UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

SNAP INC. and HISENSE USA CORPORATION,
Petitioners

v.

NOKIA TECHNOLOGIES OY,
Patent Owner.

IPR2025-01345

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

PETITION FOR *INTER PARTES* REVIEW OF U.S. PATENT NO. 9,036,701

TABLE OF CONTENTS

Page

I. INTRODUCTION 1

II. GROUNDS FOR STANDING 1

III. IDENTIFICATION OF CHALLENGE 1

- A. Prior Art 1
- B. Grounds for Challenge 2

IV. ’701 PATENT OVERVIEW 2

- A. Subject Matter Overview of the ’701 Patent 2
- B. Brief Summary of ’701 Patent Prosecution History 5

V. LEVEL OF ORDINARY SKILL 10

VI. CLAIM CONSTRUCTION 10

VII. DETAILED EXPLANATION OF GROUNDS 10

- A. Ground 1: Claims 1, 6, 8, 13, 15, 20 are obvious over Murashita 10
  - 1. Murashita 10
  - 2. Claim 1 and Dependents 13
    - a. Claim 1 13
    - b. Claim 6 46
  - 3. Claim 8 and Dependents 47
    - a. Claim 8 47
    - b. Claim 13 48
  - 4. Claim 15 and Dependents 49
    - a. Claim 15 49
    - b. Claim 20 49
- B. Ground 2: Claims 1–20 are obvious over Murashita in view of Marpe 50
  - 1. Marpe 50
  - 2. Motivation to Combine Murashita and Marpe 52
  - 3. Claim 1 and Dependents 58
    - a. Claim 1 58
    - b. Claim 2 66

c. Claim 3 ... 69
d. Claim 4 ... 71
e. Claim 5 ... 72
f. Claim 6 ... 74
g. Claim 7 ... 75

4. Claim 8 and Dependents ... 77

a. Claim 8 ... 77
b. Claim 9 ... 78
c. Claim 10 ... 78
d. Claim 11 ... 79
e. Claim 12 ... 79
f. Claim 13 ... 79
g. Claim 14 ... 79

5. Claim 15 and Dependents ... 79

a. Claim 15 ... 79
b. Claim 16 ... 80
c. Claim 17 ... 80
d. Claim 18 ... 80
e. Claim 19 ... 81
f. Claim 20 ... 81

C. Ground 3: Claims 1, 3, 6-8, 10, 13-15, 17, 20 are obvious over Yu ... 81

1. Yu 81
2. Claim 1 and Dependents ... 84

a. Claim 1 ... 84
b. Claim 3 ... 96
c. Claim 6 ... 96
d. Claim 7 ... 98

3. Claim 8 and Dependents ... 98

a. Claim 8 ... 98
b. Claim 10 ... 100
c. Claim 13 ... 100
d. Claim 14 ... 101

4. Claim 15 and Dependents ... 101

a. Claim 15 ... 101

b. Claim 17 102
c. Claim 20 102

VIII. CONCLUSION 102

IX. MANDATORY NOTICES 102

A. Real Party in Interest 102
B. Related Matters 103
C. Notice of Counsel and Service Information 103

EXHIBIT LIST

Exhibit	Reference
1001	U.S. Patent No. 9,036,701 (“the ’701 Patent” or “’701Pat.”)
1002	File History of the ’701 Patent (“’701FH”)
1003	Declaration of Dr. Eli Saber (“Saber”)
1004	Curriculum Vitae of Dr. Eli Saber
1005	Detlev Marpe, *Content-Based Adaptive Binary Arithmetic Coding in the H.264/AVC Video Compression Standard*, 13 IEEE TRANSACTIONS ON CIRCUITS & SYSTEMS FOR VIDEO TECH. 620 (July 2003) (“CABAC”)
1006	U.S. Patent Publication No. US 2005/0253740 A1 (“Marpe”)
1007	U.S. Patent No. 5,650,783 (“Murashita”)
1008	File History of EP2599315
1009	Iain E. Richardson, THE H.264 ADVANCED VIDEO COMPRESSION STANDARD (2nd ed. 2010)
1010	Steven M. Kaplan, WILEY ELECTRICAL AND ELECTRONICS ENGINEERING DICTIONARY (2004)
1011	A & C Black, DICTIONARY OF COMPUTING, (6th ed. 2010)
1012	Rafael C. Gonzales and Richard E. Woods, DIGITAL IMAGE PROCESSING (Prentice Hall pub., 2nd ed. 2001)
1013	A. Murat Tekalp, DIGITAL VIDEO PROCESSING (Prentice Hall pub., 1st ed. 1995)
1014	ISO/IEC 11172-2 (1st ed.) Information technology - Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s - Part 2: Video (August 1, 1993) (“MPEG-1”)

1015	ITU-T Recommendation H.262 (July 1995) (“H.262”)
1016	ITU-T Recommendation H.263 (February 1998) (“H.263”)
1017	ITU-T Recommendation H.264 (March 2010) (“H.264”)
1018	Wei Yu & Yun He, *A High Performance CABAC Decoding Architecture*, 51 IEEE 1352 (Nov. 2005) (“Yu”)
1019	Declaration of Sylvia Hall-Ellis (“Hall-Ellis”)
1020	U.S. Patent Publ. No. 2010/0150226 A1
1021	Exhibit 20 to ITC Preliminary Infringement Contentions for alleged infringement of U.S. Patent No. 9,036,701 by Hisense Accused Products, Inv. No. 337-TA-1448

TABLE OF ABBREVIATIONS

Abbreviation	Definition
’701 Patent/’701Pat.	U.S. Patent 9,036,701 (EX-1001)
Challenged Claims	Claims 1–20
DCT	Discrete Cosine Transform
IPR	*inter partes* review
POSITA	Person of Ordinary Skill in the Art
Snap/Petitioner	Snap Inc.

LISTING OF CHALLENGED CLAIMS

Reference	Claim Limitation[1]
Claim 1	
1[pre]	A method comprising:
1[a]	determining a frequency of occurrence threshold based on an expected frequency of occurrence of syntax elements in a bit stream;
1[b]	categorizing a plurality of syntax elements of video content into first and second categories based on the frequency of occurrence threshold, wherein syntax elements which occur greater than the frequency of occurrence threshold are categorized into the *second category* and syntax elements which occur less than the frequency of occurrence are categorized into the *first category*;
1[c]	entropy coding symbols that correspond to the first category of syntax elements and that have been subjected to a context update; and
1[d]	entropy coding symbols that correspond to the second category of syntax elements and that have bypassed context updating.
Claim 2	
2	A method according to claim 1 further comprising causing a categorization of the syntax elements to be signaled at a block, slice, picture or sequence level.
Claim 3	

[1] Italicized claim language in this table denotes corrected language, per the Certificate of Correction signed and sealed on April 5, 2016.

3	A method according to claim 1 further comprising performing binarization of the symbols that correspond to the second category of syntax elements prior to bypassing the context updating.
Claim 4	
4	A method according to claim 3 wherein performing binarization comprises performing binarization so as to target a predetermined probability.
Claim 5	
5	A method according to claim 1 further comprising causing an indication of the entropy coding to be signaled at a block, slice, picture or sequence level.
Claim 6	
6	A method according to claim 1 wherein categorizing the plurality of syntax elements comprises categorizing the plurality of syntax elements based on a relationship of the frequency of occurrence of the syntax elements to a predefined threshold.
Claim 7	
7	A method according to claim 1 further comprising performing binarization of the symbols that correspond to the first category of syntax elements prior to entropy coding the symbols that correspond to the first category of syntax elements.
Claim 8	
8[pre]	An apparatus comprising at least one processor and at least one memory including computer program code, the at least one memory and the computer program code configured to, with the processor, cause the apparatus to at least:
8[a]	determine a frequency of occurrence threshold based on an expected frequency of occurrence of syntax elements in a bit stream;
8[b]	categorize a plurality of syntax elements of video content into first and second categories based on the frequency of occurrence

	threshold, wherein syntax elements which occur greater than the frequency of occurrence threshold are categorized into the *second category* and syntax elements which occur less than the frequency of occurrence are categorized into the *first category*;
8[c]	entropy code symbols that correspond to the first category of syntax elements and that have been subjected to a context update; and
8[d]	entropy code symbols that correspond to the second category of syntax elements and that have bypassed context updating.
Claim 9	
9	An apparatus according to claim 8 wherein the at least one memory and the computer program code are further configured to, with the processor, cause the apparatus to cause a categorization of the syntax elements to be signaled at a block, slice, picture or sequence level.
Claim 10	
10	An apparatus according to claim 8 wherein the at least one memory and the computer program code are further configured to, with the processor, cause the apparatus to perform binarization of the symbols that correspond to the second category of syntax elements prior to bypassing the context updating.
Claim 11	
11	An apparatus according to claim 10 wherein the at least one memory and the computer program code are configured to, with the processor, cause the apparatus to perform binarization by performing binarization so as to target a predetermined probability.
Claim 12	
12	An apparatus according to claim 8 wherein the at least one memory and the computer program code are further configured to, with the processor, cause the apparatus to cause an indication of the entropy coding to be signaled at a block, slice, picture or sequence level.
Claim 13	
13	An apparatus according to claim 8 wherein the at least one memory and the computer program code are configured to, with the

	processor, cause the apparatus to categorize the plurality of syntax elements by categorizing the plurality of syntax elements based on a relationship of the frequency of occurrence of the syntax elements to a predefined threshold.
Claim 14	
14	An apparatus according to claim 8 wherein the at least one memory and the computer program code are further configured to, with the processor, cause the apparatus to perform binarization of the symbols that correspond to the first category of syntax elements prior to entropy coding the symbols that correspond to the first category of syntax elements.
Claim 15	
15[pre]	A computer program product comprising at least one non-transitory computer-readable storage medium having computer-executable program code instructions stored therein, the computer-executable program code instructions comprising program code instructions to:
15[a]	determine a frequency of occurrence threshold based on an expected frequency of occurrence of syntax elements in a bit stream;
15[b]	categorize a plurality of syntax elements of video content into first and second categories based on the frequency of occurrence threshold, wherein syntax elements which occur greater than the frequency of occurrence threshold are categorized into the *second category* and syntax elements which occur less than the frequency of occurrence are categorized into the *first category*;
15[c]	entropy code symbols that correspond to the first category of syntax elements and that have been subjected to a context update; and
15[d]	entropy code symbols that correspond to the second category of syntax elements and that have bypassed context updating.
Claim 16	
16	A computer program product according to claim 15 wherein the computer-executable program code instructions further comprise

	program code instructions to cause a categorization of the syntax elements to be signaled at a block, slice, picture or sequence level.
Claim 17	
17	A computer program product according to claim 15 wherein the computer-executable program code instructions further comprise program code instructions to perform binarization of the symbols that correspond to the second category of syntax elements prior to bypassing the context updating.
Claim 18	
18	A computer program product according to claim 17 wherein the comprise program code instructions to perform binarization comprise program code instructions to perform binarization so as to target a predetermined probability.
Claim 19	
19	A computer program product according to claim 15 wherein the computer-executable program code instructions further comprise program code instructions to cause an indication of the entropy coding to be signaled at a block, slice, picture or sequence level.
Claim 20	
20	A computer program product according to claim 15 comprise the program code instructions to categorize the plurality of syntax elements comprise program code instructions to categorize the plurality of syntax elements based on a relationship of the frequency of occurrence of the syntax elements to a predefined threshold.

I. INTRODUCTION

Petitioners request IPR of the Challenged Claims of the ’701 Patent.[2]

II. GROUNDS FOR STANDING

Petitioners certify that the ’701 Patent is available for IPR, and that Petitioners are not barred or estopped from requesting IPR on the grounds herein. Petitioners are unaware of any prior IPR challenges to any claims of the ’701 Patent.

III. IDENTIFICATION OF CHALLENGE

A. Prior Art

Petitioners respectfully request cancellation of the Challenged Claims based on:

Prior Art
Murashita (EX-1007); issued on July 22, 1997; prior art under at least 35 U.S.C. §102(b).[3]
Marpe (EX-1006); published as patent application publication on November 17, 2005; prior art under at least 35 U.S.C. §102(b).
Yu (EX-1018); published as a technical paper in the IEEE *Transactions on Consumer Electronics* journal in November 2005; prior art under at least 35 U.S.C. §102(b).

[2] All emphasis/annotations added to figures and text, unless stated otherwise.

[3] All references to 35 U.S.C. §§102, 103 are to the pre-AIA statute.

B. Grounds for Challenge

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

IV. ’701 PATENT OVERVIEW

A. Subject Matter Overview of the ’701 Patent

The ’701 Patent is titled “Method and Apparatus for Providing Complexity Balanced Entropy Coding.” It issued on May 19, 2015, from U.S. Patent Application 13/192,111 (the “’111 Application”), filed on Jul. 27, 2011. The ’701 Patent claims priority to U.S. Provisional Application 61/368,316, filed July 28, 2010. For purposes of this Petition only, Petitioners have applied July 28, 2010, as the priority date of the ’701 Patent.

The ’701 Patent “relate[s] generally to electronic device video coding and decoding technology.” ’701Pat., 1:12–15. The ’701 Patent explains that at the time of the alleged invention, there was a “market driven desire to keep [mobile electronics] devices relatively small,” and that this “inevitably introduce[d] challenges with respect to limits in battery life and processing resources” required for content communication. ’701Pat., 1:37–43. The ’701 Patent explains that this made it “desirable to maintain high levels of capabilities in the most efficient manner possible." *Id.*, 1:49–51. The '701 Patent purportedly addresses this desire by using "complexity balanced entropy coding." *Id.*, Abstract.

"FIG. 5 is a flowchart… for providing complexity balanced entropy coding[.]" *Id.*, 2:10–12. Step 110 involves "categorizing syntax elements of video content based on a frequency of occurrence of syntax elements in the video content." *Id.*, 12:43–47. This "may include categorizing symbols having a frequency of occurrence below a threshold as first category symbols and categorizing symbols having a frequency of occurrence above the threshold as second category symbols." *Id.*, 13:7–12. "[B]y categorizing the syntax elements based on their relative frequencies of occurrence, more complex operations may be performed with respect to less frequently occurring symbols and complex operations may be simplified or omitted with respect to more frequently occurring symbols." *Id.*, 11:66–12:3. This categorization allows for "different categories of symbols [to] be operated on differently in order to achieve relatively higher efficiency in a lower complexity implementation." *Id.*, 4:53–56.

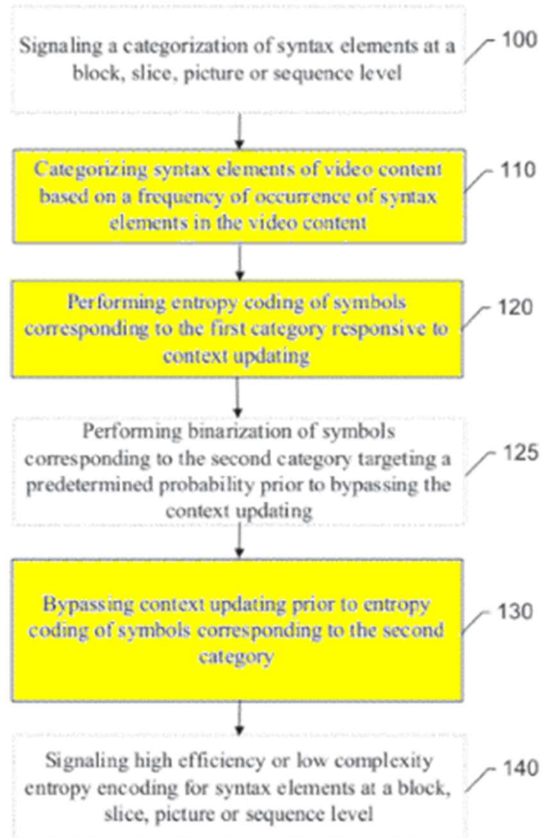


FIG. 5.

’701Pat., Fig. 5.

Step 120 involves “performing entropy coding of symbols corresponding to the first category responsive to context updating,” and Step 130 involves “bypassing context updating prior to entropy coding of symbols corresponding to the second category.” *Id.*, 12:48–53. Therefore, the “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).” *Id.*, 10:45–48.

B. Brief Summary of ’701 Patent Prosecution History

The ’111 Application was filed on July 27, 2011, and included pending independent claims 1, 8, and 15, each substantially similar claims, differing only in their preambles. Claim 1 (below) was directed to a method, claim 8 to an apparatus, and claim 15 to a computer program product.

> 1. A method comprising:
>
> categorizing a plurality of syntax elements of video content into first and second categories based on a frequency of occurrence of the syntax elements in the video content;
>
> entropy coding symbols that correspond to the first category of syntax elements and that have been subjected to a context update; and
>
> entropy coding symbols that correspond to the second category of syntax elements and that have bypassed context updating.

’701FH, 25–28.

On October 31, 2013, Examiner rejected all pending claims under 35 U.S.C. §102(6) as anticipated by US 2009/0141806 A1 (“Marpe’806”). ’701FH, 191–94. Specifically, Figures 1–16 of Marpe’806 disclose a Video Frame Encoding and Decoding that is the same apparatus/method and computer program product as the invention of the ’701 Patent. *Id.* In addressing independent claims 1, 8, and 15, Examiner identified specific elements of Figure 2 of Marpe’806 that disclose each part of the ’701 Patent’s claim language, shown below:[4]

> categorizing a plurality of syntax elements of video content (e.g. 104 in fig. 2) into first and second categories (e.g. 113 in fig. 2 (i.e. regular or bypass)) based on a frequency of occurrence of the syntax elements in the video content (e.g. based on whether the bits corresponding to a bin string have a 50/50 chance to be 1 or 0) (paragraph [0054]);
>
> entropy coding symbols that correspond to the first category of syntax elements (e.g. 118 in fig. 2) and that have been subjected to a context update (e.g. 112 in fig. 2); and
>
> entropy coding symbols that correspond to the second category of syntax elements (e.g. 120 in fig. 2) and that have bypassed context updating (e.g. as illustrated in fig. 2, when switch 113 is connected to bypass, 112 context modeler is bypassed).

’701FH, 192.

Applicant responded to the rejection on April 30, 2014; no claims were amended. ’701FH, 227–31. Applicant argued that Marpe’806 does not teach the limitation “categorizing a plurality of syntax elements of video content into first and second categories based on a frequency of occurrence of the syntax elements in the video content” in each independent claim.” ’701FH, 232–34. Applicant argued that Marpe’806’s switch does not “categorize” the syntax element by

---

[4] Notably, though the two patents do not appear on their face to be related, Figure 2 of Marpe’806 is also disclosed in Marpe (EX-1006).

determining if the bin has a 50/50 split. *Id.* Applicant argued that Marpe'806 is categorizing bit stream bins, not syntax elements. *Id.*

Examiner issued a final rejection of the pending claims on June 4, 2014, again as anticipated by Marpe'806 after finding Applicant's arguments unpersuasive. '701FH, 243–45. Examiner found that Marpe'806's Figure 2 illustrated with respect to the disclosed "binary valued syntax element" and "non-binary valued syntax element" that syntax elements can be binary or non-binary valued and, therefore, the categorization is done on binarized syntax elements. '701FH, 245.

Applicant amended each of the pending independent claims on December 2, 2014, adding the language underlined below. Notably, Applicant did not contest that Marpe'806 disclosed the categorization or the entropy coding limitations (which are identical to limitations 1[c] and 1[d] of the issued '701 Patent).

1. (Currently Amended) A method comprising:

determining a frequency of occurrence threshold based on an expected frequency of occurrence of syntax elements in a bit stream;

categorizing a plurality of syntax elements of video content into first and second categories based on ~~a~~the frequency of occurrence threshold, wherein ~~of the~~ syntax elements ~~in the video content~~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 are categorized into the second category;

entropy coding symbols that correspond to the first category of syntax elements and that have been subjected to a context update; and

entropy coding symbols that correspond to the second category of syntax elements and that have bypassed context updating.

’701FH, 260.

Following the amendment, the Examiner allowed all pending claims without specifying reasons for the allowance. ’701FH, 274–78.

Additionally, Patent Owner (“PO”) had a substantially similar European patent application EP11811919.7, issued as EP2599315. *See* EX-1008. Both patents have the same titles, are issued to the same inventors, and have substantially similar claims, e.g.:

’701 Patent, Claim 1	EP2599315, Claim 1
1. A method comprising: … categorizing a plurality of syntax elements of video content into first and	1. A method comprising: categorizing a plurality of syntax elements of video content into first and

second categories based on the frequency of occurrence threshold,…; entropy coding symbols that correspond to the first category of syntax elements and that have been subjected to a context update; and entropy coding symbols that correspond to the second category of syntax elements and that have bypassed context updating.	second categories based on a frequency of occurrence…; entropy coding symbols that correspond to the first category of syntax elements and that have been subjected to a context update; and entropy coding symbols that correspond to the second category of syntax elements and that have bypassed context updating.

That application was rejected on May 23, 2018 based on CABAC, the examiner stating it "is well known in the art that CABAC can be used to code syntax elements in image and video coding." EX-1008, 96. The European Examiner found the "CABAC coder of figure 1 in [CABAC] discloses a first coding mode comprising context updating and a second coding mode that 'bypass[es] context updating.'"[5] *Id.*

The European Examiner further found that while CABAC "does not directly disclose that the selection of the 2 coding modes is based on a threshold of

---

[5] As discussed below, the coder in Figure 1 of CABAC is identical to the coder in Figure 2 of Marpe.

occurrences of syntax elements… it discloses that for the most frequently observed bins the context modeling updating is used, while the the [sic] bypass can be used for the less frequently observed bins." *Id.*

V. LEVEL OF ORDINARY SKILL

A POSITA at the time of the invention would have had: (1) a bachelor's degree in Electrical Engineering, Computer Engineering, Computer Science, or a comparable field of study such as Physics, and (2) approximately 2–3 years of practical experience with digital video and/or image processing or coding. More education could substitute for less experience, and vice versa. Saber, ¶¶23-25.

VI. CLAIM CONSTRUCTION

Claims terms in an IPR are construed "in accordance with the ordinary and customary meaning of such claim as understood by one of ordinary skill in the art and the prosecution history pertaining to the patent." 37 C.F.R. §42.100(b).

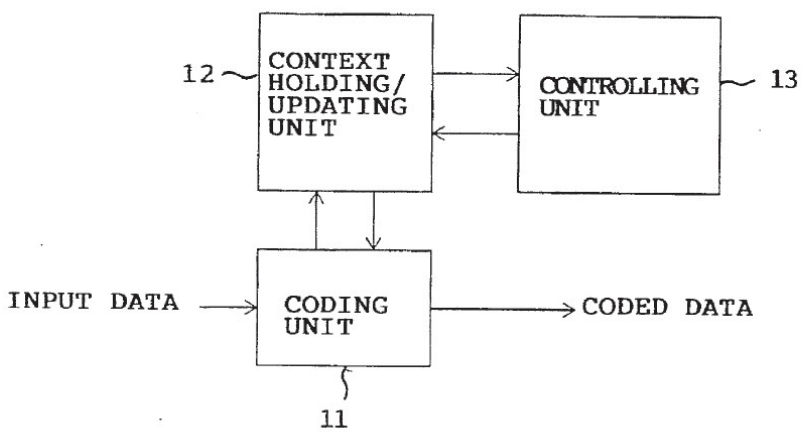
VII. DETAILED EXPLANATION OF GROUNDS

A. Ground 1: Claims 1, 6, 8, 13, 15, 20 are obvious over Murashita

1. Murashita

Murashita (EX-1007) "relates to data coding methods and devices" for "coding a series of symbols based on a multi-level arithmetic coding scheme." Murashita, 1:8–9, Abstract. Murashita's Figure 7A, below, depicts a "block diagram[] of a data coding device... according to a principle of the present invention." *Id.*, 6:24–26. Murashita's device "includes a coding unit [11] for coding the symbols based on frequency of occurrence of the symbols and orders of the symbols according to the frequency of occurrence." *Id.*, Abstract. Murashita's device includes "a context updating unit [12] for updating the frequency of occurrence and the orders upon receiving a control signal[.]" *Id.* And it includes "a controlling unit [13] for sending the control signal when one of the symbols is coded and a cumulative frequency is less than a predetermined value." *Id.*

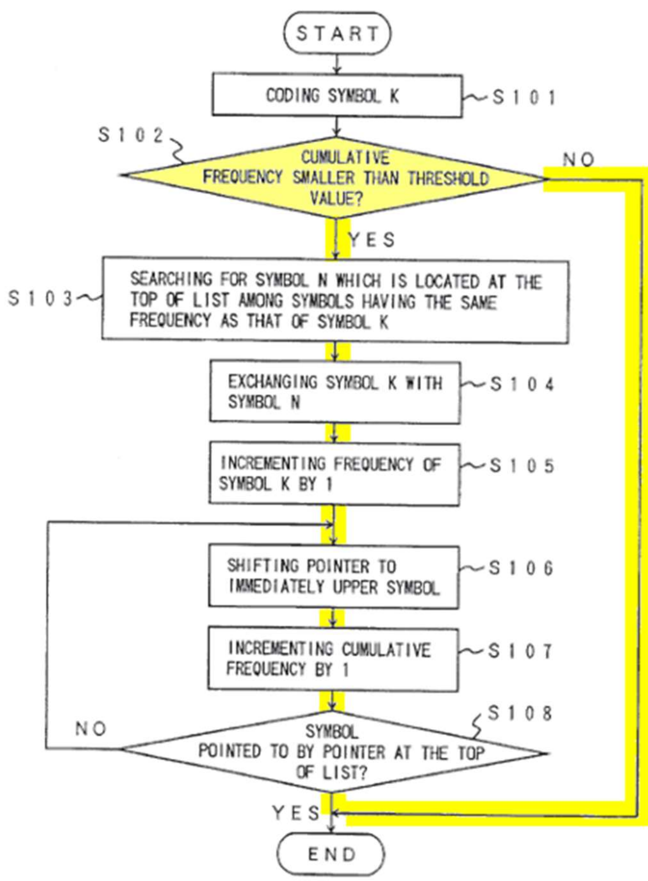
FIG. 7A



Id., Fig. 7A.

Murashita's Figure 10, below, further illustrates "a flowchart of a process of the coding device[.]" *Id.*, 6:4. "At a step S102, a check is made by the comparison unit 109 whether the cumulative frequency is less than [a] threshold value." *Id.*, 8:36–38. "[A] result of the check is indicated by the level of the comparison-output signal generated by the comparison unit 109." *Id.*, 8:40–42. "A low level signal indicates that the cumulative frequency is less than the threshold value. A high level signal indicates the contrary." *Id.*, 8:42–44. When the signal is "low," "the procedure proceeds to a step S103… to conduct the updating process." *Id.*, 8:36–48. When the signal is "high," "the procedure ends at that point" because "the updating process is no longer necessary." *Id.*, 9:1–6.

FIG. 10



Id., Fig. 10.

2. Claim 1 and Dependents

a. Claim 1

i. 1[pre]

If limiting, Murashita teaches 1[pre]. Saber, ¶80. Murashita discloses a "Data Coding/Decoding Device and **Method**." Murashita, Title; *see also id.*, 1:8–13 ("The present invention generally relates to data coding **methods** and devices…."), 5:15–21.

ii. 1[a]

Murashita teaches 1[a]. Saber, ¶¶81–104.

"***Syntax elements in a bit stream***"

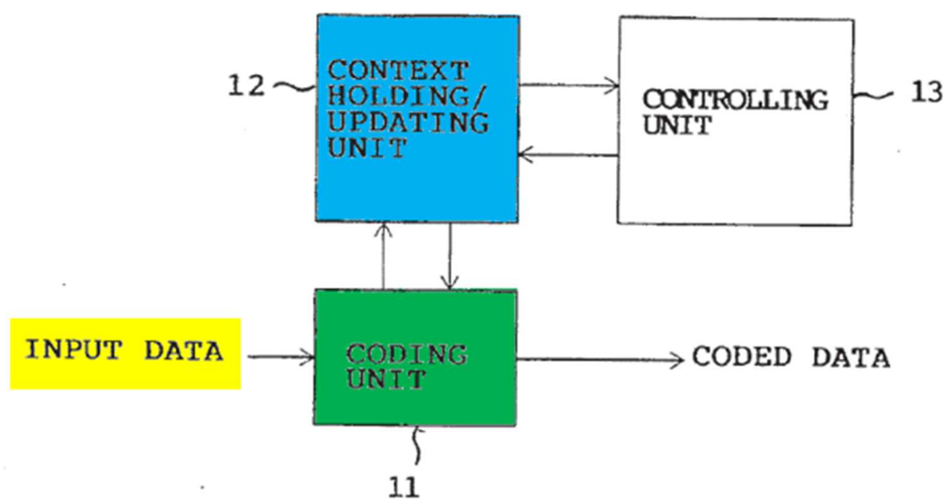
Murashita teaches receiving data symbols ("*syntax elements*") that are represented in an input data stream ("*in a bit stream*"). Saber, ¶¶87–93.

Data symbols are an example of "*syntax elements*" because they are elements of data represented in a bitstream. Saber, ¶¶82-83; *see also* '701Pat., 9:51–54 ("**syntax elements that may occur in a given bitstream (e.g., a bitstream associated with video content** being processed)"). The '701 Patent confirms that symbols are an example of syntax elements ("SEs") and often uses the terms interchangeably. *See, e.g.*, '701Pat., 9:51–63, 9:67–10:3, 13:7–12 ("**categorizing syntax elements** may include **categorizing symbols** having a frequency of occurrence below a threshold as first category symbols and

categorizing symbols having a frequency of occurrence above the threshold as second category symbols”), 13:12–15, 11:65–12:3 (“by **categorizing the syntax elements based on their relative frequencies** of occurrence, more complex operations may be performed with respect to **less frequently occurring symbols** and complex operations may be simplified or omitted with respect to **more frequently occurring symbols**”).

In Murashita’s Figure 7A, below, “input data” (highlighted yellow) represents “*syntax elements in a bit stream*.” Further, the “data coding device” includes a “coding unit 11 [highlighted green] [that] receives input data [highlighted yellow], and encodes it to generate coded data by using context data stored in the context holding/updating unit 12 [highlighted blue].” Murashita, 6:27–31.

FIG. 7A

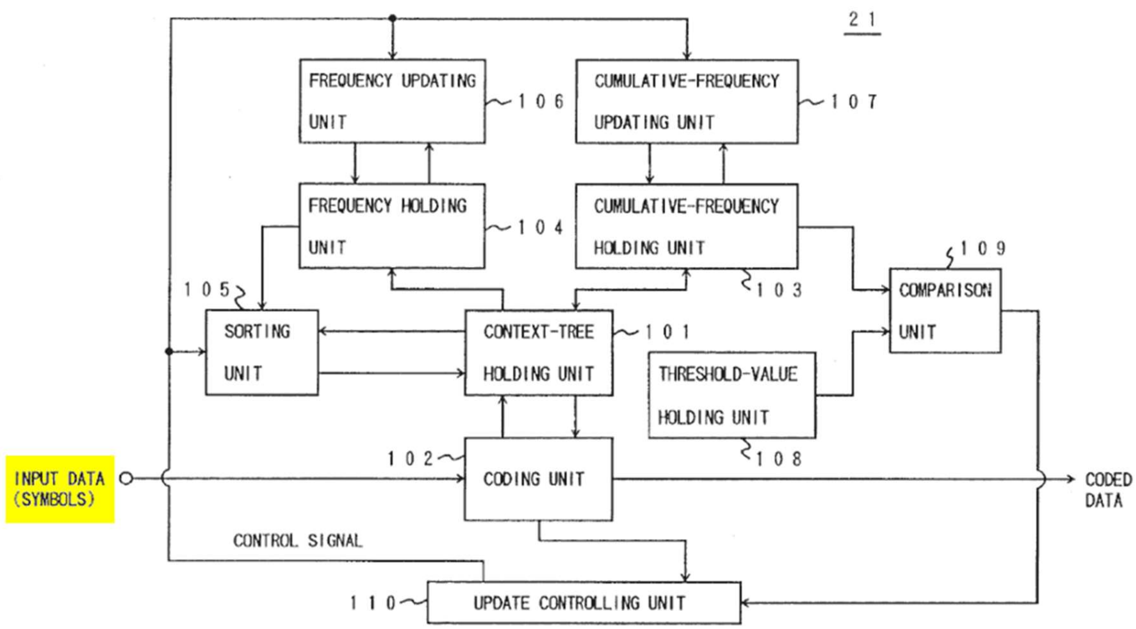


Id., Fig. 7A.

A POSITA would have understood that this input data forms a bit stream because it is a fundamental principle of digital systems. Saber, ¶¶84–85. A bit stream is the transmission of a continuous series of bits over a communications channel. *Id.*

Murashita explains that data symbols ("*syntax elements*") are represented in the input data bit stream. Referencing Figure 7A, above, Murashita explains that "[t]he context holding/updating unit 12 not only stores the context data, but also updates the context data according to the frequency of occurrence of **symbols appearing in the input data**." Murashita, 6:35–39; *see also id.*, 4:6–9 ("When a given **symbol appears in the input data**, the list in the dictionary should be rearranged according to the updated frequency of occurrence[.]"). This is depicted in Murashita's Figure 9, below, showing an "**INPUT DATA (SYMBOLS)**" input.

F I G. 9



Id., Fig. 9.

Murashita's Figure 4A and 1A, below, provide an exemplary list of symbols that can appear in the input data stream.

FIG. 1A

SYMBOL	:	FREQUENCY	:	RANGE
a	:	0.20	:	[0, 0.2)
b	:	0.10	:	[0.2, 0.3)
c	:	0.05	:	[0.3, 0.35)
d	:	0.15	:	[0.35, 0.5)
e	:	0.50	:	[0.5, 1)

Id., Fig. 1A.

FIG. 4A

ORDER OF FREQUENCY	SYMBOL	FREQUENCY	CUMULATIVE FREQUENCY
0	-----	00	327
1	00	10	317
2	01	7	310
3	02	4	306
⋮	⋮	⋮	⋮
n − 1	'M'	2	m2
n	'N'	1	m1
⋮	⋮	⋮	⋮
X	'A'	1	m0
⋮	⋮	⋮	⋮
254	FD	1	3
255	FE	1	2
256	FF	1	1
257	EOF	1	0

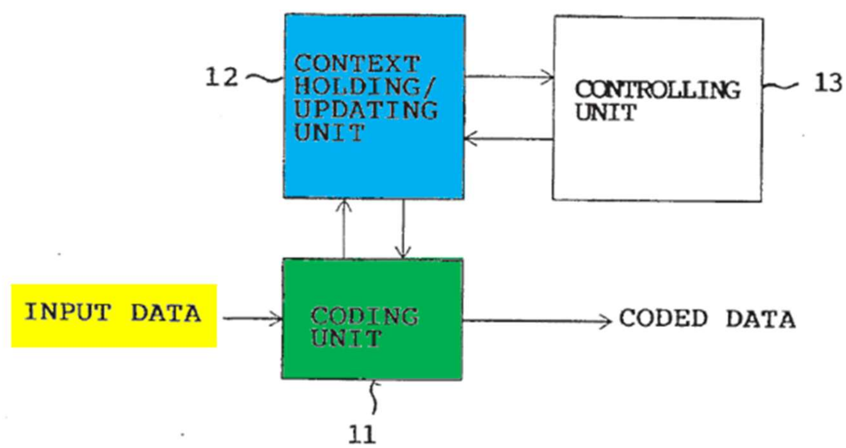
Id., Fig. 4A.

"***<u>Expected frequency of occurrence of syntax elements in a bit stream</u>***"

Murashita teaches identifying a frequency of occurrence ("*expected frequency of occurrence*") of symbols ("*of syntax elements*") in an input data stream ("*in a bit stream*"). Saber, ¶¶88–91.

Murashita explains that its system "includes a coding unit for coding the symbols based on **frequency of occurrence** ["*expected frequency of occurrence*"] **of the symbols** ["*of syntax elements in a bit stream*"] and orders of the symbols **according to the frequency of occurrence**." Murashita, Abstract, 5:22–35. Referencing Figure 7A, below, Murashita explains that "[t]he coding unit 11 [highlighted green] receives input data [highlighted yellow], and encodes it to generate coded data by using context data stored in the context holding/updating unit 12 [highlighted blue]. Here, the **context data includes frequency of occurrence** ["*expected frequency of occurrence*"], **cumulative frequency of occurrence** [also an "*expected frequency of occurrence*"], and an order according to the frequency of occurrence, **which are provided for each symbol** ["*of syntax elements in a bit stream*"] under given contexts. The context holding/updating unit 12 not only stores the context data, but also updates the context data according to the **frequency of occurrence of symbols** ["*expected frequency of occurrence of syntax elements*"] appearing in the input data." *Id.*, 6:29–39.

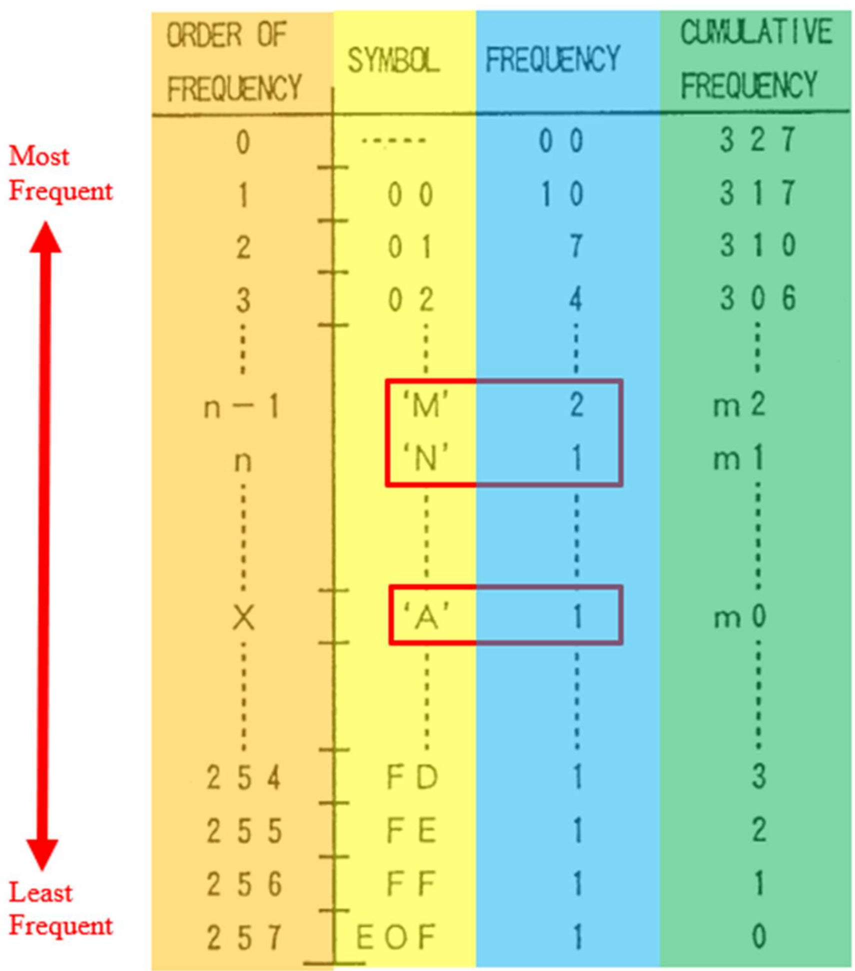
FIG. 7A



Id., Fig. 7A.

Murashita's Figures 4A–4C "are illustrative drawings for explaining an updating process of a dictionary used for recording and counting **frequency of each symbol** ["*expected frequency of occurrence of syntax elements*"]." *Id.*, 5:57–59. For instance, Figure 4A, shown below, provides a listing of symbols [highlighted yellow] (e.g., symbols 'M,' 'N,' and 'A'), their corresponding frequency of occurrence [highlighted blue] (e.g. 'M' occurs 2 times, and 'N' and 'A' each occur 1 time), and the cumulative frequency [highlighted green]. Saber, ¶90. "Symbols are arranged from the top of the list in descending order of frequency[,]" the ordering of symbols highlighted in orange. Murashita, 3:67–4:2; Saber, ¶90. "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." Murashita, 4:2–5.

FIG. 4A



Id., Fig. 4(a).

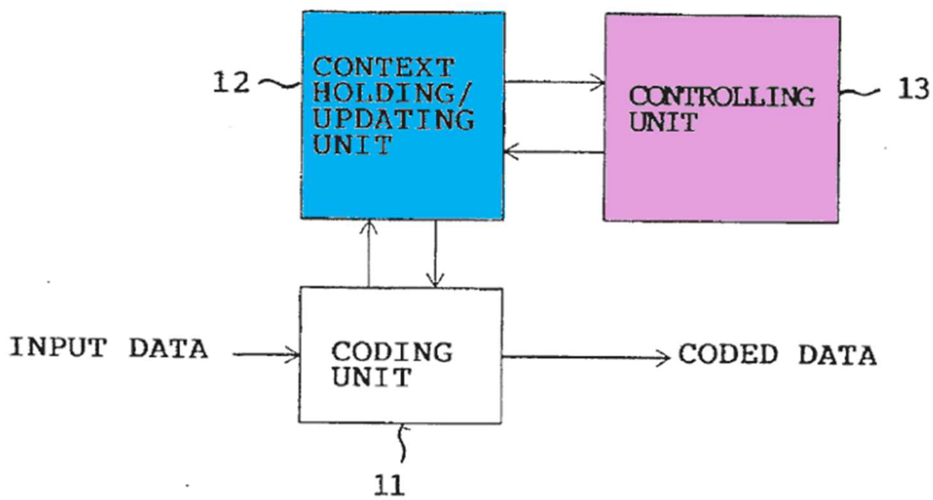
The frequency of occurrence of symbols (highlighted blue) and cumulative frequency of occurrence of symbols (highlighted green), shown above, teach "*expected frequency of occurrence of syntax elements in a bit stream*." Saber, ¶91.

"***Frequency of occurrence threshold***"

Murashita teaches a threshold value ("*frequency of occurrence threshold*"). Saber, ¶¶92–95.

Murashita explains, and shows in Figure 7A below, that its system includes a "controlling unit 13 [highlighted pink] [that] controls the context holding/updating unit 12 [highlighted blue] to update the context data. In doing so, the controlling unit 13 compares a predetermined **threshold value** ["*frequency of occurrence threshold*"] **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** ["*frequency of occurrence threshold*"], the controlling unit 13 controls the context holding/updating unit 12 to update the context data for the given context." Murashita, 6:39–47; *see also id.*, 6:57–64, 7:17–24, 7:40–57, 8:11–48, 9:1–48, 10:1–44, 10:65–11:4, 11:35–48, 12:1–23, Figs. 9–16.

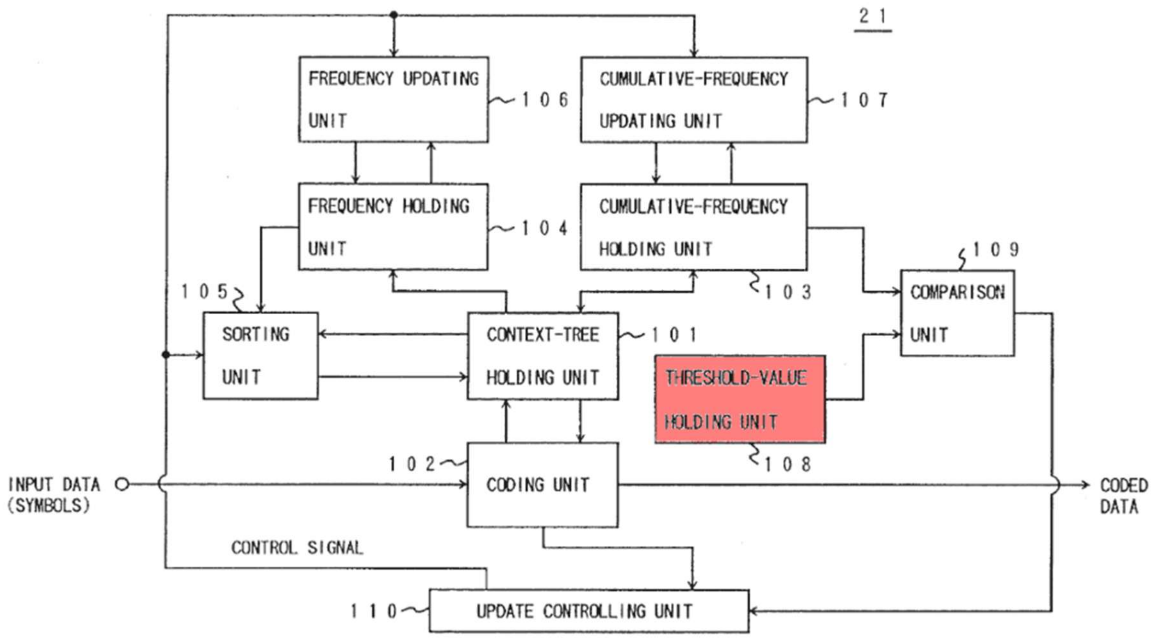
FIG. 7A



Id., Fig 7A.

Likewise, Murashita explains that its Figure 9, below, "shows a block diagram of the coding device 21. The coding device 21 include a context-tree holding unit 101, a coding unit 102, a cumulative-frequency holding unit 103, a frequency holding unit 104, a sorting unit 105, a frequency updating unit 106, a cumulative-frequency updating unit 107, a **threshold-value holding unit** 108 [highlighted red], a comparison unit 109, and an update controlling unit 110." *Id.*, 7:18–23. Murashita states that "**[t]he threshold-value holding unit 108** stores a **threshold value** ["*frequency of occurrence threshold*"] which is used in making a decision to stop the updating of the frequency, the cumulative frequency, and the context tree." *Id.*, 7:40–43.

FIG. 9

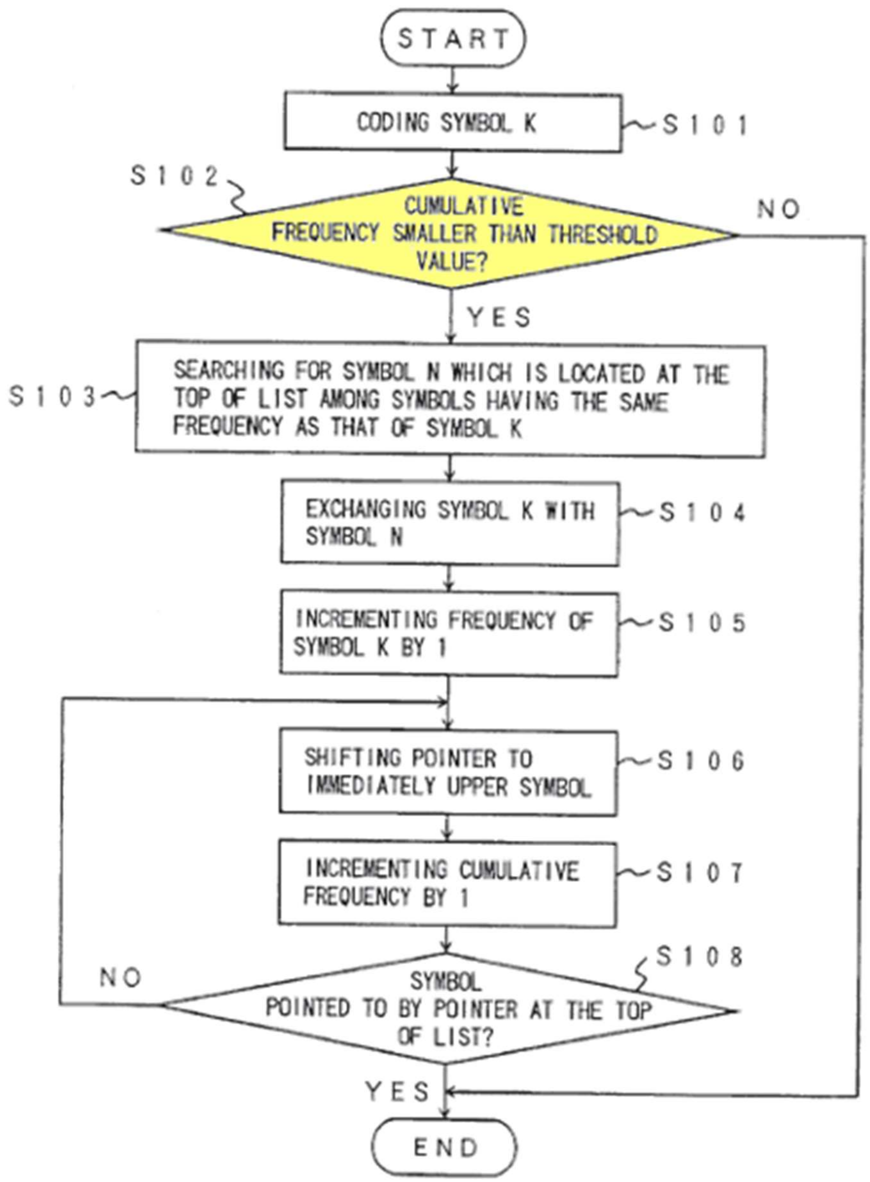


Id., Fig. 9.

Likewise, Murashita's Figure 10 (referencing components in Figure 9), below, is "a flowchart of the updating process of the coding device 21," which includes "a step S102 [highlighted yellow], [where] a check is made by the comparison unit 109 whether the cumulative frequency is less than the **threshold value** ["*frequency of occurrence threshold*"]. If it is, the procedure proceeds to a step S103. If it is not, the procedure ends. Here, a result of the check is indicated by the level of the comparison-output signal generated by the comparison unit 109.

A low level signal indicates that the cumulative frequency is less than the **threshold value**. A high level signal indicates the contrary." *Id.*, 8:32–44.

FIG. 10



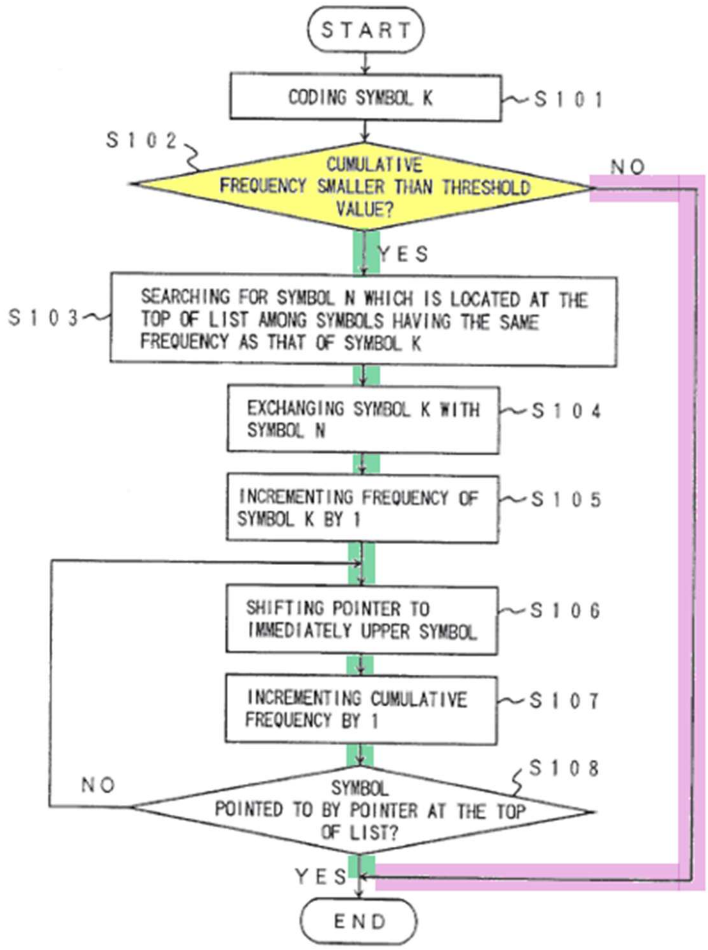
Id., Fig. 10.

"***<u>Based on an expected frequency of occurrence</u>***"

A POSITA would have understood that Murashita's threshold would be "***based on*** *[the] expected frequency of occurrence of syntax elements in a bit stream*." Saber, ¶¶96–104.

This is clear from the function of the threshold. Murashita explains that its "threshold value" identifies the "cumulative frequency of occurrence" for symbols at which to "update the context data." Murashita, 6:39–47. Depicted below in Murashita's Figure 10, context data is updated (S103–S108) if the cumulative frequency of occurrence is smaller than the threshold (green path), and context updating is bypassed (pink path) if the cumulative frequency of occurrence is greater than the threshold. Saber, ¶97. Because Murashita's threshold value indicates the frequency of occurrence of symbols ("*syntax elements*") at which context updating should occur, a POSITA would have understood that it would be based—at least in part—on an expected frequency of occurrence. *Id.*

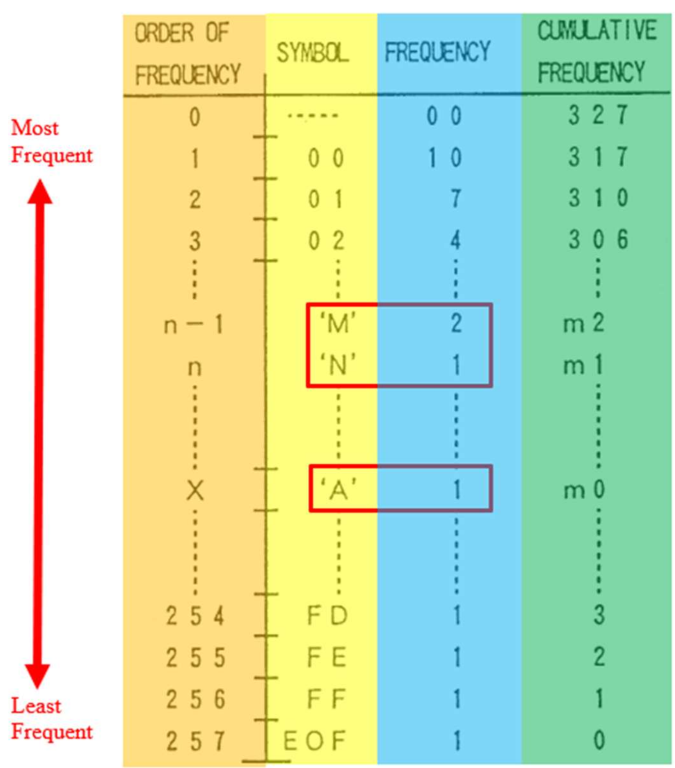
FIG. 10



Id., Fig. 10.

Murashita's threshold is not dynamically created, which indicates that there is an **expectation** for how many times Murashita's SEs will occur; otherwise, setting a threshold would be meaningless. Saber, ¶98. For example, in Figure 4A below, for the threshold to have any practical effect, its value would need to be in the range of 10 (if applied to the frequency of symbols) or in the range of 327 (if applied to the cumulative frequency of symbols). *Id*. A threshold of 500, for example, would be meaningless, as none of the symbols occur with that frequency or cumulative frequency. *Id.* Murashita's threshold is based on an **expected** frequency of occurrence of SEs. *Id.*

FIG. 4A



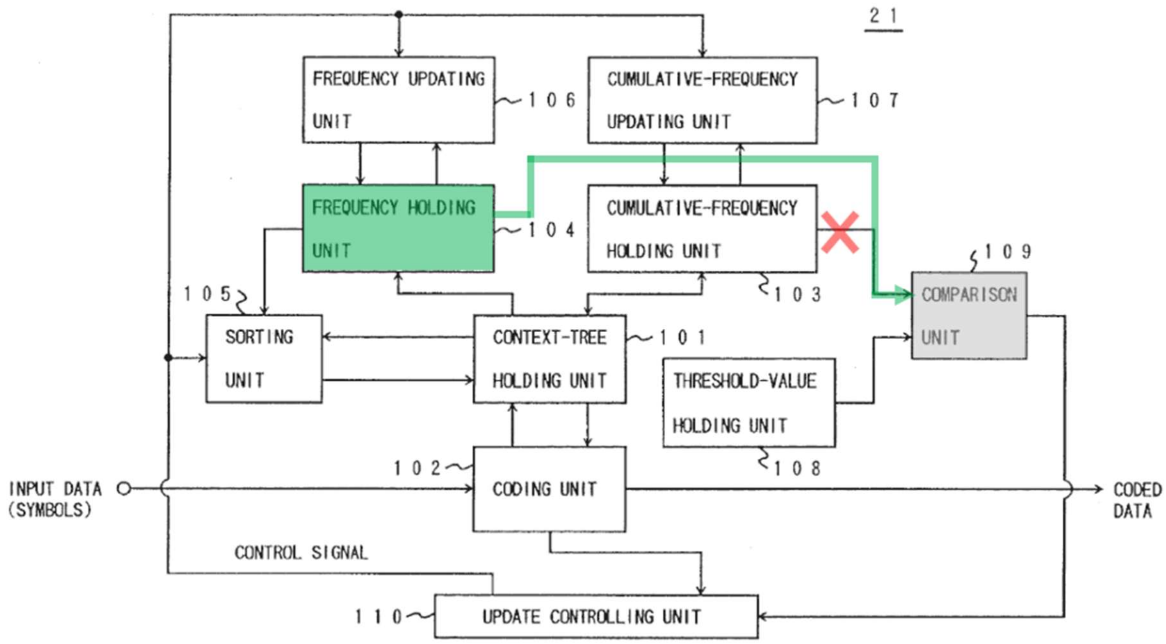
Murashita, Fig. 4(a).

Although Murashita discloses that its threshold is used for comparing whether a symbol's <u>cumulative</u> frequency of occurrence is at or below the threshold, a POSITA would have understood that such a cumulative frequency of occurrence is just another—albeit more complex— way of representing the claimed frequency of occurrence. Saber, ¶99. Referencing Figure 4(a), Murashita explains that "[t]he 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[.]" Murashita, 4:2-5. Accordingly, a POSITA would have understood that Murashita's cumulative frequency is based on and represents the expected individual frequency of occurrence of a symbol and thereby satisfies the claim language. Saber, ¶99. Further, the '701 patent explains that its threshold is based on the frequency of occurrence of symbols, plural. '701Pat, 9:57-63 ("Accordingly, for example, **a threshold frequency of occurrence may be defined for symbols**…."), 13:7-12. This is like Murashita's threshold that is based on the cumulative frequency of occurrence (i.e., also based on the frequency of occurrence of symbols, plural).

Regardless, it would have been obvious to base Murashita's threshold on and compare that threshold with the frequency of occurrence for each symbol (instead of the cumulative frequency), especially given both are calculated, updated, and stored in Murashita's system, and because it is simpler to calculate and use than the cumulative frequency of occurrence. Saber, ¶100; *see also* Murashita, Fig. 9, 7:18-39. Murashita explains that "in the coding/decoding methods of the related art, each time a symbol is supplied, the frequency of occurrence and ***the cumulative frequency are updated***, and symbols are ***sorted*** to be arranged in a descending order of frequency. ***Thus, a long processing time is required*** for treating a large number of symbols. This leads to a lower processing speed." Murashita, 5:4-10. Murashita sought to "simplify the steps and make the processing faster." *Id.*, 1:49-52, 5:11-21. A POSITA would have understood that by basing the threshold on and comparing the threshold to a symbol's frequency of occurrence, the symbols would no longer need to be sorted and the cumulative frequency would no longer need to be calculated, making the processing time faster. Saber, ¶101.

Accordingly, referencing Murashita's Figure 9, below, a POSITA would have found it obvious to feed a symbol's frequency of occurrence (stored in the Frequency Holding Unit 104) (highlighted green) into the Comparison Unit (109) (highlighted gray), rather than the cumulative frequency of occurrence (stored in the Cumulative Frequency Holding Unit 103), for comparison with the threshold. *Id.*, ¶101.

F I G. 9



Murashita, Fig. 9; Saber, ¶102.

Further, Murashita's system is capable of computing both the frequency of occurrence and the cumulative frequency of occurrence of symbols, thus a POSITA would have understood basing the threshold value on "*an expected frequency of occurrence*." *See* Murashita, Fig. 4A–4C, 7, 9. A POSITA would therefore have known how to determine a frequency of occurrence threshold based on the expected frequency of occurrence of syntax elements in a bit stream. Saber, ¶103–04.

iii. 1[b]

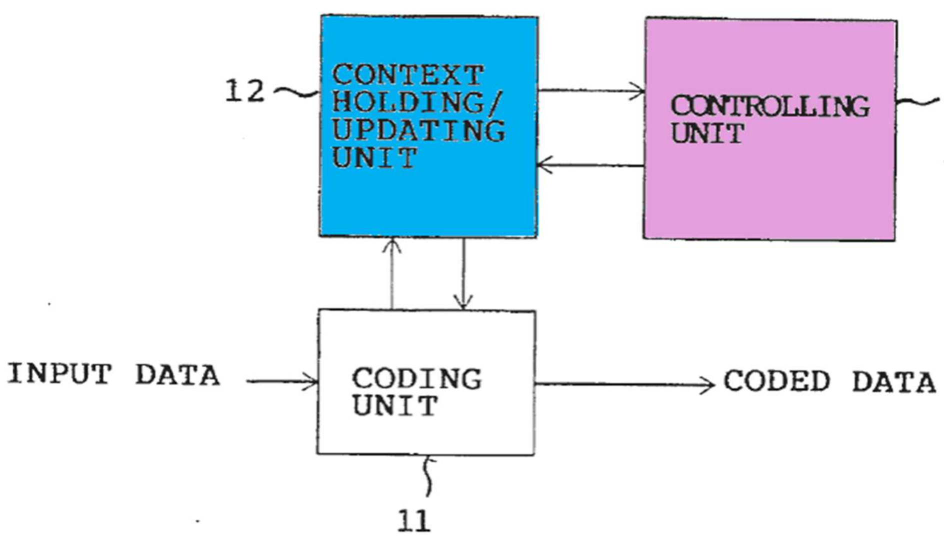
Murashita teaches categorizing symbols from input data ("*categorizing a plurality of syntax elements of video content*") based on the threshold ("*based on the frequency of occurrence threshold*"), where symbols that occur with a frequency greater than the threshold are categorized into a category ("*wherein syntax elements which occur greater than the frequency of occurrence threshold are categorized into the second category*") and symbols that occur with a lesser frequency are categorized into another category ("*syntax elements which occur less than the frequency of occurrence are categorized into the first category*"). Saber, ¶¶105–09.

As discussed in 1[a], Murashita teaches "*determining a frequency of occurrence threshold*." *See* §VII.A.2.a.ii. Murashita's cumulative frequency of occurrence disclosure teaches the use of the frequency of occurrence for each symbol. Saber, ¶105.

Referencing Figure 7A, below, Murashita explains that "[t]he controlling unit 13 [highlighted pink] controls the context holding/updating unit 12 [highlighted blue] to update the context data. In doing so, the controlling unit 13 **compares a** predetermined **threshold value** ["*frequency of occurrence threshold*"] **with a cumulative frequency of occurrence for all the symbols** ["*occur[ence]*" of "*syntax elements*"] registered under a given context. Then, **only when the**

cumulative frequency is <u>smaller than</u> the threshold value [“*syntax elements which occur less than the frequency of occurrence are categorized into the first category*”], the controlling unit 13 controls the context holding/updating unit 12 to **update the context data for the given context**.” Murashita, 6:39–47.

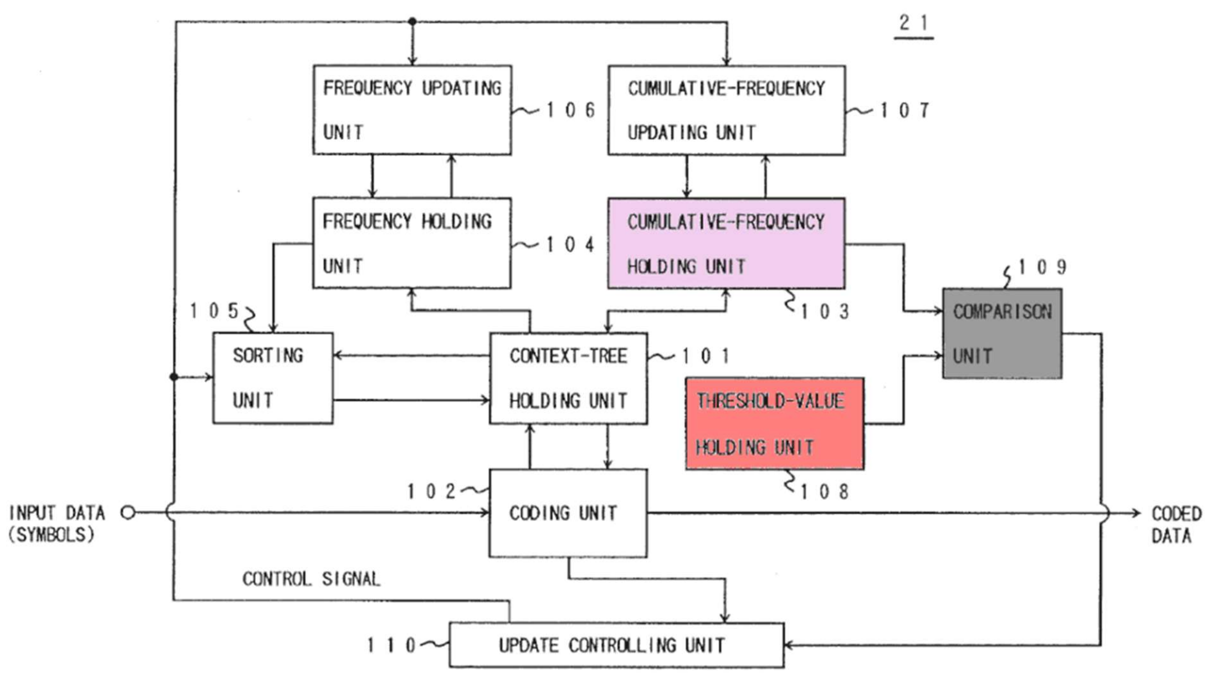
FIG. 7A



Id., Fig. 7A.

This is confirmed in Murashita’s Figure 9, below. Murashita explains that “[t]he threshold-value holding unit 108 [highlighted red] stores a threshold value which is used in making a decision to stop the updating of the frequency, the cumulative frequency, and the context tree. The comparison unit 109 [highlighted gray] **compares the threshold value** [“*frequency of occurrence threshold*”] stored in the threshold-value holding unit 108 **with the cumulative frequency** [“*occur[ence]*” of “*syntax elements*”] stored in the cumulative-frequency holding unit 103 [highlighted pink]. Then, the comparison unit 109 outputs a comparison-output signal, **which is a <u>low level</u> signal when the cumulative frequency is <u>less than</u> the threshold value** [“*syntax elements which occur less than the frequency of occurrence are categorized into the first category*”]**, and which is a <u>high level</u> when the cumulative frequency is <u>not less than</u> the threshold value** [“*syntax elements which occur greater than the frequency of occurrence threshold are categorized into the second category*”].” *Id.*, 7:40–57; *see also id.*, 8:11–19.

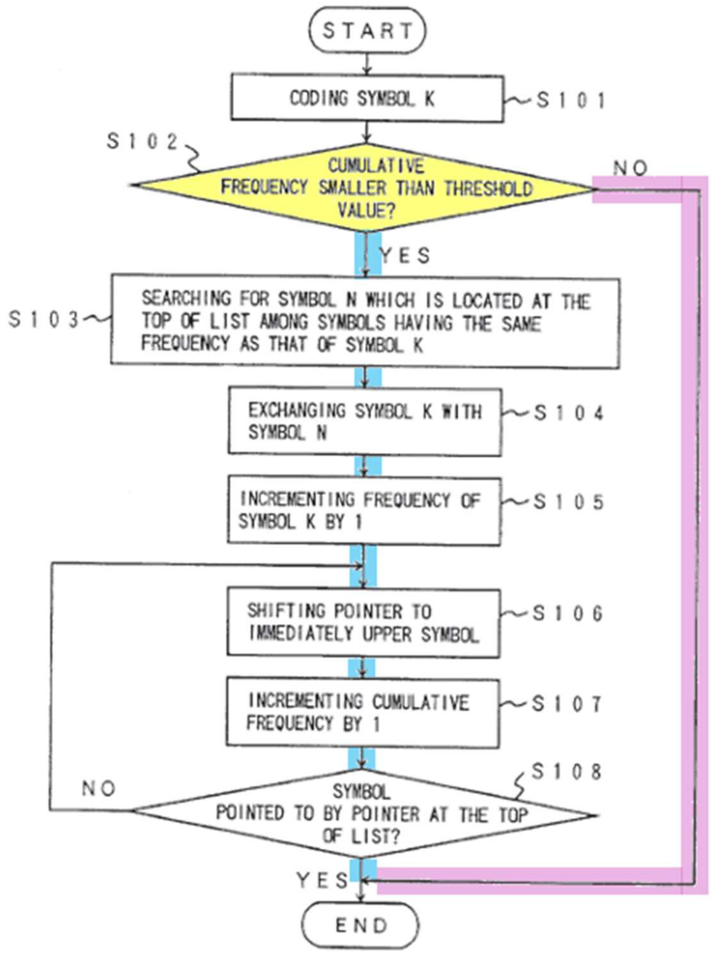
F I G. 9



Id., Fig. 9.

This is likewise shown in Murashita's Figure 10, below (referencing components in Figure 9). "FIG. 10 shows a flowchart of the updating process of the coding device 21." *Id.*, 8:33–34. "At a step S102 [highlighted yellow], **a check is made by the comparison unit 109 whether the cumulative frequency** ["*occur[ence]*" *of* "*syntax elements*"] **is less than the threshold value** ["*frequency of occurrence threshold*"]. If it is, the procedure proceeds to a step S103. If it is not, the procedure ends. Here, **a result of the check is indicated by the level** of the comparison-output signal generated by the comparison unit 109. **<u>A low level</u> signal indicates that the cumulative frequency is <u>less</u> than the threshold value** ["*syntax elements which occur less than the frequency of occurrence are categorized into the first category*"]. **A <u>high level</u> signal indicates the contrary** ["*syntax elements which occur greater than the frequency of occurrence threshold are categorized into the second category*"]. Upon receiving the low level comparison-output signal, the update controlling unit 110 controls the sorting unit 105, the frequency updating unit 106, and the cumulative-frequency updating unit 107 to conduct the updating process." *Id.*, 8:36–48.

FIG. 10



Id., Fig. 10.

Therefore, Murashita's system categorizes symbols into categories based on its threshold ("*categorizing a plurality of syntax elements of video content into first and second categories based on the frequency of occurrence threshold*")—**category 1**: update context data (comparison unit 109 outputs low level signal) when cumulative frequency of occurrence is below threshold ("*syntax elements* *which occur less than the frequency of occurrence are categorized into the first category*"); **category 2**: do not update context data (comparison unit 109 outputs high level signal) when cumulative frequency of occurrence is above threshold ("*wherein syntax elements which occur greater than the frequency of occurrence threshold are categorized into the second category*").

Murashita discloses categorizing symbols for video content ("*categorizing a plurality of syntax elements of video content*"). Saber, ¶109. Murashita explains that "data used in computers has come to include a **wider variety of data types such as character codes, vector information, images, etc. With this widening variety of data types**, a rapid increase has been seen in the amount of data treated in computers." Murashita, 1:15–19. Because video is a sequence of images, Murashita anticipates applying its invention to video content. Saber, ¶109. To the extent this is not disclosed, a POSITA would have found it obvious to apply Murashita's invention to video content, given the express disclosure of a wide variety of data types, including images, as well as its arithmetic coding disclosures. *Id.* A POSITA would have understood that arithmetic coding could be used to code symbols from video content. *Id.* Arithmetic coding is one of the entropy coding techniques adopted in the H.264/AVC video compression standard. *Id.*

iv. 1[c]

Murashita teaches using arithmetic coding to code symbols (“*entropy coding symbols*”) whose context has been updated (“*that have been subjected to a context update*”) because the cumulative frequency of occurrence was below the threshold (“*symbols that correspond to the first category of syntax elements*”). Saber, ¶¶110–21.

“***<u>Entropy Coding Symbols</u>***”

Murashita discloses “*entropy coding symbols*.” Saber, ¶¶111–16.

Arithmetic coding is a form of entropy coding. Saber, ¶111. The inventors of the ’701 patent admit this. EX-1020, [0007] (“**The entropy coding mechanisms, such as** Huffman coding, **arithmetic coding**, exploit statistical probabilities of symbol[s]….”).

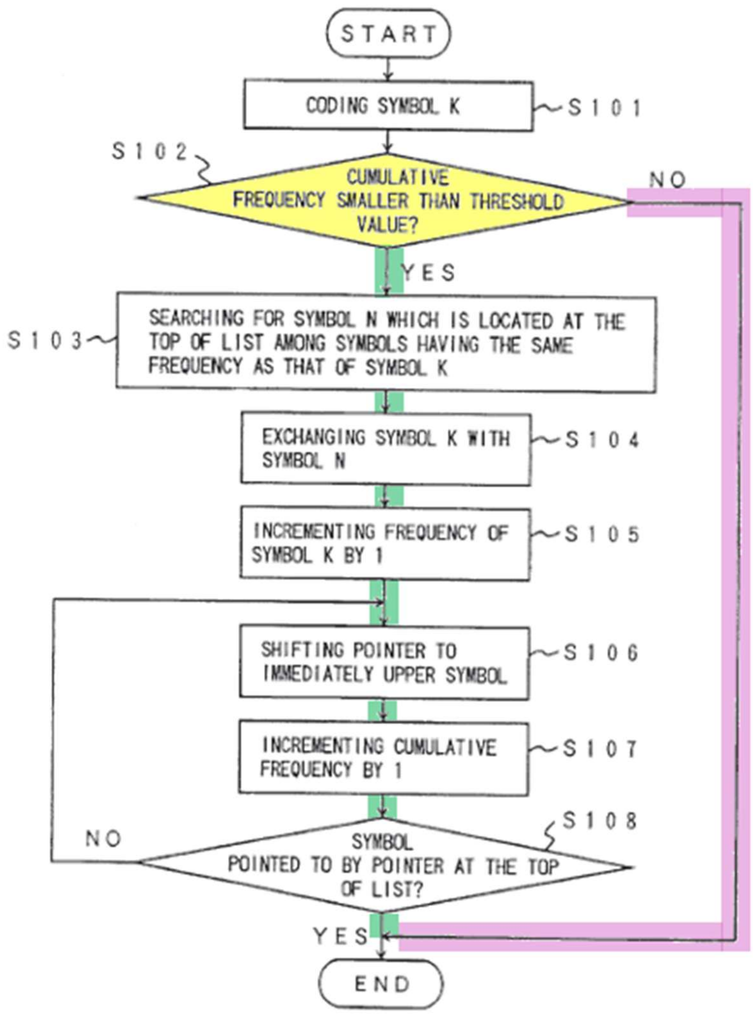
Murashita teaches using the arithmetic entropy coding technique. Murashita explains that “there are two are **two types of the arithmetic coding** method, i.e., a binary arithmetic coding method and a multi-level (more than two) arithmetic coding method;” and both were well known in the prior art. *Id.*, 1:53–62. Murashita discloses “a device for coding a series of symbols based on a **multi-level arithmetic coding scheme**.” *Id.*, 5:23–25. Murashita describes “Figs 1A through 1C [to be] illustrative drawings for explaining a **multi-level arithmetic coding scheme**[.]” *Id.*, 5:48–50, Figs. 1A–1C. Murashita also claims “[a] device for coding a series of symbols based on a **multi-level arithmetic coding scheme**…." *Id.*, claim 1; *see also id.*, claims 2–22. Murashita explains that in this scheme, "redundancies included in the data are eliminated as much as possible." *Id.*, 1:20–24. This is accomplished by utilizing the statistical distribution of symbols to render an optimal code, yielding shorter code words for frequent type symbols and longer code words with infrequent symbols. Saber, ¶¶112-15.

"***Symbols that correspond to the first category of syntax elements and that have been subjected to a context update***"

Murashita teaches entropy coding "*symbols that correspond to the first category of syntax elements and that have been subjected to a context update*." Saber, ¶¶116–21.

Murashita's Figure 10, below, "shows a flowchart of the updating process of the coding device 21." Murashita, 8:33–34. "At a step S102 [highlighted yellow], **a check is made by the comparison unit 109 whether the cumulative frequency is less than the threshold value**. **If it is** ["*symbols that correspond to the first category of syntax elements*"], **the procedure proceeds to a step S103** ["*that have been subjected to a context update*"]. If it is not, the procedure ends." *Id.*, 8:35–39. In other words, when symbols correspond to the first category (cumulative frequency is less than the threshold value), the symbols go through the green pathway below. Saber, ¶117.

F I G. 1 0

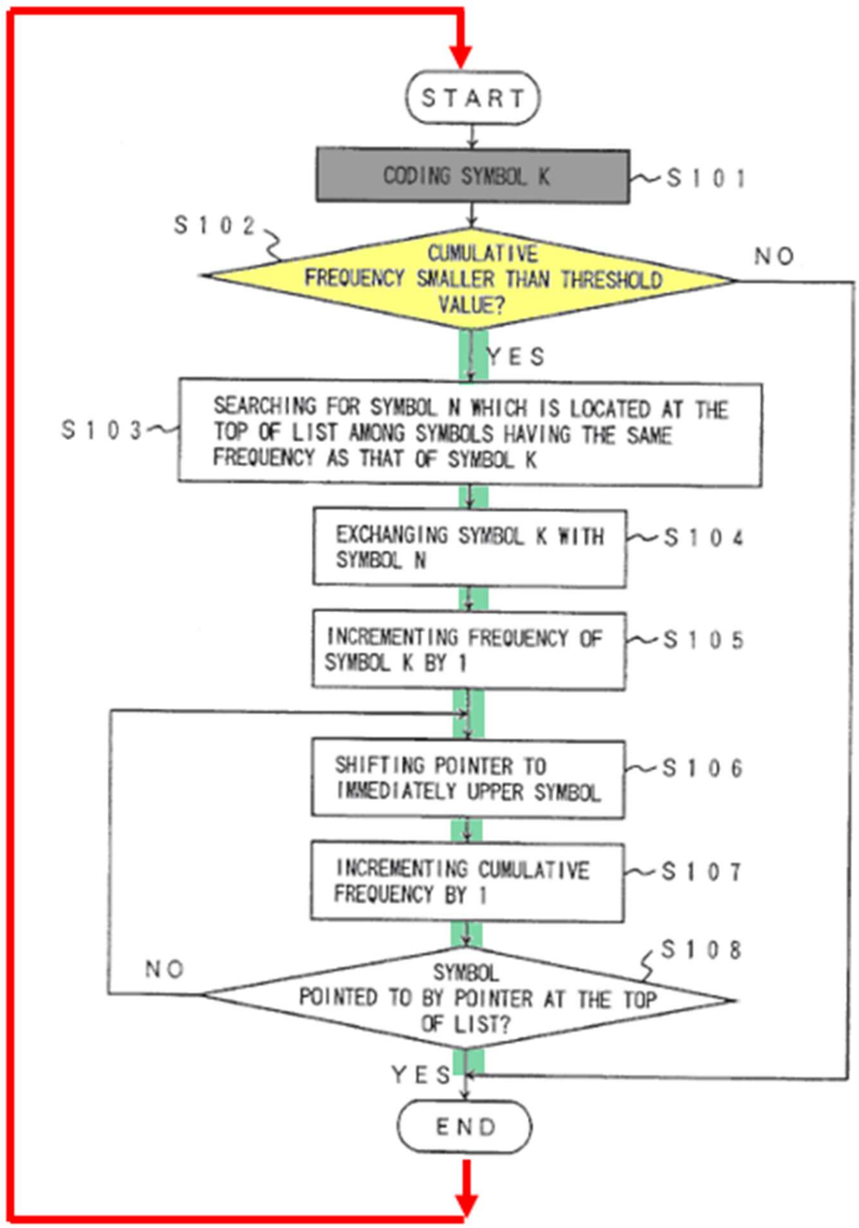


Id., Fig. 10.

The green pathway above—steps S103–S108—constitutes receiving a context update. Saber, ¶118; Murashita, 8:49–67; *see also* Murashita, 8:40–48 ("Upon receiving the **low level** comparison-output signal, the update controlling unit 110 controls the sorting unit 105, the frequency updating unit 106, and the cumulative-frequency updating unit 107 **to conduct the updating process**."), 6:39–47 ("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**."), 7:54–8:20 ("In doing so, the update controlling unit 110 uses the comparison-output signal supplied thereto **for determining whether to initiate the updating processes** of the frequency, the cumulative frequency, and the context tree…. When the cumulative frequency is **less than** the threshold value, the update controlling unit 110 controls the frequency updating unit 106, the cumulative-frequency updating unit 107, and the sorting unit 105 **to carry out a respective updating process**."), Figs. 7A, 9.

The symbols are coded (at S101, highlighted gray) using the updated context (i.e., from steps S103–S108), given the cycle of Figure 10 repeats itself, as shown below. Saber, ¶121. Thus, Murashita teaches "***entropy coding symbols*** *that correspond to the first category of syntax elements and* ***that have been*** *subjected to a context update*." *Id.*

FIG. 10



Murashita, Fig. 10.

v. 1[d]

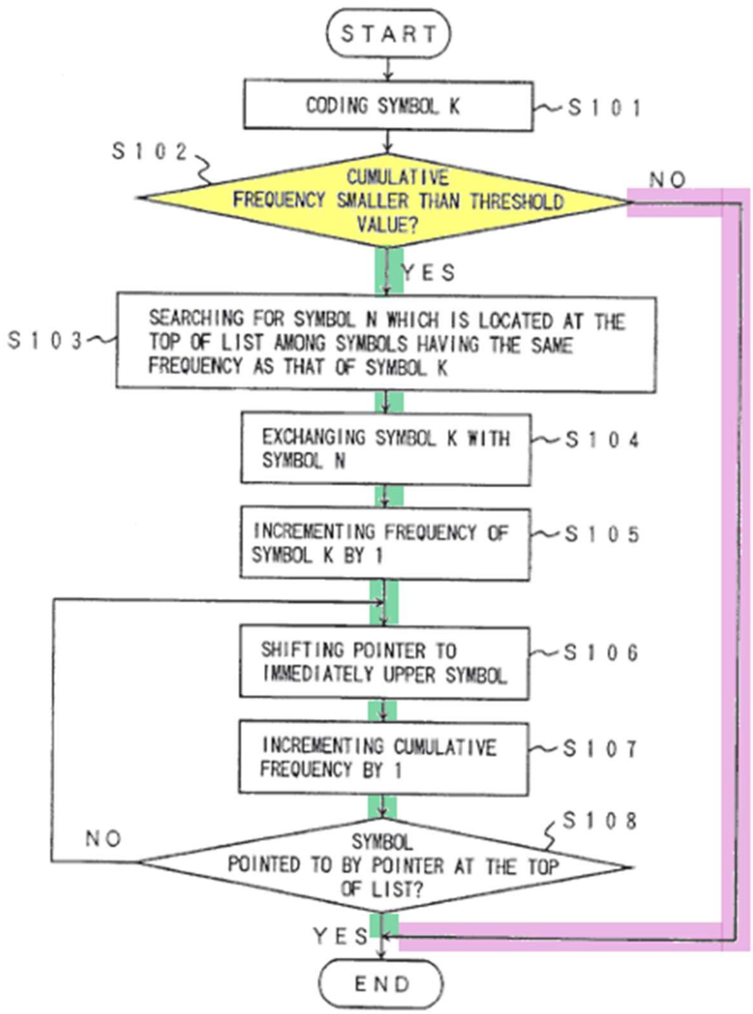
Murashita teaches using arithmetic coding to code symbols ("*entropy coding symbols*") whose context updating has been bypassed ("*that have bypassed context* *updating*") because the cumulative frequency of occurrence was above the threshold ("*symbols that correspond to the second category of syntax elements*"). Saber, ¶¶122–25.

As explained previously, Murashita discloses "*entropy coding symbols*" and "*context updat[ing]*." *See* §VII.A.2.a.iv.

Further, Murashita teaches entropy coding "*symbols that correspond to the second category of syntax elements and that have bypassed context updating*." Saber, ¶¶124–25.

Murashita's Figure 10, below, "shows a flowchart of the updating process of the coding device 21." *Id.*, 8:33–34. "At a step S102 [highlighted yellow], **a check is made by the comparison unit 109 whether the cumulative frequency is less than the threshold value**. If it is, the procedure proceeds to a step. **If it is not** ["*symbols that correspond to the second category of syntax*"], **the procedure ends** ["*that have bypassed context updating*"]." Murashita, 8:35–39. In other words, when symbols correspond to the second category (cumulative frequency is greater than the threshold value), the symbols go through the pink pathway below. Saber, ¶124.

FIG. 10



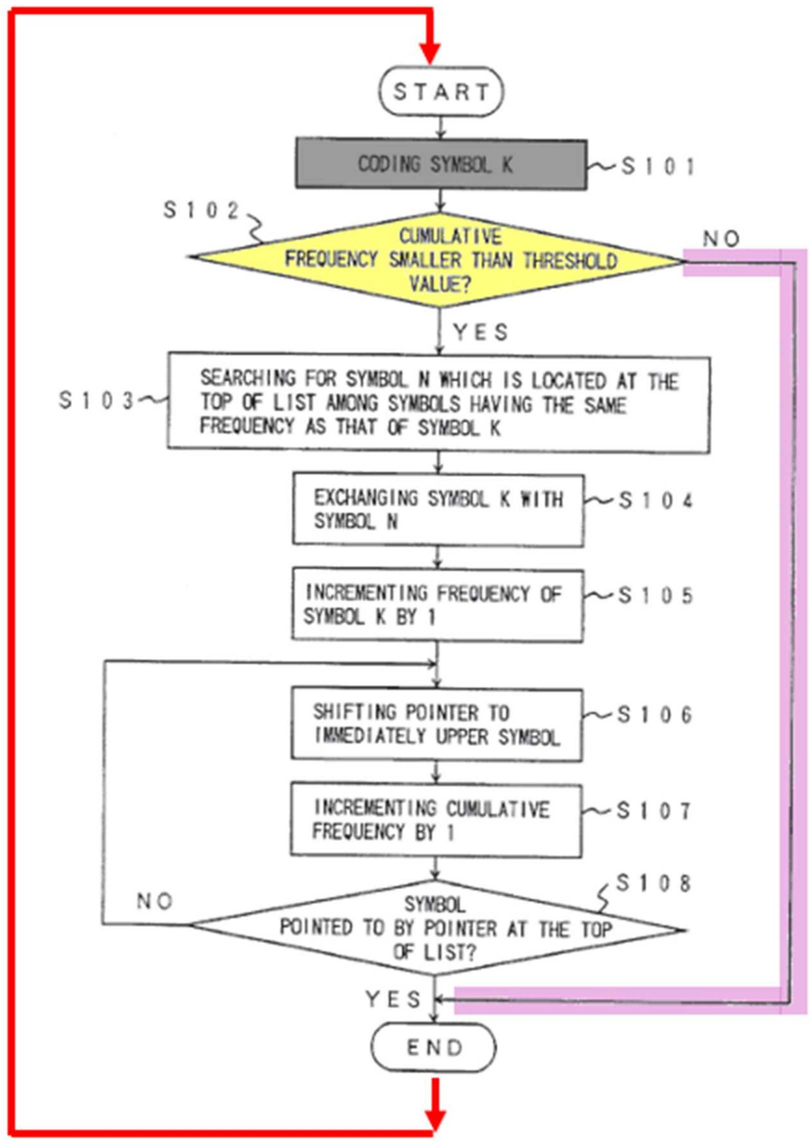
Murashita, Fig. 10.

The pink pathway above bypasses context updating. Saber, ¶125; Murashita, 8:49–67; *see also id.*, 8:28–32 ("When the comparison-output signal indicates that **the cumulative frequency is not less than the threshold value, the updating process described above will not take place**."), 9:1–6 ("As mentioned, if it turns out at the step S102 that **the cumulative frequency is not less than the threshold**

value, the procedure ends at that point. This is because the update controlling unit 110 receives the high level comparison output signal, and decides that the **updating process is no longer necessary**.")

In this case, the symbols are coded (at S101, highlighted gray) using the old and non-updated context (i.e., steps S103–S108 having been bypassed), given the cycle of Figure 10 repeats itself, as shown below. Saber, ¶125. Thus, Murashita teaches "***entropy coding symbols*** *that correspond to the second category of syntax elements and* ***that have*** *bypassed context updating*." *Id.*

F I G. 1 0



Murashita, Fig. 10.

b. Claim 6

Murashita teaches claim 6. Saber, ¶¶126–27.

Claim 6 is substantially similar to claim 1[b], as shown below, except that the threshold is "*predefined*." *Id. See* §VII.A.2.a.iii.[6]

Claim 1[b]	**Claim 6**
categorizing a plurality of syntax elements… based on the frequency of occurrence threshold, wherein syntax elements which occur greater than the frequency of occurrence …;	A method according to claim 1 wherein categorizing the plurality of syntax elements comprises categorizing the plurality of syntax elements based on a relationship of the frequency of occurrence of the syntax elements to a **predefined** threshold.

Murashita explicitly discloses using a pre-defined threshold. For example, with respect to Figure 7A, Murashita explains that "[t]he controlling unit 13 controls the context holding/updating unit 12 to update the context data. In doing so, the controlling unit 13 compares a **predetermined threshold value** ["*predefined threshold*"] with a cumulative frequency of occurrence for all the symbols registered under a given context. Then, only when the cumulative

---

[6] Petitioners do not concede that claim 6 is valid under 35 U.S.C. §112, given, for example, it is unclear what "the frequency of occurrence of the syntax elements" in claim 6 refers to.

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." Murashita, 6:39–47; *see also id.*, Abstract ("**predetermined** value"), 6:58–61 ("**predetermined** threshold value"), claims 1–22 ("**predetermined** value").

3. Claim 8 and Dependents

a. Claim 8

i. 8[pre]

If limiting, Murashita teaches 8[pre]. Saber, ¶128.

Murashita discloses a "Data Coding/Decoding **Device** and Method." Murashita, Title; *see also id.*, 1:8–13 ("The present invention generally relates to data coding methods and **devices**, and particularly relates to a method and a **device** which encodes input data according to a variable-length coding scheme based on frequency of data series occurrence in past data, and decodes coded data based on a history of past decoded data."), 5:15–21. Murashita's Figure 2 discloses "a block diagram which carries out the statistical type coding method by incorporating the inter-symbol dependency." *Id.*, 2:62–65. Murashita states that "[w]hen a large amount of data is treated, redundancies included in the data are eliminated as much as possible to compress the data, which leads to a reduction in memory volumes and an increase in data-transmission speed." *Id.*, 1:20–23. A POSITA reading Murashita would have known that the most common way to implement the claimed invention would be to use an apparatus with one or more processors and memory executing computer program code. Saber, ¶128; *see also* Murashita, 7:4–5 (referencing "**processing** speed"), 1:20–24 (referencing "**memory** volumes").

ii. 8[a]

Murashita teaches 8[a] for the reasons described for 1[a]. *See* §VII.A.2.a.ii; Saber, ¶129.

iii. 8[b]

Murashita teaches obvious 8[b] for the reasons described for 1[b]. *See* §VII.A.2.a.iii; Saber, ¶130.

iv. 8[c]

Murashita teaches obvious 8[c] for the reasons described for 1[c]. *See* §VII.A.2.a.iv; Saber, ¶131.

v. 8[d]

Murashita teaches obvious 8[d] for the reasons described for 1[d]. *See* §VII.A.2.a.v; Saber, ¶132.

b. Claim 13

Murashita teaches claim 13 for the reasons described for claims 6 and 8. *See* §§VII.A.2.b, VII.A.3.a; Saber, ¶133.

4. Claim 15 and Dependents

a. Claim 15

i. 15[pre]

If limiting, Murashita teaches 15[pre] for the reasons described for 8[pre]. *See* §VII.A.3.a.i; Saber, ¶134.

ii. 15[a]

Murashita teaches 15[a] for the reasons described for 1[a]. *See* §VII.A.2.a.ii; Saber, ¶135.

iii. 15[b]

Murashita teaches 15[b] for the reasons described for 1[b]. *See* §VII.A.2.a.iii; Saber, ¶136.

iv. 15[c]

Murashita teaches 15[c] for the reasons described for 1[c]. *See* §VII.A.2.a.iv; Saber, ¶137.

v. 15[d]

Murashita teaches 15[d] for the reasons described for 1[d]. *See* §VII.A.2.a.v; Saber, ¶138.

b. Claim 20

Murashita teaches claim 20 for the reasons described for claims 6 and 15. *See* §§VII.A.2.b, VII.A.4.a; Saber, ¶¶139–41.

B. Ground 2: Claims 1–20 are obvious over Murashita in view of Marpe

1. Marpe

Marpe (EX-1006) "relate[s] to… binarization and arithmetic coding schemes for use in video coding applications." Marpe, [0003]. Marpe's Figure 1 shows "a general view of [a] video coder environment to which the present invention could be applied." *Id.*, [0036]. "A picture 10 is fed to a video precoder 12" where "syntax elements are generated[.]" *Id.*, [0036]–[0037]. "The precoder 12 transfers the syntax elements to a final coder stage 14 which is [an] entropy coder and explained in more detail with respect to FIG. 2." *Id.*, [0038]. "Each syntax element is a data value having a certain meaning in the video signal bit stream that is passed to the entropy coder. The entropy coder 14 outputs a final compressed video bit stream." *Id.* The entropy coders work "by exploiting the redundancy in the information contained in the input bit stream." *Id.*, [0005].

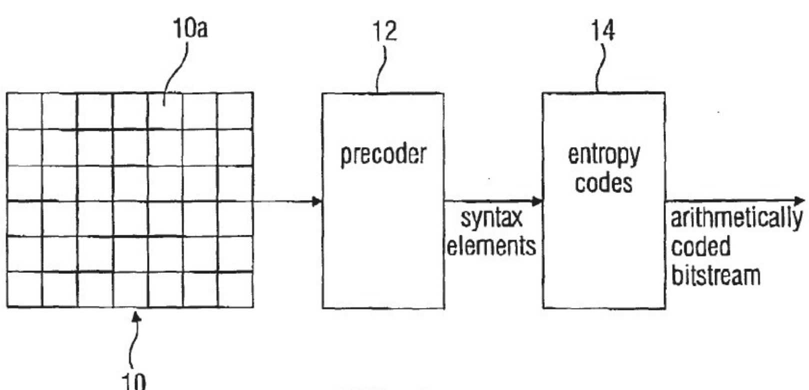


FIG. 1

Id., Fig. 1.

Marpe's Figure 2, below, "shows the arrangement for coding the syntax elements into the final arithmetic code bit stream." *Id.*, [0039]. Figure 2 depicts Marpe's very well-known CABAC engine, based on binarization and binary arithmetic entropy coding. *Id.*, Fig. 2, [0021], [0114]–[0117]. Saber, ¶148. In the CABAC engine, "[t]he coding arrangement 100 is divided into three stages 100a, 100b, and 100c." Marpe, [0039]. "The first stage 10[0]a is the binarization stage." *Id.*, [0040]. "Stage 100b is a context modelling stage" with a "switch 113 [that] pass[es] the bits or bins of the bin string at binarization stage output 108 to either the context modeler 112 or to a bypass coding input terminal 116 of stage 100c, thereby bypassing context modeler 112." *Id.*, [0044]. The decision to undergo or bypass context modeling can be made based on how "frequently observed [the] bins" are. *Id.*, [0016]. "The third stage 100c of coding arrangement 100 is the arithmetic coding stage," which includes a "regular coding engine 118" and a "bypass coding engine 120." *Id.*, [0049].

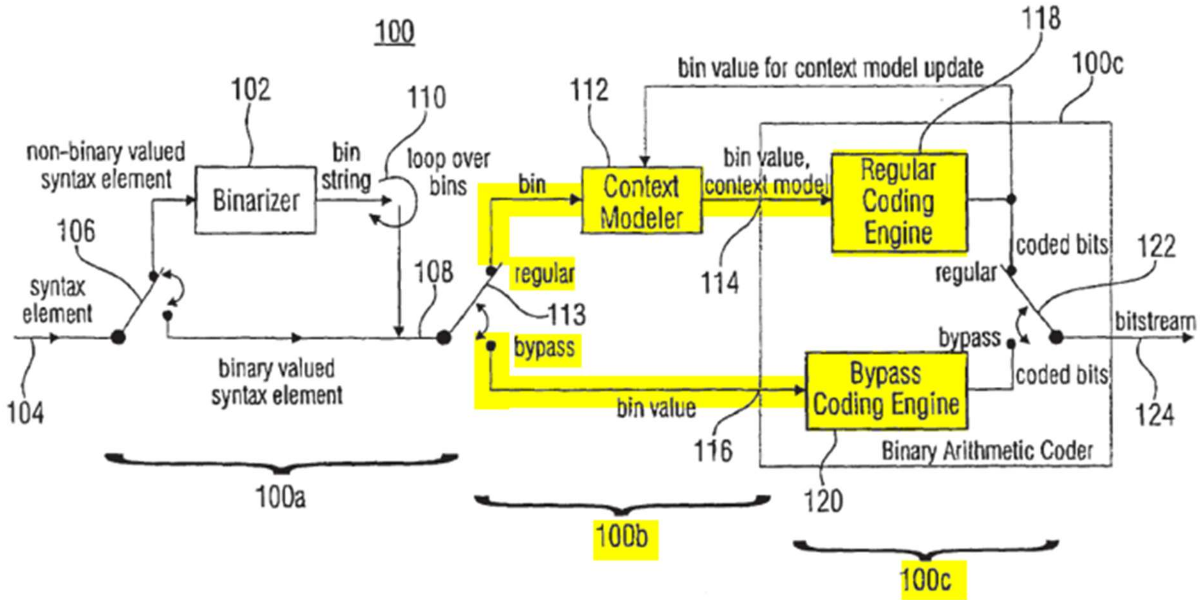


FIG. 2

Id., Fig. 2.

2. Motivation to Combine Murashita and Marpe

A POSITA considering the teachings of Murashita would have also considered the teachings of Marpe, because they are analogous art. Saber, ¶¶149–54. Both references are in the same field of endeavor, directed to coding/decoding techniques utilizing entropy coding methods such as variable-length coding and arithmetic coding. *Id.*; Murashita, Title, Abstract, 5:15–43; Marpe, Title, Abstract, [0003], [0010], [0014]. Both references are reasonably pertinent to the problem addressed in the ’701 Patent. Saber, ¶149. The ’701 Patent explains that there was a market desire to “maintain high levels of capabilities in the most efficient manner possible” for content consumption and communication, and that its invention therefore sought to "enable the provision of efficient processing in the area of video coding and decoding… to provide a high level of capability[.]" '701Pat., 1:34–51, 1:54–60. Likewise, Murashita and Marpe sought to provide gains in coding efficiency for content communication, thereby increasing capabilities. Murashita, 5:4–21 (explaining that "in the coding/decoding methods of the related art,... a long processing time is required for treating a large number of symbols" which "leads to a lower processing speed[,]" and that a "specific object of the present invention [is] to provide… a faster processing speed."), 5:36-43 ("According to the present invention... the updating process can be stopped when sufficient data is collected… which permits a higher processing speed"); Marpe, [0010] (explaining that it sought to provide a "coding scheme" that "enable[s] effective compression of data values… with moderate computational overhead"), [0043] (explaining that its invention "enabl[es] very efficient binary arithmetic coding"), [0048]; Saber, ¶150.

A POSITA would have been motivated to combine Murashita and Marpe for efficiency gains. Specifically, a POSITA would have been motivated to swap Murashita's "multi-level arithmetic coding" engine for Marpe's very well-known and more efficient CABAC engine, based on binary arithmetic coding. Saber, ¶151. Murashita explains that "there are two types of the arithmetic coding method, i.e., a **binary arithmetic coding** method and a **multi-level (more than**

two) arithmetic coding method." Murashita, 1:53–55. Murashita utilizes a "multi-level arithmetic coding scheme" (*e.g., id.*, 5:22–24), without explaining why multi-level arithmetic coding is advantageous or more efficient than binary arithmetic coding. Instead, Murashita focuses on the efficiencies that can be gained by solving a separate issue with "the related art, [where] **each** time a symbol is supplied, the frequency of occurrence and the cumulative frequency are updated, and symbols are sorted to be arranged in a descending order of frequency," leading to "long processing time" and "lower processing speed." *Id.*, 5:4–10. Murashita introduces a solution that only "carr[ies] out the updating of the frequency of occurrence and the orders" (i.e., "updat[ing] the context data") when a symbol's "cumulative frequency is smaller than [a] threshold value." *Id.*, 5:30–43, 6:39–47.

A POSITA would have been motivated to build on Murashita's efficiency gains by swapping out its "multi-level arithmetic coding" engine for Marpe's very well-known and more efficient CABAC engine, which is based on a binary arithmetic coding scheme rather than multi-level. Saber, ¶152; Murashita, 5:22–24; Marpe, Fig. 2, [0021], [0114]–[0117]. Marpe explains that "[t]he advantage of using a binary arithmetic coding engine instead of a m-ary arithmetic[7] coder… is that the complexity of the probability estimation is reduced since the determination of the probabilities for the two possible bit values can be defined by just one probability estimation value. Adaptive m-ary arithmetic coding, for m>2, would be in general a computationally complex operation requiring at least two multiplications for each symbol to encode as well as a number of fairly complex operations to perform the update of the probability estimation." Marpe, [0015]. Marpe further explains that it is advantageous that "binary arithmetic coding enables context modeling on a sub-symbol level," unlike m-ary arithmetic coding. *Id.*, [0016]. Marpe explains that by using "binarization" and "binary arithmetic coding instead of m-ary arithmetic coding," "[a] very effective compression of data values may be achieved." Marpe, [Abstract], [0014]–[0016].

A POSITA would have had a reasonable expectation of success in swapping out Murashita's multi-level arithmetic coding engine for Marpe's CABAC engine. Saber, ¶153. This is illustrated below, where Marpe's CABAC engine (Marpe, Fig. 2) (outlined in orange) replaces Murashita's Coding Unit 11 and Context

---

[7] M-ary arithmetic coding is equivalent to Murashita's multi-level arithmetic coding. Saber, ¶152.

Holding/Updating Unit 12 (both highlighted orange). *Id.* For this combination, a POSITA would have understood that Murashita's Controlling Unit 13 (highlighted red) would control switches 113 and 122 in Marpe's CABAC engine to dictate whether to "bypass[] context modeler 112" (Marpe, [0044], [0051])—i.e., whether to utilize the green or pink pathway. Saber, ¶153. This would be an easy implementation, given that Murashita's Controlling Unit 13 already indicates whether to bypass "updat[ing] the context data" by outputting a "low" or "high" signal (Murashita, 6:40–47, 7:40–57, 8:10–20). Saber, ¶153.

FIG. 7A

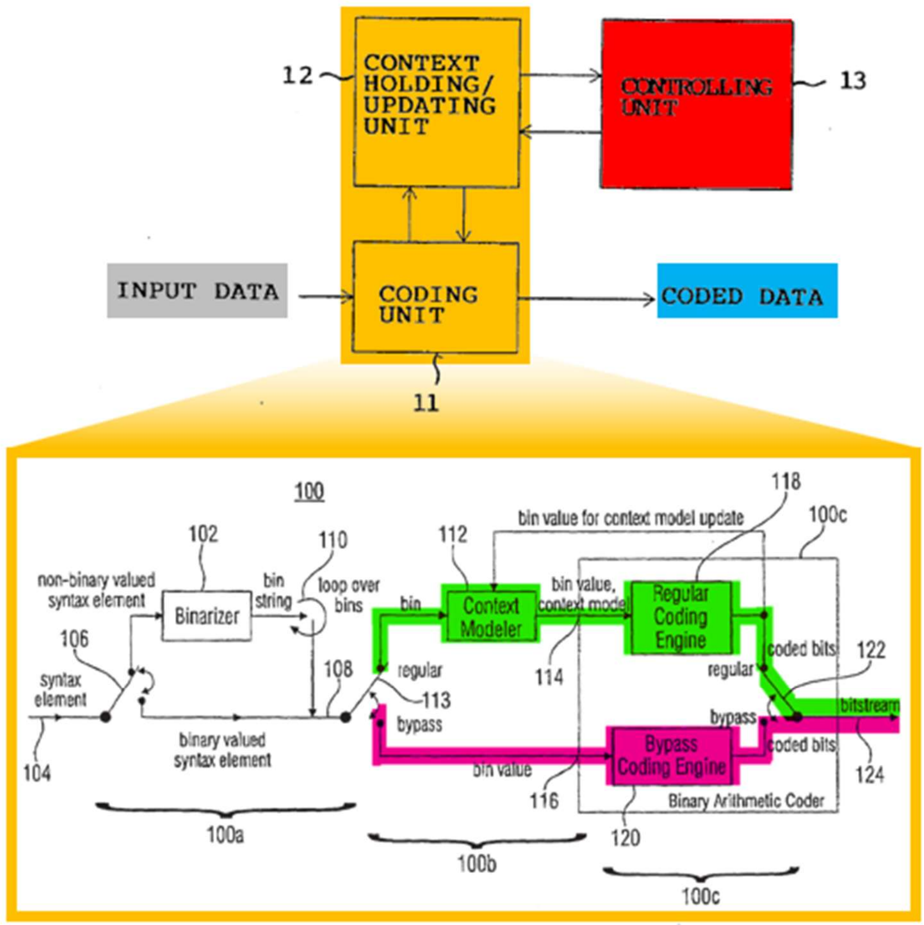


FIG. 2

Murashita, Fig. 7A; Marpe, Fig. 2; Saber, ¶153.

The combined Murashita/Marpe system would expectedly operate as follows. Saber, ¶154. First (Step 1), the combined system would "receive[] input data" (highlighted gray) (Murashita, 6:29), which consists of SEs for the reasons described previously (*see* §VII.A.2.a.ii). Saber, ¶154. Second (Step 2), Murashita's

Controlling Unit 13 (highlighted red) would "compare[] a predetermined threshold value with a cumulative frequency of occurrence for all the symbols registered under a given context" to signify whether to "update the context data for the given context" (Murashita, 6:39–47). Saber, ¶154. This would, in turn, output "a low level signal when the cumulative frequency is less than the threshold value" and a "high level [signal] when the cumulative frequency is not less than the threshold value" (Murashita, 7:40–57, 8:10–20). Saber, ¶154. Third (Step 3), the input data would undergo Marpe's CABAC process (Marpe, Fig. 2), using the control signals from Step 2 to indicate to switches 113 and 122 which pathway (green or pink) to take, thereby undergoing or bypassing context modeling. Saber, ¶154. Fourth (Step 4), the combined system would output the resulting "coded data" (highlighted blue) (Murashita, 6:29–31). Saber, ¶154.

By combining Murashita and Marpe, Murashita's system would become more efficient, thereby providing "faster processing speed[s]." (Murashita, 5:11–13, 5:19–21). Saber, ¶154.

3. Claim 1 and Dependents

a. Claim 1

i. 1[pre]

Murashita teaches 1[pre] for the reasons described above. *See* §VII.A.2.a.i; Saber, ¶155.

ii. 1[a]

Murashita teaches 1[a] for the reasons described above. *See* §VII.A.2.a.ii; Saber, ¶156.

iii. 1[b]

Murashita teaches 1[b] for the reasons described above. *See* §VII.A.2.a.iii; Saber, ¶157.

iv. 1[c]

Murashita teaches 1[c] for the reasons described above. *See* §VII.A.2.a.iv; Saber, ¶158.

To the extent it is argued that Murashita by itself does not teach 1[c], a POSITA would have been motivated to combine the teachings of Murashita and Marpe, to improve Murashita's coding technique by utilizing Marpe's binary arithmetic coding and binarization engine (i.e., CABAC engine), for the reasons previously described. *See* §VII.B.2. Murashita/Marpe teaches 1[c]. Saber, ¶159.

Murashita/Marpe teaches using binary arithmetic coding to code SEs ("*entropy coding symbols*") whose context has been updated ("*that have been subjected to a context update*") because the cumulative frequency of occurrence was below the threshold ("*symbols that correspond to the first category of syntax elements*"). Saber, ¶¶160–67.

"***Entropy Coding Symbols***"

Marpe teaches "*entropy coding symbols*." Saber, ¶¶160–62. As explained above, data symbols are an example of SEs. *See* §VII.A.2.a.ii.

Marpe's Figure 1 depicts entropy coding SEs. Referencing Figure 1, Marpe explains that "[t]he precoder 12 transfers the **syntax elements** to a final coder stage 14 which is a [sic] **entropy coder** and explained in more detail with respect to FIG. 2. Each **syntax element** is a data value having a certain meaning in the video signal bit stream that is passed to the **entropy coder**. The **entropy coder 14** outputs a final compressed video bit stream." Marpe, [0038].

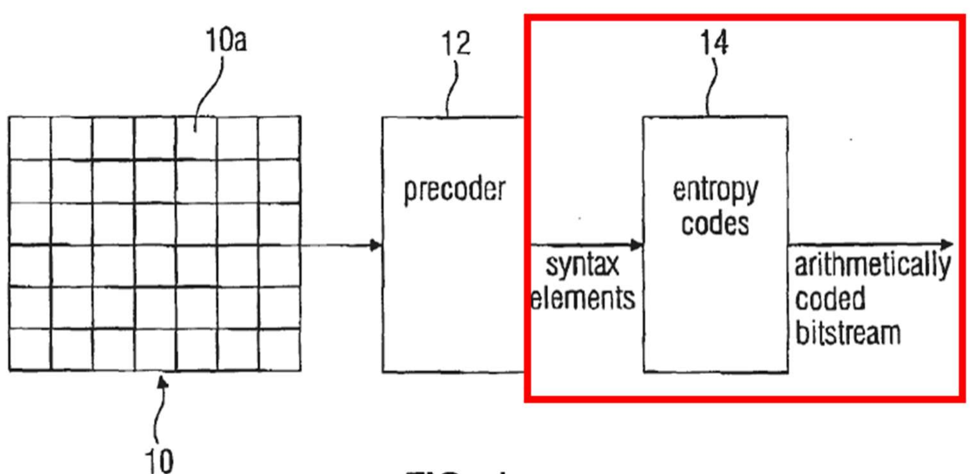


FIG. 1

Marpe, Fig. 1.

Marpe's entropy encoder is shown in more detail in Figure 2, below, where SEs (e.g., data symbols) that are "already in a wanted binarized form" are "directly pass[ed]" to the context modelling and arithmetic coding stages. *Id.*, [0041],

[0044], [0049]. And "non-binary valued syntax elements" are mapped to bin strings through binarization, and subsequently arithmetic coded. *Id.*

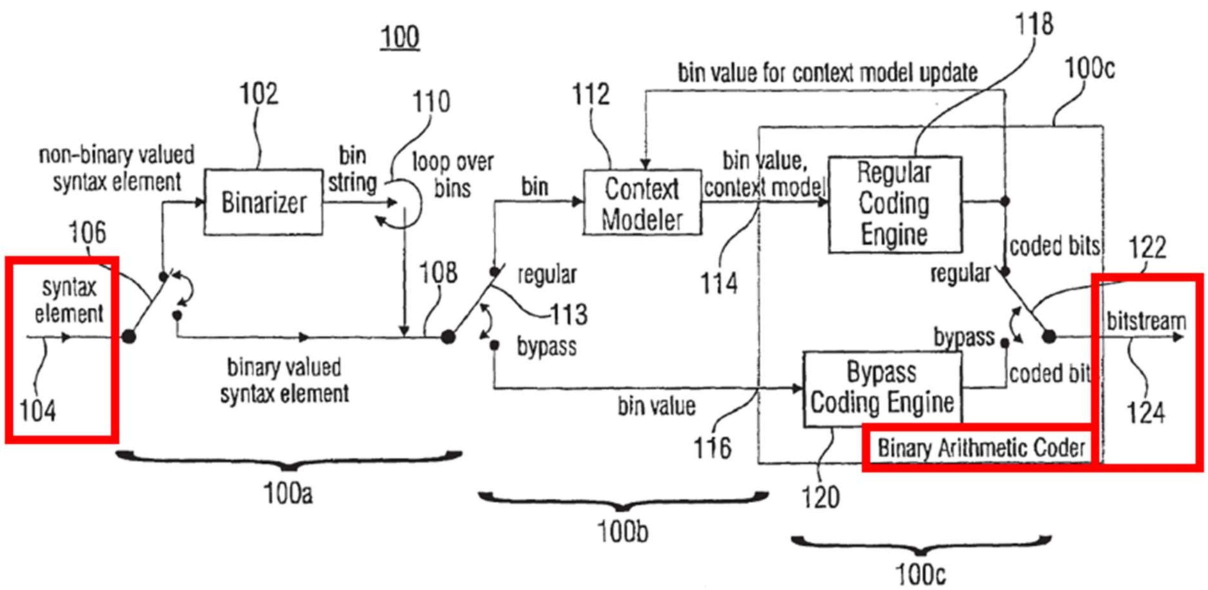


FIG. 2

Marpe, Fig. 2.

"***<u>Symbols that correspond to the first category of syntax elements and that have been subjected to a context update</u>***"

Marpe discloses entropy coding "*[s]ymbols that correspond to the first category of syntax elements and that have been subjected to a context update*." Saber, ¶¶163–67. This is depicted in Marpe's Figure 2, below, as the green "regular" pathway. *Id.*, ¶163.

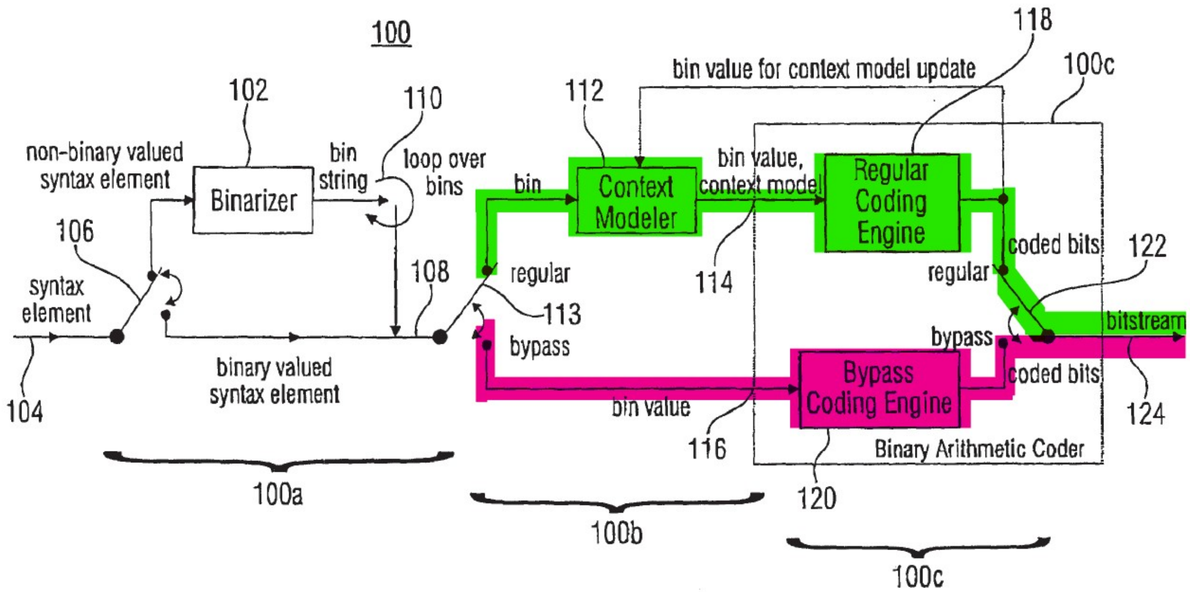


FIG. 2

Marpe, Fig. 2.

Marpe explains that switch 113 determines which pathway is used. Marpe, [0044] ("Stage 100b is a context modelling stage.… **The function of switch 113 is to pass the bits or bins of the bin string at binarization stage output 108 to either the context modeler 112** [green path] or to a bypass coding input terminal 116 [pink path] of stage 100c, thereby bypassing context modeler 112.").

Marpe explains that the green pathway includes a context modeler 112 where the context gets updated. *Id.*, [0047] ("The context modeler 112 adapts an actual bit or bin value probability distribution estimation in accordance with a predetermined context model type which is associated with that bit or bin. The **estimation adaptation or estimation <u>update</u> is based on past or prior bits**

values which the context modeler 112 may receive at the feedback input from stage 10[0]c or may temporarily store ["*symbols… that have been subjected to a context update*"].”), [0114] (“This bin value is also fed back to the context modeler **so that a context model <u>update</u> can be obtained**.”).

Marpe explains that these context-updated symbols are then entropy coded in the regular coding engine 118. *Id.*, [0047] (“**The context modeler 1[1]2 passes the bits further to arithmetic coding stage** 10[0]c together with the new adaptively varied bit value probability distribution estimation, whereby **the context modeler 112 drives the arithmetic coding stage 100c to generate a sequence of bits as a coded representation of the syntax elements according to the adaptive bit value probability distribution estimation**.”); *see also id.*, [0114].

A POSITA would have understood that the green pathway in Marpe's Figure 2 above—whereby symbols are context-updated before entropy coding—would be applied to those symbols that correspond to the “*first category of syntax elements*” in Murashita (i.e., symbols whose cumulative frequency of occurrence is less than a threshold). Saber, ¶167. This is especially so given Murashita's teachings where symbols are only context updated that correspond to the first category of SEs (*see* §VII.A.2.a.iv), and Marpe's teachings that context updating can be based on how “frequently observed” the bins are (Marpe, [0016]). *Id.*

v. 1[d]

Murashita teaches 1[d] for the reasons described above. *See* §VII.A.2.a.v; Saber, ¶¶168–74.

To the extent it is argued that Murashita by itself does not teach 1[d], a POSITA would have been motivated to combine the teachings of Murashita and Marpe, to improve Murashita's coding technique by utilizing Marpe's binary arithmetic coding and binarization engine (i.e., CABAC engine), for the reasons previously described. *See* §VII.B.2. Murashita/Marpe teaches 1[d]; Saber, ¶169.

Explained above for 1[c], Marpe teaches "*entropy coding symbols*." *See* §VII.B.3.a.iv. Marpe also teaches "*symbols that correspond to the second category of syntax elements and that have bypassed context updating*." Saber, ¶¶170–74. This is depicted in Marpe's Figure 2, below, as the pink "bypass" pathway. *Id.*, ¶170.

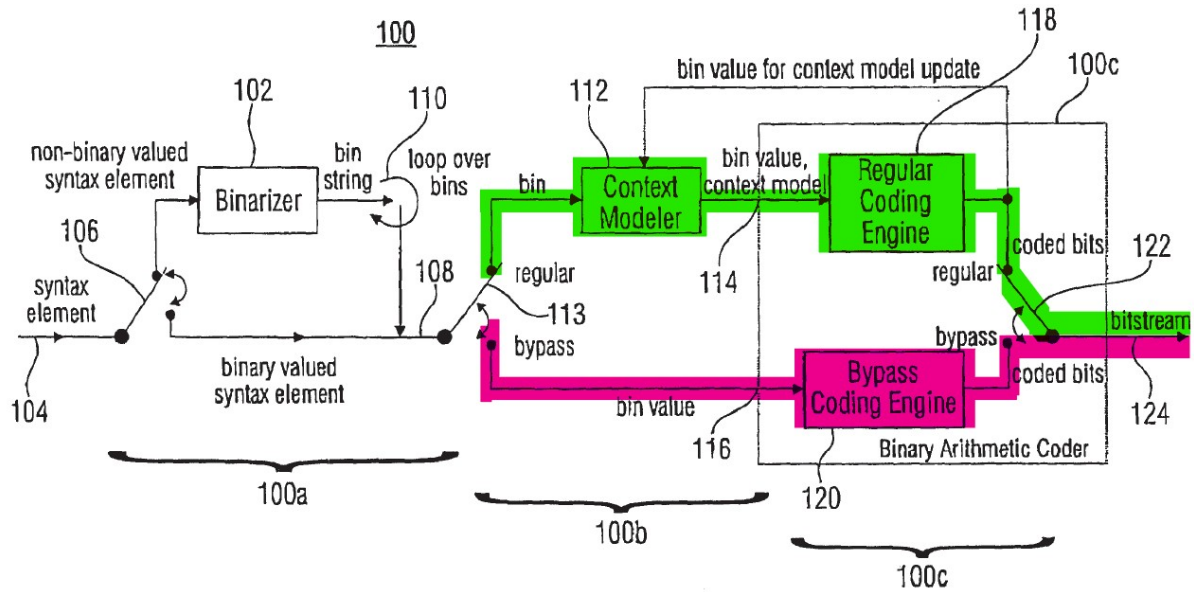


FIG. 2

Marpe, Fig. 2.

Marpe explains that switch 113 determines which pathway is used. Marpe, [0044] ("Stage 100b is a context modelling stage.… **The function of switch 113 is to pass the bits or bins of the bin string at binarization stage output 108 to** either the context modeler 112 [green path] or to **a bypass coding input terminal 116** [pink path] of stage 100c, **thereby bypassing context modeler 112**.").

Marpe further explains that the pink pathway bypasses context updating, instead using the bypass coding engine. *Id.*, [0114] ("**Contrary to the regular coding engine, the bypass coding engine is not an adaptive coding engine but works** preferably with a fixed probability model **without any context adaption** ["*symbols… that have bypassed context updating*"]."), [0050] ("The **bypass**

coding engine 120 is for binary arithmetically coding the bin values passed directly from binarization stage output 108 via switch 113 **by use of a static** predetermined **probability distribution** estimation and also outputs coded bits.”), [0051].

A POSITA would have understood that the pink pathway in Marpe’s Figure 2 above—whereby symbols bypass context-updating before entropy coding—would be applied to those symbols that correspond to the “*second category of syntax elements*” in Murashita (i.e., symbols whose cumulative frequency of occurrence is greater than a threshold). Saber, ¶174. This is especially so given Murashita’s teachings, where context updating is bypassed only for those symbols that correspond to the second category of SEs (*see* §VII.A.2.a.v), and Marpe’s teachings that context updating can be based on how “frequently observed” the bins are (Marpe, [0016]). *Id.*

b. Claim 2

Murashita/Marpe teaches claim 2. Saber, ¶¶175–78. A POSITA would have been motivated to combine the teachings of Murashita and Marpe, for the reasons previously described. *See* §VII.B.2.

Marpe’s Figure 1, below, “shows a general view of [a] video coder environment to which the present invention could be applied. A picture 10 [highlighted yellow] is fed to a video precoder 12. The video precoder treats the picture 10 in units of so called macro blocks 10a." Marpe, [0036]. Marpe explains that "[t]he macroblocks into which the picture 10 is partitioned are **grouped into several slices. For each slice a number of syntax elements are generated** ("*categorization of the syntax elements to be signaled at a... slice... level*") which form a coded version of the macroblocks of the slice." *Id.*, [0037]. Marpe further explains that "**[t]hese syntax elements are dividable into two categories**. **The first category contains** the elements related to macroblock type, sub-macroblock type and information of prediction modes both of spatial and of temporal types as well as **slice-based and macroblock-based** control information, such as components of motion vector differences, which are prediction residuals." *Id.*; *see also id.*, [0092]–[0095].

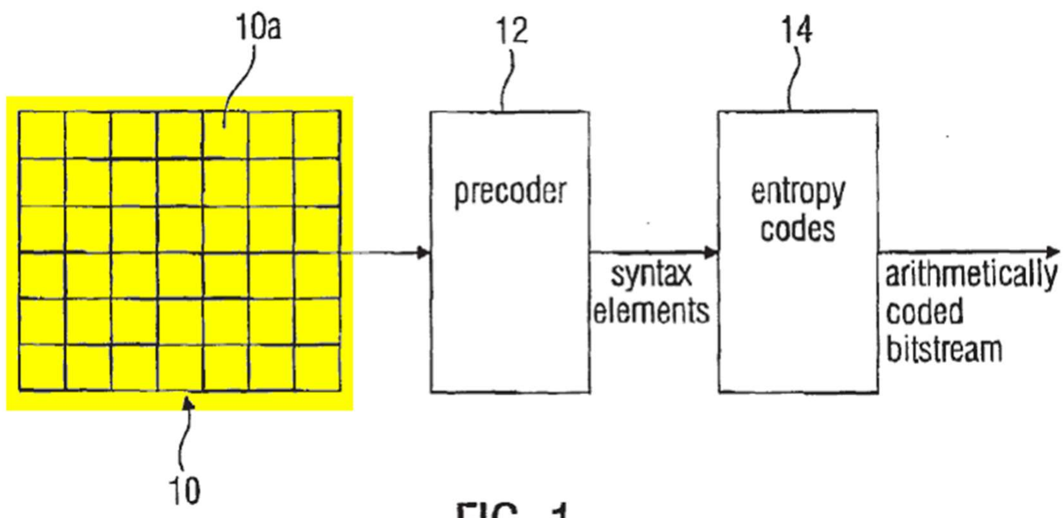


FIG. 1

Id., Fig. 1.

Marpe's Figure 12, below, "shows the basic coding structure for the emerging H.264/AVC standard for a macroblock**. The input video signal is, split into macroblocks**, each macroblock having 16x16 pixels. Then, **the association of macroblocks to slice groups and slices is selected** ("*categorization of the syntax elements to be signaled at a... slice... level*"), and, then, each macroblock of each slice is processed by the network of operating blocks in FIG. 12." *Id.*, [0092].

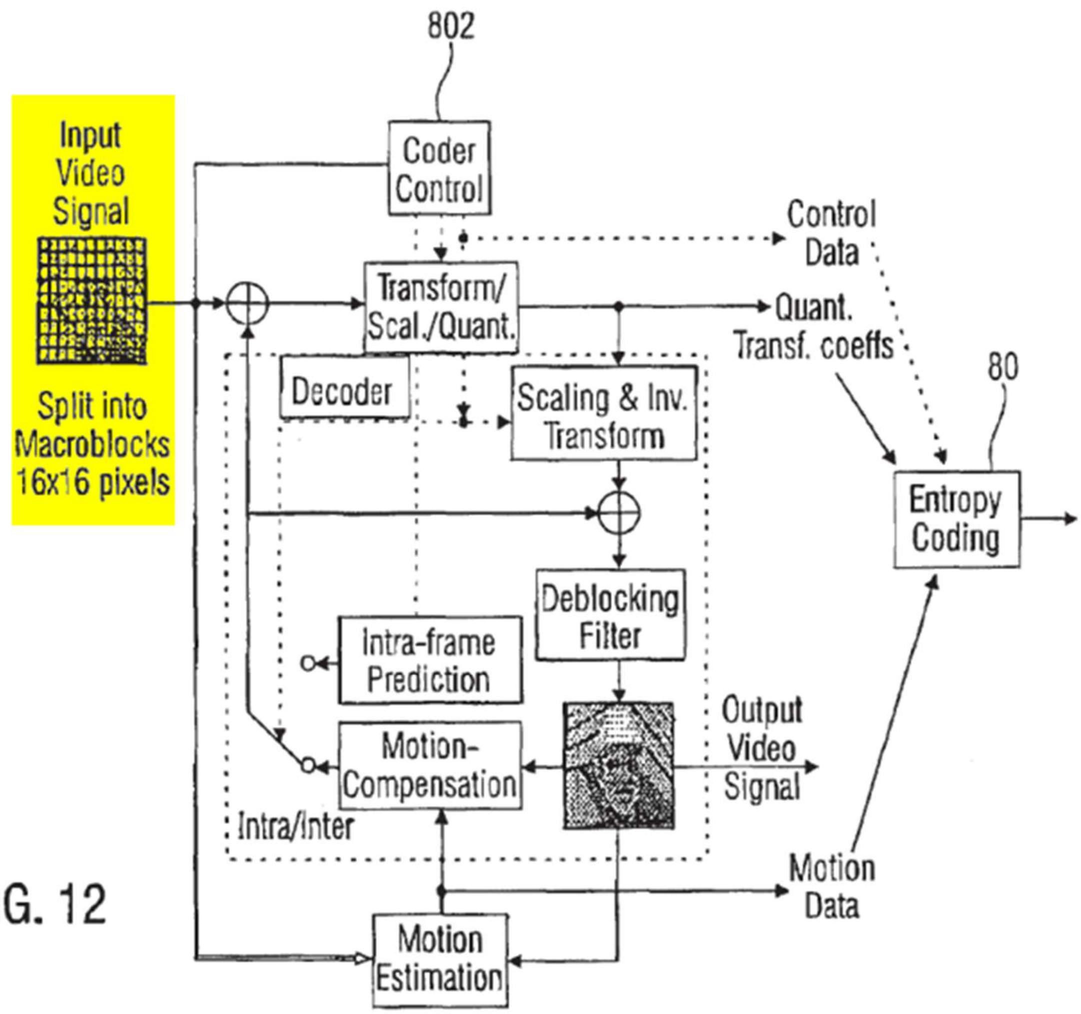


FIG. 12

Id., Fig. 12.

c. Claim 3

Murashita/Marpe teaches claim 3. Saber, ¶¶179–82. A POSITA would have been motivated to combine the teachings of Murashita and Marpe, for the reasons previously described. *See* §VII.B.2.

As discussed previously, the pink path below depicts symbols that correspond to the second category of SEs that have bypassed context updating. *See* §VII.B.3.a.v.

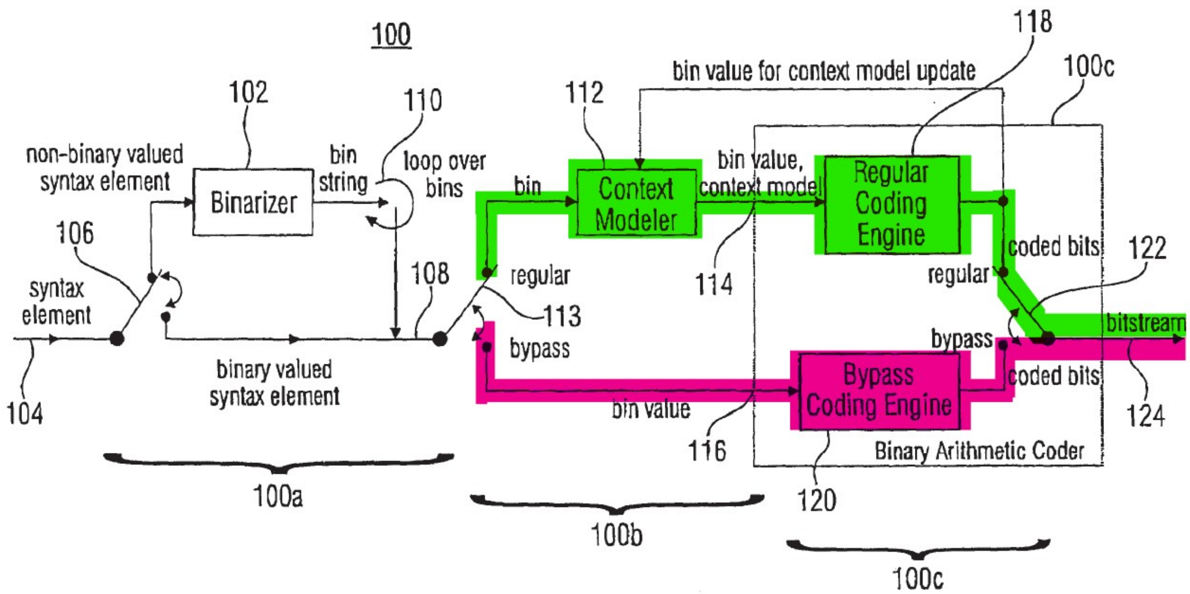


FIG. 2

Marpe, Fig. 2.

Marpe explains that its "present invention is based on the finding that a very effective compression of data values... may be achieved by using **a binarization scheme for preparing the syntax elements for the arithmetic coding**" [“*performing binarization of the symbols*”]. *Id.*, [0014]; *see also id.*, [0015]. Referencing Marpe’s Figure 2, below, Marpe explains that **“[t]he first stage 10[0]a is the binarization stage and comprises a binarizer 102** [“*performing binarization of the symbols*”].... **The output of binarizer 102** is connected to an output 108 of stage 10[0]a, which, at the same time, **forms the input of stage 100b**.” *Id.*, [0039]–[0041]. Marpe explains that “**[b]inarizer 102** maps the non-binary valued syntax elements to a code word, or a so called bin string, in accordance with a **binarization scheme** embodiments[.]” *Id.*, [0041]. As shown below, the binarization (in stage 100a) of the symbols that correspond to the second category of syntax elements occurs **prior** to bypassing the context updating (in stage 100b) and entropy coding (in stage 100c). Saber, ¶180.

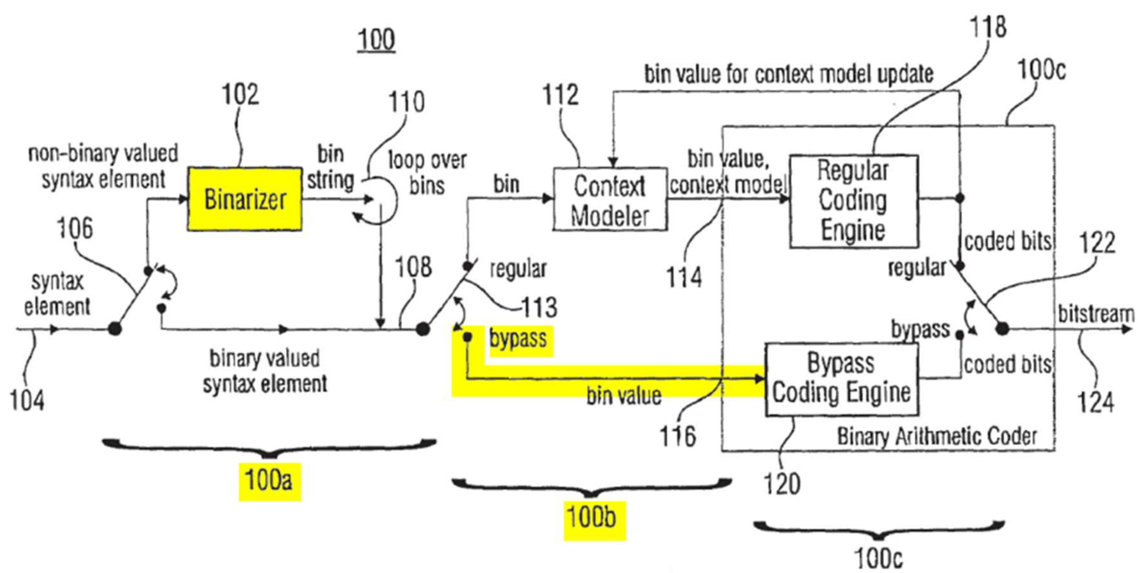


FIG. 2

Marpe, Fig. 2.

Marpe explains, in Figure 2 above, "[t]he bypass coding engine 112 has also an input and an output terminal, the input terminal being connected to the bypass coding input terminal 116. **The bypass coding engine 120 is for binary arithmetically coding the bin values passed directly from binarization stage output 108 via switch 113** ["*performing binarization of the symbols that correspond to the second category of syntax elements prior to bypassing the context updating*"] by use of a static predetermined probability distribution estimation and also outputs coded bits." *Id.*, [0050]; *see also id.*, [0044].

d. Claim 4

Murashita/Marpe teaches claim 4. Saber, ¶¶183–85. A POSITA would have been motivated to combine the teachings of Murashita and Marpe, for the reasons previously described. *See* §VII.B.2.

Marpe teaches performing binarization for the reasons described in connection with claim 3. *See* §VII.B.3.c. Marpe also discloses doing so "*as to target a predetermined probability*." Saber, ¶¶184–85.

Marpe explains that "[t]he use of the inventive binarization scheme enables, in connection with binary arithmetic coding, an effective way of **adapting the binarization representation of the data value to the probability distribution of the data values**." Marpe, [0017]; *see also id.*, [0048]. Murashita's scheme does so by using a "suitably **selected cut-off value** [that] may lead to the least probable data values to be binarized into code words having **equally probable bit values**, whereas the most probable data values may be binarized into code words leading to a very effective arithmetic coding bit stream when using adaptive context modeling." Marpe, [0017]. A POSITA would understand Marpe's binarization scheme—that targets equally probable bit values for the least probable data values—to target a predetermined probability. Saber, ¶185.

Marpe explains that "coding based on a **static probability estimation may be employed for the second binarization scheme** suffix of the code word[.]" Marpe, [0017]. Likewise, a POSITA would have understood this to mean that Marpe's binarization targets a predetermined probability. Saber, ¶185.

e. Claim 5

Murashita/Marpe teaches claim 5. Saber, ¶¶186–93. A POSITA would have been motivated to combine the teachings of Murashita and Marpe, for the reasons previously described. *See* §VII.B.2.

As discussed with respect to claim 2, Marpe teaches "*causing a **categorization of the syntax elements** to be signaled at a block, **slice**, picture or sequence level*." *See* §VII.B.3.b. Marpe discloses "*causing an **indication of the entropy coding** to be signaled at a block, **slice**, picture or sequence level*."

Marpe explains that Figure 12, below, shows "in more detail than in FIG. 1, the **complete setup of a video encoder engine** including an entropy-encoder as it is shown in FIG. 12 in block 80[] in which the aforementioned binarization and binary arithmetic coding is used. In particular, FIG. 12 shows the basic coding structure for the emerging H.264/AVC standard for a macroblock. The input video signal is, split into macroblocks, each macroblock having 16x16 pixels. Then, the association of macroblocks to slice groups and slices is selected, and, then, **each macroblock of each slice is processed by the network of operating blocks in FIG. 12** ["*causing an indication of the entropy coding to be signaled at a... slice... level*"]." Marpe, [0092]. Marpe teaches several slices that can be entropy coded:

> I slice: A slice in which **all macroblocks of the slice are coded using intra prediction**.
>
> P slice: In addition, to the coding types of the I slice, **some macroblocks of the P slice can also be coded using inter prediction** with at most one motion-compensated prediction signal per prediction block.
>
> B slice: In addition, to the coding types available in a P slice, **some macroblocks of the B slice can also be coded using inter prediction** with two motion-compensated prediction signals per prediction block....
>
> SP slice: A so-called switching **P slice that is coded** such that efficient switching between different precoded pictures becomes possible.
>
> SI slice: A so-called switching I slice that allows an exact match of a macroblock in an SP slice for random access and error recovery purposes.

Id., [0092]–[0098]. Marpe's entropy coder is highlighted below, in Marpe's Figure 12, which entropy codes at a slice level. *Id.*, [0092].

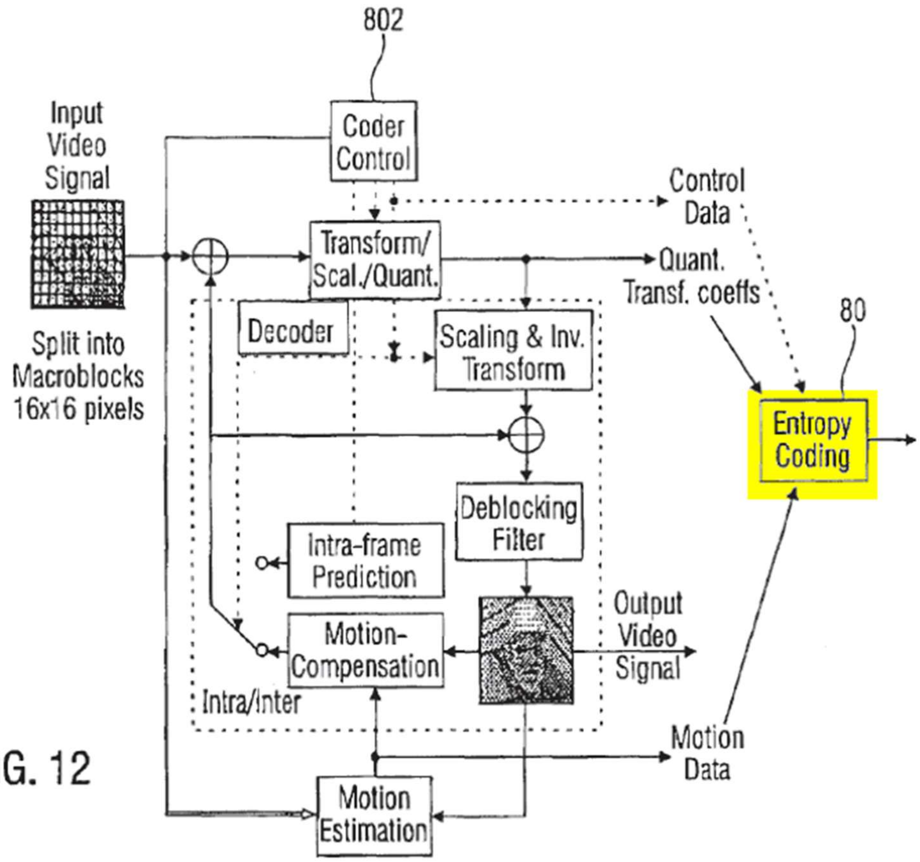


FIG. 12

Id., Fig. 12.

f. Claim 6

Murashita/Marpe teaches claim 6 for the reasons described above. *See* §§VII.A.2.b, VII.B.3.a; Saber, ¶¶194.

g. Claim 7

Murashita/Marpe teaches claim 7. Saber, ¶¶195–99. A POSITA would have been motivated to combine the teachings of Murashita and Marpe, for the reasons previously described. *See* §VII.B.2.

As discussed in connection with claim 3, Marpe discloses performing binarization of symbols at stage 100a in Figure 2. *See* §VII.B.3.c. As discussed previously, the green path below depicts symbols that correspond to the first category of SEs that undergo context updating. *See* VII.B.3.a.iv.

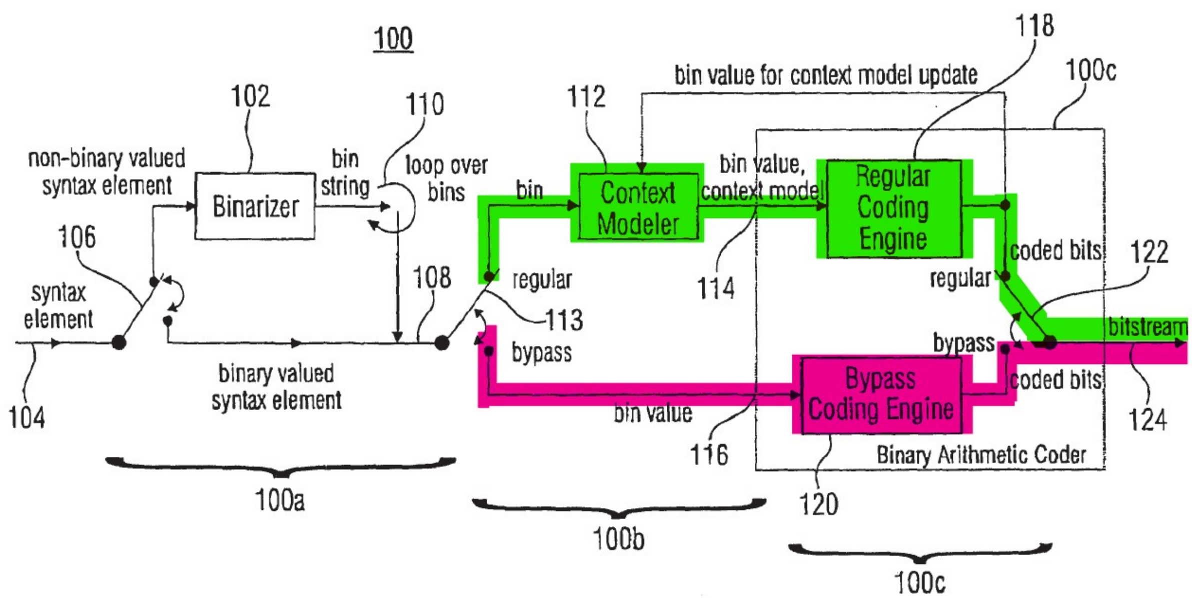


FIG. 2

Marpe, Fig. 2.

As shown below, the binarization (in stage 100a) of the symbols that correspond to the first category of SEs occurs **prior** to context updating (in stage 100b) and entropy coding (in stage 100c). Saber, ¶¶196–97.

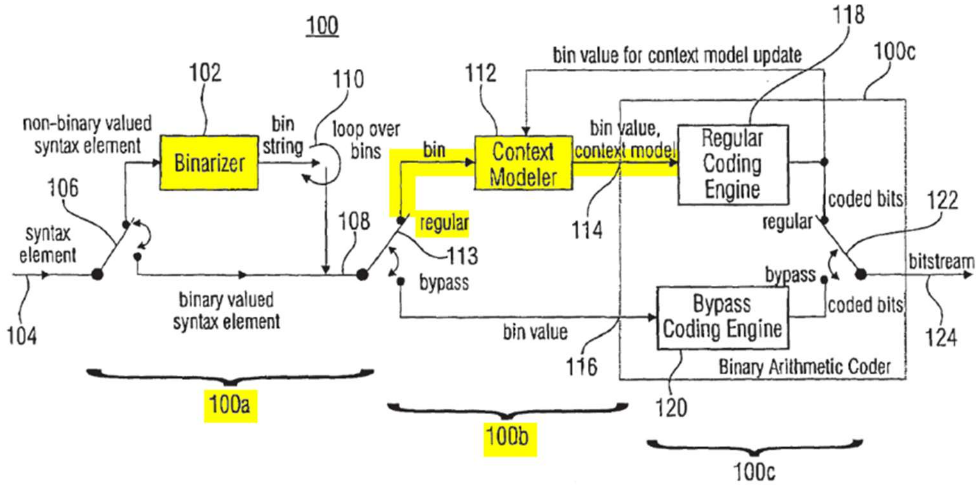


FIG. 2

Marpe, Fig. 2.

Marpe explains, in Figure 2 above, "[t]he **function of switch 113 is to pass the bits or bins of the bin string at binarization stage output 108** ["*performing binarization*"] **to either the context modeler 112** ["*of the symbols that correspond to the first category of syntax elements*"] or to a bypass coding input terminal 116 of stage 100c, thereby bypassing context modeler 112." *Id.*, [0044]. Marpe explains that **"[t]he regular coding engine 118 binary arithmetically codes the bin values passed from context modeler 112** ["*performing binarization… prior to entropy coding the symbols that correspond to the first category of syntax* *elements*"] by use of the context model also passed from context modeler 112 and outputs coded bits." *Id.*, [0049].

4. Claim 8 and Dependents

a. Claim 8

i. 8[pre]

If limiting, Murashita/Marpe teaches 8[pre]. Saber, ¶¶200–01.

Marpe discloses a "Method and **Apparatus** for Binarization and Arithmetic Coding of a Data Value." Marpe, Title; *see also id.*, [0012]. Marpe continues, stating that "[d]epending on an actual implementation, the inventive encoding/decoding and binarization/recovering methods can be implemented in **hardware** or in **software**. Therefore, the present invention also relates to a **computer program**, which can be stored on a **computer-readable medium** such as a CD, a disk or any other data carrier. The present invention is, therefore, also a computer program having a **program code** which, when executed on a **computer**, performs the inventive method of encoding or binarizing or the inventive method of decoding or recovering described in connection with the above figures." *Id.*, [0138]. A POSITA would have understood that a processor, memory, and computer program code are universal components of computer systems and thus would be utilized in the combined system of Murashita and Marpe. Saber, ¶201.

ii. 8[a]

Murashita/Marpe teaches 8[a] for the reasons described for 1[a]. *See* §VII.A.2.a.ii; Saber, ¶202.

iii. 8[b]

Murashita/Marpe teaches 8[b] for the reasons described for 1[b]. *See* §VII.A.2.a.iii; Saber, ¶203.

iv. 8[c]

Murashita/Marpe teaches 8[c] for the reasons described for 1[c]. *See* §VII.B.3.a.iv; Saber, ¶204.

v. 8[d]

Murashita/Marpe teaches 8[d] for the reasons described for 1[d]. *See* §VII.B.3.a.v; Saber, ¶205.

b. Claim 9

Murashita/Marpe teaches claim 9 for the reasons described for claims 2 and 8. *See* §§VII.B.3.b, VII.B.4.a; Saber, ¶¶206–08.

c. Claim 10

Murashita/Marpe teaches claim 10 for the reasons described for claims 3 and 8. *See* §§VII.B.3.c, VII.B.4.a; Saber, ¶¶209–11.

d. Claim 11

Murashita/Marpe teaches claim 11 for the reasons described for claims 4 and 10. *See* §§VII.B.3.d, VII.B.4.c; Saber, ¶¶212–14.

e. Claim 12

Murashita/Marpe teaches claim 12 for the reasons described for claims 5 and 8. *See* §§VII.B.3.e, VII.B.4.a; Saber, ¶¶215–17.

f. Claim 13

Murashita/Marpe teaches claim 13 for the reasons described above. *See* §§VII.A.2.b, VII.B.4.a; Saber, ¶218.

g. Claim 14

Murashita/Marpe teaches claim 14 for the reasons described for claims 7 and 8. *See* §§VII.B.3.f, VII.B.4.a; Saber, ¶¶219–21.

5. Claim 15 and Dependents

a. Claim 15

i. 15[pre]

If limiting, Murashita/Marpe teaches 15[pre] for the reasons described for 8[pre]. *See* §VII.B.4.a.i. Saber, ¶222.

ii. 15[a]

Murashita/Marpe teaches 15[a] for the reasons described for 1[a]. *See* §VII.A.2.a.ii; Saber, ¶223.

iii. 15[b]

Murashita/Marpe teaches 15[b] for the reasons described for 1[b]. *See* §VII.A.2.a.iii; Saber, ¶224.

iv. 15[c]

Murashita/Marpe teaches 15[c] for the reasons described for 1[c]. *See* §VII.B.3.a.iv; Saber, ¶225.

v. 15[d]

Murashita/Marpe teaches 15[d] for the reasons described for 1[d]. *See* §VII.B.3.a.v; Saber, ¶226.

b. Claim 16

Murashita/Marpe teaches claim 16 for the reasons described for claims 2 and 15. *See* §§VII.B.3.b, VII.B.5.a; Saber, ¶¶227–29.

c. Claim 17

Murashita/Marpe teaches claim 17 for the reasons described for claims 3 and 15. *See* §§VII.B.3.c, VII.B.5.a; Saber, ¶¶230–32.

d. Claim 18

Murashita/Marpe teaches claim 18 for the reasons described for claims 4 and 17. *See* §§VII.B.3.d, VII.B.5.c; Saber, ¶¶233–35.

e. Claim 19

Murashita/Marpe teaches claim 19 for the reasons described for claims 5 and 15. *See* §§VII.B.3.e, VII.B.5.a; Saber, ¶¶236–38.

f. Claim 20

Murashita/Marpe teaches claim 20 for the reasons described above. *See* §§VII.A.2.b, VII.B.5.a; Saber, ¶239.

C. Ground 3: Claims 1, 3, 6-8, 10, 13-15, 17, 20 are obvious over Yu

1. Yu

Yu (EX-1018) is a printed publication. Yu was publicly available in December 2005. Hall-Ellis, ¶¶46–53. Therefore, Yu was available before the earliest priority date on the ’701 Patent’s face and is prior art under §102(b). *Id.*; Saber, ¶240.

Yu describes “a highly efficient architecture for CABAC decoding” in which “[t]he two most time[-]consuming parts… i.e. binary arithmetic decoding process and context model maintenance[] are carefully designed to make [a] good tradeoff between decoding efficiency and chip cost.” Yu, 1352. Yu’s “arithmetic decoding engine keeps updating two 9-bit registers *range* and *offset* during the whole decoding process.” Yu, 1354 (emphasis in original). The *range* register tracks “the width of the current interval” (i.e., the range of possible fractional values that can be assigned to a sequence during encoding (Saber, ¶242)), while *offset* tracks “the input bit stream.” Yu, 1354. When encoding or decoding a bin—“the basic arithmetic decoding unit in CABAC" (Yu, 1352)—*range* is split into two sub-intervals designated *rLPS* and *rMPS*. *rLPS* "correspond[s] to the estimated probability interval of the *LPS*"[8] and is calculated as $rLPS = R \cdot p_{LPS}$, where R is interval width, and p_{LPS} is the estimated probability of an *LPS*. Similarly, *rMPS* "correspond[s] to the estimated probability interval of the *MPS*"[9] and is calculated as $rMPS = R - rLPS$. Yu, 1353. In "regular mode," "the *LPS* probability p_{LPS} is estimated by the corresponding context model," but in "bypass mode," "p_{LPS} is estimated as 0.5 and no context model is needed." Yu, 1354.

Yu compares the *offset* value with *range* to determine whether a bin is *MPS* or *LPS*. This is illustrated in Figure 2 (below): "the left-side plot shows the case that *MPS* occurs, where the *offset* is less than *rMPS*," and "[t]he right-side plot shows the case that *LPS* occurs, where the *offset* is greater than or equal to *rMPS*." *Id.*

---

[8] *LPS* is the estimated probability of the least probable symbol. Yu, 1353.

[9] *MPS* is the estimated probability of the most probable symbol. *Id.*

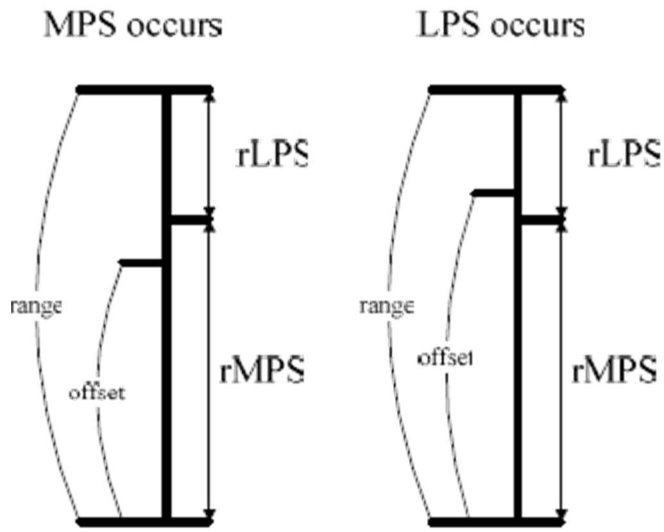


Fig. 2. Arithmetic decoding engine for one bin

Id., Fig. 2.

Yu classifies SEs into two classes based on frequency of occurrence: a first class with SEs that "occur only once or a few times in one MB," and a second class with SEs that "occur tens or hundreds of times in one MB" (e.g., coeff_abs_level_minus1). Yu, 1355. Because SEs "of the first class contribute only a small part to the arithmetic decoding workload… one bin is processed per cycle… to decrease the complexity and hardware cost." *Id.* But "to accelerate decoding of the syntax elements in the second class," Yu discloses a decoder architecture that "can decode two regular bins together with one bypass bin in one cycle." *Id.* Bins encoded in bypass mode "only appear in two cases, either as sign or as Exp-Golomb (EG) code." *Id.* Both cases use the SE coeff_abs_level_minus1, indicating that high-occurrence elements are decoded in bypass mode, where context is not needed. *Id.*

2. Claim 1 and Dependents[10]

a. Claim 1

i. 1[pre]

If limiting, Yu teaches 1[pre]. Saber, ¶245. As discussed above, Yu is directed to and describes an architecture for the well-known CABAC entropy coding *method* described in, for example, the H.264/AVC video standard. Yu, 1352.

ii. 1[a]

Yu teaches 1[a]. Saber, ¶¶246–58.

---

[10] In a co-pending ITC investigation (337-TA-1448), PO accuses products that include "hardware-accelerated video decoding" of infringing claims 1, 3, 4, 6-8, 10, 11, 13-15, 17, 18, and 20 of the '701 patent. *See generally* EX-1021. At least to the extent PO argues that the Challenged Claims can be met by a decoder—rather than an encoder—the claims would have been obvious in view of Yu's disclosures, as explained in this Petition.

"***Syntax elements in a bit stream***"[11]

Yu explains that the CABAC encoding process includes "1) binarization, which is the unique mapping process of possible values of **a syntax element** onto **a set of bin strings**;" "2) probability modeling," where "[c]ontext models are used for the probability estimation, each of which corresponds to a certain bin or **several bins of a certain syntax element**;" and then "3) [b]inary arithmetic coding" of each SE. Yu, 1352–53. Yu further explains that a "**syntax element is an element of data represented in the bit stream** to be encoded," and a "bin string is… an intermediate binary representation of value **of syntax elements**." Yu, 1353. Moreover, "[t]he CABAC decoding process is the inverse of the CABAC encoding" such that "decoded bins… reconstruct" a given syntax element. *Id.* The CABAC decoding process is illustrated in Figure 1 (below), including the input bitstream through the Software/Hardware Interface and the output of decoded SEs.

---

[11] At least to the extent PO argues that "bin strings" are "syntax elements" under the Challenged Claims—as PO appears to argue in co-pending ITC investigation (337-TA-1448) for many of the Challenged Claims (*see generally* EX-1021)—the claims would have been obvious in view of Yu's disclosures, as explained in this Petition.

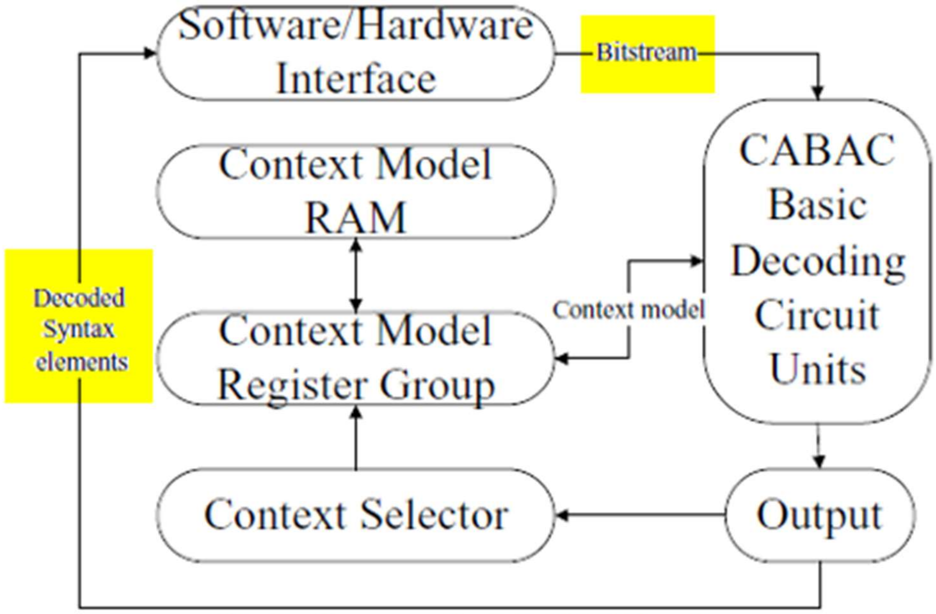


Fig. 1. Framework of CABAC decoder

Id., Fig. 1; Saber, ¶247.

"***Expected frequency of occurrence of syntax elements in a bit stream***"

Yu discloses that during the "probability modeling" step of encoding SEs in one or more bins, a context model is selected that "comprises… a 6-bit state value *state* as the probability index, and a binary value of the [] *MPS*," where *state* "is used as an index of the estimated probability value of the [] '*LPS*'." Yu, 1353. And during the "binary arithmetic coding" step, the *range* interval is further split into two sub-intervals—*rLPS* and *rMPS*—"correspond[ing] to the estimated probability interval of the *LPS*" and "the estimated probability interval of the *MPS*," respectively. *Id.* In "regular mode," "the *LPS* probability p_{LPS} is estimated by the corresponding context model" such that when the bin is subsequently decoded, whether it is decoded as *MPS* or *LPS* depends on whether the *offset* value falls into the *rLPS* or *rMPS* subinterval. Yu, 1354. This determination is illustrated in Figure 2:

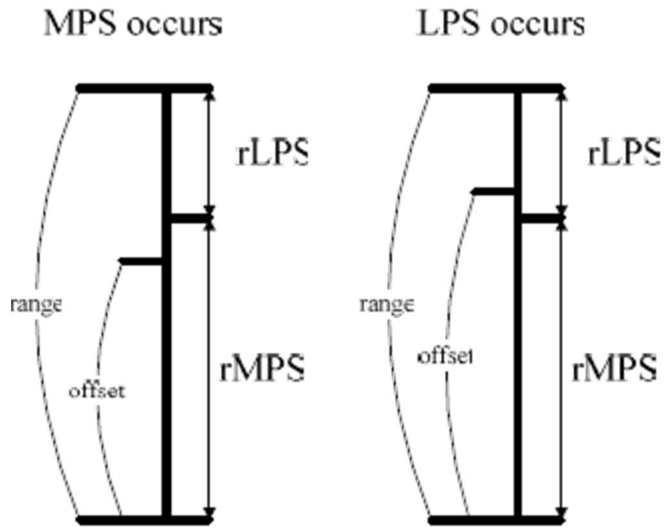


Fig. 2. Arithmetic decoding engine for one bin

Id., Fig. 2. In "bypass mode," on the other hand, "p_{LPS} is estimated as 0.5 and no context model is needed." Yu, 1354.

Yu's probability interval p_{LPS} thus indicates the *expected frequency of occurrence* of a given *syntax element* in the encoded bin stream ("*in a bit stream*"). Saber, ¶248. If the syntax element has a high frequency, the probability of occurrence of bins constituting that syntax element is also high, and its probability interval is narrower. Saber, ¶246–50.

"***Frequency of occurrence threshold***"

Yu teaches the use of a threshold value when encoding and decoding a bin as LPS or MPS ("*frequency of occurrence threshold*"). Saber, ¶¶251–53. As explained above, CABAC encoding includes selecting a context model that indicates whether a bin is *MPS* or *LPS* and then coding based on a subdivision of the *range* interval using the probability estimation (p_{LPS}) that a given bin is LPS. Yu, 1353. Then when decoding, Yu discloses determining whether the incoming bin is *MPS* or *LPS* by comparing its *offset* value to the *rLPS* and *rMPS* subintervals calculated using p_{LPS} (in "regular mode") or to half the *range* (in "bypass mode"). Yu, 1354. A POSITA would therefore have understood that there is a threshold value between the *rLPS* and *rMPS* subintervals in "regular mode" so that the decoder can successfully make the comparison, as illustrated in Figure 2. Saber, ¶251.

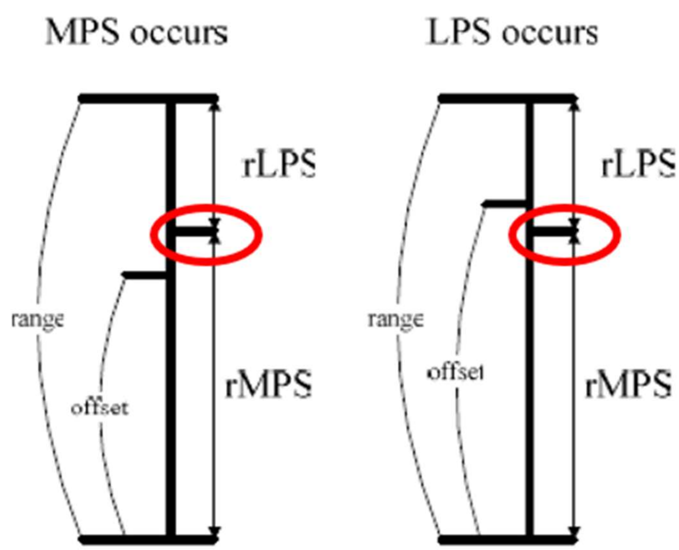


Fig. 2. Arithmetic decoding engine for one bin

Yu, Fig. 2. Or in "bypass mode," the threshold would be "*range*/2 to determine the bin is *MPS* or *LPS*." *Id.*, 1354.

Yu describes classifying SEs into two classes based on the frequency of occurrence of bins present in those SEs: a first class of SEs that "contain a few bins and occur only once or a few times in one MB" (i.e., bins where the *offset* value is below the threshold, indicating an LPS); and a second class of SEs that "contain several bins and occur tens or hundreds of times in one MB" (i.e., bins where the *offset* value is above the threshold, indicating an MPS). Yu, 1355.

"***<u>Based on an expected frequency of occurrence</u>***"

Yu teaches *determining* the *threshold… based on an expected frequency of occurrence.* Saber, ¶¶254–58. In "bypass mode," the threshold for distinguishing between *MPS* and *LPS* is half the interval *range* (Yu, 1354), which represents the probability of occurrence of a given SE. And in "regular mode," Yu sets *rLPS* and *rMPS* based on their probabilities (*see* Yu, 1354), which, as explained above, are likewise linked to the frequency of occurrence of the bin(s) corresponding to a given SE. *See* Yu, 1353–55; Saber, ¶254. In either mode, MPS indicates a more probable SE (higher occurrence frequency), and LPS indicates a less probable SE (lower occurrence frequency). Saber, ¶254.

Although Yu does not provide a calculation for computing its threshold to distinguish between MPS and LPS in "regular mode," a POSITA would have understood that it is also based on the "*expected frequency of occurrence*" of SEs. Saber, ¶255. First, the threshold is set before bins are classified as LPS or MPS, indicating the system is designed with an expectation of how often SEs will occur so that they can be properly classified according to "their occurring frequency during the decoding process" as Yu describes. Yu, 1355. For example, referring to Figure 2 above, a POSITA would have understood that the magnitude of *rLPS* and *rMPS* (and consequently the threshold between them) should be set so that that differentiating between *LPS* and *MPS* allows the decoder to achieve Yu's objective of "efficiently organiz[ing] the decoding engines... to achieve high decoding speed" (Yu, 1354).

Second, a POSITA would have recognized that Yu's threshold is based on frequency of occurrence because Yu classifies SEs according to "their occurring frequency" so that the decoder can "accelerate decoding of the syntax elements in the second class"—i.e., SEs that "occur tens or hundreds of times in one MB"—using "a new architecture that can decode two regular bins together with one bypass bin in one cycle" without the need for context updating. Yu, 1355. 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. Saber, ¶¶255–58.

iii. 1[b]

Yu teaches 1[b]. Saber, ¶¶259–61.

As discussed above for 1[a] (*see* §VII.C.2.a.ii), Yu explains that "during the decoding process" of incoming SEs ("*syntax elements of video content*"), "there are first and second classes of syntax elements." Yu, 1355; *see also* §VII.C.2.a.ii. An SE in the first class ("*first category*") consists of bins where the *offset* value is greater than *rMPS* (i.e., bins "*occur less than the frequency of occurrence*"), indicating the SE will be decoded as an LPS; and an SE in the second class ("*second category*") consists of bins where the *offset* value is less than *rMPS* (i.e., bins "*occur greater than the frequency of occurrence threshold*"), indicating the SE will be decoded as an MPS. Yu, 1353–54; Saber, ¶260. This is illustrated in Figure 2:

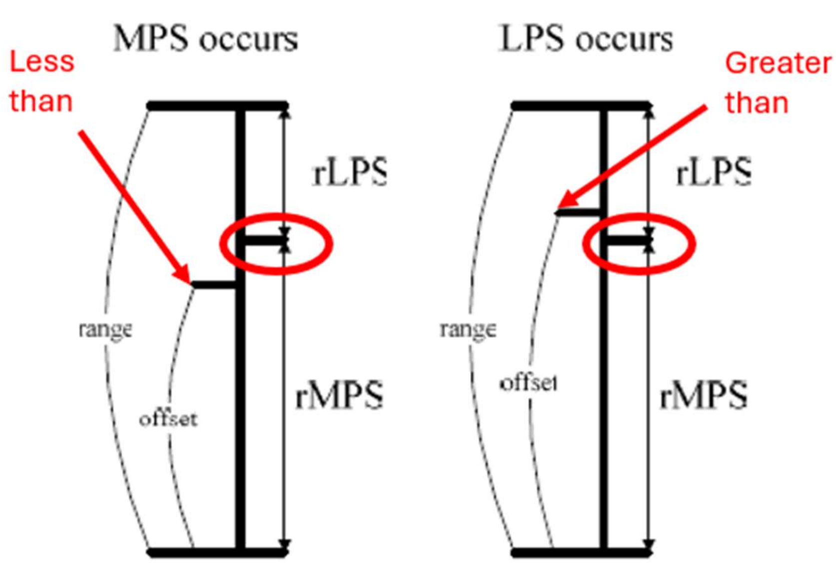


Fig. 2. Arithmetic decoding engine for one bin

This categorization allows Yu's system "to accelerate decoding of the syntax elements in the second class [using] a new architecture that can decode two regular bins together with one bypass bin in one cycle" as described. Yu, 1355.

iv. 1[c]

Yu teaches 1[c]. Saber, ¶¶262–67.

As discussed above for 1[pre], Yu is directed to an architecture for CABAC, which "is one of the two alternative "*entropy coding*" method[s] specified in," for example, H.264. Yu, 1352. Saber, ¶263. And Yu describes encoding bins (corresponding to SEs) using "context models" for "probability estimation" during the "probability modeling" phase of encoding and also for *LPS* probability estimation during decoding. Yu, 1353–54. During encoding, the first step of "binarization" maps SEs—or **symbols**—to a string of bins, and then during the "probability modeling" step, "the corresponding context model is selected for the bin and the model is updated after the bin is encoded." Yu, 1353. That context model "comprises a pair of two values, a 6-bit state value *state* as the probability index, and a binary value of the most probable **symbol**, *MPS*." *Id.* As explained above, *state* "is used as an index of the estimated probability value of the [] '*LPS*'." *Id.*

During decoding, the CABAC decoder stores context models in "Context Model RAM" and provides them to decoding circuitry as illustrated in Figure 1:

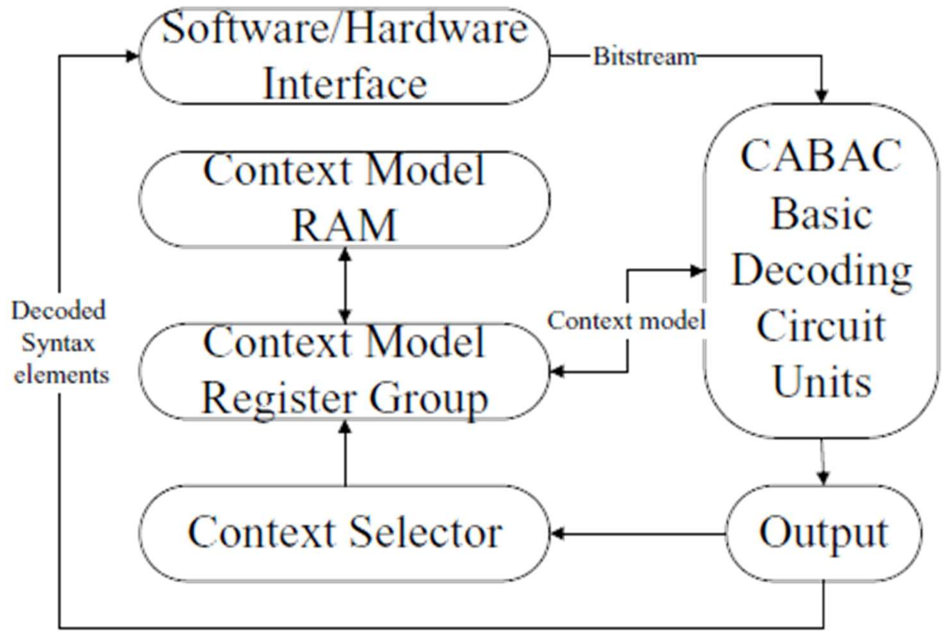


Fig. 1. Framework of CABAC decoder

Id., Fig. 1.

Yu also notes that "[d]uring the decoding process, the context model should be updated," but "[t]he bottleneck for efficient CABAC decoder implementation rests with the context model maintenance and the binary arithmetic decoding process." Yu, 1353. Yu therefore proposes updating context models by dividing them "into several groups according to the order of being called" and then "load[ing] and writ[ing] back all the context models in the same group at the same time, and only once for each group during decoding one MB." Yu, 1356. To accomplish this, Yu's decoder architecture pre-fetches four *rLPS* values based on the context state, enabling rapid selection through a multiplexer. *See* Yu, 1357–58. And while decoding a current block ("RB1"), the system concurrently calculates *rLPS* for the next block ("RB2"), including *state* renewal for RB2 using the updated *state* from RB1. Yu, 1357. This architecture is illustrated in Figures 6 and 7, respectively (below).

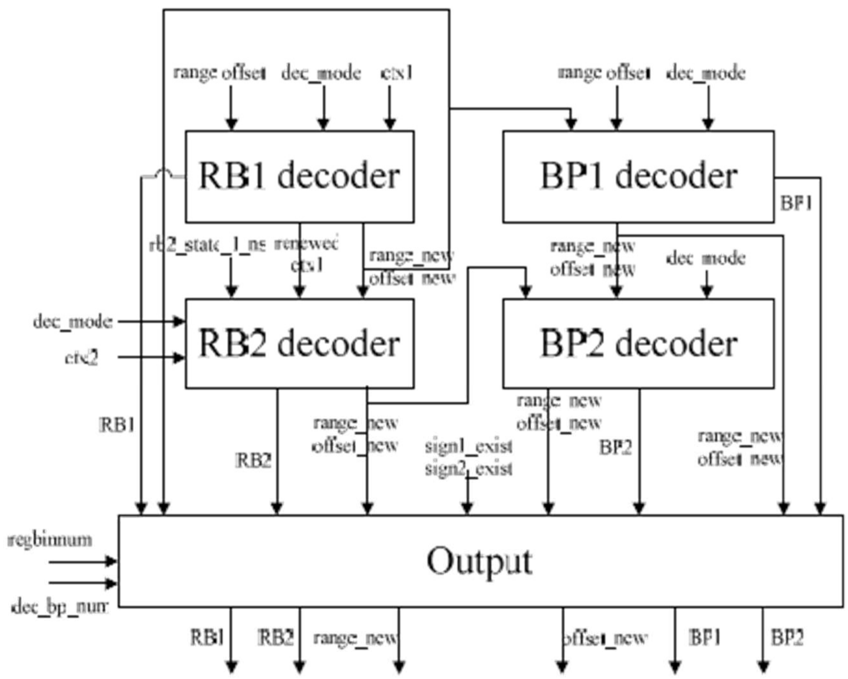


Fig. 6 Basic decoding circuit units

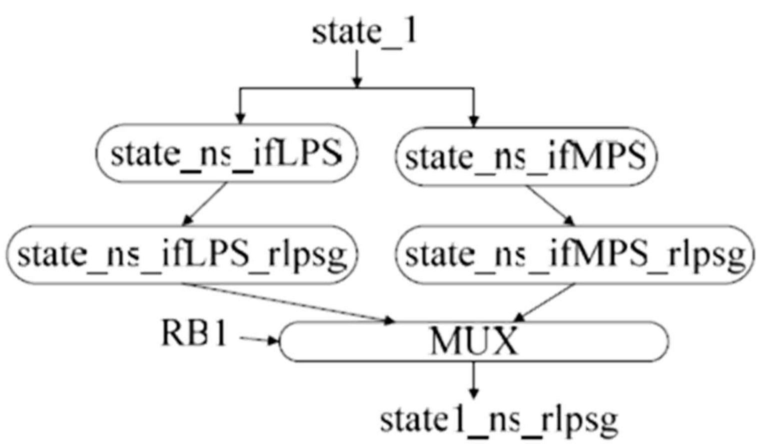


Fig. 7 state_1 renewal and preparation for rlpsg_2

A POSITA would have understood that this design avoids the long processing paths that result if *rLPS* calculations happen after RB1 decoding and *state* updates. Saber, ¶266.

Finally, as explained for 1[a] and 1[b], an SE in Yu's "*first category*" is encoded and decoded as an LPS in "regular mode." *See* §§VII.C.2.a.ii, VII.C.2.a.iii. And in that mode, Yu applies the above architecture for context mapping and updating from one SE to the next. Saber, ¶267.

v. 1[d]

Yu teaches 1[d]. Saber, ¶¶268–70.

As explained for 1[c], Yu's architecture for CABAC entropy coding includes a mechanism for maintaining and updating context for symbols encoded and decoded in "regular mode," for example. *See* §VII.C.2.a.iv. In "bypass mode," on the other hand, Yu notes that "no context model is needed." Yu, 1354.

As explained for 1[a] and 1[b], Yu discloses that an SE in the "*second category*" is encoded and decoded as an MPS in "bypass mode." *See* §§VII.C.2.a.ii, VII.C.2.a.iii. Such SEs include, for example, abs_mvd, significant_coeff_flag, last_significant_coeff_flag and coeff_abs_level_minus1, and Yu explains that "bins encoded by bypass mode only appear in two cases, either as sign or as Exp-Golomb (EG) code." Yu, 1355. Both cases use the SE coeff_abs_level_minus1, confirming that high-occurrence elements (i.e., MPS) are decoded in bypass mode. *Id.* A POSITA would have understood that Yu uses CABAC *entropy coding* for SEs that are MPS ("*symbols that correspond to the second category of syntax elements*") in "bypass mode," which does not use a context model ("*that have bypassed context updating*").

b. Claim 3

Yu teaches claim 3. Saber, ¶¶271–72.

As explained above, Yu discloses that the first step in the CABAC encoding process is "1) binarization, which is the unique mapping process of possible values of a syntax element onto a set of bin strings." Yu, 1353. This occurs for any SE in Yu's system, regardless of whether it is an LPS ("*first category*") or an MPS ("*second category*"). Saber, ¶267. This is confirmed in Yu's explanation that "**[a]fter exploring the binarization modes of different syntax elements** and their occurring frequency during the decoding process, we find there are two classes of syntax elements," including "the first class [that] contain[s] a few bins and occur[s] only once or a few times in one MB" and "the second class [that] contain[s] several bins and occur tens or hundreds of times in one MB." Yu, 1355. Thus, binarization of symbols occurs *prior to* when an MPS ("*the second category of syntax elements*") is processed in "bypass mode" ("*bypassing the context updating*").

c. Claim 6

Yu teaches claim 6. Saber, ¶¶273–77.

Claim 6 is substantially similar to Claim 1[b], except that the threshold be "*predefined*." *See* §VII.C.3.a.iii; *see also* §VII.C.3.a.ii (1[a]).

It would have been obvious to a POSITA that the threshold between *rLPS* and *rMPS* subintervals in Yu's "regular mode" (illustrated below) must be defined before determining whether an incoming bin is to be decoded as *MPS* or *LPS*. Saber, ¶275.

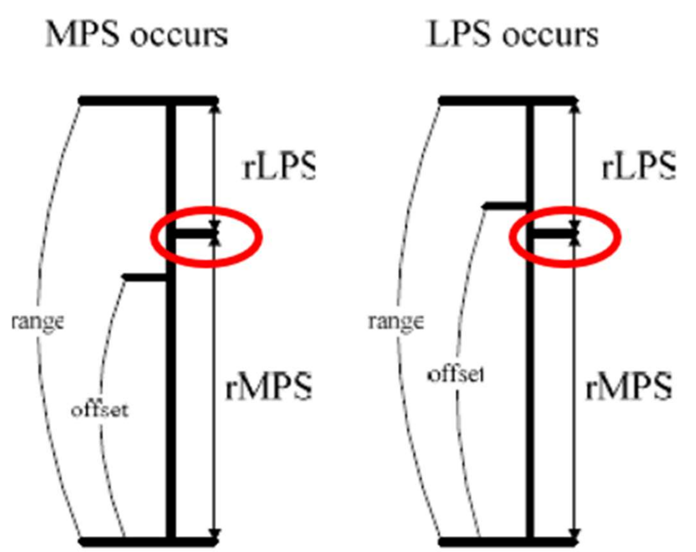


Fig. 2. Arithmetic decoding engine for one bin

Id., Fig. 2.

Yu explains that *rLPS* is calculated as $rLPS = R \cdot p_{LPS}$ (where *R* is interval width, and p_{LPS} is the estimated probability of an *LPS*), and *rMPS* is calculated as $rMPS = R - rLPS$. Yu, 1353. Yu thus provides the calculation for predefining the threshold before categorizing a bin. And for "bypass mode," Yu discloses a predefined threshold of "*range*/2 to determine the bin is *MPS* or *LPS*." Yu, 1354.

d. Claim 7

Yu teaches claim 7 for the reasons discussed for claim 3. *See* §VII.C.2.b; Saber, ¶278. Yu teaches that binarization of symbols occurs *prior to* when an LPS ("*the first category of syntax elements*") is processed in "regular mode" ("*entropy coding the symbols that correspond to the first category of syntax elements*").

3. Claim 8 and Dependents12

a. Claim 8

i. 8[pre]

If limiting, Yu teaches 8[pre]. Saber, ¶¶279–81.

Figure 1 (below) shows "CABAC decoder architecture" with a "Software/hardware interface unit" for "interfacing of the hardware units with the outside software engine" ("*computer program code*") that "provides bit stream to hardware and reads the decoded syntax elements from the output unit." Yu, 1354.

[12] *See supra* fns. 10-11.

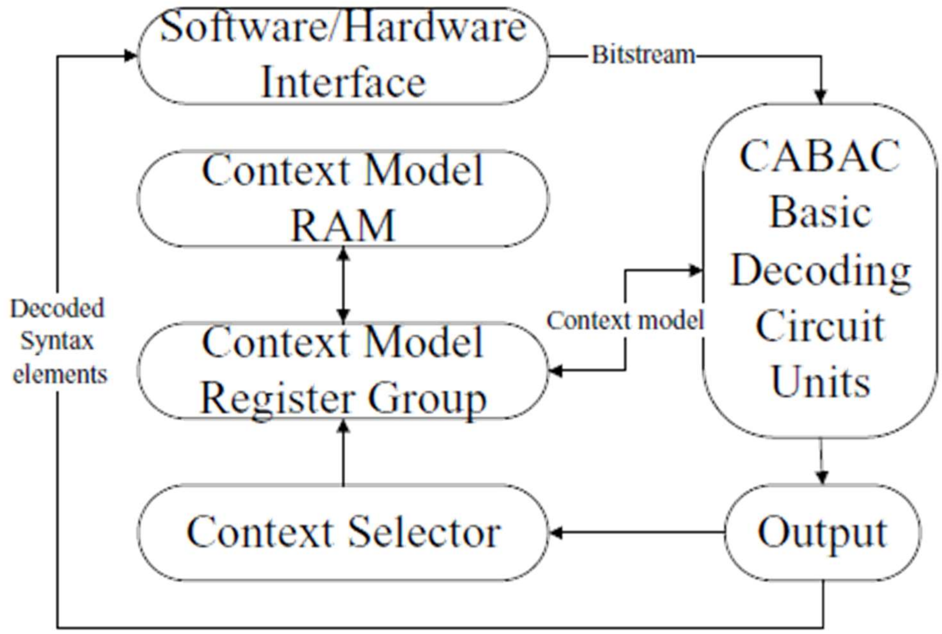


Fig. 1. Framework of CABAC decoder

Id., Fig. 1.

The hardware includes "CABAC Basic Decoding Circuit Units" that form "the processing core of the arithmetic decoding process" and include "high speed decoding engines that can decode several bins per cycle" ("*at least one processor*"). Yu. 1354. "Context Model RAM" is responsible for storing all context models, and "Context Model Register Group" "controls a group of registers, and manages their loading context models from RAM, updating models and writing back models to RAM tasks" ("*at least one memory*"). *Id.* "Context Selector" "selects the correct context model dependent on the neighboring MB information." *Id.* The output of decoded bins is "post-processed and reorganized to reconstruct the syntax elements in the module 'Output', and the software reads the decoded syntax elements from this module." *Id.*

ii. 8[a]

Yu teaches 8[a] for the reasons described for 1[a]. *See* §VII.C.2.a.ii; Saber, ¶282.

iii. 8[b]

Yu teaches 8[b] for the reasons described for 1[b]. *See* §VII.C.2.a.iii; Saber, ¶283.

iv. 8[c]

Yu teaches 8[c] for the reasons described for 1[c]. *See* §VII.C.2.a.iv; Saber, ¶284.

v. 8[d]

Yu teaches 8[d] for the reasons described for 1[d]. *See* §VII.C.2.a.v; Saber, ¶285.

b. Claim 10

Yu teaches claim 10 for the reasons described for claim 3. *See* §VII.C.2.b; Saber, ¶286.

c. Claim 13

Yu teaches claim 13 for the reasons described for claim 6. *See* §VII.C.2.c; Saber, ¶287.

d. Claim 14

Yu teaches claim 14 for the reasons described for claim 7. *See* §VII.C.2.d; Saber, ¶288.

4. Claim 15 and Dependents[13]

a. Claim 15

i. 15[pre]

If limiting, Yu teaches 15[pre] for the reasons described for 8[pre]. *See* §VII.C.3.a.i; Saber, ¶289.

ii. 15[a]

Yu teaches 15[a] for the reasons described for 1[a]. *See* §VII.C.2.a.ii; Saber, ¶290.

iii. 15[b]

Yu teaches 15[b] for the reasons described for 1[b]. *See* §VII.C.2.a.iii; Saber, ¶291.

iv. 15[c]

Yu teaches 15[c] for the reasons described for 1[c]. *See* §VII.C.2.a.iv; Saber, ¶292.

---

[13] *See supra* fns.10-11.

v. **15[d]**

Yu teaches 15[d] for the reasons described for 1[d]. *See* §VII.C.2.a.v; Saber, ¶293.

b. Claim 17

Yu teaches claim 17 for the reasons described for claim 3. *See* §VII.C.2.b; Saber, ¶294.

c. Claim 20

Yu teaches claim 20 for the reasons described for claim 6. *See* §VII.C.2.c; Saber, ¶295.

VIII. CONCLUSION

Inter partes review of the Challenged Claims is respectfully requested.[14]

IX. MANDATORY NOTICES

A. Real Party in Interest

Petitioners identify themselves as real parties in interest.

[14] This petition does not address discretionary issues in accordance with the Acting Director's March 26, 2025 Memorandum on Interim Processes for PTAB Workload Management.

B. Related Matters

To the best of Petitioners' knowledge, the '701 Patent has been involved in the following district court litigation:

- *In the Matter of Certain Video-Capable Laptop, Desktop Computers, Handheld Computers, Tablets, Televisions, Projectors, and Components and Modules Thereof*, Inv. No. 337-TA-1448, USITC (Apr. 11, 2025),
- *Nokia Technologies Oy v. ASUSTeK Computer Inc. et al.*, No. 2:25-cv-03053 (C.D. Cal. Apr. 7, 2025),
- *Nokia Technologies Oy v. Hisense Co. Ltd. et al.*, No. 1:25-cv-01871 (N.D. Ga. Apr. 7, 2025),
- *Nokia Technologies Oy v. Acer Inc. et al.*, No. 1:25-cv-00523 (W.D. Tex. Apr. 7, 2025).

C. Notice of Counsel and Service Information

Lead Counsel
Brian E. Ferguson Reg. No. 36,801 WINSTON & STRAWN LLP 1901 L Street, N.W. Washington, D.C. 20036 T: 202-282-5276 beferguson@winston.com *Attorney for Petitioner Snap Inc.*

Back-Up Counsel	
Eimeric Reig (*pro hac vice* to be submitted) WINSTON & STRAWN LLP 101 California Street, 21st Floor San Francisco, CA 94111 T: 415-591-6808 ereigplessis@winston.com Joseph C. Masullo (*pro hac vice* to be submitted) WINSTON & STRAWN LLP 1901 L Street, N.W. Washington, D.C. 20036 T: 202-282-5897 jmasullo@winston.com Lauren M. Rennecker (*pro hac vice* to be submitted) WINSTON & STRAWN LLP 1901 L Street, N.W. Washington, D.C. 20036 T: 202-282-5897 lrennecker@winston.com *Attorneys for Petitioner Snap Inc.*	Patrick J. McKeever Reg. No. 66,019 PERKINS COIE LLP 11452 El Camino Real, Suite 300 San Diego, CA 92130 T: (858) 720-5722 mckeever-ptab@perkinscoie.com Hisense-NokiaIPR@perkinscoie.com Jon R. Carter Reg. No. 75,145 PERKINS COIE LLP 1155 Ave. of the Americas, 22nd Fl. New York, NY 10036-2711 T: (212) 262-6900 carter-ptab@perkinscoie.com Wei Yuan Reg. No. 71,772 PERKINS COIE LLP 11452 El Camino Real, Suite 300 San Diego, CA 92130 T: (858) 720-5798 yuan-ptab@perkinscoie.com Nicholas Lipperd Reg. No. 83,084 PERKINS COIE LLP 11452 El Camino Real, Suite 300 San Diego, CA 92130 T: (858) 720-5798 lipperd-ptab@perkinscoie.com *Attorneys for Petitioner Hisense USA Corporation*

Dated: August 21, 2025

Respectfully Submitted,

<u>/s/ Brian E. Ferguson</u>
Brian E. Ferguson
Reg. No. 36,801
WINSTON & STRAWN LLC
1901 L Street, N.W.
Washington, D.C. 20036
T: 202-282-5276
beferguson@winston.com

CERTIFICATE OF COMPLIANCE

I hereby certify that this brief complies with the type-volume limitations of 37 C.F.R. § 42.24, because it contains 14,000 words (as determined by the Microsoft Word word-processing system used to prepare the brief and including annotations to the figures), excluding the parts of the brief exempted by 37 C.F.R. § 42.24.

Dated: August 21, 2025

Respectfully Submitted,

/s/ Brian E. Ferguson

Brian E. Ferguson
Reg. No. 36,801
WINSTON & STRAWN LLC
1901 L Street, N.W.
Washington, D.C. 20036
T: 202-282-5276
beferguson@winston.com

CERTIFICATE OF SERVICE

Pursuant to 37 C.F.R. § 42.6(e), I hereby certify that on this 21st day of August, 2025, I caused to be served a true and correct copy of the foregoing and any accompanying exhibits by priority mail on the following:

Nokia Corporation and Alston & Bird LLP
Vantage South End
1120 South Tryon Street
Suite 300
Charlotte, NC 28203-6818

Dated: August 21, 2025

Respectfully Submitted,

/s/ Brian E. Ferguson
Brian E. Ferguson
Reg. No. 36,801
WINSTON & STRAWN LLC
1901 L Street, N.W.
Washington, D.C. 20036
T: 202-282-5276
beferguson@winston.com