

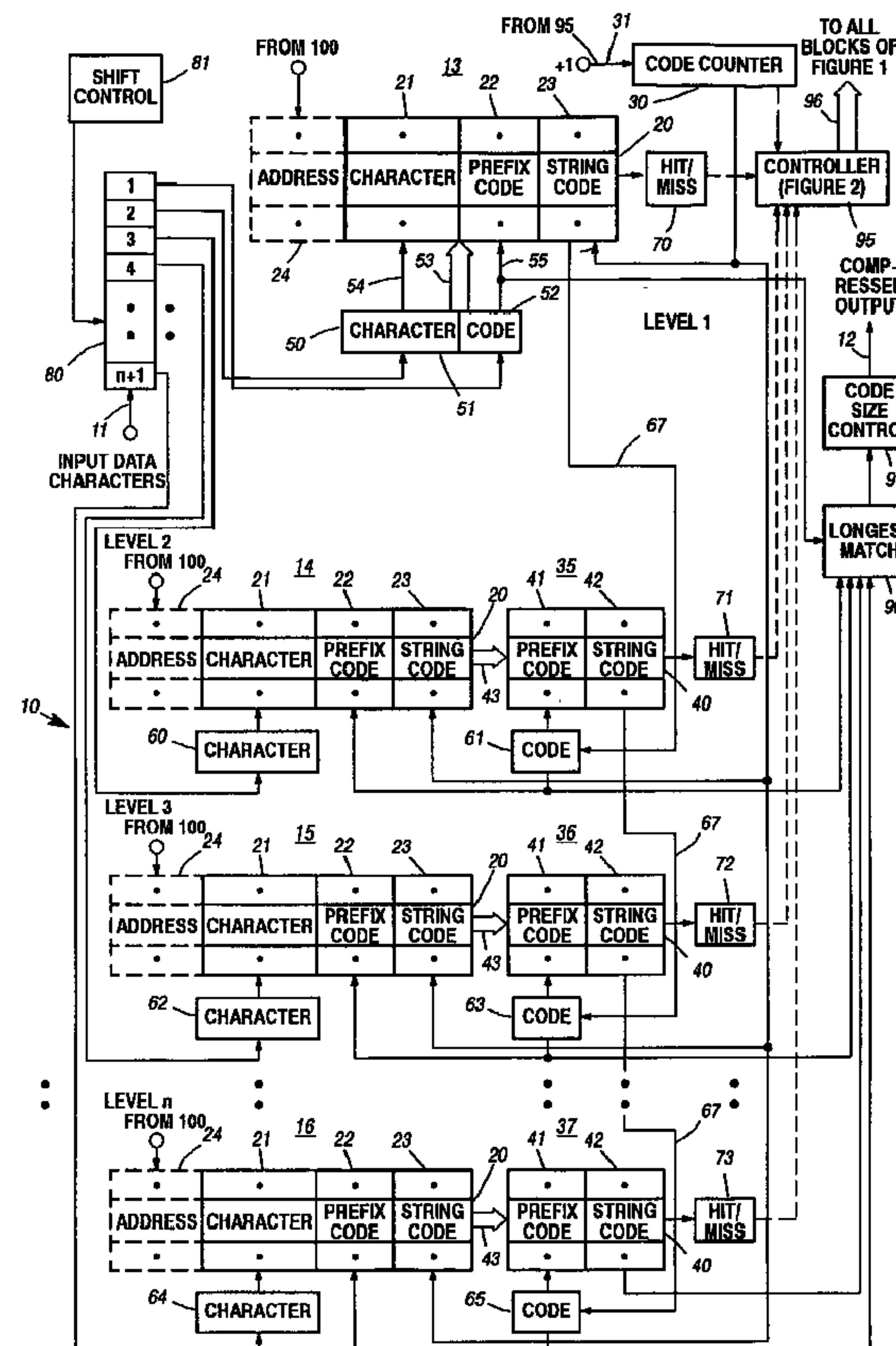


(86) Date de dépôt PCT/PCT Filing Date: 2002/06/20
(87) Date publication PCT/PCT Publication Date: 2003/03/20
(45) Date de délivrance/Issue Date: 2010/03/09
(85) Entrée phase nationale/National Entry: 2004/03/12
(86) N° demande PCT/PCT Application No.: US 2002/019710
(87) N° publication PCT/PCT Publication No.: 2003/023972
(30) Priorité/Priority: 2001/09/13 (US09/951,263)

(51) Cl.Int./Int.Cl. *H03M 7/30* (2006.01)
(72) Inventeur/Inventor:
COOPER, ALBERT B., US
(73) Propriétaire/Owner:
UNISYS CORPORATION, US
(74) Agent: R. WILLIAM WRAY & ASSOCIATES

(54) Titre : PROCÉDE ET DISPOSITIF DE COMPRESSION DE DONNÉES FAISANT APPEL A DES SOUS-DICTIONNAIRES EN CASCADE

(54) Title: DATA COMPRESSION METHOD AND APPARATUS UTILIZING CASCADED SUBDICTIONARIES



(57) Abrégé/Abstract:

The data compressor utilizes a plurality of subdictionaries. The subdictionary at the first level stores two character strings and a subdictionary at a subsequent level stores strings that are one character longer than the strings stored in the subdictionary at the



(57) Abrégé(suite)/Abstract(continued):

prior level thereto. Data characters are fetched into an input buffer and applied to the respective levels. The subdictionary at a level is searched for the string comprising the string matched at the prior level extended by the fetched character applied to the level. The string code of a string matched at a level is cascaded to the next level. The longest match with the fetched characters is determined by one of the fetched characters resulting in a mismatch at one of the levels. The string code associated with the longest match is output. An extended string comprising the longest match extended by the fetched character that resulted in the mismatch is inserted into the subdictionary at the mismatching level.

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau(43) International Publication Date
20 March 2003 (20.03.2003)

PCT

(10) International Publication Number
WO 03/023972 A3(51) International Patent Classification⁷: H03M 7/30(81) Designated States (*national*): CA, JP.

(21) International Application Number: PCT/US02/19710

(84) Designated States (*regional*): European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR).

(22) International Filing Date: 20 June 2002 (20.06.2002)

(25) Filing Language: English

Published:

(26) Publication Language: English

— with international search report
— before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments

(30) Priority Data:
09/951,263 13 September 2001 (13.09.2001) US

(71) Applicant: UNISYS CORPORATION [US/US]; Unisys Way, MS/E8-114, Blue Bell, PA 19424-0001 (US).

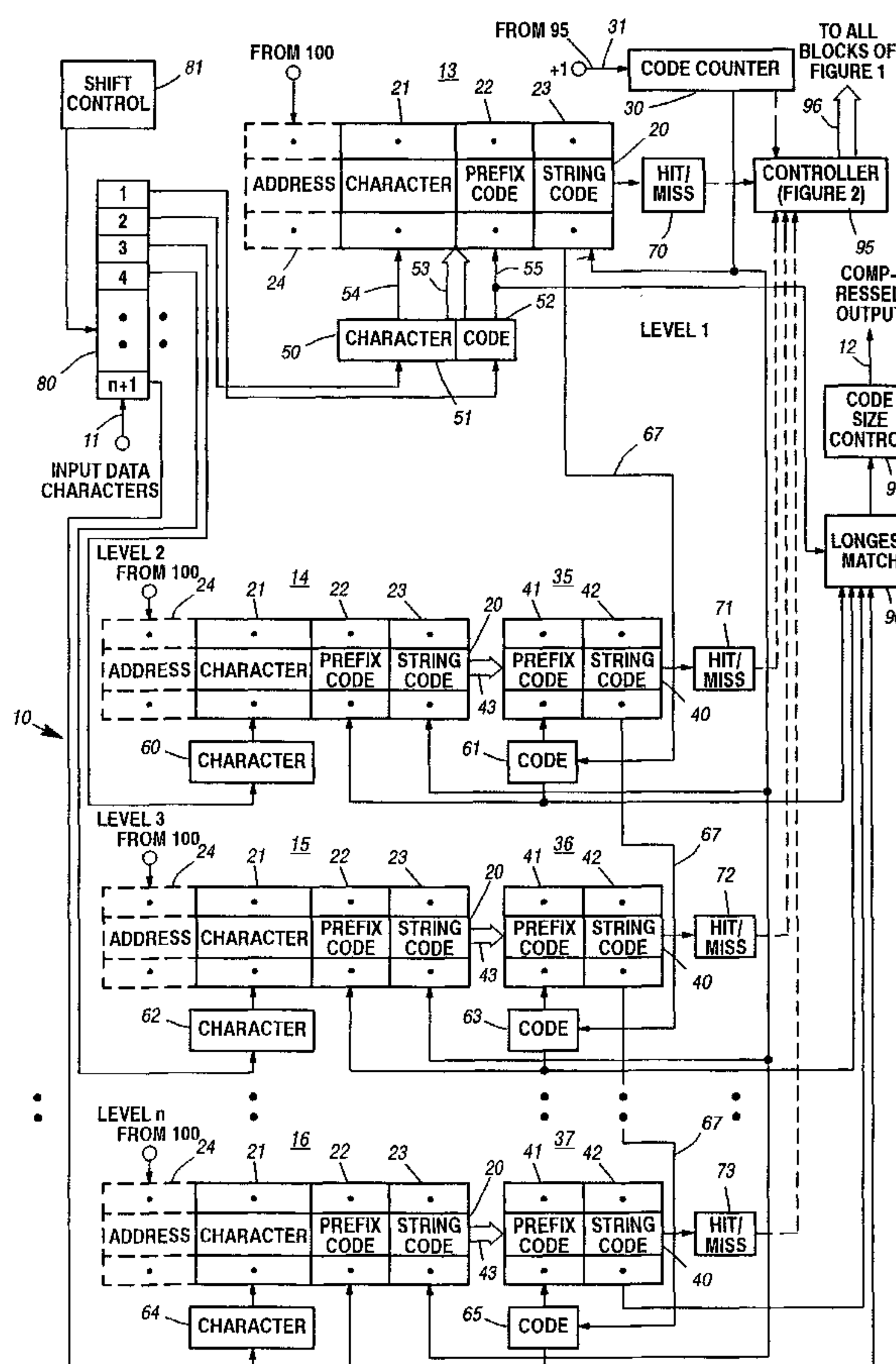
(88) Date of publication of the international search report:
31 July 2003

(72) Inventor: COOPER, Albert, B.; 20 East 74th Street, Apt. 14C, New York, NY 10021 (US).

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(74) Agents: STARR, Mark, T. et al.; Unisys Corporation, Unisys Way, MS/E8-114, Blue Bell, PA 19424-0001 (US).

(54) Title: DATA COMPRESSION METHOD AND APPARATUS UTILIZING CASCADED SUBDICTIONARIES



(57) Abstract: The data compressor utilizes a plurality of subdictionaries. The subdictionary at the first level stores two character strings and a subdictionary at a subsequent level stores strings that are one character longer than the strings stored in the subdictionary at the prior level thereto. Data characters are fetched into an input buffer and applied to the respective levels. The subdictionary at a level is searched for the string comprising the string matched at the prior level extended by the fetched character applied to the level. The string code of a string matched at a level is cascaded to the next level. The longest match with the fetched characters is determined by one of the fetched characters resulting in a mismatch at one of the levels. The string code associated with the longest match is output. An extended string comprising the longest match extended by the fetched character that resulted in the mismatch is inserted into the subdictionary at the mismatching level.

WO 03/023972 A3

DATA COMPRESSION METHOD AND APPARATUS
UTILIZING CASCADED SUBDICTIONARIES

BACKGROUND OF THE INVENTION

1 1. Field of the Invention

 The invention relates to LZ dictionary based data compression systems particularly with respect to the LZW compression methodology. More particularly, the invention relates to a parallel architecture for storing and accessing data character strings in the compressor.

2. Description of the Prior Art

10 Professors Abraham Lempel and Jacob Ziv provided the theoretical basis for LZ data compression and decompression systems that are in present day widespread usage. Two of their seminal papers appear in the IEEE Transactions on Information Theory, IT-23-3, May 1977, pp. 337-343 and in the IEEE Transactions on Information Theory, IT-24-5, September 1978, pp. 530-536. A ubiquitously used data compression and decompression system known as LZW, adopted as the standard for V.42 bis modem compression and decompression, is described in U.S. Patent 4,558,302 by Welch, issued December 10, 1985. LZW has been adopted as the compression and decompression standard used in the GIF image communication protocol and is utilized in the TIFF image communication protocol. GIF is a development of CompuServe Incorporated and the name GIF is a Service Mark thereof. A reference to the GIF specification is found in GRAPHICS INTERCHANGE FORMAT, Version 89a, 31 July, 1990. TIFF is a development of Aldus Corporation and the name TIFF is a Trademark thereof. Reference to the TIFF specification is found in TIFF, Revision 6.0, Final - June 3, 1992.

 Further examples of LZ dictionary based compression and decompression systems are described in the following U.S. patents: patent 4,464,650 by Eastman

- 2 -

1 et al., issued August 7, 1984; patent 4,814,746 by Miller
et al., issued March 21, 1989; patent 4,876,541 by Storer,
issued October 24, 1989; patent 5,153,591 by Clark, issued
October 6, 1992; patent 5,373,290 by Lempel et al., issued
5 December 13, 1994; patent 5,838,264 by Cooper, issued
November 17, 1998; patent 5,861,827 by Welch et al.,
issued January 19, 1999; and patent 6,188,333 by Cooper,
issued February 13, 2001.

In dictionary based LZ compression and
10 decompression systems, the compressor and decompressor
dictionaries may be initialized with all of the single
character strings of the character alphabet. In some
implementations, the single character strings are
considered as recognized although not explicitly stored.
15 In such systems the value of the single character may
be utilized as its code and the first available code
utilized for multiple character strings would have a
value greater than the single character values. In this
way the decompressor can distinguish between a single
20 character string and a multiple character string and
recover the characters thereof. For example, in the
ASCII environment, the alphabet has an 8 bit character
size supporting an alphabet of 256 characters. Thus,
the characters have values of 0-255. The first available
25 multiple character string code can, for example, be 258
where the codes 256 and 257 are utilized as control codes
as is well known.

In the prior art dictionary based LZ compression
systems, data character strings are stored and accessed
30 in the compressor dictionary utilizing well known search
tree architectures and protocols. Typically, the search
for the longest matching string stored in the dictionary
is an iterative process where sequentially matched strings
in the dictionary are extended by sequentially fetched
35 input characters, respectively, until the longest matching
string is determined. At each iteration, the dictionary
is accessed to determine if the new string extension

- 3 -

1 is a previously stored dictionary entry. Potentially,
at each iteration, access to all of the strings stored
in the dictionary may be effected to determine the
required information. For example, in systems implemented
5 utilizing an associative memory dictionary, such as in
said patents 5,373,290 and 5,838,264, it may be necessary,
at an iteration, to access all dictionary locations to
determine that an extended string is not stored therein.

The conventional iterative protocols, therefore,
10 tend to be time consuming. Although the known dictionary
architectures and protocols provide efficient data
compression systems, it is a continuing objective in
the art to improve compressor performance.

15

SUMMARY OF THE INVENTION

The present invention provides new string storage
and access architecture and protocols which, it is
believed, will improve the speed of dictionary based
LZ type data compressors.

20

In the present invention, a plurality of
subdictionaries are arranged in levels for storing strings
of data characters encountered in the input stream.
The strings stored in a subdictionary have the same number
of characters with respect to each other and the strings
25 stored in the subdictionary of a level have one character
more than the strings stored in the subdictionary at
the level prior thereto. A plurality of data characters
are fetched from the input and applied to the levels,
respectively. The fetched characters are searched by
30 comparing the fetched characters to the stored strings
to determine the longest match therewith. The longest
match is determined by one of the fetched characters
resulting in a mismatch at one of the levels. The string
code associated with the longest match is output so as
35 to provide the output stream of compressed codes
corresponding to the input stream of data characters.
An extended string comprising the longest match extended

- 4 -

1 by the fetched character that resulted in the mismatch
is inserted into the subdictionary at the mismatching
level.

In particular, the fetched characters are searched
5 by searching the strings stored at a particular level
for a string comprising a string matched at the level
prior thereto extended by the fetched character applied
to the particular level. The string code of a string
matched at the particular level is applied to the next
10 level in cascade fashion.

A best mode embodiment of the invention includes
an auxiliary table at each of the levels except for the
first level. While the subdictionary at the first level
is being searched for a 2 character string match, each
15 of the subdictionaries at the remaining levels is screened
for stored strings ending in the fetched input character
applied to the level. The screened strings are
transferred to the corresponding auxiliary table. At
a level, the cascaded string code from the previous level
20 is searched in the auxiliary table.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a schematic block diagram of a data
compressor for compressing data in accordance with a
25 best mode embodiment of the present invention.

Figure 2 is a schematic block diagram illustrating
details of the controller of the embodiment of Figure 1.

Figure 3 is a chart summarizing the functions
and illustrating the operational flow of the Figures 1
30 and 2 embodiment.

Figure 4 is a schematic block diagram of a data
compressor for compressing data in accordance with an
alternative best made embodiment of the invention.

Figure 5 is a schematic block diagram illustrating
35 details of the controller of the embodiment of Figure 4.

Figure 6 is a chart, similar to Figure 3,
summarizing the functions and illustrating the operational

- 5 -

1 flow of the Figures 4 and 5 embodiment.

Figure 7 is a schematic block diagram of an addition to the embodiments of Figures 1 and 2 and Figures 4 and 5 for providing enhanced processing of data character runs.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

The best mode embodiments described below are predicated, generally, on the LZW methodology. The
10 embodiments conveniently utilize an implementation feature similar to that described above where the single character strings are considered as recognized by the compressor although not explicitly initialized therein. It is appreciated that the invention may be embodied with
15 alternative implementation features such as a non-initialized arrangement as well as a configuration where the compressor is explicitly initialized with all of the single character strings.

Referring to Figure 1, a data compressor 10 is
20 illustrated that compresses a stream of input data characters applied at an input 11 into a stream of corresponding compressed codes at an output 12. The compressor 10 is arranged in levels denoted as levels 1 to n which include respective subdictionaries 13-16.
25 Each of the subdictionaries 13-16 is comprised of a plurality of subdictionary locations 20 for storing strings of data characters encountered at the input 11. Each subdictionary location 20 is comprised of a character field 21, a prefix code field 22 and a string code field
30 23. The subdictionary locations 20 are accessed by respective addresses 24.

A data character string is comprised of a prefix string of one or more characters of the alphabet over which compression is being performed followed by an
35 extension character. A string is stored in a subdictionary by storing the code of the string prefix in the prefix code field 22 and the extension character

- 6 -

1 of the string in the character field 21. The code of
the string is stored in the string code field 23. The
string code assigned to the stored string is provided
from a code counter 30 which is incremented by 1 via
5 an input 31. The addresses 24 are utilized to access
the locations 20 for string storage and read-out. By
the operations of the compressor 10 to be described,
it will be appreciated that the subdictionaries 13-16
will store exclusively 2, 3, 4,... and n+1 character
10 strings, respectively.

Included at levels 2 to n are respective auxiliary
tables 35-37. Each of the tables 35-37 is comprised
of a plurality of table locations 40 for storing the
prefix code and string code of strings selectively
15 transferred from the associated subdictionary 14-16.
Accordingly, each of the locations 40 includes a prefix
code field 41 and a string code field 42. It is
appreciated that the plurality of prefix codes 41 in
an auxiliary table form a prefix code list utilized for
20 table look-up in a manner to be described. The transfer
of the prefix codes and string codes from each of the
subdictionaries 14-16 to the associated auxiliary table
35-37 is denoted by an arrow 43. In a manner to be
further described, the strings transferred from a
25 subdictionary to the associated auxiliary table during
a search for a longest match will be those strings stored
in the subdictionary that have a particular extension
character.

At level 1, a register 50 is included having
30 a character field 51 and a code field 52. The register
50 is arranged with respect to the subdictionary 13 for
associatively searching for a string stored in the
subdictionary 13 having a prefix code in the field 22
matching the code in the register field 52 and an
35 extension character in the field 21 matching the character
in the register field 51. The associative search is
denoted by an arrow 53.

- 7 -

1 An extended string is stored in the subdictionary
13 by writing the code in the register code field 52
and the character in the register character field 51
into the prefix code field 22 and character field 21,
5 respectively, at a location in the subdictionary 13
selected by the address 24. The extended string storage
function is denoted by arrows 54 and 55. Operations
of the subdictionary 13 as an associative memory are
described in detail in said patent 5,838,264.

10 At level 2, a character register 60 is associated
with the character field 21 of the subdictionary 14 and
a code register 61 is associated with the prefix code
field 41 of the auxiliary table 35. The character
register 60 is utilized to identify the strings stored
15 in the subdictionary 14 having as the string extension
character, the character stored in the register 60.
The prefix code and string code of each identified string
is transferred to the auxiliary table 35.

 The code register 61 is utilized to determine
20 if the code in the register 61 is contained in the prefix
code list 41 of the auxiliary table 35. The code in
the register 61 is utilized to perform a look-up in the
auxiliary table 35 to locate a matching prefix code in
the prefix code list 41 so as to provide the corresponding
25 string code. Additionally, in the manner described above
with respect to level 1, the character in the character
register 60 and the code in the code register 61 are
utilized to store an extended string in the subdictionary
14.

30 Levels 3 to n include character and code registers
62-65 to perform the functions described with respect
to the character register 60 and code register 61 of
level 2.

 For reasons to be described below, the string
35 code from the field 23 of the subdictionary 13 is cascaded
into the code register 61. In a similar manner, the
string code from the field 42 of the auxiliary table

- 8 -

1 35 is cascaded into the code register 63. The cascading
continues from the string code field 42 of auxiliary
table 36 of level 3 through the intervening levels to
the code register 65 of level n. The cascading is
5 implemented by paths 67.

A hit/miss circuit 70 is coupled to the
subdictionary 13 of level 1 to provide a "hit" signal
if the string in the register 50 is stored in the
subdictionary 13 or a "miss" signal if the string in
10 the register 50 is not stored in the subdictionary 13.
While the subdictionary 13 is being searched pending
the hit or the miss, neither signal is provided. In
this manner the hit/miss circuit 70 provides a definite
indication when the determination is made that the string
15 is or is not stored. The search may be performed
associatively, sequencing through the addresses 24 of
all of the strings previously stored in the subdictionary
13.

At level 2, a similar hit/miss circuit 71 is
20 coupled to the auxiliary table 35. In the manner
described above with respect to the hit/miss circuit
70, the hit/miss circuit 71 provides the "hit" signal
when the code in the code register 61 matches a code
in the prefix code list 41 of the auxiliary table 35.
25 The "miss" signal is provided by the circuit 71 when
the determination is made that the code in the register
61 is not in the prefix code list 41. The levels 3 to
n include similar respective hit/miss circuits, such
as the circuits 72 and 73 where the descriptions given
30 with respect to the hit/miss circuit 71 of level 2 also
apply thereto.

The compressor 10 includes an input character
buffer 80 for holding n+1 consecutive characters from
the input 11. Accordingly, the stages of the buffer
35 80 are denoted as 1 to n+1. The first and second stages
of the buffer 80 are coupled, respectively, to the code
field 52 and the character field 51 of the level 1

- 9 -

1 register 50. The stages 3 to $n+1$ of the buffer 80 are
coupled to the character registers 60, 62 and 64 of levels
2 to n , respectively. A shift control circuit 81 controls
shifting of the buffer 80 so that new input characters
5 are fetched into the $n+1$ stage while processed characters
are discarded from the first stage.

By operations to be described, the compressor
10 determines the longest match of the $n+1$ characters
in the buffer 80 with the strings stored in the
subdictionaries 13-16. The appropriate code of the
determined longest match is transferred to a longest
match register 90. The code of the longest match is
selectively provided from either the code field 52 of
the register 50 of level 1, one of the code registers
15 61, 63, 65 of levels 2 to n , or the string code field
of a selected location in the auxiliary table 37 of level
 n . A code size control circuit 91 is utilized, in a
well known manner, to control the number of bits used
for transmitting the compressed code from the output
20 12.

The compressor 10 includes a controller 95 for
controlling the functions to be performed. The controller
95 receives control inputs (denoted by dashed lines)
from the code counter 30 and the hit/miss circuits 70-73
25 and provides control signals to all of the blocks of
Figure 1 as depicted by an arrow 96. Further details
of the controller 95 are illustrated in Figure 2.

Referring to Figure 2, with continued reference
to Figure 1, the controller 95 includes a plurality of
30 subdictionary address counters 100 corresponding,
respectively, to the subdictionaries 13-16. The
subdictionary address counter corresponding to a
subdictionary is utilized by the controller 95 to access
locations in the subdictionary via the addresses 24.
35 When an extended string is stored in a subdictionary,
the string is stored at the next available address
indicated by the corresponding subdictionary address

- 10 -

1 counter 100 and the address counter 100 is thereafter
advanced to the next count. For example, the
subdictionary address counts may be advanced sequentially.
Thus, the subsidiary address counter 100 corresponding
5 to the subsidiary provides an indication utilized
by the controller 95 to determine which addresses in
the subsidiary contain stored strings.

The controller 95 further includes a plurality
of auxiliary table counters 101 corresponding,
10 respectively, to the auxiliary tables 35-37. The
controller 95 utilizes the auxiliary table counter 101
corresponding to an auxiliary table for maintaining a
count of the number of strings stored in the auxiliary
table.

15 The controller 95 further includes 2-character
string search logic 102, subsidiary screening logic
103, auxiliary table search logic 104, longest match
determining logic 105 and subsidiary extended string
storage logic 106.

20 The 2-character string search logic 102 is
utilized by the controller 95 to determine if the string
represented by the character and prefix code held in
the register 50 is stored in the subsidiary 13. The
subsidiary 13 is searched associatively utilizing
25 the corresponding subsidiary address counter 100.
Since this corresponding subsidiary address counter
100 provides an indication of the address in the
subsidiary 13 at which the last string was stored,
the logic 102 only examines the locations of the
30 subsidiary 13 at which strings are stored to determine
if a match exists.

The subsidiary screening logic 103 is utilized
by the controller 95 to simultaneously screen from the
respective subsidiaries 14-16 of levels 2 to n, the
35 strings stored therein that end in the characters held
in the corresponding character registers 60, 62 and 64.
The prefix code and string code of the screened strings

- 11 -

1 are transferred to the corresponding auxiliary tables
35-37. The subdictionary screening logic 103 utilizes
the corresponding auxiliary table counters 100 to maintain
respective counts of the number of strings transferred
5 and stored in the corresponding auxiliary tables. Thus,
when the subdictionary screening logic 103 transfers
a screened string from the subdictionary to the
corresponding auxiliary table, the corresponding auxiliary
table counter 101 is incremented.

10 The auxiliary table search logic 104 is utilized
by the controller 95 to determine if the codes held in
the code registers 61, 63 and 65 of levels 2 to n are
contained in the respective prefix code lists 41 of the
corresponding auxiliary tables 35-37. The auxiliary
15 table search logic 104 utilizes the auxiliary table
counters 101 corresponding to the auxiliary tables 35-37
so that only locations in the auxiliary tables 35-37
that are storing strings are examined.

The longest match determining logic 105 examines
20 the status of the hit/miss circuits 70-73. A mismatching
level is determined when a hit/miss circuit at a level
provides the "miss" signal while all of the hit/miss
circuits at the prior levels are providing the "hit"
signal. The level at which the hit/miss circuit provides
25 the "miss" signal is the mismatching level. The code
held in the code register 61, 63 or 65 at the mismatching
level is the code of the longest match and this code
is transferred to the longest match register 90. The
character held in the character register 60, 62 or 64
30 at the mismatching level is the mismatching character
that "broke" the match.

If the hit/miss circuit 70 indicates the "miss"
signal, the code of the longest match is taken from the
code field 52 of the register 50 of level 1.

35 If all of the hit/miss circuits 70-73 provide
the "hit" signal, the code of the longest match is taken
from the string code field 42 of the auxiliary table

- 12 -

1 37 at the location containing the prefix code matching
the code held in the register 65.

The subdictionary extended string storage logic
106 is utilized to store the appropriate extended string
5 in the appropriate subdictionary. At the mismatching
level 2 to n, the character held in the associated
character register 60, 62 or 64 and the code held in
the corresponding code register 61, 63 or 65 are stored
in the character field 21 and prefix code field 22,
10 respectively, of the associated subdictionary 14-16.
If level 1 is the mismatching level, the character and
prefix code of the extended string are provided to the
subdictionary 13 by the register 50.

The string code for the stored extended string
15 is provided to the string code field 23 of the associated
subdictionary from the code counter 30. The code counter
30 is advanced to the next available string code. The
location in the associated subdictionary in which to
store the extended string is determined by the
20 corresponding subdictionary address counter 100. After
storing the extended string in the appropriate
subdictionary, the associated subdictionary address
counter 100 is advanced to the next address.

The controller 95 further includes a code size
25 register 110 for determining the code size utilized by
the code size control 91 for transmitting the compressed
codes on the output 12. The code size in the code size
register 110 is set in accordance with the count in the
code counter 30 in a well known manner. The code size
30 is set to an initial value and incremented at
predetermined codes of the code counter 30 until a maximum
code size is attained. For example, in an ASCII variable
length code implementation, the code size may begin with
9 bits and sequentially increase to 10, 11 and 12 bits
35 at codes 512, 1024 and 2048, respectively. It is
appreciated that a fixed code size may also be utilized
with appropriate modifications to the embodiment.

- 13 -

1 The controller 95 also includes control circuit
111 for controlling the detailed operations to be executed
by the compressor 10. The control circuit 111 is
considered as containing appropriate circuitry, such
5 as state machines, or appropriate software, to control
execution of the operations.

Referring to Figure 3, with continued reference
to Figures 1 and 2, a summary and flow diagram of the
operations performed by the compressor 10 is illustrated.
10 The operations are described in terms of an initialization
phase and repetitive operational phases 1, 2 and 3.
The initialization phase is denoted by a block 120, phase
1 by a block 121, phase 2 by blocks 122 and 123 and phase
3 by blocks 124 and 125.

15 At initialization phase block 120, the
subdictionary address counters 100 and the auxiliary
table counters 101 are cleared as are the code registers
61, 63 and 65 of levels 2 to n. Additionally, at the
block 120, the code counter 30 and the code size register
20 110 are set to initial values. For example, in an ASCII
environment, the code counter 30 may be set to a first
available code of 258 and the code size register 110
may be set to an initial code size of 9 bits.

Further at the block 120, the first n+1 characters
25 from the input 11 are entered into the input buffer 80
with the first character of the input stream residing
in the first stage of the buffer. Although serial data
entry is illustrated, parallel data entry into the n+1
stages of the buffer 80 may alternatively be utilized.
30 It is appreciated that the clearing, setting and data
entering operations of the block 120 may be simultaneously
performed.

During each iteration of the compressor 10, the
operations of phases 1-3 determine the longest string
35 in the buffer 80 that matches a string stored in the
subdictionaries 13-16 of levels 1 to n. The code of
the longest match is provided at the compressed output

- 14 -

1 12 and the input buffer 80 is shifted, in preparation
for the next longest match search iteration, so that
the mismatching character resides in the first stage
thereof. New characters from the input 11 are entered
5 at stage n+1 to replace the already processed characters.

At the beginning of each iteration of the
compressor 10, processing enters the phase 1 block 121.
At the block 121, the subdictionary 13 is searched,
utilizing the logic 102, for the 2 character string
10 represented by the prefix code and character held in
the register 50. The search may be performed
associatively, as described above, by examining the
locations of the subdictionary 13 that are storing
previously entered strings. The associated subdictionary
15 address counter 100 is utilized by the controller 95
to determine which addresses 24 of the subdictionary
13 to examine so as to only access locations at which
strings are stored. If the string held in the register
50 is stored at an examined location, the "hit" signal
20 is provided by the hit/miss circuit 70. If the string
is not stored in the subdictionary 13, the "miss" signal
is indicated. It is appreciated that the examined
locations of the subdictionary 13 accessed by the
addresses 24 are searched for the 2 character string
25 comprising the character in the first stage of the input
buffer 80 extended by the character in the second stage
of the input buffer 80. For all of the longest match
search iterations but the initial iteration, the first
stage of the buffer 80 is holding the mismatching
30 character from the previous longest match iteration and
the second stage of the buffer 80 is holding the next
following character in the input data stream.

Further at the phase 1 block 121, and
simultaneously with the search for the 2 character string
35 at level 1, the subdictionaries 14-16 at the levels 2
to n are simultaneously screened, utilizing the logic
103, for the strings in each subdictionary that end with

- 15 -

1 the character in the corresponding character register
60, 62 and 64. As previously described, the character
registers 60, 62 and 64 are holding the respective
characters in the third to $n+1$ stages of the input buffer
5 80. To perform the screening, the controller 95 examines
only the locations in each subdictionary 14-16 storing
previously entered strings. The subdictionary address
counters 100 corresponding to the subdictionaries 14-16
are utilized by the controller 95 to access only addresses
10 24 at which strings are stored. In order to screen the
appropriate strings, the characters in the fields 21
are compared with the character values held in the
corresponding character register 60, 62 and 64. As the
appropriate strings are identified, the prefix codes
15 and string codes thereof are transferred from the prefix
code fields 22 and string code fields 23 of the
subdictionaries 14-16 to the prefix code fields 41 and
the string code fields 42 of the corresponding auxiliary
tables 35-37. As each string transfer is effected, the
20 corresponding auxiliary table counter 101 is incremented
to maintain a count of the number of strings transferred
to the corresponding auxiliary table.

It is appreciated that the strings transferred
to the auxiliary tables are potential candidate 3, 4,...,
25 $n+1$ character strings for matching with the $n+1$ character
string held in the input buffer 80 in the search for
the longest match.

If the "miss" signal from the hit/miss circuit
70 occurs at the block 121, no 2 character string match
30 is found at level 1 and processing enters the phase 2a
block 122. At the block 122, the code of the longest
match is transferred from the code field 52 of the
register 50 to the longest match register 90. The code
of the longest match is provided at the output 12 in
35 the appropriate number of bits controlled by the code
size control circuit 91. It is appreciated that the
longest match with the string held in the input buffer 80

- 16 -

1 is the character held in the first stage thereof. The character held in the second stage of the buffer 80 is the character that caused the mismatch.

The appropriate extended string is stored in
5 the subdictionary 13, utilizing the logic 106, by storing the code and character in the register 50 into the prefix code field 22 and character field 21, respectively, at the next available location in the subdictionary 13. The appropriate string code is assigned to the stored
10 string by entering the extant code in the code counter 30 into the string code field 23 at the location storing the string.

The location in the subdictionary 13 at which to store the extended string is selected by accessing
15 the addresses 24 utilizing the corresponding subdictionary address counter 100. After the extended string is stored, the corresponding subdictionary address counter 100 is incremented to indicate the next available location in the subdictionary 13.

20 Since, at the block 122, only the character in the first stage of the buffer 80 has been processed, the input buffer 80 is shifted once discarding the processed character from the first stage and entering one new character from the input 11 at stage n+1. The
25 code counter 30 is advanced to the next available code utilizing the control input 31. The code size in the code size register 110 is advanced if required.

At the phase 1 block 121, as the 2 character string search was being performed at level 1, candidate
30 matching extended strings were being transferred to the auxiliary tables 35-37. Since, at the block 122, no 2 character string match was found at level 1, the auxiliary table counters 101 are cleared.

It is appreciated that most of the operations
35 described with respect to the phase 2a block 122 are performed simultaneously by the controller 95. The outputting of the code of the longest match, the entering

- 17 -

1 of the extended string into the subdictionary 13, the
assigning of the string code from the code counter 30,
the shifting of the input buffer and the clearing of
the auxiliary table counters may be executed
5 simultaneously. After the functions of the phase 2a
block 122 are performed, control returns to the phase
1 block 121 to find the next longest match.

If at the phase 1 block 121, a 2 character string
match is found at level 1, as indicated by the "hit"
10 signal from the hit/miss circuit 70, and the screened
strings have been transferred to the auxiliary tables
35-37, control proceeds from the block 121 to the phase
2b block 123. At the block 123, the string codes of
strings matched at the respective levels are cascaded
15 for matching in the prefix code lists of the auxiliary
tables of the next lower levels until a mismatching level
occurs or all levels match.

Specifically, the string code from the string
code field 23 of the 2 character string match location
20 in the subdictionary 13 is cascaded to the code register
61 of level 2. If a match is found in the prefix code
list 41 of the auxiliary table 35, the corresponding
string code is cascaded to the code register 63 of level
3. Similarly, if matches are found at levels 3 to n-1,
25 the corresponding string codes are cascaded to the next
lower levels with the string code of the match from the
auxiliary table of level n-1 cascaded to the code register
65 of level n. If, however, a match does not occur at
a level, the cascading to lower levels is terminated.

30 A specific protocol for performing the cascading
is that, beginning with level 2, when the "hit" signal
is provided by the hit/miss circuit at a level, the string
code of the match is cascaded to the next lower level.
In the cascading process, when a hit/miss circuit at
35 a level indicates the "miss" signal, the mismatching
level is identified, terminating the cascading process.
The cascading process also terminates at level n if all

- 18 -

1 of the hit/miss circuits 70-73 indicate the "hit" signal.

It is appreciated that when the table look-up is performed in the prefix code list 41 of the auxiliary table 35-37 of a level, the count in the corresponding auxiliary table counter 101 is utilized so that only locations that are storing strings in the auxiliary table are examined. This feature contributes to the rapid determination of the longest matching string by reducing the time for a hit/miss circuit to provide the appropriate "hit" or "miss" signal.

If, at the phase 2b block 123, a mismatching level occurs, control continues with the phase 3a block 124. The mismatching level occurs when the string code cascaded from the previous level is not in the prefix code list 41 of the auxiliary table at the mismatching level. Accordingly, at the block 124, the code of the string that comprises the longest match with the string in the input buffer 80 is output as the string code applied to the mismatching level from the prior level. The code of the longest match, therefore, is provided by the code register 61, 63 or 65 of the mismatching level and transferred to the longest match register 90. The code of the longest match is provided at the compressed output 12 via the code size control 91 as previously described. The mismatching character that "broke" the match resides in the character register 60, 62 or 64 of the mismatching level. The logic 105 is utilized in determining the longest match.

The appropriate extended string is entered, utilizing the logic 106, into the subdictionary of the mismatching level by entering the code in the code register and the character in the character register at the mismatching level into the prefix code field 22 and the character field 21, respectively, of the subdictionary at the mismatching level. The string code is assigned to the extended string by entering the code from the code counter 30 into the string code field 23

- 19 -

1 of the subdictionary location at which the extended string
is stored. The location at which to store the extended
string is provided by the corresponding subdictionary
address counter 100. After storing the extended string,
5 the corresponding subdictionary address counter 100 is
incremented to the next available address 24. The code
counter 30 is advanced to the next available string code
and the code size in the code size register 110 is
advanced if required.

10 The input buffer 80 is shifted entering new
characters from the input 11 at the stage $n+1$ until the
mismatching character is in the first buffer stage.
The level at which the mismatch occurs determines the
number of shifts to be applied by the shift control 81
15 to the input buffer 80. For example, if the mismatching
level is level 3, three shifts are required to bring
the mismatching character in the fourth stage of the
buffer 80 into the first stage thereof. As part of the
processing of the phase 3a block 124, the auxiliary table
20 counters 101 and the code registers 61, 63 and 65 of
levels 2 to n are cleared.

It is appreciated that many of the functions
of the phase 3a block 124 are simultaneously performed.
The outputting of the longest match, the entering of
25 the extended string into the subdictionary, the assigning
of the string code from the code counter 30, the shifting
of the input buffer 80 and the clearing of the auxiliary
table counters 101 may be concurrently executed. After
the phase 3a functions of the block 124 are performed,
30 control returns to the phase 1 block 121 to find the
next longest match.

If, at the phase 2b block 123, all levels match,
control proceeds from the block 123 to the phase 3b block
125. At the block 125, all levels match since at each
35 of levels 2 to n , the applied string code from the
previous level is in the prefix code list 41 of the
auxiliary table of the level. When this occurs, the

- 20 -

1 n+1 character string held in the input buffer 80 is
already stored in the subdictionary 16 of level n.
Accordingly, the code of the longest match is derived
from the string code field 42 of the auxiliary table
5 37. The code of the longest match is the string code
corresponding to the prefix code in the prefix code list
41 that matches the code in the code register 65.

The code of the longest match is provided at
the compressed output 12 via the longest match register
10 90 and the code size control 91 as previously described.
The input buffer 80 is shifted entering n+1 new characters
from the input 11. The code counter 30 is advanced to
the next available code utilizing the input 31 as
previously described. The code size in the code size
15 register 110 is advanced if required. The auxiliary
table counters 101 and the code registers 61, 63 and
65 at the levels 2 to n are cleared. After the functions
of the phase 3b block 125 are executed, control returns
to the phase 1 block 121 to find the next longest match.

20 It is appreciated that operations of the phase
3b block 125 are simultaneously performed. The outputting
of the longest match, the shifting of the input buffer
80, the advancing of the code counter 30 and the clearing
of the auxiliary table counters 101 may be concurrently
25 executed.

Referring to Figures 4 and 5, with continued
reference to Figures 1 and 2, an alternative embodiment
of the invention is illustrated. Since numerous elements
of the Figures 4 and 5 embodiment are the same as elements
30 of the Figures 1 and 2 embodiment, an element of Figures
4 and 5 is given a reference numeral that is, for
correlation, 200 greater than the reference numeral of
the corresponding element in Figures 1 and 2. The
descriptions given above with respect to the elements
35 of Figures 1 and 2 apply to the corresponding elements
of Figures 4 and 5.

Basically, the Figures 4 and 5 embodiment differs

- 21 -

1 from that of Figures 1 and 2 in that the auxiliary tables
35-37 are not utilized in Figure 4. Accordingly, the
individual character and code registers 60-65 of Figure
1 are combined into registers 400-402 at levels 2 to
5 n that are each identical to the register 50 of Figure
1. For correlation, the character and code fields of
the registers 400-402 are given reference numerals that
are 200 greater than the reference numerals of the
corresponding individual character and code registers
10 60-65 of Figure 1.

Generally, in the manner described above with
respect to Figure 1, the characters in the first and
second stages of the input buffer 280 are applied,
respectively, to the code field 252 and character field
15 251 of the register 250. Similarly, the characters in
the stages 3 to n+1 of the buffer 280 are applied to
the respective character fields 260, 262 and 264 of the
registers 401-402. A string code in the field 223 of
a matching string in a subdictionary of levels 1 to n-1
20 is cascaded, via paths 267, into the code field 261,
263, 265 of the register 400-402 of the next level.

Since the auxiliary tables 35-37 are not utilized
in Figure 4, the hit/miss circuits 271-273 are coupled
directly to the respective subdictionaries 214-216.
25 The hit/miss circuits 271-273 operate in the manner
described above with respect to the hit/miss circuit
70 coupled to the subdictionary 13. It is appreciated
that the code of the longest match is provided to the
longest match register 290 from the code field 252, 261,
30 263 or 265 of the register 250, 400, 401 or 402 of the
mismatching level generally in the manner described above
with respect to the code field 52 of register 50 and
the code registers 61, 63 and 65 of Figure 1.

In Figure 5, which illustrates details of the
35 controller 295, the subdictionary string search logic
302 is utilized for searching each of the subdictionaries
213-216 for matching strings in the manner described

- 22 -

1 above with respect to the subdictionary 13 of Figure
1. Furthermore, the subdictionary extended string storage
logic 306 stores extended strings in the subdictionaries
213-216 from the respective registers 250, 400-402 in
5 the manner described above with respect to the
subdictionary 13 and register 50 of Figure 1.

It will be appreciated that the subdictionaries
213-216 will store exclusively 2, 3, 4,...and $n+1$
character strings, respectively, by the operations of
10 the compressor 210 in a manner similar to that described
above with respect to the Figures 1 and 2 embodiment.

Referring to Figure 6, with continued reference
to Figures 4 and 5, a summary and flow diagram of the
operations performed by the compressor 210 is illustrated.
15 The operations are described in terms of an initialization
phase and repetitive operational phases 1 and 2. The
initialization phase is denoted by a block 410, phase
1 by a block 411 and phase 2 by blocks 412 and 413.

At initialization phase block 410, operations
20 similar to those described above with respect to the
block 120 of Figure 3 are performed in that the
subdictionary address counters 300 are cleared and the
code counter 230 and code size register 310 are set to
initial values. Further, at the block 410, the first
25 $n+1$ characters from the input 211 are entered into the
input buffer 280 with the first character of the input
stream residing in the first stage of the buffer. It
is appreciated that the clearing, setting and data
entering operations of the block 410 may be simultaneously
30 performed.

During each iteration of the compressor 210,
the operations of phases 1 and 2 determine the longest
string in the buffer 280 that matches a string stored
in the subdictionaries 213-216 of levels 1 to n . The
35 code of the longest match is provided at the compressed
output 212 and the input buffer 280 is shifted, in
preparation for the next longest match search iteration,

- 23 -

1 so that the mismatching character resides in the first stage thereof. New characters from the input 211 are entered at stage n+1 to replace the already processed characters.

5 At the beginning of each iteration of the compressor 210, processing enters the phase 1 block 411. At the block 411, a search is performed for 2 to n+1 character string matches at levels 1 to n, respectively, using corresponding subdictionary address counters.

10 Thus, at the block 411, a subdictionary 210-216 is searched, utilizing the logic 302, for the data character string represented by the prefix code and character held in the register 250, 400-402 associated with the subdictionary. The search may be performed
15 associatively, as described above, by examining the locations in the subdictionary that are storing previously entered strings. The associated subdictionary address counter 300 is utilized by the controller 295 to determine which addresses 224 of the subdictionary to examine so
20 as to only access locations at which strings are stored. If the string held in the register is stored at an examined location, the "hit" signal is provided by the associated hit/miss circuit. If the string is not stored in the subdictionary, the "miss" signal is indicated.

25 Further, at the phase 1 block 411, the string code of a string matched at a level (except level n) is applied to the code field of the subdictionary register of the next level and the subdictionary is searched for a matching string. Thus, at the block 411, the string
30 codes of strings matched at the respective levels are cascaded for matching in the subdictionaries of the next lower levels until a mismatching level occurs or all levels match.

Specifically, the string code from the string
35 code field 223 of a data character string match location in the subdictionary 213 is cascaded to the code field 261 of the register 400 of level 2. If a match is found

- 24 -

1 in the subdictionary 214, the corresponding string code
is cascaded to the code field 263 of the register 401
of level 3. Similarly, if matches are found at levels
3 to n-1, the corresponding string codes are cascaded
5 to next lower levels with the string code of the match
from the subdictionary of level n-1 cascaded to the code
field 265 of the register 402 of level n. If, however,
a match does not occur at a level, the cascading to lower
levels is terminated.

10 A specific protocol for performing the cascading
is that when the "hit" signal is provided by the hit/miss
circuit at a level, the string code of the match is
cascaded to the next lower level. In the cascading
process, when a hit/miss circuit at a level indicates
15 the "miss" signal, the mismatching level is identified,
terminating the cascading process. The cascading process
also terminates at level n if all of the hit/miss circuits
270-273 indicate the "hit" signal.

If, at the phase 1 block 411, a mismatching level
20 occurs, control continues with the phase 2a block 412.
A mismatching level occurs when the string represented
by the string code cascaded from the previous level,
extended by the input character applied to the level,
is not in the subdictionary at the level. Accordingly,
25 at the block 412, the code of the string that comprises
the longest match with the string in the input buffer
280 is output as the string code applied to the
mismatching level from the prior level. It is appreciated
that if the mismatch should occur at level 1, the longest
30 match is the single character string provided from the
first stage of the input buffer 280 to the code field
252 of the register 250.

The code of the longest match, therefore, is
provided by the code field 252, 261, 263 or 265 of the
35 respective register 250, 400-402 of the mismatching level
and transferred to the longest match register 290. The
code of the longest match is provided at the compressed

- 25 -

1 output 212 via the code size control 291 as previously
described. The mismatching character that "broke" the
match resides in the character field 251, 260, 262 or
264 of the respective register 250, 400-402 of the
5 mismatching level. The logic 305 is utilized in
determining the longest match.

The appropriate extended string is entered,
utilizing the logic 306, into the subdictionary of the
mismatching level by entering, into the prefix code field
10 222 and the character field 221 of the subdictionary
at the mismatching level, the code in the code field
and the character in the character field, respectively,
of the register at the mismatching level. The string
code is assigned to the extended string by entering the
15 code from the code counter 230 into the string code field
223 of the subdictionary location at which the extended
string is stored.

The location at which to store the extended string
is provided by the corresponding subdictionary address
20 counter 300. After storing the extended string, the
corresponding subdictionary address counter 300 is
incremented to the next available address 224. The code
counter 230 is advanced to the next available string
code and the code size in the code size register 310
25 is advanced if required. Additionally, the code fields
261, 263 and 265 of the registers 400-402 at the levels
2 to n are cleared.

The input buffer 280 is shifted entering new
characters from the input 211 at the stage n+1 until
30 the mismatching character is in the first buffer stage.
The level at which the mismatch occurs determines the
number of shifts to be applied by the shift control 281
to the input buffer 280. For example, if the mismatching
level is level 3, three shifts are required to bring
35 the mismatching character in the fourth stage of the
buffer 280 into the first stage thereof.

It is appreciated that many of the functions

- 26 -

1 of the phase 2a, block 412 are simultaneously performed.
The outputting of the longest match, the entering of
the extended string into the subdictionary, the assigning
of the string code from the code counter 230 and the
5 shifting of the input buffer 280 may be concurrently
executed. After the phase 2a functions of the block
412 are performed, control returns to the phase 1 block
411 to find the next longest match.

If, at the phase 1 block 411, all levels match,
10 control proceeds from the block 411 to the phase 2b block
413. At the block 413, all levels match since at each
of levels 1 to n, the string defined in the register
at the level is in the subdictionary of the level. When
this occurs, the n+1 character string held in the input
15 buffer 280 is already stored in the subdictionary 216
of level n. Accordingly, the code of the longest match
is derived from the string code field 223 of the
subdictionary 216 at the subdictionary location storing
the string matched at level n.

20 The code of the longest match is provided at
the compressed code output 212 via the longest match
register 290 and the code size control 291 in the manner
previously described. The input buffer 280 is shifted
entering n+1 new characters from the input 211. The
25 code counter 230 is advanced to the next available code
utilizing the input 231 in the manner previously
described. The code size in the code size register 310
is advanced if required. The code fields 261, 263 and
265 of the registers 400-402 at the levels 2 to n are
30 cleared. After the functions of the phase 2b block 413
are executed, control returns to the phase 1 block 411
to find the next longest match.

It is appreciated that operations of the phase
2b block 413 are simultaneously performed. The outputting
35 of the longest match, the shifting of the input buffer
280 and the advancing of code counter 230 may be
concurrently executed.

- 27 -

1 Referring to Figure 7, where like reference
numerals indicate like components with respect to Figures
1 and 4 and with continued reference to Figures 1 and
4, an addition to Figure 1 or Figure 4 for providing
5 enhanced processing of data character runs is illustrated.
A block 420 schematically represents the run detection
and processing of the compressor of said patent 6,188,333
which is incorporated herein in its entirety. The
compression apparatus of the block 420 communicates with
10 the controller 95 (295) via a bus 421 and provides
compressed codes at an output 422. The compressor 420
utilizes the input buffer 80 (280) and the shift control
81 (281) to provide the input look-ahead data required
by the apparatus 420. In the embodiment of Figure 7,
15 data character runs are processed by the apparatus 420
while non-run input data is processed by the above
described embodiments.

It is believed that by utilizing the architecture
of the present invention, fewer strings will be searched
20 in determining a longest match than in conventional
arrangements. For example, in the Figures 1 and 2
embodiment, only the potential matches extracted from
the subdictionaries to the auxiliary tables are searched
in determining the longest matching string.

25 It is appreciated that the compressed code output
provided by the above described embodiments is compatible
with standard LZW decompressors and the data character
stream corresponding to the compressed code can be
recovered thereby. Although the above described
30 embodiments of the invention are LZW based, it is
appreciated that the architecture of the present invention
can be utilized with other known dictionary based
compression methodologies. Although the search and
storage functions at the levels in the above embodiments
35 are explained in terms of associative memory
subdictionaries, it is appreciated that other search
and storage methodologies, such as hashing and sibling

- 28 -

1 lists, may also be utilized.

It is further appreciated that the input data characters can be over any size alphabet having any corresponding character bit size. For example, the data characters can be textual data, image pixel data or bit map data. The input data can also be binary characters over the two-character binary alphabet 1 and 0 having a 1-bit size character.

In selecting a number of levels in which to implement an embodiment, the statistics of the input data can be analyzed to choose an appropriate number of levels so as to optimize performance. The number of levels can also be chosen so that there will usually be a mismatching level at which to store an extended string.

Additionally, it is appreciated that the number of locations in the subdictionaries can vary from level to level and can also be chosen in accordance with the statistics of the data. For example, since it is expected that 2-character string matches may be more prevalent than longer string matches, the subdictionary of level 1 can include more locations than a subdictionary at a lower level. If the statistics of the string length distribution is known for the type of input data, the number of locations in each subdictionary can be selected in accordance with the statistics so as to minimize memory requirements.

It is appreciated that with respect to the Figures 1 and 2 embodiment, it is not necessary to clear the auxiliary tables between longest match determinations. By the use of the auxiliary table counters, only strings in the auxiliary tables that are currently applicable are examined.

Although an appropriate string code cascading protocol is described above, an alternative protocol may be utilized to provide an additional speed enhancement. In the alternative protocol, the operations

- 29 -

1 described above for an embodiment can be simultaneously
initiated with all of the stored auxiliary table entries
or subdictionary strings continuously searched, under
control of the auxiliary table counters or subdictionary
5 address counters, until "hit" signals begin to occur
starting with level 1. The controller 95 can examine
the hits and misses determining when a final status has
occurred at a mismatching level or when all of the levels
match. This action may be compared to that of a "slot
10 machine" where the different levels are spinning until
they sequentially settle to a final result.

For decompressor compatibility, a maximum code
limit can be imposed on the code counter 30 (230)
signifying that the compressor storage is full. Such
15 a maximum code can be 4096 as is typically utilized.
When the code counter attains the maximum count, the
compressor is considered as full with no further string
storage occurring. A decompressor can also terminate
string storage when the decompressor code counter attains
20 the maximum code for system compatibility.

A number of refinements can be incorporated into
the above described embodiments. For example, if an
auxiliary table counter or a subdictionary address counter
is zero, no strings are stored at the associated level
25 so matching can be terminated thereat. Additionally,
the controller 95 (295) can perform efficient operations
based on the state of the compressor. For example, when
the first n+1 characters are entered into the input
buffer, abbreviated operations can be performed since
30 at that point no strings are stored at any level.

It is appreciated that the above described
embodiments of the invention may be implemented in
hardware, firmware, software or a combination thereof.
Discrete circuit embodiments may readily be implemented
35 for performing the various described functions. In a
software embodiment, appropriate modules programmed with
coding readily generated from the above descriptions

- 30 -

1 may be utilized.

While the invention has been described in its preferred embodiments, it is to be understood that the words which have been used are words of description rather than of limitation and that changes may be made within the purview of the appended claims without departing from the true scope and spirit of the invention in its broader aspects.

10

15

20

25

30

35

- 31 -

CLAIMS

- 1 1. Data compression apparatus for compressing an
input stream of data characters into an output stream
of compressed codes, comprising
- 5 a plurality of storage means for storing strings
of data characters encountered in said input stream,
said stored strings having respective string codes
associated therewith, said storage means being arranged
in respective levels so that the strings stored in the
storage means at a level comprise one character more
10 than the strings stored in the storage means at the level
prior thereto,
- means for fetching a plurality of data characters
from said input stream and applying respective fetched
characters to said levels,
- 15 means for searching said fetched characters by
comparing said fetched characters to said stored strings
to determine the longest match therewith, said longest
match being determined by one of said fetched characters
resulting in a mismatch at one of said levels,
- 20 said means for searching including means for
searching the strings stored in the storage means at
a particular level for a string comprising a string
matched at the level prior thereto extended by the fetched
character applied to said particular level and means
25 for applying the string code of a string matched at said
particular level to the next level,
- means for outputting the string code associated
with said longest match so as to provide said output
stream of compressed codes, and
- 30 means for inserting an extended string into said
storage means, said extended string comprising said
longest match extended by the fetched character resulting
in said mismatch, said extended string being stored in
said storage means at the mismatching level.

35

- 32 -

1 2.

The compression apparatus of claim 1 wherein said means for searching is operative for determining a level whereat the storage means at the level is not storing a string comprising the string matched at the level prior thereto extended by the fetched character applied to the level, thereby determining said mismatching level,

said string code associated with said longest match comprises the string code applied to said mismatching level from the level prior thereto, and said extended string comprises said code associated with said longest match extended by the fetched character applied to said mismatching level.

15 3.

The compression apparatus of claim 2 wherein said means for fetching comprises

an input buffer for holding said plurality of data characters fetched from said input stream and for applying said fetched characters to said levels, and means for shifting said plurality of data characters in said input buffer so that the character applied to said mismatching level is shifted to a first stage of said input buffer so as to provide the first character of a next longest match.

25

. 30

35

- 33 -

1 4. The compression apparatus of claim 1 wherein
said data characters are from an alphabet of
data characters,

5 a string stored in said storage means comprises
a prefix string of at least one of said characters
followed by an extension character,

said storage means at said particular level
comprises a subdictionary storing strings having the
same number of data characters,

10 said storage means at said particular level
includes an auxiliary table, and

said means for searching includes means for
screening strings from said subdictionary where the
screened strings have an extension character matching
15 the fetched character applied to said particular level
and storing the screened strings in said auxiliary table.

20

25

30

35

34

- 11 -

5

1

pp. Data compression apparatus for compressing an input stream of data characters into an output stream of compressed codes, comprising

- 5 a plurality of storage means for storing strings of data characters encountered in said input stream, said stored strings having respective string codes associated therewith, said storage means being arranged in respective levels so that the strings stored in the storage means at a level comprise one character more
- 10 than the strings stored in the storage means at the level prior thereto, said levels including a last level, means for fetching a plurality of data characters from said input stream and applying the fetched characters to said levels, respectively,
- 15 means for searching said fetched characters by comparing said fetched characters to said stored strings to determine the longest match therewith, said means for searching being operative for searching the strings stored in the storage means at a particular level for
- 20 a string comprising a string matched at the level prior thereto extended by the fetched character applied to said particular level and for applying the string code of a string matched at said particular level to the next level, said longest match being determined by matching
- 25 the plurality of fetched data characters with a string stored in the storage means at said last level, and means for outputting the string code associated with the string matched in the storage means at said last level so as to provide said output stream of
- 30 compressed codes.

35

14-04-2003

35

- 11 -

6

1 ~~11~~. A data compression method for compressing an
input stream of data characters into an output stream
of compressed codes, comprising
storing strings of data characters encountered
5 in said input stream in a plurality of storage means,
said stored strings having respective string codes
associated therewith, said storage means being arranged
in respective levels so that the strings stored in the
storage means at a level comprise one character more
10 than the strings stored in the storage means at the level
prior thereto,
fetching a plurality of data characters from
said input stream and applying respective fetched
characters to said levels,
15 searching said fetched characters by comparing
said fetched characters to said stored strings to
determine the longest match therewith, said longest match
being determined by one of said fetched characters
resulting in a mismatch at one of said levels,
20 said searching step including searching the
strings stored in the storage means at a particular level
for a string comprising a string matched at the level
prior thereto extended by the fetched character applied
to said particular level and applying the string code
25 of a string matched at said particular level to the next
level,
outputting the string code associated with said
longest match so as to provide said output stream of
compressed codes, and
30 inserting an extended string into said storage
means, said extended string comprising said longest match
extended by the fetched character resulting in said
mismatch, said extended string being stored in said
storage means at the mismatching level.

35

36

- 1A -

6

7.
1 7A.

The compression method of claim 6 wherein
said searching step includes determining a level
whereat the storage means at the level is not storing
a string comprising the string matched at the level prior
thereto extended by the fetched character applied to
the level, thereby determining said mismatching level,
said string code associated with said longest
match comprises the string code applied to said
mismatching level from the level prior thereto, and
said extended string comprises said code
associated with said longest match extended by the fetched
character applied to said mismatching level.

8 7A. The compression method of claim 7 wherein said
fetching step comprises

utilizing an input buffer holding said plurality
of data characters fetched from said input stream and
applying said fetched characters to said levels, and
shifting said plurality of data characters in
said input buffer so that the character applied to said
mismatching level is shifted to a first stage of said
input buffer so as to provide the first character of
a next longest match.

25

30

35

37

- 11 -

6/1

1 9

11.

The compression method of claim ^{6/1} wherein
said data characters are from an alphabet of
data characters,

5 a string stored in said storage means comprises
a prefix string of at least one of said characters
followed by an extension character,

said storage means at said particular level
comprises a subdictionary storing strings having the
same number of data characters,

10 said storage means at said particular level
includes an auxiliary table; and

said searching step includes screening strings
from said subdictionary where the screened strings have
an extension character matching the fetched character
15 applied to said particular level and storing the screened
strings in said auxiliary table.

20

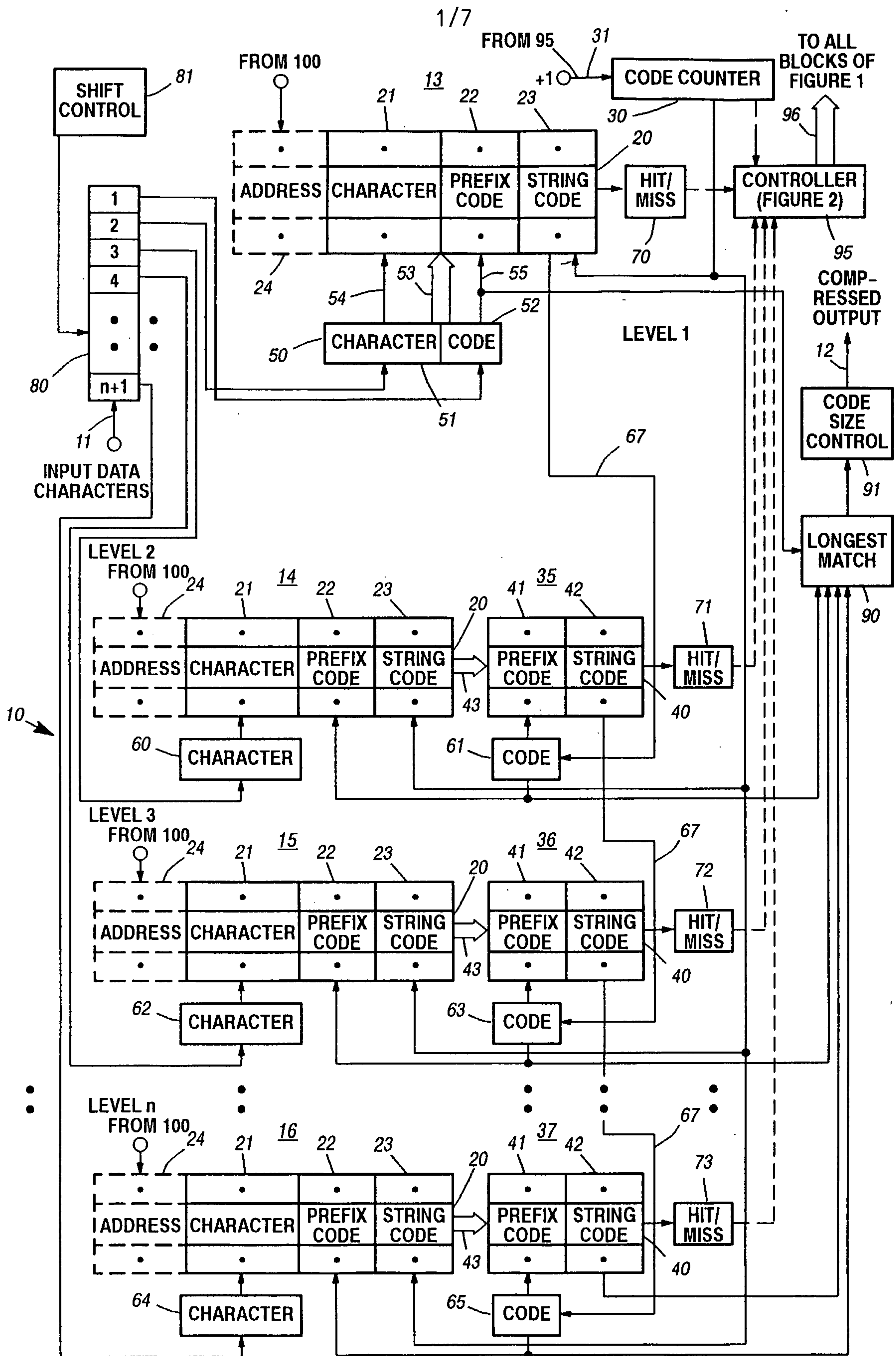
25

30

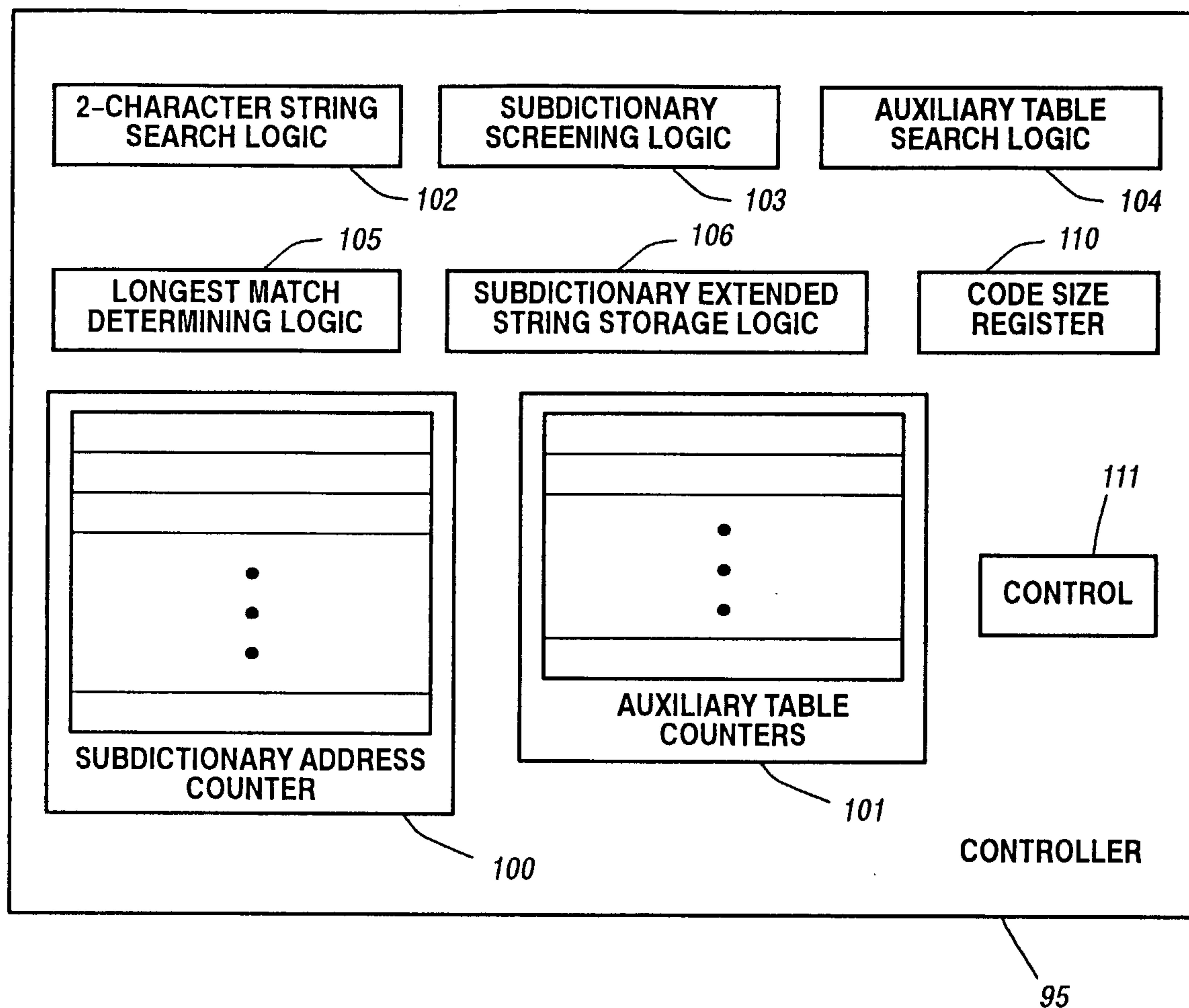
35

-38-

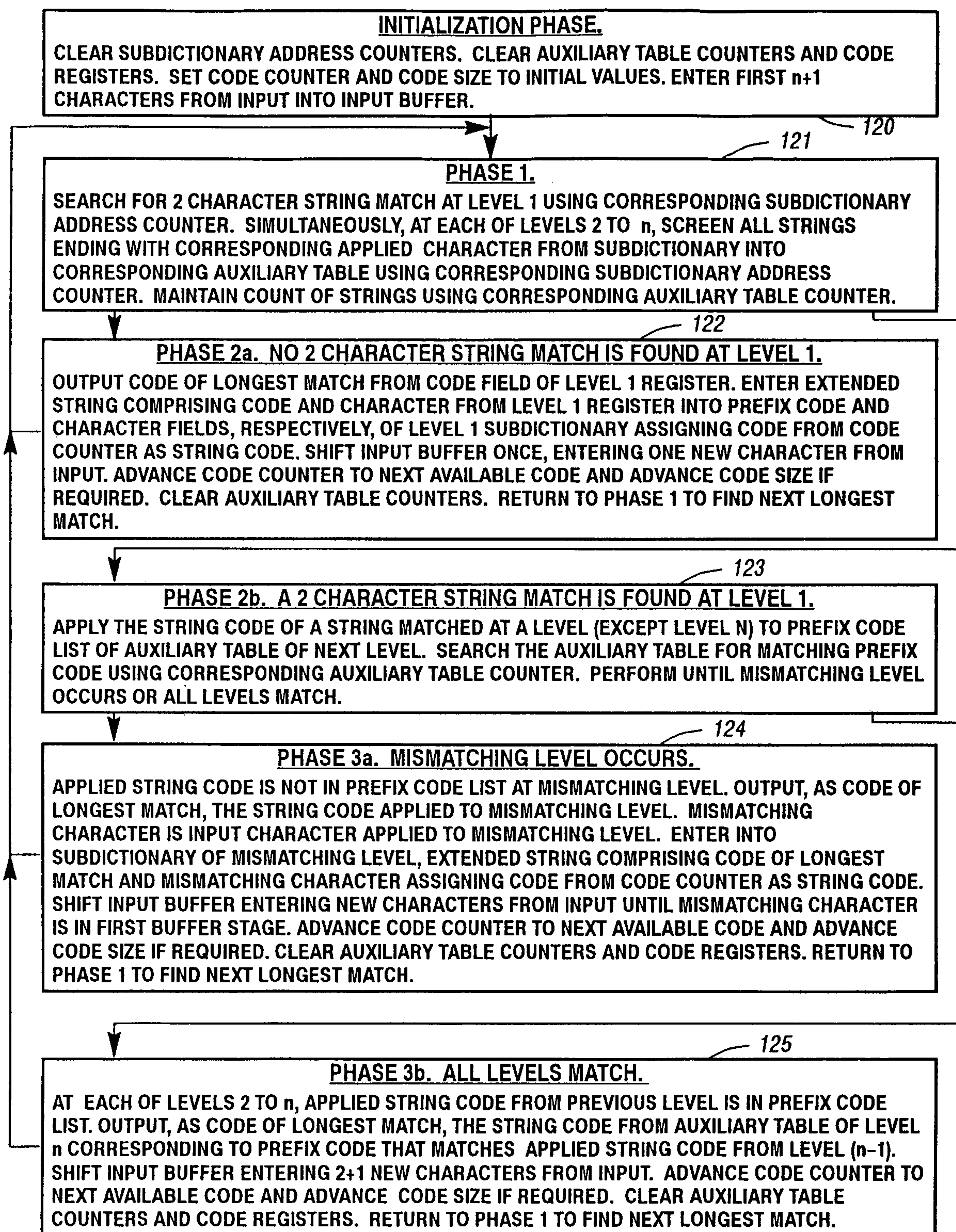
10. A data compression method for compressing an input stream of data characters into an output stream of compressed codes, comprising storing strings of data characters encountered in said input stream in a plurality of storage means, said stored strings having respective string codes associated therewith, said storage means being arranged in respective levels so that the strings stored in the storage means at a level comprise one character more than the strings stored in the storage means at the level prior thereto, said levels including a last level, fetching a plurality of data characters from said input stream and applying the fetched characters to said levels, respectively, searching said fetched characters by comparing said fetched characters to said stored strings to determine the longest match therewith, said searching step including searching the strings stored in the storage means at a particular level for a string comprising a string matched at the level prior thereto extended by the fetched character applied to said particular level and for applying the string code of a string matched at said particular level to the next level, said longest match being determined by matching the plurality of fetched data characters with a string stored in the storage means at said last level, and outputting the string code associated with the string matched in the storage means at said last level so as to provide said output stream of compressed codes.

**Figure 1**

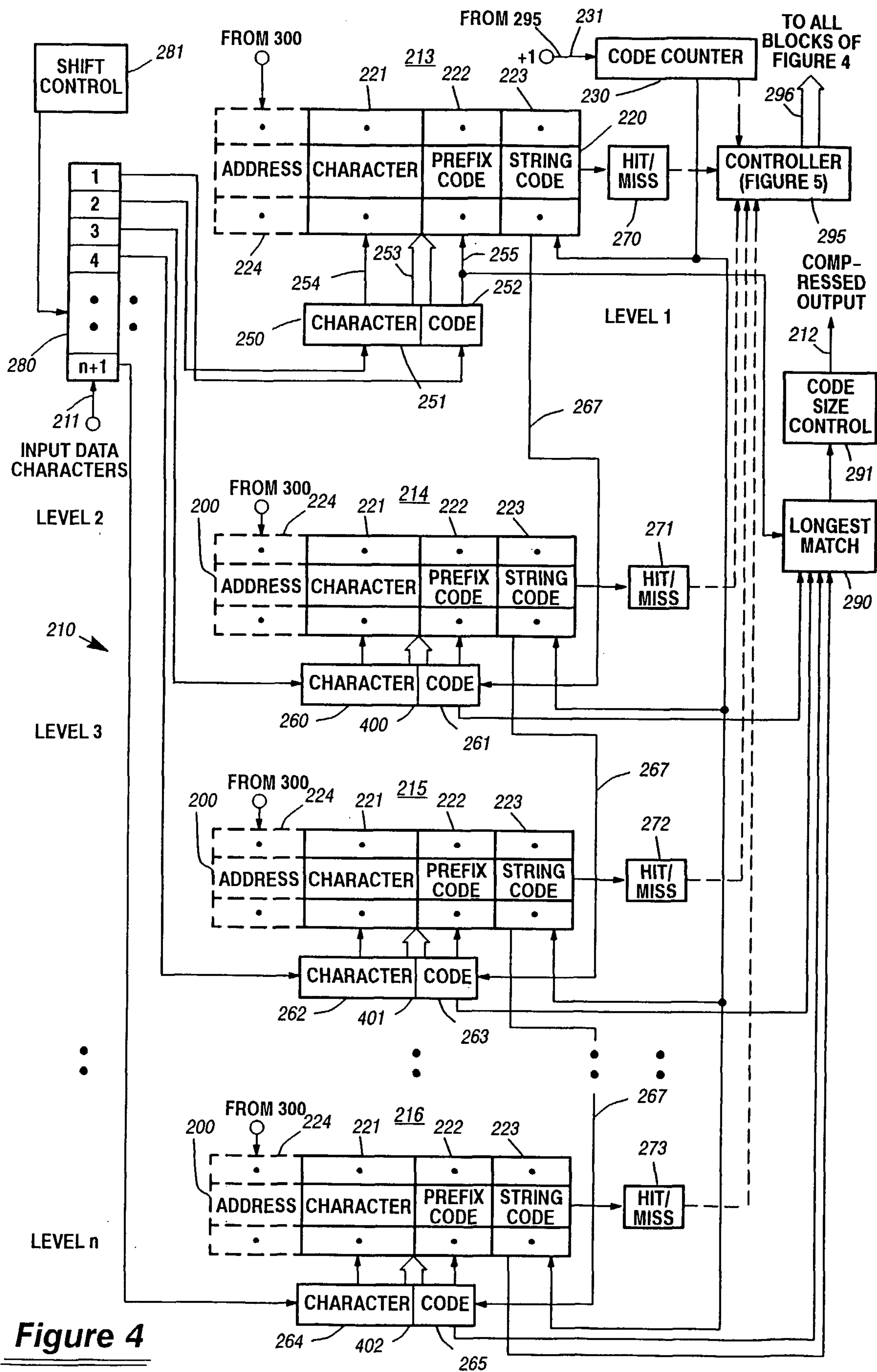
2/7

Figure 2

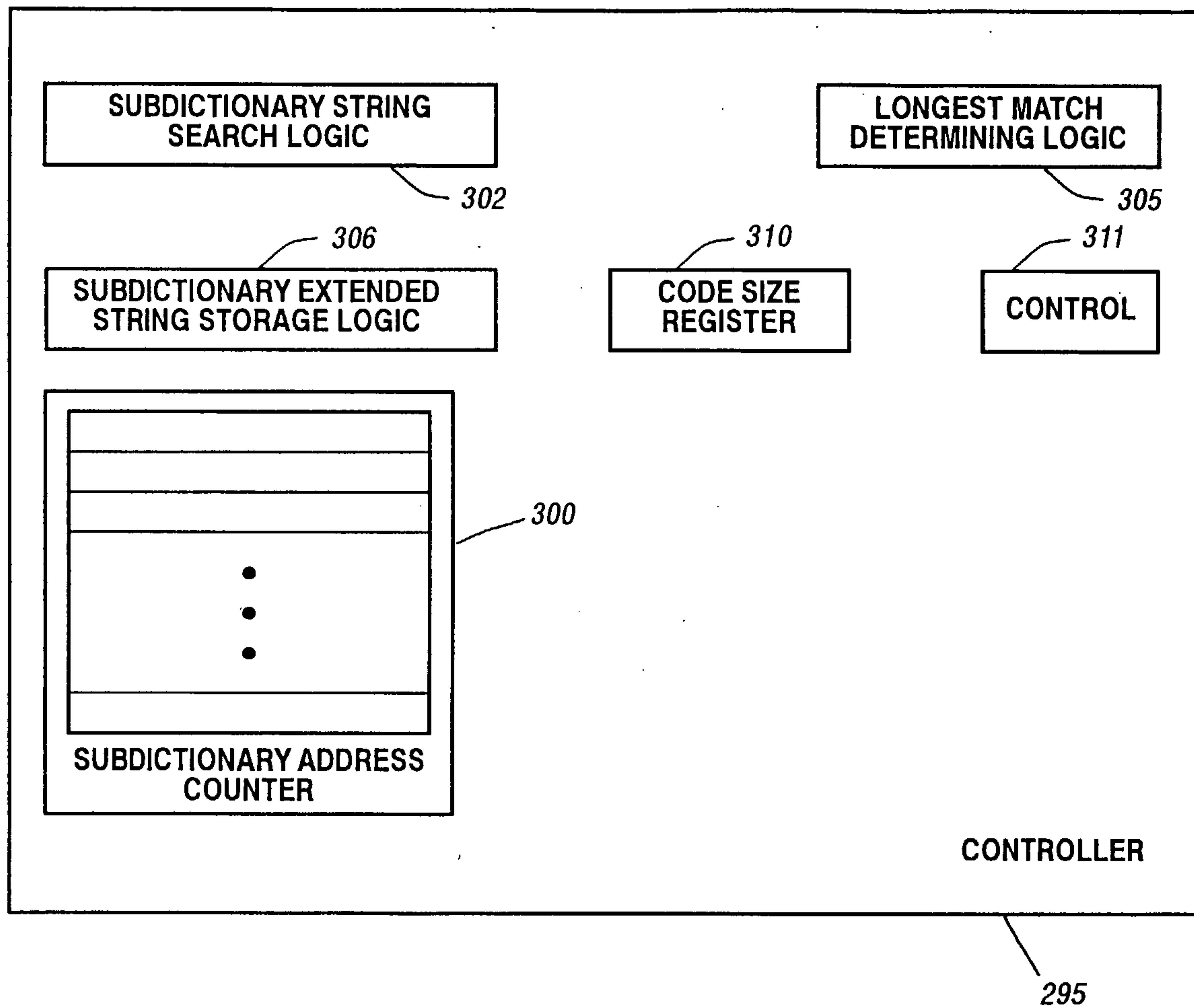
3/7

Figure 3

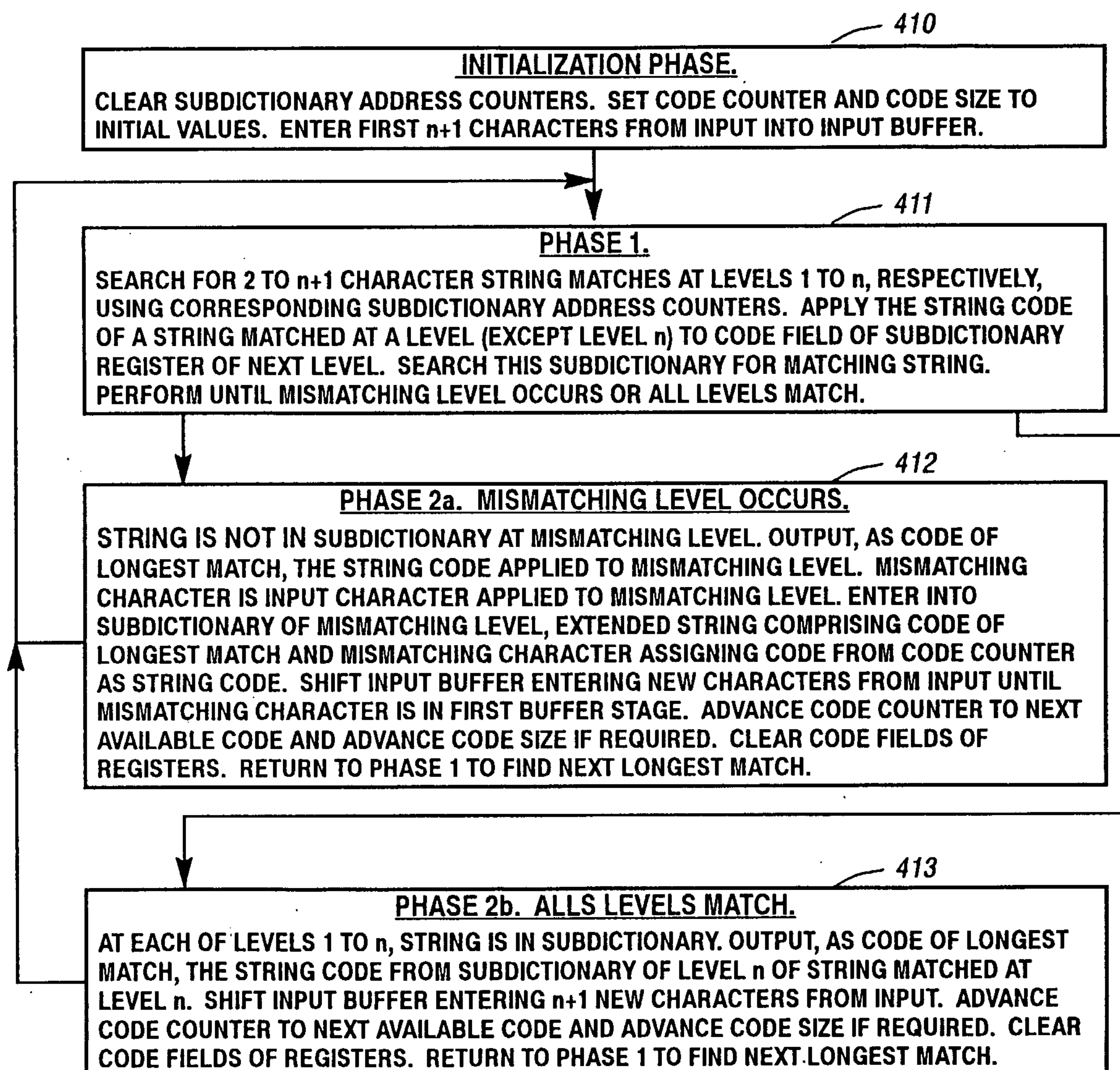
4/7



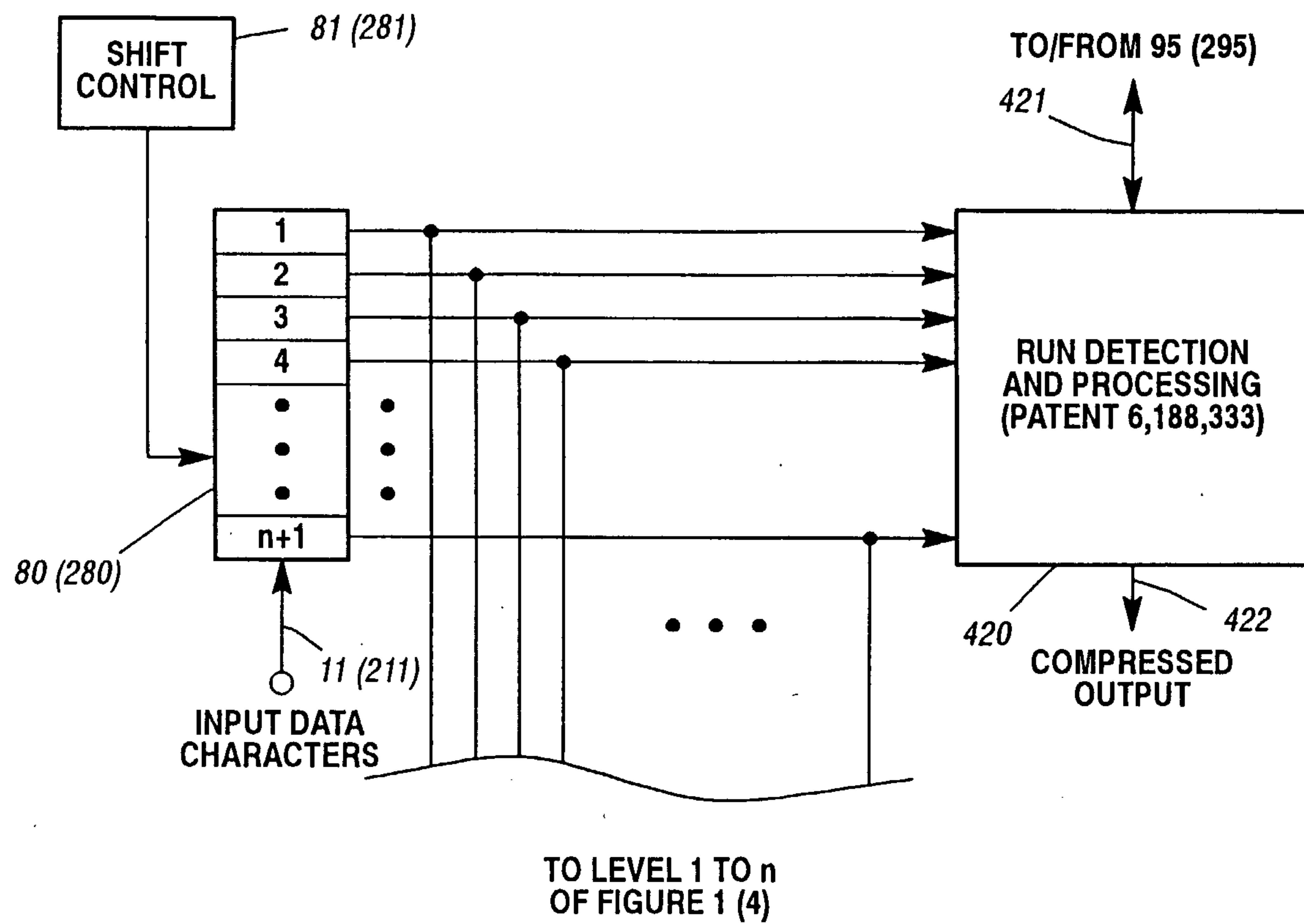
5/7

Figure 5

6/7

**Figure 6**

7/7

Figure 7

