IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent of:	Kemal Ugur et al.	
U.S. Patent No.:	11,805,267	Attorney Docket No.: 54587-0013IP1
Issue Date:	October 31, 2023	
Appl. Serial No.:	17/328,750	
Filing Date:	May 24, 2021	
Title:	MOTION PREDICTION IN VIDEO CODING	

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

PETITION FOR *INTER PARTES* REVIEW OF UNITED STATES PATENT NO. 11,805,267 PURSUANT TO 35 U.S.C. §§ 311–319, 37 C.F.R. § 42

TABLE OF CONTENTS

EXHIBITS

ASUS Exhibit No.	Description
ASUS-1001	U.S. Patent No. 11,805,267 to Ugur et al. ("the '267 patent")
ASUS-1002	Prosecution File History for the '267 patent
ASUS-1003	Declaration of Joseph Havlicek
ASUS-1004	RESERVED
ASUS-1005	U.S. Patent Application Publication No. 2011/0007799 ("Karczewicz-I")
ASUS-1006	U.S. Patent Application Publication No. 2009/0257499 ("Karczewicz-II")
ASUS-1007	Prosecution History for U.S. Patent No. 9,432,693
ASUS-1008	U.S. Patent No. 9,344,744 ("Kirchhoffer")
ASUS-1009	Srinivasan, An Overview of VC-1
ASUS-1010	U.S. Patent Application Publication No. 2003/0112864 ("Karczewicz-864")
ASUS-1011	Wiegand, Overview of the H.264/AVC Video Coding Standard
ASUS-1012	Richardson, The H.264 Advanced Video Compression Standard
ASUS-1013	U.S. Patent Application Publication No. 2008/0198935 ("Srinivasan-935")
ASUS-1014	H.264 Advanced Video Coding for Generic Audiovisual Services (March 2009)

ASUS-1015	RESERVED
ASUS-1016	U.S. Patent No. 8,594,188 ("Demos")
ASUS-1017 – ASUS-1023	RESERVED
ASUS-1024	Deposition Transcript of Dr. Iain Richardson
ASUS-1025 – ASUS-1029	RESERVED
ASUS-1030	Deposition Transcript of Immanuel Freedman
ASUS-1031	Patent Trial and Appeal Board (PTAB) Boardside Chat: Interim Processes Relating to institution in AIA Proceedings (Apr. 17, 2025)
ASUS-1032	Coke Morgan Stewart, Interim Processes for PTAB Workload Management (Mar. 26, 2025)
ASUS-1033	Institution Decision IPR2024-00626, Paper 13, September 23, 2024
ASUS-1034	Institution Decision IPR2024-00627, Paper 16, September 25, 2024

LISTING OF CLAIMS

Claim 1	
[1a]	A method for encoding a block of pixels, the method comprising:
[1b]	determining, for a current block, a first reference block based on a first motion vector and a second reference block based on a second motion vector, wherein the pixels of the current block, the first reference block, and the second reference block have values with a first precision;
[1c]	using said first reference block to obtain a first prediction, said first prediction having a second precision, which is higher than said first precision;
[1d]	using said second reference block to obtain a second prediction, said second prediction having the second precision;
[1e]	obtaining a combined prediction based at least partly upon said first prediction and said second prediction;
[1f]	decreasing a precision of said combined prediction by shifting bits of the combined prediction to the right; and
[1g]	encoding residual data in a bitstream, wherein the residual data is determined based upon a difference between the combined prediction and the block of pixels.
Claim 2	
2	The method according to claim 1, wherein in an instance in which said first motion vector points to a subpixel, said first prediction is obtained by interpolation using pixel values of said first reference block.
Claim 3	
3	The method according to claim 2, wherein said first prediction is obtained by interpolation using values of said first reference block by: right shifting a sum of a P-tap filter using values of said first reference block.

Claim 4	
4	The method according to claim 2, wherein in an instance in which said second motion vector points to an integer sample, said second prediction is obtained by shifting values of said second reference block to the left.
Claim 5	
5	The method according to claim 1, wherein said decreasing said precision of said combined prediction by shifting bits of the combined prediction to the right, further comprises: inserting a rounding offset to the combined prediction before said decreasing.
Claim 6	
6	The method according to claim 1, wherein the first precision indicates a number of bits needed to represent the values of the pixels, and the second precision indicates the number of bits needed to represent values of said first prediction and values of said second prediction.
Claim 7	
[7a]	An apparatus for encoding a block of pixels, the apparatus comprising: at least one processor and at least one memory including computer program code, the at least one memory and computer program code configured to, with the at least one processor, cause the apparatus to:
[7b]	determine, for a current block, a first reference block based on a first motion vector and a second reference block based on a second motion vector, wherein the pixels of the current block, the first reference block, and the second reference block have values with a first precision;
[7c]	use said first reference block to obtain a first prediction, said first prediction having a second precision, which is higher than said first precision;
[7d]	use said second reference block to obtain a second prediction, said second prediction having the second precision;
[7e]	obtain a combined prediction based at least partly upon said first prediction and said second prediction;

[7f]	decrease a precision of said combined prediction by shifting bits of the combined prediction to the right; and
[7g]	encode residual data in a bitstream, wherein the residual data is determined based upon a difference between the combined prediction and the block of pixels.
Claim 8	
8	The apparatus according to claim 7, wherein in an instance in which said first motion vector points to a subpixel, said first prediction is obtained by interpolation using pixel values of said first reference block.
Claim 9	
9	The apparatus according to claim 8, wherein said first prediction is obtained by interpolation using values of said first reference block by: right shifting a sum of a P-tap filter using values of said first reference block.
Claim 10	
10	The apparatus according to claim 8, wherein in an instance in which said second motion vector points to an integer sample, said second prediction is obtained by shifting values of said second reference block to the left.
Claim 11	
11	The apparatus according to claim 7, wherein the at least one memory and computer code are configured to cause the apparatus to decrease said precision of said combined prediction by shifting bits of the combined prediction to the right, by: inserting a rounding offset to the combined prediction before said decreasing.
Claim 12	
12	The apparatus according to claim 7, wherein the first precision indicates a number of bits needed to represent the values of the pixels, and the second precision indicates the number of bits needed to represent values of said first prediction and values of said second prediction.

Claim 13	
[13a]	A computer program product for encoding a block of pixels, the computer program product comprising at least one non-transitory computer readable storage medium having computer executable program code portions stored therein, the computer executable program code portions comprising program code instructions configured to:
[13b]	determine, for a current block, a first reference block based on a first motion vector and a second reference block based on a second motion vector, wherein the pixels of the current block, the first reference block, and the second reference block have values with a first precision;
[13c]	use said first reference block to obtain a first prediction, said first prediction having a second precision, which is higher than said first precision;
[13d]	use said second reference block to obtain a second prediction, said second prediction having the second precision;
[13e]	obtain a combined prediction based at least partly upon said first prediction and said second prediction;
[13f]	decrease a precision of said combined prediction by shifting bits of the combined prediction to the right; and
[13g]	encode residual data in a bitstream, wherein the residual data is determined based upon a difference between the combined prediction and the block of pixels.
Claim 14	
14	The computer program product according to claim 13, wherein in an instance in which said first motion vector points to a subpixel, said first prediction is obtained by interpolation using pixel values of said first reference block.
Claim 15	

15	The computer program product according to claim 14, wherein said first prediction is obtained by interpolation using values of said first reference block by: right shifting a sum of a P-tap filter using values of said first reference block.
Claim 16	
16	The computer program product according to claim 14, wherein in an instance in which said second motion vector points to an integer sample, said second prediction is obtained by shifting values of said second reference block to the left.
Claim 17	
17	The computer program product according to claim 13, wherein the program code instructions configured to decrease said precision of said combined prediction by shifting bits of the combined prediction to the right, further comprise program code instructions configured to: insert a rounding offset to the combined prediction before said decreasing.
Claim 18	
18	The computer program product according to claim 13, wherein the first precision indicates a number of bits needed to represent the values of the pixels, and the second precision indicates the number of bits needed to represent values of said first prediction and values of said second prediction.
Claim 19	
[19a]	A method for decoding a block of pixels, the method comprising:
[19b]	determining, for a current block, a first reference block based on a first motion vector and a second reference block based on a second motion vector, wherein the pixels of the current block, the first reference block, and the second reference block have values with a first precision;
[19c]	using said first reference block to obtain a first prediction, said first prediction having a second precision, which is higher than said first precision;

[19d]	using said second reference block to obtain a second prediction, said second prediction having the second precision;
[19e]	obtaining a combined prediction based at least partly upon said first prediction and said second prediction;
[19f]	decreasing a precision of said combined prediction by shifting bits of the combined prediction to the right; and
[19g]	reconstructing the block of pixels based on the combined prediction.
Claim 20	
20	The method according to claim 19, wherein in an instance in which said first motion vector points to a subpixel, said first prediction is obtained by interpolation using pixel values of said first reference block.
Claim 21	
21	The method according to claim 20, wherein said first prediction is obtained by interpolation using values of said first reference block by: right shifting a sum of a P-tap filter using values of said first reference block.
Claim 22	
22	The method according to claim 20, wherein in an instance in which said second motion vector points to an integer sample, said second prediction is obtained by shifting values of said second reference block to the left.
Claim 23	
23	The method according to claim 19, wherein said decreasing said precision of said combined prediction by shifting bits of the combined prediction to the right, further comprises: inserting a rounding offset to the combined prediction before said decreasing.
Claim 24	

24	The method according to claim 19, wherein the first precision indicates a number of bits needed to represent the values of the pixels, and the second precision indicates the number of bits needed to represent values of said first prediction and values of said second prediction.
Claim 25	
[25a]	An apparatus for decoding a block of pixels, the apparatus comprising: at least one processor and at least one memory including computer program code, the at least one memory and computer program code configured to, with the at least one processor, cause the apparatus to:
[25b]	determine, for a current block, a first reference block based on a first motion vector and a second reference block based on a second motion vector, wherein the pixels of the current block, the first reference block, and the second reference block have values with a first precision;
[25c]	use said first reference block to obtain a first prediction, said first prediction having a second precision, which is higher than said first precision;
[25d]	use said second reference block to obtain a second prediction, said second prediction having the second precision;
[25e]	obtain a combined prediction based at least partly upon said first prediction and said second prediction;
[25f]	decrease a precision of said combined prediction by shifting bits of the combined prediction to the right; and
[25g]	reconstruct the block of pixels based on the combined prediction.
Claim 26	
26	The apparatus according to claim 25, wherein in an instance in which said first motion vector points to a subpixel, said first prediction is obtained by interpolation using pixel values of said first reference block.
Claim 27	
27	The apparatus according to claim 26, wherein said first prediction is obtained by interpolation using values of said first reference block by:

	right shifting a sum of a P-tap filter using values of said first reference block.
Claim 28	
28	The apparatus according to claim 26, wherein in an instance in which said second motion vector points to an integer sample, said second prediction is obtained by shifting values of said second reference block to the left.
Claim 29	
29	The apparatus according to claim 25, wherein the at least one memory and computer code are configured to cause the apparatus to decrease said precision of said combined prediction by shifting bits of the combined prediction to the right, by: inserting a rounding offset to the combined prediction before said decreasing.
Claim 30	
30	The apparatus according to claim 25, wherein the first precision indicates a number of bits needed to represent the values of the pixels, and the second precision indicates the number of bits needed to represent values of said first prediction and values of said second prediction.
Claim 31	
[31a]	A computer program product for decoding a block of pixels, the computer program product comprising at least one non-transitory computer readable storage medium having computer executable program code portions stored therein, the computer executable program code portions comprising program code instructions configured to:
[31b]	determine, for a current block, a first reference block based on a first motion vector and a second reference block based on a second motion vector, wherein the pixels of the current block, the first reference block, and the second reference block have values with a first precision;
[31c]	use said first reference block to obtain a first prediction, said first prediction having a second precision, which is higher than said first

	precision;
[31d]	use said second reference block to obtain a second prediction, said second prediction having the second precision;
[31e]	obtain a combined prediction based at least partly upon said first prediction and said second prediction;
[31f]	decrease a precision of said combined prediction by shifting bits of the combined prediction to the right; and
[31g]	reconstruct the block of pixels based on the combined prediction.
Claim 32	
32	The computer program product according to claim 31, wherein in an instance in which said first motion vector points to a subpixel, said first prediction is obtained by interpolation using pixel values of said first reference block.
Claim 33	
33	The computer program product according to claim 32, wherein said first prediction is obtained by interpolation using values of said first reference block by: right shifting a sum of a P-tap filter using values of said first reference block.
Claim 34	
34	The computer program product according to claim 32, wherein in an instance in which said second motion vector points to an integer sample, said second prediction is obtained by shifting values of said second reference block to the left.
Claim 35	
35	The computer program product according to claim 31, wherein the program code instructions configured to decrease said precision of said combined prediction by shifting bits of the combined prediction to the right, further comprise program code instructions configured to: insert a rounding offset to the combined prediction before said decreasing.

Claim 36	
36	The computer program product according to claim 31, wherein the first precision indicates a number of bits needed to represent the values of the pixels, and the second precision indicates the number of bits needed to represent values of said first prediction and values of said second prediction.

ASUSTeK Computer Inc. ("Petitioner" or "ASUS") petitions for *Inter Partes* Review ("IPR") of claims 1-36 ("the Challenged Claims") of U.S. Patent No. 11,805,267 ("the '267 patent"). Compelling evidence presented in this Petition demonstrates at least a reasonable likelihood that ASUS will prevail with respect to at least one of the Challenged Claims.

I. REQUIREMENTS FOR IPR UNDER 37 C.F.R. § 42.104

A. Grounds for Standing Under 37 C.F.R. § 42.104(a)

ASUS certifies that the '267 patent is available for IPR. This Petition is being filed within one year of service of a complaint against ASUS in *Nokia Technologies Oy* v. *ASUSTeK Computer Inc. et al.*, 2-25-cv-03053 (CDCA), filed on April 7, 2025. ASUS is not barred or estopped from requesting review of the Challenged Claims on the below-identified grounds.

B. Challenge Under 37 C.F.R. § 42.104(b) and Relief Requested

ASUS requests an IPR of the Challenged Claims on the ground indicated below.[1] Ground 1 is supported and corroborated by evidence cited throughout this

[1] The Board previously instituted IPR on the same Challenged Claims of the '267 patent in IPR2024-00626 and IPR2024-00627 based on the same prior art and obviousness contentions as those presented in Ground 1 of this Petition. ASUS-1052-1053.

Petition, including by the expert declaration of Dr. Joseph Havlicek.[2] ASUS-1003.

Ground	Claims	35 U.S.C. § 103
1	1-36	Karczewicz-I in view of Karczewicz-II

The '267 patent's earliest possible priority date is January 7, 2011 ("Critical Date"). Each of the references in Ground 1 pre-date the Critical Date.

Reference	Filed	Published	Pre-AIA Prior Art Basis
Karczewicz-I	7/9/2009	1/13/2011	§102(e)

[2] Dr. Havlicek's declaration incorporates as Appendices B and C the declarations of Dr. Immanuel Freedman. *See* ASUS-1003, ¶24, Appx-B, Appx-C. Dr. Havlicek carefully reviewed Dr. Freedman's declarations and explains that he has adopted Dr. Freedman's analysis and opinions as his own. *Id.* Rather than repeat Dr. Freedman's earlier declaration testimony, Dr. Havlicek focuses his testimony by referring to Dr. Freedman's declarations in addressing the background of the '267 patent's technology and the obviousness of the Challenged Claims in Ground 1. *Id.* Citations with paragraph numbers labeled Appx-B refer to paragraphs in Dr. Freedman's declaration submitted in Appendix B and citations with paragraph numbers labeled Appx-C refer to paragraphs in Dr. Freedman's declaration submitted in Appendix C.

Reference	Filed	Published	Pre-AIA Prior Art Basis
Karczewicz-II	4/8/2009	10/15/2009	§102(a), (b), (e)

II. THE '267 PATENT

A. Brief Description

The '267 patent is directed to "utilizing motion prediction in video coding." ASUS-1001, Abstract. The patent discusses an "existing" process for generating a bi-directional prediction for a current block. *See, e.g.*, ASUS-1001, 13:43-55, Fig. 10.

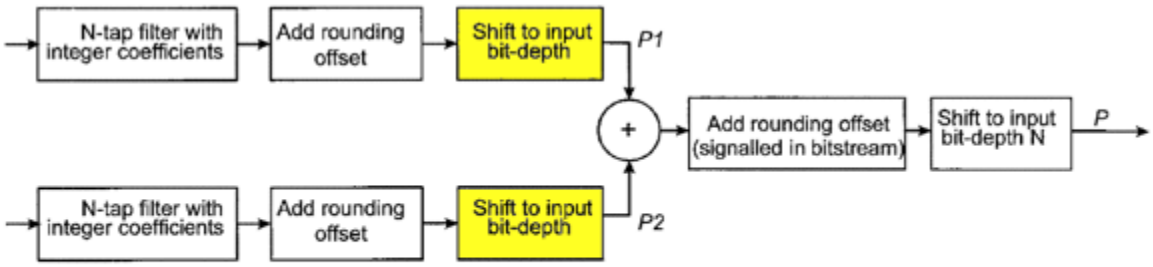

Fig. 10

ASUS-1001, FIG. 10

The '267 patent does not purport to invent this conventional process, admitting that the "Background" art includes video coding standards, e.g., H.264 (ASUS-1001, 1:34-46), and "bi-directional prediction" (*id.*, 3:49-55), with reference blocks determined by motion vectors (*id.*, 2:20-34, 3:12-18), predictions determined based on reference blocks (*id.*, 1:34-46), bi-directional prediction obtained by combining two predictions (*id.*, 3:49-55, 3:66-4:20), and residual data

calculated as a difference between the prediction and current block (*id*., 1:52-59, 3:25-30, 2:1-12).

The patent further admits that the prior art included various techniques for addressing rounding errors when averaging two blocks. ASUS-1001, 3:66-4:20. According to the '267 patent, "these methods increase somewhat the complexity" of encoders. ASUS-1001, 4:21-25.

The patent purports to "reduc[e] the effect of rounding errors in bi-directional… prediction" by maintaining prediction signals "in a higher precision during the prediction calculation" and rounding the prediction down to a lower precision "after the two or more prediction signals have been combined with each other." ASUS-1001, 4:29-43. In particular, it describes pixels having a first precision (e.g., N) and predictions having a second precision (e.g., M) that is higher than the first. ASUS-1001, 12:41-13:55, Fig. 11.

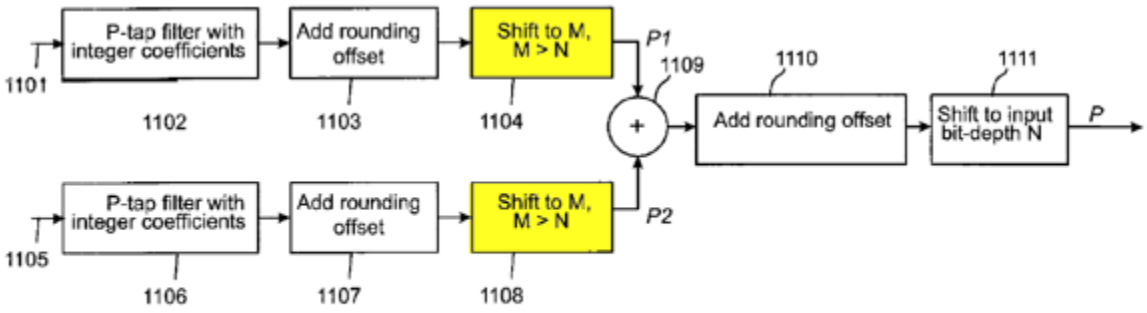

Fig. 11

ASUS-1001, FIG. 11

The patent only purports to change the number of right-shifted bits for

predictions before they are combined. However, this approach was already known and used in video coding. ASUS-1003, Appx-B, §§I.A-C. For example, the Karczewicz references taught pixel interpolation with the same concept—using higher precision for intermediate calculations—for the same calculations, years before the '267 patent.

B. Summary of the Prosecution History

During prosecution of a parent application, the Examiner issued §103 rejections based on U.S. 2013/0142262 ("Ye"). ASUS-1003, Appx-B, ¶¶26-27. The Examiner explained that Ye's weighted-prediction teachings "increase[d] the precision of $P_0(x,y)$" by multiplying it by a weight ("$w \cdot P_0(x,y)$"), which suggests a higher second precision. ASUS-1007, 249-251. However, the Examiner acknowledged that Ye did not explicitly state "the number of bits needed to represent values of said first prediction and values of said second prediction," which is "higher than said first precision[.]" ASUS-1007, 252.

In response, the Applicant did not dispute that the weighted prediction (e.g., "$w \cdot P_0(x,y)$") teaches the claimed first or second prediction, and instead argued that Ye lacks teachings about increased precision. ASUS-1007, 280-282; ASUS-1003, Appx-B, ¶¶28-30.

C. Level of Ordinary Skill in the Art

The field of the patent is video encoding/decoding. A POSITA at the time of

the alleged invention of the '267 patent would have had a (1) Bachelor's degree in electrical engineering, computer engineering, computer science, or a comparable field of study such as physics, and (2) approximately two to three years of practical experience with video encoding/decoding. ASUS-1003, ¶29, Appx-B, ¶¶33-36. Additional experience can substitute for the level of education, and vice-versa. *Id.*

D. Claim Construction

For purposes of this proceeding only, ASUS submits constructions for the following terms. All remaining terms should be given their plain meaning.

1. "precision"

A POSITA would have understood "precision" is satisfied by, but not necessarily limited to, "a number of bits needed to represent possible values." ASUS-1003, ¶33, Appx-B, ¶¶39-42. Dependent claims 6, 12, and 18 specify that "precision indicates a number of bits needed to represent the values of the pixels" and predictions. Likewise, the specification uses "precision" to refer to a number of bits representing possible values for pixels/predictions. ASUS-1001, 12:41-13:55, 14:4-10. During prosecution, the Applicant stated, "precision indicates the number of bits needed to represent values of said pixels" and "prediction[s.]" ASUS-1007, 280.

III. THE CHALLENGED CLAIMS ARE UNPATENTABLE

A. Ground 1: Obviousness based on Karczewicz-I in view of Karczewicz-II

1. Karczewicz-I (ASUS-1005)

Karczewicz-I is U.S. Patent Application Publication 2011/0007799, filed July 9, 2009; it is prior art under §102(e). ASUS-1003, Appx-B, ¶128.

Karczewicz-I teaches block-based techniques for motion prediction, including bi-directional predictions calculated by averaging predictions based on two reference blocks, which may be calculated using sub-pixel interpolation. *E.g.*, ASUS-1005, ¶35, ¶¶41-44, ¶¶55-60; ASUS-1003, Appx-B, ¶129.

Karczewicz-I is analogous art in the same field as the ’267 patent because it is directed to video encoding/decoding, motion prediction, and H.264. *E.g.*, ASUS-1005, ¶89, ¶¶35-37; ASUS-1003, Appx-B, ¶130.

2. Karczewicz-II (ASUS-1006)

Karczewicz-II is U.S. Patent Application Publication 2009/0257499, published October 2009; it is prior art under §§102(a), (b) and (e). ASUS-1003, Appx-B, ¶132.

Karczewicz-II reduced rounding inaccuracies with interpolated pixels by maintaining higher precision for intermediate values while delaying rounding until later in the process. *E.g.*, ASUS-1006, ¶10, ¶39, ¶53, ¶59, ¶¶93-106; ASUS-1003, Appx-B, ¶134. For example, Karczewicz-II “generates half-pixel values… stores

the half-pixel values as ***non-rounded*** versions" and combines them "based on the ***non-rounded versions*** of the half-pixel values and the integer pixel values." ASUS-1006, ¶17, ¶¶96-108; ASUS-1003, Appx-B, ¶¶133-134.

These are the same calculations as embodiments of the '267 patent, which likewise perform two half-pixel interpolations (one for each reference block) and then average the higher-precision half-pixel predictions together. ASUS-1001, 12:41-13:55.

Karczewicz-II is analogous art in the same field as the '267 patent because it is directed to video encoding/decoding and H.264. *E.g.*, ASUS-1006, ¶2, ¶¶46-47; ASUS-1003, Appx-B, ¶135.

3. Motivation to Combine

Karczewicz-I and Karczewicz-II are Qualcomm patent applications by the same inventors. Both apply their teachings to similar video-coding architectures. ASUS-1005, ¶2, ¶¶29-50, Fig. 1; ASUS-1006, ¶2, ¶¶40-53, Fig. 1. Both are directed to block-based H.264 motion prediction. ASUS-1005, ¶35, ¶44, ¶¶55-60; ASUS-1006, ¶8, ¶35-36, ¶46, ¶54. Both teach a video encoder that performs motion estimation and compensation for inter-predictive coding:

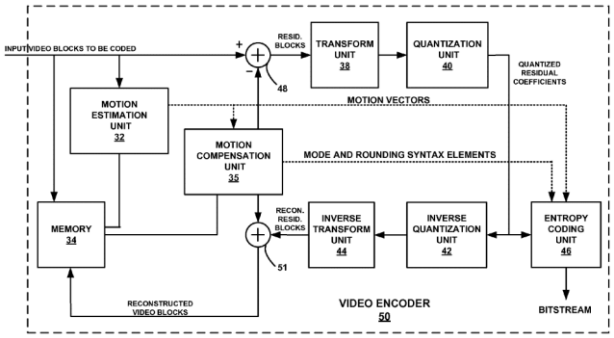

FIG. 2

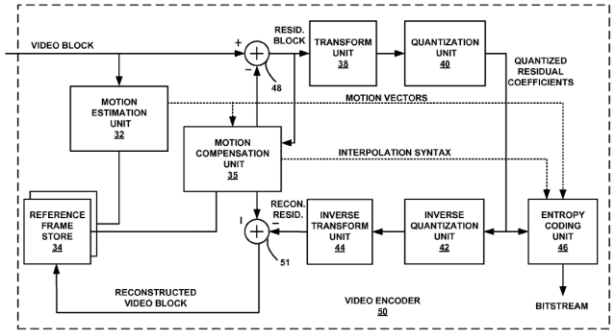

FIG. 2

ASUS-1005, FIG. 2, ¶53; ASUS-1006, FIG. 2, ¶56

ASUS-1003, Appx-B, ¶¶136-137.

Both also teach a video decoder that performs motion compensation for block-based decoding:

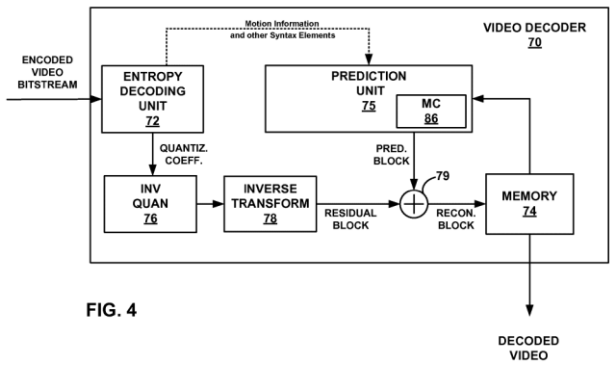

FIG. 4

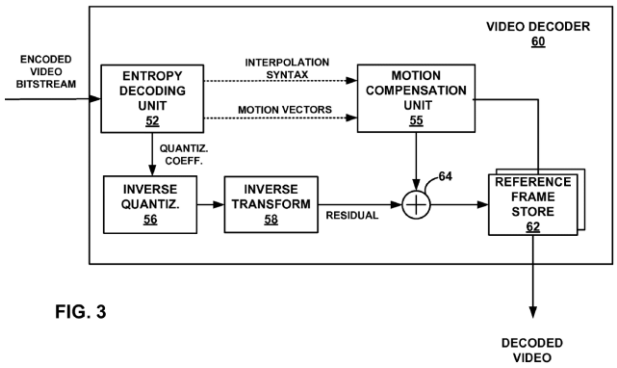

FIG. 3

ASUS-1005, FIG. 4, ¶89; ASUS-1006, FIG. 3, ¶63

ASUS-1003, Appx-C, ¶¶136-137.

These similarities would have motivated a POSITA to implement teachings from Karczewicz-I and Karczewicz-II using their common architecture, combining prior art elements according to known methods. *KSR Int'l Co. v. Teleflex Inc.*, 550 U.S. 398, 401 (2007) ("[I]f… a person of ordinary skill in the art would recognize that it would improve similar devices in the same way…"). Karczewicz-I and

Karczewicz-II teach the same calculation of averaging interpolated pixel values. ASUS-1003, ¶35. Karczewicz-I teaches bi-prediction techniques for H.264 with two motion vectors pointing to two blocks of pixels or integer values that are averaged together. ASUS-1005, ¶53, ¶35, ¶44, ¶60; ASUS-1024, 81:3-12, 101:6-102:7 (confirming that Karczewicz-I describes "bi-prediction" that involves "averaging of two predictions," where the predictions include integer or interpolated (fractional) pixel positions). Beyond integer pixels, Karczewicz-II optimizes this calculation by preserving higher-precision intermediate values. ASUS-1003, ¶35. Karczewicz-II teaches that H.264 motion vectors can also point to fractional sub-pixels (ASUS-1006, ¶¶56-58, ¶¶93-102),[3] and Karczewicz-II teaches improved calculations for averaging two integer or interpolated pixel values, where rounding is delayed until later in the process, thereby maintaining higher precision for intermediate calculations (ASUS-1006, ¶10, ¶¶20-24, ¶39, ¶¶99-106). Therefore, Karczewicz-II provides complementary teachings that improve Karczewicz-I by reducing rounding inaccuracies. *See, e.g.,* ASUS-1006, ¶102, ¶39, ¶53, ¶59; ASUS-1003, Appx-B, ¶¶138-140. The combination simply applies the exact optimization of Karczewicz-II to modify the corresponding

[3] This was known in the art, as admitted in the '267 patent's "Background Information" section. ASUS-1003, Appx-B, §I.A-C; ASUS-1001, 2:60-3:11.

calculation in Karczewicz-I. ASUS-1003, ¶35.

Karczewicz-I and Karczewicz-II both encompass motion prediction calculations that average interpolated pixel values. The calculations taught by Karczewicz-II correspond to the calculations used for Karczewicz-I's bi-predicted pixel values when applied to integer and sub-pixel values. ASUS-1006, ¶102, ¶39, ¶53, ¶59; ASUS-1003, Appx-B, ¶¶138-140. A POSITA would have been motivated to use Karczewicz-II's known techniques to improve similar devices/methods, as taught by Karczewicz-I, in the same way to improve prediction accuracy. ASUS-1006, ¶102, ¶10. This simply uses a known technique to improve a similar device/method. ASUS-1003, ¶35. And it would have had predictable results because it applies teachings from Karczewicz-II to corresponding mathematical calculations used for Karczewicz-I, using similar video codec architectures. ASUS-1003, Appx-B, ¶141.

A POSITA would have recognized the applicability of Karczewicz-II to the corresponding calculations in Karczewicz-I, which was a simple matter given the level of ordinary skill. The references share the same architecture, and the combination does not change the architecture; it simply uses more bits for intermediate calculations, as Karczewicz-II teaches. ASUS-1003, ¶35. Notably, a POSITA would have understood fundamental computer logic concepts (e.g., binary arithmetic, bit shifting, precision control, rounding and

offsetting, and error analysis) and mathematical calculations for known motion estimation and compensation techniques (e.g., bi-directional prediction, determination and use of motion vectors, interpolation for fractional pixels) because they are integral to working with video codecs. ASUS-1003, Appx-B, ¶141.

Pixel Interpolation. H.264 includes interpolated fractional pixel positions. Karczewicz-II explains that H.264 calculates predictions for half-pixel positions using 6-tap filters that interpolate the sub-pixel based on nearby pixels in the same row (e.g., half-pixel "b") or column (e.g., half-pixel "h"). ASUS-1006, ¶74, ¶¶93-94, Fig. 4A-4B.

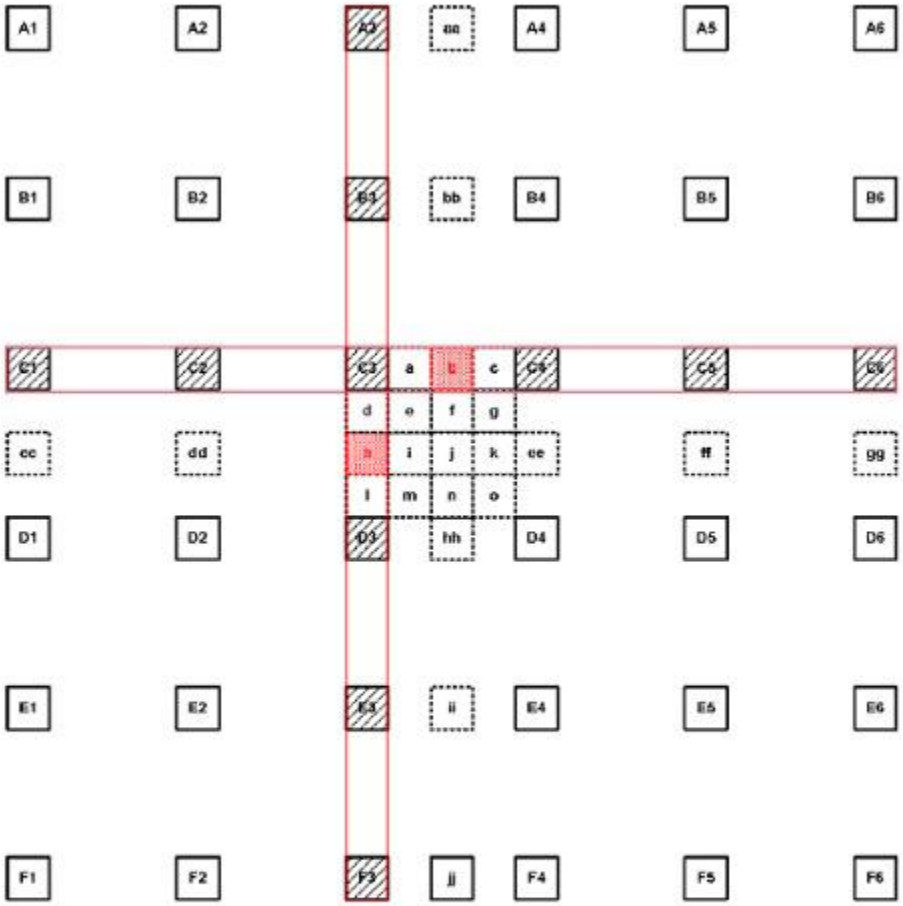

FIG. 4B

ASUS-1006, Fig. 4A

ASUS-1003, Appx-B, ¶144.

This process involved two steps. First, the filter multiplied the six pixels in the row (or column) by filter values and added the products together to produce a non-rounded prediction (e.g., b1):

$$b1=C1-5*C2+20*C3+20*C4-5*C5+C6$$

Second, the result was rounded, e.g., right-shifted (">>") producing rounded prediction b:

$$b=\max(0, \min(255, b1+16)>>5))$$

ASUS-1006, ¶¶93-94, Fig. 4B; ASUS-1003, Appx-B, ¶145.

For half-pixel positions in the center of four integer pixels (e.g., j), interpolation was applied in two rounds: first to interpolate a middle row of half-pixels, second to interpolate that middle row into the center-pixel, applying the same 6-tap interpolation filter and two-step process described above to the interpolated middle row. ASUS-1006, ¶95, Fig. 4C; ASUS-1003, Appx-B, ¶146.

For quarter-pixel predictions, Karczewicz-II averaged the two nearest integer or half-pixel predictions as explained above. ASUS-1006, ¶¶96-97. Karczewicz-II teaches an improvement by "keep[ing] the highest possible precision through the intermediate steps" and avoiding "any shifting, rounding and clipping operations… until the very last step of the interpolation process." ASUS-1006, ¶99, ¶102, ¶10, ¶39, ¶53, ¶59; ASUS-1003, Appx-B, ¶147.

Calculation Scenarios. Karczewicz-I teaches weighted bi-directional prediction that averages two predictions based on two reference blocks (e.g., pred0(i,j), pred1(i,j)) with an offset (e.g., +1), including a default mode (ASUS-1005, ¶55) with equal weights that performs a simple average. ASUS-1005, ¶60.

$$pred(i,j)=(pred0(i,j)+pred1(i,j)+1)>>1$$

ASUS-1005, ¶¶58-60. Because H.264 allows motion vectors to point to integer pixels or fractional pixels/subpixels, the predictions pred0(i,j) and pred1(i,j) encompass scenarios that include integer pixel prediction, a half-pixel prediction, or a center-pixel prediction, and Karczewicz-I calculates an average of these pixel values. ASUS-1005, ¶41; ASUS-1006, ¶¶93-102, ¶¶56-58; ASUS-1003, Appx, B, §§I.A-C.

Consistent with H.264, Karczewicz-I teaches that the inter-predictive coding process includes interpolation, providing express teaching, suggestion, and motivation ("TSM") to combine with known interpolation teachings. ASUS-1005, ¶41. Karczewicz-II teaches optimizations for interpolating and averaging integer, half-, and center-pixels, which a POSITA would have been motivated to apply to at least three scenarios for Karczewicz-I's teachings that follow the exact optimization scenarios taught by Karczewicz-II for averaging integer, half, and center pixel values. ASUS-1003, Appx-B, ¶150. The modifications for each scenario implement Karczewicz-II's optimization of preserving higher-precision intermediate values. ASUS-1003, ¶36.

Scenario 1 (with the first motion vector pointing to a half-pixel position and the second motion vector pointing to an integer pixel position). Beyond integer pixels, Karczewicz-II explains how, for H.264, motion vectors can also point to half-pixel positions (ASUS-1006, ¶¶93-102, ¶¶56-58), which therefore teaches or

suggests the default weighted prediction calculated as an average of a half-pixel and integer pixel. *See* ASUS-1005, ¶60, ¶55; ASUS-1003, Appx-B, ¶151.

Karczewicz-II teaches an improved calculation for averaging a half-pixel value with an integer pixel value. ASUS-1006, ¶96, Fig. 4D. Karczewicz-II first explains the conventional calculation for averaging two numbers. ASUS-1006, Fig. 4D, Table 1.

TABLE 1

a = (C3 + b + 1) >> 1
c = (C4 + b + 1) >> 1
d = (C3 + h + 1) >> 1
l = (D3 + h + 1) >> 1
f = (j + b + 1) >> 1
i = (j + h + 1) >> 1
k = (j + ee + 1) >> 1
n = (j + hh + 1) >> 1

ASUS-1006, Table 1

ASUS-1003, Appx-B, ¶152.

Karczewicz-II improves this conventional approach by replacing the equations in Table 1 with those in Table 3, where the pixel values are combined at a higher precision. ASUS-1006, 99. The integer pixel value (e.g., C3) is multiplied by 32 (left-shifted 5 bits), taking its precision from 8 to 13 bits. ASUS-1006, Table 5. Instead of using a rounded 8-bit half-pixel (e.g., b), Karczewicz-II delays the rounding step and instead uses a *non-rounded* half-pixel prediction (e.g., b1) that is

15 bits. *Id.* These pixel values are combined, along with a rounding offset of 32, before rounding to reduce the precision at the end, e.g., shifting 6 bits to the right (“>>6”). ASUS-1006, Table 3.

TABLE 3

a = (C3 << 5 + b1 + 32) >> 6
c = (C4 << 5 + b1 + 32) >> 6
d = (C3 << 5 + h1 + 32) >> 6
l = (D3 << 5 + h1 + 32) >> 6
f = (j1 >> 5 + b1 + 32) >> 6
i = (j1 >> 5 + h1 + 32) >> 6
k = (j1 >> 5 + ee1 + 32) >> 6
n = (j1 >> 5 + hh1 + 32) >> 6

ASUS-1006, Table 3

ASUS-1003, Appx-B, ¶153.

The operations for this improved approach are shown in Table 5 of Karczewicz-II. ASUS-1006, ¶103, Table 5.

TABLE 5

positions {a, c, d, l} of FIGS. 4A-4D

Operation	Comment	Min value	Max value	Register size
r1 = x	r1 is integer pixel x	0	255	8u
r1 = r1 << 5	r1 is 32 * x	0	8160	13u
r2 = y0	r2 is y0 (y0 is a one-dimensional (1-D) half-pixel such as b1, h1, ee1 and hh1 before shifting down)	−2550	10710	15s
r1 = r1 + r2	r1 is 32 * x + y0	−2550	18870	16s
r1 = r1 + 32	r1 is 32 * x + y0 + 32	−2518	18902	16s
r1 = r1 >> 6	r1 is (32 * x + y0 + 32) >> 6	−39	295	11s
r1 = max (0, r1)	clip r1 on the low side	0	295	10u
r1 = min (255, r1)	clip r1 on the high side	0	255	8u

ASUS-1006, Table 5

ASUS-1003, Appx-B, ¶154.

Since Karczewicz-II teaches this optimization for averaging an interpolated half-pixel and an integer pixel, a POSITA would have been motivated to apply that improved calculation to Karczewicz-I's default weighted prediction equation, which likewise calculates an average, with the half-pixel prediction (e.g.,

pred0(i,j)) kept at a higher, non-rounded precision and the integer pixel prediction (e.g., pred1(i,j)) left-shifted. *See* ASUS-1005, ¶60, ¶55; ASUS-1006, ¶¶93-94, ¶99, Table 3. As Karczewicz-II teaches, these values are combined with a rounding offset of 32 and then right-shifted 6 bits to reduce the precision. This combination results in the following equation:

$$pred(i,j)=(non\text{-}rounded(pred0(i,j))+pred1(i,j)<<5+32)>>6$$

ASUS-1006, ¶96, ¶99, Tables 1 and 3; ASUS-1005, ¶60; ASUS-1003, Appx-B, ¶¶155-156.

Scenario 2 (with the first motion vector pointing to a center-pixel position and the second motion vector pointing to a half-pixel position). The default weighted prediction is calculated as an average of a center-pixel and a half-pixel. *See* ASUS-1005, ¶60, ¶55; ASUS-1003, Appx-B, ¶157.

Karczewicz-II teaches an improved calculation for averaging a center- and half-pixel. ASUS-1006, Fig. 4D. Karczewicz-II first explains the conventional calculation for averaging two numbers. ASUS-1006, ¶96, Fig. 4D, Table 1.

TABLE 1

a = (C3 + b + 1) >> 1
c = (C4 + b + 1) >> 1
d = (C3 + h + 1) >> 1
l = (D3 + h + 1) >> 1
f = (j + b + 1) >> 1
i = (j + h + 1) >> 1
k = (j + ee + 1) >> 1
n = (j + hh + 1) >> 1

ASUS-1006, Table 1

ASUS-1003, Appx-B, ¶158.

Karczewicz-II improves this conventional approach by replacing the equations in Table 1 with those in Table 3, where the pixel values are combined at a higher precision. ASUS-1006, ¶99. Whereas Table 1 used a fully-rounded center-pixel (e.g., "j"), Table 3 uses a partially-rounded pixel (e.g., "j1>>5") that remains 5 bits longer than the rounded value in Table 1.[4] *Infra* §III.A.4(c). Likewise, Table 3 replaces the rounded half-pixel (e.g., "b") with a non-rounded half-pixel (e.g., "b1") that is 15 bits. These pixel values are combined, along with a rounding offset of 32, before rounding to reduce the precision at the end, e.g., shifting 6 bits to the

[4] Here, the partially-rounded prediction is obtained by shifting the non-rounded prediction j1 to the right by 5 bits. This is fewer bits than the 10 bits shifted for calculating the fully rounded prediction j. ASUS-1006, ¶95.

right (“>>6”). ASUS-1006, Table 3.

TABLE 3

a = (C3 << 5 + b1 + 32) >> 6
c = (C4 << 5 + b1 + 32) >> 6
d = (C3 << 5 + h1 + 32) >> 6
l = (D3 << 5 + h1 + 32) >> 6
f = (j1 >> 5 + b1 + 32) >> 6
i = (j1 >> 5 + h1 + 32) >> 6
k = (j1 >> 5 + ee1 + 32) >> 6
n = (j1 >> 5 + hh1 + 32) >> 6

ASUS-1006, Table 3

ASUS-1003, Appx-B, ¶159.

The operations for this improved approach are shown in Table 8 of Karczewicz-II. ASUS-1006, ¶105, Table 8.

TABLE 8

positions {f, i, k, n} of FIGS. 4A-4D

Operation	Comment	Min value	Max value	Register size
r1 = y0	r1 is y0 (1-D half-pixel such as b1, h1, ee1 and hh1 before shifting down)	−2550	10710	15s
r2 = j1	r2 is j1 (2-D half-pixel j1 before shifting down)	−9914	26232	16s
r2 = r2 >> 1	r2 is j1 >> 1	−4957	13116	15s
r1 = r1 + r2	r1 is y0 + (j1 >> 1)	−7507	23826	16s
r1 = r1 + 32	r1 is y0 + (j1 >> 1) + 32	−7491	23842	16s
r1 = r1 >> 6	r1 is (y0 + (j1 >> 1) + 32) >> 6	−235	745	11s
r1 = max (0, r1)	clip r1 on the low side	0	745	10u
r1 = min (255, r1)	clip r1 on the high side	0	255	8u

ASUS-1006, Table 8

ASUS-1003, Appx-B, ¶160.

Since Karczewicz-II teaches this optimization for averaging interpolated center- and half-pixels, a POSITA would have been motivated to apply that improved calculation to Karczewicz-I's default weighted prediction equation, which likewise calculates an average, with the center-pixel prediction partially rounded and the half-pixel prediction non-rounded. ASUS-1005, ¶60, ¶55; *e.g.*, ASUS-1006, ¶10, ¶20, ¶¶23-24, ¶39, ¶74, ¶93, ¶¶99-103. This combination results in the following equation:

pred(i,j)=(non-rounded(pred0(i,j))>>5+non-rounded(pred1(i,j)) +32)>>6

ASUS-1006, ¶96, ¶99, Tables 1 and 3; ASUS-1005, ¶60; ASUS-1003, Appx-B, ¶¶161-162.

Scenario 3 (with both motion vectors pointing to half-pixel positions). The default weighted prediction is calculated as an average of two half-pixels. ASUS-1005, ¶60, ¶55; ASUS-1003, Appx-B, ¶163.

Karczewicz-II teaches an improved calculation for averaging two half-pixel values. ASUS-1006, Fig. 4D. Karczewicz-II explains the conventional calculation for averaging two numbers. ASUS-1006, ¶97, Fig. 4D, Table 2.

TABLE 2

e = (b + h + 1) >> 1
g = (b + ee + 1) >> 1
m = (h + hh + 1) >> 1
o = (ee + hh + 1) >> 1

ASUS-1006, Table 2

ASUS-1003, Appx-B, ¶164.

Karczewicz-II improves this conventional approach replacing the equations in Table 2 with those in Table 4, where the pixel values are combined at a higher precision. ASUS-1006, ¶101. Instead of using a rounded 8-bit half-pixel (e.g., "b" or "h"), Karczewicz-II delays the rounding step and instead uses *non-rounded* half-pixels (e.g., "b1" and "h1") that are 15 bits each. *Id.* These pixel values are combined, along with a rounding offset of 32, before rounding to reduce the

precision at the end, e.g., shifting 6 bits to the right (“>>6”). ASUS-1006, Table 4.

TABLE 4

e = (b1 + h1 + 32) >> 6
g = (b1 + ee1 + 32) >> 6
m = (h1 + hh1 + 32) >> 6
o = (ee1 + hh1 + 32) >> 6

ASUS-1006, Table 4

ASUS-1003, Appx-B, ¶165.

The operations for this improved approach are shown in Table 6 of Karczewicz-II. ASUS-1006, ¶103, Table 6.

TABLE 6

positions {e, g, m, o} of FIGS. 4A-4D

Operation	Comment	Min value	Max value	Register size
r1 = y0	r1 is y0 (y0 is a 1-D half-pixel such as b1, h1, ee1 and hh1 before shifting down)	−2550	10710	15s
r2 = y1	r2 is y1 (y1 is a 1-D half-pixel such as b1, h1, ee1 and hh1 before shifting down)	−2550	10710	15s
r1 = r1 + r2	r1 is y0 + y1	−5100	21420	16s
r1 = r1 + 32	r1 is y0 + y1 + 32	−5068	21452	16s
r1 = r1 >> 6	r1 is (y0 + y1 + 32) >> 6	−79	335	11s
r1 = max (0, r1)	clip r1 on the low side	0	335	10u
r1 = min (255, r1)	clip r1 on the high side	0	255	8u

ASUS-1006, Table 6

ASUS-1003, Appx-B, ¶166.

Since Karczewicz-II teaches this optimization for averaging two interpolated half-pixels, a POSITA would have been motivated to apply that improved calculation to Karczewicz-I's default weighted prediction equation, which likewise calculates an average, with the half-pixel predictions non-rounded. ASUS-1005, ¶60, ¶55; ASUS-1006, ¶¶97-101. This combination results in the following equation:

$$pred(i,j)=(non\text{-}rounded(pred0(i,j))+non\text{-}rounded(pred1(i,j)) +32)>>6$$

ASUS-1006, ¶97, ¶101, Tables 2 and 4; ASUS-1005, ¶60; ASUS-1003, Appx-B, ¶¶167-168.

Across the three scenarios, Karczewicz-II treats each type of motion vector the same. For example, when a motion vector points to a half-pixel, all three scenarios use a non-rounded half-pixel value. ASUS-1003, ¶36; ASUS-1006, ¶103, ¶105, Tables 5, 6, 8; ASUS-1024, 118:14-121:4 (confirming that Karczewicz-II teaches one method of calculating non-rounded half-pixel values). Therefore, in the combination, only one value needs to be stored for the first motion vector. *Id.*

Since H.264 already included bi-prediction and interpolation—with two motion vectors that could point to integer, half-pixel, or center-pixel positions—the three scenarios of motion vectors pointing to different permutations of integer or sub-pixel locations were already present for H.264. ASUS-1003, ¶37; ASUS-1024,

81:3-12, 101:6-102:7; ASUS-1006, ¶¶93-102; ASUS-1014, 190-93. The combination simply uses more bits when calculating these scenarios that were encountered during pre-existing H.264 encoding/decoding. *Id*. Therefore, even if code branches or logic costs were needed to handle these three scenarios for the combination, they are generic costs that were already needed for H.264 even without the combination. ASUS-1024, 271:7-16, 267:9-16 (admitting that code branches are ubiquitous, stating "I don't think I could write a video codex even a very simple one without using conditional execution."); ASUS-1030, 102:7-103:22 ("it would be convenient and simple to maintain the higher [precision] throughout the sequence of calculations"; the modification of Karczewicz-I involves "a tiny change" and "a few lines of code."). The '267 patent's discussion of code branches as "Background Information" proves that separate code branches were known and used in the prior art, despite their alleged inefficiency or cost. ASUS-1001, 4:14-43.

Further, Karczewicz-II already provides motivation to use its optimizations for averaging interpolated pixel values, applied to three scenarios, and Karczewicz-II provides motivation to use its teachings despite Karczewicz-II's calculations being carried out millions of times per second and despite the encoder's need to repeatedly test potential predictions, which is present regardless of whether higher-precision intermediate values are used and are not caused by the modifications of

Karczewicz-I. ASUS-1024, 145:21-146:18. A POSITA would have recognized that there were ways of minimizing the complexity and limiting the number of searches to a small number. ASUS-1024, 159:5-160:11. For video codecs, performance is often improved at the expense of increased computational complexity. ASUS-1024, 271:17-272:8. Therefore, even if there were a tradeoff, it would not obviate the motivation to combine. *See, e.g., Raytheon Co. v. Sony Corp*., 727 F. App'x 662, 667 (Fed. Cir. 2018) (combination rendering the CCD sensor design less efficient than the original does not obviate the motivation to combine); *Medichem, S.A. v. Rolabo, S.L.*, 437 F.3d 1157, 1165 (Fed. Cir. 2006) ("[A] given course of action often has simultaneous advantages and disadvantages, and this does not necessarily obviate motivation to combine.") (citation omitted); ASUS-1003, ¶38.

Express TSM in the Art. It was well-known to use higher-precision intermediate values for video-coding computations; the prior art provides express TSM to apply such teachings from Karczewicz-II to related techniques in Karczewicz-I to achieve a higher accuracy. ASUS-1016, Abstract; ASUS-1003, Appx-B, ¶169.

Preserving higher-precision intermediate values was known to benefit motion compensation. ASUS-1008, 7:4-19 ("a prediction and a reconstruction for a block of a picture to be predicted… *in a higher precision*."). Specifically, Kirchhoffer touts the value of higher quality results, even though those results are

accompanied by at least some increase in computational complexity. ASUS-1008, 16:22–17:36. This approach was known to result in smaller residual information. *Id*.; ASUS-1003, Appx-B, ¶170. It was also known that preserving higher-precision intermediate values improved the accuracy of interpolation. ASUS-1009, 5; ASUS-1010, ¶¶61-62; ASUS-1003, Appx-B, ¶171.

Compatible Teachings. The combination would not have changed the principle of operation of either reference, but merely includes the use Karczewicz-II's known techniques to improve Karczewicz-I's similar devices or methods in the same way. ASUS-1005, Figs. 1-2, ¶¶29-50, ¶53; ASUS-1006, Figs. 1-2, ¶¶40-53, ¶56. Given the similarities between Karczewicz-I and Karczewicz-II, a POSITA would have understood that Karczewicz-II's techniques are readily applicable to Karczewicz-I. ASUS-1003, Appx-B, ¶172.

Moreover, the combination merely changes when rounding occurs for calculations that already included rounding. ASUS-1005, ¶60, ¶55; ASUS-1006, ¶¶96-106, Tables 1-8. The affected calculations already involve rounding. *See id*. The calculations themselves include basic mathematical and logical operations, such as binary arithmetic, addition, rounding, and bit shifting. ASUS-1003, ¶39. This minor implementation detail would not have changed the principle of operation of Karczewicz-I or Karczewicz-II. ASUS-1003, Appx-B, ¶173.

Reasonable expectation of success. A POSITA would have had a reasonable

expectation of success because the combination applies the math behind Karczewicz-II's improved calculations to Karczewicz-I's scenarios without further modification. ASUS-1003, Appx-B, ¶174.

Furthermore, a POSITA would have been more than capable of applying Karczewicz-II's teachings because Karczewicz-II's calculations involve basic mathematic and logical operations (e.g., addition and bit-shifting) that were well known or taught in high-school or undergraduate-level courses, and basic video codec operations that were a core part of industry work in video codecs, as discussed above. ASUS-1003, Appx-B, §I.D, ¶175.

4. Claim Element Analysis

(a) Independent Claims 1, 7, and 13

[1a]. A method for encoding a block of pixels, the method comprising:

The Ground 1 combination teaches the preamble. ASUS-1003, Appx-B, ¶¶176-182. Karczewicz-I "relates to… video encoding techniques that use bi-directional prediction" including for encoders. ASUS-1005, ¶2, ¶¶8-9, ¶22, ¶¶29-30, ¶32, Fig. 1. The encoder "perform[s]… inter-coding of blocks within video frames" that includes motion estimation and compensation operations. ASUS-1005, ¶¶51-76, Fig. 2; ASUS-1003, Appx-B, ¶177. Karczewicz-I encodes "blocks of pixel data[.]" *See, e.g.*, ASUS 1005, ¶¶39-40; ASUS-1003, Appx-B, ¶178.

Karczewicz-II "describes various interpolation techniques performed by an

encoder and a decoder during the motion compensation process of video coding." ASUS-1006, Abstract, ¶2, ¶12, ¶4, ¶¶40-41, ¶43, ¶¶54-62, Figs. 1-2. Karczewicz-II encodes a block of pixels. ASUS-1006, ¶49, ¶50; ASUS-1003, Appx-B, ¶¶179-180.

It was obvious to modify Karczewicz-I's encoding method, based on Karczewicz-II's interpolation techniques. *Supra* §III.A.3 (explaining combination). The combination teaches a method for encoding a block of pixels, comprising the operations explained below for [1b]-[1g]. ASUS-1003, Appx-B, ¶¶181-182.

[7a]. An apparatus for encoding a block of pixels, the apparatus comprising: at least one processor and at least one memory including computer program code, the at least one memory and computer program code configured to, with the at least one processor, cause the apparatus to:

The Ground 1 combination teaches [7a]. ASUS-1003, Appx-B, ¶¶183-190. The Ground 1 combination teaches a video encoder for encoding a block of pixels that performs operations described below for [7b]-[7g]. *Supra* [1a] (explaining encoder); ASUS-1005, ¶¶29-30, ¶32, Fig. 1, ¶¶51-76, Fig. 2; ASUS-1006, ¶¶40-41, ¶43, Fig. 1, ¶¶54-62, Fig. 2; ASUS-1003, Appx-B, ¶¶184-185.

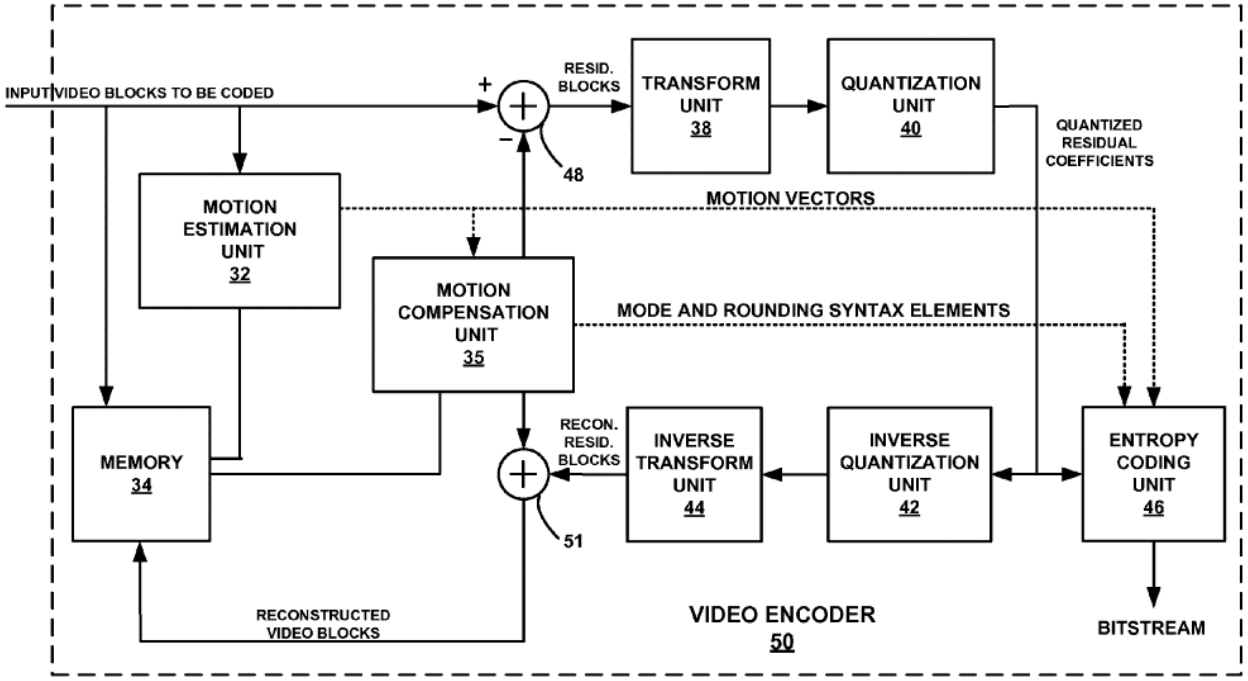

FIG. 2

ASUS-1005, Fig. 2

The video encoder encodes a block of pixels. ASUS-1005, ¶¶39-40; ASUS-1006, ¶¶49-50. Karczewicz-I and Karczewicz-II implement their encoder teachings in various types of devices (ASUS-1005, ¶3; ASUS-1006, ¶3) using a processor and memory (computer readable medium) that includes software, which comprises computer program code. ASUS-1005, ¶12, ¶38, ¶¶98-100; ASUS-1006, ¶¶25-27, ¶¶118-120. The medium comprises well-known memory types (e.g., RAM, ROM). ASUS-1005, ¶99; ASUS-1006, ¶119; ASUS-1003, Appx-B, ¶¶186-188.

Karczewicz-I and Karczewicz-II teach the processor executing computer program code in memory to carry out their teachings. ASUS-1005, ¶¶99-100; ASUS-1006, ¶48, ¶¶119-120. The apparatus further performs operations described

below for [7b]-[7g]. ASUS-1003, Appx-B, ¶¶189-190.

[13a]. A computer program product for encoding a block of pixels, the computer program product comprising at least one non-transitory computer readable storage medium having computer executable program code portions stored therein, the computer executable program code portions comprising program code instructions configured to:

The Ground 1 combination teaches the preamble. ASUS-1003, Appx-B, ¶¶191-193. As explained above, Karczewicz-I and Karczewicz-II implement their video-encoder teachings and perform operations for encoding blocks of pixels using software stored in computer-readable storage media and executed by a processor. *Supra* [7a]; ASUS-1005, ¶12, ¶38, ¶¶98-100; ASUS-1006, ¶¶25-26, ¶48, ¶¶119-120. These types of media (e.g., RAM, ROM) are non-transitory. ASUS-1005, ¶99; ASUS-1006, ¶119. The software includes computer-executable program code portions that comprise program code instructions. ASUS-1005, ¶100; ASUS-1006, ¶26, ¶120; ASUS-1003, Appx-B, ¶192.

The software performs operations taught by the Ground 1 combination, including those described below for [13b]-[13g]. The combination further teaches forming "a computer program product" using the software media. ASUS-1005, ¶99; ASUS-1006, ¶119. Therefore, the Ground 1 combination teaches a computer program product as recited in [13a] that performs the operations described below for [13b]-[13g]. ASUS-1003, Appx-B, ¶193.

[1b]/[7b]/[13b] [determining/determine], for a current block, a first reference block based on a first motion vector and a second reference block based on a second motion vector, wherein the pixels of the current block, the first reference block, and the second reference block have values with a first precision;

The Ground 1 combination teaches [1b], [7b], and [13b]. ASUS-1003, Appx-B, ¶¶194-202. Karczewicz-I and Karczewicz-II perform motion estimation and compensation. ASUS-1005, ¶4, ¶51, ¶53, Fig. 2; ASUS-1006, ¶4, ¶6, ¶54, ¶56, Fig. 2; ASUS-1003, Appx-B, ¶¶194-195.

Motion estimation "generates motion vectors" that point to reference blocks (predictive video blocks) **for a current block** and "indicate the displacement of the video blocks relative to corresponding prediction video blocks in predictive reference frame(s) or other coded units." ASUS-1005, ¶7, ¶¶53-54; ASUS-1006, ¶4, ¶56; ASUS-1003, Appx-B, ¶196.

For motion compensation, the encoder determines reference blocks for further calculations and ascertains values of reference blocks, thereby **determining reference blocks** (e.g., predictive video blocks), **based on motion vectors**. ASUS-1005, ¶7, ¶53; ASUS-1006, ¶73, ¶4, ¶56, ¶58; ASUS-1003, Appx-B, ¶197.

Karczewicz-I teaches video encoding using bi-directional prediction "from two lists of data[.]" ASUS-1005, ¶6, ¶5, ¶22, ¶42, ¶54; ASUS-1003, Appx-B, ¶198. The reference data from the two lists include data in two reference blocks and when the current block is a B block having two motion vectors, two reference

blocks are determined based on these two motion vectors. ASUS-1005, ¶7, ¶¶53-54. Therefore, the combination teaches **determining, for a current block, a first reference block based on a first motion vector and a second reference block based on a second motion vector**. ASUS-1003, Appx-B, ¶199.

Karczewicz-II teaches that the pixels of the current block, the first reference block, and the second reference block have values with a first precision (8 bits). "Precision" is at least satisfied by "a number of bits needed to represent possible values." *Supra* §II.D.1. Karczewicz-II teaches this precision (ASUS 1006, ¶89; *supra* §II.D.1), with 8 bits needed to represent possible pixel values for the two reference blocks. Table 5 shows that integer pixels (e.g., integer pixel x) include possible values between 0-255, and 8-bit unsigned numbers ("8u") are needed to represent these possible values. ASUS-1006, ¶103, Table 5.

TABLE 5

positions {a, c, d, l} of FIGS. 4A-4D

Operation	Comment	Min value	Max value	Register size
r1 = x	r1 is integer pixel x	0	255	8u
r1 = r1 << 5	r1 is 32 * x	0	8160	13u
r2 = y0	r2 is y0 (y0 is a one-dimensional (1-D) half-pixel such as b1, h1, ee1 and hh1 before shifting down)	−2550	10710	15s
r1 = r1 + r2	r1 is 32 * x + y0	−2550	18870	16s
r1 = r1 + 32	r1 is 32 * x + y0 + 32	−2518	18902	16s
r1 = r1 >> 6	r1 is (32 * x + y0 + 32) >> 6	−39	295	11s
r1 = max (0, r1)	clip r1 on the low side	0	295	10u
r1 = min (255, r1)	clip r1 on the high side	0	255	8u

ASUS-1006, Table 5

ASUS-1003, Appx-B, ¶¶200-201.

Therefore, the combination teaches or suggests that the pixels of the current block, the first reference block, and the second reference block have values with a first precision (8 bits). ASUS-1003, Appx-B, ¶202.

[1c]/[7c]/[13c] [using/use] said first reference block to obtain a first prediction, said first prediction having a second precision, which is higher than said first precision;

The Ground 1 combination teaches limitations [1c], [7c], and [13c]. ASUS-1003, Appx-B, ¶¶203-217.

Karczewicz-I teaches calculating a bi-directional prediction using a default weighted prediction mode, with equal weights assigned to two reference blocks (ASUS-1005, ¶55, ¶24, ¶44, ¶48), calculated as an average of two predictions (pred0(i,j) and pred1(i,j):

Bidirectional prediction: pred(*i,j*)=(pred0(*i,j*)+pred1(*i,j*)+1)>>1

ASUS-1005, ¶60; ASUS-1003, Appx-B, ¶¶204-205.

Here, pred0(i,j) is a first prediction based on a "motion compensated reference area[]… obtained from list 0… reference picture." ASUS-1005, ¶¶57-58. Likewise, pred1(i,j) is obtained from list 1. *Id*. The motion compensated reference area refers to a reference block since Karczewicz-I teaches bi-directional prediction based on reference blocks. *Supra* [1b]/[7b]/[13b]; ASUS-1005, ¶7, ¶¶53-54. The combination of Karczewicz-I and Karczewicz-II does not change the reference blocks on which Karczewicz-I's predictions are based. *Supra* §III.A.3 (explaining combination). Thus, the combination teaches **using said first reference block** (block from list 0) **to obtain a first prediction**. ASUS-1003, Appx-B, ¶¶206-207.

Karczewicz-I uses the reference block to obtain predictions in three

scenarios (among others). *Supra* §III.A.3 (subsections explaining "Scenarios 1-3"). In each, the combination teaches said first prediction having a second precision higher than the first. ASUS-1003, Appx-B, ¶208.

Scenario 1. For bi-directional prediction averaging a half-pixel prediction and an integer pixel prediction, the combination of Karczewicz-II to Karczewicz-I uses a non-rounded half-pixel prediction (non-rounded(pred0(i,j))) and a left-shifted integer pixel prediction (pred1(i,j)<<5):

$$pred(i,j)=(non\text{-}rounded(pred0(i,j))+pred1(i,j)<<5+32)>>6$$

Supra §III.A.3 (explaining combined teachings for Scenario 1); ASUS-1003, Appx-B, ¶209.

Karczewicz-II teaches that the non-rounded half-pixel prediction (e.g., b1) has possible values from -2550 to 10710 and that a 15-bit signed number ("15s") is needed to represent these possible values. Karczewicz-II further teaches that the left-shifted integer pixel prediction (r1<<5) has possible values from 0 to 8160 and that a 13-bit unsigned number ("13u") is needed to represent these possible values. ASUS-1006, ¶103, Table 5.

TABLE 5

positions {a, c, d, l} of FIGS. 4A-4D

Operation	Comment	Min value	Max value	Register size
r1 = x	r1 is integer pixel x	0	255	8u
r1 = r1 << 5	r1 is 32 * x	0	8160	13u
r2 = y0	r2 is y0 (y0 is a one-dimensional (1-D) half-pixel such as b1, h1, ee1 and hh1 before shifting down)	−2550	10710	15s
r1 = r1 + r2	r1 is 32 * x + y0	−2550	18870	16s
r1 = r1 + 32	r1 is 32 * x + y0 + 32	−2518	18902	16s
r1 = r1 >> 6	r1 is (32 * x + y0 + 32) >> 6	−39	295	11s
r1 = max (0, r1)	clip r1 on the low side	0	295	10u
r1 = min (255, r1)	clip r1 on the high side	0	255	8u

ASUS-1006, Table 5

ASUS-1003, Appx-B, ¶210.

The first and second predictions each need at least 13 bits to represent their possible values, which is higher than the 8-bit precision for the pixels of the current block, the first reference block, and the second reference block. *Supra* [1b]/[7b]/[13b]. Thus, the combination teaches the first prediction having a second precision (13 bits), which is higher than the first (8 bits) under Scenario 1. *Supra*

§II.D.1; ASUS-1003, Appx-B, ¶211.

Scenario 2. For a center-pixel prediction and a half-pixel prediction, the combination of Karczewicz-II to Karczewicz-I uses a partially-rounded center-pixel prediction (non-rounded(pred0(i,j))>>5) and a non-rounded half-pixel prediction (non-rounded(pred1(i,j))):

pred(*i*,*j*)=(non-rounded(pred0(*i*,*j*))>>5+non-rounded(pred1(*i*,*j*)) +32)>>6

Supra §III.A.3 (subsection "Scenario 2" explaining calculations and combined teachings); ASUS-1003, Appx-B, ¶212.

Karczewicz-II teaches that the partially-rounded center-pixel prediction (j1>>1)[5] has possible values from -4957 to 13116 and that a 15-bit signed number ("15s") is needed to represent these possible values. The non-rounded half-pixel prediction (e.g., b1) has possible values from -2550 to 10710, and a 15-bit signed integer ("15s") is needed to represent these possible values. ASUS-1006, ¶105,

[5] The equations in Table 3 of Karczewicz-II show that j1 is shifted to the right by 5 bits. Table 8 accomplishes this in two steps. The first step ("r2 = j1") is to slightly round the value of j1 to fit in a 16-bit register, which shaves off 4 bits from the right side (the least significant bits). ASUS-1006, ¶59, ¶10, ¶39, ¶53; *supra* §III.A.3. The second step ("r2 = r2 >> 1") right shifts one more bit, bringing the total right-shift amount to 5 bits.

Table 8.

TABLE 8

positions {f, i, k, n} of FIGS. 4A-4D

Operation	Comment	Min value	Max value	Register size
r1 = y0	r1 is y0 (1-D half-pixel such as b1, h1, ee1 and hh1 before shifting down)	−2550	10710	15s
r2 = j1	r2 is j1 (2-D half-pixel j1 before shifting down)	−9914	26232	16s
r2 = r2 >> 1	r2 is j1 >> 1	−4957	13116	15s
r1 = r1 + r2	r1 is y0 + (j1 >> 1)	−7507	23826	16s
r1 = r1 + 32	r1 is y0 + (j1 >> 1) + 32	−7491	23842	16s
r1 = r1 >> 6	r1 is (y0 + (j1 >> 1) + 32) >> 6	−235	745	11s
r1 = max (0, r1)	clip r1 on the low side	0	745	10u
r1 = min (255, r1)	clip r1 on the high side	0	255	8u

ASUS-1006, Table 8

ASUS-1003, Appx-B, ¶213.

Therefore, for Scenario 2, the first and second predictions each have a second precision of 15 bits, which is higher than the first precision (8 bits) for, e.g., pixels of the current block. *Supra* [1b]/[7b]/[13b], §II.D.1; ASUS-1003, Appx-B, ¶214.

Scenario 3. For two half-pixel predictions, the combination of Karczewicz-II to Karczewicz-I uses non-rounded half-pixel predictions:

$$pred(i,j)=(non\text{-}rounded(pred0(i,j))+non\text{-}rounded(pred1(i,j)) +32)>>6$$

Supra §III.A.3 (subsection "Scenario 3" explaining calculations and combined teachings); ASUS-1003, Appx-B, ¶215.

Karczewicz-II teaches that the non-rounded half-pixel predictions (e.g., b1, h1) each has values from -2550 to 10710, and a 15-bit signed number ("15s") is needed to represent these possible values. ASUS-1006, ¶103, Table 6.

TABLE 6

positions {e, g, m, o} of FIGS. 4A-4D

Operation	Comment	Min value	Max value	Register size
r1 = y0	r1 is y0 (y0 is a 1-D half-pixel such as b1, h1, ee1 and hh1 before shifting down)	−2550	10710	15s
r2 = y1	r2 is y1 (y1 is a 1-D half-pixel such as b1, h1, ee1 and hh1 before shifting down)	−2550	10710	15s
r1 = r1 + r2	r1 is y0 + y1	−5100	21420	16s
r1 = r1 + 32	r1 is y0 + y1 + 32	−5068	21452	16s
r1 = r1 >> 6	r1 is (y0 + y1 + 32) >> 6	−79	335	11s
r1 = max (0, r1)	clip r1 on the low side	0	335	10u
r1 = min (255, r1)	clip r1 on the high side	0	255	8u

ASUS-1006, Table 6

ASUS-1003, Appx-B, ¶216.

Thus, for Scenario 3, the first and second predictions each have a second precision of 15 bits, which is higher than the first precision (8 bits) for, e.g., pixels of the current block. *Supra* [1b]/[7b]/[13b], §II.D.1; ASUS-1003, Appx-B, ¶217.

[1d]/[7d]/[13d] [using/use] said second reference block to obtain a second prediction, said second prediction having the second precision;

The Ground 1 combination teaches [1d], [7d], and [13d]. ASUS-1003, Appx-B, ¶¶218-224.

As explained above, Karczewicz-I teaches calculating a bi-directional prediction as an average of two predictions, and pred1(i,j) is a second prediction obtained based on a reference block. *Supra* [1c]/[7c]/[13c]; ASUS-1005, ¶¶57-58, ¶60. The combination of Karczewicz-I and Karczewicz-II does not change the reference blocks on which Karczewicz-I's predictions are based. *Supra* §III.A.3 (explaining combination). Thus, the combination teaches **using said second reference block** (block from list 1) **to obtain a second prediction** in three scenarios. *Supra* §III.A.3 (subsections explaining Scenarios 1-3); ASUS-1003, Appx-B, ¶¶219-220.

As explained above, in Scenario 1, the combination teaches **said second prediction** (left-shifted integer pixel prediction pred1(i,j)<<5) **having the second precision** (13 bits). *Supra* [1c]/[7c]/[13c]; ASUS-1003, Appx-B, ¶222.

In Scenario 2, the combination teaches **said second prediction** (non-rounded half-pixel prediction non-rounded(pred1(i,j))) **having the second precision** (15 bits). *Supra* [1c]/[7c]/[13c]; ASUS-1003, Appx-B, ¶223.

In Scenario 3, the combination teaches **said second prediction** (non-

rounded half-pixel prediction non-rounded(pred1(i,j))) **having the second precision** (15 bits). *Supra* [1c]/[7c]/[13c]; ASUS-1003, Appx-B, ¶224.

[1e]/[7e]/[13e] [obtaining/obtain] a combined prediction based at least partly upon said first prediction and said second prediction;

The Ground 1 combination teaches [1e], [7e], and [13e]. ASUS-1003, Appx-B, ¶¶225-231.

Karczewicz-I teaches obtaining a bi-directional prediction by averaging the first and second predictions:

Bidirectional prediction: pred(i,j)=(pred0(i,j)+pred1(i,j)+1)>>1

ASUS-1005, ¶60 (adding predictions and dividing by two with ">>1"); *supra* [1c]-[1d]/[7c]-[7d]/[13c]-[13d]. The Ground 1 combination combines these bi-prediction teachings with Karczewicz-II. *Supra* §III.A.3 (explaining MTC); ASUS-1003, Appx-B, ¶¶226-227.

Under each of the three scenarios discussed above (*supra* §III.A.3), the combination teaches **obtaining a combined prediction** (a sum of the first prediction, the second prediction, and a rounding offset) **based at least partly upon said first prediction and said second prediction.** ASUS-1003, Appx-B, ¶228.

For Scenario 1, it was obvious to calculate the bi-directional prediction based on a non-rounded half-pixel prediction and a left-shifted integer pixel

prediction:

$$pred(i,j)=(non\text{-}rounded(pred0(i,j))+pred1(i,j)<<5+32)>>6$$

Supra §III.A.3 (subsection "Scenario 1" explaining calculations). The combination calculates a sum of the first prediction (e.g., non-rounded(pred0(i,j))), the second prediction (e.g., pred1(i,j)<<5) and a rounding offset (32). *See* ASUS-1006, ¶103, Table 5; ASUS-1003, Appx-B, ¶229.

Scenario 2, it was obvious to calculate the bi-directional prediction based on a partially-rounded center-pixel prediction and a non-rounded half-pixel prediction:

$$pred(i,j)=(non\text{-}rounded(pred0(i,j))>>5+non\text{-}rounded(pred1(i,j))+32)>>6$$

Supra §III.A.3 (subsection "Scenario 2" explaining calculations). The combination calculates a sum of the first prediction (e.g., non-rounded(pred0(i,j))>>5), the second prediction (e.g., non-rounded(pred1(i,j))), and a rounding offset (32). *See* ASUS-1006, ¶105, Table 8; ASUS-1003, Appx-B, ¶230.

Scenario 3, it was obvious to calculate the bi-directional prediction based on two non-rounded half-pixel predictions:

$$pred(i,j)=(non\text{-}rounded(pred0(i,j))+non\text{-}rounded(pred1(i,j))+32)>>6$$

Supra §III.A.3 (subsection "Scenario 3" explaining calculations). The combination calculates a sum of the first prediction (e.g., non-rounded(pred0(i,j))), the second prediction (e.g., non-rounded(pred1(i,j))), and a rounding offset (32). ASUS-1006,

¶103, Table 6; ASUS-1003, Appx-B, ¶231.

[1f]/[7f]/[13f] [decreasing/decrease] a precision of said combined prediction by shifting bits of the combined prediction to the right; and

The Ground 1 combination teaches [1f], [7f], and [13f]. ASUS-1003, Appx-B, ¶¶232-239.

As explained above, the Ground 1 combination teaches the combined prediction as a sum of the first prediction, the second prediction, and a rounding offset. *Supra* [1e]/[7e]/[13e]. Karczewicz-I and Karczewicz-II teach the operation ">>" for shifting bits to the right. ASUS-1005, ¶57 (">> is a right shift operation[.]"), ¶55, ¶60; ASUS-1006, ¶94. The combination teaches **decreasing a precision of said combined prediction by shifting bits of the combined prediction to the right** for each of the three scenarios discussed above. ASUS-1003, Appx-B, ¶233.

Scenario 1. Here, the combination calculates bi-directional prediction based on a non-rounded half-pixel prediction and a left-shifted integer pixel prediction:

$$pred(i,j)=(non\text{-}rounded(pred0(i,j))+pred1(i,j)<<5+32)>>6$$

Supra §III.A.3 (subsection "Scenario 1" explaining calculations). Bits of the combined prediction (non-rounded(pred0(i,j))+pred1(i,j)<<5+32) are shifted to the right by 6 bits (e.g., ">>6"). ASUS-1003, Appx-B, ¶234.

The shifting decreases a precision of said combined prediction. ASUS-1006,

¶103, Table 5.

TABLE 5

positions {a, c, d, l} of FIGS. 4A-4D

Operation	Comment	Min value	Max value	Register size
r1 = x	r1 is integer pixel x	0	255	8u
r1 = r1 << 5	r1 is 32 * x	0	8160	13u
r2 = y0	r2 is y0 (y0 is a one-dimensional (1-D) half-pixel such as b1, h1, ee1 and hh1 before shifting down)	−2550	10710	15s
r1 = r1 + r2	r1 is 32 * x + y0	−2550	18870	16s
r1 = r1 + 32	r1 is 32 * x + y0 + 32	−2518	18902	16s
r1 = r1 >> 6	r1 is (32 * x + y0 + 32) >> 6	−39	295	11s
r1 = max (0, r1)	clip r1 on the low side	0	295	10u
r1 = min (255, r1)	clip r1 on the high side	0	255	8u

ASUS-1006, Table 5

In this table, the operation “r1=r1+32 calculates the sum of a rounding offset (32) with r1, which in turn is the sum of a non-rounded half-pixel prediction (y0 which can take the value of half-pixel prediction b1) and an integer pixel prediction that was left-shifted by 5 bits (32*x). Karczewicz-II teaches that this sum has possible values between -2518 and 18902 and that a 16-bit signed number (“16s”) is needed

to represent these possible values. Next, the operation "r1=r1>>6" shifts the sum 6 bits to the right. The result has possible values between -39 and 295; an 11-bit signed number ("11s") is needed to represent these possible values. Thus, the combination decreases a precision of said combined prediction from 16 bits to 11-bits. *Supra* §II.D.1; ASUS-1003, Appx-B, ¶235.

Scenario 2. Here, the combination calculates bi-directional prediction based on a partially-rounded center-pixel prediction (non-rounded(pred0(i,j))>>5) and a non-rounded half-pixel prediction:

pred(*i*,*j*)=(non-rounded(pred0(*i*,*j*))>>5+non-rounded(pred1(*i*,*j*)) +32)>>6

Supra §III.A.3 (subsection "Scenario 2" explaining calculations). Bits of the combined prediction (non-rounded(pred0(i,j))>>5+non-rounded(pred1(i,j)) +32) are shifted to the right (">>6"). ASUS-1003, Appx-B, ¶236.

The shifting decreases a precision of said combined prediction. ASUS-1006, ¶105, Table 8.

TABLE 8

positions {f, i, k, n} of FIGS. 4A-4D

Operation	Comment	Min value	Max value	Register size
r1 = y0	r1 is y0 (1-D half-pixel such as b1, h1, ee1 and hh1 before shifting down)	−2550	10710	15s
r2 = j1	r2 is j1 (2-D half-pixel j1 before shifting down)	−9914	26232	16s
r2 = r2 >> 1	r2 is j1 >> 1	−4957	13116	15s
r1 = r1 + r2	r1 is y0 + (j1 >> 1)	−7507	23826	16s
r1 = r1 + 32	r1 is y0 + (j1 >> 1) + 32	−7491	23842	16s
r1 = r1 >> 6	r1 is (y0 + (j1 >> 1) + 32) >> 6	−235	745	11s
r1 = max (0, r1)	clip r1 on the low side	0	745	10u
r1 = min (255, r1)	clip r1 on the high side	0	255	8u

ASUS-1006, Table 8

The operation “r1=r1+32” calculates the sum of a partially-rounded center-pixel prediction (“j1>>1”), a non-rounded half-pixel prediction (y0, which may take the value of b1), and a rounding offset (32). Karczewicz-II teaches that this sum has possible values between -7491 and 23842 and that a 16-bit signed number (“16s”) is needed to represent these possible values. Next, the operation “r1=r1>>6” shifts the sum 6 bits to the right. The result has possible values between -235 and 745; an 11-bit signed number (“11s”) is needed to represent these possible values. Thus, the combination decreases a precision of said combined prediction from 16 bits to 11 bits. *Supra* §II.D.1; ASUS-1003, Appx-B, ¶237.

Scenario 3. Here, the combination calculates bi-directional prediction based on two non-rounded half-pixel predictions:

pred(*i*,*j*)=(non-rounded(pred0(*i*,*j*))+non-rounded(pred1(*i*,*j*)) +32)>>6

Supra §III.A.3 (subsection "Scenario 3" explaining calculations). Bits of the combined prediction (non-rounded(pred0(i,j))+non-rounded(pred1(i,j)) +32) are shifted to the right (">>6"). ASUS-1003, Appx-B, ¶238.

The shifting decreases a precision of said combined prediction. ASUS-1006, ¶103, Table 6.

TABLE 6

positions {e, g, m, o} of FIGS. 4A-4D

Operation	Comment	Min value	Max value	Register size
r1 = y0	r1 is y0 (y0 is a 1-D half-pixel such as b1, h1, ee1 and hh1 before shifting down)	−2550	10710	15s
r2 = y1	r2 is y1 (y1 is a 1-D half-pixel such as b1, h1, ee1 and hh1 before shifting down)	−2550	10710	15s
r1 = r1 + r2	r1 is y0 + y1	−5100	21420	16s
r1 = r1 + 32	r1 is y0 + y1 + 32	−5068	21452	16s
r1 = r1 >> 6	r1 is (y0 + y1 + 32) >> 6	−79	335	11s
r1 = max (0, r1)	clip r1 on the low side	0	335	10u
r1 = min (255, r1)	clip r1 on the high side	0	255	8u

ASUS-1006, Table 6

The operation "r1=r1+32" calculates the sum of two non-rounded half-pixel predictions (y0 and y1, which may take the values of b1 and h1), and a rounding

offset (32). Karczewicz-II teaches that this sum has possible values from -5068 to 21452, and a 16-bit signed number ("16s") is needed to represent these possible values. Next, the operation "r1=r1>>6" shifts the sum 6 bits to the right. The result has possible values from-79 to 335; an 11-bit signed number("11s") is needed to represent these possible values. Thus, the combination decreases a precision of said combined prediction from 16 to 11 bits. *Supra* §II.D.1; ASUS-1003, Appx-B, ¶239.

[1g]/[7g]/[13g] [encoding/encode] residual data in a bitstream, wherein the residual data is determined based upon a difference between the combined prediction and the block of pixels.

The Ground 1 combination teaches [1g], [7g], and [13g]. ASUS-1003, Appx-B, ¶¶240-242.

The combination teaches determining the combined prediction in the motion compensation process. *Supra* [1b]-[1f]/[7b]-[7f]/[13b]-[13f]. The Karczewicz references teach **determining residual data** (e.g., a residual video block) **based upon a difference between the combined prediction** (e.g., prediction data) **and the block of pixels** (e.g., original video block). ASUS-1005, ¶73 ("Video encoder 50 forms a residual video block by subtracting the prediction data from the original video block being coded."), ¶7, ¶89; ASUS-1006, ¶4, ¶6, ¶35, ¶50, ¶58, ¶60, ¶73, Fig. 2; ASUS-1003, Appx-B, ¶241. The combination further teaches **encoding** (e.g., transforming, quantizing, and entropy coding) **residual data in a bitstream**.

ASUS-1005, ¶¶73-75, Fig.2; ASUS-1006, ¶5, ¶35, ¶50, ¶¶60-61, ¶110, Fig. 2; ASUS-1003, Appx-B, ¶242.

(b) Dependent Claims 2, 8, and 14

2. The method according to claim 1, …
8. The apparatus according to claim 7, …
14. The computer program product according to claim 13,
wherein in an instance in which said first motion vector points to a subpixel, said first prediction is obtained by interpolation using pixel values of said first reference block.

The Ground 1 combination teaches claims 1, 7, and 13, and the limitations of claims 2, 8, and 14. *Supra* §III.A.3, §III.A.4(a); ASUS-1003, Appx-B, ¶¶243-251.

Karczewicz-II teaches obtaining predictions by interpolation using pixel values of reference blocks when motion vectors point to subpixels. ASUS-1006, ¶42 ("The interpolation techniques of this disclosure may be performed by any encoding device that supports motion compensated interpolation to sub-pixel resolution."), ¶7, ¶66, ¶¶68-72, Figs. 4A-4D; ASUS-1003, Appx-B, ¶244.

The Ground 1 combination satisfies this claim limitation for any of the three scenarios discussed above. ASUS-1003, Appx-B, ¶245.

Scenario 1 is **an instance in which said first motion vector points to a subpixel** (a half-pixel position). *Supra* §III.A.3 (subsection "Scenario 1" explaining interpolation teachings). The half-pixel position refers to a subpixel.

ASUS-1006, ¶74, Figs. 4A-4D, ¶10; ASUS-1003, Appx-B, ¶246.

Karczewicz-II teaches **obtaining the first prediction** (the non-rounded half-pixel prediction) **by interpolation using pixel values of said first reference block**. ASUS-1006, ¶93 (H.264 defines one interpolation process for sub-pixels …). The interpolation is performed using pixel values of "six closest integer pixels that surround" the subpixel, which are located in the reference block that the first motion vector points to. ASUS-1006, Fig. 4B.

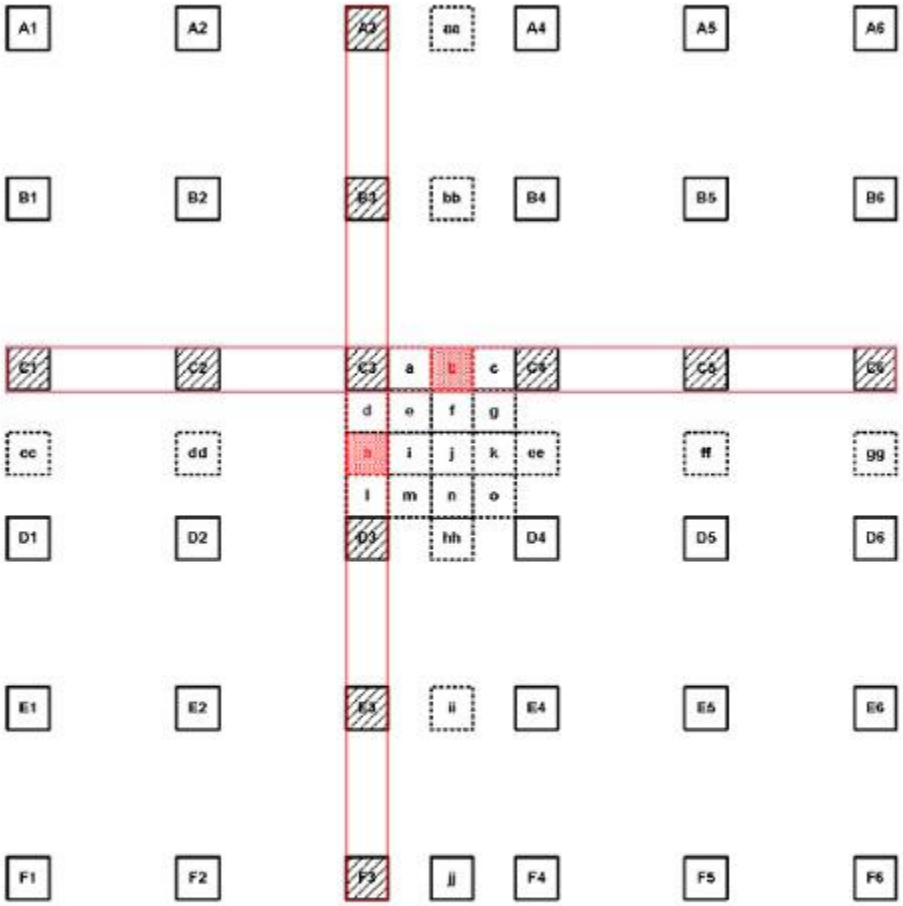

FIG. 4B

ASUS-1006, Fig. 4B

Thus, the interpolation is performed using pixel values of said first reference block. ASUS-1003, Appx-B, ¶247.

Scenario 2 is an instance in which said first motion vector points to a **subpixel** (a center-pixel position). *Supra* §III.A.3 (subsection "Scenario 2" explaining interpolation teachings). The center-pixel position refers to a subpixel.

ASUS-1006, ¶74, Figs. 4A-4D, ¶10; ASUS-1003, Appx-B, ¶248.

Karczewicz-II teaches **obtaining the first prediction** (the partially-rounded center-pixel prediction) **by interpolation using pixel values of said first reference block**. ASUS-1006, ¶95 ("To interpolate sub-pixel 'j,' …"). The interpolation is performed using non-rounded half-pixel predictions, which are located in the reference block that the first motion vector points to. *See* ASUS-1006, Fig. 4C.

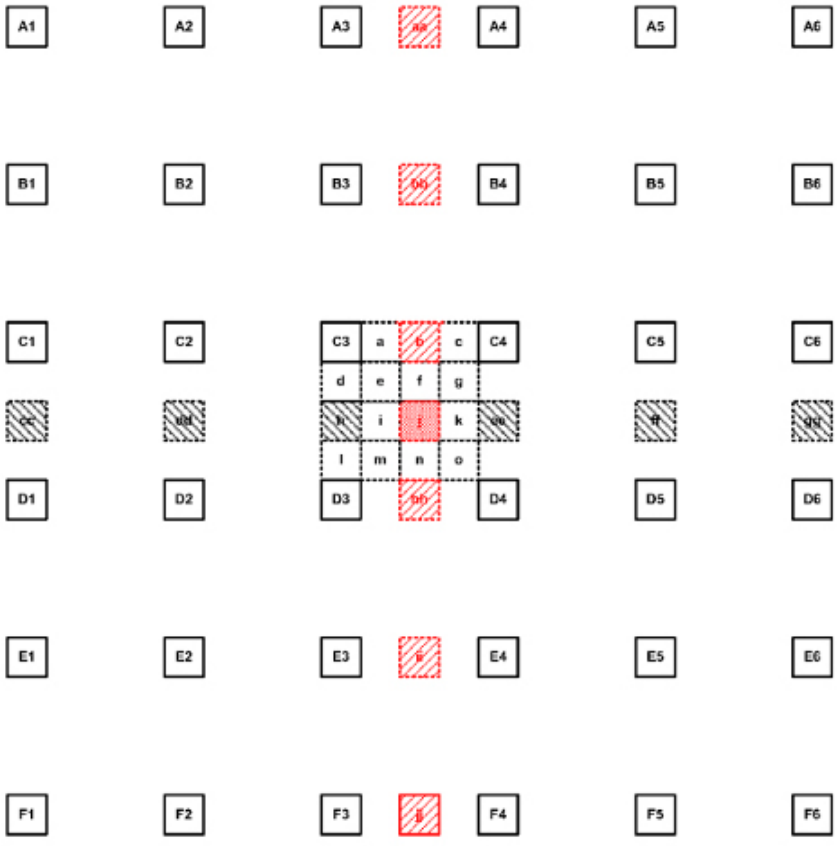

FIG. 4C

ASUS-1006, Fig. 4C

Thus, the interpolation is performed using pixel values of said first reference block. ASUS-1003, Appx-B, ¶249.

Scenario 3. Here, the first motion vector points to a half-pixel position. *Supra* §III.A.3 (subsection "Scenario 3" explaining interpolation teachings). For the same reasons as explained for Scenario 1, the combination teaches that, in an instance in which said first motion vector points to a subpixel, said first prediction is obtained by interpolation using pixel values of said first reference block. ASUS-1003, Appx-B, ¶250.

Additionally, the '267 patent confirms that the limitations of claims 2, 8, and 14 were known in the prior art by describing it in the "Background Information" section. ASUS-1001, 2:60-3:11; ASUS-1003, Appx-B, ¶251.

(c) Dependent Claims 3, 9, and 15

3. The method according to claim 2, …
9. The apparatus according to claim 8, …
15. The computer program product according to claim 14,
wherein said first prediction is obtained by interpolation using values of said first reference block by: right shifting a sum of a P-tap filter using values of said first reference block.

The Ground 1 combination teaches claims 2, 8, and 14, and the limitations of claims 3, 9, and 15 under Scenario 2. *Supra* §III.A.4(b); ASUS-1003, Appx-B, ¶¶252-257.

In Scenario 2, the combination teaches the following equation for bi-

directional prediction:

pred(*i*,*j*)=(non-rounded(pred0(*i*,*j*))>>5+non-rounded(pred1(*i*,*j*)) +32)>>6

Supra §III.A.3 (subsection "Scenario 2" explaining prediction teachings). The first prediction "non-rounded(pred0(i,j))>>5" is a partially-rounded center-pixel prediction. *Id.*; *supra* [1c]/[7c]/[13c]; ASUS-1003, Appx-B, ¶253.

Karczewicz-II teaches calculating the partially-rounded center-pixel prediction (j1>>5) by first determining a sum of a 6-tap filter (j1):

j1=aa1−5*bb1+20*b1+20*hh1−5*ii1+jj1

ASUS-1006, ¶95; *supra* §III.A.4(b). This sum is a non-rounded version of the center-pixel prediction. ASUS-1003, Appx-B, ¶254.

Next, the sum is right-shifted by 5 bits to obtain the partially-rounded center-pixel prediction (j1>>5). *Supra* §III.A.3; ASUS-1006, ¶99, ¶105, Tables 8, 3.

TABLE 3

a = (C3 << 5 + b1 + 32) >> 6
c = (C4 << 5 + b1 + 32) >> 6
d = (C3 << 5 + h1 + 32) >> 6
l = (D3 << 5 + h1 + 32) >> 6
f = (j1 >> 5 + b1 + 32) >> 6
i = (j1 >> 5 + h1 + 32) >> 6
k = (j1 >> 5 + ee1 + 32) >> 6
n = (j1 >> 5 + hh1 + 32) >> 6

ASUS-1006, Table 3

ASUS-1003, Appx-B, ¶255.

Therefore, the combination teaches that **said first prediction** (partially-

rounded center-pixel prediction) **is obtained by interpolation using values of said first reference block by: right shifting** (">>5") **a sum of a P-tap filter** (non-rounded center-pixel prediction "non-rounded(pred0(i,j))") **using values of said first reference block**. ASUS-1003, Appx-B, ¶256.

(d) Dependent Claims 4, 10, and 16

4. The method according to claim 2, …
10. The apparatus according to claim 8, …
16. The computer program product according to claim 14,
wherein in an instance in which said second motion vector points to an integer sample, said second prediction is obtained by shifting values of said second reference block to the left.

The Ground 1 combination teaches claims 2, 8, and 14, and the limitations of claims 4, 10, and 16 under Scenario 1. *Supra* §III.A.4(b); ASUS-1003, Appx-B, ¶¶258-263.

Scenario 1 **is an instance in which said second motion vector points to an integer sample**. *Supra* §III.A.3 (subsection "Scenario 1" explaining motion vector pointing to integer pixel position). ASUS-1003, Appx-B, ¶259.

The combination calculates bi-directional prediction:

$$pred(i,j)=(non\text{-}rounded(pred0(i,j))+pred1(i,j)<<5+32)>>6$$

Supra §III.A.3 (subsection "Scenario 1"). The second prediction "pred1(i,j)<<5" is a left-shifted integer pixel prediction. *Supra* §III.A.3, [1c]-[1d]/[7c]-[7d]/[13c]-[13d]; ASUS-1003, Appx-B, ¶260.

Karczewicz-II teaches calculating the left-shifted integer pixel prediction by

shifting the pixel value to the left (e.g., C3<<5). *Supra* §III.A.3; ASUS-1006, ¶99, Table 3.

TABLE 3

a = (C3 << 5 + b1 + 32) >> 6
c = (C4 << 5 + b1 + 32) >> 6
d = (C3 << 5 + h1 + 32) >> 6
l = (D3 << 5 + h1 + 32) >> 6
f = (j1 >> 5 + b1 + 32) >> 6
i = (j1 >> 5 + h1 + 32) >> 6
k = (j1 >> 5 + ee1 + 32) >> 6
n = (j1 >> 5 + hh1 + 32) >> 6

ASUS-1006, Table 3

The left-shifted value is of an integer pixel sample (e.g., C3) that is part of the second reference block. ASUS-1006, ¶74 ("integer-pixel sample 'C3'"), ¶93; ASUS-1003, Appx-B, ¶261.

Karczewicz-I and Karczewicz-II teach that motion compensation is performed on a block basis. *Supra* [1b], [1g]/[7b], [7g]/[13b], [13g]; ASUS-1005, ¶7; ASUS-1006, ¶4, ¶73. Thus, the left-shift operation is performed for the pixels of the current block based on corresponding pixels of the reference block. *Supra* [1b]/[7b]/[13b]; ASUS-1005, ¶7, ¶¶53-54; ASUS-1006, ¶4, ¶56; ASUS-1003, Appx-B, ¶262.

Therefore, the combination teaches that **in an instance in which said second motion vector points to an integer sample, said second prediction** (left-

shifted integer pixel prediction) **is obtained by shifting values of said second reference block** (values of integer pixels) **to the left** ("<<5"). ASUS-1003, Appx-B, ¶263.

(e) Dependent Claims 5, 11, and 17

5. The method according to claim 1, wherein said decreasing said precision of said combined prediction by shifting bits of the combined prediction to the right, further comprises: …
11. The apparatus according to claim 7, wherein the at least one memory and computer code are configured to cause the apparatus to decrease said precision of said combined prediction by shifting bits of the combined prediction to the right, by: …
17. The computer program product according to claim 13, wherein the program code instructions configured to decrease said precision of said combined prediction by shifting bits of the combined prediction to the right, further comprise program code instructions configured to: [inserting/insert] a rounding offset to the combined prediction before said decreasing.

The Ground 1 combination teaches claims 1, 7, and 13, and the limitations of claims 5, 11, and 17. *Supra* §III.A.4(a); ASUS-1003, Appx-B, ¶¶264-269.

As explained above, the combination teaches obtaining a combined prediction by summing the first prediction, the second prediction, and the rounding offset (e.g., 32) and decreasing a precision of the combined prediction. *Supra* [1e]-[1f]/[7e]-[7f]/[13e]-[13f]. The combination thus teaches **inserting a rounding offset (32) to the combined prediction, before said decreasing** of the precision. ASUS-1003, Appx-B, ¶265.

The value added to the sum of the first and second predictions is a rounding

offset. Karczewicz-I refers to the term that is added to the weighted sum of the first and second predictions as a "rounding adjustment." ASUS-1005, ¶63, ¶55.[6] Because "rounding adjustment" was used interchangeably with "rounding offset" and the value (32) is inserted to the combined prediction right before the rounding operation, this value is a rounding offset. ASUS-1003, Appx-B, ¶266.

Karczewicz-I and Karczewicz-II teach this rounding offset as part of its rounding process, which decreases the precision as recited by the claims. ASUS-1005, ¶55, ¶63; ASUS-1006, ¶¶96-101, Tables 1-4. Additionally, it was obvious for said decreasing to include the rounding offset because the insertion of the rounding offset is performed immediately before the right-shifting to affect the direction of the rounding. The combination includes a rounding offset to control rounding error resulting from the right-shift operation that decreases precision. This was common in the art. ASUS-1003, Appx-B, §I.D, ¶267.

Because the combination applies its operations to a computer implementation, it teaches 1) at least one memory and computer program code

[6] Karczewicz-I teaches a rounding adjustment of 2^{r-1} prior to right-shifting r bits, consistent with the Ground 1 combination's modified equation for calculating bi-directional prediction under each scenario (*supra* §III.A.3), which teaches right-shifting 6 bits and a rounding offset of $2^{6-1}=2^5=32$.

being configured to cause the apparatus to perform the operations of claim 11 (*supra* [7a] and 2) and program code instructions configured to perform the operations of claim 17 (*supra* [13a]). ASUS-1003, Appx-B, ¶¶268-269.

(f) Dependent Claims 6, 12, and 18

6. The method according to claim 1, …
12. The apparatus according to claim 7, …
18. The computer program product according to claim 13,
wherein the first precision indicates a number of bits needed to represent the values of the pixels, and the second precision indicates the number of bits needed to represent values of said first prediction and values of said second prediction.

The Ground 1 combination teaches claims 1, 7, and 13, and the limitations of claims 6, 12, and 18. *Supra* §III.A.4(a); ASUS-1003, Appx-B, ¶¶270-272.

The first precision indicates a number of bits needed to represent the values of the pixels. The combination teaches that the pixels of the current block, the first reference block, and the second reference block have values with a first precision because 8 bits are needed to represent the possible pixel values of these blocks. *Supra* [1b]/[7b]/[13b]; ASUS-1003, Appx-B, ¶271.

The second precision indicates the number of bits needed to represent values of said first prediction and values of said second prediction. The combination teaches said first prediction and second prediction having a second precision that is higher than said first precision because more bits are needed to represent the possible values of the predictions under each of Scenarios 1, 2, and 3.

Supra [1c]-[1d]/[7c]-[7d]/[13c]-[13d]; ASUS-1003, Appx-B, ¶272.

(g) Independent Claims 19, 25, and 31

[19a]. A method for decoding a block of pixels, the method comprising:

The Ground 1 combination teaches the preamble. ASUS-1003, Appx-C, ¶¶176-182.

Karczewicz-I "describes video encoding and decoding techniques applicable to bi-directional prediction" including for encoders and decoders. ASUS-1005, ¶8, ¶22, ¶¶29-30, ¶33, Fig. 1. The decoder "perform[s] the reciprocal decoding techniques to the encoding techniques."[7] ASUS-1005, ¶¶83-86, Fig. 4; ASUS-1003, Appx-C, ¶177. Karczewicz-I decodes "blocks of pixel data[.]" ASUS-1005, ¶¶39-40; ASUS-1003, Appx-C, ¶178.

Karczewicz-II "describes various interpolation techniques performed by an encoder and a decoder during the motion compensation process of video coding." ASUS-1006, Abstract, ¶2, ¶6, ¶¶40-41, 44, ¶¶63-66, Figs. 1, 3. Karczewicz-II decodes a block of pixels. ASUS-1006, ¶64, ¶66, ¶50; ASUS-1003, Appx-C, ¶¶179-180.

[7] Karczewicz-I explains that its teachings regarding bi-directional prediction (e.g., ¶¶55-60) are applicable to video decoding. ASUS-1005, ¶8, ¶35; ASUS-1003, Appx-B, §§I.A-C.

It was obvious to modify Karczewicz-I's decoding method based on Karczewicz-II's interpolation techniques. *Supra* §III.A.3 (explaining combination). The combination teaches a method for decoding a block of pixels, comprising the operations explained below for [19b]-[19g]. ASUS-1003, Appx-C, ¶¶181-182.

[25a]. An apparatus for decoding a block of pixels, the apparatus comprising: at least one processor and at least one memory including computer program code, the at least one memory and computer program code configured to, with the at least one processor, cause the apparatus to:

The Ground 1 combination teaches [25a]. ASUS-1003, Appx-C, ¶¶183-190.

The Ground 1 combination teaches a video decoder for decoding a block of pixels that performs operations described below for [25b]-[25g]. *Supra* [19a] (explaining decoder); ASUS-1005, ¶¶29-30, ¶33, Fig. 1, ¶¶83-86, Fig. 4; ASUS-1006, ¶¶15-17, ¶¶22-¶23, ¶¶40-41, ¶44, Fig. 1, ¶¶63-66, Fig. 3.

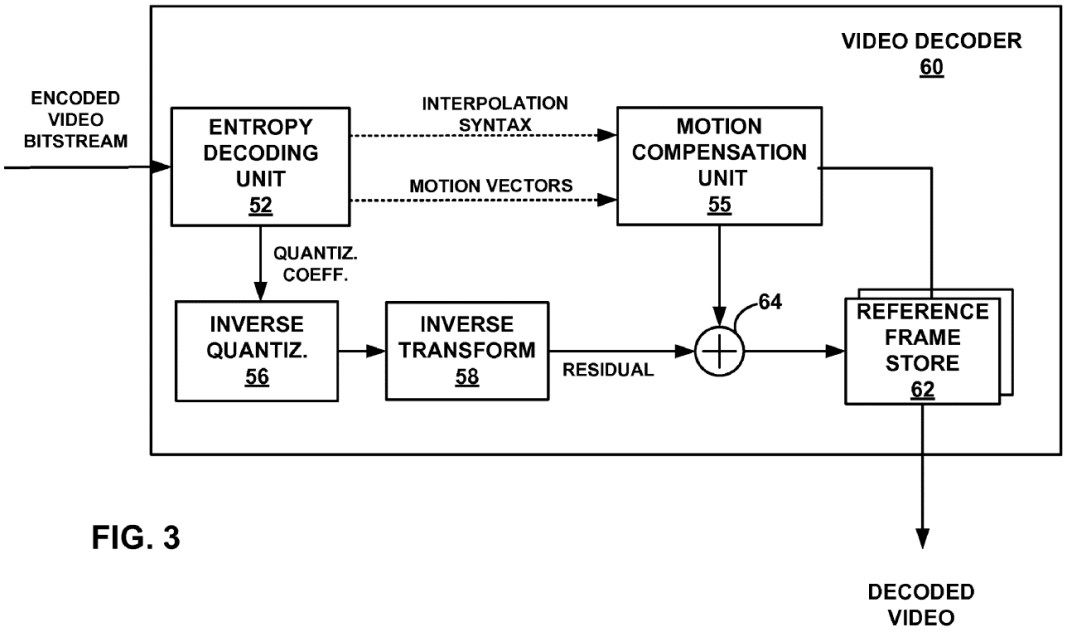

FIG. 3

ASUS-1006, FIG. 3

ASUS-1003, Appx-C, ¶¶184-185.

The video decoder decodes a block of pixels. ASUS-1005, ¶¶39-40; ASUS-1006, ¶50, ¶64, ¶66. Karczewicz-I and Karczewicz-II implement their decoder teachings in various types of devices (ASUS-1005, ¶3; ASUS-1006, ¶3) using a processor and memory (computer readable medium) that includes software, which comprises computer program code. ASUS-1005, ¶12, ¶38, ¶¶98-100; ASUS-1006, ¶¶25-27, ¶¶118- 120. The medium comprises well-known memory types (e.g., RAM, ROM). ASUS-1005, ¶99; ASUS-1006, ¶119; ASUS-1003, Appx-C, ¶¶186-188.

Karczewicz-I and Karczewicz-II teach the processor executing computer program code in memory to carry out their teachings. ASUS-1005, ¶¶99-100; ASUS 1006, ¶48, ¶¶119-120. The apparatus further performs operations described below for [25b]-[25g]. ASUS-1003, Appx-C, ¶¶189-190.

[31a]. A computer program product for decoding a block of pixels, the computer program product comprising at least one non-transitory computer readable storage medium having computer executable program code portions stored therein, the computer executable program code portions comprising program code instructions configured to:

The Ground 1 combination teaches the preamble. ASUS-1003, Appx-C, ¶¶191-193.

As explained above, Karczewicz-I and Karczewicz-II implement their video-decoder teachings and perform operations for decoding blocks of pixels using

software stored in computer-readable storage media and executed by a processor. *Supra* [25a]; ASUS-1005, ¶12, ¶38, ¶¶98-100; ASUS-1006, ¶¶25-26, ¶48, ¶¶119-120. These types of media (e.g., RAM, ROM) are non-transitory. ASUS-1005, ¶99; ASUS-1006, ¶119. The software includes computer-executable program code portions that comprise program code instructions. ASUS-1005, ¶100; ASUS-1006, ¶26, ¶120; ASUS-1003, Appx-C, ¶192.

The software performs operations taught by the Ground 1 combination, including those described below for [31b]-[31g]. The combination further teaches forming "a computer program product" using the software media. ASUS-1005, ¶99; ASUS-1006, ¶119. Therefore, the Ground 1 combination teaches a computer program product as recited in [31a] that performs the operations described below for [31b]-[31g]. ASUS-1003, Appx-C, ¶193.

[19b]/[25b]/[31b] [determining/determine], for a current block, a first reference block based on a first motion vector and a second reference block based on a second motion vector, wherein the pixels of the current block, the first reference block, and the second reference block have values with a first precision;

The Ground 1 combination teaches [19b], [25b], and [31b]. ASUS-1003, Appx-C, ¶¶194-202.

Karczewicz-I and Karczewicz-II generate motion vectors, in the encoding process, that point to **reference blocks** (predictive video blocks) **for a current block** and "indicate the displacement" between the reference blocks and the

current block. ASUS-1005, ¶7, ¶¶53-54; ASUS-1006, ¶4, ¶56; ASUS-1003, Appx-C, ¶195.

The decoder performs the inverse process (ASUS-1005, ¶83), including motion compensation, to reconstruct the current block based on data received from the encoder. ASUS-1005, ¶89, ¶85, ¶88, Figs. 4, 6; ASUS-1003, Appx-C, ¶196.

As part of motion compensation, the decoder **determines reference blocks** (predictive blocks) **based on motion vectors** and reconstructs the target block based on the determined reference blocks. ASUS-1006, ¶6 ("…Given the motion vectors, the decoder is able to reconstruct the predictive blocks that were used to code the residual."), ¶63, Fig. 3; ASUS-1003, Appx-C, ¶197.

Karczewicz-I teaches video decoding using bi-directional prediction "based on two different lists of predictive reference data." ASUS-1005, ¶8, ¶22, ¶42, ¶¶5-6. The reference data from the two lists includes data in two reference blocks, and when the current block is a B block having two motion vectors, two reference blocks are determined based on these two motion vectors. ASUS-1005, ¶¶6-7, ¶¶53-54. Therefore, the combination teaches **determining, for a current block, a first reference block based on a first motion vector and a second reference block based on a second motion vector**. ASUS-1003, Appx-C, ¶¶198-199.

Karczewicz-II teaches **that the pixels of the current block, the first reference block, and the second reference block have values with a first**

precision (8 bits). "Precision" is satisfied by "a number of bits needed to represent possible values." *Supra* §II.D.1. Karczewicz-II teaches this precision (ASUS-1006, ¶89; *supra* §II.D.1), with 8 bits needed to represent possible pixel values for the two reference blocks. Table 5 shows that integer pixels (e.g., integer pixel x) include possible values between 0-255, and 8-bit unsigned numbers ("8u") are needed to represent these possible values. ASUS-1006, ¶103, Table 5.

TABLE 5

positions {a, c, d, l} of FIGS. 4A-4D

Operation	Comment	Min value	Max value	Register size
r1 = x	r1 is integer pixel x	0	255	8u
r1 = r1 << 5	r1 is 32 * x	0	8160	13u
r2 = y0	r2 is y0 (y0 is a one-dimensional (1-D) half-pixel such as b1, h1, ee1 and hh1 before shifting down)	−2550	10710	15s
r1 = r1 + r2	r1 is 32 * x + y0	−2550	18870	16s
r1 = r1 + 32	r1 is 32 * x + y0 + 32	−2518	18902	16s
r1 = r1 >> 6	r1 is (32 * x + y0 + 32) >> 6	−39	295	11s
r1 = max (0, r1)	clip r1 on the low side	0	295	10u
r1 = min (255, r1)	clip r1 on the high side	0	255	8u

ASUS-1006, Table 5

ASUS-1003, Appx-C, ¶¶200-201.

Therefore, the combination teaches or suggests that the pixels of the current block, the first reference block, and the second reference block have values with a first precision (8 bits). ASUS-1003, Appx-C, ¶202.

[19c]/[25c]/[31c] [using/use] said first reference block to obtain a first prediction, said first prediction having a second precision, which is higher than said first precision;

The Ground 1 combination teaches [19c], [25c], and [31c]. ASUS-1003, Appx-C, ¶¶203-217.

Karczewicz-I teaches calculating a bi-directional prediction using a default weighted prediction mode, with equal weights assigned to two reference blocks (ASUS-1005, ¶55, ¶24, ¶44, ¶48), calculated as an average of two predictions (pred0(i,j) and pred1(i,j)):

Bidirectional prediction: pred(i,j)=(pred0(i,j)+pred1(i,j)+1)>>1

ASUS-1005, ¶60; ASUS-1003, Appx-C, ¶¶204-205.

Here, pred0(i,j) is a first prediction based on a "motion compensated reference area[]… obtained from list 0… reference picture." ASUS-1005, ¶¶57-58. Likewise, pred1(i,j) is obtained from list 1. *Id*. The motion-compensated reference area refers to a reference block since Karczewicz-I teaches bi-directional prediction based on reference blocks. *Supra* [19b]/[25b]/[31b]; ASUS-1005, ¶7, ¶¶53-54. The combination of Karczewicz-I and Karczewicz-II does not change the reference

blocks on which Karczewicz-I's predictions are based. *Supra* §III.A.3 (explaining combination). Thus, the combination teaches **using said first reference block** (block from list 0) **to obtain a first prediction**. ASUS-1003, Appx-C, ¶¶206-207.

Karczewicz-I uses the reference block to obtain predictions in three scenarios (among others). *Supra* §III.A.3 (subsections explaining Scenarios 1-3). In each, the combination teaches said first prediction having a second precision higher than the first. ASUS-1003, Appx-C, ¶208.

Scenario 1. For bi-directional prediction averaging a half-pixel prediction and an integer pixel prediction, the combination of Karczewicz-II to Karczewicz-I uses a non-rounded half-pixel prediction (non-rounded(pred0(i,j))) and a left-shifted integer pixel prediction (pred1(i,j)<<5):

$$pred(i,j)=(non\text{-}rounded(pred0(i,j))+pred1(i,j)<<5+32)>>6$$

Supra §III.A.3 (subsection explaining "Scenario 1" calculations and combined teachings); ASUS-1003, Appx-C, ¶209.

Karczewicz-II teaches that the non-rounded half-pixel prediction (e.g., b1) has possible values from -2550 to 10710 and that a 15-bit signed number ("15s") is needed to represent these possible values. Karczewicz-II further teaches that the left-shifted integer pixel prediction (r1<<5) has possible values from 0 to 8160 and that a 13-bit unsigned number ("13u") is needed to represent these possible values. ASUS-1006, ¶103, Table 5.

TABLE 5

positions {a, c, d, l} of FIGS. 4A-4D

Operation	Comment	Min value	Max value	Register size
r1 = x	r1 is integer pixel x	0	255	8u
r1 = r1 << 5	r1 is 32 * x	0	8160	13u
r2 = y0	r2 is y0 (y0 is a one-dimensional (1-D) half-pixel such as b1, h1, ee1 and hh1 before shifting down)	−2550	10710	15s
r1 = r1 + r2	r1 is 32 * x + y0	−2550	18870	16s
r1 = r1 + 32	r1 is 32 * x + y0 + 32	−2518	18902	16s
r1 = r1 >> 6	r1 is (32 * x + y0 + 32) >> 6	−39	295	11s
r1 = max (0, r1)	clip r1 on the low side	0	295	10u
r1 = min (255, r1)	clip r1 on the high side	0	255	8u

ASUS-1006, Table 5

ASUS-1003, Appx-C, ¶210.

The first and second prediction each need at least 13 bits to represent their possible values, which is higher than the 8-bit precision for the pixels of the current block, the first reference block, and the second reference block. *Supra* [19b]/[25b]/[31b]. Thus, the combination teaches the first prediction having a second precision (13 bits), which is higher than the first (8 bits) under Scenario 1.

Supra §II.D.1; ASUS-1003, Appx-C, ¶211.

Scenario 2. For a center-pixel prediction and a half-pixel prediction, the combination of Karczewicz-II to Karczewicz-I uses a partially-rounded center-pixel prediction (non-rounded(pred0(i,j))>>5) and a non-rounded half-pixel prediction(non-rounded(pred1(i,j))):

pred(*i*,*j*)=(non-rounded(pred0(*i*,*j*))>>5+non-rounded(pred1(*i*,*j*)) +32)>>6

Supra §III.A.3 (subsection "Scenario 2" explaining calculations and combined teachings); ASUS-1003, Appx-C, ¶212.

Karczewicz-II teaches that the partially-rounded center-pixel prediction (j1>>1)[8] has possible values from -4957 to 13116, and a 15-bit signed integer prediction (e.g., b1) has possible values from -2550 to 10710, and a 15-bit signed ("15s") is needed to represent these possible values. The non-rounded half-pixel

[8] The equations in Table 3 of Karczewicz-II show that j1 is shifted to the right by 5 bits. Table 8 accomplishes this in two steps. The first step ("r2 = j1") is to slightly round the value of j1 to fit in a 16-bit register, which shaves off 4 bits from the right side (the least significant bits). ASUS-1006, ¶59, ¶10, ¶39, ¶53; *supra* §III.A.3. The second step ("r2 = r2 >> 1") right-shifts one more bit, bringing the total right-shift amount to 5 bits.

integer ("15s") is needed to represent these possible values. ASUS-1006, ¶105, Table 8.

TABLE 8

positions {f, i, k, n} of FIGS. 4A-4D

Operation	Comment	Min value	Max value	Register size
r1 = y0	r1 is y0 (1-D half-pixel such as b1, h1, ee1 and hh1 before shifting down)	−2550	10710	15s
r2 = j1	r2 is j1 (2-D half-pixel j1 before shifting down)	−9914	26232	16s
r2 = r2 >> 1	r2 is j1 >> 1	−4957	13116	15s
r1 = r1 + r2	r1 is y0 + (j1 >> 1)	−7507	23826	16s
r1 = r1 + 32	r1 is y0 + (j1 >> 1) + 32	−7491	23842	16s
r1 = r1 >> 6	r1 is (y0 + (j1 >> 1) + 32) >> 6	−235	745	11s
r1 = max (0, r1)	clip r1 on the low side	0	745	10u
r1 = min (255, r1)	clip r1 on the high side	0	255	8u

ASUS-1006, Table 8

ASUS-1003, Appx-C, ¶213.

Therefore, for Scenario 2, the first and second predictions each have a second precision of 15 bits, which is higher than the first precision (8 bits) for, e.g., pixels of the current block. *Supra* [19b]/[25b]/[31b]; §II.D.1; ASUS-1003, Appx-C, ¶214.

Scenario 3. For two half-pixel predictions, the combination of Karczewicz-II to Karczewicz-I uses non-rounded half-pixel predictions:

pred(*i*,*j*)=(non-rounded(pred0(*i*,*j*))+non-rounded(pred1(*i*,*j*)) +32)>>6

Supra §III.A.3 (subsection "Scenario 3" explaining calculations and combined teachings); ASUS-1003, Appx-C, ¶215.

Karczewicz-II teaches that the non-rounded half-pixel predictions (e.g., b1, h1) each has values from -2550 to 10710, and a 15-bit signed number ("15s") is needed to represent these possible values. ASUS-1006, ¶103, Table 6.

TABLE 6

positions {e, g, m, o} of FIGS. 4A-4D

Operation	Comment	Min value	Max value	Register size
r1 = y0	r1 is y0 (y0 is a 1-D half-pixel such as b1, h1, ee1 and hh1 before shifting down)	−2550	10710	15s
r2 = y1	r2 is y1 (y1 is a 1-D half-pixel such as b1, h1, ee1 and hh1 before shifting down)	−2550	10710	15s
r1 = r1 + r2	r1 is y0 + y1	−5100	21420	16s
r1 = r1 + 32	r1 is y0 + y1 + 32	−5068	21452	16s
r1 = r1 >> 6	r1 is (y0 + y1 + 32) >> 6	−79	335	11s
r1 = max (0, r1)	clip r1 on the low side	0	335	10u
r1 = min (255, r1)	clip r1 on the high side	0	255	8u

ASUS-1006, Table 6

ASUS-1003, Appx-C, ¶216.

Thus, for Scenario 3, the first and second predictions each have a second precision of 15 bits, which is higher than the first precision (8 bits) for, e.g., pixels of the current block. *Supra* [19b]/[25b]/[31b], §II.D.1; ASUS-1003, Appx-C, ¶217.

[19d]/[25d]/[31d] [using/use] said second reference block to obtain a second prediction, said second prediction having the second precision;

The Ground 1 combination teaches [19d], [25d], and [31d]. ASUS-1003, Appx-C, ¶¶218-224.

As explained above, Karczewicz-I teaches calculating a bi-directional prediction as an average of two predictions, and pred1(i,j) is a second prediction obtained based on a reference block. *Supra* [19c]/[25c]/[31c]; ASUS-1005, ¶¶57-58, ¶60. The combination of Karczewicz-I and Karczewicz-II does not change the reference blocks on which Karczewicz-I's predictions are based. *Supra* §III.A.3 (explaining combination). Thus, the combination teaches **using said second reference block** (block from list 1) **to obtain a second prediction** in three scenarios. *Supra* §III.A.3 (subsections "Scenarios 1-3" explaining prediction teachings); ASUS-1003, Appx-C, ¶¶219-220.

As explained above, in Scenario 1, the combination teaches **said second prediction** (left-shifted integer pixel prediction pred1(i,j)<<5) **having the second precision** (13 bits). *Supra* [19c]/[25c]/[31c]; ASUS-1003, Appx-C, ¶222.

In Scenario 2, the combination teaches **said second prediction** (non-rounded half-pixel prediction non-rounded(pred1(i,j))) **having the second precision** (15 bits). *Supra* [19c]/[25c]/[31c]; ASUS-1003, Appx-C, ¶223.

In Scenario 3, the combination teaches **said second prediction** (non-

rounded half-pixel prediction non-rounded(pred1(i,j))) **having the second precision** (15 bits). *Supra* [19c]/[25c]/[31c]; ASUS-1003, Appx-C, ¶224.

[19e]/[25e]/[31e] [obtaining/obtain] a combined prediction based at least partly upon said first prediction and said second prediction;

The Ground 1 combination teaches [19e], [25e], and [31e]. ASUS-1003, Appx-C, ¶¶225-231.

Karczewicz-I teaches obtaining a bi-directional prediction by averaging the first and second predictions:

Bidirectional prediction: pred(*i*,*j*)=(pred0(*i*,*j*)+pred1(*i*,*j*)+1)>>1

ASUS-1005, ¶60 (adding predictions and dividing by two with ">>1"); *supra* [19c]-[19d]/[25c]-[25d]/[31c]-[31d]. The Ground 1 combination combines these bi-prediction teachings with Karczewicz-II. *Supra* §III.A.3 (explaining MTC); ASUS-1003, Appx-C, ¶¶226-227.

Under each of the three scenarios discussed above (*supra* §III.A.3), the combination teaches **obtaining a combined prediction** (a sum of the first prediction, the second prediction, and a rounding offset) **based at least partly upon said first prediction and said second prediction.** ASUS-1003, Appx-C, ¶228.

For Scenario 1, it was obvious to calculate the bi-directional prediction based on a non-rounded half-pixel prediction and a left-shifted integer pixel

prediction:

$$pred(i,j)=(non\text{-}rounded(pred0(i,j))+pred1(i,j)<<5+32)>>6$$

Supra §III.A.3 (subsection "Scenario 1" explaining calculations). The combination calculates a sum of the first prediction (e.g., non-rounded(pred0(i,j))), the second prediction (e.g., pred1(i,j)<<5), and a rounding offset (32). *See* ASUS-1006, ¶103, Table 5; ASUS-1003, Appx-C, ¶229.

For Scenario 2, it was obvious to calculate the bi-directional prediction based on a partially-rounded center-pixel prediction and a non-rounded half-pixel prediction:

$$pred(i,j)=(non\text{-}rounded(pred0(i,j))>>5+non\text{-}rounded(pred1(i,j))+32)>>6$$

Supra §III.A.3 (subsection "Scenario 2" explaining calculations). The combination calculates a sum of the first prediction (e.g., non-rounded(pred0(i,j))>>5), the second prediction (e.g., non-rounded(pred1(i,j))), and a rounding offset (32). *See* ASUS-1006, ¶105, Table 8; ASUS-1003, Appx-C, ¶230.

For Scenario 3, it was obvious to calculate the bi-directional prediction based on two non-rounded half-pixel predictions:

$$pred(i,j)=(non\text{-}rounded(pred0(i,j))+non\text{-}rounded(pred1(i,j))+32)>>6$$

Supra §III.A.3 (subsection "Scenario 3" explaining calculations). The combination calculates a sum of the first prediction (e.g., non-rounded(pred0(i,j))), the second prediction (e.g., non-rounded(pred1(i,j))), and a rounding offset (32). *See* ASUS-

1006, ¶103, Table 6; ASUS-1003, Appx-C, ¶231.

[19f]/[25f]/[31f] [decreasing/decrease] a precision of said combined prediction by shifting bits of the combined prediction to the right; and

The Ground 1 combination teaches [19f], [25f], and [31f]. ASUS-1003, Appx-C, ¶¶232-239.

As explained above, the Ground 1 combination teaches the combined prediction as a sum of the first prediction, the second prediction, and a rounding offset. *Supra* [19e]/[25e]/[31e]. Karczewicz-I and Karczewicz-II teach the operation ">>" for shifting bits to the right. ASUS-1005, ¶57 (">> is a right shift operation[.]"), ¶55, ¶60; ASUS-1006, ¶94. The combination teaches **decreasing a precision of said combined prediction by shifting bits of the combined prediction to the right** for each of the three scenarios discussed above. ASUS-1003, Appx-C, ¶233.

Scenario 1. Here, the combination calculates bi-directional prediction based on a non-rounded half-pixel prediction and a left-shifted integer pixel prediction:

$$pred(i,j)=(non\text{-}rounded(pred0(i,j))+pred1(i,j)<<5+32)>>6$$

Supra §III.A.3 (subsection "Scenario 1" explaining calculations). Bits of the combined prediction (non-rounded(pred0(i,j))+pred1(i,j)<<5+32) are shifted to the right by 6 bits (">>6"). ASUS-1003, Appx-C, ¶234.

The shifting decreases a precision of said combined prediction. ASUS-1006,

¶103, Table 5.

TABLE 5

positions {a, c, d, l} of FIGS. 4A-4D

Operation	Comment	Min value	Max value	Register size
r1 = x	r1 is integer pixel x	0	255	8u
r1 = r1 << 5	r1 is 32 * x	0	8160	13u
r2 = y0	r2 is y0 (y0 is a one-dimensional (1-D) half-pixel such as b1, h1, ee1 and hh1 before shifting down)	−2550	10710	15s
r1 = r1 + r2	r1 is 32 * x + y0	−2550	18870	16s
r1 = r1 + 32	r1 is 32 * x + y0 + 32	−2518	18902	16s
r1 = r1 >> 6	r1 is (32 * x + y0 + 32) >> 6	−39	295	11s
r1 = max (0, r1)	clip r1 on the low side	0	295	10u
r1 = min (255, r1)	clip r1 on the high side	0	255	8u

ASUS-1006, Table 5

In this table, the operation “r1=r1+32” calculates the sum of a rounding offset (32) with r1, which in turn is the sum of a non-rounded half-pixel prediction (y0 which can take the value of half-pixel prediction b1) and an integer pixel prediction that was left-shifted by 5 bits (32*x). Karczewicz-II teaches that this sum has possible values between -2518 and 18902, and that a 16-bit signed number (“16s”) is

needed to represent these possible values. Next, the operation "r1=r1>>6" shifts the sum 6 bits to the right. The result has possible values between -39 and 295; an 11-bit signed number ("11s") is needed to represent these possible values. Thus, the combination decreases a precision of said combined prediction from 16 to 11 bits. *Supra* §II.D.1; ASUS-1003, Appx-C, ¶235.

Scenario 2. Here, the combination calculates bi-directional prediction based on a partially-rounded center-pixel prediction and a non-rounded half-pixel prediction:

pred(*i*,*j*)=(non-rounded(pred0(*i*,*j*))>>5+non-rounded(pred1(*i*,*j*)) +32)>>6

Supra §III.A.3 (Subsection "Scenario 2" explaining calculations). Bits of the combined prediction (non-rounded(pred0(i,j))>>5+non-rounded(pred1(i,j)) +32) are shifted to the right (">>6"). ASUS-1003, Appx-C, ¶236.

The shifting decreases a precision of said combined prediction. ASUS-1006, ¶105, Table 8.

TABLE 8

positions {f, i, k, n} of FIGS. 4A-4D

Operation	Comment	Min value	Max value	Register size
r1 = y0	r1 is y0 (1-D half-pixel such as b1, h1, ee1 and hh1 before shifting down)	−2550	10710	15s
r2 = j1	r2 is j1 (2-D half-pixel j1 before shifting down)	−9914	26232	16s
r2 = r2 >> 1	r2 is j1 >> 1	−4957	13116	15s
r1 = r1 + r2	r1 is y0 + (j1 >> 1)	−7507	23826	16s
r1 = r1 + 32	r1 is y0 + (j1 >> 1) + 32	−7491	23842	16s
r1 = r1 >> 6	r1 is (y0 + (j1 >> 1) + 32) >> 6	−235	745	11s
r1 = max (0, r1)	clip r1 on the low side	0	745	10u
r1 = min (255, r1)	clip r1 on the high side	0	255	8u

ASUS-1006, Table 8

The operation “r1=r1+32” calculates the sum of a partially-rounded center-pixel prediction (“j1>>1”), a non-rounded half-pixel prediction (y0, which may take the value of b1), and a rounding offset (32). Karczewicz-II teaches that this sum has possible values between -7491 and 23842 and that a 16-bit signed number (“16s”) is needed to represent these possible values. Next, the operation “r1=r1>>6” shifts the sum 6 bits to the right. The result has possible values between -235 and 745; an 11-bit signed number (“11s”) is needed to represent these possible values. Thus, the combination decreases a precision of said combined prediction from 16 to 11 bits. *Supra* §II.D.1; ASUS-1003, Appx-C, ¶237.

Scenario 3. Here, the combination calculates bi-directional prediction based on two non-rounded half-pixel predictions:

pred(*i*,*j*)=(non-rounded(pred0(*i*,*j*))+non-rounded(pred1(*i*,*j*)) +32)>>6

Supra §III.A.3 (subsection "Scenario 3" explaining calculations). Bits of the combined prediction (non-rounded(pred0(i,j))+non-rounded(pred1(i,j)) +32) are shifted to the right (">>6"). ASUS-1003, Appx-C, ¶238.

The shifting decreases a precision of said combined prediction. ASUS-1006, ¶103, Table 6.

TABLE 6

positions {e, g, m, o} of FIGS. 4A-4D

Operation	Comment	Min value	Max value	Register size
r1 = y0	r1 is y0 (y0 is a 1-D half-pixel such as b1, h1, ee1 and hh1 before shifting down)	−2550	10710	15s
r2 = y1	r2 is y1 (y1 is a 1-D half-pixel such as b1, h1, ee1 and hh1 before shifting down)	−2550	10710	15s
r1 = r1 + r2	r1 is y0 + y1	−5100	21420	16s
r1 = r1 + 32	r1 is y0 + y1 + 32	−5068	21452	16s
r1 = r1 >> 6	r1 is (y0 + y1 + 32) >> 6	−79	335	11s
r1 = max (0, r1)	clip r1 on the low side	0	335	10u
r1 = min (255, r1)	clip r1 on the high side	0	255	8u

ASUS-1006, Table 6

The operation "r1=r1+32" calculates the sum of two non-rounded half-pixel predictions (y0 and y1, which may take the values of b1 and h1), and a rounding

offset (32). Karczewicz-II teaches that this sum has possible values from -5068 to 21452, and a 16-bit signed number ("16s") is needed to represent these possible values. Next, the operation "r1=r1>>6" shifts the sum 6 bits to the right. The result has possible values from -79 to 335; an 11-bit signed number ("11s") is needed to represent these possible values. Thus, the combination decreases a precision of said combined prediction from 16 to 11 bits. *Supra* §II.D.1; ASUS-1003, Appx-C, ¶239.

[19g]/[25g]/[31g] [reconstructing/reconstruct] the block of pixels based on the combined prediction.

The Ground 1 combination teaches [19g], [25g], and [31g]. ASUS-1003, Appx-C, ¶¶240-241.

The combination teaches determining the combined prediction in the motion compensation process. *Supra* [19b]-[19f]/[25b]-[25f]/[31b]-[31f]. Karczewicz-I teaches **reconstructing the block of pixels** (reconstructed video block) **based on the combined prediction** (prediction block). ASUS-1005, ¶86 ("Adder 79 combines the prediction data (e.g., a prediction block) generated by prediction unit 75 with the residual block from inverse transform unit 78 to create a reconstructed video block[.]"), ¶89, Figs. 4, 6. Karczewicz-II includes similar teachings. ASUS-1006, ¶6, ¶65, ¶111, Figs. 3, 6; ASUS-1003, Appx-C, ¶241.

(h) Dependent Claims 20, 26, and 32

20. The method according to claim 19, …
26. The apparatus according to claim 25, …
32. The computer program product according to claim 31,
wherein in an instance in which said first motion vector points to a subpixel, said first prediction is obtained by interpolation using pixel values of said first reference block.

The Ground 1 combination teaches claims 19, 25, and 31 and the limitations of claims 20, 26, and 32. *Supra* §III.A.3, §III.A.4(g); ASUS-1003, Appx-C, ¶¶242-250.

Karczewicz-II teaches obtaining predictions by interpolation using pixel values of reference blocks when motion vectors point to subpixels. ASUS-1006, ¶42 ("The interpolation techniques of this disclosure may be performed by any encoding device that supports motion compensated interpolation to sub-pixel resolution."), ¶7, ¶66, ¶¶68-72, Figs. 4A-4D; ASUS-1003, Appx-C, ¶243.

The Ground 1 combination satisfies this claim limitation for any of the three scenarios discussed above. ASUS-1003, Appx-C, ¶244.

Scenario 1 is **an instance in which said first motion vector points to a subpixel** (a half-pixel position). *Supra* §III.A.3 (subsection "Scenario 1" explaining interpolation teachings). The half-pixel position refers to a subpixel. ASUS-1006, ¶74, Figs. 4A-4D, ¶10; ASUS-1003, Appx-C, ¶245.

Karczewicz-II teaches **obtaining the first prediction** (the non-rounded half-

pixel prediction) **by interpolation using pixel values of said first reference block**. ASUS-1006, ¶93 (H.264 defines one interpolation process for sub-pixels…). The interpolation is performed using pixel values of "six closest integer pixels that surround" the subpixel, which are located in the reference block that the first motion vector points to. *See* ASUS-1006, Fig. 4B.

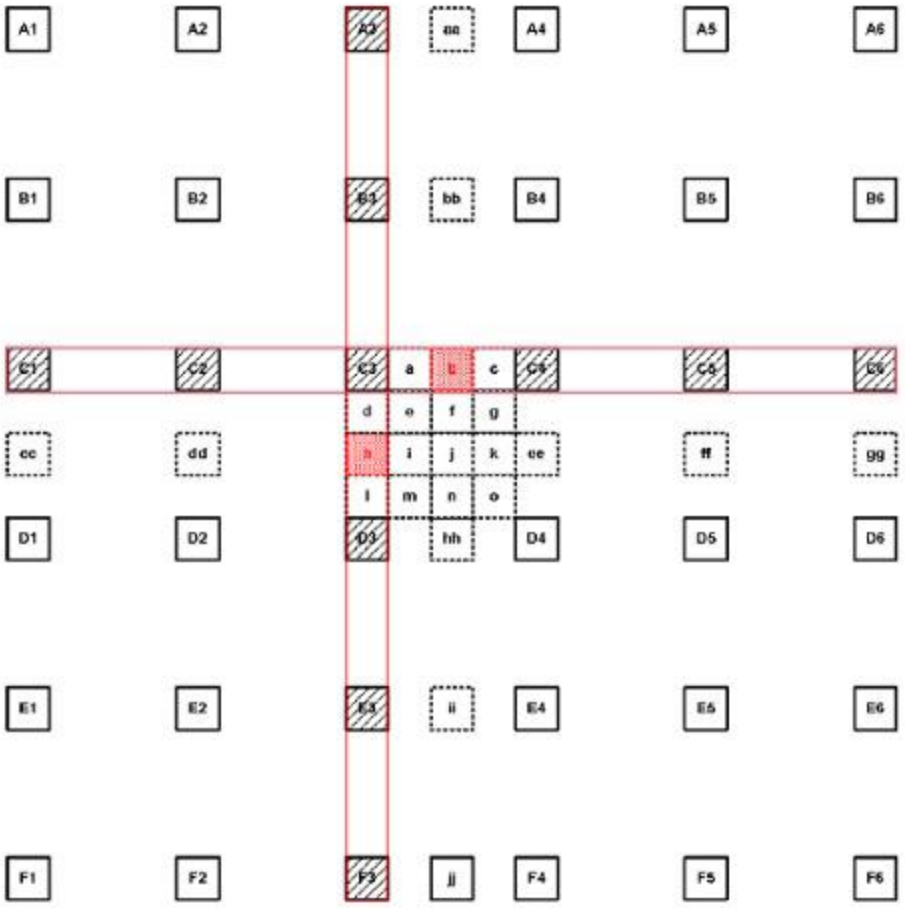

FIG. 4B

ASUS-1006, Fig. 4B

Thus, the interpolation is performed using pixel values of said first reference block. ASUS-1003, Appx-C, ¶246.

Scenario 2 is **an instance in which said first motion vector points to a subpixel** (a center-pixel position). *Supra* §III.A.3 (subsection "Scenario 2" explaining interpolation teachings). The center-pixel position refers to a subpixel. ASUS-1006, ¶74, Figs. 4A-4D, ¶10; ASUS-1003, Appx-C, ¶247.

Karczewicz-II teaches **obtaining the first prediction** (the partially-rounded center-pixel prediction) **by interpolation using pixel values of said first reference block**. ASUS-1006, ¶95 ("To interpolate sub-pixel 'j,' …"). The interpolation is performed using non-rounded half-pixel predictions, which are located in the reference block that the first motion vector points to. *See* ASUS-1006, Fig. 4C.

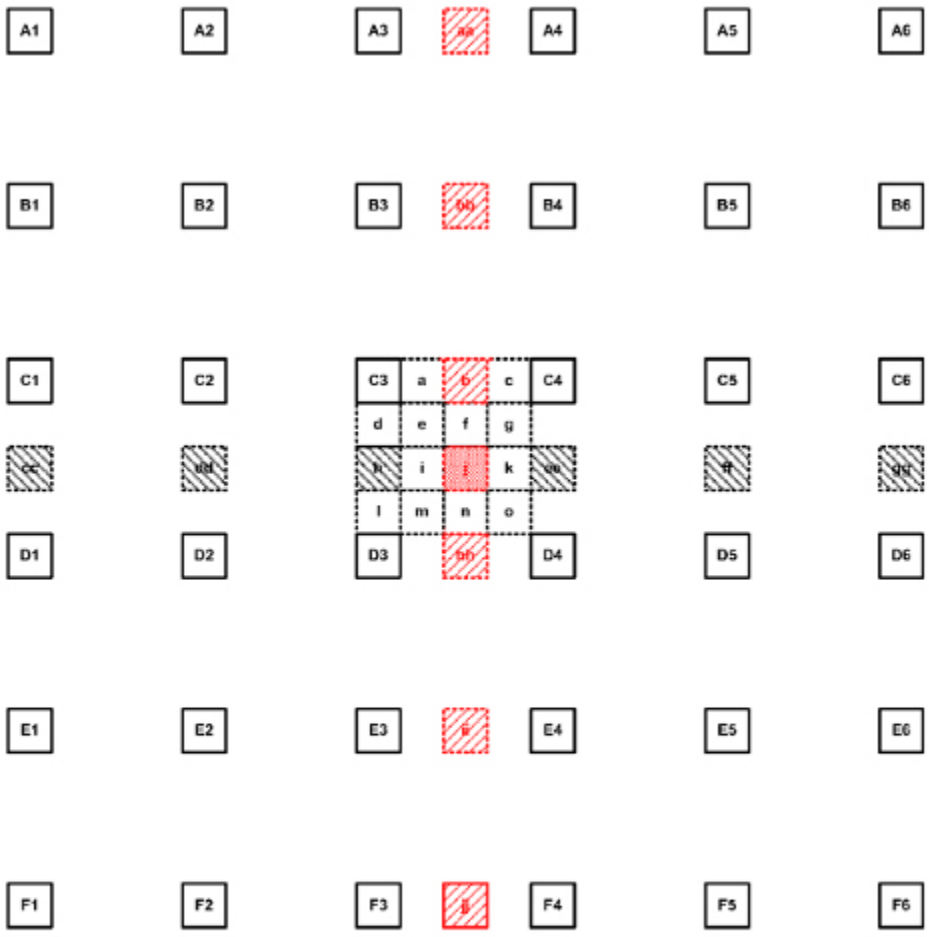

FIG. 4C

ASUS-1006, Fig. 4C

Thus, the interpolation is performed using pixel values of said first reference block. ASUS-1003, Appx-C, ¶248.

Scenario 3. Here, the first motion vector points to a half-pixel position. *Supra* §III.A.3 (subsection "Scenario 3" explaining interpolation teachings). For the same reasons as explained for Scenario 1, the combination teaches that, in an

instance in which said first motion vector points to a subpixel, said first prediction is obtained by interpolation using pixel values of said first reference block. ASUS-1003, Appx-C, ¶249.

Additionally, the ’267 patent confirms that the limitations of claims 20, 26, and 32 were known in the prior art by describing it in the “Background Information” section. ASUS-1001, 2:60-3:11; ASUS-1003, Appx-C, ¶250.

(i) Dependent Claims 21, 27, and 33

21. The method according to claim 20, …
27. The apparatus according to claim 26, …
33. The computer program product according to claim 32,
wherein said first prediction is obtained by interpolation using values of said first reference block by: right shifting a sum of a P-tap filter using values of said first reference block.

The Ground 1 combination teaches claims 20, 26, and 32, and the limitations of claims 21, 27, and 33 under Scenario 2. *Supra* §III.A.4(h); ASUS-1003, Appx-C, ¶¶251-255.

In Scenario 2, the combination teaches the following equation for bi-directional prediction:

pred(*i*,*j*)=(non-rounded(pred0(*i*,*j*))>>5+non-rounded(pred1(*i*,*j*))+32)>>6

Supra §III.A.3 (subsection “Scenario 2” explaining prediction teachings). The firstprediction “non-rounded(pred0(i,j))>>5” is a partially-rounded center-pixel prediction. *Id.*; *supra* [19c]/[25c]/[31c]; ASUS-1003, Appx-C, ¶252.

Karczewicz-II teaches calculating the partially-rounded center-pixel

prediction (j1>>5) by first determining a sum of a 6-tap filter (j1):

$$j1=aa1-5*bb1+20*b1+20*hh1-5*ii1+jj1$$

ASUS-1006, ¶95; *supra* §III.A.4(h). This sum is a non-rounded version of the center-pixel prediction. ASUS-1003, Appx-C, ¶253.

Next, the sum is right-shifted 5 bits to obtain the partially-rounded center-pixel prediction (j1>>5). *Supra* §III.A.3; ASUS-1006, ¶99, ¶105, Tables 8, 3.

TABLE 3

a = (C3 << 5 + b1 + 32) >> 6
c = (C4 << 5 + b1 + 32) >> 6
d = (C3 << 5 + h1 + 32) >> 6
l = (D3 << 5 + h1 + 32) >> 6
f = (j1 >> 5 + b1 + 32) >> 6
i = (j1 >> 5 + h1 + 32) >> 6
k = (j1 >> 5 + ee1 + 32) >> 6
n = (j1 >> 5 + hh1 + 32) >> 6

ASUS-1006, Table 3

ASUS-1003, Appx-C, ¶253.

Therefore, the combination teaches that **said first prediction** (partially rounded center-pixel prediction) **is obtained by interpolation using values of said first reference block by: right shifting** (“>>5”) **a sum of a P-tap filter** (non-rounded center-pixel prediction “non-rounded(pred0(i,j))”) **using values of said first reference block**. ASUS-1003, Appx-C, ¶254.

(j) Dependent Claims 22, 28, and 34

22. The method according to claim 20, …
28. The apparatus according to claim 26, …
34. The computer program product according to claim 32,
wherein in an instance in which said second motion vector points to an integer sample, said second prediction is obtained by shifting values of said second reference block to the left.

The Ground 1 combination teaches claims 20, 26, and 32, and the limitations of claims 22, 28, and 34 under Scenario 1. *Supra* §III.A.4(h); ASUS-1003, Appx-C, ¶¶256-261.

Scenario 1 is **an instance in which said second motion vector points to an integer sample**. *Supra* §III.A.3 (subsection "Scenario 1" explaining motion vector pointing to integer pixel position). ASUS-1003, Appx-C, ¶257.

The combination calculates bi-directional prediction:

$$pred(i,j)=(non\text{-}rounded(pred0(i,j))+pred1(i,j)<<5+32)>>6$$

Supra §III.A.3 (Subsection "Scenario 1"). The second prediction "pred1(i,j)<<5" is a left-shifted integer pixel prediction. *Supra* §III.A.3, [19c]-[19d]/[25c]-[25d]/[31c]-[31d]; ASUS-1003, Appx-C, ¶258.

Karczewicz-II teaches calculating the left-shifted integer pixel prediction by shifting the pixel value to the left (e.g., C3<<5). *Supra* §III.A.3; ASUS-1006, ¶99, Table 3.

TABLE 3

a = (C3 << 5 + b1 + 32) >> 6
c = (C4 << 5 + b1 + 32) >> 6
d = (C3 << 5 + h1 + 32) >> 6
l = (D3 << 5 + h1 + 32) >> 6
f = (j1 >> 5 + b1 + 32) >> 6
i = (j1 >> 5 + h1 + 32) >> 6
k = (j1 >> 5 + ee1 + 32) >> 6
n = (j1 >> 5 + hh1 + 32) >> 6

ASUS-1006, Table 3

The left-shifted value is of an integer pixel sample (e.g., C3) that is part of the second reference block. ASUS-1006, ¶74 ("integer-pixel sample 'C3'"), ¶93; ASUS-1003, Appx-C, ¶259.

Karczewicz-I and Karczewicz-II teach that motion compensation is performed on a block basis. *Supra* [19b], [19g]/[25b], [25g]/[31b], [31g]; Ex-1005, ¶7; ASUS-1006, ¶4, ¶73. Thus, the left-shift operation is performed for the pixels of the current block based on corresponding pixels of the reference block. *Supra* [19b]/[25b]/[31b]; ASUS-1005, ¶7, ¶¶53-54; ASUS-1006, ¶4, ¶56; ASUS-1003, Appx-C, ¶260.

Therefore, the combination teaches that **in an instance in which said second motion vector points to an integer sample, said second prediction** (left-shifted integer pixel prediction) **is obtained by shifting values of said second reference block** (values of integer pixels) **to the left** ("<<5"). ASUS-1003, Appx-C, ¶261.

(k) Dependent Claims 23, 29, and 35

23. The method according to claim 19, wherein said decreasing said precision of said combined prediction by shifting bits of the combined prediction to the right, further comprises: …
29. The apparatus according to claim 25, wherein the at least one memory and computer code are configured to cause the apparatus to decrease said precision of said combined prediction by shifting bits of the combined prediction to the right, by: …
35. The computer program product according to claim 31, wherein the program code instructions configured to decrease said precision of said combined prediction by shifting bits of the combined prediction to the right, further comprise program code instructions configured to: [inserting/insert] a rounding offset to the combined prediction before said decreasing.

The Ground 1 combination teaches claims 19, 25, and 31, and the limitations of claims 23, 29, and 35. *Supra* §III.A.4(g); ASUS-1003, Appx-C, ¶¶262-267.

As explained above, the combination teaches obtaining a combined prediction by summing the first prediction, the second prediction, and the rounding offset (e.g., 32) and decreasing a precision of the combined prediction. *Supra* [19e]-[19f]/[25e]-[25f]/[31e]-[31f]. The combination thus teaches **inserting a rounding offset** (32) **to the combined prediction**, **before said decreasing** of the precision. ASUS-1003, Appx-C, ¶263.

The value added to the sum of the first and second predictions is a rounding offset. Karczewicz-I refers to the term that is added to the weighted sum of the first

and second predictions as a "rounding adjustment." ASUS-1005, ¶63, ¶55.[9] Because "rounding adjustment" was used interchangeably with "rounding offset" and the value (32) is inserted to the combined prediction right before the rounding operation, this value is a rounding offset. ASUS-1003, Appx-C, ¶264.

Karczewicz-I and Karczewicz-II teach this rounding offset as part of its rounding process, which decreases the precision as recited by the claims. ASUS-1005, ¶55, ¶63; ASUS-1006, ¶¶96-101, Tables 1-4. Additionally, it was obvious for said decreasing to include the rounding offset because the insertion of the rounding offset is performed immediately before the right-shifting to affect the direction of the rounding. The combination includes a rounding offset to control rounding error resulting from the right-shift operation that decreases precision. This was common in the art. ASUS-1003, Appx-C, §I.D, ¶265.

Because the combination applies its operations to a computer implementation, it teaches 1) at least one memory and computer program code being configured to cause the apparatus to perform the operations of claim 29

[9] Karczewicz-I teaches a rounding adjustment of 2^{r-1} prior to right-shifting r bits, consistent with the Ground 1 combination's modified equation for calculating bi-directional prediction under each scenario (*supra* §III.A.3), which teaches right-shifting 6 bits and a rounding offset of $2^{6-1}=2^5=32$.

(*supra* [25a] and 2) and program code instructions configured to perform the operations of claim 35 (*supra* [31a]). ASUS-1003, Appx-C, ¶¶266-267.

(l) Dependent Claims 24, 30, and 36

24. The method according to claim 19, …
30. The apparatus according to claim 25, …
36. The computer program product according to claim 31,
wherein the first precision indicates a number of bits needed to represent the values of the pixels, and the second precision indicates the number of bits needed to represent values of said first prediction and values of said second prediction.

The Ground 1 combination teaches claims 19, 25, and 31 and the limitations of claims 24, 30, and 36. *Supra* §III.A.4(g); ASUS-1003, Appx-C, ¶¶268-270.

The first precision indicates a number of bits needed to represent the values of the pixels. The combination teaches that the pixels of the current block, the first reference block, and the second reference block have values with a first precision because 8 bits are needed to represent the possible pixel values of these blocks. *Supra* [19b]/[25b]/[31b]. ASUS-1003, Appx-C, ¶269.

The second precision indicates the number of bits needed to represent values of said first prediction and values of said second prediction. The combination teaches said first prediction and second prediction having a second precision that is higher than said first precision because more bits are needed to represent the possible values of the predictions under each of Scenarios 1, 2, and 3. *Supra* [19c]-[19d]/[25c]-[25d]/[31c]-[31d]; ASUS-1003, Appx-C, ¶270.

IV. PTAB DISCRETION SHOULD NOT PRECLUDE INSTITUTION

Petitioner believes that discretionary denial is unwarranted, and yet, Petitioner intends to utilize the bifurcated briefing process contemplated by the March 26, 2025, Stewart Memorandum to rebut contentions if offered by Patent Owner to the contrary. ASUS-1031; ASUS-1032.

VI. PAYMENT OF FEES – 37 C.F.R. § 42.103

Petitioner authorizes charge of fees to Deposit Account 06-1050.

VII. CONCLUSION

The Challenged Claims are unpatentable.

VIII. MANDATORY NOTICES UNDER 37 C.F.R. § 42.8(a)(1)

A. Real Party-In-Interest Under 37 C.F.R. § 42.8(b)(1)

The real parties-in-interest here are ASUSTeK Computer Inc. and ASUS Computer International. No other parties directed, controlled, or funded this *Inter Partes* Review proceeding (IPR).

B. Related Matters Under 37 C.F.R. § 42.8(b)(2)

Petitioner is not aware of any disclaimers or reexamination certificates for the ’267 patent. The ’267 patent was the subject of *inter partes* reviews in IPR2024-00626 and IPR2024-00627, both of which were terminated due to settlement. The ’267 patent is the subject of a number of civil actions including:

Certain Video-Capable Laptop, Desktop Computers, Handheld Computers,

Tablets, Televisions, Projectors, and Components and Modules Thereof; Inv. No. 337-TA-1448 (Violation), 337-TA-1448 (ITC), filed on April. 11, 2025;

Nokia Technologies Oy v. *ASUSeK Computer Inc. et al.,* 2-25-cv-03053 (CDCA), filed on April 7, 2025;

Nokia Technologies Oy v. *Hisense Co. Ltd. et al.*, 1-25-cv-01871 (NDGA), filed on April 7, 2025;

Nokia Technologies Oy v. *Acer Inc. et al.*, 1-25-cv-00523 (WDTX), filed on April 7, 2025;

Element Television Company, LLC et al. v. *Nokia Corporation a/k/a Nokia of America Corporation et al.*, 0-24-cv-04269 (DMN), Nov. 25, 2024;

Amazon.com, Inc. et al v. *Nokia Technologies Oy,* IPR2024-00626 (PTAB), filed on March 15, 2024;

Amazon.com, Inc. et al. v. *Nokia Technologies Oy,* IPR2024-00627 (PTAB), filed on March 15, 2024;

Nokia Technologies Oy v. *Amazon.com, Inc. et al.*, 1-23-cv-01236 (DDE), filed on October 31, 2023;

Nokia Technologies Oy v. *HP Inc. f/k/a Hewlett-Packard Company,* 1-23-cv-01237 (DDE), filed on October 31, 2023;

Certain Video Capable Electronic Devices, Including Computers, Streaming Devices, Televisions, and Components and Modules Thereof; Inv. No. 337-TA-

1380 (Violation), 337-TA-1380 (ITC), filed on October 31, 2023;

Nokia Corporation et al v. *Amazon.com, Inc.*,1-23-cv-01232 (DDE), filed on October 27, 2023.

C. Lead And Back-Up Counsel Under 37 C.F.R. § 42.8(b)(3)

Petitioner provides the following designation of counsel.

Lead Counsel	Backup Counsel
Jeremy J. Monaldo, Reg. No. 58,680 Fish & Richardson P.C. 60 South Sixth Street, Suite 3200 Minneapolis, MN 55402 Tel: 202-783-5070 Fax: 877-769-7945 Email: IPR54587-0013IP1@fr.com	Kim Leung, Reg. No. 64,399 Joseph V. Colaianni, Reg. No. 39,948 Linhong Zhang, Reg. No. 64,749 Jack R. Wilson IV, Reg. No. 75,011 Fish & Richardson P.C. 60 South Sixth Street, Suite 3200 Minneapolis, MN 55402 Tel: 202-783-5070 Fax: 877-769-7945

D. Service Information

Please address all correspondence and service to the address listed above. Petitioner consents to electronic service by email at IPR54587-0013IP1@fr.com.

Respectfully submitted,

Dated June 17, 2025

/Jeremy J. Monaldo/

Jeremy J. Monaldo, Reg. No. 58,680
Kim Leung, Reg. No. 64,399
Joseph V. Colaianni, Reg. No. 39,948
Linhong Zhang, Reg. No. 64,749
Jack R. Wilson IV, Reg. No. 75,011
Fish & Richardson P.C.
60 South Sixth Street, Suite 3200
Minneapolis, MN 55402
T: 202-783-5070
F: 877-769-7945

(Control No. IPR2025-01154)

Attorneys for Petitioner

CERTIFICATION UNDER 37 CFR § 42.24

Under the provisions of 37 CFR § 42.24(d), the undersigned hereby certifies that the word count for the foregoing Petition for *Inter partes* Review totals 13,440 words, which is less than the 14,000 allowed under 37 CFR § 42.24.

Dated June 17, 2025

/Jeremy J. Monaldo/

Jeremy J. Monaldo, Reg. No. 58,680
Kim Leung, Reg. No. 64,399
Joseph V. Colaianni, Reg. No. 39,948
Linhong Zhang, Reg. No. 64,749
Jack R. Wilson IV, Reg. No. 75,011
Fish & Richardson P.C.
60 South Sixth Street, Suite 3200
Minneapolis, MN 55402
T: 202-783-5070
F: 877-769-7945

Attorneys for Petitioner

CERTIFICATE OF SERVICE

Pursuant to 37 CFR §§ 42.6(e)(4)(i) *et seq.* and 42.105(b), the undersigned certifies that on June 17, 2025, a complete and entire copy of this Petition for *Inter partes* Review, Power of Attorney, and all supporting exhibits were provided via Federal Express, to the Patent Owner by serving the correspondence address of record as follows:

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

/Hoi Cheung/
Hoi Cheung
Fish & Richardson P.C.
60 South Sixth Street, Suite 3200
Minneapolis, MN 55402
hcheung@fr.com