IN-17
159834

NASA CASE NO. NPO-17,310-1CU

PRINT FIG. 2

35P.

NRO-JPL

(NASA-Case-NPO-17310-1-CU) METHOD FOR VITERBI DECODING OF LARGE CONSTRAINT LENGTH CONVOLUTIONAL CODES Patent Application (NASA) 35 p CSCL 09F

N88-28946

Unclas
G3/17 0159834

METHOD FOR VITERBI DECODING OF LARGE CONSTRAINT LENGTH CONVOLUTIONAL CODES

Inventor: In-Shek Hsu, Trieu-Kie Truong
Irving S. Reed and Jing Sun
Contractor: Jet Propulsion
Laboraratory

JPL Case No. 17310
NASA Case No. NPO-17310-1-CU
May 20, 1988

AWARDS ABSTRACT

The invention relates to a pipeline Viterbi decoding of a convolutional code used in a concatenated coding system with a Reed-Solomon outer code for down link telemetry.

A prior-art encoder for a (1/2,3) convolutional code is shown in **FIG. 1** as a simple example of the code to be decoded. A trellis diagram for the encoder is shown in **FIG. 2**. The same trellis diagram for the decoder is then shown in **FIG. 3** with the path for a received message of 10, 10, 10, 11, 00, 11, 11 emphasized by heavy lines. A functional block diagram of the decoder is shown in **FIG. 4** with a "metric computation and path decision" unit **12** recycles for the next time unit the node metrics m_0^{t+1}, m_1^{t+1}, m_2^{t+1}, m_3^{t+1} through a "data multiplexing" unit **13** shown in **FIG. 9** and a "data rearrangement" unit **11** shown in **FIG. 5** which sorts the node data for metric generation and comparison of nodes 0 and 2, and nodes 1 and 3 in selecting node metrics in the manner shown in **FIG. 6** using four sets of "path metric generating" circuits PM-0, -1, -2 and -3. Each metric generating circuit is implemented as shown in **FIG. 8**. "Metric comparison" circuits **23** and **24** shown in **FIG. 6** perform the four comparisons illustrated in **FIG. 7a** and **7b**. Together, the "path metric generating" and "metric comparison" units produce on separate lines

$$m_0^{t+1},\ p_0^{t+1},\ q_0^{t+1}$$

$$m_1^{t+1},\ p_1^{t+1},\ q_1^{t+1}$$

and

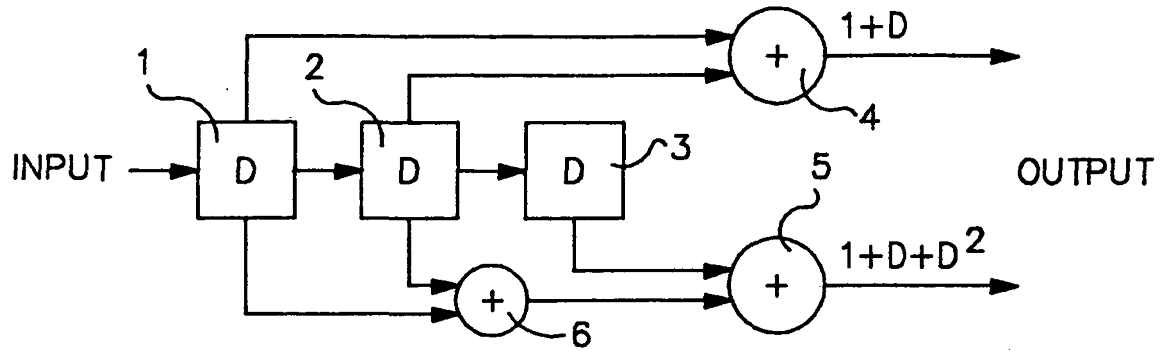
$$m_2^{t+1},\ p_2^{t+1},\ q_2^{t+1}$$

$$m_3^{t+1},\ p_3^{t+1},\ q_3^{t+1}$$

The "data multiplexing" unit **13** sorts and transmits the "p" and "q" information (survivors of state nodes "0", "1", "2" and "3", and branch metrics of the state nodes, respectively) to a "path metric comparison unit **25** which transmits a path metric decision to a "data storage and path selection" unit **14** shown in **FIG. 4**. Meantime, the "data multiplexing" unit sorts and transmits the new partial path metric data m_0^{t+1}, m_1^{t+1}, m_2^{t+1} and m_3^{t+1} to be used for the operation of the "metric computation and path decision" unit **12** during the next time unit t+2. **FIG. 10** illustrates the "data storage and path selection" unit **14** from which the path to be read out as the decoded output bits "q" from a RAM **30** shown in **FIG. 10** is selected. This RAM storage arrangement permits the "decision" from the "metric computation and path decision" unit **12** to select the last branch metric "q" from the stream of data stored of the constraint length K, and to then "trace back" to the beginning of that data stored of the constraint length K to read out the branch metrics (q's) as the decoded data in sequence. In the meantime, data from another constraint length is stored in another block of the RAM.

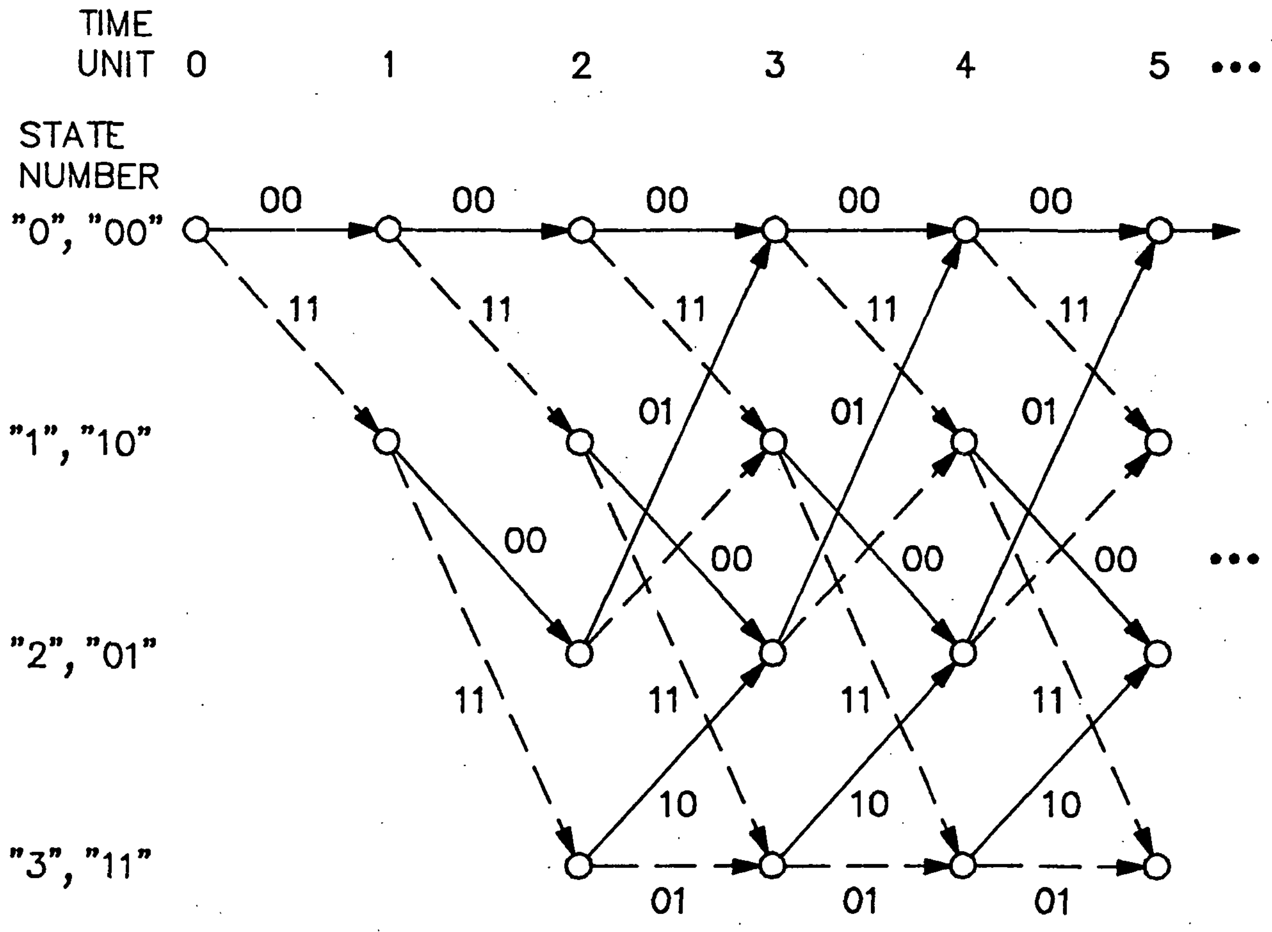
The novelty of the invention resides in the pipeline architecture for a Viterbi decoder of relatively long constraint length K=7 to 14, for example, which lends itself to VLSI fabrication on a chip. The central part is the "metric computation and path decision" unit **12**. This architecture uses a RAM to store the "p" and "q" data of all node states for a constraint length K, and to then select the decoded metric path from the last node, tracing back to the first, and then outputing as the decoded bits the q data stored for the constraint length K.

Serial No.	200,874	
Filing Date	6/1/88	
Contract No.	NAS7-918	
Contractor	Caltech/JPL	
Pasadena	CA	91109
(City)	(State)	(Zip)



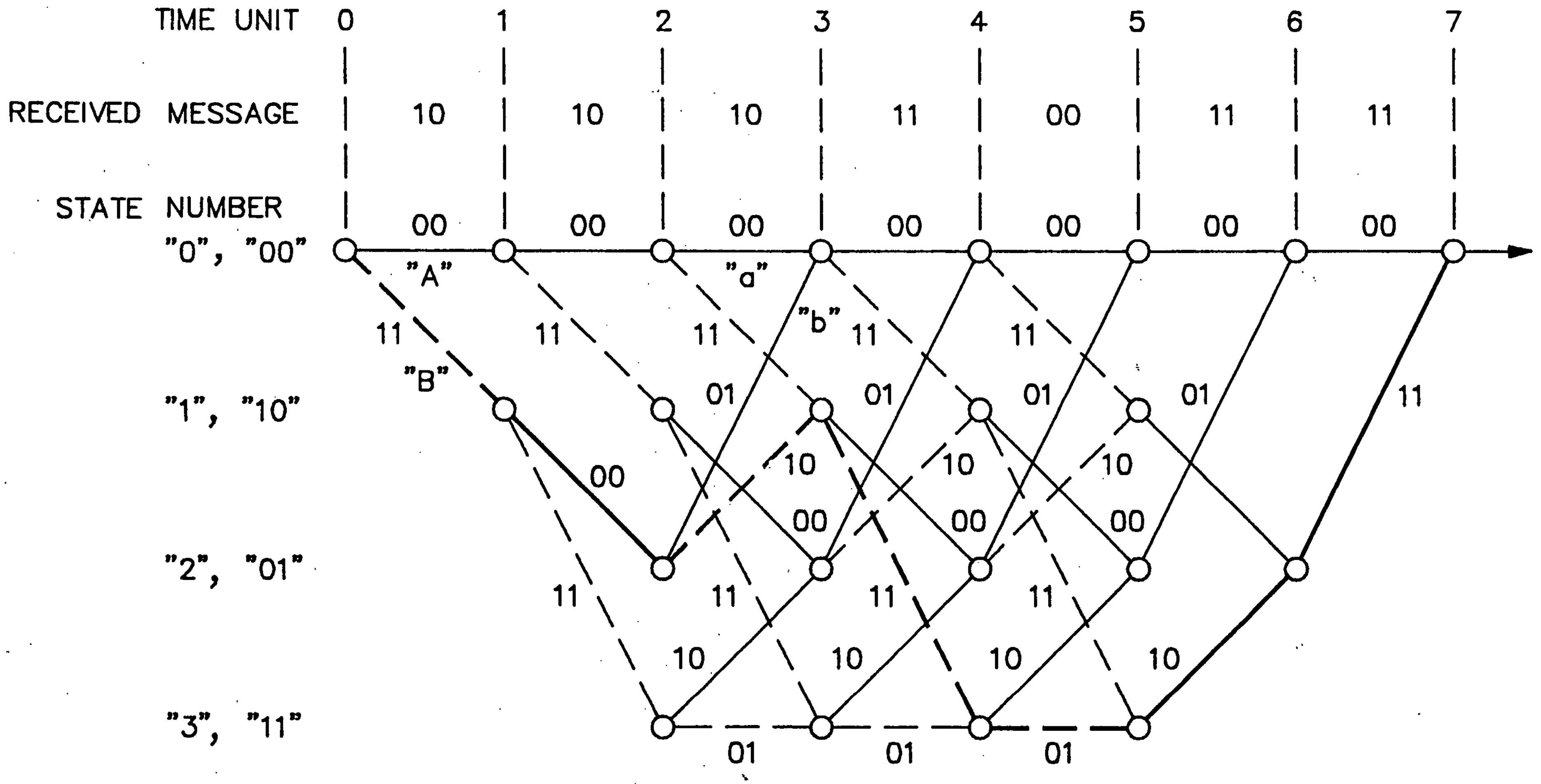
CONVENTIONAL PRIOR-ART ENCODER

FIG. 1



TRELLIS DIAGRAM FOR ENCODER

FIG. 2



TIME UNIT
0
1
2
3
4
5
6
7
RECEIVED MESSAGE
10
10
10
11
00
11
11
STATE NUMBER
"0", "00"
"1", "10"
"2", "01"
"3", "11"
00
"A"
"a"
"b"
"B"
11
01
10
TRELLIS DIAGRAM FOR DECODER

FIG. 3

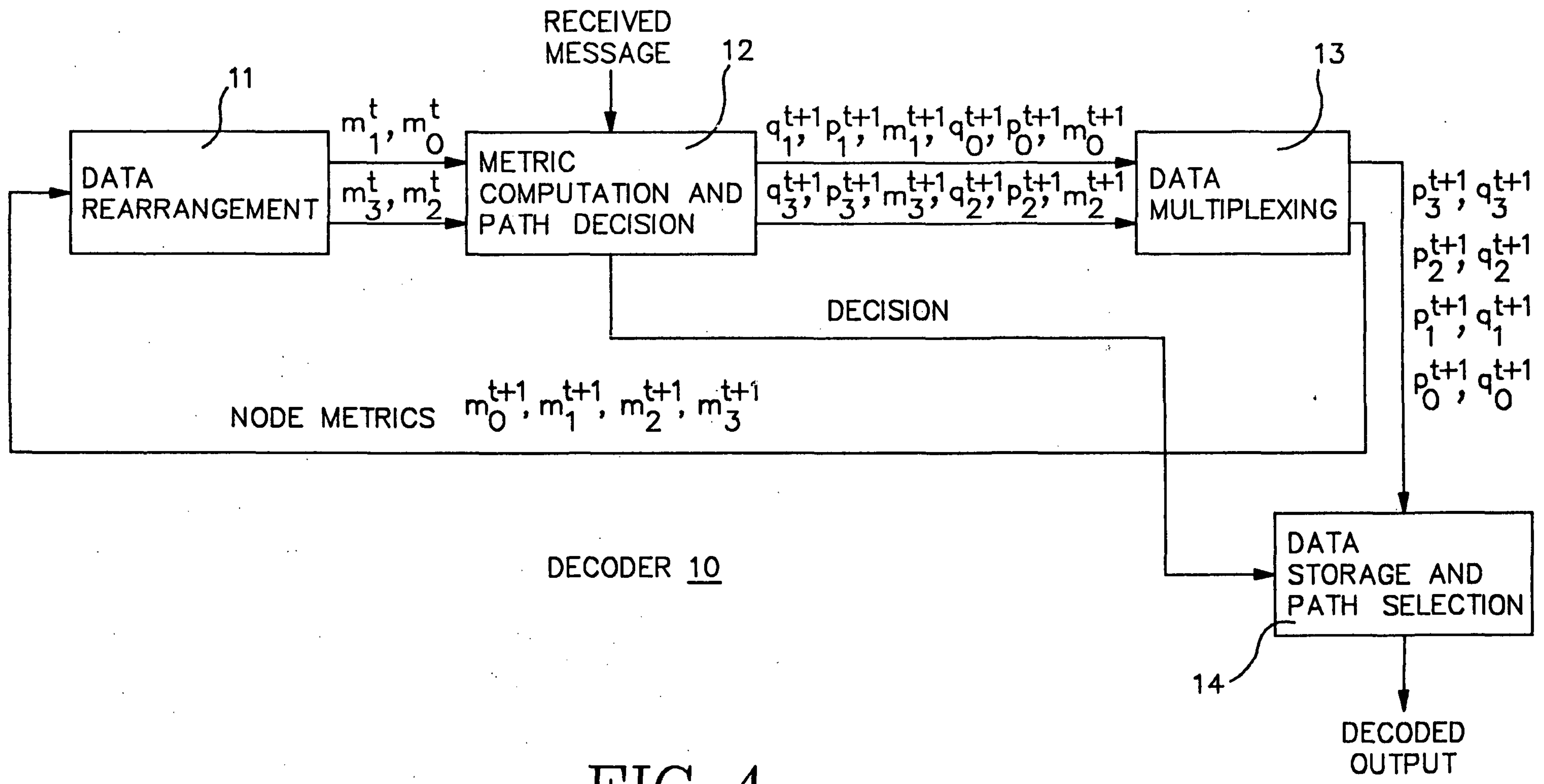

RECEIVED MESSAGE
11
12
13
DATA REARRANGEMENT
m_1^t, m_0^t
m_3^t, m_2^t
METRIC COMPUTATION AND PATH DECISION
$q_1^{t+1}, p_1^{t+1}, m_1^{t+1}, q_0^{t+1}, p_0^{t+1}, m_0^{t+1}$
$q_3^{t+1}, p_3^{t+1}, m_3^{t+1}, q_2^{t+1}, p_2^{t+1}, m_2^{t+1}$
DATA MULTIPLEXING
p_3^{t+1}, q_3^{t+1}
p_2^{t+1}, q_2^{t+1}
p_1^{t+1}, q_1^{t+1}
p_0^{t+1}, q_0^{t+1}
DECISION
NODE METRICS $m_0^{t+1}, m_1^{t+1}, m_2^{t+1}, m_3^{t+1}$
DECODER 10
DATA STORAGE AND PATH SELECTION
14
DECODED OUTPUT

FIG. 4

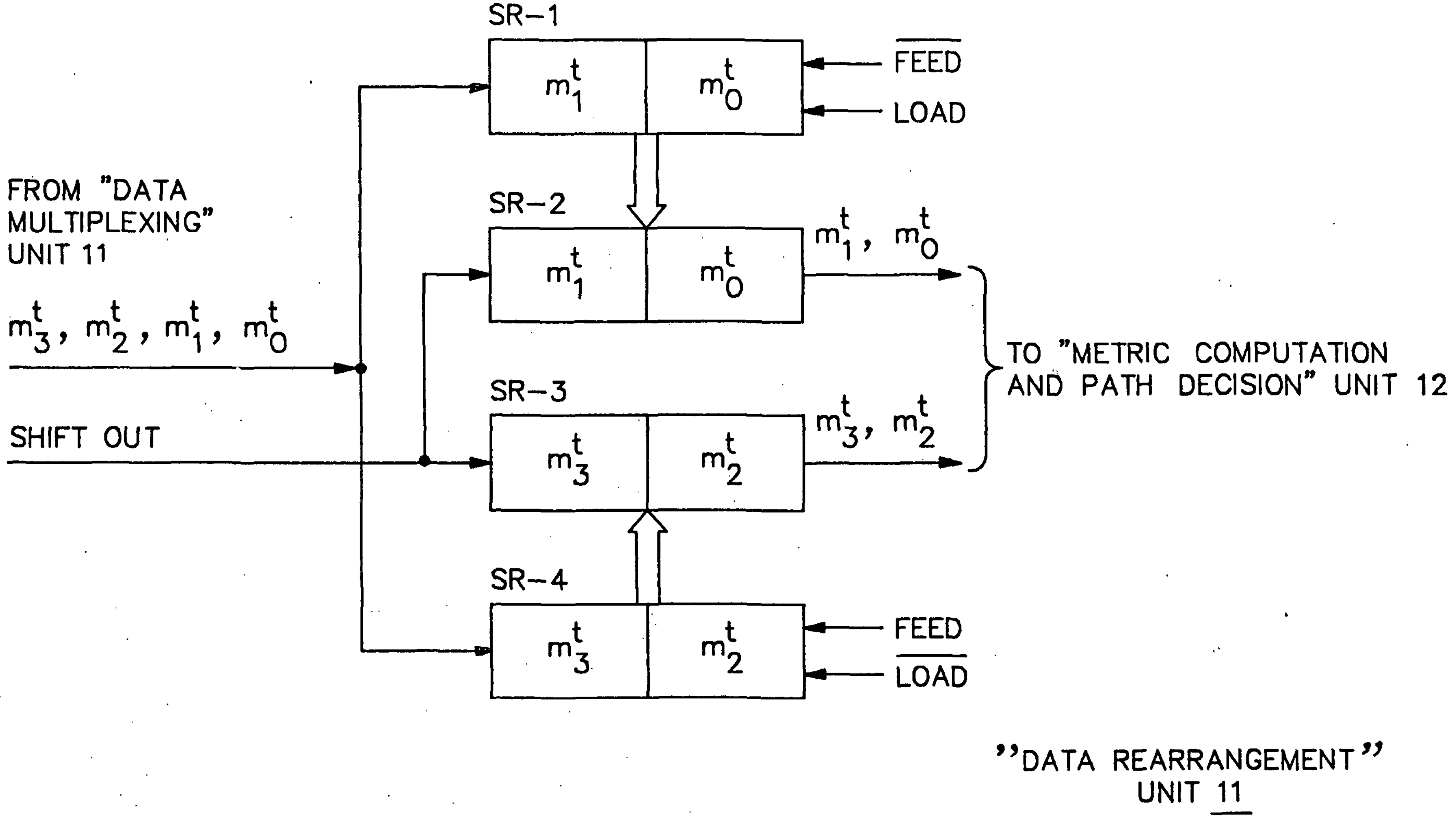
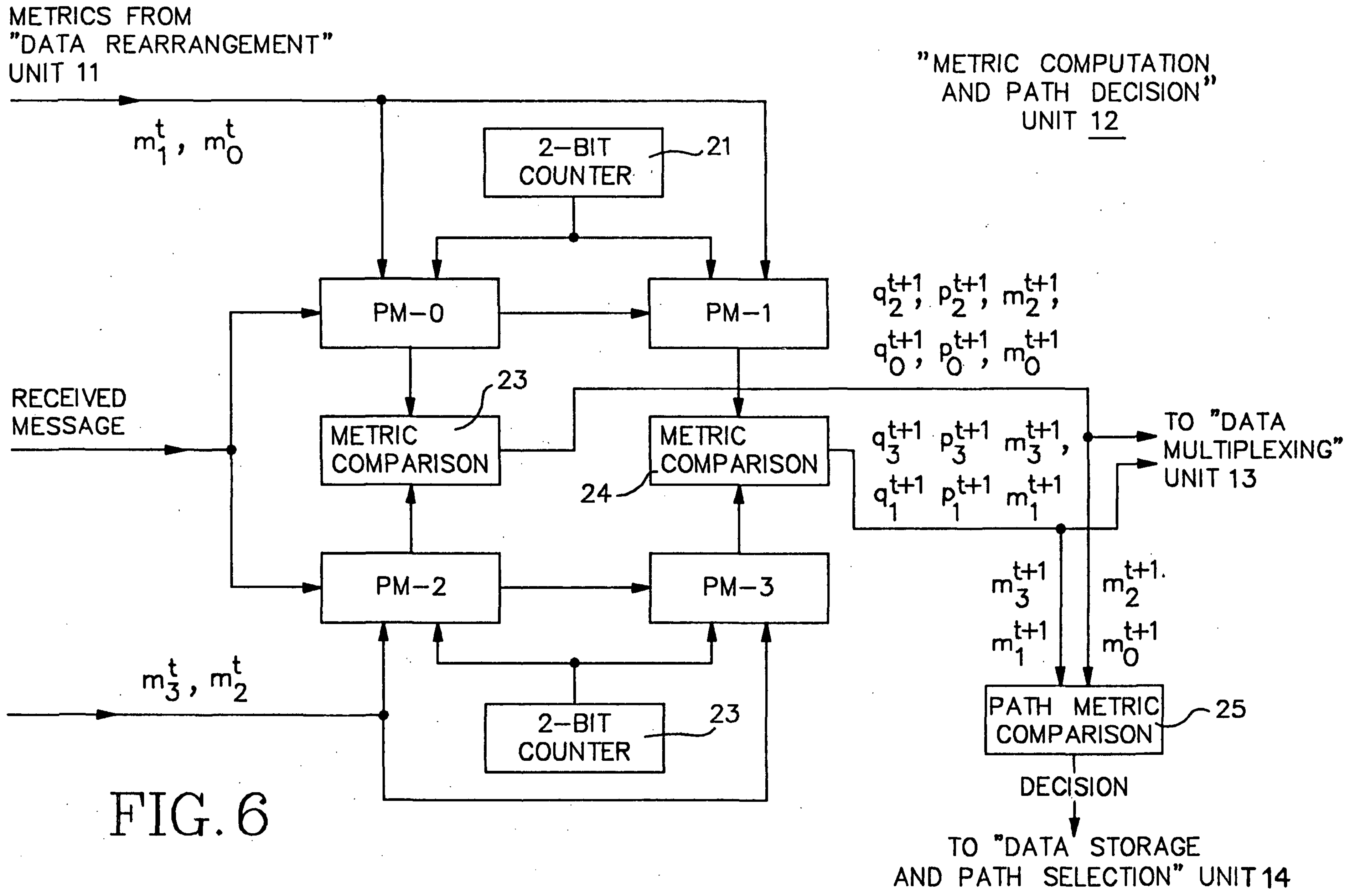

SR–1
m_1^t
m_0^t
$\overline{\text{FEED}}$
LOAD
FROM "DATA MULTIPLEXING" UNIT 11
SR–2
m_1^t
m_0^t
m_1^t, m_0^t
m_3^t, m_2^t, m_1^t, m_0^t
TO "METRIC COMPUTATION AND PATH DECISION" UNIT 12
SR–3
SHIFT OUT
m_3^t
m_2^t
m_3^t, m_2^t
SR–4
m_3^t
m_2^t
FEED
$\overline{\text{LOAD}}$
"DATA REARRANGEMENT" UNIT 11

FIG.5



METRICS FROM "DATA REARRANGEMENT" UNIT 11
m_1^t, m_0^t
"METRIC COMPUTATION AND PATH DECISION" UNIT 12
2-BIT COUNTER
21
PM-0
PM-1
q_2^{t+1}, p_2^{t+1}, m_2^{t+1},
q_0^{t+1}, p_0^{t+1}, m_0^{t+1}
RECEIVED MESSAGE
23
METRIC COMPARISON
METRIC COMPARISON
24
q_3^{t+1} p_3^{t+1} m_3^{t+1},
q_1^{t+1} p_1^{t+1} m_1^{t+1}
TO "DATA MULTIPLEXING" UNIT 13
PM-2
PM-3
m_3^{t+1}
m_1^{t+1}
m_2^{t+1}
m_0^{t+1}
m_3^t, m_2^t
2-BIT COUNTER
23
PATH METRIC COMPARISON
25
DECISION
TO "DATA STORAGE AND PATH SELECTION" UNIT 14

FIG. 6

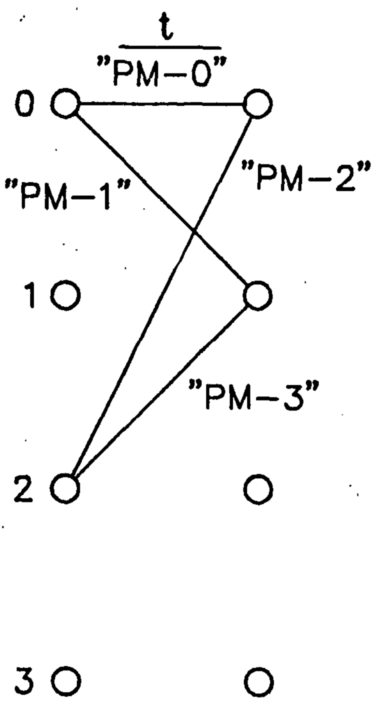


FIG. 7a

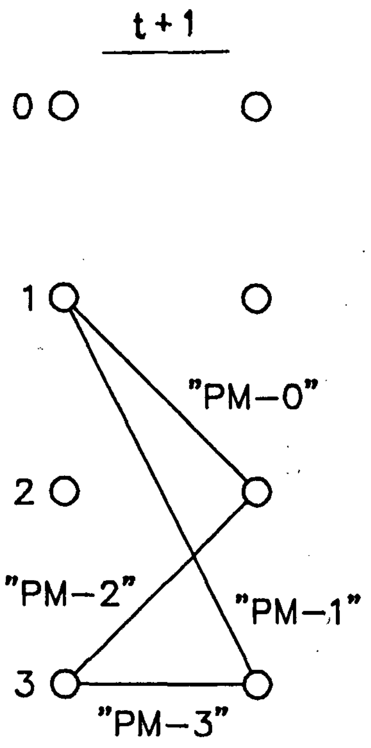
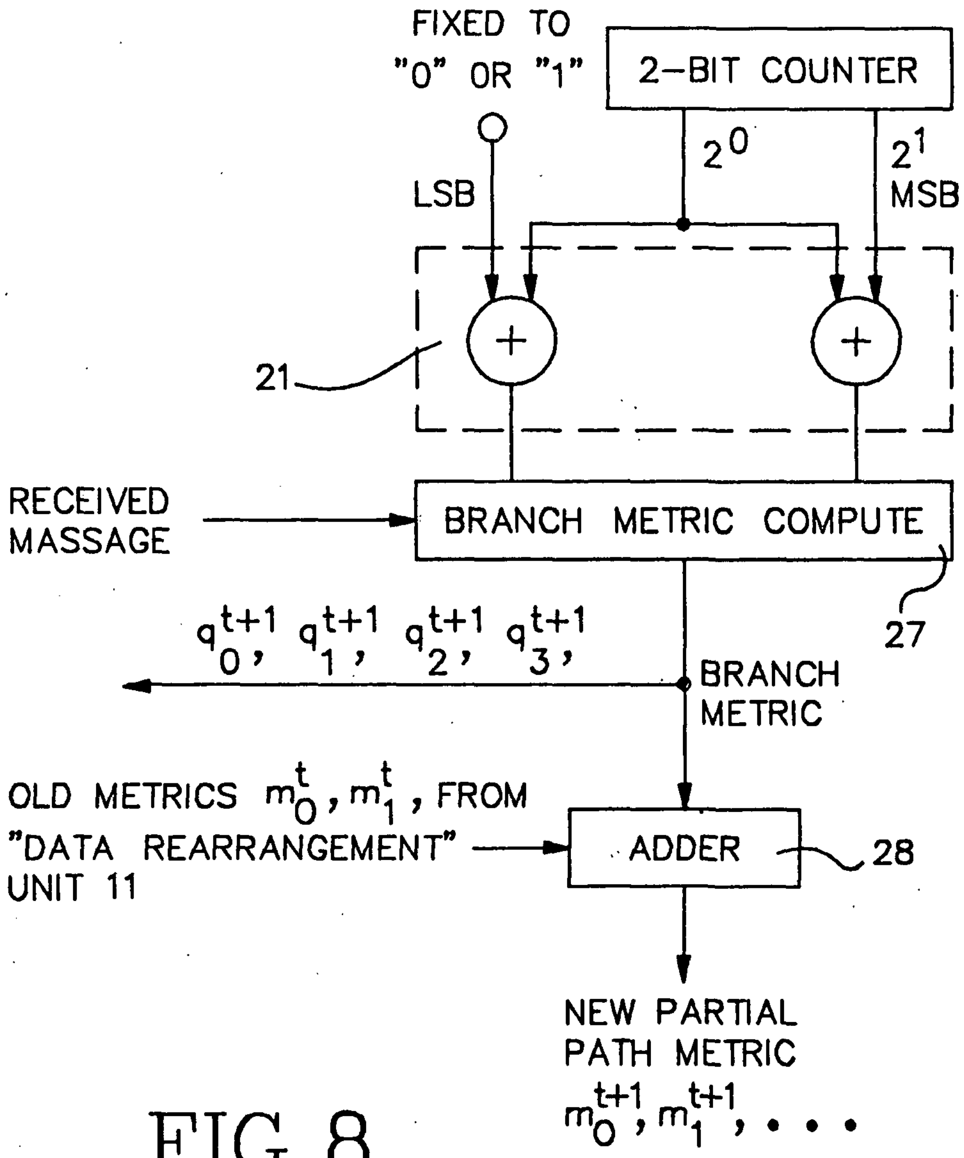


FIG. 7b



FIXED TO "0" OR "1"
2-BIT COUNTER
LSB
2^0
2^1 MSB
21
RECEIVED MASSAGE
BRANCH METRIC COMPUTE
27
$q_0^{t+1}, q_1^{t+1}, q_2^{t+1}, q_3^{t+1}$,
BRANCH METRIC
OLD METRICS m_0^t, m_1^t, FROM "DATA REARRANGEMENT" UNIT 11
ADDER
28
NEW PARTIAL PATH METRIC $m_0^{t+1}, m_1^{t+1}, \ldots$

FIG. 8

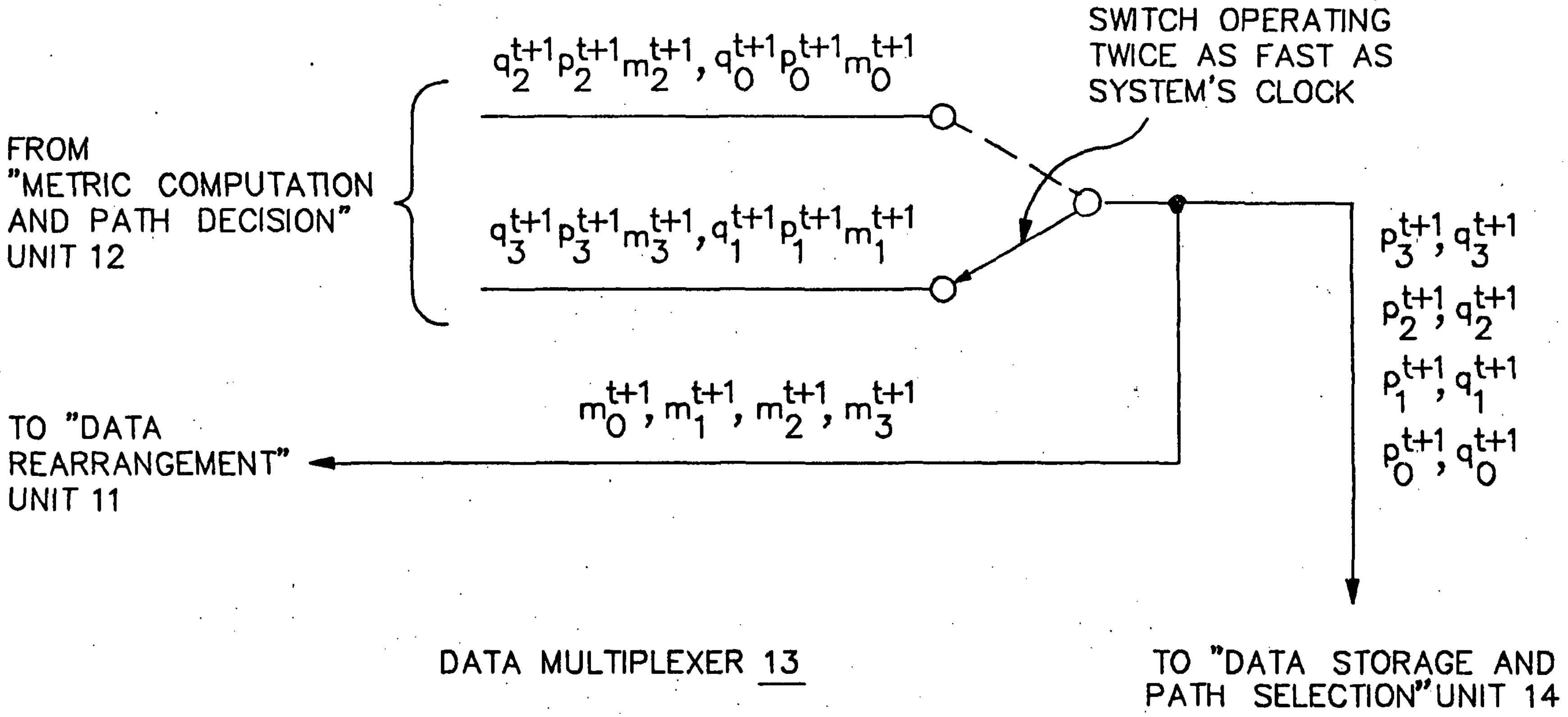

FROM "METRIC COMPUTATION AND PATH DECISION" UNIT 12
$q_2^{t+1} p_2^{t+1} m_2^{t+1}, q_0^{t+1} p_0^{t+1} m_0^{t+1}$
$q_3^{t+1} p_3^{t+1} m_3^{t+1}, q_1^{t+1} p_1^{t+1} m_1^{t+1}$
SWITCH OPERATING TWICE AS FAST AS SYSTEM'S CLOCK
p_3^{t+1}, q_3^{t+1}
p_2^{t+1}, q_2^{t+1}
p_1^{t+1}, q_1^{t+1}
p_0^{t+1}, q_0^{t+1}
$m_0^{t+1}, m_1^{t+1}, m_2^{t+1}, m_3^{t+1}$
TO "DATA REARRANGEMENT" UNIT 11
DATA MULTIPLEXER 13
TO "DATA STORAGE AND PATH SELECTION" UNIT 14

FIG. 9

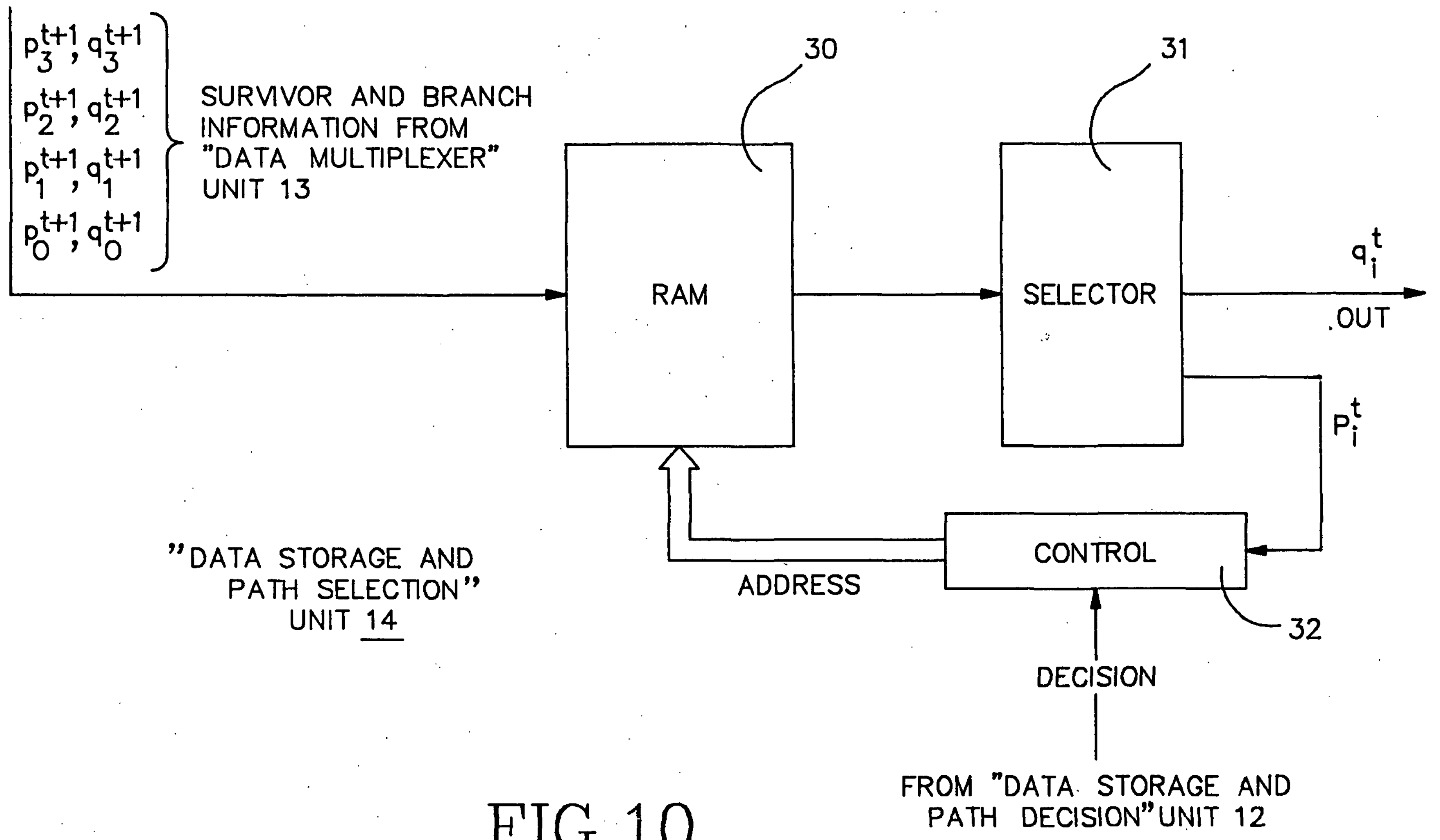

p_3^{t+1}, q_3^{t+1}
p_2^{t+1}, q_2^{t+1}
p_1^{t+1}, q_1^{t+1}
p_0^{t+1}, q_0^{t+1}
SURVIVOR AND BRANCH INFORMATION FROM "DATA MULTIPLEXER" UNIT 13
30
31
RAM
SELECTOR
q_i^t
OUT
P_i^t
CONTROL
32
ADDRESS
DECISION
"DATA STORAGE AND PATH SELECTION" UNIT 14
FROM "DATA STORAGE AND PATH DECISION" UNIT 12

FIG. 10

JPL Case No. 17310
NASA Case No. NPO-17310-1-CU
87/322

Serial No.	200, 874	
Filing Date	6/1/88	
Contract No.	NAS7-918	
Contractor	Caltech/JPL	PATENT
Pasadena	CA.	91109
(City)	(State)	(Zip)

METHOD FOR VITERBI DECODING OF LARGE CONSTRAINT LENGTH CONVOLUTIONAL CODES

Origin of the Invention

The invention described herein was made in the performance of work under a NASA contract, and is subject to the provisions of Public Law 96-517 (35 USC 202) in which the Contractor has elected not to retain title.

Technical Field

The invention relates to a pipeline VLSI architecture for a Viterbi decoder of convolutional codes to be used in future space missions for down link telemetry.

Background Art

A concatenated coding system, consisting of a convolutional inner code and a Reed-Solomon (RS) outer code, has been adopted as the guideline for down link telemetry for future space missions by the Consultative Committee for Space Data Systems. The (7, 1/2) convolutional inner code used by NASA's Voyager project with an 8-bit (255,223) RS outer code is capable of correcting up to 16 symbol errors. An investigation of the performance of this scheme was reported by R. L. Miller, L. J. Deutsch and S. A. Butman, "On the Error Statistics of Viterbi Decoding and the Performance of Concatenated Codes," JPL Publication 81-9, Jet Propulsion Laboratory, Pasadena, California, September 1, 1981, where it is shown that such a concatenated communication channel provides a coding gain of almost 2 dB over a convolutional code only channel at a decoded bit error rate of 10^{-5}. However, communication systems with higher quality may be needed for future space missions. These new space explorations may require a bit error rate (BER) of 10^{-6} at a signal-to-noise ratio (SNR) of about 0.5 dB. This means an improvement of almost 2 dB in coding gain is needed over that which is currently obtained in the Voyager system which uses a (7, 1/2) convolutional inner code and an 8-bit (255,223) RS outer code.

Several methods are possible to achieve higher performance. These include an increase of the spacecraft transmitter power, the antenna size, and coding complexity. However, a study by J. H. Yuen and Q. D. Vo, "In Search of a 2-dB Coding Gain," TDA Progress Report, 42-83, Jet Propulsion Laboratory, Pasadena, California, July-September 1985, has shown that the most cost effective means is to increase the coding complexity with a (15, 1/5), (14, 1/6), or (15, 1/6) convolutional code that is concatenated with a 10-bit (1023, 959) RS code. Thus to achieve this new performance requirement of 10^{-6} BER, it is necessary either to increase the constraint length (K=7) or to decrease the code rate (R=1/2) of the convolutional code as compared to the (7, 1/2) convolutional inner code of the Voyager system.

In 1967, a maximum likelihood decoding scheme was proposed by A. J. Viterbi, "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm," IEEE Trans. Inf. Theory, IT-13, pp. 260-269, April 1967. That scheme was relatively easy to implement for codes with small memory. Recently a Viterbi decoder on a single VLSI chip with constraint length K=7 was reported by J. B. Cain and R. A. Kriete, "A VLSI R=1/2, K=7 Viterbi Decoder," IEEE Proc. of the 1984 NAECON, May 1984. Considerable difficulty arises when one attempts, with the same architecture, to implement a Viterbi decoder with a large constraint length, for example, K=14. This is because the number of states in the Viterbi decoder trellis increases exponentially with the size of constraint length of the code used. For instance, for a code with constraint length K=7, the number of states in the decoder trellis is $2^6=64$, while for K=14, the number of states in the trellis increases dramatically to $2^{13}=8192$, i.e., while increasing the constraint length by a mere factor of two, the trellis states increases by a factor of 128. The latter is too large for the implementation of such a Viterbi decoder on a single chip with today's VLSI technology, unless a new architecture is found.

Recently, several Viterbi decoder architectures have been proposed for VLSI implementation. Examples include the systolic array method by C. Y. Chang and K. Yao, "Viterbi Decoding by Systolic Array," published in Proceedings of Allterton Conference, October 1985, and the parallel processor method by F. Pollara, "Viterbi Algorithm on a Hypercube: Concurrent Formulation," TDA Progress Report, 42-84, Jet Propulsion Laboratory, Pasadena, California, October-December 1985. However, each method has disadvantages which make it difficult for VLSI implementation. Though the parallel processor scheme of F. Pollara can have a high throughput rate, it suffers from the fact that the topological arrangement is extremely difficult to realize. On the other hand, the number of processors needed in the systolic array method of C. Y. Chang, et al., increases exponentially with the constraint length of the code used. Consequently, the chip area consumed in the implementation of a large constraint length Viterbi decoder becomes prohibitively large using the systolic array method.

One of the bottlenecks faced in today's VLSI designs is the limited number of input/output pins available on a VLSI chip. Since it is extremely difficult to put a large constraint length Viterbi decoder on a single chip, the partition of the whole decoder into several separate chips is generally unavoidable. The partitioning of the system into several separate chips, and at the same time not causing severe communication problems between the chips, is very difficult. This is a consequence of the large number of states involved in the Viterbi decoding algorithm so that each separate chip must have a substantial number of input/output pins for communicating among themselves. Unfortunately, the maximum number of pins currently available on a chip is limited to a few hundred. This is far less than needed for the implementation of a large constraint length Viterbi decoder using conventional methods. Therefore a decoding architecture which is easy to partition is needed.

Statement of the Invention

Accordingly, it is an object of this invention to provide a Viterbi decoding method which lends itself to architecture suitable for VLSI fabrication.

A further object is to provide a method for a Viterbi decoder of relatively large constraint length K, for example K=7, 11, or 14, which lends itself to architecture suitable for VLSI fabrication.

In accordance with the present invention, a Viterbi decoding method for a rate 1/n, where n is preferably equal to 2, and a large constraint length K, where K is an integer selected in a range of about 7 to 14, receives convolutionally encoded messages of serial binary digits and decodes them in blocks of NK where N is a selected integer, such as 5, using a "trace back" method by storing in a memory survivors, p_i^t, of partial path metrics m_i^t for each trellis node the survivors at each unit time, together with their estimated information bits, q_i^t. A decision is made every NK interval of the trellis data p_i^t, q_i^t by selecting from the block of stored information the lowest path metric. This selection is made at the time of the last time unit of the block NK based on the trellis data at the last time unit of the block. The trace back method is utilized while trellis data is stored for the first time unit of the next NK interval in a second block of memory to trace the p_i's from the last time unit of the first NK block to the first time unit of that first NK block. The p_i^t in that first time unit is then used to address the next trellis node of the selected path to read out the coded information q_i^{t+1}, and the p_i's, q_i's of subsequent nodes of the selected path metrics each address the next node of the next time unit until the q_i's of the NK block have all been read out. The same procedure is then followed on the second block stored while decoded information in the first block is read out. The process is continued for all subsequent NK blocks using the memory blocks alternately. The partial path metrics are computed for all the Viterbi trellis state nodes by computing new partial path metrics for all paths entering a node. That is done by adding a computed branch metric entering each node at the current time unit to the connecting metric from survivors at the preceding time unit. The partial path metrics of all paths entering each node are compared, and the path branch with the smallest partial metric is selected as the survivor for the current time unit. The estimated information bit q_i^{t+1} and the branch metric survivor p_i^{t+1} for each partial metric m_i^t then stored for reading out the correct decoded information bits after determining the survivor with the smallest metric at the end of NK partial metrics, i.e., the end of a block corresponding to NK encoded bits in the message received, at which time the "trace back" method is used to trace back to the beginning of the path selected which leads to the survivor p_i at the end of the NK block in order to read out decoded data in proper sequence.

The novel features that are considered characteristic of this invention are set forth with particularity in the appended claims. The invention will best be understood from the following description when read in connection with the accompanying drawings.

Brief Description of the Drawings

FIG. 1 is a block diagram of a convolutional encoder for R=1/2, K=3, G(D)=[1+D, 1+D+D^2].

FIG. 2 is the trellis diagram of a convolutional encoder for R=1/2, K=3, and G(D)=[1+D, 1+D+D^2].

FIG. 3 illustrates the trellis diagram of **FIG. 2** used for an example of decoding five 2-bit symbols by the architecture of the present invention using a Viterbi trellis decoder R=1/2, K=3.

FIG. 4 is a block diagram of a new architecture for decoding a convolutional code R=1/2, K=3.

FIG. 5 is a block diagram of a "data rearrangement" unit **11** for the architecture of **FIG. 4**.

FIG. 6 is a block diagram of a "metric computation and path decision" unit for the architecture of **FIG. 4**.

FIG. 7a is the partial trellis diagram where the branch labeled as "PM-i" corresponds to the path metric calculated by circuit labeled "PM-i" in **FIG. 6** at the first clock time.

FIG. 7b is the partial trellis diagram where the branch labeled as "PM-i" corresponds to the path metric calculated by circuit labeled "PM-i" in **FIG. 6** at the second clock time.

FIG. 8 is a block diagram of the path metric generating circuit of **FIG. 6**.

FIG. 9 illustrates schematically the "data multiplexing" unit **14** for the architecture of **FIG. 4**.

FIG. 10 illustrates a "data storage and path selection" unit **14** for the architecture of **FIG. 4** using a decoding block-by-block scheme.

Detailed Description of the Invention

A new Viterbi decoder architecture suitable for VLSI implementation will now be described in detail. It is expected that with this new architecture, a single chip implementation of a Viterbi decoder of relatively large constraint length is quite likely. If, on the other hand, the constraint length is too large to prohibit a single chip implementation of a Viterbi decoder, the ability to partition the system into several separate chips while at the same time maintaining the connectivity between chips is made easy by this new realization of the Viterbi algorithm.

Before proceeding with a description of the invention, the Viterbi decoding algorithm as described in S. Lin and J. Costello, "Error Control Coding," Prentice-Hall, New Jersey, 1983, is briefly reviewed for the purposes of exposition and illustration. The algorithm processes the received message in an interative manner. At each step (unit time), it compares the metrics of all paths entering each state, and then stores the path with the smallest metric, called the "survivor" together with its metric. The process can be summarized in the following steps:

Step 1. Begin at time unit j=m, where m is the length of the encoder memory. Compute the partial path metrics for each single path entering each state. Select the path having the smallest metric, called the survivor for that state. Store the survivor and its metric for each state.

Step 2. Increase the time unit j by 1. Compute the partial path metric for all the paths entering a state by adding the branch metric entering that state to the metric of the connecting survivor at the preceding time unit. For each state, store the partial path with metric m_i^t the smallest metric (the survivor), together with its metric, and eliminate all other paths.

Step 3. If j<L+m, where L is the length of information sequence, repeat Step 2. Otherwise stop and determine the path metric consisting of the survivor paths and determine from that the decoded sequence.

A difficulty with the algorithm arises when L is large, because the required storage becomes excessive, i.e., too large to be practical. Consequently, for large L, compromises must be made in the Viterbi decoding processes. The approach usually taken is to truncate the path memory of the decoder by storing only the most recent 5K blocks of information bits for each survivor, where K is equal to one constraint length of the code. After 5K time units, the decoder decodes the received message bit-by-bit. Another commonly used approach is to choose a most likely path for every 5K time units. In this approach, the algorithm decodes the received message block by block with each block equal to 5K bits for every 5K time units.

In the following, a simple example of convolutional encoding is given with reference to **FIGs. 1** and **2** for a rate 1/2, constraint length K=3, convolutional code with generator matrix:

$$G(D)=[1+D,\ 1+D+D^2]$$

where D is a unit delay operator indicated in **FIG. 1** by delay blocks **1**, **2** and **3**. At each time unit, one information bit enters into the first delay operator **1** and two message bits are generated by a plurality of Exclusive OR gates **4**, **5** and **6**, as shown in **FIG. 1** for the encoding trellis diagram of the rate 1/2, constraint length K=3 convolutional code.

As shown in **FIG. 2**, there are two branches leaving each node of the trellis illustrating nodes by circles arrayed in rows and columns. One branch shown in a solid line denotes a "zero" input to the next node and the other next branch, a dashed line, represents a "one" input to the node. The numbers on each of two branches represent the two bit outputs of the node for that particular time unit and input. For example, after two time units in **FIG. 2**, there are four nodes in rows "0", "1", "2" and "3" for each successive time unit "2", "3", "4" and "5". The states of the nodes in the respective rows are indicated on the left of the trellis as "00", "10", "01", and "11," which represent four states "0", "1", "2" and "3" in the encoder memory.

Assume that an information string of length L=7 shown in **FIG. 3** enters the encoder shown in **FIG. 1**, and that a code word from the encoder using the trellis of **FIG. 2** is transmitted over a binary symmetric channel (BSC). Next let the received message be:

10, 10, 10, 11, 00, 11, 11

FIG. 3 then shows the trellis diagram which illustrates decoding of the same rate 1/2, K=3 convolutional code with the received messages shown just below the time units at the top of the diagram. At each time unit of decoding, the difference between the received message and the symbols on the branch of the encoding trellis is adopted as the metric of that particular branch, and is called the Hamming metric.

In this example illustrated in **FIG. 3**, at the time unit zero the metric of the branch labeled "A" is 1 since there is one difference between the received message "10" and the number "00" on that particular branch at this time unit. Similarly, the metric of the other branch, branch "B", is also 1 because the received message "10" at this time unit differs in one bit position with the number "11" on this particular branch. The metrics of other branches on this diagram can be obtained in a similar fashion.

It should be noted that for all time units of the connecting branches for node "0" of state "00" are labeled 00, while all descending branches to node "1" of state "10" for all time units are labeled 11, the complement of the state "00." All upper connecting branches for nodes of state "10" to nodes of state "01" are similarly labeled 00, while the lower branches for connecting the nodes of state "10" to the nodes of state "01" are labeled 11. All connecting branches from nodes of either states "00" or "01" are either horizontal (the special cases of connecting two nodes in the "00" state or connecting two nodes in the state "11") or descending, but connecting branches from nodes of states "01" and "11" are always ascending. Adjacent to the two-bit state numbers are Arabic 0, 1, 2, and 3 which are sometimes used hereafter without quotation marks to refer to nodes at a particular time unit. The state numbers are also used hereafter without quotation marks.

For the nodes 2 in state 01 connecting to nodes 0 in the state 00, the connecting branches are labeled 01, and in the case of connecting to nodes 1 in state 10, they are labeled 10. And finally, for the nodes in state 11 connecting to nodes in states 01 and 11, the upper branch is labeled 10, and the lower branch is labeled 01. The lower connecting branches from any state node at one time unit to another state node at the next time unit are shown in dashed lines, while the upper connecting branches are shown in solid lines. At each state node for a preceding time unit t, the metric for the connecting branch is computed and added to the metrics of survivors at the preceding time unit before selecting the survivor path p_i^t for this time unit.

In review at time unit 1, the symbol **10** received in the time unit 1, following time unit 0 is compared with the symbol 00 in the branch labeled "A" to determine that the branch metric at the state number 00 is one because there is one difference in the bits. Similarly, the received symbol is compared with the symbol 11 in the branch labeled "B" to determine that the branch metric at the state number 10 at the time unit **1** is also one because it too has a difference of one in the bits. Computing the connecting branch metrics w(i,j,t) continues in a similar way as time progresses to time units 2 though 6. Note that after time unit 1 there are four nodes corresponding to the four state numbers 00, 10, 01 and 11. At each time unit, all connecting branch metrics are computed for all four state numbers (insides in the fields), and the connecting branch with the smallest cumulative metrics is selected, but not until after at least 5K. In this example illustrated in **FIG. 3**, the received message of 7 is within the constraint length of K=7, and therefore within 5K, but messages are almost always much longer. Consequently, it is necessary to store the partial path metrics at states i and time units t in order to determine the survivor path p_i^t at state i and time t for the decoding sequence of connecting branch metrics w(i,j,t) connecting state i to state j at time t, such as the branch connecting node 1 at time 1 to the node 2 at time 2.

These metrics of branches entering a node, when added to the metrics of the survivors at the preceding time unit, form new partial path metrics m_i^t of that particular node for the present time unit, such as the symbol m_i^t dendtes the partial path metrics in state i and time unit t. Also, p_i^t denotes the survivor path at state i and time unit t. The symbol w(i,j,t) represents a connecting branch metric, i.e., a metric of the branch connecting state i to state j at time t.

Taking the rate 1/2, K=3 convolutional code mentioned above for encoding (**FIG. 2**) and decoding (**FIG. 3**) as an example, the new partial path metric at node state 0, time unit 2, can be written as

$$m_0^2=\min\{m_0^1+w(0,0,2),m_2^1+w(2,0,2)\}=2+1.$$

Similarly, at time unit 2, the partial path metric at time unit 2, state 1 is

$$m_1^2=m_2^1+0=2+0=2.$$

The partial path metric at time unit 2, state 2 is

$$m_2^2=m_3^1+0=2+0=2.$$

Finally, the partial path metric at time unit 2, state 3 is

$$m_3^2=m_1^1+1=2+1=3.$$

In this example, at time unit 1 for m_i^1 equals 2 for state i=0, 1, 2, 3.

The same procedure continues until the condition specified at Step 3 in the summary of the process set forth above is met. Then the decoding process stops. Sometimes the information sequence is so long that it is impractical to store all the metrics and survivors. A feature of this invention is to make and store a decoding decision every 5T time units, where T time units equals one constraint length. The path with the smallest metric at every 5T time unit is selected and the information embedded in that particular path decoded. The decoding decision made in this way is no longer "maximum likelihood," but can be almost as good if T is not too small. The final sequence of survivor paths, v, for the above example are:

v=(1 1, 0 0, 1 0, 1 1, 0 1, 1 0, 1 1)

is shown as the highlighted path in **FIG. 3.** Thus, the decoded sequence of binary digits is determined to be u=(1 0 1 1 1).

Method and Apparatus for Viterbi Decoding According to the Present Invention

Referring to **FIG. 4**, the method and apparatus for a Viterbi decoder **10** of convolutional codes with large constraint length according to the present invention will now be described with reference to **FIG. 4**, et seq. Basically this

Viterbi decoder executes the conventional Viterbi decoding process described above, but with several new techniques employed to realize each decoding step of the Viterbi algorithm in VLSI architecture that is simple, regular, expandable, and naturally suitable for VLSI implementation.

This resulting new architecture is best described by an example. The example used here is the same rate 1/2, K=3, convolutional code given in the review above of the conventional Viterbi process. Consequently, the same final sequence of survivor paths shown in **FIG. 3** with heavy lines apply. **FIG. 4** depicts the block diagram for the architecture of this improved Viterbi decoder divided into four major blocks **11**, **12**, **13** and **14**. Each is described below in sequence.

Data Rearrangement Unit 11

This unit shown in **FIG. 5** rearranges the order of data fed back serially from the "data multiplexing" unit **13**. They are the partial path metrics m_0^t, m_1^t, m_2^t, and m_3^t of nodes 0, 1, 2 and 3 at time unit t needed to process metric data for time t+1. The ouputs of this unit **11** are the rearranged parallel metrics as shown in **FIG. 5** which shows the block diagram of the "data rearrangement" unit.

Referring to **FIG. 5**, the "data rearrangement" unit consists of four shift registers, namely SR-1, SR-2, SR-3 and SR-4. These registers are of length 2^{K-2}, where K is the constraint length of the convolutional code. In this example they are of length $2^{3-2}=2$ since K=3. Initially, these registers are reset to zeros. Among these four registers, SR-1 and SR-4 are operated at twice the clock rate of the system's master clock as described in the following. The first two sequential outputs from the "data multiplexing" unit **13** at time unit t, i.e., m_0^t and m_1^t are shifted serially into register SR-1 under control of a signal $\overline{\text{FEED}}$ at a clock rate twice the rate of a master clock rate. In that manner, m_0^t and m_1^t are fed serially into the register SR-1 in one master clock period. Its contents are then transferred in parallel into register SR-2 and latched there at the end of the master loading clock cycle under control of a signal LOAD. During the next master clock cycle, register SR-4 is fed sequentially the second half output from the "data multiplexing" unit, i.e., m_2^{t+1} and m_3^{t+1}, at the time unit t+1 under control of a FEED signal. As soon as SR-4 is full, all of its contents are loaded into SR-3 in parallel under control of a $\overline{\text{LOAD}}$ signal. It is evident that the FEED and LOAD control signals are squarewave signals generated out of phase 180° by the master clock. Once both of the registers SR-2 and SR-3 are loaded, are shifted out serially to the input of a "metric computation and path decision" unit **12** shown in **FIG. 6** under control of a SHIFT SIGNAL.

Metric Computation and Path Decision Unit 12

The purpose of this unit shown in **FIG. 6** is to compute the new partial path metric m_i^{t+1} for all the Viterbi trellis paths entering a node or state by adding the computed branch metrics q_i^{t+1}, which has the encoded data bits embedded in it, entering that state to the metric of the connecting survivor at the preceding time unit. The partial path metrics m_i's of all paths entering each state node are compared, and the path with the lowest partial metric, called the survivor p_i, together with the branch metric q_i are selected while other partial paths and branch metrics are eliminated.

As shown in **FIG. 4**, inputs to this unit **12** are partial metrics m_0^t, m_1^t, and m_2^t, m_3^t at time unit t from the "data rearrangement" unit **11** shown in **FIG. 5**. Outputs of this unit **12** are:

$$m_0^{t+1},\ p_0^{t+1},\ q_0^{t+1}$$

$$m_1^{t+1},\ p_1^{t+1},\ q_1^{t+1}$$

$$m_2^{t+1},\ p_2^{t+1},\ q_2^{t+1}$$

$$m_3^{t+1},\ p_3^{t+1},\ q_3^{t+1}\ ,$$

where m_0^{t+1}, m_1^{t+1}, m_2^{t+1} and m_3^{t+1} represent the new partial path metrics at time unit t+1 of state nodes "0", "1", "2", and "3", respectively; p_0^{t+1}, p_1^{t+1}, p_2^{t+1} and p_3^{t+1} represent the survivors of state nodes "0", "1", "2" and "3" at time unit t+1, respectively; and q_0^{t+1}, q_1^{t+1}, q_2^{t+1}, q_3^{t+1} are branch metrics that correspond to the estimated information bits at the time unit t+1.

FIG. 6 illustrates the "metric computation and path decision" unit **12** for the rate 1/2, K=3 Viterbi decoder which contains a set of four path metric generating circuits labeled PM-0, PM-1, PM-2 and PM-3. These circuits are used to generate the partial path metrics of each node state at the same time unit t+1 in the trellis diagram. For the first clock cycle, PM-0 and PM-2 generate the two competing path metrics 00 and 11 entering into node state "0" at time t+1, as shown in **FIG. 3**. These two path metrics are compared in the metric-comparison circuit **23** shown in **FIG. 6**. The path with the lowest path metrics is saved in the comparison circuit as the new path metrics of the state node "0" for the time t+1, while the other one is discarded. At the same instant, PM-1 and PM-3 generate the other two competing path metrics entering node state "1" at time t+1. Their outputs are also sent to the metric-comparison circuit **24** compared and selected there. In **FIG. 7a**, the branch labeled as "PM-0" denotes that the PM-0 circuit in **FIG. 6** is used to generate its partial path metric. The other branches in **FIG. 7a** are similarly labeled as being used to generate its partial path metric m_2^{t+1}. At the next clock time shown in **FIG. 7b**, PM-0 and PM-2 generate the two competing path metrics entering node state "2" at time unit t+1. Also PM-1 and PM-3 generate the two competing path metrics entering node "3".

It should be recalled that the "data rearrangement" unit **11** (shown in **FIG. 5**) divides the partial metrics m_0^t, m_1^t, m_2^t and m_3^t for nodes 0, 1, 2 and 3 into pairs m_0^t, m_1^t and m_2^t, m_3^t so that the path metric computation circuit receives m_0^t and m_2^t to compute m_0^{t+1} and m_1^{t+1}, and then receive m_1^t and m_3^t. It is in that manner that four path metric generating units PM-0, PM-1, PM-2 and PM-3 divided into pairs of two are able to take first two path metrics m_0^t, m_2^t to generate two partial path metrics and then m_1^t and m_3^t to generate two more partial path metrics, for a total of four partial path metrics m_0^{t+1}, m_1^{t+1}, m_2^{t+1} and m_3^{t+1}.

As shown in **FIG. 6**, PM-0 and PM-1 share a two-bit binary counter **21** and PM-2 and PM-3 share another two-bit binary counter **22**. The contents of these counters represent the states in the trellis diagram. Therefore, the change in the counter states resembles the change of state on the trellis diagram. At the start of a time unit, the 2-bit binary counter **21**, shared by circuits PM-0 and PM-1, is reset to 00 which corresponds to state "00" at time unit 0 in the trellis diagram. At the first clock cycle, the branch metrics of the branch from node 0 in time unit t to node 0 in time unit t+1 is computed by the metric generating circuit PM-0. The metric of the branch from node 0 in time unit t to node 1 in time unit t+1 is computed by the metric generating circuit PM-1 at the same time.

The same procedure applies to metric generating circuits PM-2 and PM-3, except that the 2-bit binary counter **22** shared by these circuits are initially set to "10" which corresponds to state "10" in the trellis diagram. The metric of the branch from state node 1 in time unit t to node 0 in time unit t+1 is computed by the metric generating circuit PM-3. The metric of the branch from node 1 in time unit t to node 2 in time unit t+1 is computed by the metric generating circuit PM-4.

The contents of the counters **21** and **22** are incremented by one each master clock cycle. Thus, at the next master clock time, the counter **21** shared by PM-0 and PM-1 is changed to 01 which corresponds to state node 1 in the trellis diagram and the counter **22** shared by PM-2 and PM-3 is changed to 11 which corresponds to state 4 in the trellis diagram. The contents of the counters **21** and **22** are then shifted to their associated circuits PM-0, PM-1 and PM-2, PM-3.

When the content of the counter **21** is shifted into the encoder circuit PM-0, a fixed 0 is appended in the least significant bit (LSB) position, while a fixed 1 is appended in the LSB of the PM-1 circuit. The outputs from these circuits are sent to metric comparison circuits **23** and **24** as shown in **FIG. 6** where the branch metrics are calculated by the use of both the outputs from these circuits and the received messages per unit time. These branch metrics, when added to the previous partial path metrics to form a new partial path metric at time unit t+1, are sent to the metric-comparison circuits **23** and **24** where the survivor of each state is selected.

Since two metric-comparison operations are to be performed at one master clock, two metric-comparison circuits **23** and **24** are needed as shown in **FIG. 6**. The path with the lowest metric is selected by these metric-comparison circuits and sent out. Finally, the purpose of the path metric comparison circuit **25** in **FIG. 6** is to compare and select the path with the lowest metric every 5T time units when, in accordance with this invention, a decoding process is to be carried out in the next 5T time interval.

As shown in **FIG. 8**, each of the partial path metric generating circuits PM-0, PM-1, PM-2 and PM-3 consists of an encoder **26** consisting of two Exclusive OR gates, a "branch metric compute" circuit **27** and an adder **28**. The function of the encoder **26** in combination with the" branch metric compute" circuit **27** in any one of the partial path metric generating circuits **PM-0**, **PM-1**, **PM-2** and **PM-3** is essentially the same as the encoder **4** used in the transmitter. They are used to generate the numbers 00, 10, 01 and 11 of the connecting branch for each time unit on the trellis. These numbers, together with the received messages, are input to the branch metric compute circuit **27** to compute the branch metrics $q_0^{t+1}, \ldots q_3^{t+1}$. The computed branch metric is added by the adder **28** to the previous partial path metric in order to form a new partial path metric. The two path metrics from **PM-0** and **PM-2** are sent to the "metric comparison" unit **23**, as shown in **FIG. 6**, so that the survivors of nodes 0 and 1 can be selected at this time slot. Similarly, the two path metrics from **PM-1** and **PM-3** are sent to another "metric comparison" unit **24** in **FIG. 6** to select the survivors of nodes 2 and 4.

Both the path information and the partial path metric of the selected path are sent to the "data multiplexing" unit **13**. These new partial path metrics are also fed to a "path metric comparison" circuit **25** in **FIG. 6** whenever the decoding decision is needed, i.e., every 5T times in this exarple of the present invention. The time lag needed to perform the decoding process is usually chosen as 5T where T equals one constraint length. Thus, the path with the lowest metric is selected every 5T interval. This path information is sent to the "data storage and path selection" unit **14** where they are decoded as outputs. The details of this "data storage and path selection" unit **14** will be discussed after the "data multiplexing" unit **13**.

Data Multiplexing Unit 13

As shown in **FIG. 4**, the output data of the "metric computation and path decision" unit **12** are obtained in pairs. That is, at the first clock time, m_0^{t+1}, p_0^{t+1}, q_0^{t+1} of node "0" and m_1^{t+1}, p_1^{t+1}, q_1^{t+1} of node "1" are available. At the next clock, m_2^{t+1}, p_2^{t+1}, q_2^{t+1} of node "2" and m_3^{t+1}, p_3^{t+1}, q_3^{t+1} of node "3" are available.

The purpose of this "data multiplexing" unit **13** is to change the two parallel outputs of the "metric computation and path decision" unit **12** (**FIG. 6**) to sequential order. This may be achieved by using a switch operating at twice the rate of the master clock as shown in **FIG. 9** so that both of the two sets output data from the "metric computation and path selection" unit **13** can be sampled adequately. Thus, a system's master clock with a fixed period is divided into two switch clocks with half the period of the master clock. For the first switch clock period, the path metric of node 0, i.e., m_0^{t+1}, is sampled and sent to the "data rearrangement" unit **11** and at the same time p_0^{t+1} and q_0^{t+1} are sent to the "data storage and path selection" unit **14**. For the second switch clock period, m_1^{t+1} of node "1" is sampled and sent to the "data rearrangement" unit **11** while p_1^{t+1} and q_1^{t+1} are sent to the "data storage and path selection" unit, and so on. The output sequence of the "data multiplexing" unit **13** is then rearranged in the node order of "0," "1," "2" and "3".

Data Storage and Path Selection unit 14

This unit shown in **FIG. 10** stores both the surviving paths p_i's and the estimated information bits q_i's in a RAM **30**. It is shown in Lin, et al., supra, that the decoding decision can be made after 5K. The q_i's stored in the "data storage" unit are then read out sequentially which is the most likely information for this particular received message.

A "trace back" scheme is used in this design due to its simplicity in hardware implementation. The concept of the trace back method is to store the most recent candidate bit of the hypothesized information sequence for each state at each decoding stage. After several stages have been processed, the whole sequence is constructed in the reverse order.

FIG. 10a shows a block diagram of the "data storage and path selection" unit **14** where the received messages are to be decoded block by block with each block of lengths equal to 5K. This unit is organized as an array of memory cells, such as in the RAM **30**, a selector **31** and a control register **32**. The array of memory cells may be organized into rows and columns. The number of rows equals the number of states in the trellis. As shown in **FIG. 3**, the rows are numbered in accordance with the number of nodes in the trellis diagram, namely 0, 1, 2 and 3.

Each memory cell is for storing both the p_i's (surviving paths) and the q_i's (information bits). The elements in the same column of the following table represents the p_i's and q_i's of the same time unit in the RAM.

	t+6	t+5	t+4	t+3	t+2	t+1	t
"3"	X	"2",1	X	X	X	X	X
"2"	"0",1	X	X	X	"0",1	X	X
"1"	X	X	"3",0	X	X	"2",0	X
"0"	X	X	X	"1",0	X	X	"1",0

The numbers with quotation marks denote the node number of the previous time unit of the same path. The other number represents the information bit (q_i). Paths in the RAM not selected are indicated with x's for the sake of clarity. The data from the "data-multiplexing" unit **13** (**FIG. 9**) is fed into this RAM sequentially, column by column from right to left. Each time the right-most column of the data storage array has been filled, the RAM address is shifted left one column to accept the next set of data. While this is processed, a similar array is filled. Then the roles of the two arrays are switched. After 5T clock cycles, the output from the path metric comparison circuit **25** shown in **FIG. 6** is sent to the control

32 unit 12. These outputs represent the node number with the lowest metric among all the paths at the end of time unit 5T, at the time the 5K block of metric data has been stored.

The control includes a K-1 bit register. In this example, it is a K-1=2 bit register. The output of the control register is sent to the selector and used as the address to select the next p_i and q_i and read them out in sequence. The survivor p_i is fed back to the control 32 to perform the selection of the next set of data while q_i is sent out as a decoded information bit.

For an example, assume that the sequence of data to be read out is 1100100, as shown in the table below, and that node "0" is the initial node of the path with the lowest metric selected from the path metric comparison circuit 25 in the "metric computation and path decision" unit 12. The number 0 which is 00 in binary representation, is loaded into the control 32 to address the selector. The content in the first column on the right in row zero, where p_i equals 1 and q_i equals 0, is selected and read out. The number 0 is the first information bit. The number 1 is then loaded into the control 32 and used as the next address so that the second information bit 0 is shifted out as the decoded bit. At the same time, the control 3 shifts the address to the left one column to read out the content of node 2 under t+2, i.e., three columns over to the left in the block of memory cells. This process continues until the last column of information is read out. The output string of decoded bits 1100100 is thus the desired output in the example.

A new architecture for a VLSI implementation of a Viterbi decoder has been presented. The distinct features of this new Viterbi decoder can be summarized in the following:
(a) A single processor is used to iteratively compute partial path metrics. Each time unit is divided into several smaller time slots, and in each time slot n new partial metrics of n

different nodes at the same time unit are computed by this single processor. In general, the number of time slots in a time unit is 2^{k-2}. For example, since this is a rate 1/2 code, the new partial path metrics of nodes 0 and 1 are calculated in the first time slot of a unit time. At the second time slot, the same processor is used to compute the new partial path metrics of nodes 2 and 3. This process thus completes path metrics of four nodes in a unit time. See **FIG. 3.**

It is recognized that a Viterbi decoder with this single-processor architecture might not operate as fast as a parallel-processor architecture of a Viterbi decoder. However, this new architecture is the only feasible way to implement a large constraint length VLSI Viterbi decoder. However, this single-processor architecture can be easily extended to include several processors together to so compute the path metrics as to increase the decoding speed if required.

(b) A number n of circuits are used to compute the partial metrics of n nodes at the same time unit. This number of circuits is needed because n branch metrics are computed simultaneously to generate the numbers on each branch of the trellis diagram. **FIG. 3** shows a trellis for a rate 1/2 code. Since, in this design, the Hamming metric is adopted as the branch metric, the branch metric is just the difference between the numbers on each branch in the trellis diagram and the received message at that particular time unit.

(c) Because a single processor is used to compute the partial path metric, the partial path metrics at the preceding time unit must be fed back to the processor and added to the newly computed branch metric to form a new partial path metric of each node in the trellis diagram. This may cause data congestion problems because, as mentioned above, n new partial path metrics are obtained in one time slot. They are fed back to the processor one by one sequentially. It is necessary that this feedback operation be finished in one time slot. Hence a multiplexer is needed to convert these n parallel data into n serial data at the output of this processor. This means that a switch, acting as the multiplexer, must operate n times faster than the system's master clock speed to meet this requirement. The system's master clock speed equals the reciprocal of one time slot, as defined above. Furthermore, since at each time slot the processor takes two parallel data input to compute the partial path metrics, a buffer is required to convert the n serial data output from the multiplexer back to n parallel data. The operating speed of this buffer must be n times faster than the system's master clock rate. Therefore, the buffer will have enough time to convert the serial data type into parallel data type in one master clock time.

(d) The "trace back" scheme is used in this architecture to select the most likely path. An array type of storage is used to both store the survivors and perform the path selection operation. In other words, the necessary storage is arranged in an array of columns and rows so that each row corresponds to a node in the Viterbi trellis diagram, and each column corresponds to a time unit t. Elements in the same column are the survivors and estimated information bits of the same time unit. This storage operates in such a manner that the output from the processor is first fed into the left-most column of this unit serially. In the preferred embodiment illustrated, the length of this storage unit is 5K which is sufficient to store all the survivors for a constraint length K. The depth of this storage is equal to the number of states in the trellis, i.e., 2^{k-1}. Therefore, this storage unit consists of 5K times 2^{k-1} storage cells. In the decoding procedure, if the decision is to be made to select the most likely path in one time interval of 5K which equals 5T, the last state of this path is determined first by comparing the metrics of every node at the last time unit. This information is sent to this storage unit to select one of the cells in the right-most column which represents the first time unit t of this path. Its content is read out and decoded. At the same time, control is shifted to read the designated node in the column of t+1. This output is fed back and used as the next address to select the designated node in the next node t+2. The selected content is then read out. This operation continues until all the columns of this storage of constraint length 5K are read out and decoded.

It is estimated that with this new architecture, a Viterbi decoder for the rate 1/2, constraint length K=11 convolutional code needs about 150,000 transistors for VLSI implementation. This is within the capability for a single-chip implementation of the decoder with current VLSI technology. Furthermore, due to the sequential processor structure used in this architecture, it is not too difficult to partition the whole decoder into several chips if the constraint length is too large for a single-chip implementation. This may lead to the implementation of a Viterbi decoder of convolutional codes with constraint length K=14. Finally, if the decoding rate of this new Viterbi decoder is not fast enough, several processors can be used in parallel to enhance the decoding rate.

Although particular embodiments of the invention have been described and illustrated herein, it is recognized that modifications and variations may readily occur to those skilled in the art. Consequently, it is intended that the claims be interpreted to cover such modifications and variations.

METHOD FOR VITERBI DECODING OF LARGE
CONSTRAINT LENGTH CONVOLUTIONAL CODES

Abstract of the Disclosure

A new method for Viterbi decoding of convolutional codes lends itself to a pipeline VLSI architecture using a single sequential processor to compute the path metrics in the Viterbi trellis. An array method is used to store the path information for NK intervals where N is a number, and K is constraint length. The selected path at the end of each NK interval is then selected from the last entry in the array. A "trace-back" method is used for returning to the beginning of the selected path back, i.e., to the first time unit of the interval NK to read out the stored branch metrics of the selected path which correspond to the message bits. The decoding decision made in this way is no longer "maximum likelihood," but can be almost as good, provided the constraint length K is not too small. The advantage is that for a long message, it is not necessary to provide a large memory to store the trellis derived information until the end of the message to select the path that is to be decoded; the selection is made at the end of every NK time units, thus decoding a long message in successive blocks.