[s] to the interval of the MPS. Ex-2009 at ¶ 114 (quoting Ex-1018 at 1353). This means Yu’s decoder determines whether a bin is an MPS or LPS by checking “whether the offset falls into the subinterval of rLPS or rMPS.” *Id.* Petition relies on this comparison as an alleged “threshold.” However, it is nothing more than a probability-model decision applied to a single bin value (i.e., whether that bin is more likely a 0 or 1). Ex-2009 at ¶ 114. Yu’s “threshold” is not linked to whether a bin is coded with bypass or context updating and

is unrelated to the frequency of occurrence of a syntax element. *Id.* Further, Yu's alleged threshold has no relationship with Yu's "first class" and "second class" of syntax elements. Ex 1018 at 1355.

Dr. Richardson also points out that "Yu's alleged threshold (rMPS) is only applied to bins that the encoder/decoder has already determined are to be coded using a context update. In the event that the encoder/decoder is performing bypass coding, a static 50/50 probability is used and there is no LPS or MPS, as they are equiprobable. Bypass bins therefore do not make any use of the rMPS and instead the '*offset*' value is compared with *range* to determine [sic] the bin is 1 or 0.'" Ex-2009 at ¶ 115 (quoting Ex-1018 at 1354).

This means that even if rMPS were the claimed threshold (which it is not), the Yu rMPS is not used to categorize syntax elements or bins into categories such that one category is coded using bypass mode. Ex-2009 at ¶ 116.

The Petition attempts to arbitrarily define a threshold that suits its needs. Ex-2009 at ¶ 117. The Challenged Claims of the '701 Patent require a categorization of syntax elements based on the threshold frequency of occurrence and different entropy coding based on such categorization. For the reasons discussed above, Yu does not teach independent claims 1, 8, and 15.

Claims 3, 6-7, 10, 13-14, 17, 20 depend on the independent claims. For the reasons discussed above, Petitioner has failed to show that Yu discloses these claims.

6. As to Dependent Claims 6, 13, and 20, Yu's Threshold is Not "Predefined"

Yu does not disclose the use of a "predefined" threshold for determining a bin value. Ex-2009 at ¶ 120. Rather, Yu states that the probability model stored in the context state is used to dynamically generate probability models, which in turn are used to as Petitioner's alleged "threshold." Ex-1018 at 1353. Yu explains, the decoder divides the coding interval into two subintervals representing the probability of the rLPS and rMPS. *Id.*

Following this division, the decoder determines the bin value by checking "whether the offset falls into the subinterval of rLPS or rMPS." *Id.* Both rLPS and rMPS are obtained from a lookup table indexed by the *current state* of the context model—"a 6-bit state value... used as an index of the estimated probability value." *Id.*

Yu confirms this probability state is dynamic, as it is "updated after encoding each bin." Ex-1018 at 1353. As such, Yu's probability decision is neither fixed nor "predefined." The probability state updates with each bin and cannot operate as a predetermined or "predefined threshold." Ex-2009 at ¶ 122. Thus, for the reasons discussed above, Yu does not teach dependent claims 6, 13, and 20.

IV. CONCLUSION

For any or all of the above-described reasons, Patent Owner respectfully requests that the Board deny institution.

Dated: November 28, 2025

Respectfully submitted,

/s/ Scott W. Hejny

Scott W. Hejny

Reg. No. 45,882

shejny@mckoolsmith.com

Eric S. Hansen

Reg. No. 63,095

ehansen@mckoolsmith.com

McKool Smith, P.C.

300 Crescent Court, Suite 1200

Dallas, TX 75201

Telephone: 214.978.4000

Facsimile: 214.978.4044

*Counsel for Patent Owner,
Nokia Technologies Oy*

CERTIFICATION OF WORD COUNT

The undersigned hereby certifies that the portions of the above-captioned **PATENT OWNER'S PRELIMINARY RESPONSE** specified in 37 C.F.R. §42.24 have 8,466 words, in compliance with the 14,000 word limit set forth in 37 C.F.R. §42.24(b)(1). This word count was prepared using Microsoft Word 2010.

Dated: November 28, 2025

By: /s/ Scott W. Hejny
Scott W. Hejny
Reg. No. 45,882

CERTIFICATION OF SERVICE

The undersigned hereby certifies that a copy of this **PATENT OWNER'S PRELIMINARY RESPONSE PURSUANT TO 37 C.F.R. § 42.10(b)** has been served on Petitioner via email on the following counsel of record for Petitioner:

Brian E. Ferguson
beferguson@winston.com

Eimeric Reig
ereigplessis@winston.com

Joseph C. Masullo
jmasullo@winston.com

Lauren M. Rennecker
lrennecker@winston.com

Patrick J. McKeever
Mckeever-ptab@perkinscoie.com

Jon R. Carter
Carter-ptab@perkinscoie.com

Wei Yuan
Yuan-ptab@perkinscoie.com

Nicholas Lipperd
Lipperd-ptab@perkinscoie.com

Hisense-NokiaIPR@perkinscoie.com

Dated: November 28, 2025

By: /s/ Scott W. Hejny
Scott W. Hejny
Reg. No. 45,882