

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

AMAZON.COM, INC., AMAZON.COM SERVICES, LLC,

Petitioner

v.

NOKIA TECHNOLOGIES OY,

Patent Owner

Case: IPR2024-01175
U.S. Patent No. 6,856,701

PATENT OWNER'S PRELIMINARY RESPONSE

Mail Stop "PATENT BOARD"
Patent Trial and Appeal Board
U.S. Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450

TABLE OF CONTENTS

I. INTRODUCTION	1
II. BACKGROUND	1
A. TECHNICAL BACKGROUND	6
B. INTRA-PREDICTION AND INTER-PREDICTION	7
C. THE DISCRETE COSINE TRANSFORM.....	8
D. QUANTIZATION OF DCT COEFFICIENTS	12
E. RUN-LEVEL CODING	14
F. VARIABLE LENGTH CODING	16
G. ARITHMETIC CODING.....	20
H. U.S. PATENT NO. 6,856,701	29
I. THE ASSERTED PRIOR ART	32
1. Tsai (Ex. 1005)	33
2. VCEG-L28 (Ex-1006).....	38
3. VCEG-L13 (Ex-1007).....	39
4. Savatier (Ex-1008).....	41
5. AAPA	42
J. POSITA	43
K. CLAIM CONSTRUCTION	43
III. ARGUMENT	44
A. PETITIONER HAS NOT ESTABLISHED A REASONABLE LIKELIHOOD THAT THE CHALLENGED CLAIMS ARE UNPATENTABLE IN LIGHT OF GROUND 1A.	44
1. Claim 1 is not Rendered Obvious in Light of a Combination of Tsai and VCEG-L28.....	44
2. Claim 10 is not Rendered Obvious in Light of Tsai alone.....	53
3. Petitioner’s Arguments Regarding the Remaining Independent Claims Fail for the Same Reasons	55
4. Petitioner’s Arguments Regarding the Dependent Claims fail	56

B. PETITIONER HAS NOT ESTABLISHED A REASONABLE LIKELIHOOD THAT THE CHALLENGED CLAIMS ARE UNPATENTABLE IN LIGHT OF GROUND 1B.	56
C. PETITIONER HAS NOT ESTABLISHED A REASONABLE LIKELIHOOD THAT THE CHALLENGED CLAIMS ARE UNPATENTABLE IN LIGHT OF GROUND 2.	57
1. Claim 1 is not Rendered Obvious in Light of AAPA and Savatier	57
2. Petitioner’s Arguments Regarding the Remaining Independent Claims Fail for the Same Reasons	58
3. Petitioner’s Arguments Regarding the Dependent Claims fail	59
D. PETITIONER HAS NOT ESTABLISHED A REASONABLE LIKELIHOOD THAT THE CHALLENGED CLAIMS ARE UNPATENTABLE IN LIGHT OF GROUND 3.	59
1. Claim 10 is not Rendered Obvious in Light of a Combination of AAPA and VCEG-L28.....	60
IV. CONCLUSION.....	63

TABLE OF AUTHORITIES

	Page(s)
CASES	
<i>In re Fine</i> , 837 F.2d 1071 (Fed. Cir. 1988)	56, 59
<i>Regents of Univ. of Cal. v. Broad Inst., Inc.</i> , 903 F.3d 1286 (Fed. Cir. 2018)	52, 58

TABLE OF EXHIBITS

Exhibit	Description
2001	Declaration of Dr. Doug Chrissan
2002	CV of Dr. Doug Chrissan

I. INTRODUCTION

Petitioner advances three grounds in the Petition. All three grounds fail on the merits.

Petitioner's first Ground is split into Ground 1A and 1B. Ground 1A is based on Tsai (Ex-1005) and VCEG-L28 (Ex-1006). Ground 1B relates to dependent claims and further relies on VCEG-L13 (Ex-1007). Ground 2 is based on a combination of "Applicant Admitted Prior Art" ("AAPA") and Savatier (Ex-1008). Ground 3 relies on AAPA and VCEG-L13.

The '701 Patent relates to context assignment for arithmetic coding in the field of video coding. The Challenged Claims are directed to assigning contexts for transform coefficient information based on previously processed information. In coding transform coefficient information, a first number is used to represent the level value of the coefficient and a second number is used to represent the run value. The level corresponds to the absolute value of the coefficient and the run value corresponds to the number of zero value coefficients either preceding or following the level value in the block. Petitioner's grounds rely on a combination of prior art arithmetic coding techniques and prior art variable length coding techniques. Variable length coding is a separate form of coding from arithmetic coding and the two are mutually exclusive in video coding.

Ground 1A fails for at least four reasons. First, Ground 1A fails to render obvious exemplary claim limitations 1[b] and 10[b]. These same limitations are found in all other independent claims, so Ground 1A also fails to render obvious all other independent claims. Tsai discloses a method of arithmetic coding wherein run and level values are assigned to separate contexts. Tsai discloses a system for switching between the two contexts based solely on the content of the data to be encoded. Additionally, Tsai provides only a single context for run values and a single context for level values. Petitioner relies solely on Tsai to disclose the claim limitation “assigning the first numbers to one of a plurality of contexts representative of the first numbers.”¹ As Tsai discloses only a single context for level values (the first number), Petitioner’s Ground 1A fails to disclose the Challenged Claims. Petitioner offers no combination to satisfy this limitation of the Challenged Claims.

Second, Petitioner relies on a combination of Tsai and VCEG-L28 to satisfy the limitation of “such that the first number of a first number pair is assigned to a context at least partly in dependence on a first number of a second number pair” found in Claims 1, 18, 35, 37, and 39. This argument fails. Petitioner’s proposed

¹ This limitation is present in independent Claims 1, 18, 35, 37, and 39. The remaining independent claims (Claims 10, 27, 36, 38, and 40) contain a similar limitation of “assigning the second numbers to one of a plurality of contexts representative of the second numbers.” Tsai fails to disclose this limitation as well for the same reasons.

combination relates to the selection of a table for symbol mapping, not to the selection of contexts. Petitioner relies on one element of Tsai to disclose context selection for the first half of claim 1[b] and a different element of Tsai to satisfy the limitation of context selection for the second half of claim 1[b]. These two disclosures are distinct processes found within Tsai, relating to different processes. They cannot both be “context assignment.”

Third, Petitioner’s argument relies on a misrepresentation of the prior art references as well as a technically infeasible combination. Tsai relates to arithmetic coding while VCEG-L28 is directed to a form of variable length coding that utilizes multiple codeword tables. Variable length coding is a distinct coding process in video coding and the techniques used in variable length coding are not easily combined with arithmetic coding. Even if a POSITA were able to combine the two references, Petitioner’s proposed combination would not relate to context selection, but rather symbol mapping.

Fourth, Petitioner relies on Tsai alone to satisfy the limitation of “such that the second number of a number pair is assigned to a context at least partly in dependence on the first number of a number pair” found in Claims 10, 27, 36, 38, and 40. Tsai fails to disclose this limitation. Tsai’s context assignment is very clearly based on the content of the data to be coded, not the content of any other coded data. Petitioner admits as much in the Petition. However, Petitioner relies on the fact that

Tsai discloses a method wherein the least significant bit of the level value is coded along with the run value in a single context to support its argument. Petitioner's argument ignores the claim language at issue, which is not limited to a run value assigned a context based on a level value, but rather a second number in a number pair to be assigned a context based, at least in part, on the second number. Tsai's numbers are assigned independently.

Petitioner's Ground 1B addresses only dependent claims and does not remedy the flaws in Ground 1A regarding the independent claims. For these reasons, Grounds 1A and 1B both fail.

Petitioner's Ground 2 relies on a combination of AAPA and Savatier to render obvious Claims 1-9, 18-26, 35, 37, and 39. Petitioner asserts that Context Adaptive Binary Arithmetic Coding (CABAC) is AAPA and proposes a combination with Savatier's variable length coding. CABAC is a prior art arithmetic coding mode discussed in the background of the '701 Patent that utilized set contexts for arithmetic coding. Savatier, much like VCEG-L28, discloses a system for variable length coding using multiple codeword tables. As with Ground 1A, Ground 2 fails because the proposed combination relates to a system for selecting codeword tables, not contexts. Contexts are distinct to arithmetic coding and the codeword tables of Savatier do not perform the same function as context selection in the '701 Patent.

Petitioner's proposed combination would take place and conclude prior to the function of the Challenged Claims in the context of arithmetic coding.

Ground 3 fares no better. Petitioner relies on a combination of AAPA and VCEG-L13 for Ground 3 to disclose claims 10-17, 27-34, 36, 38, and 40. VCEG-L13 discloses a system for context assignment in arithmetic coding wherein the contexts are assigned based on the block type being coded. Petitioner's argument relies on the fact that when a zero value is transmitted for a level value, no run value is coded. This is because VCEG-L13 assigns a zero level value to be the end of block symbol, meaning that no non-zero coefficients remain to be coded in the block. Because the remaining coefficients are zero, no run value needs to be coded. Petitioner asserts that this lack of run value equates to the limitation "such that the second number of a number pair is assigned to a context at least partly in dependence on the first number of the number pair" found in the challenged independent claims. Petitioner's argument is both technically and logically flawed. To begin with, VCEG-L28 is clear that the context assignment for run/level values is based on the block type, not the values to be coded. Second, Petitioner's argument ignores the claim limitations that require a "number pair." The end of block symbol relied upon by Petitioner is a single number to be coded. There is no pair because the run value does not exist with an end of block symbol.

II. BACKGROUND

A. Technical Background

Video compression, or video coding, is directed to transmitting or storing video with reduced file sizes. This is achieved through performing several data processing steps on an input video to produce an “encoded bitstream.” The encoded bitstream is a compressed version of the input video that can be stored and/or transmitted to another device. For video playback, a video decoder performs decompression (“video decoding”) and produces an output video for display. Ex-2001 at ¶ 34.

For some video streaming applications, such as video conferencing or digital broadcast of live events, a system is needed that can perform these steps in real time. This calls for encoding, transmission, and decoding with little to no delay so that the end user perceives no delay in the output video. Ex-2001 at ¶ 35.

Because uncompressed (“raw”) video requires a large number of bits for representation and storage (representing over 8 million pixels for one frame of an 4K video), it is necessary for an encoded bitstream to contain fewer bits. This is because the current bandwidth of communications networks cannot keep up with the high bitrate requirement to transmit raw video. Ex-2001 at ¶ 36.

To achieve this compression, several video coding techniques have been developed. For example, in the case of H.264-compliant video, a compression ratio

of 250:1 can be achieved. However, this compression comes at a cost. Video compression is a “lossy” process, which means that it is not possible to perfectly recreate the raw video from a compressed video. The ultimate goal of video compression is therefore to reduce distortions in the decoded video compared with raw video to an acceptable level while reducing the size of the bitstream as much as possible. An overview of several techniques used in video coding is provided below. Ex-2001 at ¶ 37.

B. Intra-prediction and Inter-prediction

Video coding achieves compression through the exploitation of redundancies in video sequences. There are two main types of redundancy, spatial and temporal. Spatial redundancy refers to correlations between neighboring pixels in a frame. Temporal redundancy refers to redundancies found between frames, such as an object appearing in subsequent frames. Ex-2001 at ¶ 44.

One technique that takes advantage of temporal redundancy is known as “motion compensated prediction.” In motion compensated prediction, the content of one frame is predicted from one or more other frames by tracking the motion of objects or regions that are similar across frames. *See*, Ex-1001 at 3:46-62. Coded pictures using motion compensated prediction are known as INTER coded or P-frames. For INTER frames, motion compensation can generally predict a close, but not exact, recreation of the original frame. To compensate for distortions between

the predicted frame and the original input frame, coders use prediction error (PE) to correct the prediction. Prediction error represents differences between the original input frame and the predicted frame and is added to the predicted frame at the decoder to form the output image. *See*, Ex-1001 at 3:62-4:14, Ex-2001 at ¶ 45.

Frames that are coded using spatial coding are known as Intra-coded or I-frames. INTRA-coded frames use previously coded blocks within the same frame for prediction. This is known as INTRA-prediction. Spatial coding is applied directly to the pixel values of the frame to be coded or the prediction error after INTRA-prediction. *See*, Ex-1001 at 4:15-4:24; Ex-2001 at ¶ 46.

C. The Discrete Cosine Transform

A key component to video coding is the use of block-based transform coding of the prediction error. Transform coding applies a two-dimensional spatial frequency transform to values in a block. This transforms the spatial values into the spatial frequency domain. Typically, this transform utilizes a Discrete Cosine Transform (DCT) or equivalent transform. Ex-2001 at ¶ 47.

Transform coding is capable of accurately representing all values for an $N \times M$ block of pixels through the use of an $N \times M$ block of transform coefficients. As an example, for an 8×8 block of values, the DCT operation outputs an 8×8 array of transform coefficients, accurately representing the input values. Transform coding is a reversible process wherein the transform coefficients may be used as an input to

an inverse DCT operation. The inverse operation will perfectly return the original values. Ex-2001 at ¶ 48. Transform coding is commonly performed on prediction error in video coding with the prediction error values being used as the input values. Ex-2001 at ¶ 48.

As an example, the equation to calculate a set of DCT coefficients (as an N x N array $\{F(u,v), u = 0 \text{ to } N-1, v = 0 \text{ to } N-1\}$ from a two-dimensional input array $\{f(i,j), i = 0 \text{ to } N-1, j = 0 \text{ to } N-1\}$ is provided below:

$$F(u, v) = \frac{2}{N} C(u) C(v) \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f(i, j) \cos\left(\frac{(2i+1)u\pi}{2N}\right) \cos\left(\frac{(2j+1)v\pi}{2N}\right)$$

where

$$C(x) = \begin{cases} \frac{1}{\sqrt{2}} & , x = 0 \\ 1 & \text{otherwise} \end{cases}$$

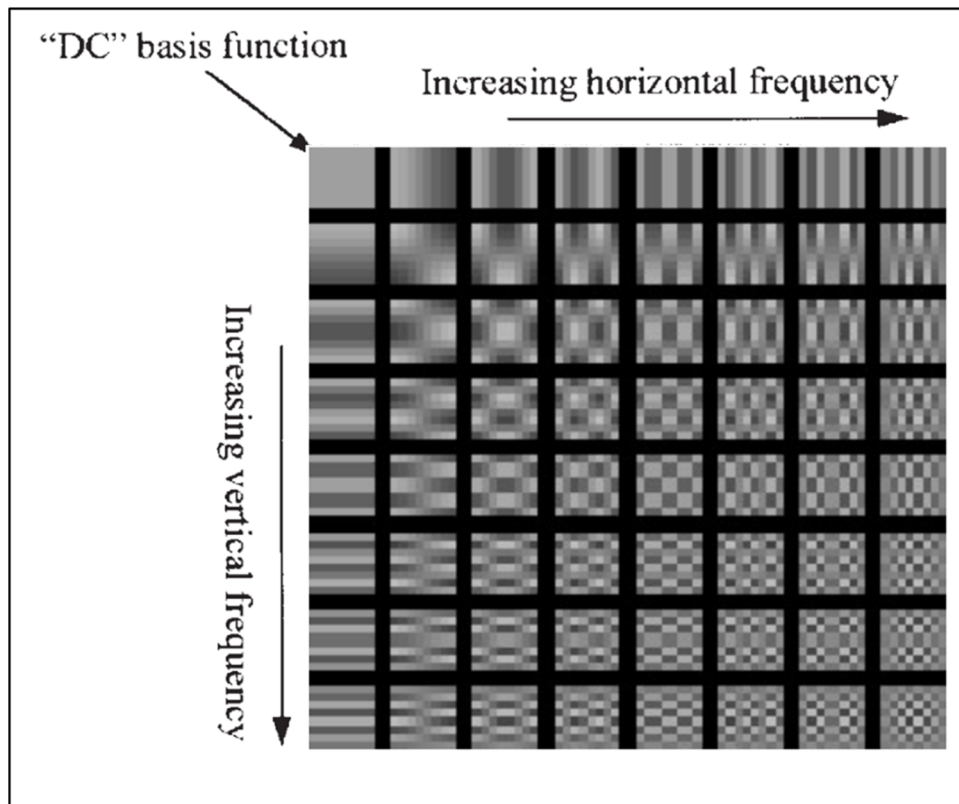
Ex-1022, p. 23; Ex-2001 at ¶ 49. As noted above, this transform process is reversible through the use of an “inverse DCT” operation. To reverse the DCT, the N x N array of transform coefficients $F(u,v)$ is input into the equation below to recover the original N x N array.

$$f(i, j) = \frac{2}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} C(u) C(v) F(u, v) \cos\left(\frac{(2i+1)u\pi}{2N}\right) \cos\left(\frac{(2j+1)v\pi}{2N}\right)$$

with $C(x)$ as above.

Ex-2001 at ¶ 49; Ex-1022, p. 23.

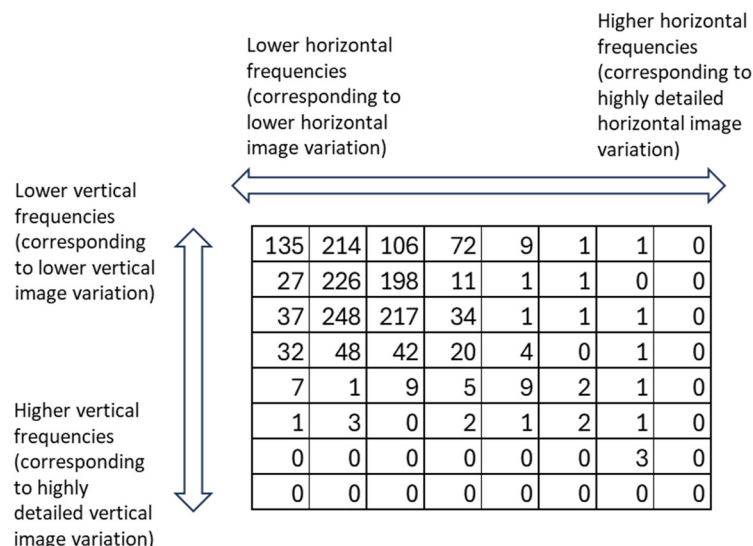
The DCT process transforms the input information from the spatial domain to the spatial frequency domain, with the spatial frequency domain information being represented by the transform coefficients. The transform coefficients represent weightings for a set of N^2 “basis functions.” These basis functions are $\cos\left(\frac{(2i+1)u\pi}{2N}\right)\cos\left(\frac{(2i+1)v\pi}{2N}\right)$, $u = 0$ to $N-1$, $v = 0$ to $N-1$ in the equations above. For an 8×8 block, the 64 resulting basis functions can be visualized below. Ex-2001 at ¶ 50.



Ex-2001 at ¶ 50, Ex-1002, p. 22, Figure 3.3 (modified). In this figure, the u values (starting at 0 on the left and progressing through 7 on the right) are represented by the “Increasing horizontal frequency” arrow. The v values (starting at 0 on the top

and progressing through 7 on the bottom) is represented by the “Increasing vertical frequency” arrow. Ex-2001 at ¶ 51. The basis functions are represented by the patterns shown in each location within the array.

The basis functions for transform coefficients are arranged so that the lowest frequency functions are included in the top left corner of the block, as shown in the illustration below. The lower frequency basis functions represent details that are more noticeable to the human visual system and are more significant for a suitable recreation of the transformed block of data. Coefficients in the lower right region typically represent higher frequency basis functions which are not as noticeable to the human visual system. An exemplary table of transform coefficients created by Dr. Chrissan is provided below:



The diagram illustrates an 8x8 transform coefficient array. A horizontal double-headed arrow at the top indicates frequency progression from left to right. A vertical double-headed arrow on the left indicates frequency progression from top to bottom. The array is an 8x8 grid of numerical values.

Lower horizontal frequencies (corresponding to lower horizontal image variation)				Higher horizontal frequencies (corresponding to highly detailed horizontal image variation)							
Lower vertical frequencies (corresponding to lower vertical image variation) Higher vertical frequencies (corresponding to highly detailed vertical image variation)				135	214	106	72	9	1	1	0
				27	226	198	11	1	1	0	0
				37	248	217	34	1	1	1	0
				32	48	42	20	4	0	1	0
				7	1	9	5	9	2	1	0
				1	3	0	2	1	2	1	0
				0	0	0	0	0	0	3	0
				0	0	0	0	0	0	0	0

Ex-2001 at ¶¶ 51-54.

The transform coefficients are generated by the encoder and sent to the decoder in lieu of sending the pixel values for each location within a block. The transform coefficients represent a “weighting” factor for each basis function. This means that, when the basis functions are combined using the appropriate weighting function, the original pixel values are recreated. This operation is performed several thousands of times per second in modern video coding contexts due to the large number of blocks that comprise higher definition videos. Ex-2001 at ¶¶ 55-55.

Transform coding allows for the efficient representation of blocks of values because more important information for the human visual system is found in the lower frequency coefficients. This allows for large values to be concentrated in fewer coefficients. Additionally, many coefficients result in zero or close to zero values. This efficiency is further improved with quantization and run-level coding. Ex-2001 at ¶ 56.

D. Quantization of DCT Coefficients

Typically, an encoder will perform a process known as quantization to reduce the precision of the transform coefficients included in a bitstream. Quantization reduces number of the bits in the bitstream but also lowers the decoded image quality. This is because quantization is a “lossy” process, meaning that a perfect recreation of the input values is no longer achievable upon decoding. Ex-2001 at ¶ 57.

In a simplified quantization process, quantization can be achieved by rounding an input value to the nearest multiple of N. In this scenario, N is referred to as the quantization level. As an example of this process, if the input values have a possible range of 0-99, a quantization level of 10 could be used. This would approximate any value 0-9 as zero, 10-19 as 10, 20-29 as 20, and so on. This results in 10 possible values that need to be encoded instead of the original 99, which takes fewer bits to signal. As such, the quantization operation reduces the precision but also improves compression. Ex-2001 at ¶ 58.

When applied to transformed pixel values or prediction error data, the quantization process will typically result in numerous zero valued coefficients. The zero values are usually found in the areas representing higher frequency basis functions because the higher frequency basis functions tend to have smaller values initially. Dr. Chrissan performed an exemplary quantization on the transform coefficients provided above:

Transform Coefficients								
135	214	106	72	9	1	1	0	
27	226	198	11	1	1	0	0	
37	248	217	34	1	1	1	0	
32	48	42	20	4	0	1	0	
7	1	9	5	9	2	1	0	
1	3	0	2	1	2	1	0	
0	0	0	0	0	0	3	0	
0	0	0	0	0	0	0	0	

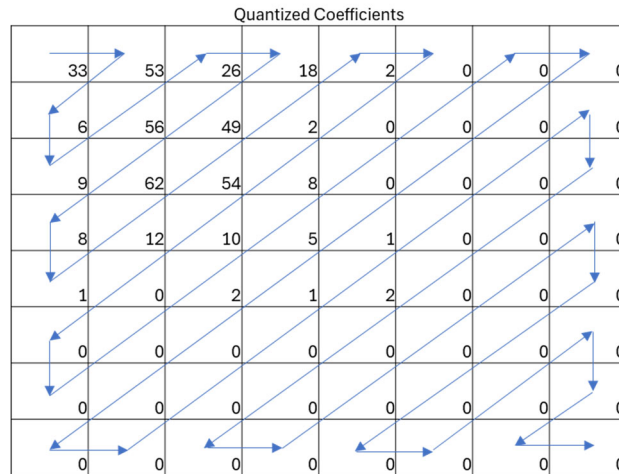


Quantized Coefficients								
33	53	26	18	2	0	0	0	
6	56	49	2	0	0	0	0	
9	62	54	8	0	0	0	0	
8	12	10	5	1	0	0	0	
1	0	2	1	2	0	0	0	
0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	

Ex-2001 at ¶ 59. In the above example, the transform coefficients in the left block are quantized by rounding to the nearest multiple of 4 and then divided by 4. This results in the quantized coefficient block on the right. Looking to a specific example, the top left value is 135 before quantization. In this case, 135 is rounded down to the nearest multiple of 4 (132), which is divided by 4 to arrive at 33. This results in the quantized coefficient value of 33. On the decoder side, the decoder will perform inverse quantization by multiplying 33 by 4 which results in a value of 132. While 132 is close to the original value of 135, it is not a perfect match, so some error (distortion) is introduced in the process. This process is performed for all values in the block. Consequently, any transform coefficient value that is less than four is quantized to zero. Ex-2001 at ¶ 59.

E. Run-level Coding

After prediction, transform coding, and quantization is performed, the coder has a compressed representation of the data to be transmitted. However, additional compression can be achieved through the way this data is represented in the bitstream. Certain coding techniques, such as run-length coding or other techniques, are used to take advantage of the statistics of the data to be put in the bitstream. When processing transform data for insertion into the bitstream, typically a “zigzag” scan pattern is used as shown below:



Ex-2001 at ¶ 60. The '701 Patent provides a similar scan pattern for a 4x4 block of data in Fig. 6a:

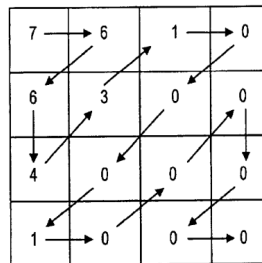


FIG. 6a

Ex-1001 at Fig. 6a, 14:46-65; Ex-2001 at ¶ 60.

The zigzag scanning pattern often results in long runs of consecutive zeros between non-zero coefficients and a long run of zeros at the end of the scan. The long run of zeros at the end results from the fact that the higher frequency components are located at the end of the scan and tend to be lower valued and quantized to zero. To efficiently code this kind of data, video coders utilize “run-level” or “run length” coding. Run-level coding provides a two-value entry in a one-

dimensional array for every non-zero quantized coefficient value. These two values are known as the “level” and “run”. The level value represents the value of the non-zero quantized coefficient and the run value is the number of consecutive zero-value coefficients preceding (or following) the coefficient in question. A level value equal to zero is used as an EOB (end-of-block) symbol, which indicates there are no more non-zero coefficient values in the block. Ex-2001 at ¶ 61.

The '701 Patent provides an example of run-level coding the block of Fig. 6a in Fig. 6b. This example provides an example with run values before (top) and after the level value (bottom):

07	06	06	04	03	01	31	0
70	60	60	40	30	10	13	0

↑ Run
↑ Level

FIG. 6b

Ex-1001 at Fig. 6b, 14:46-65; Ex-2001 at ¶ 63.

F. Entropy Coding

Once the run-level values are determined by the encoder, they must be inserted into the bitstream, along with all other pertinent information for decoding the video, for transmission to the decoder. At a fundamental level, the bitstream is a sequence of binary values (1s and 0s). This means that any parameter values (such

as the run-level coded values described above) must be converted into an efficient binary representation. To achieve this, coders use a technique known as “entropy coding.” Entropy coding is a lossless process that takes symbols as an input and outputs a binary representation of the input data. Because entropy coding is a lossless process, the decoder is able to recover the exact values that were encoded. Ex-2001 at ¶ 64.

Entropy coding relies on probabilities of occurrence of symbols within a data set. This means that, instead of using a fixed number of bits to represent each symbol, a variable number of bits can be assigned to each symbol. Typically this is done so that symbols that are more likely to occur are represented by shorter binary code-words. In run-level coding, this means that values that are more likely to occur will be assigned shorter code-words. Ex-2001 at ¶¶ 65-67.

G. Variable Length Coding

Entropy coding can be implemented in a variety of ways. One common means for entropy coding run-level values is through the use of look-up tables which define the mapping between the symbols in the data set and a corresponding set of variable length codewords. This is known as Variable Length Coding (VLC). The look-up tables are typically created through the use of statistical analysis of training material that has similar statistical properties as the data to be coded. Ex-1001, 6:33-40; Ex-2001 at ¶ 68.

VCEG-L28 provides an example of such a VLC look-up table, with two possible tables available to the coder.

Table 1. The first nine codewords of UVLC and UVLC2

Index	UVLC	UVLC2
0	1	11
1	001	001
2	011	011
3	00001	1001
4	00011	1011
5	01001	00001
6	01011	00011
7	0000001	01001
8	0000011	01011

Ex-1006 at 1. In this example, there are nine possible codewords in both the UVLC and UVLC2 look-up tables. This means that the tables are capable of representing all possible input values of a given integer “X” that is to be coded, allowing for an integer range of zero through eight for X. Ex-2001 at 69-70.

Coding the value of X without the use of VLC would require four bits: a zero is 0000, a one is 0001, and so on through eight being 1000. This is the binary representation of the value. However, the use of the UVLC table provides a new mapping of the value of X to bits. Using the UVLC table, if X has an equal probability to be any given value, then the average number of bits required to code X is $1/9 (1 + 3 + 3 + 5 + 5 + 5 + 5 + 7 + 7) = 41/9 = 4.56$ bits. This calculation is performed by adding up the total number of bits required to represent every value and dividing by the number of possible values. In this scenario, where all values have an equal probability, the VLC method does not provide any bit savings.

However, different probabilities can lead to different results. As one example, assume that the probability of a zero value for X is 60%, and the probability of one to eight is 5% each. Calculating the average number of bits in this scenario results in: $(0.6 \times 1) + (0.05 \times (3 + 3 + 5 + 5 + 5 + 5 + 7 + 7)) = 2.6$ bits. This is a significant saving over four bits. Ex-2001 at ¶ 71.

Using the UVLC2 table provided in VCEG-L28 also affects the savings. For example, the equal probability case results in $1/9 (2 + 3 + 3 + 4 + 4 + 5 + 5 + 5 + 5) = 36/9 = 4$ bits, providing the same bit requirement as no coding. For the second scenario, using the 60% probability for a zero and 5% probability for each other value, the average bits required is $(0.6 \times 2) + (0.05 \times (3 + 3 + 4 + 4 + 5 + 5 + 5 + 5)) = 2.9$ bits. Ex-2001 at ¶ 72.

The use of variable length codewords is made possible because no codeword begins with another codeword. As an example, in the UVLC column, zero is encoded with a “1” bit and no other codeword starts with a “1.” It is also true that no other codeword in the list starts with any other codeword. Thus, each codeword is uniquely identifiable, allowing the decoder to recognize when it has reached the end of a codeword and should treat the next incoming bit as the first bit of a new codeword. Ex-2001 at ¶ 73.

Deriving benefit from the use of VLC depends on several factors. One such factor is the probabilities of the possible values of an input value (e.g., as described

in the example above, an equal probability of 5% for all non-zero values as opposed to 60% probability for zero). Another factor is the development of a VLC table that allows for efficient coding of the given probabilities (such as the UVLC v. UVLC2 tables of VCEG-L28). This means that the efficient use of VLC requires a significant investment in research and experimentation to analyze probabilities in representative video sets and develop optimal table mappings. For example, VCEG-L28 is a proposed improvement to known VLC coding modes. Ex-2001 at ¶ 74.

VLC coding is a relatively simple method of entropy coding that maps possible input data values to unique coded bit fields. The tables must be known to both the encoder and decoder to function properly, but VLC has a relatively low computational complexity. Ex-2001 at ¶75.

H. Arithmetic Coding

Other methods of entropy coding also exist. One such method is known as Arithmetic Coding. Arithmetic coding can be used instead of VLC coding to perform lossless coding of video parameters. Instead of using lookup tables, arithmetic coding uses fractional values between zero and one as codewords. This results in a more computationally complex process than VLC. Ex-2001 at ¶ 69. Additionally, arithmetic coding takes into account an entire block of data during coding. This is because probability estimates used in arithmetic coding are updated throughout the

process. On the other hand, VLC addresses data elements one at a time and does not generate new codewords based on other data within the block. Ex-2001 at ¶ 76.

Dr. Chrissan provides an example of arithmetic coding in his declaration regarding the decimal value of 0.345. The value 0.345 is the same as the fraction 345/1000. It can also be represented as $3 \times 10^{-1} + 4 \times 10^{-2} + 5 \times 10^{-3}$. Additionally, the decimal value of 0.345... falls within the range of [0.345, 0.346).² This means that no matter what additional digits follow the “5” of 0.345, the resulting value is still within the given range (i.e., it is always smaller than 0.346). As another example, consider the value 0.2... Any higher order decimal value provided after the “2” will be within the range of [0.2, 0.3). The principle is true for any other decimal number. Arithmetic coding uses this logic to map all possible input symbols to unique ranges between [0 and 1). When the arithmetic coder must determine a codeword to represent a set of input data symbols, it will choose the shortest fractional value within that input symbol’s assigned range. This is how arithmetic coding generates codewords. Ex-2001 at ¶ 77.

On the decoding side, the mapping between symbols and range must be known as well. This means that the encoder and the decoder must use the same

² In this example, “...” represents anywhere between zero and an infinite number of additional decimal digits after the 5, “[“ means inclusive and “)” means not inclusive.

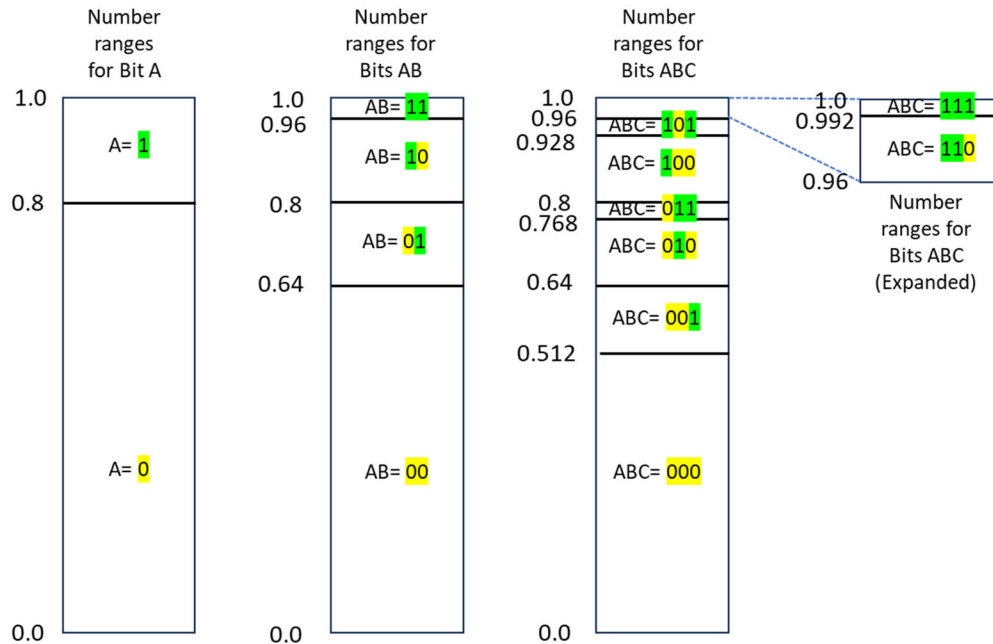
symbol mapping to allow for accurate decoding of the arithmetically coded values. Ex-2001 at ¶ 78.

While the examples above use decimal fractions, video coders typically use binary fractions for arithmetic coding. An exemplary binary fraction is 0.10010011, which is equivalent to the base-10 fraction $147/256$. This means, regardless of what higher order binary digits follow 0.10010011, the resulting value will still be in the base 10 range of $[147/256, 148/256]$. This allows for the codeword “10010011” to represent a particular input symbol sequence that has an assigned mapping range that wholly encompasses $[147/256, 148/256)$. Additionally, the single digit codeword “0” could be used to represent an input symbol sequence that has an assigned mapping that wholly includes $[0, 0.5)$. Ex-2001 at ¶ 79.

This means that, generally, wider range mapping allows for shorter codewords. As such, arithmetic coding typically maps more probable input symbol sequences to wider number ranges allowing them to be represented by shorter codewords.. Ex-2001 at ¶ 80.

An example process of arithmetic coding is described below. In this example, three binary digits (bits) ABC (which can each have either a value of 0 or 1) are input values to the arithmetic coder. The arithmetic coder will code the three ABC bits as a single representation through a sequential process, coding A first, then B, then C. The example assumes that the probability for a zero bit is 80% and the

probability of a one is 20%. The diagram provided by Dr. Chrissan below depicts the mapping for eight possible values of ABC.



Ex-2001 at ¶ 81. In this example, if bit A has the value 0 it is mapped to the range [0, 0.8) because it is known, according to the statistics of the data set to which A belongs, the probability of a zero value is 80%. If bit A has the value 1, it is mapped to the range [0.8, 1) representing the 20% probability of a 1 occurring (as illustrated in the left-hand column in the figure above). Using different probabilities (e.g., a 50/50 split) would result in different mappings. When coding bit B, the A = 0 range is further divided depending on whether the combination of the A and B bit values is AB = 00 or AB = 01 (see second column in the figure above). When coding the value of bit C, a similar subdivision is then performed for each possible combination

of the values of C with the previously coded values of A and B (as shown in the third column of the figure above). The third column shows the final numerical codeword ranges for the eight possible combinations of input values A, B and C (000, 001, 010, 011, 100, 101, 110 and 111). The number ranges represented in the diagram are summarized below:

Three bit input:	Range beginning:	Range end:
000	0	0.512
001	0.512	0.64
010	0.64	0.768
011	0.768	0.8
100	0.8	0.928
101	0.928	0.96
110	0.96	0.992
111	0.992	1

Ex-2001 at ¶¶ 82-84.

In order to facilitate accurate coding and decoding, both the encoder and decoder need to know of these ranges. This is achieved through either 1) using a pre-determined ranges inherent to the design of the algorithm, 2) by including them in the encoded bitstream ahead of the codewords, and/or 3) adjusting them correspondingly at both the encoder and decoder throughout the coding process based on the actual count of previously transmitted ones and zeros, so as to update the probabilities and the corresponding ranges in real-time. Adjusting the ranges in real-time is known as “Context-based Adaptive Binary Arithmetic Coding,” or “CABAC.” Ex-2001 at ¶ 84; *See also* Ex-1001 at 12:35-49.

Using the table of ranges determined above, the encoder then determines the following coding words for the possible ABC values:

ABC bits	Codeword	Codeword length
000	0	1
001	1001	4
010	1011	4
011	110010	6
100	1101	4
101	111100	6
110	111110	6
111	1111111	7

Ex-2001 at ¶ 85. For $ABC = 000$, a one bit codeword of “0” is assigned because any binary fraction that starts with 0.0 is within the range of $[0, \frac{1}{2})$. This range is wholly within the range $[0, 0.512)$, which is the range allocated to $ABC = 000$ in the range table above. On the other hand, the single bit “1” cannot be used as a codeword since $0.1\dots$ (representing the base 10 range of $[\frac{1}{2}, 1)$), does not fall wholly within any of the defined ranges. The next available codeword is 1001, representing the binary decimal 0.1001, or base 10 $\frac{9}{16}$ (representing the base 10 range of $[\frac{9}{16}, \frac{10}{16})$ or $[0.512, 0.64)$). This binary decimal 0.1001 falls wholly within the range set for $ABC = 001$, and thus is available as a codeword. Ex-2001 at ¶ 85.

Understanding that, in the example above, the probability of each possible ABC value is determined by the size of its associated range, the average number of bits for a codeword is the probability of each of the eight possible ABC values times its corresponding codeword’s length: $0.512 \times 1 + 0.128 \times 4 + 0.128 \times 4 + 0.032 \times 6$

$+ 0.128 \times 4 + 0.032 \times 6 + 0.032 \times 6 + 0.008 \times 7 = 2.68$ bits. This results in a significant savings over the 3 bits that would otherwise be required to send each uncompressed value. Ex-2001 at ¶ 86.

In this example, three bits (ABC) are coded. The three bits can represent e.g. a single integer having a possible value of 0 to 7 (the binary representation of 0 being 000 and the binary representation of 7 being 111). Each of the three individual bits (A, B, and C) are considered a “bin” and each bin has a separate probability that the bin will be a 1 or 0. This probability can be represented with a probability model associated with the bin. In this example, all three bins have the same probability of a 0. Because the bins share the same probability of a 0, the bins can be grouped together in a context, which is a grouping of bins with similar statistical characteristics.. However, if the probability for one of the bins were to change, a new context would be needed for that bin. Notably, in prior art solutions, contexts were not tailored to the input data in a way that allowed for the most efficient means of probability estimation. Rather, contexts were pre-assigned to specific bins. Ex-2001 at ¶ 87.

1. Contexts

An integer can be mapped to bins and symbols in multiple ways. Video encoders and decoders must know the mapping used. As the '701 Patent describes:

CABAC is a binary arithmetic coding method and thus data

symbols to be coded which have non-binary values are first converted to binary values ('binarized') in binary mapping block 710. The binarization process involves mapping a symbol to be coded to a sequence of bins, each of which has a corresponding bin number and can be assigned a value of either 0 or 1. An example of such a mapping is given below in Table 1. In principle other binarization schemes can be envisaged.

TABLE 1

value		Bin Sequence							
0	1								
1	0	1							
2	0	0	1						
3	0	0	0	1					
4	0	0	0	0	1				
5	0	0	0	0	0	1			
6	0	0	0	0	0	0	1		
...									
bin_nr.	1	2	3	4	5	6	7		

In the CABAC method each of the bins is assigned to a so-called 'context' (hence the name context-based arithmetic coding). A context can be thought of as grouping together bins, which have similar statistical characteristics. In other words, each bin assigned to a particular context is assumed to have a similar probability of containing the value 1 or 0 as the other bins belonging to that context. In this way, the probability estimates used to generate code-words in the arithmetic coder are defined for each context rather than for each possible bin to be encoded. Each context is defined according to a 'context model', established in advance and based on information about the statistical characteristics of the data symbols (and thus the bins) to be encoded.

Ex. 1001, 12:42–13:10.

In the Table 1 example of the '701 patent, a parameter value is mapped to a sequence of bins prior to CABAC encoding. Ex-2001 at ¶¶ 88-89.

The arithmetic coding takes place after this symbol mapping by assigning contexts to the bins. The assigned contexts represent the probability of a one or zero in each bin. Contexts group together bins of similar statistical characteristics and provide information relating to the probability assignments for the bins. In this way, arithmetic coding is significantly different than variable length coding and much more complex. VLC provides a one-to-one mapping between input data symbols and codewords. Arithmetic coding, on the other hand, maps multiple data symbols to a single codeword. Furthermore, context-adaptive binary arithmetic coding utilizes a complex arrangement of bins and contexts to represent the probabilities of individual symbols and combinations of symbols. Because of this, concepts used in symbol mapping for VLC cannot easily be adapted for use in context-adaptive binary arithmetic coding. Ex-2001 at ¶ 85.

The following simple example is provided for explanation purposes. In the English language, the letter “q” generally has a low probability of occurrence. However, when a “q” does occur, it is almost always followed by the letter “u.” In prior art arithmetic coding methods, where inter-symbol dependencies and contexts were not considered, occurrence of a “q” would lead to an increase in the probability estimate for a “q” occurring, i.e. the probability model would treat “q” appearing as

an indication that “q” is more likely to appear. However, as is apparent from this simple English language example, the occurrence of a “q” does not make “q” any more likely to occur, but instead indicates with a high probability that the next symbol to appear will be a “u”. Thus, an example “context” specific to the English language would increase the probability estimate of a “u” occurring after a “q”, rather than increasing the probability estimate of another “q” occurring. The contexts of the ’701 Patent are adapted for run-level coding in a similar, albeit much more complex, manner.

I. U.S. Patent No. 6,856,701

U.S. Patent No. 6,856,701 claims priority to U.S. Provisional Application No. 60/322,112, which was filed on Sept. 14, 2001. The ’701 Patent describes “new context models for the coding of DCT transform coefficients, which take into account [] relationships between level and run values.” Ex-1001 at 16:21-24. This allows more efficient entropy coding of transform coefficients. This improves upon prior art methods because “certain relationships exist between the run and level values associated with DCT transform coefficients.” Ex-1001 at 11:25-36; Ex-2001 at ¶ 91.

Prior to the inventions claimed in the ’701 Patent, arithmetic coding was typically performed using a Context Adaptive Binary Arithmetic Coding (CABAC) process. In CABAC, arithmetic coding is achieved through the binarization of data

symbols into bins. The bins are assigned different contexts based on the statistical characteristics of the bins (probability of having a 1 or a 0). Contexts are defined via a “context model,” which is “established in advance and based on information about the statistical characteristics of the data symbols (and thus the bins).” Ex-1001 at 13:5-10. Context models are defined prior to coding and, for H.26L CABAC, four contexts are used for level values and three contexts are used for run values. For level values, the first context corresponds to bin 1, the second to bin 2, the third to all remaining bins, and the fourth context is used for the sign of the level. Ex-1001 at 14:14-24. For run values, the first context is used for the first bin, the second context is used for the second bin, and the third context is used for all remaining bins. Ex-1001 at 14:20-24. Additionally, the contexts have a “probability estimate” assigned to them. The probability estimates represent the statistical characteristics of the bins assigned to the context in question (e.g., the likelihood of a bin assigned to the context containing a 1 or a 0). While probability estimates can be updated throughout encoding, the assignment of contexts to bins remains fixed. Ex-1001 at 13:20-43; Ex-2001 at ¶¶ 92-95.

The inventor of the '701 Patent realized that video coding could exploit certain potential relationships between the run and level values associated with DCT transform coefficients, such that new context models could be created to exploit this relationship. In H.26L's CABAC method, most run and level values were coded in

context 3, creating a large amount of overlap within context 3 and minimizing the probability that a given bin would contain a 1 or a 0. Ex-1001 at 14:66-15:15. A lower probability estimate for a context creates a need to send more information for a coded symbol. Because of this, it is advantageous to maximize the probability estimates of a context with adaptive context models. Ex-1001 at 12:65-13:19. Ex-2001 at ¶ 96.

The '701 Patent improves upon the prior art context assignment systems with improved context models that account for the relationships between level and run values. Ex-1001 at 16:5-24. The improved system allows for probability estimates to be adapted specifically for run-level coding based on the assigned context. The '701 Patent provides examples of how the context assignments can be adapted to run-level coding:

[W]ithin a given block of transform coefficients, the level of a coefficient to be encoded has, in general, a magnitude substantially similar to the level of the previously encoded coefficient. The inventors have also determined an inverse relationship between the level and run values. In particular, larger level values are more likely to be preceded by smaller run values. The converse is also true, namely smaller level values are likely to be preceded by larger runs. Consequently, the present invention proposes the creation of new context models for the coding of DCT transform coefficients, which take into account these relationships between level and run values.

In a first such context model, intended for implementation in a context-based arithmetic encoder, the context assigned to the bins of a binarized coefficient level value depends on the previously encoded coefficient level. In a second such context model, intended for implementation in a context-based arithmetic decoder, the context assigned to the bins of a binarized coefficient level value depends on the previously decoded coefficient level. In a third context model, implemented in either a context-based arithmetic encoder or a context-based arithmetic decoder, the context assigned to the bins of a binarized coefficient run value depends on the coefficient's level value. Ex-1001 at 16:10-36.

Rather than assigning contexts based on bin numbers, the '701 Patent assigns contexts based on information of coefficient level and run values within the block. This allows the contexts to be tailored to the data being coded and exploit additional statistical relationships. Ex-2001 at ¶ 96.

By adapting context models throughout the coding of transform coefficient values, the contexts can be tailored to fit the data being coded, thus reducing bitstream overhead.

J. The Asserted Prior Art

Petitioner relies on five prior art references in its petition, Tsai, VCEG-L28, VECEG-L13, Savatier, and AAPA. Petitioner does not assert that any reference alone anticipates the '701 Patent. Instead, Petitioner relies on various combinations

of prior art references. Every proposed combination fails to disclose one or more limitations of the challenged claims and any modifications alleged by Petitioner would not have been obvious.

1. Tsai (Ex. 1005)

U.S. Patent No. 5,818,877 (“Tsai”) claims priority to U.S. Provisional Application No. 60/013,605, which was filed on Mar. 14, 1996. Tsai relates to “data compression, and in particular, to compression of run/level coded data or other sequential, digital data.” Ex-1005 at 1:13-15. Tsai focuses on “representing a stream of data values, wherein the data values have variable numbers of significant digits.” Ex-1005 at 2:58-60. Tsai teaches that this is achieved through “representing a most significant digit of each data value with a symbol from a first symbol set, and representing each significant digit other than the most significant digit of each data value with a symbol from a second symbol set.” Ex-1005 at 2:61-65. Relevant here, Tsai includes a “context analyzer” that tells the entropy coder which context to use based on the type of data being coded. Ex-2001 at ¶ 99.

Tsai teaches that “the symbols comprise two pairs” with the first pair representing “binary 1’s and 0’s of run lengths, and its two members appear with greater frequency in the first context” and the second pair representing “binary 1’s and 0’s of the second context, related to the nonzero levels.” Ex-1005 at 4:26-37. In other words, Tsai teaches to code run/level pairs using a first context for run lengths

and a second context for level lengths. The symbols representing the run/level values also include information regarding the least significant digits of the corresponding run or level value. Tsai discloses mapping run/level pairs to symbols through the use of two contexts: one for run and one for level. Ex-1005 at 4:6-5:9; Pet. at 10; Ex-2001 at ¶ 100.

Tsai also teaches that, when mapping the run/level pairs to symbols, “it is desirable that shorter words be used to represent run values that have a higher probability of occurrence.” Ex-1005 at 10:35-39. Additionally, it is advantageous to “perform the mapping such that the symbols comprising words appear with unequal probability.” Ex-1005 at 10:42-45. Tsai illustrates this point by providing Figure 5 with two Symbol Maps:

502 SYMBOL MAP 1		504 SYMBOL MAP 2
RUN VALUE	WORD	WORD
1	+	+
2	-	-
3	++	++
4	--	+-
5	+-	-+
6	-+	--
7	+++	+++
8	---	++-
9	+--	+-+
10	-+-	-++
11	++-	+-
12	--+	-+-
13	+-+	--+
14	-++	---
15	++++	++++
16	----	+++
.	.	.
.	.	.
.	.	.

Ex-1005 at Fig. 5.

Figure 5 “illustrates two examples of symbol maps used to map symbol groups to run values.” Ex-1005 at 5:57-58. Tsai explains that “[f]or typical image and sound data, the run-length coder **108** is more likely to produce runs that are

small,” as such, “[s]ymbol maps **1** and **2 (502, 504)** in FIG. 5 takes advantage of this fact and associates small run values with short words.” Ex-1005 at 35-41. Symbol Map 2 is also arranged such that, “among any group of words having the same number of symbols, words associated with the smallest run values contain the most “+” symbols.” Ex-1005 at 10:49-52. This facilitates a mapping where symbols comprising words appear with unequal probability. Tsai provides a similar comparison of symbol maps for level values in Figure 7. Notably, these symbol map comparisons are unrelated to the choice of contexts discussed further in Tsai. Ex-2001 at ¶ 100.

Tsai discusses the use of two contexts for use in “the entropy coding step that follows the mapping to symbols.” Ex-1005 at 4:44-50. Tsai teaches that these contexts change based on the category of information being coded (i.e., one context used for run values and another used for level values). These contexts are not chosen based on information of other run/level values. For example, Tsai illustrates a context selector in Fig. 11:

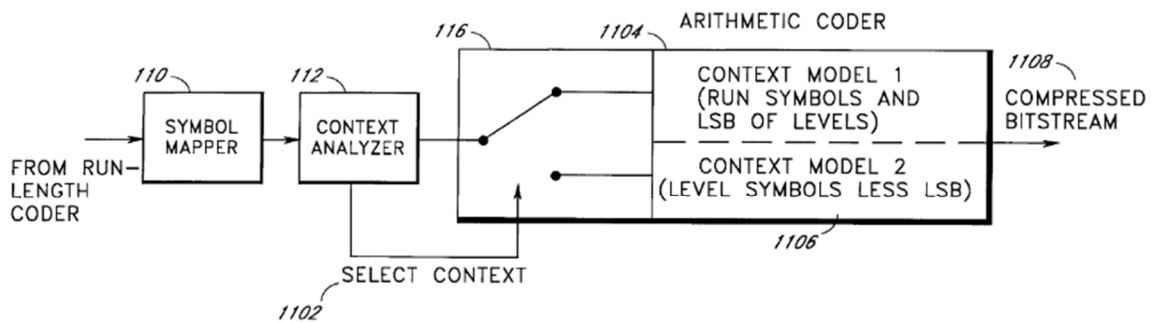


FIG. 11

Ex-1005 at FIG. 11.

Figure 11 depicts “the flow of data from the symbol mapper **110** into the context analyzer **112** and then into the arithmetic coder **116**.” Ex-1005 at 14:65-67. The context analyzer “examines the input symbols and detects changes in context (run context or level context).” Ex-1005 at 15:2-5. The context analyzer then performs two functions. First “the context analyzer **112** partitions the symbol stream into two different types of symbol groups ... by breaking up the symbol stream into groups of symbols each of which is slightly different than a run word or a level word.” Ex-1005 at 15:7-14. Second, the context analyzer “controls the context of arithmetic coding.” *Id.* In other words, the context analyzer tells the entropy coder which context to use based on the type of data being coded. Ex-2001 at ¶ 100.

2. VCEG-L28 (Ex-1006)

International Telecommunication Union-Telecommunications Sector (“ITU-T”), *VLC concatenation for Intra coding* (2001), https://www.itu.int/wftp3/av-arch/video-site/0101_Eib/VCEG-L28.doc (“VCEG-L38”) is a Video Coding Experts Group proposal submitted to ITU-T in January of 2001. Ex-2001 at ¶ 103.

VCEG-L28 describes a method to improve the “universal-VLC [(Variable Length Coding)] approach of H.26L.” Ex-1006 at 1. This purported improvement is achieved through “switching between two VLC tables.” *Id.* VCEG-L28 provides two VLC tables: UVLC and UVLC2.

Index	UVLC	UVLC2
0	1	11
1	001	001
2	011	011
3	00001	1001
4	00011	1011
5	01001	00001
6	01011	00011
7	0000001	01001
8	0000011	01011

Ex-1006 at 1. The two UVLC tables provide a mapping of the input value to codeword symbols to be used in entropy coding. VCEG-L28’s “switching applies

to Intra luma coefficients in double scan.” Ex-1006 at 2. VCEG-L28’s switching logic is reproduced below:

4. Switching between UVLC and UVLC2

The switching applies to Intra luma coefficients in double scan with the following rules:

- First coefficient, first scan: use UVLC2 if the above or left block is unavailable or if $\text{abs}(\text{dc_above}) + \text{abs}(\text{dc_left}) > 2$, where $\text{dc}=1$ is used outside the present macroblock.
- First coefficient, second scan: use UVLC2 if $\text{abs}(\text{dc}) > 2$ or if first scan had >3 coefficients.
- Second coefficients: use UVLC2 if $\text{abs}(\text{previous_level}) > 2(1)$ for first (second) scan.
- Third through sixth coefficients: use UVLC2 if $\text{abs}(\text{previous_level})/(\text{previous_run}+2) > 0$.

Ex-1006 at 2. This results in the possibility to switch the symbol mapping for input variables for the second through sixth coefficients in a block. However, VCEG-L28 does not relate to assigning distinct contexts to symbols or input variables. Ex-2001 at ¶ 104.

3. VCEG-L13 (Ex-1007)

ITU-T, *Adaptive Codes for H.26L* (2001), https://www.itu.int/wftp3/av-arc/video-site/0101_Eib/VCEG-L13.doc (“VCEG-L13”) is a VCEG proposal submitted to ITU-T in January of 2001.

VCEG-L13 describes a method to improve entropy in H.26L through the implementation of Context Adaptive Binary Arithmetic Coding (CABAC) instead of the “universal variable length code (UVLC) used in the ... current H.26L test model.” Ex-1007 at 1. VCEG-L13 provides “a short overview of the main coding

elements or [the] proposed entropy coding scheme.” Ex-1007 at 1. VCEG-L13 also provides that the “process of constructing a model is commonly referred to as context modeling and it depends in [the] approach on the given syntax element.” Ex-1007 at 1. Ex-2001 at ¶ 103.

VCEG-L13 provides context models for run/level in section 3.3. Run/level coding is dependent upon “the scanning mode, the DC/AC block type, the luminance/chrominance, and the intra/inter macroblock decision.” Ex-1007 at 4. Based on these criteria, there are 9 different block types, “which, in turn, results in 9 different contexts.” Ex-1007 at 4. The context for a given block is determined by the block type. Within each block of a given type, every run/level value coded will be assigned the same context. VCEG-L13 provides a table illustrating this point:

<i>ctx_run_level</i>	Block Type
0	Double Scan
1	Single Scan, Inter
2	Single Scan, Intra
3	Intra16x16, DC
4	Intra16x16, AC
5	Chroma, DC, Inter
6	Chroma, DC, Intra
7	Chroma, AC, Inter
8	Chroma, AC, Intra

Table 1 : Numbering of the different context models used for coding of RUN and LEVEL

Ex-1007 at 4; Ex-2001 at ¶ 104.

VCEG-L13 goes on to disclose that, based on the context for a given block, “four different models are chosen” for coding the level values. EX-1007 at 5. The

first model is used for coding the sign of the level value while “the remaining 3 models are used for the first, the second and all remaining bins of the binarization of LEVEL.” VCEG-L13 at 5. VCEG-L13 also uses a level value of 0 to indicate an “end of block,” (no remaining non-zero level values remain in a given block).

Where the level value is not equal to zero, the corresponding run value is coded using “two separate models ... one model for the first bin and the second model for all remaining bins.” Ex-1007 at 5. Where the level value is equal to zero (signifying the end of block), no run value is coded. Ex-1007 at 5.

To summarize, VCEG-L13 proposes a method for coding run/level values where the contexts are assigned based on the coding mode of the block in question. Where an end of block signal is sent for the level value, there is no corresponding run value to be coded and the run value coding is skipped.

4. Savatier (Ex-1008)

U.S. Patent No. 5,400,075 (“Savatier”) was filed on January 13, 1993 and issued on March 21, 1995. Savatier discloses a coding system “for adaptively providing VLC changes without additional signal overhead, includ[ing] a plurality of VLC encoding tables.” Ex-1008 at Abstract. Ex-2001 at ¶ 105.

Savatier discloses a system in which a plurality of VLC encoding tables are provided with selected “VLC codes associated with anticipated signal statistics.” Ex-1008 at 2:6-10. Within each signal, there are “blocks of data subject to common

parameters which parameters may be associated with signal statistics” and the blocks “include information indicative of such parameters.” Ex-1008 at 2:10-14. These parameters inform the coder to determine which VLC coding/decoding table is assigned to a given block. Ex-2001 at ¶ 105-106.

Savatier also discloses an additional embodiment, wherein “the VLC table used for each successive codeword is made dependent upon the table utilized for the lastmost codeword and the relative value of the lastmost codeword.” Ex-1008 at 2:15-19. Savatier explains that this VLC table selection is “dependent upon the block type ... and the information contained in respective codewords.” Ex-1008 at 21-25. Ex-2001 at ¶ 107.

Notably, Savatier is directed to the assignment of codewords to unique input values, not the assignment of probabilities or contexts for unique bins to contain a 1 or 0.

5. AAPA

The purported AAPA relied upon by Petitioner refers to Context Adaptive Binary Arithmetic Coding. This is a form of arithmetic coding that functions similarly to the overview of arithmetic coding provided in § II. G.

K. POSITA

Patent Owner has applied Petitioner's POSITA for the sake of this Response. (Pet., 9.) This Response does not depend on this skill level. Patent Owner reserves the right to argue in favor of different level of skill should trial be instituted.

L. Claim Construction

Petitioner did not propose any claim constructions. For the sake of this Response, Patent Owner does not propose any claim constructions. No constructions are necessary for Patent Owner to demonstrate validity of the '701 Patent. Patent Owner reserves the right to propose claim constructions in the future.

While no construction is necessary, and Petitioner proposes no construction, Patent Owner notes that Petitioner relies on a definition of "context" that is at odds with the specification of the '701 Patent and how a POSITA would understand the term. Petitioner equates Savatier's codewords table selection to a context assignment. Pet. at 70. However, the '701 Patent is clear that a context relates to statistical characteristics of bins, not codewords. *E.g.*, Ex-1001 at 12:67-13:1. This Preliminary Response discusses Petitioner's mistake in equating Savatier's codewords table selection to a context assignment in the section on Ground 3, below.

III. ARGUMENT

A. Petitioner has not Established a Reasonable Likelihood that the Challenged Claims are Unpatentable in Light of Ground 1A.

Petitioner has failed to show that Claims 1, 10, 18, 27, and 35-40 are obvious in light of Tsai and VCEG-L28 and therefore failed to show that all claims depending from Claim 1 (Claims 2-11) and Claim 10 (Claims 11-17), Claim 18 (Claims 19-26), and Claim 27 (Claims 28-34) are obvious.

1. Claim 1 is not Rendered Obvious in Light of a Combination of Tsai and VCEG-L28

Petitioner relies on a combination of Tsai and VCEG-L28 to show that Claim 1[b] is rendered obvious. Petitioner's argument relies on the idea that a POSITA would have been motivated to combine Tsai with VCEG-L28 and would have had a reasonable expectation of success. However, Tsai discloses a system related to arithmetic coding, while VCEG-L28 discloses a system related to variable length coding. As noted above, these are distinct coding techniques that are mutually exclusive in the realm of video coding. Ex-2001 at ¶ 118. As such, a POSITA would not have been motivated to combine the two, nor arrive at the necessary modification to disclose claim 1[b].

Claim 1[b] is reproduced below:

assigning the first numbers to one of a plurality of contexts representative of the first numbers such that the first number of a first number pair is assigned to a context at

least partly in dependence on a first number of a second pair.

Ex-1001 at Claim 1; Ex-2001 at ¶ 110.

Petitioner relies on Tsai alone to satisfy the first half of this limitation and a combination of Tsai and VCEG-L28 to satisfy the second half. Both arguments fail.

a) Tsai alone does not disclose “assigning the first numbers to one of a plurality of contexts representative of the first numbers”

To begin, Claim 1[b] requires a “plurality of contexts representative of the first numbers.” A context is a component of arithmetic coding alone and is wholly unrelated to variable length coding. As the ’701 Patent explains, a “context can be thought of a grouping together bins, which have similar statistical characteristics.” Ex-1001 at 12:67-13:2. The ’701 Patent teaches that, regarding a run/level pair, the “first number” is the level value. Ex-1001 at 6:12-14. Therefore, Claim 1[b] requires that there be a plurality of contexts representative of the level value for a run/level pair. Ex-2001 at ¶ 114.

Petitioner asserts that Tsai “teaches *assigning* the level words in the level group (*the first numbers*) to the level context of the two contexts (*one of a plurality of contexts representative of the first numbers*).” Pet. at 26. However, Tsai does not teach “a plurality of contexts representative of the first numbers.” Rather, Tsai

teaches a single context for run values and a single context for level values. Petitioner acknowledges this in providing an overview of Tsai.³

Tsai discloses a specific mapping of (run, level) pairs to symbols and the use of two contexts: one for run and one for level.

Pet. at 10 (citing Ex-1006 at 4:6-5:9; Ex-2001 at ¶ 114).

Petitioner's reliance on the process depicted in Figure 11 is misguided. As Tsai's Figure 11 illustrates, the two proposed contexts are assigned based on "changes in context (run context or level context)." Ex-1005 at 15:2-5. Context Model 1 is representative of the run symbols and Context Model 2 is representative of the level symbols.

³ It should be noted that Tsai teaches to code the run value plus the least significant bit of the level value and separately code the level value less the least significant bit. For the sake of this section of the Response, Patent Owner uses Petitioner's assigned terminology to the terms.

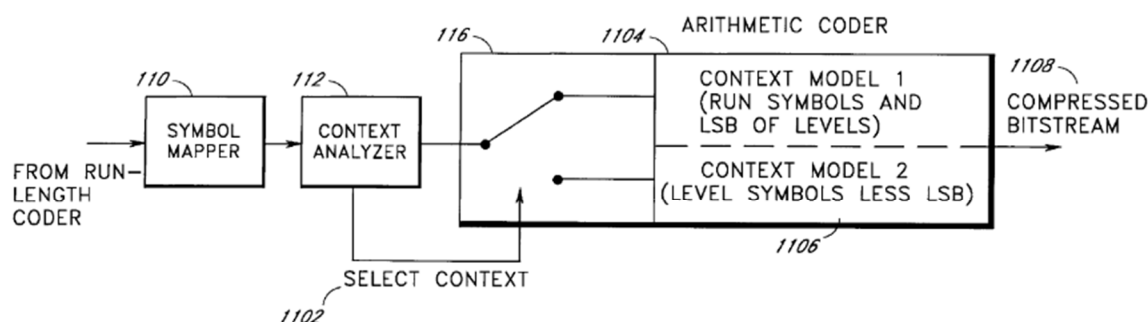


FIG. 11

Ex-1005 at FIG. 11; Ex-2001 at ¶ 111.

Petitioner’s own citations regarding Figure 11 confirm this. Petitioner states that “when the context analyzer determines ‘a group of symbols comprising a level word with no LSB,’ the context analyzer ‘issues a second select context command 1102 to the arithmetic coder 116’ to cause ‘the arithmetic coder 116 to use the context model 1106 it is adapting to code groups of symbols consisting of a level word having no LSB.’” Pet. at 26 (quoting Tsai at 6:12-18). In other words, when the context analyzer sees a level symbol, it selects the level context.

Because Tsai teaches only a single context representative of the first numbers, it fails to disclose “assigning the first numbers to one of a plurality of contexts representative of the first numbers.” Petitioner offers no additional argument to show disclosure of this claim limitation. As such, Petitioner failed to show that Claim 1[b]

is rendered obvious under Ground 1. Additionally, Petitioner's combination to satisfy the second half of Claim 1[b] fails as well.

b) Petitioner's proposed combination of Tsai and VCEG-L28 fail to disclose the limitation of "the first number of a first number pair is assigned to a context at least partly in dependence on a first number of a second number pair"

Petitioner's argument regarding the second half of limitation 1[b] departs from the teachings of Tsai and proposes a combination of two distinct coding methods.

i. Tsai's symbol mapper is unrelated to context selection

Petitioner points to a combination of Tsai's "symbol mapper" and VCEG-L28's *previous_level* criteria to disclose this limitation. Pet. at 29. This combination is flawed on multiple fronts. First, Petitioner points to Tsai's context analyzer and arithmetic coder to satisfy the assignment of contexts in the first half of limitation 1[b], which is incorrect, as discussed above. Second, Tsai's symbol mapper does not select between two symbol maps. Third, in Tsai's exemplary coder, the symbol mapper operates prior to the context analyzer and context assignment. Ex-2001 at ¶ 112.

As discussed in § III. A. 1. a., Petitioner points to the Context Analyzer 112 and Arithmetic Coder 118 of Figure 11 to perform the step of assigning a context to

a first number. Petitioner also refers to Context Model 1 and Context Model 2 as the plurality of contexts representative of the first number. Pet. at 26.

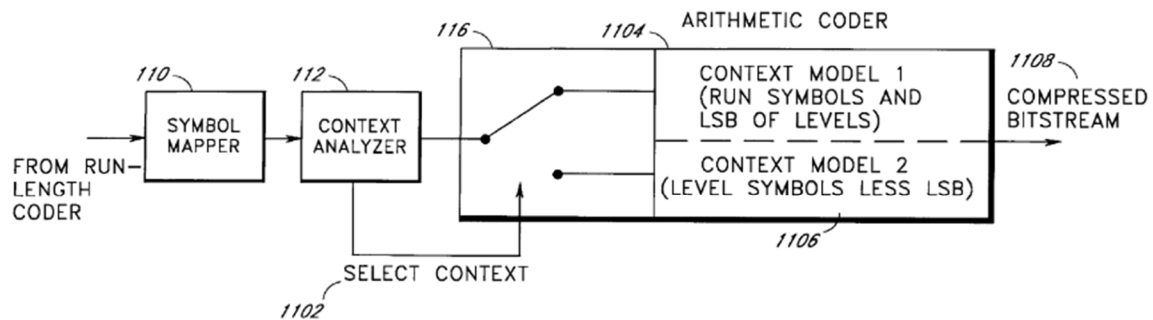


FIG. 11

Ex-1005 at FIG. 11.

However, in an about face, Petitioner then makes the argument that Tsai's symbol mapper 110 performs the process of context assignment and contains the plurality of contexts. Petitioner attempts to brush over this fact by describing a combination where "Tsai's context assignment (i.e., selection between Symbol Map 1 or Symbol Map 2) is based on the *previous_level* criteria taught in VCEG-L28." Pet. at 29. This is inconsistent with Petitioner's previous reasoning (*see* Pet. at 26) and inconsistent with Tsai. For example, in Figure 11, the symbol mapper 110 comes before Context Analyzer 112, the flow path for Select Context 1102, and the two contexts provided in the arithmetic coder. Ex-2001 at ¶ 112.

Petitioner's logic also relies on Tsai supporting the use of two distinct symbol maps during coding. However, Petitioner makes no attempt to explain how Tsai discloses a system for this. Petitioner instead focuses on the fact that Tsai describes two possible symbol maps that may be advantageous in certain coding scenarios. However, Tsai is clear that these symbol maps are "examples" of different maps that may be used in the system described in Tsai. Ex-1006 at 5:58-63. At no point does Tsai discuss switching between the two symbol maps during coding. Petitioner states that "selection of the symbol table based on the previous level affects the output symbol stream and thereby also affects the context assignment." Pet. at 29. Petitioner makes no attempt to elaborate on or cite support for this premise. Petitioner's unsupported premise is contrary to the system disclosed in Tsai. Ex-2001 at ¶ 112.

Tsai makes clear that the context assignment is dependent upon whether the content to be encoded relates to the run symbols and least significant bit of the levels (Context Model 1) or to the level symbol less the least significant bit (Context Model 2). *See* Ex-1006 at Fig. 11. Changing the symbol map used to map the values to symbols will not change this context assignment. Ex-2001 at ¶ 112.

Petitioner fails to provide a feasible combination of Tsai and VCEG-L28. Even with Petitioner's proposed combination, Petitioner still fails to show that the entirety of Limitation 1[b] is rendered obvious because Petitioner relies on different components of Tsai to satisfy the context assignment throughout the limitation.

Additionally, Petitioner has failed to show that a POSITA would combine the two references.

ii. A POSITA would not be motivated to combine and/or modify Tsai and VCEG-L28 and would not have a reasonable expectation of success in doing so

Arithmetic coding and Variable Length Coding are distinct coding techniques that do not operate together. In fact, variable length coding and arithmetic coding modes are mutually exclusive in video coding because they both perform the same function according to different operating principles with tradeoffs in complexity and compression. Ex-2001 at ¶ 118.

A POSITA would have understood that using a variable length coding table in arithmetic coding would only affect the assignment of input variables to symbols. This is a distinct process from assigning a context to the symbols for use in arithmetic coding. Additionally, a combination of arithmetic coding and VLC would have created increased complexity. Conventionally, VLC coding was regarded as less efficient, but overall less computationally complex than arithmetic coding. This is one reason why H.264 allows for compliant bitstreams that rely on one of arithmetic coding or VLC. Encoders with less computational power were able to use VLC coding at the cost of efficiency, while high end encoders could use arithmetic coding to increase compression efficiency. The combination proposed by the Petitioner

would disregard the known trade-offs of the two techniques. Ex-2001 at ¶ 118. The combination proposed by Petitioner would in effect be the worst of all possible worlds and not something a POSITA would reasonable expect to succeed.

Even if a POSITA were motivated to combine the arithmetic coding of Tsai with the Variable Length Coding of VCEG-L18, any combination would still require modification to disclose the Challenged Claims. As discussed above, the proposed combination of using VCEG-L28's *previous_level* criteria with Tsai's symbol map would have no impact on the context assignments. This is because the symbol map is assigned prior to any context analysis and the only context analysis present in Tsai is related to whether the values are directed toward run or level values. Rather than modifying Tsai so that multiple symbol maps may be used determined by VCEG-L28's *previous_level* criteria, a POSITA would need to somehow add multiple contexts and an additional switching mechanism into the Context Assignment of Tsai. Petitioner offers no explanation as to how such a modification might be achieved, why a POSITA would be motivated to make such a modification, or why a POSITA would have a reasonable expectation of success in undertaking to make such a modification.. Ex-2001 at ¶119. As a result, Ground 1 fails. *See Regents of Univ. of Cal. v. Broad Inst., Inc.*, 903 F.3d 1286, 1291 (Fed. Cir. 2018) (“An obviousness determination requires finding that a person of ordinary skill in the art

would have been motivated to combine or modify the teachings in the prior art and would have had a reasonable expectation of success in doing so.”)

2. Claim 10 is not Rendered Obvious in Light of Tsai alone

As discussed above with regards to Claim 1[b], Tsai does not disclose “a plurality of contexts.” Claim 10[b] provides a similar limitation to claim 1[b]:

assigning the second numbers to one of a plurality of contexts representative of the second numbers such that the second number of a number pair is assigned to a context at least partly in dependence on the first number of the number pair.

Ex-1001 at Claim 10.

A main distinction between Claims 1 and 10 is that Claim 10[b] is directed to the assignment of the second number (e.g., run value) based on the assignment of the first number (e.g., level value) as opposed to the assignment of the first number based on a first number of a second pair. Petitioner’s arguments regarding 10[b] closely resemble the arguments for 1[b] and fail as well for corresponding reasons. Ex-2001 at ¶ 114.

To begin with, Tsai does not disclose “a plurality of contexts representative of the second numbers.” This is because Tsai only discloses a single context representative of the run value plus the least significant bit of the level. There is only one context available for run values. Petitioner admits as much:

That is, the second numbers (the run words in the location/run group) are assigned to the run context.

Pet. at 39.

This inevitable assignment to the run context also contradicts Petitioner's argument that Tsai discloses "the second number of a number pair is assigned to a context at least partly in dependence on the first number of the number pair." Petitioner draws attention to the fact that Tsai's run symbol also includes the least significant bit (LSB) of the level value. This entire symbol is assigned to the "run" context. Petitioner attempts to use this LSB as its basis for showing that the context assignment is made dependent upon the first number. However, Petitioner ignores the fact that the LSB is included in the second number in Tsai. The LSB and run value are coded as a symbol and placed into a context together. There is no dependency upon the first number, which in Tsai's case is the level value less the LSB. Ex-2001 at ¶ 115.

Consider this in the scenario of English language coding discussed above in Section II.H.1, wherein contexts may be assigned to specific letters based on the probability of the next occurring letter. Tsai's selection of context depends not on the prior coded letter, but rather what category of information is being coded. For example, Tsai would more closely resemble a system wherein a first context is assigned to the first sentence of a paragraph and a second context is assigned to all

following sentences. The contexts are not dependent on the values coded, but rather the categorization of information provided to the coder.

3. Petitioner's Arguments Regarding the Remaining Independent Claims Fail for the Same Reasons

Claims 18, 35, 37, and 39 all contain the same limitation of “assigning the first numbers to one of a plurality of contexts representative of the first numbers such that the first number of a first number pair is assigned to a context at least partly in dependence on a first number of a second number pair” found in Claim 1. Petitioner's additional arguments regarding these limitations do not remedy the shortcomings of the arguments made regarding Claim 1. Indeed, Petitioner repeatedly incorporates its Claim 1 arguments into the additional claim sections. Because Petitioner failed to show that this limitation was met for claim 1, its arguments regarding the additional claims fail as well.

Claims 27, 36, 38, and 40 all contain the same limitation of assigning the second numbers to one of a plurality of contexts representative of the second numbers, and operative to assign the second number of a number pair to a context at least partly in dependence on the first number of the number pair” found in Claim 10. Petitioner's additional arguments regarding these limitations do not remedy the shortcomings of the arguments made regarding Claim 10. Indeed, Petitioner repeatedly incorporates its Claim 10 arguments into the additional claim sections.

Because Petitioner failed to show that this limitation was met for claim 10, its arguments regarding the additional claims fail as well.

4. Petitioner’s Arguments Regarding the Dependent Claims fail

Petitioner’s arguments regarding Claims 2-9, 11-14, 17, 19-26, 28-31 and 34 fail because Petitioner failed to show that the independent claims from which these claims depend are obvious. Petitioner failed to show that Claims 1, 10, 18 and 27 are obvious in light of Tsai and VCEG-L28, Petitioner’s arguments regarding the independent claims fail as well. *See In re Fine*, 837 F.2d 1071, 1076 (Fed. Cir. 1988) (indicating that “[d]ependent claims are nonobvious under section 103 if the independent claims from which they depend are nonobvious.”).

B. Petitioner has not Established a Reasonable Likelihood that the Challenged Claims are unpatentable in Light of Ground 1B.

In addition to Tsai and VCEG-L28, Petitioner’s Ground 1B further relies on VCEG-L13 (Ex-1007). Ground 1B only relates to dependent claims 15-16, and 32-33 and accordingly fails for the same reasons as Ground 1A. Therefore, Ground 1B rises and falls with Ground 1A. *See In re Fine*, 837 F.2d 1071, 1076 (Fed. Cir. 1988) (indicating that “[d]ependent claims are nonobvious under section 103 if the independent claims from which they depend are nonobvious.”)

C. Petitioner has not Established a Reasonable Likelihood that the Challenged Claims are Unpatentable in Light of Ground 2.

Petitioner has failed to show that Claims 1 and 18, 35, 37, and 39 are obvious in light of AAPA and Savatier and therefore failed to show that all claims depending from Claim 1 (Claims 2-9) and Claim 18 (Claims 19-26) are obvious.

1. Claim 1 is not Rendered Obvious in Light of AAPA and Savatier

Petitioner's reliance on a combination of Savatier and AAPA is misguided. Savatier discloses a system for variable length coding while the purported AAPA relates to Context Adaptive Binary Coding. As already discussed, VLC coding and Arithmetic coding are distinct coding methods. The methods used in each are not interchangeable. Ex-2001 at ¶ 122.

The methods of symbol mapping described in Savatier do not relate to the probability of a value occurring in a context and adapting context assignment as necessary for context-adaptive arithmetic coding. Rather, Savatier describes a method of choosing a variable length code for symbols. This is wholly unrelated to the choice of contexts in arithmetic coding. *See Supra* § II. G; Ex-2001 at ¶ 123.

Petitioner is incorrect that "Savatier's table selection is context assignment." Pet. at 70. Petitioner refers back to § VI.C.2 to support this argument, however the term "context" does not even appear in the referenced section. *See Pet.* at 60-

62. This is unsurprising, as Savatier itself makes no mention of “contexts.” *See generally*, Ex-1008. This is because the system of Savatier has no need for contexts. Savatier discloses variable length coding, a coding mode which is distinctly devoid of contexts. Instead, VLC relies on unique codewords to signal values. Petitioner’s reliance on Savatier’s “table selection” as a context assignment is misguided and technically incorrect. Ex-2001 at ¶ 124.

Moreover, even if a POSITA would have been motivated to combine CABAC and Savatier as Petitioner suggests, the resulting system would not assign one of a plurality of contexts to a first number based on the first number of a second pair. Instead, it would determine what symbol would be used to represent an input value prior to the context assignment. This is because the VLC tables do not assign a context but rather a symbol to be coded. Ex-2001 at ¶ 127.

Petitioner also fails to demonstrate that why a POSITA would have a reasonable expectation of success in combining AAPA and Savatier. As a result, Ground 2 fails. *See Regents of Univ. of Cal. V. Broad Inst., Inc.*, 903 F.3d at 1291.

2. Petitioner’s Arguments Regarding the Remaining Independent Claims Fail for the Same Reasons

Claims 18, 35, 37, and 39 all contain the same limitation of “assigning the first numbers to one of a plurality of contexts representative of the first numbers such that the first number of a first number pair is assigned to a context at least partly in

dependence on a first number of a second number pair” found in Claim 1. Petitioner’s additional arguments regarding these limitations do not remedy the shortcomings of the arguments made regarding Claim 1. Indeed, Petitioner repeatedly incorporates its Claim 1 arguments into the additional claim sections. Because Petitioner failed to show that this limitation was met for claim 1, its arguments regarding the additional claims fail as well.

3. Petitioner’s Arguments Regarding the Dependent Claims Fail

Petitioner’s arguments regarding Claims 2-9 and 19-26 fail because Petitioner failed to show that the independent claims from which these claims depend are obvious. Petitioner failed to show that Claims 1, and 18 are obvious in light of Savatier and AAPA, Petitioner’s arguments regarding the independent claims fail as well. *See In re Fine*, 837 F.2d 1071, 1076 (Fed. Cir. 1988) (indicating that “[d]ependent claims are nonobvious under section 103 if the independent claims from which they depend are nonobvious.”).

D. Petitioner has not Established a Reasonable Likelihood that the Challenged Claims are Unpatentable in Light of Ground 3.

Petitioner has failed to show that Claims 10, 27, 36, 38, and 40 are obvious in light of AAPA and VCEG-L13 and therefore failed to show that all claims depending from Claim 10 (Claims 11-17) and Claim 27 (Claims 28-34) are obvious.

1. Claim 10 is not Rendered Obvious in Light of a Combination of AAPA and VCEG-L28

Petitioner's argument regarding a combination of VCEG-L13 and AAPA is flawed for three reasons. First, Petitioner attempts to broaden the Challenged Claims beyond their ordinary scope. Second, Petitioner mischaracterizes the disclosures of VCEG-L13. Third, even with Petitioner's broadening of the Claims and mischaracterizations of VCEG-L28, Petitioner's proposed combination does not disclose the Challenged Claims. Ex-2001 at ¶ 130.

Petitioner's argument depends on VCEG-L13's end of block symbol. In VCEG-L13, the end of block is represented with a level value of 0. EX-1007 at 5. When the end of block symbol is received, there is no need to code a run value; this is why VCEG-L13 only codes the run value if LEVEL≠0. Ex-1007 at 5. However, coding a run value and assigning a context to a run value are distinct concepts. Petitioner's argument relies on broadening the claim language of Claim 10[b] so that the term "assigning" is read as "coding." Petitioner equates the two in its argument.

VCEG-L13 teaches that the run value of a run-level pair (*the second number of a number pair*) is coded only when the value of the same run-level pair (*the first number of the number pair*) is not equal to 0. VCEG-L13, §3.3.1 ('If LEVEL≠0 (EOB), the corresponding RUN is coded in a subsequent coding routine.') In other words, assigning a context model to the run value in a run-level pair (*the second number of a number pair is assigned to a context*) is selectively performed based on the [sic] whether the level value in the same run-level pair is equal to 0 (*at least*

partly in dependence on the first number of the number pair).

Pet. at 91-92; Ex-2001 at ¶ 131.

Coding run/level pairs based on the already assigned contexts is distinct from the process of assigning the contexts. Under Petitioner's interpretation of the Challenged Claims, the lack of a value altogether is equal to making a determination for context assignment. Ex-2001 at ¶ 132.

Petitioner mischaracterizes the context selection in VCEG-L13. Petitioner asserts that the context assignment for the run value is based on the level value. This is contrary to the disclosures of VCEG-L13. In VCEG-L13, contexts are assigned based on the characteristics of the block in question i.e., at the "block level" in coding hierarchy, whereas the claims relate to the coding of level and run values within a given block.

Coding of (RUN, LEVEL) pairs is conditioned on the scanning mode, the DC/AC block type, the luminance/chrominance, and the intra/inter macroblock decision. Thus, a total number of 9 different block types are given according to Table 1, which, in turn, results in 9 different contexts.

The contexts assigned based on block type are then used in the coding process. EX-1007 at 5 ("For a given block C, the LEVEL information is first separated into sign and magnitude according to its context *ctx_run_level(C)*"), ("For each context *ctx_run_level(C)* two separate models are provided for the coding of RUN.")

Notably, this does not mean that there is a scenario where a context isn't assigned to a run value. In the event that an EOB block indication is sent for the level value, no run value exists. This is because all remaining values in the block are already signaled as zero. Petitioner attempts to twist this logic into disclosing the Challenged Claims. There is no conditional assignment of contexts to the run values in VCEG-L13. Either they are present – and assigned a context – or they aren't present – and not assigned a context. Ex-2001 at ¶ 131.

Even if Petitioner were correct in its interpretation of the claims and function of VCEG-L13, it still fails to render obvious the Challenged Claims. Specifically, the Petitioner's argument regarding the end of block symbol does not relate to “a plurality of number pairs.” This is because the end of block symbol is a single number, not a pair. There is no run value associated with an end of block symbol. Under VCEG-L13, any time that a number pair exists, both values are assigned a context based on block type. This is because, so long as the level value is not equal to zero, there will be a run value associated with it. Together, these values form the number pair. This is also the condition that VCEG-L13 requires to assign contexts to the run value in the first place. There is no situation that VCEG-L13 discloses where the second “second number of a number pair” is assigned a context based on “the first number of the number pair.” Either the number pair exists and both values are assigned a context, or the number pair does not exist.

IV. CONCLUSION

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

Dated: January 21, 2025

Respectfully submitted,

/s/Scott W. Hejny

Scott W. Hejny

Reg. No. 45,882

shejny@mckoolsmith.com

Eric S. Hansen

Reg. No. 63095

ehansen@mckoolsmith.com

McKool Smith, P.C.

300 Crescent Court, Suite 1500

Dallas, Texas 75201

Telephone: (214) 978-4000

Facsimile: (214) 978-4044

*Counsel for Patent Owner, Nokia
Technologies Oy*

CERTIFICATION OF WORD COUNT

The undersigned hereby certifies that the portions of the above-captioned **PATENT OWNER'S PRELIMINARY RESPONSE** specified in 37 C.F.R. §42.24 have 12,878 words, in compliance with the 14,000 word limit set forth in 37 C.F.R. §42.24(b)(1). This word count was prepared using Microsoft Word 2010.

Dated: January 21, 2025

Respectfully submitted,

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

CERTIFICATION OF SERVICE

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

Jessica Kaiser
kaiser-ptab@perkinscoie.com

Robin Brewer
Brewer-ptab@perkinscoie.com

Atanas Baitchev
Baitchev-ptab@perkinscoie.com

Wei Yuan
wyuan@perkinscoie.com

Jingyuan Huang
Huang-ptab@perkinscoie.com

Jon R. Carter
carter-ptab@perkinscoie.com

Amazon-NokiaIPR@perkinscoie.com

Dated: January 21, 2025

Respectfully submitted,

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