jc714 U.S. PTO
09/579221
05/26/00

ISSUE CLASSIFICATION	
Class	382
Subclass	232

PATENT NUMBER

6597812

6597812

U.S. UTILITY Patent Application

O.I.P.E.	PATENT DATE
SCANNED CIH Q.A. UA8	JUL 22 2003

APPLICATION NO.	CONT/PRIOR	CLASS	SUBCLASS	ART UNIT	EXAMINER
09/579221	D	~~710~~ 382	232	~~2782~~ 2623	Wu, J

APPLICANTS

James Fallon
Steven Bo

TITLE

System and method for lossless data compression and decompression

PTO-2040
12/99

ISSUING CLASSIFICATION

ORIGINAL		CROSS REFERENCE(S)						
CLASS	SUBCLASS	CLASS	SUBCLASS (ONE SUBCLASS PER BLOCK)					
382	232	382	245					
INTERNATIONAL CLASSIFICATION		341	51					
[illegible]								
						☐ Continued on Issue Slip Inside File Jacket		

☐ TERMINAL DISCLAIMER	DRAWINGS			CLAIMS ALLOWED	
	Sheets Drwg. 6	Figs. Drwg. 6	Print Fig. FIG 1	Total Claims 30	Print Claim for O.G. claim 1
☐ The term of this patent subsequent to ________ (date) has been disclaimed	________ (Assistant Examiner) ________ (Date)			NOTICE OF ALLOWANCE MAILED 2-28-03	
☐ The term of this patent shall not extend beyond the expiration date of U.S Patent. No ________	JINGGE WU PATENT EXAMINER ________ (Primary Examiner) 2/26/03 (Date)			ISSUE FEE CW Amount Due $650-00 Date Paid 6-2-03	
☐ The terminal ____ months of this patent have been disclaimed.	________ (Legal Instruments Examiner) 3-5-03 (Date)			ISSUE BATCH NUMBER	

WARNING:
The information disclosed herein may be restricted. Unauthorized disclosure may be prohibited by the United States Code Title 35, Sections 122, 181 and 368. Possession outside the U.S. Patent & Trademark Office is restricted to authorized employees and contractors only.

Form PTO-436A
(Rev. 6/99)

FILED WITH: ☐ DISK (CRF) ☐ FICHE ☐ CD-ROM
(Attached in pocket on right inside flap)

FORMAL DRAWINGS

(FACE)

SAP AMERICA, INC. ET AL.
EXHIBIT 1015

PATENT APPLICATION

09579221

jc714 U.S. PTO
09/579221
05/26/00

JUN 0 9 00 16
INITIALS ______

CONTENTS

Date Received (Incl. C. of M.) or Date Mailed

1. Application 6 prints papers.
2-27-03 2. Allow. ______ 2-28-03
6-12-03 3. Formal [illegible] 6 shts) set 6·2-03
4. ______ ______
5. ______ ______
6. ______ ______
7. ______ ______
8. ______ ______
9. ______ ______
10. ______ ______
11. ______ ______
12. ______ ______
13. ______ ______
14. ______ ______
15. ______ ______
16. ______ ______
17. ______ ______
18. ______ ______
19. ______ ______
20. ______ ______
21. ______ ______
22. ______ ______
23. ______ ______
24. ______ ______
25. ______ ______
26. ______ ______
27. ______ ______
28. ______ ______
29. ______ ______
30. ______ ______
31. ______ ______
32. ______ ______
33. ______ ______
34. ______ ______
35. ______ ______
36. ______ ______
37. ______ ______
38. ______ ______
39. ______ ______
40. ______ ______
41. ______ ______

Date Received (Incl. C. of M.) or Date Mailed

42. ______ ______
43. ______ ______
44. ______ ______
45. ______ ______
46. ______ ______
47. ______ ______
48. ______ ______
49. ______ ______
50. ______ ______
51. ______ ______
52. ______ ______
53. ______ ______
54. ______ ______
55. ______ ______
56. ______ ______
57. ______ ______
58. ______ ______
59. ______ ______
60. ______ ______
61. ______ ______
62. ______ ______
63. ______ ______
64. ______ ______
65. ______ ______
66. ______ ______
67. ______ ______
68. ______ ______
69. ______ ______
70. ______ ______
71. ______ ______
72. ______ ______
73. ______ ______
74. ______ ______
75. ______ ______
76. ______ ______
77. ______ ______
78. ______ ______
79. ______ ______
80. ______ ______
81. ______ ______
82. ______ ______

(LEFT OUTSIDE)

SEARCHED

Class	Sub.	Date	Exmr.
382	232-251	2/25/03	JW
341	51-99		
348	400.16 400.2		
375	240-341		

INTERFERENCE SEARCHED

Class	Sub.	Date	Exmr.
382	232 245	1/26/03	JW
341	51		

SEARCH NOTES
(INCLUDING SEARCH STRATEGY)

	Date	Exmr.
EAST	2/24/03 2/25/03	JW

(RIGHT OUTSIDE)

ISSUE SLIP STAPLE AREA (for additional cross references)

POSITION	INITIALS	ID NO.	DATE
FEE DETERMINATION	PS	66621	6/2
O.I.P.E. CLASSIFIER			
FORMALITY REVIEW	DW	72346	7-21-10
RESPONSE FORMALITY REVIEW			

INDEX OF CLAIMS

✓	Rejected	N	Non-elected
=	Allowed	I	Interference
— (Through numeral)	Canceled	A	Appeal
÷	Restricted	O	Objected

Claim Final	Claim Original	Date 11/26/10								
1	1	=								
2	2									
3	3									
4	4									
5	5									
6	6									
7	7									
8	8									
9	9									
10	10									
11	11									
12	12									
13	13									
14	14									
15	15									
16	16									
17	17									
18	18									
19	19									
20	20									
21	21									
22	22									
23	23									
24	24									
25	25									
26	26									
27	27									
28	28									
29	29									
30	30									
	31									
	32									
	33									
	34									
	35									
	36									
	37									
	38									
	39									
	40									
	41									
	42									
	43									
	44									
	45									
	46									
	47									
	48									
	49									
	50									

Claim Final	Claim Original	Date								
	51									
	52									
	53									
	54									
	55									
	56									
	57									
	58									
	59									
	60									
	61									
	62									
	63									
	64									
	65									
	66									
	67									
	68									
	69									
	70									
	71									
	72									
	73									
	74									
	75									
	76									
	77									
	78									
	79									
	80									
	81									
	82									
	83									
	84									
	85									
	86									
	87									
	88									
	89									
	90									
	91									
	92									
	93									
	94									
	95									
	96									
	97									
	98									
	99									
	100									

Claim Final	Claim Original	Date								
	101									
	102									
	103									
	104									
	105									
	106									
	107									
	108									
	109									
	110									
	111									
	112									
	113									
	114									
	115									
	116									
	117									
	118									
	119									
	120									
	121									
	122									
	123									
	124									
	125									
	126									
	127									
	128									
	129									
	130									
	131									
	132									
	133									
	134									
	135									
	136									
	137									
	138									
	139									
	140									
	141									
	142									
	143									
	144									
	145									
	146									
	147									
	148									
	149									
	150									

If more than 150 claims or 10 actions
staple additional sheet here

(LEFT INSIDE)

US006597812B1

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

(10) **Patent No.:** **US 6,597,812 B1**
(45) **Date of Patent:** **Jul. 22, 2003**

(54) **SYSTEM AND METHOD FOR LOSSLESS DATA COMPRESSION AND DECOMPRESSION**

(75) Inventors: **James J. Fallon**, Armonk, NY (US); **Steven L. Bo**, Bayside, NY (US)

(73) Assignee: **Realtime Data, LLC**, New York, NY (US)

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

(21) Appl. No.: **09/579,221**

(22) Filed: **May 26, 2000**

Related U.S. Application Data

(60) Provisional application No. 60/136,561, filed on May 28, 1999.

(51) **Int. Cl.**^{7} **G06K 9/36**

(52) **U.S. Cl.** **382/232**; 382/245; 341/51

(58) **Field of Search** 382/232–251; 341/51–79; 348/420.11–424.2; 375/240–241

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,870,036 A	*	2/1999	Franaszek et al.	341/51
5,883,975 A	*	3/1999	Narita et al.	382/232
5,955,976 A	*	9/1999	Heath	341/87
6,195,024 B1	*	2/2001	Fallon	341/51
6,489,902 B2	*	12/2002	Heath	341/87

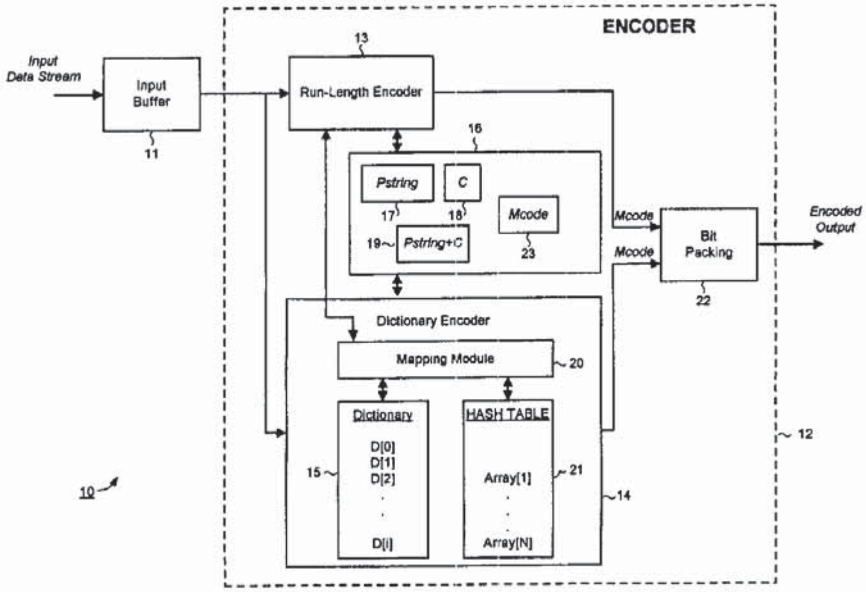
* cited by examiner

Primary Examiner—Jingge Wu
(74) *Attorney, Agent, or Firm*—F. Chau & Associates, LLP; Frank V. DeRosa, Esq.

(57) **ABSTRACT**

Systems and methods for providing lossless data compression and decompression are disclosed which exploit various characteristics of run-length encoding, parametric dictionary encoding, and bit packing to comprise an encoding/decoding process having an efficiency that is suitable for use in real-time lossless data compression and decompression applications. In one aspect, a method for compressing input data comprising a plurality of data blocks comprises the steps of: detecting if the input data comprises a run-length sequence of data blocks; outputting an encoded run-length sequence, if a run-length sequence of data blocks is detected; maintaining a dictionary comprising a plurality of code words, wherein each code word in the dictionary is associated with a unique data block string; building a data block string from at least one data block in the input data that is not part of a run-length sequence; searching for a code word in the dictionary having a unique data block string associated therewith that matches the built data block string; and outputting the code word representing the built data block string.

30 Claims, 6 Drawing Sheets



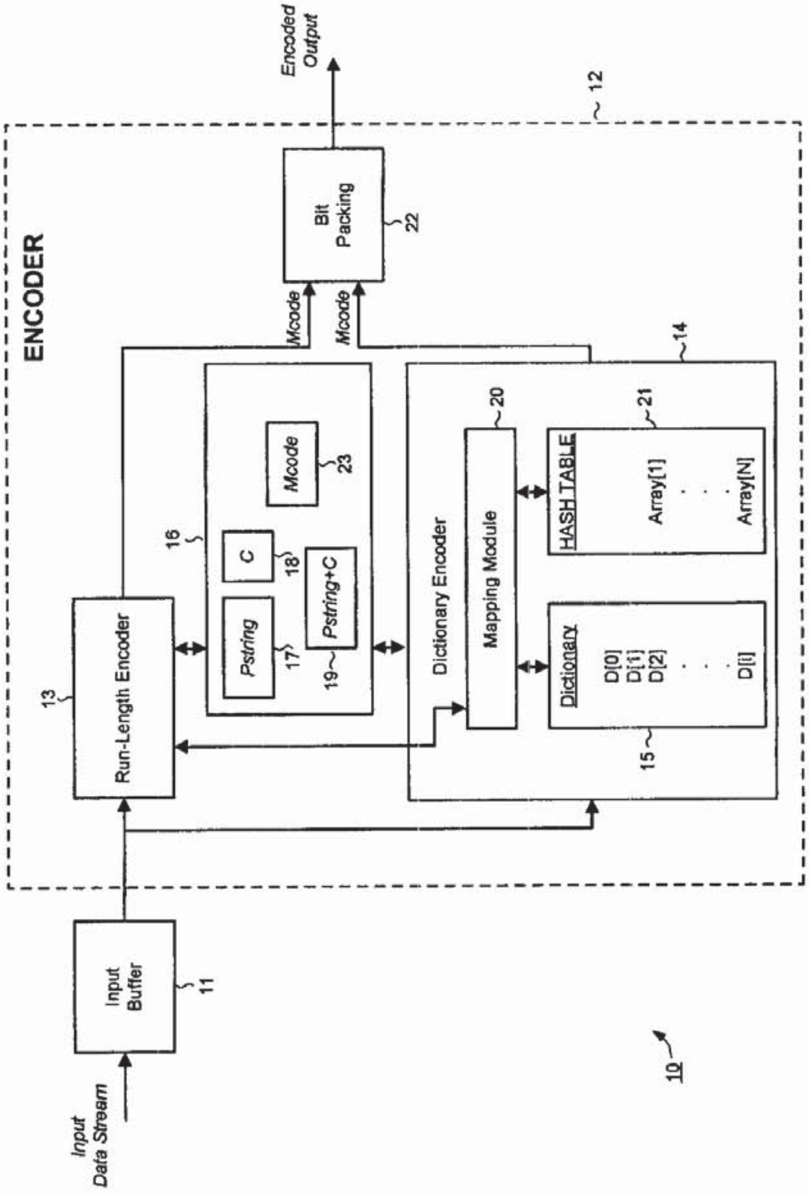

ENCODER
Input Data Stream
Input Buffer
11
13
Run-Length Encoder
16
Pstring
17
C
18
Pstring+C
19
Mcode
23
Mcode
Mcode
Bit Packing
22
Encoded Output
12
14
Dictionary Encoder
Mapping Module
20
Dictionary
D[0]
D[1]
D[2]
D[i]
15
HASH TABLE
Array[1]
Array[N]
21
10

FIGURE 1

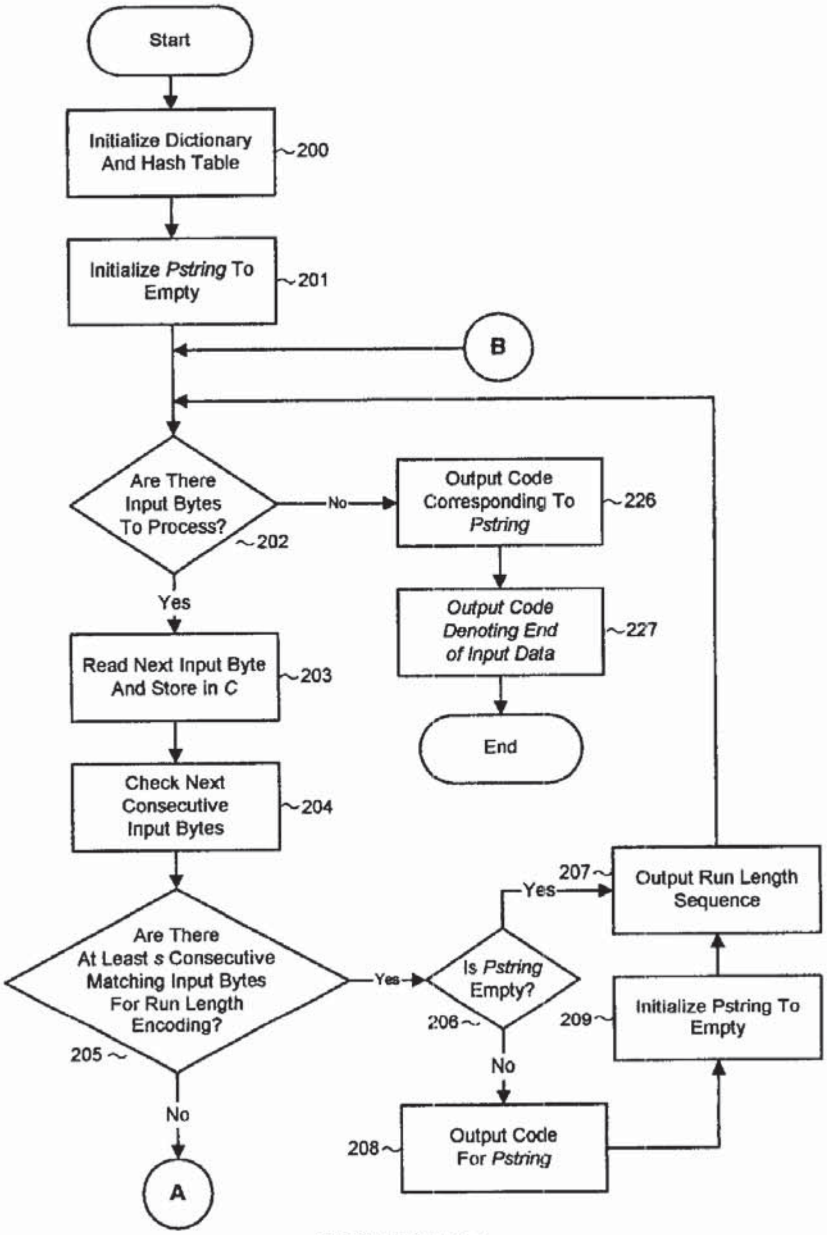

Start
Initialize Dictionary And Hash Table
200
Initialize *Pstring* To Empty
201
B
Are There Input Bytes To Process?
202
No
Output Code Corresponding To *Pstring*
226
Output Code Denoting End of Input Data
227
End
Yes
Read Next Input Byte And Store in *C*
203
Check Next Consecutive Input Bytes
204
Are There At Least *s* Consecutive Matching Input Bytes For Run Length Encoding?
205
Yes
Is *Pstring* Empty?
206
Yes
Output Run Length Sequence
207
Initialize Pstring To Empty
209
No
Output Code For *Pstring*
208
No
A

FIGURE 2A

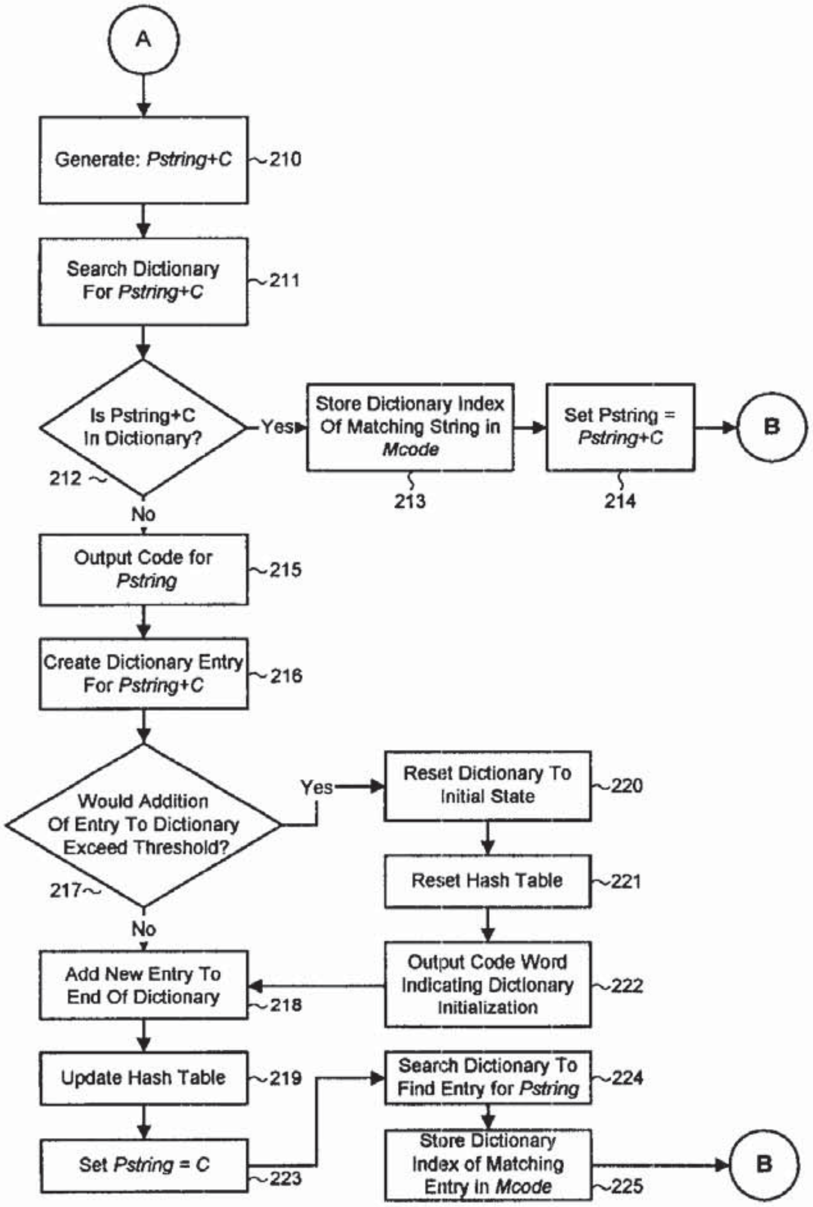

A
Generate: *Pstring+C*
210
Search Dictionary For *Pstring+C*
211
Is Pstring+C In Dictionary?
212
Yes
Store Dictionary Index Of Matching String in *Mcode*
213
Set Pstring = *Pstring+C*
214
B
No
Output Code for *Pstring*
215
Create Dictionary Entry For *Pstring+C*
216
Would Addition Of Entry To Dictionary Exceed Threshold?
217
Yes
Reset Dictionary To Initial State
220
Reset Hash Table
221
Output Code Word Indicating Dictionary Initialization
222
No
Add New Entry To End Of Dictionary
218
Update Hash Table
219
Set *Pstring = C*
223
Search Dictionary To Find Entry for *Pstring*
224
Store Dictionary Index of Matching Entry in *Mcode*
225
B

FIGURE 2B

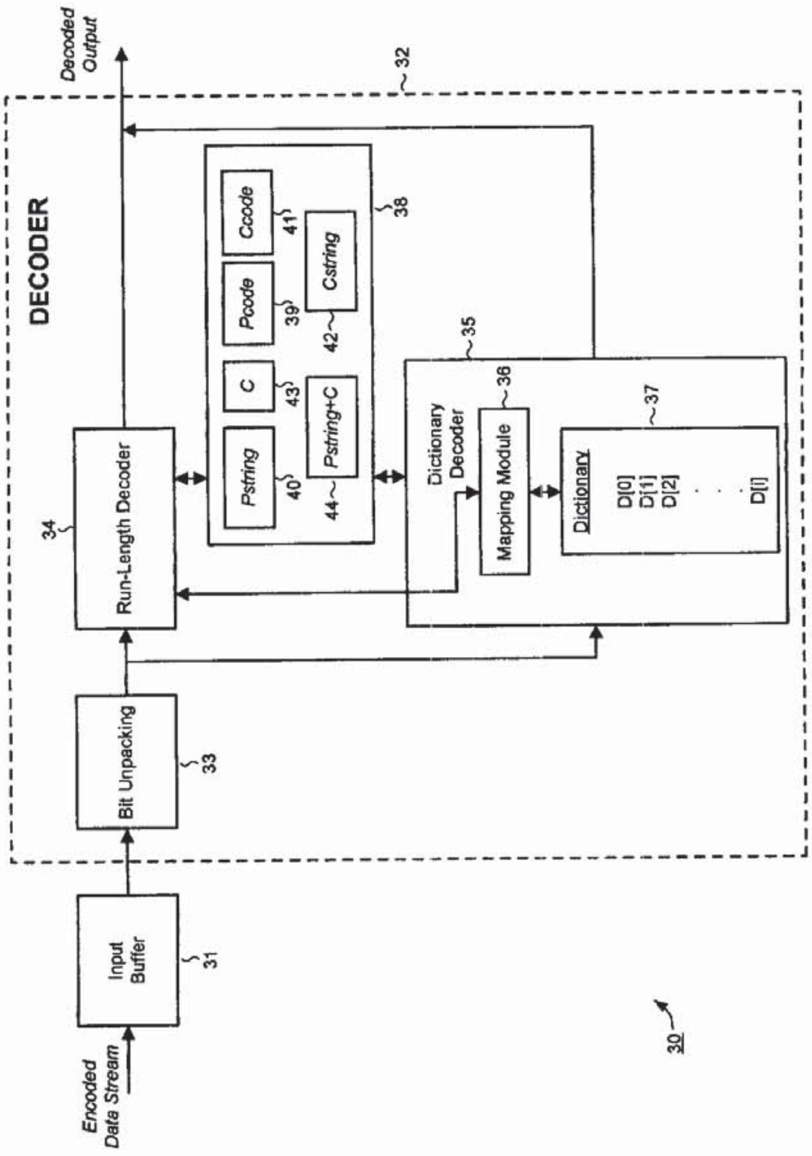

DECODER
Encoded Data Stream
Input Buffer
31
Bit Unpacking
33
Run-Length Decoder
34
Decoded Output
32
Pstring
40
C
43
Pcode
39
Ccode
41
Pstring+C
44
Cstring
42
38
Dictionary Decoder
35
Mapping Module
36
Dictionary
D[0]
D[1]
D[2]
. . .
D[i]
37
30

FIGURE 3

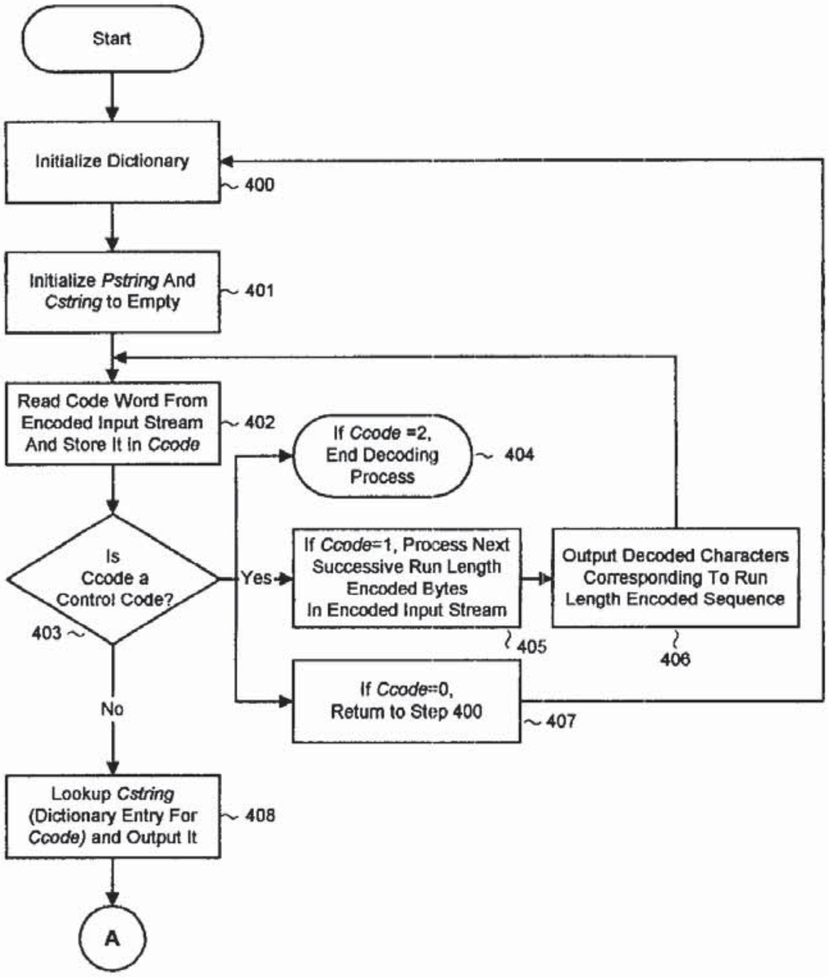

Start
Initialize Dictionary
400
Initialize *Pstring* And *Cstring* to Empty
401
Read Code Word From Encoded Input Stream And Store It in *Ccode*
402
Is Ccode a Control Code?
403
Yes
No
If *Ccode* =2, End Decoding Process
404
If *Ccode*=1, Process Next Successive Run Length Encoded Bytes In Encoded Input Stream
405
Output Decoded Characters Corresponding To Run Length Encoded Sequence
406
If *Ccode*=0, Return to Step 400
407
Lookup *Cstring* (Dictionary Entry For *Ccode)* and Output It
408
A

FIGURE 4A

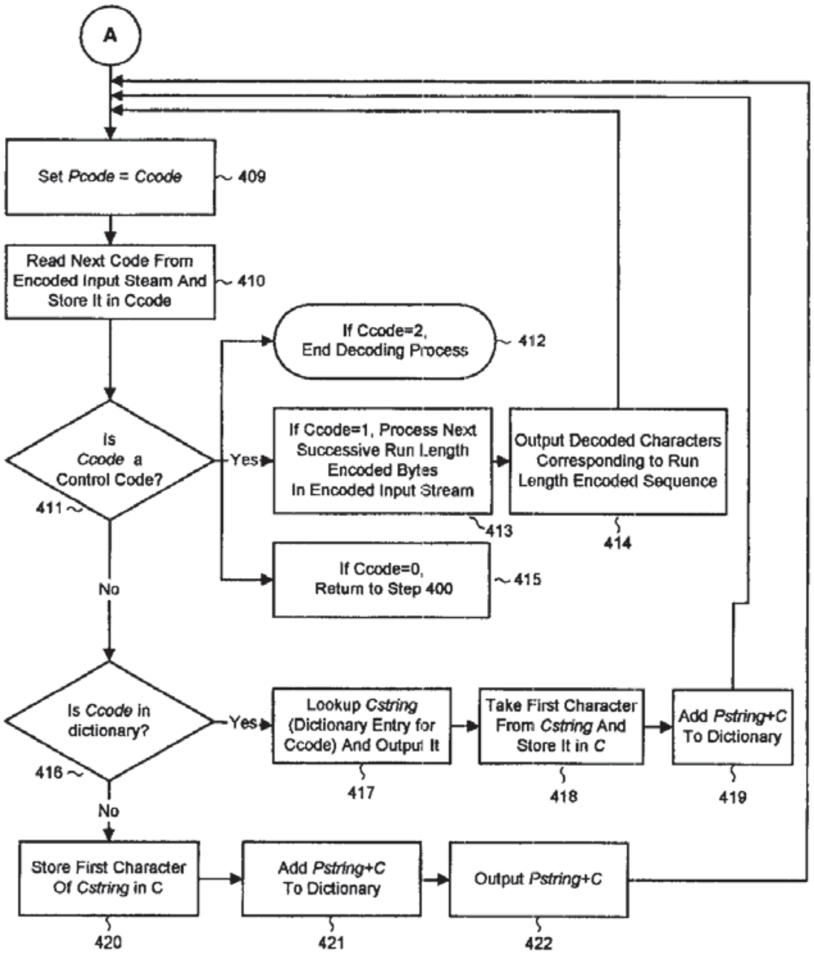

A
Set Pcode = Ccode
409
Read Next Code From Encoded Input Steam And Store It in Ccode
410
If Ccode=2, End Decoding Process
412
Is Ccode a Control Code?
411
Yes
If Ccode=1, Process Next Successive Run Length Encoded Bytes In Encoded Input Stream
413
Output Decoded Characters Corresponding to Run Length Encoded Sequence
414
If Ccode=0, Return to Step 400
415
No
Is Ccode in dictionary?
416
Yes
Lookup Cstring (Dictionary Entry for Ccode) And Output It
417
Take First Character From Cstring And Store It in C
418
Add Pstring+C To Dictionary
419
No
Store First Character Of Cstring in C
420
Add Pstring+C To Dictionary
421
Output Pstring+C
422

FIGURE 4B

SYSTEM AND METHOD FOR LOSSLESS DATA COMPRESSION AND DECOMPRESSION

CROSS-REFERENCE TO RELATED APPLICATION

This application is based on provisional application U.S. Application Ser. No. 60/136,561 filed on May 28, 1999.

BACKGROUND

1. Technical Field

The present invention relates generally to data compression and decompression and, more particularly to systems and methods for providing lossless data compression and decompression using a combination of dictionary and run length encoding.

2. Description of Related Art

Information may be represented in a variety of manners. Discrete information such as text and numbers are easily represented in digital data. This type of data representation is known as symbolic digital data. Symbolic digital data is thus an absolute representation of data such as a letter, figure, character, mark, machine code, or drawing.

Continuous information such as speech, music, audio, images and video frequently exists in the natural world as analog information. As is well-known to those skilled in the art, recent advances in very large scale integration (VLSI) digital computer technology have enabled both discrete and analog information to be represented with digital data. Continuous information represented as digital data is often referred to as diffuse data. Diffuse digital data is thus a representation of data that is of low information density and is typically not easily recognizable to humans in its native form.

There are many advantages associated with digital data representation. For instance, digital data is more readily processed, stored, and transmitted due to its inherently high noise immunity. In addition, the inclusion of redundancy in digital data representation enables error detection and/or correction. Error detection and/or correction capabilities are dependent upon the amount and type of data redundancy, available error detection and correction processing, and extent of data corruption.

One outcome of digital data representation is the continuing need for increased capacity in data processing, storage, retrieval and transmittal. This is especially true for diffuse data where continuing increases in fidelity and resolution create exponentially greater quantities of data. Within the current art, data compression is widely used to reduce the amount of data required to process, transmit, store and/or retrieve a given quantity of information. In general, there are two types of data compression techniques that may be utilized either separately or jointly to encode and decode data: lossy and lossless data compression

Lossy data compression techniques provide for an inexact representation of the original uncompressed data such that the decoded (or reconstructed) data differs from the original unencoded/uncompressed data. Lossy data compression is also known as irreversible or noisy compression. Negentropy is defined as the quantity of information in a given set of data. Thus, one obvious advantage of lossy data compression is that the compression ratios can be larger than that dictated by the negentropy limit, all at the expense of information content. Many lossy data compression techniques seek to exploit various traits within the human senses to eliminate otherwise imperceptible data. For example, lossy data compression of visual imagery might seek to delete information content in excess of the display resolution or contrast ratio of the target display device.

On the other hand, lossless data compression techniques provide an exact representation of the original uncompressed data. Simply stated, the decoded (or reconstructed) data is identical to the original unencoded/uncompressed data. Lossless data compression is also known as reversible or noiseless compression. Thus, lossless data compression has, as its current limit, a minimum representation defined by the negentropy of a given data set.

It is well known within the current art that data compression provides several unique benefits. First, data compression can reduce the time to transmit data by more efficiently utilizing low bandwidth data links. Second, data compression economizes on data storage and allows more information to be stored for a fixed memory size by representing information more efficiently.

A rich and highly diverse set of lossless data compression and decompression algorithms exist within the current art. These range from the simplest "adhoc" approaches to highly sophisticated formalized techniques that span the sciences of information theory, statistics, and artificial intelligence. One fundamental problem with almost all modern approaches is the compression ratio verses the encoding and decoding speed achieved. As previously stated, the current theoretical limit for data compression is the entropy limit of the data set to be encoded. However, in practice, many factors actually limit the compression ratio achieved. Most modern compression algorithms are highly content dependent. Content dependency exceeds the actual statistics of individual elements and often includes a variety of other factors including their spatial location within the data set.

Within the current art there also presently exists a strong inverse relationship between achieving the maximum (current) theoretical compression ratio, referred to as "algorithmic effectiveness", and requisite processing time. For a given single algorithm the "effectiveness" over a broad class of data sets including text, graphics, databases, and executable object code is highly dependent upon the processing effort applied. Given a baseline data set, processor operating speed and target architecture, along with its associated supporting memory and peripheral set, "algorithmic efficiency" is defined herein as the time required to achieve a given compression ratio. Algorithmic efficiency assumes that a given algorithm is implemented in an optimum object code representation executing from the optimum places in memory. This is virtually never achieved in practice due to limitations within modern optimizing software compilers. In addition, an optimum algorithmic implementation for a given input data set may not be optimum for a different data set. Much work remains in developing a comprehensive set of metrics for measuring data compression algorithmic performance, however for present purposes the previously defined terms of algorithmic effectiveness and efficiency should suffice.

Of the most widely utilized compression techniques, arithmetic coding possesses the highest degree of algorithmic effectiveness but, as expected, is the slowest to execute. This is followed in turn by dictionary compression, Huffman coding, and run-length coding techniques with respectively decreasing execution times. What is not apparent from these algorithms, that is also one major deficiency within the current art, is knowledge of their algorithmic efficiency. More specifically, given a compression ratio that is within the effectiveness of multiple algorithms, the question arises as to their corresponding efficiency on various data sets.

SUMMARY OF THE INVENTION

The present invention is directed to systems and methods for providing lossless data compression and decompression. The present invention exploits various characteristics of run-length encoding, parametric dictionary encoding, and bit packing to comprise an encoding/decoding process having an efficiency that is suitable for use in real-time lossless data compression and decompression applications.

In one aspect of the present invention, a method for compressing input data comprising a plurality of data blocks comprises the steps of:

detecting if the input data comprises a run-length sequence of data blocks;

outputting an encoded run-length sequence, if a run-length sequence of data blocks is detected;

maintaining a dictionary comprising a plurality of code words, wherein each code word in the dictionary is associated with a unique data block string;

building a data block string from at least one data block in the input data that is not part of a run-length sequence;

searching for a code word in the dictionary having a unique data block string associated therewith that matches the built data block string; and

outputting the code word representing the built data block string.

In another aspect of the present invention, the dictionary is dynamically maintained and updated during the encoding process by generating a new code word corresponding to a built data block string, if the built data block string does not match a unique data block string in the dictionary, and then adding the new code word in the dictionary.

In yet another aspect of the present invention, the dictionary is initialized during the encoding process if the number of code words (e.g., dictionary indices) in the dictionary exceeds a predetermined threshold. When the dictionary is initialized, a code word is output in the encoded data stream to indicate that the dictionary has been initialized at that point in the encoding process. An initialization process further comprises resetting the dictionary to only include each possible code word corresponding to a unique data block string comprising a single data block. By way of example, if each data block comprises a byte of data, there will be 256 possible code words for a data block string comprising a single byte. In this instance, the dictionary reset to its initial state will comprise 256 entries.

In another aspect of the present invention, the dictionary further comprises a plurality of control code words, wherein a control code word is designated to represent a dictionary initialization, a run-length encoded sequence, and the end of the input data (or completion of the encoding process). These control words are used in the decoding process to re-create the input data.

In yet another aspect of the present invention, a bit-packing process is employed to pack the bits of successive output code words representing encoded run-length sequences and data block strings.

In another aspect of the present invention, a method for decompressing an encoded data stream comprising a plurality of code words, which is generated using the encoding method, comprises the steps of:

maintaining a dictionary comprising a plurality of code words utilized to generate the encoded data stream, wherein the code words in the dictionary comprise control code words and code words that are each associated with a unique data block string;

decoding and outputting a run-length sequence of data blocks associated with an input code word of the encoded data stream, if the input code word is a control code word in the dictionary that indicates an encoded run-length sequence;

outputting a unique data block string in the dictionary that is associated with an input code word of the encoded data stream, if the input code word is found in the dictionary; and

if the input code word is not found in the dictionary, building a new data block string comprising (1) the unique data block string associated with a previous control word found in the dictionary and (2) the first data block of the unique data block string, adding the new string to the dictionary, and outputting the new string.

These and other aspects, features and advantages of the present invention will become apparent from the following detailed description of preferred embodiments, which is to be read in connection with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram of a system for providing lossless data compression according to an embodiment of the present invention;

FIGS. 2*a* and 2*b* comprise a flow diagram of a method for providing lossless data compression according to one aspect of the present invention;

FIG. 3 is a block diagram of a system for providing lossless data decompression according to an embodiment of the present invention; and

FIGS. 4A and 4B comprise a flow diagram of a method for providing lossless data decompression according to one aspect of the present invention.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

The present invention is directed to systems and methods for providing lossless data compression and decompression. It is to be understood that the present invention may be implemented in various forms of hardware, software, firmware, or a combination thereof. In particular, the present invention may be implemented in hardware comprising general purpose microprocessors, digital signal processors, and/or dedicated finite state machines. Preferably, the present invention is implemented as an application program, tangibly embodied on one or more data storage mediums, which is executable on any machine, device or platform comprising suitable architecture. It is to be further understood that, because the present invention is preferably implemented as software, the actual system configurations and process flow illustrated in the accompanying Figures may differ depending upon the manner in which the invention is programmed. Given the teachings herein, one of ordinary skill in the related art will be able to contemplate these and similar implementations or configurations of the present invention.

Data Compression

Referring now to FIG. 1, a block diagram illustrates a system **10** for providing lossless data compression according to an embodiment of the present invention. In general, the data compression system **10** comprises an input buffer **11** for temporarily buffering an input data stream and an encoder 12 for compressing the input data stream. It is to be understood that the compressed data stream output from the encoder may, for example, be stored in a storage medium for subsequent retrieval and decoded using a decompression method described below, or transmitted over a local or global computer network (for purposes of increased bandwidth transmission) and decompressed at a desired location. It is to be further understood that the input buffer 11 is an optional component that may be employed, for example, in real-time compression applications where the rate of compression of the encoder 12 is slower than the bandwidth of the input data stream.

In general, the encoder 12 employs a unique combination of compression techniques preferably including run-length encoding and hash table dictionary encoding to compress an input data stream, as well as bit-packing to increase the final compression ratio. More specifically, the encoder 12 comprises a run-length encoder 13 and dictionary encoder 14, both of which utilize a code word dictionary 15 to output one or more "code words" representing a "character string" identified by the respective encoder 13, 14 in the input data stream. It is to be understood that the term "character" as used herein refers to an input byte of data that can take on any one of 256 values, and the term "string" as used herein refers to a grouping of one or more characters (bytes). Furthermore, as described in further detail below, in a preferred embodiment, a "code word" for a given character string comprises a dictionary index (denoted herein as D[i]) of the character string in the dictionary 15.

During an encoding process in which bytes of data in the input stream are input to the encoder 12, the run-length encoder 13 will identify a run-length sequence in the data stream, i.e., a character string comprising a plurality of consecutively similar characters (bytes), and output one or more code words from the dictionary 15 to represent the run-length sequence (as explained in detail below). Moreover, the dictionary encoder 14 will build a character string comprising two or more characters (which does not comprise a run-length sequence), search the dictionary 15 for a code word that corresponds to the character string, and then output the code word representing the character string. In addition, if the character string that is built by the dictionary encoder 14 does not match a character string in the dictionary 15, the dictionary encoder 14 will cause the character string to be added to the dictionary and a new code word (e.g., dictionary index) will be associated with that string. An encoding process according to one aspect of the present invention will be described in detail below with reference, for example, to the flow diagram of FIGS. 2A and 2B.

The encoder 12 utilizes a plurality of data storage structures 16 for temporarily storing data during an encoding process. For example, in the illustrative embodiment of FIG. 1, a Pstring data structure 17 is employed for temporarily storing a working character string, Pstring. A C data structure 18 is employed for temporarily storing a next character (byte) C in the input stream. In addition, a Pstring+C data structure 19 is used for temporarily storing a character string Pstring+C which is a string comprising all of the characters in Pstring plus the character in C. Moreover, an Mcode data structure 23 is used for temporarily storing a code word (Mcode) (e g., dictionary index) corresponding to a previous successful string match in the dictionary. The use of these data structures will be discussed in further detail below.

The code word dictionary 15 comprises a plurality of dictionary indices D[i], wherein each index in the dictionary 15 is mapped (via a mapping module 20) to either a predefined control code or a different code word corresponding to a character (byte) string. The mapping module 20 preferably employs a hash function to, inter alia, map each character string (e.g., strings of one or more bytes) into a unique index D[i] in the dictionary 15 (although other mapping techniques known to those skilled in the art may be employed). As indicated above, in a preferred embodiment, the dictionary indices D[i] are output as the "code words" (also referred to herein as "Mcodes")by the encoder to create an encoded file. These code words are processed by a decoder to decompress an encoded file (as discussed below with reference to FIGS 3, 4*a* and 4*b*.)

In a preferred embodiment, the first three entries in the dictionary 15, indices D[**0**], D[**1**], and D[**3**], are reserved as control codes. In particular, the entry for the dictionary index D[**0**], or code word "0", is output to indicate (to the decoder) that the dictionary 15 has been reset to its initial state. As explained in detail below, the dictionary 15 is preferably reset at the commencement of an encoding process before a new input stream is processed and, preferably, during an encoding process when the total number of entries D[i] in the dictionary 15 exceeds a predetermined limit. In addition, the dictionary index D[**1**], or code word "1", is utilized for the run-length encoding process. More specifically, the code word "1" is output to indicate that the next two consecutive output numbers (in the encoded sequence) represent a run-length encoding sequence comprising (1) a character code and (2) a number denoting the amount of consecutive characters found in the data stream corresponding to the character code. Furthermore, the dictionary index D[**2**], or code word "2" is output to indicate the end of the data stream and completion of the encoding process.

The next 256 entries in the dictionary 15 (i.e., index numbers 3 through 258) each comprise a single character sting (e.g., one byte) corresponding to one of the 256 possible character codes. Accordingly, in a preferred embodiment, the dictionary indices D[**0**] through D[**258**] are the only entries that exist in the dictionary 15 upon initialization of the dictionary 15. Any additional character strings that are dynamically added to the dictionary 15 by the dictionary encoder 14 during an encoding process will be consecutively added beginning at index D[**260**].

It is to be appreciated that, as indicated above, for a given character string under consideration, the encoder 12 will output (as a code word) the dictionary index number D[i] corresponding to a matching character string. Since the dictionary index number is usually less than two bytes and the input character strings are typically longer than six bytes, the reduction in the number of bits output can be significant.

In one embodiment of the present invention, the dictionary encoder 14 can search the code word dictionary 15 for a matching character string therein by comparing each entry in the dictionary 15 to the input character string under consideration. In certain instances, however, the amount of entries D[i]0 in the dictionary 15 can increase significantly, potentially rendering this search process slow, inefficient and computationally intensive. Accordingly, the data compression system 10 preferably comprises a hash table 21 which is utilized by the dictionary encoder 14 during an encoding process to reduce the search time for finding a matching character string in the dictionary 15.

More specifically, in one embodiment, the hash table 21 comprises a plurality of arrays Array[N], wherein each array comprises every dictionary index number D[i] in the dictionary 15 having an entry (i.e., character strings) that begins with a character code corresponding to the array index. For example, the third hash table array Arrary[**3**] comprises all the dictionary indices D[i] having a dictionary entry in which the first character (byte) of the string has decimal value of "three." In the preferred embodiment where the encoder processes individual bytes of data in the input stream, since there are 256 possible characters, there are 256 arrays, i.e., Array[N], where N=1 . . . 256. Advantageously, the use of the hash table **21** for finding matching strings in the dictionary reduces the number of string comparisons by 256.

In another embodiment, the hash table **21** comprises a plurality of nested hash tables. For example, a first level of hashing can use the first character to subdivide the dictionary **15** into 256 sub-dictionaries and a second level of hashing may use the 2^{nd} character of the input string to further subdivide each of the initial 256 entries. Each additional level of hashing subdivides each dictionary into an additional 256 sub-dictionaries. For example, 2 levels of hashing yields 256^2 sub-dictionaries and n levels yields 256^n sub-dictionaries. The purpose of this hashing function is to reduce the time for searching the dictionary **15**. For example, using an n level hashing scheme reduces the search time by $256^n-(n*256)$.

Furthermore, as explained in detail below with reference to the process depicted in FIGS. **2***a* and **2***b*, the hash table is dynamically modified to incorporate new entries D[i] that are added to the dictionary **15** during the encoding process.

In addition, the data compression system **10** optionally comprises a bit packing module **22** for providing additional compression of the encoded data stream. As explained above, the maximum size (i.e., number of entries D[i]) of the dictionary **15** is predefined and, consequently, the maximum number of bits of information needed to represent any index in the dictionary **15** is known a priori. For example, if the maximum dictionary size is 4000 entries, only 12 bits are needed to represent any index number. Since data is typically transferred in groups of 8 or 16 bits, in the above example where 12 bits maximum are need to represent the index number, 4 bits out of every 16 bits would be wasted.

Accordingly, to provide additional compression, the encoder **12** preferably implements the bit-packing module **22** to pack the bits of successive output code words. It is to be understood that any suitable bit-packing technique known to those skilled in the art may be employed. In a preferred embodiment, the bit-packing module employs a shift register to output at least 16 bits of data when the data is ready for output. By way of example, assume a 12-bit code word is initially input to the shift register. The next 12-bit code word that is output is also placed in the shift register, and the shift register would contain 24 bits of information. Then, 16 bits would be output from the shift register, leaving 8 bits remaining. When the next 12-bit code word is input to the shift register, the shift register will contain 20 bits, and 16 will be output. This bit packing process is repeated for every output code word until the encoding process is complete.

Advantageously, the bit packing process according to the present invention improves the compression by a factor of $^{16}/_{12}$, or 1.33. Moreover, it is to be appreciated that the processing time required for the bit-packing is negligible. Consequently, the bit packing process provides increased compression ("algorithmic effectiveness") without a significant increase in processing overhead ("algorithmic efficiency").

Referring now to FIGS. **2***a* and **2***b*, a flow diagram illustrates a method for compressing data according to one

aspect of the present invention. In particular, the encoding process depicted in FIGS. **2***a* and **2***b* illustrates a mode of operation of the system **10** of FIG. **1**. Initially, the dictionary **15** and hash table **21** are initialized (step **200**). For example, as noted above, the dictionary **15** is initialized to include 259 entries, i.e., the first three entries D[**0**]–D[**2**] comprise the control codes and the next 256 entries D[**3**]–D[**259**] comprise the 256 possible character codes (assuming, of course, that the encoder processes data blocks each comprising a byte). Furthermore, the hash table will be initialized such that each array Arrary[**1**]–[N] comprises one entry—the dictionary index D[i] for the corresponding character code. Next, the Pstring data structure **17** (or "Pstring")is initialized to be empty (i.e., it contains no characters at initialization) (step **201**). It is to be understood that neither the C data structure **18** (or "C") nor the Mcode data structure **23** (or "Mcode") require initialization.

After the initialization process, a determination is made as to whether there are any input characters for processing (step **202**). If there is input data (affirmative result in step **202**), the first (or next) character (e.g., byte) in the input stream will be read and temporarily stored in C (step **203**). Then, the next consecutive characters in the input stream are checked (step **204**) to determine if there is a string of at least s consecutive characters that match the character stored in C to trigger a run-length sequence (step **205**), where s is a predetermined minimum number of consecutive characters that are required to trigger a run-length encoding sequence.

If there are at least s consecutively similar characters in the input stream (affirmative determination in step **205**), then a determination is made as to whether Pstring is empty (step **206**). If Pstring is empty (affirmative determination in step **206**), then code words representing the run-length sequence are output (step **207**). In a preferred embodiment, the encoded run-length sequence comprises the predefined control code "1" (which is first output from the dictionary **15**), followed by the code word for the character stored in C (which is also obtained from the dictionary), which is then followed by the number of consecutive characters that were found in the input stream to match the character in C.

On the other hand, if Pstring is not empty (negative determination in step **206**) upon the triggering of run-length encoding process, before the run-length encoding sequence is generated and output (step **207**), the code word having an entry (character string) that matches the current value of Pstring is output (step **208**), and Pstring is set to empty (step **209**) It is to be understood that the code word for the current value of Pstring in this instance would be the code word that was determined (and temporarily stored in Mcode) from a last successful dictionary search.

If there are not enough consecutively similar characters to trigger an run-length encoding sequence (negative determination in step **205**), referring now to FIG. **2***b*, the character string Pstring+C is generated (step **210**). A dictionary search is then performed to determine if there is an indexed character string that matches Pstring+C (step **211**). This search is performed using, for example, the search techniques described above, e.g., searching each entry in the dictionary starting from index D[**3**] to find an entry that matches Pstring+C, or using the hash table to first determine each dictionary index having a character string entry that begins with the first character in the string Pstring+C. It is to be understood that, during the initial search, there is always a match found in the dictionary for Pstring+C because Pstring is empty and C contains a single character (i.e., in the illustrative embodiment, the dictionary is initialized to include all possible character codes ranging from **0** to **255**).

If a match for Pstring+C is found in the dictionary (affirmative result in step **212**), the dictionary index D[i] (code word) corresponding to the matching entry is stored in Mcode (step **213**). Next, the string Pstring+C is stored in the Pstring data structure (step **214**). Then, assuming there are additional bytes to process (affirmative result in step **202**) and assuming a run-length encoding process is not triggered (step **205**), the process (i.e., steps **210**–**214**) is repeated until the current value of Pstring+C is not found in the dictionary (negative determination in step **212**). It is to be appreciated that for each iteration of this process, as each input character C is added to the current string Pstring, a dictionary search is performed for the most current value Pstring+C and the value of Mcode is updated (but not output) to include the code word (dictionary index) of the current string Pstring+C if it is found in the dictionary.

When there is no match found between an indexed string in the dictionary and the current Pstring+C (negative determination in step **212**), the code word stored in Mcode corresponding to the last successful dictionary search (in which a match for the current Pstring was found) is output (step **215**). As explained above, the output code word may be further-processed using a bit-packing process as described above to provide additional compression.

Next, a dictionary entry is created for the new string Pstring+C (step **216**) in anticipation of the new string being added to the dictionary. A determination is then made as to whether the addition of the new entry would exceed the predefined maximum number of entries for the dictionary (step **217**). If the addition of the new entry would not result in exceeding this threshold (negative determination in step **217**), the new entry will be added to the end of the dictionary (step **218**), i.e., the entry will be indexed with the next available dictionary index. The appropriate hash table will then be updated (step **219**), i.e., the new dictionary index will be added to the appropriate hash table array.

On the other hand, if the addition of the new entry would result in exceeding the maximum number of dictionary entries(affirmative determination in step **217**), the dictionary will be reset to its initial state as described above (step **220**). In addition, the hash table will be reset to reflect the initialization of the dictionary (step **221**). Then, a predefined code word (e.g , code word "0") will be output to indicate that the dictionary has been reset (step **222**). After initialization of the dictionary and hash table, the new entry will be added to the dictionary (step **218**) and the appropriate hash table array will be updated to reflect the new entry (step **219**).

In any event, once the new entry for Pstring+C has been added to the dictionary and the hash table has been updated appropriately, the Pstring data structure is set to include only the character in C (step **223**). The dictionary is then searched for the string Pstring (step **224**) and the index number of the matching string in stored in Mcode (step **225**). It is to be understood that since Pstring contains one character C and since all possible characters are in the dictionary, the search is assured to find a match. Steps **224** and **225** are performed to ensure that if no match is found the during the next dictionary search, the code word (stored in step **225**) corresponding to the match found in step **224** will be output.

Referring back to FIG. ***2a***, if there are more characters in the input stream, the process described above is repeated until it is determined that there are no more characters in the input stream (negative determination in step **202**). Then, the code word (current value of Mcode) corresponding to a match for the current value of Pstring is output (step **226**). Finally, a predefined control code word (e.g., code word "2") will be output to indicate the end of the encoding process (step **227**).

The following example illustrates several iterations of a portion of the encoding process described above in FIGS. **2A** and **2B**. Assume the input stream comprises the following string of characters "a b a b c a . . . ", wherein each character comprises a byte of information. An initialization process is first performed as discussed above. Then, the first character a in the input stream is read and stored in the data structure C (step **203**). The next character in the input stream b is checked to determine if it matches a (step **204**). In this instance, it will be determined that there is no match and, consequently, a run-length encoding process is not triggered.

Accordingly, the string Pstring+C is created (step **210**). Since Pstring is empty (due to initialization), the new string Pstring+C is simply a. The dictionary is searched for the new string. A matching entry for the character string a will be found since all possible one character strings are indexed in the dictionary. The index D[i] of the match is stored in Mcode (step **213**). The string a (i.e., Pstring+C) is stored in Pstring data structure (step **214**).

The next character in the input stream b is read and stored in the C data structure (step **203**). The next character in the input stream a is checked to determine if it matches b(step **204**) In this instance, it will be determined that there is no match and, consequently, a run-length encoding process is not triggered.

Accordingly, the string Pstring+C is created (step **210**). Since Pstring contains the character a and C contains the character b, the new string is ab. The dictionary is searched for the new string (step 211). In this instance, a match will not be found since there is no entry in the dictionary for the string ab.

Since no match was found (negative result in step **212**), the code word corresponding to the last match is output, i.e., the value in Mcode corresponding to the character a is output. Then, the string ab added to the dictionary at index D[**259**] (steps **216**–**218**) (assuming of course that this is the first new entry after initialization of the dictionary and the addition would not exceed the maximum number of allowed entries).

Then, Pstring is set to include only the character in C, which is b (step **223**), and the dictionary is searched for the indexed entry corresponding to a match for Pstring (step **224**). Since, in this instance, Pstring contains only a single character b, a match is guaranteed. The index of the match is stored in Mcode (step **225**).

Then, the next character in the input stream a is read and stored in the C data structure (step **203**). The next character b is checked to determine if it matches a (step **204**). In this instance, it will be determined that there is no match and, consequently, a run-length encoding process is not triggered.

Accordingly, the string ba (i.e., Pstring+C) is created (step **210**). The dictionary is searched for the new string ba. A match will not be found since there is no entry for the string ba.

Since no match was found (negative result in step **212**), the code word corresponding to the last match is output, i.e., the value in Mcode corresponding to the character b.

Then, the string ba added to the dictionary at index D[**260**] (steps **216**–**218**) (assuming of course that this is the second new entry after initialization of the dictionary and the addition would not exceed the maximum number of allowed entries).

Then, Pstring is set to store the character in C, which is a (step **223**) and the dictionary is searched for the indexed entry corresponding to a match for Pstring (step **224**). Since, in this instance, Pstring contains only a single character a, a match is guaranteed. The index of the match is stored in Mcode (step **225**).

Then, the next character in the input stream b is read and stored in the C data structure (step **203**). The next character c is checked to determine if it matches b (step **204**). In this instance, it will be determined that there is no match and, consequently, a run-length encoding process is not triggered.

Accordingly, the string ab (i.e., Pstring+C) is created (step **210**). The dictionary is searched for the new string ab (step **211**). In this instance, a match will be found since there was a previous entry added to the dictionary for the string ab. Accordingly, the code word (dictionary index) of the entry ab (which is this example is D[**259**]) is stored in Mcode (step **213**). The new string ab is stored in Pstring (step **214**).

The next character in the input stream c is read and stored in the C data structure (step **203**). The next character in the input stream a is checked to determine if it matches c(step **204**). In this instance, it will be determined that there is no match and, consequently, a run-length encoding process is not triggered.

Accordingly, the string abc (i.e., Pstring+C) is created (step **210**). The dictionary is searched for the new string abc. A match will not be found since there is no entry for the string abc.

Since no match was found (negative result in step **212**), the code word corresponding to the last match is output, i.e., the previously stored value in Mcode corresponding to the character string ab. Then, the string abc is added to the dictionary at index D[**261**] (steps **216**–**218**) (assuming of course that this is the third new entry after initialization of the dictionary and the addition would not exceed the maximum number of allowed entries).

Then, Pstring is set to store the character in C, which is c (step **223**) and the dictionary is searched for the indexed entry corresponding to a match for Pstring (step **224**). Since Pstring contains only a single character c, a match is guaranteed. The index of the match is stored in Mcode (step **225**). Again, this process is repeated for all characters in the input stream.

Data Decompression

Referring now to FIG. **3**, a block diagram illustrates a system **30** for providing lossless data decompression according to an embodiment of the present invention. In general, the data decompression system **30** comprises an input buffer **31** for temporarily buffering an encoded data stream and a decoder **32** for decompressing the encoded data stream. It is to be understood that the encoded data stream may be, e.g., received from a storage medium for decoding, or received at a desired location over a communication channel and decoded at the location. It is to be further understood that the input buffer **31** is an optional component that may be employed, for example, in real-time decompression applications where the rate of decompression of the decoder **32** is slower than the bandwidth of the transmitted encoded data stream.

In general, the decoder **32** performs, for the most part, the inverse of the encoding process described above. As an encoded data stream is received by the decoder **32**, a bit unpacking module **33** unpacks the bits and restores the original code words generated by the encoder **12** (FIG. **1**). Again, it is to be understood that the bit packing module **22** (FIG. **1**) is an optional component that may be employed to provide additional compression of the code words. Therefore, if bit packing is not implemented for the encoding process, bit unpacking is not employed in the decoding process.

The decoder **32** comprises a run-length decoder **34** for processing encoded run-length sequences in the encoded data stream and outputting the decoded data corresponding to such encoded run-length sequences. As explained below, if the run-length decoder detects a control word "1" in the input data stream, it will read and process the next two successive words in the encoded stream to output the decoded data.

A dictionary decoder **35** is employed to build a dictionary **37** which is identical to the dictionary built by the encoder **12** (as discussed above). Using a mapping module **36** (or any suitable dictionary lookup function), the dictionary decoder will output character strings that are entries in the dictionary **37** to recreate the original file.

It is to be understood that the state of the dictionary of the encoder is always at least one step ahead of the state of the dictionary of the decoder. Therefore, it is possible that the encoder will output a code word for a unique data block string that the decoder has not yet entered in the decoding dictionary. This special case occurs when a character string is encoded using the string immediately preceding it. When this special situation occurs, the first and last characters of the string must be the same. Accordingly, when the decoder receives a code word that is not in the decoding dictionary, the decoder will know that the first character of the string that was encoded is equal to the last character. This a priori knowledge enables the decoder to handle this special case. It is to be appreciated that because there are no lengthy dictionary searches performed during the decoding process, it is much less computationally intensive than the encoding process. A decoding process according to one aspect of the present invention is described below with reference to FIGS. **4**A and **4**B.

The decoder **32** utilizes a plurality of data storage structures **38** for temporarily storing data during a decoding process. For example, in the illustrative embodiment of FIG. **3**, a Pcode data structure **39** (or "Pcode") is used for temporarily storing a previous code word received by the decoder **32**. A Pstring data structure **40** ("Pstring") is employed for temporarily storing a dictionary string corresponding to Pcode. A Ccode data structure **41** ("Ccode") is employed for temporarily storing a code word that is currently being processed. A Cstring data structure **42** ("Cstring") is employed for temporarily storing a dictionary string corresponding to Ccode. A C data structure **43** is employed for temporarily storing a next code word (byte) C in the encoded input stream. Finally, a Pstring+C data structure **44** is used for temporarily storing a character string Pstring+C which is a string comprising all of the characters in Pstring plus the character in C. The use of these data structures will be discussed in further detail below.

Referring now to FIGS. **4***a* and **4***b*, a flow diagram illustrates a method for decompressing data according to one aspect of the present invention. In particular, the decoding process depicted in FIGS. **4**A and **4**B illustrates a mode of operation of the system **30** of FIG. **3**. Initially, the dictionary **37** will be initialized in the same manner as discussed above (step **400**) i.e., the dictionary will comprises an index for each of the three control words and an index for each of the 256 characters). In addition, Pstring and Cstring are initialized to empty (step **401**). It is to be understood that Pcode, Ccode, and C do not require initialization.

After initialization, the first code word in the encoded input stream will be read and stored in Ccode (step **402**). A determination is then made as to whether the current code word (stored in Ccode) is a (predefined) control word (step **403**). If Ccode is a control word (affirmative determination in step **403**), the decoding process will be terminated if the control word is "2" (step **404**). If the control word is "1", then a run-length decoding process is commenced by reading and processing the next two words in the encoded input stream (step **405**). In particular, as explained above, a code word "1" is output during the encoding process to indicate that the next two consecutive output numbers (in the encoded sequence) represent a run-length encoding sequence comprising (1) a character code and (2) a number denoting the amount of consecutive characters found in the data stream corresponding to the character code. Accordingly, assuming "X" represents the character code and "N" represents the number of consecutive "X"s, the decoder will output the character X, N times (step **406**). Finally, if the control word is "0" (step **407**), the decoding process is initialized (return to step **400**).

On the other hand, if the current Ccode does not comprise a control word (negative determination in step **403**), the dictionary will be searched to find the string Cstring corresponding to the current Ccode (step **408**). It is to be understood that the first (non-control) code word in the input stream will always be found in the dictionary, i.e., the first non-control word will correspond to one of the 256 code words that are initialized in the dictionary.

Referring now to FIG. **4B**, Pcode is set to be equal to Ccode (step **409**) (and the string Pstring is set based on the value of Pcode). The next code word will be read from the encoded input stream and stored in Ccode (step **410**).

A determination is then made as to whether the current code word (stored in Ccode) is a (predefined) control word (step **411**). As explained above, if Ccode is a control word (affirmative determination in step **411**), the decoding process will be terminated if the control word is "2" (step **412**). If the control word is "1", then a run-length decoding process is commenced by reading and processing the next two words ("X" and "N", respectively) in the encoded input stream (step **413**) and the decoder will output the character X, N times (step **414**). If the control word is "0" (step **415**), the decoding process is initialized (return to step **400**).

If, on the other hand, the current Ccode is not a control code (negative determination in step **411**), a determination is made as to whether there is an indexed entry (Cstring) in the decoding dictionary corresponding to Ccode (step **416**). If there is an entry (affirmative determination in step **416**) then Cstring corresponding to that Ccode is output (step **417**). Then, the first character of Cstring is stored in the C data structure (step **418**). A new string Pstring+C is then formed and added to the decoding dictionary (step **419**).

If there is no entry in the dictionary for the current Ccode (negative determination in step **416**) this is the special case described above and the decoder performs the following steps. First, the first character from Pstring is stored in the C data structure (step **420**). Then, a new string Pstring+C is formed and added to the decoding dictionary (step **421**). The new string Pstring+C is then output by the decoder (step **422**).

The following example illustrates several iterations of the decoding process using the output from the above encoding example which was based on the input string "a b a b c a . . " The data structure are initialized as described above (steps **400** and **401**). The first code is read and stored in the data structure Ccode. Since the first input code corresponds to character a, the current Ccode is determined not to be a control code (step **403**). Accordingly, the dictionary entry Cstring (i.e., a) corresponding to Ccode is output.

Pcode is then set equal to Ccode (step **409**). The next code word is read and stored in the data structure Ccode. Since the code word corresponds to character b, Ccode is not a control code (step **411**). The decoding dictionary is then searched for a match for Ccode (step **416**). Since a single character string (i.e., b in this instance) is always in the dictionary, a match will be found. Since a match is guaranteed, the dictionary entry Cstring (i.e., b) is output (step **417**). Next, the first character of Cstring (i.e.,b) is stored in C (step **418**). A new string Pstring+C is formed and added to the dictionary (step **419**). In this example, since Pstring is the string corresponding to Pcode , which is the character a, and C contains the character b, the new string Pstring+C is ab, which is added to the dictionary at the next available index, D[**259**]. Again, Pcode is set equal to Ccode.

Then, the next code word (corresponding to character ab) is read and stored in the data structure Ccode. Since this is not a control code, the dictionary is searched for a match for Ccode. Again, in this instance, there will be a match. Accordingly, Cstring, i.e., ab, is output.

Then, the first character of Cstring (which is a) is stored in C (step **418**). A new string Pstring+C is formed comprising ba (i.e., Pstring is the string corresponding to Pcode, b, and C contains a) and then added to the dictionary (step **419**) at, the next available index D[**260**]. Then, Pcode is set equal to Ccode, and the process is repeated.

It is to be appreciated the present invention exploits various traits within run-length encoding, parametric dictionary encoding, and bit packing to provide an encoding/decoding process whose efficiency is suitable for use in real-time lossless data compression and decompression systems such as the systems disclosed in U.S. patent application Ser. No. 09/210,491, filed on Dec. 11, 1998, entitled "Content Independent Data Compression Method and System," which is commonly assigned and fully incorporated herein by reference.

In particular, although dictionary class encoding techniques, in general, are considered superior to run-length encoding techniques, run-length encoding techniques can process and compress contiguous strings of data blocks far more optimally than dictionary encoding techniques. We have analyzed the manner in which certain programs store data By way of example, we have determined that MICROSOFT OFFICE™ applications use large string of repetitive characters in certain portions of programs and data files such as in the headers and footers of the files, although these run-lengths can occur in the middle of files such as .dll files, data base files and those files with embedded data structures.

Using an analysis tool that analyzes the frequency of characters (i.e., a histogram analysis of the frequency (count) of byte values), we have found that .exe files and .doc files comprise an inordinate quantity of bytes that are equal to 00hex (0s) and FFhex (255). These frequently occurring byte values often appear in contiguous strings as header, footer or byte padding values for data structures internal to the Word format As indicated above, a run-length algorithm exploits these occurrences far more optimally than any known dictionary technique.

In addition, a further analysis of these file types on a block basis, e.g., an 8 kilobyte block or 4 kilobyte block, underscores the advantage of using a combination of dictionary and run-length encoding—the contiguous nature of the data strings that we have found in these files amplifies the benefit of the run-length encoding over the dictionary encoding since the dictionary encoding has been determined to typically provide a lower compression ratio when applied to smaller quantities of data. Therefore, while dictionary compression techniques typically yield higher compression ratios than run-length, this may not be true, e.g., for most MICROSOFT WINDOWS™ operating system, program and data files. Accordingly, an encoding process such as described herein using a combination of run-length and dictionary encoding is far superior to compress data files, etc., that characteristically include contiguous strings of similar data blocks.

Moreover, as indicated above, the use of bit-packing in combination with the dictionary and run-length encoding advantageously provides additional compression, with a negligible increase in the overhead or processing time required for the bit-packing.

Further, the parametric nature of the algorithm allows for tailoring to a wide variety of applications and target processing architectures, wherein trades in processor throughput and instruction set mix, memory hierarchy and bandwidth, and requisite input/output bandwidth requirements may be accommodated By way of example, various memory bandwidths and sizes within the processing hierarchy may dictate the size of the dictionary in terms of the number of entries (or "dictionary depth"), and maximum length of each entry (or "dictionary width"). For example, the Texas Instruments Digital Signal Processor TMS320C6x and TMS320C5x employ separate onboard caches for program and data memory in a Harvard Architecture Arrangement. The caching may further have multiple levels of cached commonly known as L1 (lowest level) and L2 (higher level) onboard cache. Typically the lowest levels of cache have highest throughput. Also, caches are typically faster that external memory.

In one aspect of the present invention, by fixing the dictionary depth to place it in the appropriate level of caching, one can obtain a desired balance between the compression ratio and compression throughput. Indeed, although a larger dictionary typically produces a higher compression ratio, the larger dictionary results in slower throughput. With the current technology limit, L1 cache is typically too small to store a full dictionary and the dictionary is maintained at its optimum size in L2 cache. However, this trade is specific to the desired compression ratio and throughput.

In another aspect of the present invention, the throughput of, e.g., the encoding process can be monitored as a function of compression ratio and dictionary size. If the compression throughput is found to fall below a desired level or is otherwise desired to be increased the compression algorithm may dynamically enlarge the dictionary to increase compression ratio or decrease the dictionary to improve throughput. It should be noted that the relationship is dependent upon the entropy content of the input data stream and may be multivalued and/or non-linear. In yet another aspect of the present invention, a learning algorithm may be further applied to learn the optimum ratios using a time weighted average of throughput.

Another approach is to page dictionary entries from memory to L2 cache, L2 cache to L1 cache, or L1 cache to on board registers within the processor. This methodology can be extended to any memory hierarchy within a single or multiprocessor architecture.

In another embodiment, the present invention may adopt the use of a control signal that would affect the compression technique used by the encoder. The control signal could originate from the same source as the data. It would indicate to the encoder whether to place emphasis on the compression speed or the compression ratio during the encoding process. As indicated above, when it comes to compression speed and compression ratio, one can often be sacrificed to benefit the other.

An example of the use of such a control signal is as follows. Assume the encoder resides in a hard disk controller of a computer. The operating system driver that sends the information to be stored on the disk would generate the control signal. The driver may use an algorithm that normally sends a control signal to the encoder indicating that the encoder should use a form of the compression process that yields a very high compression ratio even if the encoding process is not very fast. When the driver has accumulated sufficient amount of data to be written to the disk, then the driver could generate a control signal to the encoder which would cause the controller to use a very fast implementation of its compression algorithm, even if it does not produce the best compression ratio.

In a particular example, the use of a control signal may be employed to set the appropriate parameters within the encoding/decoding algorithms described herein to facilitate data storage and retrieval bandwidth acceleration and provide data compression and decompression at rates faster than the input data stream such as disclosed in U.S. patent Ser. No. 09/266,394, filed on Mar. 11, 1999, entitled "System and Methods For Accelerated Data Storage and Retrieval," which is commonly assigned and fully incorporated herein by reference. For example, if a data stream inputs 30 megabytes per second the losslessly compressed, real-time, output stream is 10 megabytes per second, assuming a 3:1 compression ratio. Conversely, if a compressed input data stream is 10 megabytes per second, the corresponding decompressed, real-time output stream is 30 megabytes per second, again assuming an original 3:1 lossless compression ratio. Again, using the methods described above, the accelerated data storage and retrieval rates may be modified based on the desired compression and throughput.

Although illustrative embodiments of the present invention have been described herein with reference to the accompanying drawings, it is to be understood that the present invention is not limited to those precise embodiments, and that various other changes and modifications may be affected therein by one skilled in the art without departing from the scope or spirit of the invention. All such changes and modifications are intended to be included within the scope of the invention as defined by the appended claims.

What is claimed is:

1. A method for compressing input data comprising a plurality of data blocks, the method comprising the steps of:

detecting if the input data comprises a run-length sequence of data blocks;

outputting an encoded run-length sequence, if a run-length sequence of data blocks is detected;

maintaining a dictionary comprising a plurality of code words, wherein each code word in the dictionary is associated with a unique data block string;

building a data block string from at least one data block in the input data that is not part of a run-length sequence;

searching for a code word in the dictionary having a unique data block string associated therewith that matches the built data block string; and outputting the code word representing the built data block string.

2. The method of claim **1**, wherein the step of detecting a run-length sequence comprises the steps of:

receiving an input data block;

identifying a run-length sequence if at least the next s successive data blocks in the input data are similar to the input data block.

3. The method of claim **2**, wherein the step of outputting an encoded run-length sequence comprises the step of consecutively outputting a first control code word indicating a run-length sequence, a code word in the dictionary having a unique data block string associated therewith that corresponds to the input data block, and a word corresponding to the number of successive data blocks that are similar to the input data block.

4. The method of claim **1**, wherein the step of maintaining a dictionary comprises the steps of:

dynamically generating a new code word corresponding to a built data block string, if the built data block string does not match a unique data block string in the dictionary; and

adding the new code word in the dictionary.

5. The method of claim **4**, wherein the step of maintaining the dictionary further comprises the step of initializing the dictionary if the number of code words exceeds a predetermined threshold.

6. The method of claim **5**, wherein the step of initializing the dictionary comprises the steps of:

resetting the dictionary to include all possible code words corresponding to a unique data block string comprising a single data block; and

outputting a control code word indicating that the dictionary has been initialized.

7. The method of claim **1**, wherein the code words in the dictionary further comprises at least one control code word representing one of dictionary initialization, a run-length encoded sequence, an end of the input data, and a combination thereof.

8. The method of claim **1**, wherein each code word in the dictionary comprises a dictionary index.

9. The method of claim **1**, further comprising the step of bit-packing encoded run-length sequences and code words that are output.

10. The method of claim **1**, wherein the step of building a data block string comprises the steps of·

(a) iteratively storing in a first data structure, a next successive data block in the input data to build a current data block string; and

(b) for each iteration in step (a), updating a previous code word stored in a second data structure to a current code word corresponding to the current data block string in the first data structure, if the code word for the current data block string in the first data structure is found in the dictionary; and

further wherein the step of outputting the code word representing the built data block string comprises the steps of outputting the previous code word stored in the second data structure, if a code word is not found in the dictionary corresponding to the current data block string in the first data structure.

11. The method of claim **10**, further comprising the step of adding the current data block string to the dictionary.

12 The method of claim **11**, further comprising the steps of·

storing, in a third data structure, the last data block input in the first data structure, if the current data block string is not found in the dictionary; and

repeating steps (a) and (b) starting with the data block in the third data structure, if the data block in the third data structure is not part of a run-length sequence.

13. The method of claim **1**, further comprising the step of maintaining a hash table comprising a plurality of arrays, wherein each array comprises all code words in the dictionary that are associated with a unique data block having a first data block whose value corresponds with an index of the array, and wherein the hash table is used for the step of searching for a code word in the dictionary

14. A program storage device readable by a machine, tangibly embodying a program of instructions executable by the machine to perform method steps for compressing input data comprising a plurality of data blocks, the method comprising the steps of:

detecting if the input data comprises a run-length sequence of data blocks;

outputting an encoded run-length sequence, if a run-length sequence of data blocks is detected;

maintaining a dictionary comprising a plurality of code words, wherein each code word in the dictionary is associated with a unique data block string;

building a data block string from at least one data block in the input data that is not part of a run-length sequence;

searching for a code word in the dictionary having a unique data block string associated therewith that matches the built data block string; and

outputting the code word representing the built data block string.

15. The program storage device of claim **14**, wherein the instructions for performing the step of detecting a run-length sequence comprise instructions for performing the steps of:

receiving an input data block;

identifying a run-length sequence if at least the next s successive data blocks in the input data are-similar to the input data block.

16 The program storage device of claim **15**, wherein the instructions for performing the step of outputting an encoded run-length sequence comprise instructions for performing the step of consecutively outputting a first control code word indicating a run-length sequence, a code word in the dictionary having a unique data block string associated therewith that corresponds to the input data block, and a word corresponding to the number of successive data blocks that are similar to the input data block.

17. The program storage device of claim **14**, wherein the instructions for performing the step of maintaining a dictionary comprise instructions for performing the steps of:

dynamically generating a new code word corresponding to a built data block string, if the built data block string does not match a unique data block string in the dictionary; and

adding the new code word in the dictionary.

18. The program storage device of claim **17**, wherein the instructions for performing the step of maintaining the dictionary comprise instructions for performing the step of initializing the dictionary if the number of code words exceeds a predetermined threshold.

19. The program storage device of claim **18**, wherein the instructions for performing the step of initializing the dictionary comprise instructions for performing the steps of:

resetting the dictionary to include all possible code words corresponding to a unique data block string comprising a single data block; and outputting a control code word indicating that the dictionary has been initialized.

20. The program storage device of claim **14**, wherein the code words in the dictionary further comprise at least one control code word representing one of dictionary initialization, a run-length encoded sequence, an end of the input data, and a combination thereof.

21. The program storage device of claim **14**, wherein each code word in the dictionary comprises a dictionary index.

22. The program storage device of claim **14**, further comprising instructions for performing the step of bit-packing encoded run-length sequences and code words that are output.

23. The program storage device of claim **14**, wherein the instructions for performing the step of building a data block string comprise instructions for performing the steps of:

(a) iteratively storing in a first data structure, a next successive data block in the input data to build a current data block string; and

(b) for each iteration in step (a), updating a previous code word stored in a second data structure to a current code word corresponding to the current data block string in the first data structure, if the code word for the current data block string in the first data structure is found in the dictionary; and

further wherein the instructions for performing the step of outputting the code word representing the built data block string comprise instructions for performing the step of outputting the previous code word stored in the second data structure, if a code word is not found in the dictionary corresponding to the current data block string in the first data structure.

24. The program storage device of claim **23**, further comprising instructions for performing the step of adding the current data block string to the dictionary.

25. The program storage device of claim **24**, further comprising instructions for performing the steps of:

storing, in a third data structure, the last data block input in the first data structure, if the current data block string is not found in the dictionary; and

repeating steps (a) and (b) starting with the data block in the third data structure, if the data block in the third data structure is not part of a run-length sequence.

26. The program storage device of claim **14**, further comprising instructions for performing the step of maintaining a hash table comprising a plurality of arrays, wherein each array comprises all code words in the dictionary that are associated with a unique data block having a first data block whose value corresponds with an index of the array, and wherein the hash table is used for the step of searching for a code word in the dictionary.

27. A method for decompressing an encoded data stream comprising a plurality of code words, the method comprising the steps of:

maintaining a dictionary comprising a plurality of code words utilized to generate the encoded data stream, wherein the code words in the dictionary comprise control code words and code words that are each associated with a unique data block string;

decoding and outputting a run-length sequence of data blocks associated with an input code word of the encoded data stream, if the input code word is a control code word in the dictionary that indicates an encoded run-length sequence;

outputting a unique data block string in the dictionary that is associated with an input code word of the encoded data stream, if the input code word is found in the dictionary; and

if the input code word is not found in the dictionary, building a new data block string comprising (1) the unique data block string associated with a previous control word found in the dictionary and (2) the first data block of the unique data block string, adding the new string to the dictionary and outputting the new string.

28. A system for compressing input data comprising a plurality of data blocks, the system comprising:

a dictionary comprising a plurality of code words, wherein the code words comprise control code words and code words that are each mapped to a unique data block string;

a run-length encoder for encoding a sequence of similar data blocks in the input data using at least one code word in the dictionary; and

a dictionary encoder for encoding a data block string comprising at least one data block in the input data using a code word in the dictionary, wherein output of the run-length encoder and dictionary encoder are combined to form an encoded data stream.

29. The system of claim **28**, further comprising a system for decompressing the encoded data stream, wherein the system for decompressing the encoded data stream comprises:

a dictionary comprising a plurality of code words utilized to generate the encoded data stream, wherein the code words in the dictionary comprise control code words and code words that are each associated with a unique data block string;

a run-length decoder for decoding and outputting a run-length sequence of data blocks associated with an input code word of the encoded data stream, if the input code word is a control code word in the dictionary that indicates an encoded run-length sequence;

a dictionary decoder for outputting a unique data block string in the dictionary that is associated with an input code word of the encoded data stream, if the input code word is found in the dictionary; and if the input code word is not found in the dictionary, building a new data block string comprising (1) the unique data block string associated with a previous control word found in the dictionary and (2) the first data block of the unique data block string, adding the new string to the dictionary and outputting the new string.

30. The system of claim **29**, wherein the compression and decompression systems are employed for accelerated data storage and retrieval.

* * * * *

PATENT APPLICATION SERIAL NO. ____________________

U.S. DEPARTMENT OF COMMERCE
PATENT AND TRADEMARK OFFICE
FEE RECORD SHEET

```
06/02/2000 PSTANBAC 00000036 09579221

01 FC:201                      345.00 OP
02 FC:203                       90.00 OP
03 FC:202                       39.00 OP
```

PTO-1556
(5/87)

*U S GPO 1999-459-082/19144

Bib Data Sheet

UNITED STATES DEPARTMENT OF COMMERCE
Patent and Trademark Office
Address COMMISSIONER OF PATENTS AND TRADEMARKS
Washington, D C 20231

SERIAL NUMBER 09/579,221	**FILING DATE** 05/26/2000 **RULE** —	**CLASS** 710	**GROUP ART UNIT** 2782	**ATTORNEY DOCKET NO.** 8011-3

APPLICANTS

James J. Fallon, Armonk, NY ;
Steven L. Bo, Bayside, NY ;

** **CONTINUING DATA** ************************* SW YES

THIS APPLN CLAIMS BENEFIT OF 60/136,561 05/28/1999

** **FOREIGN APPLICATIONS** ********************* SW NONE

IF REQUIRED, FOREIGN FILING LICENSE GRANTED ** SMALL ENTITY **
** 07/21/2000

Foreign Priority claimed 35 USC 119 (a-d) conditions met Verified and Acknowledged	□ yes ☒ no □ yes ☒ no □ Met after Allowance SW Examiner's Signature Initials	**STATE OR COUNTRY** NY	**SHEETS DRAWING** 6	**TOTAL CLAIMS** 30	**INDEPENDENT CLAIMS** 4

ADDRESS

Frank Chau Esq
F Chau & Associates LLP
1900 Hempstead Turnpike
Suite 501
East Meadow ,NY 11554

TITLE

System and method for lossless data compression and decompression

FILING FEE RECEIVED 474	FEES: Authority has been given in Paper No _________ to charge/credit DEPOSIT ACCOUNT No. ________ for following:	□ All Fees □ 1.16 Fees (Filing) □ 1.17 Fees (Processing Ext of time) □ 1.18 Fees (Issue) □ Other _______________ □ Credit

ABSTRACT OF THE DISCLOSURE

Systems and methods for providing lossless data compression and decompression are disclosed which exploit various characteristics of run-length encoding, parametric dictionary encoding, and bit packing to comprise an encoding/decoding process having an efficiency that is suitable for use in real-time lossless data compression and decompression applications. In one aspect, a method for compressing input data comprising a plurality of data blocks comprises the steps of: detecting if the input data comprises a run-length sequence of data blocks; outputting an encoded run-length sequence, if a run-length sequence of data blocks is detected; maintaining a dictionary comprising a plurality of code words, wherein each code word in the dictionary is associated with a unique data block string; building a data block string from at least one data block in the input data that is not part of a run-length sequence; searching for a code word in the dictionary having a unique data block string associated therewith that matches the built data block string; and outputting the code word representing the built data block string.

SYSTEM AND METHOD FOR LOSSLESS DATA COMPRESSION AND DECOMPRESSION

Cross-Reference To Related Application

This application is based on provisional application U.S. Application Serial No. 60/136,561 filed on May 28, 1999.

BACKGROUND

1. Technical Field:

The present invention relates generally to data compression and decompression and, more particularly to systems and methods for providing lossless data compression and decompression using a combination of dictionary and run length encoding.

2. Description of Related Art:

Information may be represented in a variety of manners. Discrete information such as text and numbers are easily represented in digital data. This type of data representation is known as symbolic digital data. Symbolic digital data is thus an absolute representation of data such as a letter, figure, character, mark, machine code, or drawing.

Continuous information such as speech, music, audio, images and video frequently exists in the natural world as analog information. As is well-known to those skilled in the art, recent advances in very large scale integration (VLSI) digital computer technology have enabled both discrete and analog information to be represented with digital data. Continuous information represented as digital data is often referred to as diffuse data. Diffuse digital data is thus a representation of data that is of low information density and is typically not easily recognizable to humans in its native form.

There are many advantages associated with digital data representation. For instance, digital data is more readily processed, stored, and transmitted due to its inherently high noise immunity. In addition, the inclusion of redundancy in digital data representation enables error detection and/or correction. Error detection and/or correction capabilities are dependent upon the amount and type of data redundancy, available error detection and correction processing, and extent of data corruption.

One outcome of digital data representation is the continuing need for increased capacity in data processing, storage, retrieval and transmittal. This is especially true for diffuse data where continuing increases in fidelity and resolution create exponentially greater quantities of data. Within the current art, data compression is widely used to reduce the amount of data required to process, transmit, store and/or retrieve a given quantity of information. In general, there are two types of data compression techniques that may be utilized either separately or jointly to encode and decode data: lossy and lossless data compression.

Lossy data compression techniques provide for an inexact representation of the original uncompressed data such that the decoded (or reconstructed) data differs from the original unencoded/uncompressed data. Lossy data compression is also known as irreversible or noisy compression. Negentropy is defined as the quantity of information in a given set of data. Thus, one obvious advantage of lossy data compression is that the compression ratios can be larger than that dictated by the negentropy limit, all at the expense of information content. Many lossy data compression techniques seek to exploit various traits within the human senses to eliminate otherwise imperceptible data. For example, lossy data compression of visual imagery might seek to delete information content in excess of the display resolution or contrast ratio of the target display device.

On the other hand, lossless data compression techniques provide an exact representation of the original uncompressed data. Simply stated, the decoded (or reconstructed) data is identical to the original unencoded/uncompressed data. Lossless data compression is also known as reversible or noiseless compression. Thus, lossless data compression has, as its current limit, a minimum representation defined by the negentropy of a given data set.

It is well known within the current art that data compression provides several unique benefits. First, data compression can reduce the time to transmit data by more efficiently utilizing low bandwidth data links. Second, data compression economizes on data storage and allows more information to be stored for a fixed memory size by representing information more efficiently.

A rich and highly diverse set of lossless data compression and decompression algorithms exist within the current art. These range from the simplest "adhoc" approaches to highly sophisticated formalized techniques that span the sciences of information theory, statistics, and artificial intelligence. One fundamental problem with almost all modern approaches is the compression ratio verses the encoding and decoding speed achieved. As previously stated, the current theoretical limit for data compression is the entropy limit of the data set to be encoded. However, in practice, many factors actually limit the compression ratio achieved. Most modern compression algorithms are highly content dependent. Content dependency exceeds the actual statistics of individual elements and often includes a variety of other factors including their spatial location within the data set.

Within the current art there also presently exists a strong inverse relationship between achieving the maximum (current) theoretical compression ratio, referred to as "algorithmic effectiveness", and requisite processing time. For a given single algorithm the "effectiveness" over a broad class of data sets including text, graphics, databases, and executable object code is highly dependent upon the processing effort applied. Given a baseline data set, processor operating speed and target architecture, along with its associated supporting memory and peripheral set, "algorithmic efficiency" is defined herein as the time required to achieve a given compression ratio. Algorithmic efficiency assumes that a given algorithm is implemented in an optimum object code representation executing from the optimum places in memory. This is virtually never achieved in practice due to limitations within modern optimizing software compilers. In addition, an optimum algorithmic implementation for a given input data set may not be optimum for a different data set. Much work remains in developing a comprehensive set of metrics for measuring data compression algorithmic performance, however for present purposes the previously defined terms of algorithmic effectiveness and efficiency should suffice.

Of the most widely utilized compression techniques, arithmetic coding possesses the highest degree of algorithmic effectiveness but, as expected, is the slowest to execute. This is followed in turn by dictionary compression, Huffman coding, and run-length coding techniques with respectively decreasing execution times. What is not apparent from these algorithms, that is also one major deficiency within the current art, is knowledge of their algorithmic efficiency. More specifically, given a compression ratio that is within the effectiveness of multiple algorithms, the question arises as to their corresponding efficiency on various data sets.

SUMMARY OF THE INVENTION

The present invention is directed to systems and methods for providing lossless data compression and decompression. The present invention exploits various characteristics of run-length encoding, parametric dictionary encoding, and bit packing to comprise an encoding/decoding process having an efficiency that is suitable for use in real-time lossless data compression and decompression applications.

In one aspect of the present invention, a method for compressing input data comprising a plurality of data blocks comprises the steps of:

detecting if the input data comprises a run-length sequence of data blocks;

outputting an encoded run-length sequence, if a run-length sequence of data blocks is detected;

maintaining a dictionary comprising a plurality of code words, wherein each code word in the dictionary is associated with a unique data block string;

building a data block string from at least one data block in the input data that is not part of a run-length sequence;

searching for a code word in the dictionary having a unique data block string associated therewith that matches the built data block string; and

outputting the code word representing the built data block string.

In another aspect of the present invention, the dictionary is dynamically maintained and updated during the encoding process by generating a new code word corresponding to a built data block string, if the built data block string does not match a unique data block string in the dictionary, and then adding the new coce word in the dictionary.

In yet another aspect of the present invention, the dictionary is initialized during the encoding process if the number of code words (e.g., dictionary indices) in the dictionary exceeds a predetermined threshold. When the dictionary is initialized, a code word is output in the encoded data stream to indicate that the dictionary has been initialized at that point in the encoding process. An initialization process further comprises resetting the dictionary to only include each possible code word corresponding to a unique data block string comprising a single data block. By way of example, if each data block comprises a byte of data, there will be 256 possible code words for a data block string comprising a single byte. In this instance, the dictionary reset to its initial state will comprise 256 entries.

In another aspect of the present invention, the dictionary further comprises a plurality of control code words, wherein a control code word is designated to represent a dictionary initialization, a run-length encoded sequence, and the end of the input data (or completion of the encoding process). These control words are used in the decoding process to re-create the input data.

In yet another aspect of the present invention, a bit-packing process is employec to pack the bits of successive output code words representing encoded run-length sequences and data block strings.

In another aspect of the present invention, a method for decompressing an encoded data stream comprising a plurality of code words, wrich is generated using the encoding method, comprises the steps of:

maintaining a dictionary comprising a plurality of code words utilized to generate the encoded data stream, wherein the code words in the dictionary comprise control code words and code words that are each associated with a unique data block string;

decoding and outputtirg a run-length sequence of data blocks associated with an input code word of the encoded data stream, if the input code word is a control code word in the dictionary that indicates an encoded run-length sequence;

outputting a unique data block string in the dictionary that is associated with an input code word of the encoded data stream, if the input code word is found in the dictionary; and

if the input code word is not found in the dictionary, building a new data block string comprising (1) the unique data block string associated with a previous control word found in the dictionary anc (2) the first data block of the unique data block string, adding the new string to the dictionary, and outputting the new string.

These and other aspects, features and advantages of the present invention will beccme apparent from the following detailed description of preferred embodiments, which is to be read in connection with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a block diagram of a system for providing lossless data compression according to an embodiment of the present invention;

Figs. 2a and 2b comprise a flow diagram of a method for providing lossless data compression according to one aspect of the present invention;

Fig. 3 is a block diagram of a system for providing lossless data decompression according to an embodiment of the present invention; and

Figs. 4A and 4B comprise a flow diagram of a method for providing lossless data decompression according to one aspect of the present invention.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENTS

The present invention is directed to systems and methods for providing lossless data compression and decompression. It is to be understood that the present invention may be implemented in various forms of hardware, software, firmware, or a ccmbination thereof. In particular, the present invention may be implemented in hardware comprising general purpose microprocessors, digital signal processors, and/or cedicated finite state machines. Preferably, the present invention is implemented as an application program, tangibly embodied on one or more data storage mediums, which is executable on any machine, device or platform comprising suitable architecture. It is to be further understood that, because the present invention is preferably implemented as software, the actual system configurations and process flow illustrated in the accompanying Figures may differ depending upon the manner in which the invention is programmed. Given the teachings herein, one of ordinary skill in the related art will be able to contemplate these and similar implementations or configurations of the present invention.

Data Compression

Referring now to Fig. 1, a block diagram illustrates a system 10 for providing lossless data compression according to an embodiment of the present invention. In general, the data compression system 10 comprises an input buffer 11 for temporarily buffering an input data stream and an encoder 12 for compressing the input cata stream. It is to be understood that the compressed data stream output from the encoder may, for example, be stored in a storage medium for subsequent retrieval and decoded using a decompression method described below, or transmitted over a local or global computer network (for purposes of increased bandwidth transmission) and decompressed at a desired location. It is to be further understood trat the input buffer 11 is an optional component that may be employed, for example, in real-time compression applications where the rate of compression of the encoder 12 is slower than the bandwidth of the input data stream.

In general, the encoder 12 employs a unique combination of compression techniques preferably including run-length encoding and hash table dictionary encoding to compress an input data stream, as well as bit-packing to increase the final compression ratio. More specifically, the encoder 12 comprises a run-length enccder 13 and dictionary encoder 14, both of which utilize a code word dictionary 15 to output one or more "code words" representing a "character string" identified by the respective encoder 13, 14 in the input data stream. It is to be understood that the term "character" as used herein refers to an input byte of data that can take on any one of 256 values, and the term "string" as used herein refers to a grouping of one or more characters (bytes). Furthermore, as described in further detail below, in a preferred embodiment, a "code word" for a given character string comprises a dictionary index (denoted herein as **D[i]**) of the character string in the dictionary 15.

During an encoding process in which bytes of data in the input stream are input to the encoder 12, the run-length encoder 13 will identify a run-length sequence in the data stream, i.e., a character strıng comprising a plurality of consecutively similar characters (bytes), and output one or more code words from the dictionary 15 to represent the run-length sequence (as explained in detail below). Moreover, the dictionary encoder 14 will build a character string comprising two or more characters (which does not comprise a run-length sequence), search the dictionary 15 for a code word that corresponds to the character string, and then output the code word representing the character string. In addition, if the character string that is built by the dictionary encoder 14 does not match a character string in the dictionary 15, the dictionary encoder 14 will cause the character string to be added to the dictionary and a new code word (e.g., dictionary index) will be associated with that string. An encoding process according to one aspect of the present invention will be described in detail below with reference, for example, to the flow diagram of Figs. 2A and 2B.

The encoder 12 utilizes a plurality of data storage structures 16 for temporarily storing data during an encoding process. For example, in the illustrative embodiment of Fig. 1, a Pstring data structure 17 is employed for temporarily storing a working character string, ***Pstring***. A C data structure 18 is employed for temporarily storing a next character (byte) ***C*** in the input stream. In addition, a Pstring+C data structure 19 is used for temporarily storing a character string ***Pstring+C*** ,which is a string comprising all of the characters in ***Pstring*** plus the character in ***C***. Moreover, an Mcode data structure 23 is used for temporarily storing a code word (***Mcode***) (e.g., dictionary index) corresponding to a previous successful string match in the dictionary. The use of these data structures will be discussed in further detail below.

The code word dictionary 15 comprises a plurality of dictionary indices **D[i]**, wherein each index in the dictionary 15 is mapped (via a mapping module 20) to either a predefined control code or a different code word corresponding to a character (byte) string. The mapping module 20 preferably employs a hash function to, *inter alia*, map each character string (e.g., strings of one or more bytes) into a unique index **D[i]** in the dictionary 15 (although other mapping techniques known to those skilled in the art may be employed). As indicated above, in a preferred embodiment, the dictionary indices **D[i]** are output as the "code words" (also referred to herein as "***Mcodes***") by the encoder to create an encoded file. These code words are processed by a decoder to decompress an encoded file (as discussed below with reference to Figs. 3, 4a and 4b.)

In a preferred embodiment, the first three entries in the dictionary 15, indices **D[0]**, **D[1]**, and **D[3]**, are reserved as control codes. In particular, the entry for the dictionary index **D[0]**, or code word "0", is output to indicate (to the decoder) that the dictionary 15 has been reset to its initial state. As explained in detail below, the dictionary 15 is preferably reset at the commencement of an encoding process before a new input stream is processed and, preferably, during an encoding process when the total number of entries **D[i]** in the dictionary 15 exceeds a predetermined limit. In addition, the dictionary index **D[1]**, or code word "1", is utilized for the run-length encoding process. More specifically, the code word "1" is output to indicate that the next two consecutive output numbers (in the encoded sequence) represent a run-length encoding sequence comprising (1) a character code and (2) a number denoting the amount of consecutive characters found in the data stream corresponding to the character code. Furthermore, the dictionary index **D[2]**, or code word "2" is output to indicate the end of the data stream and completion of the encoding process.

The next 256 entries in the dictionary 15 (i.e., index numbers 3 through 258) each comprise a single character sting (e.g., one byte) corresponding to one of the 256 possible character codes. Accordingly, in a preferred embodiment, the dictionary indices **D[0]** through **D[258]** are the only entries that exist in the dictionary 15 upon initialization of the dictionary 15. Any additional character strings that are dynamically added to the dictionary 15 by the dictionary encoder 14 during an encoding process will be consecutively added beginning at index **D[260]**.

It is to be appreciated that, as indicated above, for a given character string under consideration, the encoder 12 will output (as a code word) the dictionary index number **D[i]** corresponding to a matching character string. Since the dictionary index number is usually less than two bytes and the input character strings are typically longer than six bytes, the reduction in the number of bits output can be significant.

In one embodiment of the present invention, the dictionary encoder 14 can search the code word dictionary 15 for a matching character string therein by comparing each entry in the dictionary 15 to the input character string under consideration. In certain instances, however, the amount of entries **D[i]** in the dictionary 15 can increase significantly, potentially rendering this search process slow, inefficient and computationally intensive. Accordingly, the data compression system 10 preferably comprises a hash table 21 which is utilized by the dictionary encoder 14 during an encoding process to reduce the search time for finding a matching character string in the dictionary 15.

More specifically, in one embodiment, the hash table 21 comprises a plurality of arrays **Array[N],** wherein each array comprises every dictionary index number **D[i]** in the dictionary 15 having an entry (i.e., character strings) that begins with a character code corresponding to the array index. For example, the third hash table array **Arrary[3]** comprises all the dictionary indices **D[i]** having a dictionary entry in which the first character (byte) of the string has decimal value of "three." In the preferred embodiment where the encoder processes individual bytes of data in the input stream, since there are 256 possible characters, there are 256 arrays, i.e., **Array[N]**, where $\mathbf{N} = 1 \ldots 256$. Advantageously, the use of the hash table 21 for finding matching strings in the dictionary reduces the number of string comparisons by 256.

In another embodiment, the hash table 21 comprises a plurality of nested hash tables. For example, a first level of hashing can use the first character to subdivide the dictionary 15 int0 256 sub-dictionaries and a second level of hashing may use the 2nd character of the input string to further subdivide each of he initial 256 entries. Each additional level of hashing subdivides each dictionary into an additional 256 sub-dictionaries. For example, 2 levels of hashing yields 256^2 sub-dictionaries and **n** levels yields $256^{\mathbf{n}}$ sub-dictionaries. The purpose of this hashing function is to reduce the time for searching the dictionary 15. For example, using an **n** level hashing scheme reduces the search time by $256^{\mathbf{n}}-(\mathbf{n}*256)$.

Furthermore, as explained in detail below with reference to the process depicted in Figs. 2a and 2b, the hash table is dynamically modified to incorporate new entries **D[i|** that are added to the dictionary 15 during the encoding process.

In addition, the data compression system 10 optionally comprises a bit packing module 22 for providing additional compression of the encoded data stream. As explained above, the maximum size (i.e., number of entries **D[i]**) of the dictionary 15 is predefined and, consequently, the maximum number of bits of information needed to represent any index in the dictionary 15 is known *a priori*. For example, if the maximum dictionary size is 4000 entries, only 12 bits are needed to represent any index number. Since data is typically transferred in groups of 8 or 16 bits, in the above example where 12 bits maximum are reed to represent the index number, 4 bits out of every 16 bits would be wasted.

Accordingly, to provide additional compression, the encoder 12 preferably implements the bit-packing module 22 to pack the bits of successive output code words. It is to be understood that any suitable bit-packing technique known to those skilled in the art may be employed. In a preferred embodiment, the bit-packing module employs a shift register to output at least 16 bits of data when the data is ready for output. By way of example, assume a 12-bit code word is initially input to the shift register. The next 12-bit code word that is output is also placec in the shift register, and the shift register would contain 24 bits of information. Then, 16 bits would be output from the shift register, leaving 8 bits remaining. When the next 12-bit code word is input to the shift register, the shift register will contain 20 bits, and 16 will be output. This bit packing process is repeated for every output code word until the encoding process is complete.

Advantageously, the bit packing process according to the present invention improves the compression by a factor of 16/12, or 1.33. Moreover, it is to be appreciated that the processing time required for the bit-packing is negligible. Consequently, the bit packing process provides increased compression ("algorithmic effectiveness") without a significant increase in processing overhead ("algorithmic efficiency").

Referring now to Figs. 2a and 2b, a flow diagram illustrates a method for compressing data according to one aspect of the present invention. In particular, the encoding process depicted in Figs. 2a anc 2b illustrates a mode of operation of the system 10 of Fig. 1. Initially, the dictionary 15 and hash table 21 are initialized (step 200). For example, as noted above, the dictionary 15 is initialized to include 259 entries, i.e., the first three entries **D[0] - D[2]** comprise the control codes and the next 256 entries **D[3] - D[259]** comprise the 256 possible character codes (assuming, of course, that the encoder processes data blocks each comprising a byte). Furthermore, the hash table will be initialized such that each array **Arrary[1]-[N]** comprises one entry - the dictionary index **D[i]** for the corresponding character code. Next, the Pstring data structure 17 (or "***Pstring***")is initialized to be empty (i.e., it contains no characters at initialization) (step 201). It is to be understood that neither the C data structure 18 (or "***C***") nor the Mcode data structure 23 (or "***Mcode***") require initialization.

After the initialization process, a determination is made as to whether there are any input characters for processing (step 202). If there is input data (affirmative result in step 202), the first (or next) character (e.g., byte) in the input stream will be read and temporarily stored in ***C*** (step 203). Then, the next consecutive characters in the input stream are checked (step 204) to determine if there is a string of at least ***s*** consecutive characters that match the character stored in ***C*** to trigger a run-length sequence (step 205), where ***s*** is a predetermined minimum number of consecutive characters that are required to trigger a run-length encoding sequence.

If there are at least ***s*** consecutively similar characters in the input stream (affirmative determination in step 205), then a determination is made as to whether ***Pstring*** is empty (step 206). If ***Pstring*** is empty (affirmative determination in step 206), then code words representing the run-length sequence are output (step 207). In a preferred embodiment, the encoded run-length sequence comprises the predefined control code "1" (which is first output from the dictionary 15), followed by the code word for the character stored in ***C*** (which is also obtained from the dictionary), which is then followed by the number of consecutive characters that were found in the input stream to match the character in ***C***.

On the other hand, if ***Pstring*** is not empty (negative determination in step 206) upon the triggering of run-length encoding process, before the run-length encoding sequence is generated and output (step 207), the code word having an entry (character string) that matches the current value of ***Pstring*** is output (step 208), and ***Pstring*** is set to empty (step 209). It is to be understood that the code word for the current value of ***Pstring*** in this instance would be the code word that was determined (and temporarily stored in ***Mcode***) from a last successful dictionary search.

If there are not enough consecutively similar characters to trigger an run-length encoding sequence (negative determination in step 205), referring now to Fig. 2b, the character string ***Pstring+C*** is generated (step 210). A dictionary search is then performed to determine if there is an indexed character string that matches ***Pstring+C*** (step 211). This search is performed using, for example, the search techniques described above, e.g., searching each entry in the dictionary starting from index **D[3]** to find an entry that matches ***Pstring+C*** , or using the hash table to first determine each dictionary index having a character string entry that begins with the first character in the string ***Pstring+C.*** It is to be understood that, during the initial search, there is always a match found in the dictionary for ***Pstring+C*** because ***Pstring*** is empty and ***C*** contains a single character (i.e., in the illustrative embodiment, the dictionary is initialized to include all possible character codes ranging from 0 to 255).

If a match for ***Pstring+C*** is found in the dictionary (affirmative result in step 212), the dictionary index **D[i]** (code word) corresponding to the matching entry is stored in ***Mcode*** (step 213). Next, the string ***Pstring+C*** is stored in the ***Pstring*** data structure (step 214). Then, assuming there are additional bytes to process (affirmative result in step 202) and assuming a run-length encoding process is not triggered (step 205), the process (i.e., steps 210-214) is repeated until the current value of ***Pstring+C*** is not found in the dictionary (negative determination in step 212). It is to be appreciated that for each iteration of this process, as each input character ***C*** is added to the current string ***Pstring,*** a dictionary search is performed for the most current value ***Pstring+C*** and the value of ***Mcode*** is updated (but not output) to include the code word (dictionary index) of the current string ***Pstring+C*** if it is found in the dictionary.

When there is no match found between an indexed string in the dictionary and the current ***Pstring+C*** (negative determination in step 212), the code word stored in ***Mcode*** corresponding to the last successful dictionary search (in which a match for the current ***Pstring*** was found) is output (step 215). As explained above, the output code word may be further processed using a bit-packing process as described above to provide additional compression.

Next, a dictionary entry is created for the new string ***Pstring+C*** (step 216) in anticipation of the new string being added to the dictionary. A determination is then made as to whether the addition of the new entry would exceed the predefined maximum number of entries for the dictionary (step 217). If the addition of the new entry would not result in exceeding this threshold (negative determination in step 217), the new entry will be added to the end of the dictionary (step 218), i.e., the entry will be indexed with the next available dictionary index. The appropriate hash table will then be updated (step 219), i.e., the new dictionary index will be added to the appropriate hash table array.

On the other hand, if the addition of the new entry would result in exceeding the maximum number of dictionary entries(affirmative determination in step 217), the dictionary will be reset tc its initial state as described above (step 220). In addition, the hash table will be reset to reflect the initialization of tne dictionary (step 221). Then, a predefined code word (e.g., code word "0") will be output to indicate that the dictionary has been reset (step 222). After initialization of the dictionary and hash table, the new entry will be added to the dictionary (step 218) and the appropriate hash table array will be updated to reflect the new entry (step 219).

In any event, once the new entry for ***Pstring+C*** has been added to the dictionary and the hash table has been updated appropriately, the ***Pstring*** data structure is set to include only the character in ***C*** (step 223). The dictionary is then searched for the string ***Pstring*** (step 224) and the index number of the matching string in stored in ***Mcode*** (step 225). It is to be understood that since ***Pstring*** contains one character ***C*** and since all possible characters are in the dictionary, the search is assured to find a match. Steps 224 and 225 are performed to ensure that if no match is found the during the next dictionary search, the code word (stored in step 225) corresponding to the match found in step 224 will be output.

Referring back to Fig. 2a, if there are more characters in the input stream, the process described above is repeated until it is determined that there are no more characters in the input stream (negative determination in step 202). Then, the code word (current value of ***Mcode***) corresponding to a match for the current value of ***Pstring*** is output (step 226). Finally, a predefined control code word (e.g., code word "2") will be output to indicate the end of the encoding process (step 227).

The following example illustrates several iterations of a portion of the encoding process described above in Figs. 2A and 2B. Assume the input stream comprises the following string of characters "***a b a b c a …***", wherein each character comprises a byte of information. An initialization process is first performed as discussed above. Then, the first character ***a*** in the input stream is read and stored in the data structure ***C*** (step 203). The next character in the input stream ***b*** is checked to determine if it matches ***a*** (step 204). In this instance, it will be determined that there is no match and, consequently, a run-length encoding process is not triggered.

Accordingly, the string ***Pstring+C*** is created (step 210). Since ***Pstring*** is empty (due to initialization), the new string ***Pstring+C*** is simply ***a***. The dictionary is searched for the new string. A matching entry for the character string ***a*** will be found since all possible one character strings are indexed in the dictionary. The index **D[i]** of the match is stored in ***Mcode*** (step 213). The string ***a*** (i.e., ***Pstring+C***) is stored in ***Pstring*** data structure (step 214).

The next character in the input stream ***b*** is read and stored in the ***C*** data structure (step 203). The next character in the input stream ***a*** is checked to determine if it matches ***b*** (step 204). In this instance, it will be determined that there is no match and, consequently, a run-length encoding process is not triggered.

Accordingly, the string ***Pstring+C*** is created (step 210). Since ***Pstring*** contains the character ***a*** and ***C*** contains the character ***b***, the new string is ***ab***. The dictionary is searched for the new string (step 211). In this instance, a match will *not* be found since there is no entry in the dictionary for the string ***ab***.

Since no match was found (negative result in step 212), the code word corresponding to the last match is output, i.e., the value in ***Mcode*** corresponding to the character ***a*** is output. Then, the string ***ab*** added to the dictionary at index **D[259]** (steps 216-218) (assuming of course that this is the first new entry after initialization of the dictionary and the addition would not exceed the maximum number of allowed entries).

Then, ***Pstring*** is set to include only the character in ***C***, which is ***b*** (step 223), and the dictionary is searched for the indexed entry corresponding to a match for ***Pstring*** (step 224). Since, in this instance, ***Pstring*** contains only a single character ***b***, a match is guaranteed. The index of the match is stored in ***Mcode*** (step 225).

Then, the next character in the input stream ***a*** is read and stored in the ***C*** data structure (step 203). The next character ***b*** is checked to determine if it matches ***a*** (step 204). In this instance, it will be determined that there is no match and, consequently, a run-length encoding process is not triggered.

Accordingly, the string ***ba*** (i.e., ***Pstring+C***) is created (step 210). The dictionary is searched for the new string ***ba***. A match will not be found since there is no entry for the string ***ba***.

Since no match was found (negative result in step 212), the code word corresponding to the last match is output, i.e., the value in ***Mcode*** corresponding to the character ***b***.

Then, the string ***ba*** added to the dictionary at index **D[260]** (steps 216-218) (assuming of course that this is the second new entry after initialization of the dictionary and the addition would not exceed the maximum number of allowed entries).

Then, ***Pstring*** is set to store the character in ***C***, which is ***a*** (step 223) and the dictionary is searched for the indexed entry corresponding to a match for ***Pstring*** (step 224). Since, in this instance, ***Pstring*** contains only a single character ***a***, a match is guaranteed. The index of the match is stored in ***Mcode*** (step 225).

Then, the next character in the input stream ***b*** is read and stored in the ***C*** data structure (step 203). The next character ***c*** is checked to determine if it matches ***b*** (step 204). In this instance, it will be determined that there is no match and, consequently, a run-length encoding process is not triggered.

Accordingly, the string ***ab*** (i.e., ***Pstring+C***) is created (step 210). The dictionary is searched for the new string ***ab*** (step 211). In this instance, a match *will* be found since there was a previous entry added to the dictionary for the string ***ab***. Accordingly, the code word (dictionary index) of the entry ***ab*** (which is this example is **D[259]**) is stored in ***Mcode*** (step 213). The new string ***ab*** is stored in ***Pstring*** (step 214).

The next character in the input stream ***c*** is read and stored in the ***C*** data structure (step 203). The next character in the input stream ***a*** is checked to determine if it matches ***c*** (step 204). In this instance, it will be determined that there is no match and, consequently, a run-length encoding process is not triggered.

Accordingly, the string ***abc*** (i.e., ***Pstring+C***) is created (step 210). The dictionary is searched for the new string

abc. A match will *not* be found since there is no entry for the string ***abc***.

Since no match was found (negative result in step 212), the code word corresponding to the last match is output, i.e., the previously stored value in ***Mcode*** corresponding to the character string ***ab***. Then, the string ***abc*** is added to the dictionary at index **D[261]** (steps 216-218) (assuming of course that this is the third new entry after initialization of the dictionary and the addition would not exceed the maximum number of allowed entries).

Then, ***Pstring*** is set to store the character in ***C***, which is ***c*** (step 223) and the dictionary is searched for the indexed entry corresponding to a match for ***Pstring*** (step 224). Since ***Pstring*** contains only a single character ***c***, a match is guaranteed. The index of the match is stored in ***Mcode*** (step 225). Again, this process is repeated for all characters in the input stream.

Data Decompression

Referring now to Fig. 3, a block diagram illustrates a system 30 for providing lossless data decompression according to an embodiment of the present invention. In general, the data decompression system 30 comprises an input buffer 31 for temporarily buffering an encoded data stream and a decoder 32 for decompressing the encoded data stream. It is to be understood that the encoded data stream may be, e.g., received from a storage medium for decoding, or received at a desired location over a communication channel and decoded at the locatior. It is to be further understood that the input buffer 31 is an optional component that may be employed, for example, in real-time decompression applications where the rate of decompression of the decoder 32 is slower than the bandwidth of the transmitted encoded data stream.

In general, the decoder 32 performs, for the most part, the inverse of the encoding process described above. As an encoded data stream is received by the decoder 32, a bit unpacking module 33 unpacks the bits and restores the original code words generated by the encoder 12 (Fig. 1). Again, it is to be understood that the bit packing module 22 (Fig. 1) is an optional component that may be employed to provide additional compression of the code words. Therefore, if bit packing is not implemented for the encoding process, bit unpacking is not employed in the decoding process.

The decoder 32 comprises a run-length decoder 34 for processing encoded run-length sequences in the encoded data stream and outputting the decoded data corresponding to such encoded run-length sequences. As explained below, if the run-length decoder detects a control word "1" in the input data stream, it will read and process the next two successive words in the encoded stream to output the decoded data.

A dictionary decoder 35 is employed to build a dictionary 37 which is identical to the dictionary built by the encoder 12 (as discussed above). Using a mapping module 36 (or any suitable dictionary lookup function), the dictionary decoder will output character strings that are entries in the dictionary 37 to recreate the original file.

It is to be understood that the state of the dictionary of the encoder is always at least one step ahead of the state of the dictionary of the decoder. Therefore, it is possible that the encoder will output a code word for a unique data block string that the decoder has not yet entered in the decoding dictionary. This special case occurs when a character string is encoded using the string immediately preceding it. When this special situation occurs, the first and last characters of the string must be the same. Accordingly, when the decoder receives a code word that is not in the decoding dictionary, the decoder will know that the first character of the string that was encoded is equal to the last character. This *a priori* knowledge enables the decoder to handle this special case. It is to be appreciated that because there are no lengthy dictionary searches performed during the decoding process, it is much less computationally intensive than the encoding process. A decoding process according to one aspect of the present invention is described below with reference to Figs. 4A and 4B.

The decoder 32 utilizes a plurality of data storage structures 38 for temporarily storing data during a decoding process. For example, in the illustrative embodiment of Fig. 3, a Pcode data structure 39 (or "***Pcode***") is used for temporarily storing a previous code word received by the decoder 32. A Pstring data structure 40 ("***Pstring***") is employed for temporarily storing a dicticnary string corresponding to ***Pcode.*** A Ccode data structure 41 ("***Ccode***") is employed for temporarily storing a code word that is currently being processed. A Cstring data structure 42 ("***Cstring***") is employed for temporarily storing a dictionary string corresponding to ***Ccode***. A C data structure 43 is employed for temporarily storing a next code word (byte) ***C*** in the encoded input stream. Finally, a Pstring+C data structure 44 is used for temporarily storing a character string ***Pstring+C*** ,which is a string comprising all of the characters in ***Pstring*** plus the character in ***C***. The use of these data structures will be discussed in further detail below.

Referring now to Figs. 4a and 4b, a flow diagram illustrates a method for decompressing data according to one aspect of the present invention. In particular, the decoding process depicted in Figs. 4A and 4B illustrates a mode of operation of the system 30 of Fig. 3. Initially, the dictionary 37 will be initialized in the same manner as discussed above (step 400) i.e., the dictionary will comprises an index for each of the three control words and an index for each of the 256 characters). In addition, ***Pstring*** and ***Cstring*** are initialized to empty (step 401). It is to be understood that ***Pcode***, ***Ccode***, and ***C*** do not require initialization.

After initialization, the first code word in the encoded input stream will be read and stored in ***Ccode*** (step 402). A determination is then made as to whether the current code word (stored in ***Ccode***)is a (predefined) control word (step 403). If ***Ccode*** is a control word (affirmative determination in step 403), the decoding process will be terminated if the control word is "2" (step 404). If the control word is "1", then a run-length decoding process is commenced by reading and processing the next two words in the encoded input stream (step 405). In particular, as explained above, a code word "1" is output during the encoding process to indicate that the next two consecutive output numbers (in the encoded sequence) represent a run-length encoding sequence comprising (1) a character code and (2) a number denoting the amount of consecutive characters found in the data stream corresponding to the character code. Accordingly, assuming "X" represents the character code and "N" represents the number of consecutive "X"s, the decoder will output the character X, N times (step 406). Finally, if the control word is "0" (step 407), the decoding process is initialized (return to step 400).

On the other hand, if the current ***Ccode*** does not comprise a control word (negative determination in step 403), the dictionary will be searched to find the string ***Cstring*** corresponding to the current ***Ccode*** (step 408). It is to be understood that the first (non-control) code word in the input stream will always be found in the dictionary, i.e., the first non-control word will correspond to one of the 256 code words that are initialized in the dictionary.

Referring now to Fig. 4B, ***Pcode*** is set to be equal to ***Ccode*** (step 409) (and the string ***Pstring*** is set based on the value of ***Pcode***). The next code word will be read from the encoded input stream and stored in ***Ccode*** (step 410).

A determination is then made as to whether the current code word (stored in ***Ccode***) is a (predefined) control word (step 411). As explained above, if ***Ccode*** is a control word (affirmative determination in step 411), the decoding process will be terminated if the control word is "2" (step 412). If the control word is "1", then a run-length decoding process is commenced by reading and processing the next two words ("X" and "N", respectively) in the encoded input stream (step 413) and the decoder will output the character X, N times (step 414). If the control word is "0" (step 415), the decoding process is initialized (return to step 400).

If, on the other hand, the current ***Ccode*** is not a control code (negative determination in step 411), a determination is made as to whether there is an indexed entry (***Cstring***) in the decoding dictionary corresponding to ***Ccode*** (step 416). If there is an entry (affirmative determination in step 416) then ***Cstring*** corresponding to that ***Ccode*** is output (step 417). Then, the first character of ***Cstring*** is stored in the ***C*** data structure (step 418). A new string ***Pstring+C*** is then formed and added to the decoding dictionary (step 419).

If there is no entry in the dictionary for the current ***Ccode*** (negative determination in step 416) this is the special case described above and the decoder performs the following steps. First, the first character from ***Pstring*** is stored in the ***C*** data structure (step 420). Then, a new string ***Pstring+C*** is formed and added to the decoding dictionary (step 421). The new string ***Pstring+C*** is then output by the decoder (step 422).

The following example illustrates several iterations of the decoding process using the output from the above encoding example which was based on the input string ***"a b a b c a ..."*** The data structure are initialized as described above (steps 400 and 401). The first code is read and stored in the data structure ***Ccode***. Since the first input code corresponds to character ***a***, the current ***Ccode*** is determined not to be a control code (step 403). Accordingly, the dictionary entry ***Cstring*** (i.e., ***a***) corresponding to ***Ccode*** is output.

Pcode is then set equal to ***Ccode*** (step 409). The next code word is read and stored in the data structure ***Ccode***. Since the code word corresponds to character ***b***, Ccode is not a control code (step 411). The decoding dictionary is then searched for a match for ***Ccode*** (step 416). Since a single character string (i.e., ***b*** in this instance) is always in the dictionary, a match will be found. Since a match is guaranteed, the dictionary entry ***Cstring*** (i.e., ***b***) is output (step 417). Next, the first character of ***Cstring*** (i.e.,***b***) is stored in ***C*** (step 418). A new string ***Pstring+C*** is formed and added to the dictionary (step 419). In this example, since ***Pstring*** is the string corresponding to ***Pcode*** , which is the character ***a***, and ***C*** contains the character ***b***, the new string ***Pstring+C*** is ***ab***, which is added to the dictionary at the next available index, **D[259]**. Again, ***Pcode*** is set equal to ***Ccode***.

Then, the next code word (corresponding to character ***a*b**) is read and stored in the data structure ***Ccode***. Since this is not a control code, the dictionary is searched for a match for ***Ccode***. Again, in this instance, there will be a match. Accordingly, ***Cstring***, i.e.,***ab***, is output.

Then, the first character of ***Cstring*** (which is ***a***) is stored in ***C*** (step 418). A new string ***Pstring+C*** is formed comprising ***ba*** (i.e., ***Pstring*** is the string corresponding to ***Pcode***, ***b***, and ***C*** contains ***a***) and then added to the dictionary (step 419) at the next available index **D[260]**. Then, ***Pcode*** is set equal to ***Ccode***, and the process is repeated.

It is to be appreciated the present invention exploits various traits within run-length encoding, parametric dictionary encoding, and bit packing to provide an encoding/decoding process whose efficiency is suitable for use in real-time lossless data compression and decompression systems such as the systems disclosed in U.S. Patent Application Serial No 09/210,491, filed on December 11, 1998, entitled "Content Independent Data Compression Method and System," which is commonly assigned and fully incorporated herein by reference.

In particular, although dictionary class encoding techniques, in general, are considered superior to run-length encoding techniques, run-length encoding techniques can process and compress contiguous strings of data blocks far more optimally than dictionary encoding techniques. We have analyzed the manner in which certain programs store data. By way of example, we have determined that MICROSOFT OFFICE ™ applications use large string of repetitive characters in certain portions of programs and data files such as in the headers and footers of the files, although these run-lengths can occur in the middle of files such as .dll files, data base files and those files with embedded data structures.

Using an analysis tool that analyzes the frequency of characters (i.e., a histogram analysis of the frequency (count) of byte values), we have found that .exe files and .doc files comprise an inordinate quantity of bytes that are equal to 00hex (0s) and FFhex (255). These frequently occurring byte values often appear in contiguous strings as header, footer or byte padding values for data structures internal to the Word format. As indicated above, a run-length algorithm exploits these occurrences far more optimally than any known dictionary technique.

In addition, a further analysis of these file types on a block basis, e.g., an 8 kilobyte block or 4 kilobyte block, underscores the advantage of using a combination of dictionary and run-length encoding - the contiguous nature of the data strings that we have found in these files amplifies the benefit of the run-length encoding over the dictionary encoding since the dictionary encoding has been determined to typically provide a lower compression ratio when applied to smaller quantities of data. Therefore, while dictionary compression techniques typically yield higher compression ratios than run-length, this may not be true, e.g., for most MICROSOFT WINDOWS ™ operating system, program and data files. Accordingly, an encoding process such as described herein using a combination of run-length and dictionary encoding is far superior to compress data files, etc., that characteristically include contiguous strings of similar data blocks.

Moreover, as indicated above, the use of bit-packing in combination with the dictionary and run-length encoding advantageously provides additional compression, with a negligible increase in the overhead or processing time required for the bit-packing.

Further, the parametric nature of the algorithm allows for tailoring to a wide variety of applications and target processing architectures, wherein trades in processor throughput and instruction set mix, memory hierarchy and bandwidth, and requisite irput/output bandwidth requirements may be accommodated. By way of example, various memory bandwidths and sizes within the processing hierarchy may dictate the size of the dictionary in terms of the number of entries (or "dictionary depth"), and maximum length of each entry (or "dictionary width"). For example, the Texas Instruments Digital Signal Processor TMS320C6x and TMS320C5x employ separate onboard caches for program and data memory in a Harvard Architecture Arrangement. The caching may further have multiple levels of cached commonly known as L1 (lowest level) and L2 (higher level) onboard cache. Typically the lowest levels of cache have highest throughput. Also, caches are typically faster that external memory.

In one aspect of the present invention, by fixing the dictionary depth to place it in the appropriate level of caching, one can obtain a desired balance between the compression ratio and compression throughput. Indeed, although a larger dictionary typically produces a higher compression ratio, the larger dictionary results in slower throughput. With the current technology limit, L1 cache is typically too small to store a full dictionary and the dictionary is maintained at its optimum size in L2 cache. However, this trade is specific to the desired compression ratio and throughput.

In another aspect of the present invention, the throughput of, e.g., the encoding process can be monitored as a function of compression ratio and dictionary size. If the compression throughput is found to fall below a desired level or is otherwise desired to be increased the compression algorithm may dynamically enlarge the dictionary to increase compression ratio or decrease the dictionary to improve throughput. It should be noted that the relationship is dependent upon the entropy content of the input data stream and may be multivalued and/or non-linear. In yet another aspect of the present invention, a learning algorithm may be further applied to learn the optimum ratios using a time weighted average of throughput.

Another approach is to page dictionary entries from memory to L2 cache, L2 cache to L1 cache, or L1 cache to on board registers within the processor. This methodology can be extended to any memory hierarchy within a single or multiprocessor architecture.

In another embodiment, the present invention may adopt the use of a control signal that would affect the compression technique used by the encoder. The control signal could originate from the same source as the data. It would indicate to the encoder whether to place emphasis on the compression speed or the compression ratio during the encoding process. As indicated above, when it comes to compression speed and compression ratio, one can often be sacrificed to benefit the other.

An example of the use of such a control signal is as follows. Assume the encoder resices in a hard disk controller of a computer. The operating system driver that sends the information to be stored on the disk would generate the control signal. The driver may use an algorithm that normally sends a control signal to the encoder indicating that the encoder should use a form of the compression process that yields a very high compression ratio even if the encoding process is not very fast. When the driver has accumulated sufficient amount of data to be written to the disk, then the driver could generate a control signal to the encoder which would cause the controller to use a very fast implementation of its compression algorithm, even if it does not produce the best compression ratio.

In a particular example, the use of a control signal may be employed to set the appropriate parameters within the encoding/decoding algorithms described herein to facilitate data storage and retrieval bandwidth acceleration and provide data compression and decompressicn at rates faster than the input data stream such as disclosed in U.S. Patent Serial No. 09/266,394, filed on March 11, 1999, entitled "System and Methods For Accelerated Data Storage and Retrieval," which is commonly assigned and fully incorporated herein by reference. For example, if a data stream inputs 30 megabytes per second the losslessly compressed, real-time, output stream is 10 megabytes per second, assuming a 3:1 compression ratio. Conversely, if a compressed input data stream is 10 megabytes per second, the corresponding decompressed, real-time output stream is 30 megabytes per second, again assuming an original 3:1 lossless compression ratio. Again, using the methods described above, the accelerated data storage and retrieval rates may be modified based on the desired compression and throughput.

Although illustrative embodiments of the present invention have been described herein with reference to the accompanying drawings, it is to be understood that the present invention is not limited to those precise embodiments, and that various other changes and modifications may be affected therein by one skilled in the art without departing from the scope or spirit of the invention. All such changes and modifications are intended to be included within the scope of the invention as defined by the appended claims.

WHAT IS CLAIMED IS:

1. A method for compressing input data comprising a plurality of data blocks, the method comprising the steps of:

detecting if the input data comprises a run-length sequence of data blocks;

outputting an encoded run-length sequence, if a run-length sequence of data blocks is detected;

maintaining a dictionary comprising a plurality of code words, wherein each code word in the dictionary is associated with a unique data block string;

building a data block string from at least one data block in the input data that is not part of a run-length sequence;

searching for a code word in the dictionary having a unique data block string associated therewith that matches the built data block string; and

outputting the code word representing the built data block string.

2. The method of claim 1, wherein the step of detecting a run-length sequence comprises the steps of:

receiving an input data block;

identifying a run-length sequence if at least the next *s* successive data blocks in the input data are similar to the input data block.

3. The method of claim 2, wherein the step of outputting an encoded run-length sequence comprises the step of consecutively outputting a first control code word indicating a run-length sequence, a code word in the dictionary having a unique data block string associated therewith that corresponds to the input data block, and a word corresponding to the number of successive data blocks that are similar to the input data block.

4. The method of claim 1, wherein the step of maintaining a dictionary comprises the steps of:

dynamically generating a new code word corresponding to a built data block string, if the built data block string does not match a unique data block string in the dictionary; and

adding the new code word in the dictionary.

5. The method of claim 4, wherein the step of maintaining the dictionary further comprises the step of initializing the dictionary if the number of code words exceeds a predetermined threshold.

6. The method of claim 5, wherein the step of initializing the dictionary comprises the steps of:

resetting the dictionary to include all possible code words corresponding to a unique data block string comprising a single data block; and

outputting a control code word indicating that the dictionary has been initialized.

7. The method of claim 1, wherein the code words in the dictionary further comprises at least one control code word representing one of d:ctionary initialization, a run-length encoded sequence, an end of the input data, and a combination thereof.

8. The method of claim 1, wherein each code word in the dictionary comprises a dictionary index.

9. The method of claim 1, further comprising the step of bit-packing encoded run-length sequences and code words that are output.

10. The method of claim 1, wherein the step of building a data block string comprises the steps of:

(a) iteratively storing in a first data structure, a next successive data block in the input data to build a current data block string; and

(b) for each iteratior in step (a), updating a previous code word stored in a secord data structure to a current code word corresponding to the current data block string in the first data structure, if the code word for the current data block string in the first data structure is found in the dictionary; and

further wherein the step of outputting the code word representing the built data block string comprises the steps of outputting the previous code word stored in the second data structure, if a code word is not found in the dictionary corresponding to the current data block string in the first data structure.

11. The method of claim 10, further comprising the step of adding the current data block string to the dictionary.

12. The method of claim 11, further comprising the steps of:

storing, in a third data structure, the last data block input in the first data structure, if the current data block string is not found in the dictionary; and repeating steps (a) and (b) starting with the data block in the third data structure, if the data block in the third data structure is not part of a run-length sequence.

13. The method of claim 1, further comprising the step of maintaining a hash table comprising a plurality of arrays, wherein each array comprises all code words in the dictionary that are associated with a unique data block having a first data block whose value corresponds with an index of the array, and wherein the hash table is used for the step of searching for a code word in the dictionary.

14. A program storage device readable by a machine, tangibly embodying a program of instructions executable by the machine to perform method steps for compressing input data comprising a plurality of data blocks, the method comprising the steps of:

detecting if the input data comprises a run-length sequence of data blocks;

outputting an encoded run-length sequence, if a run-length sequence of data blocks is detected;

maintaining a dictionary comprising a plurality of code words, wherein each code word in the dictionary is associated with a unique data block string;

building a data block string from at least one data block in the input data that is not part of a run-length sequence;

searching for a code word in the dictionary having a unique data block string associated therewith that matches the built data block string; and

outputting the code word representing the built data block string.

15. The program storage device of claim 14, wherein the instructions for performing the step of detecting a run-length sequence comprise instructions for performing the steps of:

receiving an input data block;

identifying a run-length sequence if at least the next *s* successive data blocks in the input data are similar to the input data block.

16. The program storage device of claim 15, wherein the instructions for performing the step of outputting an encoded run-length sequence comprise instructions for performing the step of consecutively outputting a first control code word indicating a run-length sequence, a code word in the dictionary having a unique data block string

associated therewith that corresponds to the input data block, and a word corresponding to the number of successive data blocks that are similar to the input data block.

17. The program storage device of claim 14, wherein the instructions for performing the step of maintaining a dictionary comprise instructions for performing the steps of:

dynamically generating a new code word corresponding to a built data block string, if the built data block string does not match a unique data block string in the dictionary; and

adding the new code word in the dictionary.

18. The program storage device of claim 17, wherein the instructions for performing the step of maintaining the dictionary comprise instructions for performing the step of initializing the dictionary if the number of code words exceeds a predetermined threshold.

19. The program storage device of claim 18, wherein the instructions for performing the step of initializing the dictionary comprise instructions for performing the steps of:

resetting the dictionary to include all possible code words corresponding to a unique data block string comprising a single data block; and

outputting a control code word indicating that the dictionary has been initialized.

20. The program storage device of claim 14, wherein the code words in the dictionary further comprise at least one control code word representing one of dictionary initialization, a run-length encoded sequence, an end of the input data, and a combination thereof.

21. The program storage device of claim 14, wherein each code word in the dictionary comprises a dictionary index.

22. The program storage device of claim 14, further comprising instructions for performing the step of bit-packing encoded run-length sequences and code words that are output.

23. The program storage device of claim 14, wherein the instructions for performing the step of building a data block string comprise instructions for performing the steps

of:

(a) iteratively storing in a first data structure, a next successive data block in the input data to build a current data block string; and

(b) for each iteration in step (a), updating a previous code word stored in a second data structure to a current code word corresponding to the current data block string in the first data structure, if the code word for the current data block string in the first data structure is found in the dictionary; and

further wherein the instructions for performing the step of outputting the code word representing the built data block string comprise instructions for performing the step of outputting the previous code word stored in the second data structure, if a code word is not found in the dictionary corresponding to the current data block string in the first data structure.

24. The program storage device of claim 23, further comprising instructions for performing the step of adding the current data block string to the dictionary.

25. The program storage device of claim 24, further comprising instructions for performing the steps of:

storing, in a third data structure, the last data block input in the first data structure, if the current data block string is not found in the dictionary; and

repeating steps (a) and (b) starting with the data block in the third data structure, if the data block in the third data structure is not part of a run-length sequence.

26. The program storage device of claim 14, further comprising instructions for performing the step of maintaining a hash table comprising a plurality of arrays, wherein each array comprises all code words in the dictionary that are associated with a unique data block having a first data block whose value corresponds with an index of the array, and wherein the hash table is used for the step of searching for a code word in the dictionary.

27. A method for decompressing an encoded data stream comprising a plurality of code words, the method comprising the steps of:

maintaining a dictionary comprising a plurality of code words utilized to generate the encoded data stream, wherein the code words in the dictionary comprise control code words and code words that are each associated with a unique data block string;

decoding and outputting a run-length sequence of data blocks associated with an input code word of the encoded data stream, if the input code word is a control code word in the dictionary that indicates an encoded run-length sequence;

outputting a unique data block string in the dictionary that is associated with an input code word of the encoded data stream, if the input code word is found in the dictionary; and

if the input code word is not found in the dictionary, building a new data block string comprising (1) the unique data block string associated with a previous control word found in the dictionary and (2) the first data block of the unique data block string, adding the new string to the dictionary and outputting the new string.

28. A system for compressing input data comprising a plurality of data blocks, the system comprising:

a dictionary comprising a plurality of code words, wherein the code words comprise control code words and code words that are each mapped to a unique data block string;

a run-length encoder for encoding a sequence of similar data blocks in the input data using at least one code word in the dictionary; and a dictionary encoder for encoding a data block string comprising at least one data block in the input data using a code word in the dictionary, wherein output of the run-length encoder and dictionary encoder are combined to form an encoded data stream.

29. The system of claim 28, further comprising a system for decompressing the encoded data stream, wherein the system for decompressing the encoded data stream comprises:

a dictionary comprising a plurality of code words utilized to generate the encoded data stream, wherein the code words in the dictionary comprise control code words and code words that are each associated with a unique data block string;

a run-length decoder for decoding and outputting a run-length sequence of data blocks associated with an input code word of the encoded data stream, if the input code word is a control code word in the dictionary that indicates an encoded run-length sequence;

a dictionary decoder for outputting a unique data block string in the dictionary that is associated with an input code word of the encoded data stream, if the input code word is found in the dictionary; and if the input code word is not found in the dictionary, building a new data block string comprising (1) the unique data block string associated with a previous control word found in the dictionary and (2) the first data block of the unique data block string, adding the new string to the dictionary and outputting the new string.

30. The system of claim 29, wherein the compression and decompression systems are employed for accelerated data storage and retrieval.

PTO/SB/01 (6/95)

DECLARATION

Attorney Docket No. 8011-3

AS A BELOW NAMED INVENTOR, I hereby declare that:

My residence, post office address and citizenship are as stated next to my name.

I believe that I am the original, first and sole *(if only one name is listed below)*, or an original, first and joint inventor *(if plural names are listed below)*, of the subject matter which is claimed and for which a patent is sought on the invention entitled:

TITLE: **SYSTEM AND METHOD FOR LOSSLESS DATA COMPRESSION AND DECOMPRESSION**

the specification of which either is attached hereto or indicates an attorney docket no. 8011-3, or:

□ was filed in the U.S. Patent & Trademark Office on _______ and assigned Serial No. _____,

□ and *(if applicable)* was amended on ____,

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claims, as amended by any amendment referred to above. I acknowledge the duty to disclose information which is material to patentability and to the examination of this application in accordance with Title 37 of the Code of Federal Regulations §1.56. I hereby claim foreign priority benefits under Title 35, U.S. Code §119(a)-(d) or §365(b) of any foreign application(s) for patent or inventor's certificate, or §365(a) of any PCT international application which designated at least one country other than the United States, listed below and have also identified below any foreign applications for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

			Priority Claimed:
			Yes [ ] No [ ]
(Application Number)	*(Country)*	*Day/Month/Year filed)*	
			Yes [ ] No [ ]
(Application Number)	*(Country)*	*(Day/Month/Year filed)*	

I hereby claim the benefit under Title 35, U.S. Code, §120, of any United States application(s), or §119(e) of any United States provisional application(s), or §365(c) of any PCT International application designating the United States, listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States or PCT International application(s) in the manner provided by the first paragraph of Title 35, U.S. Code, §112, I acknowledge the duty to disclose information material to patentability as defined in Title 37, The Code of Federal Regulations, §1.56(a) which became available between the filing date of the prior application and the national or PCT international filing date of this application:

60/136,561	**May 28, 1999**	**Pending**
(Application Serial Number)	*(Filing Date)*	*(STATUS: patented, pending, abandoned)*
(Application Serial Number)	*(Filing Date)*	*(STATUS: patented, pending, abandoned)*

I hereby appoint the following attorneys: **FRANK CHAU**, Reg. No. 34,136; **JAMES J. BITETTO**, Reg. No. 40,513, **FRANK V. DeROSA**, Reg. No. 43,584; and **GASPARE J. RANDAZZO**, Reg. No. 41,528, each of them of **F. CHAU & ASSOCIATES, LLP**, 1900 Hempstead Turnpike, Suite 501, East Meadow, New York 11554 to prosecute this application and to transact all business in the U.S. Patent and Trademark Office connected therewith and with any divisional, continuation, continuation-in-part, reissue or re-examination application, with full power of appointment and with full power to substitute an associate attorney or agent, and to receive all patents which may issue thereon, and request that all correspondence be addressed to:

Frank Chau, Esq.
F. CHAU & ASSOCIATES, LLP
1900 Hempstead Turnpike, Suite 501
East Meadow, New York 11554
Area Code: 516-357-0091

I HEREBY DECLARE that all statements made herein of my own knowledge are true and th_at all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under §1001 of Title 18 U.S. Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

FULL NAME OF FIRST OR SOLE INVENTOR: James J. Fallon Citizenship USA

Inventor's signature: ______ Date: 5/24/2000

Residence & Post Office Address: 11 Wampus Close,
Armonk, New York 10504

FULL NAME OF SECOND INVENTOR: Steven L. Bo Citizenship USA

Inventor's signature: ______ Date: 5-24-00

Residence & Post Office Address: 42-09 217th Street
Bayside, New York 11361

PRINT OF DRAWINGS
AS ORIGINALLY FILE

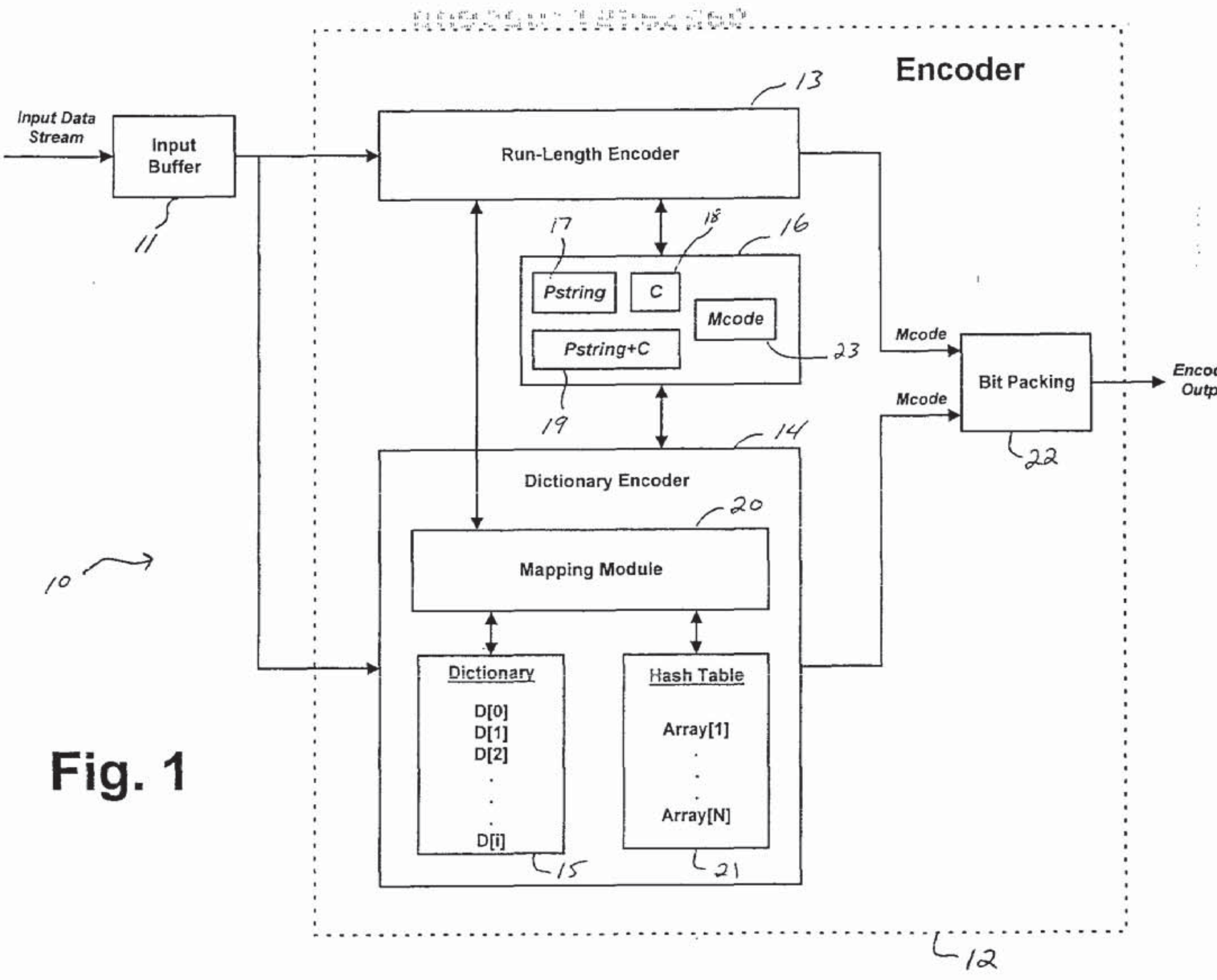
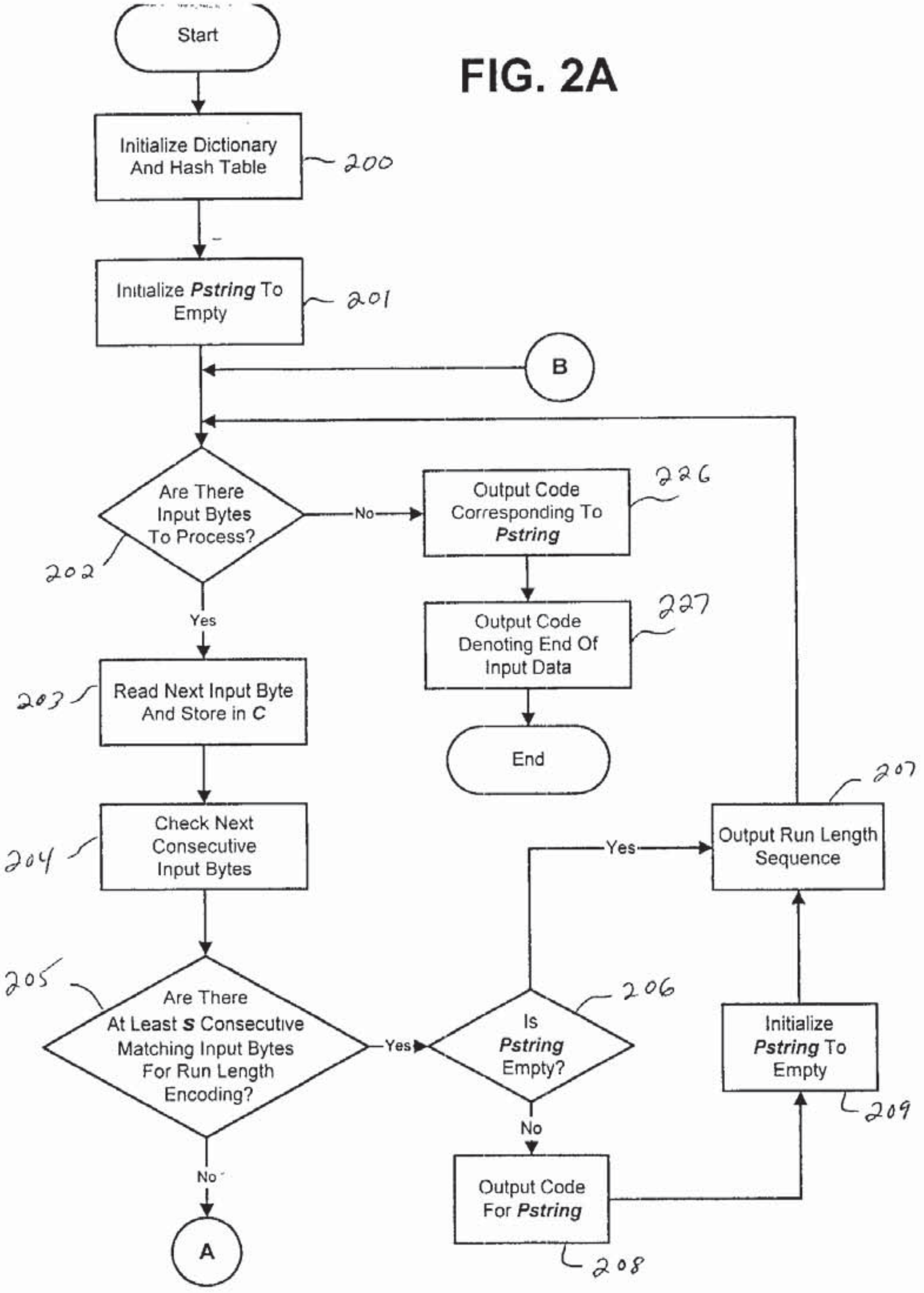


Fig. 1

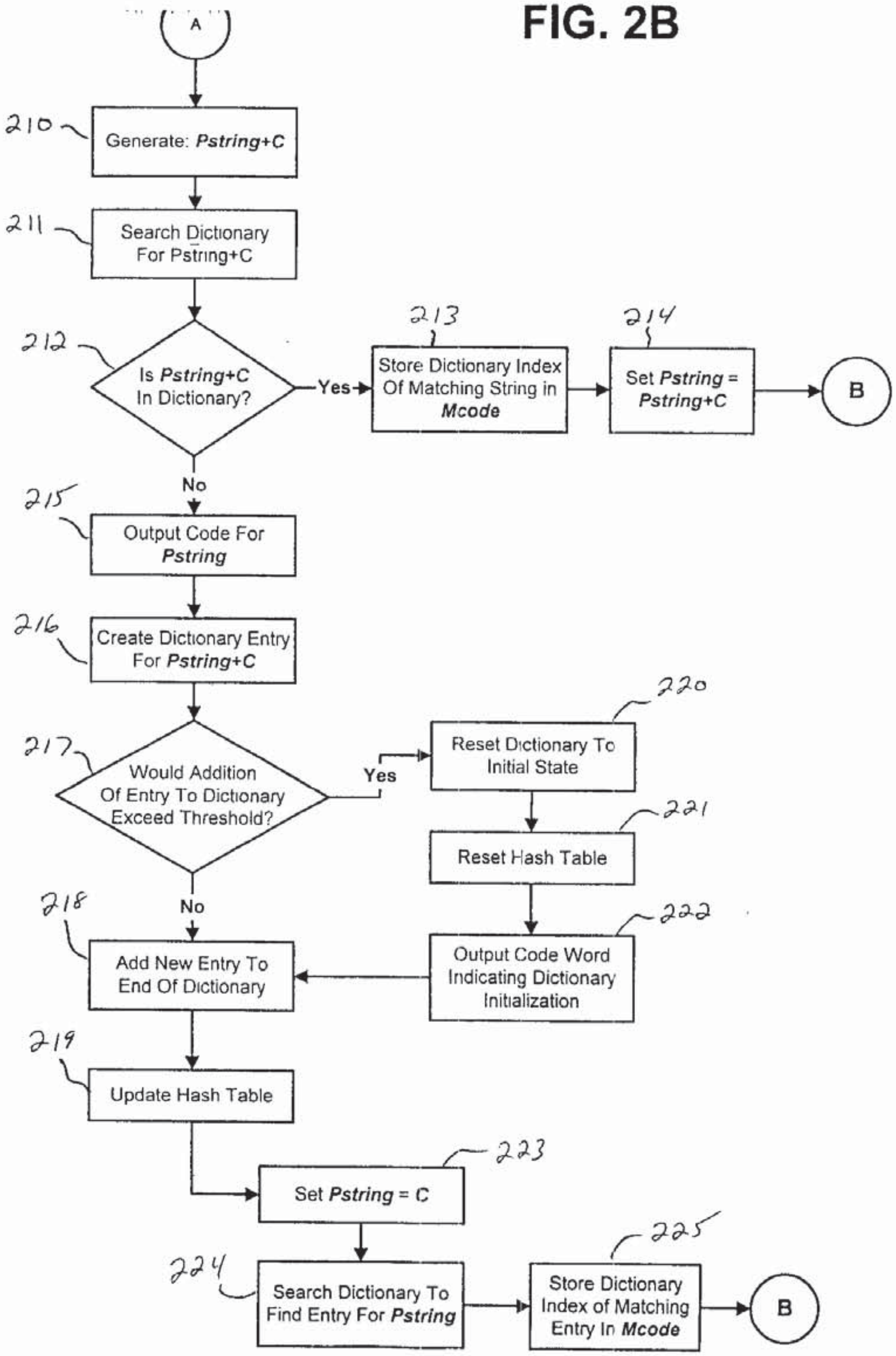
PRINT OF DRAWINGS
AS ORIGINALLY FIL

FIG. 2A


Start
Initialize Dictionary And Hash Table
200
Initialize Pstring To Empty
201
B
Are There Input Bytes To Process?
202
No
Output Code Corresponding To Pstring
226
Output Code Denoting End Of Input Data
227
End
Yes
Read Next Input Byte And Store in C
203
Check Next Consecutive Input Bytes
204
Are There At Least S Consecutive Matching Input Bytes For Run Length Encoding?
205
Yes
Is Pstring Empty?
206
Yes
Output Run Length Sequence
207
No
Output Code For Pstring
208
Initialize Pstring To Empty
209
No
A

PRINT OF DRAWINGS
AS ORIGINALLY FILF

FIG. 2B


A
210
Generate: Pstring+C
211
Search Dictionary For Pstring+C
212
Is Pstring+C In Dictionary?
Yes
213
Store Dictionary Index Of Matching String in Mcode
214
Set Pstring = Pstring+C
B
No
215
Output Code For Pstring
216
Create Dictionary Entry For Pstring+C
217
Would Addition Of Entry To Dictionary Exceed Threshold?
Yes
220
Reset Dictionary To Initial State
221
Reset Hash Table
222
Output Code Word Indicating Dictionary Initialization
No
218
Add New Entry To End Of Dictionary
219
Update Hash Table
223
Set Pstring = C
224
Search Dictionary To Find Entry For Pstring
225
Store Dictionary Index of Matching Entry In Mcode
B

PRINT OF DRAWINGS
AS ORIGINALLY FILE

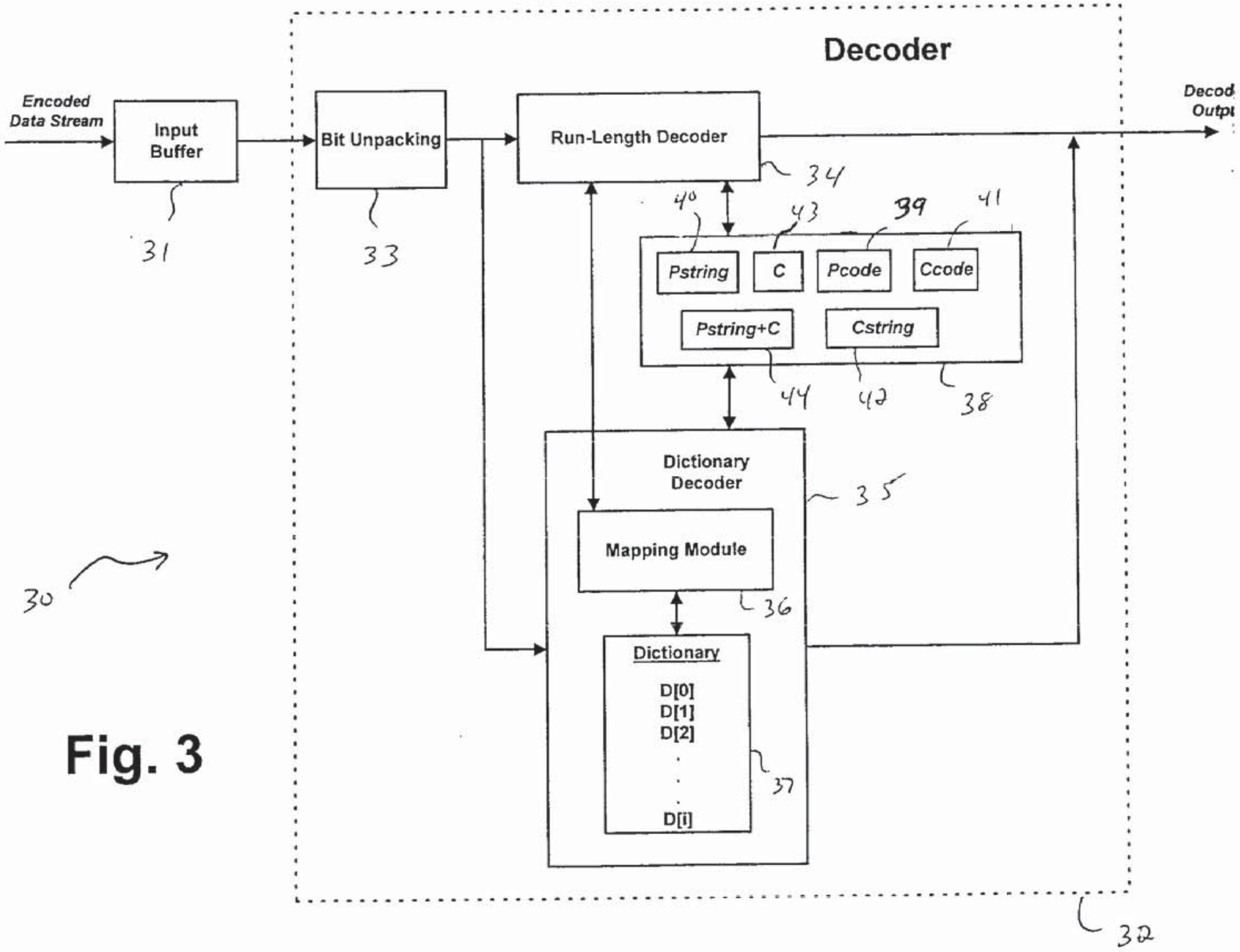


Fig. 3

PRINT OF DRAWINGS
AS ORIGINALLY FILE

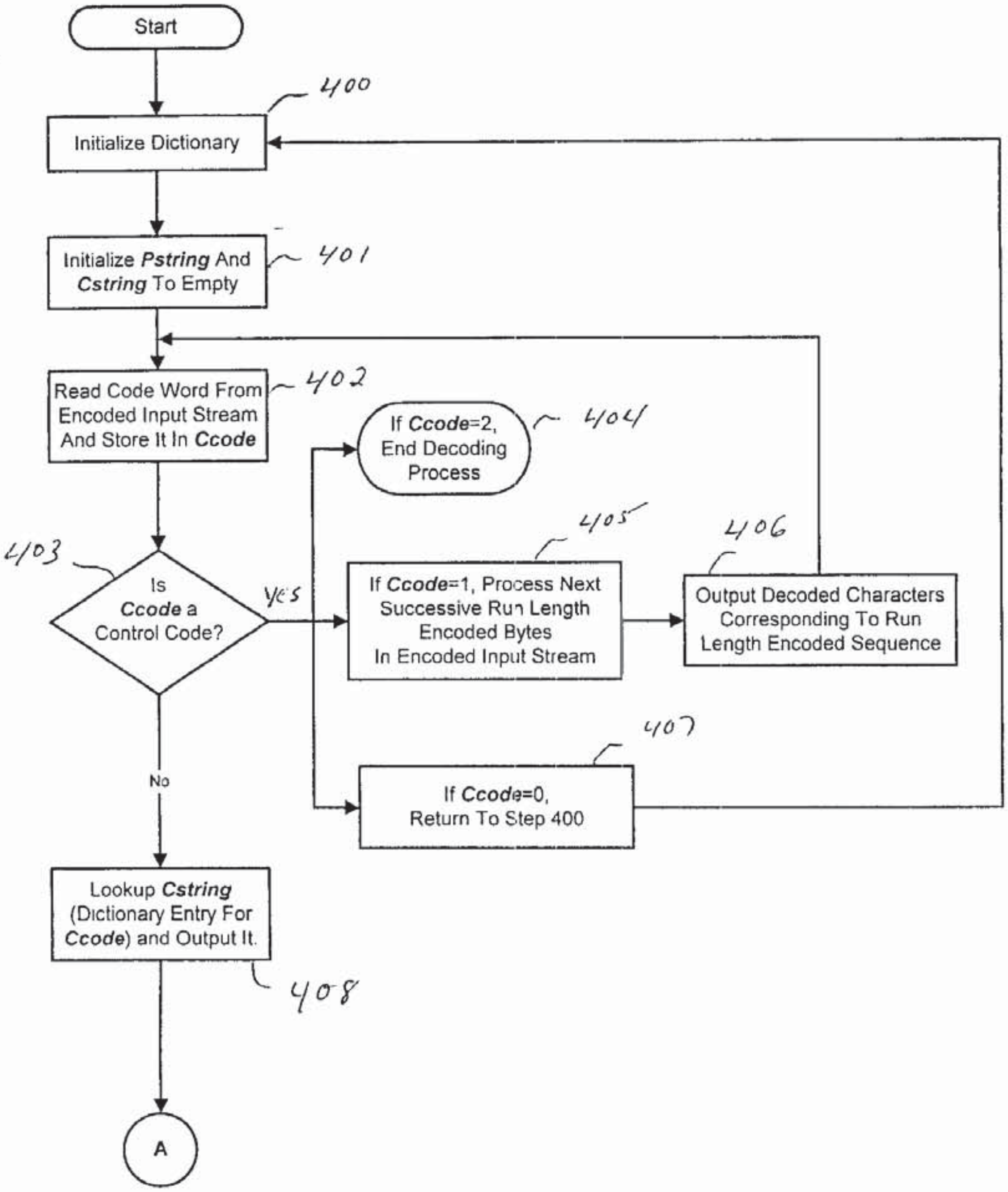

Start
400
Initialize Dictionary
Initialize Pstring And Cstring To Empty
401
Read Code Word From Encoded Input Stream And Store It In Ccode
402
If Ccode=2, End Decoding Process
404
403
Is Ccode a Control Code?
yes
405
If Ccode=1, Process Next Successive Run Length Encoded Bytes In Encoded Input Stream
406
Output Decoded Characters Corresponding To Run Length Encoded Sequence
407
If Ccode=0, Return To Step 400
No
Lookup Cstring (Dictionary Entry For Ccode) and Output It.
408
A

Fig. 4A

PRINT OF DRAWINGS
AS ORIGINALLY FILF

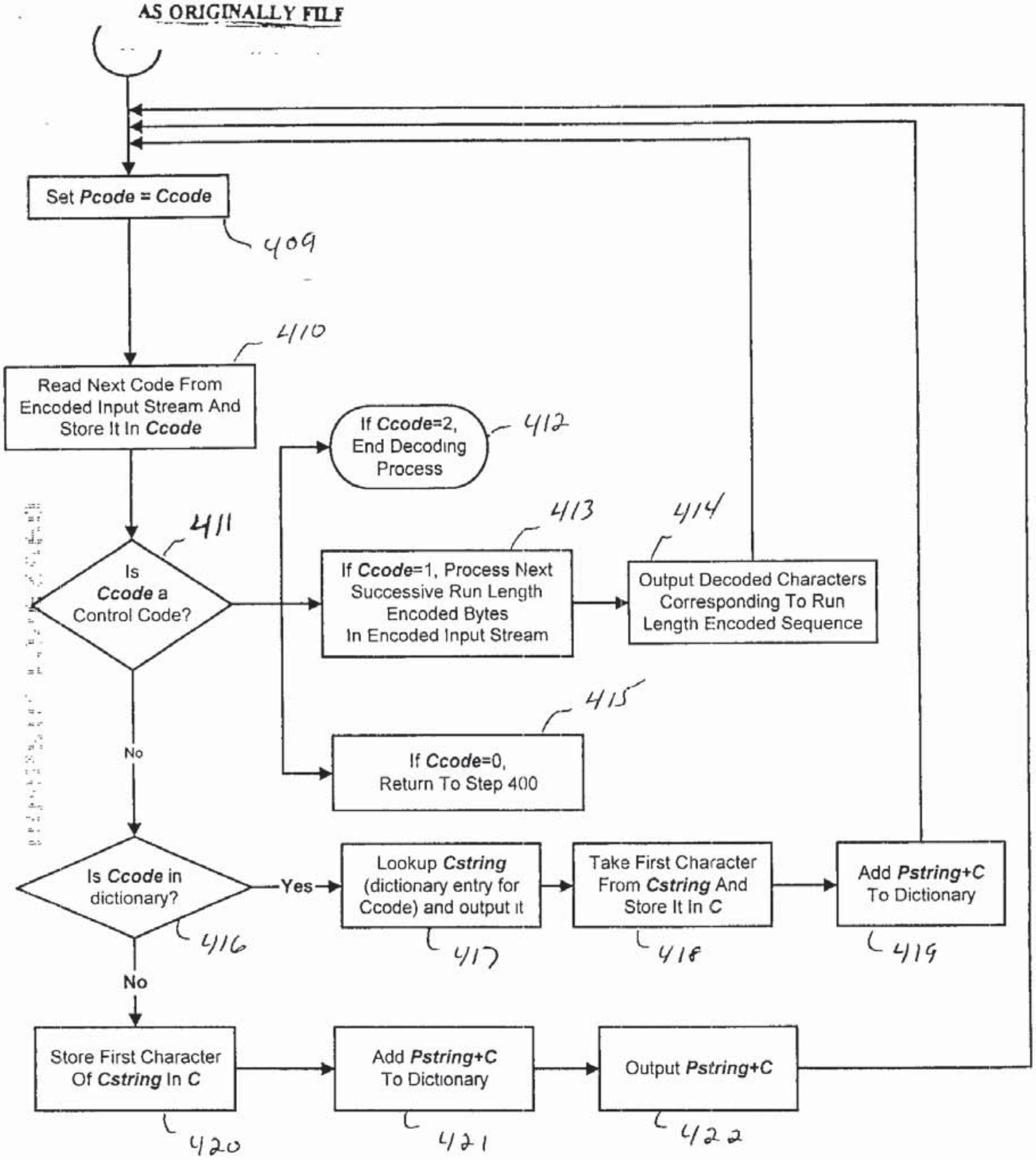

Set Pcode = Ccode
409
410
Read Next Code From Encoded Input Stream And Store It In Ccode
If Ccode=2, End Decoding Process
412
411
Is Ccode a Control Code?
413
If Ccode=1, Process Next Successive Run Length Encoded Bytes In Encoded Input Stream
414
Output Decoded Characters Corresponding To Run Length Encoded Sequence
415
If Ccode=0, Return To Step 400
No
Is Ccode in dictionary?
416
Yes
Lookup Cstring (dictionary entry for Ccode) and output it
417
Take First Character From Cstring And Store It In C
418
Add Pstring+C To Dictionary
419
No
Store First Character Of Cstring In C
420
Add Pstring+C To Dictionary
421
Output Pstring+C
422

Fig. 4B

JC836 U.S. PTO
05/26/00

05-30-00 A

PATENT APPLICATION

Atty. Docket No. 8011-3

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

JC714 U.S. PTO
09/579221
05/26/00

Assistant Commissioner for Patents
Washington, D.C. 20231

UTILITY APPLICATION FEE TRANSMITTAL

Sir:

Transmitted herewith for filing is the patent application of

Inventor(s): James J. Fallon, Steven L. Bo

For: SYSTEM AND METHOD FOR LOSSLESS DATA COMPRESSION AND DECOMPRESSION

Enclosed are:

[X] 46 page(s) of specification

[X] 1 page(s) of Abstract

[X] 13 page(s) of claims

[X] 6 sheets of drawings [ ] formal [X] informal

[X] 2 page(s) of Declaration and Power of Attorney

[ ] An Assignment of the invention to ____________________

CERTIFICATION UNDER 37 C.F.R. § 1.10

I hereby certify that this New Application Transmittal and the documents referred to as enclosed therein are being deposited with the United States Postal Service on this date May 26, 2000 in an envelope as "Express Mail Post Office to Addressee" Mail Label Number EL433927031US addressed to: Assistant Commissioner for Patents, Washington, D.C. 20231.

Frank V. DeRosa
(Type or print name of person mailing paper)

Frank DeR

(Signature of person mailing paper)

Page 1 of 3

[X] This application claims the benefit under 35 U.S.C. §119(e) of U.S. Provisional Application(s) No(s).:

APPLICATION NO(S) :	FILING DATE
60/136,561	May 28, 1999
__/	

[ ] Certified copy of applications

Country	Appln. No.	Filed

from which priority under Title 35 United States Code, § 119 is claimed

[ ] is enclosed.

[ ] will follow.

CALCULATION OF UTILITY APPLICATION FEE

For	Number Filed	Number Extra	Rate	Basic Fee $690.00
Total Claims*	30 -20 =	10	x $ 18.00	$180.00
Independent Claims	4 -3 =	1	x $ 78.00	$ 78.00
Multiple Dependent	[ ] yes	Add'l. Fee	$260.00	$
Claims	[ ] no	Add'l. Fee	None =	$
			TOTAL	$ 948.00

[X] Verified Statement of "Small Entity" Status Under 37 C.F.R. § 1.27. Reduced fees under 37 C.F.R. § 1.9(f) (50% of total) paid herewith $474.00.

*Includes all independent and single dependent claims and all claims referred to in multiple claims. See 37 C.F.R. § 1.75(c)

[ ] A check in the amount of $40.00 is enclosed for recording the attached Assignment.

[X] A check in the amount of $474.00 to cover the filing fee is attached.

[ ] Charge fee to Deposit Account No. 50-0679. Order No. 50-0679. TWO (2) COPIES OF THIS SHEET ARE ENCLOSED.

[X] Please charge any deficiency as well as any other fee(s) which may become due under 37 C.F.R. § 1.16 and 1.17, at any time during the pendency of this application, or credit any overpayment of such fee(s) to Deposit Account No. 50-0679. Also, in the event any extensions of time for responding are required for the pending application(s), please treat this paper as a petition to extend the time as required and charge Deposit Account No. 50-0679 therefor. TWO (2) COPIES OF THIS SHEET ARE ENCLOSED.

Date: 5/26/2000

SIGNATURE OF ATTORNEY
Frank V. DeRosa
Reg. No. 43,584

F. CHAU & ASSOCIATES, LLP
1900 Hempstead Turnpike
Suite 501
East Meadow, New York 11554
Tel. No. (516) 357-0091
Fax. (516) 357-0092
FVD:pg

05/26/00
1c836 U.S. PTO

Jc821 U.S. PTO
05/26/00

Please type a plus sign (+) inside this box → [+]

PTO/SB/05
Approved for use through 09/30/2000. OMB 0651-0032
Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

UTILITY PATENT APPLICATION TRANSMITTAL

(Only for new nonprovisional applications under 37 C.F.R. § 1.53(b))

Attorney Docket No.	8011-3
First Inventor or Application Identifier	FALLON
Title	SYSTEM AND METHOD FOR LOSSLESS DATA....
Express Mail Label No.	EL433927031US

APPLICATION ELEMENTS

See MPEP chapter 600 concerning utility patent application contents.

ADDRESS TO: Assistant Commissioner for Patents
Box Patent Application
Washington, DC 20231

1. ☒ * Fee Transmittal Form *(e.g., PTO/SB/17)* *(Submit an original and a duplicate for fee processing)*
2. ☒ Specification [*Total Pages* 60]
   (preferred arrangement set forth below)
   - Descriptive title of the Invention
   - Cross References to Related Applications
   - Statement Regarding Fed sponsored R & D
   - Reference to Microfiche Appendix
   - Background of the Invention
   - Brief Summary of the Invention
   - Brief Description of the Drawings (*if filed*)
   - Detailed Description
   - Claim(s)
   - Abstract of the Disclosure
3. ☒ Drawing(s) *(35 U.S.C. 113)* [*Total Sheets* 6]
4. Oath or Declaration [*Total Pages* 2]
   a. ☒ Newly executed (original or copy)
   b. □ Copy from a prior application (37 C.F.R. § 1.63(d)) *(for continuation/divisional with Box 16 completed)*
      i. □ DELETION OF INVENTOR(S) Signed statement attached deleting inventor(s) named in the prior application, see 37 C.F.R. §§ 1.63(d)(2) and 1.33(b).

** NOTE FOR ITEMS 1 & 13: IN ORDER TO BE ENTITLED TO PAY SMALL ENTITY FEES, A SMALL ENTITY STATEMENT IS REQUIRED (37 C.F.R. § 1.27), EXCEPT IF ONE FILED IN A PRIOR APPLICATION IS RELIED UPON (37 C.F.R. § 1.28).*

5. □ Microfiche Computer Program *(Appendix)*
6. Nucleotide and/or Amino Acid Sequence Submission (*if applicable, all necessary*)
   a. □ Computer Readable Copy
   b. □ Paper Copy (identical to computer copy)
   c. □ Statement verifying identity of above copies

ACCOMPANYING APPLICATION PARTS

7. □ Assignment Papers (cover sheet & document(s))
8. □ 37 C.F.R.§3.73(b) Statement *(when there is an assignee)* ☒ Power of Attorney
9. □ English Translation Document *(if applicable)*
10. □ Information Disclosure Statement (IDS)/PTO-1449 □ Copies of IDS Citations
11. □ Preliminary Amendment
12. ☒ Return Receipt Postcard (MPEP 503) *(Should be specifically itemized)*
13. □ * Small Entity Statement(s) *(PTO/SB/09-12)* ☒ Statement filed in prior application, Status still proper and desired
14. □ Certified Copy of Priority Document(s) *(if foreign priority is claimed)*
15. ☒ Other: Check in the sum of $474.00

16. If a CONTINUING APPLICATION, *check appropriate box, and supply the requisite information below and in a preliminary amendment:*
□ Continuation □ Divisional □ Continuation-in-part (CIP) of prior application No: ______/______
Prior application information: Examiner ______ *Group / Art Unit:* ______
For CONTINUATION or DIVISIONAL APPS only: The entire disclosure of the prior application, from which an oath or declaration is supplied under Box 4b, is considered a part of the disclosure of the accompanying continuation or divisional application and is hereby incorporated by reference. The incorporation can only be relied upon when a portion has been inadvertently omitted from the submitted application parts.

17. CORRESPONDENCE ADDRESS

□ *Customer Number or Bar Code Label* *(Insert Customer No. or Attach bar code label here)* or □ *Correspondence address below*

Name	Frank V. DeRosa				
Address	F. Chau & Associates, LLP 1900 Hempstead Turnpike, Suite 501				
City	East Meadow	*State*	New York	*Zip Code*	11554
Country	USA	*Telephone*	516-357-0091	*Fax*	516-357-0092

Name (Print/Type)	Frank V. DeRosa	*Registration No. (Attorney/Agent)*	43,584
Signature		*Date*	5/26/00

Burden Hour Statement: This form is estimated to take 0.2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents, Box Patent Application, Washington, DC 20231

PTO/SB/17 (12/99)
Approved for use through 09/30/2000 OMB 0651-0032
Patent and Trademark Office U S DEPARTMENT OF COMMERCE

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

FEE TRANSMITTAL for FY 2000

Patent fees are subject to annual revision

Small Entity payments must be supported by a small entity statement, otherwise large entity fees must be paid See Forms PTO/SB/09-12 See 37 C F.R §§ 1 27 and 1 28

TOTAL AMOUNT OF PAYMENT ($) 474.00

Complete if Known	
Application Number	
Filing Date	May 26, 2000
First Named Inventor	James J. Fallon
Examiner Name	
Group / Art Unit	
Attorney Docket No.	8011-3

METHOD OF PAYMENT (check one)

1. □ The Commissioner is hereby authorized to charge indicated fees and credit any overpayments to:

Deposit Account Number: 50-0679

Deposit Account Name: F. CHAU & ASSOCIATES, LLP

□ Charge Any Additional Fee Required Under 37 CFR §§ 1 16 and 1 17

2. ☒ Payment Enclosed: ☒ Check □ Money Order □ Other

FEE CALCULATION

1. BASIC FILING FEE

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description	Fee Paid
101	690	201	345	Utility filing fee	345
106	310	206	155	Design filing fee	
107	480	207	240	Plant filing fee	
108	690	208	345	Reissue filing fee	
114	150	214	75	Provisional filing fee	

SUBTOTAL (1) ($) 345.00

2. EXTRA CLAIM FEES

			Extra Claims		Fee from below		Fee Paid
Total Claims	30	-20** =	10	x	9	=	90
Independent Claims	4	- 3** =	1	x	39	=	39
Multiple Dependent					130	=	

***or number previously paid, if greater; For Reissues, see below*

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description
103	18	203	9	Claims in excess of 20
102	78	202	39	Independent claims in excess of 3
104	260	204	130	Multiple dependent claim, if not paid
109	78	209	39	** Reissue independent claims over original patent
110	18	210	9	** Reissue claims in excess of 20 and over original patent

SUBTOTAL (2) ($) 129.00

FEE CALCULATION (continued)

3. ADDITIONAL FEES

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description	Fee Paid
105	130	205	65	Surcharge - late filing fee or oath	
127	50	227	25	Surcharge - late provisional filing fee or cover sheet	
139	130	139	130	Non-English specification	
147	2 520	147	2,520	For filing a request for reexamination	
112	920*	112	920*	Requesting publication of SIR prior to Examiner act on	
113	1,840*	113	1,840*	Requesting publication of SIR after Examiner action	
115	110	215	55	Extension for reply within first month	
116	380	216	190	Extension for reply within second month	
117	870	217	435	Extension for reply within third month	
118	1,360	218	680	Extension for reply within fourth month	
128	1,850	228	925	Extension for reply within fifth month	
119	300	219	150	Notice of Appeal	
120	300	220	150	Filing a brief in support of an appeal	
121	260	221	130	Request for oral hearing	
138	1,510	138	1,510	Petition to institute a public use proceeding	
140	110	240	55	Petition to revive - unavoidable	
141	1,210	241	605	Petition to revive - unintentional	
142	1,210	242	605	Utility issue fee (or reissue)	
143	430	243	215	Design issue fee	
144	580	244	290	Plant issue fee	
122	130	122	130	Petitions to the Commissioner	
123	50	123	50	Petitions related to provisional applications	
126	240	126	240	Submission of Information Disclosure Stmt	
581	40	581	40	Recording each patent assignment per property (times number of properties)	
146	690	246	345	Filing a submission after final rejection (37 CFR § 1 129(a))	
149	690	249	345	For each additional invention to be examined (37 CFR § 1.129(b))	

Other fee (specify) ______________________

Other fee (specify) ______________________

* Reduced by Basic Filing Fee Paid

SUBTOTAL (3) ($)

SUBMITTED BY				Complete (if applicable)	
Name (Print/Type)	Frank V. DeRosa	Registration No (Attorney/Agent)	43,584	Telephone	(516) 357-0091
Signature	Frank De			Date	5/26/00

WARNING:

Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.

Burden Hour Statement: This form is estimated to take 0 2 hours to complete Time will vary depending upon the needs of the individual case Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, Patent and Trademark Office, Washington, DC 20231 DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS SEND TO Assistant Commissioner for Patents, Washington, DC 20231

Notice of Allowability	**Application No.** 09/579,221	**Applicant(s)** FALLON ET AL.	
	Examiner Jingge Wu	**Art Unit** 2623	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address--

All claims being allowable, PROSECUTION ON THE MERITS IS (OR REMAINS) CLOSED in this application. If not included herewith (or previously mailed), a Notice of Allowance (PTOL-85) or other appropriate communication will be mailed in due course. **THIS NOTICE OF ALLOWABILITY IS NOT A GRANT OF PATENT RIGHTS.** This application is subject to withdrawal from issue at the initiative of the Office or upon petition by the applicant. See 37 CFR 1.313 and MPEP 1308.

1. ☒ This communication is responsive to *5/26/2000*.

2. ☒ The allowed claim(s) is/are *1-30*

3. ☐ The drawings filed on _____ are accepted by the Examiner.

4. ☐ Acknowledgment is made of a claim for foreign priority under 35 U S C. § 119(a)-(d) or (f).

a) ☐ All b) ☐ Some* c) ☐ None of the:

1. ☐ Certified copies of the priority documents have been received.

2. ☐ Certified copies of the priority documents have been received in Application No. _____ .

3. ☐ Copies of the certified copies of the priority documents have been received in this national stage application from the International Bureau (PCT Rule 17.2(a)).

* Certified copies not received: _____.

5. ☐ Acknowledgment is made of a claim for domestic priority under 35 U.S C. § 119(e) (to a provisional application).

(a) ☐ The translation of the foreign language provisional application has been received.

6. ☐ Acknowledgment is made of a claim for domestic priority under 35 U.S C §§ 120 and/or 121.

Applicant has THREE MONTHS FROM THE "MAILING DATE" of this communication to file a reply complying with the requirements noted below. Failure to timely comply will result in ABANDONMENT of this application. **THIS THREE-MONTH PERIOD IS NOT EXTENDABLE**

7. ☐ A SUBSTITUTE OATH OR DECLARATION must be submitted Note the attached EXAMINER'S AMENDMENT or NOTICE OF INFORMAL PATENT APPLICATION (PTO-152) which gives reason(s) why the oath or declaration is deficient.

8. ☒ CORRECTED DRAWINGS must be submitted.

(a) ☒ including changes required by the Notice of Draftsperson's Patent Drawing Review (PTO-948) attached

1) ☒ hereto or 2) ☐ to Paper No. ____.

(b) ☐ including changes required by the proposed drawing correction filed _____, which has been approved by the Examiner.

(c) ☐ including changes required by the attached Examiner's Amendment / Comment or in the Office action of Paper No. _____.

Identifying indicia such as the application number (see 37 CFR 1.84(c)) should be written on the drawings in the top margin (not the back) of each sheet. The drawings should be filed as a separate paper with a transmittal letter addressed to the Official Draftsperson.

9. ☐ DEPOSIT OF and/or INFORMATION about the deposit of BIOLOGICAL MATERIAL must be submitted. Note the attached Examiner's comment regarding REQUIREMENT FOR THE DEPOSIT OF BIOLOGICAL MATERIAL.

Attachment(s)

1☒ Notice of References Cited (PTO-892)

3☒ Notice of Draftperson's Patent Drawing Review (PTO-948)

5☐ Information Disclosure Statements (PTO-1449), Paper No. ____.

7☐ Examiner's Comment Regarding Requirement for Deposit of Biological Material

2☐ Notice of Informal Patent Application (PTO-152)

4☐ Interview Summary (PTO-413), Paper No.____.

6☐ Examiner's Amendment/Comment

8☒ Examiner's Statement of Reasons for Allowance

9☐ Other

JINGGE WU
PATENT EXAMINER

Reasons for Allowance

1. The following is an examiner's statement of reasons for allowance:

Independent claims 1, 14, 27 and 28 are allowable over the prior art of record. Claims 2-13, 15-26, and 29-30 depend from claims 1, 14, and 28 respectively, therefore, are allowed.

Independent claims 1 and 14 recite the limitations of : building a data block string from at least one data block in the input data that is not part of a run-length sequence; searching for a code word in the dictionary having a unique data block string associated therewith that matches the built data block string. The combination of these features as cited in the claims in combination with the other limitations of the claims, are neither disclosed nor suggested by the prior art of record.

Independent claim 27 recites the limitations of : if the input code word is not found in the dictionary, building a new data block string comprising (1) the unique data block string associated with a previous control word found in the dictionary and (2) the first data block of the unique data block string , adding the new string to the dictionary and output the new string. The combination of these features as cited in the claims in combination with the other limitations of the claims, are neither disclosed nor suggested by the prior art of record.

Independent claim recites the limitations of : a dictionary comprising a plurality of code words, wherein the code words comprise control code words and code words that are each mapped to a unique data block string; an run-length encoder, and a dictionary encoder for encoding a data block string comprising at least one data block in the input data using a code word in the dictionary, wherein output of the run-length encoder and dictionary encoder are combined to form an encoded data stream.. The combination of these features as cited in the claims in combination with the other limitations of the claims, are neither disclosed nor suggested by the prior art of record.

The closest references of US 5883975 to Narita et al. discloses using run-length encoder, and dictionary. However, he does not teach the limitations of cited above.

Conclusion

2. The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

US 6489902 and 5995976 to Heath, US 5870036 to Franaszek et al., and US 6195024 to Fallon disclose methods for using run-length encoding and dictionary.

Contact Information

3. Any inquiry concerning this communication or earlier communications should be directed to Jingge Wu whose telephone number is (703) 308-9588. He can normally be reached Monday through Thursday from 8:00 am to 4:30 pm. The examiner can be also reached on second alternate Fridays.

Any inquiry of a general nature or relating to the status of this application should be directed to TC customer service whose telephone number is (703) 306-0377.

If attempts to reach the Examiner by telephone are unsuccessful, the Examiner's supervisor, Amelia Au, can be reached at (703) 308-6604.

The Working Group Fax number is (703) 872-9314.

Jingge Wu
Primary Patent Examiner

Notice of References Cited	Application/Control No. 09/579,221	Applicant(s)/Patent Under Reexamination FALLON ET AL.	
	Examiner Jingge Wu	Art Unit 2623	Page 1 of 1

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
*	A	US-6,195,024	02-2001	Fallon	341/51
*	B	US-5,883,975	03-1999	Narita et al.	382/232
*	C	US-5,870036	02-1999	Franaszek et al.	341/51
*	D	US-6,489,902	12-2002	Heath	341/87
*	E	US-5,955,976	09-1999	Heath	341/87
	F	US-			
	G	US-			
	H	US-			
	I	US-			
	J	US-			
	K	US-			
	L	US-			
	M	US-			

FOREIGN PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	N					
	O					
	P					
	Q					
	R					
	S					
	T					

NON-PATENT DOCUMENTS

*		Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)
	U	
	V	
	W	
	X	

*A copy of this reference is not being furnished with this Office action (See MPEP § 707 05(a))
Dates in MM-YYYY format are publication dates Classifications may be US or foreign

Form PTO 948 (Rev 03/01) U S DEPARTMENT OF COMMERCE - Patent and Trademark Office Application No. 09/579221

NOTICE OF DRAFTSPERSON'S PATENT DRAWING REVIEW

The drawing(s) filed (insert date) 5/26/00

A. □ approved by the Draftsperson under 37 CFR 1.84 or 1 152

B. ☑ objected to by the Draftsperson under 37 CFR 1 84 or 1.152 for the reasons indicated below. The Examiner will require submission of new, corrected drawings when necessary Corrected drawing must be submitted according to the instructions on the back of this notice.

1. DRAWINGS 37 CFR 1 84(a). Acceptable categories of drawings Black ink Color.
   - ____ Color drawings are not acceptable until petition is granted. Fig(s) ____
   - ____ Pencil and non black ink not permitted Fig(s) ____
2. PHOTOGRAPHS. 37 CFR 1 84(b)
   - ____ 1 full-tone set is required Fig(s) ____
   - ____ Photographs may not be mounted 37 CFR 1 84(e)
   - ____ Poor quality (half-tone) Fig(s) ____
3. TYPE OF PAPER 37 CFR 1 84(e)
   - ____ Paper not flexible strong, white, and durable Fig(s) ____
   - ____ Erasures, alterations, overwritings, interlineations, folds, copy machine marks not accepted Fig(s) ____
   - ____ Mylar, velum paper is not acceptable (too thin) Fig(s) ____
4. SIZE OF PAPER 37 CFR 1 84(f) Acceptable sizes
   - ____ 21 0 cm by 29 7 cm (DIN size A4)
   - ____ 21 6 cm by 27.9 cm (8 1/2 x 11 inches)
   - ____ All drawing sheets not the same size Sheet(s) ____
   - ____ Drawings sheets not an acceptable size Fig(s) ____
5. MARGINS 37 CFR 1 84(g) Acceptable margins:

   Top 2.5 cm Left 2 5cm Right 1 5 cm Bottom 1 0 cm
   SIZE A4 Size

   Top 2 5 cm Left 2 5 cm Right 1 5 cm Bottom 1 0 cm
   SIZE 8 1/2 x 11

   Margins not acceptable Fig(s) 1,2A,2B,3,4A,4B
   ✓ Top (T) ✓ Left (L)
   ____ Right (R) ____ Bottom (B)
6. VIEWS 37 CFR 1 84(h)

   REMINDER: Specification may require revision to correspond to drawing changes

   Partial views 37 CFR 1 84(h)(2)
   - ____ Brackets needed to show figure is one entity Fig(s) ____
   - ____ Views not labeled separately or properly Fig(s) ____
   - ____ Enlarged view not labeled separately or properly Fig(s) ____
7. SECTIONAL VIEWS 37 CFR 1 84 (h)(3)
   - ____ Hatching not indicated for sectional portions of an object Fig(s) ____
   - ____ Sectional designation should be noted with Arabic or Roman numbers Fig(s) ____
8. ARRANGEMENT OF VIEWS 37 CFR 1 84(i)
   - ____ Words do not appear on a horizontal, left-to-right fashion when page is either upright or turned so that the top becomes the right side, except for graphs Fig(s) ____
9. SCALE 37 CFR 1 84(k)
   - ____ Scale not large enough to show mechanism without crowding when drawing is reduced in size to two-thirds in reproduction Fig(s) ____
10. CHARACTER OF LINES, NUMBERS, & LETTERS 37 CFR 1 84(l)
    - ____ Lines, numbers & letters not uniformly thick and well defined, clean, durable, and black (poor line quality) Fig(s) ____
11. SHADING 37 CFR 1.84(m)
    - ____ Solid black areas pale Fig(s) ____
    - ____ Solid black shading not permitted Fig(s) ____
    - ____ Shade lines, pale, rough and blurred Fig(s) ____
12. NUMBERS, LETTERS, & REFERENCE CHARACTERS. 37 CFR 1 84(p)
    - ✓ Numbers and reference characters not plain and legible Fig(s) All
    - ____ Figure legends are poor. Fig(s) ____
    - ____ Numbers and reference characters not oriented in the same direction as the view. 37 CFR 1.84(p)(1) Fig(s) ____
    - ____ English alphabet not used. 37 CFR 1.84(p)(2) Figs ____
    - ____ Numbers, letters and reference characters must be at least .32 cm (1/8 inch) in height 37 CFR 1 84(p)(3) Fig(s) ____
13. LEAD LINES 37 CFR 1.84(q)
    - ____ Lead lines cross each other Fig(s) ____
    - ____ Lead lines missing Fig(s) ____
14. NUMBERING OF SHEETS OF DRAWINGS 37 CFR 1 84(t)
    - ____ Sheets not numbered consecutively, and in Arabic numerals beginning with number 1. Sheet(s) ____
15. NUMBERING OF VIEWS 37 CFR 1 84(u)
    - ____ Views not numbered consecutively, and in Arabic numerals, beginning with number 1 Fig(s) ____
16. CORRECTIONS 37 CFR 1.84(w)
    - ____ Corrections not made from prior PTO-948 dated ____
17. DESIGN DRAWINGS 37 CFR 1.152
    - ____ Surface shading shown not appropriate Fig(s) ____
    - ____ Solid black shading not used for color contrast Fig(s) ____

COMMENTS

REVIEWER CBR DATE 2/20/03 TELEPHONE NO. 703 308 1359

ATTACHMENT TO PAPER NO. ____

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address COMMISSIONER OF PATENTS AND TRADEMARKS
Washington, D.C. 20231
www.uspto.gov

NOTICE OF ALLOWANCE AND FEE(S) DUE

7590 02/28/2003

Frank Chau Esq
F Chau & Associates LLP
1900 Hempstead Turnpike
Suite 501
East Meadow, NY 11554

EXAMINER	
WU, JINGGE	
ART UNIT	CLASS-SUBCLASS
2623	382-232000

DATE MAILED: 02/28/2003

APPLICATION NO	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO	CONFIRMATION NO
09/579,221	05/26/2000	James J. Fallon	8011-3	8196

TITLE OF INVENTION SYSTEM AND METHOD FOR LOSSLESS DATA COMPRESSION AND DECOMPRESSION

APPLN TYPE	SMALL ENTITY	ISSUE FEE	PUBLICATION FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	YES	$650	$0	$650	05/28/2003

THE APPLICATION IDENTIFIED ABOVE HAS BEEN EXAMINED AND IS ALLOWED FOR ISSUANCE AS A PATENT. PROSECUTION ON THE MERITS IS CLOSED. THIS NOTICE OF ALLOWANCE IS NOT A GRANT OF PATENT RIGHTS. THIS APPLICATION IS SUBJECT TO WITHDRAWAL FROM ISSUE AT THE INITIATIVE OF THE OFFICE OR UPON PETITION BY THE APPLICANT. SEE 37 CFR 1.313 AND MPEP 1308.

THE ISSUE FEE AND PUBLICATION FEE (IF REQUIRED) MUST BE PAID WITHIN THREE MONTHS FROM THE MAILING DATE OF THIS NOTICE OR THIS APPLICATION SHALL BE REGARDED AS ABANDONED. THIS STATUTORY PERIOD CANNOT BE EXTENDED. SEE 35 U.S.C. 151. THE ISSUE FEE DUE INDICATED ABOVE REFLECTS A CREDIT FOR ANY PREVIOUSLY PAID ISSUE FEE APPLIED IN THIS APPLICATION. THE PTOL-85B (OR AN EQUIVALENT) MUST BE RETURNED WITHIN THIS PERIOD EVEN IF NO FEE IS DUE OR THE APPLICATION WILL BE REGARDED AS ABANDONED.

HOW TO REPLY TO THIS NOTICE:

I Review the SMALL ENTITY status shown above.

If the SMALL ENTITY is shown as YES, verify your current SMALL ENTITY status.

A If the status is the same, pay the TOTAL FEE(S) DUE shown above.

B. If the status is changed, pay the PUBLICATION FEE (if required) and twice the amount of the ISSUE FEE shown above and notify the United States Patent and Trademark Office of the change in status, or

If the SMALL ENTITY is shown as NO:

A. Pay TOTAL FEE(S) DUE shown above, or

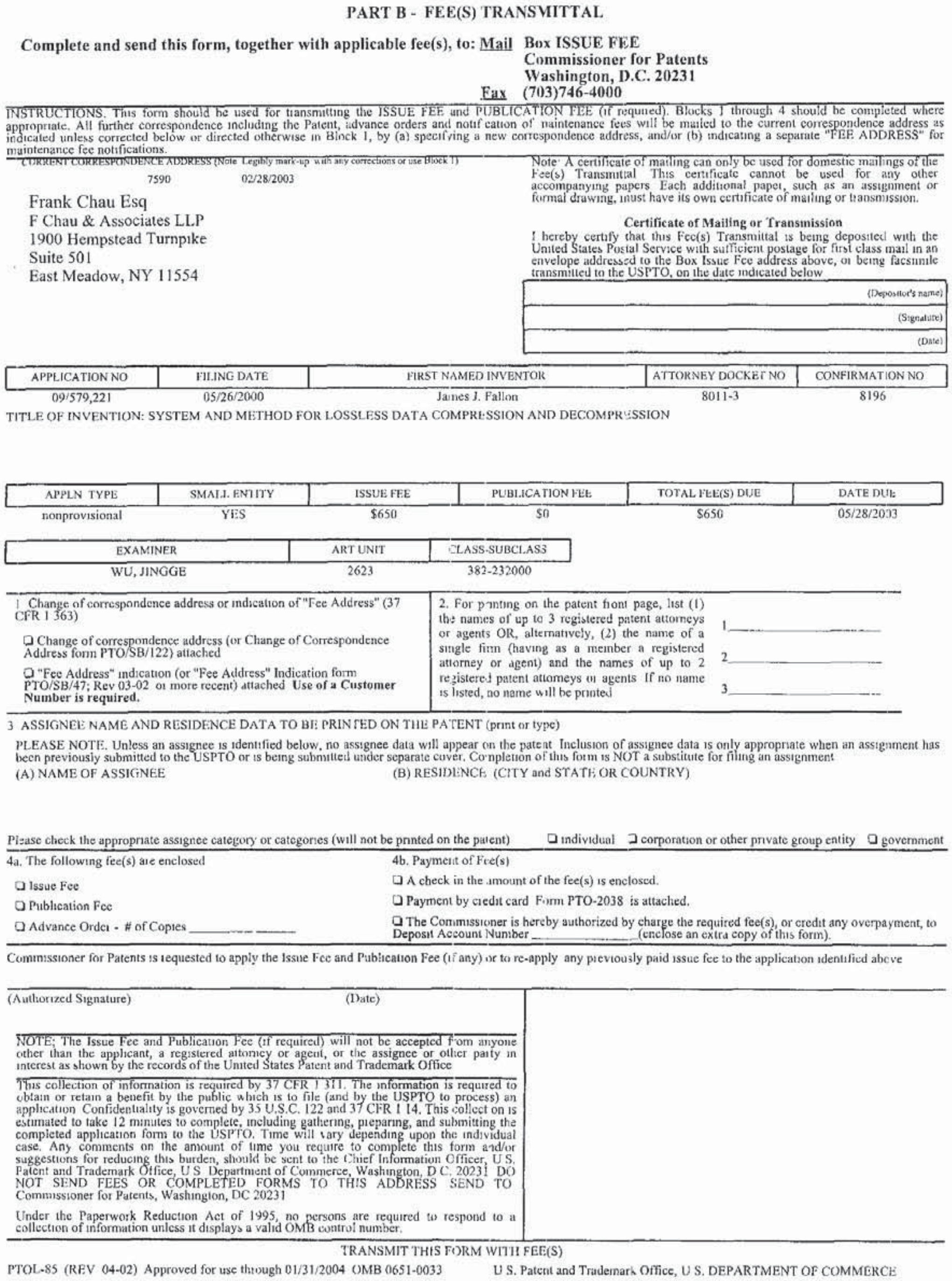
B If applicant claimed SMALL ENTITY status before, or is now claiming SMALL ENTITY status, check the box below and enclose the PUBLICATION FEE and 1/2 the ISSUE FEE shown above

□ Applicant claims SMALL ENTITY status.
See 37 CFR 1.27.

II. PART B - FEE(S) TRANSMITTAL should be completed and returned to the United States Patent and Trademark Office (USPTO) with your ISSUE FEE and PUBLICATION FEE (if required) Even if the fee(s) have already been paid, Part B - Fee(s) Transmittal should be completed and returned. If you are charging the fee(s) to your deposit account, section "4b" of Part B - Fee(s) Transmittal should be completed and an extra copy of the form should be submitted

III. All communications regarding this application must give the application number. Please direct all communications prior to issuance to Box ISSUE FEE unless advised to the contrary.

IMPORTANT REMINDER: Utility patents issuing on applications filed on or after Dec. 12, 1980 may require payment of maintenance fees. It is patentee's responsibility to ensure timely payment of maintenance fees when due.



PART B - FEE(S) TRANSMITTAL

Complete and send this form, together with applicable fee(s), to: Mail Box ISSUE FEE
Commissioner for Patents
Washington, D.C. 20231
Fax (703)746-4000

INSTRUCTIONS. This form should be used for transmitting the ISSUE FEE and PUBLICATION FEE (if required). Blocks 1 through 4 should be completed where appropriate. All further correspondence including the Patent, advance orders and notif cation of maintenance fees will be mailed to the current correspondence address as indicated unless corrected below or directed otherwise in Block 1, by (a) specifying a new correspondence address, and/or (b) indicating a separate "FEE ADDRESS" for maintenance fee notifications.

CURRENT CORRESPONDENCE ADDRESS (Note Legibly mark-up with any corrections or use Block 1)

7590 02/28/2003

Frank Chau Esq
F Chau & Associates LLP
1900 Hempstead Turnpike
Suite 501
East Meadow, NY 11554

Note: A certificate of mailing can only be used for domestic mailings of the Fee(s) Transmittal This certificate cannot be used for any other accompanying papers Each additional paper, such as an assignment or formal drawing, must have its own certificate of mailing or transmission.

Certificate of Mailing or Transmission

I hereby certify that this Fee(s) Transmittal is being deposited with the United States Postal Service with sufficient postage for first class mail in an envelope addressed to the Box Issue Fee address above, or being facsimile transmitted to the USPTO, on the date indicated below

(Depositor's name)

(Signature)

(Date)

APPLICATION NO	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO	CONFIRMATION NO
09/579,221	05/26/2000	James J. Fallon	8011-3	8196

TITLE OF INVENTION: SYSTEM AND METHOD FOR LOSSLESS DATA COMPRESSION AND DECOMPRESSION

APPLN TYPE	SMALL ENTITY	ISSUE FEE	PUBLICATION FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	YES	$650	$0	$650	05/28/2003

EXAMINER	ART UNIT	CLASS-SUBCLASS
WU, JINGGE	2623	382-232000

1 Change of correspondence address or indication of "Fee Address" (37 CFR 1 363)

□ Change of correspondence address (or Change of Correspondence Address form PTO/SB/122) attached

□ "Fee Address" indication (or "Fee Address" Indication form PTO/SB/47; Rev 03-02 or more recent) attached **Use of a Customer Number is required.**

2. For printing on the patent front page, list (1) the names of up to 3 registered patent attorneys or agents OR, alternatively, (2) the name of a single firm (having as a member a registered attorney or agent) and the names of up to 2 registered patent attorneys or agents If no name is listed, no name will be printed

1 ____________

2 ____________

3 ____________

3 ASSIGNEE NAME AND RESIDENCE DATA TO BE PRINTED ON THE PATENT (print or type)

PLEASE NOTE. Unless an assignee is identified below, no assignee data will appear on the patent Inclusion of assignee data is only appropriate when an assignment has been previously submitted to the USPTO or is being submitted under separate cover. Completion of this form is NOT a substitute for filing an assignment

(A) NAME OF ASSIGNEE

(B) RESIDENCE (CITY and STATE OR COUNTRY)

Please check the appropriate assignee category or categories (will not be printed on the patent) □ individual □ corporation or other private group entity □ government

4a. The following fee(s) are enclosed

□ Issue Fee

□ Publication Fee

□ Advance Order - # of Copies ____________

4b. Payment of Fee(s)

□ A check in the amount of the fee(s) is enclosed.

□ Payment by credit card Form PTO-2038 is attached.

□ The Commissioner is hereby authorized by charge the required fee(s), or credit any overpayment, to Deposit Account Number ____________ (enclose an extra copy of this form).

Commissioner for Patents is requested to apply the Issue Fee and Publication Fee (if any) or to re-apply any previously paid issue fee to the application identified above

(Authorized Signature) (Date)

NOTE; The Issue Fee and Publication Fee (if required) will not be accepted from anyone other than the applicant, a registered attorney or agent, or the assignee or other party in interest as shown by the records of the United States Patent and Trademark Office

This collection of information is required by 37 CFR 1 311. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1 14. This collect on is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S Department of Commerce, Washington, D C. 20231 DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS SEND TO Commissioner for Patents, Washington, DC 20231

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

TRANSMIT THIS FORM WITH FEE(S)

PTOL-85 (REV 04-02) Approved for use through 01/31/2004 OMB 0651-0033 U S. Patent and Trademark Office, U S. DEPARTMENT OF COMMERCE

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address COMMISSIONER OF PATENTS AND TRADEMARKS
Washington D C 20231
www uspto gov

APPLICATION NO	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO	CONFIRMATION NO
09/579,221	05/26/2000	James J Fallon	8011-3	8196

7590 02/28/2003

Frank Chau Esq
F Chau & Associates LLP
1900 Hempstead Turnpike
Suite 501
East Meadow, NY 11554
UNITED STATES

EXAMINER	
WU, JINGGE	

ART UNIT	PAPER NUMBER
2623	

DATE MAILED 02/28/2003

Determination of Patent Term Extension under 35 U.S.C. 154 (b)

(application filed after June 7, 1995 but prior to May 29, 2000)

The patent term extension is 0 days. Any patent to issue from the above identified application will include an indication of the 0 day extension on the front page.

If a continued prosecution application (CPA) was filed in the above-identified application, the filing date that determines patent term extension is the filing date of the most recent CPA.

Applicant will be able to obtain more detailed information by accessing the Patent Application Information Retrieval (PAIR) system. (http://pair.uspto.gov)

Any questions regarding the patent term extension or adjustment determination should be directed to the Office of Patent Legal Administration at (703)305-1383.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address COMMISSIONER OF PATENTS AND TRADEMARKS
Washington, D C 20231
www uspto gov

APPLICATION NO	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO
09/579,221	05/26/2000	James J Fallon	8011-3	8196

7590 02/28/2003

EXAMINER	
WU, JINGGE	
ART UNIT	PAPER NUMBER
2623	

Frank Chau Esq
F Chau & Associates LLP
1900 Hempstead Turnpike
Suite 501
East Meadow, NY 11554
UNITED STATES

DATE MAILED 02/28/2003

Notice of Fee Increase on January 1, 2003

If a reply to a "Notice of Allowance and Fee(s) Due" is filed in the Office on or after January 1, 2003, then the amount due will be higher than that set forth in the "Notice of Allowance and Fee(s) Due" since there will be an increase in fees effective on January 1, 2003. See Revision of Patent and Trademark Fees for Fiscal Year 2003; Final Rule, 67 Fed. Reg. 70847, 70849 (November 27, 2002).

The current fee schedule is accessible from: http://www.uspto.gov/main/howtofees.htm.

If the issue fee paid is the amount shown on the "Notice of Allowance and Fee(s) Due," but not the correct amount in view of the fee increase, a "Notice to Pay Balance of Issue Fee" will be mailed to applicant. In order to avoid processing delays associated with mailing of a "Notice to Pay Balance of Issue Fee," if the response to the Notice of Allowance and Fee(s) due form is to be filed on or after January 1, 2003 (or mailed with a certificate of mailing on or after January 1, 2003), the issue fee paid should be the fee that is required at the time the fee is paid. If the issue fee was previously paid, and the response to the "Notice of Allowance and Fee(s) Due" includes a request to apply a previously-paid issue fee to the issue fee now due, then the difference between the issue fee amount at the time the response is filed and the previously paid issue fee should be paid. See Manual of Patent Examining Procedure, Section 1308.01 (Eighth Edition, August 2001).

Questions relating to issue and publication fee payments should be directed to the Customer Service Center of the Office of Patent Publication at (703) 305-8283.

OIPE JC109
JUN 0 2 2003
TRADEMARK OFFICE

BB CW

PART B - FEE(S) TRANSMITTAL

Complete and send this form, together with applicable fee(s), to: Mail Box ISSUE FEE
Commissioner for Patents
Washington, D.C. 20231
Fax (703)746-4000

INSTRUCTIONS: This form should be used for transmitting the ISSUE FEE and PUBLICATION FEE (if required). Blocks 1 through 4 should be completed where appropriate. All further correspondence including the Patent, advance orders and notification of maintenance fees will be mailed to the current correspondence address as indicated unless corrected below or directed otherwise in Block 1, by (a) specifying a new correspondence address; and/or (b) indicating a separate "FEE ADDRESS" for maintenance fee notifications.

CURRENT CORRESPONDENCE ADDRESS (Note: Legibly mark-up with any corrections or use Block 1)

7590 02/28/2003

Frank Chau Esq
F Chau & Associates LLP
1900 Hempstead Turnpike
Suite 501
East Meadow, NY 11554

Note: A certificate of mailing can only be used for domestic mailings of the Fee(s) Transmittal. This certificate cannot be used for any other accompanying papers. Each additional paper, such as an assignment or formal drawing, must have its own certificate of mailing or transmission.

Certificate of Mailing or Transmission

I hereby certify that this Fee(s) Transmittal is being deposited with the United States Postal Service with sufficient postage for first class mail in an envelope addressed to the Box Issue Fee address above, or being facsimile transmitted to the USPTO, on the date indicated below.

Frank V. DeRosa (Depositor's name)
(Signature)
May 28, 2003 (Date)

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO
09/579,221	05/26/2000	James J. Fallon	8011-3	8196

TITLE OF INVENTION: SYSTEM AND METHOD FOR LOSSLESS DATA COMPRESSION AND DECOMPRESSION

APPLN. TYPE	SMALL ENTITY	ISSUE FEE	PUBLICATION FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	YES	$650	$0	$650	05/28/2003

EXAMINER	ART UNIT	CLASS-SUBCLASS
WU, JINGGE	2623	382-232000

1. Change of correspondence address or indication of "Fee Address" (37 CFR 1.363).

□ Change of correspondence address (or Change of Correspondence Address form PTO/SB/122) attached.

□ "Fee Address" indication (or "Fee Address" Indication form PTO/SB/47; Rev 03-02 or more recent) attached. **Use of a Customer Number is required.**

2. For printing on the patent front page, list (1) the names of up to 3 registered patent attorneys or agents OR, alternatively, (2) the name of a single firm (having as a member a registered attorney or agent) and the names of up to 2 registered patent attorneys or agents. If no name is listed, no name will be printed.

1 F. Chau & Associates, LLP
2 Frank V. DeRosa, Esq.
3 __________

3. ASSIGNEE NAME AND RESIDENCE DATA TO BE PRINTED ON THE PATENT (print or type)

PLEASE NOTE: Unless an assignee is identified below, no assignee data will appear on the patent. Inclusion of assignee data is only appropriate when an assignment has been previously submitted to the USPTO or is being submitted under separate cover. Completion of this form is NOT a substitute for filing an assignment.

(A) NAME OF ASSIGNEE: Realtime Data, LLC

(B) RESIDENCE: (CITY and STATE OR COUNTRY): New York, New York

Please check the appropriate assignee category or categories (will not be printed on the patent) □ individual ☒ corporation or other private group entity □ government

4a. The following fee(s) are enclosed:

☒ Issue Fee
□ Publication Fee
☒ Advance Order - # of Copies 1

4b. Payment of Fee(s):

□ A check in the amount of the fee(s) is enclosed.
☒ Payment by credit card. Form PTO-2038 is attached.
□ The Commissioner is hereby authorized by charge the required fee(s), or credit any overpayment, to Deposit Account Number 50-0679 (enclose an extra copy of this form).

Commissioner for Patents is requested to apply the Issue Fee and Publication Fee (if any) or to re-apply any previously paid issue fee to the application identified above.

(Authorized Signature) (Date)
Frank V. DeRosa, Reg. No. 43,584 5/28/03

NOTE: The Issue Fee and Publication Fee (if required) will not be accepted from anyone other than the applicant; a registered attorney or agent; or the assignee or other party in interest as shown by the records of the United States Patent and Trademark Office.

This collection of information is required by 37 CFR 1.311. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, Washington, D.C. 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, Washington, DC 20231.

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

06/04/2003 MAHMED2 00000029 09579221

01 FC:2501 650.00 OP
02 FC:8001 3.00 OP

TRANSMIT THIS FORM WITH FEE(S)

PTOL-85 (REV. 04-02) Approved for use through 01/31/2004. OMB 0651-0033 U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

OIPE JC109
JUN 0 2 2003
PATENT & TRADEMARK OFFICE

3PM 6/3 AH

PATENT

Atty. Docket No. 8011-3

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICANT(S): James J. Fallon   Examiner: Jingge Wu

SERIAL NO.: 09/579,221   Group Art Unit: 2623

FILED: May 26, 2000

FOR: SYSTEM AND METHOD FOR LOSSLESS DATA COMPRESSION AND DECOMPRESSION

Dated: May 28, 2003

Mail Stop Issue Fee
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

TRANSMITTAL OF FORMAL DRAWINGS

Sir:

Applicant submits herewith six (6) sheets of formal drawings depicting FIGS. 1-4B for this application.

Respectfully submitted,

Frank V. DeRosa
Reg. No. 43,584
Attorney for Applicant(s)

F. CHAU & ASSOCIATES, LLP
1900 Hempstead Turnpike
Suite 501
East Meadow, New York 11554
(516) 357-0091

CERTIFICATE OF MAILING UNDER 37 C.F.R. §1.8(a)

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail, postpaid in an envelope addressed to the Commissioner for Patents, P.O Box 1450, Alexandria, VA 22313-1450 on May 28, 2003.

Dated. May 28, 2003

Frank V DeRosa

2/03

OIPE JC139
JUN 0 2 2003
PATENT & TRADEMARK OFFICE

6597812

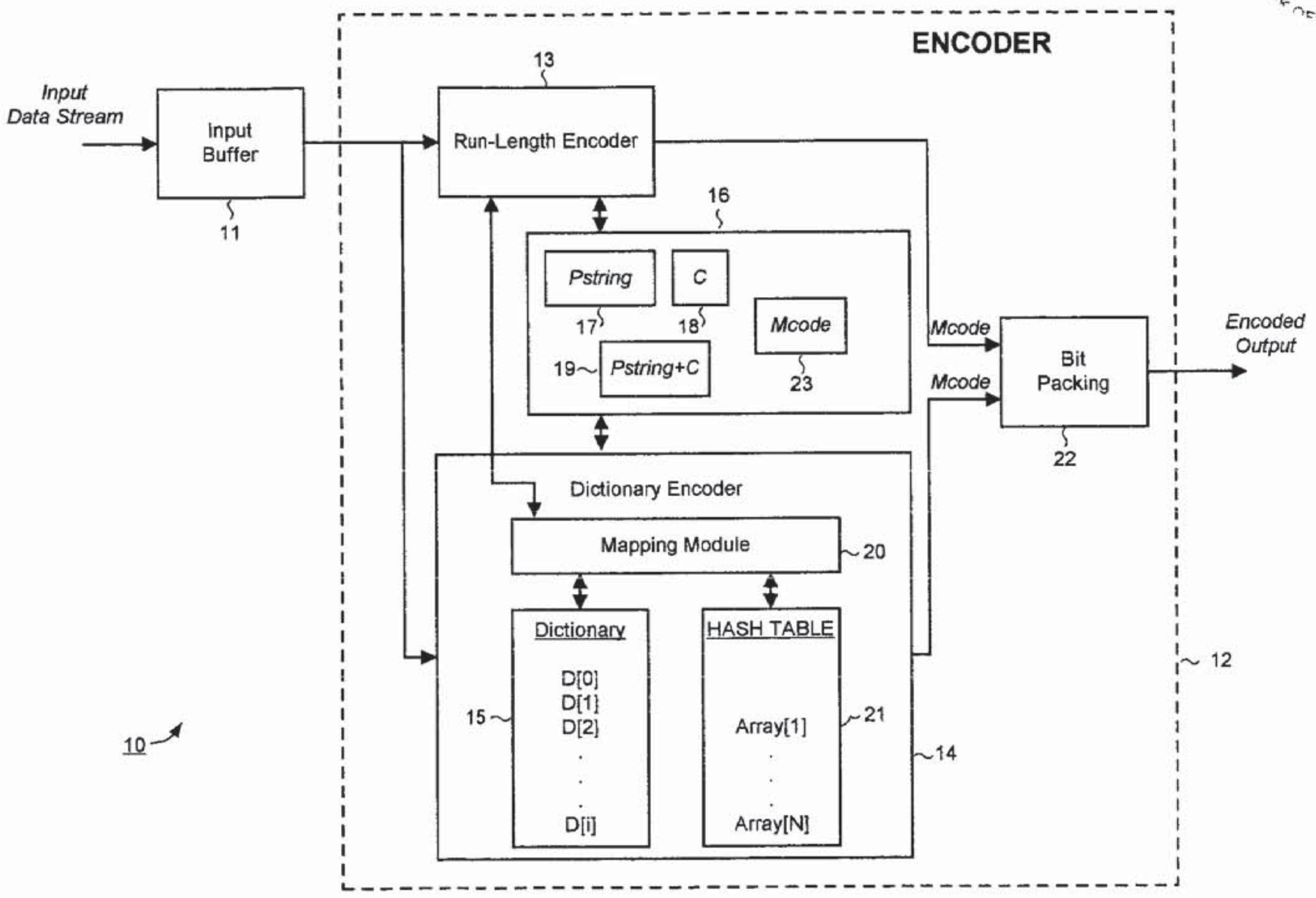


FIGURE 1

OIPE JC139
JUN 0 2 2003
TRADEMARK OFFICE

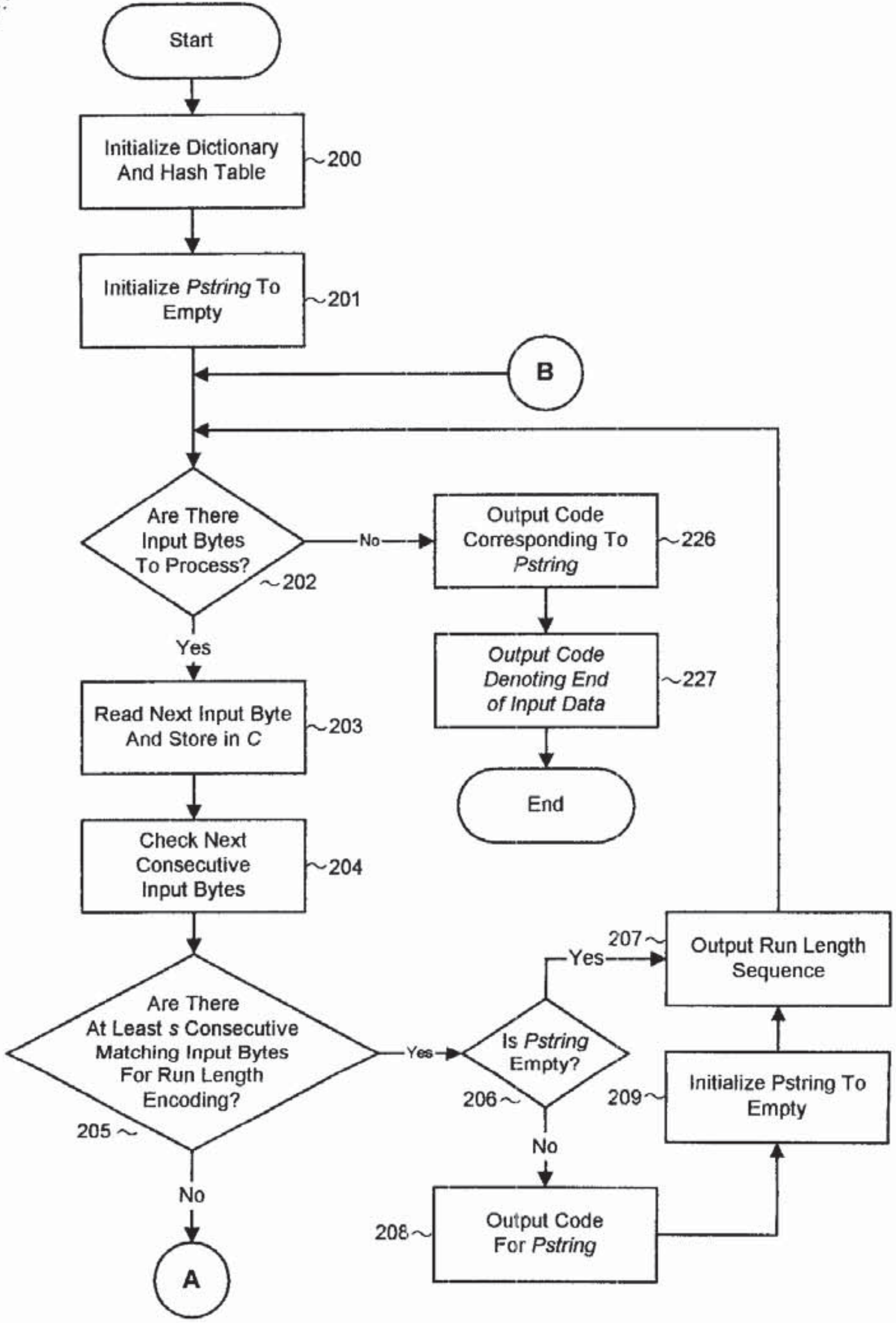

Start
Initialize Dictionary And Hash Table
200
Initialize *Pstring* To Empty
201
B
Are There Input Bytes To Process?
202
No
Output Code Corresponding To *Pstring*
226
Output Code Denoting End of Input Data
227
End
Yes
Read Next Input Byte And Store in *C*
203
Check Next Consecutive Input Bytes
204
Are There At Least *s* Consecutive Matching Input Bytes For Run Length Encoding?
205
Yes
Is *Pstring* Empty?
206
Yes
207
Output Run Length Sequence
209
Initialize Pstring To Empty
No
208
Output Code For *Pstring*
No
A

FIGURE 2A

OIPE JC109
JUN 0 2 2003
PATENT & TRADEMARK OFFICE

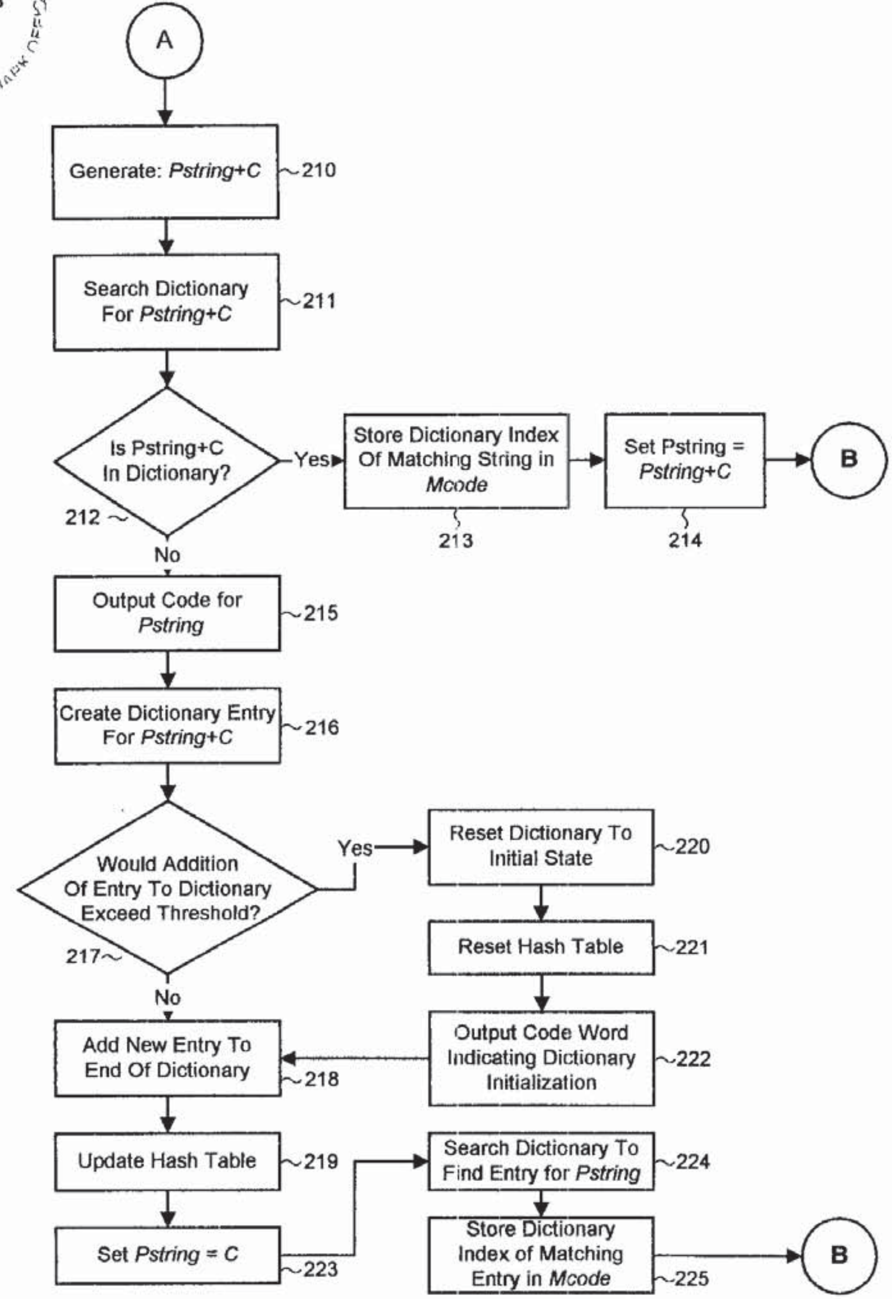

A
Generate: Pstring+C
210
Search Dictionary For Pstring+C
211
Is Pstring+C In Dictionary?
212
Yes
Store Dictionary Index Of Matching String in Mcode
213
Set Pstring = Pstring+C
214
B
No
Output Code for Pstring
215
Create Dictionary Entry For Pstring+C
216
Would Addition Of Entry To Dictionary Exceed Threshold?
217
Yes
Reset Dictionary To Initial State
220
Reset Hash Table
221
Output Code Word Indicating Dictionary Initialization
222
No
Add New Entry To End Of Dictionary
218
Update Hash Table
219
Set Pstring = C
223
Search Dictionary To Find Entry for Pstring
224
Store Dictionary Index of Matching Entry in Mcode
225
B

FIGURE 2B

OIPE JC159
JUN 0 2 2003
PATENT & TRADEMARK OFFICE

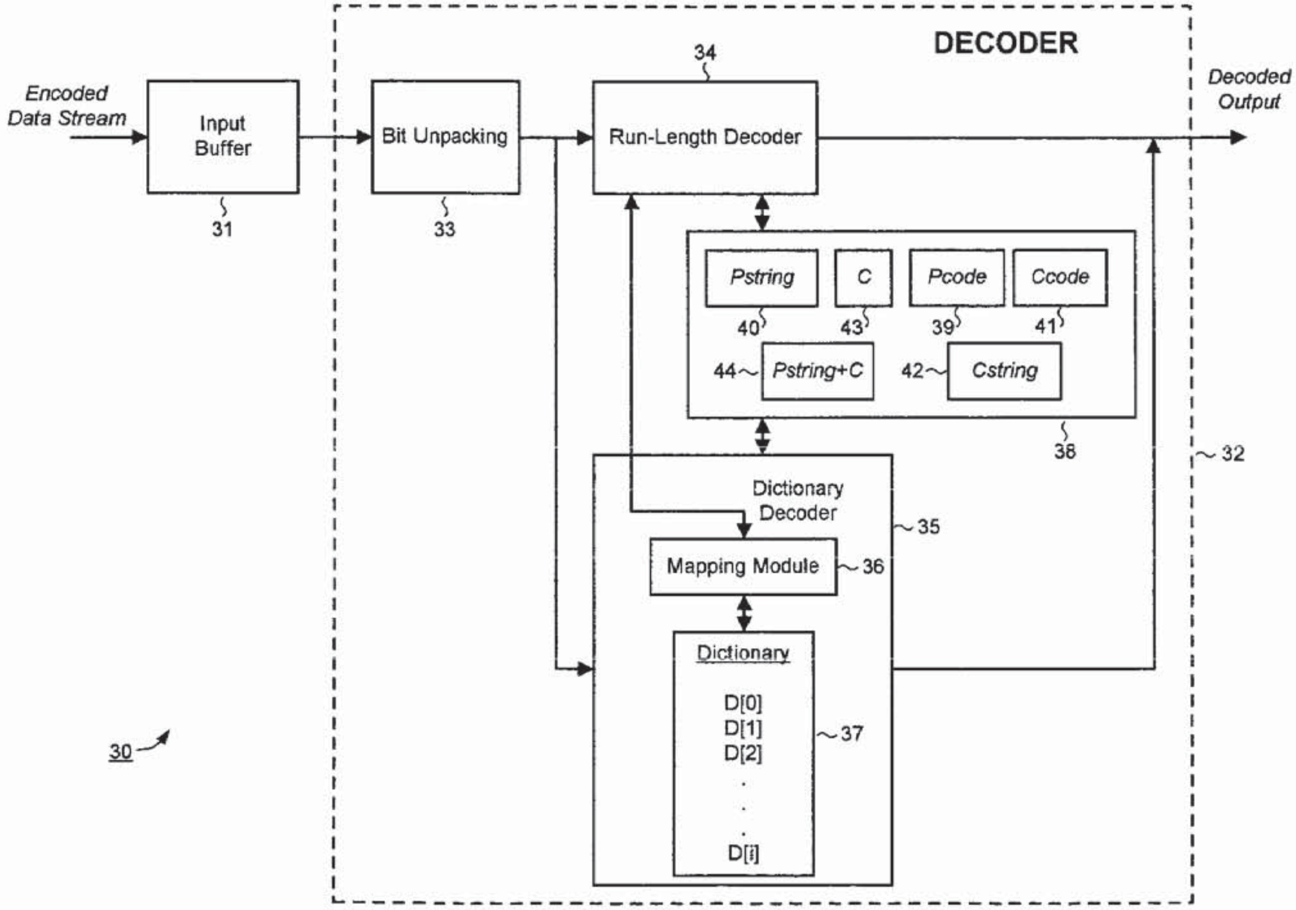


FIGURE 3

OIPE JC109
JUN 0 2 2003
PATENT & TRADEMARK OFFICE

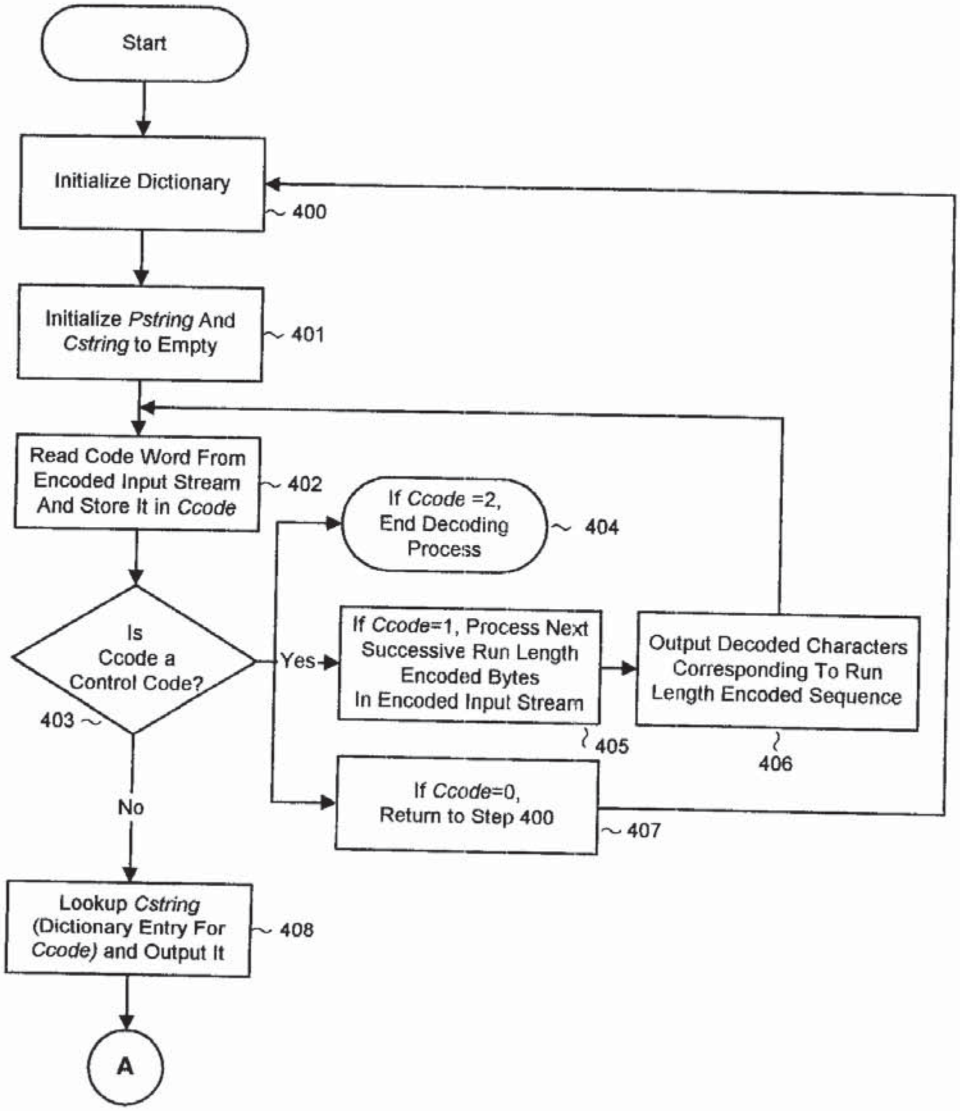

Start
Initialize Dictionary
400
Initialize *Pstring* And *Cstring* to Empty
401
Read Code Word From Encoded Input Stream And Store It in *Ccode*
402
If *Ccode* =2, End Decoding Process
404
Is Ccode a Control Code?
403
Yes
If *Ccode*=1, Process Next Successive Run Length Encoded Bytes In Encoded Input Stream
405
Output Decoded Characters Corresponding To Run Length Encoded Sequence
406
If *Ccode*=0, Return to Step 400
407
No
Lookup *Cstring* (Dictionary Entry For *Ccode)* and Output It
408
A

FIGURE 4A

OIPE JC109
JUN 0 2 2003
PATENT & TRADEMARK OFFICE

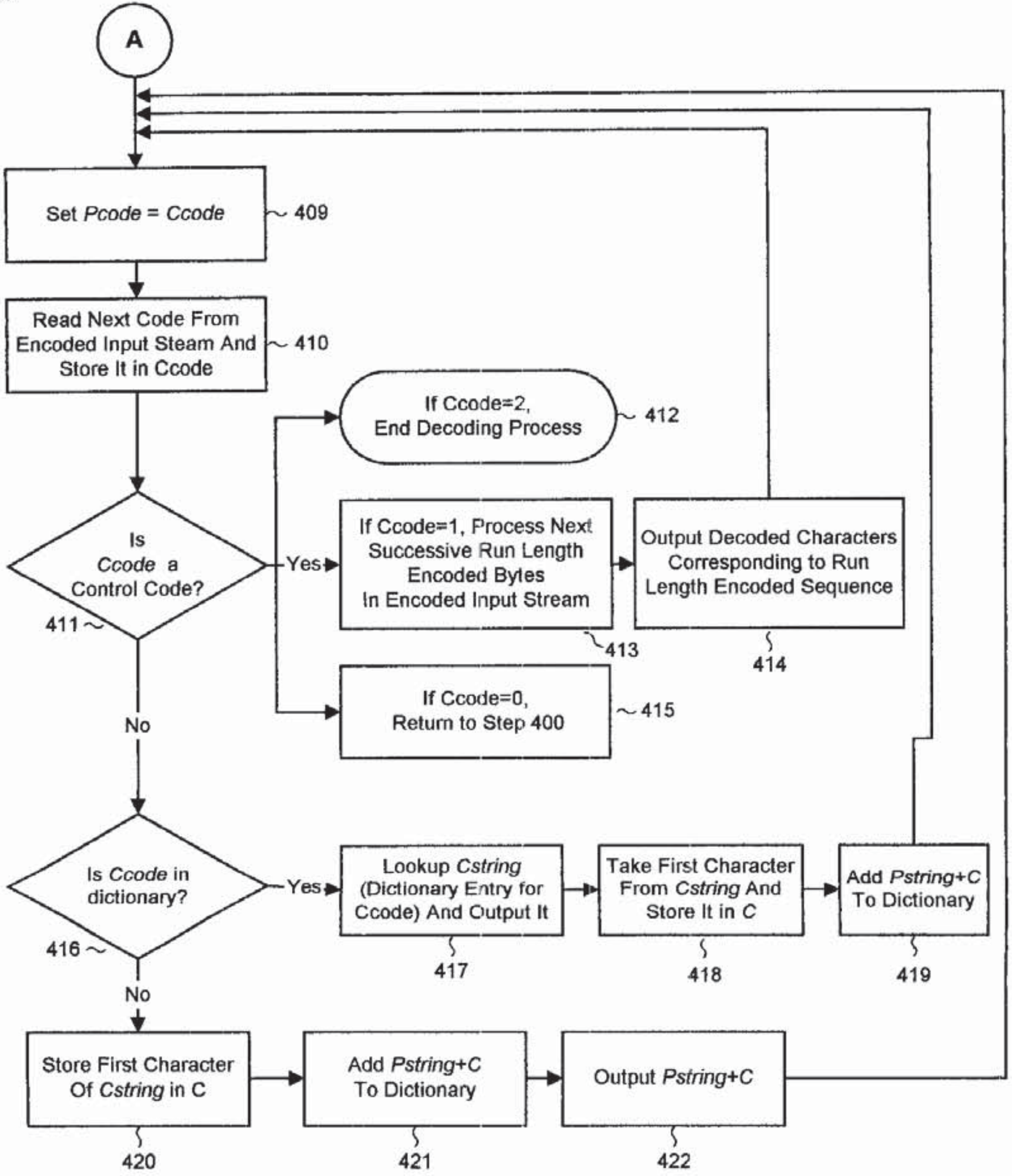
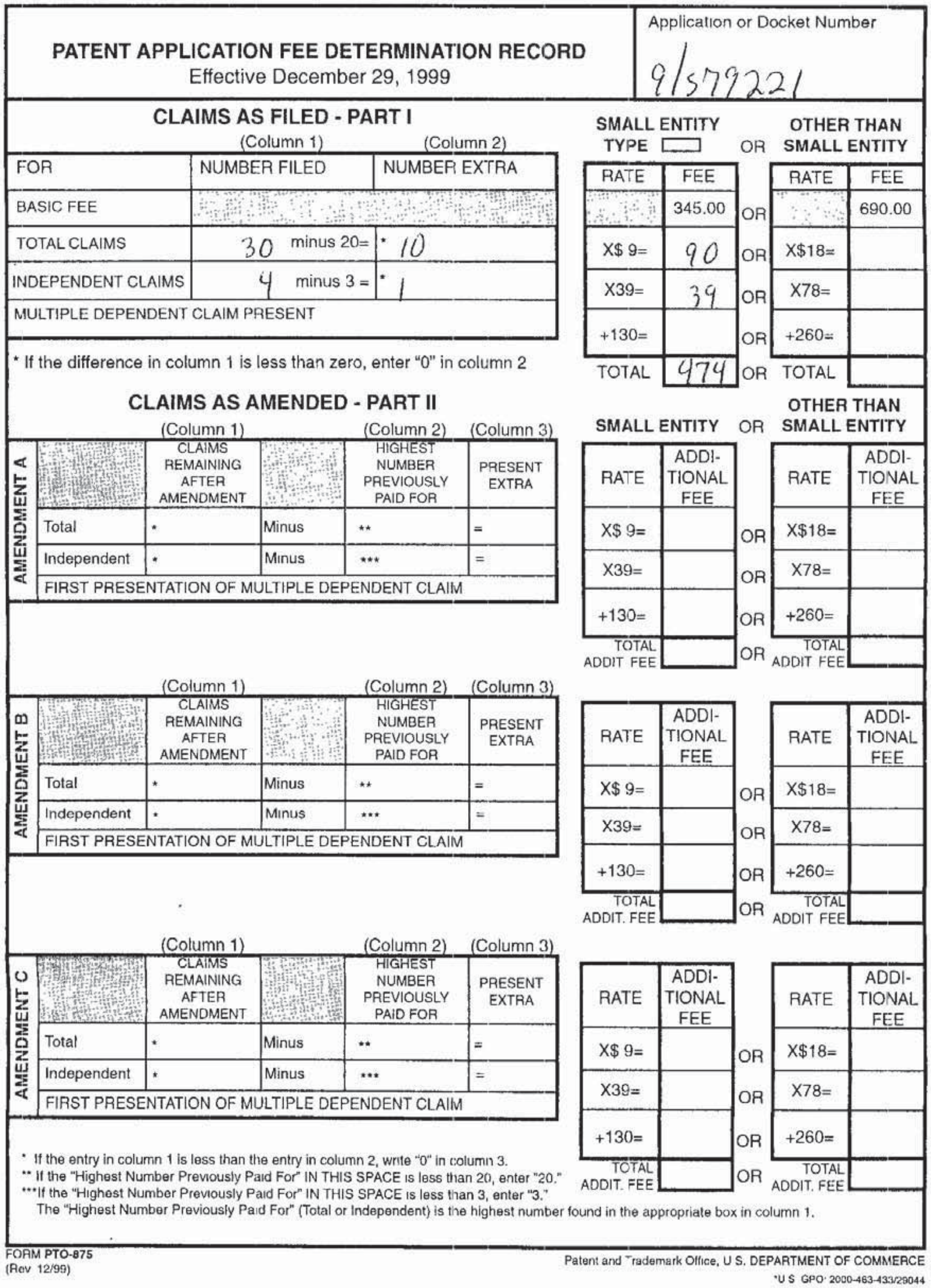

A
Set *Pcode* = *Ccode*
409
Read Next Code From Encoded Input Steam And Store It in Ccode
410
If Ccode=2, End Decoding Process
412
Is *Ccode* a Control Code?
411
Yes
If Ccode=1, Process Next Successive Run Length Encoded Bytes In Encoded Input Stream
413
Output Decoded Characters Corresponding to Run Length Encoded Sequence
414
If Ccode=0, Return to Step 400
415
No
Is *Ccode* in dictionary?
416
Yes
Lookup *Cstring* (Dictionary Entry for Ccode) And Output It
417
Take First Character From *Cstring* And Store It in *C*
418
Add *Pstring+C* To Dictionary
419
No
Store First Character Of *Cstring* in C
420
Add *Pstring+C* To Dictionary
421
Output *Pstring+C*
422

FIGURE 4B



PATENT APPLICATION FEE DETERMINATION RECORD

Effective December 29, 1999

Application or Docket Number

9/579221

CLAIMS AS FILED - PART I

	(Column 1)	(Column 2)	SMALL ENTITY TYPE □		OR	OTHER THAN SMALL ENTITY	
FOR	NUMBER FILED	NUMBER EXTRA	RATE	FEE		RATE	FEE
BASIC FEE	▨	▨	▨	345.00	OR	▨	690.00
TOTAL CLAIMS	30 minus 20=	* 10	X$ 9=	90	OR	X$18=	
INDEPENDENT CLAIMS	4 minus 3 =	* 1	X39=	39	OR	X78=	
MULTIPLE DEPENDENT CLAIM PRESENT			+130=		OR	+260=	
			TOTAL	474	OR	TOTAL	

* If the difference in column 1 is less than zero, enter "0" in column 2

CLAIMS AS AMENDED - PART II

AMENDMENT A

	(Column 1)		(Column 2)	(Column 3)	SMALL ENTITY		OR	OTHER THAN SMALL ENTITY	
▨	CLAIMS REMAINING AFTER AMENDMENT	▨	HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA	RATE	ADDI-TIONAL FEE		RATE	ADDI-TIONAL FEE
Total	*	Minus	**	=	X$ 9=		OR	X$18=	
Independent	*	Minus	***	=	X39=		OR	X78=	
FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM					+130=		OR	+260=	
					TOTAL ADDIT FEE		OR	TOTAL ADDIT FEE	

AMENDMENT B

	(Column 1)		(Column 2)	(Column 3)					
▨	CLAIMS REMAINING AFTER AMENDMENT	▨	HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA	RATE	ADDI-TIONAL FEE		RATE	ADDI-TIONAL FEE
Total	*	Minus	**	=	X$ 9=		OR	X$18=	
Independent	*	Minus	***	=	X39=		OR	X78=	
FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM					+130=		OR	+260=	
					TOTAL ADDIT. FEE		OR	TOTAL ADDIT FEE	

AMENDMENT C

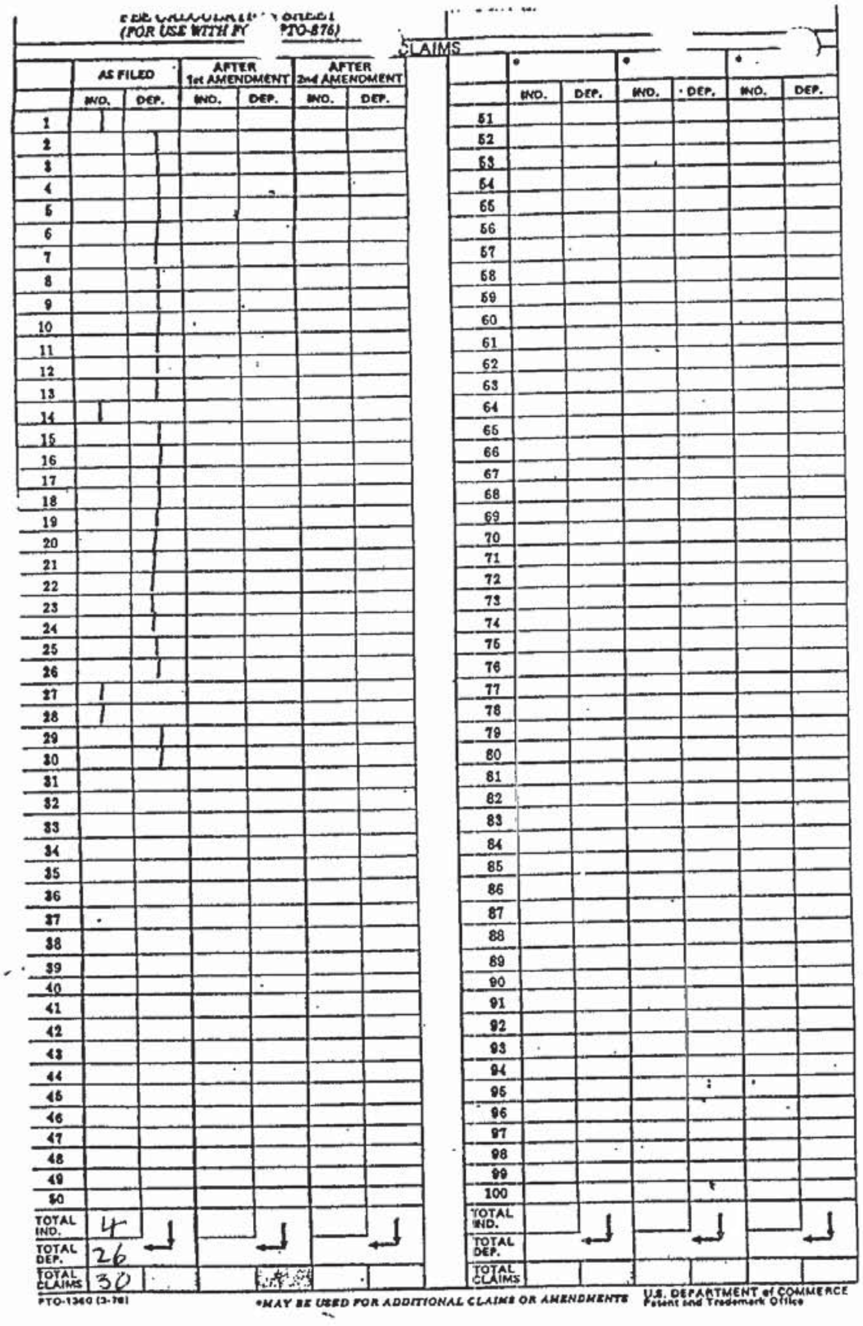
	(Column 1)		(Column 2)	(Column 3)					
▨	CLAIMS REMAINING AFTER AMENDMENT	▨	HIGHEST NUMBER PREVIOUSLY PAID FOR	PRESENT EXTRA	RATE	ADDI-TIONAL FEE		RATE	ADDI-TIONAL FEE
Total	*	Minus	**	=	X$ 9=		OR	X$18=	
Independent	*	Minus	***	=	X39=		OR	X78=	
FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM					+130=		OR	+260=	
					TOTAL ADDIT. FEE		OR	TOTAL ADDIT. FEE	

* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.
** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20."
***If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3."
The "Highest Number Previously Paid For" (Total or Independent) is the highest number found in the appropriate box in column 1.

FORM PTO-875
(Rev 12/99)

Patent and Trademark Office, U S. DEPARTMENT OF COMMERCE

*U S GPO: 2000-463-433/29044



FEE CALCULATION SHEET
(FOR USE WITH FORM PTO-875)

CLAIMS

	AS FILED		AFTER 1st AMENDMENT		AFTER 2nd AMENDMENT			*		*		*	
	IND.	DEP.	IND.	DEP.	IND.	DEP.		IND.	DEP.	IND.	DEP.	IND.	DEP.
1	1						51						
2		1					52						
3		1					53						
4		1					54						
5		1					55						
6		1					56						
7		1					57						
8		1					58						
9		1					59						
10		1					60						
11		1					61						
12		1					62						
13		1					63						
14	1						64						
15		1					65						
16		1					66						
17		1					67						
18		1					68						
19		1					69						
20		1					70						
21		1					71						
22		1					72						
23		1					73						
24		1					74						
25		1					75						
26		1					76						
27	1						77						
28	1						78						
29		1					79						
30		1					80						
31							81						
32							82						
33							83						
34							84						
35							85						
36							86						
37							87						
38							88						
39							89						
40							90						
41							91						
42							92						
43							93						
44							94						
45							95						
46							96						
47							97						
48							98						
49							99						
50							100						
TOTAL IND.	4						TOTAL IND.						
TOTAL DEP.	26						TOTAL DEP.						
TOTAL CLAIMS	30						TOTAL CLAIMS						

PTO-1360 (3-78)

*MAY BE USED FOR ADDITIONAL CLAIMS OR AMENDMENTS

U.S. DEPARTMENT of COMMERCE
Patent and Trademark Office

MPI Family Report (Family Bibliographic and Legal Status)

In the MPI Family report, all publication stages are collapsed into a single record, based on identical application data. The bibliographic information displayed in the collapsed record is taken from the latest publication.

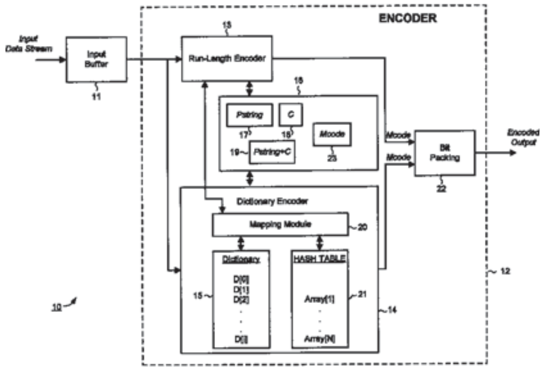
Report Created Date: 2009-11-09

Name of Report:

Number of Families: 1

Comments:

Table of Contents



Family1

1 records in the family.

US6597812B1 20030722

(ENG) System and method for lossless data compression and decompression

Assignee: REALTIME DATA LLC US

Inventor(s): FALLON JAMES J US ; BO STEVEN L US

Application No: US 57922100 A

Filing Date: 20000526

Issue/Publication Date: 20030722

Abstract: (ENG) Systems and methods for providing lossless data compression and decompression are disclosed which exploit various characteristics of run-length encoding, parametric dictionary encoding, and bit packing to comprise an encoding/decoding process having an efficiency that is suitable for use in real-time lossless data compression and decompression applications. In one aspect, a method for compressing input data comprising a plurality of data blocks comprises the steps of: detecting if the input data comprises a run-length sequence of data blocks; outputting an encoded run-length sequence, if a run-length sequence of data blocks is detected; maintaining a dictionary comprising a plurality of code words, wherein each code word in the dictionary is associated with a unique data block string; building a data block string from at least one data block in the input data that is not part of a run-length sequence; searching for a code word in the dictionary having a unique data block string associated therewith that matches the built data block string; and outputting the code word representing the built data block string.

Priority Data: US 13656199 19990528 P; US 57922100 20000526 A;

Related Application(s): 60/136561 19990528 00

IPC (International Class): G06K00936

ECLA (European Class): H03M00730Z2; H03M00746

US Class: 382232; 382245; 341051

Agent(s): F. Chau & Associates, LLP; DeRosa, Esq. Frank V. 0

Examiner Primary: Wu, Jingge

Assignments Reported to USPTO:

Reel/Frame: 11039/0865 **Date Signed:** 20000803 **Date Recorded:** 20000808
Assignee: REALTIME DATA, LLC 206 EAST 63RD STREET NEW YORK NEW YORK 10021

Assignor: BO, STEVEN L.; FALLON, JAMES J.

Corres. Addr: F. CHAU & ASSOCIATES, LLP FRANK V. DEROSA, ESQ. 1900 HEMPSTEAD TURNPIKE, SUITE 501 EAST MEADOW, NEW YORK 11554
Brief: ASSIGNMENT OF ASSIGNORSINTEREST (SEE DOCUMENT FOR DETAILS).

Legal Status:

Date	+/-	Code	Description
20000808	( )	AS	ASSIGNMENT New owner name: REALTIME DATA, LLC 206

			EAST 63RD STREET NEW YORK N; : ASSIGNMENT OF ASSIGNORS INTEREST;ASSIGNORS:FALLON, JAMES J.;BO, STEVEN L.;REEL/FRAME:011039/0865; Effective date: 20000803;
20000808	()	AS	New owner name: REALTIME DATA, LLC, NEW YORK; : ASSIGNMENT OF ASSIGNORS INTEREST;ASSIGNORS:FALLON, JAMES J.;BO, STEVEN L.;REEL/FRAME:011039/0865; Effective date: 20000803;
20000808	()	AS	New owner name: REALTIME DATA, LLC 206 EAST 63RD STREET NEW YORK N; : ASSIGNMENT OF ASSIGNORS INTEREST;ASSIGNORS:FALLON, JAMES J.;BO, STEVEN L.;REEL/FRAME:011039/0865; Effective date: 20000803;

USPTO Maintenance Report			11/09/2009 12:43 PM		
Patent Number:	6597812		Application Number:	09579221	
Issue Date:	07/22/2003		Filing Date:	05/26/2000	
Title:	SYSTEM AND METHOD FOR LOSSLESS DATA COMPRESSION AND DECOMPRESSION				
Status:	8th year fee window opens: 07/22/2010			Entity:	Small
Window Opens:	07/22/2010	Surcharge Date:	01/25/2011	Expiration:	N/A
Fee Amt Due:	Window not open	Surchg Amt Due:	Window not open	Total Amt Due:	Window not open
Fee Code:	2552	MAINTENANCE FEE DUE AT 7.5 YEARS			
Surcharge Fee Code:					
Most recent events (up to 7):	01/22/2007	Payment of Maintenance Fee, 4th Yr, Small Entity. --- End of Maintenance History ---			
Address for fee purposes:	ROPES & GRAY LLP PATENT DOCKETING 39/361 1211 AVENUE OF THE AMERICAS NEW YORK, NY 100368704				