CHRIS R. OTTENWELLER (STATE BAR NO. 73649)
cottenweller@orrick.com
G. HOPKINS GUY, III (STATE BAR NO. 124811)
hopguy@orrick.com
VICKIE L. FEEMAN (STATE BAR NO. 177487)
vfeeman@orrick.com
BAS DE BLANK (STATE BAR NO. 191487)
basdeblank@orrick.com
ORRICK, HERRINGTON & SUTCLIFFE LLP
1000 Marsh Road
Menlo Park, CA 94025
Telephone: +1-650-614-7400
Facsimile: +1-650-614-7401

Attorneys for Plaintiff
APPLE INC.

E-FILING

ADR

ORIGINAL FILED
JAN 1[illegible] 2011
Richard W. [illegible]
Clerk, U.S. Dist. Court
Northern District of California
San Jose

UNITED STATES DISTRICT COURT

NORTHERN DISTRICT OF CALIFORNIA

HRL

APPLE INC., a California corporation, Plaintiff, v. S3 GRAPHICS CO., LTD., a Cayman Islands corporation, and S3 GRAPHICS, INC., a Delaware corporation, Defendants.	Case No. CV 11-00210 **COMPLAINT FOR DECLARATORY JUDGMENT OF NON-INFRINGEMENT AND INVALIDITY OF PATENTS** **DEMAND FOR JURY TRIAL**

Plaintiff Apple Inc. ("Apple") alleges against Defendants S3 Graphics Co., Ltd. ("S3G Cayman") and S3 Graphics, Inc. ("S3G Delaware") (collectively, "S3 Graphics") as follows:

NATURE OF THE ACTION

1. This is an action brought pursuant to the Declaratory Judgment Act, 28 U.S.C. § 2210, for a declaratory judgment of non-infringement and invalidity of patents S3 Graphics has asserted against Apple in proceedings before the United States International Trade Commission.

2. Apple seeks a declaratory judgment of non-infringement and invalidity of United States Patent Nos. 6,658,146 ("the '146 Patent") (attached as Exhibit A); 6,683,978 (the "the '978 Patent") (attached as Exhibit B); 6,775,417 ("the '417 Patent") (attached as Exhibit C); and 7,043,087 ("the '087 Patent") (attached as Exhibit D) (collectively, "the Asserted Patents") under the Patent Laws of the United States, 35 U.S.C. §§ 101, *et seq.*

THE PARTIES

3. Plaintiff Apple is a California corporation with its principal place of business at 1 Infinite Loop, Cupertino, California 95014.

4. On information and belief, Defendant S3G Cayman is a Cayman Islands corporation with its principal place of business at 2nd Floor, Zephyr House, Mary Street, P.O. Box 709, Grand Cayman, Grand Cayman Islands, British West Indies.

5. On information and belief, Defendant S3G Delaware is a Delaware corporation with its principal place of business at 1025 Mission Court, Fremont, California, 94539. On information and belief, S3G Delaware is a wholly owned subsidiary of S3G Cayman.

JURISDICTION AND VENUE

6. Apple brings this action under the Declaratory Judgment Act, 28 U.S.C. § 2201, for a declaratory judgment of non-infringement and invalidity of the Asserted Patents under the Patent Laws of the United States, 35 U.S.C. §§ 101 *et seq.* This Court has subject matter jurisdiction pursuant to 28 U.S.C. §§ 1331 and 1338(a).

7. This Court has personal jurisdiction over S3G Cayman because S3G Cayman has constitutionally sufficient contacts with California to make personal jurisdiction proper in this Court. On information and belief, S3G Cayman does business in the Northern District of

California, including business conducted through its subsidiary S3G Delaware.

8. On information and belief, S3G Cayman has negotiated and entered into to licenses under the Asserted Patents with other entities in the Northern District of California, including S3G Delaware. On information and belief, S3G Cayman derives the benefit of these licenses, including benefits arising out of activities in California.

9. Further, on information and belief, S3G Delaware acts as the general agent of S3G Cayman in that it was established for, and is engaged in, activities that, but for the existence of the subsidiary, the parent would have to undertake itself. For example, on information and belief, S3G Delaware engages in licensing activities out of its Fremont, California facility on behalf of S3G Cayman relating to patents owned or controlled by S3G Cayman, including the Asserted Patents.

10. On information and belief, S3G Delaware is an alter ego of S3G Cayman in that the subsidiary is the mere instrumentality of the parent and there is such unity of interest and ownership that the separate personalities of the two entities no longer exist. On information and belief, S3G Cayman controls the activities of S3G Delaware, and the two entities do not maintain separate, distinct businesses.

11. Venue in this district is established under 28 U.S.C. §§ 1391(b) and (c).

12. An actual controversy exists between S3 Graphics and Apple as to whether Apple infringes the Asserted Patents. On May 28, 2010, S3 Graphics filed a complaint (the "ITC Complaint") with the United States International Trade Commission under section 337 of the Tariff Act of 1930. The ITC Complaint alleges that Apple infringes the Asserted Patents through the manufacture and distribution of certain hardware and software products. The ITC Complaint further alleges that Apple contributes to and induces infringement of the Asserted Patents by others. A true and correct copy of the ITC Complaint is attached hereto as Exhibit E.

BACKGROUND

13. On information and belief, the Asserted Patents were issued by the United States Patent and Trademark Office and are assigned to S3G Cayman.

14. The Asserted Patents stem from the same parent application and are directed to

various aspects of an image processing system involving compression and decompression of images. The '146, '978, and '417 Patents are all entitled "Fixed-Rate Block-Based Image Compression with Inferred Pixel Values."

15. Apple designs and sells electronic devices, including the iPhone, iPod touch, iPad, and Mac computers. Apple also provides a software development kit ("SDK") that allows third party developers to design applications for its products.

16. As stated above, S3 Graphics has alleged in the ITC Complaint that Apple infringes the Asserted Patents through the manufacture and distribution of certain hardware and software products.

17. On June 25, 2010, the International Trade Commission instituted Investigation No. 337-TA-724, styled *In the Matter of Certain Electronic Devices with Image Processing Systems, Components Thereof, and Associated Software*, to determine whether there is a violation of section 337 based on the allegations of the ITC Complaint.

18. In its Response to the ITC Complaint, Apple denied that there has been a violation of section 337. Apple further has asserted that the Asserted Patents are not infringed and are invalid.

19. Therefore, at the present time, an actual controversy exists between Apple, on the one hand, and S3 Graphics, on the other, as to the validity of the Asserted Patents and the infringement of those patents by Apple. This controversy is of such immediacy and reality as to warrant declaratory relief so that the parties may ascertain their rights and duties with respect to the Asserted Patents.

20. An actual controversy also exists between Apple and S3 Graphics with respect to patent licenses granted by S3 Graphics to third parties who supply components to Apple. Apple contends that these licenses bar S3 Graphics' assertions of infringement, but S3 Graphics has denied the existence of these licenses or denied that Apple is a beneficiary of them.

COUNT ONE

Declaratory Judgment of Non-Infringement of the Asserted Patents

21. Apple realleges and incorporates herein by reference the allegations in paragraphs

1 through 20 above.

22. S3 Graphics has alleged and continues to assert that Apple and its products infringe the Asserted Patents.

23. Apple has not infringed and is not now infringing directly or indirectly, and has not induced or contributed to and is not now inducing or contributing to the infringement of, any claim of the Asserted Patents, either literally or by application of the doctrine of equivalents.

24. Apple seeks a declaratory judgment from this Court under Rule 57 of the Federal Rules of Civil Procedure and 28 U.S.C. § 2201 declaring that Apple is not infringing and has not infringed the Asserted Patents and granting Apple all other declaratory relief to which it may be entitled.

COUNT TWO

Declaratory Judgment of Invalidity of the Asserted Patents

25. Apple realleges and incorporates herein by reference the allegations in paragraphs 1 through 24 above.

26. The claims of the Asserted Patents are invalid because they fail to comply with one or more requirements of the Patent Laws of the United States, including, but not limited to, 35 U.S.C. §§ 101, 102, 103, 112 and/or 116.

27. Apple seeks a declaratory judgment from this Court under Rule 57 of the Federal Rules of Civil Procedure and 28 U.S.C. § 2201 declaring that the Asserted Patents are invalid and granting Apple all other declaratory relief to which it maybe entitled.

PRAYER FOR RELIEF

WHEREFORE, Apple prays for judgment and relief as follows:

1. That Apple has not infringed, contributed to the infringement of, nor induced infringement of any claim of the Asserted Patents;

2. That the claims of the Asserted Patents are invalid;

3. That this case is exceptional under 35 U.S.C. § 285;

4. For reasonable attorneys' fees pursuant to 35 U.S.C. § 285;

5. For further necessary or proper relief pursuant to 28 U.S.C. § 2202; and

6. For such other relief as the Court deems just and proper.

DEMAND FOR JURY TRIAL

Plaintiff hereby demands a jury trial in this action.

Dated: January 13, 2011

ORRICK, HERRINGTON & SUTCLIFFE LLP

Vickie L. Feeman

Vickie L. Feeman
Attorneys for Plaintiff
APPLE INC.

Exhibit A

US006658146B1

(12) **United States Patent**
Iourcha et al.

(10) **Patent No.: US 6,658,146 B1**
(45) **Date of Patent: *Dec. 2, 2003**

(54) FIXED-RATE BLOCK-BASED IMAGE COMPRESSION WITH INFERRED PIXEL VALUES

(75) Inventors: **Konstantine I. Iourcha**, San Jose, CA (US); **Krishna S. Nayak**, Stanford, CA (US); **Zhou Hong**, San Jose, CA (US)

(73) Assignee: **S3 Graphics Co., Ltd.**, Grand Cayman (KY)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **09/351,930**

(22) Filed: **Jul. 12, 1999**

Related U.S. Application Data

(63) Continuation of application No. 08/942,850, filed on Oct. 2, 1997, now Pat. No. 5,956,431.

(51) **Int. Cl.**^{7} **G06K 9/00**
(52) **U.S. Cl.** **382/166**; 382/162; 382/232
(58) **Field of Search** 382/232, 253, 382/239, 236, 166, 162, 272; 345/593; 341/106; 358/1.9, 500

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,821,208 A * 4/1989 Ryan et al. 364/158

FOREIGN PATENT DOCUMENTS

JP 405216993 A * 8/1993 .... G06F/15/70

OTHER PUBLICATIONS

Feng et al., "A Dynamic Address Vector Quantization Algorithm Based on Inter–Block and Inter–Color Correction for Color image Coding", IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 3, May 1989, pps. 1755–1758.*

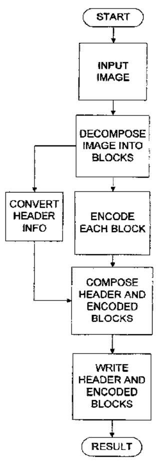
(List continued on next page.)

Primary Examiner—Anh Hong Do
(74) *Attorney, Agent, or Firm*—Carr & Ferrell LLP

(57) **ABSTRACT**

An image processing system includes an image encoder system and a image decoder system that are coupled together. The image encoder system includes a block decomposer and a block encoder that are coupled together. The block encoder includes a color quantizer and a bitmap construction module. The block decomposer breaks an original image into blocks. Each block is then processed by the block encoder. Specifically, the color quantizer selects some number of base points, or codewords, that serve as reference pixel values, such as colors, from which quantized pixel values are derived. The bitmap construction module then maps each pixel colors to one of the derived quantized colors. The codewords and bitmap are output as encoded image blocks. The decoder system includes a block decoder. The block decoder includes a block type detector, one or more decoder units, and an output selector. Using the codewords of the encoded data blocks, the comparator and the decoder units determine the quantized colors for the encoded image block and map each pixel to one of the quantized colors. The output selector outputs the appropriate color, which is ordered in an image composer with the other decoded blocks to output an image representative of the original image. A method for encoding an original image and for decoding the encoded image to generate a representation of the original image is also disclosed.

22 Claims, 16 Drawing Sheets


START
INPUT IMAGE
DECOMPOSE IMAGE INTO BLOCKS
CONVERT HEADER INFO
ENCODE EACH BLOCK
COMPOSE HEADER AND ENCODED BLOCKS
WRITE HEADER AND ENCODED BLOCKS
RESULT

U.S. PATENT DOCUMENTS

4,887,151	A	*	12/1989	Wataya	358/539
5,734,744	A		3/1998	Wittenstein et al.	382/166
5,748,904	A	*	5/1998	Huang et al.	345/544
5,787,192	A	*	7/1998	Takaichi et al.	382/166
5,822,465	A		10/1998	Normile et al.	382/253
5,956,425	A	*	9/1999	Yoshida	382/234
5,956,431	A	*	9/1999	Iourcha et al.	382/253
6,075,619	A	*	6/2000	Iizuka	358/432

OTHER PUBLICATIONS

A. Schilling, et al.; “Texram: A Smart Memory for Texturing”; IEEE Computer Graphics & Applications; May 1996; 16(3) pp. 9–19.

G. Knittel, et al.; “Hardware and Software for Superior Texture Performance”; In 10; Eurographics Hardware Workshop ’95; Maastricht, NL; Aug. 28–29, 1995; pp. 1–8.

G. Campbell, et al.; “Two Bit/Pixel Full Color Encoding”; Computer Graphics, (Proc. Siggraph ’86); Aug. 18–22, 1986; vol. 20, No. 4, Dallas TX; pp. 215–219.

* cited by examiner

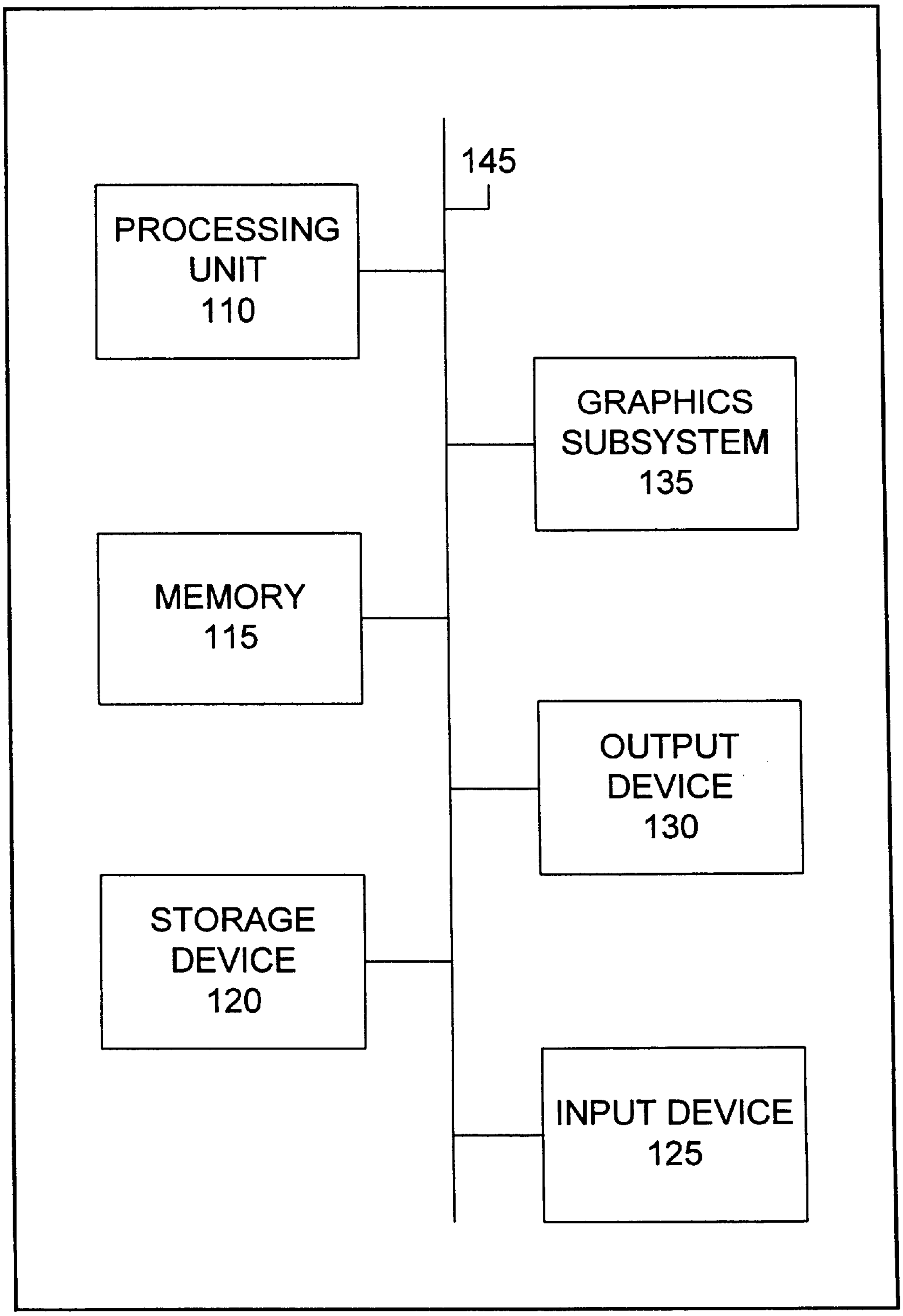

145
PROCESSING UNIT 110
GRAPHICS SUBSYSTEM 135
MEMORY 115
OUTPUT DEVICE 130
STORAGE DEVICE 120
INPUT DEVICE 125

FIG. 1

105

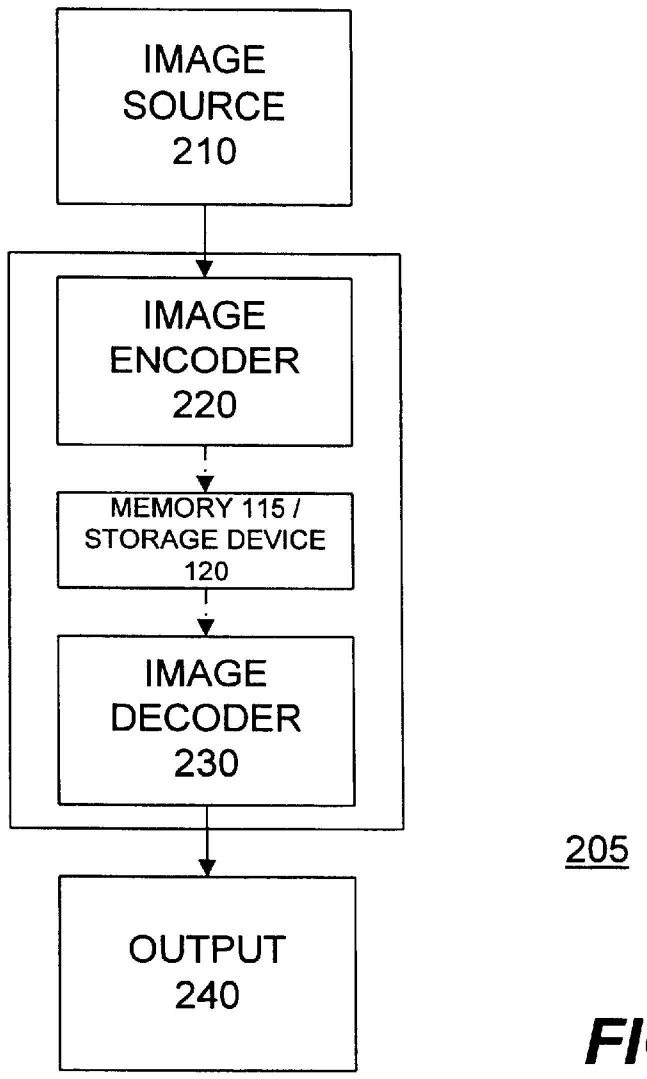


IMAGE SOURCE 210
IMAGE ENCODER 220
MEMORY 115 / STORAGE DEVICE 120
IMAGE DECODER 230
OUTPUT 240
205

FIG. 2A

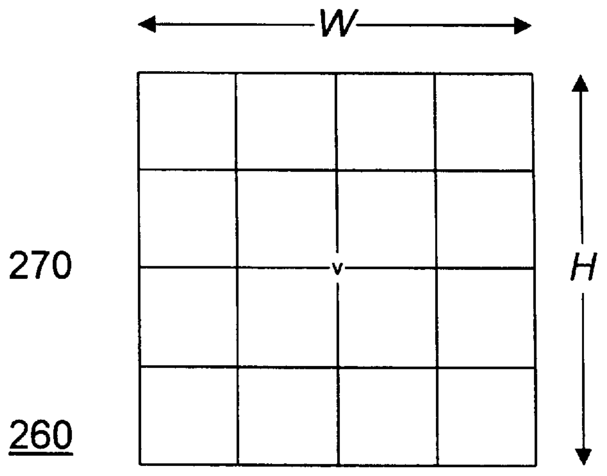

W
H
v
270
260

FIG. 2B

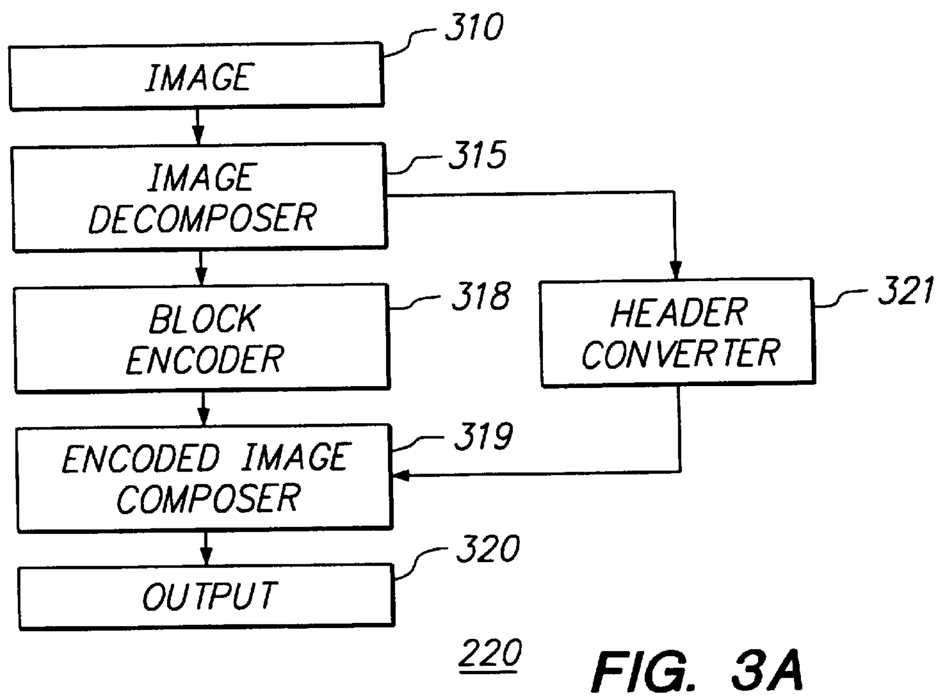

310
IMAGE
315
IMAGE DECOMPOSER
318
BLOCK ENCODER
321
HEADER CONVERTER
319
ENCODED IMAGE COMPOSER
320
OUTPUT
220

FIG. 3A

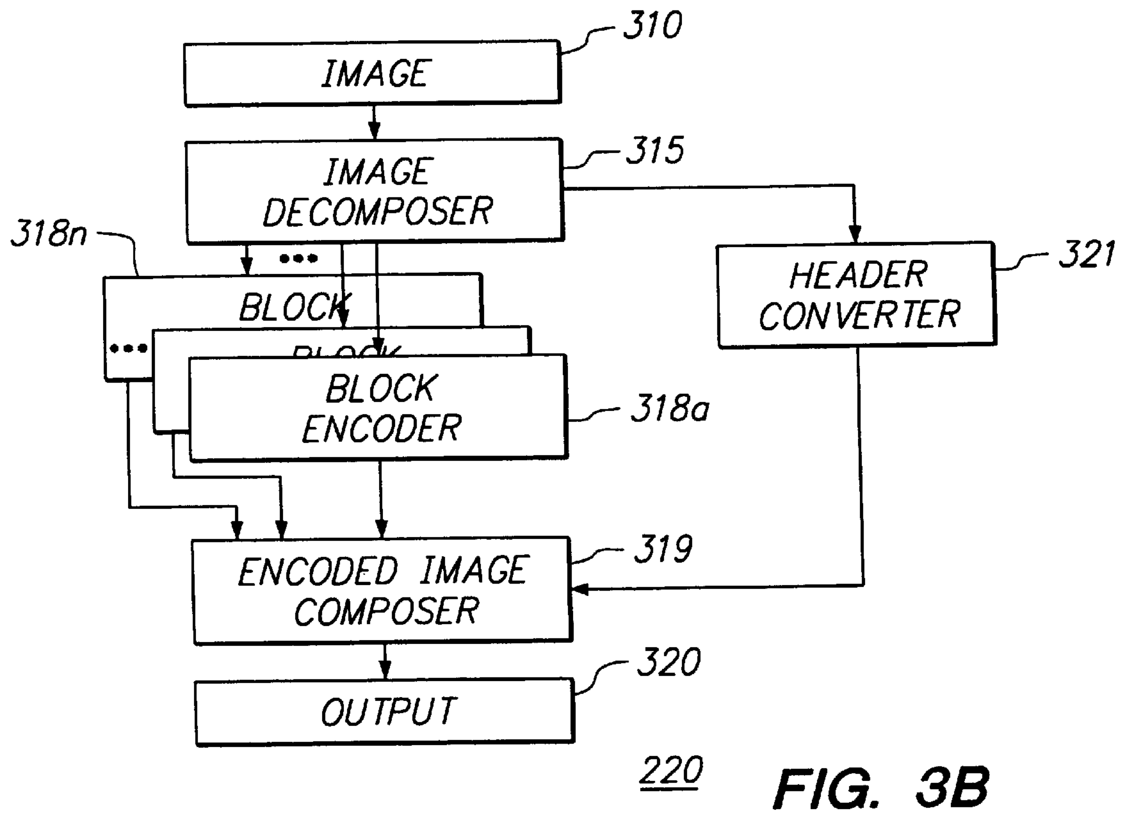
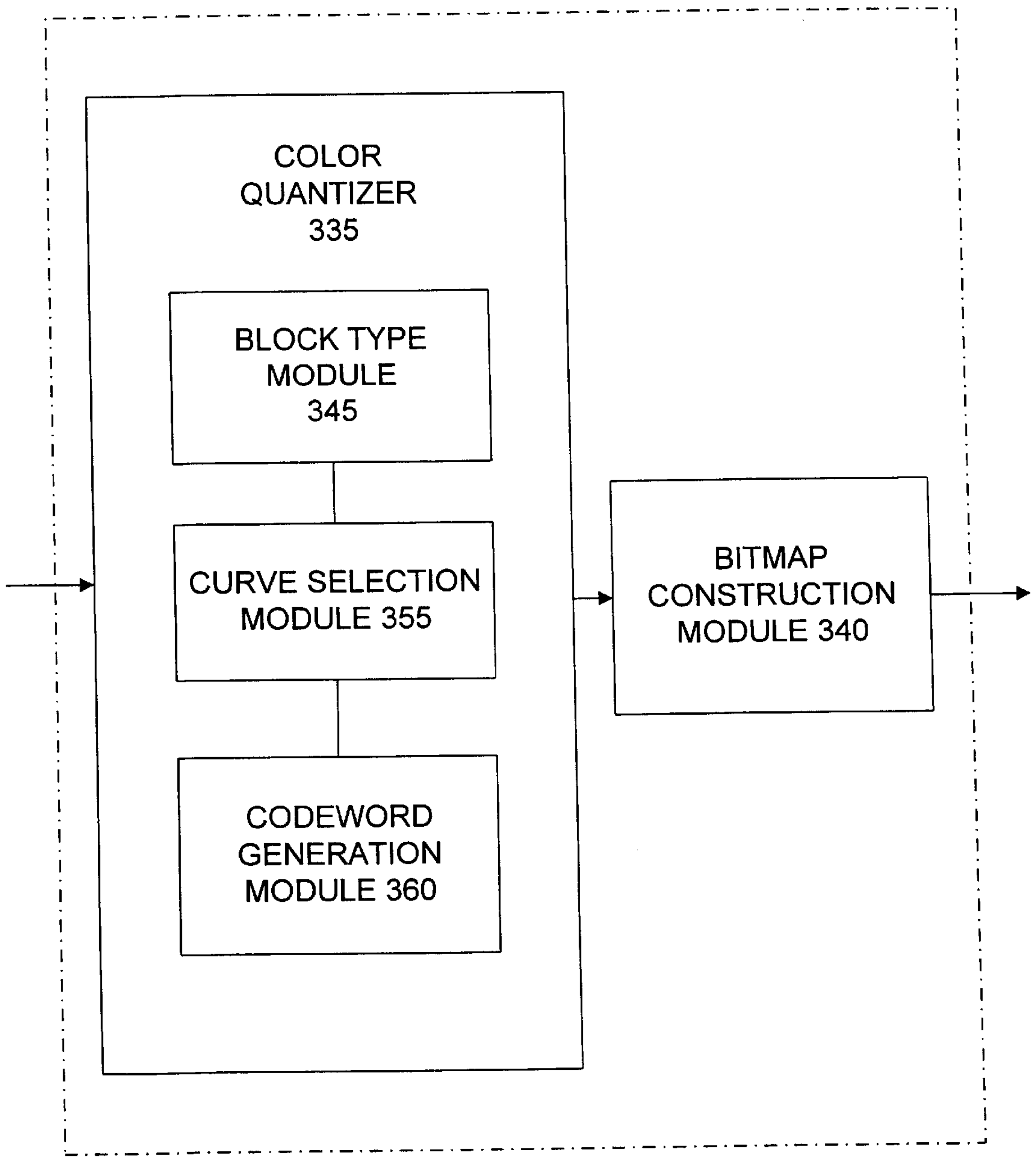

310
IMAGE
315
IMAGE DECOMPOSER
318n
BLOCK
BLOCK
BLOCK ENCODER
318a
321
HEADER CONVERTER
319
ENCODED IMAGE COMPOSER
320
OUTPUT
220

FIG. 3B



COLOR QUANTIZER 335
BLOCK TYPE MODULE 345
CURVE SELECTION MODULE 355
CODEWORD GENERATION MODULE 360
BITMAP CONSTRUCTION MODULE 340

FIG. 3C

318

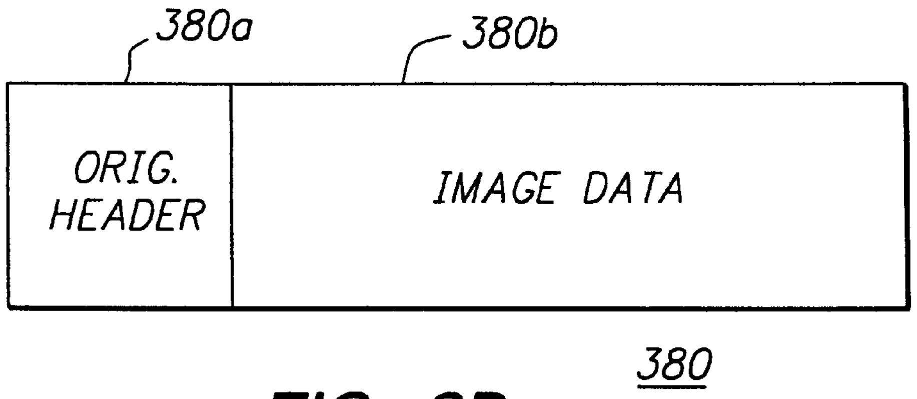


FIG. 3D

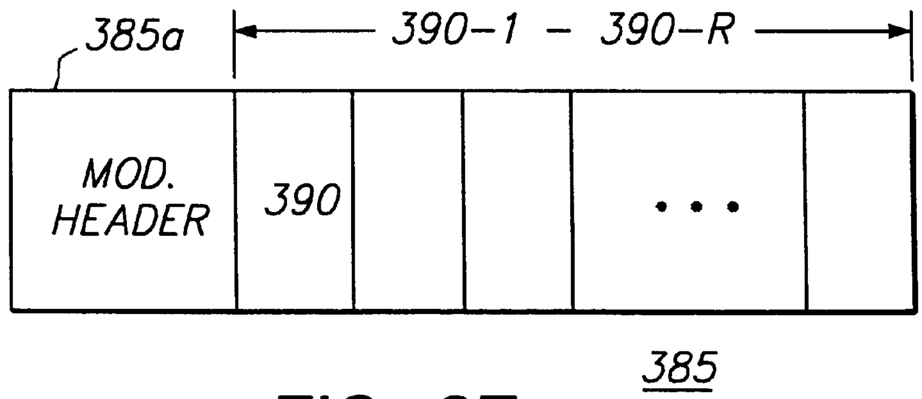


FIG. 3E

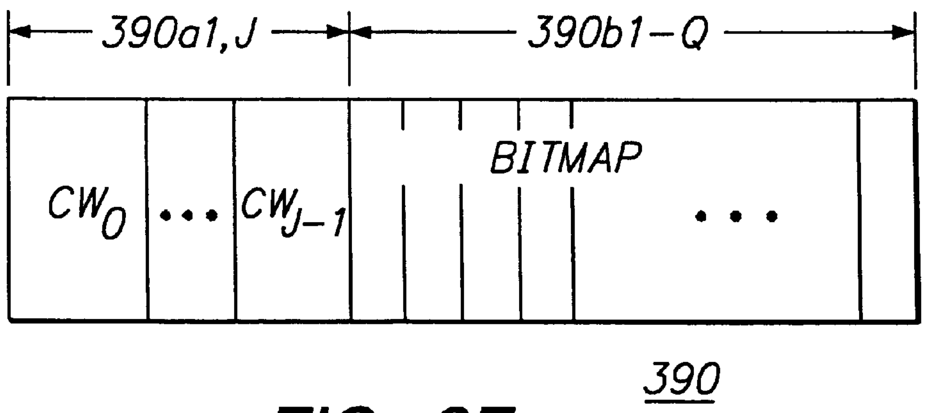


FIG. 3F

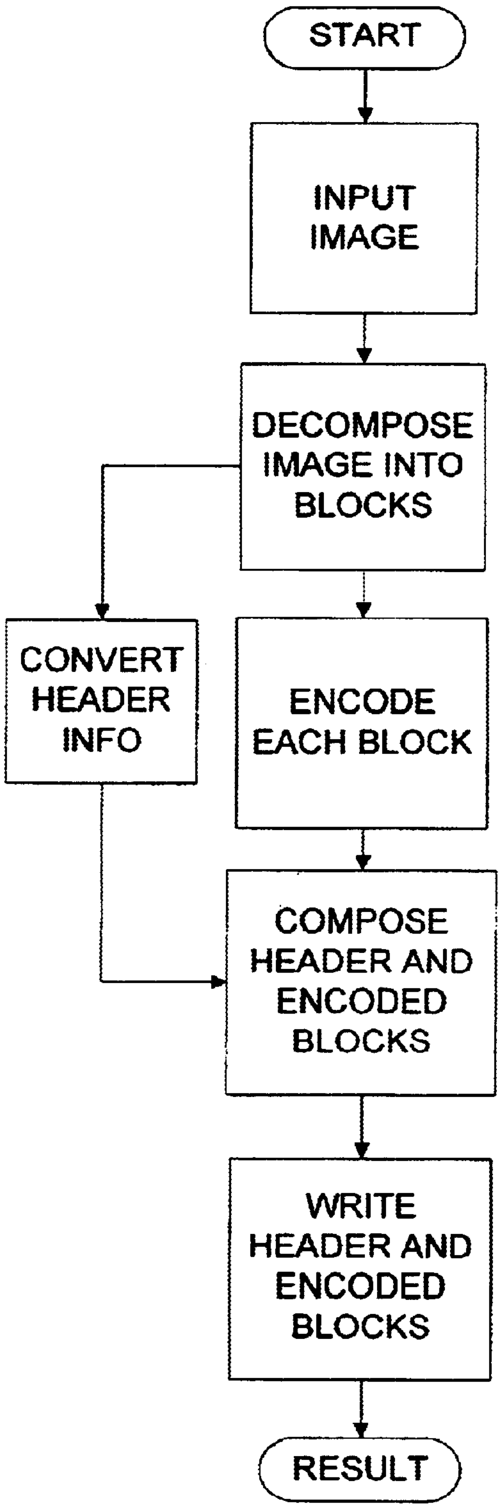

START
INPUT IMAGE
DECOMPOSE IMAGE INTO BLOCKS
CONVERT HEADER INFO
ENCODE EACH BLOCK
COMPOSE HEADER AND ENCODED BLOCKS
WRITE HEADER AND ENCODED BLOCKS
RESULT

Fig 4A

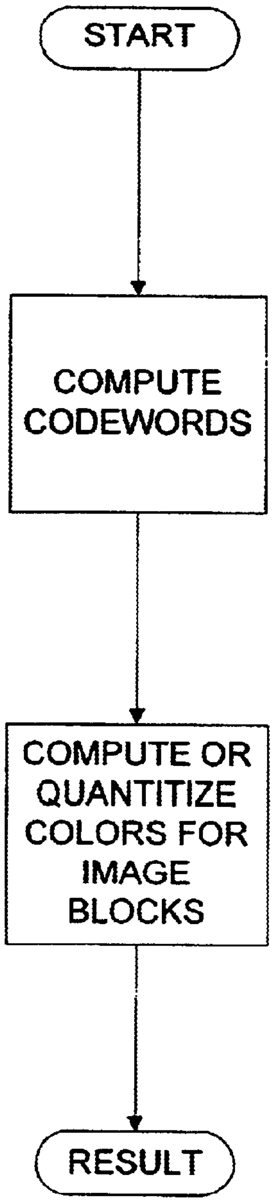

START
COMPUTE CODEWORDS
COMPUTE OR QUANTITIZE COLORS FOR IMAGE BLOCKS
RESULT

Fig 4B

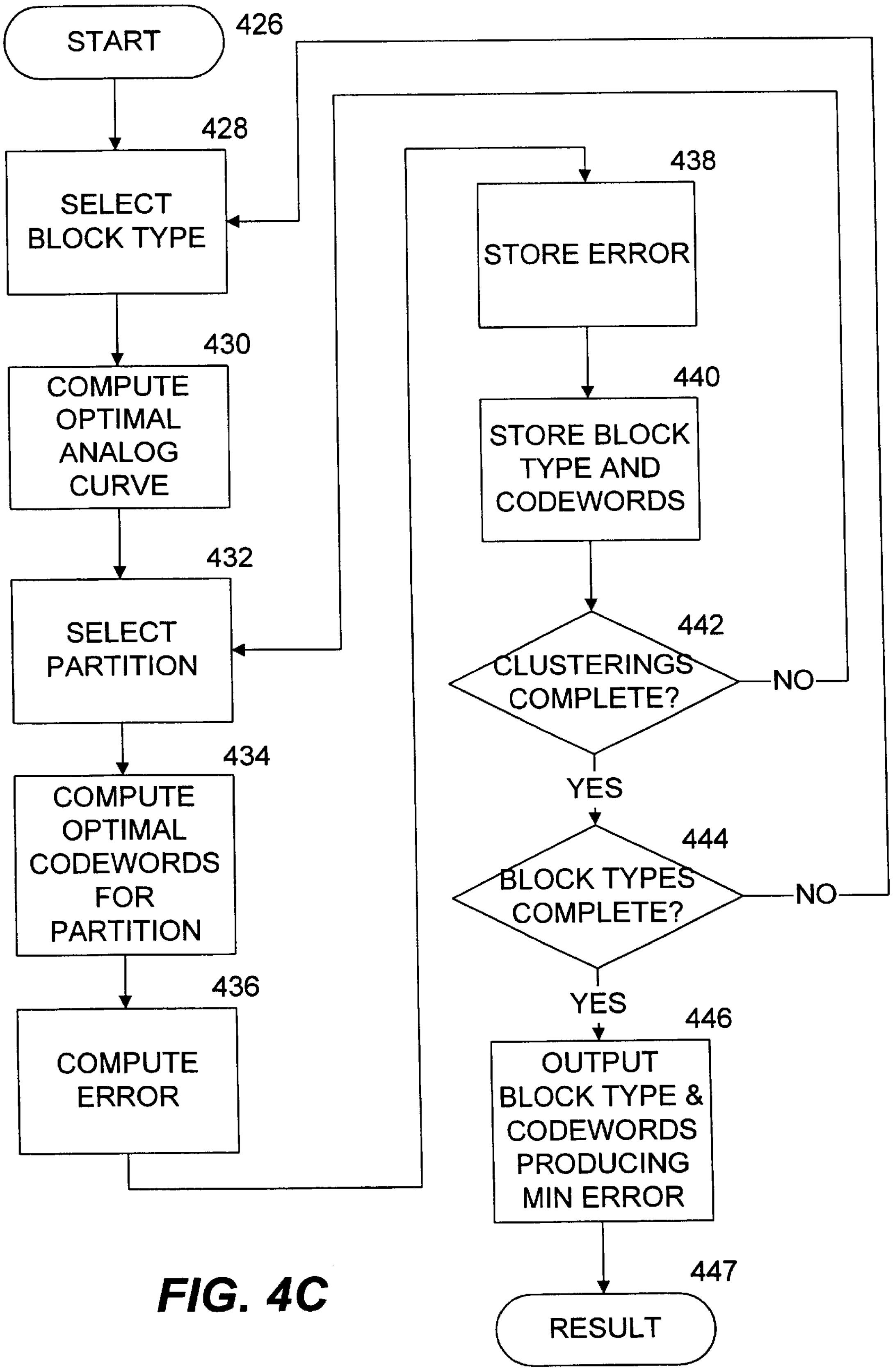

START
426
428
SELECT BLOCK TYPE
430
COMPUTE OPTIMAL ANALOG CURVE
432
SELECT PARTITION
434
COMPUTE OPTIMAL CODEWORDS FOR PARTITION
436
COMPUTE ERROR
438
STORE ERROR
440
STORE BLOCK TYPE AND CODEWORDS
442
CLUSTERINGS COMPLETE?
NO
YES
444
BLOCK TYPES COMPLETE?
NO
YES
446
OUTPUT BLOCK TYPE & CODEWORDS PRODUCING MIN ERROR
447
RESULT

FIG. 4C

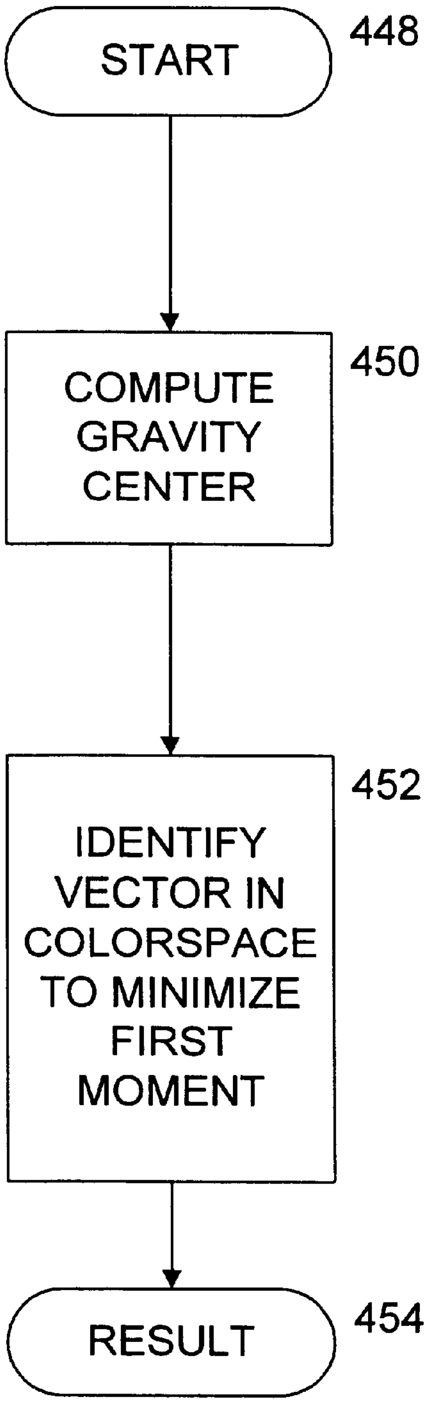

START
448
COMPUTE GRAVITY CENTER
450
IDENTIFY VECTOR IN COLORSPACE TO MINIMIZE FIRST MOMENT
452
RESULT
454

FIG. 4D

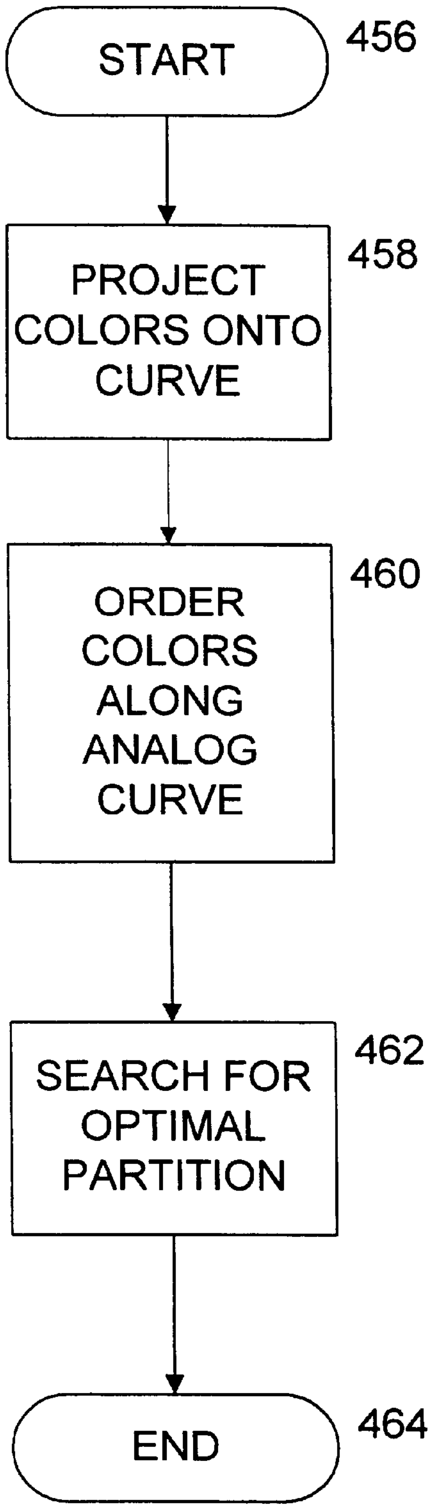

START
456
PROJECT COLORS ONTO CURVE
458
ORDER COLORS ALONG ANALOG CURVE
460
SEARCH FOR OPTIMAL PARTITION
462
END
464

FIG. 4E

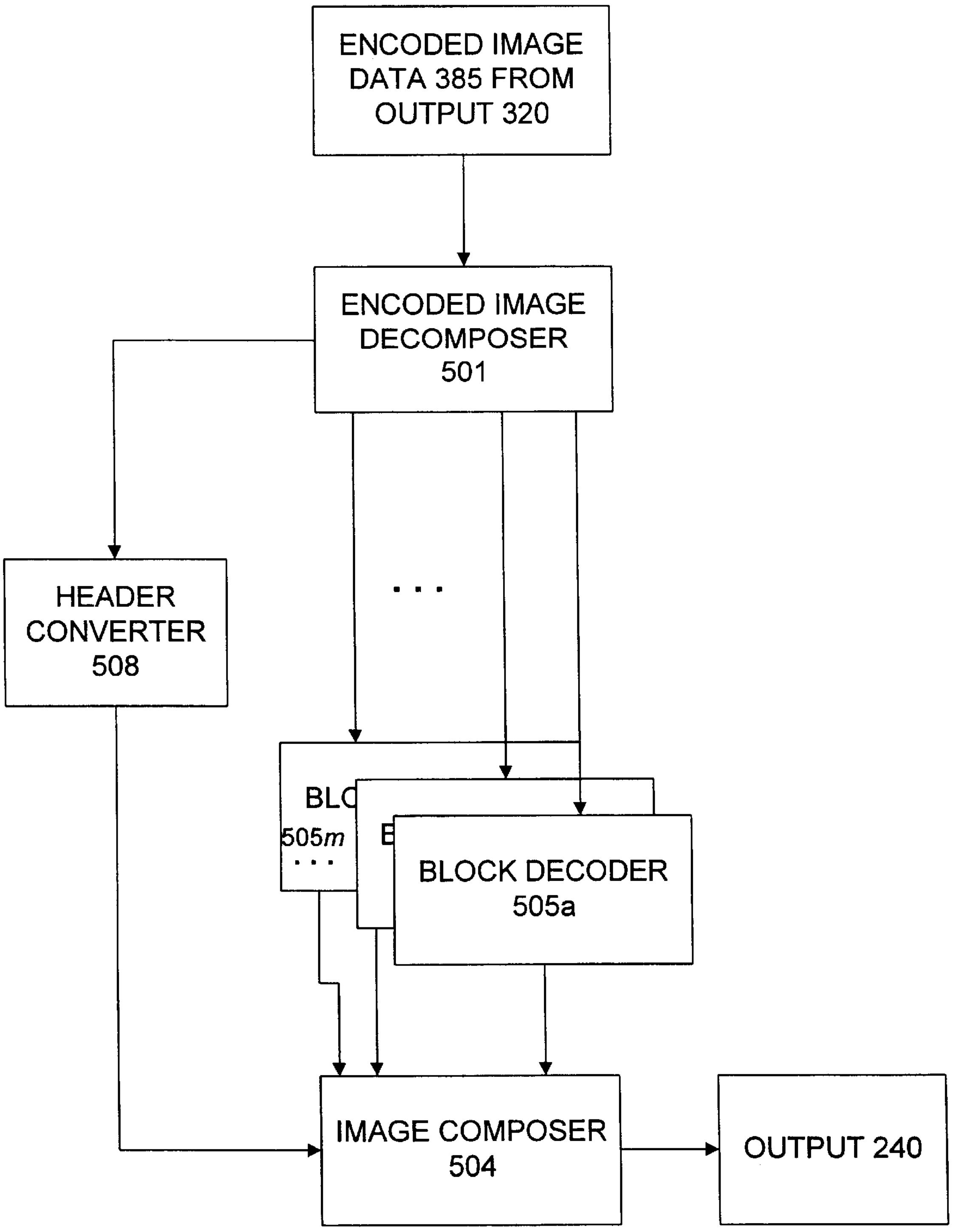

ENCODED IMAGE DATA 385 FROM OUTPUT 320
ENCODED IMAGE DECOMPOSER 501
HEADER CONVERTER 508
. . .
BLC
505m
. . .
E
BLOCK DECODER 505a
IMAGE COMPOSER 504
OUTPUT 240

FIG. 5A

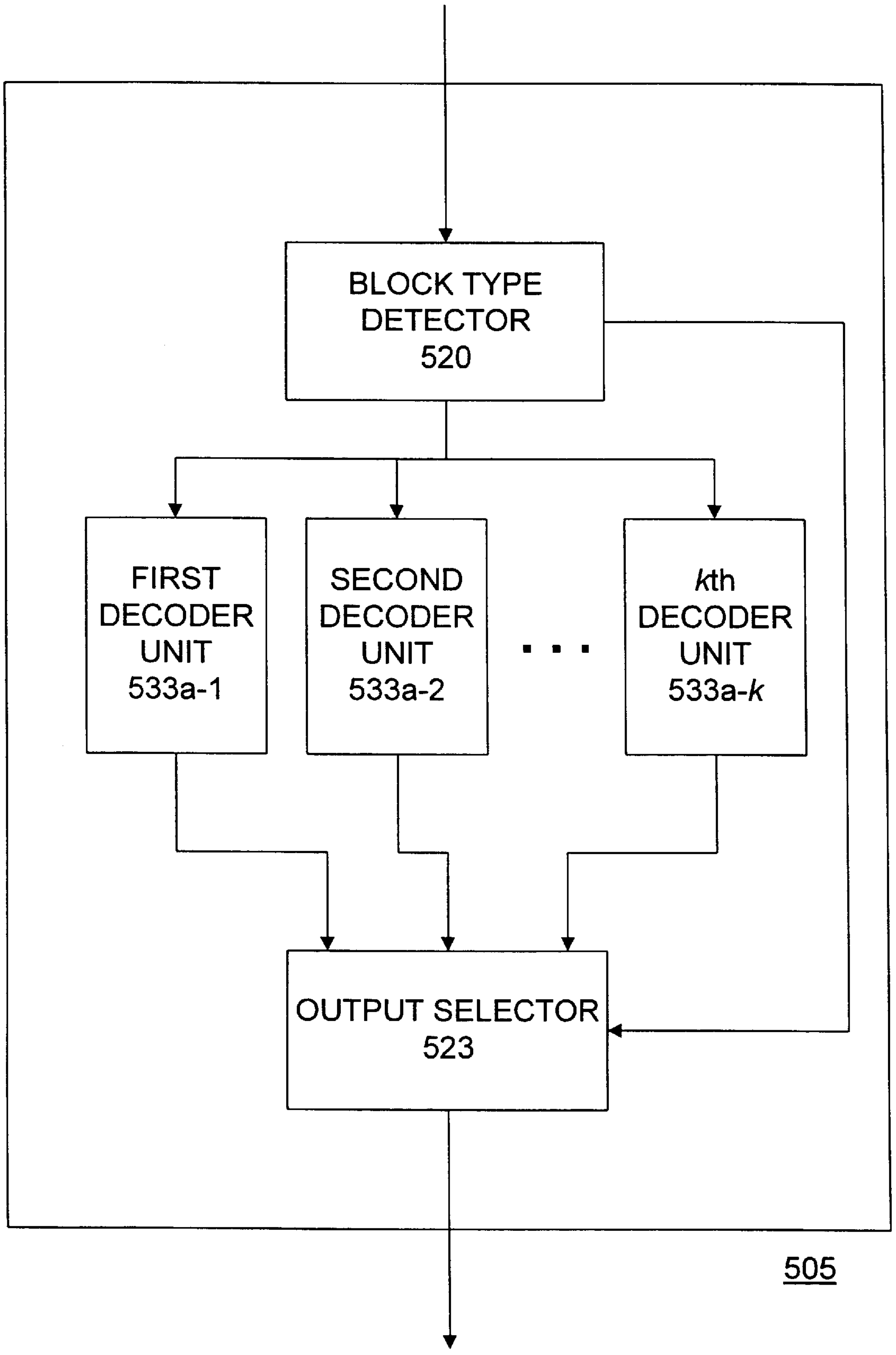

BLOCK TYPE DETECTOR 520
FIRST DECODER UNIT 533a-1
SECOND DECODER UNIT 533a-2
. . .
*k*th DECODER UNIT 533a-*k*
OUTPUT SELECTOR 523
505

FIG. 5B

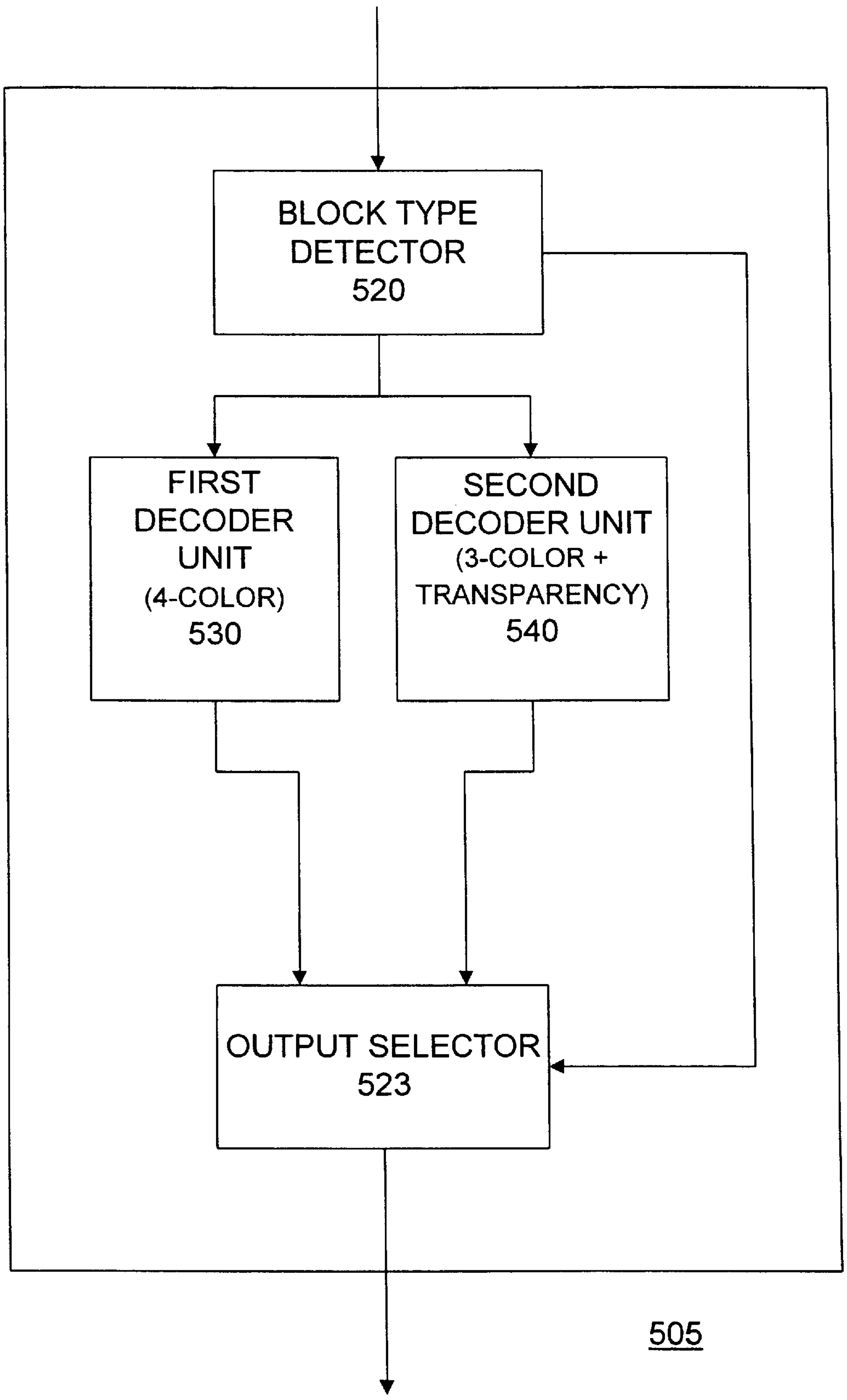

BLOCK TYPE DETECTOR 520
FIRST DECODER UNIT (4-COLOR) 530
SECOND DECODER UNIT (3-COLOR + TRANSPARENCY) 540
OUTPUT SELECTOR 523
505

FIG. 5C

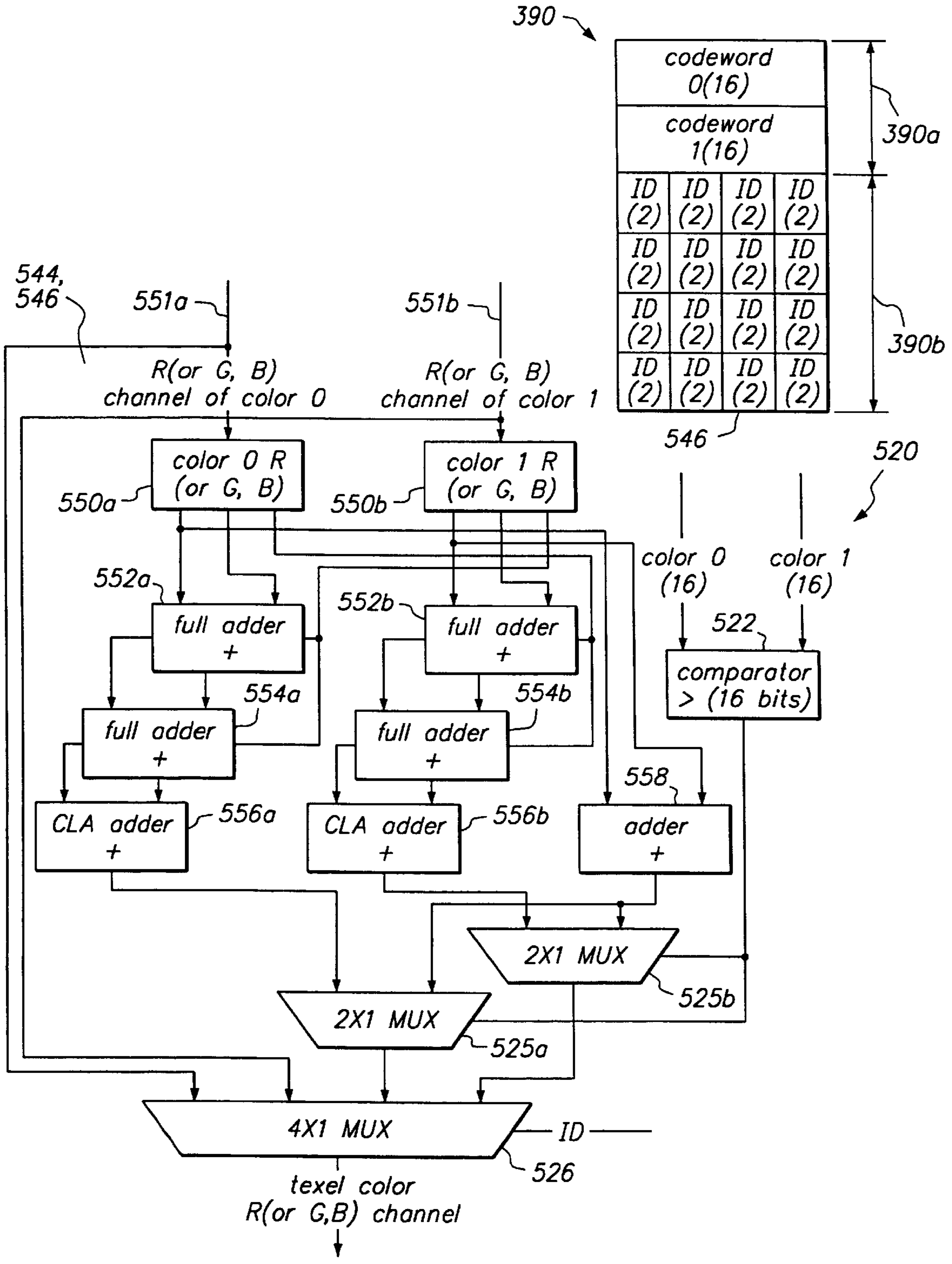

390
codeword 0(16)
codeword 1(16)
ID (2)
390a
390b
546
520
544, 546
551a
551b
R(or G, B) channel of color 0
R(or G, B) channel of color 1
color 0 R (or G, B)
color 1 R (or G, B)
550a
550b
color 0 (16)
color 1 (16)
522
comparator > (16 bits)
552a
552b
full adder +
554a
554b
558
CLA adder +
556a
556b
adder +
2X1 MUX
525b
525a
4X1 MUX
ID
526
texel color R(or G,B) channel

FIG. 5D

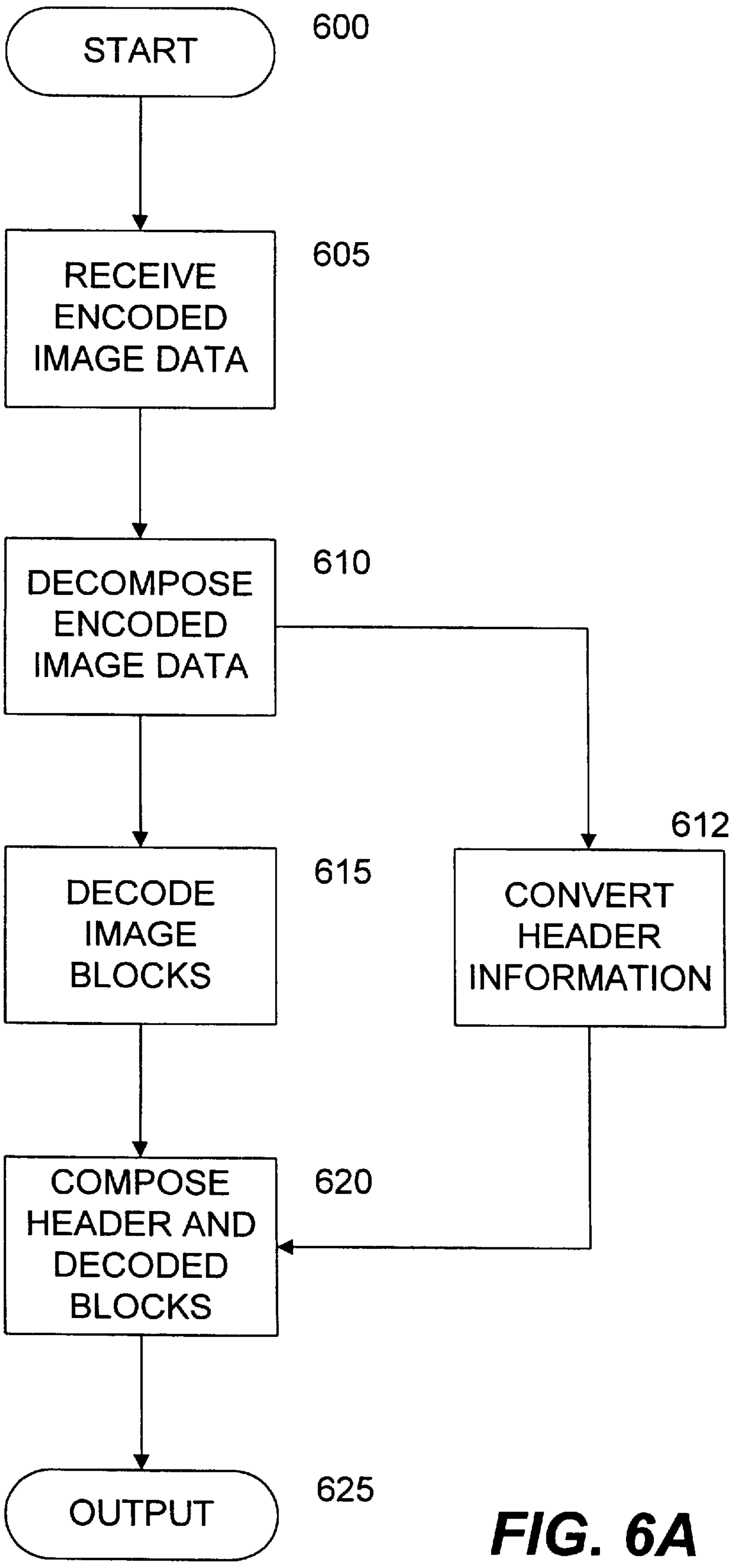

START
600
RECEIVE ENCODED IMAGE DATA
605
DECOMPOSE ENCODED IMAGE DATA
610
DECODE IMAGE BLOCKS
615
CONVERT HEADER INFORMATION
612
COMPOSE HEADER AND DECODED BLOCKS
620
OUTPUT
625

FIG. 6A

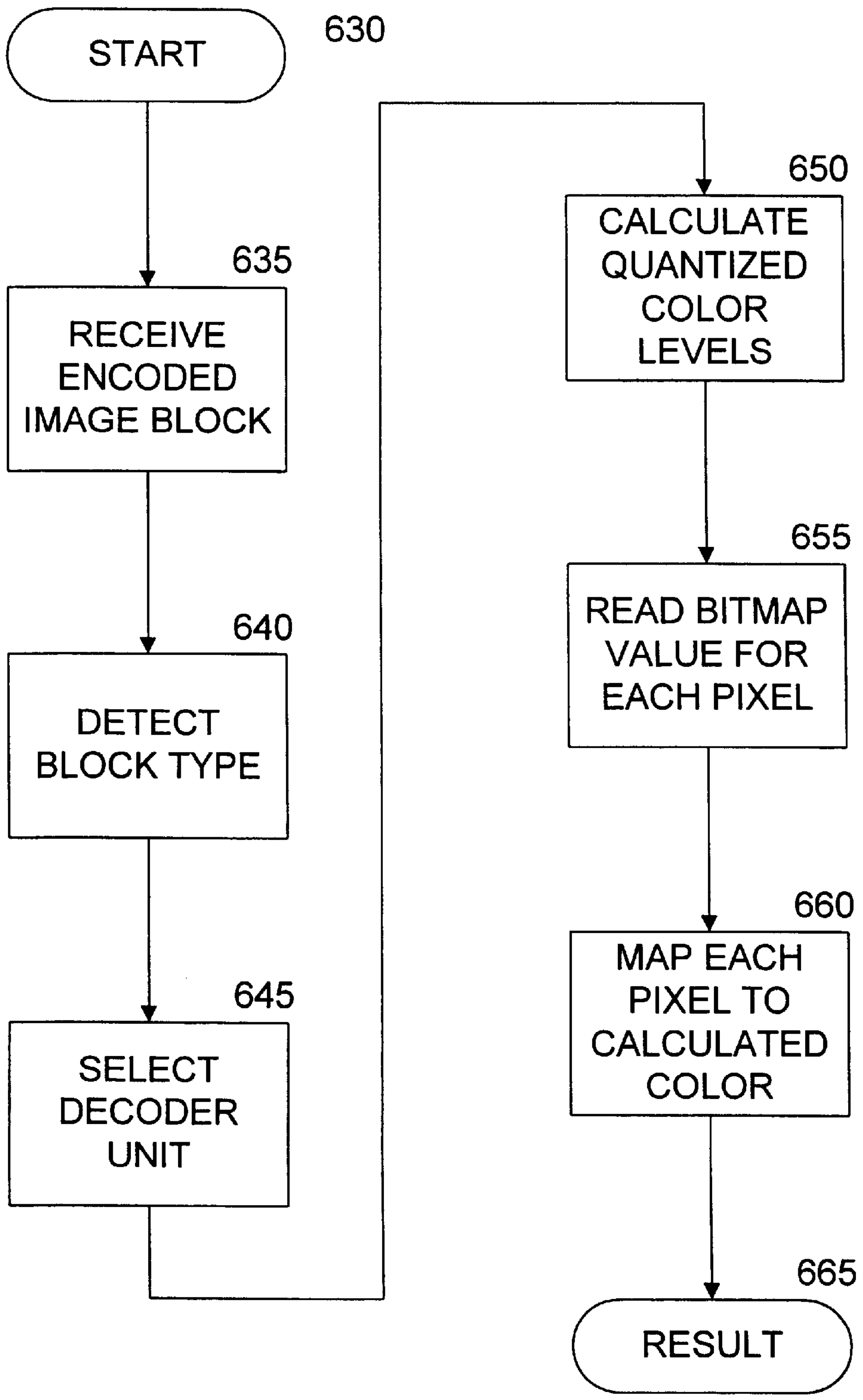

START
630
635
RECEIVE ENCODED IMAGE BLOCK
640
DETECT BLOCK TYPE
645
SELECT DECODER UNIT
650
CALCULATE QUANTIZED COLOR LEVELS
655
READ BITMAP VALUE FOR EACH PIXEL
660
MAP EACH PIXEL TO CALCULATED COLOR
665
RESULT

FIG. 6B

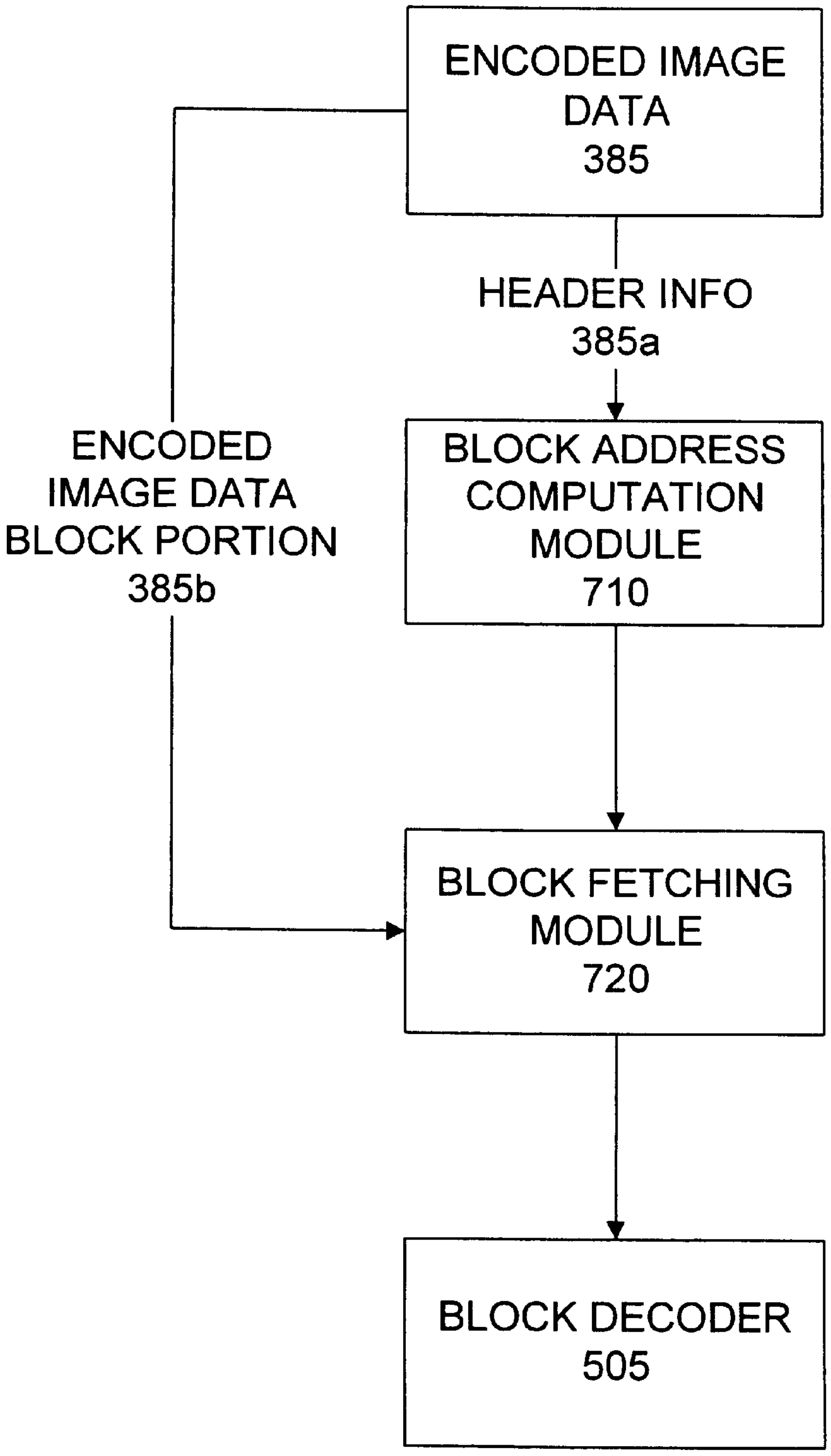

ENCODED IMAGE DATA 385
HEADER INFO 385a
BLOCK ADDRESS COMPUTATION MODULE 710
ENCODED IMAGE DATA BLOCK PORTION 385b
BLOCK FETCHING MODULE 720
BLOCK DECODER 505
700

FIG. 7A

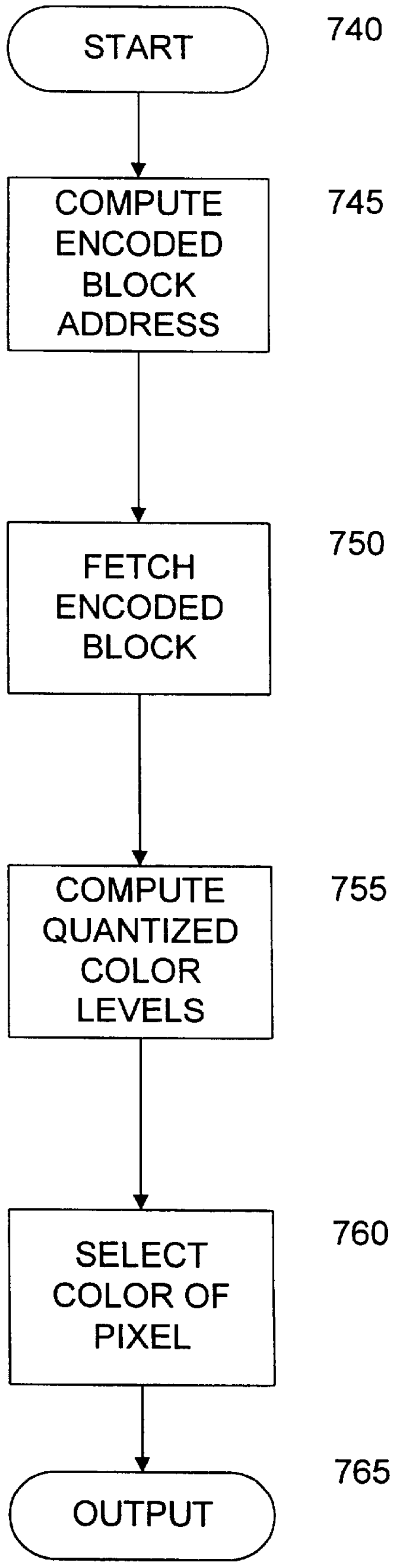

START
740
COMPUTE ENCODED BLOCK ADDRESS
745
FETCH ENCODED BLOCK
750
COMPUTE QUANTIZED COLOR LEVELS
755
SELECT COLOR OF PIXEL
760
OUTPUT
765

FIG. 7B

FIXED-RATE BLOCK-BASED IMAGE COMPRESSION WITH INFERRED PIXEL VALUES

This is a CON of Ser. No. 08/942,850, filed Oct. 2, 1997, now U.S. Pat. No. 5,956,431.

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to image processing systems, and more specifically, to three-dimensional rendering systems using fixed-rate image compression for textures.

2. Description of the Related Art

The art of generating images, such as realistic or animated graphics on a computer is known. To generate such images requires tremendous memory bandwidth and processing power on a graphics subsystem. To reduce the bandwidth and processing power requirements, various compression methods and systems were developed. These methods and systems included Entropy or lossless encoders, discrete cosine transform or JPEG type compressors, block truncation coding, color cell compression, and others. Each of these methods and systems, however, have numerous drawbacks.

Entropy or lossless encoders include Lempel-Ziv encoders and are used for many different purposes. Entropy coding relies on predictability. For data compression using Entropy encoders, a few bits are used to encode the most commonly occurring symbols. In stationary systems where the probabilities are fixed, Entropy coding provides a lower bound for the compression than can be achieved with a given alphabet of symbols. A problem with Entropy coding is that it does not allow random access to any given symbol. The part of the compressed data preceding a symbol of interest must be first fetched and decompressed to decode the symbol which takes considerable processing time and resources as well as decreasing memory throughput. Another problem with existing Entropy methods and systems is that they do not provide any guaranteed compression factor which makes this type of encoding scheme impractical where the memory size is fixed.

Discrete Cosine Transform ("DCT") or JPEG-type compressors, allow users to select a level of image quality. With DCT, uncorrelated coefficients are produced so that each coefficient can be treated independently without loss of compression efficiency. The DCT coefficients can be quantized using visually-weighted quantization values which selectively discard the least important information.

DCT, however, suffers from a number of shortcomings. One problem with DCT and JPEG-type compressors is that they require usually bigger blocks of pixels, typically 8×8 or 16×16 pixels, as a minimally accessible unit in order to obtain a reasonable compression factor and quality. Access to a very small area, or even a single pixel involves fetching a large quantity of compressed data, thus requiring increased processor power and memory bandwidth. A second problem with DCT and JPEG-type compressors is that the compression factor is variable, therefore requiring a complicated memory management system that, in turn, requires greater processor resources. A third problem with DCT and JPEG-type compression is that using a large compression factor significantly degrades image quality. For example, the image may be considerably distorted with a form of a ringing around the edges in the image as well as noticeable color shifts in areas of the image. Neither artifact can be removed with subsequent low-pass filtering.

A fourth problem with DCT and JPEG-type compression is that such a decompressor is complex and has a significant associated hardware cost. Further, the high latency of the decompressor results in a large additional hardware cost for buffering throughout the system to compensate for the latency. Finally, a fifth problem with DCT and JPEG-type compressors is that it is not clear whether a color keyed image can be compressed with such a method and system.

Block truncation coding ("BTC") and color cell compression ("CCC") use a local one-bit quantizer on 4×4 pixel blocks. The compressed data for such a block consists of only two colors and 16-bits that indicate which one of the two colors is assigned to each of the 16 pixels. Decoding a BTC/CCC image consists of using a multiplexer with a look-up table so that once a 16-texel-block (32-bits) is retrieved from memory, the individual pixels are decoded by looking up the two possible colors for that block and selecting the color according to the associated bit from the 16 decision bits.

The BTC/CCC methods quantize each block to just two color levels resulting in significant image degradation. Further, a two-bit variation of CCC stores the two colors as eight-bit indices into a 256-entry color lookup table. Thus, such pixel blocks cannot be decoded without fetching additional information that can consume additional memory bandwidth.

The BTC/CCC methods and systems can use a three-bit per pixel scheme which store the two colors as 16-bit values (not indices into a table) resulting in pixel blocks of six bytes. Fetching such units, however, decreases system performance because of additional overhead due to memory misalignment. Another problem with BTC/CCC is that when it is used to compress images that use color keying to indicate transparent pixels, there will be a high degradation of image quality.

Therefore, there is a need for a method and system that maximizes the accuracy of compressed images while minimizing storage, memory bandwidth requirements, and decoding hardware complexities, while also compressing image data blocks into convenient sizes to maintain alignment for random access to any one or more pixels.

SUMMARY OF THE INVENTION

An image processing system includes an image encoder system and an image decoder system that are coupled together. The image encoder system includes a block decomposer and a block encoder that are coupled together. The block encoder includes a color quantizer and a bitmap construction module. The block decomposer breaks an original image into image blocks, each having a plurality of pixel values (e.g. colors) or equivalent color points. Each image block is then processed by the block encoder. Specifically, the color quantizer computes some number of base points, or codewords, that serve as reference pixel values, such as colors, from which computed or quantized pixel values are derived. The bitmap construction module then maps at least one pixel value in the image block to one of the computed or quantized colors or one of the codewords. The codewords and bitmap are output as encoded image blocks.

The decoder system includes a block decoder having one or more decoder units and an output selector. The block decoder may also include a block type detector for determining the block type of an image block. The block type determines the number of computed colors to use for mapping each pixel color from an image block. Using the codewords of the encoded data blocks, the comparator and the decoder units determine the computed colors for the encoded image block and map each pixel to one of the computed colors. The output selector outputs the appropriate color, which is ordered in an image composer with the other decoded blocks to output an image representative of the original image.

The present invention also includes a method of compressing an original image block having a set of original colors. The method includes: computing a set of codewords from the set of original colors; computing a set of computed colors using the set of codewords; and mapping each original color to one of the computed colors or one of the codewords to produce an index for each original color.

The compressed or encoded image block, which has a first set of indices and a set of codewords, where a set is equal to or greater than one, is decoded by: computing at least one computed color using the set of codewords; and mapping an index within the first set of indices to one of the computed colors or one of the codewords.

Those of ordinary skill in the art will readily recognize that the present invention may be practiced using any general purpose computer system, such as the computer system described below, or any "hardwired" device specifically designed to perform the method, such as but not limited to devices implemented using ASIC or FPGA technology and the like.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a block diagram of a data processing system in accordance with the present invention;

FIG. **2A** is a block diagram of an image processing system in accordance with the present invention;

FIG. **2B** is a graphical representation of an image block in accordance with the present invention;

FIG. **3A** is a block diagram of a first embodiment an image encoder system in accordance with the present invention;

FIG. **3B** is a block diagram of a second embodiment of an image encoder system in accordance with the present invention;

FIG. **3C** is a block diagram of an image block encoder in accordance with the present invention;

FIG. **3D** is a data sequence diagram of an original image in accordance with the present invention;

FIG. **3E** is a data sequence diagram of encoded image data of the original image output from the image encoder system in accordance with the present invention;

FIG. **3F** is a data sequence diagram of an encoded image block from the image block encoder in accordance with the present invention;

FIGS. **4A–4F** are flow diagrams illustrating an encoding process in accordance with the present invention;

FIG. **5A** is a block diagram of an image decoder system in accordance with the present invention;

FIG. **5B** is a block diagram of a first embodiment of a block decoder in accordance with the present invention;

FIG. **5C** is a block diagram of a second embodiment of a block decoder in accordance with the present invention;

FIG. **5D** is a logic diagram illustrating a first embodiment of a decoder unit in accordance with the present invention;

FIGS. **6A–6B** are flow diagrams illustrating a decoding process in accordance with the present invention;

FIG. **7A** is a block diagram of a subsystem for random access to a pixel or an image block in accordance with the present invention; and

FIG. **7B** is a flow diagram illustrating random access to a pixel or an image block in accordance with the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

FIG. **1** is a block diagram of a data processing system **105** constructed in accordance with the present invention. The data processing system **105** includes a processing unit **110**, a memory **115**, a storage device **120**, an input device **125**, an output device **130**, and a graphics subsystem **135**. In addition, the data processing system **105** includes a data bus **145** that couples each of the other components **110, 115, 120, 125, 130, 135** of the data processing system **105**.

The data bus **145** is a conventional data bus and while shown as a single line it may be a combination of a processor bus, a PCI bus, a graphical bus, and an ISA bus. The processing unit **110** is a conventional processing unit such as the Intel Pentium processor, Sun SPARC processor, or Motorola PowerPC processor, for example. The processing unit **110** processes data within the data processing system **105**. The memory **115**, the storage device **120**, the input device **125**, and the output device **130** are also conventional components as recognized by those skilled in the art. The memory **115** and storage device **120** store data within the data processing system **105**. The input device **125** inputs data into the system while the output device **130** receives data from the data processing system **105**.

FIG. **2A** is a block diagram of an image processing system **205** constructed in accordance with-the present invention. In one embodiment, the image processing system **205** runs within the data processing system **105**. The image processing system **205** includes an image encoder system **220** and an image decoder system **230**. The image processing system **205** may also include a unit for producing an image source **210** from which images are received, and an output **240** to which processed images are forwarded for storage or further processing. The image encoder system **220** is coupled to receive an image from the image source **210**. The image decoder system **230** is coupled to output the image produced by the image processing system **205**. The image encoder system **220** is coupled to the image decoder system **230** through a data line and may be coupled via a storage device **120** and/or a memory **115**, for example.

Within the image encoder system **220**, the image is broken down into individual blocks and processed before being forwarded to, e.g., the storage device **140**, as compressed or encoded image data. When the encoded image data is ready for further data processing, the encoded image data is forwarded to the image decoder system **230**. The image decoder system **230** receives the encoded image data and decodes it to generate an output that is a representation of the original image that was received from the image source **210**.

FIGS. **3A** and **3B** are block diagrams illustrating two separate embodiments of the image encoder system **220** of the present invention. The image encoder system **220** includes an image decomposer **315**, a header converter **321**, one or more block encoders **318** (**318***a*–**318***n*, where n is the nth encoder, n being any positive integer), and an encoded image composer **319**. The image decomposer **315** is coupled to receive an original image **310** from a source, such as the image source **210**. The image decomposer **315** is also coupled to the one or more block encoders **318** and to the header converter **321**. The header converter **321** is also coupled to the encoded image composer **319**. Each block encoder **318** is also coupled to the encoded image composer **319**. The encoded image composer **319** is coupled to the output **320**.

The image decomposer **315** receives the original image **310** and forwards information from a header of the original image **310** to the header converter **321**. The header converter **321** modifies the original header to generate a modified header, as further described below. The image decomposer **315** also breaks, or decomposes, the original image **310** into R number of image blocks, where R is some integer value. The number of image blocks an original image **310** is broken into may depend on the number of image pixels. For example, in a preferred embodiment an image **310** comprised of A image pixels by B image pixels will typically be (A/4)*(B/4) blocks, where A and B are integer values. For example, where an image is 256 pixels by 256 pixels, there will be 64×64 blocks. In other words, the image is decomposed such that each image block is 4 pixels by 4 pixels (16 pixels). Those skilled in the art will recognize that the number of pixels or the image block size may be varied, for example m×n pixels, where m and n are positive integer values.

Briefly turning to FIG. **2**B, there is illustrated an example of a single image block **260** in accordance with the present invention. The image block **260** is comprised of pixels **270**. The image block **260** may be defined as an image region W pixels **270** in width by H pixels **270** in height, where W and H are integer values. In a preferred embodiment, the image block **260** is comprised of W=4 pixels **270** by H=4 pixels **270** (4×4).

Turning back to FIGS. **3**A and **3**B, each block encoder **318** receives an image block **260** from the image decomposer **315**. Each block encoder **318** encodes or compresses each image block **260** that it receives to generate an encoded or compressed image block. Each encoded image block is received by the encoded image composer **319** which orders the encoded blocks in a data file. The data file from the encoded image composer **319** is concatenated with a modified header from the header converter **321** to generate an encoded image data file that is forwarded to the output **320**. Further, it is noted that having more than one block encoder **318***a*–**318***n* allows for encoding multiple image blocks simultaneously, one image block per block encoder **318***a*–**318***n*, within the image encoder system **220** to increase image processing efficiency and performance.

The modified header and the encoded image blocks together form the encoded image data that represents the original image **310**. The function of each element of the image encoder system **220**, including the block encoder **318**, will be further described below with respect to FIGS. 4A–4E.

The original image **310** may be in any one of a variety of formats including red-green-blue ("RGB"), YUV **420**, YUV **422**, or a proprietary color space. It may be useful in some cases to convert to a different color space before encoding the original image **310**. It is noted that in one embodiment of the present invention, each image block **260** is a 4×4 set of pixels where each pixel **270** is 24-bits in size. For each pixel **270** there are 8-bits for a Red(R)-channel, 8-bits for a Green(G)-channel, and 8-bits for a Blue(B)-channel channel in a red-green-blue ("RGB") implementation color space. Further, each encoded image block is also a 4×4 set of pixels, but, each pixel is only 2-bits in size and has an aggregate size of 4-bits as will be further described below.

FIG. **3**C is a block diagram illustrating a block encoder **318** of the present invention in greater detail. The block encoder **318** includes a color quantizer **335** and a bitmap construction module **340**. The color quantizer **335** is coupled to the bitmap construction module **340**. Further, the color quantizer **335** further emphasizes a block type module **345**, a selection module **355**, and a codeword generation module **360**. The block type module **345** is coupled to the selection module **355**. The selection module **355** is coupled to the codeword generation module **360**.

Each image block **260** of the decomposed original image **310** is received and initially processed by the color quantizer **335** before being forwarded to the bitmap construction module **340** for further processing. The bitmap construction module **340** outputs encoded image blocks for the encoded image composer **319** to order. The bitmap construction module **340** and the color quantizer **335**, including the block type module **345**, the selection module **355**, and the codeword generation module **360**, are further discussed below in FIGS. **4**A–**4**E.

Briefly, FIG. **3**D is a diagram of a data sequence or string **380** representing the original image **310** that is received by the block decomposer **315**. The data string **380** of the original image **310** includes an a-bit header **380***a* and a b-bit image data **380***b*, where a and b are integer values. The header **380***a* may include information such as the pixel width of the image **310**, the pixel height of the image **310**, and the format of the image **310**, e.g., the number of bits to the pixel in RGB or YUV format, for example, as well as other information. The image data is the data **380***b* representing the original image **310** itself.

FIG. **3**E is a diagram of a data sequence or string **385** representing encoded image data **385** that is generated and output **320** by the image encoder system **220**. The data string for the encoded image data **385** includes a modified header portion **385***a* and an encoded image block portion **390-1**–**390**-R. The modified header portion **385***a* is generated by the header converter **321** from the original header **380***a* for the original image **310**. The modified header generated by the header converter **321** includes information about file type, a number of bits per pixel of the original image **310**, addressing into the original image **310**, other miscellaneous encoding parameters, as well as the width and height information indicating the size of that original image **310**. The encoded image block portion **390-1**-R includes the encoded image blocks **390-1**–**390**-R from the block encoders **318**, where R is an integer value that is the number of blocks resulting from the decomposed original image **310**.

FIG. **3**F is a diagram of a data sequence or string **390** representing an encoded image block in accordance with the present invention. It is understood that the data string **390** representing the encoded image block may be similar to any one of the encoded image blocks **390-1**–**390**-R shown in the encoded image data string **385**.

The data string **390** of the encoded image block includes a codeword section **390***a* which includes J codewords, where J is an integer value, and a bitmap section **390***b*. The codeword section **390***a* includes J codewords **390***a* that are used to compute the colors indexed by the bitmap **390***b*. A codeword is a n-bit data string, where n is an integer value, that identifies a pixel property, for example a color component. In a preferred embodiment, there are two 16-bit codewords **390***a*, CW0, CW1 (J=2). The bitmap is a Q-bit data portion and is further discussed below in FIG. **4**B.

Further, in a preferred embodiment, each encoded image block is 64-bits, which includes two 16-bit codewords and a 32-bit (4×4×2 bit) bitmap **395**. Encoding the image block **260** as described provides greater system flexibility and increased data processing efficiency as will be further discussed below.

FIGS. **4**A–**4**E describe the operation of the image encoder system **220**. FIG. **4**A describes the general operation of the image encoder system **220**. At the start **402** of operation, data string **380** of the original image **310**, that includes the a-bit header **380***a* and the b-bit image data **380***b*, is input **404** into the block decomposer **315** from the image source **210**. The block decomposer **315** decomposes **406** the original image **310** to extract the a-bit header **380***a* and forward it to the header converter **321**. The block decomposer also **315** decomposes, **406** the original image **310** into image blocks. Each image block **260** is independently compressed, or encoded, **410** in the one or more block encoders **318**.

The header converter **321** converts **408** the a-bit header to generate a modified header **385***a*. The modified header **385***a* is forwarded to the encoded image composer **319**. Simultaneous with the header converter **321** converting **408** the α-bit header, each image block is encoded **410** by the one or more image encoders **318***a*–**318***n* to generate the encoded image blocks **390-1**–**390**-R. Again, it is noted that each image block **260** may be processed sequentially in one block encoder **318***a* or multiple image blocks **260** may be processed in parallel in multiple block encoders **318***a*–**318***n*.

The encoded image blocks **390** are output from the block encoders **318** and are placed into a predefined order by the encoded image composer **319**. In a preferred embodiment, the encoded image blocks **390** are ordered in a file from left to right and top to bottom in the same order in which they were broken down by the block decomposer **315**. The image encoder system **220** continues by composing **412** the modified header information **385***a* from the header converter **321** and the encoded image blocks **390**. Specifically, the modified header **385***a* and the ordered encoded image blocks **390** are concatenated to generate the encoded image data file **385**. The encoded image data file **385** is written **414** as encoded output **320** to the memory **115**, the storage device **120**, or the output device **130**, for example.

FIG. 4B shows the encoding process **410** for the encoder system **220** described above in FIG. **2**. At the start **418** of operation, codewords are computed **420**. As discussed above in FIG. **3**F, in a preferred embodiment there are two codewords **390***a*, CW**0**, CW**1**. The process for computing codewords is further described below in FIG. **4**C.

Once the codewords are computed **420** pixel values or properties, such as colors, for the image block **260** are computed or quantized quantized **422**. Specifically, the codewords **390***a* provide points in a pixel space from which M quantized pixel values may be inferred, where M is an integer value. The M quantized pixel values are a limited subset of pixels in a pixel space that are used to represent the current image block. The process for quantizing pixel values, and more specifically colors, will be described below in FIGS. **4**D and **4**E. Further, it is noted that the embodiments will now be described with respect to colors of a pixel value although one skilled in the art will recognize that in general any pixel value may be used with respect to the present invention.

In a preferred embodiment, each pixel is encoded with two bits of data which can index one of M quantized colors (M=4). Further, in a preferred embodiment the four quantized colors are derived from the two codewords **390***a* where two colors are the codewords themselves and the other two colors are inferred from the codewords, as will be described below. It is also possible to use the codewords **390***a* so that there is one index to indicate a transparent color and three indices to indicate colors, of which one color is inferred.

In a preferred embodiment, the bitmap **390***b* is a 32-bit data string. The bitmap **390***b* and codewords **390***a* are output **424** as a 64-bit data string representing an encoded image block **390**. Specifically, the encoded image block **390** includes the two 16-bit codewords **390***a* (n=16) and a 32-bit bitmap **390***b*. Each codeword **390***a* CW**0**, CW**1** that is a 16-bit data string includes a 5-bit red-channel, 6-bit green-channel, and 5-bit blue-channel.

Each of the encoded image blocks **390** is placed together **390***a***1**–**390***a*R, and concatenated with header information **385***a* derived from the original header **380***a* of the original image **310**. The resulting **424** output is the encoded image data **385** representing the original image **310**.

FIG. **4**C describes the process for computing **420** the codewords for the image blocks **260** in more detail. At the start **426** of the process, the color quantizer **335** uses the block type module **345** to select **428** the first block type for the image block **260** that is being processed. For example, one block type selected **428** may be a four-color and another block type selected **428** may be a three-color plus transparency, where the colors within the particular block type have equidistant spacing in a color space.

Those of ordinary skill in the art will readily recognize that selecting a block type for each image is not intended to be limiting in any way. Instead, the present invention may be limited to processing image blocks that are of a single block type. This eliminates the need to distinguish between different block types, such as the three and four color block types discussed above. Consequently, the block type module **345** in FIG. **3**B and reference number **428** in FIG. **4**C are optional and are not intended to limit the present invention in any way.

Once the block type is selected **428**, the process computes **430** an optimal analog curve for the block type. Computation **430** of the optimal analog curve **430** will be further described below in FIG. **4**D. The analog curve is used to simplify quantizing of the colors in the image block. After computing **430** the optimal analog curve, the process selects **432** a partition of the points along the analog curve. A partition may be defined as a grouping of indices {1 . . . . (W×H)} into M nonintersecting sets. In a preferred embodiment, the indices (1 . . . 16) are divided into three or four groups, or clusters, (M=3 or 4) depending on the block type.

Once a partition is selected **432**, the optimal codewords for that **20** particular partition are computed **434**. Computation **434** of the optimal codewords is further described below in FIG. **4**E. In addition to computing **434** the codewords, an error value (squared error as describe below) for the codewords is also computed **436**. Computation **436** of the error values is further described below with respect to FIG. **4**E also. If the computed **436** error value is the first error value it is stored. Otherwise, the computed **436** error value is stored **438** only if it is less than the previously stored error value. For each stored **438** error value, the corresponding block type and codewords are also stored **440**. It is noted that the process seeks to find the block type and codewords that minimize the error function.

The process continues by determining **442** if the all the possible partitions are complete. If there are more partitions possible, the process selects **432** the next partition and once again computes **434** the codewords, computes **436** the associated error value, and stores **438** the error value and stores **440** associated block type and codewords only if the error value is less than the previously stored error value.

After all the possible partitions are completed, the process determines **444** whether all the block types have been selected. If there are more block types, the process selects **428** the next block type. Once again, the process will compute **430** the optimal analog curve, select **432**, **442** all the possible partitions, for each partition it will compute **434**, **436** the codewords and associated error value, and store **438**, **440** the error value and associated block type and codeword only if the error value is less than the previously stored error value. After the last block type is processed, the process outputs **446** a result **447** of the block type and codewords **390***a* having the minimum error.

In an alternative embodiment, the optimal analog curve may be computed **430** before searching the block type. That is, the process may compute **430** the optimal analog curve before proceeding with selecting **428** the block type, selecting **432** the partition, computing **434** the codewords, computing **436** the error, storing **438** the error, and storing **440** the block type and codeword. Computing **430** the optimal analog curve first is useful if all the block types use the same analog curve and color space because the analog curve does not need to be recomputed for each block type.

FIG. 4D further describes the process of identifying the optimal analog curve. The selection module **355** starts **448** the process by computing a center of gravity **450** for pixel **270** colors of an image block **260**. Computing **450** the center of gravity includes averaging the pixel **270** colors of the image block **260**. Once the center of gravity is computed **450**, the process identifies **452** a vector in color space to minimize the first moment of the pixel **270** colors of the image block **260**.

Specifically, for identifying **452** the vector the process fits a straight line to a set of data points, which are the original pixel **270** colors of the image block **260**. A straight line is chosen passing through the center of gravity of the set of points such that it minimizes the "moment of inertia" (the means square error). For example, for three pixel properties, to compute the direction of the line minimizing the moment of inertia, tensor inertia, T, is calculated from the individual colors as follows:

$$T = \sum \begin{matrix} C_{1i}^2 + C_{2i}^2 & -C_{0i}C_{1i} & -C_{0i}C_{2i} \\ -C_{0i}C_{1i} & C_{0i}^2 + C_{2i}^2 & -C_{1i}C_{2i} \\ -C_{0i}C_{2i} & -C_{2i}C_{1i} & C_{0i}^2 + C_{1i}^2 \end{matrix}$$

where C_0, C_1, and C_2 represent pixel properties, for example color components in RGB or YUV, relative to a center of gravity. In a preferred embodiment of an RGB color space, C_{0i} is the value of red, C_{1i} is the value of green, and C_{2i} is the value of blue for each pixel, i, of the image block. Further, i takes on integer values from 1 to W×H, so that if W=4 and H=4, i ranges from 1 to 16.

The eigenvector of tensor, T, with the smallest eigenvalue is calculated using conventional methods known to those skilled in the art. The eigenvector direction along with the calculated gravity center, defines the axis that minimizes the moment of inertia. This axis is used as the optimal analog curve, which in a preferred embodiment is a straight line. Those of ordinary skill in the art will readily recognize that the term optimal analog curve is not limited solely to a straight line but may include a set of parameters, such as pixel values or colors, that minimizes the moment of inertia or means square error when fitted to the center of gravity of the pixel colors in the image block. The set of parameters may define any geometric element, such as but not limited to a curve, a plane, a trapezoid, or the like.

FIG. 4E illustrates the process undertaken by the codeword generation module **360** for selecting **432** the partitions, computing **434**, **436** the codewords for the partitions and the

associated error, and storing **438**, **440** the error value, block type, and codeword if the error value is less than a previously stored error value. The process starts **456** with the codeword generation module **360** projecting **458** the W×H color values onto the previously constructed optimal analog curve. The value of W×H is the size in number of pixels **270** of an image block **260**. In a preferred embodiment, where Wand Hare both **4** pixels, W×H is 16 pixels.

Once the colors are projected **458** onto the analog curve, the colors are ordered **460** sequentially along that analog curve based on the position of the color on the one-dimensional analog curve. After the colors are ordered **460**, the codeword generation module **360** searches **462** for optimal partitions. That is, the codeword generation module **360** takes the W×H colors (one color associated with each pixel) that are ordered **460** along the analog curve and partitions, or groups, them into a finite number of clusters with a predefined relative spacing. In a preferred embodiment, where W=4 and H=4, so that W×H is 16, the 16 colors are placed in three or four clusters (M=3 or 4).

In conducting the search **462** for the optimal partition, the color selection module **360** finds the best M clusters for the W×H points projected onto the optimal curve, so that the error associated with the selection is minimized. The best M clusters are determined by minimizing the mean square error with the constraint that the points associated with each cluster are spaced to conform to the predefined spacing.

In a preferred embodiment, for a block type of four equidistant colors, the error may be defined as a squared error along the analog curve, such as

$$E^2 = \Sigma_{cluster0}(x_i - p_0)^2 + \Sigma_{cluster1}(x_i - ((2/3)p_0 + (1/3)p_1))^2 + \Sigma_{cluster2}(x_i - ((1/3)p_0 + (2/3)p_1))^2 + \Sigma_{cluster3}(x_i - p_1)^2$$

where E is the error for the particular grouping or clustering, p_0 and p_1 are the coded colors, and x_i are the projected points on the optimal analog curve.

In instances where the block type indicates three equidistant colors, the error may be defined as a squared error along the analog curve, such as

$$E^2 = \Sigma_{cluster0}(x_i - p_0)^2 + \Sigma_{cluster1}(x_i - ((1/2)p_0 + (1/2)p_1))^2 + \Sigma_{cluster2}(x_i - p_1)^2$$

where, again, E is the error for the particular grouping or clustering, p_0 and p_1 are the coded colors, and x_i are the projected points on the optimal analog curve.

After the resulting **447** optimal codewords **390***a* are identified, they are forwarded to the bitmap construction module **340**. The bitmap construction module **340** uses the codewords **390***a* to identify the M colors that may be specified or inferred from those codewords **390***a*. In a preferred embodiment, the bitmap construction module **340** uses the codewords **390***a*, e.g., **CW0**, **CW1**, to identify the three or four colors that may be specified or inferred from those codewords **390***a*.

The bitmap construction module **340** constructs a block bitmap **390***b* using the codewords **390***a* associated with the image block **260**. Colors in the image block **260** are mapped to the closest color associated with one of the quantized colors specified by, or inferred from, the codewords **390***a*. The result is a color index, referenced as ID, per pixel in the block identifying the associated quantized color.

Information indicating the block type is implied by the codewords **390***a* and the bitmap **390***b*. In a preferred embodiment, the order of the codewords **390***a* **CW0**, **CW1**, indicate the block type. If a numerical value of **CW0** is greater than a numerical value of **CW1**, the image block is a four color block. Otherwise, the block is a three color plus transparency block.

As discussed above, in a preferred embodiment, there are two image block types. One image block type has four equidistant colors, while the other image block type has three equidistant colors with the fourth color index used to specify that a pixel is transparent. For both image block types the color index is two bits.

The output of the bitmap construction module **340** is an encoded image block **390** having the M codewords **390***a* plus the bitmap **390***b*. Each encoded image block **390** is received by the encoded image composer **319** that, in turn, orders the encoded image blocks **390** in a file. In a preferred embodiment, the encoded image blocks **390** are ordered from left to right and from top to bottom in the same order as the blocks were broken down by the block decomposer **315**. The ordered file having the encoded image blocks **390** is concatenated with the header information **385***a* that is derived from the header **380***a* of the original image **310** to generate the encoded image data **385** that is the image encoder system **220** output **320**. The image encoder system **220** output **320** may be forwarded to the memory **115**, the storage device **120**, or the output device **130**, for example.

The image encoder system **220** of the present invention advantageously reduces the effective data size of an image, for example, from 24-bits per pixel to 4-bits per pixel. Further, the present invention beneficially addresses transparency issues by allowing for codewords to be used with a transparency identifier.

FIG. 5A is a block diagram of an image decoder system **230** in accordance with the present invention. The image decoder system **230** includes an encoded image decomposing unit **501**, a header converter **508**, one or more block decoders **505** (**505***a*–**505***m*, where m is any positive integer value representing the last block decoder), and an image composer **504**. The encoded image decomposer **501** is coupled to receive the encoded image data **385** that was output **320** from the image encoder system **220**. The encoded image decomposer **501** is coupled to the one or more block decoders **505***a*–**505***m*. The one or more block decoders **505***a*–**505***m* are coupled to the image composer **504** that, in turn, is coupled to the output **240**.

The encoded image decomposer **501** receives the encoded image data **385** and decomposes, or breaks, it into its header **385***a* and the encoded image blocks **390-1** –**390**-R. The encoded image decomposer **501** reads the modified header **385***a* of the encoded image data **385** and forwards the modified header **385***a* to the header converter **508**. The encoded image decomposer **501** also decomposes the encoded image data **385** into the individual encoded image blocks **390-1**–**390**-R that are forwarded to the one or more block decoders **505***a*–**505***m*.

The header converter **508** converts the modified header **385***a* to an output header. Simultaneously, the encoded image blocks **390-1**–**390**-R are decompressed or decoded by the one or more block decoders **505***a*–**505***m*. It is noted that the each encoded image block **390** may be processed sequentially in one block decoder **505***a* or multiple encoded image blocks **390-1**–**390**-R may be processed in parallel with one block decoder **505***a*–**505***m* for each encoded image block **390**-–**390**-R. Thus, multiple block decoders **505***a*–**505***m* allows for parallel processing that increases the processing performance and efficiency of the image decoder system **230**.

The image composer **504** receives each decoded image block from the one or more block decoders **505***a*–**505***m* and orders them in a file. Further, the image composer **504** receives the converted header from the header converter **508**. The converted header and the decoded image blocks are placed together to generate output **240** data representing the original image **310**.

FIG. 5B is a block diagram of a first embodiment of a block decoder **505** in accordance with the present invention. Each block decoder **505***a*–**505***m* includes a block type detector **520**, one or more decoder units, e.g., **533***a–l* to **533***a–k* (k is any integer value), and an output selector **523**. The block type detector **520** is coupled to the encoded image decomposer **501**, the output selector **523**, and each of the one or more decoder units, e.g., **533***a–l*–**533***a–k*. Each of the decoder units, e.g., **533***a–l*–**533***a–k*, is coupled to the output selector **523** that, in turn, is coupled to the image composer **504**.

The block type detector **520** receives the encoded image blocks **390** and determines the block type for each encoded image block **390**. Specifically, the block type detector **520** passes a selector signal to the output selector **523** that will be used to select an output corresponding to the block type detected. The block type is detected based on the codewords **390***a*. After the block type is determined, the encoded image blocks **390** are passed to each of the decoder units, e.g., **533***a–l*–**533***a–k*. The decoder units, e.g., **533***a–l*–**533***a–k*, decompress or decode each encoded image block **390** to generate the colors for the particular encoded image block **390**. The decoder units, e.g., **533***a–l*–**53***a–k*, may be c-channels wide (one channel for each color component (or pixel property) being encoded), where c is any integer value. Using the selector signal, the block type detector **520** enables the output selector **523** to output the color of the encoded image block **390** from one of the decoder units, e.g., **533***a–l*–**533***a–k* that corresponds with the block type detected by the block type detector **520**. Alternatively, using the selector signal, the appropriate decoder unit **533** could be selected so the encoded block is processed through that decoder unit only.

FIG. 5C is a block diagram of a second embodiment of a block decoder **505** in accordance with the present invention. In a second embodiment, the block decoder **505** includes a block type detector **520**, a first and a second decoder unit **530**, **540**, and the output selector **523**. The block type detector **520** is coupled to receive the encoded image blocks **390** and is coupled to the first and the second decoder units **530**, **540** and the output selector **523**.

The block type detector **520** receives the encoded image blocks **390** and determines, by comparing the codewords **390***a* of the encoded image block **390**, the block type for each encoded image block **390**. For example, in a preferred embodiment, the block type is four quantized colors or three quantized colors and a transparency. Once the block type is selected and a selector signal is forwarded to the output selector **523**, the encoded image blocks **390** are decoded by the first and the second decoder units **530**, **540**. The first and the second decoder units **530**, **540** decode the encoded image block **390** to produce the pixel colors of each image block. The output selector **523** is enabled by the block type detector **520** to output the colors from the decoder unit **530**, **540** that corresponds to the block type selected.

FIG. 5D is a logic diagram illustrating one embodiment of a decoder unit through a red-channel of the that decoder unit in accordance with the present invention. Specifically, the decoder unit is similar to the decoder units **530**, **540** illustrated in FIG. 5C. Moreover, the functionality of each of those decoder units **530**, **540** is merged into the single logic diagram illustrated in FIG. 5D. Further, those skilled in the art will understand that although described with respect to the red-channel of the decoder units **530**, **540** the remaining channels, e.g., the green-channel and the blue-channel, in each decoder unit **530**, **540** are similarly coupled and functionally equivalent.

The logic diagram illustrating the decoder units **530, 540** is shown to include portions of the block type detector **520**, for example a comparator unit **522**. The comparator unit **522** works with a first 2×1 multiplexer **525***a* and a second 2×1 multiplexer **525***b*. The comparator unit **522** is coupled to the first and the second 2×1 multiplexers **525***a*, **525***b*. Both 2×1 multiplexers **525***a*, **525***b* are coupled to a 4×1 multiplexer **526** that serves to select the appropriate color to output.

The red-channel **544**, **546** of the first decoder unit **530** includes a first and a second red-channel line **551***a*, **551***b* and a first and a second red-color block **550***a*, **550***b*. Along the path of each red-color block **550***a*, **550***b* is a first full adder **552***a*, **552***b*, a second full adder **554***a*, **554***b*, and a CLA ("carry-look ahead") adder **556***a*, **556***b*. The first and the second red-channel lines **551***a*, **551***b* are coupled to the first and the second red-color blocks **550***a*, **550***b*, respectively. Each red-color block **550***a*, **550***b* is coupled to the first full adder **552***a*, **552***b* associated with that red-color block **550***a*, **550***b*. Each first full adder **552***a*, **552***b* is coupled to the respective second full adder **554***a*, **554***b*. Each second full adder **554***a*, **554***b* is coupled to the respective CLA adder **556***a*, **556***b*.

The second decoder unit **540** comprises the first and the second red-channel lines **551***a*, **551***b* and the respective first and second red-color blocks **550***a*, **550***b* and an adder **558**. The first and the second channel lines **551***a*, **551***b* are coupled to their respective red-color blocks **550***a*, **550***b* as described above. Each red-color block **550***a*, **550***b* is coupled to the adder **558**.

The CLA adder **556***a* from the path of the first red-color block **550***a* of the first decoder unit **530** is coupled to the first 2×1 multiplexer **525***a* and the CLA adder **556***b* from the path of the second red-color block **550***b* of the first decoder unit **530** is coupled to the second 2×1 multiplexer **525***b*. The adder **558** of the second decoder unit **540** is coupled to both the first and the second 2×1 multiplexers **525***a*, **525***b*.

The 4×1 multiplexer **526** is coupled to the first and the second red-channel lines **551***a*, **551***b*, as well as to the first and the second 2×1 multiplexers **525***a*, **525***b*. The 4×1 multiplexer **526** is also coupled to receive a transparency indicator signal that indicates whether or not a transparency (no color) is being sent. The 4×1 multiplexer **526** selects a color for output based on the value of the color index, referenced as the ID signal, that references the associated quantized color for an individual pixel of the encoded image block **390**.

FIG. **6**A is a flow diagram illustrating operation of the decoder system **230** in accordance with the present invention. For purposes of illustration only, the process for the decoder system **230** will be described with a single block encoder **505** having two decoding units, e.g., **530**, **540**. Those skilled in the art will recognize that the process is functionally equivalent for decoder systems having more than one block decoder **505** and more than one decoder units, e.g., **533***a*–*l*–**533***a*–*k*.

The process starts **600** with the encoded image decomposer **501** receiving **605** the encoded, or compressed, image data **385** from the encoder system **220**, for example, through the memory **115** or the storage device **120**. The encoded image decomposer **501** decomposes **610** the encoded image data **385** by forwarding the modified header **385***a* to the header converter **508**. In addition, the encoded image decomposer **501** also decomposes **610** the encoded image data **385** into the individual encoded image blocks **390-1**–**390**-R.

The header converter **508** converts **612** the header information to generate an output header that is forwarded to the image composer **504**. Simultaneously, the one or more block decoders **505***a*–**505***m* decodes **615** the pixel colors for each encoded image block **390**. It is again noted that each encoded image block **390** may be decoded **615** sequentially in one block decoder **505***a* or multiple encoded image blocks **390-1**–**390**-R may be decoded **615** in parallel in multiple block decoders **505***a*–**505***m*, as described above. The process for decoding the encoded image blocks **390** is further described in FIG. 6B. Each decoded **615** image block is then composed **620** into a data file with the converted **612** header information by the image composer **504**. The image composer **504** generates the data file as an output **625** that represents the original image **310**.

FIG. 6B is a flow diagram illustrating operation of the block encoder **505** in accordance with the present invention. Once the process is started **630**, each encoded image block **390** is received by the block decoder **505** and the block type for each encoded image block **390** is detected **640**. Specifically, for a preferred embodiment the first and the second codewords **390***a*, CW**0**, CW**1**, respectively, are received **635** by the block type detector **520** of the block decoder **505**. As discussed above, comparing the numerical values of CW**0** and CW**1** reveals the block type.

In addition, the first five bits of each codeword **390***a*, e.g., CW**0**, CW**1**, that represent the red-channel color are received by the red-channel **545** of each of the first and the second decoder units **530**, **540**, the second 6-bits of each codeword **390***a* CW**0**, CW**1** that represent the green-channel color are received by the green-channel of each of the first and the second decoder units **530, 540**, and the last 5-bits of each codeword **390***a* CW**0**, CW**1** that represent the blue-channel color are received by the blue-channel of each of the first and the second decoder units **530, 540**.

The block type detector **520** detects **640** the block type for an encoded image block **390**. Specifically, the comparator **522** compares the first and the second codewords **390***a*, CW**0**, CW**1**, and generates a flag signal to enable the first 2×1 multiplexers **525***a* or the second 2×1 multiplexers **525***b* which, in turn, selects **645** either the first decoding unit **530** or the second decoding unit **540**, respectively. The process then calculates **650** the quantized color levels for the decoder units **530, 540**.

To calculate **650** the quantized color levels, the first decoding unit **530** calculates the four colors associated with the two codewords **390***a*, CW**0**, CW**1**, using the following relationship:

CW**0**=first codeword=first color;

CW**1**=second codeword=second color;

CW**2**=third color=($\frac{2}{3}$)CW**0**+($\frac{1}{3}$)CW**1**;

CW**3**=fourth color=($\frac{1}{3}$)CW**0**+($\frac{2}{3}$)CW**1.**

In one embodiment, the first decoder unit **530** may estimate the above equations for CW**2** and CW**3**, for example, as follows:

CW**2**=($\frac{5}{8}$)CW**0**+($\frac{3}{8}$)CW**1**; and

CW**3**=($\frac{3}{8}$)CW**0**+($\frac{5}{8}$)CW**1**.

The red-color blocks **550***a*, **550***b* serve as a one-bit shift register to get ($\frac{1}{2}$)CW**0** or ($\frac{1}{2}$)CW**1** and each full adder **552***a*, **552***b*, **554***a*, **554***b* also serves to shift the signal left by 1-bit. Thus, the signal from the first full adders **552***a*, **552***b* is ($\frac{1}{4}$)CW**0** or ($\frac{1}{4}$)CW**1**, respectively, because of a two-bit overall shift and the signal from the second full adders **554***a*, **554***b* is ($\frac{1}{8}$)CW**0** or ($\frac{1}{8}$)CW**1**, respectively, because of a three-bit overall shift. These values allow for the above approximations for the color signals.

The second decoder unit **540** calculates 650 three colors associated with the codewords **390***a*, CW**0**, CW**1**, and includes a fourth signal that indicates a transparency is being passed. The second decoder unit **540** calculates colors, for example, as:

CW**0**=first codeword=first color;

CW**1**=second codeword=second color;

CW**3**=third color=(½)CW**0**+(½)CW**1**; and

T=Transparency.

In one embodiment the second decoder unit **540** has no approximation because the signals received from the red-color blocks **550***a*, **550***b* is shifted left by one-bit so that the color is already calculated to (½)CW**0** and (½)CW**1**, respectively.

After the quantized color levels for the selected **645** decoder unit **530**, **540** have been calculated **650**, each bitmap value for each pixel is read **655** from the encoded image data block **385**. As each index is read **655** it is mapped **660** to one of the four calculated colors if the first decoder unit **530** is selected **645** or one of the three colors and transparency if the second decoder unit **540** is selected. The mapped **660** colors are selected by the 4×1 multiplexer **526** based on the value of the ID signal from the bitmap **390***b* of the encoded image block **390**. As stated previously, a similar process occurs for selection of colors in the green-channel and the blue-channel.

As the colors are output from the red-, green-, and blue-channels, the output is received by the image composer **504**. The image composer **504** orders the output from the block encoders **505** in the same order as the original image **310** was decomposed. The resulting **665** image that is output from the image decoder system **230** is the original image that is forwarded to an output source **240**, e.g., a computer screen, which displays that image.

The system and method of the present invention beneficially allows for random access to any desired image block **260** within an image, and any pixel **270** within an image block **260**. FIG. **7**A is a block diagram of a subsystem **700** that provides random access to a pixel **270** or an image block **260** in accordance with the present invention.

The random access subsystem **700** includes a block address computation module **710**, a block fetching module **720**, and the one or more block decoders **505**. The block address computation module **710** is coupled to receive the header information **385***a* of the encoded image data **385**. The block address computation module **710** is also coupled to the block fetching module **720**. The block fetching module **720** is coupled to receive the encoded image block portion **390-1**-R of the encoded image data **385**. The block fetching module **720** is also coupled to the block decoders **505**.

FIG. **7**B is a flow diagram illustrating a process of random access to a pixel **270** or an image block **260** using the random access subsystem **700** in accordance with the present invention. When particular pixels **270** have been identified for decoding, the process starts **740** with the image decoder system **230** receiving the encoded image data **385**. The modified header **385***a* of the encoded image data **385** is forwarded to the block address computation module **710** and the encoded image block portion **390-1**-R of the encoded image data **385** is forwarded to the block fetching module **720**.

The block address computation module **710** reads the modified header **385***a* to compute **745** the address of the encoded image block portion **390-1**-R having the desired pixels **270**. The address computed **745** is dependent upon the pixel coordinates within an image. Using, the computed **745** address, the block fetching module **720** identifies the encoded image block **390** of the encoded image block portion **390-1**-R that has the desired pixels **270**. Once the encoded image block **390** having the desired pixels **270** has been identified, only the identified encoded image block **390** is forwarded to the block decoders **505** for processing.

Similar to the process described above in FIG. **6**B, the block decoders **505** compute **755** the quantized color levels for the identified encoded image blocks **390** having the desired pixels. After the quantized color levels have been computed **755**, the color of the desired pixel is selected **760** and output **765** from the image decoder system **230**.

Random access to pixels **270** of an image block **260** advantageously allows for selective decoding of only needed portions or sections of an image. Random access also allows the image to be decoded in any order the data is required. For example, in three-dimensional texture mapping only portions of the texture may be required and these portions will generally be required in some non-sequential order. Thus, the present invention increases processing efficiency and performance when processing only a portion or section of an image.

The present invention beneficially encodes, or compresses, the size of an original image **310** from 24-bits per pixel to an aggregate 4-bits per pixel and then decodes, or decompresses the encoded image data **385** to get a representation of the original image **310**. Further, the claimed invention uses, for example, two base points or codewords from which additional colors are derived so that extra bits are not necessary to identify a pixel **270** color.

Moreover, the present invention advantageously accomplishes the data compression on an individual block basis with the same number of bits per block so that the compression rate can remain fixed. Further, because the blocks are of fixed size with a fixed number of pixels **270**, the present invention beneficially allows for random access to any particular pixel **270** in the block. The present invention provides for an efficient use of system resources because entire blocks of data are not retrieved and decoded to display data corresponding to only a few pixels **270**.

In addition, the use of a fixed-rate 64-bit data blocks in the present invention provides the advantage of having simplified header information that allows for faster processing of individual data blocks. Also, a 64-bit data block allows for data blocks to be processed rapidly, e.g., within one-clock cycle, as the need to wait until a full data string is assembled is eliminated. Further, the present invention also reduces the microchip space necessary for a decoder system because the decoder system only needs to decode each pixel to a set of colors determined by, e.g., the two codewords.

While particular embodiments and applications of the present invention have been illustrated and described, it is to be understood that the invention is not limited to the precise construction and components disclosed herein and that various modifications, changes and variations which will be apparent to those skilled in the art may be made in the arrangement, operation and details of the method and apparatus of the present invention disclosed herein without departing from the spirit and scope of the invention as defined in the appended claims.

What is claimed is:

1. A system for encoding an image, comprising:

an image decomposer, coupled to receive an image, for breaking the image into one or more image blocks, each image block having a set of colors;

at least one block encoder for receiving each image block and for compressing each image block to generate an encoded image block, wherein each block encoder includes a color quantizer for receiving each image block and for generating at least one codeword from which at least one quantized color is derived, the color quantizer having a selection module for computing a set of parameters from the set of colors, the at least one codeword derived from the set of parameters; and

an encoded image composer for receiving and ordering the encoded image blocks into a data file.

2. The system of claim **1**, further comprising a header converter, coupled to the image decomposer and the encoded image composer, for receiving a header from the image, modifying the header, and outputting the modified header with the data file.

3. The system of claim **1**, wherein each block encoder comprises:

a bitmap construction module for mapping the colors of an image block to one of the at least one quantized colors.

4. The system of claim **3**, wherein the color quantizer further comprises:

a block type module, coupled to receive the image block, for selecting a block type for the image block; and

a codeword generation module for generating the least one codeword from the set of parameters generated by the selection module.

5. A system for decoding a compressed image, comprising:

an encoded image decomposer, coupled to receive encoded image data file having at least one compressed image block, for breaking the encoded image data file into individual compressed image blocks, each compressed image block having at least one associated codeword, each codeword generated by computing a set of parameters, partitioning the set of parameters into a plurality of partitions, and computing each codeword from one of the partitions;

at least one block decoder for decompressing the compressed image blocks into decompressed image blocks; and

an image composer for ordering the decompressed image blocks in an output file.

6. The system of claim **5**, further comprising a header converter, coupled to the encoded image decomposer and the image composer, for receiving a modified header associated with the encoded image data file, generating an output header, and outputting the output header with the output file.

7. The system of claim **6**, wherein each block decoder further comprises:

a block type detector for selecting a block type for each compressed image block received from the encoded image decomposer;

at least one decoder unit for decompressing each compressed image block based on the block type selected by the block type detector; and

an output selector for outputting the image block from the decoder unit in response to the block type selected by the block type detector.

8. A method for generating an encoded image of an original image having a header, comprising:

converting the header to a modified header;

decomposing the original image into image blocks, each image block having a set of colors;

encoding each image block to generate an encoded image block for each image block by computing a set of codewords from the set of colors, computing a set of computed colors using the set of codewords, and mapping each original color to one of the computed colors or one of the codewords to produce an index for each original color; and

composing the modified header and each encoded image block in a file to generate the encoded image.

9. The method of claim **8**, wherein computing the set of codewords further comprises:

selecting a block type for each image block, wherein the goemetric element is computed using the block type;

partitioning the set of parameters into a plurality of partitions;

computing a set of codewords for each partition in the plurality of partitions;

computing an error for each computed set of codewords; and

outputting the block type and set of codewords producing the minimum computed error for each computed set of codewords.

10. A method for generating an original image from an encoded image including a modified header and at least one encoded image block, comprising:

receiving the encoded image data;

decomposing the encoded image into the modified header and the individual encoded image blocks;

reading the modified header to generate an output header;

decoding each individual encoded image block to generate a decoded image block, each individual encoded image block having a set of codewords and a set of indices;

calculating at least one quantized color level for the encoded image block using the set of codewords;

mapping at least one index from the set of indices to one of the calculated quantized color levels or to a codeword from the set of codewords; and

composing the output header and the individual decoded image blocks to generate an output file of the original image.

11. A system for processing any identified pixel from an encoded image data file having header information, including at least once codeword computed from a set of parameters, the set of parameters computed from a set of colors within an original image block, and an encoded image block portion including at least one encoded image block, the system comprising:

a block address computation module, coupled to receive each codeword from the header information, for computing an address of an encoded image block having the identified pixel;

a block fetching module, coupled to receive the encoded image block portion and the computed address, for fetching the encoded image block having the identified pixel; and

a block decoder, coupled to receive the fetched encoded image block, for decoding the image block to generate a quantized color associated with the identified pixel.

12. A method for processing any identified pixel of an encoded image data file having a header, including at least once codeword computed from a set of parameters, the set of parameters computed from a set of colors within an original image block, and an encoded image block portion including at least one encoded image block, the method comprising:

computing an address for an encoded image block having the identified pixel, the address computed from the at least one codeword for the encoded image block;

fetching the encoded image block using the computed address;

computing quantized color levels for the fetched encoded image block; and selecting a color of the identified pixel from the quantized color levels to output.

13. A method of compressing an original image block having a first set of color points defined within a selected color space, comprising:

fitting a geometric element to the first set of color points so that the geometric element includes a second set of color points having a minimal moment of inertia when fitted to the center of gravity of the first set of color points;

computing a set of codewords from the second set of color points;

computing a set of computed colors using the set of codewords;

mapping each of the first set of color points to one of the computed colors or one of the codewords to produce an index for each of the first set of color points; and

using the indices produced by the mapping each of the first set of color points and the set of codewords to represent the first set of color points.

14. The method of claim **13**, wherein the set of parameters defines at least two color points in the selected color space.

15. The method of claim **13**, further including generating an encoded image block having the set of codewords and the indices produced in mapping the first set of color points.

16. The method of claim **13**, wherein mapping further includes mapping a first set color point to a predefined index, if the first set color point represents an alpha value.

17. The method of claim **13**, wherein mapping further includes mapping a first set color point to a predefined index, if the first set color point represents a color key value.

18. A method of compressing an original image having a set of pixel parameters, each pixel parameter including a color point parameter defined within an RGB color space, comprising:

dividing the original image into at least one block of pixel parameters;

identifying a block type of the at least one block of pixel parameters;

computing a center of gravity for a set of color point parameters associated with the block of pixel parameters;

fitting a geometric element to the set of color point parameters associated with the block of pixel parameters so that the geometric element includes a subset of color point parameters having a minimal moment of inertia when fitted to the center of gravity;

computing a set of codewords from the subset of color point parameters;

computing a set of computed color point parameters using the set of codewords;

mapping each of the pixel parameters within the block of pixel parameters to one of the computed color point parameters or to one of the codewords to produce an index for each of the pixel parameters within the block of pixel parameters; and

representing the block of pixel parameters by using the set of codewords, and the block type, and each index produced by mapping.

19. The method of claim **18**, wherein mapping further includes mapping a pixel parameter within the block of pixel parameters to a predefined index, if the pixel parameter represents a transparency identifier.

20. The method of claim **18**, wherein mapping further includes mapping a pixel parameter within the block of pixel parameters to a predefined index, if the pixel parameter represents an alpha value.

21. The method of claim **18**, wherein mapping further includes mapping a pixel parameter within the block of pixel parameters to a predefined index, if the pixel parameter represents a color key value.

22. A method of reducing a number of original colors in an image block to at least three different colors and a bitmap table, the method comprising:

selecting a geometric element;

fitting the geometric element to the original colors so that the geometric element includes a set of colors having a minimal moment of inertia when fitted to the center of gravity of the original colors;

computing a set of codewords from the set of colors;

computing a set of computed colors using the set of codewords; and

generating the bitmap table by mapping each original color to one of the at least three different colors.

* * * * *

Exhibit B

US006683978B1

(12) **United States Patent**
Iourcha et al.

(10) **Patent No.:** **US 6,683,978 B1**
(45) **Date of Patent:** **Jan. 27, 2004**

(54) **FIXED-RATE BLOCK-BASED IMAGE COMPRESSION WITH INFERRED PIXEL VALUES**

(75) Inventors: **Konstantine I. Iourcha**, San Jose, CA (US); **Krishna S. Nayak**, Stanford, CA (US); **Zhou Hong**, San Jose, CA (US)

(73) Assignee: **S3 Graphics Co., Ltd.**, Grand Cayman (KY)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/442,114**

(22) Filed: **Nov. 17, 1999**

Related U.S. Application Data

(63) Continuation of application No. 09/351,930, filed on Jul. 12, 1999, which is a continuation of application No. 08/942,860, filed on Oct. 2, 1997, now Pat. No. 5,956,431.

(51) **Int. Cl.**^{7} **G06K 9/00**

(52) **U.S. Cl.** **382/166**; 382/232; 725/146

(58) **Field of Search** 382/166, 239, 382/253, 232; 345/550; 725/146; 358/462, 1.15; 348/63

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,821,208 A	* 4/1989	Ryan et al.	....	345/550
4,887,151 A	12/1989	Wataya	....	358/539
5,734,744 A	3/1998	Wittenstein et al.	....	382/166
5,742,892 A	* 4/1998	Chaddha	....	725/146
5,748,904 A	5/1998	Huang et al.	....	345/544
5,787,192 A	7/1998	Takaichi et al.	....	382/166
5,822,465 A	10/1998	Normile et al.	....	382/253
5,956,425 A	9/1999	Yoshida	....	382/234
5,956,431 A	9/1999	Iourcha et al.	....	382/253
6,075,619 A	6/2000	Iizuka	....	382/166

FOREIGN PATENT DOCUMENTS

JP	405216993	8/1993	.... G06F/15/70

OTHER PUBLICATIONS

A. Schilling et al., "Texram: A Smart Memory for Texturing", IEEE Computer Graphics & Applications, May 1996, 16(3), pp. 9–19.

G. Knittel et al., "Hardware and Software for Superior Texture Performance"In 10, Eurographics Hardware Workshop, Maastricht, NL, Aug. 28–29 1995, pp. 1–8.

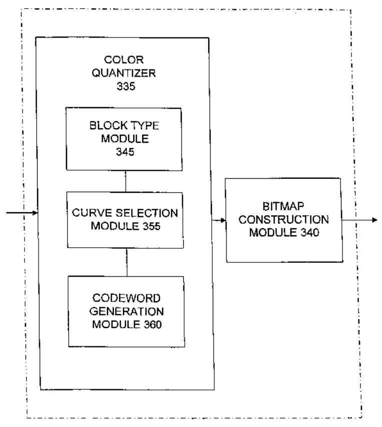
(List continued on next page.)

Primary Examiner—Anh Hong Do
(74) *Attorney, Agent, or Firm*—Carr & Ferrell LLP

(57) **ABSTRACT**

An image processing system includes an image encoder system and a image decoder system that are coupled together. The image encoder system includes a block decomposer and a block encoder that are coupled together. The block encoder includes a color quantizer and a bitmap construction module. The block decomposer breaks an original image into blocks. Each block is then processed by the block encoder. Specifically, the color quantizer selects some number of base points, or codewords, that serve as reference pixel values, such as colors, from which quantized pixel values are derived. The bitmap construction module then maps each pixel colors to one of the derived quantized colors. The codewords and bitmap are output as encoded image blocks. The decoder system includes a block decoder. The block decoder includes a block type detector, one or more decoder units, and an output selector. Using the codewords of the encoded data blocks, the comparator and the decoder units determine the quantized colors for the encoded image block and map each pixel to one of the quantized colors. The output selector outputs the appropriate color, which is ordered in an image composer with the other decoded blocks to output an image representative of the original image. A method for encoding an original image and for decoding the encoded image to generate a representation of the original image is also disclosed.

29 Claims, 16 Drawing Sheets


COLOR QUANTIZER 335
BLOCK TYPE MODULE 345
CURVE SELECTION MODULE 355
CODEWORD GENERATION MODULE 360
BITMAP CONSTRUCTION MODULE 340

318

OTHER PUBLICATIONS

G. Campbell et al., "Two Bit/Pixel Full Color Encoding", Computer Graphics (Proc. SIGGRAPH '86), Aug. 18–22, 1986, vol. 20, No. 4, Dallas, TX, pp. 215–219.

Yang, Ching Yung et al., "Hybrid Adaptive Block Truncation Coding for Image Compression," Optical Engineering, Soc. of Photo–Optical Instrumentation Engineers, Apr. 1, 1997, p. 1021–1027, vol. 36, No. 4, Bellingham, WA, USA.

Kugler, A. "High–Performance Texture Decompression Hardware," Visual Computer, Springer–Verlag, 1997, p. 51–63, vol. 13, No. 2, Berlin, Germany.

Nasiopoulos, Panos, et al., "Adaptive Compression Coding," IEEE Transactions on Communications, IEEE Inc., Aug. 1, 1991, p. 1245–1254, vol. 39, No. 8, New York, USA.

Knittel, G. et al., "Hardware for Superior Texture Performance," Eurographics Workshop on Graphics Hardware, Jul. 28, 1995, p 33–40.

Campbell, G. et al., "Two Bit/Pixel Full Color Encoding," Computer Graphics, Aug. 1986, p. 215–219, vol. 20, No. 4, New York, New York, USA.

Delp E. J. et al., "Image Compression Using Block Truncation Coding," IEEE Transactions on Communcations, Sep. 1979, p. 1335–1342, vol. COM–27, No. 9, IEEE Inc., New York, USA.

Yang, Ching Yung et al., "Use of Radius Weighted Mean to Cluster Two–Class Data," Electronics Letters, May 12, 1994, p. 757–759, vol. 30, No. 10, IEE Stevenage, Great Britain.

Russ, J. C., "Optimal Grey Scale Images From Multiplane Color Images," Journal of Computer–Assisted Microscopy, Dec. 1995, p. 221–223, vol. 7, No. 4, Plenum, USA.

Knittel, G., et al., "Hardware and Software for Superior Texture Performance," Eurographics Hardware Wokshop '95, Aug. 28–29, 1995, pp. 1–8, Masstricht, NL.

Schilling, A., et al., "Texram: A Smart Memory for Texturing," IEEE Computer Graphics & Applications, May 1996, pp. 9–19, vol. 16, No. 3.

Feng et al., "A Dynamic Address Vector Quantization Algorithm Based on Inter–Block and Inter–Color Correction for Color Image Coding," IEEE International conference on Acoustics, Speech, and Signal Processing, May 1989, pp. 1755–1758, vol. 3.

* cited by examiner

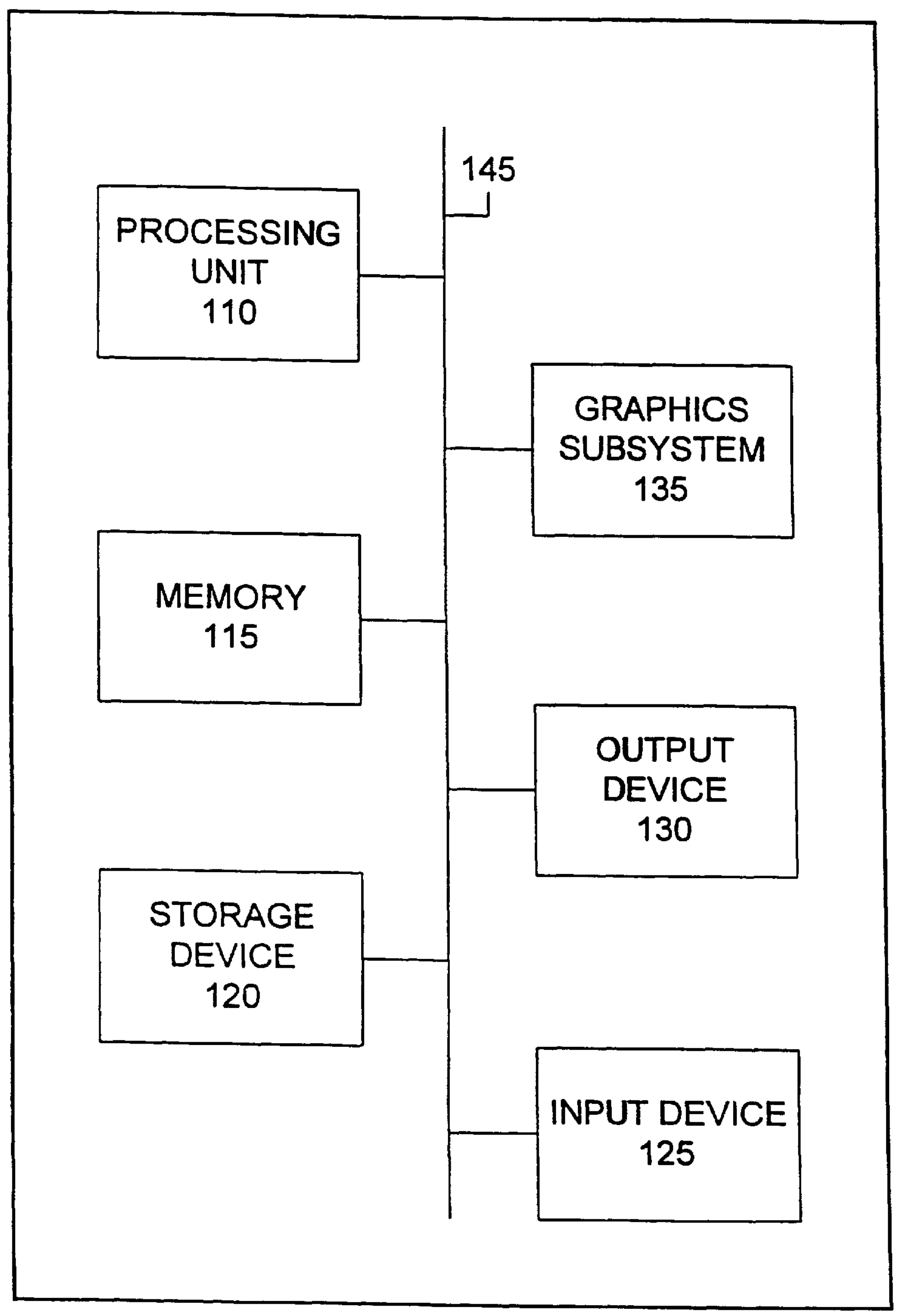

145
PROCESSING UNIT 110
GRAPHICS SUBSYSTEM 135
MEMORY 115
OUTPUT DEVICE 130
STORAGE DEVICE 120
INPUT DEVICE 125

FIG. 1

105

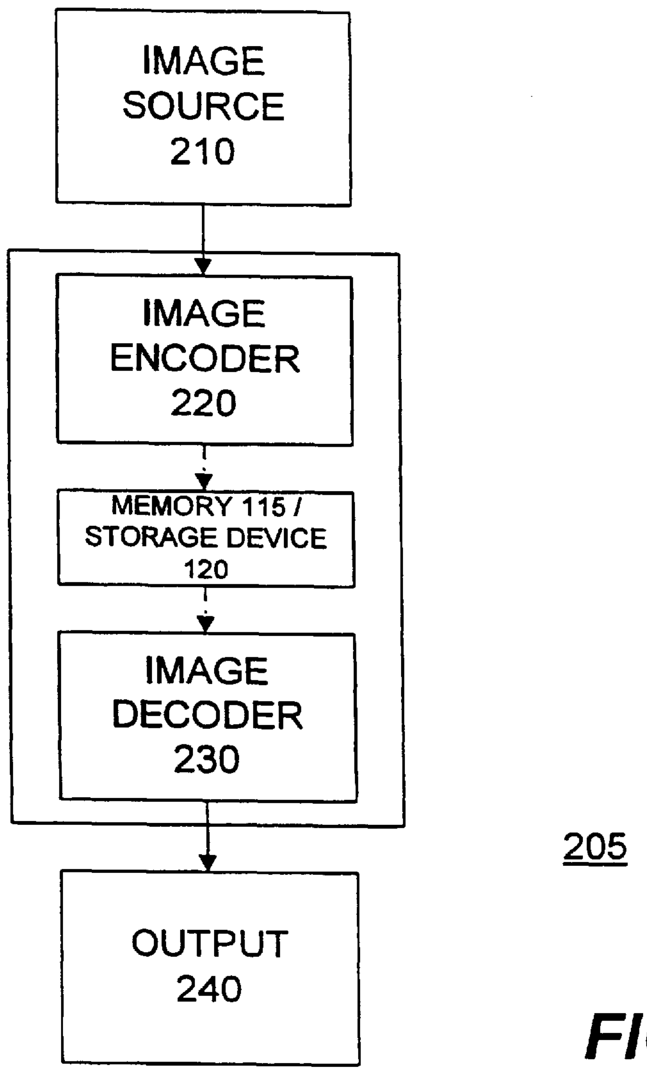


IMAGE SOURCE 210
IMAGE ENCODER 220
MEMORY 115 / STORAGE DEVICE 120
IMAGE DECODER 230
OUTPUT 240
205

FIG. 2A

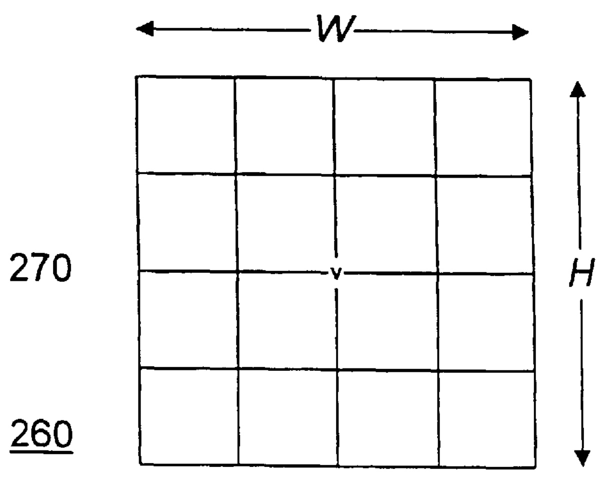

W
H
270
260
v

FIG. 2B

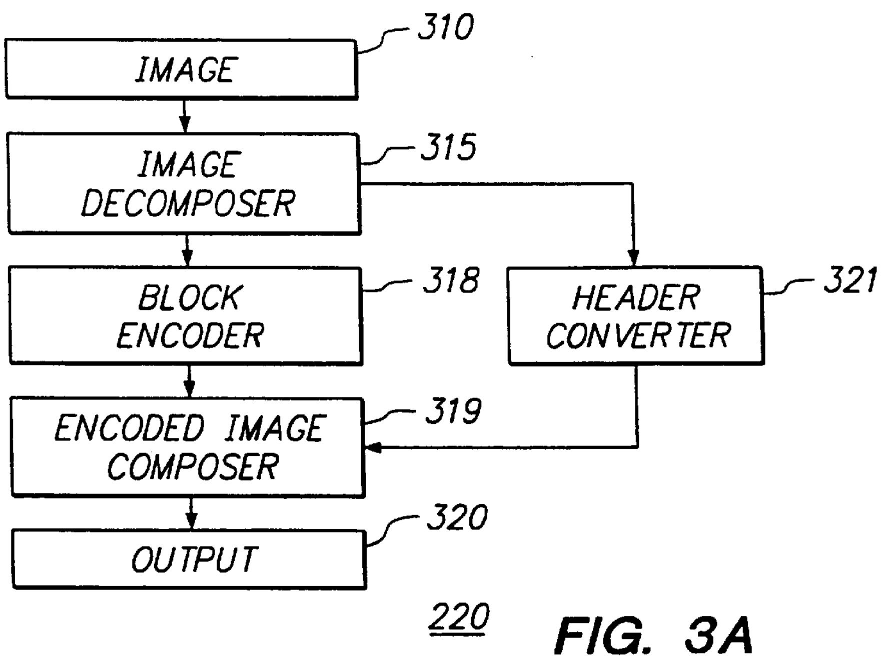

310
IMAGE
315
IMAGE DECOMPOSER
318
BLOCK ENCODER
321
HEADER CONVERTER
319
ENCODED IMAGE COMPOSER
320
OUTPUT
220

FIG. 3A

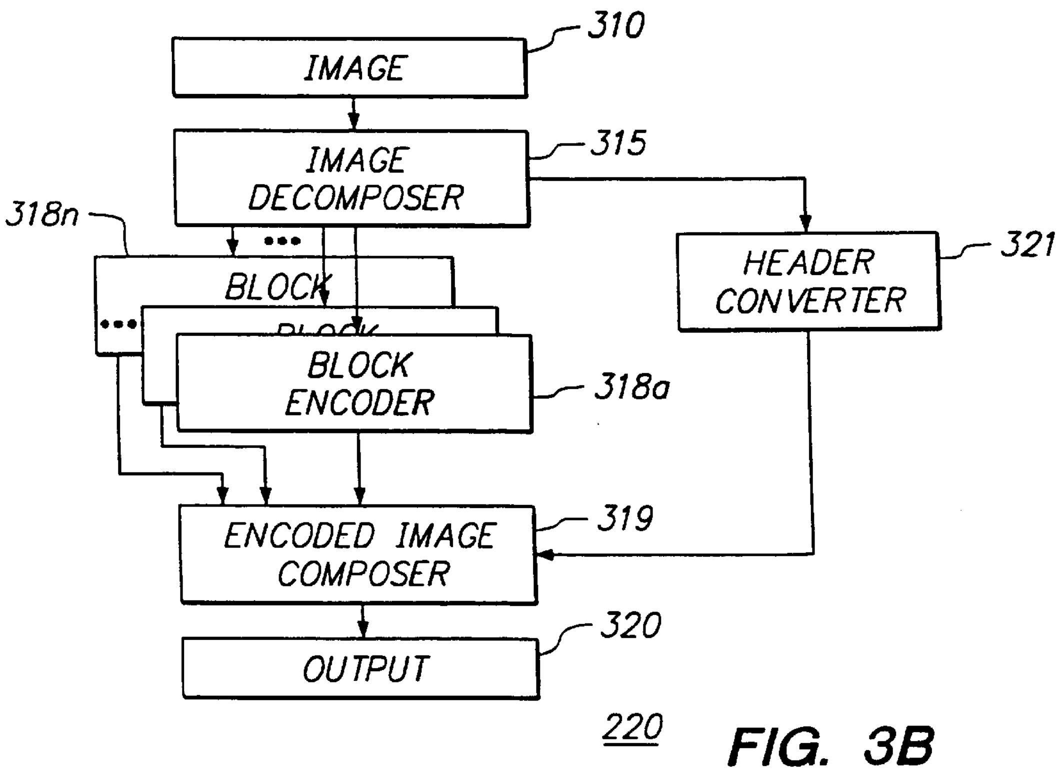

310
IMAGE
315
IMAGE DECOMPOSER
318n
BLOCK
BLOCK
BLOCK ENCODER
318a
321
HEADER CONVERTER
319
ENCODED IMAGE COMPOSER
320
OUTPUT
220

FIG. 3B

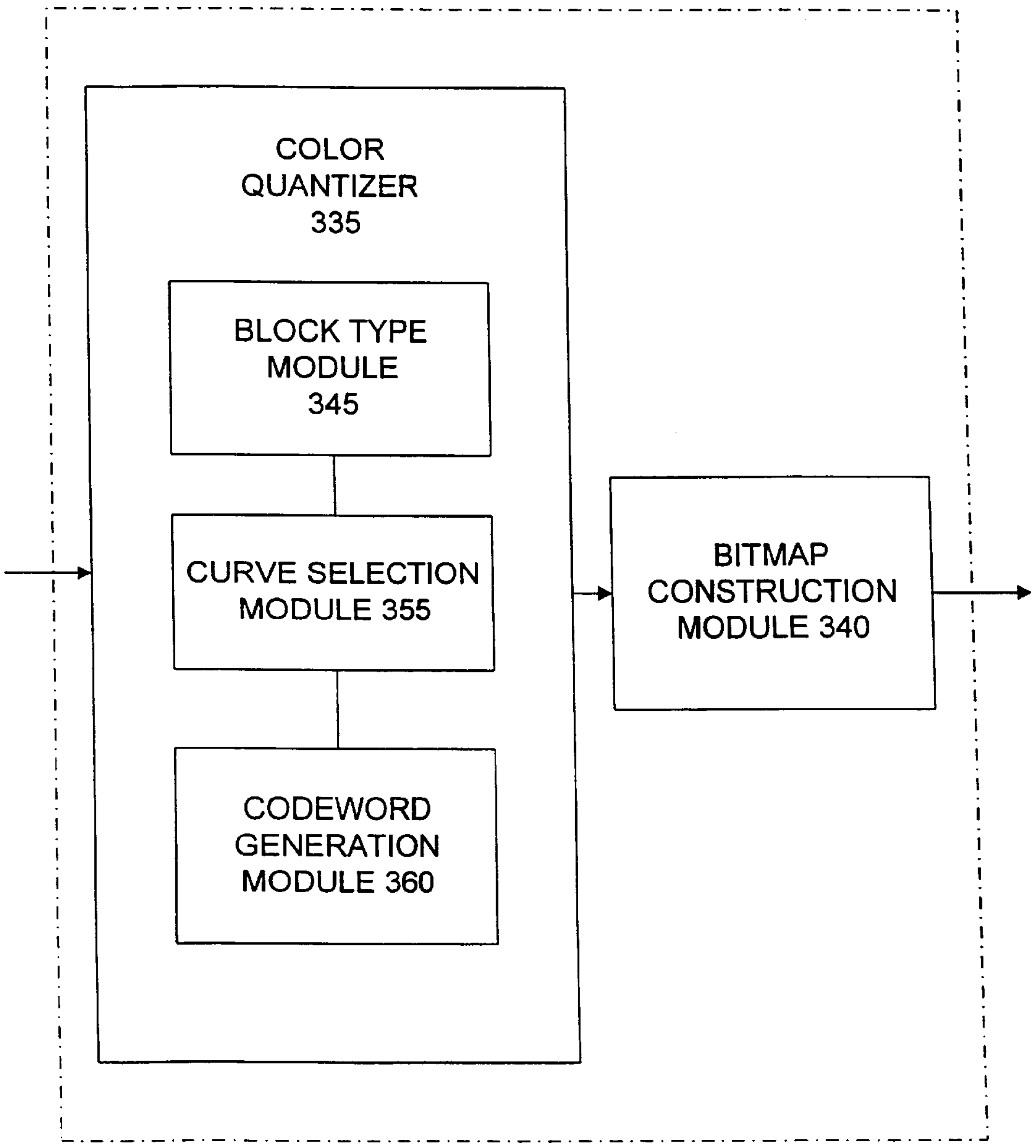

COLOR QUANTIZER 335
BLOCK TYPE MODULE 345
CURVE SELECTION MODULE 355
CODEWORD GENERATION MODULE 360
BITMAP CONSTRUCTION MODULE 340

FIG. 3C

318

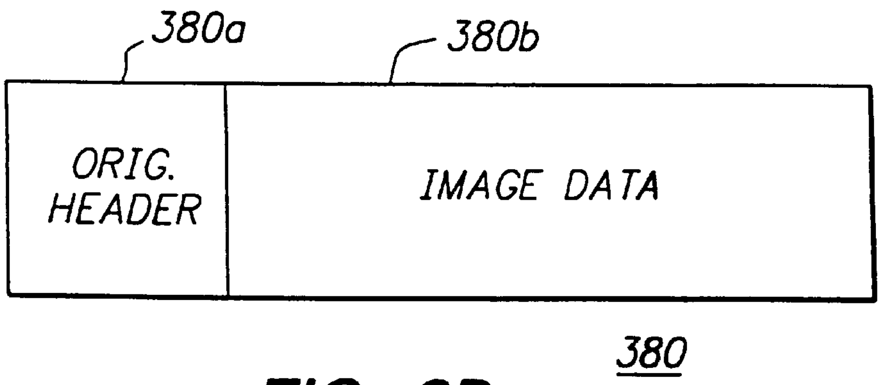


FIG. 3D

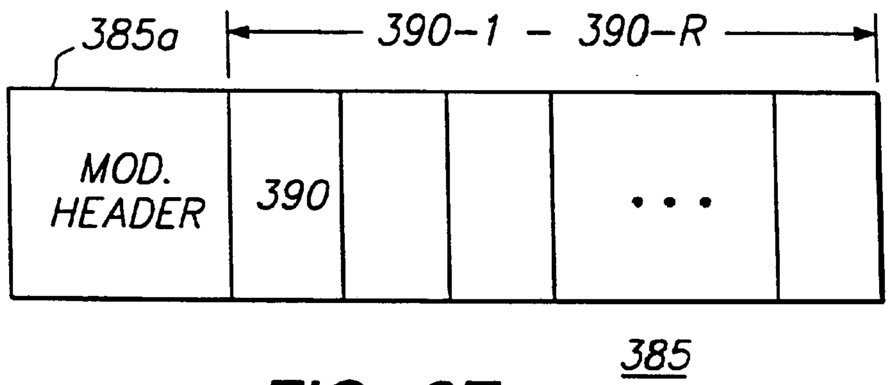


FIG. 3E

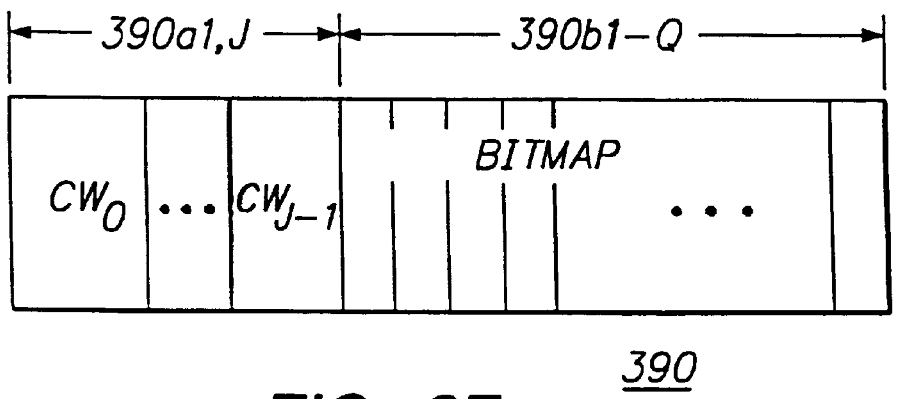


FIG. 3F

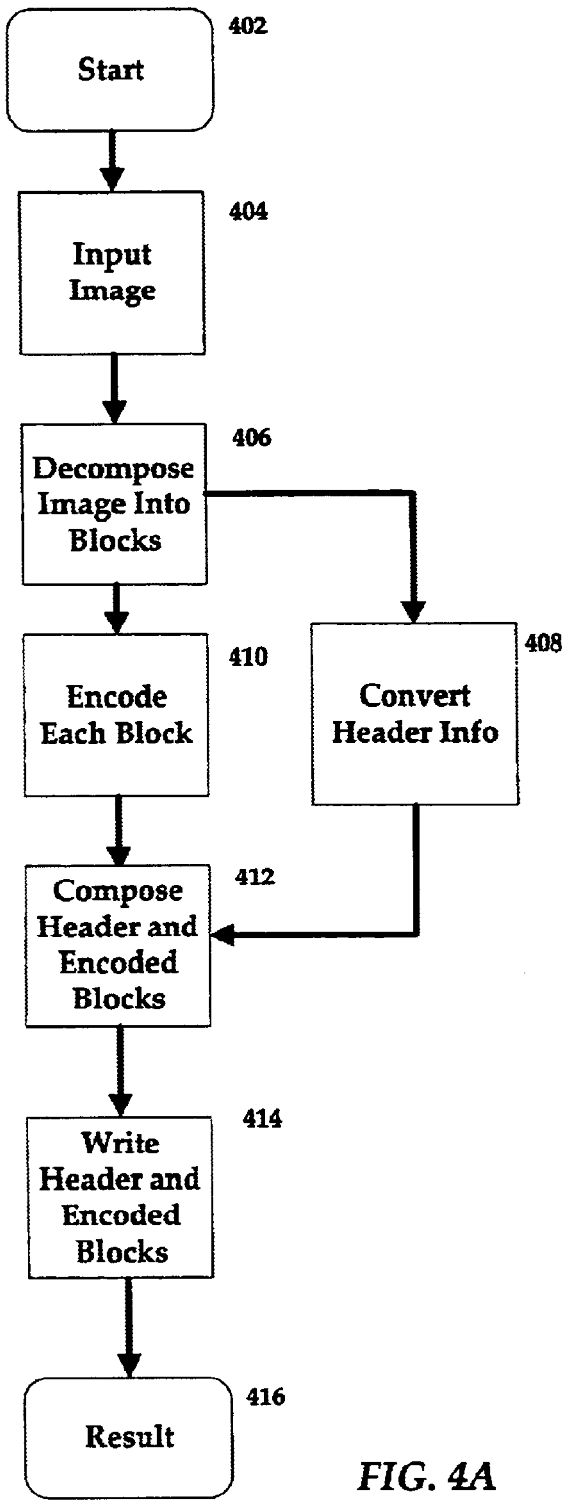

Start
402
Input Image
404
Decompose Image Into Blocks
406
Encode Each Block
410
Convert Header Info
408
Compose Header and Encoded Blocks
412
Write Header and Encoded Blocks
414
Result
416

FIG. 4A

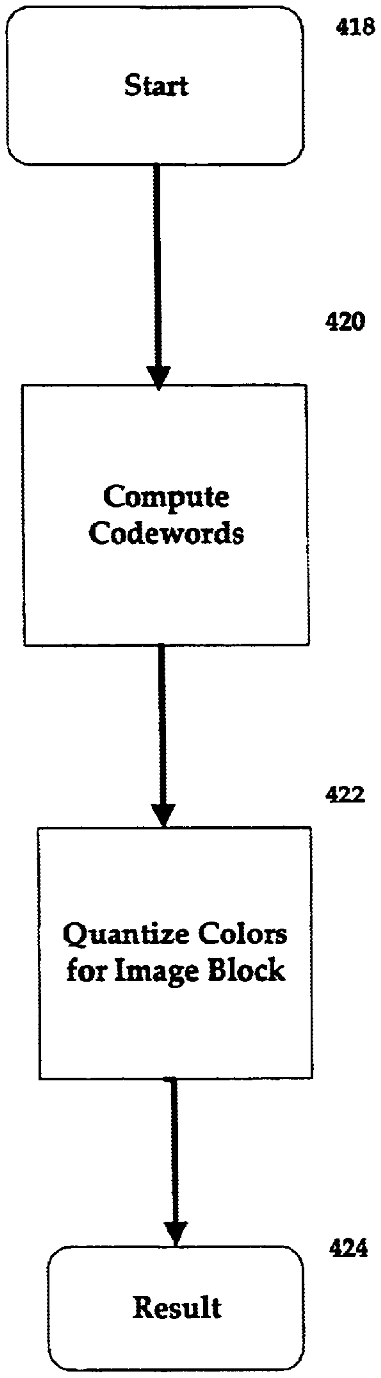

Start
418
420
Compute Codewords
422
Quantize Colors for Image Block
424
Result

FIG. 4B

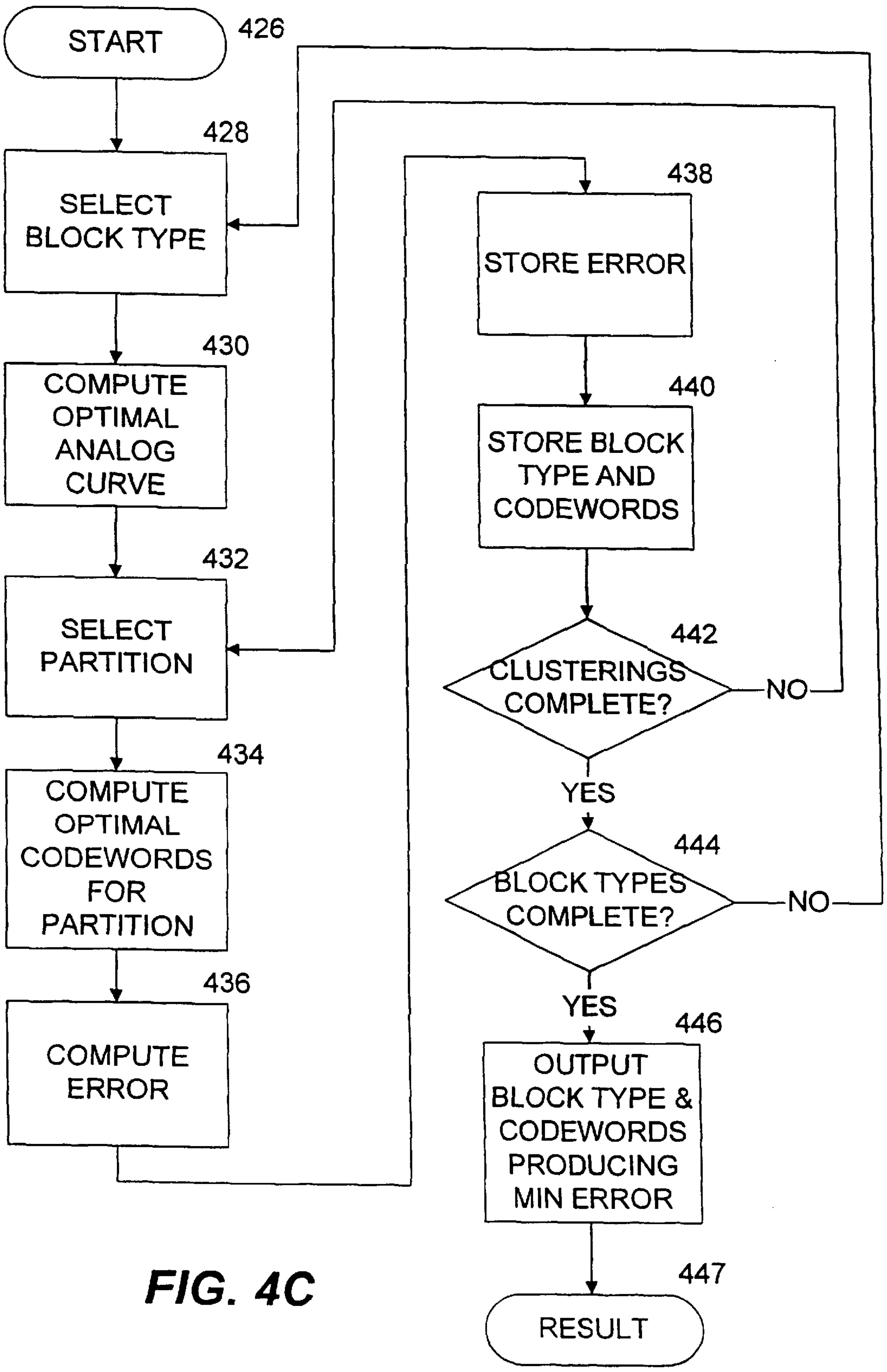

START
426
428
SELECT BLOCK TYPE
430
COMPUTE OPTIMAL ANALOG CURVE
432
SELECT PARTITION
434
COMPUTE OPTIMAL CODEWORDS FOR PARTITION
436
COMPUTE ERROR
438
STORE ERROR
440
STORE BLOCK TYPE AND CODEWORDS
442
CLUSTERINGS COMPLETE?
NO
YES
444
BLOCK TYPES COMPLETE?
NO
YES
446
OUTPUT BLOCK TYPE & CODEWORDS PRODUCING MIN ERROR
447
RESULT

FIG. 4C

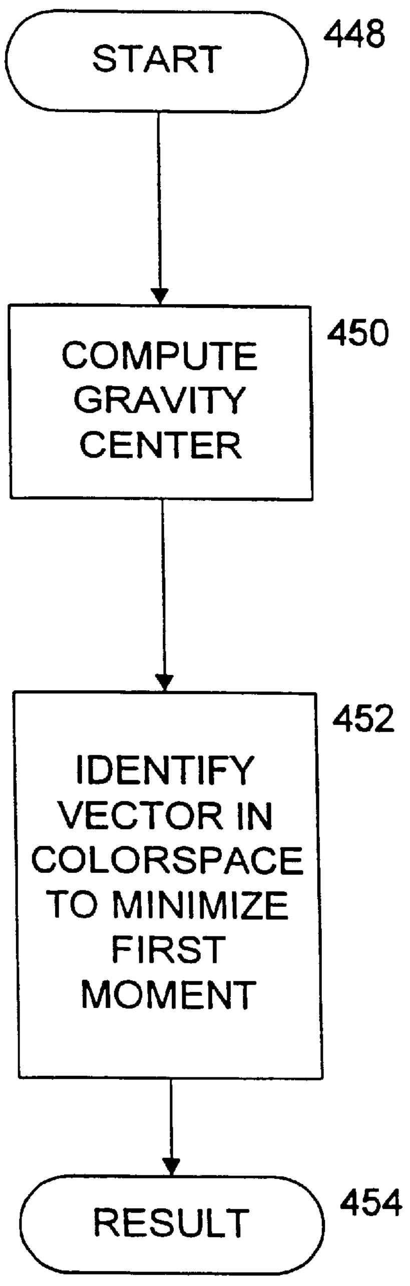

START
448
COMPUTE GRAVITY CENTER
450
IDENTIFY VECTOR IN COLORSPACE TO MINIMIZE FIRST MOMENT
452
RESULT
454

FIG. 4D

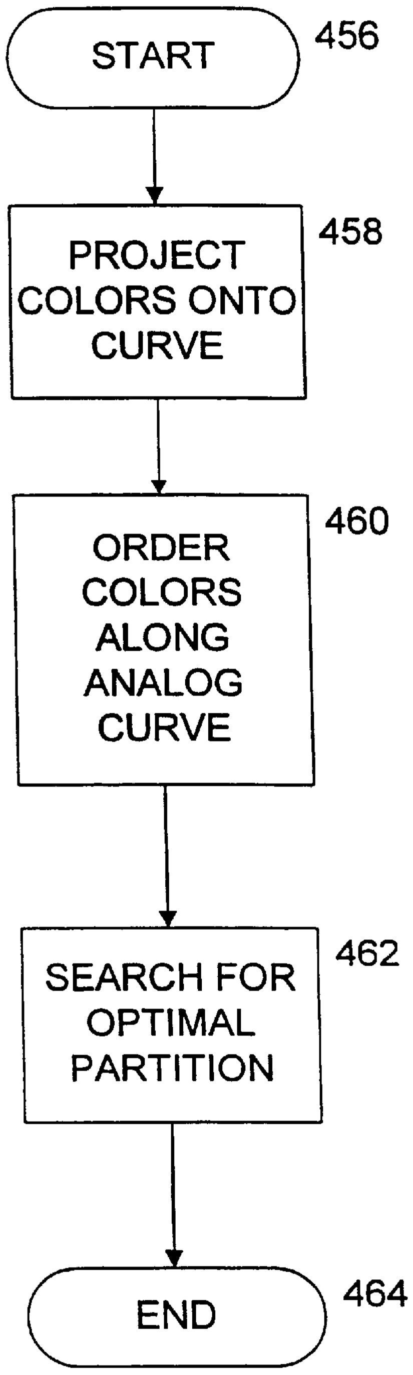

START
456
PROJECT COLORS ONTO CURVE
458
ORDER COLORS ALONG ANALOG CURVE
460
SEARCH FOR OPTIMAL PARTITION
462
END
464

FIG. 4E

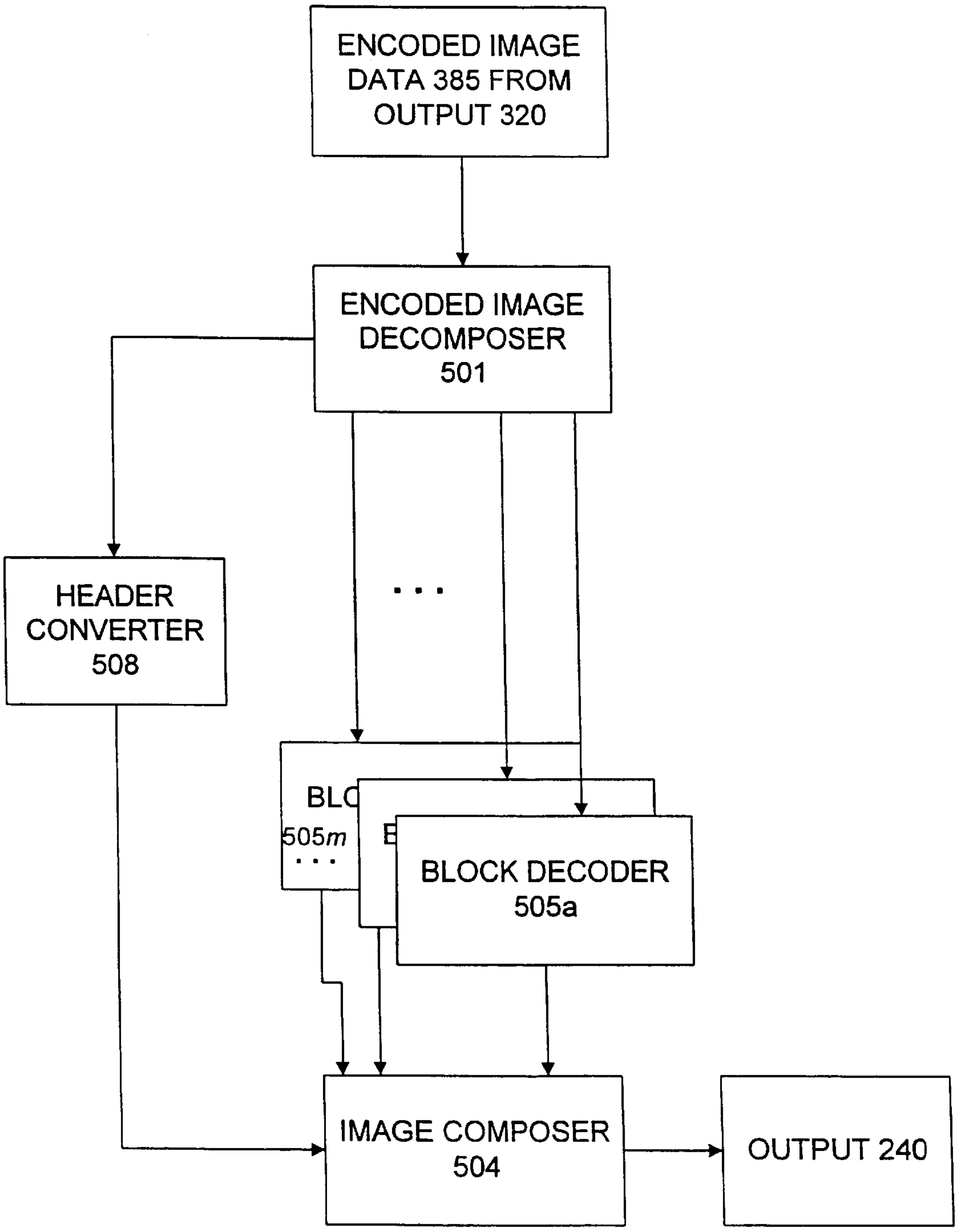

ENCODED IMAGE DATA 385 FROM OUTPUT 320
ENCODED IMAGE DECOMPOSER 501
HEADER CONVERTER 508
. . .
BLC
505m
. . .
B
BLOCK DECODER 505a
IMAGE COMPOSER 504
OUTPUT 240

FIG. 5A

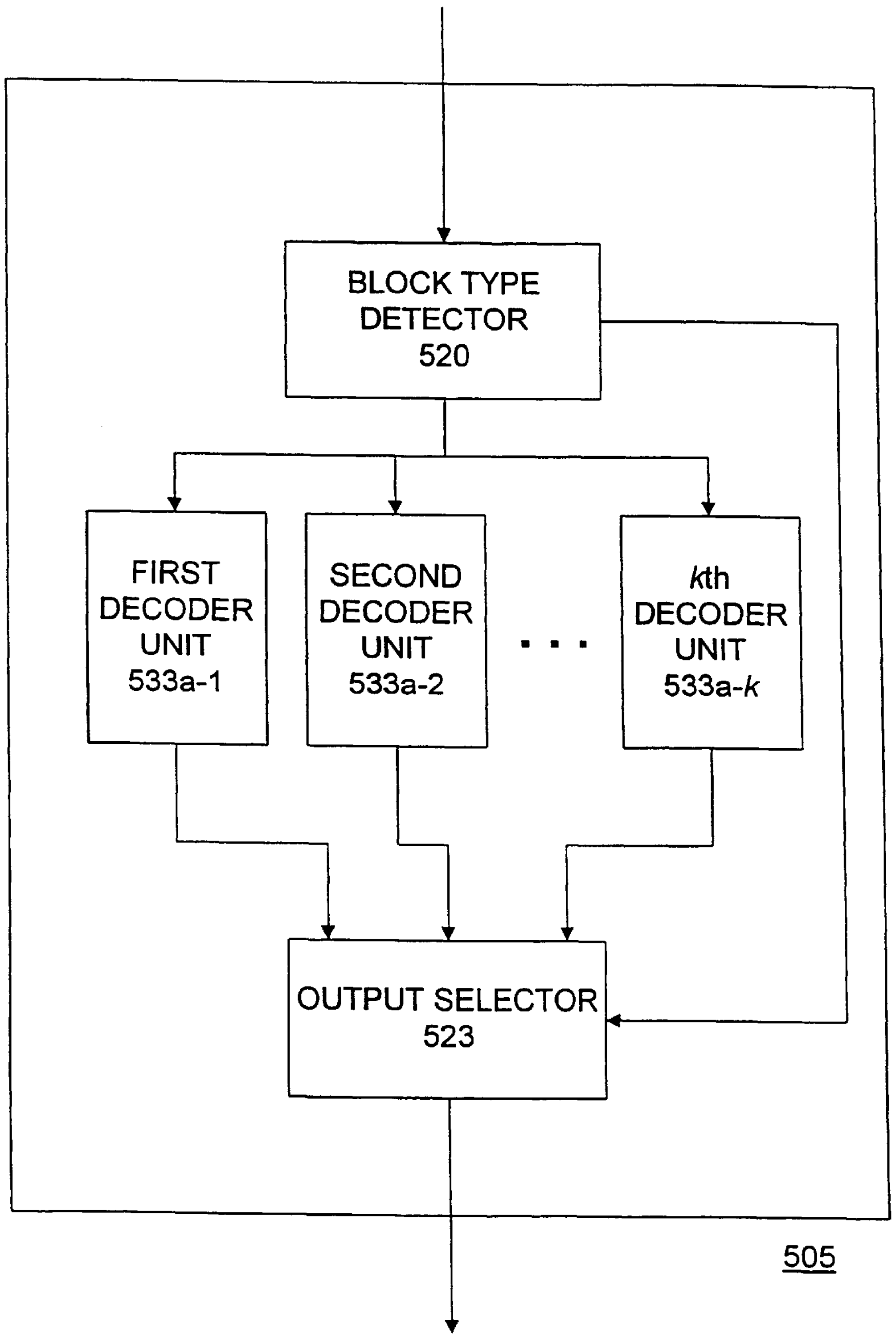

BLOCK TYPE DETECTOR 520
FIRST DECODER UNIT 533a-1
SECOND DECODER UNIT 533a-2
. . .
kth DECODER UNIT 533a-k
OUTPUT SELECTOR 523
505

FIG. 5B

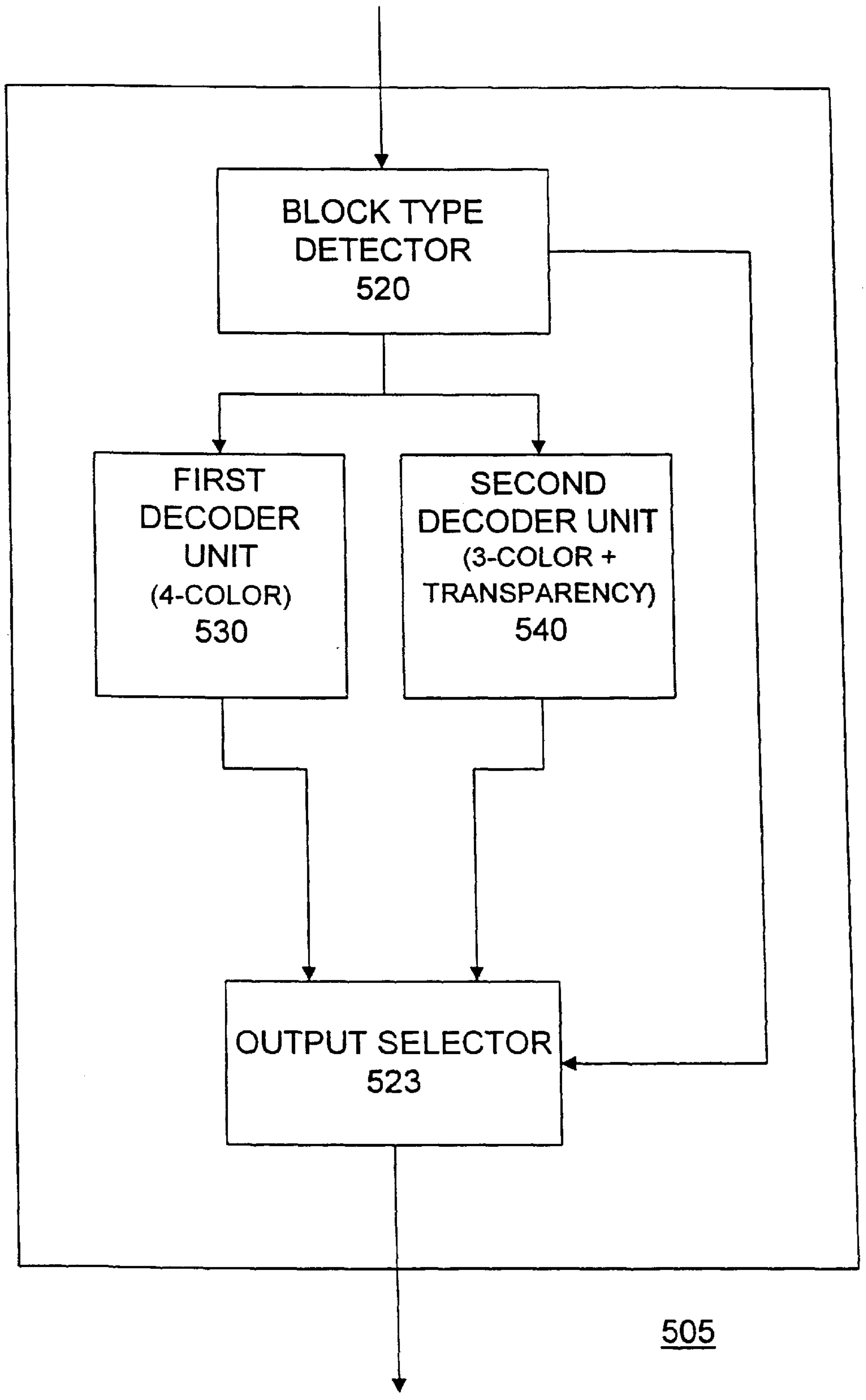

BLOCK TYPE DETECTOR 520
FIRST DECODER UNIT (4-COLOR) 530
SECOND DECODER UNIT (3-COLOR + TRANSPARENCY) 540
OUTPUT SELECTOR 523
505

FIG. 5C

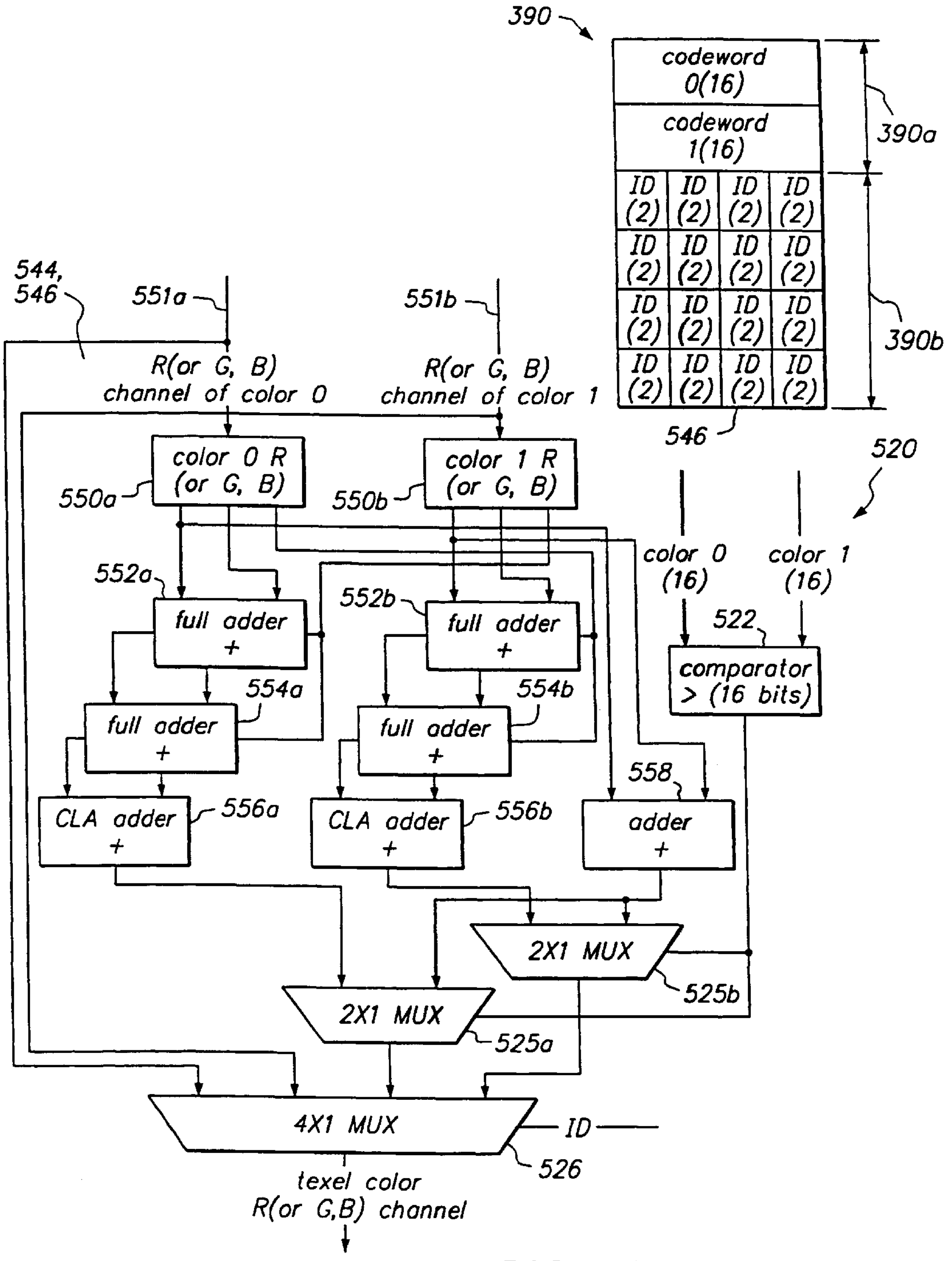

390
codeword 0(16)
codeword 1(16)
ID (2)
390a
390b
546
520
544, 546
551a
551b
R(or G, B) channel of color 0
R(or G, B) channel of color 1
color 0 R (or G, B)
color 1 R (or G, B)
550a
550b
color 0 (16)
color 1 (16)
522
comparator > (16 bits)
552a
552b
full adder +
554a
554b
CLA adder +
556a
556b
558
adder +
2X1 MUX
525a
525b
4X1 MUX
ID
526
texel color R(or G,B) channel

FIG. 5D

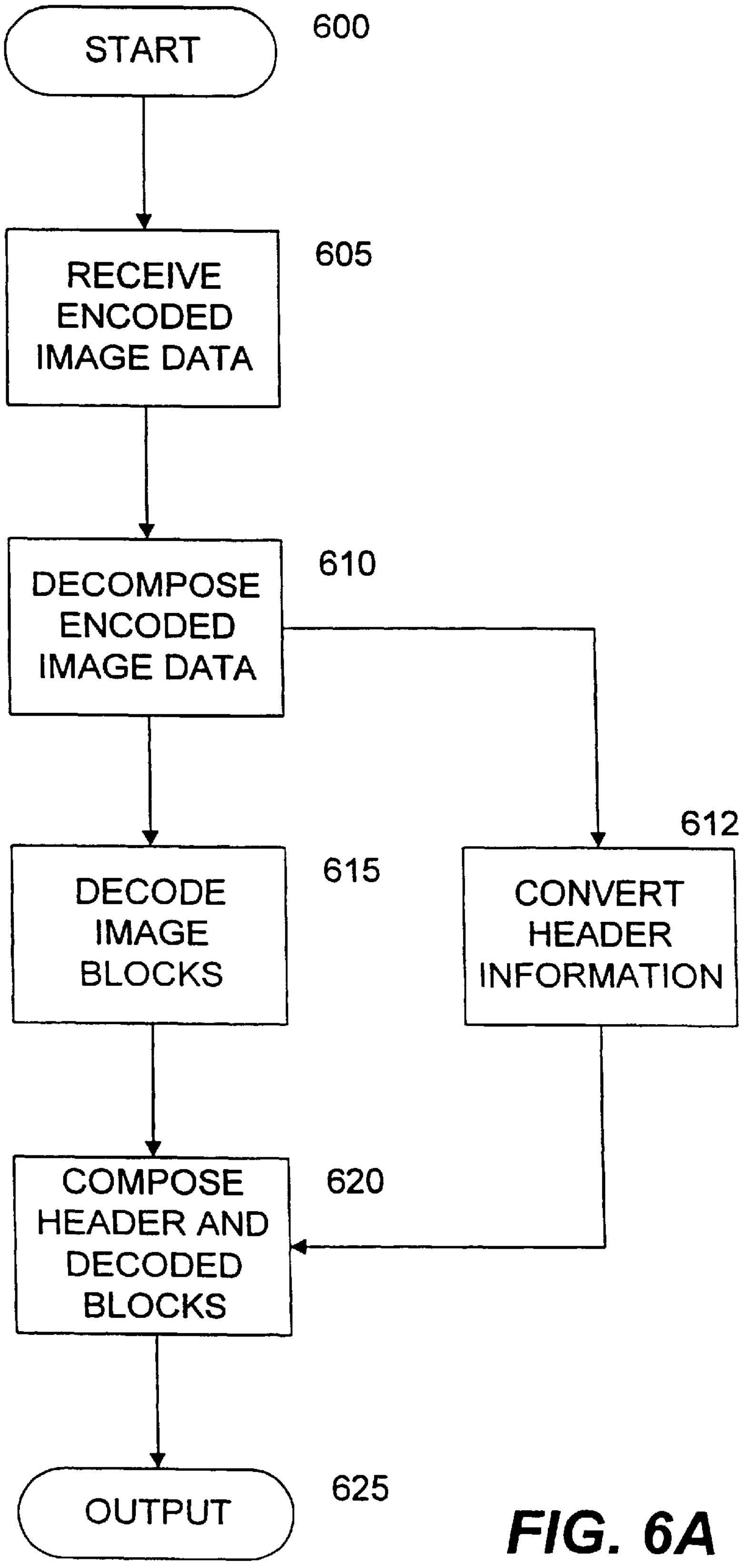

START
600
RECEIVE ENCODED IMAGE DATA
605
DECOMPOSE ENCODED IMAGE DATA
610
DECODE IMAGE BLOCKS
615
CONVERT HEADER INFORMATION
612
COMPOSE HEADER AND DECODED BLOCKS
620
OUTPUT
625

FIG. 6A

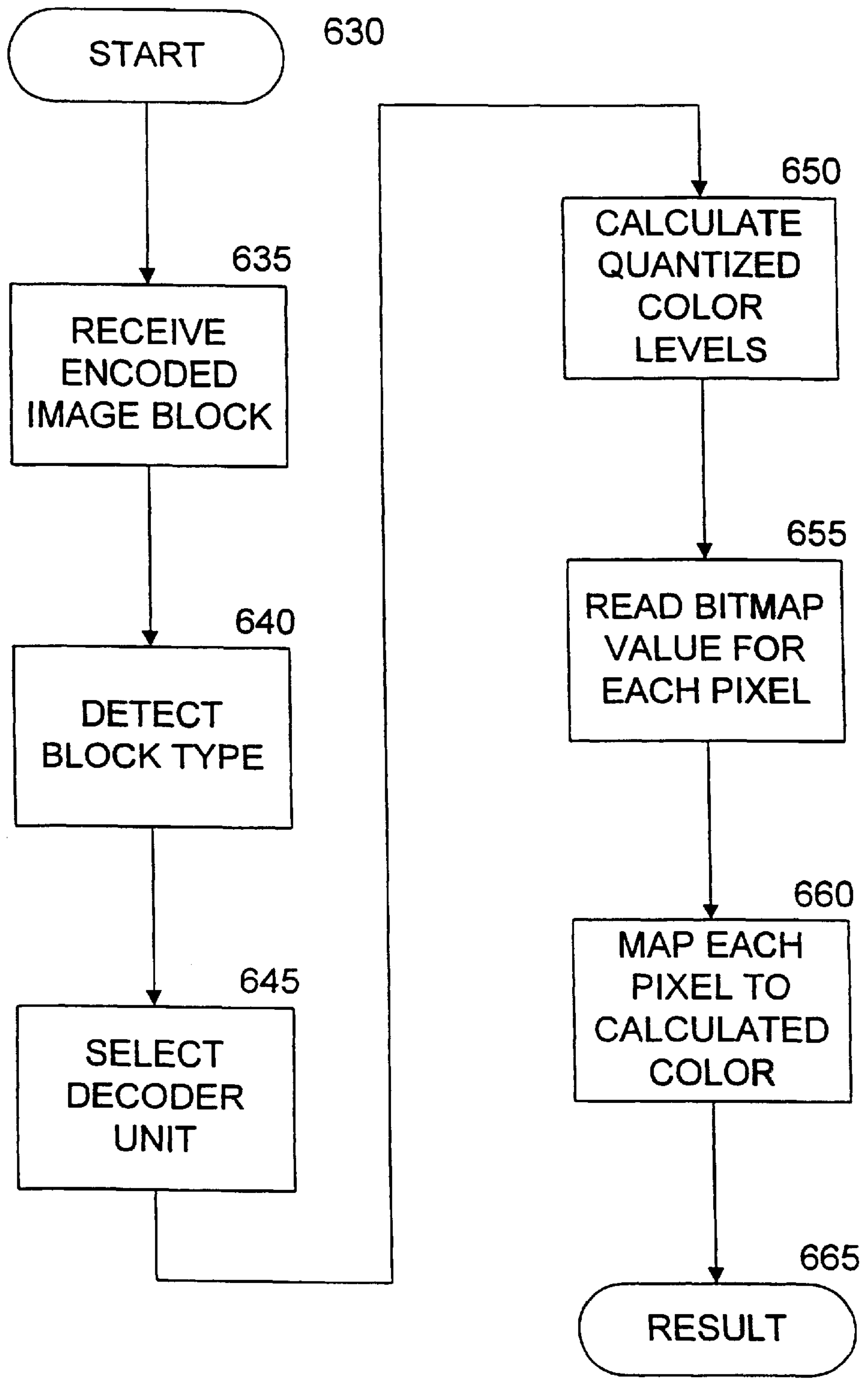

START
630
635
RECEIVE ENCODED IMAGE BLOCK
640
DETECT BLOCK TYPE
645
SELECT DECODER UNIT
650
CALCULATE QUANTIZED COLOR LEVELS
655
READ BITMAP VALUE FOR EACH PIXEL
660
MAP EACH PIXEL TO CALCULATED COLOR
665
RESULT

FIG. 6B

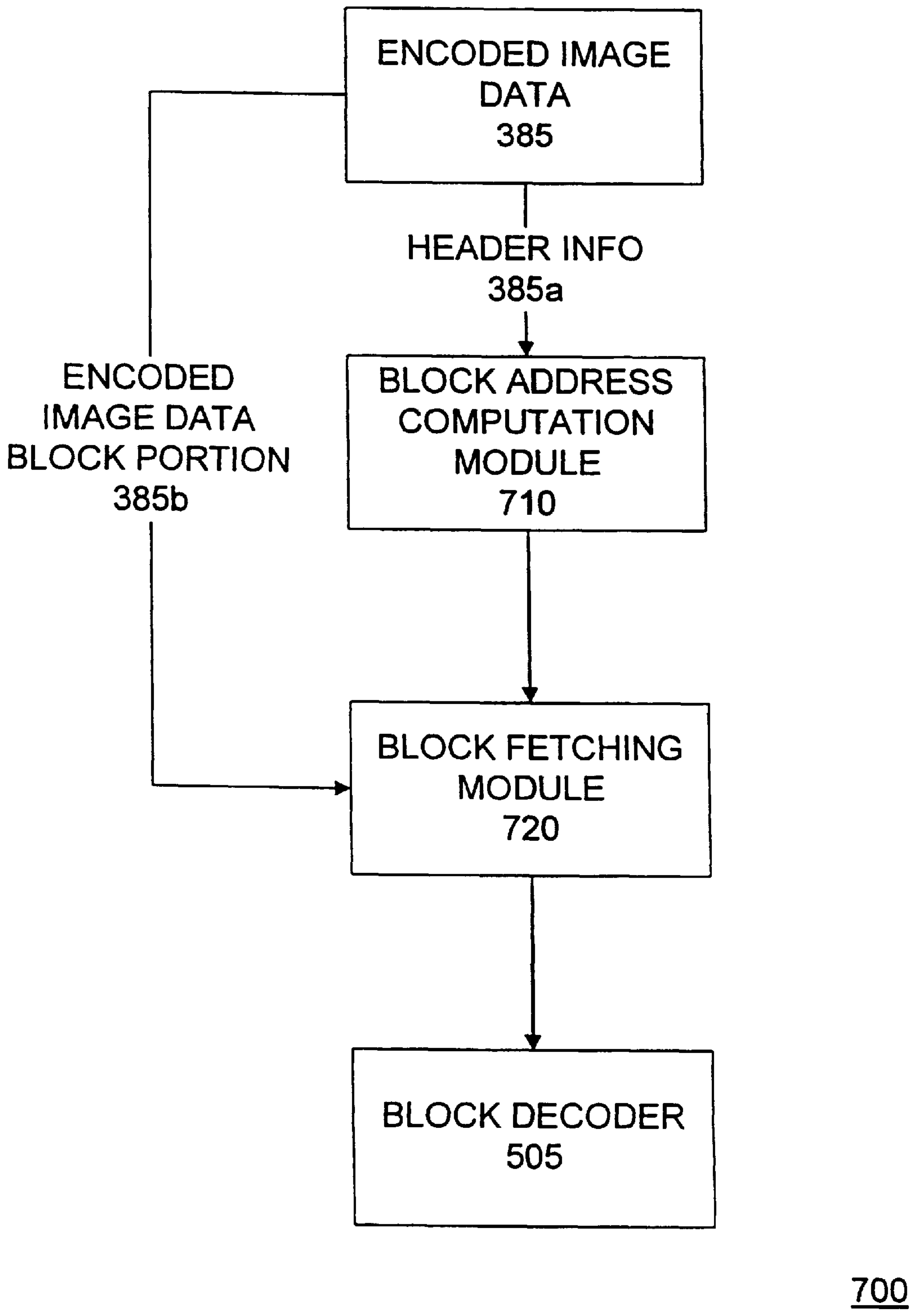

ENCODED IMAGE DATA 385
HEADER INFO 385a
BLOCK ADDRESS COMPUTATION MODULE 710
ENCODED IMAGE DATA BLOCK PORTION 385b
BLOCK FETCHING MODULE 720
BLOCK DECODER 505
700

FIG. 7A

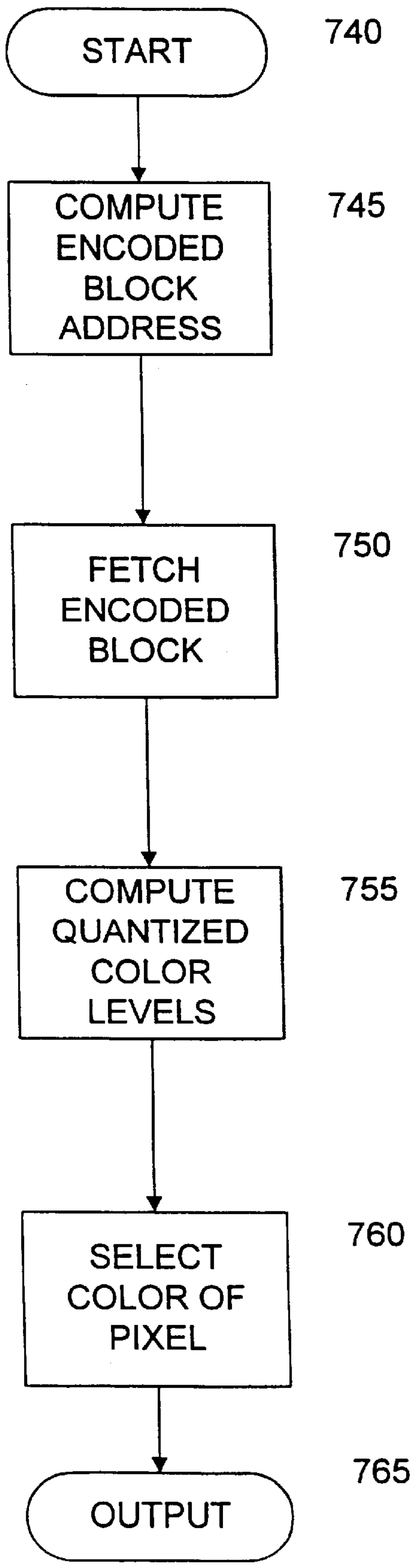

START
740
COMPUTE ENCODED BLOCK ADDRESS
745
FETCH ENCODED BLOCK
750
COMPUTE QUANTIZED COLOR LEVELS
755
SELECT COLOR OF PIXEL
760
OUTPUT
765

FIG. 7B

FIXED-RATE BLOCK-BASED IMAGE COMPRESSION WITH INFERRED PIXEL VALUES

This application is a continuation of application Ser. No. 09/351,930 filed Jul. 12, 1999, which is a continuation of application Ser. No. 08/942,860 filed Oct. 2, 1997, now U.S. Pat. No. 5,956,431 issued Sep. 21, 1999.

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to image processing systems, and more specifically, to three-dimensional rendering systems using fixed-rate image compression for textures.

2. Description of the Related Art

The art of generating images, such as realistic or animated graphics on a computer is known. To generate such images requires tremendous memory bandwidth and processing power on a graphics subsystem. To reduce the bandwidth and processing power requirements, various compression methods and systems were developed. These methods and systems included Entropy or lossless encoders, discrete cosine transform or JPEG type compressors, block truncation coding, color cell compression, and others. Each of these methods and systems, however, have numerous drawbacks.

Entropy or lossless encoders include Lempel-Ziv encoders and are used for many different purposes. Entropy coding relies on predictability. For data compression using Entropy encoders, a few bits are used to encode the most commonly occurring symbols. In stationary systems where the probabilities are fixed, Entropy coding provides a lower bound for the compression than can be achieved with a given alphabet of symbols. A problem with Entropy coding is that it does not allow random access to any given symbol. The part of the compressed data preceding a symbol of interest must be first fetched and decompressed to decode the symbol which takes considerable processing time and resources as well as decreasing memory throughput. Another problem with existing Entropy methods and systems is that they do not provide any guaranteed compression factor which makes this type of encoding scheme impractical where the memory size is fixed.

Discrete Cosine Transform ("DCT") or JPEG-type compressors, allow users to select a level of image quality. With DCT, uncorrelated coefficients are produced so that each coefficient can be treated independently without loss of compression efficiency. The DCT coefficients can be quantized using visually-weighted quantization values which selectively discard the least important information.

DCT, however, suffers from a number of shortcomings. One problem with DCT and JPEG-type compressors is that they require usually bigger blocks of pixels, typically 8×8 or 16×16 pixels, as a minimally accessible unit in order to obtain a reasonable compression factor and quality. Access to a very small area, or even a single pixel involves fetching a large quantity of compressed data, thus requiring increased processor power and memory bandwidth. A second problem with DCT and JPEG-type compressors is that the compression factor is variable, therefore requiring a complicated memory management system that, in turn, requires greater processor resources. A third problem with DCT and JPEG-type compression is that using a large compression factor significantly degrades image quality. For example, the image may be considerably distorted with a form of a ringing around the edges in the image as well as noticeable color

shifts in areas of the image. Neither artifact can be removed with subsequent low-pass filtering.

A fourth problem with DCT and JPEG-type compression is that such a decompressor is complex and has a significant associated hardware cost. Further, the high latency of the decompressor results in a large additional hardware cost for buffering throughout the system to compensate for the latency. Finally, a fifth problem with DCT and JPEG-type compressors is that it is not clear whether a color keyed image can be compressed with such a method and system.

Block truncation coding ("BTC") and color cell compression ("CCC") use a local one-bit quantizer on 4×4 pixel blocks. The compressed data for such a block consists of only two colors and 16-bits that indicate which one of the two colors is assigned to each of the 16 pixels. Decoding a BTC/CCC image consists of using a multiplexer with a look-up table so that once a 16-texel-block (32-bits) is retrieved from memory, the individual pixels are decoded by looking up the two possible colors for that block and selecting the color according to the associated bit from the 16 decision bits.

The BTC/CCC methods quantize each block to just two color levels resulting in significant image degradation. Further, a two-bit variation of CCC stores the two colors as eight-bit indices into a 256-entry color lookup table. Thus, such pixel blocks cannot be decoded without fetching additional information that can consume additional memory bandwidth.

The BTC/CCC methods and systems can use a three-bit per pixel scheme which store the two colors as 16-bit values (not indices into a table) resulting in pixel blocks of six bytes. Fetching such units, however, decreases system performance because of additional overhead due to memory misalignment. Another problem with BTC/CCC is that when it is used to compress images that use color keying to indicate transparent pixels, there will be a high degradation of image quality.

Therefore, there is a need for a method and system that maximizes the accuracy of compressed images while minimizing storage, memory bandwidth requirements, and decoding hardware complexities, while also compressing image data blocks into convenient sizes to maintain alignment for random access to any one or more pixels.

SUMMARY OF THE INVENTION

An image processing system includes an image encoder system and an image decoder system that are coupled together. The image encoder system includes a block decomposer and a block encoder that are coupled together. The block encoder includes a color quantizer and a bitmap construction module. The block decomposer breaks an original image into image blocks, each having a plurality of pixel values (e.g. colors) or equivalent color points. Each image block is then processed by the block encoder. Specifically, the color quantizer computes some number of base points, or codewords, that serve as reference pixel values, such as colors, from which computed or quantized pixel values are derived. The bitmap construction module then maps at least one pixel value in the image block to one of the computed or quantized colors or one of the codewords. The codewords and bitmap are output as encoded image blocks.

The decoder system includes a block decoder having one or more decoder units and an output selector. The block decoder may also include a block type detector for determining the block type of an image block. The block type determines the number of computed colors to use for mapping each pixel color from an image block. Using the codewords of the encoded data blocks, the comparator and the decoder units determine the computed colors for the encoded image block and map each pixel to one of the computed colors. The output selector outputs the appropriate color, which is ordered in an image composer with the other decoded blocks to output an image representative of the original image.

The present invention also includes a method of compressing an original image block having a set of original colors. The method includes: computing a set of codewords from the set of original colors; computing a set of computed colors using the set of codewords; and mapping each original color to one of the computed colors or one of the codewords to produce an index for each original color.

The compressed or encoded image block, which has a first set of indices and a set of codewords, where a set is equal to or greater than one, is decoded by: computing at least one computed color using the set of codewords; and mapping an index within the first set of indices to one of the computed colors or one of the codewords.

Those of ordinary skill in the art will readily recognize that the present invention may be practiced using any general purpose computer system, such as the computer system described below, or any "hardwired" device specifically designed to perform the method, such as but not limited to devices implemented using ASIC or FPGA technology and the like.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a block diagram of a data processing system in accordance with the present invention;

FIG. **2**A is a block diagram of an image processing system in accordance with the present invention;

FIG. **2**B is a graphical representation of an image block in accordance with the present invention;

FIG. **3**A is a block diagram of a first embodiment an image encoder system in accordance with the present invention;

FIG. **3**B is a block diagram of a second embodiment of an image encoder system in accordance with the present invention;

FIG. **3**C is a block diagram of an image block encoder in accordance with the present invention;

FIG. **3**D is a data sequence diagram of an original image in accordance with the present invention;

FIG. **3**E is a data sequence diagram of encoded image data of the original image output from the image encoder system in accordance with the present invention;

FIG. **3**F is a data sequence diagram of an encoded image block from the image block encoder in accordance with the present invention;

FIGS. **4**A–**4**E are flow diagrams illustrating an encoding process in accordance with the present invention;

FIG. **5**A is a block diagram of an image decoder system in accordance with the present invention;

FIG. **5**B is a block diagram of a first embodiment of a block decoder in accordance with the present invention;

FIG. **5**C is a block diagram of a second embodiment of a block decoder in accordance with the present invention;

FIG. **5**D is a logic diagram illustrating a first embodiment of a decoder unit in accordance with the present invention;

FIGS. **6**A–**6**B are flow diagrams illustrating a decoding process in accordance with the present invention;

FIG. **7**A is a block diagram of a subsystem for random access to a pixel or an image block in accordance with the present invention; and

FIG. **7**B is a flow diagram illustrating random access to a pixel or an image block in accordance with the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

FIG. **1** is a block diagram of a data processing system **105** constructed in accordance with the present invention. The data processing system **105** includes a processing unit **110**, a memory **115**, a storage device **120**, an input device **125**, an output device **130**, and a graphics subsystem **135**. In addition, the data processing system **105** includes a data bus **145** that couples each of the other components **110**, **115**, **120**, **125**, **130**, **135** of the data processing system **105**.

The data bus **145** is a conventional data bus and while shown as a single line it may be a combination of a processor bus, a PCI bus, a graphical bus, and an ISA bus. The processing unit **110** is a conventional processing unit such as the Intel Pentium processor, Sun SPARC processor, or Motorola PowerPC processor, for example. The processing unit **110** processes data within the data processing system **105**. The memory **115**, the storage device **120**, the input device **125**, and the output device **130** are also conventional components as recognized by those skilled in the art. The memory **115** and storage device **120** store data within the data processing system **105**. The input device **125** inputs data into the system while the output device **130** receives data from the data processing system **105**.

FIG. **2**A is a block diagram of an image processing system **205** constructed in accordance with the present invention. In one embodiment, the image processing system **205** runs within the data processing system **105**. The image processing system **205** includes an image encoder system **220** and an image decoder system **230**. The image processing system **205** may also include a unit for producing an image source **210** from which images are received, and an output **240** to which processed images are forwarded for storage or further processing. The image encoder system **220** is coupled to receive an image from the image source **210**. The image decoder system **230** is coupled to output the image produced by the image processing system **205**. The image encoder system **220** is coupled to the image decoder system **230** through a data line and may be coupled via a storage device **120** and/or a memory **115**, for example.

Within the image encoder system **220**, the image is broken down into individual blocks and processed before being forwarded to, e.g., the storage device **140**, as compressed or encoded image data. When the encoded image data is ready for further data processing, the encoded image data is forwarded to the image decoder system **230**. The image decoder system **230** receives the encoded image data and decodes it to generate an output that is a representation of the original image that was received from the image source **210**.

FIGS. **3**A and **3**B are block diagrams illustrating two separate embodiments of the image encoder system **220** of the present invention. The image encoder system **220** includes an image decomposer **315**, a header converter **321**, one or more block encoders **318** (**318***a*–**318***n*, where n is the nth encoder, n being any positive integer), and an encoded image composer **319**. The image decomposer **315** is coupled to receive an original image **310** from a source, such as the image source **210**. The image decomposer **315** is also coupled to the one or more block encoders **318** and to the header converter **321**. The header converter **321** is also coupled to the encoded image composer **319**. Each block encoder **318** is also coupled to the encoded image composer **319**. The encoded image composer **319** is coupled to the output **320**.

The image decomposer **315** receives the original image **310** and forwards information from a header of the original image **310** to the header converter **321**. The header converter **321** modifies the original header to generate a modified header, as further described below. The image decomposer **315** also breaks, or decomposes, the original image **310** into R number of image blocks, where R is some integer value. The number of image blocks an original image **310** is broken into may depend on the number of image pixels. For example, in a preferred embodiment an image **310** comprised of A image pixels by B image pixels will typically be (A/4)*(B/4) blocks, where A and B are integer values. For example, where an image is 256 pixels by 256 pixels, there will be 64×64 blocks. In other words, the image is decomposed such that each image block is 4 pixels by 4 pixels (16 pixels). Those skilled in the art will recognize that the number of pixels or the image block size may be varied, for example m×n pixels, where m and n are positive integer values.

Briefly turning to FIG. **2**B, there is illustrated an example of a single image block **260** in accordance with the present invention. The image block **260** is comprised of pixels **270**. The image block **260** may be defined as an image region W pixels **270** in width by H pixels **270** in height, where W and H are integer values. In a preferred embodiment, the image block **260** is comprised of W=4 pixels **270** by H=4 pixels **270** (4×4).

Turning back to FIGS. **3**A and **3**B, each block encoder **318** receives an image block **260** from the image decomposer **315**. Each block encoder **318** encodes or compresses each image block **260** that it receives to generate an encoded or compressed image block. Each encoded image block is received by the encoded image composer **319** which orders the encoded blocks in a data file. The data file from the encoded image composer **319** is concatenated with a modified header from the header converter **321** to generate an encoded image data file that is forwarded to the output **320**. Further, it is noted that having more than one block encoder **318***a*–**318***n* allows for encoding multiple image blocks simultaneously, one image block per block encoder **318***a*–**318***n*, within the image encoder system **220** to increase image processing efficiency and performance.

The modified header and the encoded image blocks together form the encoded image data that represents the original image **310**. The function of each element of the image encoder system **220**, including the block encoder **318**, will be further described below with respect to FIGS. **4**A–**4**E.

The original image **310** may be in any one of a variety of formats including red-green-blue ("RGB"), YUV **420**, YUV **422**, or a proprietary color space. It may be useful in some cases to convert to a different color space before encoding the original image **310**. It is noted that in one embodiment of the present invention, each image block **260** is a 4×4 set of pixels where each pixel **270** is 24-bits in size. For each pixel **270** there are 8-bits for a Red(R)-channel, 8-bits for a Green(G)-channel, and 8-bits for a Blue(B)-channel in a red-green-blue ("RGB") implementation color space. Further, each encoded image block is also a 4×4 set of pixels, but, each pixel is only 2-bits in size and has an aggregate size of 4-bits as will be further described below.

FIG. **3**C is a block diagram illustrating a block encoder **318** of the present invention in greater detail. The block encoder **318** includes a color quantizer **335** and a bitmap construction module **340**. The color quantizer **335** is coupled to the bitmap construction module **340**. Further, the color quantizer **335** further emphasizes a block type module **345**, a selection module **355**, and a codeword generation module **360**. The block type module **345** is coupled to the selection module **355**. The selection module **355** is coupled to the codeword generation module **360**.

Each image block **260** of the decomposed original image **310** is received and initially processed by the color quantizer **335** before being forwarded to the bitmap construction module **340** for further processing. The bitmap construction module **340** outputs encoded image blocks for the encoded image composer **319** to order. The bitmap construction module **340** and the color quantizer **335**, including the block type module **345**, the selection module **355**, and the codeword generation module **360**, are further discussed below in FIGS. **4**A–**4**E.

Briefly, FIG. **3**D is a diagram of a data sequence or string **380** representing the original image **310** that is received by the block decomposer **315**. The data string **380** of the original image **310** includes an a-bit header **380***a* and a b-bit image data **380***b*, where a and b are integer values. The header **380***a* may include information such as the pixel width of the image **310**, the pixel height of the image **310**, and the format of the image **310**, e.g., the number of bits to the pixel in RGB or YUV format, for example, as well as other information. The image data is the data **380***b* representing the original image **310** itself.

FIG. **3**E is a diagram of a data sequence or string **385** representing encoded image data **385** that is generated and output **320** by the image encoder system **220**. The data string for the encoded image data **385** includes a modified header portion **385***a* and an encoded image block portion **390**-**1**–**390**-R. The modified header portion **385***a* is generated by the header converter **321** from the original header **380***a* for the original image **310**. The modified header generated by the header converter **321** includes information about file type, a number of bits per pixel of the original image **310**, addressing into the original image **310**, other miscellaneous encoding parameters, as well as the width and height information indicating the size of that original image **310**. The encoded image block portion **390**-**1**-R includes the encoded image blocks **390**-**1**–**390**-R from the block encoders **318**, where R is an integer value that is the number of blocks resulting from the decomposed original image **310**.

FIG. **3**F is a diagram of a data sequence or string **390** representing an encoded image block in accordance with the present invention. It is understood that the data string **390** representing the encoded image block may be similar to any one of the encoded image blocks **390**-**1**–**390**-R shown in the encoded image data string **385**.

The data string **390** of the encoded image block includes a codeword section **390***a* which includes J codewords, where J is an integer value, and a bitmap section **390***b*. The codeword section **390***a* includes J codewords **390***a* that are used to compute the colors indexed by the bitmap **390***b*. A codeword is a n-bit data string, where n is an integer value, that identifies a pixel property, for example a color component. In a preferred embodiment, there are two 16-bit codewords **390***a*, CW0, CW1 (J=2). The bitmap is a Q-bit data portion and is further discussed below in FIG. **4**B.

Further, in a preferred embodiment, each encoded image block is 64-bits, which includes two 16-bit codewords and a 32-bit (4×4×2 bit) bitmap **395**. Encoding the image block **260** as described provides greater system flexibility and increased data processing efficiency as will be further discussed below.

FIGS. **4A–4E** describe the operation of the image encoder system **220**. FIG. **4A** describes the general operation of the image encoder system **220**. At the start **402** of operation, data string **380** of the original image **310**, that includes the a-bit header **380***a* and the b-bit image data **380***b*, is input **404** into the block decomposer **315** from the image source **210**. The block decomposer **315** decomposes **406** the original image **310** to extract the a-bit header **380***a* and it to the header converter **321**. The block decomposer also **315** decomposes, **406** the original image **310** into image blocks. Each image block **260** is independently compressed, or encoded, **410** in the one or more block encoders **318**.

The header converter **321** converts **408** the a-bit header to generate a modified header **385***a*. The modified header **385***a* is forwarded to the encoded image composer **319**. Simultaneous with the header converter **321** converting **408** the a-bit header, each image block is encoded **410** by the one or more image encoders **318***a*–**318***n* to generate the encoded image blocks **390-1**–**390**-R. Again, it is noted that each image block **260** may be processed sequentially in one block encoder **318**a or multiple image blocks **260** may be processed in parallel in multiple block encoders **318***a*–**318***n*.

The encoded image blocks **390** are output from the block encoders **318** and are placed into a predefined order by the encoded image composer **319**. In a preferred embodiment, the encoded image blocks **390** are ordered in a file from left to right and top to bottom in the same. order in which they were broken down by the block decomposer **315**. The image encoder system **220** continues by composing **412** the modified header information **385***a* from the header converter **321** and the encoded image blocks **390**. Specifically, the modified header **385***a* and the ordered encoded image blocks **390** are concatenated to generate the encoded image data file **385**. The encoded image data file **385** is written **414** as encoded output **320** to the memory **115**, the storage device **120**, or the output device **130**, for example.

FIG. **4B** shows the encoding process **410** for the encoder system **220** described above in FIG. **2**. At the start **418** of operation, codewords are computed **420**. As discussed above in FIG. **3F**, in a preferred embodiment there are two codewords **390***a*, CW**0**, CW**1**. The process for computed codewords is further described below in FIG. **4C**.

Once the codewords are computed **420** pixel values or properties, such as colors, for the image block **260** are computed or quantized **422**. Specifically, the codewords **390***a* provide points in a pixel space from which M quantized pixel values may be inferred, where M is an integer value. The M quantized pixel values are a limited subset of pixels in a pixel space that are used to represent the current image block. The process for quantizing pixel values, and more specifically colors, will be described below in FIGS. **4D** and **4E**. Further, it is noted that the embodiments will now be described with respect to colors of a pixel value although one skilled in the art will recognize that in general any pixel value may be used with respect to the present invention.

In a preferred embodiment, each pixel is encoded with two bits of data which can index one of M quantized colors (M=4). Further, in a preferred embodiment the four quantized colors are derived from the two codewords **390***a* where two colors are the codewords themselves and the other two colors are inferred from the codewords, as will be described below. It is also possible to use the codewords **390***a* so that there is one index to indicate a transparent color and three indices to indicate colors, of which one color is inferred.

In a preferred embodiment, the bitmap **390***b* is a 32-bit data string.

The bitmap **390***b* and codewords **390***a* are output **424** as a 64-bit data string representing an encoded image block **390**. Specifically, the encoded image block **390** includes the two 16-bit codewords **390***a* (n=16) and a 32-bit bitmap **390***b*. Each codeword **390***a* CW**0**, CW**1** that is a 16-bit data string includes a 5-bit red-channel, 6-bit green-channel, and 5-bit blue-channel.

Each of the encoded image blocks **390** is placed together **390***a***1**–**390***a*R, and concatenated with header information **385***a* derived from the original header **380***a* of the original image **310**. The resulting **424** output is the encoded image data **385** representing the original image **310**.

FIG. **4C** describes the process for computing the codewords for the image blocks **260** in more detail. At the start **426** of the process, the color quantizer **335** uses the block type module **345** to select **428** the first block type for the image block **260** that is being processed. For example, one block type selected **428** may be a four-color and another block type selected **428** may be a three-color plus transparency, where the colors within the particular block type have equidistant spacing in a color space.

Those of ordinary skill in the art will readily recognize that selecting a block type for each image is not intended to be limiting in any way Instead, the present invention may be limited to processing image blocks that are of a single block type. This eliminates the need to distinguish between different block types, such as the three and four color block types discussed above. Consequently, the block type module **345** in FIG. **3B** and reference number **428** in FIG. **4C** are optional and are not intended to limit the present invention in any way.

Once the block type is selected **428**, the process computes **430** an optimal analog curve for the block type. Computation **430** of the optimal analog curve **430** will be further described below in FIG. **4D**. The analog curve is used to simplify quantizing of the colors in the image block. After computing **430** the optimal analog curve, the process selects **432** a partition of the points along the analog curve. A partition may be defined as a grouping of indices {1 . . . (W×H)} into M nonintersecting sets. In a preferred embodiment, the indices (1 . . . 16) are divided into three or four groups, or clusters, (M=3 or 4) depending on the block type.

Once a partition is selected **432**, the optimal codewords for that particular partition are computed **434**. Computation **434** of the optimal codewords is further described below in FIG. **4E**. In addition to computing **434** the codewords, an error value (squared error as describe below) for the codewords is also computed **436**. Computation **436** of the error values is further described below with respect to FIG. **4E** also. If the computed **436** error value is the first error value it is stored. Otherwise, the computed **436** error value is stored **438** only if it is less than the previously stored error value. For each stored **438** error value, the corresponding block type and codewords are also stored **440**. It is noted that the process seeks to find the block type and codewords that minimize the error function.

The process continues by determining **442** if the all the possible partitions are complete. If there are more partitions possible, the process selects **432** the next partition and once again computes **434** the codewords, computes **436** the associated error value, and stores **438** the error value and stores **440** associated block type and codewords only if the error value is less than the previously stored error value.

After all the possible partitions are completed, the process determines **444** whether all the block types have been selected. If there are more block types, the process selects **428** the next block type. Once again, the process will compute **430** the optimal analog curve, select **432**, **442** all the possible partitions, for each partition it will compute **434**, **436** the codewords and associated error value, and store **438**, **440** the error value and associated block type and codeword only if the error value is less than the previously stored error value. After the last block type is processed, the process outputs **446** a result **447** of the block type and codewords **390***a* having the minimum error.

In an alternative embodiment, the optimal analog curve may be computed **430** before searching the block type. That is, the process may compute **430** the optimal analog curve before proceeding with selecting **428** the block type, selecting **432** the partition, computing **434** the codewords, computing **436** the error, storing **438** the error, and storing **440** the block type and codeword. Computing **430** the optimal analog curve first is useful if all the, block types use the same analog curve and color space because the analog curve does not need to be recomputed for each block type.

FIG. **4**D further describes the process of identifying the optimal analog curve. The selection module **355** starts **448** the process by computing a center of gravity **450** for pixel **270** colors of an image block **260**. Computing **450** the center of gravity includes averaging the pixel **270** colors of the image block **260**. Once the center of gravity is computed **450**, the process identifies **452** a vector in color space to minimize the first moment of the pixel **270** colors of the image block **260**.

Specifically, for identifying **452** the vector the process fits a straight line to a set of data points, which are the original pixel **270** colors of the image block **260**. A straight line is chosen passing through the center of gravity of the set of points such that it minimizes the "moment of inertia" (the means square error). For example, for three pixel properties, to compute the direction of the line minimizing the moment of inertia, tensor inertia, T, is calculated from the individual colors as follows:

$$T = \sum \begin{matrix} C_{1i}^2 + C_{2i}^2 & -C_{0i}C_{1i} & -C_{0i}C_{2i} \\ -C_{0i}C_{1i} & C_{0i}^2 + C_{2i}^2 & -C_{1i}C_{2i} \\ -C_{0i}C_{2i} & -C_{2i}C_{1i} & C_{0i}^2 + C_{1i}^2 \end{matrix}$$

where C_0, C_1, and C_2 represent pixel properties, for example color components in RGB or YUV, relative to a center of gravity. In a preferred embodiment of an RGB color space, C_{0i} is the value of red, C_{1i} is the value of green, and C_{2i} is the value of blue for each pixel, i, of the image block. Further, i takes on integer values from 1 to W×H, so that if W=4 and H=4, i ranges from 1 to 16.

The eigenvector of tensor, T, with the smallest eigenvalue is calculated using conventional methods known to those skilled in the art. The eigenvector direction along with the calculated gravity center, defines the axis that minimizes the moment of inertia. This axis is used as the optimal analog curve, which in a preferred embodiment is a straight line. Those of ordinary skill in the art will readily recognize that the term optimal analog curve is not limited solely to a straight line but may include a set of parameters, such as pixel values or colors, that minimizes the moment of inertia or means square error when fitted to the center of gravity of the pixel colors in the image block. The set of parameters may define any geometric element, such as but not limited to a curve, a plane, a trapezoid, or the like.

FIG. **4**E illustrates the process undertaken by the codeword generation module **360** for selecting **432** the partitions, computing **434**, **436** the codewords for the partitions and the associated error, and storing **438**, **440** the error value, block type, and codeword if the error value is less than a previously stored error value. The process starts **456** with the codeword generation module **360** projecting **458** the W×H color values onto the previously constructed optimal analog curve. The value of W×H is the size in number of pixels **270** of an image block **260**. In a preferred embodiment, where W and H are both 4 pixels, W×H is 16 pixels.

Once the colors are projected **458** onto the analog curve, the colors are ordered **460** sequentially along that analog curve based on the position of the color on the one-dimensional analog curve. After the colors are ordered **460**, the codeword generation module **360** searches **462** for optimal partitions. That is, the codeword generation module **360** takes the W×H colors (one color associated with each pixel) that are ordered **460** along the analog curve and partitions, or groups, them into a finite number of clusters with a predefined relative spacing. In a preferred embodiment, where W=4 and H=4, so that W×H is 16, the 16 colors are placed in three or four clusters (M=3 or 4).

In conducting the search **462** for the optimal partition, the color selection module **360** finds the best M clusters for the W×H points projected onto the optimal curve, so that the error associated with the selection is minimized. The best M clusters are determined by minimizing the mean square error with the constraint that the points associated with each cluster are spaced to conform to the predefined spacing.

In a preferred embodiment, for a block type of four equidistant colors, the error may be defined as a squared error along the analog curve, such as

$$E^2=\Sigma_{cluster0}(x_i-p_0)^2+\Sigma_{cluster1}(x_i-((2/3)p_0+((1/3)p_{1))}^2+\Sigma_{cluster2}(x_i-((1/3)p_0+(2/3)p_1))^2+\Sigma_{cluster3}(x_i-p_1)^2$$

where E is the error for the particular grouping or clustering, p_0 and p_1 are the coded colors, and x_i are the projected points on the optimal analog curve.

In instances where the block type indicates three equidistant colors, the error may be defined as a squared error along the analog curve, such as

$$E^2=\Sigma_{cluster}0(x_i-p_0)^2+\Sigma_{cluster1}(x_i-((1/2)p_0+(1/2)p_1))^2+\Sigma_{cluster2}(x_i-p_1)^2$$

where, again, E is the error for the particular grouping or clustering, p_0 and p_1 are the coded colors, and x_i are the projected points on the optimal analog curve.

After the resulting **447** optimal codewords **390***a* are identified, they are forwarded to the bitmap construction module **340**. The bitmap construction module **340** uses the codewords **390***a* to identify the M colors that may be specified or inferred from those codewords **390***a*. In a preferred embodiment, the bitmap construction module **340** uses the codewords **390***a*, e.g., CW**0**, CW**1**, to identify the three or four colors that may be specified or inferred from those codewords **390***a*.

The bitmap construction module **340** constructs a block bitmap **390***b* using the codewords **390***a* associated with the image block **260**. Colors in the image block **260** are mapped to the closest color associated with one of the quantized colors specified by, or inferred from, the codewords **390***a*.

The result is a color index, referenced as ID, per pixel in the block identifying the associated quantized color.

Information indicating the block type is implied by the codewords **390***a* and the bitmap **390***b*. In a preferred embodiment, the order of the codewords **390***a* CW**0**, CW**1**, indicate the block type. If a numerical value of CW**0** is greater than a numerical value of CW**1**, the image block is a four color block. Otherwise, the block is a three color plus transparency block.

As discussed above, in a preferred embodiment, there are two image block types. One image block type has four equidistant colors, while the other image block type has three equidistant colors with the fourth color index used to specify that a pixel is transparent. For both image block types the color index is two bits.

The output of the bitmap construction module **340** is an encoded image block **390** having the M codewords **390***a* plus the bitmap **390***b*. Each encoded image block **390** is received by the encoded image composer **319** that, in turn, orders the encoded image blocks **390** in a file. In a preferred embodiment, the encoded image blocks **390** are ordered from left to right and from top to bottom in the same order as the blocks were broken down by the block decomposer **315**. The ordered file having the encoded image blocks **390** is concatenated with the header information **385***a* that is derived from the header **380***a* of the original image **310** to generate the encoded image data **385** that is the image encoder system **220** output **320**. The image encoder system **220** output **320** may be forwarded to the memory **115**, the storage device **120**, or the output device **130**, for example.

The image encoder system **220** of the present invention advantageously reduces the effective data size of an image, for example, from 24-bits per pixel to 4-bits per pixel. Further, the present invention beneficially addresses transparency issues by allowing for codewords to be used with a transparency identifier.

FIG. **5**A is a block diagram of an image decoder system **230** in accordance with the present invention. The image decoder system **230** includes an encoded image decomposing unit **501**, a header converter **508**, one or more block decoders **505** (**505***a*–**505***m*, where m is any positive integer value representing the last block decoder), and an image composer **504**. The encoded image decomposer **501** is coupled to receive the encoded image data **385** that was output **320** from the image encoder system **220**. The encoded image decomposer **501** is coupled to the one or more block decoders **505***a*–**505***m*. The one or more block decoders **505***a*–**505***m* are coupled to the image composer **504** that, in turn, is coupled to the output $^{240.}$

The encoded image decomposer **501** receives the encoded image data **385** and decomposes, or breaks, it into its header **385***a* and the encoded image blocks **390**-**1**–**390**-R. The encoded image decomposer **501** reads the modified header **385***a* of the encoded image data **385** and forwards the modified header **385***a* to the header converter **508**. The encoded image decomposer **501** also decomposes the encoded image data **385** into the individual encoded image blocks **390**-**1**–**390**-R that are forwarded to the one or more block decoders **505***a*–**505***m*.

The header converter **508** converts the modified header **385***a* to an output header. Simultaneously, the encoded image blocks **390**-**1**–**390**-R are decompressed or decoded by the one or more block decoders **505***a*–**505***m*. It is noted that the each encoded image block **390** may be processed sequentially in one block decoder **505***a* or multiple encoded image blocks **390**-**1**–**390**-R may be processed in parallel with one block decoder **505***a*–**505***m* for each encoded image block **390**-**1**–**390**-R. Thus, multiple block decoders **505***a*–**505***m* allows for parallel processing that increases the processing performance and efficiency of the image decoder system **230**.

The image composer **504** receives each decoded image block from the one or more block decoders **505***a*–**505***m* and orders them in a file. Further, the image composer **504** receives the converted header from the header converter **508**. The converted header and the decoded image blocks are placed together to generate output **240** data representing the original image **310**.

FIG. **5**B is a block diagram of a first embodiment of a block decoder **505** in accordance with the present invention. Each block decoder **505***a*–**505***m* includes a block type detector **520**, one or more decoder units, e.g., **533***a*-**1** to **533***a*-k (k is any integer value), and an output selector **523**. The block type detector **520** is coupled to the encoded image decomposer **501**, the output selector **523**, and each of the one or more decoder units, e.g., **533***a*-**1**–**533***a*-k. Each of the decoder units, e.g., **533***a*-**1**–**533***a*-k, is coupled to the output selector **523** that, in turn, is coupled to the image composer **504**.

The block type detector **520** receives the encoded image blocks **390** and determines the block type for each encoded image block **390**. Specifically, the block type detector **520** passes a selector signal to the output selector **523** that will be used to select an output corresponding to the block type detected. The block type is detected based on the codewords **390***a*. After the block type is determined, the encoded image blocks **390** are passed to each of the decoder units, e.g., **533***a*-**1**–**533***a*-k The decoder units, e.g., **533***a*-**1**–**533***a*-k, decompress or decode each encoded image block **390** to generate the colors for the particular encoded image block **390**. The decoder units, e.g., **533***a*-**1**–**533***a*-k, may be c-channels wide (one channel for each color component (or pixel property) being encoded), where c is any integer value. Using the selector signal, the block type detector **520** enables the output selector **523** to output the color of the encoded image block **390** from one of the decoder units, e.g., **533***a*-**1**–**533***a*-k that corresponds with the block type detected by the block type detector **520**. Alternatively, using the selector signal, the appropriate decoder unit **533** could be selected so the encoded block is processed through that decoder unit only.

FIG. **5**C is a block diagram of a second embodiment of a block decoder **505** in accordance with the present invention. In a second embodiment, the block decoder **505** includes a block type detector **520**, a first and a second decoder unit **530**, **540**, and the output selector **523**. The block type detector **520** is coupled to receive the encoded image blocks **390** and is coupled to the first and the second decoder units **530**, **540** and the output selector **523**.

The block type detector **520** receives the encoded image blocks **390** and determines, by comparing the codewords **390***a* of the encoded image block **390**, the block type for each encoded image block **390**. For example, in a preferred embodiment, the block type is four quantized colors or three quantized colors and a transparency. Once the block type is selected and a selector signal is forwarded to the output selector **523**, the encoded image blocks **390** are decoded by the first and the second decoder units **530**, **540**. The first and the second decoder units **530**, **540** decode the encoded image block **390** to produce the pixel colors of each image block. The output selector **523** is enabled by the block type detector **520** to output the colors from the decoder unit **530**, **540** that corresponds to the block type selected.

FIG. **5**D is a logic diagram illustrating one embodiment of a decoder unit through a red-channel of the that decoder unit in accordance with the present invention. Specifically, the decoder unit is similar to the decoder units **530**, **540** illustrated in FIG. **5**C. Moreover, the functionality of each of those decoder units **530**, **540** is merged into the single logic diagram illustrated in FIG. **5**D. Further, those skilled in the art will understand that although described with respect to the red-channel of the decoder units **530**, **540** the remaining channels, e.g., the green-channel and the blue-channel, in each decoder unit **530**, **540** are similarly coupled and functionally equivalent.

The logic diagram illustrating the decoder units **530**, **540** is shown to include portions of the block type detector **520**, for example a comparator unit **522**. The comparator unit **522** works with a first 2×1 multiplexer **525***a* and a second 2×1 multiplexer **525***b*. The comparator unit **522** is coupled to the first and the second 2×1 multiplexers **525***a*, **525***b*. Both 2×1 multiplexers **525***a*, **525***b* are coupled to a 4×1 multiplexer **526** that serves to select the appropriate color to output.

The red-channel **544**, **546** of the first decoder unit **530** includes a first and a second red-channel line **551***a*, **551***b* and a first and a second red-color block **550***a*, **550***b*. Along the path of each red-color block **550***a*, **550***b* is a first full adder **552***a*, **552***b*, a second full adder **554***a*, **554***b*, and a CLA ("carry-look ahead") adder **556***a*, **556***b*. The first and the second red-channel lines **551***a*, **551***b* are coupled to the first and the second red-color blocks **550***a*, **550***b*, respectively. Each red-color block **550***a*, **550***b* is coupled to the first full adder **552***a*, **552***b* associated with that red-color block **550***a*, **550***b*. Each first full adder **552***a*, **552***b* is coupled to the respective second full adder **554***a*, **554***b*. Each second full adder **554***a*, **554***b* is coupled to the respective CLA adder **556***a*, **556***b*.

The second decoder unit **540** comprises the first and the second red-channel lines **551***a*, **551***b* and the respective first and second red-color blocks **550***a*, **550***b* and an adder **558**. The first and the second channel lines **551***a*, **551***b* are coupled to their respective red-color blocks **550***a*, **550***b* as described above. Each red-color block **550***a*, **550***b* is coupled to the adder **558**.

The CLA adder **556***a* from the path of the first red-color block **550***a* of the first decoder unit **530** is coupled to the first 2×1 multiplexer **525***a* and the CLA adder **556***b* from the path of the second red-color block **550***b* of the first decoder unit **530** is coupled to the second 2×1 multiplexer **525***b*. The adder **558** of the second decoder unit **540** is coupled to both the first and the second 2×1 multiplexers **525***a*, **525***b*.

The 4×1 multiplexer **526** is coupled to the first and the second red-channel lines **551***a*, **551***b*, as well as to the first and the second 2×1 multiplexers **525***a*, **525***b*. The 4×1 multiplexer **526** is also coupled to receive a transparency indicator signal that indicates whether or not a transparency (no color) is being sent. The 4×1 multiplexer **526** selects a color for output based on the value of the color index, referenced as the ID signal, that references the associated quantized color for an individual pixel of the encoded image block **390**.

FIG. **6**A is a flow diagram illustrating operation of the decoder system **230** in accordance with the present invention. For purposes of illustration only, the process for the decoder system **230** will be described with a single block encoder **505** having two decoding units, e.g., **530**, **540**. Those skilled in the art will recognize that the process is functionally equivalent for decoder systems having more than one block decoder **505** and more than one decoder units, e.g., **533***a*-**1**–**533***a*-k.

The process starts **600** with the encoded image decomposer **501** receiving **605** the encoded, or compressed, image

data **385** from the encoder system **220**, for example, through the memory **115** or the storage device **120**. The encoded image decomposer **501** decomposes **610** the encoded image data **385** by forwarding the modified header **385***a* to the header converter **508**. In addition, the encoded image decomposer **501** also decomposes **610** the encoded image data **385** into the individual encoded image blocks **390**-**1**–**390**-R.

The header converter **508** converts **612** the header information to generate an output header that is forwarded to the image composer **504**. Simultaneously, the one or more block decoders **505***a*–**505***m* decodes **615** the pixel colors for each encoded image block **390**. It is again noted that each encoded image block **390** may be decoded **615** sequentially in one block decoder **505***a* or multiple encoded image blocks **390**-**1**–**390**-R may be PATENT decoded **615** in parallel in multiple block decoders **505***a*–**505***m*, as described above. The process for decoding the encoded image blocks **390** is further described in FIG. **6**B. Each decoded **615** image block is then composed **620** into a data file with the converted **612** header information by the image composer **504**. The image composer **504** generates the data file as an output **625** that represents the original image **310**.

FIG. **6**B is a flow diagram illustrating operation of the block encoder **505** in accordance with the present invention. Once the process is started **630**, each encoded image block **390** is received by the block decoder **505** and the block type for each encoded image block **390** is detected **640**. Specifically, for a preferred embodiment the first and the second codewords **390***a*, CW**0**, CW**1**, respectively, are received **635** by the block type detector **520** of the block decoder **505**. As discussed above, comparing the numerical values of CW**0** and CW**1** reveals the block type.

In addition, the first five bits of each codeword **390***a*, e.g., CW**0**, CW**1**, that represent the red-channel color are received by the red-channel **545** of each of the first and the second decoder units **530**, **540**, the second 6-bits of each codeword **390***a* CW**0**, CW**1** that represent the green-channel color are received by the green-channel of each of the first and the second decoder units **530**, **540**, and the last 5-bits of each codeword **390***a* CW**0**, CW**1** that represent the blue-channel color are received by the blue-channel of each of the first and the second decoder units **530**, **540**.

The block type detector **520** detects **640** the block type for an encoded image block **390**. Specifically, the comparator **522** compares the first and the second codewords **390***a*, CW**0**, CW **1**, and generates a flag signal to enable the first 2×1 multiplexers **525***a* or the second 2×1 multiplexers **525***b* which, in turn, selects **645** either the first decoding unit **530** or the second decoding unit **540**, respectively. The process then calculates **650** the quantized color levels for the decoder units **530**, **540**.

To calculate **650** the quantized color levels, the first decoding unit **530** calculates the four colors associated with the two codewords **390***a*, CW**0**, CW**1**, using the following relationship:

$$CW0=\text{first codeword}=\text{first color};$$

$$CW1=\text{second codeword}=\text{second color};$$

$$CW2=\text{third color}=(2/3)CW0+(1/3)CW1;$$

$$CW3=\text{fourth color}=(1/3)CW0+(2/3)CW1.$$

In one embodiment, the first decoder unit **530** may estimate the above equations for CW**2** and CW**3**, for example, as follows:

$$CW2=(5/8)CW0+(3/8)CW1;\text{ and}$$

$$CW3=(3/8)CW0+(5/8)CW1.$$

The red-color blocks **550***a*, **550***b* serve as a one-bit shift register to get (½)CW**0** or (½)CW**1** and each full adder **552***a*, **552***b*, **554***a*, **554***b* also serves to shift the signal left by 1-it. Thus, the signal from the first full adders **552***a*, **552***b* is (¼)CW**0** or (¼)CW**1**, respectively, because of a two-bit overall shift and the signal from the second full adders **554***a*, **554***b* is (⅛)CW**0** or (⅛)CW**1**, respectively, because of a three-bit overall shift. These values allow for the above approximations for the color signals.

The second decoder unit **540** calculates **650** three colors associated with the codewords **390***a*, CW**0**, CW**1**, and includes a fourth signal that indicates a transparency is being passed. The second decoder unit **540** calculates colors, for example, as:

$CW0$=first codeword=first color;

$CW1$=second codeword=second color;

$CW3$=third color=$(1/2)CW0+(1/2)CW1$; and

T=Transparency.

In one embodiment the second decoder unit **540** has no approximation because the signals received from the red-color blocks **550***a*, **550***b* is shifted left by one-bit so that the color is already calculated to (½)CW**0** and (½)CW**1**, respectively.

After the quantized color levels for the selected **645** decoder unit **530**, **540** have been calculated **650**, each bitmap value for each pixel is read **655** from the encoded image data block **385**. As each index is read **655** it is mapped **660** to one of the four calculated colors if the first decoder unit **530** is selected **645** or one of the three colors and transparency if the second decoder unit **540** is selected. The mapped **660** colors are selected by the 4×1 multiplexer **526** based on the value of the ID signal from the bitmap **390***b* of the encoded image block **390**. As stated previously, a similar process occurs for selection of colors in the green-channel and the blue-channel.

As the colors are output from the red-, green-, and blue-channels, the output is received by the image composer **504**. The image composer **504** orders the output from the block encoders **505** in the same order as the original image **310** was decomposed. The resulting **665** image that is output from the image decoder system **230** is the original image that is forwarded to an output source **240**, e.g., a computer screen, which displays that image.

The system and method of the present invention beneficially allows for random access to any desired image block **260** within an image, and any pixel **270** within an image block **260**. FIG. 7A is a block diagram of a subsystem **700** that provides random access to a pixel **270** or an image block **260** in accordance with the present invention. PATENT The random access subsystem **700** includes a block address computation module **710**, a block fetching module **720**, and the one or more block decoders **505**. The block address computation module **710** is coupled to receive the header information **385***a* of the encoded image data **385**. The block address computation module **710** is also coupled to the block fetching module **720**. The block fetching module **720** is coupled to receive the encoded image block portion **390-1**-R of the encoded image data **385**. The block fetching module **720** is also coupled to the block decoders **505**.

FIG. 7B is a flow diagram illustrating a process of random access to a pixel **270** or an image block **260** using the random access subsystem **700** in accordance with the

present invention. When particular pixels **270** have been identified for decoding, the process starts **740** with the image decoder system **230** receiving the encoded image data **385**. The modified header **385***a* of the encoded image data **385** is forwarded to the block address computation module **710** and the encoded image block portion **390-1**-R of the encoded image data **385** is forwarded to the block fetching module **720**.

The block address computation module **710** reads the modified header **385***a* to compute **745** the address of the encoded image block portion **390-1**-R having the desired pixels **270**. The address computed **745** is dependent upon the pixel coordinates within an image. Using the computed **745** address, the block fetching module **720** identifies the encoded image block **390** of the encoded image block portion **390-1**-R that has the desired pixels **270**. Once the encoded image block **390** having the desired pixels **270** has been identified, only the identified encoded image block **390** is forwarded to the block decoders **505** for processing.

Similar to the process described above in FIG. 6B, the block decoders **505** compute **755** the quantized color levels for the identified encoded image blocks **390** having the desired pixels. After the quantized color levels have been computed **755**, the color of the desired pixel is selected **760** and output **765** from the image decoder system **230**.

Random access to pixels **270** of an image block **260** advantageously allows for selective decoding of only needed portions or sections of an image. Random access also allows the image to be decoded in any order the data is required. For example, in three-dimensional texture mapping only portions of the texture may be required and these portions will generally be required in some non-sequential order. Thus, the present invention increases processing efficiency and performance when processing only a portion or section of an image.

The present invention beneficially encodes, or compresses, the size of an original image **310** from 24-bits per pixel to an aggregate 4-bits per pixel and then decodes, or decompresses the encoded image data **385** to get a representation of the original image **310**. Further, the claimed invention uses, for example, two base points or codewords from which additional colors are derived so that extra bits are not necessary to identify a pixel **270** color.

Moreover, the present invention advantageously accomplishes the data compression on an individual block basis with the same number of bits per block so that the compression rate can remain fixed. Further, because the blocks are of fixed size with a fixed number of pixels **270**, the present invention beneficially allows for random access to any particular pixel **270** in the block. The present invention provides for an efficient use of system resources because entire blocks of data are not retrieved and decoded to display data corresponding to only a few pixels **270**.

In addition, the use of a fixed-rate 64-bit data blocks in the present invention provides the advantage of having simplified header information that allows for faster processing of individual data blocks. Also, a 64-bit data block allows for data blocks to be processed rapidly, e.g., within one-clock cycle, as the need to wait until a full data string is assembled is eliminated. Further, the present invention also reduces the microchip space necessary for a decoder system because the decoder system only needs to decode each pixel to a set of colors determined by, e.g., the two codewords.

While particular embodiments and applications of the present invention have been illustrated and described, it is to be understood that the invention is not limited to the precise construction and components disclosed herein and that various modifications, changes and variations which will be apparent to those skilled in the art may be made in the arrangement, operation and details of the method and apparatus of the present invention disclosed herein without departing from the spirit and scope of the invention as defined in the appended claims.

What is claimed is:

1. A data format for representing an original image block having a pixel color set, comprising:

a codeword portion for storing at least two codewords;

a bitmap portion for storing a set of indices, the bitmap portion constructed by a bitmap construction module utilizing the codeword portion associated with the bitmap portion; and

wherein said codewords define at least three colors that approximate the pixel color set, and said indices map the pixel color set to at least one of said at least three colors.

2. The data format of claim **1**, wherein said set of indices includes a predefined index.

3. The data format of claim **2**, wherein said predefined index is for mapping a transparency identifier associated with the original image block.

4. The data format of claim **2**, wherein said predefined index is for mapping an alpha value associated with the original image block.

5. The data format of claim **2**, wherein said predefined index is for mapping a color key value associated with the original image block.

6. The data format of claim **1**, wherein said codeword portion includes a first portion for storing a first codeword and a second portion for storing a second codeword; and wherein said first codeword and said second codeword are used to indicate a block type for the original image block.

7. The data format of claim **1**, wherein said codeword portion includes a first portion for storing a first codeword and a second portion for storing a second codeword; and wherein said at least three colors includes at least two computed colors if said first codeword is greater than said second codeword.

8. The data format of claim **1**, wherein said codeword portion includes a first portion for storing a first codeword and a second portion for storing a second codeword; and wherein said at least three colors includes at least one computed color and said set of indices includes a predefined index if said first codeword is less than said second codeword.

9. The data format of claim **1**, wherein said at least three colors are computed using a geometric element fitted to said pixel color set so that said geometric element has a minimal moment of inertia.

10. The data format of claim **1**, wherein said at least three colors includes one of said at least two codewords.

11. A data format for representing an original image block having a pixel color set, comprising:

a codeword portion for storing at least one codeword;

a bitmap portion for storing a set of indices, said set of indices includes an available index for representing a transparency identifier, the bitmap portion constructed by a bitmap construction module utilizing the codeword portion associated with the bitmap portion; and

wherein said codeword defines a set of colors that approximate the pixel color set, and said indices map the pixel color set to at least one color in said set of colors.

12. The data format of claim **11**, wherein said set of colors includes said at least one codeword and a computed color.

13. The data format of claim **11**, wherein said set of colors includes at least three colors.

14. A data format for representing an original image block having a pixel color set, comprising:

a codeword portion for storing at least one codeword;

a bitmap portion for storing a set of indices;

wherein said at least one codeword defines a set of colors that approximate the pixel color set, and said indices map the pixel color set to at least one color in said set of colors; and

wherein said set of colors are computed using a geometric element fitted to said pixel color set so that said geometric element has a minimal moment of inertia.

15. An encoded image data format for representing an original image partitioned into at least two image blocks, said image blocks each having a corresponding pixel color set, the data format comprising:

at least two encoded image block portions, one of said encoded image block portions having a codeword portion for storing at least two codewords, and a bitmap portion for storing a set of indices, the bitmap portion constructed by a bitmap construction module utilizing the codeword portion associated with the bitmap portion; and

wherein said at least two codewords define at least three colors that approximate the pixel color set of one of the original image blocks, and said indices map the pixel color set to at least one of said at least three colors.

16. The data format of claim **15**, further including a header portion.

17. The data format of claim **15**, wherein said set of indices includes a predefined index.

18. The data format of claim **17**, wherein said predefined index is for mapping a transparency identifier associated with the original image block.

19. The data format of claim **17**, wherein said predefined index is for mapping an alpha value associated with the original image block.

20. The data format of claim **17**, wherein said predefined index is for mapping a color key value associated with the original image block.

21. The data format of claim **15**, wherein said set of colors are computed using a geometric element fitted to said pixel color set so that said geometric element has a minimal moment of inertia.

22. The data format of claim **15**, wherein said at least three colors includes one of said at least two codewords.

23. An encoded image data format for representing an original image partitioned into at least a first image block having a first pixel color set and a second image block having a second pixel color set, the data format comprising:

a first encoded image block having a first portion for storing a first codeword, a second codeword, and a first bitmap portion for storing a first set of indices;

a second encoded image block having a second portion for storing a third codeword, a fourth codeword, and a second bitmap portion for storing a second set of indices;

wherein said first and second codewords define a first set of colors that approximate the first pixel color set, and said first set of indices map the first set of colors to the first pixel color set; and

wherein said third and fourth codewords define a second set of colors that approximate the second pixel color set, and said second set of indices map the second set of colors to the second pixel color set.

24. The encoded image data format of claim **23**, wherein said first set of colors includes at least three colors.

25. The data format of claim **23**, wherein said first set of colors includes at least said first codeword.

26. The data format of claim **23**, wherein said first set of colors includes said second codeword.

27. The data format of claim **23**, wherein said second set of colors includes at least three colors.

28. The data format of claim **23**, wherein said second set of colors includes said third codeword.

29. The data format of claim **23**, wherein said second set of colors includes said fourth codeword.

* * * * *

Exhibit C

US006775417B2

(12) **United States Patent**
Hong et al.

(10) **Patent No.: US 6,775,417 B2**
(45) **Date of Patent: Aug. 10, 2004**

(54) FIXED-RATE BLOCK-BASED IMAGE COMPRESSION WITH INFERRED PIXEL VALUES

(75) Inventors: **Zhou Hong**, Cupertino, CA (US); **Konstantine I. Iourcha**, San Jose, CA (US); **Krishna S. Nayak**, Palo Alto, CA (US)

(73) Assignee: **S3 Graphics Co., Ltd.**, Grand Cayman (KN)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 165 days.

(21) Appl. No.: **10/052,613**

(22) Filed: **Jan. 17, 2002**

(65) **Prior Publication Data**

US 2003/0053706 A1 Mar. 20, 2003

(Under 37 CFR 1.47)

Related U.S. Application Data

(63) Continuation-in-part of application No. 09/351,930, filed on Jul. 12, 1999, now Pat. No. 6,658,146, which is a continuation of application No. 08/942,860, filed on Oct. 2, 1997, now Pat. No. 5,956,431.

(51) **Int. Cl.**^{7} **G06K 9/38**
(52) **U.S. Cl.** **382/253**; 382/232; 725/146
(58) **Field of Search** 382/166, 162, 382/232, 253; 725/146; 345/549, 550; 358/539

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,821,208	A	4/1989	Ryan et al.	345/550
4,887,151	A	12/1989	Wataya	358/539
5,734,744	A	3/1998	Wittenstein et al.	382/166
5,742,892	A	4/1998	Chaddha	725/146
5,748,904	A	5/1998	Huang et al.	345/544
5,787,192	A	7/1998	Takaichi et al.	382/166
5,822,465	A	10/1998	Normile et al.	382/253
5,956,425	A	9/1999	Yoshida	382/234
5,956,431	A	9/1999	Iourcha et al.	382/253
6,075,619	A	6/2000	Iizuka	382/166

FOREIGN PATENT DOCUMENTS

JP	401284188 A *	11/1989	H04N/7/137
JP	405216993	8/1993	G06F/15/70

OTHER PUBLICATIONS

Feng et al., "A Dynamic Address Vector Quantization Based on Inter–Block and Inter–Color Correction for Color Image Coding,"IEEE International Conference on Acoustics, Speech and Signal Processing, vol. 3, May 1989, pp. 1755–1758.

Schilling et al., "Texram: A Smart Memory for Texturing," IEEE Computer Graphics & Applications, May 1996, vol. 16, No. 3, pp. 9–19.

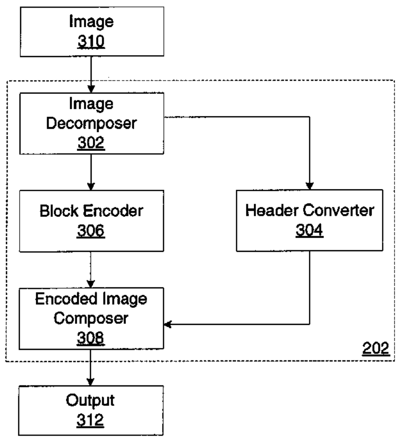
(List continued on next page.)

Primary Examiner—Anh Hong Do
(74) *Attorney, Agent, or Firm*—Carr & Ferrell LLP

(57) **ABSTRACT**

An image processing system including an image encoder and image decoding system is provided. The image encoder system includes an image decomposer, a block encoder, and an encoded image composer. The image decomposer decomposes the image into blocks. The block encoder which includes a selection module, a codeword generation module and a construction module, processes the blocks. Specifically, the selection module computes a set of parameters from image data values of a set of image elements in the image block. The codeword generation module generates codewords which the construction module uses to derive a set of quantized image data values. The construction module then maps each of the image element's original image data values to an index to one of the derived image data values. The image decoding system reverses this process to reorder decompressed image blocks in an output data file.

30 Claims, 16 Drawing Sheets



OTHER PUBLICATIONS

Knittel et al., "Hardware and Software for Superior Texture Performance," In 10: Eurographics Hardware Workshop 1995, Maastricht, The Netherlands, Aug. 28–29, 1995, pp. 1–8.

Campbell et al., "Two Bit/Pixel Full Color Encoding," Computer Graphics (Proc. SIGPRAPH 1986), Aug. 18–22, 1986, vol. 20, No. 4, Dallas, TX, pp.215–219.

Yang et al., "Hybrid Adaptive Block Truncation Coding for Image Compression," Optical Engineering, Soc. of Photo–Optical Instrumentation Engineers, Bellingham, USA, Vo. 36, No. 4, Apr. 1, 1997, pp. 1021–1027.

Kugler, "High–Performance Texture Decompression Hardware," Visual Computer, Springer, Berlin, Germany, vol. 13, No. 2, 1997, pp. 51–63.

Panos Nasiopoulos et al., "Adaptive Compression Coding," IEEE Transactions on Communications, IEEE Inc., New York, USA, vol. 39, No. 8, Aug. 1, 1991, pp. 1245–1254.

Delp E.J. et al., "Image Compression Using Block Truncation Coding," IEEE Inc., New York, USA, vol. COM–27, No. 9, Sep. 1979, pp. 1335–1342.

Yang et al., "Use of Radius Weighted Mean to Cluster Two–Class Data," Electronics Letters, IEE Stevenage, Great Britain, vol. 30, No. 10, May 12, 1994, pp. 757–759.

Russ, J.C. et al., "Optimal Grey Scale Images from Multiplane Color Images," Journal of Computer–Assisted Microscopy, Dec. 1995, Plenum, USA, vol. 7, No. 4, pp. 221–233.

Knittel et al., "Hardware for Superior Texture Performance," Eurographics Workshop on Graphics Hardware, Jul. 28, 1995, pp. 33–40.

* cited by examiner

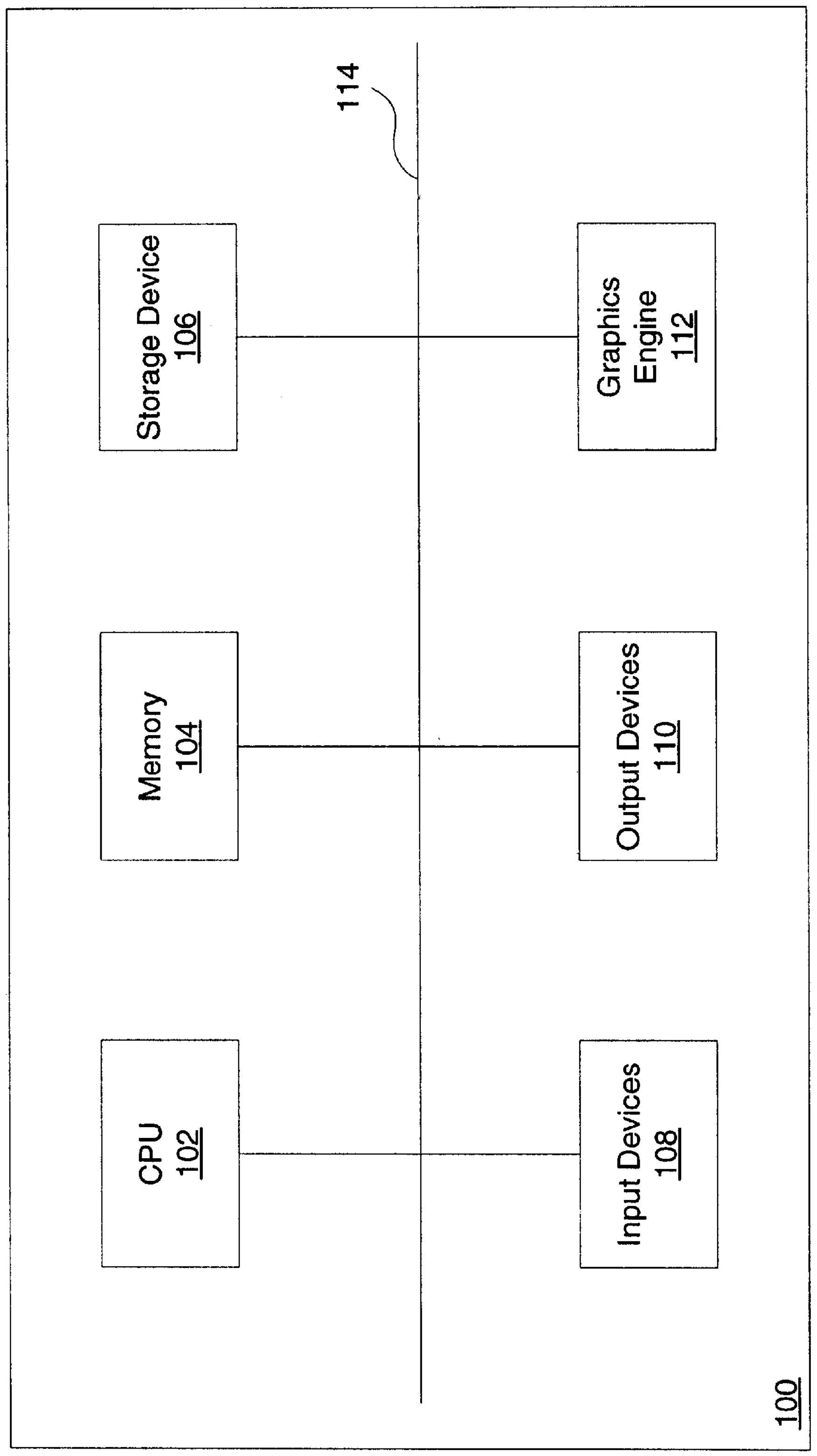

Storage Device
106
Graphics Engine
112
Memory
104
Output Devices
110
CPU
102
Input Devices
108
114
100

FIG. 1

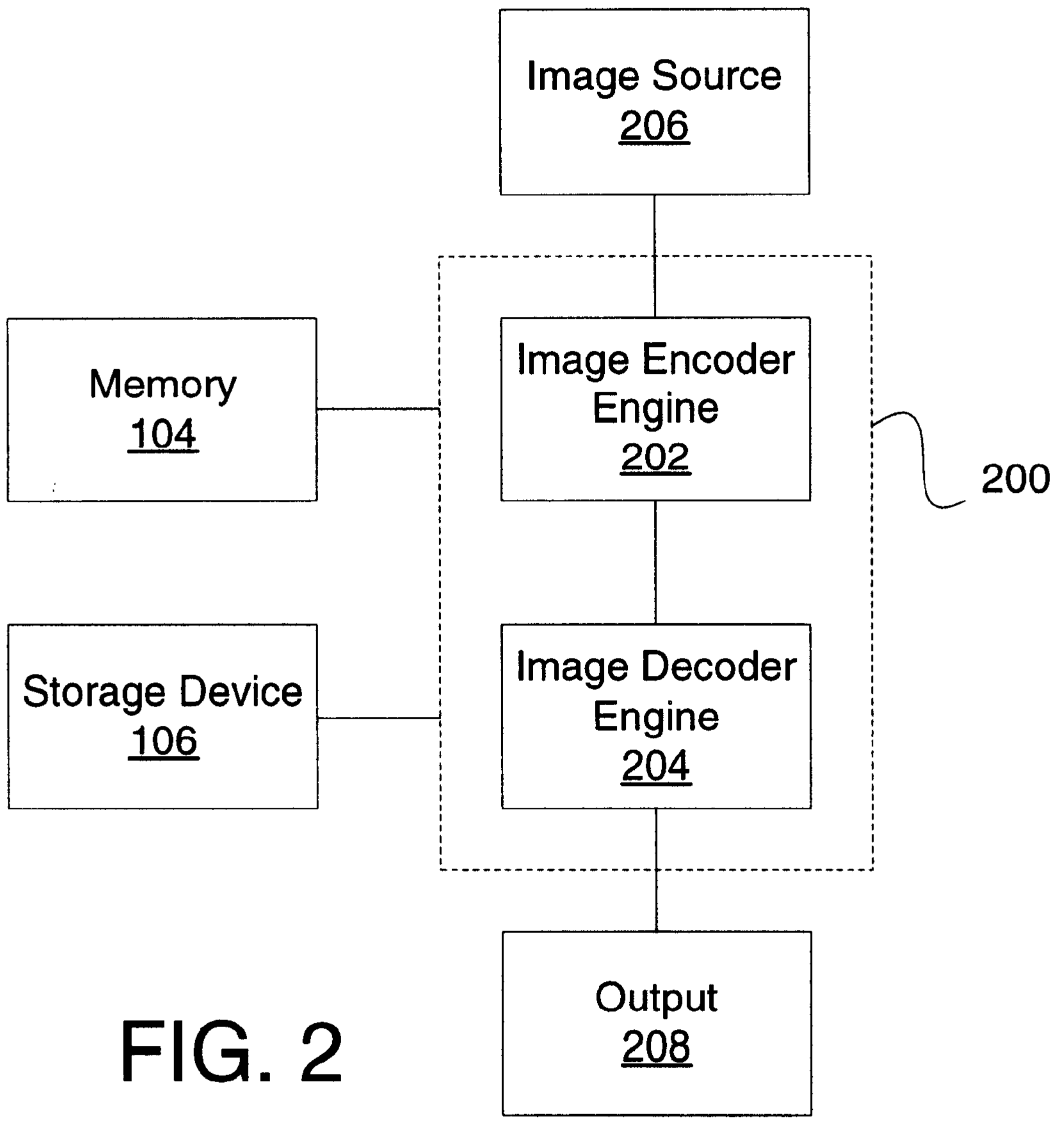


Image Source
206
Memory
104
Image Encoder Engine
202
200
Storage Device
106
Image Decoder Engine
204
Output
208

FIG. 2

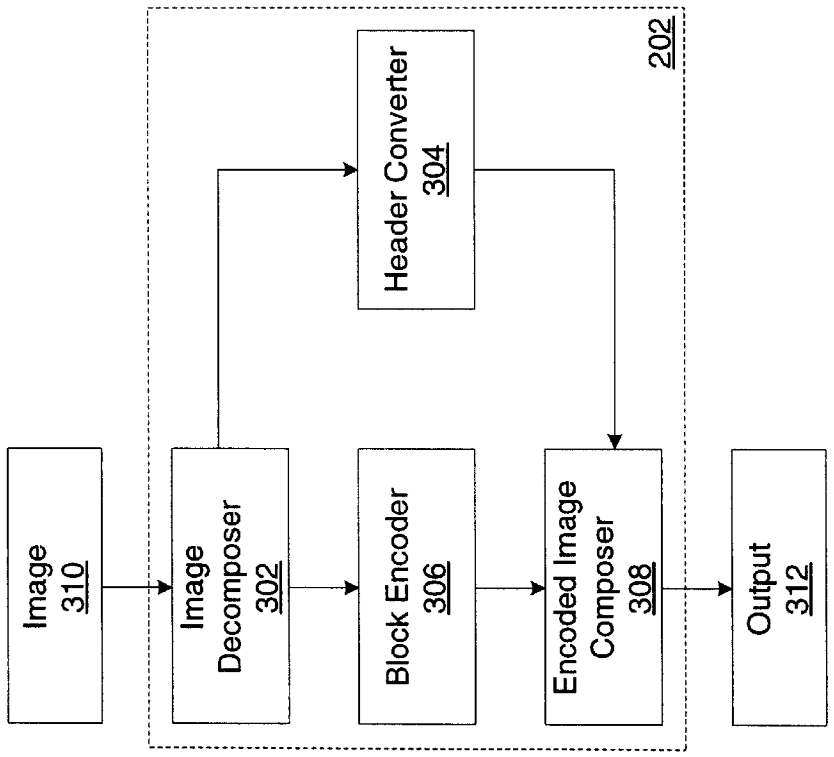


Image
310
Image
Decomposer
302
Block Encoder
306
Encoded Image
Composer
308
Output
312
Header Converter
304
202

FIG. 3A

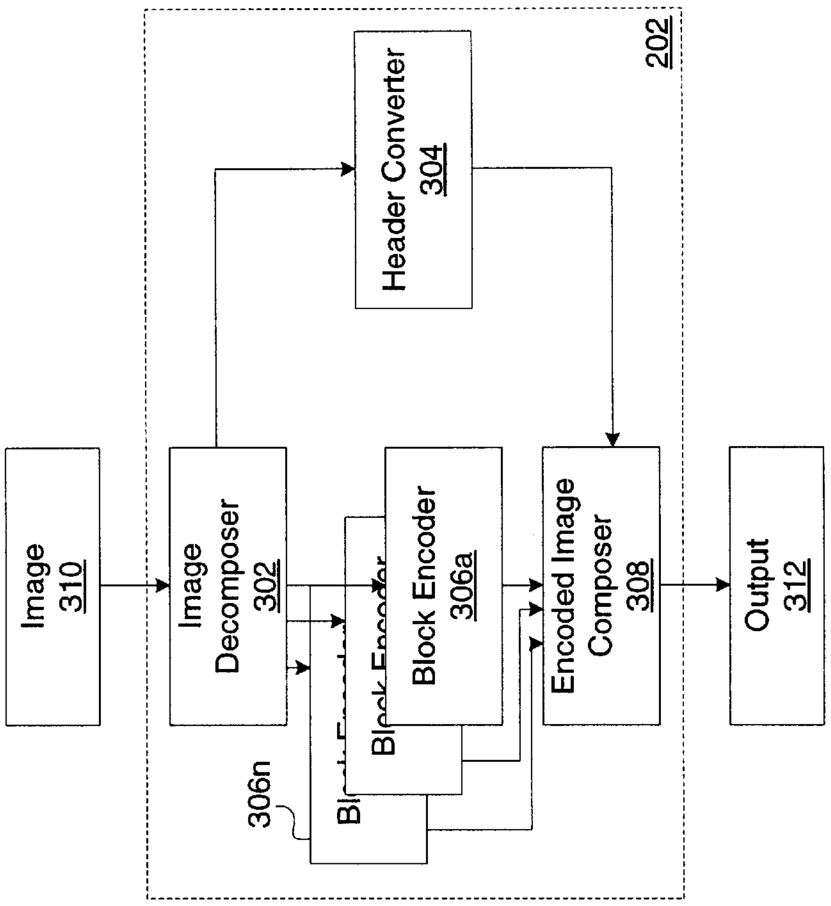


Image
310
Image
Decomposer
302
Block Encoder
306a
306n
Encoded Image
Composer
308
Output
312
Header Converter
304
202

FIG. 3B

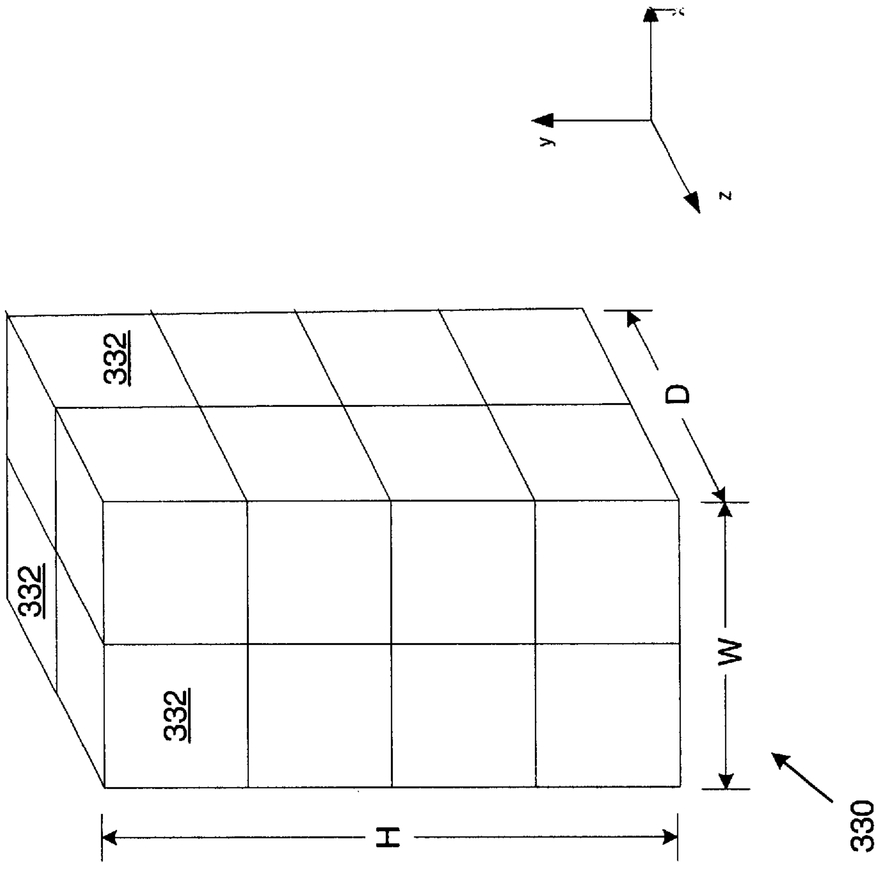


FIG. 3D

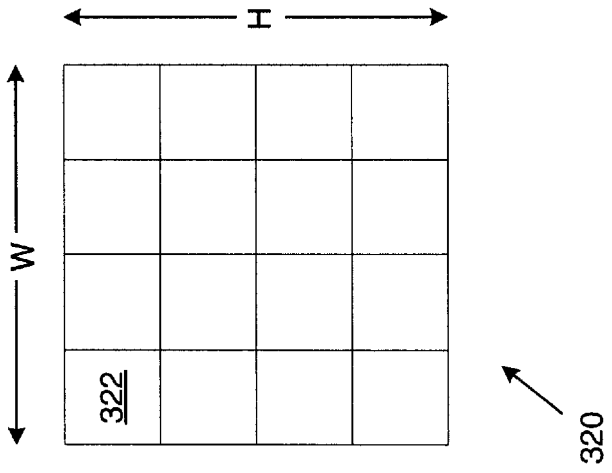


FIG. 3C

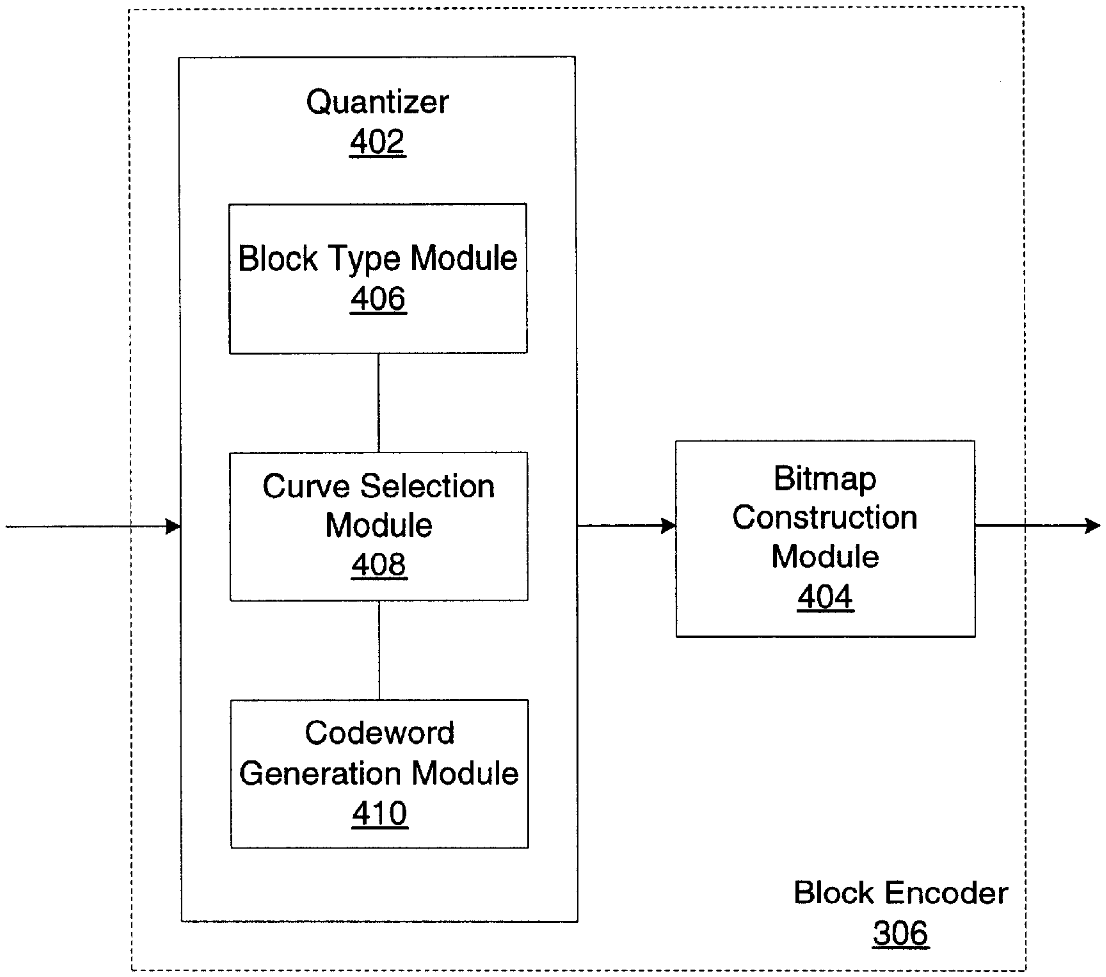

Quantizer
402
Block Type Module
406
Curve Selection Module
408
Codeword Generation Module
410
Bitmap Construction Module
404
Block Encoder
306

FIG. 4

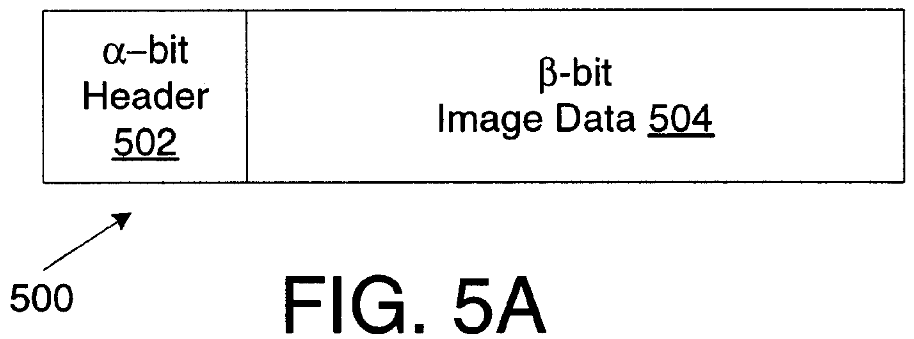


FIG. 5A

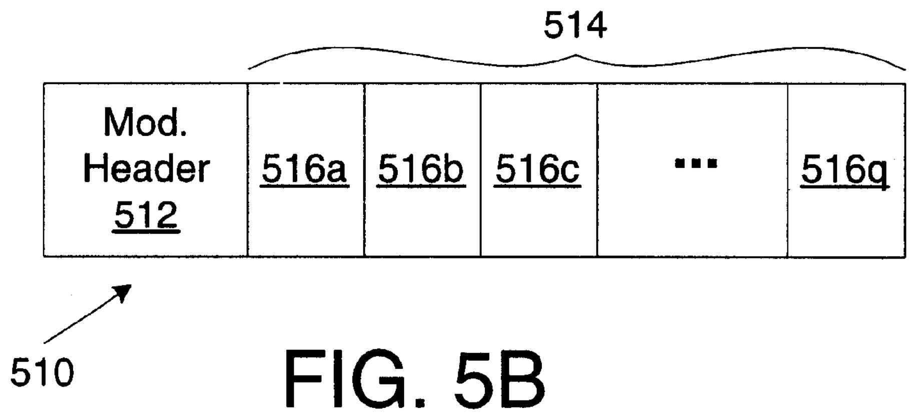


FIG. 5B

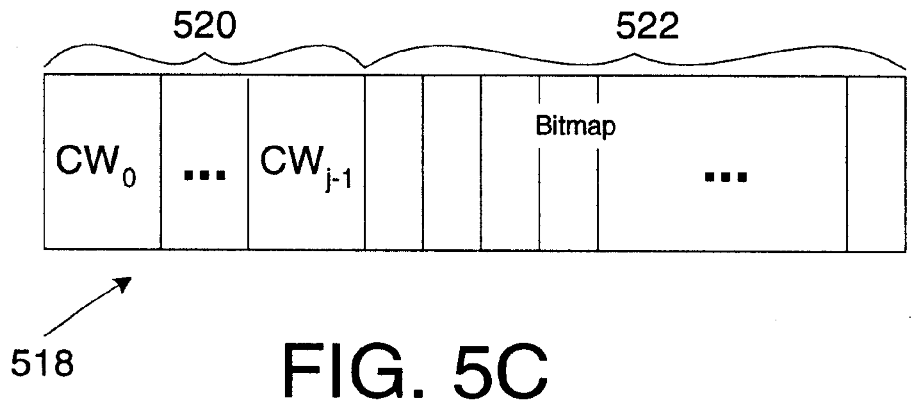


FIG. 5C

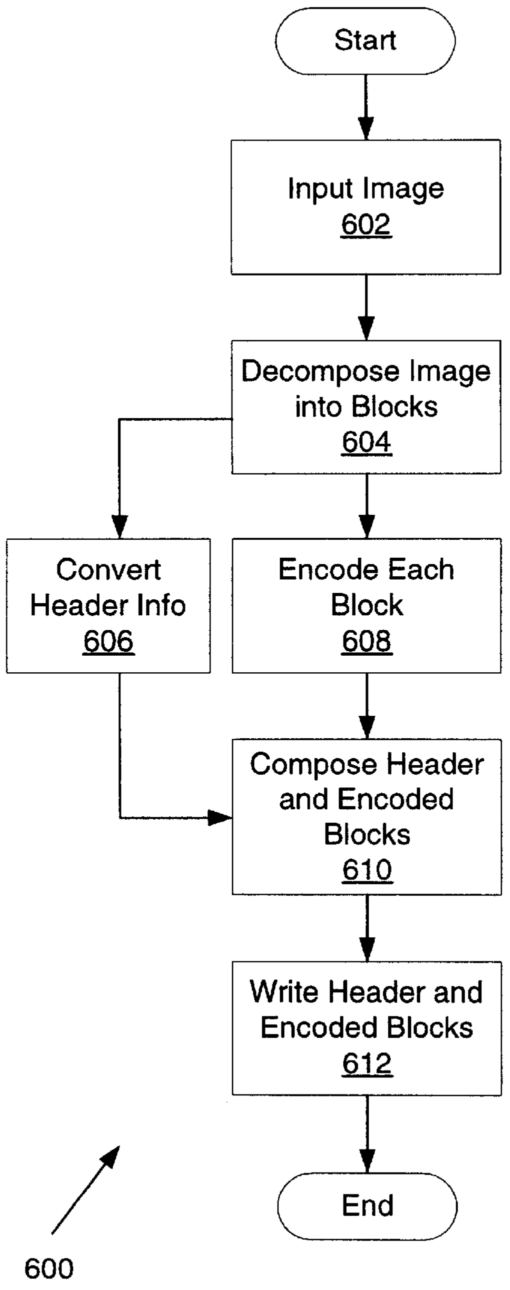

Start
Input Image 602
Decompose Image into Blocks 604
Convert Header Info 606
Encode Each Block 608
Compose Header and Encoded Blocks 610
Write Header and Encoded Blocks 612
End
600

FIG. 6A

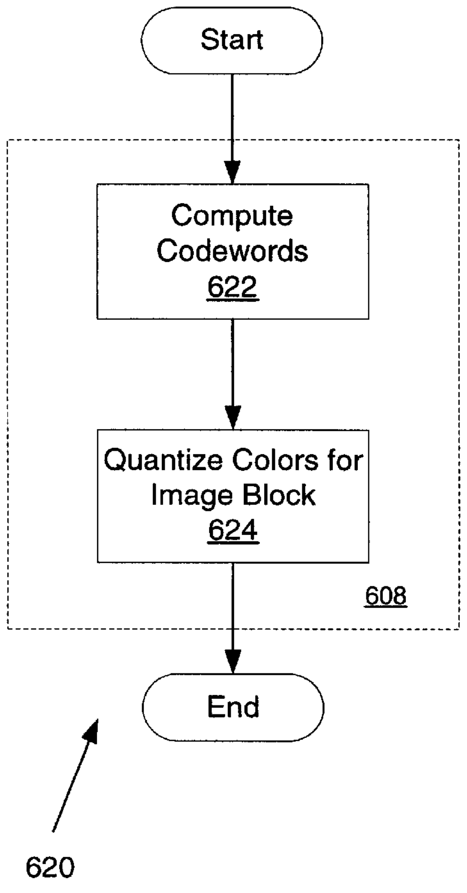

Start
Compute Codewords 622
Quantize Colors for Image Block 624
608
End
620

FIG. 6B

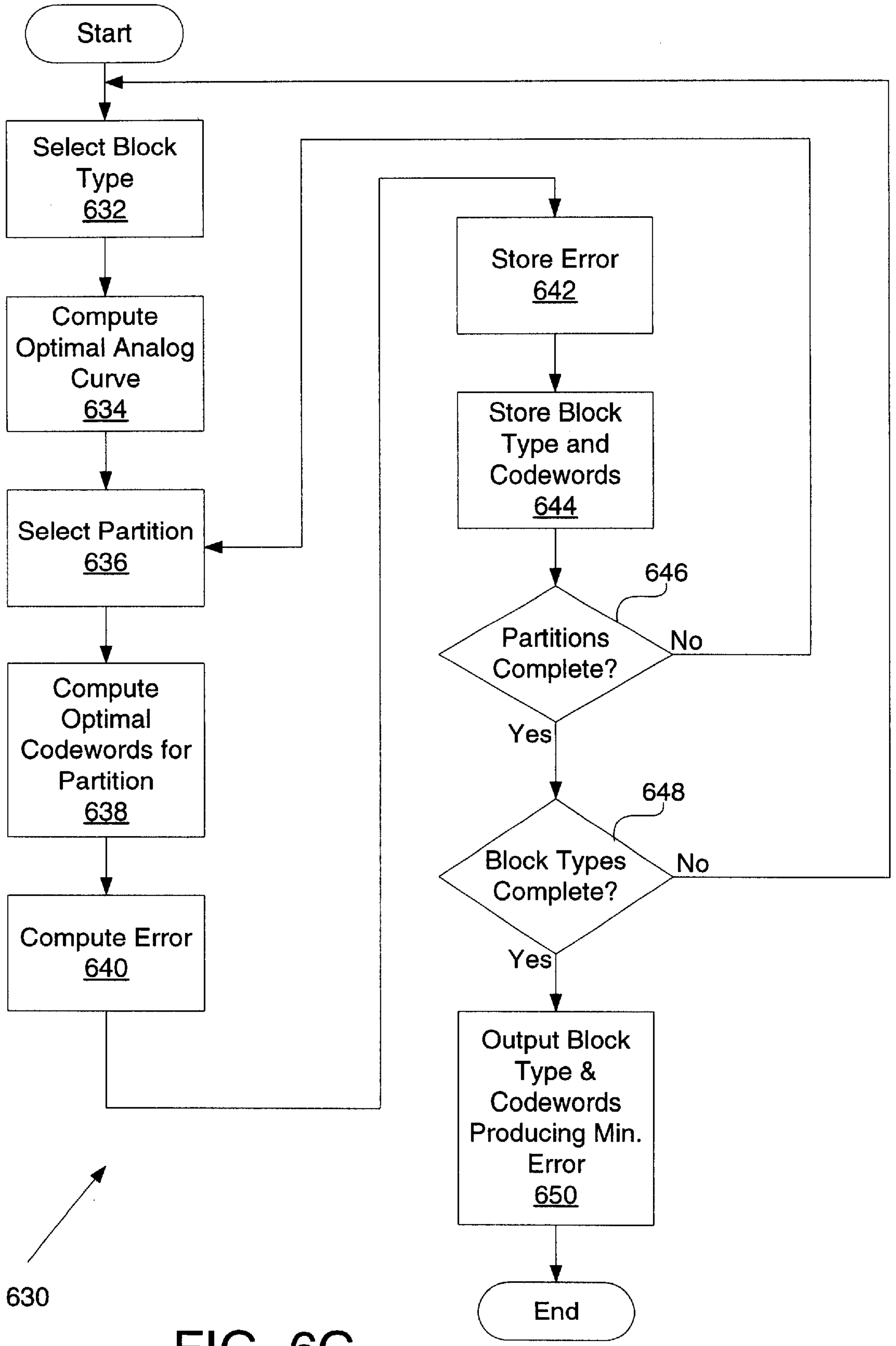

Start
Select Block Type 632
Compute Optimal Analog Curve 634
Select Partition 636
Compute Optimal Codewords for Partition 638
Compute Error 640
Store Error 642
Store Block Type and Codewords 644
646
Partitions Complete?
No
Yes
648
Block Types Complete?
No
Yes
Output Block Type & Codewords Producing Min. Error 650
End
630

FIG. 6C

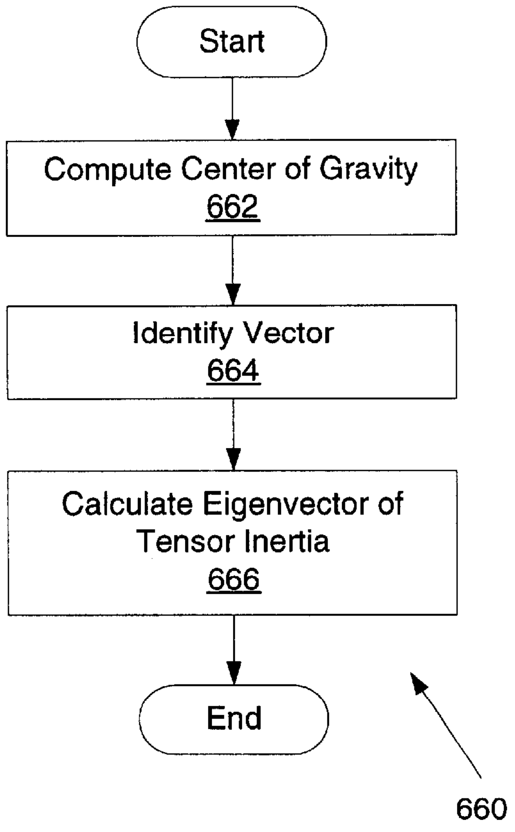

Start
Compute Center of Gravity 662
Identify Vector 664
Calculate Eigenvector of Tensor Inertia 666
End
660

FIG. 6D

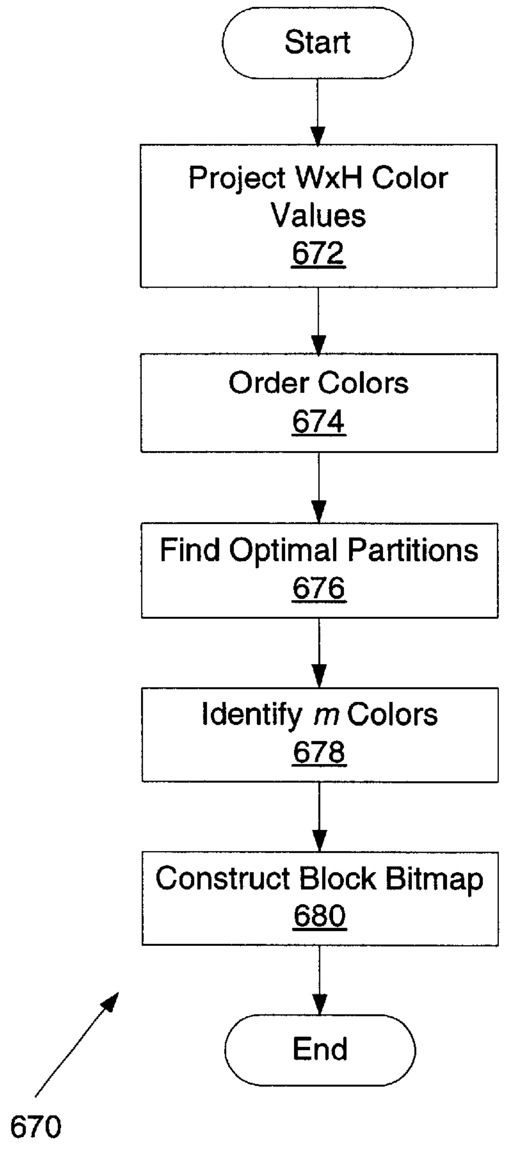

Start
Project WxH Color Values 672
Order Colors 674
Find Optimal Partitions 676
Identify *m* Colors 678
Construct Block Bitmap 680
End
670

FIG. 6E

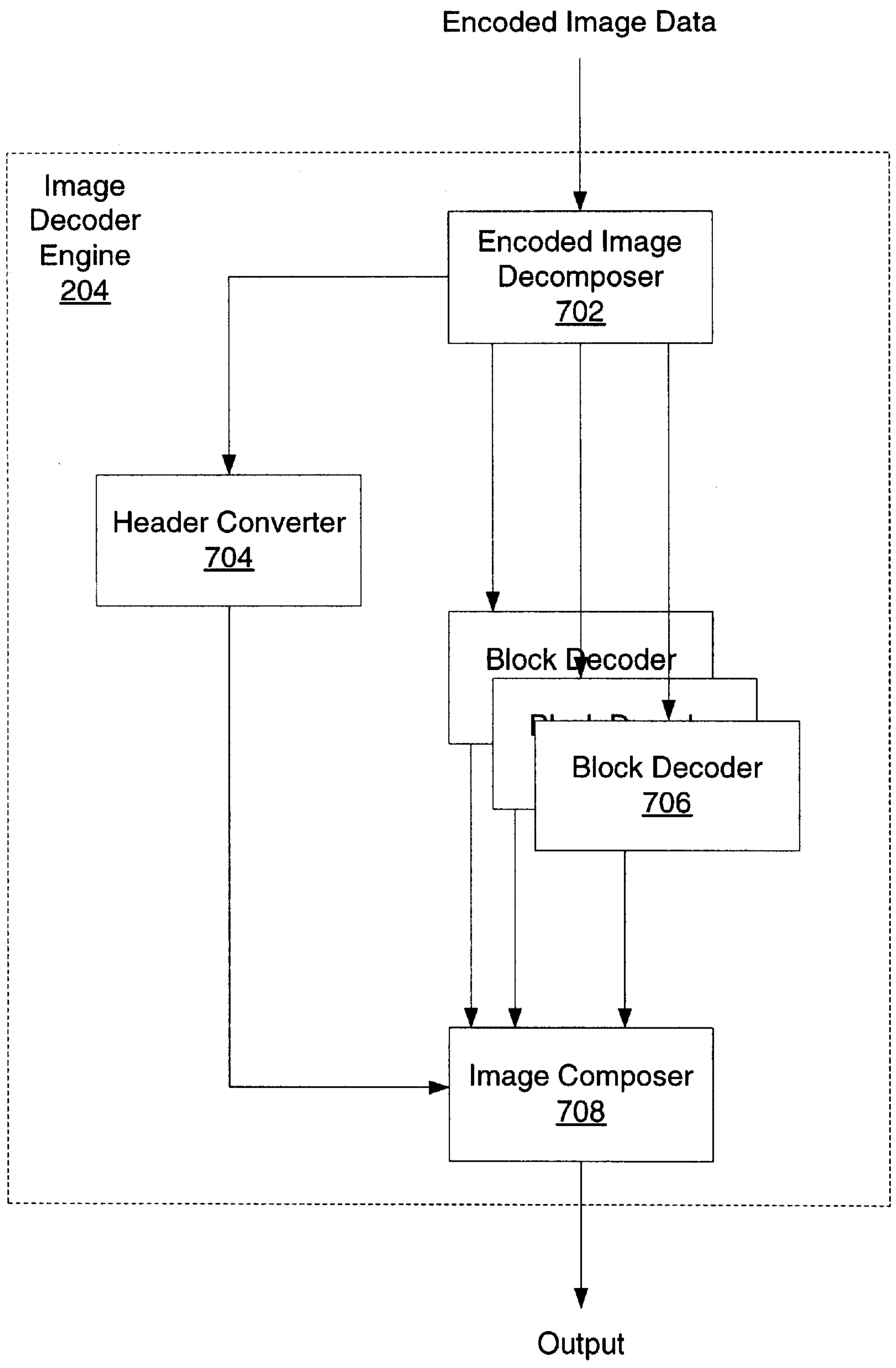

Encoded Image Data
Image Decoder Engine 204
Encoded Image Decomposer 702
Header Converter 704
Block Decoder
Block Decoder 706
Image Composer 708
Output

FIG. 7A

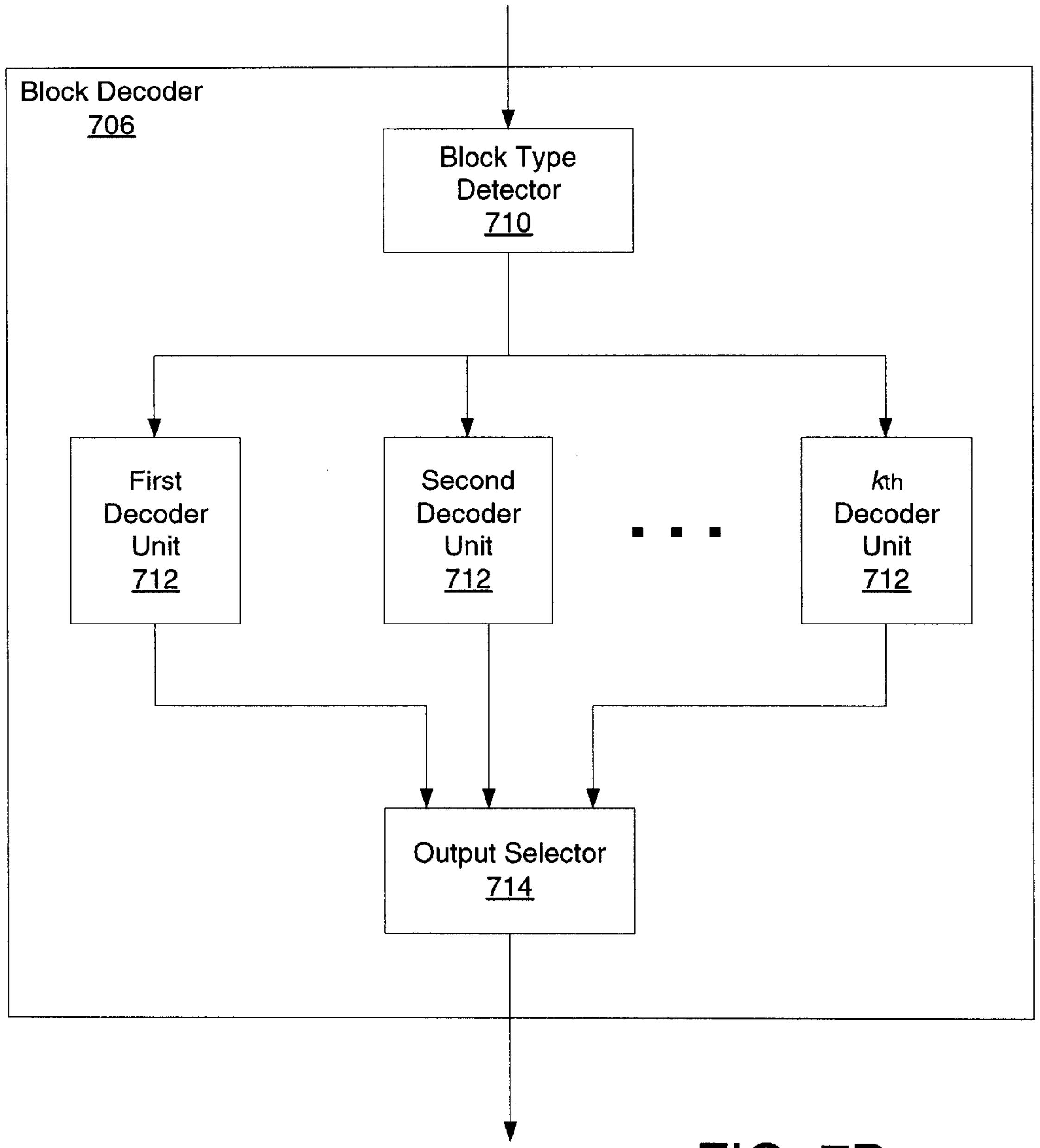

Block Decoder
706
Block Type Detector
710
First Decoder Unit
712
Second Decoder Unit
712
kth Decoder Unit
712
Output Selector
714

FIG. 7B

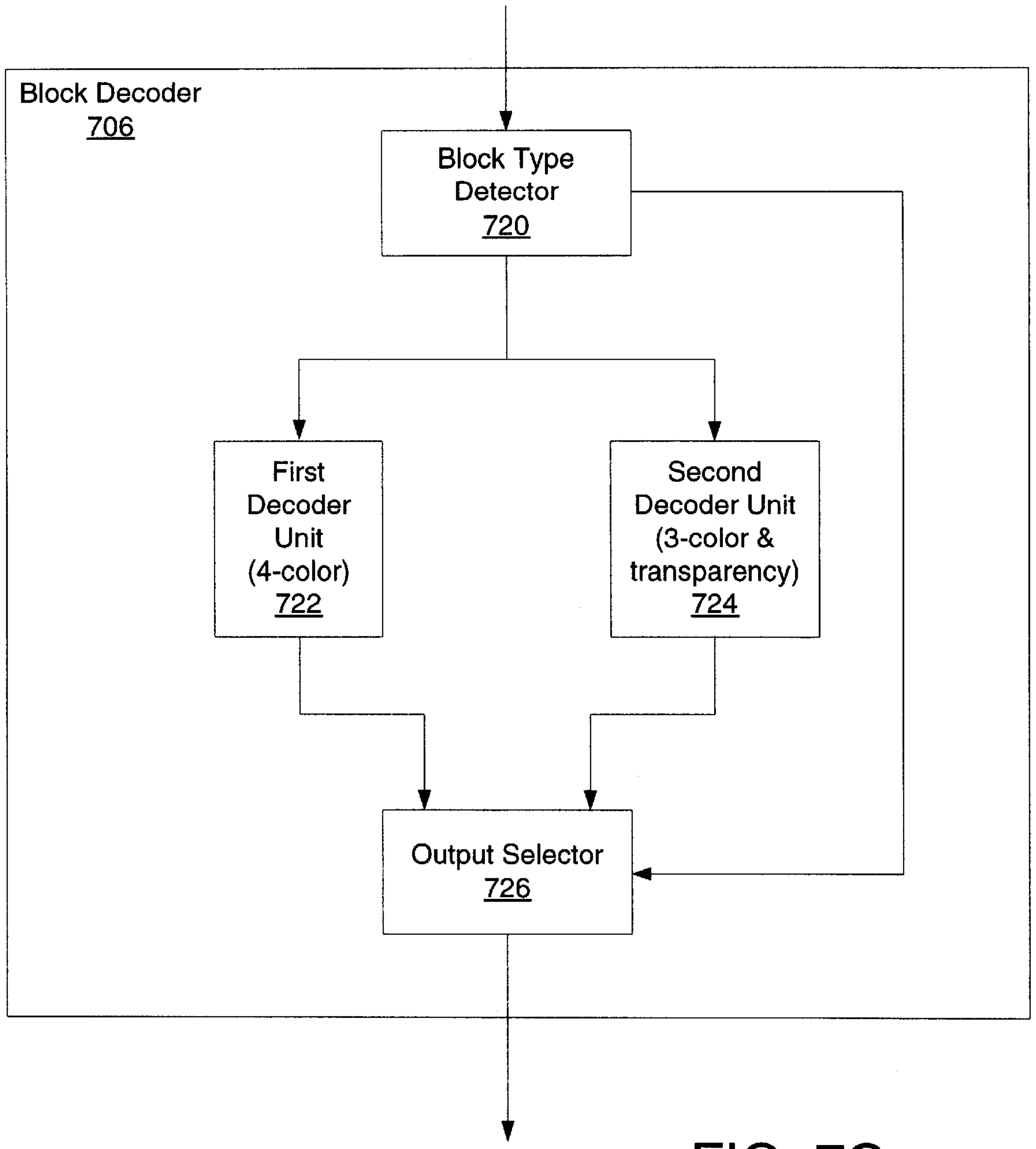

Block Decoder 706
Block Type Detector 720
First Decoder Unit (4-color) 722
Second Decoder Unit (3-color & transparency) 724
Output Selector 726

FIG. 7C

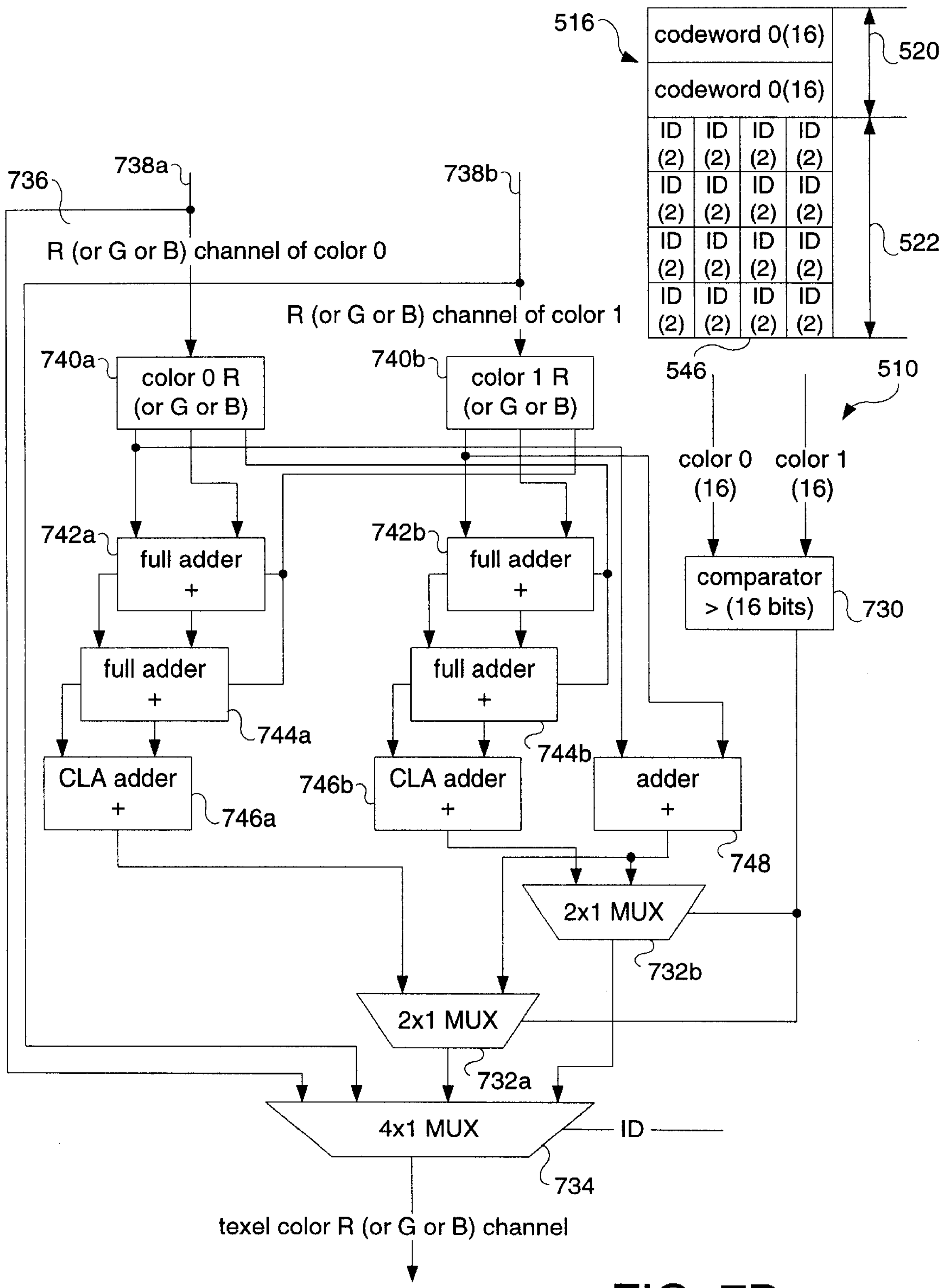

516
codeword 0(16)
codeword 0(16)
520
ID (2)
522
546
510
736
738a
738b
R (or G or B) channel of color 0
R (or G or B) channel of color 1
740a
color 0 R (or G or B)
740b
color 1 R (or G or B)
color 0 (16)
color 1 (16)
742a
full adder +
742b
full adder +
comparator > (16 bits)
730
full adder +
744a
full adder +
744b
CLA adder +
746a
746b
CLA adder +
adder +
748
2x1 MUX
732b
2x1 MUX
732a
4x1 MUX
ID
734
texel color R (or G or B) channel

FIG. 7D

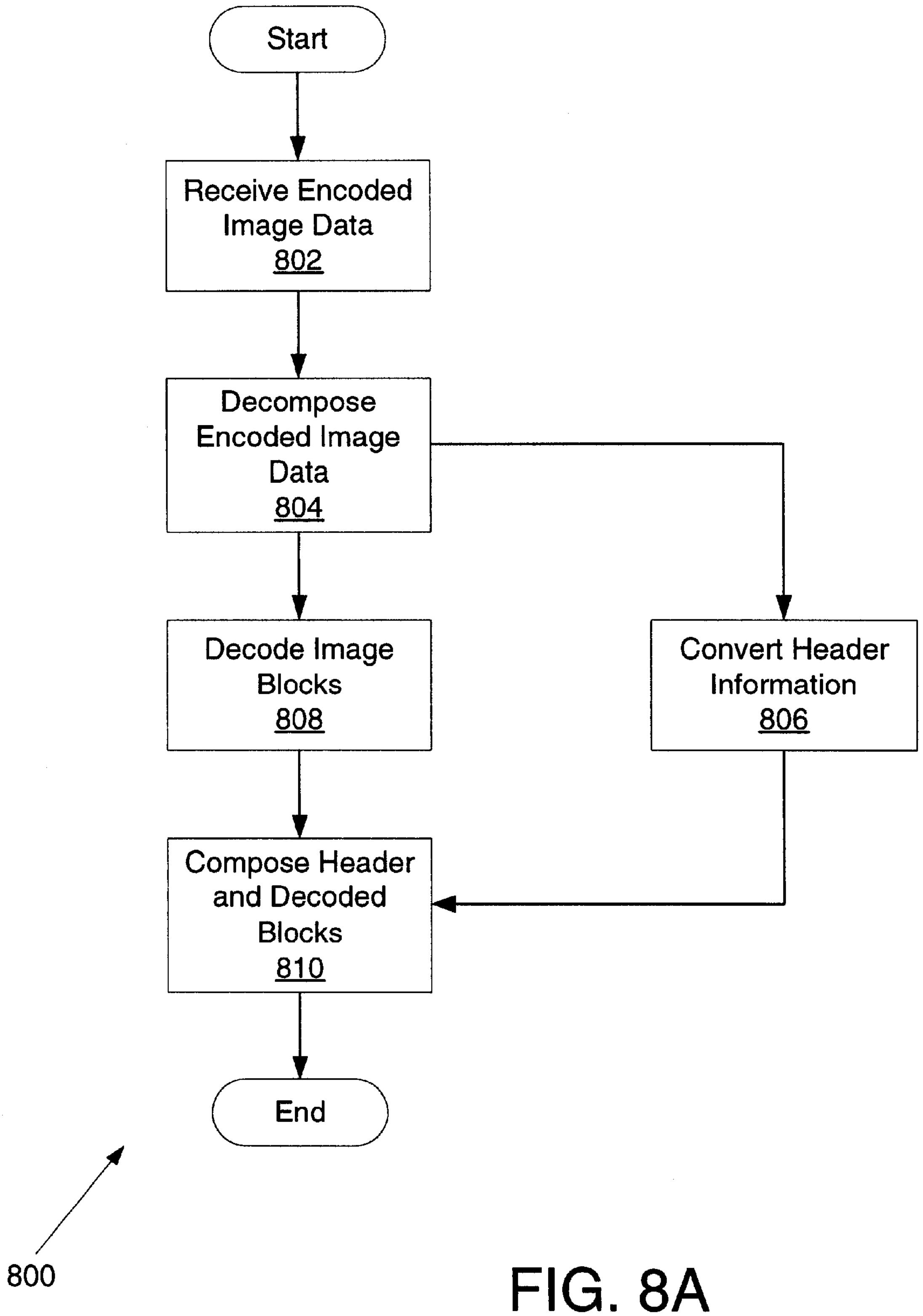

Start
Receive Encoded Image Data 802
Decompose Encoded Image Data 804
Decode Image Blocks 808
Convert Header Information 806
Compose Header and Decoded Blocks 810
End
800

FIG. 8A

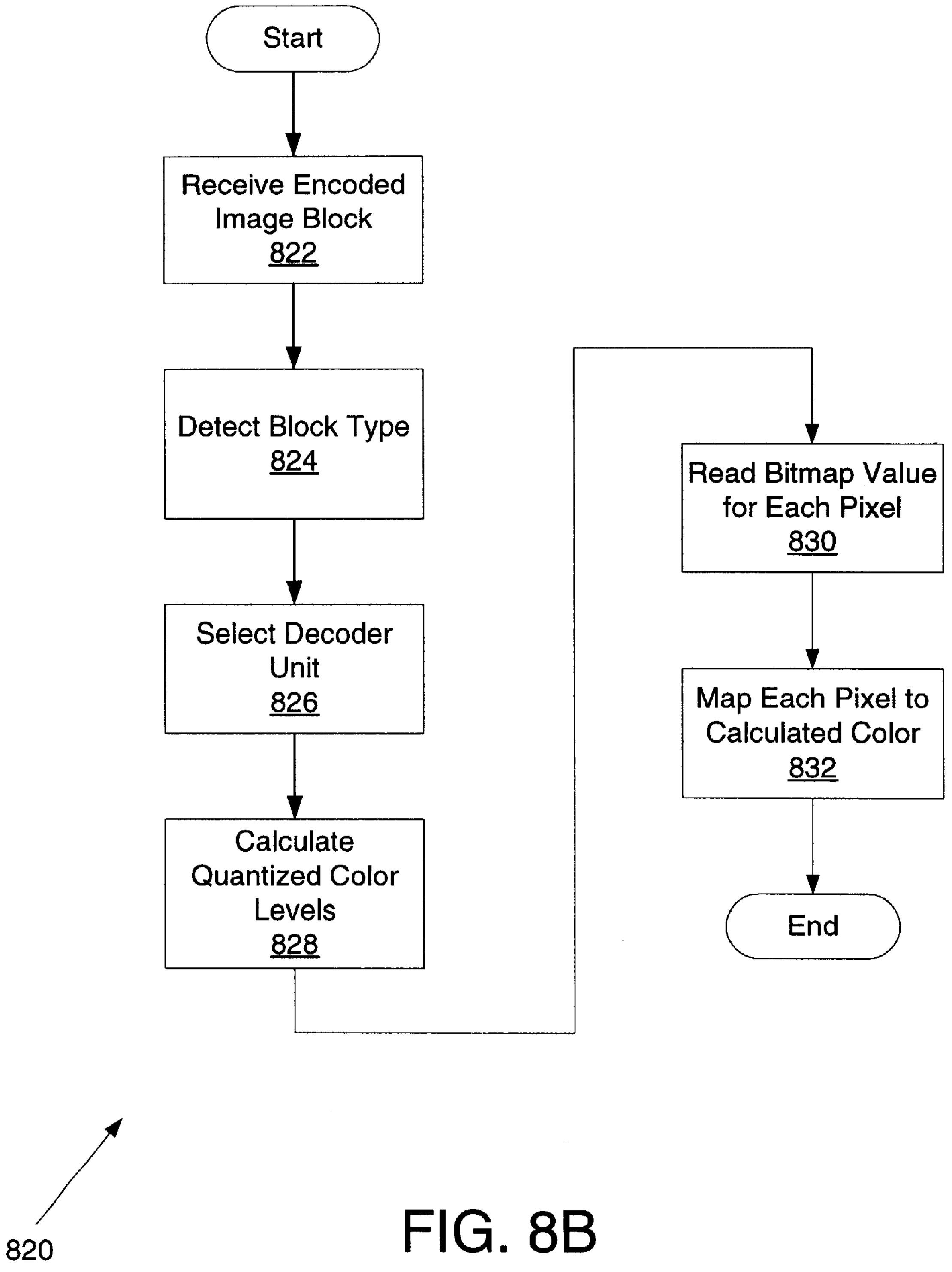

Start
Receive Encoded Image Block 822
Detect Block Type 824
Select Decoder Unit 826
Calculate Quantized Color Levels 828
Read Bitmap Value for Each Pixel 830
Map Each Pixel to Calculated Color 832
End
820

FIG. 8B

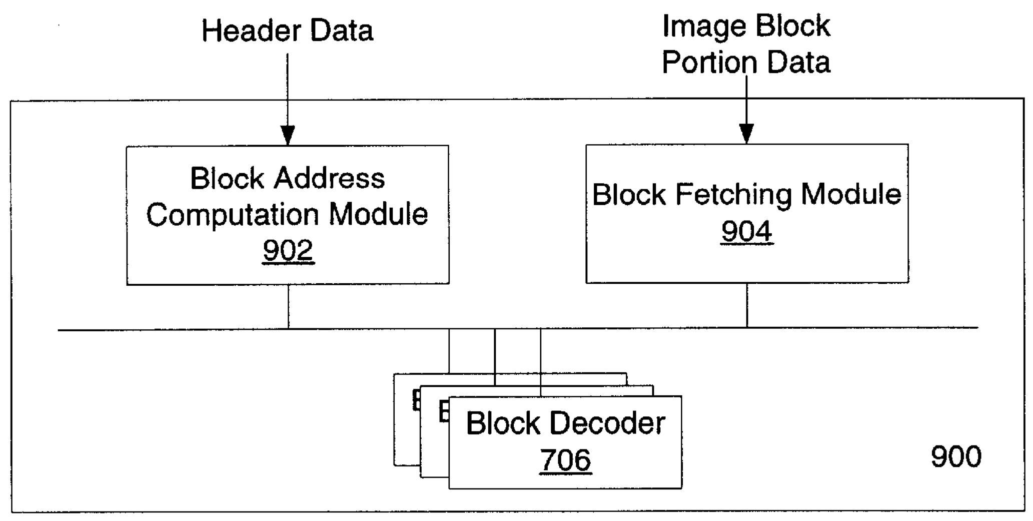

Header Data
Image Block Portion Data
Block Address Computation Module 902
Block Fetching Module 904
Block Decoder 706
900

FIG. 9A

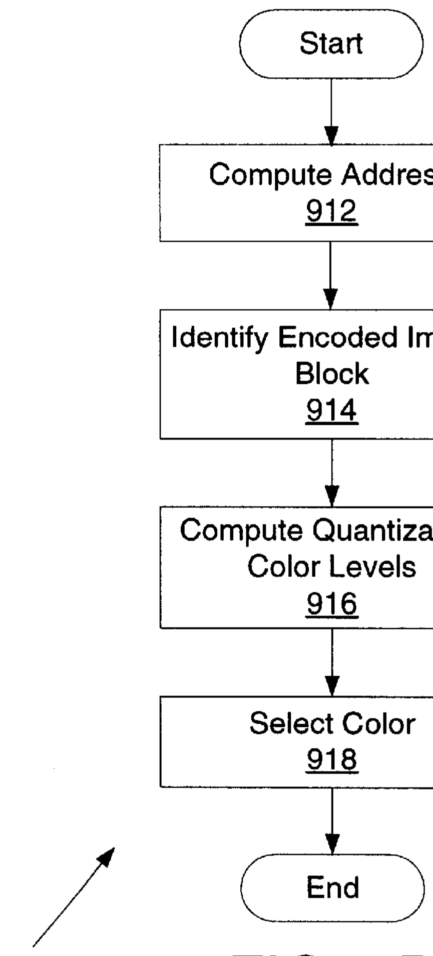

Start
Compute Address 912
Identify Encoded Image Block 914
Compute Quantization Color Levels 916
Select Color 918
End
910

FIG. 9B

FIXED-RATE BLOCK-BASED IMAGE COMPRESSION WITH INFERRED PIXEL VALUES

CROSS-REFERENCES TO RELATED APPLICATIONS

This application is a continuation-in-part application of Ser. No. 09/351,930 filed Jul. 12, 1999 now U.S. Pat. No. 6,658,146, which is a continuation of Ser. No. 08/942,860 filed Oct. 2, 1997, now U.S. Pat. No. 5,956,431 issued Sep. 21, 1999.

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates generally to image processing, and more particularly to three-dimensional rendering using fixed-rate image compression.

2. Description of Related Art

Conventionally, generating images, such as realistic and animated graphics on a computing device, required tremendous memory bandwidth and processing power on a graphics system. Requirements for memory and processing power are particularly true when dealing with three-dimensional images. In order to reduce bandwidth and processing power requirements, various compression methods and systems have been developed including Entropy or lossless encoders, Discrete Cosine Transform ("DCT") or JPEG type compressors, block truncation coding, and color cell compression. However, these methods and systems have numerous disadvantages.

Entropy or lossless encoders include Lempel-Ziv encoders which rely on predictability. For data compression using entropy encoders, a few bits are used to encode most commonly occurring symbols. In stationary systems where probabilities are fixed, entropy coding provides a lower bound for compression than can be achieved with a given alphabet of symbols. However, coding does not allow random access to any given symbol. Part of the compressed data preceding a symbol of interest must be first fetched and decompressed to decode the symbol, requiring considerable processing time and resources, as well as decreasing memory throughput. Another problem with existing Entropy methods and systems is that no guaranteed compression factor is provided. Thus, this type of encoding scheme is impractical where memory size is fixed.

Discrete Cosine Transform or JPEG-type compressors allow users to select a level of image quality. With DCT, uncorrelated coefficients are produced so that each coefficient can be treated independently without loss of compression efficiency. The DCT coefficients can be quantized using visually-weighted quantization values which selectively discard least important information.

DCT, however, suffers from a number of shortcomings. One problem with DCT and JPEG-type compressors is a requirement of large blocks of pixels, typically, 8×8 or 16×16 pixels, as a minimally accessible unit in order to obtain a reasonable compression factor and quality. Access to a very small area, or even a single pixel involves fetching a large quantity of compressed data; thus requiring increased processor power and memory bandwidth. A second problem is that the compression factor is variable, therefore requiring a complicated memory management system that, in turn, requires greater processor resources. A third problem with DCT and JPEG-type compression is that using a large compression factor significantly degrades image quality. For

example, an image may be considerably distorted with a form of ringing around edges in the image as well as noticeable color shifts in areas of the image. Neither artifact can be removed with subsequent low-pass filtering.

A further disadvantage with DCT and JPEG-type compression is the complexity and significant hardware cost for a compressor and decompressor ("CODEC"). Furthermore, high latency of a decompressor results in a large additional hardware cost for buffering throughout the system to compensate for the latency. Finally, DCT and JPEG-type compressors may not be able to compress a color keyed image.

Block truncation coding ("BTC") and color cell compression ("CCC") use a local one-bit quantizer on 4×4 pixel blocks. Compressed data for such a block consists of only two colors and 16-bits that indicate which of the two colors is assigned to each of 16 pixels. Decoding a BTC/CCC image consists of using a multiplexer with a look-up table so that once a 16-texel (or texture element, which is the smallest addressable unit of a texture map) block (32-bits) is retrieved from memory, the individual pixels are decoded by looking up the two possible colors for that block and selecting the color according to an associated bit from 16 decision bits.

Because the BTC/CCC methods quantize each block to just two color levels, significant image degradation may occur. Further, a two-bit variation of CCC stores the two colors as 8-bit indices into a 256-entry color lookup table. Thus, such pixel blocks cannot be decoded without fetching additional information which may consume additional memory bandwidth.

The BTC/CCC methods and systems can use a 3-bit per pixel scheme which stores the two colors as 16-bit values (not indices into a table) resulting in pixel blocks of six bytes. Fetching such units, however, decreases system performance because of additional overhead due to memory misalignment. Another problem associated with BTC/CCC methods is a high degradation of image quality when used to compress images that use color keying to indicate transparent pixels.

Therefore, there is a need for a system and method that maximizes accuracy of compressed images while minimizing storage, memory bandwidth requirements, and decoding hardware complexities. There is a further need for compressing image data blocks into convenient sizes to maintain alignment for random access to any one or more pixels.

SUMMARY OF THE INVENTION

The present invention provides for fixed-rate block based image compression with inferred pixel values. An image processing system includes an image encoder engine and an image decoder engine. The image encoder engine includes an image decomposer, at least one block encoder, and an encoded image composer. The block decomposer decomposes an original image into a header and a plurality of blocks which are composed of a plurality of image elements or pixels. The block encoder subsequently processes each block. The block encoder includes a selection module, a codeword generation module, and a construction module. Specifically, the selection module computes a set of parameters from image data values of each set of image elements. The codeword generation module then generates codewords which are reference image data values such as colors or density values. Subsequently, the construction module uses the codewords to derive a set of quantized image data values. The construction module then maps each of the image element's original image data values with an index to one of the derived image data values. Finally, the codewords and indices are output as encoded image blocks.

Conversely, the image decoder engine includes an encoded image decomposer, at least one block decoder, and an image composer. The image decomposer takes the encoded image and decomposes the encoded image into a header and plurality of encoded image blocks. The block decoder uses the codewords in the encoded image blocks to generate a set of derived image data values. Subsequently, the block decoder maps the index values for each image element to one of the derived image data values. The image composer then reorders the decompressed image blocks in an output data file, which is forwarded to a display device.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a block diagram of a data processing system, according to an embodiment of the present invention;

FIG. **2** is a block diagram of an image processing system;

FIG. **3**A is a block diagram of one embodiment of an image encoder system;

FIG. **3**B is a block diagram of an alternative embodiment of an image encoder system;

FIG. **3**C is a graphical representation of an image block;

FIG. **3**D is a graphical representation of a three-dimensional image block;

FIG. **4** is a block diagram of an image block encoder of FIG. **2**A, **3**A, or **3**B;

FIG. **5**A is a data sequence diagram of an original image;

FIG. **5**B is a data sequence diagram of encoded image data of an original image output from the image encoder system;

FIG. **5**C is a data sequence diagram of an encoded image block from the image block encoder of FIG. **4**;

FIGS. **6**A–**6**E are flowcharts illustrating encoding processes, according to the present invention;

FIG. **7**A is a block diagram of an image decoder system;

FIG. **7**B is a block diagram of one embodiment of a block decoder of FIG. **7**A;

FIG. **7**C is a block diagram of an alternative embodiment of a block decoder of FIG. **7**A;

FIG. **7**D is a logic diagram illustrating an exemplary decoder unit, according to the present invention;

FIG. **8**A is a flowchart illustrating a decoding process of the image decoder of FIG. **2**;

FIG. **8**B is a flowchart illustrating operations of the block encoder of FIG. **7**A;

FIG. **9**A is a block diagram of a subsystem for random access to a pixel or an image block; and

FIG. **9**B is a flowchart illustrating random access to a pixel or an image block.

DESCRIPTION OF THE PREFERRED EMBODIMENT

FIG. **1** is a block diagram of a data processing system **100** for implementing the present invention. The data processing system **100** includes a CPU **102**, a memory **104**, a storage device **106**, input devices **108**, output devices **110**, and a graphics engine **112** all of which are coupled to a system bus **114**. The memory **104** and storage device **106** store data within the data processing system **100**. The input device **108** inputs data into the data processing system **100**, while the output device **110** receives data from the data processing system **100**. Although the data bus **114** is shown as a single line, alternatively, the data bus **114** may be a combination of a processor bus, a PCI bus, a graphic bus, or an ISA bus.

FIG. **2** is a block diagram of an exemplary image processing system **200**. In one embodiment, the image processing system **200** is contained within the graphics engine **112** (FIG. **1**). The image processing system **200** includes an image encoder engine **202** and an image decoder engine **204**. The image processing system **200** may also include, or be coupled to, an image source unit **206** which provides images to the image encoder engine **202**. Further, the image processing system **200** may include or be coupled to an output unit **208** to which processed images are forwarded for storage or further processing. Additionally, the image processing system **200** may be coupled to the memory **104** (FIG. **1**) and the storage device **106** (FIG. **1**). In an alternative embodiment, the image encoder engine **202** and the image decoder engine **204** are contained within different computing devices, and the encoded images pass between the two engines **202** and **204**.

Within the image encoder engine **202**, images are broken down into individual blocks and processed before being forwarded, for example, to the storage device **106** as compressed or encoded image data. When the encoded image data are ready for further processing, the encoded image data are forwarded to the image decoder engine **204**. The image decoder engine **204** receives the encoded image data and decodes the data to generate an output that is a representation of the original image that was received from the image source unit **206**.

FIGS. **3**A and **3**B are block diagrams illustrating two exemplary embodiments of the image encoder engine **202** of FIG. **2**. The image encoder engine **202** includes an image decomposer **302**, a header converter **304**, one or more block encoders **306** in FIG. **3**A (**306***a*–**306***n*, where n is the nth encoder in FIG. **3**B), and an encoded image composer **308**. The image decomposer **302** is coupled to receive an original image **310** from a source, such as the image source unit **206** (FIG. **2**), and forwards information from a header of the original image **310** to the header converter **304**. Subsequently, the header converter **304** modifies the original header to generate a modified header, as will be described further in connection with FIG. **5**B. The image decomposer **302** also breaks, or decomposes, the original image **310** into R numbers of image blocks, where R is any integer value. The number of image blocks the original image **310** is broken into may depend on the number of image pixels. In an exemplary embodiment, the image **310** having A image pixels by B image pixels will, typically, be (A/4)×(B/4) blocks. For example, an image that is 256 pixels by 256 pixels will be broken down into 64×64 blocks. In the present embodiment, the image is decomposed such that each image block is 4 pixels by 4 pixels (16 pixels). Those skilled in the art will recognize that the number of pixels or the image block size may be varied.

Briefly turning to FIG. **3**C, an example of a single image block **320** is illustrated. The image block **320** is composed of image elements (pixels) **322**. The image block **320** may be defined as an image region W pixels in width by H pixels in height. In the embodiment of FIG. **3**C, the image block **320** is W=4 pixels by H=4 pixels (4×4).

In an alternative embodiment, the original image **310** (FIG. **3**A or **3**B) may be a three-dimensional volume data set as shown in FIG. **3**D. FIG. **3**D illustrates an exemplary three-dimensional image block **330** made up of sixteen image elements (volume pixels or voxels) **332**. Image block **330** is defined as an image region W voxels in width, H voxels in height, and D voxels in depth.

The three-dimensional volume data set may be divided into image blocks of any size or shape. For example, the image may be divided along a z-axis into a plurality of x×y×z sized images, where z=1. Each of these x×y×1 images may be treated similarly with two-dimensional images, where each x×y×1 image is divided into two-dimensional image blocks, as described above with respect to FIG. **3**C. However, decomposing the three-dimensional image into two-dimensional "slices" for compression does not fully utilize the graphical similarities that may exist in the z (depth) direction in a three-dimensional image. To utilize such similarities, the volume data may be decomposed into a plurality of three-dimensional image blocks. It will be understood that in alternative embodiments, other combinations of W×H×D are possible, and may be more desirable, depending on the data being compressed.

This type of three-dimensional image data is used, for example, in medical imaging applications such as ultrasound or magnetic resonance imaging ("MRI"). In such an application, a body part is scanned to produce a three-dimensional matrix of image elements (i.e., image block comprised of voxels **320**). The image is x voxels wide by y voxels high by z voxels deep. In this example, each voxel provides density data regarding characteristics of body tissue. In ultrasound applications, each voxel may be provided with a brightness level indicating the strength of echoes received during scanning.

In the embodiment of FIG. **3**D, the original image **310** is a three-dimensional data volume where the image data are density values. In alternative embodiments, other scalar data types may be represented in the original image **310**, such as transparency or elevation data. In further embodiments, vector data, such as the data used for "bump maps", may be represented.

Referring back to FIGS. **3**A and **3**B, each block encoder **306** receives an image block **320** from the image decomposer **302**, and encodes or compresses each image block **320**. Subsequently, each encoded image block is forwarded to the encoded image composer **308** which orders the encoded image blocks in a data file. Next, the data file from the encoded image composer **308** is concatenated with the modified header from the header converter **304** to generate an encoded image data file that is forwarded to an output **312**. Thus, the modified header and the encoded image blocks together form the encoded image data that represent the original image **310**. Alternatively, having more than one block encoder **306***a*–**306***n*, as shown in FIG. **3**B, allows for encoding multiple image blocks simultaneously, one image block per block encoder **306***a*–**306***n*, within the image encoder engine **202**. Advantageously, simultaneous encoding increases image processing efficiency and performance.

The image data associated with the original image **310** may be in any one of a variety of formats including red-green-blue ("RGB"), YUV **420** (YUV are color models representing luminosity and color difference signals), YUV **422**, or a propriety color space. In some cases, conversion to a different color space before encoding the original image **310** may be useful. In one embodiment, each image block **320** is a 4×4 set of pixels where each pixel **322** is 24-bits in size. For each pixel **322**, there are 8-bits for a Red("R")-channel, 8-bits for a Green("G")-channel, and 8-bits for a Blue("B")-channel in an RGB implementation color space. Alternatively, each encoded image block is also a 4×4 set of pixels with each pixel being only 2-bits in size and having an aggregate size of 4-bits as will be described further below.

FIG. **4** is a block diagram illustrating an exemplary block encoder **306** of FIGS. **3**A and **3**B. The block encoder **306** includes a quantizer **402** and a bitmap construction module **404**. Further, the quantizer **402** includes a block type module **406**, a curve selection module **408**, and a codeword generation module **410**.

Each image block **320** (FIG. **3**C) of the decomposed original image **310** (FIGS. **3**A and **3**B) is received and initially processed by the quantizer **402** before being forwarded to the bitmap construction module **404**. The bitmap construction module **404** outputs encoded image blocks for the encoded image composer **308** (FIGS. **3**A and **3**B) to order. The bitmap construction module **404** and the modules of the quantizer **402** are described in more detail below.

Briefly, FIG. **5**A is a diagram of a data sequencer or string **500** representing the original image **310** (FIGS. **3**A and **3**B) that is received by the block decomposer **302** (FIGS. **3**A and **3**B). The data string **500** includes an α-bit header **502** and a β-bit image data **504**. The header **502** may include information such as pixel width, pixel height, format of the original image **310** (e.g., number of bits to the pixel in RGB or YUV format), as well as other information. The image data **504** are data representing the original image **310**, itself.

FIG. **5**B is a diagram of a data sequence or string **510** representing encoded image data that are generated by the image encoder engine **202** (FIG. **2**). The encoded image data string **510** includes a modified header portion **512** and an encoded image block portion **514**. The modified header portion **512** is generated by the header converter **304** (FIGS. **3**A and **3**B) from the original α-bit header **502** (FIG. **5**A) and includes information about file type, number of bits per pixel of the original image **310** (FIGS. **3**A and **3**B), addressing in the original image **310**, other miscellaneous encoding parameters, as well as the width and height information indicating size of the original image **310**. The encoded image block portion **514** includes encoded image blocks **516***a*–*q* from the block encoders **306** (FIGS. **3**A and **3**B) where q is the number of blocks resulting from the decomposed original image **310**.

FIG. **5**C is a diagram of a data sequence or string **518** representing an encoded image block. The data string **518** may be similar to any one of the encoded image blocks **516***a*–*q* (FIG. **5**B) shown in the encoded image data string **510** of FIG. **5**B.

The encoded image block data string **518** includes a codeword section **520** and a bitmap section **522**. The codeword section **520** includes j codewords, where j is an integer value, that are used to compute colors of other image data indexed by the bitmap section **522**. A codeword is an n-bit data string that identifies a pixel property, such as color component, density, transparency, or other image data values. In one embodiment, there are two 16-bit codewords CW_0 and CW_1 (j=2). The bitmap section **522** is a Q-bit data portion and is described in more detail in connection with FIG. **6**B.

In an alternative embodiment, each encoded image block is 64-bits, which includes two 16-bit codewords and a 32-bit (4×4×2 bit) bitmap **522**. Encoding the image block **320** (FIG. **3**C) as described above provides greater system flexibility and increased data processing efficiency. In a further exemplary embodiment, each 32-bit bitmap section **522** may be a three-dimensional 32-bit bitmap.

FIGS. **6**A–**6**E describe operations of the image encoder engine **202** (FIG. **2**). In flowchart **600**, a general operation of the image encoder engine **202** is shown. In block **602**, a data string **500** (FIG. **5**A) of the original image **310** (FIGS. **3**A and **3**B), which includes the α-bit header **502** (FIG. **5**A) and the β-bit image data **504** (FIG. **5**A), is input into the image decomposer **302** (FIGS. **3**A and **3**B). The image decomposer **302** decomposes the image **310** into the α-bit header and a plurality of blocks in block **604**. The α-bit header **502** is then forwarded to the header converter **304** (FIGS. **3**A and **3**B). Subsequently, the header converter **304** generates a modified header **512** (FIG. **5**B) from the α-bit header **502** in block **606**. The modified header **512** is then forwarded to the encoded image composer **308** (FIGS. **3**A and **3**B).

Simultaneous with the header conversion process, each image block **320** is encoded in block **608** by one or more of the block encoders **306***a*–**306***n* (FIGS. **3**A and **3**B) to generate the encoded image blocks **516** (FIG. **5**B). Each image block **320** may be processed sequentially in one block encoder **306**, or multiple image blocks **320** may be processed in parallel in multiple block encoders **306***a*–**306***n*.

The encoded image blocks **516** are output from the block encoders **306**, and are placed into a predefined order by the encoded image composer **308**. In one embodiment, the encoded image blocks **516** are arranged in a file from left to right and top to bottom in the same order in which the encoded image blocks **516** were broken down by the image decomposer **302** (FIGS. **3**A and **3**B). The image encoder engine **202** subsequently composes the modified header information **512** from the header converter **304** and the encoded image blocks **516***a*–**516***q* in block **610**. Specifically, the modified header **512** and the ordered encoded image blocks **516** are concatenated to generate the encoded image data file **510** (FIG. **5**B), which may be written as encoded output **312** (FIGS. **3**A and **3**B) to the memory **104**, storage device **106**, or any output device **110** (FIG. **1**) in block **612**.

FIG. **6**B is a flowchart **620** showing the encoding process of block **608** (FIG. **6**A) in more detail. In block **622**, codewords **520** (FIG. **5**C) are computed by the codeword generation module **410** (FIG. **4**). The process for computing these codewords **520** is described in more detail in connection with FIG. **6**C.

Once the codewords **520** have been computed, pixel values or properties, such as colors, for the image block **320** (FIG. **3**C) are computed or quantized in block **624**. Specifically, the codewords **520** provide points in a pixel space from which m quantized pixel values may be inferred. The m quantized pixel values are a limited subset of pixels in a pixel space that are used to represent the current image block. The process for quantizing pixel values, and more specifically colors, will be described infra in connection with FIGS. **8**A and **8**B. Further, the embodiments will now be described with respect to colors of a pixel value although one skilled in the art will recognize that, in general, any pixel value may be used with respect to the present invention. Therefore, the image data which is quantized may be any form of scalar or vector data, such as density values, transparency values, and "bump map" vectors.

In an exemplary embodiment, each pixel is encoded with two bits of data which can index one or m quantized colors, where m=4 in this embodiment. Further, four quantized colors are derived from the two codewords **520** where two colors are the codewords **520**, themselves, and the other two colors are inferred from the codewords **520**, as will be described below. It is also possible to use the codewords **520** so that there is one index to indicate a transparent color and three indices to indicate colors, of which one color is inferred.

In another embodiment, the bitmap **522** (FIG. **5**C) is a 32-bit data string. The bitmap **522** and codewords **520** are output in block **624** as a 64-bit data string representing an encoded image block **518**. Specifically, the encoded image block **514** (FIG. **5**B) includes two 16-bit codewords **520** (n=16) and a 32-bit bitmap **522**. Every codeword **520** that is a 16-bit data string includes a 5-bit red-channel, 6-bit green-channel, and 5-bit blue-channel.

Each of the encoded image blocks **516** is placed together and concatenated with modified header information **512** derived from the original α-bit header **502** of the original image **310** (FIGS. **3**A and **3**B). A resulting output is the encoded image data **510** representing the original image **310**.

FIG. **6**C is a flowchart **630** illustrating a process for computing codewords for the image blocks **320** (FIG. **3**C), and relates to color quantizing using quantizer **402** (FIG. **4**). The process for computing codewords can be applied to all scalar and vector image data types. In select block type **632**, the quantizer **402** uses the block type module **406** (FIG. **4**) to select a first block type for the image block **320** that is being processed. For example, a selected block type may be a four-color or a three-color plus transparency block type, where the colors within the particular block type have equidistant spacing in a color space. Those of ordinary skill in the art will readily recognize that selecting a block type for each image is not intended to be limiting in any way. Instead, the present invention processes image blocks that are of a single block type, which eliminates the need to distinguish between different block types, such as the three- and four-color block types discussed above. Consequently, the block type module **406** and select block type **632** are optional.

Once the block type is selected, the quantizer **402** computes an optimal analog curve for the block type in block **634**. Computation of the optimal analog curve will be further described in connection with FIG. **6**D. The analog curve is used to simplify quantizing of the colors in the image block. Subsequently in block **636**, the quantizer **402** selects a partition of points along the analog curve, which is used to simplify quantizing of the colors in the image block. A partition may be defined as a grouping of indices {1 . . . (W×H)} into m nonintersecting sets. In one embodiment, the indices (1 . . . 16) are divided into three or four groups or clusters (i.e., m=3 or 4) depending on the block type.

Once a partition is selected, optimal codewords for the particular partition are computed in block **638**. In addition to computing the codewords, an error value (square error as described infra) for the codeword is also computed in block **640**. Both computations will be described in more detail in connection with FIG. **6**E. If the computed error value is the first error value, the error value is stored in block **642**. Alternatively, the computed error value is stored if it is less than the previously stored error value. For each stored error value, corresponding block type and codewords are also stored in block **644**. The process of flowchart **630** seeks to find the block type and codewords that minimize the error function.

Next in block **646**, the code generation module **410** (FIG. **4**) determines if all possible partitions are completed. If there are more partitions, the code generation module **410** selects the next partition, computes the codewords and associated error values, and stores the error values, associated block types, and codewords if the error value is less than the previously stored error value.

After all the possible partitions are completed, the codeword generation module **410** determines, in block **648**, whether all block types have been selected. If there are more block types, the codeword generation module **410** selects the next block type and computes the codeword and various values as previously described. After the last block type has been processed, the codeword generation module **410** outputs a result of the block type and codewords **520** (FIG. **5**C) having the minimum error in block **650**.

In an alternative embodiment, the optimal analog curve may be computed before selecting the block type. That is, the optimal analog curve is computed before the selection of the block type and partition, computation of the codewords and error values, and storage of the error value, block type, and codeword. Computing the optimal analog curve first is useful if all block types use the same analog curve and color space because the analog curve does not need to be recomputed for each block type.

FIG. **6**D is a flowchart **660** describing a process of identifying the optimal analog curve. The curve selection module **408** (FIG. **4**) first computes a center of gravity for pixel colors of an image block **320** (FIG. **3**C) in block **662**. The center of gravity computation includes averaging the pixel colors. Once the center of gravity is computed, a vector in color space is identified in block **664** to minimize the first moment of the pixel colors of the image block **320**. Specifically for identifying a vector, a straight line is fit to a set of data points, which are the original pixel colors of the image block **320**. The straight line is chosen passing through the center of gravity of the set of data points such that it minimizes a "moment of inertia" (i.e., square error). For example, to compute a direction of a line minimizing the moment of inertia for three pixel properties, tensor inertia, T, is calculated from individual colors as follows:

$$T = \sum_{i=1}^{W\times H} \begin{bmatrix} C_{1i}^2 + C_{2i}^2 & -C_{0i}C_{1i} & -C_{0i}C_{2i} \\ -C_{0i}C_{1i} & C_{0i}^2 + C_{2i}^2 & -C_{1i}C_{2i} \\ -C_{0i}C_{2i} & -C_{2i}C_{1i} & C_{0i}^2 + C_{1i}^2 \end{bmatrix}$$

where C_0, C_1, and C_2 represent pixel properties (e.g., color components in RGB or YUV) relative to a center of gravity. In one embodiment of an RGB color space, C_{0i} is a value of red, C_{1i} is a value of green, and C_{2i}, is a value of blue for each pixel, i, of the image block. Further, i takes on integer values from 1 to W×H, so that if W=4 and H=4, i ranges from 1 to 16.

An eigenvector of tensor inertia, T, with the smallest eigenvalue is calculated in block **666** using conventional methods. An eigenvector direction along with the calculated gravity center, defines an axis that minimizes the moment of inertia. This axis is used as the optimal analog curve, which in one embodiment, is a straight line. Those of ordinary skill in the art will readily recognize that the optimal analog curve is not limited to a straight line, but may include a set of parameters, such as pixel values or colors, that minimizes the moment of inertia or mean-square-error when fit to the center of gravity of the pixel colors in the image block. The set of parameters may define any geometric element, such as a curve, plate, trapezoid, or the like.

FIG. **6**E is a flowchart **670** describing the process undertaken by the codeword generation module **410** (FIG. **4**) for selecting the partitions, computing the codewords and associated error for the partitions, and storing the error value, block type, and codeword if the error value is less than a previously stored error value. In block **672**, the codeword generation module **410** projects the W×H color values onto the previously constructed optimal analog curve. The value of W×H is the size in number of pixels of an image block **320** (FIG. **3**C). In one embodiment where W and H are both four pixels, W×H is 16 pixels.

Subsequently in block **674**, the colors are ordered sequentially along the analog curve based on a position of the color on a one-dimensional analog curve. After the colors are ordered, the codeword generation module **410** searches, in block **676**, for optimal partitions. Thus, the codeword generation module **410** takes the W×H colors (one color associated with each pixel) that are ordered along the analog curve and partitions and groups the colors into a finite number of clusters with a predefined relative spacing. In one embodiment where W=4 and H=4 (i.e., W×H is 16), the 16 colors are placed in three and four clusters (i.e., m=3 or 4).

In conducting the search for the optimal partition, a color selection module within the codeword generation module **410** finds the best m clusters from the W×H points projected onto the optimal curve, so that the error associated with the selection is minimized. The best m clusters are determined by minimizing the mean-square-error with the constraint that the points associated with each cluster are spaced to conform to the predefined spacing.

In one embodiment for a block type of four equidistant colors, the error may be defined as a square error along the analog curve, such as

$$E^2 = \sum_{cluster\,0} (x_i - p_0)^2 + \sum_{cluster\,1} \left[x_i - \left(\frac{2}{3}p_0 + \frac{1}{3}p_1\right)\right]^2 + \sum_{cluster\,2} \left[x_i - \left(\frac{1}{3}p_0 + \frac{2}{3}p_1\right)\right]^2 + \sum_{cluster\,3} (x_i - p_1)^2$$

where E is the error for the particular grouping or clustering, p_0 and p_1 are the coded colors, and x_i are the projected points on the optimal analog curve.

In instances where the block type indicates three equidistant colors, the error may be defined as a squared error along the analog curve, such as

$$E^2 = \sum_{cluster\,0} (x_i - p_0)^2 + \sum_{cluster\,1} \left[x_i - \left(\frac{1}{2}p_0 + \frac{1}{2}p_1\right)\right]^2 + \sum_{cluster\,2} (x_i - p_1)^2$$

After the resulting optimal codewords **520** are identified, the codewords **520** are forwarded to the bitmap construction module **404** (FIG. **4**). The bitmap construction module **404** uses the codewords **520** to identify the m colors that may be specified or inferred from those codewords **520** in block **678**. In one embodiment, the bitmap construction module **404** uses the codewords **520** (e.g., CW_0 and CW_1) to identify the three or four colors that may be specified or inferred from those codewords **520**.

Next in block **680**, the bitmap construction module **404** constructs a block bitmap **522** (FIG. **5**C) using the codewords **520** associated with the image block **320** (FIG. **3**C). Colors in the image block **320** are mapped to the closest color associated with one of the quantized colors specified by, or inferred from, the codewords **520**. The result is a color index, referenced as ID, per pixel in the block identifying the associated quantized color.

Information indicating the block type is implied by the codewords **520** and the bitmap **522**. In one embodiment, the order of the codewords **520** indicate the block type. If a numerical value of CW_0 is greater than a numerical value of CW_1, the image block is a four-color block. Otherwise, the block is a three-color plus transparency block.

In one embodiment discussed above, there are two-color image block types. One color image block type has four equidistant colors, while the other color image block type has three equidistant colors with the fourth color index used to specify that a pixel is transparent. For both color image block types, the color index is two bits. In an embodiment with density values in place of color values, each density image block type has four equidistant density values.

The output of the bitmap construction module **404** is an encoded image block **514** (FIG. **5**B) having the m codewords **520** plus the bitmap **522**. Each encoded image block **516** is received by the encoded image composer **308** (FIGS. **3**A and **3**B) that, in turn, orders the encoded image blocks **516** in a file. In one embodiment, the encoded image blocks **516** are arranged from left to right and from top to bottom in the same order as the blocks were broken down by the image decomposer **302**. The ordered file having the encoded image blocks **516** is concatenated with the modified header information **512** that is derived from the α-bit header **502** of the original image **310** (FIGS. **3**A and **3**B) to generate the encoded image data **510** that is the output of the image encoder engine **202** (FIG. **2**). The output may then be forwarded to the memory **104**, the storage device **106**, or the output device **110** (FIG. **1**).

The exemplary embodiment of the image encoder engine **202** advantageously reduces the effective data size of an image from 24-bits per pixel to 4-bits per pixel. Further, the exemplary embodiment beneficially addresses transparency issues by allowing codewords to be used with a transparency identifier.

FIG. **7**A is a block diagram of an exemplary image decoder engine **204** (FIG. **2**). The image decoder engine **204** includes an encoded image decomposer **702**, a header converter **704**, one or more block decoders **706** (**706***a*–**706***p*, where p represents the last block decoder), and an image composer **708**. The encoded image decomposer **702** is coupled to received the encoded image data **514** (FIG. **5**B) output from the image encoder engine **202** (FIG. **2**). The encoded image decomposer **702** receives the encoded image data string **510** and decomposes, or breaks, the encoded image data string **510** into the header **512** (FIG. **5**B) and the encoded image blocks **514** (FIG. **5**B). Next, the encoded image decomposer **702** reads the modified header **512**, and forwards the modified header **512** to the header converter **704**. The encoded image decomposer **702** also decomposes the encoded image data string **510** into the individual encoded image blocks **516** (FIG. **5**B) that are forwarded to the one or more block decoders **706**.

The header converter **704** converts the modified header **512** into an output header. Simultaneously, the encoded image blocks **516** are decompressed or decoded by the one or more block decoders **706**. Each encoded image block **516** may be processed sequentially in one block decoder **706**, or multiple encoded image blocks **514** may be processed in parallel with one block decoder **706** for each encoded image block **516**. Thus, multiple block decoders **706** allow for parallel processing that increases the processing performance and efficiency of the image decoder engine **204** (FIG. **2**).

The image composer **708** receives each decoded image blocks from the one or more block decoders **706** and orders the decoded image block in a file. Further, the image composer **708** receives the converted header from the header converter **704**. The converted header and the decoded image blocks are placed together to generate output data representing the original image **310**.

FIG. **7**B is a block diagram of an exemplary embodiment of a block decoder **706**. Each block decoder **706** includes a block type detector **710**, one or more decoder units **712**, and an output selector **714**. The block type detector **710** is coupled to the encoded image decomposer **702** (FIG. **7**A), the output selector **714**, and each of the one or more decoder units **712**.

The block type detector **710** receives the encoded image blocks **514** and determines the block type for each encoded image block **516** (FIG. **5**B). The block type is detected based on the codewords **520** (FIG. **5**C). After the block type is determined, the encoded image blocks **514** are passed to each of the decoder units **712**, which decompress or decode each encoded image block **516** to generate colors for each particular encoded image block **516**. The decoder units **712** may be c-channels wide (e.g., one channel for each color component or pixel property being encoded), where c is any integer value. Using the selector signal, the block type detector **710** enables the output selector **714** to output the color of each encoded image block **516** from one of the decoder units **712** that corresponds with the block type detected by the block type detector **710**. Specifically, the block type detector **710** passes a selector signal to the output selector **714** that is used to select an output corresponding to the block type detected. Alternatively, using the selector signal, the appropriate decoder unit **712** could be selected so that the encoded block is only processed through the selected decoder unit.

FIG. **7**C is a block diagram of an alternative embodiment of a block decoder **706**. In this embodiment, the block decoder **706** includes a block type detector **720**, a first decoder unit **722**, a second decoder unit **724**, and an output selector **726**. The block type detector **720** is coupled to receive each encoded image block **516** (FIG. **5**B), and determine by comparing the codewords **520** (FIG. **5**C) of the encoded image block, the block type for each encoded image block **516**. For example, the block type may be four quantized colors or three quanitized colors and a transparency. Once the block type is selected and a selector signal is forwarded to the output selector **726**, the encoded image blocks **516** are decoded by the first and second decoder units **722** and **724**, respectively, to produce the pixel colors of each image block. The output selector **726** is enabled by the block type detector **720** to output the colors from the first and second decoder units **722** and **724** that correspond to the block type selected.

FIG. **7**D is a logic diagram illustrating an exemplary embodiment of a decoder unit similar to the decoder units **722** and **724** of FIG. **7**C. For simplicity, the functionality of each of the first and second decoder units **722** and **724** is merged into the single logic diagram of FIG. **7**D. Those skilled in the art will recognize that although the diagram is described with respect to a red-channel of the decoder units, the remaining channels (i.e., the green-channel and the blue-channel) are similarly coupled and functionally equivalent.

The logic diagram illustrating the first and second decoder units **722** and **724** is shown including portions of the block type detector **710**, **720** (FIGS. **7**B and **7**C, respectively) such as a comparator unit **730**. The comparator unit **730** is coupled to and works with a first 2×1 multiplexer **732***a* and a second 2×1 multiplexer **732***b*. Both 2×1 multiplexers **732***a* and **732***b* are coupled to a 4×1 multiplexer **734** that serves to select an appropriate color to output. The 4×1 multiplexer **734** is coupled to receive a transparency indicator signal that indicates whether or not a transparency (e.g., no color) is being sent. The 4×1 multiplexer **734** selects a color for output based on the value of the color index, referenced as the ID signal, that references the associated quantized color for an individual pixel of the encoded image block **514** (FIG. **5**B).

A red-channel **736** of the first decoder unit **722** includes a first and a second red-channel line **738***a* and **738***b* and a first and a second red-color block **740***a* and **740***b*. Along the path of each red-color block **740***a* and **740***b* is a first full adder **742***a* and **742***b*, a second full adder **744***a* and **744***b*, and carry-look ahead ("CLA") adders **746***a* and **746***b*. The second decoder unit **724** contains similar components as the first decoder unit **722**.

The CLA adder **746***a* of the first red-color block **740***a* path of the first decoder unit **722** is coupled to the first 2×1 multiplexer **732***a*, while the CLA adder **746***b* of the second red-color block **740***b* path of the first decoder unit **722** is coupled to the second 2×1 multiplexer **732***b*. Further, adder **748** of the second decoder unit **724** is coupled to both the first and the second 2×1 multiplexers **732***a* and **732***b*.

FIG. **8**A is a flowchart **800** illustrating an operation of the decoder engine **204** (FIG. **2**) in accordance with an exemplary embodiment of the present invention. For purposes of illustration, the process for the decoder engine **204** will be described with a single block decoder **706** (FIG. **7**A) having two decoder units **722** and **724** as described earlier in connection with FIG. **7**C. Those skilled in the art will recognize that the process is functionally equivalent for decoder systems having more than one block decoder **706** and more than two decoder units **712**, as discussed in connection with FIG. **7**B.

In block **802**, the encoded image decomposer **702** (FIG. **7**A) receives the encoded or compressed image data **510** (FIG. **5**B) from the image encoder engine **202** (FIG. **2**), through the memory **104** (FIG. **1**) or the storage device **106** (FIG. **1**). Next, the encoded image decomposer **702** decomposes the encoded image data **510** by forwarding the modified header **512** (FIG. **5**B) to the header converter **704** (FIG. **7**A) in block **804**.

Subsequently in block **806**, the header converter **704** converts the header information to generate an output header that is forwarded to the image composer **708** (FIG. **7**A). Simultaneously, the one or more block decoders **706** (FIG. **7**A) decode pixel colors for each encoded image block **516** (FIG. **5**B) in block **808**. Each encoded image block **516** may be decoded sequentially in one block decoder **706** or multiple encoded image blocks **514** (FIG. **5**B) may be decoded in parallel in multiple block decoders **706** in block **808**. The process for decoding each encoded image block **516** is further described in connection with FIG. **8**B. Each decoded image block is then composed into a data file with the converted header information by the image composer **708** in block **810**. The image composer **708** then generates the data file as an output that represents the original image **310** (FIGS. **3**A and **3**B).

FIG. **8**B is a flowchart **820** illustrating an operation of the block decoder **706** (FIG. **7**A) in accordance with an exemplary embodiment of the present invention. Initially, each encoded image block **516** (FIG. **5**B) is received by the block decoder **706** in block **822**. Specifically, for one embodiment the first and the second codewords **520** (e.g., CW_0 and CW_1 of FIG. **5**C) are received by the block type detector **710**, **720** (FIGS. **7**B and **7**C, respectively) of the block decoder **706**. As discussed above, comparing the numerical values of CW_0 and CW_1 reveals the block type. The first five bits of each codeword **520** that represent the red-channel color are received by the red-channel of each of the first and second decoder units **722** and **724** (FIG. **7**C). Furthermore, the second 6-bits of each codeword **520** that represent the green-channel color are received by the green-channel of each of the first and the second decoder units **722** and **724**, while the last 5-bits of each codeword **520** that represent the blue-channel color are received by the blue-channel of each of the first and second decoder units **722** and **724**.

Next in block **824**, the block type detector **710** detects the block type for an encoded image block **514**. Specifically, the comparator **730** (FIG. **7**D) compares the first and the second codewords **520** (e.g., CW_0 and CW_1) and generates a flag signal to enable the first 2×1 multiplexer **732***a* or the second 2×1 multiplexer **732***b*. In block **826**, either the first decoder unit **722** or the second decoder unit **724** is selected.

Subsequently quantized color levels for the decoder units **722** and **724** are calculated in block **828**. The calculation of the quantized color levels will now be discussed in more detail. Initially, the first decoder unit **722** calculates the four colors associated with the two codewords **520** (e.g., CW_0 and CW_1) using the following exemplary relationship:

CW_0=first codeword=first color;

CW_1=second codeword=second color;

$$CW_2 = \text{third color} = \frac{2}{3}CW_0 + \frac{1}{3}CW_1;\text{ and}$$

$$CW_3 = \text{fourth color} = \frac{1}{3}CW_0 + \frac{2}{3}CW_1.$$

In one embodiment, the first decoder unit **722** may estimate the above equations for CW_2 and CW_3 as follows:

$$CW_2 = \frac{5}{8}CW_0 + \frac{3}{8}CW_1;\text{ and}$$

$$CW_3 = \frac{3}{8}CW_0 + \frac{5}{8}CW_1.$$

The red-color blocks **740***a* and **740***b* (FIG. **7**D) serve as one-bit shift registers to obtain

$$\frac{1}{2}CW_0 \text{ or } \frac{1}{2}CW_1.$$

Further, each full adder **742***a*, **742***b*, **744***a*, and **744***b* (FIG. **7**D) also serves to shift the signal left by 1-bit. Thus, the signal from the first full adders **742***a* and **742***b* is

$$\frac{1}{4}CW_0 \text{ or } \frac{1}{4}CW_1,$$

respectively, because of a 2-bit overall shift, while the signal from the second full adders **744***a* and **744***b* is

$$\frac{1}{8}CW_0 \text{ or } \frac{1}{8}CW_1,$$

respectively due to a 3-bit overall shift. These values allow for the above approximations for the color signals.

The second decoder unit **724** (FIG. **7**C) calculates three colors associated with the codewords **520** (e.g., CW_0 and CW_1), and includes a fourth signal that indicates a transparency is being passed. The second decoder unit **724** calculates colors using the following exemplary relationship:

CW_0=first codeword=first color;

CW_1=second codeword=second color;

$$CW_3 = \text{third color} = \frac{1}{2}CW_0 + \frac{1}{2}CW_1;\text{ and}$$

T=Transparency.

In one embodiment, the second decoder unit **724** has no approximation because the signals received from the red-color blocks **740***a* and **740***b* are shifted left by 1-bit so that the color is already calculated to

$$\frac{1}{2}CW_0 \text{ or } \frac{1}{2}CW_1,$$

respectively.

After the quantized color levels for the decoder units **722** and **724** selected in block **826** have been calculated in block **828**, each bitmap value for each pixel is read from the encoded image data block **510** (FIG. **5**A) in block **830**. As each index is read, it is mapped in block **832** to one of the four calculated colors if the first decoder unit **722** is selected. Alternatively, one of the three colors and transparency is mapped in block **832** if the second decoder unit **724** is selected. The mapped colors are selected by the 4×1 multiplexer **734** based on the value of the ID signal from the bitmap **522** (FIG. **5**C) of the encoded image block **514**. As stated previously, a similar process occurs for selection of colors in the green-channel and the blue-channel.

As the color data are output from the red-channel, green-channel and blue-channel, the output is received by the image composer **708** (FIG. **7**A). Subsequently, the image composer **708** arranges the output from the block encoders **706** in the same order as the original image **310** was decomposed. The resulting image is the original image **310**, which is then forwarded to an output unit **208** (FIG. **2**; e.g., a computer screen) which displays the image.

This exemplary embodiment beneficially allows for random access to any desired image block **320** (FIG. **3**C) within an image, and any pixel **322** (FIG. **3**C) within an image block **320**. FIG. **9**A is a block diagram of a subsystem **900** that provides random access to a pixel **322** or an image block **320** in accordance with one embodiment of the present invention.

The random access subsystem **900** includes a block address computation module **902**, a block fetching module **904**, and one or more block decoders **706** coupled to the block address computation module **902** and the block fetching module **904**. The block address computation module **902** receives the header information **512** (FIG. **5**B) of the encoded image data string **510** (FIG. **5**B), while the block fetching module **904** receives the encoded image block portion **514** (FIG. **5**B) of the encoded image data string **510**.

FIG. **9**B is a flowchart **910** of a process for random access to a pixel **322** (FIG. **3**C) or an image block **320** (FIG. **3**C) using the random access subsystem **900** of FIG. **9**A. When particular pixels **322** have been identified for decoding, the image decoder engine **204** (FIG. **2**) receives the encoded image data string **510** (FIG. **5**B). The modified header **512** (FIG. **5**B) of the encoded image data string **510** is forwarded to the block address computation module **902** (FIG. **9**A), and the encoded image block portion **514** (FIG. **5**B) of the encoded image data string **510** is forwarded to the block fetching module **904** (FIG. **9**A).

In block **912**, the block address computation module **902** reads the modified header **512** to compute an address of the encoded image block portion **514** having the desired pixels **322**. The address computed is dependent upon the pixel coordinates within an image. Using the computed address, the block fetching module **904** identifies each encoded image block **516** (FIG. **5**B) of the encoded image block portion **514** that contains the desired pixels **322** in block **914**. Once each encoded image block **516** having the desired pixels **322** has been identified, only the identified encoded image block **516** is forwarded to the block decoders **706** (FIG. **9**A) for processing.

FIG. **9**B is similar to the process described above in FIG. **8**B, wherein the block decoders **706** compute quantized color levels for each identified encoded image blocks **516** having the desired pixels in block **916**. After the quantized color levels have been computed, the color of the desired pixel is selected in block **918** and output from the image decoder engine **204**.

Random access to pixels **322** of an image block **320** (FIG. **3**C) advantageously allows for selective decoding of only needed portions or sections of an image. Random access also allows the image to be decoded in any order the data is required. For example, in three-dimensional texture mapping only portions of the texture may be required and these portions will generally be required in some non-sequential order. Thus, this embodiment of the present invention increases processing efficiency and performance when processing only a portion or section of an image. Further, the present invention beneficially encodes or compresses the size of an original image **310** (FIGS. **3**A and **3**B) from 24-bits per pixel to an aggregate 4-bits per pixel, and then decodes or decompresses the encoded image data string **510** (FIG. **5**B) to get a representation of the original image **310**. Additionally, the exemplary embodiment uses two base points or codewords from which additional colors are derived so that extra bits are not necessary to identify a pixel **322** color.

Moreover, the exemplary embodiment advantageously accomplishes the data compression on an individual block basis with the same number of bits per block so that the compression rate can remain fixed. Further, because the blocks are of fixed size with a fixed number of pixels **322**, random access to any particular pixel **322** in the block is allowed. Additionally, an efficient use of system resources is provided because entire blocks of data are not retrieved and decoded to display data corresponding to only a few pixels **322**.

Finally, the use of fixed-rate 64-bit data blocks provides the advantage of having simplified header information that allows for faster processing of individual data blocks. A 64-bit data block allows for faster processing as the need to wait until a full data string is assembled is eliminated. Further, an imaging system in accordance with the present invention may also reduce the microchip space necessary for a decoder system because the decoder system only needs to decode each pixel **322** to a set of colors determined by, for example, the two codewords **520** (FIG. **5**C).

The present invention has been described above with reference to specific embodiments. It will be apparent to those skilled in the art that various modifications may be made and other embodiments can be used without departing from the broader scope of the invention. Therefore, these and other variations upon the specific embodiments are intended to be covered by the present invention.

What is claimed is:

1. An image encoder engine for encoding an image, comprising:

an image decomposer for decomposing the image into a header and at least one image block, each image block having a set of image elements and each image element having an original image data value;

at least one block encoder for receiving each image block and for compressing each image block into an encoded image block by associating each original image data value of the image element with an index to a derived image data value in a set of quantized image date values; and

an encoded image composer coupled to the block encoder for ordering the encoded image blocks into a data file.

2. The image encoder engine of claim **1** further comprising a header converter coupled to the image decomposer for converting the header into a modified header.

3. The image encoder engine of claim **2** wherein the encoded image composer orders the encoded image block and the modified header into a data file.

4. The image encoder engine of claim **1** wherein the block encoder further comprises a selection module for computing a set of parameters from the image data values of the set of image elements.

5. The image encoder engine of claim **1** wherein the block encoder further comprises a codeword generation module for generating at least one codeword.

6. The image encoder engine of claim **1** wherein the block encoder further comprises a construction module for generating the set of quantized image data values including at least one codeword and at least one derived image data value.

7. The image encoder engine of claim **1** wherein the block encoder further comprises a block type module for selecting an identifiable block type for the image block.

8. An image decoder engine for decoding an encoded image data file, comprising:

an encoded image decomposer for decomposing the encoded image data file into a modified header and at least one compressed image block, each image block having at least one associated codeword and a plurality of image elements associated with an index value;

at least one block decoder coupled to the encoded image decomposer for decompressing the at least one compressed image block into at least one decompressed image block by generating a set of quantized image data values and mapping the index value to a quantized image data value from the set of quantized image data values; and

an image composer for ordering the at least one decompressed image blocks in an output data file.

9. The image decoder engine of claim **8** wherein the set of quanitized image data values include the at least one codeword and at least one image data value derived from the at least one codeword.

10. The image decoder engine of claim **8** further comprising a header converter coupled to the encoded image decomposer for converting the modified header into an output header.

11. The image decoder engine of claim **10** wherein the image composer orders the at least one decompressed image block and the output header into a data file.

12. The image decoder engine of claim **8** wherein the at least one block decoder further comprises a block type detector for selecting a block type for each of the at least one compressed image block.

13. The image decoder engine of claim **8** wherein the at least one block decoder further comprises a decoder for decompressing each of the at least one compressed image block based on a block type.

14. The image decoder engine of claim **8** wherein the at least one block decoder further comprises an output selector for outputting the at least one decompressed image block.

15. A method for fixed-rate block-based image compression of an original image, comprising the steps of:

decomposing the original image into a header and a plurality of image blocks each having a set of image elements with an original image data value;

computing at least one codeword from the original image data value for the set of image elements;

generating a set of quantized image data values including the at least one codeword and at least one image value derived from the at least one codeword; and

mapping the original image data value to one of the quantized image data values to produce an index value for each image element.

16. The method of claim **15** further comprising outputting an encoded image data file.

17. The method of claim **15** further comprising the step of converting the header into a modified header.

18. The method of claim **17** further comprising the step of composing the modified header and encoded image blocks into the encoded image data file.

19. A machine readable medium having embodied thereon a program being executable by a machine to perform method steps for fixed-rate block-based image compression of an original image, the method steps comprising:

decomposing the original image into a header and a plurality of image blocks each having a set of image elements with an original image data value;

computing at least one codeword from the original image data value for the set of image elements;

generating a set of quantized image data values including the at least one codeword and at least one image value derived from the at least one codeword; and

mapping the original image data value to one of the quantized image data values to produce an index value for each image element.

20. The machine readable medium of claim **19** further comprising the method of outputting an encoded image data file.

21. An image encoder system for encoding an original image, comprising:

means for decomposing the original image into a header and a plurality of image blocks each having a set of image elements with an original image data value;

means for computing at least one codeword from the original image data value for the set of image elements;

means for generating a set of quantized image data values including the at least one codeword and at least one image value derived from the at least one codeword; and

means for mapping the original image data value to one of the quantized image data values to produce an index value for each image element.

22. The image encoder system of claim **21** further comprising means for outputting an encoded image data file.

23. A method for fixed-rate block-based image decompression of an encoded image, comprising the steps of:

decomposing the encoded image of into a modified header and a plurality of encoded image blocks having at least one codeword and a plurality of image elements associated with an index value;

generating a set of quanitized image data values including the at least one codeword and at least one image value derived from the at least one codeword; and

mapping the index value for each image element to one of the quantized image data values.

24. The method of claim **23** further comprising outputting a decoded image data file.

25. The method of claim **23** further comprising the step of converting the modified header into an output header.

26. The method of claim **25** further comprising the step of composing the output header and decoded image blocks into the decoded image data file.

27. A machine readable medium having embodied thereon a program being executable by a machine to perform method steps for fixed-rate block-based image decompression of an encoded image, the method steps comprising:

decomposing the encoded image data file into a modified header and a plurality of encoded image blocks having at least one codeword and a plurality of image elements associated with an index value;

generating a set of quanitized image data values including the at least one codeword and at least one image value derived from the at least one codeword; and

mapping the index value for each image element to one of the quantized image data values.

28. The machine readable medium of claim **27** further comprising the method of outputting a decoded image data file.

29. An image decoder engine for decoding an encoded image data file, comprising means for decomposing the encoded image data file into a modified header and a plurality of encoded image blocks having at least one codeword and a plurality of image elements associated with an index value;

means for generating a set of quanitized image data values including the at least one codeword and at least one image value derived from the at least one codeword; and

means for mapping the index value for each image element to one of the quantized image data values.

30. The image decoder engine of claim **29** further comprising means for outputting a decoded image data file.

* * * * *

Exhibit D

US007043087B2

(12) **United States Patent**
Hong et al.

(10) **Patent No.: US 7,043,087 B2**
(45) **Date of Patent: May 9, 2006**

(54) IMAGE PROCESSING SYSTEM

(75) Inventors: **Zhou Hong**, Cupertino, CA (US); **Konstantine I. Iourcha**, San Jose, CA (US); **Krishna S. Nayak**, Palo Alto, CA (US)

(73) Assignee: **S3 Graphics Co., Ltd.**, Grand Cayman (KY)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **10/893,084**

(22) Filed: **Jul. 16, 2004**
(Under 37 CFR 1.47)

(65) **Prior Publication Data**

US 2004/0258321 A1 Dec. 23, 2004

Related U.S. Application Data

(63) Continuation of application No. 10/052,613, filed on Jan. 17, 2002, now Pat. No. 6,775,417, which is a continuation-in-part of application No. 09/351,930, filed on Jul. 12, 1999, now Pat. No. 6,658,146, which is a continuation of application No. 08/942,860, filed on Oct. 2, 1997, now Pat. No. 5,956,431.

(51) **Int. Cl.**
G06K 9/36 (2006.01)

(52) **U.S. Cl.** **382/233**; 382/166; 382/232; 382/253

(58) **Field of Classification Search** 382/253, 382/232, 233, 166, 162; 375/240.03, 240.16; 709/247; 370/394
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,821,208 A		4/1989	Ryan et al.	345/550
4,887,151 A		12/1989	Wataya	358/539
5,734,744 A		3/1998	Wittenstein et al.	382/166
5,742,892 A		4/1998	Chaddha	725/146
5,748,904 A		5/1998	Huang et al.	345/544
5,768,535 A	*	6/1998	Chaddha et al.	709/247
5,787,192 A		7/1998	Takaichi et al.	382/166
5,822,465 A		10/1998	Normile et al.	382/253
5,956,425 A		9/1999	Yoshida	382/234

(Continued)

FOREIGN PATENT DOCUMENTS

JP	401 284 188	11/1989

(Continued)

OTHER PUBLICATIONS

A. Schilling, et al.; "Texram: A Smart Memory for Texturing"; IEEE Computer Graphics & Applications; May 1996; 16(3) pp. 9-19.

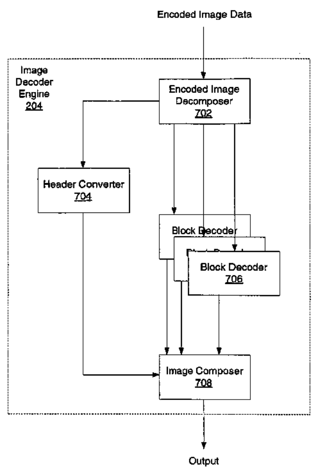
(Continued)

Primary Examiner—Anh Hong Do
(74) *Attorney, Agent, or Firm*—Carr & Ferrell LLP

(57) **ABSTRACT**

An image processing system including an image encoder and image decoding system is provided. The image encoder system includes an image decomposer, a block encoder, and an encoded image composer. The image decomposer decomposes the image into blocks. The block encoder, which includes a selection module, a codeword generation module and a construction module, processes the blocks. Specifically, the selection module computes a set of parameters from image data values of a set of image elements in the image block. The codeword generation module generates codewords, which the construction module uses to derive a set of quantized image data values. The construction module then maps each of the image element's original image data values to an index to one of the derived image data values. The image decoding system reverses this process to reorder decompressed image blocks in an output data file.

8 Claims, 16 Drawing Sheets



U.S. PATENT DOCUMENTS

5,956,431	A	9/1999	Iourcha et al.	382/253
6,075,619	A	6/2000	Iizuka	382/166
6,088,392	A *	7/2000	Rosenberg	375/240.03
6,658,146	B1	12/2003	Iourcha et al.	382/166
6,775,417	B1 *	8/2004	Hong et al.	382/253
2004/0258322	A1	12/2004	Hong et al.	382/253

FOREIGN PATENT DOCUMENTS

JP	405 216 993	8/1993

OTHER PUBLICATIONS

G. Knittel, et al.; "Hardware and Software for Superior Texture Performance": In 10; Eurographics Hardware Workshop '95; Maastricht, NL; Aug. 28-29, 1995; pp. 1-8.

G. Campbell, et al.; "Two Bit/Pixel Full Color Encoding"; Computer Graphics, (Proc. Siggraph '86); Aug. 18-22, 1986; vol. 20, No. 4, Dallas TX; pp. 215-219.

Feng et al., "A Dynamic Address Vector Quantization Algorithm . . . ", IEEE Int'l Conf. on Acoustics, Speech & Signal Proc., vol. 3, May 1989, pp. 1755-1758.

Yang et al., "Hybrid Adaptive Block Truncation Coding for Image Compression," Optical Eng., Soc. of Photo-Optical Instr. Eng., vol. 36, No. 4, Apr. 1, 1997 pp. 1021-1027.

Kugler, "High-Performance Texture Decompression Hardware," Visual Computer, Springer, Berlin, Germany, vol. 13, No. 2, 1997, pp. 51-63.

Panos Nasiopoulos et al., "Adaptive Compression Coding," IEEE Transactions on Communications, IEEE Inc., New York, USA, vol. 39, No. 8, Aug. 1, 1991, pp. 1245-1254.

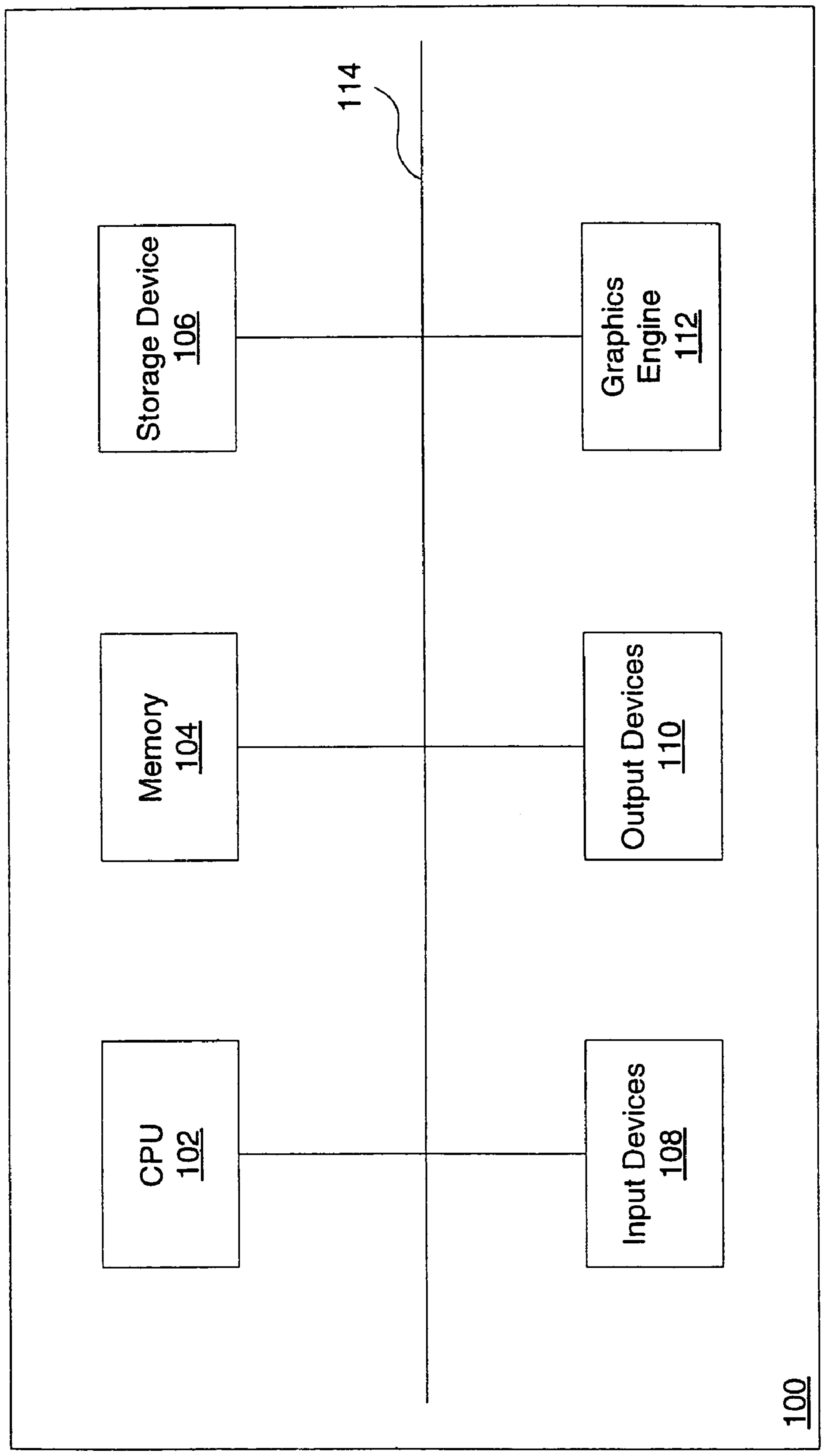
Delp E.J. et al., "Image Compression Using Block Truncation Coding," IEEE Inc., New York, USA, vol. COM-27, No. 9, Sep. 1979, pp. 1335-1342.

Yang et al., "Use of Radius Weighted Mean to Cluster Two-Class Data," Electronics Letters, IEE Stevenage, Great Britain, vol. 30, No. 10, May 12, 1994, pp. 757-759.

Russ, J.C. et al., "Optimal Grey Scale Images from Multiplane Color Images," Journal of Computer-Assisted Microscopy, Dec. 1995, Plenum, USA, vol. 7, No. 4, pp. 221-233.

Knittel et al., "Hardware for Superior Texture Performance," Eurographics Workshop on Graphics Hardware, Jul. 28, 1995, pp. 33-40.

* cited by examiner



CPU 102
Memory 104
Storage Device 106
Input Devices 108
Output Devices 110
Graphics Engine 112
114
100

FIG. 1

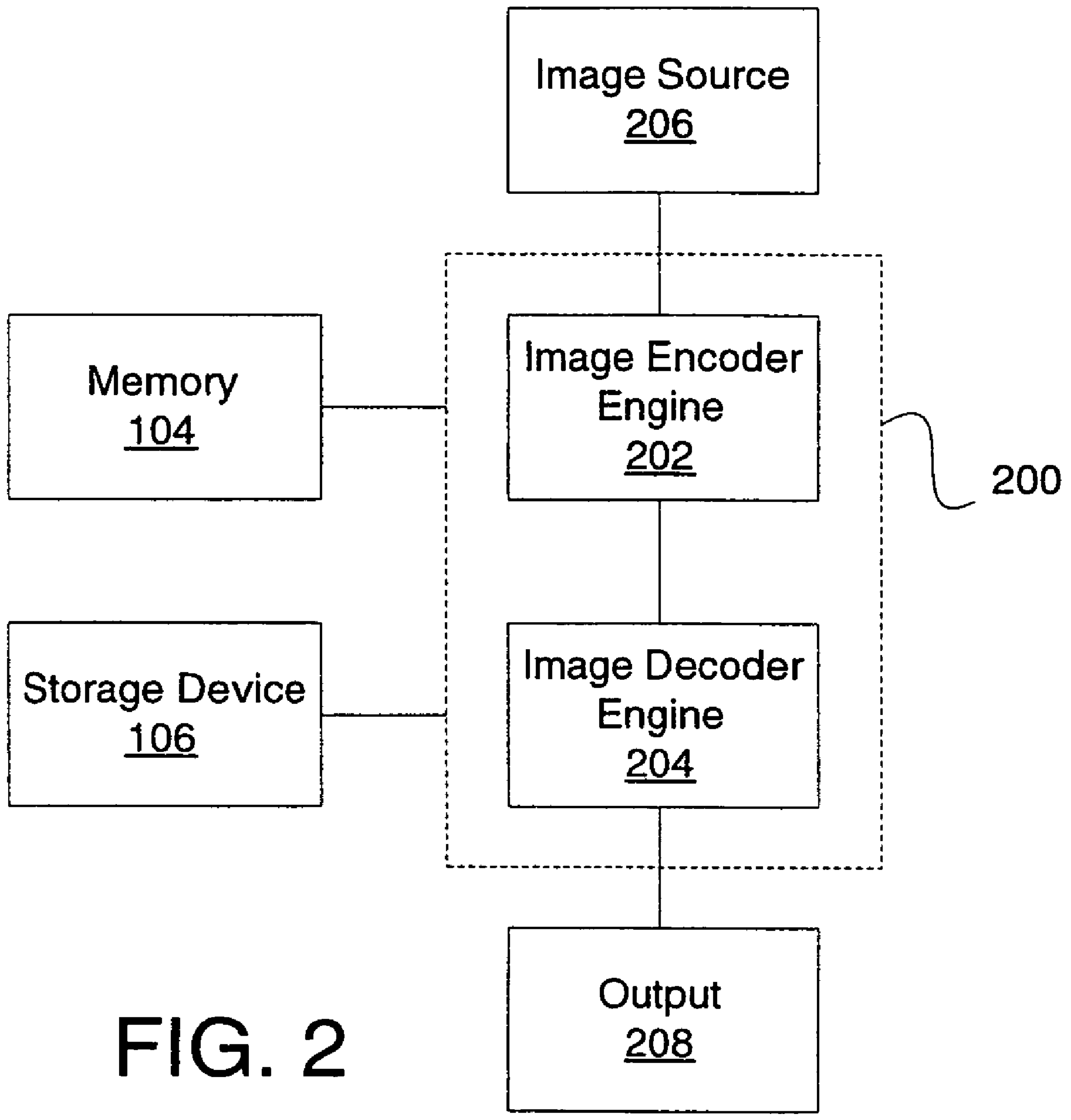


Image Source
206
Memory
104
Image Encoder Engine
202
200
Storage Device
106
Image Decoder Engine
204
Output
208

FIG. 2

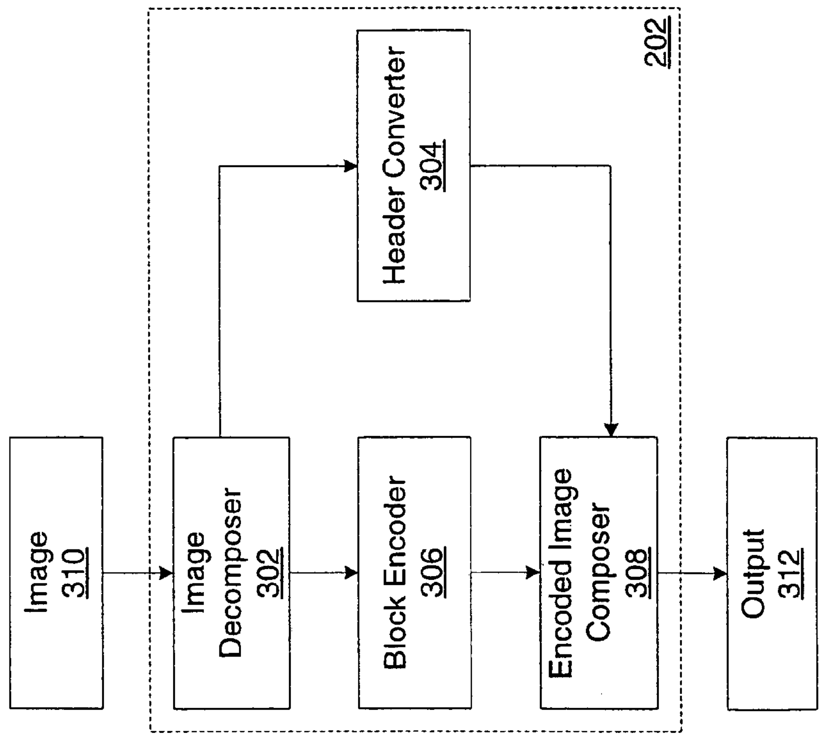


Image 310
Image Decomposer 302
Block Encoder 306
Encoded Image Composer 308
Output 312
Header Converter 304
202

FIG. 3A

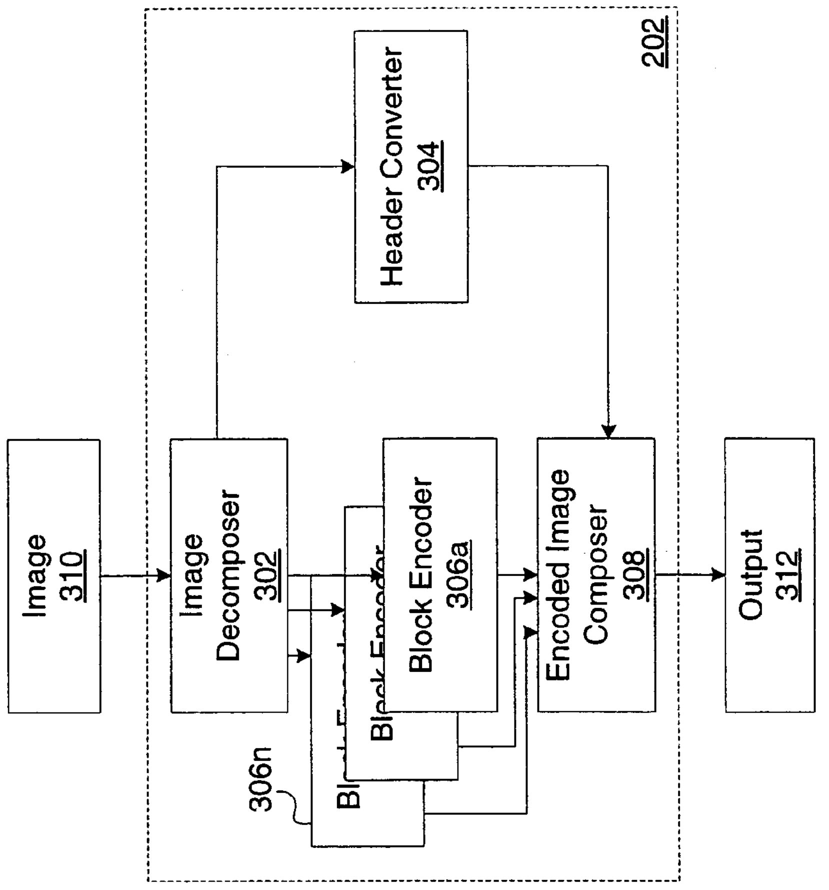


Image 310
Image Decomposer 302
Block Encoder 306a
Block Encoder
Block Encoder
306n
Encoded Image Composer 308
Output 312
Header Converter 304
202

FIG. 3B

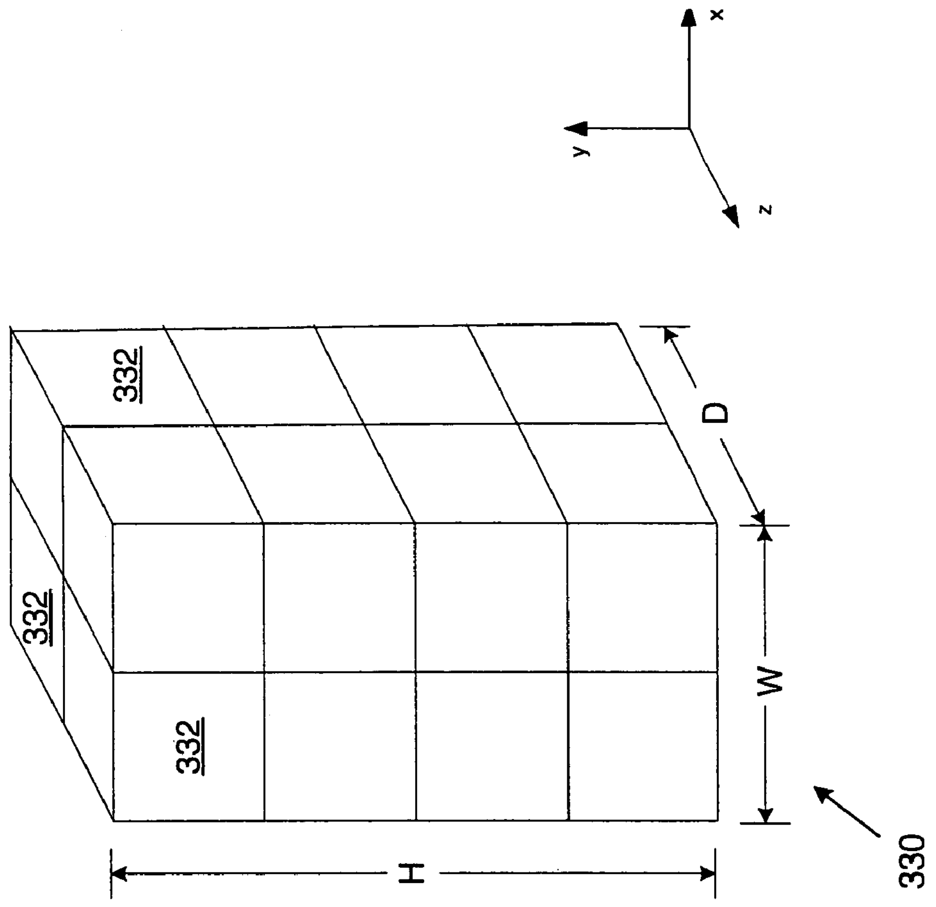


FIG. 3D

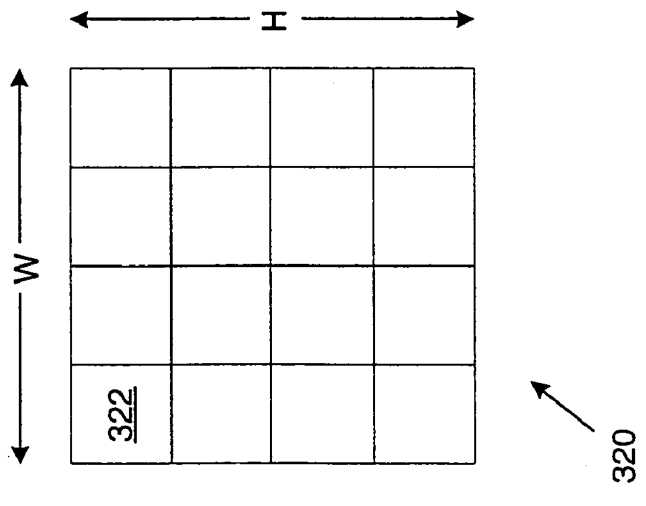


FIG. 3C

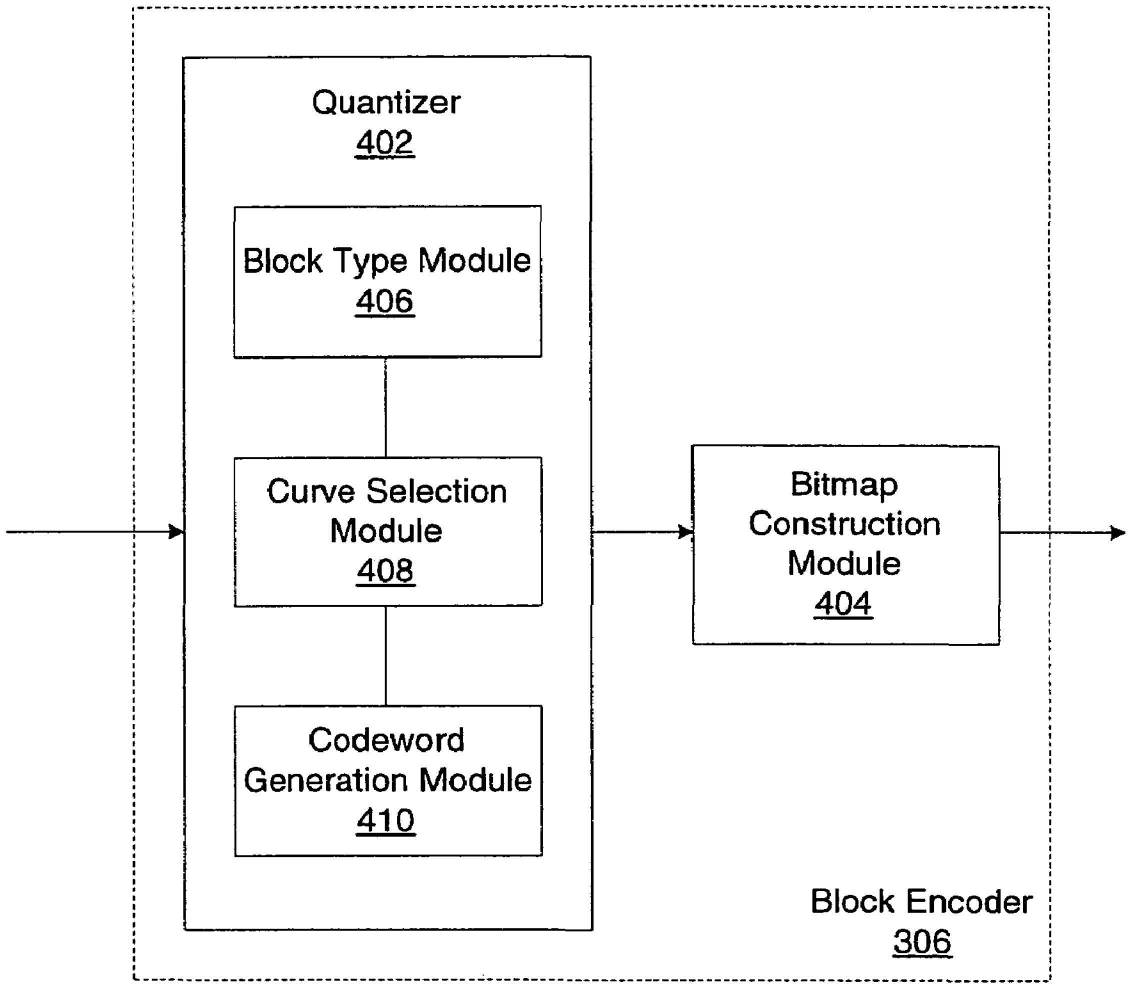

Quantizer
402
Block Type Module
406
Curve Selection Module
408
Codeword Generation Module
410
Bitmap Construction Module
404
Block Encoder
306

FIG. 4

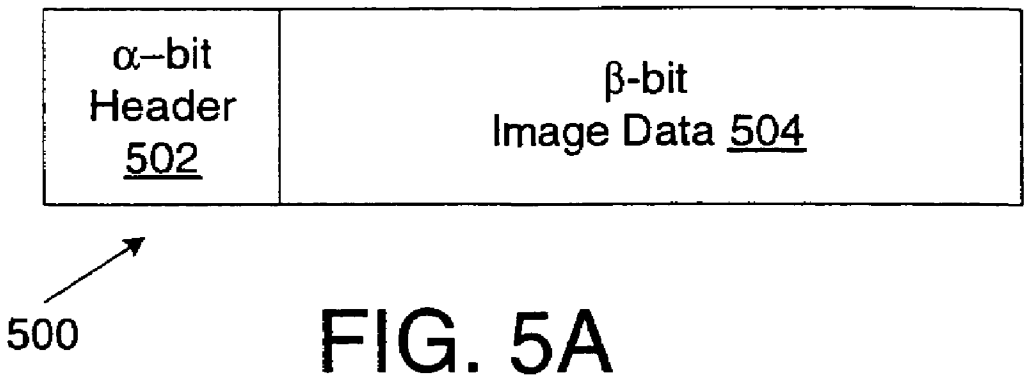


FIG. 5A

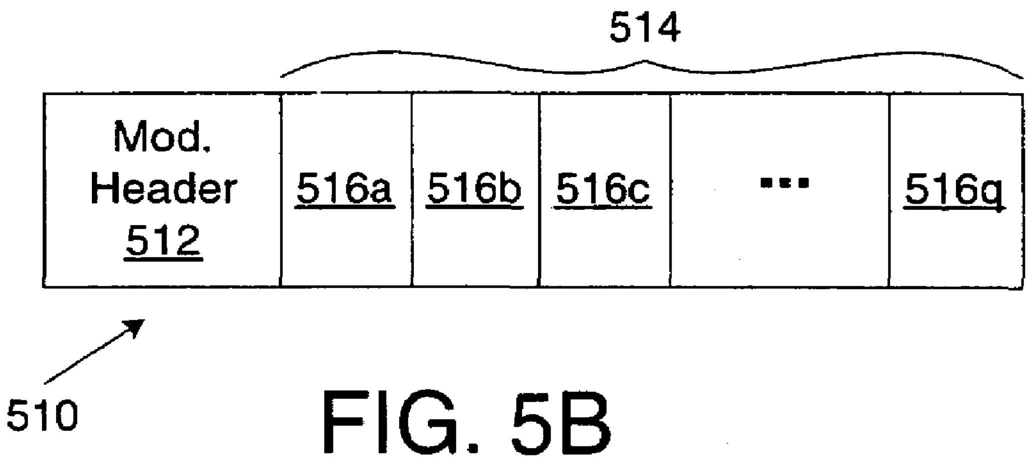


FIG. 5B

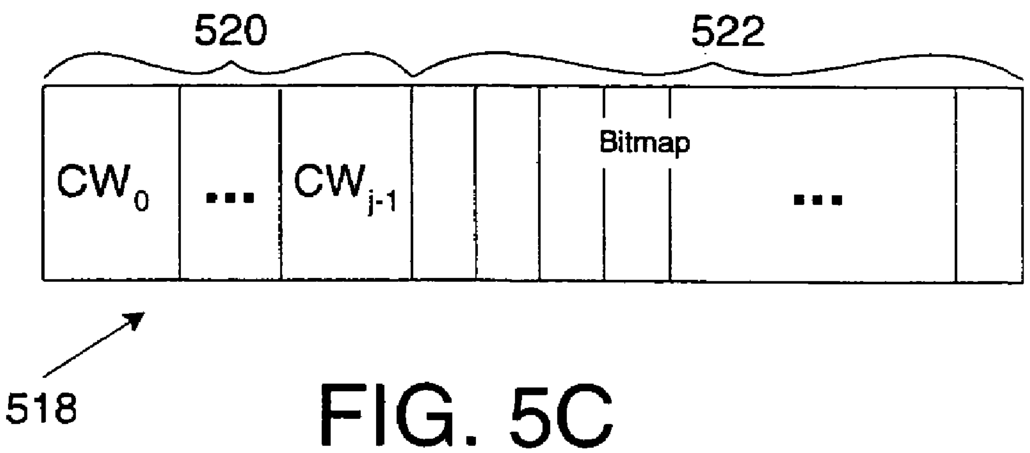


FIG. 5C

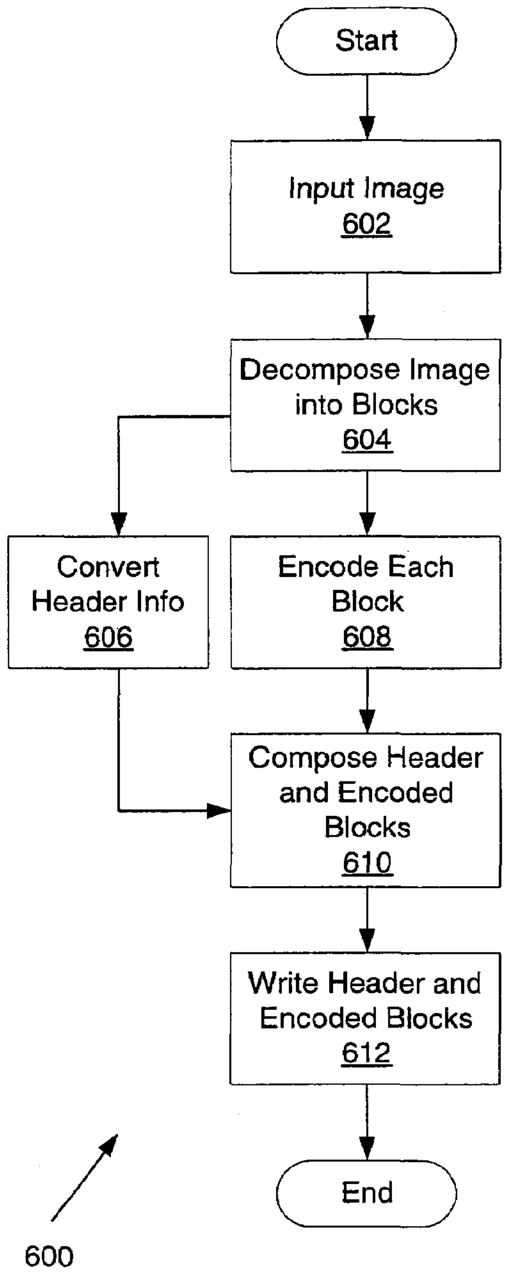

Start
Input Image 602
Decompose Image into Blocks 604
Convert Header Info 606
Encode Each Block 608
Compose Header and Encoded Blocks 610
Write Header and Encoded Blocks 612
End
600

FIG. 6A

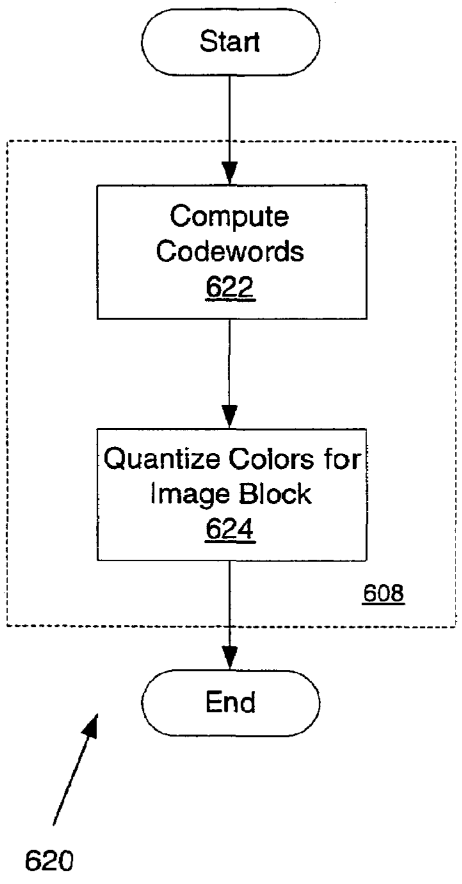

Start
Compute Codewords 622
Quantize Colors for Image Block 624
608
End
620

FIG. 6B

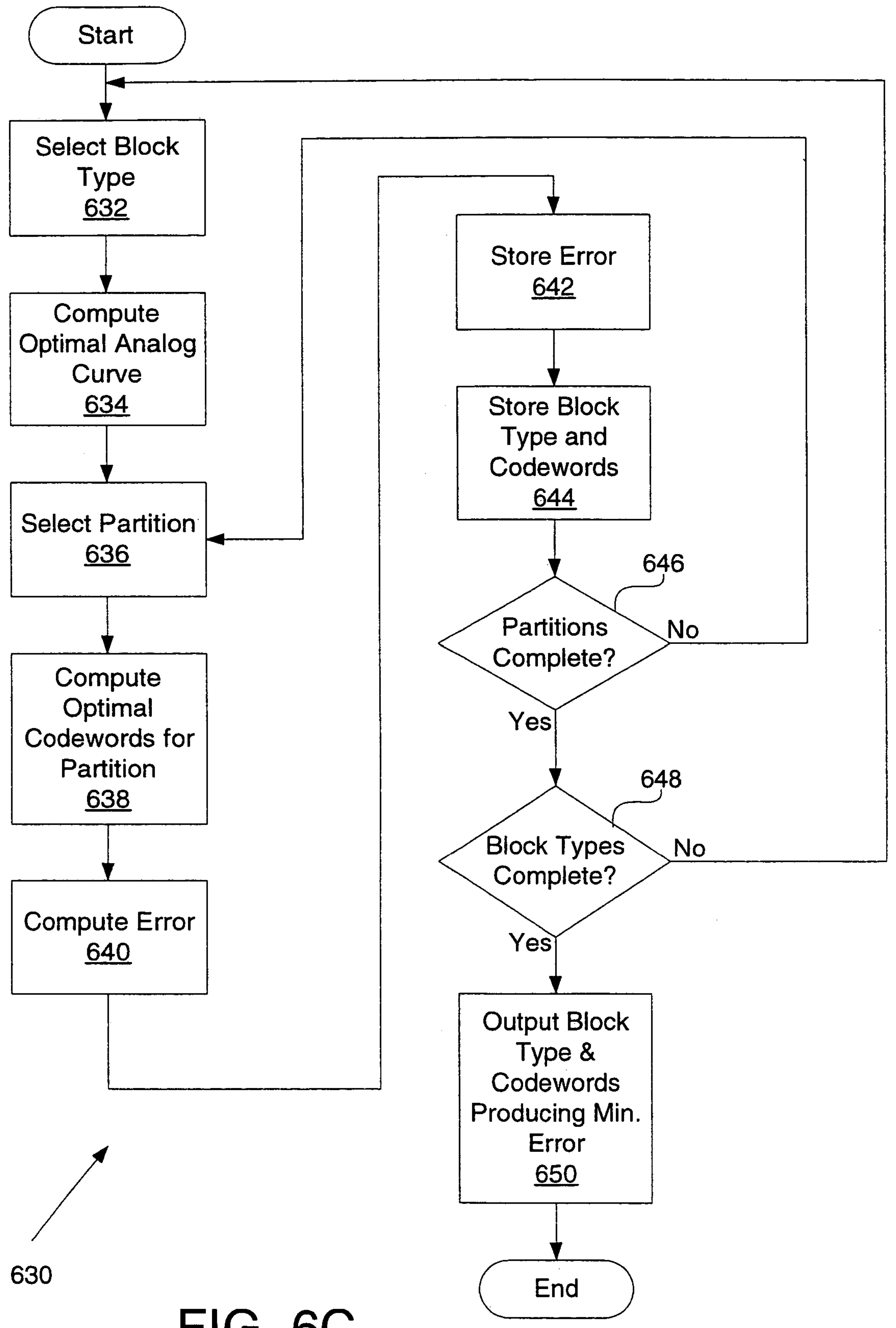

Start
Select Block Type 632
Compute Optimal Analog Curve 634
Select Partition 636
Compute Optimal Codewords for Partition 638
Compute Error 640
Store Error 642
Store Block Type and Codewords 644
646
Partitions Complete?
No
Yes
648
Block Types Complete?
No
Yes
Output Block Type & Codewords Producing Min. Error 650
End
630

FIG. 6C

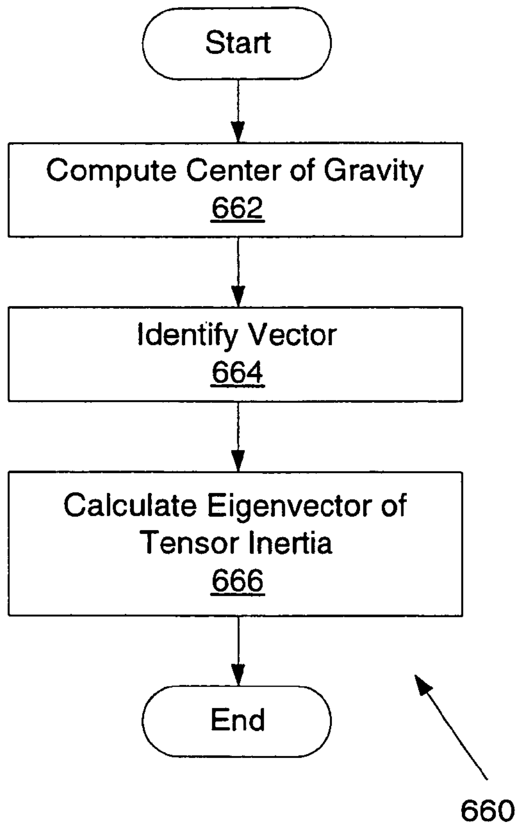

Start
Compute Center of Gravity 662
Identify Vector 664
Calculate Eigenvector of Tensor Inertia 666
End
660

FIG. 6D

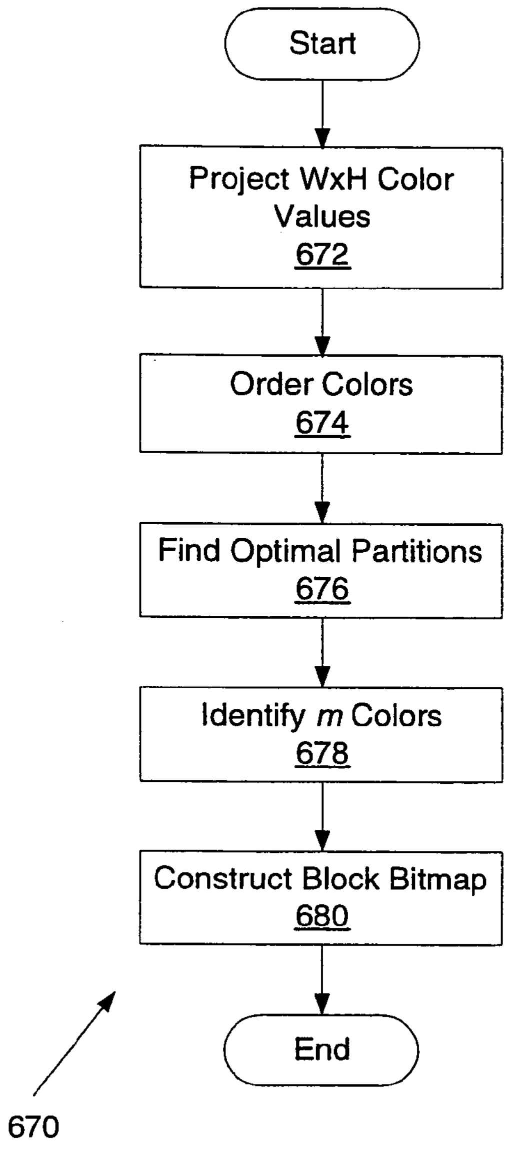

Start
Project WxH Color Values 672
Order Colors 674
Find Optimal Partitions 676
Identify *m* Colors 678
Construct Block Bitmap 680
End
670

FIG. 6E

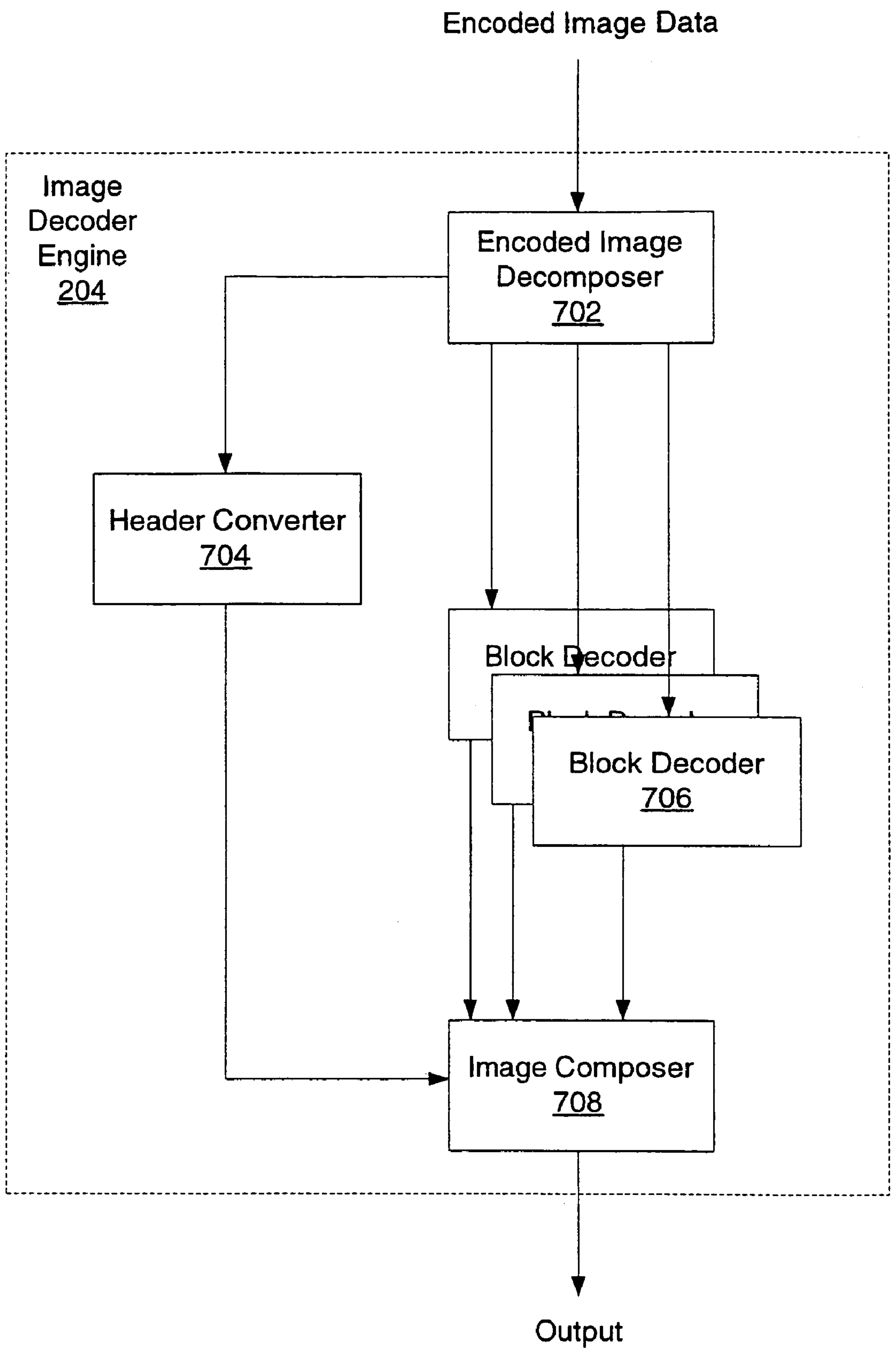

Encoded Image Data
Image Decoder Engine 204
Encoded Image Decomposer 702
Header Converter 704
Block Decoder
Block Decoder 706
Image Composer 708
Output

FIG. 7A

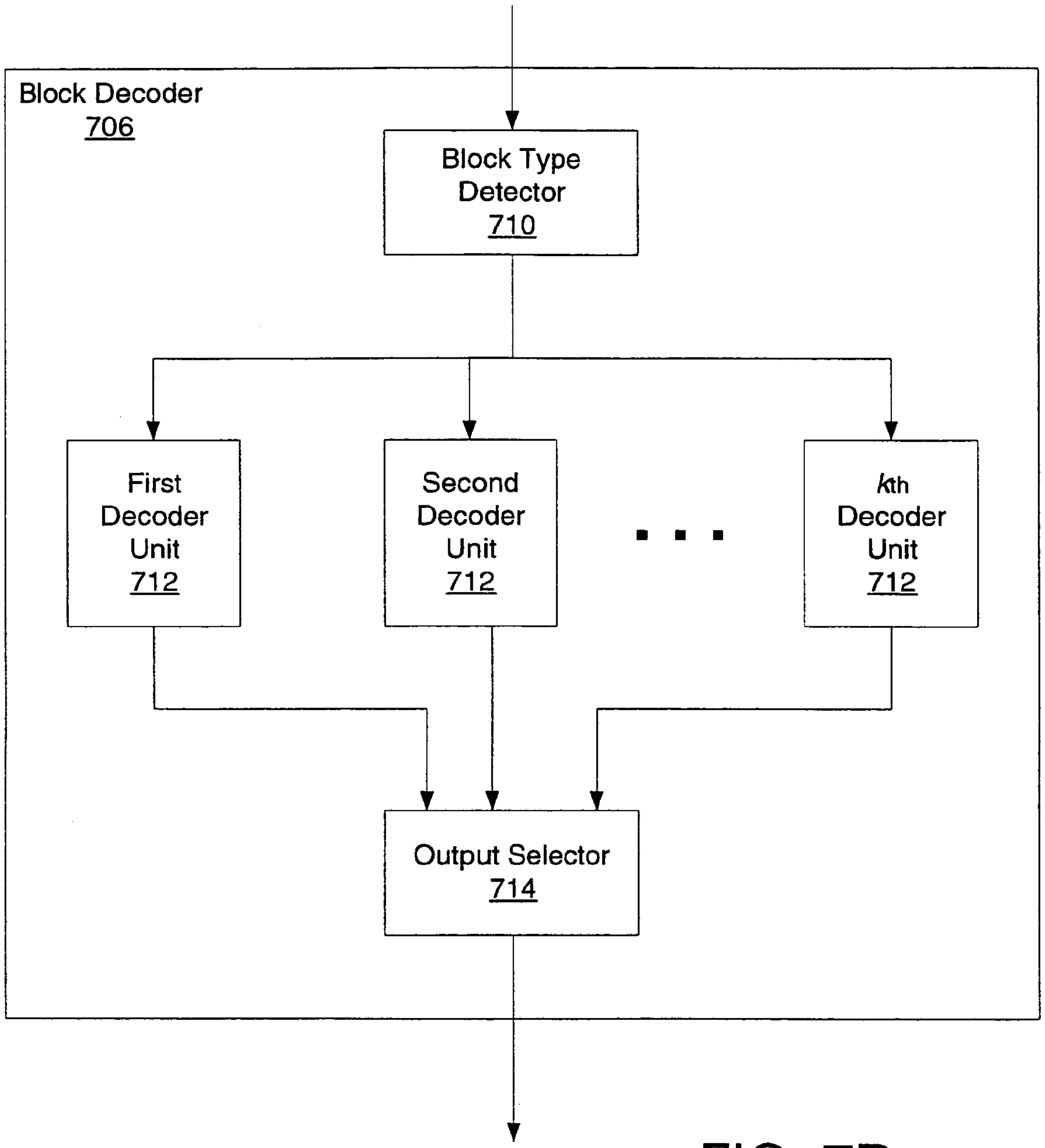

Block Decoder
706
Block Type Detector
710
First Decoder Unit
712
Second Decoder Unit
712
kth Decoder Unit
712
Output Selector
714

FIG. 7B

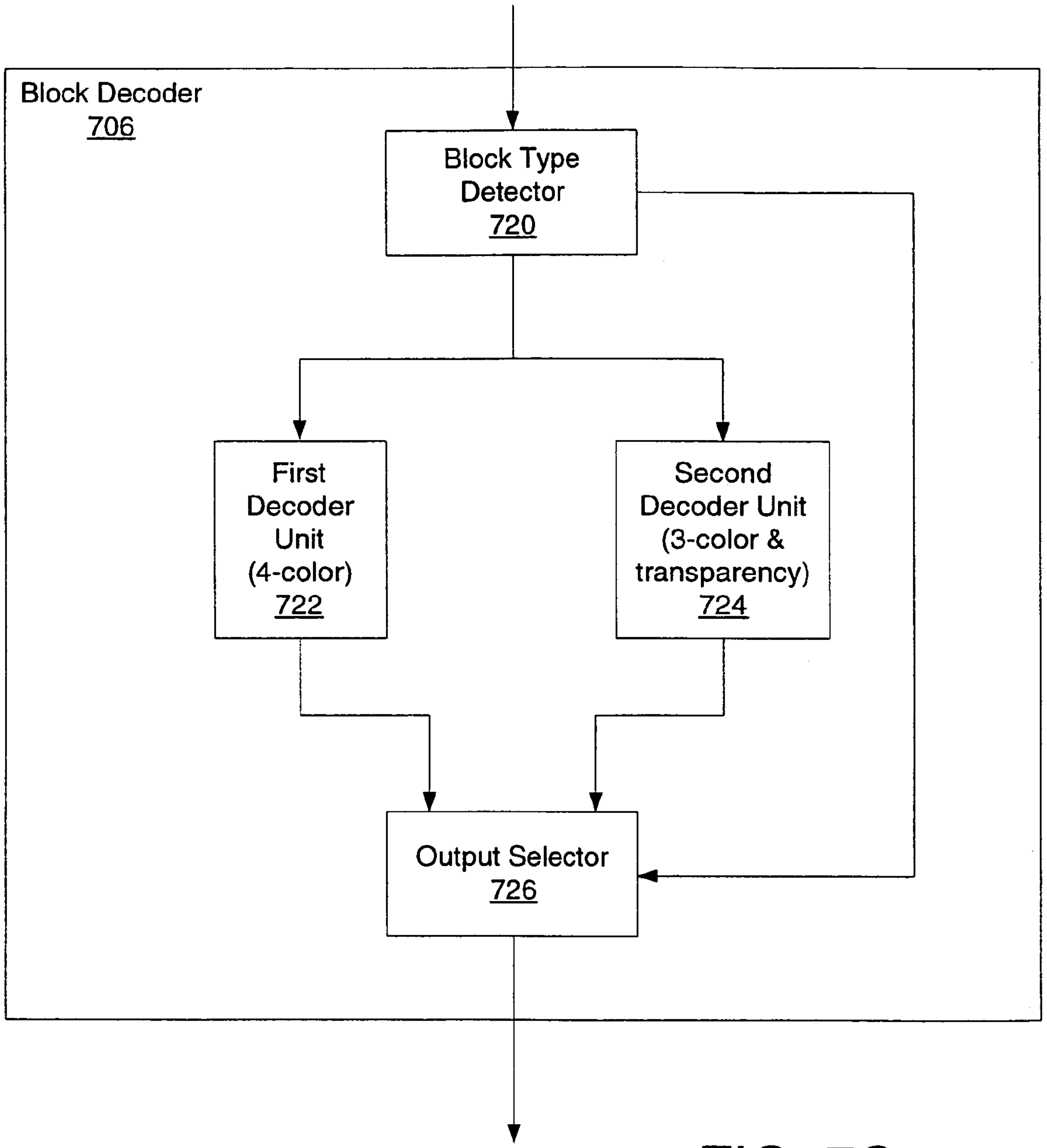

Block Decoder
706
Block Type Detector
720
First Decoder Unit (4-color)
722
Second Decoder Unit (3-color & transparency)
724
Output Selector
726

FIG. 7C

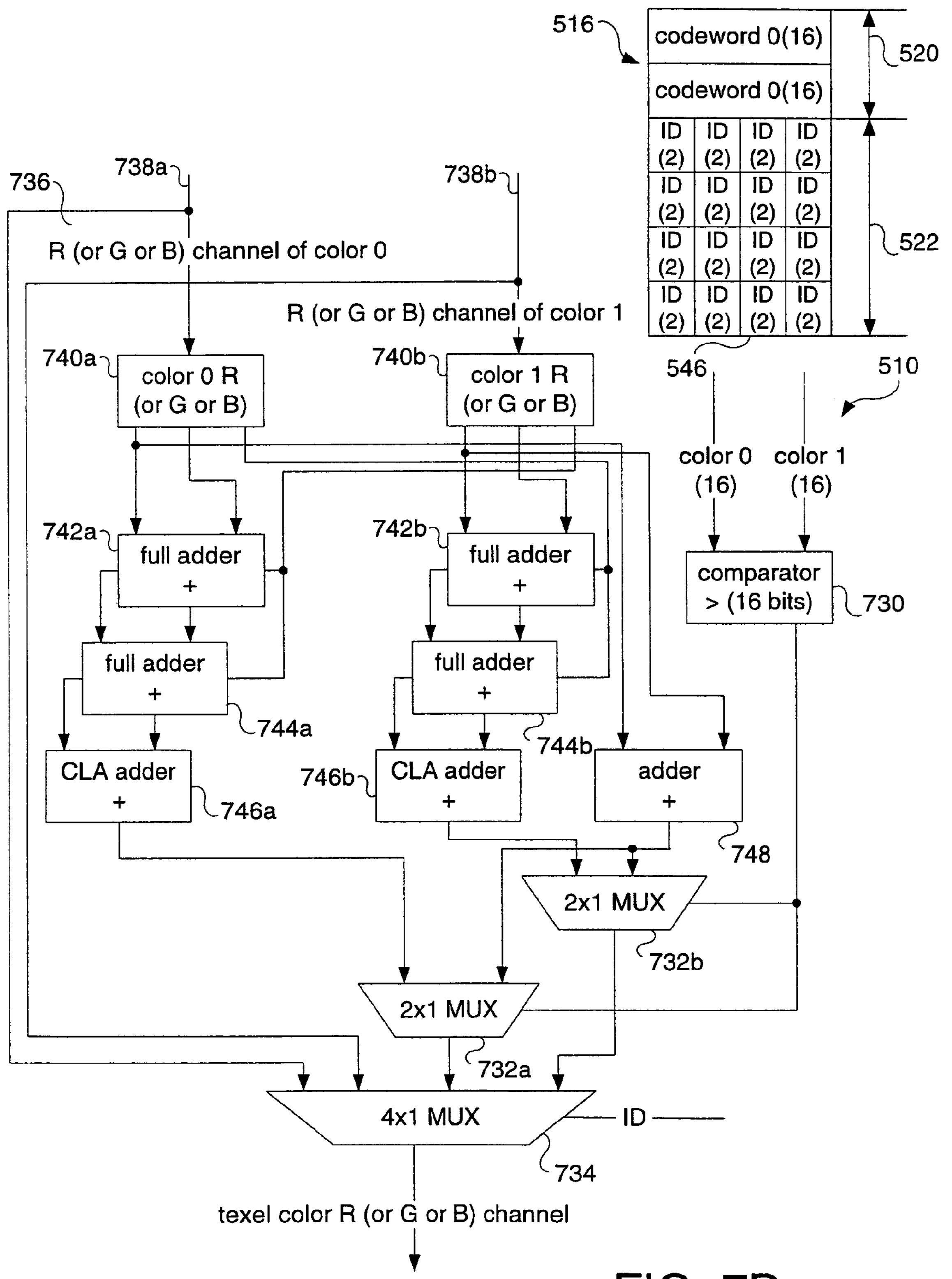

516
codeword 0(16)
codeword 0(16)
ID (2)
520
522
546
510
color 0 (16)
color 1 (16)
comparator > (16 bits)
730
736
738a
738b
R (or G or B) channel of color 0
R (or G or B) channel of color 1
740a
color 0 R (or G or B)
740b
color 1 R (or G or B)
742a
full adder +
742b
full adder +
full adder +
744a
full adder +
744b
CLA adder +
746a
746b
CLA adder +
adder +
748
2x1 MUX
732b
2x1 MUX
732a
4x1 MUX
ID
734
texel color R (or G or B) channel

FIG. 7D

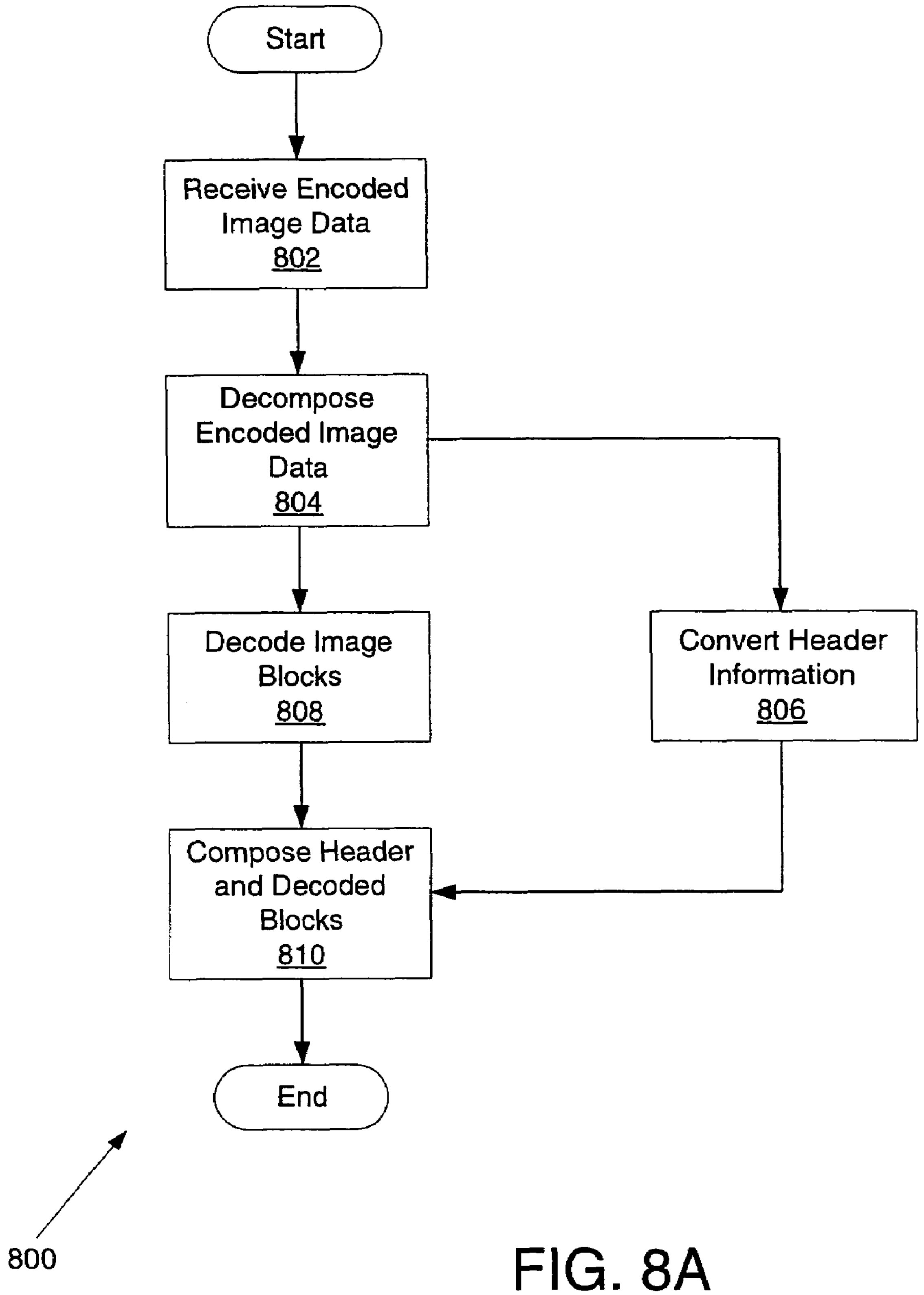

Start
Receive Encoded Image Data 802
Decompose Encoded Image Data 804
Decode Image Blocks 808
Convert Header Information 806
Compose Header and Decoded Blocks 810
End
800

FIG. 8A

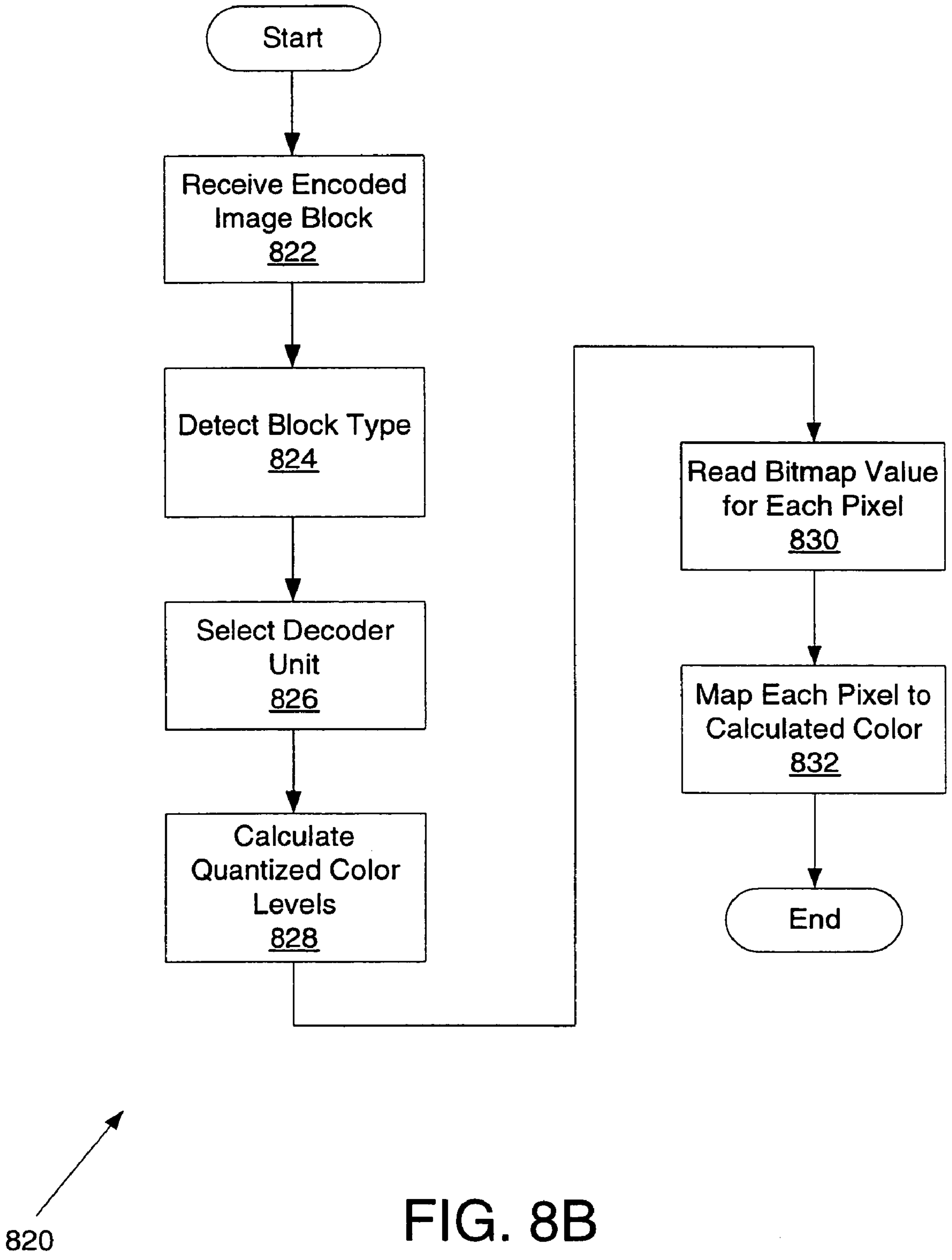

Start
Receive Encoded Image Block 822
Detect Block Type 824
Select Decoder Unit 826
Calculate Quantized Color Levels 828
Read Bitmap Value for Each Pixel 830
Map Each Pixel to Calculated Color 832
End
820

FIG. 8B

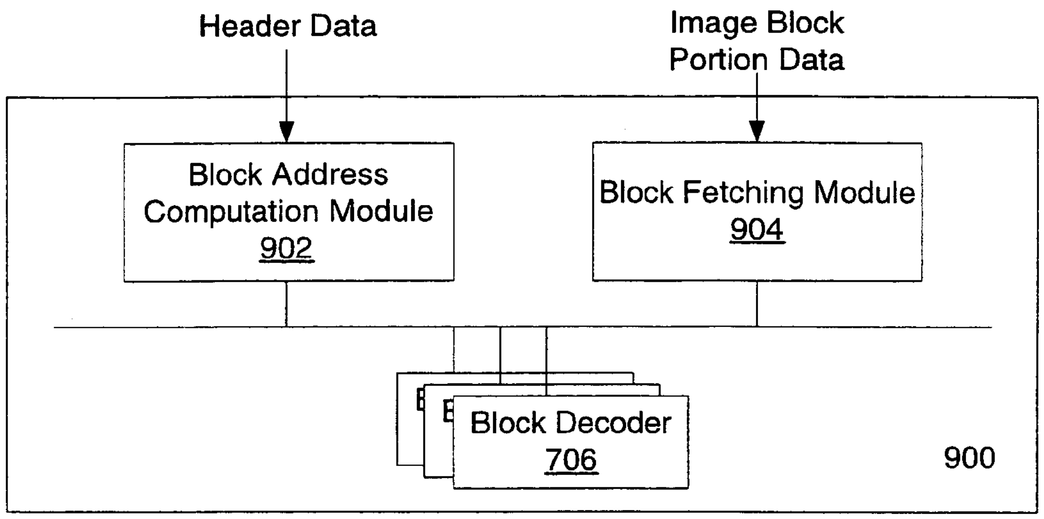

Header Data
Image Block Portion Data
Block Address Computation Module 902
Block Fetching Module 904
Block Decoder 706
900

FIG. 9A

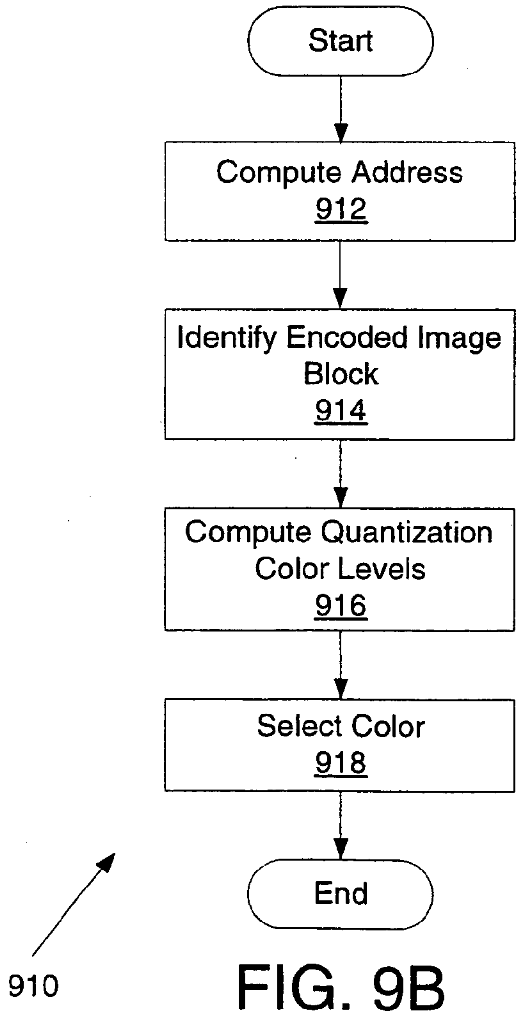

Start
Compute Address 912
Identify Encoded Image Block 914
Compute Quantization Color Levels 916
Select Color 918
End
910

FIG. 9B

IMAGE PROCESSING SYSTEM

CROSS-REFERENCE TO RELATED APPLICATIONS

This application is a continuation of and claims the priority benefit of U.S. patent application Ser. No. 10/052,613 entitled "Fixed-Rate Block-Based Image Compression with Inferred Pixel Values" filed Jan. 17, 2002 and now U.S. Pat. No. 6,775,417, which is a continuation-in-part of U.S. patent application Ser. No. 09/351,930 entitled "Fixed-Rate Block-Based Image Compression with Inferred Pixel Values" filed Jul. 12, 1999 and now U.S. Pat. No. 6,658,146 which is a continuation of U.S. patent application Ser. No. 08/942,860 entitled "System and Method for Fixed-Rate Block-Based Image Compression with Inferred Pixel Values" filed Oct. 2, 1997 and now U.S. Pat. No. 5,956,431. The disclosure of the above-referenced applications and patents are incorporated herein by reference.

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates generally to image processing, and more particularly to three-dimensional rendering using fixed-rate image compression.

2. Description of Related Art

Conventionally, generating images, such as realistic and animated graphics on a computing device, required tremendous memory bandwidth and processing power on a graphics system. Requirements for memory and processing power are particularly true when dealing with three-dimensional images. In order to reduce bandwidth and processing power requirements, various compression methods and systems have been developed including Entropy or lossless encoders, Discrete Cosine Transform (DCT) or JPEG type compressors, block truncation coding, and color cell compression. However, these methods and systems have numerous disadvantages.

Entropy or lossless encoders include Lempel-Ziv encoders, which rely on predictability. For data compression using entropy encoders, a few bits are used to encode most commonly occurring symbols. In stationary systems where probabilities are fixed, entropy coding provides a lower bound for compression than can be achieved with a given alphabet of symbols. However, coding does not allow random access to any given symbol. Part of the compressed data preceding a symbol of interest must be first fetched and decompressed to decode the symbol, requiring considerable processing time and resources, as well as decreasing memory throughput. Another problem with existing entropy methods and systems is that no guaranteed compression factor is provided. Thus, this type of encoding scheme is impractical where memory size is fixed.

Discrete Cosine Transform (DCT) or JPEG-type compressors allow users to select a level of image quality. With DCT, uncorrelated coefficients are produced so that each coefficient can be treated independently without loss of compression efficiency. The DCT coefficients can be quantized using visually-weighted quantization values which selectively discard least important information.

DCT, however, suffers from a number of shortcomings. One problem with DCT and JPEG-type compressors is a requirement of large blocks of pixels, typically, 8×8 or 16×16 pixels, as a minimally accessible unit in order to obtain a reasonable compression factor and quality. Access to a very small area, or even a single pixel involves fetching

a large quantity of compressed data; thus requiring increased processor power and memory bandwidth. A second problem is that the compression factor is variable, therefore requiring a complicated memory management system that, in turn, requires greater processor resources. A third problem with DCT and JPEG-type compression is that using a large compression factor significantly degrades image quality. For example, an image may be considerably distorted with a form of ringing around edges in the image as well as noticeable color shifts in areas of the image. Neither artifact can be removed with subsequent low-pass filtering.

A further disadvantage with DCT and JPEG-type compression is the complexity and significant hardware cost for a compressor and decompressor (CODEC). Furthermore, high latency of a decompressor results in a large additional hardware cost for buffering throughout the system to compensate for the latency. Finally, DCT and JPEG-type compressors may not be able to compress a color-keyed image.

Block truncation coding (BTC) and color cell compression (CCC) use a local one-bit quantizer on 4×4 pixel blocks. Compressed data for such a block consists of only two colors and 16-bits that indicate which of the two colors is assigned to each of 16 pixels. Decoding a BTC/CCC image consists of using a multiplexer with a look-up table so that once a 16-texel (or texture element, which is the smallest addressable unit of a texture map) block (32-bits) is retrieved from memory, the individual pixels are decoded by looking up the two possible colors for that block and selecting the color according to an associated bit from 16 decision bits.

Because the BTC/CCC methods quantize each block to just two color levels, significant image degradation may occur. Further, a two-bit variation of CCC stores the two colors as 8-bit indices into a 256-entry color lookup table. Thus, such pixel blocks cannot be decoded without fetching additional information, which may consume additional memory bandwidth.

The BTC/CCC methods and systems can use a 3-bit per pixel scheme, which stores the two colors as 16-bit values (not indices into a table) resulting in pixel blocks of six bytes. Fetching such units, however, decreases system performance because of additional overhead due to memory misalignment. Another problem associated with BTC/CCC methods is a high degradation of image quality when used to compress images that use color keying to indicate transparent pixels.

Therefore, there is a need for a system and method that maximizes accuracy of compressed images while minimizing storage, memory bandwidth requirements, and decoding hardware complexities. There is a further need for compressing image data blocks into convenient sizes to maintain alignment for random access to any one or more pixels.

SUMMARY OF THE INVENTION

The present invention provides for fixed-rate block based image compression with inferred pixel values. An image processing system includes an image encoder engine and an image decoder engine. The image encoder engine includes an image decomposer, at least one block encoder, and an encoded image composer. The block decomposer decomposes an original image into a header and a plurality of blocks, which are composed of a plurality of image elements or pixels. The block encoder subsequently processes each block. The block encoder includes a selection module, a codeword generation module, and a construction module.

Specifically, the selection module computes a set of parameters from image data values of each set of image elements. The codeword generation module then generates codewords, which are reference image data values such as colors or density values. Subsequently, the construction module uses the codewords to derive a set of quantized image data values. The construction module then maps each of the image element's original image data values with an index to one of the derived image data values. Finally, the codewords and indices are output as encoded image blocks.

Conversely, the image decoder engine includes an encoded image decomposer, at least one block decoder, and an image composer. The image decomposer takes the encoded image and decomposes the encoded image into a header and plurality of encoded image blocks. The block decoder uses the codewords in the encoded image blocks to generate a set of derived image data values. Subsequently, the block decoder maps the index values for each image element to one of the derived image data values. The image composer then reorders the decompressed image blocks in an output data file, which is forwarded to a display device.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a block diagram of a data processing system, according to an embodiment of the present invention;

FIG. **2** is a block diagram of an image processing system;

FIG. **3**A is a block diagram of one embodiment of an image encoder system;

FIG. **3**B is a block diagram of an alternative embodiment of an image encoder system;

FIG. **3**C is a graphical representation of an image block;

FIG. **3**D is a graphical representation of a three-dimensional image block;

FIG. **4** is a block diagram of an image block encoder of FIG. **2**A, **3**A, or **3**B;

FIG. **5**A is a data sequence diagram of an original image;

FIG. **5**B is a data sequence diagram of encoded image data of an original image output from the image encoder system;

FIG. **5**C is a data sequence diagram of an encoded image block from the image block encoder of FIG. **4**;

FIG. **6**A–**6**E are flowcharts illustrating encoding processes, according to the present invention;

FIG. **7**A is a block diagram of an image decoder system;

FIG. **7**B is a block diagram of one embodiment of a block decoder of FIG. **7**A;

FIG. **7**C is a block diagram of an alternative embodiment of a block decoder of FIG. **7**A;

FIG. **7**D is a logic diagram illustrating an exemplary decoder unit, according to the present invention;

FIG. **8**A is a flowchart illustrating a decoding process of the image decoder of FIG. **2**;

FIG. **8**B is a flowchart illustrating operations of the block encoder of FIG. **7**A;

FIG. **9**A is a block diagram of a subsystem for random access to a pixel or an image block; and

FIG. **9**B is a flowchart illustrating random access to a pixel or an image block.

DESCRIPTION OF THE PREFERRED EMBODIMENT

FIG. **1** is a block diagram of an exemplary data processing system **100** for implementing the present invention. The data processing system **100** comprises a CPU **102**, a memory **104**, a storage device **106**, input devices **108**, output devices

110, and a graphics engine **112** all of which are coupled to a system bus **114**. The memory **104** and storage device **106** store data within the data processing system **100**. The input device **108** inputs data into the data processing system **100**, while the output device **110** receives data from the data processing system **100**. Although the data bus **114** is shown as a single line, alternatively, the data bus **114** may be a combination of a processor bus, a PCI bus, a graphic bus, or an ISA bus.

FIG. **2** is a block diagram of an exemplary image processing system **200**. In one embodiment, the image processing system **200** is contained within the graphics engine **112** (FIG. **1**). The image processing system **200** includes an image encoder engine **202** and an image decoder engine **204**. The image processing system **200** may also include, or be coupled to, an image source unit **206**, which provides images to the image encoder engine **202**. Further, the image processing system **200** may include or be coupled to an output unit **208** to which processed images are forwarded for storage or further processing. Additionally, the image processing system **200** may be coupled to the memory **104** (FIG. **1**) and the storage device **106** (FIG. **1**). In an alternative embodiment, the image encoder engine **202** and the image decoder engine **204** are contained within different computing devices, and the encoded images pass between the two engines **202** and **204**.

Within the image encoder engine **202**, images are broken down into individual blocks and processed before being forwarded, for example, to the storage device **106** as compressed or encoded image data. When the encoded image data are ready for further processing, the encoded image data are forwarded to the image decoder engine **204**. The image decoder engine **204** receives the encoded image data and decodes the data to generate an output that is a representation of the original image that was received from the image source unit **206**.

FIGS. **3**A and **3**B are block diagrams illustrating two exemplary embodiments of the image encoder engine **202** of FIG. **2**. The image encoder engine **202** comprises an image decomposer **302**, a header converter **304**, one or more block encoders **306** in FIG. **3**A (**306***a*–**306***n*, where n is the nth encoder in FIG. **3**B), and an encoded image composer **308**. The image decomposer **302** is coupled to receive an original image **310** from a source, such as the image source unit **206** (FIG. **2**), and forwards information from a header of the original image **310** to the header converter **304**. Subsequently, the header converter **304** modifies the original header to generate a modified header, as will be described further in connection with FIG. **5**B. The image decomposer **302** also breaks, or decomposes, the original image **310** into R numbers of image blocks, where R is any integer value. The number of image blocks the original image **310** is broken into may depend on the number of image pixels. In an exemplary embodiment, the image **310** having A image pixels by B image pixels will, typically, be (A/4)×(B/4) blocks. For example, an image that is 256 pixels by 256 pixels will be broken down into 64×64 blocks. In the present embodiment, the image is decomposed such that each image block is 4 pixels by 4 pixels (16 pixels). Those skilled in the art will recognize that the number of pixels or the image block size may be varied.

Briefly turning to FIG. **3**C, an example of a single image block **320** is illustrated. The image block **320** is composed of image elements (pixels) **322**. The image block **320** may be defined as an image region, W pixels in width by H pixels in height. In the embodiment of FIG. **3**C, the image block **320** is W=4 pixels by H=4 pixels (4×4).

In an alternative embodiment, the original image **310** (FIG. **3**A or **3**B) may be a three-dimensional volume data set as shown in FIG. **3**D. FIG. **3**D illustrates an exemplary three-dimensional image block **330** made up of sixteen image elements (volume pixels or voxels) **332**. Image block **330** is defined as an image region W voxels in width, H voxels in height, and D voxels in depth.

The three-dimensional volume data set may be divided into image blocks of any size or shape. For example, the image may be divided along a z-axis into a plurality of x×y×z sized images, where z=1. Each of these x×y×1 images may be treated similarly with two-dimensional images, where each x×y×1 image is divided into two-dimensional image blocks, as described above with respect to FIG. **3**C. However, decomposing the three-dimensional image into two-dimensional "slices" for compression does not fully utilize the graphical similarities that may exist in the z (depth) direction in a three-dimensional image. To utilize such similarities, the volume data may be decomposed into a plurality of three-dimensional image blocks. It will be understood that in alternative embodiments, other combinations of W×H×D are possible, and may be more desirable, depending on the data being compressed.

This type of three-dimensional image data is used, for example, in medical imaging applications such as ultrasound or magnetic resonance imaging ("MRI"). In such an application, a body part is scanned to produce a three-dimensional matrix of image elements (i.e., image block comprised of voxels **320**). The image is x voxels wide by y voxels high by z voxels deep. In this example, each voxel provides density data regarding characteristics of body tissue. In ultrasound applications, each voxel may be provided with a brightness level indicating the strength of echoes received during scanning.

In the embodiment of FIG. **3**D, the original image **310** is a three-dimensional data volume where the image data are density values. In alternative embodiments, other scalar data types may be represented in the original image **310**, such as transparency or elevation data. In further embodiments, vector data, such as the data used for "bump maps", may be represented.

Referring back to FIGS. **3**A and **3**B, each block encoder **306** receives an image block **320** from the image decomposer **302**, and encodes or compresses each image block **320**. Subsequently, each encoded image block is forwarded to the encoded image composer **308**, which orders the encoded image blocks in a data file. Next, the data file from the encoded image composer **308** is concatenated with the modified header from the header converter **304** to generate an encoded image data file that is forwarded to an output **312**. Thus, the modified header and the encoded image blocks together form the encoded image data that represent the original image **310**. Alternatively, having more than one block encoder **306***a*–**306***n*, as shown in FIG. **3**B, allows for encoding multiple image blocks simultaneously, one image block per block encoder **306***a*–**306***n*, within the image encoder engine **202**. Advantageously, simultaneous encoding increases image processing efficiency and performance.

The image data associated with the original image **310** may be in any one of a variety of formats including red-green-blue ("RGB"), YUV **420** (YUV are color models representing luminosity and color difference signals), YUV **422**, or a propriety color space. In some cases, conversion to a different color space before encoding the original image **310** may be useful. In one embodiment, each image block **320** is a 4×4 set of pixels where each pixel **322** is 24-bits in size. For each pixel **322**, there are 8-bits for a Red ("R")-channel, 8-bits for a Green ("G")-channel, and 8-bits for a Blue ("B")-channel in an RGB implementation color space. Alternatively, each encoded image block is also a 4×4 set of pixels with each pixel being only 2-bits in size and having an aggregate size of 4-bits as will be described further below.

FIG. **4** is a block diagram illustrating an exemplary block encoder **306** of FIGS. **3**A and **3**B. The block encoder **306** includes a quantizer **402** and a bitmap construction module **404**. Further, the quantizer **402** includes a block type module **406**, a curve selection module **408**, and a codeword generation module **410**.

Each image block **320** (FIG. **3**C) of the decomposed original image **310** (FIGS. **3**A and **3**B) is received and initially processed by the quantizer **402** before being forwarded to the bitmap construction module **404**. The bitmap construction module **404** outputs encoded image blocks for the encoded image composer **308** (FIGS. **3**A and **3**B) to order. The bitmap construction module **404** and the modules of the quantizer **402** are described in more detail below.

Briefly, FIG. **5**A is a diagram of a data sequencer or string **500** representing the original image **310** (FIGS. **3**A and **3**B) that is received by the block decomposer **302** (FIGS. **3**A and **3**B). The data string **500** includes an α-bit header **502** and a β-bit image data **504**. The header **502** may include information such as pixel width, pixel height, format of the original image **310** (e.g., number of bits to the pixel in RGB or YUV format), as well as other information. The image data **504** are data representing the original image **310**, itself.

FIG. **5**B is a diagram of a data sequence or string **510** representing encoded image data that are generated by the image encoder engine **202** (FIG. **2**). The encoded image data string **510** includes a modified header portion **512** and an encoded image block portion **514**. The modified header portion **512** is generated by the header converter **304** (FIGS. **3**A and **3**B) from the original α-bit header **502** (FIG. **5**A) and includes information about file type, number of bits per pixel of the original image **310** (FIGS. **3**A and **3**B), addressing in the original image **310**, other miscellaneous encoding parameters, as well as the width and height information indicating size of the original image **310**. The encoded image block portion **514** includes encoded image blocks **516***a–q* from the block encoders **306** (FIGS. **3**A and **3**B) where q is the number of blocks resulting from the decomposed original image **310**.

FIG. **5**C is a diagram of a data sequence or string **518** representing an encoded image block. The data string **518** may be similar to any one of the encoded image blocks **516***a–q* (FIG. **5**B) shown in the encoded image data string **510** of FIG. **5**B.

The encoded image block data string **518** includes a codeword section **520** and a bitmap section **522**. The codeword section **520** includes j codewords, where j is an integer value, that are used to compute colors of other image data indexed by the bitmap section **522**. A codeword is an n-bit data string that identifies a pixel property, such as color component, density, transparency, or other image data values. In one embodiment, there are two 16-bit codewords CW_0 and CW_1 (j=2). The bitmap section **522** is a Q-bit data portion and is described in more detail in connection with FIG. **6**B.

In an alternative embodiment, each encoded image block is 64-bits, which includes two 16-bit codewords and a 32-bit (4×4×2 bit) bitmap **522**. Encoding the image block **320** (FIG. **3**C) as described above provides greater system flexibility and increased data processing efficiency. In a further exemplary embodiment, each 32-bit bitmap section **522** may be a three-dimensional 32-bit bitmap.

FIGS. **6**A–**6**E describe operations of the image encoder engine **202** (FIG. **2**). In flowchart **600**, a general operation of the image encoder engine **202** is shown. In block **602**, a data string **500** (FIG. **5**A) of the original image **310** (FIGS. **3**A and **3**B), which includes the α-bit header **502** (FIG. **5**A) and the β-bit image data **504** (FIG. **5**A), is input into the image decomposer **302** (FIGS. **3**A and **3**B). The image decomposer **302** decomposes the image **310** into the α-bit header and a plurality of blocks in block **604**. The α-bit header **502** is then forwarded to the header converter **304** (FIGS. **3**A and **3**B). Subsequently, the header converter **304** generates a modified header **512** (FIG. **5**B) from the α-bit header **502** in block **606**. The modified header **512** is then forwarded to the encoded image composer **308** (FIGS. **3**A and **3**B).

Simultaneous with the header conversion process, each image block **320** is encoded in block **608** by one or more of the block encoders **306***a*–**306***n* (FIGS. **3**A and **3**B) to generate the encoded image blocks **516** (FIG. **5**B). Each image block **320** may be processed sequentially in one block encoder **306**, or multiple image blocks **320** may be processed in parallel in multiple block encoders **306***a*–**306***n*.

The encoded image blocks **516** are output from the block encoders **306**, and are placed into a predefined order by the encoded image composer **308**. In one embodiment, the encoded image blocks **516** are arranged in a file from left to right and top to bottom and in the same order in which the encoded image blocks **516** were broken down by the image decomposer **302** (FIGS. **3**A and **3**B). The image encoder engine **202** subsequently composes the modified header information **512** from the header converter **304** and the encoded image blocks **516***a*–**516***q* in block **610**. Specifically, the modified header **512** and the ordered encoded image blocks **516** are concatenated to generate the encoded image data file **510** (FIG. **5**B), which may be written as encoded output **312** (FIGS. **3**A and **3**B) to the memory **104**, storage device **106**, or any output device **110** (FIG. **1**) in block **612**.

FIG. **6**B is a flowchart **620** showing the encoding process of block **608** (FIG. **6**A) in more detail. In block **622**, codewords **520** (FIG. **5**C) are computed by the codeword generation module **410** (FIG. **4**). The process for computing these codewords **520** is described in more detail in connection with FIG. **6**C.

Once the codewords **520** have been computed, pixel values or properties, such as colors, for the image block **320** (FIG. **3**C) are computed or quantized in block **624**. Specifically, the codewords **520** provide points in a pixel space from which m quantized pixel values may be inferred. The m quantized pixel values are a limited subset of pixels in a pixel space that are used to represent the current image block. The process for quantizing pixel values, and more specifically colors, will be described infra in connection with FIGS. **8**A and **8**B. Further, the embodiments will now be described with respect to colors of a pixel value although one skilled in the art will recognize that, in general, any pixel value may be used with respect to the present invention. Therefore, the image data, which is quantized may be any form of scalar or vector data, such as density values, transparency values, and "bump map" vectors.

In an exemplary embodiment, each pixel is encoded with two bits of data which can index one or m quantized colors, where m=4 in this embodiment. Further, four quantized colors are derived from the two codewords **520** where two colors are the codewords **520**, themselves, and the other two colors are inferred from the codewords **520**, as will be described below. It is also possible to use the codewords **520** so that there is one index to indicate a transparent color and three indices to indicate colors, of which one color is inferred.

In another embodiment, the bitmap **522** (FIG. **5**C) is a 32-bit data string. The bitmap **522** and codewords **520** are output in block **624** as a 64-bit data string representing an encoded image block **518**. Specifically, the encoded image block **514** (FIG. **5**B) includes two 16-bit codewords **520** (n=16) and a 32-bit bitmap **522**. Every codeword **520** that is a 16-bit data string includes a 5-bit red-channel, 6-bit green-channel, and 5-bit blue-channel.

Each of the encoded image blocks **516** is placed together and concatenated with modified header information **512** derived from the original α-bit header **502** of the original image **310** (FIGS. **3**A and **3**B). A resulting output is the encoded image data **510** representing the original image **310**.

FIG. **6**C is a flowchart **630** illustrating a process for computing codewords for the image blocks **320** (FIG. **3**C), and relates to color quantizing using quantizer **402** (FIG. **4**). The process for computing codewords can be applied to all scalar and vector image data types. In select block type **632**, the quantizer **402** uses the block type module **406** (FIG. **4**) to select a first block type for the image block **320** that is being processed. For example, a selected block type may be a four-color or a three-color plus transparency block type, where the colors within the particular block type have equidistant spacing in a color space. Those of ordinary skill in the art will readily recognize that selecting a block type for each image is not intended to be limiting in any way. Instead, the present invention processes image blocks that are of a single block type, which eliminates the need to distinguish between different block types, such as the three- and four-color block types discussed above. Consequently, the block type module **406** and select block type **632** are optional.

Once the block type is selected, the quantizer **402** computes an optimal analog curve for the block type in block **634**. Computation of the optimal analog curve will be further described in connection with FIG. **6**D. The analog curve is used to simplify quantizing of the colors in the image block. Subsequently in block **636**, the quantizer **402** selects a partition of points along the analog curve, which is used to simplify quantizing of the colors in the image block. A partition may be defined as a grouping of indices {1 . . . (W×H)} into m nonintersecting sets. In one embodiment, the indices (1 . . . 16) are divided into three or four groups or clusters (i.e., m=3 or 4) depending on the block type.

Once a partition is selected, optimal codewords for the particular partition are computed in block **638**. In addition to computing the codewords, an error value (square error as described infra) for the codeword is also computed in block **640**. Both computations will be described in more detail in connection with FIG. **6**E. If the computed error value is the first error value, the error value is stored in block **642**. Alternatively, the computed error value is stored if it is less than the previously stored error value. For each stored error value, corresponding block type and codewords are also stored in block **644**. The process of flowchart **630** seeks to find the block type and codewords that minimize the error function.

Next in block **646**, the code generation module **410** (FIG. **4**) determines if all possible partitions are completed. If there are more partitions, the code generation module **410** selects the next partition, computes the codewords and associated error values, and stores the error values, associated block types, and codewords if the error value is less than the previously stored error value.

After all the possible partitions are completed, the codeword generation module **410** determines, in block **648**, whether all block types have been selected. If there are more block types, the codeword generation module **410** selects the next block type and computes the codeword and various values as previously described. After the last block type has been processed, the codeword generation module **410** outputs a result of the block type and codewords **520** (FIG. **5**C) having the minimum error in block **650**.

In an alternative embodiment, the optimal analog curve may be computed before selecting the block type. That is, the optimal analog curve is computed before the selection of the block type and partition, computation of the codewords and error values, and storage of the error value, block type, and codeword. Computing the optimal analog curve first is useful if all block types use the same analog curve and color space because the analog curve does not need to be recomputed for each block type.

FIG. **6**D is a flowchart **660** describing a process of identifying the optimal analog curve. The curve selection module **408** (FIG. **4**) first computes a center of gravity for pixel colors of an image block **320** (FIG. **3**C) in block **662**. The center of gravity computation includes averaging the pixel colors. Once the center of gravity is computed, a vector in color space is identified in block **664** to minimize the first moment of the pixel colors of the image block **320**. Specifically for identifying a vector, a straight line is fit to a set of data points, which are the original pixel colors of the image block **320**. The straight line is chosen passing through the center of gravity of the set of data points such that it minimizes a "moment of inertia" (i.e., square error). For example, to compute a direction of a line minimizing the moment of inertia for three pixel properties, tensor inertia, T, is calculated from individual colors as follows:

$$T = \sum_{i=1}^{W\times H} \begin{bmatrix} C_{1i}^2 + C_{2i}^2 & -C_{0i}C_{1i} & -C_{0i}C_{2i} \\ -C_{0i}C_{1i} & C_{0i}^2 C_{2i}^2 & -C_{1i}C_{2i} \\ -C_{0i}C_{2i} & -C_{2i}C_{1i} & C_{0i}^2 + C_{1i}^2 \end{bmatrix}$$

where C_0, C_1, and C_2 represent pixel properties (e.g., color components in RGB or YUV) relative to a center of gravity. In one embodiment of an RGB color space, C_{0i} is a value of red, C_{1i} is a value of green, and C_{2i} is a value of blue for each pixel, i, of the image block. Further, i takes on integer values from 1 to W×H, so that if W=4 and H=4, i ranges from 1 to 16.

An eigenvector of tensor inertia, T, with the smallest eigenvalue is calculated in block **666** using conventional methods. An eigenvector direction along with the calculated gravity center, defines an axis that minimizes the moment of inertia. This axis is used as the optimal analog curve, which, in one embodiment, is a straight line. Those of ordinary skill in the art will readily recognize that the optimal analog curve is not limited to a straight line, but may include a set of parameters, such as pixel values or colors, that minimizes the moment of inertia or mean-square-error when fit to the center of gravity of the pixel colors in the image block. The set of parameters may define any geometric element, such as a curve, plate, trapezoid, or the like.

FIG. **6**E is a flowchart **670** describing the process undertaken by the codeword generation module **410** (FIG. **4**) for selecting the partitions, computing the codewords and associated error for the partitions, and storing the error value, block type, and codeword if the error value is less than a previously stored error value. In block **672**, the codeword generation module **410** projects the W×H color values onto the previously constructed optimal analog curve. The value of W×H is the size in number of pixels of an image block **320** (FIG. **3**C). In one embodiment where W and H are both four pixels, W×H is 16 pixels.

Subsequently in block **674**, the colors are ordered sequentially along the analog curve based on a position of the color on a one-dimensional analog curve. After the colors are ordered, the codeword generation module **410** searches, in block **676**, for optimal partitions. Thus, the codeword generation module **410** takes the W×H colors (one color associated with each pixel) that are ordered along the analog curve and partitions and groups the colors into a finite number of clusters with a predefined relative spacing. In one embodiment where W=4 and H=4 (i.e., W×H is 16), the 16 colors are placed in three and four clusters (i.e., m=3 or 4).

In conducting the search for the optimal partition, a color selection module within the codeword generation module **410** finds the best m clusters from the W×H points projected onto the optimal curve, so that the error associated with the selection is minimized. The best m clusters are determined by minimizing the mean-square-error with the constraint that the points associated with each cluster are spaced to conform to the predefined spacing.

In one embodiment for a block type of four equidistant colors, the error may be defined as a square error along the analog curve, such as

$$E^2 = \sum_{cluster0} (x_i - p_0)^2 + \sum_{cluster1} \left[x_i - \left(\frac{2}{3}p_0 + \frac{1}{3}p_1\right)\right]^2 + \sum_{cluster2} \left[x_i - \left(\frac{1}{3}p_0 + \frac{2}{3}p_1\right)\right]^2 + \sum_{cluster3} (x_i - p_1)^2$$

where E is the error for the particular grouping or clustering, p_0 and p_1 are the coded colors, and x_i are the projected points on the optimal analog curve.

In instances where the block type indicates three equidistant colors, the error may be defined as a squared error along the analog curve, such as

$$E^2 = \sum_{cluster0} (x_i - p_0)^2 + \sum_{cluster1} \left[x_i - \left(\frac{1}{2}p_0 + \frac{1}{2}p_1\right)\right]^2 + \sum_{cluster2} (x_i - p_1)^2$$

After the resulting optimal codewords **520** are identified, the codewords **520** are forwarded to the bitmap construction module **404** (FIG. **4**). The bitmap construction module **404** uses the codewords **520** to identify the m colors that may be specified or inferred from those codewords **520** in block **678**. In one embodiment, the bitmap construction module **404** uses the codewords **520** (e.g., CW_0 and CW_1) to identify the three or four colors that may be specified or inferred from those codewords **520**.

Next in block **680**, the bitmap construction module **404** constructs a block bitmap **522** (FIG. **5**C) using the codewords **520** associated with the image block **320** (FIG. **3**C).

Colors in the image block **320** are mapped to the closest color associated with one of the quantized colors specified by, or inferred from, the codewords **520**. The result is a color index, referenced as ID, per pixel in the block identifying the associated quantized color.

Information indicating the block type is implied by the codewords **520** and the bitmap **522**. In one embodiment, the order of the codewords **520** indicates the block type. If a numerical value of CW_0 is greater than a numerical value of CW_1, the image block is a four-color block. Otherwise, the block is a three-color plus transparency block.

In one embodiment discussed above, there are two-color image block types. One color image block type has four equidistant colors, while the other color image block type has three equidistant colors with the fourth color index used to specify that a pixel is transparent. For both color image block types, the color index is two bits. In an embodiment with density values in place of color values, each density image block type has four equidistant density values.

The output of the bitmap construction module **404** is an encoded image block **514** (FIG. **5**B) having the m codewords **520** plus the bitmap **522**. Each encoded image block **516** is received by the encoded image composer **308** (FIGS. **3**A and **3**B) that, in turn, orders the encoded image blocks **516** in a file. In one embodiment, the encoded image blocks **516** are arranged from left to right and from top to bottom and in the same order as the blocks were broken down by the image decomposer **302**. The ordered file having the encoded image blocks **516** is concatenated with the modified header information **512** that is derived from the α-bit header **502** of the original image **310** (FIGS. **3**A and **3**B) to generate the encoded image data **510** that is the output of the image encoder engine **202** (FIG. **2**). The output may then be forwarded to the memory **104**, the storage device **106**, or the output device **110** (FIG. **1**).

The exemplary embodiment of the image encoder engine **202** advantageously reduces the effective data size of an image from 24-bits per pixel to 4-bits per pixel. Further, the exemplary embodiment beneficially addresses transparency issues by allowing codewords to be used with a transparency identifier.

FIG. **7**A is a block diagram of an exemplary image decoder engine **204** (FIG. **2**). The image decoder engine **204** includes an encoded image decomposer **702**, a header converter **704**, one or more block decoders **706** (**706***a*–**706***p*, where p represents the last block decoder), and an image composer **708**. The encoded image decomposer **702** is coupled to receive the encoded image data **514** (FIG. **5**B) output from the image encoder engine **202** (FIG. **2**). The encoded image decomposer **702** receives the encoded image data string **510** and decomposes, or breaks, the encoded image data string **510** into the header **512** (FIG. **5**B) and the encoded image blocks **514** (FIG. **5**B). Next, the encoded image decomposer **702** reads the modified header **512**, and forwards the modified header **512** to the header converter **704**. The encoded image decomposer **702** also decomposes the encoded image data string **510** into the individual encoded image blocks **516** (FIG. **5**B) that are forwarded to the one or more block decoders **706**.

The header converter **704** converts the modified header **512** into an output header. Simultaneously, the encoded image blocks **516** are decompressed or decoded by the one or more block decoders **706**. Each encoded image block **516** may be processed sequentially in one block decoder **706**, or multiple encoded image blocks **514** may be processed in parallel with one block decoder **706** for each encoded image block **516**. Thus, multiple block decoders **706** allow for parallel processing that increases the processing performance and efficiency of the image decoder engine **204** (FIG. **2**).

The image composer **708** receives each decoded image blocks from the one or more block decoders **706** and orders the decoded image block in a file. Further, the image composer **708** receives the converted header from the header converter **704**. The converted header and the decoded image blocks are placed together to generate output data representing the original image **310**.

FIG. **7**B is a block diagram of an exemplary embodiment of a block decoder **706**. Each block decoder **706** includes a block type detector **710**, one or more decoder units **712**, and an output selector **714**. The block type detector **710** is coupled to the encoded image decomposer **702** (FIG. **7**A), the output selector **714**, and each of the one or more decoder units **712**.

The block type detector **710** receives the encoded image blocks **514** and determines the block type for each encoded image block **516** (FIG. **5**B). The block type is detected based on the codewords **520** (FIG. **5**C). After the block type is determined, the encoded image blocks **514** are passed to each of the decoder units **712**, which decompress or decode each encoded image block **516** to generate colors for each particular encoded image block **516**. The decoder units **712** may be c-channels wide (e.g., one channel for each color component or pixel property being encoded), where c is any integer value. Using the selector signal, the block type detector **710** enables the output selector **714** to output the color of each encoded image block **516** from one of the decoder units **712** that corresponds with the block type detected by the block type detector **710**. Specifically, the block type detector **710** passes a selector signal to the output selector **714** that is used to select an output corresponding to the block type detected. Alternatively, using the selector signal, the appropriate decoder unit **712** could be selected so that the encoded block is only processed through the selected decoder unit.

FIG. **7**C is a block diagram of an alternative embodiment of a block decoder **706**. In this embodiment, the block decoder **706** includes a block type detector **720**, a first decoder unit **722**, a second decoder unit **724**, and an output selector **726**. The block type detector **720** is coupled to receive each encoded image block **516** (FIG. **5**B), and determine by comparing the codewords **520** (FIG. **5**C) of the encoded image block, the block type for each encoded image block **516**. For example, the block type may be four quantized colors or three quanitized colors and a transparency. Once the block type is selected and a selector signal is forwarded to the output selector **726**, the encoded image blocks **516** are decoded by the first and second decoder units **722** and **724**, respectively, to produce the pixel colors of each image block. The output selector **726** is enabled by the block type detector **720** to output the colors from the first and second decoder units **722** and **724** that correspond to the block type selected.

FIG. **7**D is a logic diagram illustrating an exemplary embodiment of a decoder unit similar to the decoder units **722** and **724** of FIG. **7**C. For simplicity, the functionality of each of the first and second decoder units **722** and **724** is merged into the single logic diagram of FIG. **7**D. Those skilled in the art will recognize that although the diagram is described with respect to a red-channel of the decoder units, the remaining channels (i.e., the green-channel and the blue-channel) are similarly coupled and functionally equivalent.

The logic diagram illustrating the first and second decoder units **722** and **724** is shown including portions of the block type detector **710**, **720** (FIGS. **7**B and **7**C, respectively) such as a comparator unit **730**. The comparator unit **730** is coupled to and works with a first 2×1 multiplexer **732***a* and a second 2×1 multiplexer **732***b*. Both 2×1 multiplexers **732***a* and **732***b* are coupled to a 4×1 multiplexer **734** that serves to select an appropriate color to output. The 4×1 multiplexer **734** is coupled to receive a transparency indicator signal that indicates whether or not a transparency (e.g., no color) is being sent. The 4×1 multiplexer **734** selects a color for output based on the value of the color index, referenced as the ID signal, that references the associated quantized color for an individual pixel of the encoded image block **514** (FIG. **5**B).

A red-channel **736** of the first decoder unit **722** includes a first and a second red-channel line **738***a* and **738***b* and a first and a second red-color block **740***a* and **740***b*. Along the path of each red-color block **740***a* and **740***b* is a first full adder **742***a* and **742***b*, a second full adder **744***a* and **744***b*, and carry-look ahead (CLA) adders **746***a* and **746***b*. The second decoder unit **724** contains similar components as the first decoder unit **722**.

The CLA adder **746***a* of the first red-color block **740***a* path of the first decoder unit **722** is coupled to the first 2×1 multiplexer **732***a*, while the CLA adder **746***b* of the second red-color block **740***b* path of the first decoder unit **722** is coupled to the second 2×1 multiplexer **732***b*. Further, adder **748** of the second decoder unit **724** is coupled to both the first and the second 2×1 multiplexers **732***a* and **732***b*.

FIG. **8**A is a flowchart **800** illustrating an operation of the decoder engine **204** (FIG. **2**) in accordance with an exemplary embodiment of the present invention. For purposes of illustration, the process for the decoder engine **204** will be described with a single block decoder **706** (FIG. **7**A) having two decoder units **722** and **724** as described earlier in connection with FIG. **7**C. Those skilled in the art will recognize that the process is functionally equivalent for decoder systems having more than one block decoder **706** and more than two decoder units **712**, as discussed in connection with FIG. **7**B.

In block **802**, the encoded image decomposer **702** (FIG. **7**A) receives the encoded or compressed image data **510** (FIG. **5**B) from the image encoder engine **202** (FIG. **2**), through the memory **104** (FIG. **1**) or the storage device **106** (FIG. **1**). Next, the encoded image decomposer **702** decomposes the encoded image data **510** by forwarding the modified header **512** (FIG. **5**B) to the header converter **704** (FIG. **7**A) in block **804**.

Subsequently in block **806**, the header converter **704** converts the header information to generate an output header that is forwarded to the image composer **708** (FIG. **7**A). Simultaneously, the one or more block decoders **706** (FIG. **7**A) decode pixel colors for each encoded image block **516** (FIG. **5**B) in block **808**. Each encoded image block **516** may be decoded sequentially in one block decoder **706** or multiple encoded image blocks **514** (FIG. **5**B) may be decoded in parallel in multiple block decoders **706** in block **808**. The process for decoding each encoded image block **516** is further described in connection with FIG. **8**B. Each decoded image block is then composed into a data file with the converted header information by the image composer **708** in block **810**. The image composer **708** then generates the data file as an output that represents the original image **310** (FIGS. **3**A and **3**B).

FIG. **8**B is a flowchart **820** illustrating an operation of the block decoder **706** (FIG. **7**A) in accordance with an exemplary embodiment of the present invention. Initially, each encoded image block **516** (FIG. **5**B) is received by the block decoder **706** in block **822**. Specifically, for one embodiment, the first and the second codewords **520** (e.g., CW_0 and CW_1 of FIG. **5**C) are received by the block type detector **710**, **720** (FIGS. **7**B and **7**C, respectively) of the block decoder **706**. As discussed above, comparing the numerical values of CW_0 and CW_1 reveals the block type. The first five bits of each codeword **520** that represent the red-channel color are received by the red-channel of each of the first and second decoder units **722** and **724** (FIG. **7**C). Furthermore, the second 6-bits of each codeword **520** that represent the green-channel color are received by the green-channel of each of the first and the second decoder units **722** and **724**, while the last 5-bits of each codeword **520** that represent the blue-channel color are received by the blue-channel of each of the first and second decoder units **722** and **724**.

Next in block **824**, the block type detector **710** detects the block type for an encoded image block **514**. Specifically, the comparator **730** (FIG. **7**D) compares the first and the second codewords **520** (e.g., CW_0 and CW_1) and generates a flag signal to enable the first 2×1 multiplexer **732***a* or the second 2×1 multiplexer **732***b*. In block **826**, either the first decoder unit **722** or the second decoder unit **724** is selected.

Subsequently quantized color levels for the decoder units **722** and **724** are calculated in block **828**. The calculation of the quantized color levels will now be discussed in more detail. Initially, the first decoder unit **722** calculates the four colors associated with the two codewords **520** (e.g., CW_0 and CW_1) using the following exemplary relationship:

$$CW_0 = \text{first codeword} = \text{first color};$$

$$CW_1 = \text{second codeword} = \text{second color};$$

$$CW_2 = \text{third color} = \frac{2}{3}CW_0 + \frac{1}{3}CW_1; \text{ and}$$

$$CW_3 = \text{fourth color} = \frac{1}{3}CW_0 + \frac{2}{3}CW_1.$$

In one embodiment, the first decoder unit **722** may estimate the above equations for CW_2 and CW_3 as follows:

$$CW_2 = \frac{5}{8}CW_0 + \frac{3}{8}CW_1; \text{ and}$$

$$CW_3 = \frac{3}{8}CW_0 + \frac{5}{8}CW_1.$$

The red-color blocks **740***a* and **740***b* (FIG. **7**D) serve as one-bit shift registers to obtain

$$\frac{1}{2}CW_0 \text{ or } \frac{1}{2}CW_1.$$

Further, each full adder **742***a*, **742***b*, **744***a*, and **744***b* (FIG. **7**D) also serves to shift the signal left by 1-bit. Thus, the signal from the first full adders **742***a* and **742***b* is

$$\frac{1}{4}CW_0 \text{ or } \frac{1}{4}CW_1,$$

respectively, because of a 2-bit overall shift, while the signal from the second full adders **744***a* and **744***b* is

$$\frac{1}{8}CW_0 \text{ or } \frac{1}{8}CW_1,$$

respectively due to a 3-bit overall shift. These values allow for the above approximations for the color signals.

The second decoder unit **724** (FIG. **7**C) calculates three colors associated with the codewords **520** (e.g., CW_0 and CW_1), and includes a fourth signal that indicates a transparency is being passed. The second decoder unit **724** calculates colors using the following exemplary relationship:

$$CW_0 = \text{first codeword} = \text{first color;}$$

$$CW_1 = \text{second codeword} = \text{second color;}$$

$$CW_3 = \text{third color} = \frac{1}{2}CW_0 + \frac{1}{2}CW_1\text{; and}$$

$$T = \text{Transparency.}$$

In one embodiment, the second decoder unit **724** has no approximation because the signals received from the red-color blocks **740***a* and **740***b* are shifted left by 1-bit so that the color is already calculated to

$$\frac{1}{2}CW_0 \text{ and } \frac{1}{2}CW_1,$$

respectively.

After the quantized color levels for the decoder units **722** and **724** selected in block **826** have been calculated in block **828**, each bitmap value for each pixel is read from the encoded image data block **510** (FIG. **5**A) in block **830**. As each index is read, it is mapped in block **832** to one of the four calculated colors if the first decoder unit **722** is selected. Alternatively, one of the three colors and transparency is mapped in block **832** if the second decoder unit **724** is selected. The mapped colors are selected by the 4×1 multiplexer **734** based on the value of the ID signal from the bitmap **522** (FIG. **5**C) of the encoded image block **514**. As stated previously, a similar process occurs for selection of colors in the green-channel and the blue-channel.

As the color data are output from the red-channel, green-channel and blue-channel, the output are received by the image composer **708** (FIG. **7**A). Subsequently, the image composer **708** arranges the output from the block encoders **706** in the same order as the original image **310** was decomposed. The resulting image is the original image **310**, which is then forwarded to an output unit **208** (FIG. **2**; e.g., a computer screen), which displays the image.

This exemplary embodiment beneficially allows for random access to any desired image block **320** (FIG. **3**C) within an image, and any pixel **322** (FIG. **3**C) within an image block **320**. FIG. **9**A is a block diagram of a subsystem **900** that provides random access to a pixel **322** or an image block **320** in accordance with one embodiment of the present invention.

The random access subsystem **900** includes a block address computation module **902**, a block fetching module

904, and one or more block decoders **706** coupled to the block address computation module **902** and the block fetching module **904**. The block address computation module **902** receives the header information **512** (FIG. **5**B) of the encoded image data string **510** (FIG. **5**B), while the block-fetching module **904** receives the encoded image block portion **514** (FIG. **5**B) of the encoded image data string **510**.

FIG. **9**B is a flowchart **910** of a process for random access to a pixel **322** (FIG. **3**C) or an image block **320** (FIG. **3**C) using the random access subsystem **900** of FIG. **9**A. When particular pixels **322** have been identified for decoding, the image decoder engine **204** (FIG. **2**) receives the encoded image data string **510** (FIG. **5**B). The modified header **512** (FIG. **5**B) of the encoded image data string **510** is forwarded to the block address computation module **902** (FIG. **9**A), and the encoded image block portion **514** (FIG. **5**B) of the encoded image data string **510** is forwarded to the block-fetching module **904** (FIG. **9**A).

In block **912**, the block address computation module **902** reads the modified header **512** to compute an address of the encoded image block portion **514** having the desired pixels **322**. The address computed is dependent upon the pixel coordinates within an image. Using the computed address, the block-fetching module **904** identifies each encoded image block **516** (FIG. **5**B) of the encoded image block portion **514** that contains the desired pixels **322** in block **914**. Once each encoded image block **516** having the desired pixels **322** has been identified, only the identified encoded image block **516** is forwarded to the block decoders **706** (FIG. **9**A) for processing.

FIG. **9**B is similar to the process described above in FIG. **8**B, wherein the block decoders **706** compute quantized color levels for each identified encoded image blocks **516** having the desired pixels in block **916**. After the quantized color levels have been computed, the color of the desired pixel is selected in block **918** and output from the image decoder engine **204**.

Random access to pixels **322** of an image block **320** (FIG. **3**C) advantageously allows for selective decoding of only needed portions or sections of an image. Random access also allows the image to be decoded in any order the data is required. For example, in three-dimensional texture mapping only portions of the texture may be required and these portions will generally be required in some non-sequential order. Thus, this embodiment of the present invention increases processing efficiency and performance when processing only a portion or section of an image. Further, the present invention beneficially encodes or compresses the size of an original image **310** (FIGS. **3**A and **3**B) from 24-bits per pixel to an aggregate 4-bits per pixel, and then decodes or decompresses the encoded image data string **510** (FIG. **5**B) to get a representation of the original image **310**. Additionally, the exemplary embodiment uses two base points or codewords from which additional colors are derived so that extra bits are not necessary to identify a pixel **322** color.

Moreover, the exemplary embodiment advantageously accomplishes the data compression on an individual block basis with the same number of bits per block so that the compression rate can remain fixed. Further, because the blocks are of fixed size with a fixed number of pixels **322**, random access to any particular pixel **322** in the block is allowed. Additionally, an efficient use of system resources is provided because entire blocks of data are not retrieved and decoded to display data corresponding to only a few pixels **322**.

Finally, the use of fixed-rate 64-bit data blocks provides the advantage of having simplified header information that allows for faster processing of individual data blocks. A 64-bit data block allows for faster processing as the need to wait until a full data string is assembled is eliminated. Further, an imaging system in accordance with the present invention may also reduce the microchip space necessary for a decoder system because the decoder system only needs to decode each pixel **322** to a set of colors determined by, for example, the two codewords **520** (FIG. **5**C).

The present invention has been described above with reference to specific embodiments. It will be apparent to those skilled in the art that various modifications may be made and other embodiments can be used without departing from the broader scope of the invention. These and other variations of the specific embodiments are intended to be covered by the present invention.

What is claimed is:

1. An image decoder engine for decoding an encoded image data file, comprising:

an encoded image decomposer for decomposing the encoded image data file into a modified header and at least one compressed image block, each image block having at least one associated codeword and a plurality of image elements associated with an index value; and

at least one block decoder coupled to the encoded image decomposer for decompressing the at least one compressed image block into at least one decompressed image block by generating a set of quantized image data values and mapping the index value to a quantized image data value from the set of quantized image data values, the at least one block decoder further comprising,

at least one decoder configured for decompressing each of the at least one compressed image block to generate colors for each of the at least one compressed image block.

2. The image decoder engine of claim **1** further comprising an image composer configured for ordering the at least one decompressed image blocks in an output data file.

3. The image decoder engine of claim **1** wherein the set of quantized image data values comprise the at least one associated codeword and at least one image data value derived from the at least one associated codeword.

4. The image decoder engine of claim **1** further comprising a header converter coupled to the encoded image decomposer and configured for converting the modified header into an output header.

5. The image decoder engine of claim **4** wherein the image composer orders the at least one decompressed image block and the output header into a data file.

6. The image decoder engine of claim **1** wherein the at least one block decoder further comprises a block type detector configured for determining a block type for each of the at least one compressed image block based on the at least one associated codeword.

7. The image decoder engine of claim **1** wherein the decoder is configured to decompress each of the at least one compressed image block based on a block type.

8. The image decoder engine of claim **1** wherein the at least one block decoder further comprises an output selector for outputting the at least one decompressed image block.

* * * * *

Exhibit E

UNITED STATES INTERNATIONAL TRADE COMMISSION
WASHINGTON, D.C.

In the Matter of

CERTAIN ELECTRONIC DEVICES WITH IMAGE PROCESSING SYSTEMS, COMPONENTS THEREOF, AND ASSOCIATED SOFTWARE

Investigation No. 337-TA-____

COMPLAINT OF S3 GRAPHICS CO., LTD. AND S3 GRAPHICS, INC. UNDER SECTION 337 OF THE TARIFF ACT OF 1930, AS AMENDED

<u>COMPLAINANT</u>

S3 Graphics Co., Ltd.
2nd Fl., Zephyr House
Mary St., P.O. Box 709
Grand Cayman
Grand Cayman Islands
British West Indies
Telephone: (510) 683-3300

S3 Graphics, Inc.
1025 Mission Court
Fremont, CA 94539
Telephone: (510) 683-3300

Counsel for Complainant:

Thomas L. Jarvis
Thomas W. Winland
John R. Alison
Paul C. Goulet
John M. Williamson
FINNEGAN, HENDERSON, FARABOW, GARRETT & DUNNER, LLP
901 New York Avenue, N.W.
Washington, D.C. 20001-4413
Telephone: (202) 408-4000
Facsimile: (202) 408-4400

<u>PROPOSED RESPONDENT</u>

Apple Inc., a/k/a Apple Computer, Inc.
1 Infinite Loop
Cupertino, CA 95014
Telephone: (408) 996-1010

TABLE OF CONTENTS

TABLE OF EXHIBITS

Exh.	Document
Exh. 1	Certified copy of U.S. Patent No. 7,043,087
Exh. 2	Certified copy of U.S. Patent No. 6,775,417
Exh. 3	Certified copy of U.S. Patent No. 6,683,978
Exh. 4	Copy of U.S. Patent No. 6,658,146
Exh. 5	Certified copies of recorded assignments for the Asserted Patents
Exh. 6	Apple Inc., Form 10-K for the Fiscal Year Ended September 26, 2009
Exh. 7	Foreign patents and patent applications related to the Asserted Patents
Exh. 8	Infringement claim charts for U.S. Patent No. 7,043,087
Exh. 9	Infringement claim charts for U.S. Patent No. 6,775,417
Exh. 10	Infringement claim charts for U.S. Patent No. 6,683,978
Exh. 11	Infringement claim charts for U.S. Patent No. 6,658,146
Exh. 12	Documents detailing purchase of Apple iPhone 3GS product in the United States, including photographs
Exh. 13	Documents detailing purchase of Apple iPad product in the United States, including photographs
Exh. 14	Documents detailing purchase of Apple iPod Touch product in the United States, including photographs
Exh. 15	Documents detailing purchase of Apple MacBook Pro product in the United States, including photographs
Exh. 16	Documents detailing purchase of three Apple iPhone software applications in the United States, including screenshots
Exh. 17	Photographs of S3G's Chrome 440 GTX and Chrome 430 ULP Graphics Chips
Exh. 18(A)	Apple document entitled, "OpenGL ES Programming Guide for iPhone OS"
Exh. 18(B)	Apple document entitled, "iPad Programming Guide"
Exh. 18(C)	Apple document entitled, "GLES2 Sample"
Exh. 18(D)	Imagination document entitled, "PowerVR MBX Technology Overview"
Exh. 18(E)	Imagination document entitled, "PowerVR SGX Open ES 2.0 Application Development Recommendation"
Exh. 18(F)	Imagination document entitled, "PowerVR 3D Application Development Recommendation"

Exh.	Document
Exh. 18(G)	Fenney, "Texture Compression using Low-Frequency Signal Modulation"
Exh. 18(H)	3D graphics decompression source code, "PVRTDecompress.cpp"
Exh. 18(I)	Apple document entitled, "iPhone Development Guide"
Exh. 18(J)	Apple webpage containing technical specifications of MacBook models
Exh. 18(K)	Technical specifications of iMac
Exh. 18(L)	Technical specifications of Mac Pro
Exh. 18(M)	Technical specifications of Mac mini
Exh. 18(N)	Apple document entitled, "Image I/O Programming Guide"
Exh. 18(O)	The TIFF standard, revision 6.0
Exh. 18(P)	Apple webpage entitled "iPhone OS Overview"

Confidential Exhs.	Document
Exh. 19C	Identification of Licensees
Exh. 20C	S3G Domestic Industry claim chart for U.S. Patent No. 7,043,087
Exh. 21C	S3G Domestic Industry claim chart for U.S. Patent No. 6,775,417
Exh. 22C	S3G Domestic Industry claim chart for U.S. Patent No. 6,683,978
Exh. 23C	S3G Domestic Industry claim chart for U.S. Patent No. 6,658,146
Exh. 24C	Exemplary S3G licensee Domestic Industry claim chart for U.S. Patent No. 7,043,087
Exh. 25C	Exemplary S3G licensee Domestic Industry claim chart for U.S. Patent No. 6,775,417
Exh. 26C	Exemplary S3G licensee Domestic Industry claim chart for U.S. Patent No. 6,683,978
Exh. 27C	Exemplary S3G licensee Domestic Industry claim chart for U.S. Patent No. 6,658,146
Exh. 28C	S3G investments in engineering, research, and development
Exh. 29C	S3G investments in labor and capital
Exh. 30C	S3G investments in plant and equipment

Exh. 31C	S3G investment in licensing
Exh. 32C	Domestic Industry Investments of licensees
Exh. 32C (A)	Exemplary S3G licensee's Form 10-K for the Fiscal Year Ended June 30, 2009
Exh. 33C (A)	Developer documentation related to compression technology
Exh. 33C (B)	Developer documentation related to compression technology
Exh. 33C (C)	Developer documentation related to compression technology
Exh. 33C (D)	Developer documentation related to compression technology
Exh. 33C (E)	Developer documentation related to compression technology
Exh. 33C (F)	Developer documentation related to compression technology
Exh. 33C (G)	Developer documentation related to compression technology
Exh. 33C (H)	Developer documentation related to compression technology
Exh. 33C (I)	Confidential S3 architecture specification
Exh. 33C (J)	Confidential S3 source code
Exh. 33C (K)	Confidential S3 source code
Exh. 33C (L)	Confidential S3 product overview
Exh. 33C (M)	Confidential S3 source code
Exh. 34C	Photograph of Confidential Physical Exhibit 7C and screenshot of exemplary S3G license's software development kit ("SDK") and sample texture file

Physical Exhs.

Physical Exh. 1	Apple iPhone (in box with packaging)
Physical Exh. 2	Apple iPad (in box with packaging)
Physical Exh. 3	Apple iPod Touch (in box with packaging)
Physical Exh. 4	Apple MacBook Pro (in box with packaging)
Physical Exh. 5	Domestic Industry Exhibit (S3G Chrome)

Physical Exhs.	
Physical Exh. 6	Compact disc containing three software applications for the Apple iPhone
Confidential Physical Exhs.	
Physical Exh. 7C	Compact disc containing exemplary S3G licensee's Software Development Kit and texture file

APPENDICES

Appendix	Document
App. A	Certified copy of the prosecution history of U.S. Patent No. 7,043,087 and three copies thereof
App. B	Certified copy of the prosecution history of U.S. Patent No. 6,775,417 and three copies thereof
App. C	Certified copy of the prosecution history of U.S. Patent No. 6,683,978 and three copies thereof
App. D	Copy of the prosecution history of U.S. Patent No. 6,658,146 and three copies thereof
App. E	Four copies of each technical reference identified in the prosecution history of U.S. Patent No. 7,043,087
App. F	Four copies of each technical reference identified in the prosecution history of U.S. Patent No. 6,775,417
App. G	Four copies of each technical reference identified in the prosecution history of U.S. Patent No. 6,683,978
App. H	Four copies of each technical reference identified in the prosecution history of U.S. Patent No. 6,658,146

Confidential Appendix	Document
Conf. App. I	Three copies of each known current license involving the Asserted Patents

I. INTRODUCTION

1. This Complaint is filed by S3 Graphics, Inc. and S3 Graphics Co, Ltd. (collectively, "S3G") under Section 337 of the Tariff Act of 1930, as amended, 19 U.S.C. § 1337, based on the unlawful importation into the United States, the sale for importation, and the sale within the United States after importation, by proposed Respondent Apple Inc., ("Apple") of certain electronic devices with image processing systems, and components thereof, and associated software that infringe one or more of claims 1, 6, or 7 of United States Patent No. 7,043,087 ("the '087 patent"); one or more of claims 1, 7, 8, 12, 13, 15 or 23 of United States Patent No. 6,775,417 ("the '417 patent"); one or more of claims 11, 14, or 16 of United States Patent No. 6,683,978 ("the '978 patent"); and one or more of claims 2, 4, 8, 13, 16, 18, or 19 of United States Patent No. 6,658,146 ("the '146 patent") (collectively, the "Asserted Claims" of the "Asserted Patents").

2. Certified copies of the Asserted Patents are attached as Exhibit Nos. 1 through 4, respectively.[1] S3 Graphics Co., Ltd., owns all right, title, and interest in each of the Asserted Patents. S3 Graphics, Inc., a wholly owned subsidiary of S3 Graphics Co. Ltd., holds a nonexclusive license, with a right to grant sublicenses, to the Asserted Patents. (Confidential Appendix I). Certified copies of recorded assignments demonstrating the chain of title of the Asserted Patents are attached as Exhibit No. 5.

3. The proposed respondent is Apple Inc. The Accused Products are certain electronic devices with image processing systems, components thereof, and associated software including, but not limited to multimedia devices, smart phones, personal computers, and software for use

[1] S3G has not yet obtained a certified copy of the '146 patent. Exhibit No. 4, therefore, is not a certified copy of the '146 patent. S3G will supplement Exhibit No. 4 with a certified copy of the '146 patent upon receipt.

with such devices (collectively the "Accused Products"). Examples of the Accused Products are the Apple iPod Touch, iPhone, iPad, Apple computers such as the MacBook used in conjunction with an Apple software development kit ("SDK"), and other application software. The Accused Products are imported into the United States and sold after importation into the United States by Apple.

4. An industry as required by 19 U.S.C. § 1337(a)(2) and (3) exists in the United States relating to the technology protected by the Asserted Patents.

5. As set forth more fully in paragraph 119, S3G seeks as relief, a permanent exclusion order barring from entry into the United States all infringing Apple electronic devices with image processing systems, components thereof, and associated software sold for importation into the United States, imported, or sold after importation. S3G also seeks, as relief, a cease and desist order prohibiting Apple's sale for importation into the United States, importation, sale after importation into the United States, offer for sale, solicitation of sales, advertising, testing, technical support and other commercial activity related to Apple electronic devices with image processing systems, components thereof, and/or associated software that infringe one or more Asserted Claims of the Asserted Patents.

II. COMPLAINANTS

6. S3 Graphics, Inc., is a Delaware corporation with its principal place of business at 1025 Mission Court, Fremont, CA 94539. S3 Graphics Co., Ltd. is a Cayman Islands corporation with its principal place of business at 2nd Fl., Zephyr House, Mary St., P.O. Box 709, Grand Cayman, Grand Cayman Islands, British West Indies. S3 Graphics Co., Ltd. holds all right, title, and interest in the Asserted Patents. (Exhibit No. 5). S3G provides innovative graphics visualization technologies and GPU (graphics processing unit) products for mobile devices, desktop computers, and embedded systems.

7. S3G's image processing technologies enable coding of image attributes into data files that can be more efficiently stored and later displayed. Many software developers use S3G image processing technology to convert very large color image data files, particularly animated (motion) images, into compressed data files that can be efficiently distributed to and displayed by end users of the software. For example, video games typically implement life-like animation by the rapid display of a sequence of progressively modified still images to achieve the illusion of movement. Game developers can use S3G's image processing technology to encode the image data into compressed formats that are convenient for distribution and can be decoded and displayed by consumers. S3G's image processing technology is licensed by some of the largest computer hardware and software companies in the world.

8. S3G engages in research, development, engineering, and product design activities at S3 Graphics, Inc.'s principal place of business in Fremont, California including research, development, and product design for products utilizing S3G image compression technology, including the S3G Chrome series graphics products.

9. S3G operates a licensing business from S3 Graphics, Inc.'s principal place of business in Fremont, California that includes formulation of licensing strategies, identification of products and companies that currently do, or prospectively could, utilize S3G image processing technology, analyzing those products and companies for potential licensing opportunities, negotiating licenses under the S3G patent portfolio, and monitoring and enforcing compliance with those licenses and S3G patent rights.

10. On information and belief, S3G's licensees conduct in the United States certain research, development, engineering, manufacturing, and technical support of products with S3G image processing technology.

III. PROPOSED RESPONDENT

11. On information and belief, respondent Apple Inc. is a corporation organized under the laws of the State of California with its principal place of business at 1 Infinite Loop, Cupertino, CA 95014. (Exhibit No. 6).

12. On information and belief, Apple is involved in the design, development, manufacture, sale for importation, importation, and sale after importation of the Accused Products. Further, on information and belief, Apple performs several services to support the importation and sale of Accused Products into and within the United States, including marketing of the Accused Products, repair of the Accused Products, technical support, and other after-sale services, such as supporting and configuring the Accused Products, as well as interfacing with U.S.-based customers and distributors to conform the Accused Products to purchaser requests.

IV. THE TECHNOLOGY AND PRODUCTS AT ISSUE

13. The technologies at issue relate generally to apparatuses, methods, and data formats for encoding and decoding, including compressing image data, storing of compressed image data, and decompressing of such data. The Asserted Patents generally relate to aspects of an image processing system for encoding and decoding, including compressing image data files into a more compact form, a format for storing that compressed data, and a system for decompressing that data for display as an image.

14. The Asserted Patents disclose an image compression technology including an image decomposer, an encoder for computing image data values and generating codeword reference values, and a construction module for creating indices that map each image data value to a set of colors generated from the codewords. The resulting codewords and indices form an encoded image block.

15. The Asserted Patents also disclose a format for storage of encoding or compressing image data that includes a portion for storage of multiple codewords from which a set of colors can be computed and a portion for storage of indices for mapping pixel color to a computed color.

16. The Asserted Patents also disclose an image data decoding or decompressing technology that includes a decomposer for processing the encoded image data stream into a header and a plurality of encoded image bocks, a header converter for generating an output image header, one or more block data decoders for generating from the codewords and indices pixel image attributes such as color and for mapping those attributes to pixels, and an image composer that reassembles data blocks for a display device and/or a data file.

17. On information and belief, Apple provides an SDK specifically adapted for use with Apple computers to compress and decompress image data files using the technology disclosed and claimed in the Asserted Patents.

18. On information and belief, Apple's SDK and computers generate encoded image files in the format disclosed and claimed in the Asserted Patents.

19. On information and belief, Apple sells a variety of imported products, including the Apple iPod Touch, iPhone, iPad, Apple computers such as the MacBook, certain applications for those products, and associated software that incorporate the image data compression, decompression, and/or data format disclosed and claimed in the Asserted Patents.

20. The identification of a specific model, trade name, or type of electronic device with image processing systems and/or the identification of specific software or components is not intended to limit the scope of this Investigation. The remedy sought in this Complaint should

extend to all infringing electronic devices with image processing systems, components thereof, and associated software.

V. THE ASSERTED PATENTS AND NON-TECHNICAL DESCRIPTION OF THE INVENTIONS

A. Four Patents from a Single Original Application

21. On October 2, 1997, S3 Incorporated (a predecessor company to S3G) filed United States Patent Application Serial Number. 08/942,860 ("U.S. Pat. App. Ser. No. 08/942,860"). From that single original application, through continuation and continuation-in-part applications, all four of the patents at issue in this investigation were issued.

B. U.S. Patent No. 7,043,087

1. Identification and Ownership of the '087 Patent

22. United States Patent No. 7,043,087, entitled "Image Processing System," issued on May 9, 2006, to inventors Zhou Hong, Konstantine I. Iourcha, and Krishna S. Nayak. (Exhibit No. 1). The '087 patent issued from Application No. 10/893,084, filed on July 16, 2004, that claims priority from the original U.S. Pat. App. Ser. No. 08/942,860. *Id.*

23. The '087 patent has 1 independent claim and 7 dependent claims. S3G is asserting claims 1, 6, and 7 of the '087 patent in this Investigation.

24. The Asserted Claims of the '087 patent are valid, enforceable, and currently in full force and effect until its expiration on October 2, 2017.

25. S3 Graphics Co., Ltd., owns by assignment the entire right, title, and interest in and to the '087 patent. (Exhibit No. 5).

26. Pursuant to Commission Rule 210.12(c), this Complaint is accompanied by a certified copy of the prosecution history of the '087 patent and three copies thereof.

(Appendix A). Further, this Complaint is accompanied by four copies of each technical reference identified in the prosecution history of the '087 patent (Appendix E).

2. Non-Technical Description of the Invention Claimed in the '087 Patent

27. The '087 patent discloses aspects of an image processing system for encoding and decoding image data, including compressing image data files into a more compact form, a format for storing that compressed data, and a system for decoding and/or decompressing that data for display as an image and/or for storage. Asserted Claims 1, 6, and 7 of the '087 patent are directed to aspects of an engine for decoding image data files. A nontechnical description of that decoding engine is that it includes: (a) a decomposer for converting encoded image data files into a modified header and at least one encoded or compressed image block, where each image block is associated with at least one codeword and index values for a plurality of pixels; (b) at least one block decoder for decoding or decompressing image blocks by generating a set of quantized image data values and mapping the index value to one of the quantized image data values from the set of quantized image data values; and (c) at least one decoder configured for decoding or decompressing each of the image blocks. This nontechnical description does not limit or interpret the claims of the '087 patent.

3. Foreign Counterparts

28. The foreign patents and patent applications reported as related to the '087 patent are identified in Exhibit No. 7. On information and belief, no other foreign applications or patents corresponding to the '087 patent have been filed, abandoned, or rejected.

4. Licenses

29. As required under Commission Rule 210.12(a)(9)(iii), a list of licensed entities is attached to this Complaint as Confidential Exhibit No. 19C. On information and belief, there are no other current licenses involving the '087 patent.

C. U.S. Patent No. 6,775,417

1. Identification and Ownership of the '417 Patent

30. United States Patent No. 6,775,417 (the "'417 patent"), entitled "Fixed-Rate Block-Based Image Compression with Inferred Pixel Values," issued on August 10, 2004, to inventors Zhou Hong, Konstantine I. Iourcha, and Krishna S. Nayak. (Exhibit No. 2). The '417 patent issued from Application No. 10/052,613, filed on January 17, 2002, that claims priority from the original U.S. Pat. App. Ser. No. 08/942,860. *Id.*

31. The '417 patent has 8 independent claims and 22 dependent claims. S3G is asserting claims 1, 7, 8, 12, 13, 15 and 23 of the '417 patent in this Investigation.

32. The Asserted Claims of the '417 patent are valid, enforceable, and currently in full force and effect until its expiration on March 16, 2018.

33. S3 Graphics Co., Ltd., owns by assignment the entire right, title, and interest in and to the '417 patent. (Exhibit No.5).

34. Pursuant to Commission Rule 210.12(c), this Complaint is accompanied by a certified copy of the prosecution history of the '417 patent and three copies thereof. (Appendix B). Further, this Complaint is accompanied by four copies of each technical reference identified in the prosecution history of the '417 patent (Appendix F).

2. Non-Technical Description of the Invention of the '417 Patent

35. The '417 patent discloses aspects of an image processing system for encoding and decoding image data, including compressing image data files into a more compact form, a format

for storing that encoded or compressed data, and a system for decoding and/or decompressing that data for display as an image and/or for storage.

36. Asserted Claims 1, 7, and 15 of the '417 patent is directed to aspects of an apparatus and method for encoding image data files. A nontechnical description of that encoding engine is that it includes: (a) a decomposer for decomposing image data files into a header and at least one image block, where each image block has a set of image elements and each image element has an original image data value; (b) at least one block encoder for encoding or compressing each image block by associating each original image data value of the image element with an index to a derived image data value in a set of quantized image data values; and (c) an encoded image composer ordering the encoded image blocks into a data file. This nontechnical description does not limit or interpret the claims of the '417 patent.

37. Asserted Claims 8, 12, and 13 of the '417 patent are directed to aspects of an apparatus for decoding image data files. A nontechnical description of that apparatus is that it includes: (a) a decomposer for converting encoded image data files into a modified header and at least one encoded or compressed image block, where each image block is associated with at least one codeword and index values for a plurality of image elements and (b) at least one block decoder for decoding or decompressing image blocks by generating a set of quantized image data values and mapping the index value to one of the quantized image data values in the set of quantized image data values. This nontechnical description does not limit or interpret the claims of the '417 patent.

38. Asserted Claim 23 of the '417 patent is directed to aspects of a method for decoding image data files. A nontechnical description of that decoding method is that it includes: (a) decomposing an encoded image into a modified header and a plurality of encoded

image blocks having at least one codeword and a plurality of image elements associated with an index value; (b) generating a set of quantized image data values; and (c) mapping the index value for each image element to one of the quantized image data values. This nontechnical description does not limit or interpret the claims of the '417 patent.

3. Foreign Counterparts

39. The foreign patents and patent applications reported as related to the '417 patent are identified in Exhibit No.7. On information and belief, no other foreign applications or patents corresponding to the '417 patent have been filed, abandoned, or rejected.

4. Licenses

40. As required under Commission Rule 210.12(a)(9)(iii), a list of licensed entities is attached to this Complaint as Confidential Exhibit No. 19C. On information and belief, there are no other current licenses involving the '417 patent.

D. U.S. Patent No. 6,683,978

1. Identification and Ownership of the '978 Patent

41. United States Patent No. 6,683,978 (the "'978 patent"), entitled "Fixed-Rate Block Based Image Compression with Inferred Pixel Values," issued on January 17, 2004, to inventors Konstantine I. Iourcha, Krishna S. Nayak, and Zhou Hong. (Exhibit No. 3). The '978 patent issued from Application No. 09/442,114, filed on November 17, 1999, that claims priority from the original U.S. Pat. App. Ser. No. 08/942,860. *Id.*

42. The '978 patent has 5 independent claims and 24 dependent claims. S3G is asserting claims 11, 14, and 16 of the '978 patent in this Investigation.

43. The Asserted Claims of the '978 patent are valid, enforceable, and currently in full force and effect until its expiration on October 2, 2017.

44. S3 Graphics Co., Ltd., owns by assignment the entire right, title, and interest in and to the '978 patent. (Exhibit No. 5).

45. Pursuant to Commission Rule 210.12(c), this Complaint is accompanied by a certified copy of the prosecution history of the '978 patent and three copies thereof. (Appendix C). Further, this Complaint is accompanied by four copies of each technical reference identified in the prosecution history of the '978 patent (Appendix G).

2. Non-Technical Description of the Invention of the '978 Patent

46. The '978 patent discloses aspects of an image processing system for encoding and decoding image data files, including compressing image data files into a more compact form, a format for storing that compressed data, and a system for decoding and/or decompressing that data for display as an image and/or file storage. Asserted claims 11, 14, and 16 of the '978 patent are directed to aspects of a data format for representing an original color image. A nontechnical description of the data format is: (a) a codeword portion for storing at least one codeword and a bitmap portion for storing a set of indices and (b) the bitmap portion constructed by a bitmap construction module utilizing the codeword portion associated with the bitmap portion where (i) at least one codeword defines a set of colors that approximate the pixel color set and (ii) the indices map the pixel color set to at least one color in the set of colors. This nontechnical description does not limit or interpret the claims of the '978 patent.

3. Foreign Counterparts

47. The foreign patents and patent applications reported as related to the '978 patent are identified in Exhibit No. 7. On information and belief, no other foreign applications or patents corresponding to the '978 patent have been filed, abandoned, or rejected.

4. Licenses

48. As required under Commission Rule 210.12(a)(9)(iii), a list of licensed entities is attached to this Complaint as Confidential Exhibit No. 19C. On information and belief, there are no other current licenses involving the '978 patent.

E. U.S. Patent No. 6,658,146

1. Identification and Ownership of the '146 Patent

49. United States Patent No 6,658,146, entitled "Fixed-Rate Block-Based Image Compression with Inferred Pixel Values" issued December 2, 2003, to inventors Konstantine I. Iourcha, Krishna S. Nayak, and Zhou Hong. (Exhibit No. 4). The '146 patent issued from Application No. 09/351,930, filed on July 12, 1999, that claims priority from the original U.S. Pat App. Ser. No. 08/942,860. *Id.*

50. The '146 patent has 9 independent claims and 13 dependent claims. S3G is asserting claims 2, 4, 8, 13, 16, 18, and 19 of the '146 patent in this Investigation.

51. The Asserted Claims of the '146 patent are valid, enforceable, and currently in full force and effect until its expiration on October 2, 1017.

52. S3 Graphics Co., Ltd., owns by assignment the entire right, title, and interest in and to the '146 patent. (Exhibit No. 5).

53. Pursuant to Commission Rule 210.12(c), this Complaint is accompanied by a copy of the prosecution history of the '146 patent and three copies thereof.[2] (Appendix D). Further, this Complaint is accompanied by four copies of each technical reference identified in the prosecution history of the '146 patent (Appendix H).

---

[2] As of the time of filing, S3G has not yet obtained a certified copy of the prosecution history of the '146 patent. S3G will supplement Appendix D with a certified copy of the prosecution history of the '146 patent upon receipt.

2. Non-Technical Description of the Invention of the '146 Patent

54. The '146 Patent discloses aspects of an image processing system for encoding and decoding image data files, including compressing image data files into a more compact form, a format for storing that compressed data, and a system for decoding and/or decompressing that data for display as an image and/or file storage. Asserted Claims 2 and 4 of the '146 Patent are directed to aspects of an apparatus for encoding image data files. The apparatus includes: (a) a decomposer for breaking an image into one or more image blocks; (b) at least one block encoder for encoding or compressing each image block to generate an encoded image block; and (c) an encoded image composer for ordering the encoded image blocks into a data file. The block encoder also includes a color quantizer for generating codewords from which quantized colors are derived. The color quantizer further includes a block type module for selecting an identifiable block type of the image block.

55. Asserted Claims 8, 13, 16, 18, and 19 of the '146 patent are directed to aspects of a method for encoding image data files. A nontechnical description of that method is that it includes: (a) fitting a geometric element to the color points of the image block to the center of gravity of the color points of the image block while minimizing the moment of inertia; (b) computing a set of codewords; (c) computing a set of computed colors; (d) mapping each of the color points of the image block to one of the computed colors to generate an index; and (e) using the indices to represent the color points of the image bock. This nontechnical description does not limit or interpret the claims of the '146 Patent.

3. Foreign Counterparts

56. The foreign patents and patent applications reported as related to the '146 patent are identified in Exhibit No. 5. On information and belief, no other foreign applications or patents corresponding to the '146 patent have been filed, abandoned, or rejected.

4. Licenses

57. As required under Commission Rule 210.12(a)(9)(iii), a list of licensed entities is attached to this Complaint as Confidential Exhibit No. 19C. On information and belief, there are no other current licenses involving the '146 patent.

VI. UNLAWFUL AND UNFAIR ACTS OF RESPONDENTS—PATENT INFRINGEMENT

58. Apple has engaged in unfair trade practices, including the sale for importation, importation, and sale after importation of certain electronic devices with image processing systems, components thereof, and associated software that infringe the Asserted Claims of the Asserted Patents.

A. Infringement of the '087 Patent

59. On information and belief, the Accused Products infringe at least claims 1, 6, and 7 of the '087 patent. A chart that applies representative independent claim 1 of the '087 patent to the Accused Products is attached to this Complaint as Exhibit No. 8.

1. Direct Infringement of the '087 Patent

60. On information and belief, Apple sells for importation into the United States, imports, and/or sells after importation in the United States the Accused Products, including the Apple iPod Touch, iPhone, iPad, Apple computers such as the MacBook, the Apple SDK, and application software. S3G has purchased these devices and certain application software in the United States directly from Apple. (*See* Exhibit Nos. 12-16).

61. On information and belief, Apple tests or operates the Accused Products in the United States by using the Accused Products, including the Apple iPod Touch, iPhone, iPad, and Apple computers such as the MacBook in combination with associated software (e.g., encoded image data and the Apple SDK), thereby directly infringing claims 1, 6, and 7 of the '087 patent.

2. Contributory Infringement of the '087 Patent

62. Apple will have knowledge of the '087 patent and the infringing acts at least as early as its receipt of this Complaint.

63. On information and belief, Apple contributes to the infringement of claims 1, 6, and 7 of the '087 patent.

64. On information and belief, the graphics decoding components of the Apple iPod Touch, iPhone, and iPad are specially adapted for an infringing use of one or more of claims 1, 6, and 7 of the '087 patent; embody a material part of the inventions claimed in the '087 patent; and are not staple articles of commerce suitable for substantial non-infringing use. Apple sells for importation into the United States, imports, and/or sells after importation in the United States the Apple iPod Touch, iPhone, iPad, and the graphics decoding components of those products.

65. On information and belief, software application developers and consumers make and use the claimed inventions by using the Apple iPod Touch, iPhone, and iPad, in combination with associated software (e.g., encoded image data) thereby directly infringing claims 1, 6, and 7 of the '087 patent.

3. Inducement of Infringement of the '087 Patent

66. Apple will have knowledge of the '087 patent and the infringing acts at least as early as its receipt of this Complaint.

67. On information and belief, Apple induces others to infringe claims 1, 6, and 7 of the '087 patent by encouraging and facilitating others to perform actions known by Apple to infringe and with the intent that performance of the actions will infringe.

68. On information and belief, Apple encourages software application developers to make and use the claimed inventions by providing compressed image data, providing

instructions and support for developing applications, and providing a distribution channel for applications for the Apple iPod Touch, iPhone, and iPad that include the encoded image data.

69. On information and belief, Apple induces consumers to use the claimed inventions by providing the Apple iPod Touch, iPhone, and iPad, and by providing software applications for those products that include encoded image data files. Further, Apple actively encourages, promotes, distributes, provides instruction for, and supports the use of software applications for its iPod Touch, iPhone, and iPad products that include encoded image data files.

70. On information and belief, software application developers and consumers make and use the claimed inventions by using the Apple iPod Touch, iPhone, and iPad, in combination with associated software (e.g., encoded image data) thereby directly infringing claims 1, 6, and 7 of the '087 patent.

71. On information and belief, Apple induces software application developers to make and use the claimed inventions by providing the Apple SDK for use with Apple computers such as the MacBook. The Apple SDK works exclusively on Apple computers such as the MacBook, and the Apple SDK in combination with Apple computers such as the MacBook can be used to produce application software at issue. Apple actively encourages application software developers to use the Apple SDK with Apple computers such as the MacBook to develop application software by distributing the Apple SDK and providing instructions and support for its use with Apple computers such as the MacBook and by selling, promoting, distributing, and marketing applications made using the Apple SDK.

72. On information and belief, application software developers make and use the claimed inventions by using Apple computers such as the MacBook in combination with

associated Apple software (e.g., the Apple SDK) thereby directly infringing claims 1, 6, and 7 of the '087 patent.

B. Infringement of the '417 Patent

73. On information and belief, the Accused Products infringe at least claims 1, 7, 8, 12, 13, 15 and 23 of the '417 patent. Charts that apply representative independent claims 1, 8, 15 and 23 of the '417 patent to the Accused Products are attached to this Complaint as Exhibit No. 9.

1. Direct Infringement of the '417 Patent

74. On information and belief, Apple sells for importation into the United States, imports, and/or sells after importation in the United States the Accused Products, including the Apple iPod Touch, iPhone, iPad, Apple computers such as the MacBook, the Apple SDK, and application software. S3G has purchased these devices and certain application software in the United States directly from Apple. (Exhibit Nos. 12-16).

75. On information and belief, Apple tests or operates the Accused Products in the United States by performing the claimed methods and by using the Accused Products, including the Apple iPod Touch, iPhone, iPad, and Apple computers such as the MacBook in combination with associated software (e.g., encoded image data and the Apple SDK), thereby directly infringing claims 1, 7, 8, 12, 13, 15 and 23 of the '417 patent.

2. Contributory Infringement of the '417 Patent

76. Apple will have knowledge of the '417 patent and the infringing acts at least as early as its receipt of this Complaint.

77. On information and belief, Apple contributes to the infringement of claims 8, 12, 13, and 23 of the '417 patent.

78. On information and belief, the graphics decoding components of the Apple iPod Touch, iPhone, and iPad, are specially adapted for practicing an infringing method or an infringing use of one or more of claims 8, 12, 13, and 23 of the ’417 patent; embody a material part of the inventions claimed in the ’417 patent; and are not staple articles of commerce suitable for substantial non-infringing use. Apple sells for importation into the United States, imports, and/or sells after importation in the United States the Apple iPod Touch, iPhone, iPad, and the graphics decoding components of those products.

79. On information and belief, application software developers and consumers make and use the claimed inventions and practice the claimed methods by using the Apple iPod Touch, iPhone, and iPad, in combination with associated software (e.g., encoded image data) thereby directly infringing claims 8, 12, 13 and 23 of the ’417 patent.

3. Inducement of Infringement of the ’417 Patent

80. Apple will have knowledge of the ’417 patent and the infringing acts at least as early as its receipt of this Complaint.

81. On information and belief, Apple induces others to infringe claims 1, 7, 8, 12, 13, 15 and 23 of the ’417 patent by encouraging and facilitating others to perform actions known by Apple to infringe and with the intent that performance of the actions will infringe.

82. On information and belief, Apple encourages software application developers to make and use the claimed inventions by providing encoded image data, providing instructions and support for developing application software, and providing a distribution channel for application software for the Apple iPod Touch, iPhone, and iPad that include encoded image data.

83. On information and belief, Apple induces consumers to use the claimed inventions by providing the Apple iPod Touch, iPhone, and iPad and by providing application

software for those products that include encoded image data files. Further, Apple actively encourages, promotes, distributes, provides instruction for, and supports the use of software applications for its iPod Touch, iPhone, and iPad products that include encoded image data files.

84. On information and belief, application software developers and consumers make and use the claimed inventions and practice the claimed methods by using the Apple iPod Touch, iPhone, and iPad, in combination with associated software (e.g., encoded image data) thereby directly infringing claims 8, 12, 13 and 23 of the '417 patent.

85. On information and belief, Apple induces application software developers to make and use the claimed inventions and practice the claimed methods by providing the Apple SDK for use with Apple computers such as the MacBook. The Apple SDK works exclusively on Apple computers such as the MacBook, and the Apple SDK in combination with Apple computers such as the MacBook can be used to produce the software applications at issue. Further, Apple actively encourages application software developers to use the Apple SDK with Apple computers such as the MacBook to develop application software by distributing the Apple SDK and providing instructions and support for its use with Apple computers such as the MacBook and by selling, promoting, distributing, and marketing application software made using the Apple SDK.

86. On information and belief, application software developers make and use the claimed inventions and practice the claimed methods by using Apple computers such as the MacBook in combination with associated Apple software (e.g., the Apple SDK) thereby directly infringing claims 1, 7, 8, 12, 13, 15 and 23 of the '417 patent.

C. Infringement of the '978 Patent

87. On information and belief, the Accused Products infringe at least claims 11, 14, and 16 of the '978 patent. Charts that apply representative independent claims 11, 14, and 16 of the '978 patent to the Accused Products are attached to this Complaint as Exhibit No. 10.

1. Direct Infringement of the '978 Patent

88. On information and belief, Apple sells for importation into the United States, imports, and/or sells after importation in the United States the Accused Products, including certain application software containing encoded image data files, thereby directly infringing claims 11, 14, and 16 of the '978 patent. S3G has purchased certain application software containing encoded image data files in the United States directly from Apple. (Exhibit No. 16).

89. On information and belief, Apple makes and uses the claimed invention by testing or operating application software, containing encoded image data files, thereby directly infringing claims 11, 14, and 16 of the '978 patent.

2. Inducement of Infringement of the '978 Patent

90. Apple will have knowledge of the '978 patent and the infringing acts at least as early as its receipt of this Complaint.

91. On information and belief, Apple induces others to infringe claims 11, 14, and 16 of the '978 patent by encouraging and facilitating others to perform actions known by Apple to infringe and with the intent that performance of the actions will infringe.

92. On information and belief, Apple induces application software developers to make and use the claimed inventions by providing the Apple SDK for use with Apple computers such as the MacBook. The Apple SDK works exclusively on Apple computers such as the MacBook, and the Apple SDK in combination with Apple computers such as the MacBook can be used to produce the application software at issue. Further, Apple actively encourages

application software developers to use the Apple SDK with Apple computers such as the MacBook to develop application software by distributing the Apple SDK and providing instructions and support for its use with Apple computers such as the MacBook and by selling, promoting, distributing, and marketing applications made using the Apple SDK.

93. On information and belief, application software developers make and use the claimed inventions by using Apple computers such as the MacBook in combination with associated Apple software (e.g., the Apple SDK) thereby directly infringing claims 11, 14, and 16 of the '978 patent.

94. On information and belief, Apple induces consumers to use the claimed inventions by marketing, distributing, promoting, selling, and providing support for software applications, containing encoded image data files, for use with Apple's iPod Touch, iPhone, and iPad.

D. Infringement of the '146 Patent

95. On information and belief, the Accused Products infringe at least claims 2, 4, 8, 13, 16, 18, and 19 of the '146 patent. Charts that apply representative independent claims 4, 8, 13, and 18 of the '146 patent to the Accused Products are attached to this Complaint as Exhibit No. 11.

1. Direct Infringement of the '146 Patent

96. On information and belief, Apple sells for importation into the United States, imports, and/or sells after importation in the United States the Accused Products, including Apple computers such as the MacBook and associated Apple software (e.g., the Apple SDK). S3G has purchased these devices in the United States directly from Apple. (Exhibit No. 15).

97. On information and belief, Apple tests or operates the Accused Products in the United States by performing the claimed methods and by using the Accused Products, including

Apple computers such as the MacBook in combination with associated software (e.g., the Apple SDK), thereby directly infringing claims 2, 4, 8, 13, 16, 18, and 19 of the '146 patent.

2. Inducement of Infringement of the '146 Patent

98. Apple will have knowledge of the '146 patent and the infringing acts at least as early as its receipt of this Complaint.

99. On information and belief, Apple induces others to infringe claims 2, 4, 8, 13, 16, 18, and 19 of the '146 patent by encouraging and facilitating others to perform actions known by Apple to infringe and with the intent that performance of the actions will infringe.

100. On information and belief, Apple induces application software developers to make and use the claimed inventions and practice the claimed methods by providing the Apple SDK for use with Apple computers such as the MacBook. The Apple SDK works exclusively on Apple computers such as the MacBook, and the Apple SDK in combination with Apple computers such as the MacBook can be used to produce the application software at issue. Further, Apple actively encourages application software developers to use the Apple SDK with Apple computers such as the MacBook to develop software applications by distributing the Apple SDK and providing instructions and support for its use with Apple computers such as the MacBook and by selling, promoting, distributing, and marketing applications made using the Apple SDK.

101. On information and belief, application software developers make and use the claimed inventions and practice the claimed methods by using Apple computers such as the MacBook in combination with associated Apple software (e.g., the Apple SDK) thereby directly infringing claims 2, 4, 8, 13, 16, 18, and 19 of the '146 patent.

VII. SPECIFIC INSTANCES OF UNFAIR IMPORTATION AND SALE

102. During April 2010, samples of Accused Products were purchased in the United States.

103. On or about April 19, 2010, an Apple iPod Touch multimedia player was purchased in the United States. Exhibit No. 14 is a copy of the purchase receipt, and photographs of the packaging and iPod Touch device, including a label indicating that the device was "Assembled in China." A physical sample of the Apple iPod Touch, purchased as described above, is submitted as Physical Exhibit No. 3 to this Complaint.

104. On or about April 19, 2010, an Apple iPhone 3GS smart phone was purchased in the United States. Exhibit No. 12 is a copy of the purchase receipt, and photographs of the packaging and iPhone 3GS device, including a label indicating that the device was "Assembled in China." A physical sample of the Apple iPhone, purchased as described above, is submitted as Physical Exhibit No. 1 to this Complaint.

105. On or about April 20, 2010, an Apple iPad tablet computer was purchased in the United States. Exhibit No. 13 is a copy of the purchase receipt, and photographs of the packaging and iPad device, including a label indicating that the device was "Assembled in China." A physical sample of the Apple iPad, purchased as described above, is submitted as Physical Exhibit No. 2 to this Complaint.

106. On or about May 26, 2010, three software applications for the iPhone were purchased in the United States. Exhibit No. 16 is a copy of the purchase receipt, screenshots from these software applications, and information about the location of the developers. On information and belief, all three software applications were made outside of the United States. A compact disc containing the three iPhone applications, purchased as described above, is submitted as Physical Exhibit No. 6.

107. On or about April 19, 2010, an Apple MacBook Pro computer was purchased in the United States. Exhibit No. 15 is a copy of the purchase receipt, and photographs of the packaging and MacBook Pro device, including a label indicating that the device was "Assembled in China." A physical sample of the Apple MacBook Pro, purchased as described above, is submitted as Physical Exhibit No. 4 to this Complaint.

VIII. HARMONIZED TARIFF SCHEDULE ITEM NUMBERS

108. On information and belief, the Harmonized Tariff Schedule of the United States item numbers under which the infringing image processing systems, and products containing same have been imported into the United States may include at least the following HTS numbers: 8517.12.00 (mobile phones), 8519.81.40, 8519.89.30, or 8521.90.00 (portable media players), 8471.30.01, 8471.41.01, or 8471.49.00 (portable computers), and 9504.10.0000 (software).

IX. RELATED LITIGATION

109. S3G has not previously asserted any of the Asserted Patents in any other judicial or administrative proceeding.

X. THE DOMESTIC INDUSTRY

110. An industry in the United States, relating to the image processing systems protected by the Asserted Patents, exists under 19 U.S.C. § 1337(a)(3)(a)-(c), comprising significant investments in physical operations, employment of labor and capital, and exploitation of the Asserted Patents.

A. S3G's Investments in the Domestic Industry

111. S3 Graphics, Inc., employs a work force the United States that conducts research, development, engineering, product design, support, and repair in the United States for S3 Graphics, Inc.'s products that practice the Asserted Patents, including at least the S3 Graphics,

Inc. Chrome series of graphics products (Chrome S25, S27, 2300E; Chrome 430, 435ULP, 440GTX; Chrome 4300E, 4400E; Chrome 535 ULP, and 540GTX. S3 Graphics, Inc., makes significant investments in plant, equipment, labor, engineering, and research and development in the United States in connection with its research, development, design, technical support, and repair of products that practice the Asserted Patents. *See* Confidential Exhibit Nos. 28C-30C.

B. S3G's Practice of the Asserted Patents

112. S3G makes extensive use of the Asserted Patents in several of its own products. As noted above, S3G has a variety of graphics products including the Chrome S25, S27, 2300E, 430, 435ULP, 440GTX, 4300E, 4400E, 535 ULP, 540GTX, and 5400E. Each of these products practices the Asserted Patents. As an example, the Chrome 430 and 440 chips are provided with this Complaint as Physical Exhibit No. 5.

113. An exemplary claim chart comparing the Chrome 400 series to a representative claim of the '087 patent is attached as Confidential Exhibit No. 20C.

114. An exemplary claim chart comparing the Chrome 400 series to a representative claim of the '417 patent is attached as Confidential Exhibit No. 21C.

115. An exemplary claim chart comparing the Chrome 400 series to a representative claim of the '978 patent is attached as Confidential Exhibit No. 22C.

116. An exemplary claim chart comparing the Chrome 400 series to a representative claim of the '146 patent is attached as Confidential Exhibit No. 23C.

C. S3G's Licensees' Practice of the Asserted Patents

117. On information and belief, S3G's licensees practice at least one claim of each of the Asserted Patents in the United States. For example, a major computer software company holds a license to the Asserted Patents. Charts comparing a representative claim of each of the Asserted Patents to a corresponding product of this major computer software company, are

submitted as Confidential Exhibit Nos. 24C-27C. Similarly, a number of other major computer software and hardware companies hold licenses to the Asserted Patents. On information and belief, certain of S3G's licensees have a contractual obligation to incorporate S3G's image processing technologies in their products and S3G's licensees practice at least one claim of each of the Asserted Patents. On information and belief, S3G's licensees make significant investments in plant, equipment, labor, and consulting services in the United States in the course of researching, developing, engineering, manufacturing, and supporting products that practice one or more Asserted Claims of the Asserted Patents.

D. S3G's Licensing Business

118. S3G operates a licensing business from its headquarters in Fremont, California. S3G and its consultants formulate licensing strategies, identify products and companies that currently or prospectively could utilize S3G image processing technology, analyze those products and companies for potential licensing opportunities, negotiate licenses under the S3G patent portfolio, and monitor and enforce compliance with those licenses and S3G patent rights. The work force and compensation of S3 Graphics, Inc., in Fremont, California responsible for licensing S3G's patent portfolios, including the Asserted Patents, is identified in Confidential Exhibit No. 31C. S3G's domestic investments in plant, equipment, labor, and consulting services used to conduct that licensing business are set forth in Confidential Exhibit No. 31C.

XI. RELIEF REQUESTED

119. WHEREFORE, by reason of the foregoing, S3G respectfully requests that the United States International Trade Commission:

(a) Institute an immediate investigation, pursuant to Section 337 of the Tariff Act of 1930, as amended, 19 U.S.C. § 1337(a)(1)(B)(i) and (b)(1), with respect to violations of Section 337 by the Apple Inc. based upon its sale for importation, importation, and/or sale after importation into the United States of certain electronic devices with image processing systems, components thereof, and associated software, that infringe one or more of the Asserted Claims of S3G's United States Patent Nos. 7,043,087; 6,775,417; 6,683,978; and 6,658,146;

(b) Schedule and conduct a hearing on said unlawful acts and, following said hearing;

(c) Issue a limited exclusion order pursuant to 19 U.S.C. § 1337(d)(1), barring from entry into the United States all Accused Products, that infringe one or more of the Asserted Claims of S3G's United States Patent Nos. 7,043,087; 6,775,417; 6,683,978; and 6,658,146, including but not limited to the Apple iPod Touch, iPhone, iPad, certain application software for those products, and related software (e.g., the Apple SDK);

(d) Issue a permanent cease and desist order, pursuant to 19 U.S.C. § 1337(f), directing Apple Inc., to cease and desist from selling for importation into the United States, importing, selling after importation into the United States, offering for sale, marketing, advertising, demonstrating, sampling, warehousing inventory for distribution, offering for sale, selling, distributing, licensing, testing, technical support, use, or other related commercial activity involving imported Accused Products that infringe one or more of the Asserted Claims of S3G's United States Patent Nos. 7,043,087; 6,775,417; 6,683,978; and 6,658,146, including but not limited to the Apple iPod Touch, iPhone, iPad, certain application software for those products, and related software (e.g., the Apple SDK); and

(e) Grant such other and further relief as the Commission deems just and proper based on the facts determined by the investigation and the authority of the Commission.

Dated: May 28, 2010

Respectfully submitted,

[illegible]

Thomas L. Jarvis
Thomas W. Winland
John R. Alison
Paul C. Goulet
John M. Williamson
FINNEGAN, HENDERSON, FARABOW,
GARRETT & DUNNER, LLP
901 New York Avenue, N.W.
Washington, D.C. 20001-4413
Telephone: (202) 408-4000
Facsimile: (202) 408-4400

Attorneys for Complainants
S3 Graphics Co., Ltd. and
S3 Graphics, Inc.

VERIFICATION OF COMPLAINT

I, Ken Weng, declare, in accordance with 19 C.F.R. §§ 210.4 and 210.12(a), under penalty of perjury, that the following statements are true:

1. I am CEO of S3 Graphics, Inc., and am duly authorized to sign this Complaint on behalf of Complainant S3 Graphics, Inc.;

2. I am CEO of S3 Graphics Co., Ltd., and am duly authorized to sign this Complaint on behalf of Complainant S3 Graphics Co., Ltd.;

3. I have read the foregoing Complaint;

4. To the best of my knowledge, information, and belief, based on reasonable inquiry, the foregoing Complaint is well-founded in fact and is warranted by existing law or by a non-frivolous argument for the extension, modification, or reversal of existing law or the establishment of new law;

5. The allegations and other factual contentions have evidentiary support or are likely to have evidentiary support after a reasonable opportunity for further investigation or discovery; and

6. The foregoing Complaint is not being filed for an improper purpose, such as to harass or to cause unnecessary delay or needless increase in the cost of litigation.

Executed May 28, 2010.

[illegible]

Ken Weng
CEO
S3 Graphics, Inc.
S3 Graphics Co., Ltd.
1025 Mission Court
Fremont, CA 94539