(12) UK Patent Application (19) GB (11) 2 313 757 (13) A

(43) Date of A Publication 03.12.1997

(21) Application No 9712513.2

(22) Date of Filing 25.06.1996

Date Lodged 16.06.1997

(30) Priority Data
(31) 08498036 (32) 03.07.1995 (33) US

(62) Divided from Application No 9613320.2 under Section 15(4) of the Patents Act 1977

(51) INT CL^6
H03M 7/30 , H04N 1/41 7/50

(52) UK CL (Edition O)
H4P PDCFT PDCFX
H4F FD12X FD3R FD3T FD3X FD30B FD30K FRW

(56) Documents Cited
None

(58) Field of Search
UK CL (Edition O) H4F FRW , H4P PDCFT PDCFX
INT CL^6 H03M 7/30 , H04N 1/41 7/30 7/50
Online : WPI, INSPEC, JAPIO

(71) Applicant(s)
Ricoh Company,Ltd

(Incorporated in Japan)

3-6,1-chome,Nakamagome, Ota-ku, Tokyo 143, Japan

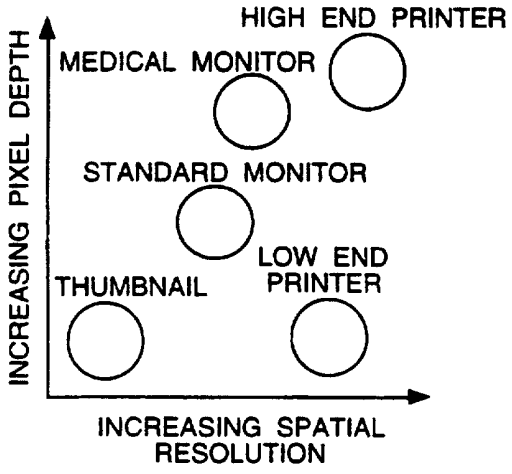
(72) Inventor(s)
Ahmad Zandi
Edward L Schwartz
Michael Gormish
Martin Boliek

(74) Agent and/or Address for Service
J A Kemp & Co
14 South Square, Gray's Inn, LONDON, WC1R 5LX, United Kingdom

(54) **Method using an embedded codestream**

(57) A target device (eg. a monitor) is identified to receive data in an embedded codestream, and each bit-plane is decoded to provide data to the target device, each bit-plane being truncated in accordance with a marker or pointer in each coding unit denoting the location where truncation can occur. Prior to decoding, the resolution of the target device is identified and decoding is then truncated to decode only the data necessary to support the target device.

Fig.34.



GB 2 313 757 A

Fig.1.

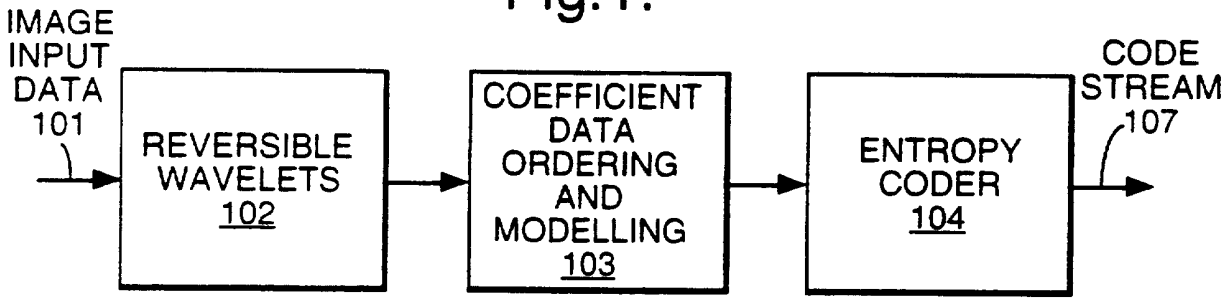


Fig.2.

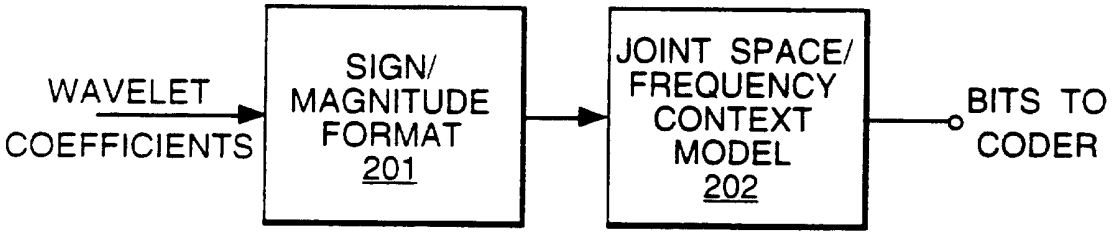


Fig.3A.

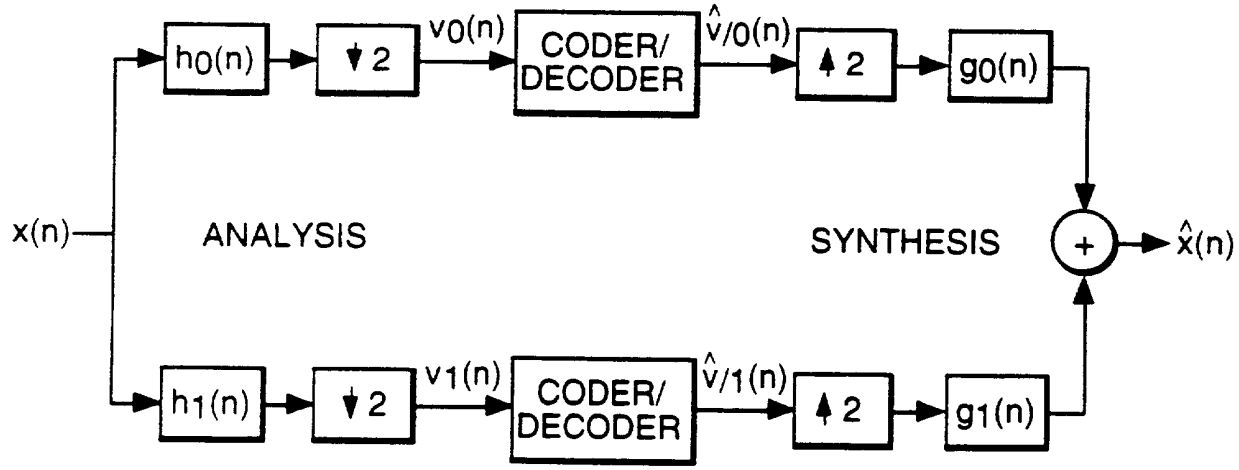


Fig.3B.

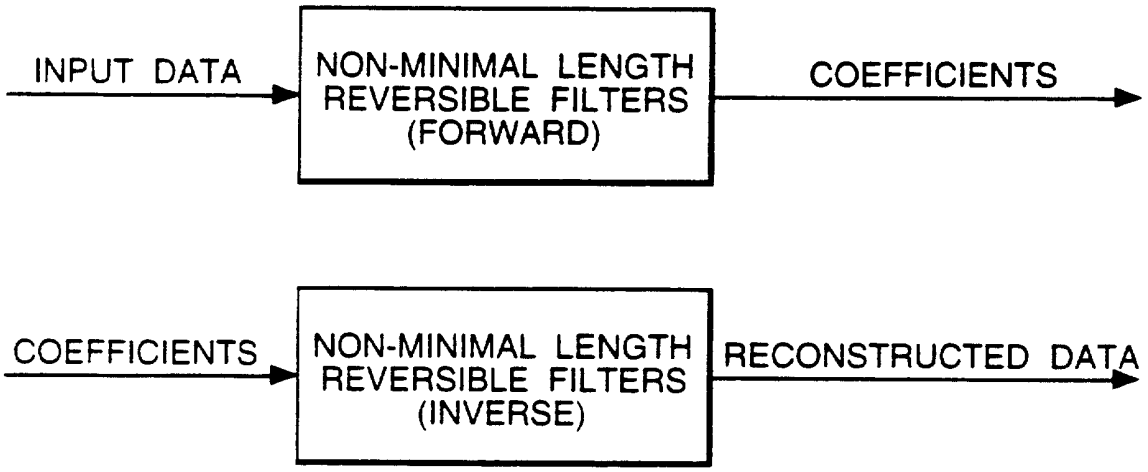
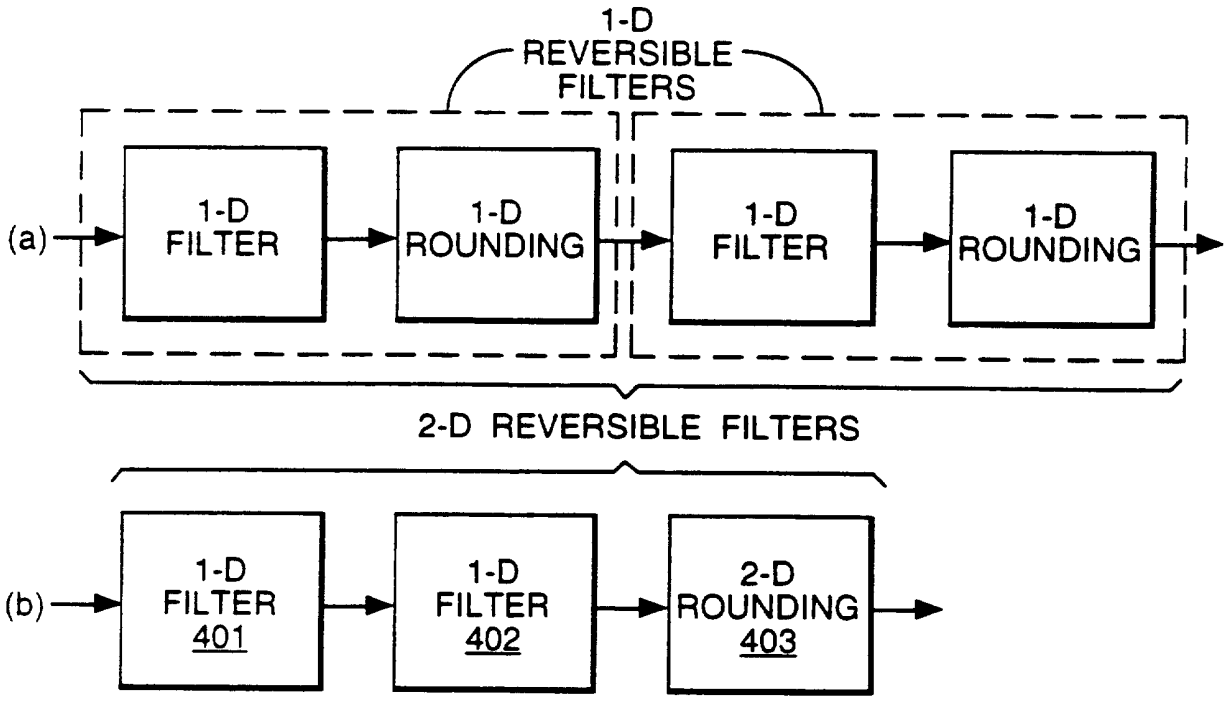

INPUT DATA
NON-MINIMAL LENGTH REVERSIBLE FILTERS (FORWARD)
COEFFICIENTS
COEFFICIENTS
NON-MINIMAL LENGTH REVERSIBLE FILTERS (INVERSE)
RECONSTRUCTED DATA

Fig.4.


1-D REVERSIBLE FILTERS
(a)
1-D FILTER
1-D ROUNDING
1-D FILTER
1-D ROUNDING
2-D REVERSIBLE FILTERS
(b)
1-D FILTER 401
1-D FILTER 402
2-D ROUNDING 403

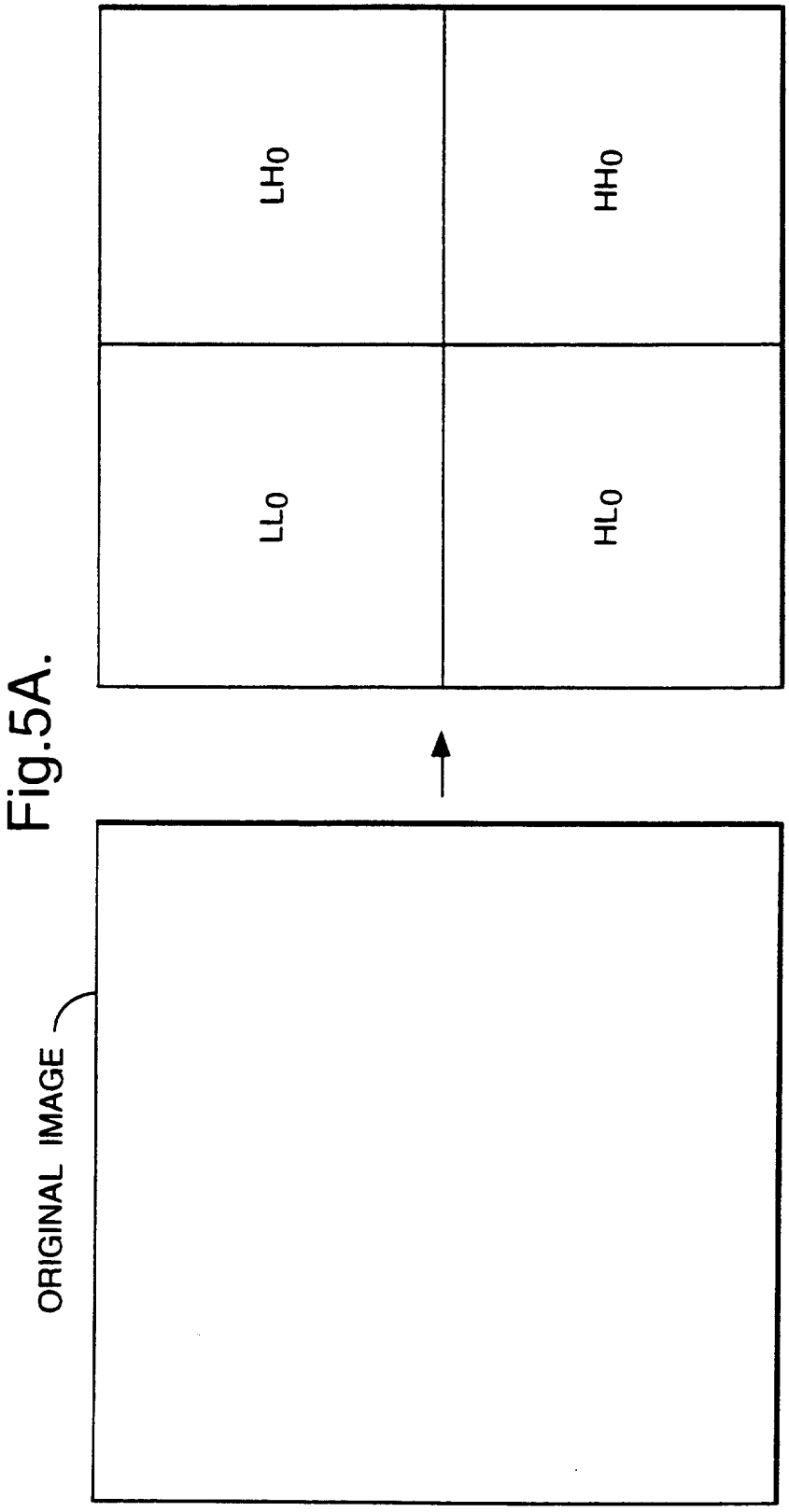


Fig.5A.

Fig.5B.

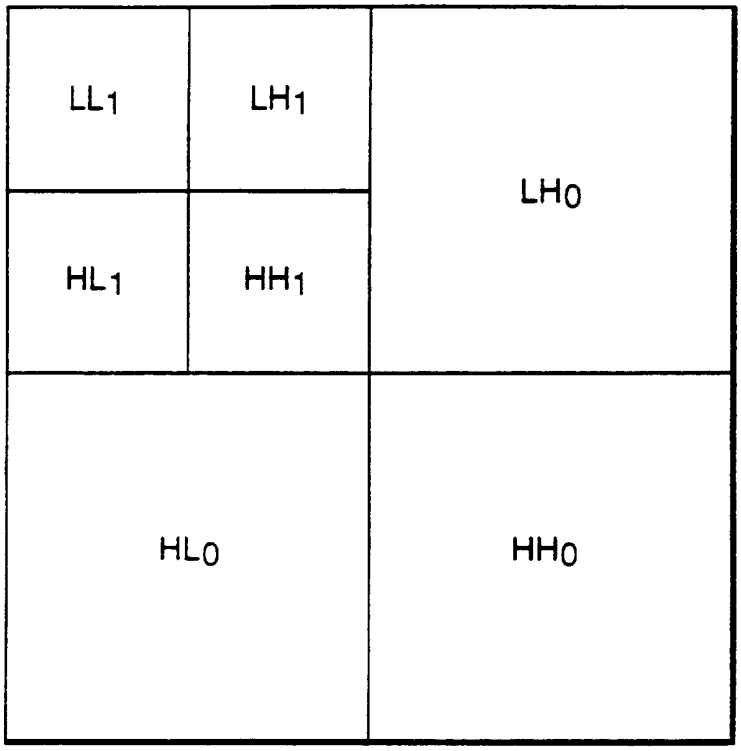

LL1
LH1
LH0
HL1
HH1
HL0
HH0

Fig.5C.

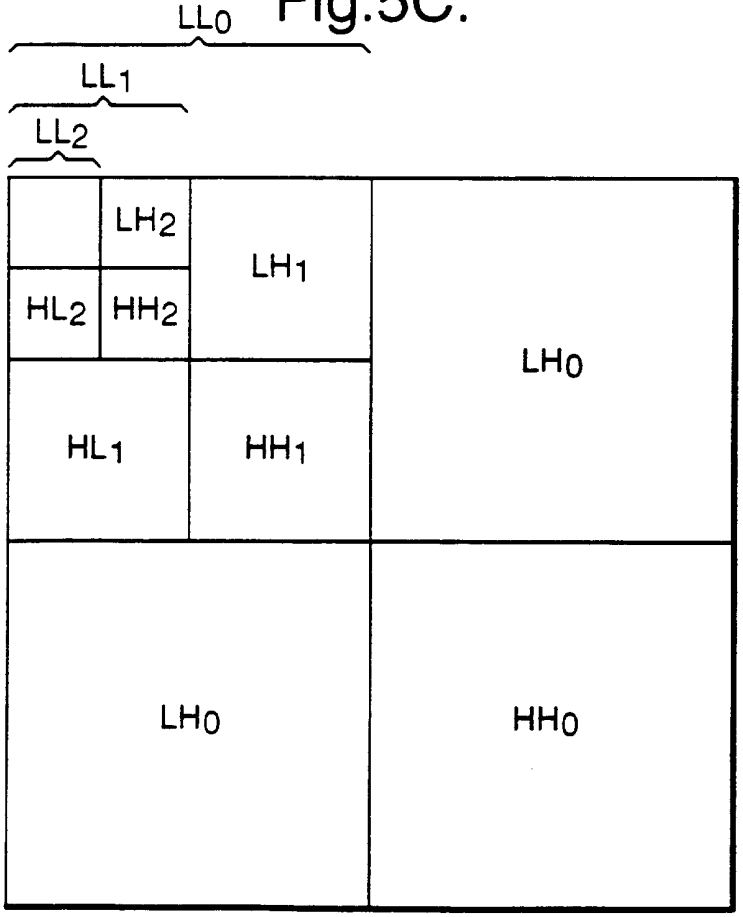

LL0
LL1
LL2
LH2
LH1
HL2
HH2
LH0
HL1
HH1
LH0
HH0

Fig.5D.

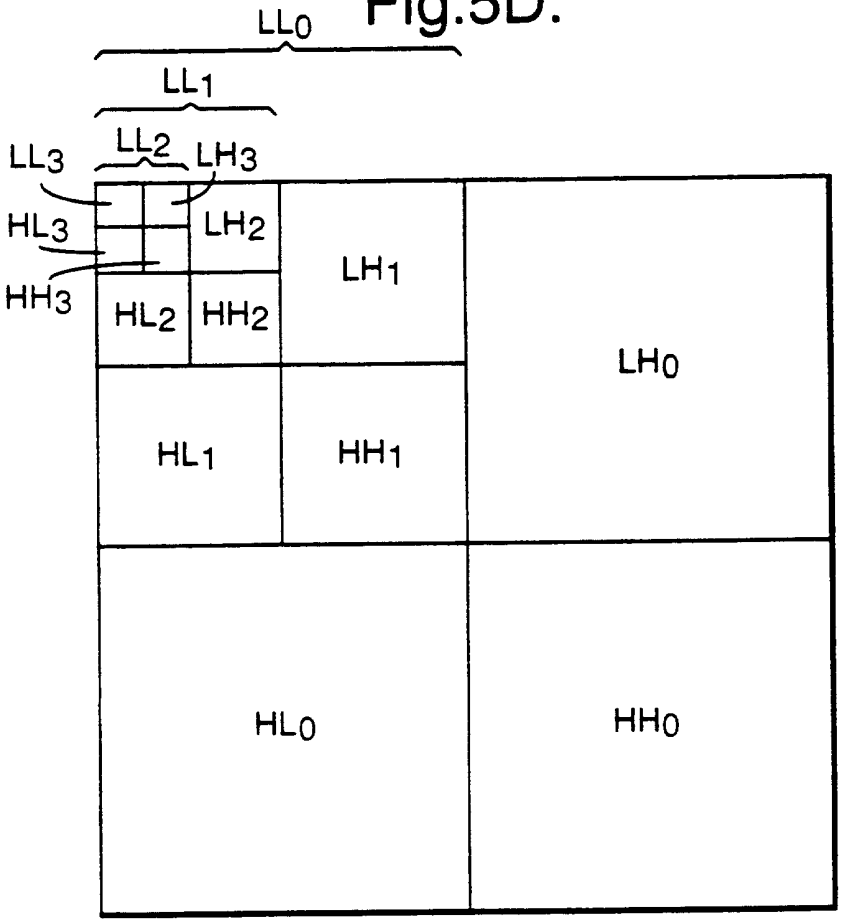

LL0
LL1
LL2
LH3
LL3
HL3
LH2
LH1
HH3
HL2
HH2
LH0
HL1
HH1
HL0
HH0

Fig.6.

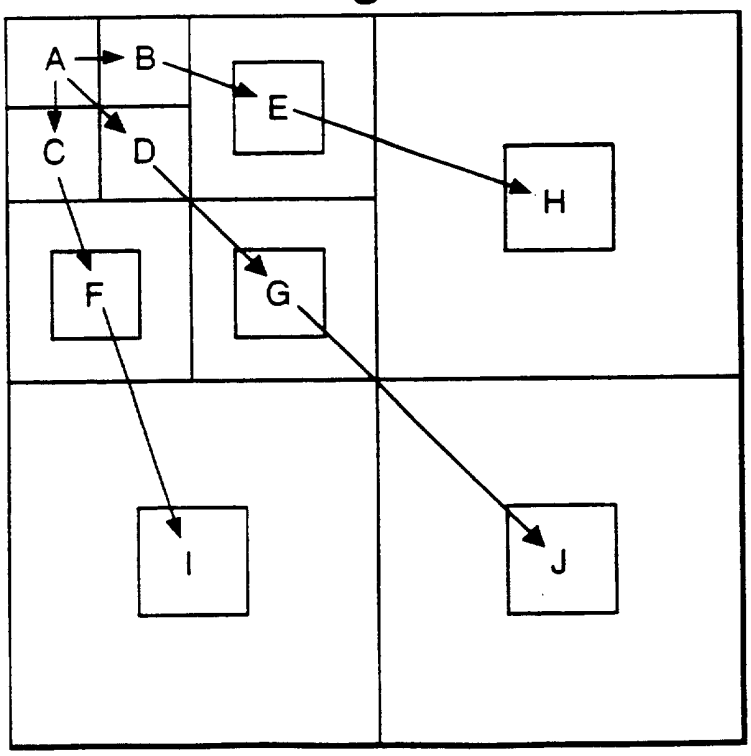
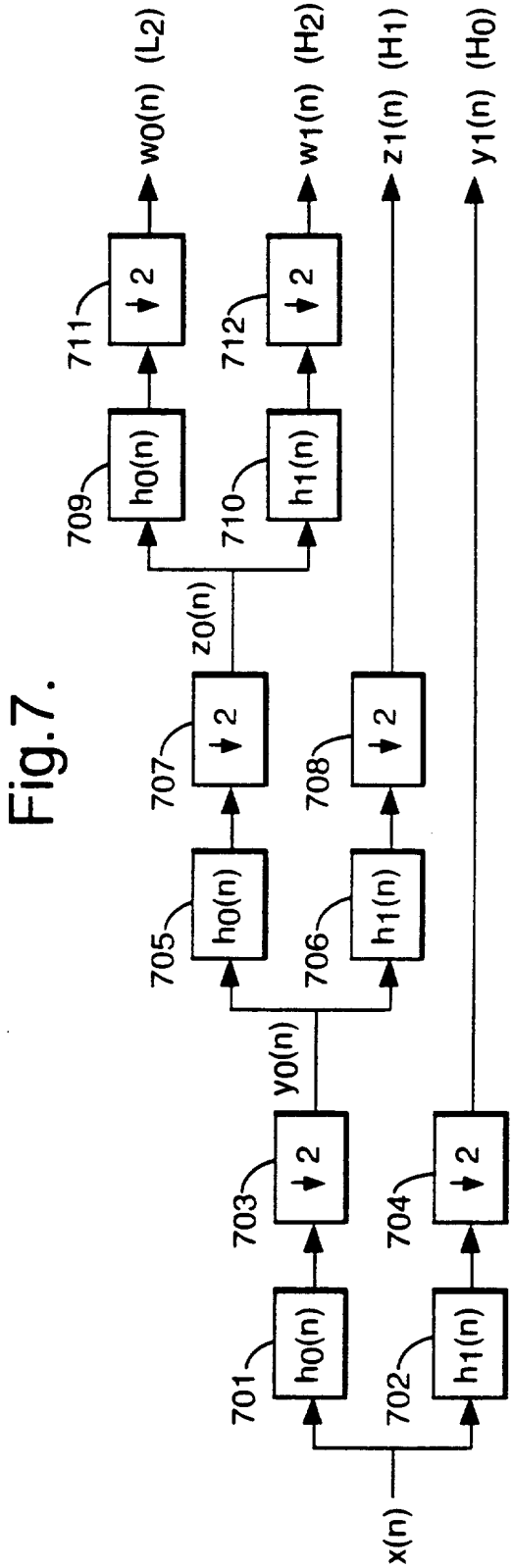

A
B
E
C
D
H
F
G
I
J

Fig.7.


x(n)
701
h0(n)
702
h1(n)
703
2
704
2
y0(n)
y1(n) (H0)
705
h0(n)
706
h1(n)
707
2
708
2
z0(n)
z1(n) (H1)
709
h0(n)
710
h1(n)
711
2
712
2
w0(n) (L2)
w1(n) (H2)

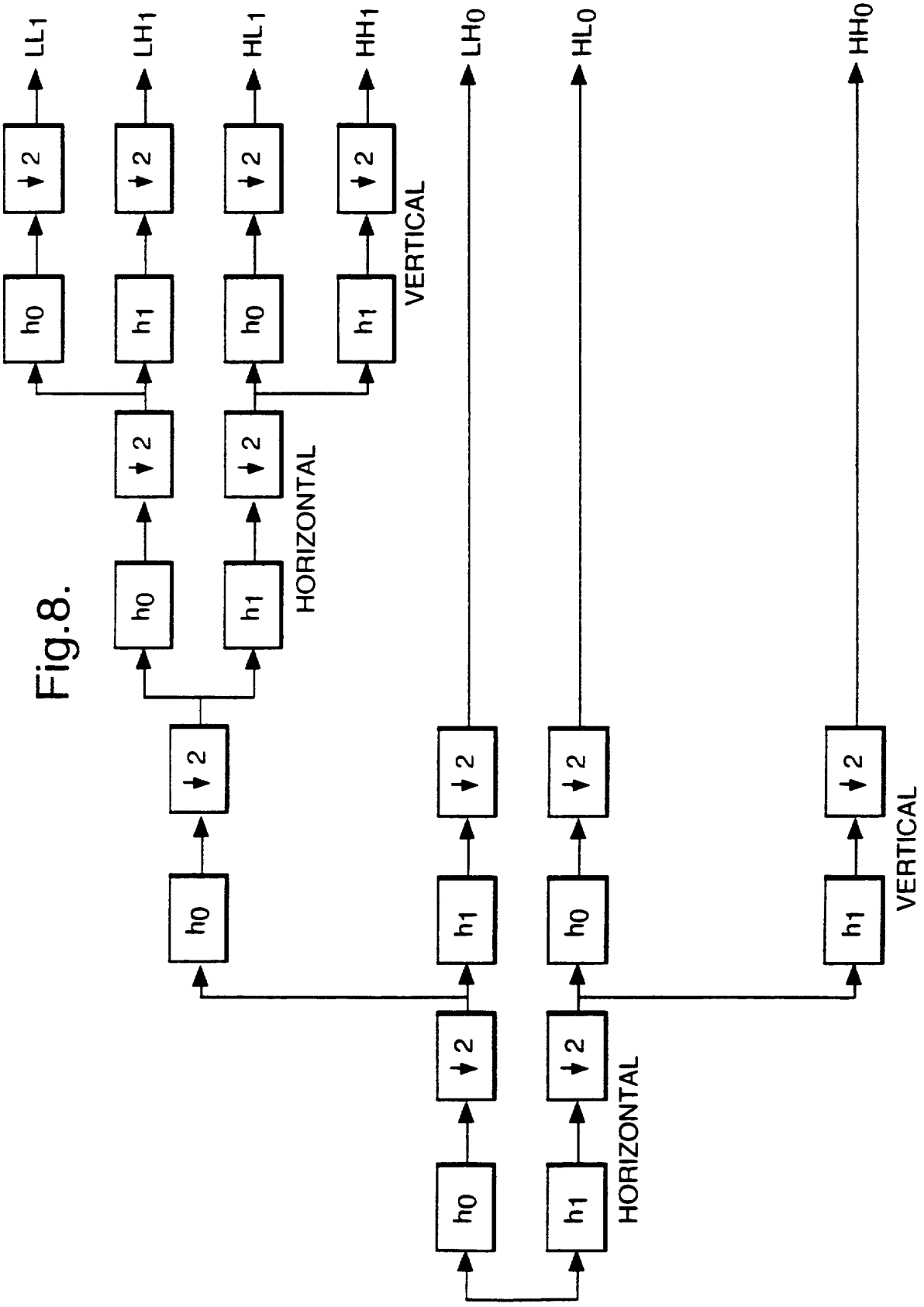

LL1
LH1
HL1
HH1
LH0
HL0
HH0
h0
h1
2
HORIZONTAL
VERTICAL
HORIZONTAL
VERTICAL

Fig.8.

Fig.9.

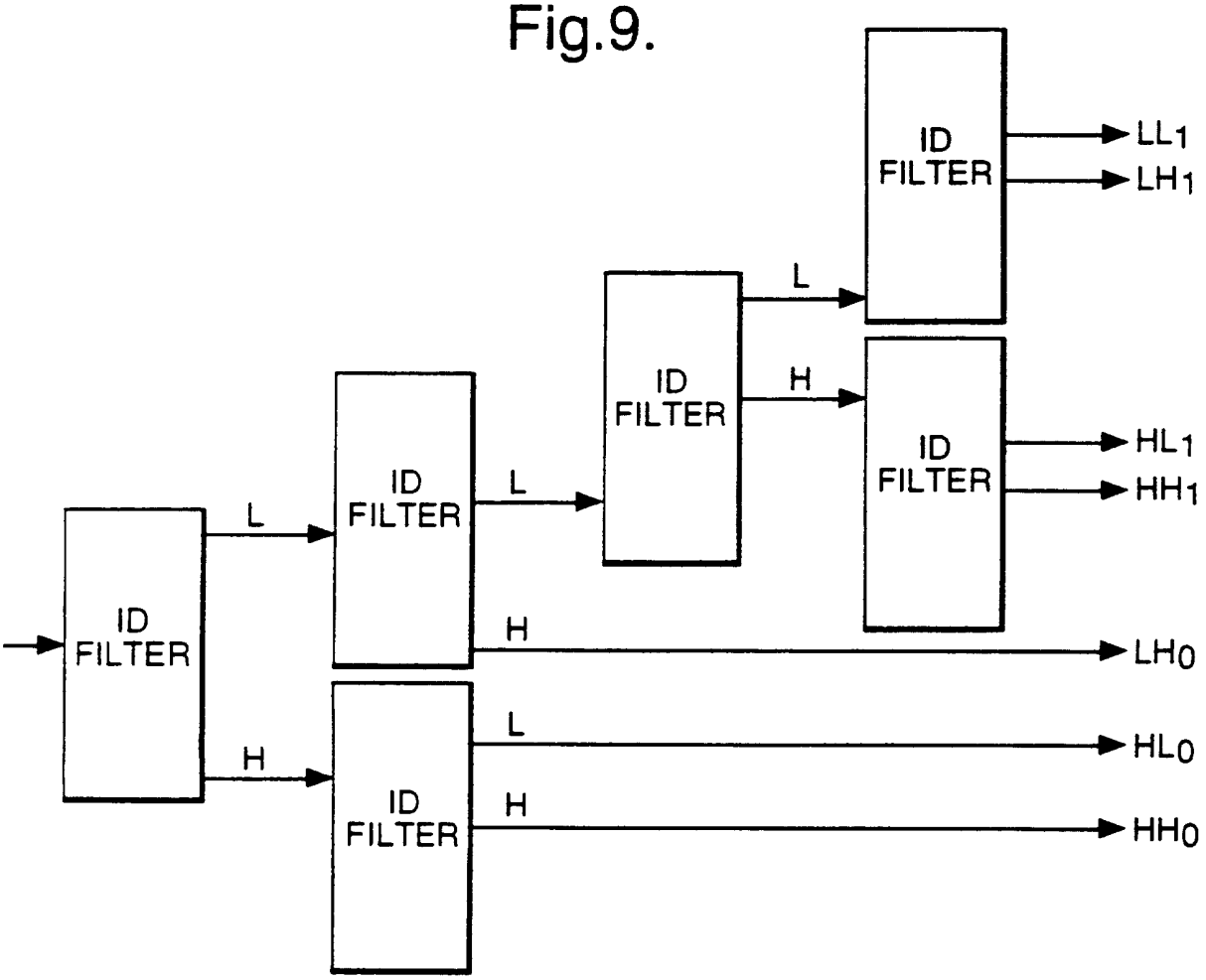

ID FILTER
L
H
ID FILTER
L
H
ID FILTER
L
H
ID FILTER
L
H
LL1
LH1
HL1
HH1
LH0
HL0
HH0

Fig.10.

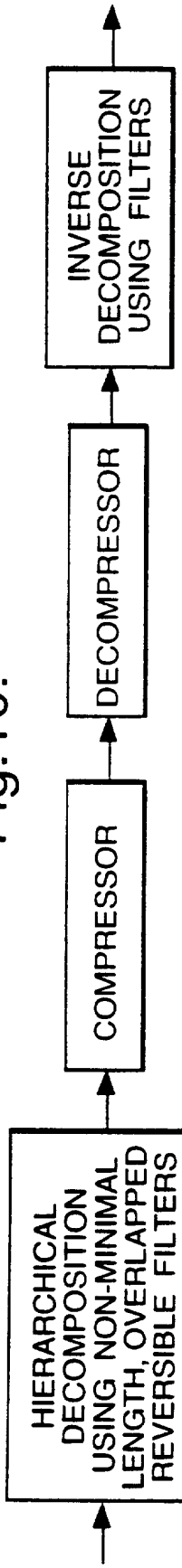

HIERARCHICAL DECOMPOSITION USING NON-MINIMAL LENGTH, OVERLAPPED REVERSIBLE FILTERS
COMPRESSOR
DECOMPRESSOR
INVERSE DECOMPOSITION USING FILTERS

Fig.11.

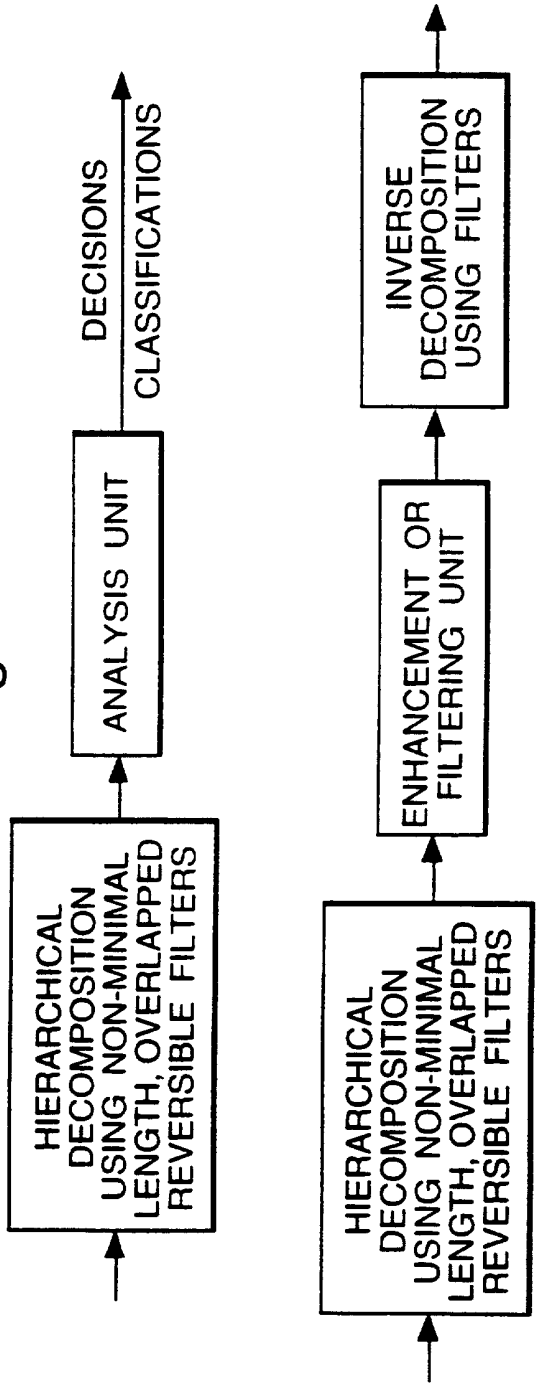

HIERARCHICAL DECOMPOSITION USING NON-MINIMAL LENGTH, OVERLAPPED REVERSIBLE FILTERS
ANALYSIS UNIT
DECISIONS
CLASSIFICATIONS
HIERARCHICAL DECOMPOSITION USING NON-MINIMAL LENGTH, OVERLAPPED REVERSIBLE FILTERS
ENHANCEMENT OR FILTERING UNIT
INVERSE DECOMPOSITION USING FILTERS

Fig.12A.

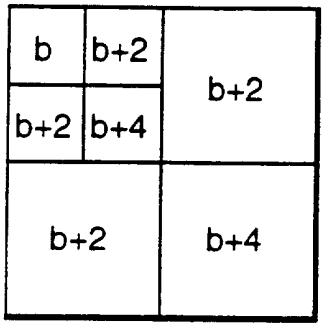


Fig.12B.

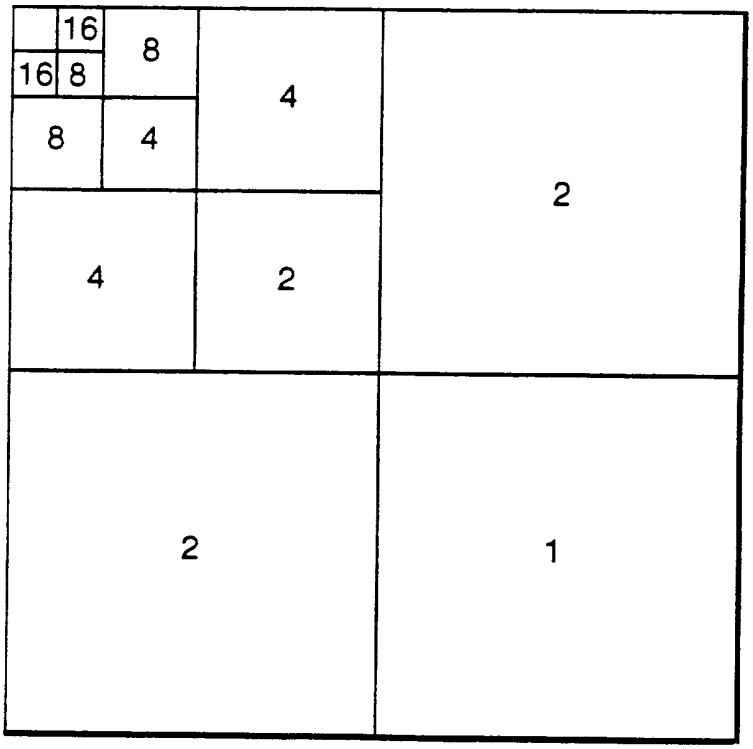


Fig.13.

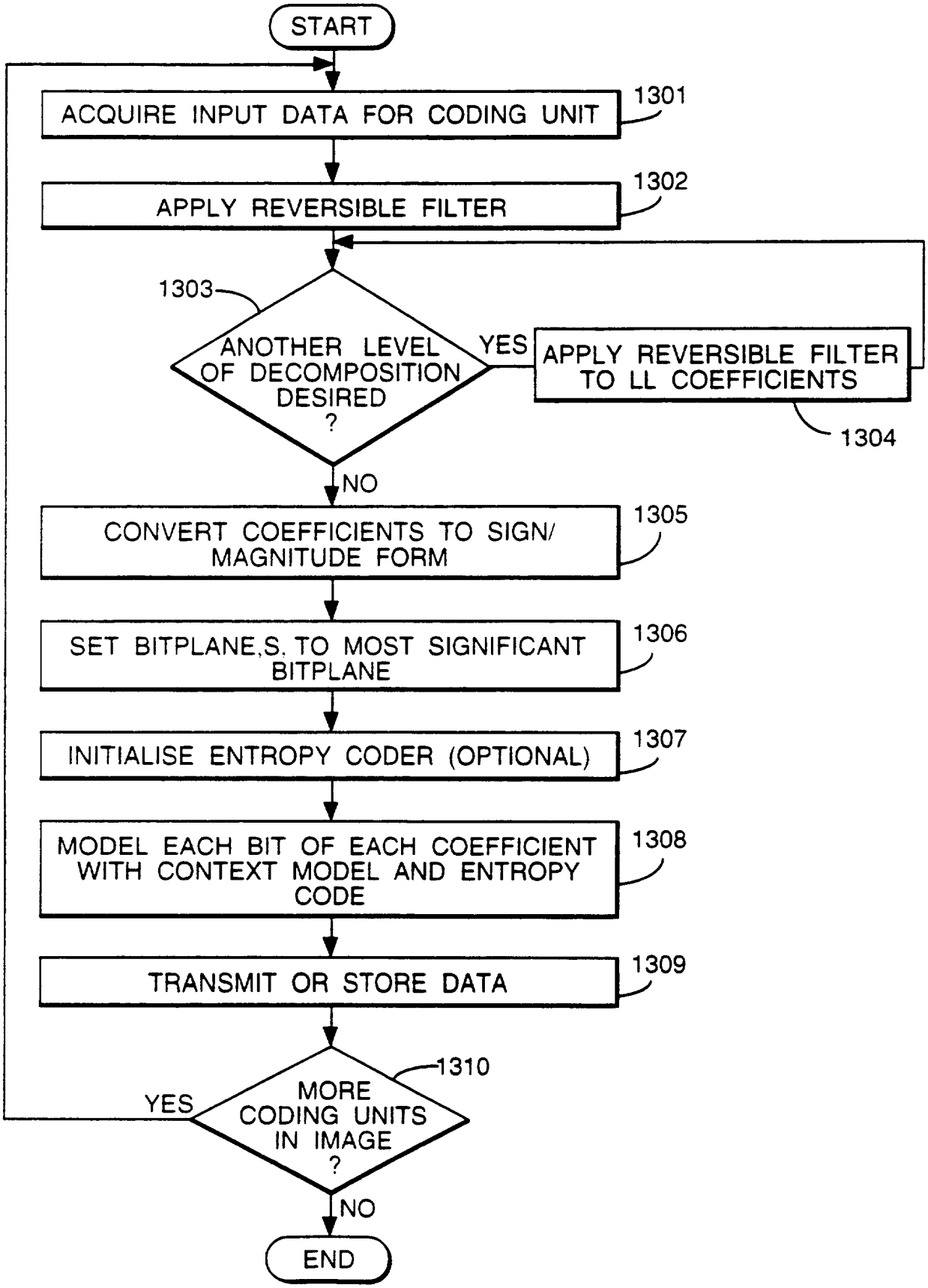

START
ACQUIRE INPUT DATA FOR CODING UNIT
1301
APPLY REVERSIBLE FILTER
1302
1303
ANOTHER LEVEL OF DECOMPOSITION DESIRED ?
YES
APPLY REVERSIBLE FILTER TO LL COEFFICIENTS
1304
NO
CONVERT COEFFICIENTS TO SIGN/ MAGNITUDE FORM
1305
SET BITPLANE,S. TO MOST SIGNIFICANT BITPLANE
1306
INITIALISE ENTROPY CODER (OPTIONAL)
1307
MODEL EACH BIT OF EACH COEFFICIENT WITH CONTEXT MODEL AND ENTROPY CODE
1308
TRANSMIT OR STORE DATA
1309
1310
MORE CODING UNITS IN IMAGE ?
YES
NO
END

Fig.14.

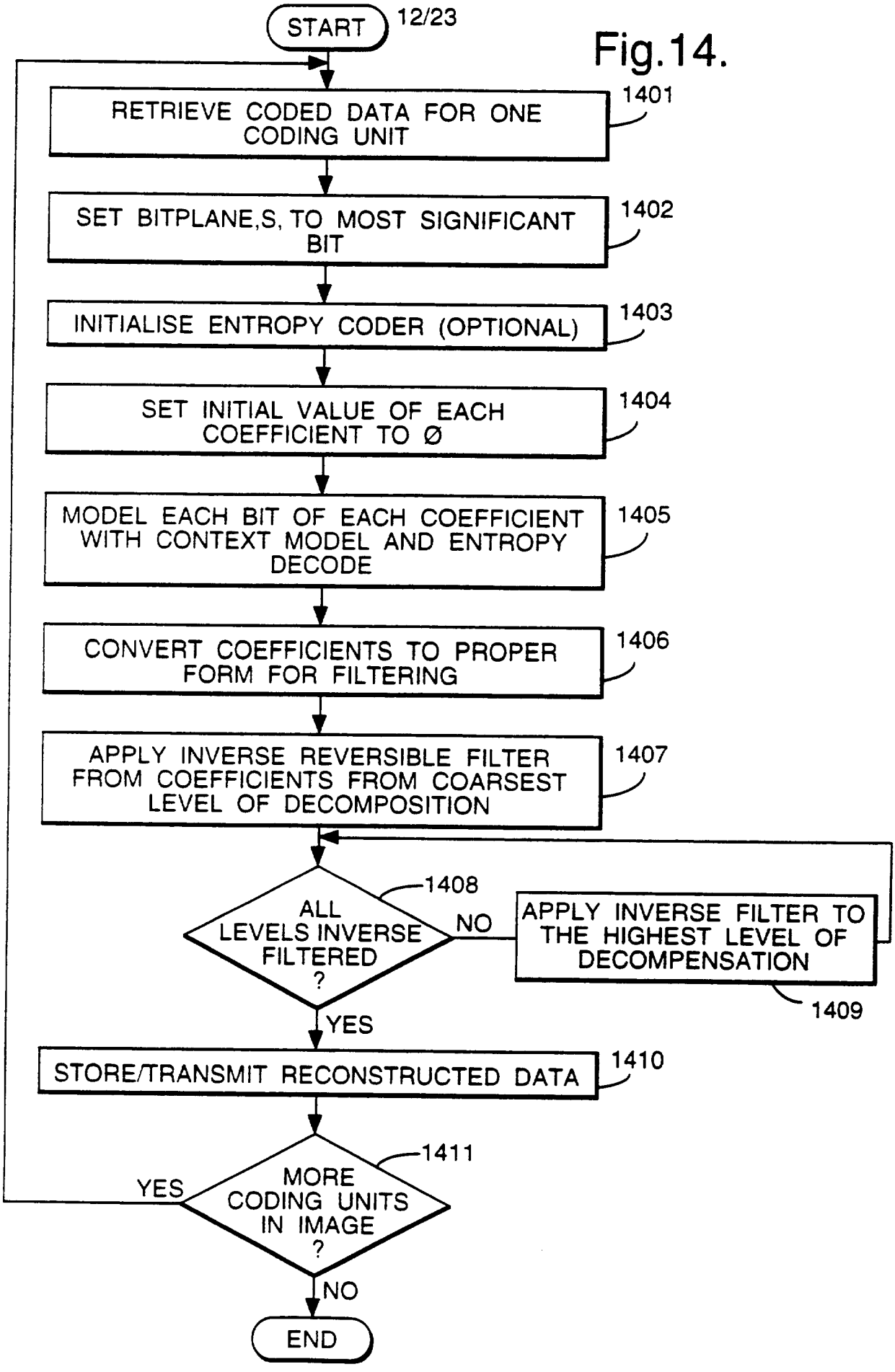
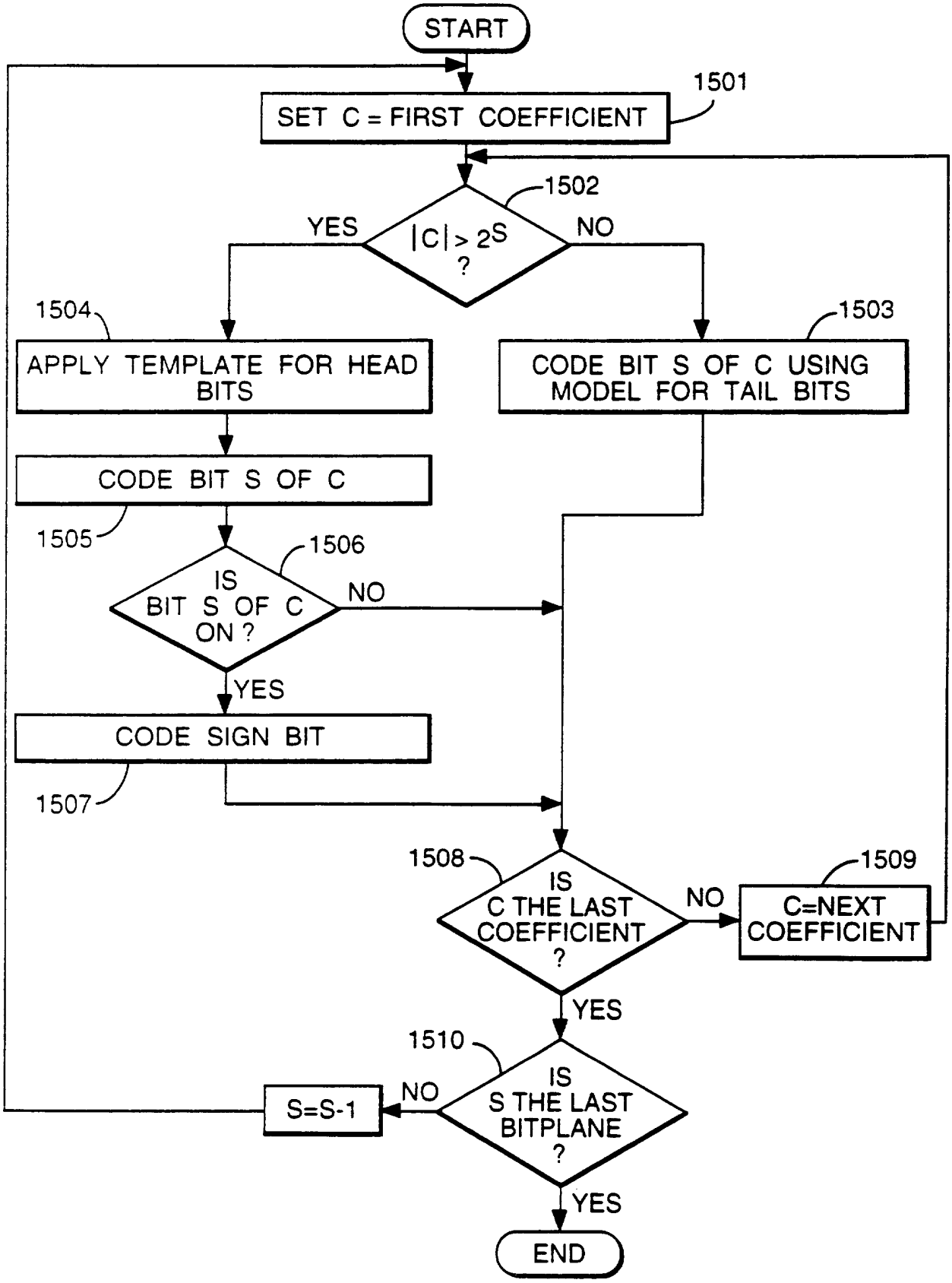

START
RETRIEVE CODED DATA FOR ONE CODING UNIT
1401
SET BITPLANE,S, TO MOST SIGNIFICANT BIT
1402
INITIALISE ENTROPY CODER (OPTIONAL)
1403
SET INITIAL VALUE OF EACH COEFFICIENT TO Ø
1404
MODEL EACH BIT OF EACH COEFFICIENT WITH CONTEXT MODEL AND ENTROPY DECODE
1405
CONVERT COEFFICIENTS TO PROPER FORM FOR FILTERING
1406
APPLY INVERSE REVERSIBLE FILTER FROM COEFFICIENTS FROM COARSEST LEVEL OF DECOMPOSITION
1407
ALL LEVELS INVERSE FILTERED ?
1408
NO
APPLY INVERSE FILTER TO THE HIGHEST LEVEL OF DECOMPENSATION
1409
YES
STORE/TRANSMIT RECONSTRUCTED DATA
1410
MORE CODING UNITS IN IMAGE ?
1411
YES
NO
END

Fig.15.


START
SET C = FIRST COEFFICIENT
1501
1502
|C| > 2^S ?
YES
NO
1504
APPLY TEMPLATE FOR HEAD BITS
1503
CODE BIT S OF C USING MODEL FOR TAIL BITS
CODE BIT S OF C
1505
1506
IS BIT S OF C ON ?
NO
YES
CODE SIGN BIT
1507
1508
IS C THE LAST COEFFICIENT ?
NO
1509
C=NEXT COEFFICIENT
YES
1510
IS S THE LAST BITPLANE ?
NO
S=S-1
YES
END

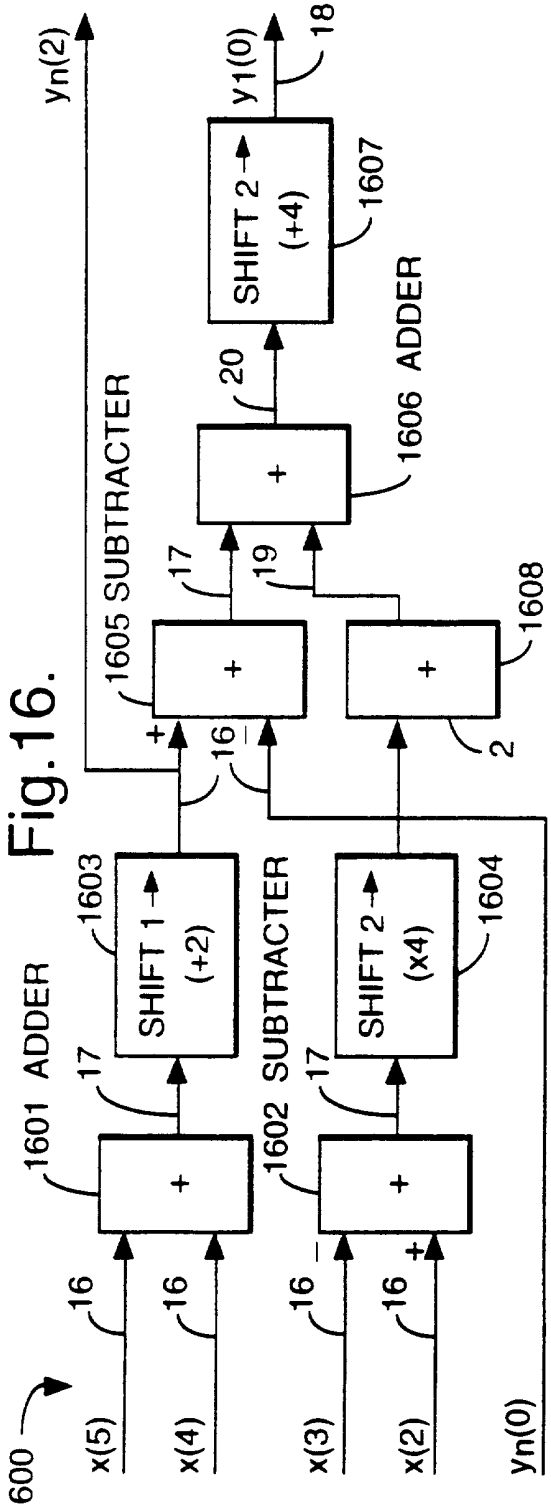

1600
x(5)
x(4)
x(3)
x(2)
yn(0)
16
1601 ADDER
1602 SUBTRACTER
17
SHIFT 1→ (+2)
1603
SHIFT 2→ (x4)
1604
1605 SUBTRACTER
16
2
1608
19
1606 ADDER
20
SHIFT 2→ (+4)
1607
y1(0)
18
yn(2)

Fig.16.

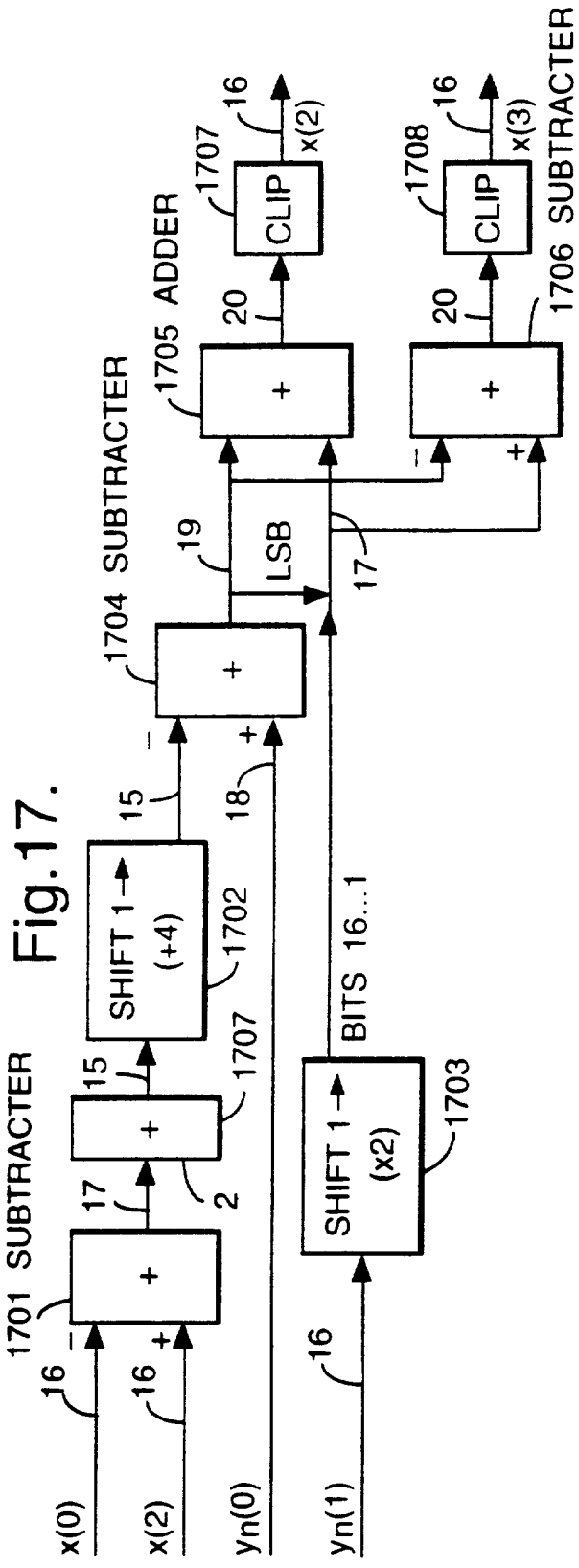

x(0)
x(2)
yn(0)
yn(1)
16
1701 SUBTRACTER
17
15
1702
SHIFT 1→ (+4)
1704 SUBTRACTER
18
19
LSB
1705 ADDER
20
1707
CLIP
x(2)
BITS 16...1
SHIFT 1→ (x2)
1703
1708
x(3)
1706 SUBTRACTER

Fig.17.

Fig.18.

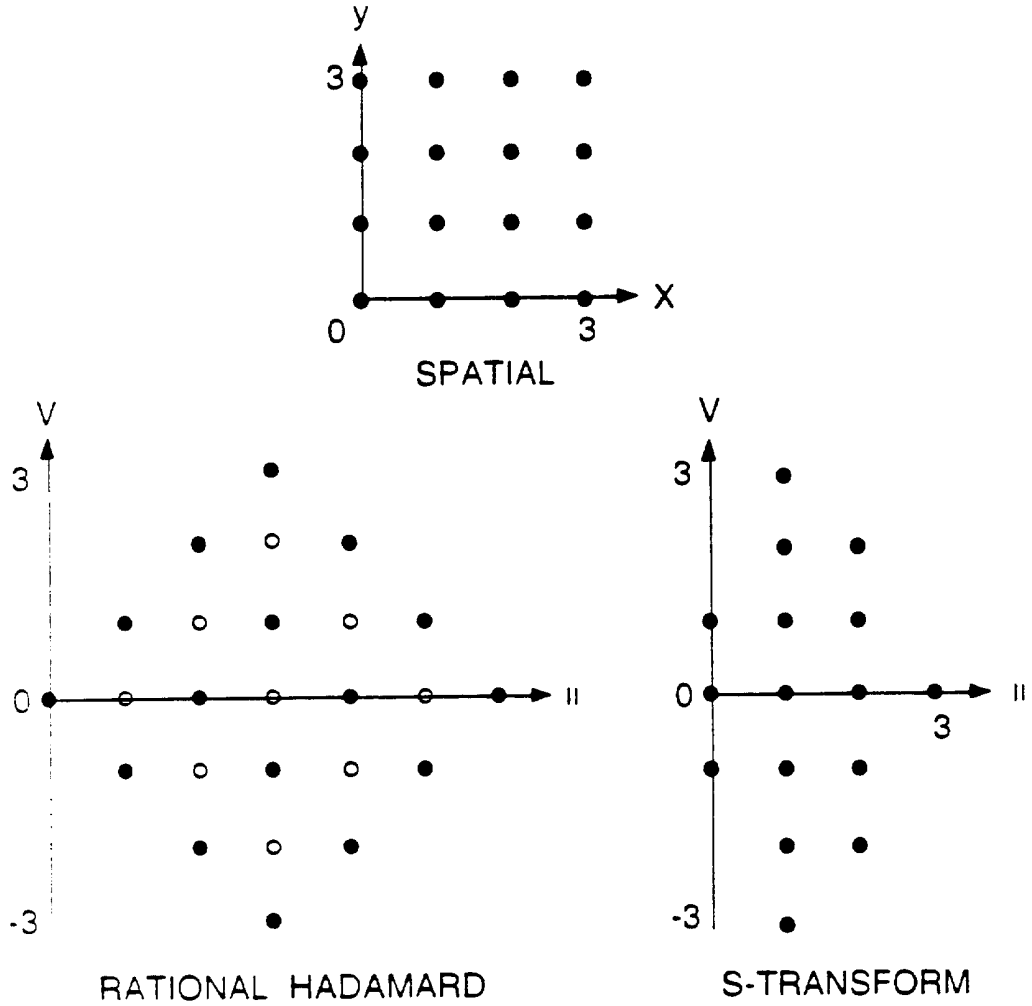

y
3
0
3
X
SPATIAL
V
3
0
-3
II
RATIONAL HADAMARD
V
3
0
-3
3
II
S-TRANSFORM

Fig.20.

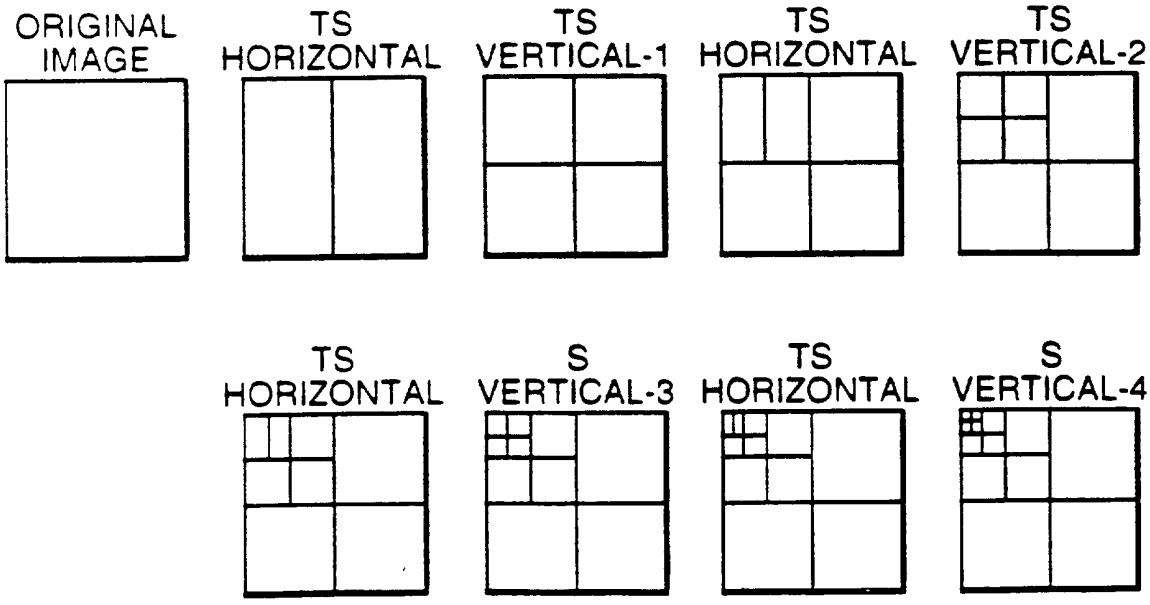

ORIGINAL IMAGE
TS HORIZONTAL
TS VERTICAL-1
TS HORIZONTAL
TS VERTICAL-2
TS HORIZONTAL
S VERTICAL-3
TS HORIZONTAL
S VERTICAL-4

Fig.19A.

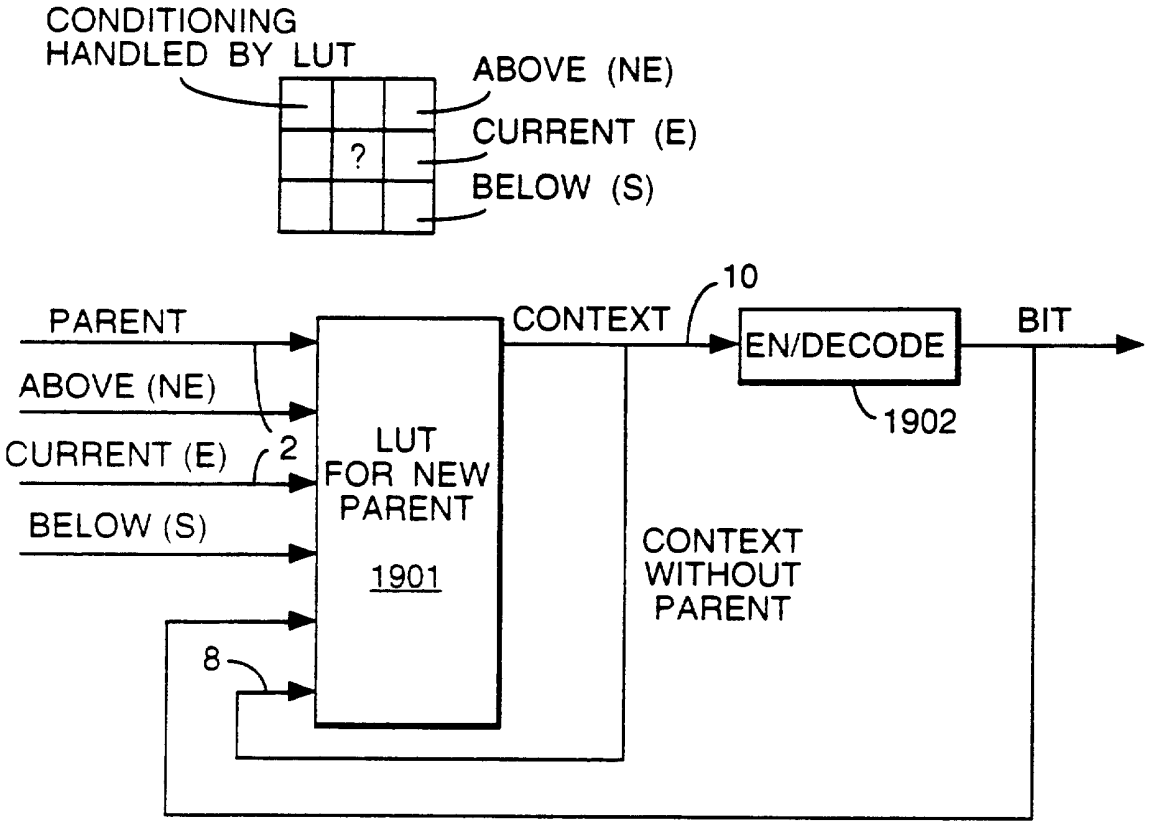


Fig.19B.

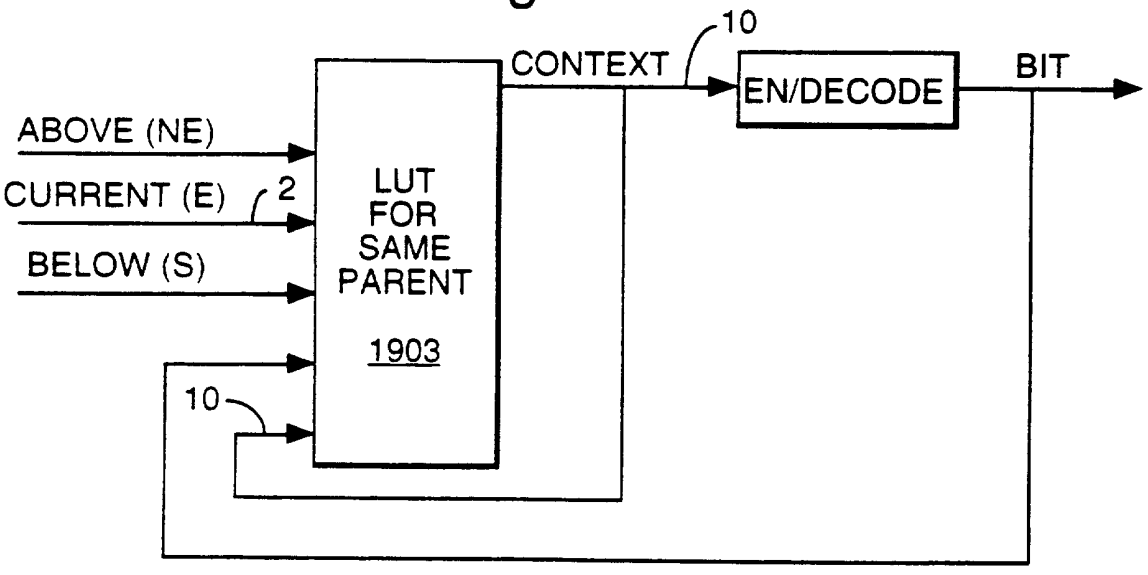


Fig.21.

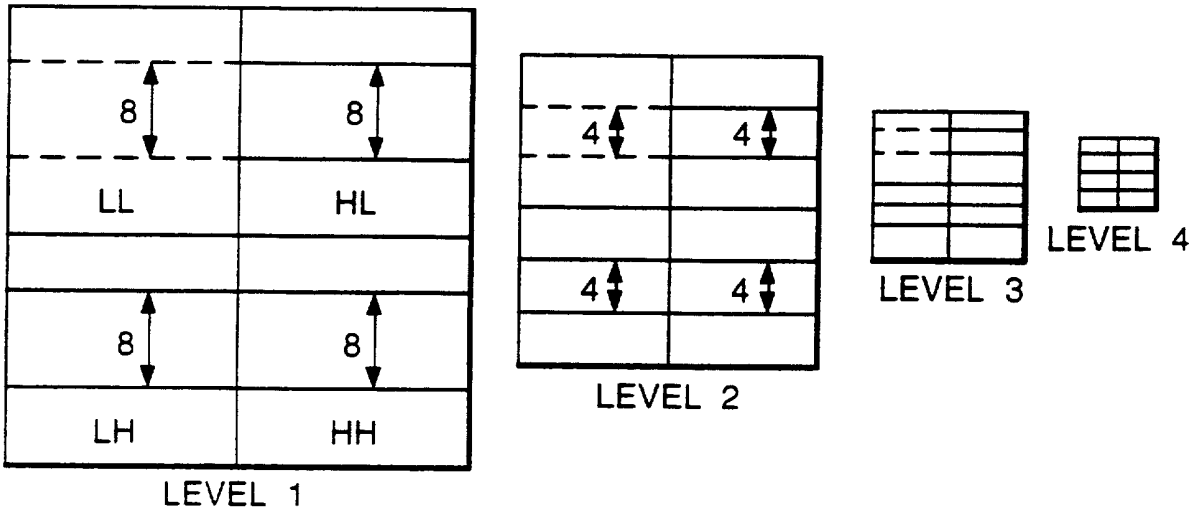

8
8
LL
HL
8
8
LH
HH
LEVEL 1
4
4
4
4
LEVEL 2
LEVEL 3
LEVEL 4

Fig.22.

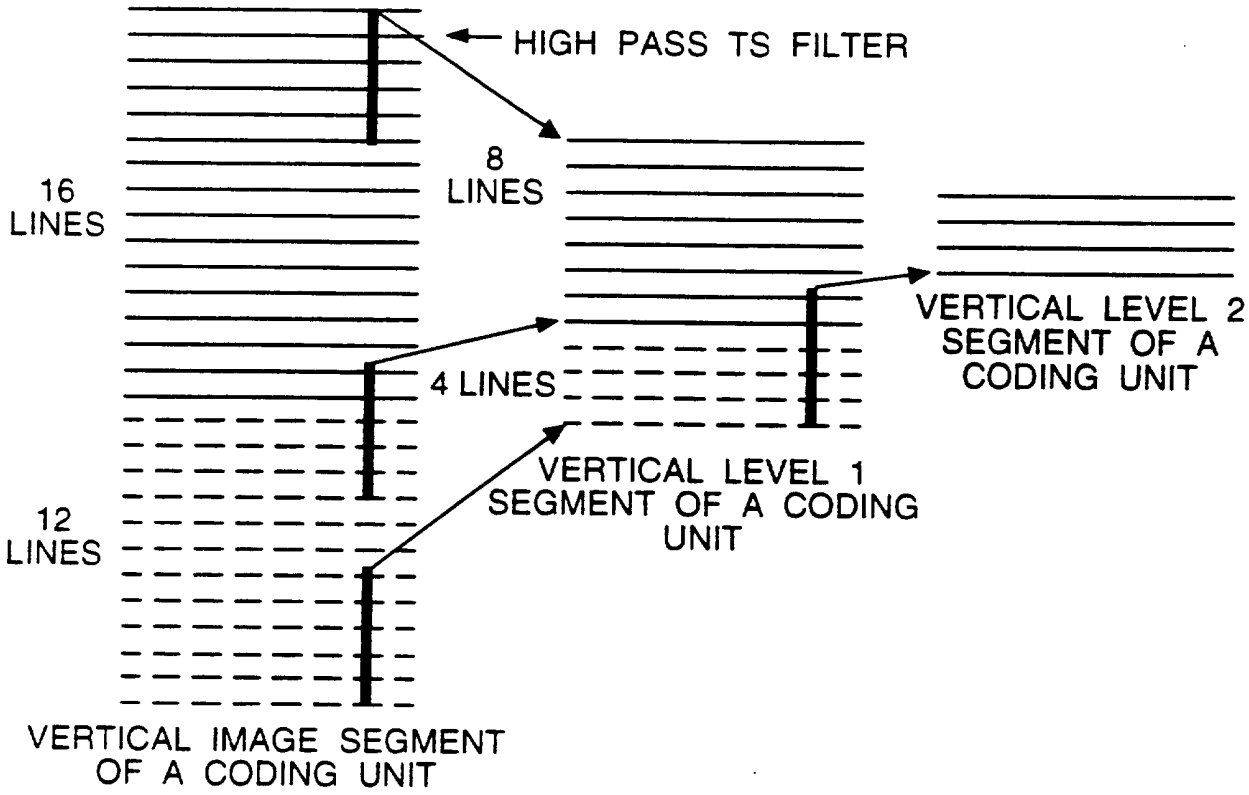

HIGH PASS TS FILTER
16
LINES
8
LINES
4 LINES
12
LINES
VERTICAL LEVEL 2
SEGMENT OF A
CODING UNIT
VERTICAL LEVEL 1
SEGMENT OF A CODING
UNIT
VERTICAL IMAGE SEGMENT
OF A CODING UNIT

Fig.23.

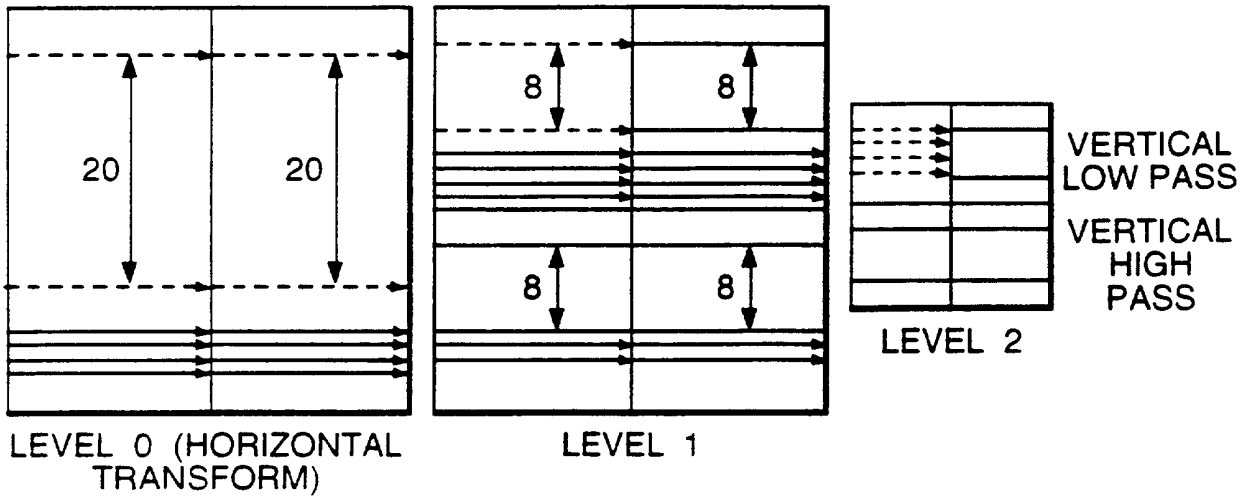


Fig.24A.

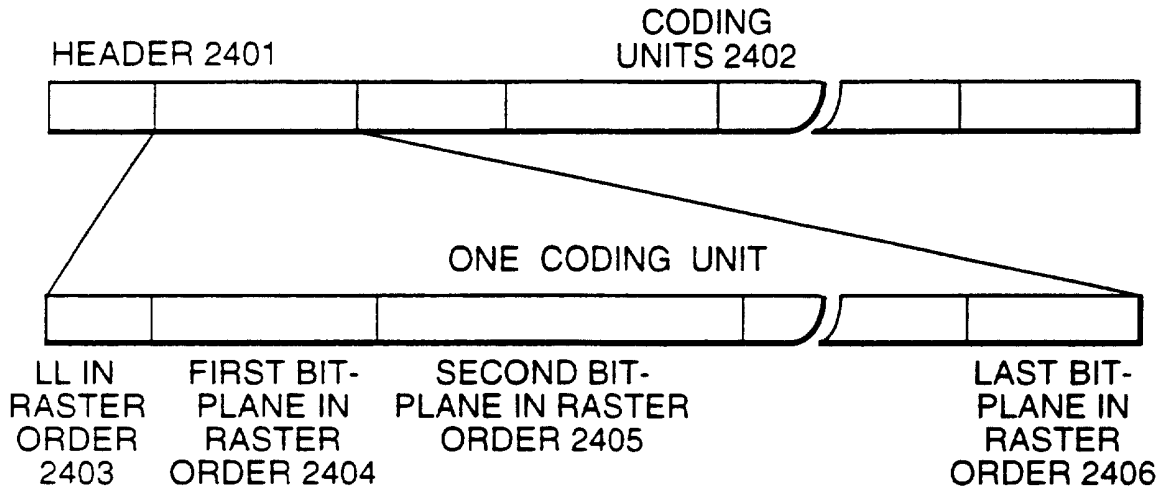


Fig.24B.

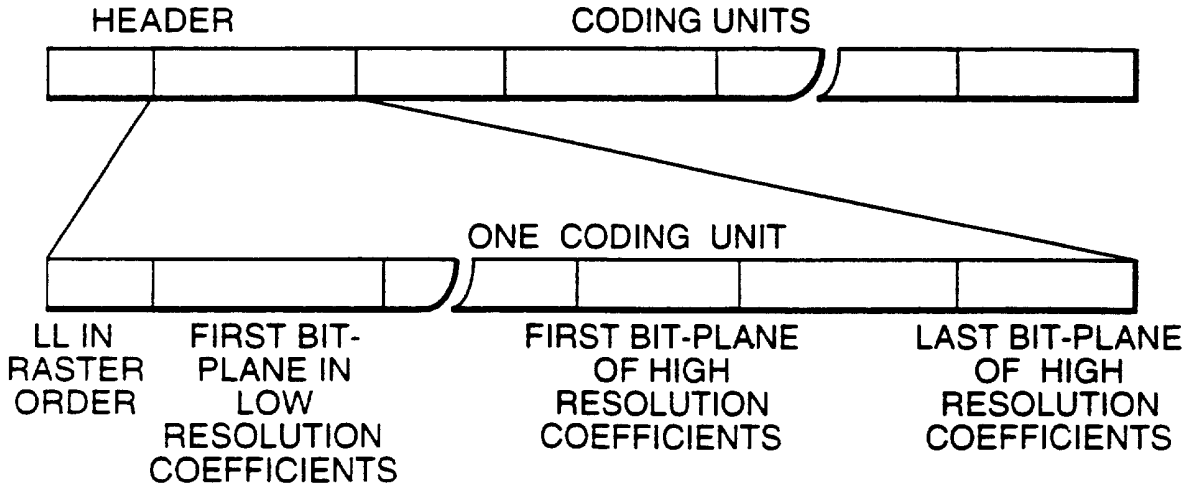


Fig.25.

NW	N	NE
W	P	E
SW	S	SE

Fig.26A.

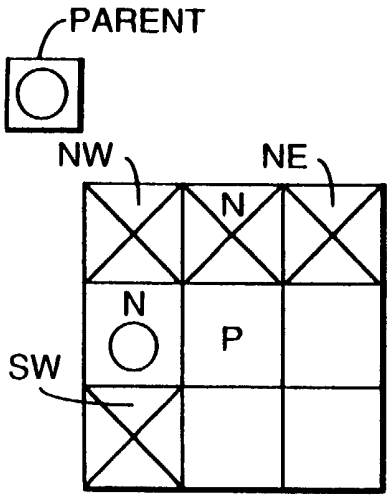


Fig.26B.

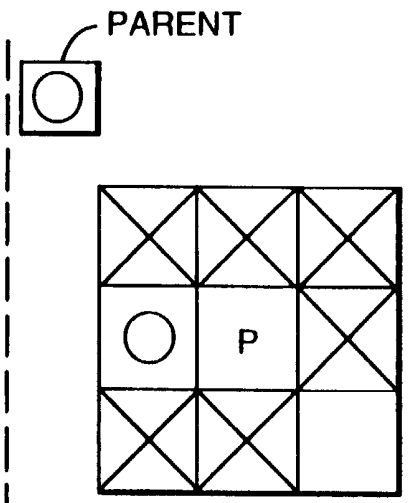


Fig.26C.

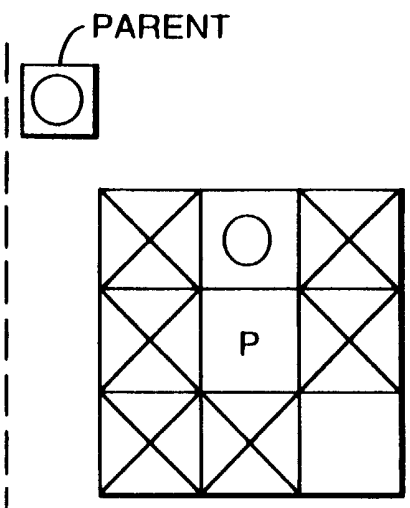
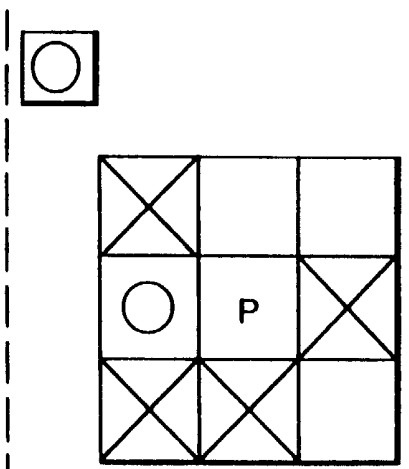


Fig.26D.



X = TAIL-ON BIT (1 BIT)

O = TAIL-ON INFORMATION (2 BITS)

Fig.27.

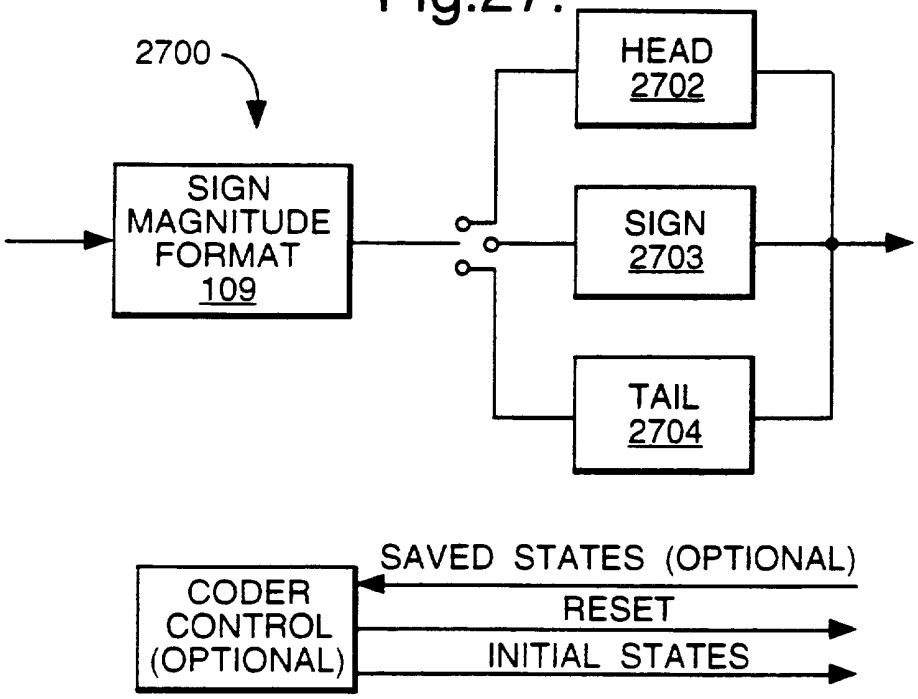


Fig.28.

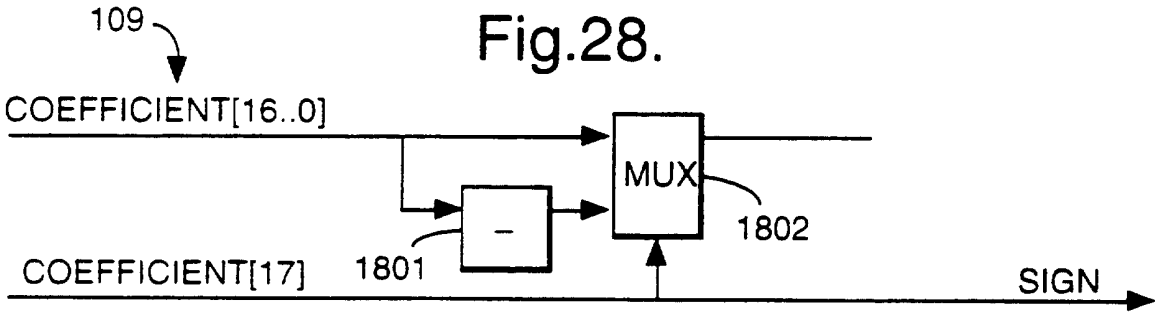


Fig.29.

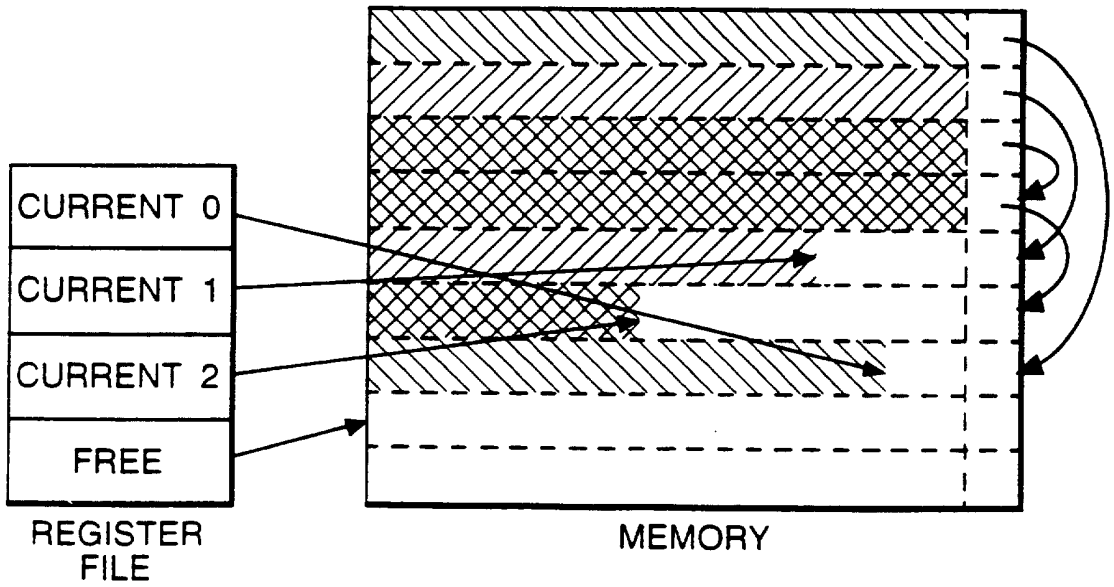


Fig.30.

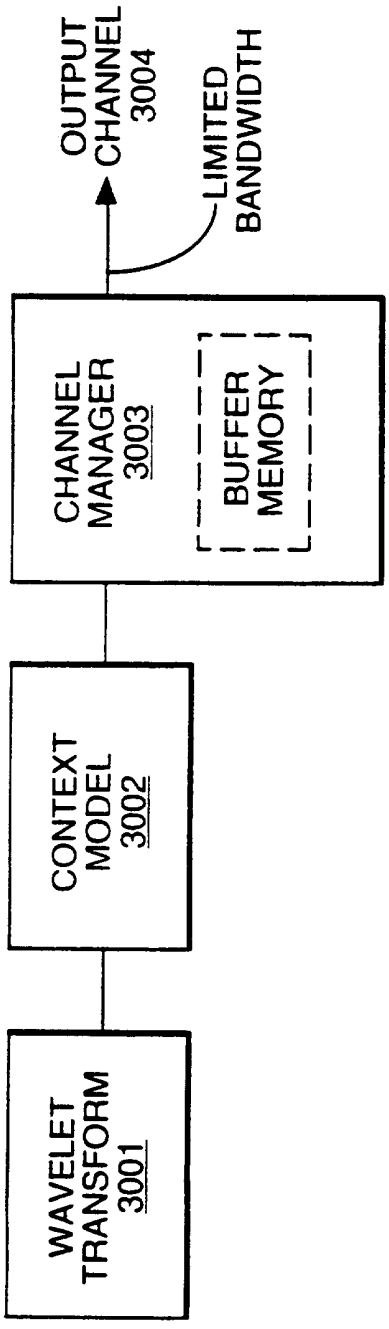

WAVELET TRANSFORM 3001
CONTEXT MODEL 3002
CHANNEL MANAGER 3003
BUFFER MEMORY
LIMITED BANDWIDTH
OUTPUT CHANNEL 3004

Fig.31.

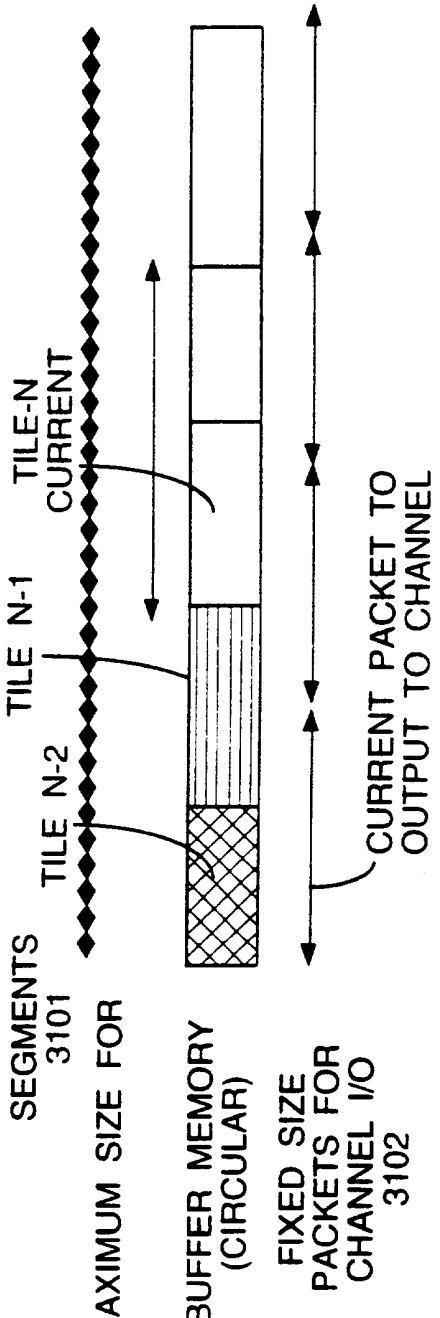

SEGMENTS 3101
MAXIMUM SIZE FOR
TILE N-2
TILE N-1
TILE-N CURRENT
BUFFER MEMORY (CIRCULAR)
FIXED SIZE PACKETS FOR CHANNEL I/O 3102
CURRENT PACKET TO OUTPUT TO CHANNEL

Fig.32.

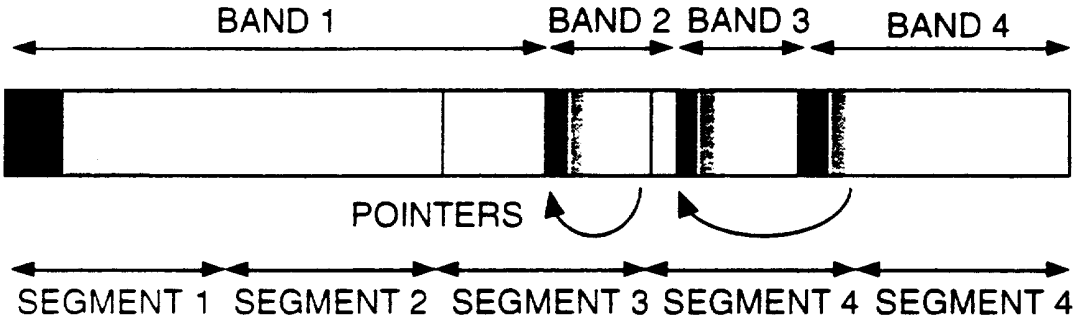

BAND 1
BAND 2
BAND 3
BAND 4
POINTERS
SEGMENT 1
SEGMENT 2
SEGMENT 3
SEGMENT 4
SEGMENT 4

Fig.33.

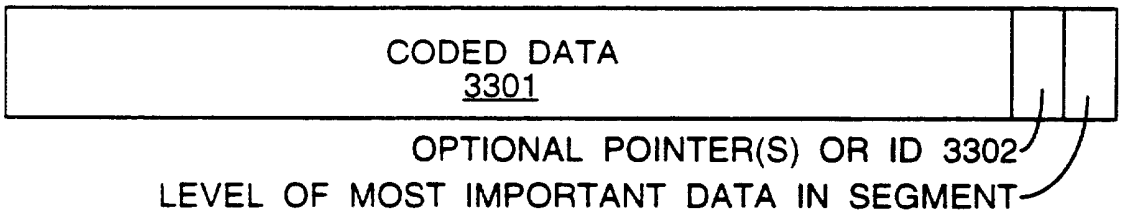

CODED DATA
3301
OPTIONAL POINTER(S) OR ID 3302
LEVEL OF MOST IMPORTANT DATA IN SEGMENT

Fig.34.

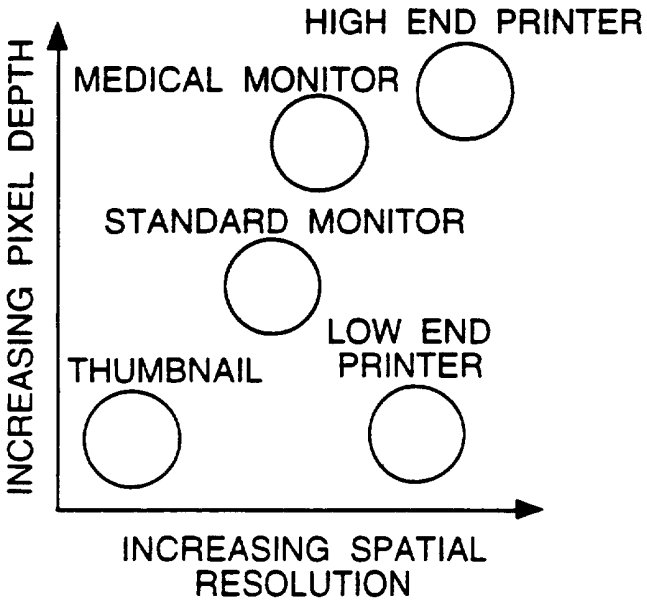
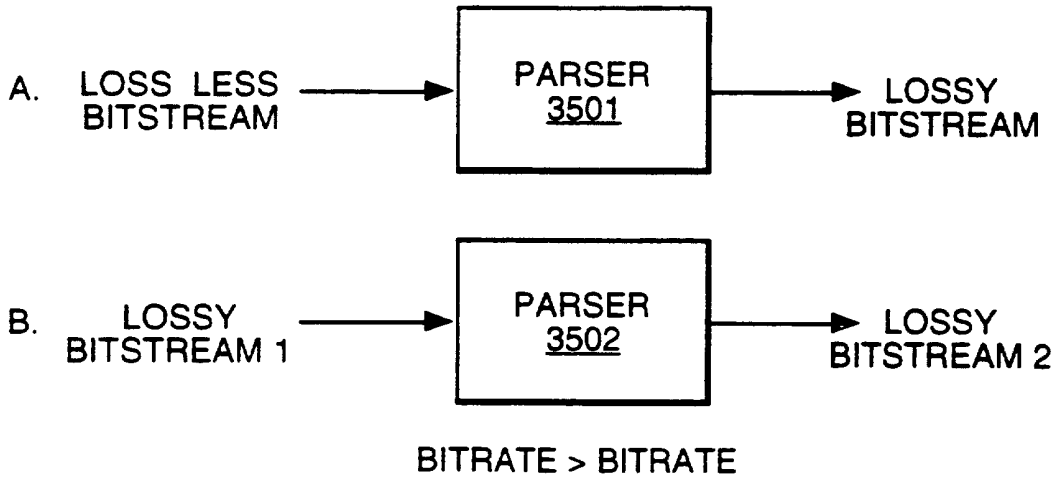

HIGH END PRINTER
MEDICAL MONITOR
STANDARD MONITOR
LOW END PRINTER
THUMBNAIL
INCREASING PIXEL DEPTH
INCREASING SPATIAL RESOLUTION

Fig.35.


A.
LOSS LESS BITSTREAM
PARSER 3501
LOSSY BITSTREAM
B.
LOSSY BITSTREAM 1
PARSER 3502
LOSSY BITSTREAM 2
BITRATE > BITRATE

METHOD USING AN EMBEDDED CODESTREAM

The present invention relates to the field of data compression and decompression systems; particularly, the present invention relates to a method using an embedded codestream.

Data compression is an extremely useful tool for storing and transmitting large amounts of data. For example, the time required to transmit an image, such as a facsimile transmission of a document, is reduced drastically when compression is used to decrease the number of bits required to recreate the image.

Many different data compression techniques exist in the prior art. Compression techniques can be divided into two broad categories, lossy coding and lossless coding. Lossy coding involves coding that results in the loss of information, such that there is no guarantee of perfect reconstruction of the original data. The goal of lossy compression is that changes to the original data are done in such a way that they are not objectionable or detectable. In lossless compression, all the information is

retained and the data is compressed in a manner which allows for perfect reconstruction.

In lossless compression, input symbols or intensity data are converted to output codewords. The input may include image, audio, one-dimensional (e.g., data changing spatially or temporally), two-dimensional (e.g., data changing in two spatial directions (or one spatial and one temporal dimension)), or multi-dimensional/multi-spectral data. If the compression is successful, the codewords are represented in fewer bits than the number of bits required for the uncoded input symbols (or intensity data). Lossless coding methods include dictionary methods of coding (e.g., Lempel-Ziv), run length encoding, enumerative coding and entropy coding. In lossless image compression, compression is based on predictions or contexts, plus coding. The JBIG standard for facsimile compression and DPCM (differential pulse code modulation - an option in the JPEG standard) for continuous-tone images are examples of lossless compression for images. In lossy compression, input symbols or intensity data are quantized prior to conversion to output codewords. Quantization is intended to preserve relevant characteristics of the data while eliminating unimportant characteristics. Prior to quantization, lossy compression system often use a transform to provide energy compaction. JPEG is an example of a lossy coding method for image data.

Recent developments in image signal processing continue to focus attention on a need for efficient and accurate forms of data compression coding. Various forms of transform or pyramidal signal processing have been proposed, including multiresolution pyramidal processing and wavelet pyramidal processing. These forms are also referred to as subband

processing and hierarchical processing. Wavelet pyramidal processing of image data is a specific type of multi-resolution pyramidal processing that may use quadrature mirror filters (QMFs) to produce subband decomposition of an original image. Note that other types of non-QMF wavelets exist. For more information on wavelet processing, see Antonini, M., et al., "Image Coding Using Wavelet Transform", IEEE Transactions on Image Processing, Vol. 1, No. 2, April 1992; Shapiro, J., "An Embedded Hierarchical Image Coder Using Zerotrees of Wavelet Coefficients", Proc. IEEE Data Compression Conference, pgs. 214-223, 1993.

One problem associated with much of prior art wavelet processing is that a large memory is required to store all of the data while it is being processed. In other words, in performing wavelet processing, all of the data must be examined before encoding is performed on the data. In such a case, there is no data output until at least one full pass has been made through all of the data. In fact, wavelet processing typically involves multiple passes through the data. Because of this, a large memory is often required. It is desirable to utilize wavelet processing, while avoiding the requirement of a large memory. Furthermore, it is desirable to perform wavelet processing using only a single pass through the data.

Many wavelet or subband transform implementations require filters in a particular canonical form. For example, low and high-pass filters must be the same length, the sum of the squares of the coefficients must be one, the high-pass filter must be the time and frequency reverse of the low-pass filter, etc. (See U.S. Patent No. 5,014,134 issued May 1991 to Lawton et al.). It is desirable to allow a wider class of filters. That is, it is desirable to provide wavelet or subband transform implementations that

use low and high-pass filters that are not the same length, the sum of the squares of the coefficients need not be one, the high-pass filter need not be the time frequency reverse of the low-pass filter, etc.

A compression and decompression system is described. The compression system includes an encoder that encodes input data into a compressed data stream. The entropy coder said coder comprises a reversible wavelet filter, an ordering and modeling mechanism and a binary entropy coder. The reversible wavelet filter transforms input data into coefficients using a pyramidal decomposition. The ordering and modeling mechanism generates an embedded codestream from the coefficients by ordering the coefficients and binary values within the coefficients. The binary entropy coder operable to binary entropy code the embedded codestream to produce the compressed data stream.

The present invention provides: a method of using an embedded codestream comprising: identifying a target device to receive data in the embedded codestream;

decoding each bit-plane to provide data to the target device, said step of decoding comprising the step of truncating each bit-plane in the embedded codestream for data necessary to support the target device.

The present invention will be understood more fully from the detailed description given below and from the accompanying drawings, in which:-

Figure 1 is a block diagram of one example of an

encoding portion of a coding system described for reference.

Figure 2 is a block diagram of one example of the coefficient data ordering and modeling of the reference system.

Figure 3A is a block diagram of a wavelet analysis/synthesis system.

Figure 3B illustrates forward and reverse representations of transform systems for filtering with non-overlapped minimal length reversible filters.

Figure 4 is a block diagram illustrating alternative examples of a 2-D reversible filter.

Figures 5 illustrate results of performing a four level decomposition.

Figure 6 illustrates the parental relationship between two consecutive levels.

Figure 7 is a block diagram of a three-level pyramidal transform.

Figure 8 is a block diagram of a two-dimensional, two level transform.

Figure 9 is a block diagram illustrating one-dimensional filters performing a multi-resolution decompression.

Figure 10 is a block diagram of a system using reversible wavelets.

Figure 11 are block diagrams of enhancement and analysis systems using reversible wavelets.

Figure 12A illustrates coefficient size in the reference system.

Figure 12B is one embodiment of the multipliers for the frequency band used for coefficient alignment in the reference system.

Figure 13 is a flow chart of one example of the encoding process of the reference system.

Figure 14 is a flow chart of one example of the decoding process of the reference system.

Figure 15 is a flow chart of the modeling process of the reference system.

Figure 16 is one example of the forward wavelet filter

of the reference system.

Figure 17 is a block diagram of one example of a reverse wavelet filter of the reference system.

Figure 18 illustrates the coefficient range of various transforms.

Figure 19A and 19B illustrate two examples of context models using look-up tables.

Figure 20 illustrates one example of a wavelet decomposition stage.

Figure 21 illustrates one coding unit.

Figure 22 illustrates vertical passes with the TS-transform.

Figure 23 illustrates buffering and coefficient computation.

Figure 24A illustrates one embodiment of a codestream configuration.

Figure 24B illustrates one embodiment of a codestream configuration for a low resolution target.

Figure 25 illustrates the neighboring relationship among coefficients (or pixels).

Figures 26A-D illustrate examples of context models.

Figure 27 is a block diagram of one example of the context model of the reference system.

Figure 28 is a block diagram of one example of the sign/magnitude unit of the reference system.

Figure 29 illustrates the dynamic allocation of coded data memory for one pass operation.

Figure 30 illustrates one example of a channel manager.

Figure 31 illustrates memory utilization in the reference system.

Figure 32 illustrates a bitstream in the reference system.

Figure 33 illustrates the structure of a segment.

Figure 34 illustrates target devices versus a parameter space.

Figures 35A and 35B illustrate various embodiments of the parser of the present invention.

A method and apparatus for compression and decompression is described.

Some portions of the detailed descriptions which follow are presented in terms of algorithms and symbolic representations of operations on data bits within a computer memory. These algorithmic descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. An algorithm is here, and generally, conceived to be a self-consistent sequence of steps leading to a desired result. The steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like.

It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically

stated otherwise as apparent from the following discussions, it is appreciated that throughout the present invention, discussions utilizing terms such as "processing" or "computing" or "calculating" or "determining" or "displaying" or the like, refer to the action and processes of a computer system, or similar electronic computing device, that manipulates and transforms data represented as physical (electronic) quantities within the computer system's registers and memories into other data similarly represented as physical quantities within the computer system memories or registers or other such information storage, transmission or display devices.

The apparatus for performing the operations herein may be specially constructed for the required purposes, or it may comprise a general purpose computer selectively activated or reconfigured by a computer program stored in the computer. The algorithms and displays presented herein are not inherently related to any particular computer or other apparatus. Various general purpose machines may be used with programs in accordance with the teachings herein, or it may prove convenient to construct more specialized apparatus to perform the required method steps. The required structure for a variety of these machines will appear from the description below. In addition, the reference system is not described with reference to any particular programming language. It will be appreciated that a variety of programming languages may be used to implement the teachings as described herein.

The following terms are used in the description that follows. A definition has been included for these various terms. However, the

definition provided should not be considered limiting to the extent that the terms are known in the art. These definitions are provided to help in the understanding of the present invention.

bit-significance:	A number representation, similar to sign magnitude, with head bits, followed by the sign bit, followed by tail bits, if any. The embedding encodes in bit-plane order with respect to this representation.
coding unit:	Unit of coefficients that are coded together and can be in arbitrary order. In one embodiment, a coding unit comprises one or more trees arranged in a rectangle. A coding unit may consist of an entire image, set of images or other data set. The coding unit has a significant impact on the buffer size needed for computing a transform. Also, in one embodiment, no contents can be derived from coefficients outside the current coding unit. However, the entropy codes may be reset within a coding unit or after many coding units. The coding unit is not necessarily randomly addressable.
context model:	Available information relative to the current bit to be coded that give historically learned information about the current bit. This enables conditional probability estimation for entropy coding.

trees: The coefficients, and the pixels, that are related to a single coefficient in the LL of the highest level wavelet decomposition. The number of coefficients is a number of the number of levels.

band: The coefficients, and the pixels, that are related to a single row or line of coefficients in the LL of the highest level wavelet decomposition for two-dimensional data. Bands are similarly defined for data of other dimensions.

decomposition level: A location in the wavelet decomposition pyramid.

embedded quantization: Quantization that is implied by the codestream. For example, if the importance levels are placed in order, from the most important to the least, then quantization is performed by simple truncation of the codestream. The same functionality is available with tags, markers, pointers, or other signaling.

entropy coder: A device that encodes a current bit based on its context. The context allows probability estimation for the best representation of the current bit (or multiple bits).

fixed-rate: An application or system that maintains a certain pixel rate and has a limited bandwidth channel. This requires achieving local average compression rather than a global average compression. Example: MPEG.

fixed-size: An application or system that has a limited size buffer. In such a case, a global average compression is achieved, e.g., a print buffer. (An application can be both fixed-rate and fixed-size or either.)

fixed-length: A system that converts a specific block of data to a specific block of compressed data, e.g., BTC. Fixed-length codes serve fixed-rate and fixed-size applications; however, the rate-distortion performance is often poor compared with variable rate systems.

Horizon context model: A context model for use with an entropy coder defined herein

head: In bit-significance representation, the head bits are the magnitude bits from the most significant up to and including the first non-zero bit

overlapped transform: A transform where a single source sample point contributes to multiple coefficients of the same frequency. Examples include many wavelets and the Lapped Orthogonal Transform.

progressive: A codestream that is ordered, such that a coherent decompressed result is available from part of the coded data which can be refined with more data. A codestream that is ordered with deepening bit-planes of data; in this case, it usually refers to wavelet coefficient data.

pyramidal: Succession of resolutions where each lower resolution is a linear factor of two greater (a factor of four in area).

reversible transform: An efficient transform implemented with integer arithmetic that has exact reconstruction.

S-transform: A specific reversible wavelet filter pair with a 2-tap low pass and a 2-tap high pass filter.

tail: In bit-significance representation, the tail bits are the magnitude bits with less significance than the most significant non-zero bit.

tail information: In one example, four states possible for a coefficient represented in bit-significance representation. It is a function of the coefficient and the current bit-plane, and is used for the Horizon context model.

tail-on: In one example, two states depending on whether the tail information state is zero or non-zero. It is used for the Horizon context model.

TS-transform: Two-Six transform, a specific wavelet filter pair with a 2-tap low pass and a 6-tap high pass filter.

unified lossless/lossy: The same compression system provides a coded data stream capable of lossless or lossy reconstruction. As will be described below, this codestream is capable of both without settings or instructions to the encoder.

visual importance levels: By definition of the specific system, the input data (pixel data, coefficients, error signals, etc.) is divided logically into groups with the same visual impact. For example, the most significant bit-plane, or planes, is probably more visually important than lessor planes. Also low frequency information is generally more important than high frequency. Most working definitions of "visual significance" are with respect to some error metric. Better visual metrics, however, could be incorporated in the system definition of visual importance. Alternate data types have alternate importance levels, for example, audio data has audio importance levels.

wavelet filters: The high and low pass synthesis and analysis filters used in wavelet transform.

wavelet transform: A transformation with both "frequency" and "time (or space)" domain constraints. In a described example, it is a transform consisting of a high pass filter and a low pass filter. The resulting coefficients are decimated by two (critically filtered) and the filters are applied to the low pass coefficients.

The reference compression/decompression system has an encoding portion and a decoding portion. The encoding portion is responsible for encoding input data to create compressed data, while the decoding portion is responsible for decoding previously encoded data to produce a reconstructed version of the original input data. The input data may comprise a variety of data types, such as image (still or video), audio, etc. In one example, the data is digital signal data; however, analog data digitized, text data formats, and other formats are possible. The source of the data may be a memory or channel for the encoding portion and/or the decoding portion.

In the reference system, elements of the encoding portion and/or the decoding portion may be implemented in hardware or software, such as that used on a computer system. The reference system provides a lossless compression/decompression system, but may also be configured to perform lossy compression/decompression.

Figure 1 is a block diagram of one example of the encoding portion of the system. Note the decoding portion of the system operates in reverse order, along with the data flow. Referring to Figure 1, input image data 101 is received by wavelet transform block 102. The output of wavelet transform block 102 is coupled to coefficient data ordering and modeling block 103. In response to the output from wavelet transform block 102, the ordering/modeling block 103 produces at least one bit stream that is received by an

entropy coder 104. In response to the input from ordering/modeling block 103, entropy coder 104 produces code stream 107.

In one example, the ordering/modeling block 103 comprises a sign/magnitude formatting unit 201 and a joint space/frequency context model 202, such as shown in Figure 2. In one example, the joint space/frequency context model 202 comprises a horizon context model, as is described below. The input of the sign/magnitude unit 201 is coupled to the output of the wavelet transform coding block 102. The output of sign/magnitude unit 201 is coupled to joint space/frequency modeling block 202. The output of JSF context model 202 is coupled to the input of entropy coder 104 which produces the output code stream 107.

Referring back to Figure 1, the image data 101 is received and transform coded using reversible wavelets in wavelet transform block 102, as defined below, to produce a series of coefficients representing a multi-resolution decomposition of the image. The reversible wavelet transforms are not computationally complicated. The transforms may be performed in software or hardware with no systematic error. Furthermore, the wavelets are excellent for energy compaction and compression performance. These coefficients are received by the ordering/modeling block 103.

The ordering/modeling block 103 provides coefficient ordering and modeling. The coefficient ordering provides an embedded data stream. The embedded data stream allows a

resulting codestream to be quantized at encode time, transmission time, or decode time. In one example, ordering/modeling block 103 orders and converts the coefficients into sign-magnitude format and, based on their significance (as described below later), the formatted coefficients are subjected to the embedded modeling method. In one example, the formatted coefficients are subjected to joint spatial/frequency modeling.

The results of ordering and modeling comprise decisions (or symbols) to be coded by the entropy coder. In one example, all decisions are sent to a single coder. In another example, decisions are labeled by significance, and decisions for each significance level are processed by different (physical or virtual) multiple coders.

Referring back to Figure 2, the bit stream(s) resulting from JSF context model block 201 are encoded in order of significance using entropy coder 104. In one example, entropy coder 104 comprises one or more binary entropy coders.

Wavelet Decomposition

The reference system initially performs decomposition of an image (in the form of image data) or another data signal using reversible wavelets. A reversible wavelet transform comprises an implementation of an exact-reconstruction system in integer arithmetic, such that a signal with integer coefficients can be losslessly recovered. By using reversible wavelets, it is possible to provide

lossless compression with finite precision arithmetic. The results generated by applying the reversible wavelet transform to the image data are a series of coefficients.

The reversible wavelet transform may be implemented using a set of filters. In one example, the filters are a two-tap low-pass filter and a six-tap high-pass filter. In one example, these filters are implemented using only addition and subtraction.

operations (plus hardwired bit shifting). Also, the high-pass filter generates its output using the results of the low-pass filter. The resulting high-pass coefficients are only a few bits greater than the pixel depth and the low-pass coefficients are the same as the pixel depth. Because only the low-pass coefficients are repeatedly filtered in a pyramidal decomposition, coefficient resolution is not increased in multi-level decompositions.

In alternate examples, the low-pass filter output coefficients could increase in size, instead of high-pass filter output coefficients.

A wavelet transform system is defined by a pair of FIR analysis filters $h_0(n)$, $h_1(n)$, and a pair of FIR synthesis filters $g_0(n)$, $g_1(n)$. In the present invention, h_0 and g_0 are the low-pass filters and h_1 and g_1 are the high-pass filters. A block diagram of the wavelet system is shown in Figure 3A. Referring to Figure 3A, for an input signal, $x(n)$, the analysis filters h_0 and h_1 are applied and the outputs are decimated by 2 (critically subsampled) to generate the transformed signals $y_0(n)$ and $y_1(n)$, referred to herein as low-passed (smooth) and high-passed (detail) coefficients respectively. The analysis filters and their corresponding decimation, or subsampling, blocks form the analysis portion of the wavelet transform system. The coder/decoder contain all the processing logic and routines performed in the transformed domain (e.g., prediction, quantization, coding, etc.). The wavelet system shown in Figure 3A also includes a synthesis portion in which the transformed signals are upsampled by 2 (e.g., a zero is inserted after every term) and then passed through synthesis filters, $g_0(n)$ and $g_1(n)$. The low-passed (smooth) coefficients $y_0(n)$ are passed through the low-pass synthesis filter g_0 and the high-passed (detail)

coefficients $y_1(n)$ are passed through the high-passed filter g_1. The output of filters $g_0(n)$ and $g_1(n)$ are combined to produce $\hat{x}(n)$.

While downsampling and upsampling are performed in some embodiments, in other embodiments, filters are used such that computations which are unneeded due to downsampling and upsampling are not performed.

The wavelet system may be described in terms of the Z-transform, where $X(Z)$, $\hat{X}(Z)$ are the input and output signals respectively, $Y_0(Z)$, $Y_1(Z)$ are the low-passed and high-passed transformed signals, $H_0(Z)$, $H_1(Z)$ the low-pass and the high-pass analysis filters and finally $G_0(Z)$, $G_1(Z)$ are the low-pass and the high-pass synthesis filters. If there is no alteration or quantization in the transform domain, the output $\hat{X}(Z)$ in Figure 3, is given by

$$\hat{X}(Z) = \frac{1}{2}[H_0(Z)G_0(Z)+H_1(Z)G_1(Z)]X(Z)+ \frac{1}{2}[H_0(-Z)G_0(Z)+H_1(-Z)G_1(Z)]X(-Z).$$

In the reference system, the second term of $\hat{X}(Z)$, referred to as the "aliasing" term, is cancelled because the synthesis filters are defined to be the quadrature mirror of the analysis filters, i.e.,

$$\begin{cases} G_0(Z) = H_1(-Z) \\ G_1(Z) = -H_0(-Z) \end{cases}$$

In terms of the filter coefficients,

$$\begin{cases} g_0(n) = (-1)^n h_1(n) \\ g_1(n) = -(-1)^n h_0(n) \end{cases}$$

Therefore, for a quadrature mirror filter pairs, after substitution, the output is:

$$\hat{X}(z)=\frac{1}{2}[H_0(Z)H_1(-Z)-H_1(Z)H_0(-Z)]X(Z).$$

Thus, in the quadrature mirror system of the reference system, the output is defined in terms of the analysis filters only. The wavelet transform is applied recursively to the transformed signals in that the outputs generated by the filters are used as inputs, directly or indirectly, into the filters. In the described system, only the low-passed transformed component $y_0(n)$ is recursively transformed such that the system is pyramidal. An example of such a pyramidal system is shown in Figure 6.

The Z transform is a convenient notation for expressing the operation of hardware and/or software on data. Multiplication by Z^{-m} models a m clock cycle delay in hardware, and an array access to the mth previous element in software. Such hardware implementations include memory, pipestages, shifters, registers, etc.

In the reference system, the signals, $x(n)$ and $\hat{x}(n)$, are identical up to a multiplicative constant and a delay term, i.e. in terms of the Z-transform,

$$\hat{X}(Z)=cZ^{-m}X(Z).$$

This is called an exact reconstruction system. Thus, the wavelet transform initially applied to the input data is exactly reconstructible.

One example of the reference system using the Hadamard Transform is an exact reconstruction system, which in normalized form has the following representation in the Z-domain:

$$\begin{cases} H_0(Z) = \frac{1}{\sqrt{2}}(1+Z^{-1}) \\ H_1(Z) = \frac{1}{\sqrt{2}}(1-Z^{-1}) \end{cases}$$

After substitution, the output is

$$\hat{X}(Z) = Z^{-1}X(Z),$$

which is clearly an exact-reconstruction. For more information on the Hadamard Transform, see Anil K. Jain, Fundamentals of Image Processing, pg. 155.

A reversible version of the Hadamard Transform is referred to herein as the S-transform. For more information on S-transform, see Said, A. and Pearlman, W. "Reversible Image Compression via Multiresolution Representation and Predictive Coding," Dept. of Electrical, Computer and Systems Engineering, Renssealaer Polytechnic Institute, Troy, NY 1993. Since the Hadamard Transform is an exact reconstruction transform, the following unnormalized version (which differs from the Hadamard Transform by constant factors) is also an exact reconstruction transform:

$$\begin{cases} h_0(Z) = \frac{1}{2}(1+Z^{-1}) \\ h_1(Z) = 1-Z^{-1} \end{cases}$$

Given the samples of the input signal as x_0, x_1, the S-transform is a reversible implementation of this system as,

$$\begin{cases} y_0(0) = \lfloor (x(0)+x(1))/2 \rfloor \\ y_1(0) = x(0)-x(1) \end{cases}$$

The S-transform may be defined by the outputs with a generic index, n, as follows:

$$\begin{cases} s(n) = \left\lfloor \dfrac{x(2n)+x(2n+1)}{2} \right\rfloor \\ d(n) = x(2n)-X(2n+1) \end{cases}$$

Note that the factor of two in the transform coefficients addressing is the result of an implied subsampling by two. This transform is reversible and the inverse is:

$$\begin{cases} x(2n) = s(n)+\left\lfloor \dfrac{d(n)+1}{2} \right\rfloor \\ x(2n+1) = s(n)-\left\lfloor \dfrac{d(n)}{2} \right\rfloor \end{cases}$$

The notation $\lfloor . \rfloor$ means to round down or truncate and is sometimes referred to as the floor function. Similarly, the ceiling function $\lceil . \rceil$ means round up to the nearest integer.

The proof that this implementation is reversible follows from the fact that the only information lost in the approximation is the least significant bit of $x(0)+x(1)$. But since the least significant bits of $x(0)+x(1)$ and $x(0)-x(1)$ are identical, this can be recovered from the high-pass output $y_1(0)$. In other words,

$$\begin{cases} x(0) = y_0(0) + \lfloor (y_1(0)+1)/2 \rfloor \\ x(1) = y_0(0) - \lfloor (y_1(0)-1)/2 \rfloor \end{cases}$$

The S-transform is a non-overlapping transform using minimal length reversible filters. Minimal length filters comprise a pair of filters, where both filters have two taps. Minimal length transforms do not provide good energy compaction. Minimal length filters implement a non-overlapped transform because the length of the filters is equal to the number of filters. Overlapped transforms use at least one filter which has length greater than the number of filters. Overlapped transforms using longer (non-minimal length) filters can provide better energy compaction. **The reference system provides non-minimal length reversible filters which permits an overlapped transform.**

Another example of an exact-reconstruction system comprises the Two/Six (TS)-Transform which has the Z-domain definition,

$$\begin{cases} H_0(Z) = \frac{1}{\sqrt{2}}(1+Z^{-1}) \\ H_1(Z) = \frac{1}{8\sqrt{2}}(-1-Z^{-1}+8Z^{-2}-8Z^{-3}+Z^{-4}+Z^{-5}) \end{cases}$$

After substitution, the output is

$$\hat{X}(Z) = 2Z^{-3}X(Z),$$

which is an exact-reconstruction transform.

The rational unnormalized version of the TS-transform comprises:

$$\begin{cases} h_0(Z) = \frac{1}{2}(1+Z^{-1}) \\ h_1(Z) = \frac{1}{8}(-1-Z^{-1}+8Z^{-2}-8Z^{-3}+Z^{-4}+Z^{-5}) \end{cases}$$

If $x(0), x(1), \ldots x(5)$ are six samples of the signal, then the first three low-passed coefficients $y_0(0), y_0(1), y_0(2)$ and the first high-passed coefficient $y_1(0)$ are given by:

$$\begin{cases} y_0(0) = \lfloor (x(0)+x(1))/2 \rfloor \\ y_0(1) = \lfloor (x(2)+x(3))/2 \rfloor \\ y_0(2) = \lfloor (x(4)+x(5))/2 \rfloor \end{cases}$$

$$y_1(0) = \left\lfloor \left(-(x(0)+x(1))\right) + 8(x(2)-x(3)) + (x(4)+x(5))/8 \right\rfloor .$$

However, the straight forward implementation of the rational unnormalized version of the TS-transform is not reversible. The following example shows that the implementation is non-reversible locally. A longer sequence can be constructed as an example for the global case. Since $-(x(0)+x(1))+(x(4)+x(5)) \neq -y_0(0)+y_0(2)$ because of rounding to compute $y_0(0)$ and $y_0(2)$, this transform is not reversible using local information.

For example, if $x(0)=1, x(1)=1, x(2)=3, x(3)=1, x(4)=1, x(5)=1$, then

$$y_0(0) = \lfloor (1+1)/2 \rfloor = 1$$
$$y_0(1) = \lfloor (3+1)/2 \rfloor = 2$$
$$y_0(2) = \lfloor (1+1)/2 \rfloor = 1$$
$$y_1(0) = \lfloor [-(1+1)+8(3-1)+(1+1)]/8 \rfloor = \lfloor (-2+16+2)/8 \rfloor = 2$$

and if $x(0)=1, x(1)=2, x(2)=4, x(3)=1, x(4)=1, x(5)=1$, then

$$y_0(0) = \lfloor (1+2)/2 \rfloor = 1$$
$$y_0(1) = \lfloor (4+1)/2 \rfloor = 2$$
$$y_0(2) = \lfloor (1+1)/2 \rfloor = 1$$
$$y_1(0) = \lfloor [-(1+2)+8(4-1)+(1+1)] \rfloor /8 = \lfloor (-3+24+2)/8 \rfloor = \lfloor 23/8 \rfloor = 2$$

Since $y_0(0)$, $y_0(1)$, $y_0(2)$ and $y_1(0)$ are the same for two different sets of inputs $x(0) \ldots x(5)$, the transform is not reversible, since given $y_0(0), \ldots$ $y_1(0)$ it cannot be determined from this local information which of the two sets were input. (Note that it can be proved that the transform is not reversible using global information from all coefficients.)

Now consider a reversible TS-transform, which is referred to herein as an RTS-transform, which provides a different high-pass filtering operation.

If $x(0)$, $x(1)$, $x(2)$, $x(3)$, $x(4)$, $x(5)$ are 6 samples of the signal, then the first three low-passed coefficients $y_0(0)$, $y_0(1)$, $y_0(2)$ and the first high-passed coefficient $y_1(0)$ are given by,

$$\begin{cases} y_0(0) = \lfloor (x(0)+x(1))/2 \rfloor \\ y_0(1) = \lfloor (x(2)+x(3))/2 \rfloor \\ y_0(2) = \lfloor (x(4)+x(5))/2 \rfloor \end{cases}$$

$$\begin{aligned} y_1(0) &= \lfloor (-\lfloor (x(0)+x(1))/2 \rfloor + 4(x(2)-x(3)) + \lfloor (x(4)+x(5))/2 \rfloor + 2)/4 \rfloor \\ &= \lfloor (-y_0(0) + 4(x(2)-x(3)) + y_0(2) + 2)/4 \rfloor . \end{aligned}$$

Since

$$x(2)-x(3) = y_1(0) - \lfloor -(y_0(0) - y_0(2) + 2)/4 \rfloor$$

then $x(2)$-$x(3)$ is completely known. With $y_0(1) = \lfloor (x(2)+x(3))/2 \rfloor$ and $x(2)-x(3)$ and $x(2)$-$x(3)$ defined above, $x(2)$ and $x(3)$ may be recovered because the least significant bits of $x(0)+(1)$ and $x(0)$-$x(1)$ are identical.

Specifically, let

$$d(0) = x(2)-x(3) = y_1(0) - \lfloor (-Y_0(0) + y_0(2) + 2)/4 \rfloor$$

$$x(2) = y_0(1) + \lfloor (d(0)+1)/2 \rfloor$$

$$x(3) = y_0(1) + \lceil (d(0)-1)/2 \rceil$$

In one example of the RTS-transform, as well as that of the S-transform, a divide-by-eight is implemented as a divide-by-two and then a divide-by-four in order to provide additional accuracy. Note that mathematically the equation

$$\frac{1}{8}\left(-1-Z^{-1}+8Z^{-2}-8Z^{-3}+Z^{-4}+Z^{-5}+4\right)$$

and the equation

$$\frac{1}{4}\left(\frac{1}{2}\left(-1-Z^{-1}\right)+4\left(Z^{-2}-Z^{-3}\right)+\frac{1}{2}\left(Z^{-4}+Z^{-5}\right)+2\right)$$

are the same when performed with infinite precision arithmetic. The reason the second equation represents a reversible filter is apparent when physically implemented with integer arithmetic. Exemplary hardware implementations of the low-pass filter and the high-pass filter are described in conjunction with Figures 16 and 17.

Note that in both the S-transform and the RTS-transform, the low-pass filter is implemented so that the range of the input signal $x(n)$ is the same as the output signal $y_0(n)$. For example, if the signal is an 8-bit image, the output of the low-pass filter is also 8 bits. This is an important property for a pyramidal system where the low-pass filter is successively applied because in prior art systems the range of the output signal is greater than that of the input signal, thereby making successive applications of the filter difficult. In addition, the low-pass filter has only two taps which makes it a non-overlapping filter. This property is important for the hardware implementation, as is described below later.

In a more generic format, the reversible TS-transform is defined by the expression of the two outputs of a low-pass and a high-pass filter:

$$\begin{cases} s(n) = \left\lfloor \dfrac{x(2n) + x(2n+1)}{2} \right\rfloor \\ d(n) = \left\lfloor - \dfrac{\left\lfloor \dfrac{x(2n) + x(2n+1)}{2} \right\rfloor + 4(x(2n+2) - x(2n+3)) + \left\lfloor \dfrac{x(2n+4) + x(2n+5))}{2} \right\rfloor}{4} \right\rfloor \end{cases}$$

The expression for d(n) can be simplified and written with the use of s(n) (moreover the integer division by 4 can be rounded by adding a 2 to the numerator). These result in:

$$\begin{cases} s(n) = \left\lfloor \dfrac{x(2n) + x(2n+1)}{2} \right\rfloor \\ d(n) = x(2n+2) - (2n+3) + \left\lfloor \dfrac{-s(n) + s(n+2) + 2}{4} \right\rfloor \end{cases}$$

The TS-transform is reversible and the inverse is:

$$\begin{cases} x(2n) = s(n) + \left\lfloor \dfrac{p(n)+1}{2} \right\rfloor \\ x(2n+1) = s(n) - \left\lfloor \dfrac{p(n)}{2} \right\rfloor \end{cases}$$

where p(n) must first be computed by,

$$p(n) = d(n-1) - \left\lfloor \frac{-s(n-1) + s(n+1) + 2}{4} \right\rfloor$$

The results from the low-pass filter may be used twice (in the first and third terms) in the high-pass filter. Therefore, only two other additions need to be performed to arrive at the results of the high-pass filter.

The TS-transform, in addition to being reversible, is also efficient. Hence, it lends itself quite well to lossless compression. The TS-transform (like the S-transform) has no growth in the smooth output, i.e., if the input signal is b bits deep, so is the smooth output. This is useful for in pyramidal systems, defined in the next section, where the smooth output is decomposed further. There is no systemic error due to rounding in the integer implementation of the transform, so all error in a lossy system can be controlled by quantization.

Among the four filters participating in a wavelet transform, the low-pass synthesis filter is the most important because it combines the quantized coefficients and also smooths the artifacts. This fact has led to the choice of a relatively long (six-tap) and particularly well behaved filter for the low-pass synthesis filter in the present invention. Note that in a QMF system, there are only two independent filters.

Many overlapped, non-minimal length reversible filters may be used in the present invention. Such forward and inverse representations of the transform system for filtering with non-overlapped minimal length reversible filters is shown in Figure 3B. For instance, the following class of filters may be used. For an integer $L \geq z$,

$$d(0) = x\left(2\left(\lfloor L/2 \rfloor + 1\right)\right) - x\left(2\left(\lfloor L/2 \rfloor + 1\right) + 1\right)$$

and

$$y_0(0) = \lfloor (x(0) + x(1))/2 \rfloor$$
$$y_0(1) = \lfloor (x(2) + x(3))/2 \rfloor$$
$$y_0(L-1) = \lfloor (x(2\lfloor (L-1)/2 \rfloor) + x(2\lfloor (L-1)/2 \rfloor + 1))/2 \rfloor$$

and

$$y_1(0) = \frac{\sum_{i=0}^{\lfloor L/2 \rfloor} a_i y_0(i) + bd(0) + \sum_{j=\lfloor L/2 \rfloor - 2}^{L-1} c_i y_0(j) + \left\lfloor \frac{k}{2} \right\rfloor}{k}$$

The length of the high-pass filter is 2L. If L is odd, the filter may be closer to a symmetric filter. If a_i, b, c_i and k are integers and $k \leq b$, then the filter is reversible. If a_i, b, c_i, and k are powers of two (or the negative or complement of a power of two), then the implementation of the filter may be simplified. If k = b (regardless of the values of a_i and c_i) then the range of the output of the high-pass filter y_1 is minimized. For each a_i, if there is exactly one c_j where $a_i = -c_j$, then the high-pass filter will have no response to a constant input. If $a_i = -c_j$ when j-(L-1)= i, then the filter may be closer to a symmetric filter.

Another useful property is

$$\sum_{i=0}^{\lfloor L/2 \rfloor} \left[(a_i)(2i)^m + (a_i)(2i+1)^m \right] + (b)\left(2(\lfloor L/2 \rfloor + 1)\right)^m$$
$$-(b)\left(2(\lfloor L/2 \rfloor + 1) + 1\right)^m + \sum_{j=\lfloor L/2 \rfloor + 2}^{L-1} \left[c_j (2_j)^m + c_j (2_j + 1)^m \right] = 0$$

This makes the high-pass filter have no response to a linearly changing input when m=1 and a quadratically changing input when m=2, etc., where m is the moment condition. This property is the principle reason that the RTS-transform has better energy compaction than the S-transform.

While filters must meet the minimum constraints for reversibility, for different applications, filters may be used that meet none, some or all of the other properties. In some examples, one of the following high-pass filters is used. The filters are listed in a notation that just lists the integer coefficients of the rational version of the filter, to avoid obscuring the invention.

1 1 -4 -4 16 -16 4 4 -1 -1

1 1 -3 -3 8 -8 3 3 -1 -1

-1 -1 0 0 16 -16 0 0 1 1

-1 -1 4 4 -16 -16 256 -256 16 16 -4 -4 1 1

3 3 -22 -22 128 -128 22 22 -3 -3

The last filter is referred to as the (Two/Ten) TT-filter, and it has the property that it has no response to a cubically increasing function. Note - that since 22=16+2x3 and 3=2+1, this filter can be implemented with a total of seven additions and subtractions.

The filters can be combined and applied to a block, such that both the horizontal and the vertical passes are performed in one operation. Figure 4 illustrates the filters to perform the combined operation. Configuration (a) shows the use of two separate 1-D reversible filters, one for each pass, that include a 1-D filter and 1-D rounding. Configuration (b) shows a 1-D filter 401, followed by another 1-D filter 402, ending with a 2-D rounding operation 403. This configuration produces more precise results in that it allows for better rounding.

The strict reversibility requirements for filters can be relaxed by noting the following. High pass coefficients are encoded and decoded in the some order. Pixel values corresponding to previously decoded high

pass coefficients are known exactly, so they can be used in current high pass filtering. For example, the following filter can be used when a raster order is used.

$$H_1(Z)=\left\lfloor\frac{1}{4}\left(\left\lfloor-\frac{1}{2}\left(1+Z^{-1}\right)\right\rfloor+\left\lfloor\frac{1}{2}\left(8\left(Z^{-2}-Z^{-3}\right)+\left(Z^{-4}+Z^{-5}\right)\right)\right\rfloor+2\right)\right\rfloor$$

The use of a single fixed high-pass filter is not required. Adaptive filters may be used or multiple filters may be used. The data used to adapt or select among multiple filters must be restricted to data that is available in the decoder prior to a particular inverse filtering operation.

One way to use multiple filters is to process the high-pass coefficients progressively. Alternate high-pass filtering operations ($y_1(0)$, $y_1(2)$, $y_1(4)$, . . .) may be processed first with a reversible filter such as the RTS high-pass filter. The remaining processing ($y_1(1)$, $y_1(3)$, $y_1(5)$, . . .) may use a non-reversible filter of up to six taps, because the exact values of the inputs to the overlap portion of the filter are known. For example, any of the following filters may be used.

-1 3 -3 1

-1 4 -4 1

-3 8 -8 3

1 -5 10 -10 5 -1

1 -4 8 -8 4 -1

Note that QMF filters are not used in some examples. The high pass filter may be replaced with a prediction/interpolation operation. A predictor/interpolator may predict the difference between a pair of inputs using any data that is available in

the decoder prior to a particular prediction/interpolation operation. The predicted difference is subtracted from the actual difference of the inputs and the result is outputted. In one example, prior art prediction methods used in DPCM, progressive coding or spatial domain coding are used.

Non-linear filters may also be used, such as morphological filters (e.g. a median filter). In one example, the 1,1 filter is used in conjunction with a different filter for the highpass. Such a filter system must be able to transmit the difference between two pixels. Based on any data the decoder has, a prediction can be made as to what the difference should be. A non-linear morphological filter may be used to do the estimate. The reference system computes the median around a pixel using the actual pixels on the causal side of the window and inputting them into the filter. On the non-causal side of the filter, the low pass coefficients are used instead of pixel values.

Two-Dimensional Wavelet Decomposition

Using the low-pass and high-pass filters of the reference system, a multi-resolution decomposition is performed. The number of levels of composition is variable and may be any number; however, currently the number of decomposition levels equals from two to five levels.

The most common way to perform the transform on two-dimensional data, such as an image, is to apply the one-dimensional filters separately, i.e., along the rows and then along the columns. The first level of decomposition leads to four different bands of coefficients, referred to herein as LL, HL, LH, and HH. The letters stand for low (L) and high (H)

corresponding to the application smooth and detail filters defined above respectively. Hence, the LL band consist of coefficients from the smooth filter in both row and column directions. It is common practice to place the wavelet coefficients in the format as in Figures 5A-5D.

Each subblock in a wavelet decomposition can be further decomposed. The most common practice is to only decompose the LL subblock further, but this can be done a number of times. Such a multiple decomposition is called pyramidal decomposition (Figures 5A-5D). The designation LL, LH, HL, HH and the decomposition level number denote each decomposition. Note that with either filters, S or TS, of the present invention, pyramidal decomposition does not increase the coefficient size.

For example, if the reversible wavelet transform is recursively applied to an image, the first level of decomposition operates on the finest detail, or resolution. At a first decomposition level, the image is decomposed into four sub-images (e.g., subbands). Each subband represents a band of spatial frequencies. The first level subbands are designated LL_0, LH_0, HL_0 and HH_0. The process of decomposing the original image involves subsampling by two in both horizontal and vertical dimensions, such that the first level subbands LL_0, LH_0, HL_0 and HH_0 each have one-fourth as many coefficients as the input has pixels (or coefficients) of the image, such as shown in Figure 5A.

Subband LL_0 contains simultaneously low frequency horizontal and low frequency vertical information. Typically a large portion of the image energy is concentrated in this subband. Subband LH_0 contains low frequency horizontal and high frequency vertical information (e.g., horizontal edge information). Subband HL_0 contains high frequency

horizontal information and low frequency vertical information (e.g., vertical edge information). Subband HH_0 contains high frequency horizontal information and high frequency vertical information (e.g., texture or diagonal edge information).

Each of the succeeding second, third and fourth lower decomposition levels is produced by decomposing the low frequency LL subband of the preceding level. This subband LL_0 of the first level is decomposed to produce subbands LL_1, LH_1, HL_1 and HH_1 of the moderate detail second level, as shown in Figure 5B. Similarly, subband LL_1 is decomposed to produce coarse detail subbands LL_2, LH_2, HL_2 and HH_2 of the third level, as shown in Figure 5C. Also, subband LL_2 is decomposed to produce coarser detail subbands LL_3, LH_3, HL_3 and HH_3 of the third level, as shown in Figure 5D. Due to subsampling by two, each second level subband is one-sixteenth the size of the original image. Each sample (e.g., pel) at this level represents moderate detail in the original image at the same location. Similarly, each third level subband is 1/64 the size of the original image. Each pel at this level corresponds to relatively coarse detail in the original image at the same location. Also, each fourth level subband is 1/256 the size of the original image.

Since the decomposed images are physically smaller than the original image due to subsampling, the same memory used to store the original image can be used to store all of the decomposed subbands. In other words, the original image and decomposed subbands LL_0 and LL_1 are discarded and are not stored in a three level decomposition.

Although only four subband decomposition levels are shown, additional levels could be developed in accordance with the requirements

of a particular system. Also, with other transformations such as DCT or linearly spaced subbands, different parent-child relationships may be defined.

There is a natural and useful tree structure to wavelet coefficients in a pyramidal decomposition. Note that there is a single LL subblock corresponding to the last level of decomposition. On the other hand, there are an many LH, HL, and HH bands as the number of levels. The tree structure defines the parent of a coefficient in a frequency band to be a coefficient in a same frequency band at a lower resolution and related to the same spatial locality. Figure 6 shows the parental relationship between two consecutive levels.

Referring to Figure 6, the coefficient at A is the direct parent to B, C, and D but is also parent to the coefficients that have B, C and D as parents. Specifically, B is parent to the four coefficients around E and the sixteen coefficients around H, etc.

The process of multi-resolution decomposition may be performed using a filtering system, such as that depicted in Figure 7. An input signal representing a one-dimensional signal with length L is low-pass and high-pass filtered by filter units 701 and 702 before being subsampled by two via units 703 and 705. A subsampled output signal from unit 703 is low-pass and high-pass filtered by units 705 and 706 before being subsampled by two via units 707 and 708, respectively. Subband components L and H appear at respective outputs of units 707 and 708. Similarly, the output signal from unit 705 is low-pass and high-pass filtered by units 709 and 710 before being subsampled by units 711 and 712, respectively. Subband components L and H appear at respective outputs of units 711 and 712. As described

above, the filters used in subband decomposition may be digital quadrature mirror filters for splitting the horizontal and vertical frequency bands into low frequency and high frequency bands.

Figure 8 illustrates a two-dimensional, two-level transform. Figure 9 also illustrates a two-dimensional, two-level transform implemented using one-dimensional filters, such as those shown in Figure 16 and 17. The one-dimensional filters are applied at every other position, to avoid computation rendered unnecessary by subsampling. One-dimensional filters may share computation between low-pass and high-pass computation.

Therefore, the reference system for compression and decompression uses non-minimal length, overlapped reversible filters. Figure 10 is a block diagram of one example of such a system. Referring to Figure 10, hierarchical decompression is initially performed. The results of the hierarchical decomposition are sent to a compressor for compression. The compression performed may include vector quantization, scalar quantization, zero run length coding, Huffman coding, Tunstall, etc. The output of the compressor compresses data representing a compressed version of the original input data. A decompressor may receive the data at sometime in the future and decompress the data. The reference system then performs an inverse decomposition using non-minimal length, overlapped reversible filters to generate a reconstructed version of the original data. Note that the non-minimal length, overlapped reversible filters comprise non-S transform filters.

The reversible wavelet filters may also be used in exemplary analysis and enhancement systems, such as shown in Figure 11. Referring to Figure 11, hierarchical decomposition is performed on input data using non-minimal length, overlapped reversible wavelet filters. The analysis unit receives the coefficients generated by the filters and classifies them into decisions, e.g., rather than encoding the coefficients completely, only relevant information is extracted. For example, in a document archiving system, blank pages might be recognized using only the coarsest low-pass subband. Another example would be to only use high pass information from a particular subband to distinguish between image of text and images of natural scenes. The hierarchical decomposition may be used for registering multiple images, such that coarse registration is done first with coarse subbands. In another embodiment, the coefficients undergo enhancement or filtering followed by inverse decomposition. Sharpening, edge enhancements, noise control, etc. may be performed using a hierarchical decomposition. Thus, the present invention provides a wavelet transform for use in joint time/space and frequency domain analysis and filtering/enhancement systems.

Ordering and Modeling of the Coefficients and Bit Planes

In the reference system, the coefficients generated as a result of the wavelet decomposition are entropy coded. The coefficients initially undergo embedded coding in which the coefficients are ordered in a visually significant order or, more generally, ordered with respect to some error metric (e.g., distortion metric). Error or distortion

metrics include peak error, and mean squared error (MSE). Additionally, ordering can be performed to give preference to bit-significance spatial location, relevance for data base querying, and directly (vertical, horizontal, diagonal, etc.).

The ordering of the data is performed to create the embedded quantization of the codestream. Two ordering systems are used: a first for ordering the coefficients and a second for ordering the binary values within a coefficient. The ordering produces a bitstream that is thereafter coded with a binary entropy coder.

The coefficient ordering and modeling may comprise M-ary coding. Alternatively, it may be embedded by band only, instead of by bit. Also, for lossless coding or single quality lossy coding (e.g. quantization specified at the encoder), non-embedded coding may be used in the coefficient ordering and modeling.

Coding Unit

A coding unit is a rectangular set of trees that are coded independently of the rest of the image. The coding unit represents the smallest unit of coded data (although there are quantization options that would allow partial coding units to be decoded). All of the data in a coding unit is available to the encoder at one time, e.g. buffered in memory.

The choice of a coding unit is implementation dependent. The coding unit may be defined as the entire image (or other data set) or a single tree or any rectangle in between. In one example, the choice of a coding unit may entail a compromise between compression efficiency and memory usage.

In one example, all the coefficients within a coding unit are available in random access memory. Since all coefficients within a coding unit are available in random access memory, the embedding order between the coefficients within a coding unit can be any arbitrary order. This order is known to both the encoder and the decoder. But since the entropy coder is causal with respect to this ordering, the order has a significant impact on the compression and is chosen with care. One example of particular ordering is described below.

Modeling

Joint spatial/frequency modeling comprises an embedded coding system used to encode the coefficients generated by the wavelet transform ~~of the present~~ invention. The joint space/frequency modeling takes advantage of both the known frequency bands and the neighboring pixels (or data). One example of the joint space/frequency modeling is referred to herein as horizon modeling.

The data is initially formatted in sign magnitude format, which is followed by the data being sorted based on significance. In another embodiment, to further reduce workspace memory, the coefficients could be stored in a magnitude/mantissa form instead of a sign/magnitude.

After the data is sorted with respect to the given significance metric, the data is encoded.

Assuming a digital signal, x(n), for each x(n) is represented with R bits of precision, then the embedded coding encodes the most significant bit (or bits) of every x(n) of the signal, then the next significant bit (or bits) and so on. For example, in the case of

visually defined ordering, an image that requires better quality in the center than along the corners or near the edges (such as some medical images) may be subjected to encoding such that the low-order bits of the central pixels might be coded prior to the higher-order bits of the boundary pixels.

Bit-Significance Representation

In one example, the embedded order used for binary values within a coefficient is by bit-plane. The coefficients are expressed in bit-significance representation. Bit-significance is a sign-magnitude representation where the sign bit, rather than being the most significant bit (MSB), is encoded with the first non-zero magnitude bit.

There are three types of bits in a number represented in bit-significance form: head, tail, and sign. The head bits are all the zero bits from the MSB to the first non-zero magnitude bit plus the first non-zero bit. The bit-plane where the first non-zero magnitude bit occurs defines the significance of the coefficient. The bits after the first non-zero magnitude bit to the LSB are the tail bits. The sign bit simply denotes the sign. A number with a non-zero bit as the MSB has only one head bit. A zero coefficient has no tail or sign bits.

In the case where the values are non-negative integers, such as occurs with respect to the intensity of pixels, the order that may be used is the bitplane order (e.g., from the most significant to the least significant bitplane). Where two's complement negative integers are also allowed, the embedded order of the sign bit is the same as the first non-zero bit of the absolute value of the integer. Therefore, the sign bit is

not considered until a non-zero bit is coded. For example, using sign magnitude notation, the 16-bit number -7 is:

1000000000000111

On a bit-plane basis, the first twelve decisions will be "insignificant" or zero. The first 1-bit occurs at the thirteenth decision. Next, the sign bit ("negative") will be coded. After the sign bit is coded, the tail bits are processed. The fifteenth and sixteenth decisions are both "1".

Coefficient Alignment

The coefficients in the different subblocks represent different frequencies similar to the FFT or the DCT. The quantization is performed by aligning coefficients with respect to each other before the bit-plane encoding. The less heavily quantized coefficients will be aligned toward the earlier bit-planes (e.g., shifted to the left). Thus, if the stream is truncated, these coefficients will have more bits defining them than the more heavily quantized coefficients.

In one example, the coefficients are aligned for the best rate-distortion performance in terms of SNR or MSE. Alternately, the alignment could allow a physchovisual quantization of the coefficient data. The alignment has significant impact on the evolution of the image quality (or in other words on the rate-distortion curve), but has negligible impact on the final compression ratio of the lossless system.

The bit depths of the various coefficients in a two-level TS-transform decomposition from an input image with b bits per pixel are shown in Figure 12. To align the coefficients, the 1-HH coefficient size is used as a reference,

and shifts are given with respect to this size. Table 1 shows an example of this alignment process.

Table 1 - Example of coefficient alignment, b=8

1-HH	Reference		10	9	8	7	6	5	4	3	2	1	0	
x-HL or x-LH	Left 3	8	7	6	5	4	3	2	1	0				
x-HL or x-LH	Left 2		8	7	6	5	4	3	2	1	0			
x-HL or x-LH	Left 1			8	7	6	5	4	3	2	1	0		
x-HL or x-LH	none				8	7	6	5	4	3	2	1	0	
x-HL or x-LH	Right 1					8	7	6	5	4	3	2	1	0

Note the sign bit is not the MSB and is encoded with the first tail bit. It is important to note that the alignment simply controls the order the bits are sent to the entropy coder. Actual padding, shifting, storage, or coding of extra zero bits is not performed.

Context Model

One example of the Horizon context model is described below. This model uses bits within a coding unit based on the spatial and spectral dependencies of the coefficients. The available binary values of the neighboring coefficients, and parent coefficients can be used to create contexts. The contexts, however, are causal for decodability and in small numbers for efficient adaptation.

Entropy Coding

In one example, the entropy coding performed by the reference system is performed by binary entropy coders. Entropy coder 104 comprises a Q-coder, a QM-coder, a finite state machine coder, a high speed parallel coder, etc. A single coder may be used to produce a single output code stream. Alternately, multiple (physical or virtual) coders may be employed to produce multiple (physical or virtual) data streams.

In one example, the binary entropy coder comprises a Q-coder. For more information on the Q-coder, see Pennebaker, W.B., et al., "An Overview of the Basic Principles of the Q-coder Adaptive Binary Arithmetic," IBM Journal of Research and Development, Vol. 32, pg. 717-26, 1988. In an alternate example, a binary entropy coder uses a QM-coder, which is a well known and efficient binary entropy coder. It is particularly efficient on bits with very high probability skew. The QM-coder is used in both the JPEG and JBIG standards.

The binary entropy coder may comprise a finite state machine (FSM) coder. Such a coder provides the simple conversion from a probability and an outcome to a compressed bit stream. A finite state machine coder is implemented using table look-ups for both decoder and encoder. A variety of probability estimation methods may be used with such a finite state machine coder. Compression is excellent for probabilities close to 0.5. Compression for highly skewed probabilities depends on the size of the lookup table used. Like the QM-coder, it is useful with embedded bit streams because the decisions are coded in the order of occurrence. There is no possibility for "carry-over" problems because the outputs are defined by a lookup table. In fact, there is a maximum delay between encoding and the

production of a compressed output bit, unlike the Q and QM coders. The finite state machine coder may comprise a B-coder defined in US Patent No. 5,272,478, entitled "Method and Apparatus for Entropy Coding", issued December 21, 1993.

The binary entropy coder may comprise a high speed parallel coder. Both the QM-coder and the FSM coder require that one bit be encoded or decoded at a time. The high-speed parallel coder handles several bits in parallel. The high speed parallel coder is implemented in VLSI hardware or multi-processor computers without sacrificing compression performance. One high speed parallel coder that may be used is described in U.S. Patent No. 5,381,145, entitled "Method and Apparatus for Parallel Decoding and Encoding of Data", issued January 10, 1995.

Most efficient binary entropy coders are limited in speed by fundamental feedback loops. A possible solution is to divide the incoming data stream into multiple streams and feed these to parallel encoders. The output of the encoders are multiple streams of variable-length coded data. One problem with this type of approach is how to transmit the data on a single channel. The high speed parallel coder described in U.S. Patent No. 5,381,145 solves this problem with a method of interleaving these coded data streams.

Many of the contexts are fixed probability, which makes a finite state machine coder, such as the B-coder especially useful. Note when a system using probabilities close to 0.5, both high speed parallel coder disclosed above and the finite state machine coder

operate with more efficiency than the Q-coder. Thus, both have a potential compression advantage with the context model.

The Encoding and Decoding Process

The following flow charts, Figures 13-15, depict one example of the encoding and decoding processes of the reference system. These processes may be performed in software or with hardware. In either case, references have been made to processing logic, which may represent either.

Figure 13 illustrates one example of the encoding process.

Referring to Figure 13, the encoding process begins by having processing logic acquiring an input data for a coding unit (processing block 1301). Next, processing logic applies reversible filter(s) to the input data unit of the coding unit (processing block 1302).

A test then determines if another level of decomposition is desired (processing block 1303). If so, processing logic applies the reversible filter to all the LL coefficients (processing block 1304), and the process loops back and continues at processing block 1303. If another level of decomposition is not desired, processing continues at processing block 1305 where processing logic converts the coefficients to sign/magnitude form.

After converting coefficients to the sign/magnitude form, a bitplane variable, S, is set to the most significant bitplane (processing block 1306). Then, the processing logic optionally initializes the entropy coder (processing block 1307).

Once the entropy coder has been initialized, processing logic models each bit of each coefficient with the context model and entropy codes the

bits (processing block 1308). After entropy coding the bit, the data is either transmitted or stored (processing block 1309).

Thereafter, a test determines if there are any more coding units in the image (processing block 1310). If there are more coding units, processing continues to processing block 1301. On the other hand, if there are no more coding units, then processing ends.

Figure 14 illustrates one example of the decoding process.

Referring to Figure 14, the process begins by processing logic retrieving coded data for one coding unit (processing block 1401). Next, a variable S is set to the most significant bitplane (processing block 1402). After setting the bitplane variable S to the most significant bitplane, processing logic optionally initializes the entropy coder (processing block 1403).

After the entropy coder has been initialized, processing logic sets the initial value of each coefficient to zero (processing block 1404). Then the processing logic models each bit of each coefficient with a context model and entropy decoder (processing block 1405) and converts coefficients to proper form for filtering (processing block 1406). This conversion may convert from bit significance to two's compliment form. Thereafter, the processing logic applies an inverse filter(s) on the coefficients starting from the highest level of decomposition (processing block 1407).

A test then determines if all the levels have been inverse filtered (processing block 1408). If all the levels have not been inverse filtered, processing logic applies the inverse filter(s) on the coefficients from the next highest level of decomposition (processing block 1409), and processing continues at processing block 1408. If all the levels have been

inverse filtered, processing continues at processing block 1410 where the reconstructed data is either stored or transmitted. After storing the transmitted reconstructed data, a test determines if there are more coding units (processing block 1411). If there are more coding units, processing loops back and continues at processing block 1401 where the process is repeated. If there are no more coding units, the process ends.

Figure 15 illustrates one example of the process for modeling bits. Referring to Figure 15, the process for modeling bits begins by setting a coefficient variable C to the first coefficient (processing block 1501). Then, a test determines if $|c|>2^S$. If yes, processing continues at processing block 1503 where processing logic codes bit S of coefficient C using the model for tail bits and processing continues at processing block 1508. The model for tail bits may be a stationary (non-adaptive) model. If $|c|$ is not greater that 2^S, then processing continues at processing block 1504 where processing logic applies a template for head bits (i.e., the initial zeros and the first on "1" bit). After applying the template, processing logic codes bit S of coefficient C (processing block 1505). Possible templates are shown in Figures 26A-C. Templates may be implemented with LUTs, as shown in Figures 19A and 19B.

Next, a test determines if bit S of coefficient C is on (processing block 1506). If bit S of coefficient C is not on, processing continues at processing block 1508. On the other hand, if bit S of coefficient C is on, processing continues at processing block 1507 where processing logic codes the sign bit. Thereafter, processing continues at processing block 1508.

At processing block 1508, a test determines if coefficient C is the last coefficient. If coefficient C is not the last coefficient, processing continues

at processing block 1509 where the coefficient variable C is set to the next coefficient and processing continues at processing block 1502. On the other hand, if coefficient C is the last coefficient, processing continues at processing block 1510 where a test determines if S is the last bitplane. If S is not the last bitplane, bitplane variable S is decremented by 1 (processing block 1511) and processing continues at processing block 1501. If S is the last bitplane, processing ends.

The reference system may be implemented in hardware and/or software. A hardware implementation requires implementation of the wavelet filters, memory/data flow management to provide the data for the filters, a context model to control the embedded coding of the present invention, memory/data flow management to provide the data for the context model, and a binary entropy coder.

Wavelet Filters

One example of a forward wavelet filter is shown in Figure 16. The wavelet filter shown in Figure 16 accommodates 4 16-bit two's complement input pixels, shown as x(2)-x(5).

Referring to Figure 16, the two tap "1 1" low-pass filter uses one 16-bit adder 1601. The outputs are called S and D, respectively. The output of the adder (S) is truncated to 16 bits using shift-by-1 block 1603. The shift-by-1 block 1603 performs a divide-by-2 function by shifting its 17-bit input to the right one bit.

The six tap "-1 -1 8 -8 1 1" high-pass filter requires the computation of $-S_0 + 4D_1 + S_2$. The function $S_2 - S_0$ is computed with 16-bit subtractor 1605 receiving the output of shift-by-1 block 1603 and the $Y_0(0)$. The $4D_1$ term is computed using subtractor 1602, shift-by-2 block 1604 and adder 1608. The output produced by 16-bit subtractor 1602 is shifted to the left two places by shift-by-two block 1604, thereby effectively multiplying its output by four. The output of block 1604 is added to 2 by adder 1608. Note that because of the shift by 2, adder 1608 may be replaced by wiring. Adding the $4D_1$ output from adder 1608 to the output of subtractor 1605 is performed by 20-bit adder 1606. The output of the adder 1606 is truncated to 18 bits using shift-by-2 block 1607. Shift-by-2 block 1607 performs a divide-by-4 function by shifting its 20 bit input to the right two bits.

Thus, the total computational hardware required (not counting registers for storing temporary results) is:

- 1 @ 16-bit adder,
- 2 @ 16-bit subtractors,
- 1 @ 19-bit adder.

Note that shifting (and adder 1608) is performed by the wiring, such that no logic is needed.

In other examples for inputs of size N, one N-bit adder, two N-bit subtractors and one (N+3) bit adder may be used.

Due to the extremely low hardware cost of these adders/subtractors, parallel implementations of the filters can be used if desired.

Note that alternatively, instead of subtracting X(3) and X(2), X(4)-X(5) can be computed and saved until needed later as X(2)-X(3) for the next shift or application of the filter. Both the forward filter (and the

inverse filter described below) may be pipelined to achieve higher throughput.

The inverse wavelet filter is shown in Figure 17. The inputs of $Y_0(0)$ and $Y_0(2)$ are subtracted by subtractor 1701. Two (2) is added from the output of subtractor 1701 by adder 1709. The result of the addition is shifted to the right two bits by shift-by-2 block 1702. This effectively divides the output of the subtractor by 4. A subtraction is performed between the output of shift-by-2 block 1704 and the $Y_1(0)$ input. The input $Y_0(1)$ is shifted one bit to the left by shift-by-1 block 1703, thereby multiplying the input by two. After $Y_0(1)$ is shifted by 1 (multiplied by two), the LSB of the shifted value is the LSB taken from the output of subtractor 1704 and combined with the 16 bits output from shift-by-1 block 1703 to form an input for adder 1705 and subtractor 1706. The other input for adder 1705 and subtractor 1706 is the output of subtractor 1704. The outputs of adder 1705 and subtractor 1706 may subsequently undergo clipping.

A choice of two clip operations may be used. In both cases, the 20-bit value is shifted by 1 (divided by 2), to a 19-bit value. For a system that only performs lossless compression, the least significant 16 bits can be output (the remaining 3 bits can be ignored). In a lossy system (or a lossy/lossless system), the 19-bit value is set to zero if it is negative or set to 2^{16}-1 if it is greater than 2^{16}-1; otherwise, the least significant 16 bits can be output.

For inputs of size N bits, one N-bit subtractor, one (N+1) bit adder, one (N+2) bit subtractor, one (N+3) bit adder and one (N+3) bit subtractor may be used, and the clip unit outputs N bits.

In one example of the wavelet transform, Monte Carlo division is used in the transform computations, wherein a pseudo random generator is used and based on its output, the results of a transform operation are either rounded up or down. Such an implementation may be used as long as a decoder is aware of the rounding being performed (i.e., uses the same random generator starting at the same point).

Memory Usage for Data Flow Management for Wavelet Filters

With respect to memory and data flow management for the wavelet filters for images where a full frame can fit in memory, memory/data flow management is not a difficult issue. Even for 1024 X 1024 16-bit medical images (e.g., 2 Mbytes in size), requiring a full frame buffer is reasonable for many applications. For larger images (e.g., A4, 400 DPI 4-color images are about 50 Mbytes in size), performing the wavelet transform with a limited amount of line buffer memory is desirable.

Note that a full frame buffer is not necessary to implement a one-pass system. Because of this, the memory required may be reduced by about a factor of 100 (compared to using a full frame buffer for large images). The one-pass system is described later.

The data stored in the filter memory is a series of coefficients that are to be subjected to the embedded coding and binary entropy coding. The embedded coding uses a context model to coordinate the use of horizon coding, and to provide data in the proper order. The context model operates in conjunction with a memory management scheme. For

systems with a full frame buffer, providing data in the proper order is not difficult.

For systems with a finite amount of workspace memory, in one embodiment, different height transforms are used to reduce the number of workspace lines of memory needed for storage. Thus, if a wider image is encountered, it may be compressed efficiently within the allotted workspace memory. For instance, the S-transform may be used vertically to reduce the number of lines.

Memory is required to buffer raster data, so a wavelet transform can be performed. In some applications, minimizing this memory is important for reducing cost. A technique for accomplishing this is described below.

One example of the wavelet 2-D transform described herein is designed for a one-pass implementation and restricted memory usage. In one example, the wavelet transforms applied to achieve the pyramidal decomposition are only TS and S transforms. In this example, there are four levels of separable pyramidal decompositions. In one example, a four level decomposition is performed using the S and TS transforms. In one example, in the horizontal (row-wise) decomposition, solely the TS-transform is used, i.e, the horizontal decomposition is formed of TS-TS-TS-TS. In the vertical (column-wise) decomposition, the S-transform and the TS-transform are both used in the form of TS-TS-S-S. Two of the TS-transforms are replaced by S-transform at a small cost to the compression, but significant impact on the memory usage. The horizontal and vertical transforms are applied alternatively as usual (Figure 20).

Note that any combination of the S and TS transforms may be used to implement the horizontal and vertical transforms. Note that although

the orders of the transforms may be mixed, the decoder must be aware of the order and must perform a reverse operation in the reverse order to be fully reversible.

Coefficient Trees

In a pyramidal system, the coefficients can be grouped into sets using a tree structure. The root of each tree is a purely low-pass coefficient. Figure 6 illustrates the tree structure of one purely low-pass coefficient of the transformed image. For a two-dimensional signal such as an image, the root of the tree has three "children" and the rest of the nodes have four children each. The tree hierarchically is not limited to two dimensional signals. For example, for a one dimensional signal, a root has one child and non-root nodes have two children each. Higher dimensions follow from the one-dimensional and two-dimensional cases.

The tree structure is also apparent from the operation of the filters shown in Figures 7-9. The operation of the pairs of filters with subsampling causes the previously described coefficients to be related.

In one example, the coefficients are coded in a bit significance, or bit-plane embedded system. Since the coefficients are coded from most significant bitplane to least significant bitplane, the number of bitplanes in the data must be determined. This is accomplished by finding an upper bound on the magnitudes of the coefficient values calculated from the data or derived from the depth of the image and the filter coefficients. For example, if the upper bound is 149, then there are 8 bits of significance or 8 bitplanes. For speed in software, bitplane coding may not be used. In an alternate example, a

bitplane is coded only when a coefficient becomes significant as a binary number.

The horizon context model may comprise the bit-significance embedded encoding of the wavelet coefficients that feeds a binary entropy coder.

Context Model Alternatives

Once the decomposition has been completed and the data coefficients ordered, the context model is used to encode the coefficients. There are various context models that may be used. Decisions may be conditioned on spatial location, level, and/or bit position. Decisions may also be conditioned on previously decoded data that is close to the current data in spatial location, level, and/or bit position.

Some examples are as follows. The most significant tail bit (and therefore most easily predicted) could use a different context than the rest of the tail bits. Head bits can be conditioned on the same bit for spatially close previous coefficients at the same transform level. Similarly, the sign bits for significant coefficients might be conditioned on the sign of spatially close previous coefficients at the same level or the sign of the coefficient of the parent.

Context model improvements might be especially important when compressing images that have spatial or multi-resolution structure. Grayscale images of line drawings or text are an example of images with both of these types of structure. Improvements are also important for

compressing files that already have to be compressed and decompressed with a specified peak error.

When encoding in software, a large amount of time is expended to obtain bits for contexts because the bits are required for conditioning (e.g. every head bit). A software implementation may be speeded up using look-up tables (LUTs). This avoids separate bit extraction operations for the North (N), Northwest (NW), West (W) and Southwest (SW) pixels that are used as contexts.

Figures 19A and 19B illustrate a state machine for head bit conditioning. Referring to Figure 19A, a LUT 1901 for a new parent is shown coupled to an encode/decode block 1902. LUT 1901 is coupled to receive bits indicative of a parent, bits representing the above (NE) coefficients, the current (E) coefficient and the coefficient below (S). The parent input and current input comprise two bits each. Other inputs to LUT 1901 include all or part of the context output from LUT 1901 and the output of encode/decode block 1902, as feedbacks. 8 of 10 bits output as a context by LUT 1901 are fed back to an input of LUT 1901.

The NE, E and S coefficients are used because they represent the leading edge of template information which comprises coefficient information associated with the previous bit-planes. Note that the Southeast (SE) coefficient may be used instead of the South (S) coefficient. In one example, if the template is outside the coding unit, the outside conditioning bits may be replaced with bits from the current pixel.

Figure 19B illustrates state machine conditioning using a LUT for the same parent. In such a case, the entire context is fed back as an input to LUT 1903.

Where data is processed in raster order, using a LUT reduces the number of memory accesses because the same memory used to generate the last context does not have to be reloaded.

To reduce the size of LUT memory, alternatively the parent conditioning can be done separately by ORing with the output of a LUT which only handles the other conditioning.

A slightly larger LUT table can also provide most of the conditioning for the next bitplane also. Another smaller LUT could take the state information from the current context LUT and combine it with the newly available data from the next bitplane. This may especially be useful if coding one tree at a time.

As described above "efficient" may be defined to mean the transform has determinant of 1. In such a case, code space is not wasted by saving room to code low probability events when the low probability is zero. However, 8-bit coefficients are still input and produce an 8-bit coefficient and one 9-bit coefficient. Therefore, efficiency may still be improved. The added inefficiency is due to the rotation of the space of possible coefficients.

It should be noted that certain results of the transform operations uniquely identify numbers used in the computations. This occurs when the results are near the bounds of those ranges of possible results. This is exemplified by Figure 18, wherein u represents the low-pass and v represents the high-pass. Because the values of u and v are not

independent, these numbers may also be easier to entropy code taking joint information into account. This is because, as shown in Figure 18, for most low pass values, some code space for the high pass is not used. In many applications there is little advantage because the probability assigned to these impossible pairs is low. However, there might be a worthwhile gain in some applications. To speed operations, more bits of the LL coefficients could be sent prior to the LH, HL and HH coefficients. This may make bounding easier.

In some examples, after each coding unit has been coded, everything is reset, all the statistics and probabilities are reset when coding the second unit. Some of the statistics or all may be saved. These then act as the initial statistics when coding of a later coding unit begins. The statistics may be saved at a predetermined point in the coding of the first or previous coding unit. For example, after coding the third bit plane, the statistics used by code the current coding unit are saved and act as the statistics for the beginning of coding of the following coding unit or later coding unit. The classes for all images may be evaluated and a hard coded set of statistics are determined. Then coding is performed using these hard coded statistics as a default. Statistics may be saved for each bit plane, such that when coding in the similar bit plane in another tile, the statistics are used.

In one example, there is no coding until the first one bit. At the occurrence of the first one bit in the coefficient, the sign is encoded. Although the head bits are image/region dependent, the tail bits are more uniform across different images and regions. Based on how far the tail bits

are from the initial one bit (in the head bit), certain probability classes are used to encode the bits in the tail. In one embodiment, the first tail bit in a coefficient is coded with a probability class including 0.7. The second and third tail bits are coded with a probability class including 0.6. Lastly, the fourth and further tail bits are coded with probability classes that includes 0.5.

A. Performing the Wavelet Transform

In the one-pass system, the wavelet transform performed is a compromise between compression performance and the amount of memory used. The coding unit size is chosen for the least memory usage with the fewest line buffers (assuming the image is delivered in raster order). The intermediate coefficients of the wavelet transform are stored in the same memory replacing the input as appropriate.

Choice of Wavelet Transform Filters

The wavelet 2-D transform described herein is designed for a one-pass implementation and restricted memory usage. There are four levels of separable pyramidal decompositions. In the horizontal decomposition, solely the TS-transform is used, i.e., the horizontal decomposition is formed of TS-TS-TS-TS. In the vertical decomposition, the S-transform and the TS-transform are both used, and the vertical decomposition is formed of TS-TS-S-S. The horizontal and vertical transforms are applied alternatively. Figure 20 illustrates the horizontal and vertical decompositions.

Two of the TS-transforms are replaced by S-transform at a small cost to the lossless compression, but significant impact on the memory usage. The

choice of using the S-transform in the last two vertical passes is solely to use less memory. The usage of the S-transform saves approximately 32 lines of coefficient buffer (e.g., 16 lines down from 48 lines). Note that using the TS-transform for all the decompositions does provide better compression performance.

Coding Unit Definition

In one example, the coding unit is defined by one line of trees (a line of LL coefficients and all their descendants. With four levels of decomposition, this implies that in the spatial domain, a coding unit is 16 lines by the width of the image. Figure 21 illustrates one coding unit. Note that Figure 21 is not to scale. The level 1 block is the image after one 2-D decomposition. To reiterate, the names LL(low-low), LH (low-high), HL (high-low), and HH (high-high) are used to address a subblock and are applied to all the level 1-4 blocks. Level 2 block is the result of the 2-D decomposition of the subblock LL in the level 1 block. Similarly, blocks 3 and 4 are 2-D decompositions of the subblocks LL in level 2 block and level 3 block respectively.

A coding unit is 8 lines high for the HH, HL, and LH coefficients in level 1, 4 lines high in level 2, 2 lines high in level 3, and 1 line in level 4 and the LL subblock. Notice that as the resolution decreases at each step, the length as well as the number of rows halve. Each coefficient in the LL of the level 4 block is the top parent of a tree.

Buffering and Coefficient Computation

In order to generate one coding unit described in Figure 21, a workspace buffer of size 2•w•m, where w is the width of the image, and m is the maximum coefficient size in bits, may be used. Because of the nature of the wavelet filters chosen for the vertical transform (i.e., column-wise) passes, the workspace memory requirement is approximately 18-20 lines. Each horizontal transform (i.e., row-wise) pass, all of which are TS-transforms, is computed one line (row) at a time and the new coefficients replace the old coefficients or pixels.

The first two vertical transform passes use the TS-transform filters. Because of the six-tap high pass filter, each high pass coefficient in the vertical pass depends on six lines of either pixel of coefficient data. A high pass TS coefficient generated is related to the top two lines with four lines below for overlap. This is shown in Figure 22. Referring to Figure 22, a vertical image segment of a coding unit is shown. The vertical image segment of the coding is the result of the original image being transformed by a horizontal pass of the TS-transform. A vertical level 1 segment of a coding unit is shown and is the first level 2-D transform of the image. The vertical level 1 segment is obtained by performing a horizontal pass with a TS-transform. A vertical level 2 segment of a coding unit is also shown and results from applying the TS transform on the LL subblock of the level 1 block on both dimensions.

It should be noted that since the TS-transform is overlapped by four pixels (or coefficients), four lines of data are saved at the end of a coding unit to be used in the computation of the coefficients of the next coding unit. In other words, to create the level 1 coefficients, two extra lines of pixels are needed at both the top and the bottom or four extra lines are needed at the bottom. To create the level 2 coefficients, two extra lines of level 1 coefficients

are needed at both the top and the bottom, or four are needed at the bottom. To generate these extra level 1 coefficients, another two lines of pixels are required on both the top and the bottom or four are needed at the bottom. Thus, each coding unit spans 28 vertical lines.

Importantly, however, no extra computation is required to generate these "extra" level 1 coefficients since they will be used in the coding units above and below the current one. Also note that only 20 lines of storage are required because only the level 2 coefficients are stored.

The final two vertical passes are S-transforms that have no overlap in the low pass and, thus, do not require extra lines.

Memory for the Transform Computation

Given pixel values or coefficient values of size b bits (in the range of $-2^{b-1} \ldots, 0, \ldots 2^{b-1}-1$) the smooth outputs, s(.), of both the S-transform and the TS-transform are also b bits. In other words, they have the same range as the input. However, the one-dimensional detail outputs, d(.), of the S-transform and TS-transform will take b+1 and b+2 bits to express respectively.

Figure 23 illustrates some of the line buffering needed in addition to the coding unit. The gray areas and arrows are the coefficients which are part of the current coding unit and need to be saved in memory for the current coding. The dashed arrows are the temporary coefficients which are needed to compute the coefficients in the coding unit. These are overwritten by the new coefficients. The solid arrows are the coefficients which are the by-products of the computation of the current coding unit coefficients and are saved to be part of the next coding unit.

The final level (level 4) of coefficients is a single line in all four subblocks (LL, LH, HL, HH). Referring only to the vertical transform, to calculate level 4 from level 3, the S-transform is used so all subblocks require only two lines of coefficients in level 3. Likewise, to calculate level 3 from level 2 requires four lines of coefficients in level 2. All of these coefficients are part of the current coding unit.

To calculate the vertical passes of level 2 and 1, the TS-transform is used. Because of the overlapped nature of the six-tap high pass overlapped filter, these levels require data from the next coding unit. That data is used to calculate the coefficients in the current coding unit and then are stored for use in the next coding unit.

To calculate the high pass subblocks in level 2 (LH, HH) from level 1, 12 lines are needed (8 lines to 4 lines from downsampling and 4 extra lines for the overlap). These lines are shown in low pass subblocks of level 1 (LL, HL) of Figure 23 as the 8 lines that are part of the current coding unit and 4 lines that are part of the next.

To calculate the 12 lines in the low pass subblocks of level 1 (LL, HL), 24 lines are needed from level 0. These 24 lines at level 0 can create the 10 lines in the high pass subblocks of level 1 (16 lines to 8 lines from downsampling and 4 extra lines for the overlap). It is most efficient to calculate all 10 of these lines and store them at level 1 even though only 8 are needed for the current coding unit. Thus, only the 4 extra lines used for the overlap need be saved at level 0.

Starting from an image of pixel depth b, for a separable 2-D transform, for the case that both row and column transforms are TS, the ranges of the coefficients are b, b+2, b+4 for LL, HL, LH, HH (Figure 12) subblocks

respectively. In the case the separable 2-D transform consists of horizontal TS and vertical S transforms, the ranges of the coefficients are, b, b+1, b+2, b+3 for LL, HL, LH, HH respectively. Tables 2, 3, 4, 5, and 6 illustrate the calculation for the memory required by each block. Note that the calculation is done in terms of size in bits for an image of width w, one for each block:

Table 2 - Memory cost for Level 0

Subblock	L	H
Memory	4•b•w/2	4•(b+2)•w/2

Table 3 - Memory cost for Level 1

Subblock	LL	HL	LH	HH
Memory	4•b•w/2	10•(b+2)•w/2	12•(b+2)•w/2	10•(b+4)•w/2

Table 4 - Memory cost for Level 2

Subblock	LL	HL	LH	HH
Memory	0•b•w/4	4•(b+2)•w/4	4•(b+2)•w/4	4•(b+4)•w/4

Table 5 - Memory cost for Level 3

Subblock	LL	HL	LH	HH
Memory	0•b•w/8	2•(b+1)•w/8	2•(b+2)•w/8	2•(b+3)•w/8

Table 6 - Memory cost for Level 4

Subblock	LL	HL	LH	HH
Memory	1•b•w/16	1•(b+1)•w/16	1•(b+2)•w/16	1•(b+3)•w/16

Adding all of the above numbers equals $\left(26b + 55\frac{7}{8}\right) \bullet w$ bits, which rounded is $(26b + 56) \bullet w$ bits. A two line computational buffer of the largest size, b + 4 bits, adds $2 \bullet (b + 4) \bullet w$ bits leading to a total memory cost of $(28b + 64) \bullet w$ bits. For example, for an 8-bit 512 pixel wide image, 147,456 bits or about 18K bytes of memory is required.

In one example, the size of the transform is selected based on the width of the image and the fixed size of the memory available. In other words, an image of a particular may be input into the system, and due to a limited amount of available transform memory, the number of levels of decomposition are reduced. If more memory is available, then the number of decomposition levels are increased. Note that this may occur dynamically as the image is being received into the system. If enough memory is available, the LL coefficients are fed through wavelet filters to perform the additional level of decomposition. Note that an effect of decreasing or increasing the number of levels is to decrease or increase, respectively, the amount of compression that may be achieved.

B. Embedded Order

Ordering of the Codestream

Figure 24A illustrates the ordering of the codestream and the ordering within a coding unit. Referring to Figure 24A, the header 2401 is followed by the coding units 2402 in order from top to bottom. Within a coding unit, the LL coefficients 2403 are uncoded in raster (line) order. Following the LL coefficients is entropy coded data one bit-plane at a time, starting from the

most significant bit-plane to the least significant bit-plane. Then the first bit-plane from every coefficient is coded followed by the second bit-plane, etc.

Alignment of the Coefficients

In one example of the reference system, the context model uses an unnormalized $1 + Z^{-1}$ low-pass filter. However, the context model may be used with normalized filters, such as

$$\frac{1 + Z^{-1}}{\sqrt{2}}$$

In order to use normalized filters, an alignment unit between the forward wavelet filter 1600 and the context model 105, can be used to compensate for the energy gained (or alternatively, lost) from the unnormalized filter, which improves compression. Because alignment allows non-uniform quantization for lossy operation, alignment can enhance the visual quality of lossy image reconstructions. In the one-dimensional case, coefficients from each level of the tree would have different alignment (divisors = $\sqrt{2}$, 2, $2\sqrt{2}$, 4, multipliers = $2\sqrt{2}$, 2, $\sqrt{2}$, 1). In the two-dimensional case, the divisors would be 2, 4, 8, 16 and the multipliers would be 8, 4, 2, 1.

Since the alignment is only for grouping similar binary decisions for coding, using the exact normalization value is not critical. The alignment must be inverted during decoding, so both multiplication and division are required. Using factors/divisors that are powers of two would allow hardware efficient shifting to be performed instead. When coefficients are multiplied by a power of two, the lessor significant zero bits added do not have to be coded.

Coefficient alignment can be used for tuning and for finer and non-uniform quantization. In case of images (two dimensional signals), one example of the RTS-transform aligns the coefficients by multiplying the frequency band by the numbers depicted in Figure 12B. Multiplying these numbers results in the RTS-transform being a very close approximation of the exact reconstruction wavelets of the TS-transforms.

This one-pass system uses only one alignment that is optimal with respect to MSE for the filter pairings. Table 7 illustrates the alignment numbers. The coefficients are coded by bit-significance where the first bit-plane is the left most magnitude bit of all the coefficients. The sign bit for each coefficient is not coded until the highest bit-plane where that coefficient has a non-zero magnitude bit. In other words, the sign bit is encoded right after the first "on-bit" is coded. This has the advantage of not coding a sign bit for any coefficient that has zero magnitude, and not coding a sign bit until the point in the embedded codestream where the sign bit is relevant. For an image of pixel depth b, the largest possible coefficient magnitude is $2^{b+3}-1$, i.e., a b+3 bit number. Therefore, every coefficient is encoded in b+3 binary decisions plus an additional one for the sign bit, if needed.

Table 7 - Coefficient alignment

1-HH	1-HL,1-LH	2-HH	2-HL,2-LH	3-HH	3-HL,3-LH	4-HH	4-HL,4-LH
reference	Left 1	Left 1	Left 2	Left 2	Left 3	Left 3	Left 4

The alignment of different sized coefficients is known to both the coder and the decoder and has no impact on the entropy coder efficiency.

Note also that every subblock of every block of a coding unit has its own possible largest magnitude range, which is known to the coder and the decoder. For most subblocks, there are several completely deterministic binary zero values that are skipped by the entropy coder for the sake of efficiency.

The order that the coefficients during each bit-plane are processed are from the low resolution to the high resolution and from low frequency to the high frequency. The coefficient coder within each bit-plane is from the high level (low resolution, low frequency) to the low level (high resolution, high frequency) as follows:

4-LL, 4-HL, 4-LH, 4-HH, 3-HL, 3-LH, 3-HH, 2-HL, 2-LH, 2-HH, 1-HL, 1-LH, 1-HH

Within each subblock, the coding is in raster scan order.

Note that coding units of the same data set may have different alignments. The alignment may be specified in a header, such as header 2401 in Figure 24A.

The Horizon Context Model

Figure 25 shows the neighborhood coefficients for every coefficient of a coding unit. Referring to Figure 25, the neighborhood coefficients are denoted with the obvious geographical notations (e.g., N=north, NE=northeast, etc).

Given a coefficient, such as P in Figure 25, and a current bit-plane, the context model can use any information from all of the coding unit prior to the given bit-plane. The parent coefficient of the present coefficient is also used for this context model.

Rather than using the neighborhood or parent coefficient values to determine the context for the present bit of the present coefficient, the information is reduce to two bits referred to herein as tail-information. This information can be stored in memory or calculated dynamically from the neighbor or parent coefficient. The tail-information relates whether or not the first non-zero magnitude bit has been observed (e.g., whether the first "on-bit" has been observed) and, is so, about how many bit-planes ago. Table 8 describes the tail-information bits.

Table 8 - Definition of the tail information

Tail	Definition
0	no on-bits is observed yet
1	the first on-bit was on the last bit-plane
2	the first on-bit was two or three bit-planes ago
3	the first on-bit was more than three bit-planes ago

From the 2-bit tail information, a "tail-on" bit of information indicates whether the tail information is zero or not. In one example the tail-information and the tail-on bits are updated immediately after the coefficient has been coded. In another example, updating occurs later to allow parallel context generation.

As an example, Table 9 shows the tail-on bit, as a function of bit-plane, for a coefficient with the magnitude expressed in binary as follows ("*" means it does not matter whether it is 0 or 1):

0	0	0	1	*	*	*	*

Table 9 - Table if tail information for the example context coefficient

Bit-plane	1	2	3	4	5	6	7	8
Prior to the occurrence of the example coefficient	0	0	0	1	2	2	3	3
Subsequent to the occurrence of the example coefficient	0	0	0	0	1	2	2	3

A third type of context information is the sign bit. Since the sign-bit is coded right after the first on-bit, the tail indicates whether the sign information is known or not. Therefore, the sign-bit has no information context unless the tail is non-zero (recall that there are three possibilities for the sign: positive, negative, or unknown).

The context model of the system uses up to 11 bits to described the context. This number is not fully specified: only 1030 or 1031 contexts are actually used, including the sign bit contexts. The meaning of every bit position depends on the previous binary values. One example follows these rules:

If the tail-on bit of the present coefficient is zero (for head bits), then 1024 contexts from the tail-information bits of the parent and W coefficient and the tail-on bit of the NW, N, NE, E, SW, and S coefficients respectively.

Adaptive coding may be used for head bits. In some examples, a single context is used to provide some "run coding" of head bits. If the next 16 bits to be coded are all head bits and their N, S, E and W neighbors and parent all have tail-information 0, a single decision will be

coded. This decision indicates if any of the 16 bits to be coded has a one bit at the current bitplane. If there is no one bit, the 16 decisions normally coded can be skipped. If any of the next 16 coefficients contain their first significant bit, then 16 decisions are used one for each bit. This "look ahead" results in fewer calls to the binary entropy coder which results in higher speed and higher compression.

If the tail-on bit of the present coefficient is one (for tail bits), then three contexts from the tail-information bits present coefficient. Fixed probability coding may be used as discussed previously.

If the present bit of the present coefficient is the first non-zero magnitude bit, then the sign bit of the present coefficient is encoded immediately after. The context for the sign bit is 3 contexts from the N_tail-on bit and the N_sign bit, where if the N_tail-on bit is zero, then the N_sign bit is unknown. If the N_ sign bit is unknown, the sign is coded with the probability 0.5. Otherwise, the sign is coded adaptively.

In summary, an 11 bit number is created denoting the context from the information available from the current, neighboring, and parent coefficients in the same coding unit.

Figures 26A-D illustrate causal and non-causal coefficients that may be used to condition a coefficient P. Each of the templates illustrated include the use of both tail-on bits and tail-on information. While the tail-on bit of each coefficient provides 1 bit, the tail-on information of each coefficient comprises 2 bits. In Figure 26A, the total number of bits provided by the template is 8. In Figures 26B and 26C, the total number of bits provided by the template is 10.

Additional bits may be used to condition the head bits of coefficient P. In one embodiment, two additional bits may specify bit position as follows:

00 first bit (MSB) and second bit

01 third bit and fourth bit

10 fifth bit and sixth bit

11 other bits

It should be noted that other templates may be designed based on neighboring and parent coefficients. Furthermore, in one example, the coefficients used to condition coefficient P are causal, if not by position by bit plane.

In one embodiment, the S-transform parents are used for conditioning, not TS-transform parents. This reduces buffering needed for conditioning by saving low pass lines before continuing to code the next one. This is not advantageous where the order of entropy coding is important and encoder memory is not important.

Note that there is a tradeoff between having more contexts to create more skewed data and the adaptation efficiency as a result of less data within a context.

The tail bits that do not need conditioning do not have to be buffered (for conditioning). They can be coded immediately as soon as they are available. In such a case, the channel manager may send the bits out immediately to the channel.

Rather than coding the coefficients on the lowest level of the decomposition in the same way as the other coefficients

or not coding them at all, the coefficients may be coded using prediction coding, such as DPCM.

For tail coding bits, either fixed probabilities or adaptive ones may be used.

With respect to conditioning, the last bit may be conditioned, in part, on the second-to-last bit. Also, bits after the first "on" bit may be conditioned on how far they are from the first "on" bit.

In one example, some tail bits are coded adaptively. For example, when there are fewer than T tail bits in a coefficient (e.g. T=2, T=3), adaptive coding is used. The context for these bits include the bit position and any previously coded tail bits in the current coefficient. This is similar to the M-ary coding of centers taught by Langdon for DPCM data.

In an alternative example, some or all data is coded with a M-ary entropy coder instead of a binary entropy coder. M-ary coders include Tunstall, fixed Huffman, Adaptive Huffman, etc. For example, one **Huffman code could be used for head bits. Instead of coding head bits one bit at a time, a priority encoder may be used to determine the position of the first "on" bit. The bits in the binary representation of the position are then coded with a binary entropy coder.**

Horizon Context Model

The context model is shown in block diagram form in Figure 27. Context model 2700 contains the sign/magnitude unit 109 (Figure 2), and three units for processing the different bits in the coefficient. Based on the bit being coded, one of the three units is selected. A switch may be included to facilitate the switching

between the units in a hardware implementation. These units include a head bit block 2701, a sign bit block 2702, and a tail bit block 2703. The head bit block 2701, a sign bit block 2702, and a tail bit block 2703 model the head bits, the sign and the tail bits, respectively, as described above. The output of these three units is sent to the entropy coder 104 (Figure 1).

The coder may include optional control that saves states (optional), provides initial states and resets the coder (for instance, at the end of a coding unit).

The contexts defined above are used with an adaptive binary entropy coder with a few exceptions. The contexts of the head bits (present coefficient tail-on bit =0) and the sign bits when N_tail-on = 1 are allowed to adapt.

However, the bits after tail-on = 1 and the sign bits when N_tail-on = 0 are modeled by a stationary source. In these cases, the adaptation feature of the entropy coder is not necessary and, in fact, can be a source of compression inefficiency. For the following contexts a fixed (non-adaptive) state, described in terms of the states of the Q-coder is used.

Statistical models

The context is for coding the sign bit when N_tail-on = 0 (the sign of the N-coefficient is not known) is coded at the fixed Q-coder state 0 - probability approximately 0.5.

The context is for coding the first binary value after the first non-zero (tail-information = 1)) bit is coded at the fixed Q-coder state 4 - probability approximately 0.7.

The context is for coding the second and the third binary values after the first non-zero bit (tail information = 2) is coded at the fixed Q-coder state 3 - probability approximately 0.6.

The context is for coding the fourth and later binary values after the first non-zero bit (tail-information = 3) is coded at the fixed Q-coder state 0 - probability approximately 0.5.

In some examples, the entropy coding is reset after each coding unit, so the adaptation cost for contexts that are allowed to adapt (e.g., contexts used to encode binary values before the first on-bit) is significant. To keep this cost to a minimum, a set of initial states may be computed for these contexts from, for instance, some training data.

The following discussion assumes that the coefficients are 18-bits and that the input data has undergone a four level decomposition.

One example of the sign/magnitude unit 109 is shown in Figure 28 and converts input coefficients into a sign/magnitude format. Sign/magnitude unit 109 is coupled to receive 18 bits of the coefficients and includes an inverter 2801 and a multiplexer (MUX) 2802. The sign/magnitude unit 109 outputs a significance indication (e.g., a 5-bit value), the mantissa of the input coefficient, (e.g., 17 bits), the sign of the input coefficient 1 bit and an index from counter 2804, (e.g., 7 bits.)

MUX 2802 is coupled to receive 17 bits of the coefficient directly input into sign/magnitude unit 109 and an inverted version of the 17 bits from two's complementer 2801. Based on the sign bit (coefficient bit 17) received on the select input of MUX 2802, the positive of the two inputs is output as the mantissa.

Coding Alternatives

The binary entropy coder is given a context and the bit to be encoded.

For bitplane by bitplane coding, the reference system uses a carry-save style computation (on a general purpose computer) so the computation is done with a data format that is suitable for fast coding by bitplane. For instance, in such an implementation, a 32 bit processor may compute 1 bit of each of 32 coefficients in the same bit plane at the same time, instead of one entire coefficient. Using such an arrangement results in increased speed when coding by bitplanes.

Since a coding unit is encoded at a time and all the coefficients in a coding unit reside in the memory, there is no memory cost for the storage of context information, except what the adaptive binary entropy coder needs. For example, the Q-coder needs to keep the binary value of the LPS (least significant symbol) for all contexts and the current state for each context that is allowed to adapt. Since Q-coder has 30 states, a 6-bit number (1 bit for the LPS and 5 bits for states) is needed for each context. Therefore, the memory cost is 1024 x 5 + 1030 = 6150 bits of memory.

Note that there is no special signaling information necessary for the one-pass system described above. If the number of levels of decomposition were a variable, that would require at least 3 bits of header information. The header used but not counted in the compressed bits, is the following:

- Width, 2 bytes,
- Height, 2 bytes,
- Bits per pixel of input image, 1 bytes.

Memory Management

Memory management for coded data in the one pass system is presented for systems that store all of the data in memory and for systems that transmit data in a channel. In the one-pass system, coded data must be stored such that it can be accessed in an embedded causal fashion, so that less significant data can be discarded without losing more significant data. Since coded data is variable length, dynamic memory allocation can be used.

In one example, the embedded coding scheme uses 18 bitplanes and, thus, assigns 18 levels of significance to the data. The coder in a one-pass system is "embedded causal." That is, the decoding events corresponding to a bitplane do not require information from lower order bitplanes. In one example, all of the bits from one tree will be coded before any of the bits in the next tree are coded, so bits of different significance are not separated. For coders that do not use internal state, like Huffman coders, this is not a problem. However, many sophisticated compressors with better compression use internal state.

One way to solve this problem for these coders is to use 18 different coders, perhaps 18 Q-coder chips. A technique that would allow the use of 9 Q-coder chips is described in U.S. Patent No. 5,097,261 (Langdon, Jr.), entitled "Data Compression for Recording on a Record Medium," issued March 17, 1992. A better way uses a pipelined coder to implement different virtual codes with a single physical coder, such as that described in U.S. Patent No. 5,381,145, entitled "Method and Apparatus for Parallel Decoding and Encoding of Data", issued January 10, 1995. In such a coder, the multiple bit generator states for each probability are each assigned to a

part of the data. For example, each of 18 states could be assigned to a particular bitplane for 18 bit data. Registers in the shifter in the coder are also assigned to each part of the data. In the encoder, no interleaving is performed; each part of the data is simply bitpacked.

In systems either with multiple physical or virtual coders, memory is allocated to each part of the data. When compression is complete, a linked list describing the memory allocated plus the contents of the allocated memory is the result.

If the memory overflows, the memory allocation routing causes more important data to overwrite less important data. For example, the least significant bit of numeric data might be overwritten first. The information that describes how memory is allocated must be stored in addition to the coded data.

Figure 29 shows an example dynamic memory allocation unit for three categories of significance. Only three categories are described to avoid obscurity; typically, a larger number of categories, such as 8, 16 or 18, would be used. A register file (or other storage) holds a pointer for each category of significance plus another pointer for indicating the next free memory location. The memory is divided into fixed size pages.

Initially, each pointer assigned to a significance category points to the start of a page of memory and the free pointer points to the next available page of memory. Coded data, identified with a significance category, is stored at the memory location addressed by the corresponding pointer. The pointer is then incremented to the next memory location.

When the pointer reaches the maximum for the current page, the address of the start of the next free page stored in the free pointer is stored with the current page as a link. The part of the coded data memory or a separate memory or register file could be used for this purpose. Then the current pointer is set to the next free page. The free pointer is incremented. These steps cause a new page of memory to be allocated to a particular significance category and provide links to pages of memory containing data for a common significance category so that the order of allocation can be determined during decoding.

When all pages in the memory are in use and there is more data that is more significant than the least significant data in memory, memory reassignment may be performed. Three such reassignment techniques are described. In all three cases, memory assigned to the least significant data is reassigned to more significant data and no more least significant data is stored.

First, the page currently being used by the least significant data is simply assigned to the more significant data. Since most typical entropy coders use internal state information, all of the least significant data stored previously in that page is lost.

Second, the page currently being used by the least significant data is assigned to the more significant data. Unlike the previous case, the pointer is set to the end of the page and as more significant data is written to the page, the corresponding pointer is decremented. This has the advantage of preserving the least significant data at the start of the page if the more significant data does not require the entire page.

Third, instead of the current page of least significant data being reassigned, any page of least significant data may be reassigned. This requires that the coded data for all pages be coded independently, which may reduce the compression achieved. It also requires that the uncoded data corresponding to the start of all pages be identified. Since any page of least significant data can be discarded, greater flexibility in quantization is available.

The third alternative might be especially attractive in a system that achieves a fixed rate of compression over regions of the image. A specified number of memory pages can be allocated to a region of the image. Whether lessor significant data is retained or not can depend on the compression achieved in a particular region. Note that the memory assigned to a region might not be fully utilized if lossless compression required less than the amount of memory assigned. Achieving a fixed rate of compression on a region of the image can support random access to the image regions.

When compression is complete, the data may be transferred, if desired, to a channel or storage device in order of significance. The various links and pointers would then no longer be needed and multi-pass decoding could be performed. Alternatively, for one-pass decoding, the pointers to the data for each significance can be kept.

In some applications, some significance categories might not be used. For example, a 16-bit compressor might be used on a 12-bit medical image, so significance categories corresponding to bitplanes 15...12 would be unused. In implementations with large pages and many unused significance categories, this would waste memory (when the system does

not know in advance that some categories are unused), since memory does not have to be allocated to them. Another solution to this memory waste would be to use a small memory (or register) to hold a count for each significance category. The count would keep track of the number of "insignificant" decisions that occur before any other decision occurs. The memory required to store these counters must be "traded-off" against the memory used by unused significance categories.

The ability to write data into each page from both ends can be used to better utilize the total amount of memory available in the system. When all pages are allocated, any page that has sufficient free space at the end can be allocated for use from the end. The ability to use both ends of a page must be balanced against the cost of keeping track of the location where the two types of data meet. Note that this is different from the case where one of the data types was not significant and could simply be overwritten.

Improvements

In one embodiment, the reference system provides lossless compression with a small memory buffer and is capable of serving many different application and device environments. The following describes techniques to implement various features to enable the system to be more flexible for different applications and target devices. Note that choices in resolution, pixel depth, random access, quantization, etc. do not have to be made at encode time.

Data Arrangement

With respect to data arrangement, there are a number of options for arranging the image and coefficient data with the system.

As is discussed in more detail below, these options include, but are not limited to, the tiling of coded units, the number of levels of decomposition, the selection of wavelet transform filters, and the alignment of coefficients. As such, each of these could be a user or system designer controlled parameter.

As discussed above, one parameter may be the tiling of coded units. The height and width of the coding unit are defined with respect to trees.

For random access, the start of the coded data for each coding unit can be designated by pointers or markers in the codestream or pointers in the header. This would allow access to blocks not the width of the image.

Another parameter that may be controlled is the number of levels of decomposition. Varying the number of levels of decomposition varies the compression performance based on the fact that the more levels of decomposition results in better compression. Note that varying the number of decomposition levels also affects the memory requirements, as more levels requires more line buffers. More levels might be needed to target a resolution below full resolution. For example, if an original image is 2000 dpi, five levels of decomposition is needed to achieve about 63 dpi. This allows a high resolution scan to be displayed at close to real size on a monitor without decompression and subsampling.

The type of wavelet transform filters for the horizontal and vertical pass at each level may also be different. This allows for different memory requirements and compression performance. Note that the coefficient size

does not increase with more levels. Also, since the wavelet transform is an order N transform, and there is less data to transform as the levels increase, there is little computational cost for more levels.

Target devices for the embedded codestream

There are many possible application targets for a particular compressed codestream. It might be desirable to have a codestream that can be sent to a monitor with lower resolution but full pixel depth, a printer with full resolution but lower pixel depth, a fixed-rate real-time device with a limited channel, or a fixed-size limited memory device. It is possible that the same codestream is required to serve all of these needs. Figure 34 shows a generalization of the relative device characteristics a single application might serve.

Transmission or Decode Codestream Parser

The system of the present invention with enough speed at the encoder and decoder and enough bandwidth can extract the required data from the decompressed image. Furthermore, the encoder can create a codestream that is intended for one of the above devices. At the encoder, the image can be quantized or down sampled in the traditional fashion.

However, a virtue of the present invention is that, with the proper signaling, a codestream can be created that can be parsed before transmission or decoding without decompression for any of the above devices. Such a parser may be shown in Figures 35A and B. Referring to Figure 35A, a parser 3501 is shown receiving a lossless bitstream and generating a lossy bitstream. Referring to Figure 35B, a parser 3502 is

shown receiving a lossy bitstream and generating another lossy bitstream; however, the relationship between the output and the input in Figure 35B in such that the present invention has the property of being idempotent, which will be described in further detail below. Note that in the case of both parsers 3501 and 3502, the bit rate of data received as an input is greater than that being outputted.

Low Resolution, High Pixel Depth Embedded Target

If the target is a low resolution, high pixel depth embedded target, this application assumes that the target device has a lower spatial resolution than is available but the full pixel depth is required. Examples of a low resolution, high pixel depth embedded target are monitors. Using the codestream shown in Figure 24A, each bit-plane is decoded for as many higher level coefficients as needed. This requires the parser to truncate each bit-plane. To assist the parser, each bit-plane of each coding unit could have markers or pointers denoting the location where the truncation can occur. In such an embodiment, if more than one target resolution is desired, more markers or pointers are required. The bit-planes are coded independently so the entropy coder can be reset for the next bit-plane.

Another approach is to embed the data differently, such as shown in Figure 24B. Referring to Figure 24B, the target resolution coefficients within each coding unit is coded first followed by the bit-planes of the remaining high resolution coefficients. In this case, there is only one truncation necessary per coding unit and the entropy coder need not be reset. Markers or pointers can denote the desired truncation point.

High Resolution, Low Pixel Depth Embedded Target

If the target is a high resolution, low pixel depth embedded target, this application assumes that the target device requires the full resolution available, or more, but cannot use the full pixel depth. Examples of the high resolution, low pixel depth embedded target include low end printers and standard monitors (when images are more than 8 bits/plane). The codestream shown in Figure 24A is embedded in this order. Each coding unit is truncated at the point at the right number of bit-planes and the transform is performed on the quantized coefficients. There is a direct relationship between coefficient depth and pixel depth. Markers or pointers can denote the desired truncation point.

Alternately, if the codestream is embedded as shown in Figure 24B, then two markers or pointers are used to denote truncation, one for the low resolution bit-planes and one for the high resolution bit-planes. The two sets of bit-planes are coded independently to allow the entropy coder to be reset.

Yet another alternative is to decode some or all of the low resolution coefficients, as described with respect to the low resolution, high pixel depth embedded target, and possibly some data from the high resolution coefficients. Then perform the interpolating wavelet transform described below.

Fixed-Rate Embedded Target

If the target is a fixed-rate embedded target, this application assumes that a real-time constant pixel output must be maintained while using a

constrained channel. In this case, there is a certain maximum codestream data rate locally in time (minimum compression ratio). To achieve this goal, first, the coding units are chosen based on the amount of buffering available at the target device. This defines the locality over which the average compression ratio is to be achieved. Then each coding unit with more data than is allowed is truncated.

Note that if the codestream data rate does not exceed the maximum channel bandwidth the image is recovered losslessly. This is not true of any other fixed-rate system.

Fixed-size Embedded Target

If the target is a fixed-size embedded target, this application assumes that a fixed-size frame buffer is available to the compressed image data. Unlike the fixed-rate application, this requires a minimum compression rate averaged over the entire image, not just locally. Of course, the fixed-rate method could be used here, but, by using the concept of averaging over the entire image rather than locally, better bit allocation and image quality can be obtained.

If the coding unit contained the entire image, it would be trivial to truncate the data that overflows the buffer. If coding units are less than the entire image and all the coding units are truncated to the same number of bits, there is no guarantee that the truncation has uniformly removed the lowest importance levels. A simple solution is to record at encode time (or a later parsing time) the number of coded bits that each importance level contributes to the codestream for each coding unit or globally, or both. The recording can be done using simple counters. These

numbers are recorded in a header and can be used for deciding how to truncate each coding unit at transmission or storage time. The header contains an importance level and its corresponding number of bits. The header may also contain this information for each of the coding units that are contained in the stream. When deciding where to truncate, the same effect on each coding unit. For example, if due to memory constrainst, it is determined that one and a half importance levels are to be discarded, then one and a half importance levels from each coding unit are discarded. This allows the effect of the truncation to be spread across the coding units in a uniform manner.

Achieving a fixed-size compressed image can be achieved at encode time as well. The memory is divided into segments for importance levels. If the memory is about to overflow, lessor important segments are overwritten with more important data.

Note that if the compressed data does not overflow the memory buffer the image is recovered losslessly. This is not true of any other fixed-size system.

Interpolating with the Wavelet Transform

Wavelets can be used to interpolate images to higher resolution. The results are visually quite comparable to bi-cubic spline techniques. If the compressed data is already in the form of wavelet coefficients, the effective additional computation for interpolation is less than bi-cubic spline.

Imagine that all the coefficients of N level decomposition are available. By creating a new lowest level of coefficients, by padding with

zeros or some other method, and then performing a N+1 level wavelet reconstruction, the new image is a 2:1 interpolated version of the original.

This method can also be used with the systems in which there are target devices for the embedded codestream, especially for high resolution, low pixel depth target devices. The coding units are truncated so only the low resolution coefficients are present (or only the low resolution coefficients and a few bits of some or all of the high resolution coefficients). The coefficients are padded to the higher resolution and the reconstruction is performed.

Using a Channel

In a system where data is transmitted in a channel instead of being stored in a memory and fixed size pages of memory are used (but only one page per significance category is needed), when a page of memory is full, it is transmitted in the channel, and memory location can be reused as soon as they are transmitted. In some applications, the page size of the memory can be the size of data packets used in the channel or a multiple of the packet size.

In some communications systems, for example ATM (Asynchronous Transfer Mode), priorities can be assigned to packets. ATM has two priority levels, priority and secondary. Secondary packets are only transmitted if sufficient bandwidth is available. A threshold can be used to determine which significance categories are priority and which are secondary. Another method would be to use a threshold at the encoder to not transmit significance categories that were less significant than a threshold.

Thus, the memory manager may control the storage of compressed data in a fixed size memory.

That is, the memory manager divides the compressed data into different importance levels. When the memory is full, lessor important data is overwritten by more important data.

In order to manage a channel using a limited amount of buffer memory (e.g., a fixed-rate),

all data is transmitted if sufficient bandwidth is available; otherwise, lessor importance data is discarded and only more important data is transmitted.

Figure 30 illustrates a system utilizing a channel manager. Referring to Figure 30, wavelet transform 3001 generates coefficients. These coefficients are subjected to context model 3002. Context model 3002 is coupled to a channel manager 3003 that includes a buffer memory. The channel manager 3003 is coupled to a limited bandwidth channel 3004.

Channel manager 3003 controls the rate at which data is output to channel 3004. As data is received into its buffer memory, channel manager 3003 determines if the amount of data is greater than the bandwidth of channel 3004. If the amount of data is not greater, then channel manager 3003 outputs all of the data. On the other hand, if the amount of data received into the buffer memory is greater than the channel bandwidth, then channel manager 3003 discards information in its buffer memory to match the bandwidth of channel 3004.

Channel 3004 may indicate its bandwidth to channel manager 3003. In another embodiment, channel manager 3003 may dynamically determine the bandwidth of channel 3004 based on the amount of time that it takes channel 3004 to send a predetermined unit (e.g., packet) of data

through channel 3004. That is, the channel bandwidth can be treated as dynamic if desired.

Channel manager 3003 operates on an image that is broken up into tiles (or bands). This is a "tile dominant over importance" scheme in contrast to the fixed size memory manager where tiling and importance are somewhat independent. Each tile is separately coded and divided by importance levels and fixed size pages. Therefore, all the coded data for each tile is grouped together. Coded data within each tile is tagged by importance level.

The buffer memory in channel manager 3403 is preferably at least two (or perhaps three) times the size of the channel's packet size and several times (perhaps four times) larger than the expected compressed data size for a tile.

A fixed maximum amount of the buffer memory is assigned to a tile. The maximum amount is matched to the bandwidth of the channel. Buffer memory is broken into fixed size segments and allocated as needed. If the memory usage reaches the maximum allowed, segments are reassigned as in the management of fixed size memory system.

Figure 31 illustrates an example of buffer memory usage. Referring to Figure 31, a circular buffer memory has multiple fixed sized segments 3101 that are divided into multiple fixed size packets 3102 for channel input/output. As shown, different tiles of data may occupy the same packet of memory. The different tiles may represent different importance levels. As the packet size amount of buffer space is used, channel manager 3103 indicates to the context model to output the data to channel 3104 (Figure 30). As shown, tile N-2 and part of tile N-1

would be output as the current packet. Thus, a packet size amount of memory is allocated and filed in order to match the bandwidth of the channel.

If the buffer does not fill up, the extra memory may be used for future tiles. In one embodiment, to avoid noticeable tile boundaries at the start of a difficult to compress region versus the next block, only some fraction (1/2, 1/3, etc.) of the extra is used by the next tile.

The channel manager may be used where data can only be transmitted in a certain period of time. Using such a channel manager, the data transmission occurs during the time period regardless of the complexity, because the data is embedded based on its importance.

Alternate Example of the Channel Manager

One goal of the channel manager is to use minimal memory. Where the channel manager does not contain buffer memory, the following may be used:

```
for each coding unit
    for each bitplane do
        for each frequency do
            for each spatial location do
```

The coder (or set it to a known state) is reset at the start of each band. A band may comprise 16 lines for a four level decomposition if the band memory is to be reduced.

Figure 32 illustrates a bitstream using the above method. Referring to Figure 32, the bitstream is divided into fixed size segments, which are

channel packets, disk sectors or whatever is a reasonable amount of buffer for the channel. Note that this division may be no more than a logical division during encoding; the encoder can output using no buffering if desired. Each fixed size segment includes an indication of the most important data in the segment.

The structure of a segment is shown in Figure 33. Referring to Figure 33, the bitstream for one segment includes coded data 3301, an optional pointer(s) or ID 3302 and a level of the most important data in the segment 3303. Bit field 3303 may comprise 2 to 5 bits. If the most important level is 0 (the most important one), the next to last M bits of the segment is a pointer that tells where in the segment the level 0 data starts at. Note that the first segment of data can be entirely coded data, no overhead is needed.

The starting point for each band may be identified using restart markers, similar to those used in the JPEG standard. However, the marker used should be that symbol that occurs least often during coding.

Now again considering Figure 31, assume it is desired to decompress only some importance levels (perhaps only the most important level). Decompression starts with the first segment. For segment 2, the "level of the most important data in segment" is checked, and perhaps the entire segment can be skipped where the most important level contained in the segment is less than the level(s) being decompressed. For the third segment, the pointer is used to find the start of band 2, and decompression of band 2 can begin.

Note that to ensure that all the most significant data in a segment is obtained, it might be required to decompress the entire segment, particularly when more than one band falls in the segment.

By selectively decompressing only a predetermined number of significant bands, a preview image maybe obtained. This may be advantageous when data is in embedded form and lossy versions of lossless data are desired.

Depending on the desired access and quantization possibilities, and whether or not the time to decompress an entire band is important, the optional pointer(s) or ID at the end of the segment can contain:

- A next segment pointer for fixed size memory management.
- An ID for the segment or ID of band(s) contained. (Supports channel quantization, would indicate if segment 2 were dropped for example)
- The number of different bands that the segment contains data for (or at least a bit indicating that a band contains more than two segments). (Supports not decompressing entire segments after decompressing the desired data).

An example of the overhead for a band of 512x16 pixels, consider 8-bit image having 2:1 lossless compression, and a segment size of 512 bytes. Note that a band typically compresses to 8 segments. For 32 importance levels, 5 bit tags are used. Assume pointers are on byte boundaries, so 9-bit pointers are used. Therefore, there are 49 overhead bits/(32K compressed bits + 49) representing a total of 0.15%.

<u>Near-lossless</u>

One concept of near lossless compression is based on absolute error in the reconstructed pixel values. Hence, in a near-lossless compressed image with an absolute error of 1 it is guaranteed that no pixel value in the decompressed image differs from the original by more than 1 unit of pixel value. This is an absolute definition independent of the pixel depth or the dynamic range of the image. An obvious and, under some reasonable assumptions, optimal to such a system is to keep the compression/decompression part lossless and use preprocessing and post-processing schemes to achieve near-lossless. This method has been adopted in this implementation.

The near-lossless compressed image with an absolute error of e is achieved by the quantization method that maps every 2e+1 consecutive integers to their middle integer. For example, for error equal to 3, the pixel values are quantized such that 0 through 6 is mapped to 3 and 7 through 13 is mapped to 10, and so on. The quantized image as such is not suited for a transform-based compression system. Hence, it is mapped one-to-one (losslessly) into an image of lower dynamic range or depth, called the shallow image. This is done by mapping the middle values (representative values) to consecutive integers preserving the order. Mathematically given a pixel value x, it is quantized to:

$$q(x) = \left(\left\lfloor \frac{x}{2e+1} \right\rfloor + 1 \right) . e$$

The one-to-one mapping of the representative values to the shallow image values is,

$$p(x) = \left\lfloor \frac{x}{e} \right\rfloor - 1$$

The inverse of the one-to-one mapping p which maps the shallow image values back to representative values is,

$$p^{-1}(x) = e \cdot (x + 1)$$

Quantization (q(x))) followed by the mapping to the shallow image values (p(x)) is the pre-processing operation, which proceeds the lossless compression. The map from the shallow image values to the representative values form the post-processing operation, which follows the lossless decompression.

Transform domain quantization can also be used. Many coefficients have an effect on peak error propagates through multiple levels of the transform. It is easier to determine the effect on peak error for the high-pass coefficients that have no children.

Consider a one-dimensional signal which is to be encoded with a maximum peak error of ±E. This can be achieved by quantizing the finest detail high-pass coefficients to ±2E. For a two-dimensional signal, since there are two applications of the high-pass filter, the finest detail HH coefficients can be quantized to ±4E.

An alternative to using quantization of the input image is to control the decisions to the entropy coder. One example is the following. For each coefficient, if setting the coefficient to zero would not cause the

error in any pixel affected by that coefficient to exceed the maximum error, the coefficient is set to zero. In some implementations only particular coefficients will be tested, perhaps only the AC coefficients that have no children. Coefficients can be considered with a greedy strategy where one is considered at a time. Other strategies can consider small groups of coefficients and choose to zero the largest possible subset of the group.

As described above, quantization is achieved by the embedding function and is optimized to maximize performance with respect to a quantitative metric such a RMSE. In one example, the quantization of the various coefficients is performed to achieve improved results with respect to the Human Visual System. In such a case, little modification of the embedding scheme is required. For instance, the coefficients are shifted to change the relation between them by a factor of two and/or to represent the number in a different type of numbering system such as Gray code.

The compressed wavelet system may be useful in an image editing situation. In the prior art, applying image processing functions to a full resolution print image is time consuming and makes interactive processing difficult.

If an image editing system saved tiles that are compressed, it could very quickly apply operations to the scale representation (the very low pass) for the user to evaluate. This can be done quickly because only the displayed pixels are operated on. It is only an approximation of the final result since the actual full resolution pixels affect the output. The user will therefore zoom in (perhaps on some text) on various portions of the image. As the user does, the image editing

system applies the operation to that part of the image. To facilitate this, a tree is stored that contains the compressed coefficients and information about which processing operations have been applied and which still need to be applied.

The importance levels may be redefined to permit lossless compression in a defined window and lossy compression for the rest of the image. The window could be fixed or selectable by the user. There could be multiple windows of different importance. In one example, the window is as small as 48x48 blocks, although it should be possible to have much finer occurrence even down to the two pixel level.

A possible application of this is satellite imagery, where satellites use a lossless window on data so that statistical studies are not messed up with JPEG artifacts, but the lossy compression allows a much wider field of view than would be possible with lossless compression.

In one system, the user draws arbitrary boxes on an image and specifies the relative importance of the data in the box. Once a box has been drawn, software increases the size of the box bigger to the smallest size which meets the required constraints and contains the user box. The file header would contain information about the boxes used and the importance level. The encoder and decoder would then provide more resolution to the coefficients in the important boxes as coding/decoding proceeds. For the satellite imagery case, the important window is likely to be predefined.

Idempotent Operation

For a lossy compressor, in general, idempotent operation as

DCDCI = DCI, where I is the image, C the compression operation and D the decompression operation. In the present invention, when data is compressed to X bits and then decompressed, they should be able to be recompressed to X bits and have the original X bits. There is an even stronger version of idempotent for an embedded system. In one embodiment, an image when compressed to X bits, decompressed and recompressed to Y bits with Y < X is the same as if the original image is compressed to Y bits.

This is important because compression and processing causes images to drift farther from the original. If the compressor is idempotent, then multiple lossy compression decompression cycles do not affect the data. In the present invention, it does not matter how many times data is compressed and decompressed at the same compression ratio. Also, a lossy input to a parser subjected to further quantization produces an identical result to the case when a lossless input is used. Thus, the reference system comprises a transform-based idempotent system that includes a wavelet transform, a context model, and an entropy coder, such that coefficients are described and stored in an order such that removing information does not change the description for prior coefficients.

Applications

The present invention may be used for a number of applications, some of which are mentioned as examples below. Specifically, high-end applications with high-resolution and deep pixels and applications that are artifact intolerant can use the present invention. The present invention enables high-end applications maintain the highest quality in high-quality

environments while applications with more limited bandwidth, data storage, or display capabilities can also use the same compressed data. This is precisely the device-independent representation that is commonly being required of modern imaging applications such as web browsers.

The superior lossless compression performance of the present invention on deep pixel images (10 bits to 16 bits per pixel) is ideal for medical imagery. In addition to the lossless compression, the present invention is a true lossy compressor without many of the artifacts known to block-based compressors. Lossy artifacts derived by using the present invention tend to be along sharp edges where they are often hidden by the visual masking phenomena of the Human Visual System.

The present invention may be used in applications involving the pre-press industry in which the images tend to be very high resolution and have high pixel depth. With the pyramidal decomposition of the present invention, it is easy for the pre-press operator to perform image processing operations on a lower resolution lossy version of the image (on a monitor). When satisfied, the same operations can be performed on the lossless version.

The present invention is also applicable for use in facsimile document applications where the time of transmission required without compression is often too long. The present invention allows very high image output from fax machines with different spatial and pixel resolutions. Since transmission time is a premium in this application, the interpolation feature of the present invention is useful.

The present invention may be used in image archival systems that require compression, particularly for increasing storage capacity. The device

independent output of the present invention is useful because the system can be accessed by systems with different resources in bandwidth, storage, and display. Also, progressive transmission capabilities of the present invention are useful for browsing. Lastly, the lossless compression is desirable for output devices in image archiving systems may be provided by the present invention.

The hierarchical progressive nature in the lossless or high quality lossy data stream of the present invention make it ideal for use in the World Wide Web, particularly where device independence, progressive transmission, high quality, and open standards are imperative.

The present invention is applicable to satellite images, particularly those that tend to be deep pixel and high resolution. Furthermore, satellite imagery applications have limited bandwidth channel. The present invention allows flexibility and with its progressive transmission qualities, it may be used to allow humans to browse or preview images.

"Fixed-rate", limited-bandwidth applications such as ATM networks need ways of reducing data if it overflows the available bandwidth. However, there should be no quality penalty if there is enough bandwidth (or the compression is high enough). Likewise, "fixed-size" applications like limited-memory frame stores in computers and other imaging devices need a way to reduce data if the memory fills. Once again, there should be no penalty for an image that can be compressed losslessly into the right amount of memory.

The embedded codestream of the present invention serves both of these applications. The embedding is implicit to allow the codestream to be trimmed or truncated for transmission or storage of a lossy image. If no trimming or truncation is required, the image arrives losslessly.

Lossy compression provided by the present invention is achieved by embedded quantization. That is, the codestream includes the quantization. The actual quantization (or visual importance) levels can be a function of the decoder or the transmission channel, not necessarily the encoder. If the bandwidth, storage, and display resources allowed it, the image is recovered losslessly. Otherwise, the image is quantized only as much as required by the most limited resource.

The embedding used in the present invention may be progressive, specifically by bitplane, i.e. MSB followed by lesser bits. Both the spatial and wavelet domains can be decomposed progressively, although the present invention is progressive in the wavelet domain specifically. For applications that have spatial resolution but lower pixel resolution, such as printers, the progressive ordering of the bits in the present invention is ideal. These features are available with the same codestream.

Reference is directed to copending application no. 9613320.2 (Publication no. 2,303,031) which claims encoders and methods of encoding described above.

C L A I M S

1. A method of using an embedded codestream comprising:

identifying a target device to receive data in the embedded codestream;

decoding each bit-plane to provide data to the target device, said step of decoding comprising the step of truncating each bit-plane in the embedded codestream for data necessary to support the target device.

2. A method according to claim 1, wherein each bit-plane is truncated based on an indication in each coding unit denoting a location where truncation may occur.

3. A method according to claim 1, wherein each bit-plane is truncated based on one of a plurality of indications in each coding unit denoting locations where truncation may occur, said step of truncating further comprising the step of selecting one of the indications based on the target device.

4. A method according to claim 2 or 3, wherein the or each indication comprises a marker.

5. A method according to claim 2 or 3, wherein the or each indication comprises a pointer.

6. A method according to any one of the preceding claims, wherein the step of truncating comprises truncating target resolution coefficients, coded separately in each coding unit, from the embedded codestream.

7. A method according to any one of claims 1 to 6, wherein the target device comprises a low resolution, high pixel depth embedded target such that the step of decoding decodes as many higher level coefficients as needed to achieve full pixel depth and low spatial resolution of the target device.

8. A method according to any one of claims 1 to 6, wherein the target device comprises a high resolution, low pixel depth embedded target and truncating each coding unit at a number of bit-planes and inverse wavelet transforming the non-truncated data of each coding unit to achieve the low pixel depth and high spatial resolution of the target device.

9. A method according to any one of claims 1 to 8, further comprising the steps of:

selecting coding units based on an amount of available buffering at the target device; and

truncating each coding unit with more data than available buffering.

10. A method according to any one of claims 1 to 9,

wherein the step of truncating further comprises the steps of:-

determining a uniform amount to truncate each coding unit;

truncating at least a portion of at least one importance level in each coding unit.

11. A method according to claim 10, wherein the step of truncation is performed using information in a header of the codestream setting forth importance level information.

12. A method according to claim 10, wherein the step of truncation is performed using information in a header of the codestream setting forth importance level information for each coding unit in the codestream.

13. A method according to claim 10, 11 or 12, wherein the steps are performed after encode time.

14. A method of using an embedded codestream substantially as hereinbefore described with reference to the accompanying drawings.

The Patent Office

Application No:	GB 9712513.2	**Examiner:**	Keith Williams
Claims searched:	1-14	**Date of search:**	18 September 1997

Patents Act 1977
Search Report under Section 17

Databases searched:

UK Patent Office collections, including GB, EP, WO & US patent specifications, in:

UK Cl (Ed.O): H4F (FRW); H4P (PDCFT, PDCFX)

Int Cl (Ed.6): H03M 7/30; H04N 1/41, 7/30, 7/50

Other: Online WPI, INSPEC, JAPIO

Documents considered to be relevant:

Category	Identity of document and relevant passage	Relevant to claims
	None	

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art.
Y	Document indicating lack of inventive step if combined with one or more other documents of same category.	P	Document published on or after the declared priority date but before the filing date of this invention.
&	Member of the same patent family	E	Patent document published on or after, but with priority date earlier than, the filing date of this application.