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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
21 March 2002 (21.03.2002)

PCT

(10) International Publication Number
WO 02/23740 A1

(51) International Patent Classification[7]: H03M 13/27, H04L 1/00

(21) International Application Number: PCT/US01/22807

(22) International Filing Date: 19 July 2001 (19.07.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/232,224 13 September 2000 (13.09.2000) US
60/260,930 11 January 2001 (11.01.2001) US

(71) Applicant: INTERDIGITAL TECHNOLOGY CORPORATION [US/US]; 300 Delaware Avenue, Suite 527, Wilmington, DE 19801 (US).

(72) Inventor: SHAHRIER, Sharif, M.; 640 American Avenue, Apt. E101, King of Prussia, PA 19406 (US).

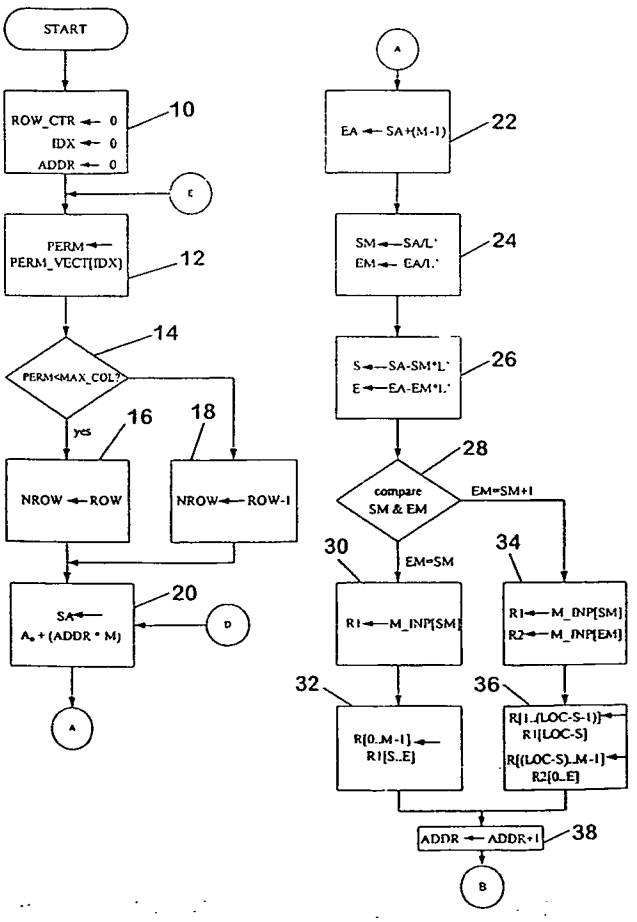
(74) Agents: VOLPE, Anthony, S. et al.; Volpe and Koenig, P.C., Suite 400, One Penn Center, 1617 John F. Kennedy Boulevard, Philadelphia, PA 19103 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) Title: THIRD GENERATION FDD MODEM INTERLEAVER



(57) Abstract: A method and apparatus are disclosed for deinterleaving expanded interleaved data blocks, particularly for use in a wireless telecommunications systems such as provided by the Third Generation Partnership Project (3G) standard. The data is processed on a sequential element basis where each element has a pre-determined number of bits M which bits are contained in a block of sequential data world W'. The elements are extracted from the block of words W' in sequential order, each element being extracted form either a single or two sequential interleaved words within the set of words W'. The elements are stored in selective location within a set of words W of a deinterleaver memory such that upon completion of the extraction and writing of all the elements, the words W from the deinterleaver memory can be sequentially read out to correspond to an original data block of bits from which the block of interleaved elements was created. Additional conventional processing results in the contraction of the deinterleaved expanded words to reproduce the data block of bits in a receiver as originally designated for transmission in a transmitter.

WO 02/23740 A1

Published:

— *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*

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.

[0001] **THIRD GENERATION FDD MODEM INTERLEAVER**

[0002] The present application relates to interleaving of data in a telecommunications system. In particular, method and apparatus for de-interleaving data.

[0003] BACKGROUND

[0004] It is known in the wireless telecommunications art to scramble data through a process known as interleaving for transmitting the data from one communication station to another communication station. The data is then de-scrambled through a de-interleaving process at the receiving station.

[0005] In Third Generation Partnership Project (3G) wireless systems, a specific type of data interleaving for frequency division duplex (FDD) modems physical channel data is specified. Physical channel data in a 3G system is processed in words having a pre-defined bit size, which is currently specified as 32 bits per word.

[0006] Blocks of arbitrary numbers of sequential data bits contained within sequential data words are designated for communication over FDD physical channels. In preparing each data block for transmission over the channel, the data is mapped row by row to a matrix having a pre-determined number of columns. Preferably there are fewer columns than the number of bits in a word. Currently 30 columns are specified in 3G for physical channel interleaving of data bit blocks contained in 32 bit words.

[0007] For example, a mapping of a 310 data bit block contained as bits $w_{0,0}$ - $w_{9,21}$ within ten 32-bit words w0-w9 to a thirty column matrix is illustrated in Fig. 1. The 310 data bit block is mapped to a 30 column matrix having 11 rows. Since the data block has a total of 310 bits, the last twenty of the columns, columns 10-29, include one fewer data bit then the first ten columns, 0-9.

[0008] Whether or not all of the matrix columns have bits of data mapped to them is dependent upon the number of bits in the block of data. For example, a block of 300 data bits would be mapped to a 30 x 10 matrix completely filling all the columns since 300 is evenly divisible by 30. In general, for mapping a block of T elements, the last r columns of a C column by N row matrix will only have data in N-1 rows where r = (C*N) -T and r ‹ C.

[0009] After the data bits have been mapped to the interleaver matrix, the order of the columns is rearranged in a pre-defined sequence and the data bits are written to a new set of words w' on a column by column sequential basis to define an interleaved data block of sequential bits $w'_{\#,\#}$ in a set of sequential words w'.

[0010] For example, the 310 bit block of data contained in words w0 - w9 of Figure 1 is selectively stored to words w'0 - w'9, in accordance with the preferred interleaver column sequence as shown in Figs. 2a, 2b. For the set of words, w0 - w9, the corresponding interleaved block of ten words w'0 - w'9 contain all of the 310 bits of data of the original words w0 - w9 in a highly rearranged/scrambled order. As shown in Figure 2a, interleaved word w'0 is formed of a sequence of bits from columns 0, 20 and

10 of Figure 1. The correspondence of the bits $w_{\#,\#}$ from the original words w0 - w9 to the bits $w'_{0,0}$ - $w'_{0,31}$ of interleaved word w'0 is illustrated in Fig. 2b.

[0011] In 3G, various processes occur concerning the interleaved data before it is transmitted to a receiving station. In order to increase the signal to noise ratio, the bit size structure is expanded M number of times. Currently the bit expansion is specified as six fold. Accordingly, each of the interleaved data bits for a block of physical channel data in a 3G system is expanded to a six bit element.

[0012] By way of example, the ten interleaved data words w'0 - w'9 of the example of Figures 2a and 2b are expanded into a block of 59 words W'0 - W'58 for transmission as reflected in Figure 3. Figures 4a-4f illustrate an example of the correspondence of the interleaved bits $w'_{0,0}$ - $w'_{0,31}$ of word w'0 to expanded interleaved six-bit elements T'0 - T'31 of words W'0 - W'5.

[0013] Since the element bit size does not evenly divide into the word bit size, some elements span two sequential words. For example, in Figures 4a and 4b, element T'5 is partially contained in word W'0 and partly contained in the next word W'1.

[0014] In the receiving station, after reception and processing, the received block of expanded interleaved elements, for example, the bit $W'_{0,0}$ - $W'_{58,3}$ in the 59 words W'0 - W'58, must be deinterleaved, i.e. descrambled, to reassemble the data in its original sequential form. It would be highly advantageous to provide a method and apparatus for deinterleaving of expanded column interleaved data blocks in a fast and efficient manner.

[0015] SUMMARY

[0016] A method and apparatus are disclosed for deinterleaving expanded interleaved data blocks, particularly for use in a wireless telecommunications system such as provided by the Third Generation Partnership Project (3G) standard. The data is processed on a sequential element basis where each element has a pre-determined number of bits M which bits are contained in a block of sequential data words W'. The elements are extracted from the block of words W' in sequential order, each element being extracted from either a single or two sequential interleaved words within the set of words W'. The elements are stored in selective location within a set of words W of a deinterleaver memory such that upon completion of the extraction and writing of all the elements, the set of words W from the deinterleaver memory can be sequentially read out to correspond to an original data block of bits from which the block of interleaved elements was created. Additional conventional processing results in the contraction of the deinterleaved expanded words to reproduce the data block of bits in a receiver as originally designated for transmission in a transmitter.

[0017] Although the method and apparatus were specifically designed for a 2nd de-interleaving function of a 3G FDD receiver modem, the invention is readily adaptable for either scrambling and descrambling expanded data blocks for other applications.

[0018] Preferably, a multi-stage pipeline configuration is employed to process the elements in conjunction with calculating a deinterleaver memory address and selective storage of the data elements therein. Data throughput of up to 60 megabits per second has been realized using a preferred three stage pipeline. Also, multiple deinterleavers may

be used parallel to process multiple blocks of data, each, for example, for a group of different physical channels, so that the deinterleaving process does not adversely impact on the overall speed of the communications system. However, since the physical channel processing of each channel is currently specified as 380 kilobits per second, the speed of a single deinterleaver in accordance with the preferred construction is more than adequate to process the data element blocks of all of physical channels of a 3G FDD receiver modem.

[0019] Other objects and advantages of the present invention will be apparent to those of ordinary skill in the art from the drawings the following detailed description.

[0020] BRIEF DESCRIPTION OF THE DRAWINGS

[0021] Figure 1 illustrates a mapping of a block of 310 data bits contained in ten 32-bit words w upon a thirty column matrix.

[0022] Figure 2a illustrates a mapping of the data bit block of Figure 1 onto a block of interleaved bits $w_{\#,\#}$ of words w' in accordance with a current 3G interleaver column sequence specification.

[0023] Figure 2b illustrates a bit mapping for one interleaved word w' from bits of data words w of Figure 1.

[0024] Figure 3 illustrates an expansion mapping of the interleaved bit block words w' of Figure 2a onto an expanded set of interleaved six-bit element words W'

[0025] Figures 4a-4f illustrate a bit mapping of one of the interleaved bit block words w' of Figure 2a onto a set of six expanded element interleaved words W'.

[0026] Figures 5a and 5b illustrate a mapping of the bits of the block of expanded interleaved elements of words W' of Figure 3 onto an interleaver matrix of thirty six-bit element columns.

[0027] Figure 6 illustrates bit and element mapping for one word W of the deinterleaved element block of data on the matrix of Figures 5a and 5b.

[0028] Figures 7a and 7b illustrate a corresponding deinterleaved expanded element and bit mapping of the matrix of Figures 5a and 5b.

[0029] Figure 8 illustrates the correspondence of the deinterleaved expanded element words W of Figure 7 with the original data bit block words w of Figure 1.

[0030] Figure 9 is a block diagram of receiver processing components of a communication system which utilizes the current invention.

[0031] Figures 10a and 10b are a flow chart of a general method of deinterleaving in accordance with the present invention.

[0032] Figures 11a - 11c are a schematic diagram of a three stage pipeline interleaver in accordance with the present invention.

[0033] DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0034] As part of the current 3G specification, blocks of expanded interleaved data, for example, data for physical channels of an FDD receiver are received and must be deinterleaved for further processing. The FDD receiver is divided up into a number of sub-blocks. One of these blocks is called the *Receiver Composite Channel (RCC)*. The RCC block diagram is shown in Figure 9. It consists of physical channel de-mapping, 2nd

de-interleaving, physical channel aggregation, 2nd stripping of DTX and P indication bits and Transmit Channel (TrCH) demultiplexing. Effectively, the receiver composite channel operations are opposite to functions performed by a transmitter modem in a transmitter composite channel.

[0035] The present invention is particularly useful for the architecture of the 2nd de-interleaver of an FDD receiver. The bit sequence to be transmitted for each physical channel (PyCH) is scrambled through an interleaver process and then expanded into equal sized packets; each packet consisting of a small number M of bits. Each of these groups of bits is termed a *data element.* Current, 3G FDD physical channel data element size is specified as six bits, i.e. M = 6 in the preferred embodiment. Figures 1-4 illustrate an example of the transmitter modem interleaving and expansion of a 310 data bit block into a block of 310 interleaved six-bit elements T'.

[0036] The expanded, interleaved data elements are transmitted in their interleaved sequence. The receiver receives the data elements over the air, and stores them in a set of sequential 32-bit data words W'. In the example of Figures 1-4, the data block of 310 bits initially stored in 32-bit words w0-w9 on the transmitter side is received and stored as data elements T'0-T'309 in 32-bit words W'0-W'58 on the receiver side.

[0037] The 2nd interleaver is a block interleaver with inter-column permutations which resequences the interleaved data elements. The interleaving matrix has 30 element columns, numbered 0, 1, 2, ..., 29 from left to right. The number of rows is provided by the user as an external parameter *N*, but can be calculated for a data block having T elements as the least integer N such that $N*30 \geq T$.

[0038] The inter-column permutation pattern for the 2nd de-interleaver for a 3G FDD modem is as follows:

Number of columns	Inter-Column Permutation Pattern
30	{0,20,10,5,15,25,3,13,23,8,18,28,1,11,21, 6,16,26,4,14,24,19,9,29,12,2,7,22,27,17}

Table 1 - Inter-Column Permutation Pattern for De-Interleaver

[0039] The output of the 2nd de-interleaver is a bit sequence read out row by row from a mapping to the inter-column permuted N x 30 matrix. Where the entire N x 30 matrix is output, the output is pruned by deleting bits that were not present in the input bit sequence of data elements.

[0040] Figures 5a and 5b illustrate a bit mapping of the example received data elements T'0 - T'309 of left and right portions of an 11 row by 30 element column interleaver matrix. In Figure 5a, for example, column 0 reflects the bit mapping of the bits of elements T'0 - T'10 which are contained in words W'0, W'1 and W'2. The bits of element T'5 are extracted from two words, W'0 and W'1; the bits of element T'10 are extracted from two words, W'1 and W'2. In Figure 5b, the bottom row has no elements since only the first ten columns are completely filled by the data elements.

[0041] Figures 6 and 7 reflect how the elements T' are reordered through the selected storing of the elements in a set of words W based on the interleaver matrix mapping. Thus, T'0, T'124, T'258, T'186 and the first two bits of T'31 are stored in the 32 bits of word W0 which, accordingly, correspond to reordered elements T0 through T4 and the first two bits of element T5. As a result of the selective storage of the elements

T'0 through T'309 based on the interleaver matrix mapping, a series of 32 bit words W, W0 through W58 is formed containing reordered elements T0 through T309 as shown in Figures 7a-7c. Figure 8 reflects how the original word w0-w9 correspond to words W0-W58 illustrating the correspondence of the reordered elements T0-T309 with the 310 original data block bits $w_{0,0}$ - $w_{9,21}$, shown in Figure 1.

[0042] In order to properly place the elements T'0 - T'309 in the matrix so that the elements T'0 - T'309 can be read out row by row in sequential words W0-W58, each element T' is selectively processed as reflected in the flow charts of Figures 10a and 10b.

[0043] In the 3G FDD modem receiver, expanded, interleaved data is separated into different physical channels and stored in a random-access-memory (RAM) named M_INP for processing by the deinterleaver. The bit stream is segmented into words of 32-bits, and the words are placed into contiguous locations in M_INP. In the example of Figures 1-4, the bit stream for elements T0 - T309 which are contained in the words W'0-W'58 would be stored at sequential addresses in M_INP. The flowchart in Figures 10a-10b explains how the de-interleaver reads data from M_INP, de-interleaves it and writes it to a local memory M_LOC. The entire process consists of reading the data out, element by element from M_INP, carrying out an address transformation, and writing the element to that location in M_LOC. This location corresponds to the original location of the element in memory before the interleaving was performed at the transmitter. Figures 5-8 illustrate the correspondence of the interleaver mapping of elements T'0 - T'309 to resequenced elements T0 - T309 in words W0-W59 and, in Figure 8, the correspondence to the original bit sequence contained in word w0-w9 on the transmitter side.

[0044] Table 2 provides a list of parameters as used in the flow chart of Figures 10a and 10b.

[0045] At the start, the variables used in the process are initialized at block 10. The address incrementer ADDR, and row counter ROW_CTR and column index pointer IDX are set to 0. The pre-defined permutation order is stored in a vector named PERM_VECT. The order of the permuted columns within PERM_VECT is preferably as shown in Table 1 for a FDD modem receiver 2nd de-interleaver. In step 12, a valve PERM is output from PERM_VECT based on the IDX value which indicates the column position for the current element being processed.

[0046] The next several actions 14, 16, 18 determine the number of rows within column number PERM, and sets the variable NROW to this value. A constant parameter MAX_COL is set such that columns 0, 1, 2, .., MAX_COL - 1 have "ROW" number of rows in them, and columns MAX_COL, ..., *C* - 1 have "ROW-1" rows in them. Based on this fact and the current value of PERM, the variable NROW is set accordingly.

PARAMETER	DESCRIPTION
ADDR	Word address incrementer in M_INP for words W' starting at address A_0
T	Total number of elements in data block
ROW_CTR (or n)	Counter for counting rows in column PERM.
PERM_VECT	Column permutation vector.
COL (or C)	Number of columns in permutation matrix.
ROW (or N)	Number of rows in permutation matrix.
PERM (or i)	PERM_VECT element pointed to by IDX.
IDX	PERM_VECT element pointer.

MAX_COL	Constant value equal to T- (C* (N-1)).
NROW	Number of rows in column number PERM.
SA	Start bit address of element.
EA	End bit address of element
SM	Start word address of element.
EM	End word address of element.
S	Start bit location of element within SM.
E	End bit location of element within EM.
M	Number of bits in each element T'# or T#.
R, R1, R2	Storage registers.
L'	Number of bits in each word of set W'.
L	Number of bits in each word of set W.

Table 2 - List of Flow Chart Parameters

[0047] In steps 20, 22, using the initial address A_0, the current ADDR value, and the element size M, start and end *bit-addresses*, SA and EA respectively, of the current data element within M_INP are determined. Dividing SA and EA by the word bit size L' and discarding any remainder (or equivalently shifting right by 5) generates the corresponding word address in the word set W'. These word addresses are SM and EM, respectively. Then in step 26, the start and end *bit-locations* of the data element within the memory word(s) identified by SM and EM is calculated as S and E, respectively. S and E may be contained within a single memory word of the set of words W', or be spread across two consecutive memory words. The next set of actions 28, 30, 32, 34, 36 demonstrates how these two scenarios are handled.

[0048] The next action 28 in the flowchart is to compare the SM and EM word locations. If the element is within a single word of the set of words W', i.e. EM=SM, then in step 30, the word in location SM is fetched from M_INP. The element is then, in step 32, extracted from its bit locations, as indicated by S and E, and the value is assigned to register R. If, on the other hand, the element is contained within two words of the set of words W', i.e. EM=SM+1, two words have to be accessed from M_INP. Accordingly, the word from SM is fetched and assigned to register R1 and the word from EM is fetched and assigned to register R2 is shown in step 34. Then in step 36 the bits of the element are extracted from R1 and R2 and assigned to register R. Thus, in either case, all of the bits of the interleaved element contained in the set of words W' stored in M_INP are extracted. Finally, the address counter ADDR is incremented for initializing the extraction of the next element.

[0049] The next set of actions 40-60, shown in Figure 10b, is to determine the word(s) and bit location within M_LOC where the extracted element will be stored, access the word(s), place the element within appropriate bit locations within the word(s), and write the word(s) back into M_LOC. These steps can be performed as a single *read-modify-write* operation.

[0050] The start and end mapping *bit addresses*, SA and EA, of where the extracted element stored in R, in step 32 or 36, will be stored into M_LOC is determined in steps 40-42. The start address is calculated in step 40 based on the row and element column mapping of the element extracted in steps 30, 32 or 34, 36. The matrix position is calculated by multiplying the row number, given by ROW_CTR, by the number of

matrix columns, COL, plus the current column number PERM derived from the PERM_VECT vector, i.e. (ROW_CTR * COL) + PERM. Since each element has M bits, the result is multiplied by M to get SA.

[0051] Dividing SA and EA by L, the bit size of the words in set W, and discarding the remainder, generates the corresponding word addresses in step 46. These word addresses are SM and EM, respectively. Finally, the start and end *bit-locations* of where the extracted element in register R is to be placed are computed as S and E, respectively. Where L is not evenly divisible by M, S and E may be contained within a single memory word, or be spread across two consecutive memory words of the set of words W. The next set of actions 48, 60 describe how these two scenarios are handled.

[0052] In step 48, the addresses SM and EM are compared. If the extracted element is to be stored is within a single word, i.e. SM=EM, then in step 50 the word in location SM is fetched from M_LOC and placed in register R1. The extracted element value in R is then, in step 52, written to the bit locations indicated by S and E within R1. Finally, R1 is written back into memory location SM of M_LOC in step 54.

[0053] If on the other hand, the extracted element is to be stored within two consecutive words having addresses SM and SM+1, those words are fetched in step 56 from M_LOC and placed in registers R1 and R2, respectively. Then, in step 58, the bits of the extracted element within R are placed into appropriate locations in registers R1 and R2, respectively, based upon S and E. Finally, the register contents of R1 and R2 are written back, in step 60, into memory locations SM and SM+1, respectively.

[0054] The next action in step 62 is to increment the row counter ROW_CTR by 1 to indicate that the next extracted element T'# will be stored in the next row of the same column. A check is made in step 64 to determine if the row counter is less than or equal to the number of rows of the current column, NROW. If that is the case, the process continues at step 20 with the next element within column member PERM.

[0055] If ROW_CTR is not less than NROW, in step 64, the next extracted element will be stored at an address corresponding to the first row (row 0) of the next column indicated by the vector PERM_VECT. Accordingly, if that is the case, ROW_CTR is reset to 0 and the PERM_VECT index, IDX, is incremented by 1 in steps 66, 68. If, in step 70, IDX is less than COL, the de-interleaving process is repeated from step 12 with a new value of PERM being assigned, otherwise the process is stopped since all T elements of the data block will have been processed.

[0056] While the general processing method is described in accordance with the flow charts of Figures 10a and 10b, a preferred implementation of the process in hardware is illustrated in Figures 11a-11c. The preferred design consists of a 3-stage pipeline, with an associated memory, LOCAL MEMORY, for storing the deinterleaved bits of data. Parallel processing components of the first stage are illustrated in Figures 11a and 11b; the second and third stage processing is illustrated in Figure 11c.

[0057] The operation of stage-1 commences with the extraction of a data element from a 2L' bit vector defined by the contents of two registers REG3 and REG4. The registers REG3 and REG4 store two consecutive L' bit words from physical channel

(PyCH) memory. For the preferred 32-bit word size, these two registers form a 64-bit vector of bits.

[0058] A register REG0, an adder 71, a substracter 72, and a selector 73, are configured to operate in conjunction with a merge device 74 to extract elements having a size of M bits from registers REG3 and REG4 on a sequential basis and store the element in a register REG2. To initialize the interleaver, first and second words of the sequential words W' are initially stored in registers REG3 and REG4, respectively, and register REG0 is initialized to 0. The merge device 74 receives the value 0 from register REG0, extracts the M bits starting at address 0 through address M-1. Thus, the first M bit from the initial word in REG3, which corresponds to the first element T'0 are extracted. The merge device 74 then stores the extracted M bits in the pipeline register REG2.

[0059] The value of register REG0 is incremented by either M via the adder 71 or M-L' via the adder 71 and the substracter 72 based upon the action of the selector 73. If incrementing the value of register REG0 by M does not exceed L', the selector 73 increments register REG0 by M. Otherwise, the selector 73 increments the register REG0 value by M - L'. This effectively operates as a modulo L' function so that the value of REG0 is always less than L' thereby assuring that the start address of the element extracted by the merge device 74 is always within the bit addresses 0 - L' - 1 of register REG3.

[0060] Where the selector 73 selects to increment register REG0 by M - L', a signal EN is sent to trigger the transfer of the contents of REG4 to REG3 and the fetching of the

next sequential word of the set of words W' from the external memory for storage in REG4. During the fetch process, the entire pipeline is stalled. The subtracting of L' in conjunction with the incrementing of the value of register REG0 corresponds with the transfer of the word W' in register REG4 to register REG3 so that the sequential extraction of elements is continued with at least the first bit of the element being extracted from the contents of register REG3.

[0061] With reference to Figure 11b, an interleaver positioning value is calculated in parallel with the extraction process for the element being extracted. The matrix mapping information is calculated by retrieving a current row value n from a register N-REG, and multiplying it in a multiplier 75 by the number of element columns COL in the interleaver matrix. An adder 76, then adds a current column value i which is output from a register file 78 containing the interleaver column sequence as a vector PERM_VECT. The output of the register file 78 is controlled by the content of an index register I-REG which increments the value of the output of the register file 78 in accordance with the vector PERM_VECT.

[0062] The matrix mapping circuitry also include elements to selectively increment the row index register N-REG and the column index register I-REG. The circuitry effectively maintains the same column until each sequential row value has been used and then increments the column to the next column in the interleaver vector starting at the initial row of that column. This is accomplished through the use of a unit incrementer 80 associated with the row register N-REG to increment the row value by one for each cycle of first stage processing. The output of register N-REG is also compared in comparator

81 against a maximum row value determined by a multiplexer 83. The maximum row value for the particular column is either the maximum row value ROW of the entire matrix or ROW-1. The multiplexer 83 generates an output in response to a comparator 84 which compares the column value currently being output by the register file 78 with the largest column value having the maximum row size ROW.

[0063] If the comparator 81 determines that the maximum row number has been reached by the output value of register N-REG, the comparator 81 issues a signal to reset N-REG to 0 and to operate a multiplexer (MUX) 86 associated with the index register I-REG. A unit incrementer 88 is also associated with the index register I-REG and the MUX 86 permits incrementation of the I-REG value by one via the incrementer 88 when a signal is received from comparator 81. Otherwise, the multiplexer 86 simply restores the same value to register I-REG during a first stage cycle.

[0064] Referring to Figure 11c, the second stage of the pipeline interleaver comprises a processing cycle where the element extracted and stored in the first pipeline register REG2 is transferred and stored into a second data pipeline register REG9. In parallel in the second stage of processing, the corresponding matrix mapping data stored in register REG1 is used to calculate corresponding start bit address data which is stored in a register REG5, end bit address data which is stored in a register REG8, start word address data which is stored in a register REG6, and end word address data which is stored in a register REG7. During a second stage cycle, the matrix mapping data from REG1 is initially multiplied by the element bit size M in a multiplier 90. The start bit address data is then calculated by subtracting from that resultant value in a substracter 91

a value to produce a modulo L equivalent, where L is the bit size of the data words of a local memory 100 where the extracted elements are to be selectively stored. The value subtracted in substracter 91 is calculated by dividing the output of multiplier 90 by L without remainder in divider 92 and multiplying that value by L in multiplier 93. The output of the divider 92 also provides the start word address of the corresponding word within which at least a first portion of an element in register REG9 is to be stored in the local memory 100.

[0065] The end bit address data is calculated by adding M-1 to the result of the multiplier 90, in an adder 95 and then subtracting from that value in a subtracter 96 a value calculated to produce a modulo L value which is then stored in register REG8. The value subtracted is derived by dividing the output of the adder 95, in a divider 97, by L without remainder and then multiplying the result by L in a multiplier 98. The output of divider 97 also provides the end word address data which is stored in register REG7.

[0066] The third stage of the pipeline interleaver performs a read-modify-write to selectively store the element value in register REG9 in the local memory based upon the data in registers REG5, REG6, REG7 and REG8. Initially, the contents of registers REG6 and REG7 are compared in a comparator 99. If the values are equal, the element in register REG9 will be stored within a single word of the local memory 100. In that case, the value from register REG6 passes through multiplexer 101 to multiplexer 102 where it may be combined with a base address which can be used to allocate overall memory resources within the system.

[0067] The output of multiplexer 102 indicates the address of the word W into which the element in register REG9 is to be written. That word is output to a de-multiplexer 103 whereupon a merged device creates a new word comprised of the bit values of the element in register REG9 in the sequential addresses within the word starting with the value in register REG5 and ending with the value in register REG8, with the remaining bits of the word being copied from the values of the word in de-multiplexer 103. The newly formed word in the merge device 105 is then stored back to the address from which the original word was output to the de-multiplexer 103.

[0068] Where the contents of registers REG6 and REG7 are different, the first and second stages of the pipeline are stalled for one cycle so that the third stage can perform a read-modify-write cycle with respect to the word identified by the data in register REG6 and then resume the pipeline cycles of all stages to perform a read-modify-write with respect to the local memory word corresponding to the end word data stored in register REG7. In that case, during the read-modify-write cycle with respect to the word corresponding to the start word address data in register REG6, the third stage stores an initial portion of the element stored in register REG9 in the last bits of the local memory word starting with the bit position indicated by the value stored in register REG5. During a second third stage cycle, where the first and second stage cycles are resumed, the remaining portion of the element in register REG9 is stored in the word corresponding to the end word address data in register REG7 starting with the initial bit of that word through the bit address indicated by the value in register REG8.

[0069] After all T elements of a block of data bits have been processed, the sequential words of the local memory are read out via the de-multiplexer 103 for further processing in the system. The output of the local memory after processing for the 310 element data block reflected in the example of Figures 5-8 correspond to the word sequence reflected in Figure 7c. During further processing within a 3G system, the expanded six bit elements are contracted to a single bit thereby, for the example, reproducing the original 310 bit data block in the same sequence as originally occurring in the transmitter unit.

[0070] Testing of the 3-stage pipeline of the second interleaver was carried out using two different techniques. First of these testing methods was a manual technique called *regression*. Regression testing was carried out by fetching 30, 32-bit words from the PyCH memory, extracting 6-bit elements from them, and passing them down the pipeline. The testing cycle was based on manual cycle-bases simulation, where the expected contents of the registers and the internal memory were determined by hand. These values were compared with the actual values obtained from simulation. The simulation was carried out for a large number of test cases and for all cases of the pipeline stall condition. The interleaver pipeline was found to function correctly under all the test scenarios of the manual setting.

[0071] Next, the interleaver was independently implemented in C-language. A set of test vectors were applied to the C-block and outputs were monitored and written to a results file. The same set of input test vectors were applied to the VHDL model. Two sets of input vectors were used in the tests:

[0072] A 201-element input vector and a 540-element input vector. Two different sets of inputs were used to create two different interleaver matrices. The 201-element matrix had two different row sizes; one row is one less than the other one. The 540-element matrix had a single row size. Thus, the tests included the two different types of interleaver matrix structures that are possible. The test results showed that the output vectors from the VHDL model and the C-language model matched the two input cases.

[0073] The hardware was synthesized using Synopsys Logic Synthesizer, Using Texas Instruments 0.18um standard cell library. The gate counts are given below.

Number of Standard Cells (TI/GS30/Std-Cell)	1034
Sequential gates	1844
Combination gates	3348
Total gates	5192

Table 3 - Total gate count estimate for the interleaver

[0074] The pipelined architecture ensures a high-rate of throughput, and a small compact area due low number of gates. While a three stage pipeline is preferred, a two stage design is easily implemented by eliminating registers REG1 and REG2 from the preferred system illustrated in Figures 11a-11c.

[0075] Other variations and modifications will be recognized by those of ordinary skill in the art as within the scope of the present invention.

* * *

CLAIMS

What is claimed is:

1. A method of processing data in a communication systems by selectively resequencing a block of T sequential expanded data elements having a bit size M contained in a sequential set of data words W' having a bit size L' to produce a set of sequential data words W having a bit size L containing the T expanded data elements in a selected sequence based upon an interleaver mapping to a matrix having C element columns and N rows where L and L' are integers larger then M, C is not equal to L and the last r columns of the matrix have N-1 rows where $r < C$ and $r = (C * N) - T$, comprising:

sequentially extracting the T data elements from the set of sequential data words W';

determining a matrix mapping position for a first extracted element at a first row of an initial column of a pre-determined interleaver column sequence;

for each subsequent extracted data element, determining a matrix mapping position of a row n and a column i at a next row of the same column as the immediately preceding data element or, if that column has no next row, the first row of the next column in the pre-determined interleaver column sequence;

defining a row by row sequential mapping of M bit sequential addresses of a local memory of data words W corresponding to the C by N matrix;

for each data element, determining a sequential bit address within one word or spanning two sequential words of the set of words W corresponding to the element's determined matrix mapping position; and

storing each data element at its determined address.

2. The method of claim 1 wherein the steps are performed in a receiver modem as a second deinterleaving process further comprising:

after storing a last of the T sequential data element, sequentially reading out the data words W from the local memory whereby the T elements are sequentially ordered in a deinterleaved sequence corresponding to a block of T data bits from which the block of T sequential expanded interleaved data elements was created in a transmitter modem.

3. The method of claim 1 wherein a sequential bit address for each element is defined in either one word or spanning two sequential words of the set of data words W' and the element extraction includes determining a start bit address and an end bit address which corresponds to a data element to be extracted and storing the data between and inclusive of the determined bit addresses in a register.

4. The method of claim 3 wherein the set of words W' are located sequential addresses of a memory starting at address A_0 wherein a start bit address for the first extracted element is A_0 and the start bit address for each subsequent extracted element is M more than an immediately preceding extracted element.

5. The method of claim 1 wherein:

the sequential set of data words W' is sequentially read into first and second registers from which each element is extracted and stored in a first pipeline register;

first and second sequential words of the set of words W' are initially stored in the first and second registers, respectively;

the first data element is extracted from the first M bits the first word in the first register;

each subsequent element is extracted starting with bits of a word in the first register; and

after all bits of a word W' in the first register have been extracted, the word W' in the second register is transferred to the first registered and a next sequential word W' is stored in the second register.

6. The method of claim 5 wherein:

the first register has addresses A_0 through $A_0 + (L'-1)$ and the second register has addresses $A_0 + L'$ through $A_0 + (2L'-1)$;

a start address $A_0 + 0$ is defined for the first data element; and

a start address for each subsequent element is defined as address $A_0 + SA$ of the first register based on the start address $A_0 + SA'$ of the immediately preceding element such that

when $SA' + M < L'$, $SA = SA' + M$, and

when $SA' + M \geq L$, $SA = (SA' + M) - L$ and, before extraction, the word in the second register is stored to the first register and the next sequential word of the set of words W' is stored to the second register.

7. The method of claim 5 wherein the matrix mapping position row and column for each element is determined in parallel with the storage of an element in the first pipeline register thereby defining a cycle of a first stage of processing.

8. The method of claim 7 wherein an element in the first pipeline register is stored to a second pipeline register and local memory address information is determined for that element in parallel thereby defining a cycle of a second stage of processing.

9. The method of claim 8 wherein the local memory includes bit addresses A'_0 through $A'_0 + (T*M) - 1$ where each sequential word of the set of words W is assigned L sequential bit addresses, and a local memory start address LSA for each data element is determined by $LSA = A'_0 + ((n*C) + i) * M)$ where $LSA = A'_0$ for the first data element.

10. The method of claim 9 wherein:

the first register has addresses A_0 through $A_0 + (L'-1)$ and the second register has addresses $A_0 + L'$ through $A_0 + (2L'-1)$;

a start address $A_0 + 0$ is defined for the first data element; and

a start address for each subsequent element is defined as address A_0 + SA of the first register based on the start address A_0 + SA' of the immediately preceding element such that

when SA' + M ‹ L, SA = SA' + M, and

when SA' + M ≤ L, SA = (SA' + M) -L and, before extraction, the word in the second register is stored to the first register and the next sequential word of the set of words W' is stored to the second register.

11. The method of claim 8 where at least a portion of an element in the second pipeline register is stored to a word of the set of words W in the local memory thereby defining a cycle of a third stage of processing.

12. The method of claim 11 where a local memory start address LSA and a local memory end address LEA is determined for each data element during a second stage processing cycle and when LSA and LEA are in two consecutive words of the set of words W of the local memory, a first portion of the element in the second pipeline register is stored in one of the two words during a third stage processing cycle while the first and second processing stages are stalled for one cycle.

13. The method of claim 12 wherein, except for the first and last elements, each time a second stage processing cycle occurs, a first stage processing cycle and a third stage processing cycle also occurs.

14. The method of claim 13 wherein first, second and third stage processing is stalled while the word in the second register is transferred to the first register and a next sequential word of the set of words W' is stored in the second register.

15. An interleaver for selective resequencing a block of T sequential data elements having a bit size M contained in a sequential set of data words W' having a bit size L' to produce a set of sequential data words W having a bit size L containing the T data elements in a selected sequence based upon an interleaver mapping to a matrix having C element columns and N rows where L and L' are integers larger then M, C is not equal to L and the last r columns of the matrix have N-1 rows, where $r < C$ and $r = (C * N) - T$, comprising:

a memory device from which sequential words W' are accessed;

a first pipeline register for receiving data elements extracted from the memory device;

a matrix position register device for receiving matrix position data relating to an element being stored in the first pipeline register;

a second pipeline register for sequentially receiving data elements from the first pipeline register;

a local memory;

a local address register device for receiving local memory address data relating to an element being stored in the second pipeline register;

first stage processing circuitry including

circuitry for sequentially extracting data elements from the memory device and storing each sequentially extracted element in the first pipeline register, and

matrix mapping circuitry for generating and storing corresponding matrix position data in the matrix position register device;

second stage processing circuitry for generating and storing local memory address data in the local address register device from matrix position data stored in the matrix position register device corresponding to an element being transferred from the first pipeline register and stored in the second pipeline register; and

third stage processing circuitry for retrieving data elements stored in the second pipeline register and selectively storing each data element in sequential words of the set of words W of the local memory based on the corresponding address data stored in the local address register device.

16. An interleaver according to claim 15 wherein L' is not equally divisible by M, the memory device comprises first and second L' bit registers to which words from the set of words W' are sequentially transferred, and the first stage extracting circuitry extracts elements from M sequential bit addresses which start at selectively determined bit addresses of the first L' bit register.

17. An interleaver according to claim 15 wherein the matrix position register device is a single register and the matrix mapping circuitry includes a column index register, a row index register, and an interleaver vector memory which are selectively

coupled such that the row index register is incremented or reset in conjunction with generating matrix position data for each element, the index register is incremented in conjunction with resetting the row register, and the index register is used to increment an output of the interleaver vector memory in accordance with an interleaver vector stored therein.

18. An interleaver according to claim 17 wherein for each data element, the row register outputs a value n and the interleaver vector memory outputs a value i and the value (n*C) + i is stored in the matrix position register in connection with the storage of an element in the first pipeline register.

19. An interleaver according to claim 18 wherein L' = 32, C = 30, M = 6, the memory device comprises first and second L bit registers to which words of the set of words W' are sequentially transferred, and the first stage extracting circuitry extracts elements from M sequential bit addresses which start at selectively determined bit addresses of the first L' bit register.

20. An interleaver according to claim 15 wherein L is not equally divisible by M, the local address register device includes a start bit address register, an end bit address register, a start word address register, and an end word address register, and the second stage processing circuitry calculates start bit address data, end bit address data, start word address data, and end word address data for the respective registers of the local address

register device based upon matrix position data (MPD) stored in the matrix position register device, the data element bit size M and the data word bit size L in conjunction with the transfer of an element from the first pipeline register to the second pipeline register.

21. An interleaver according to claim 20 wherein the second stage processing circuitry calculates the start bit address data to be MPD modulo L, the end bit address data to be (MPD + M) modulo L, the start word address data to be the largest integer resulting from MPD/L and the end word address data to be the largest integer resulting from (MPD + M) / L.

22. An interleaver according to claim 21 wherein a first stage cycle is defined when the first stage processing circuitry extracts a new element from the memory device and stores it in the first pipeline register, a second stage cycle is defined when an element is transferred from the first pipeline register to the second pipeline register, and a third stage cycle is defined when the third stage circuitry selectively stores at least a portion of an element from the second stage pipeline register in a word of the set of words W of the local memory, wherein the second stage processing circuitry is associated with the first stage processing circuitry and the third stage processing circuitry such that, except for first and last of the T sequential elements, each time a second stage cycle occurs, a first stage cycle and a third stage cycle also occur.

23. An interleaver according to claim 22 wherein the third stage processing circuitry includes a comparator for comparing the contents of the start word address register and the end word address register which is associated with data storage circuitry and the first and with second stage processing circuitry such that:

when the start word address data equals the end word address data, the element in the second pipeline register is stored in a word of the set of words W corresponding to the start word address data at a bit location within that word corresponding to the start bit address data stored in the start address register through an address corresponding to the end bit address data stored in the end bit address register; and

when the start word address data is not equal to the end word address data, the first and second stage processing circuitry is stalled for one cycle while a first portion of the element stored in the second pipeline register is stored in a word of the set of words W corresponding to the start word address data at a bit location starting with the bit address corresponding to the start bit address data stored in the start bit address register through an end bit of that word thereby completing one third stage cycle and in a subsequent third stage cycle, when the first and second stage processing circuitry is not stalled, a second portion of the element stored in the second pipeline register is stored in a next sequential word of the set of words W corresponding to the end word address data starting at a first bit of that word through a bit location corresponding to the end bit address data stored in the end bit address register.

24. An interleaver for selective resequencing a block of T sequential data elements having a bit size M contained in a sequential set of data words W' having a bit size L' where to produce a set of sequential data words W having a bit size L containing the T data elements in a selected sequence based upon an interleaver mapping to a matrix having C element columns and N rows where L and L' are integers larger then M, C is not equal to L and the last r columns of the matrix have N-1 rows, where $r < C$ and $r = (C * N) - T$, comprising:

a memory device from which sequential words of the set of words W' are accessed;

a pipeline register device for receiving data elements extracted from the memory device;

a local memory;

a local address register device for receiving local memory address data relating to an element being stored in the pipeline register device;

element extracting circuitry for sequentially extracting data elements from the memory device and storing each sequentially extracted element in the pipeline register device;

matrix mapping circuitry for generating matrix position data for each extracted element;

address processing circuitry for generating and storing local memory address data in the local address register device from matrix position data corresponding to an element being stored in the pipeline register device; and

storage processing circuitry for sequentially retrieving data elements stored in the pipeline register device and selectively storing each data element in sequential words of the set of words W of the local memory based on corresponding address data stored in the local address register device.

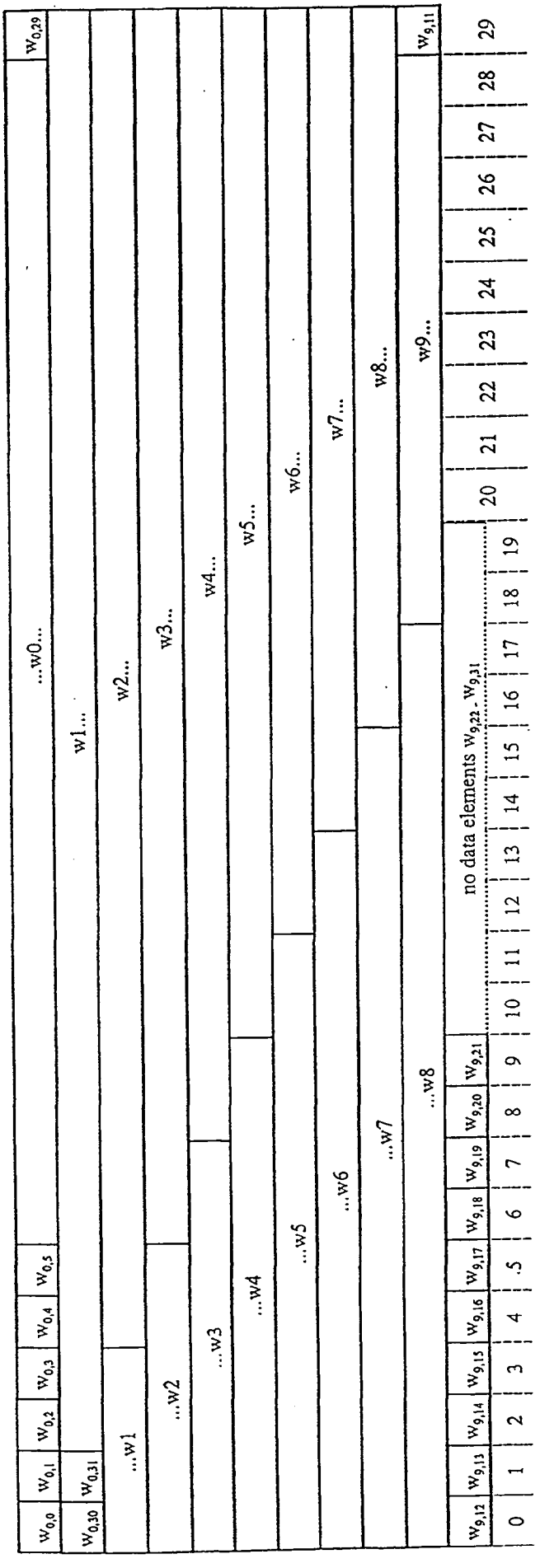


FIG. 1 (Prior Art)

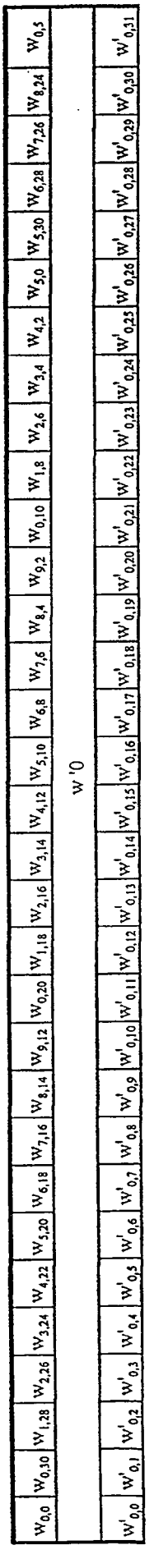


FIG. 2b (Prior Art)

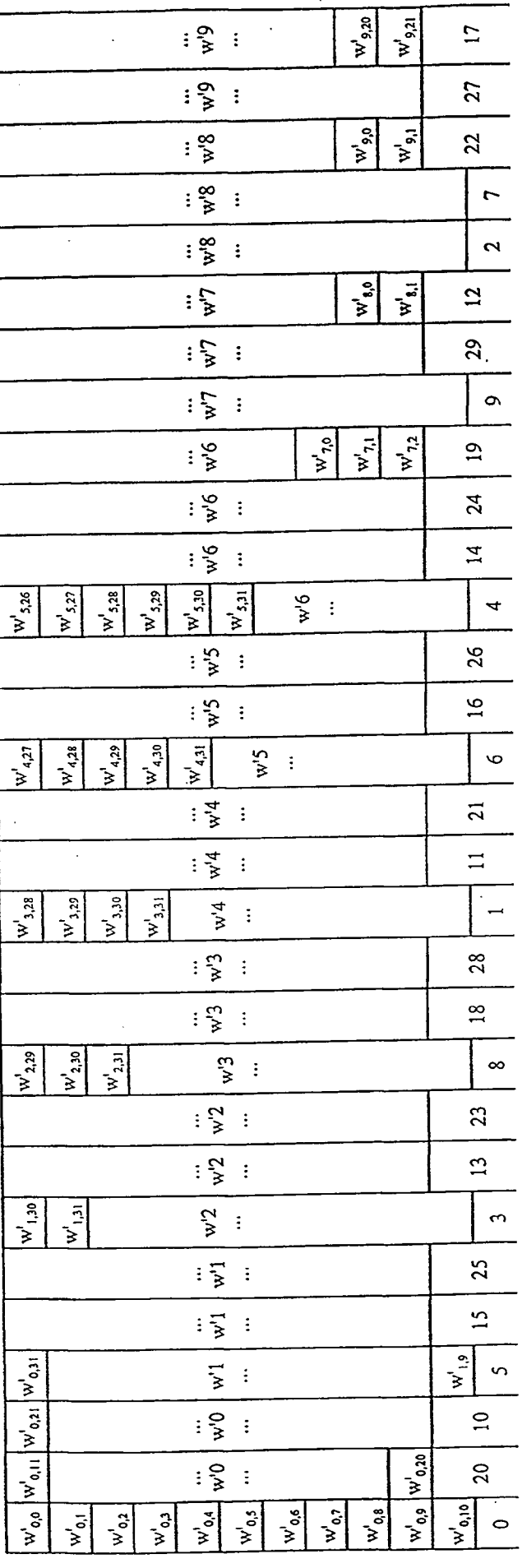


FIG. 2a (Prior Art)

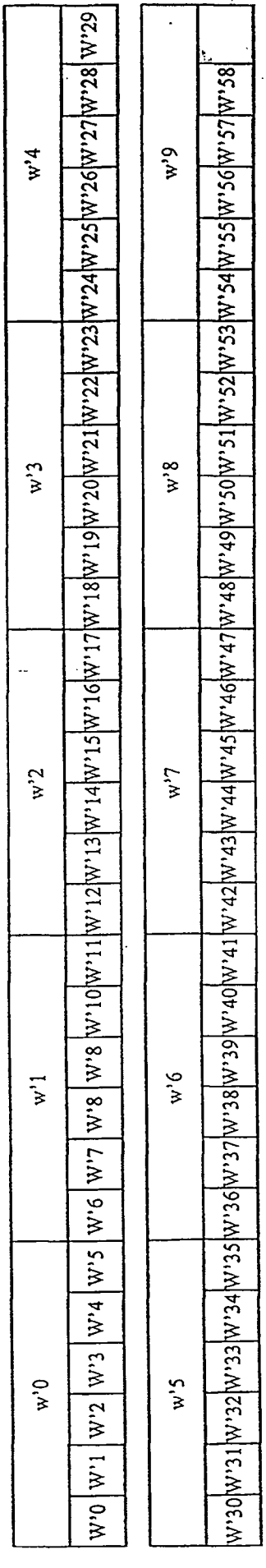


FIG. 3

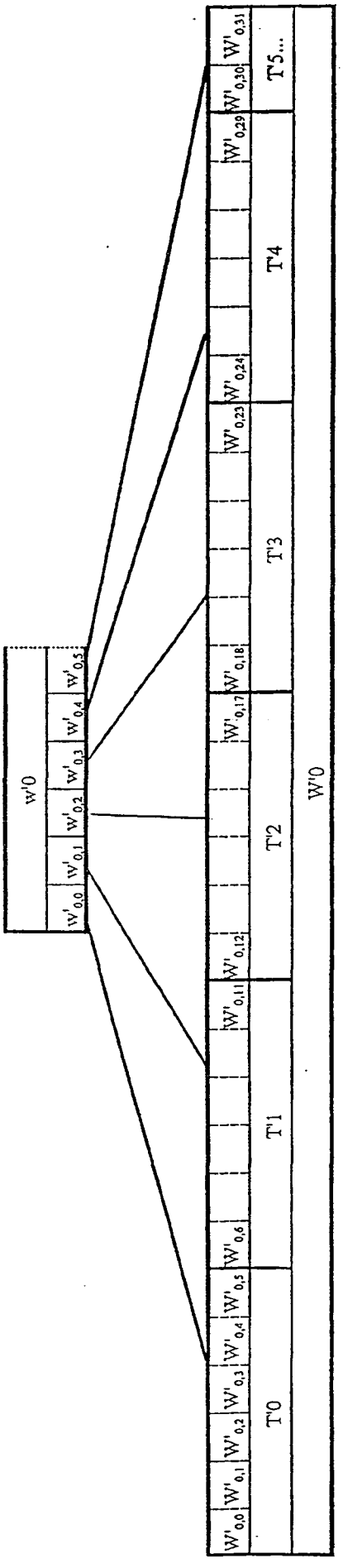


FIG. 4a (Prior Art)

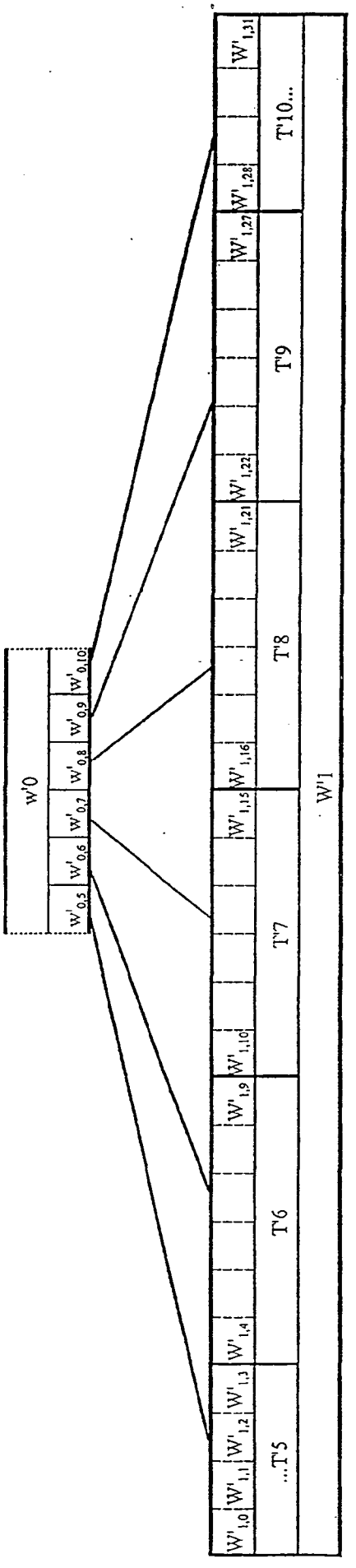


FIG. 4b (Prior Art)

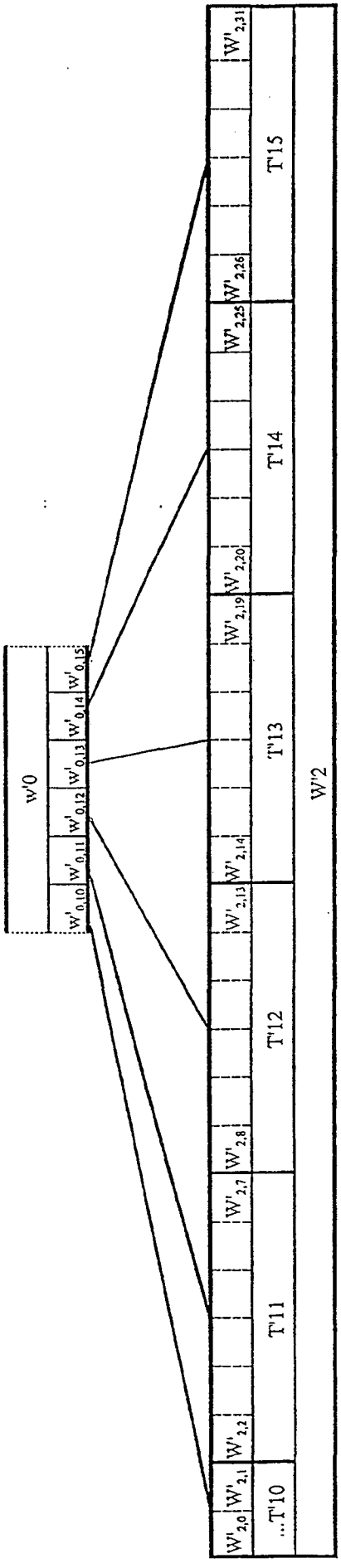


FIG. 4c (Prior Art)

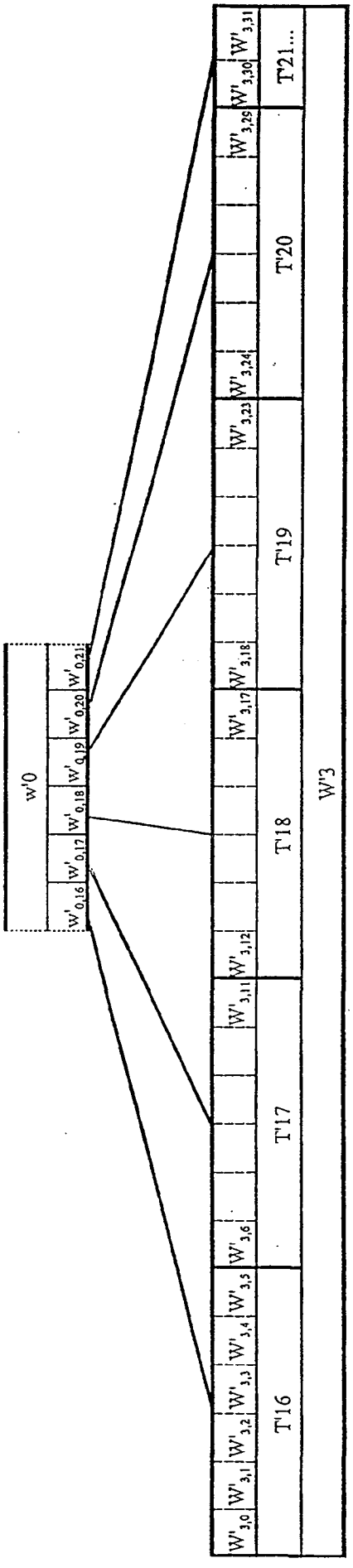


FIG. 4d (Prior Art)

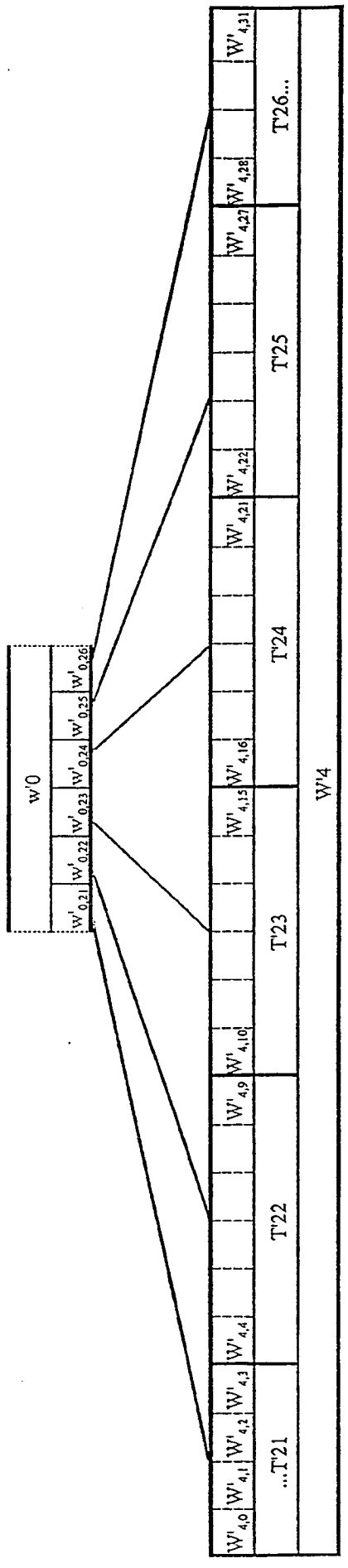


FIG. 4e (Prior Art)

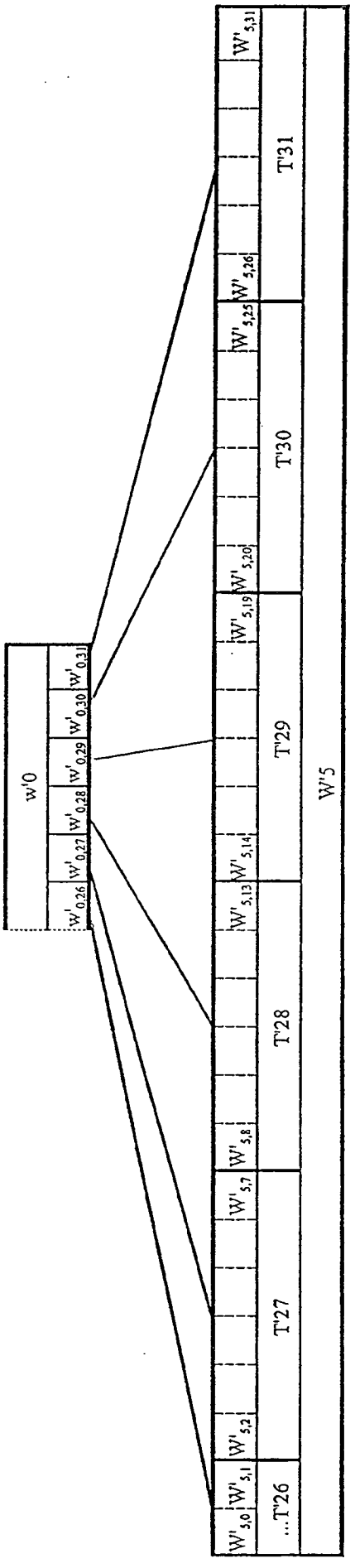
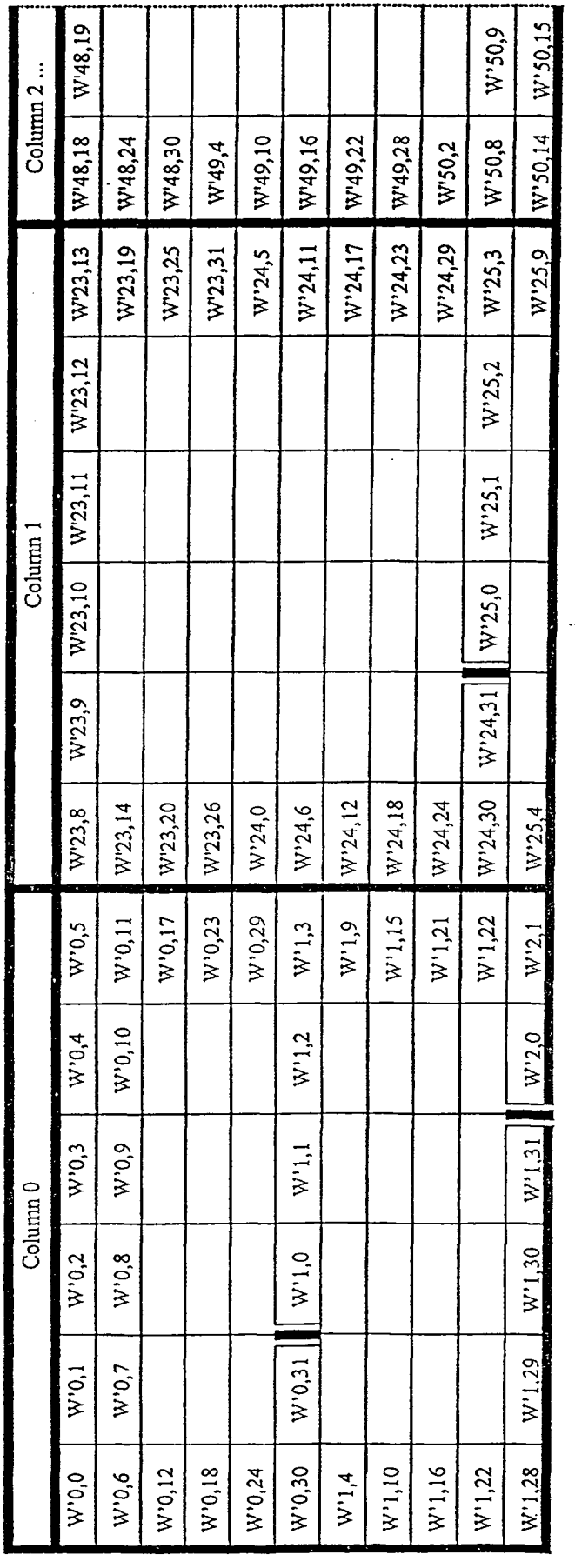


FIG. 4f (Prior Art)



Column 0						Column 1						Column 2 ...	
W'0,0	W'0,1	W'0,2	W'0,3	W'0,4	W'0,5	W'23,8	W'23,9	W'23,10	W'23,11	W'23,12	W'23,13	W'48,18	W'48,19
W'0,6	W'0,7	W'0,8	W'0,9	W'0,10	W'0,11	W'23,14					W'23,19	W'48,24	
W'0,12					W'0,17	W'23,20					W'23,25	W'48,30	
W'0,18					W'0,23	W'23,26					W'23,31	W'49,4	
W'0,24					W'0,29	W'24,0					W'24,5	W'49,10	
W'0,30	W'0,31	W'1,0	W'1,1	W'1,2	W'1,3	W'24,6					W'24,11	W'49,16	
W'1,4					W'1,9	W'24,12					W'24,17	W'49,22	
W'1,10					W'1,15	W'24,18					W'24,23	W'49,28	
W'1,16					W'1,21	W'24,24					W'24,29	W'50,2	
W'1,22					W'1,22	W'24,30	W'24,31	W'25,0	W'25,1	W'25,2	W'25,3	W'50,8	W'50,9
W'1,28	W'1,29	W'1,30	W'1,31	W'2,0	W'2,1	W'25,4					W'25,9	W'50,14	W'50,15

FIG. 5a

Column 27	Column 28							Column 29						
W'54,10	W'21,12						W'21,17	W'44,26	W'44,27	W'44,28	W'44,29	W'44,30	W'44,31
W'54,16	W'21,18						W'21,23	W'45,0					W'45,5
W'54,22	W'21,24						W'21,29	W'45,6					W'45,11
W'54,28	W'21,30	W'21,31	W'22,0				W'22,3	W'45,12					W'45,17
W'55,2	W'22,4	W'22,5	W'22,6	W'22,7	W'22,8		W'22,9	W'45,18					W'45,23
W'55,8	W'22,10	W'22,11	W'22,12	W'22,13	W'22,14		W'22,15	W'45,24					W'45,29
W'55,14	W'22,16						W'22,21	W'45,30	W'45,31	W'46,0			W'46,3
W'55,20	W'22,22						W'22,27	W'46,4	W'46,5	W'46,6	W'46,7	W'46,8	W'46,9
W'55,26	W'22,28	W'22,29	W'22,30	W'22,31	W'23,0		W'23,1	W'46,10	W'46,11	W'46,12	W'46,13	W'46,14	W'46,15
W'56,0	W'56,1	W'23,2					W'23,7	W'46,16					W'46,21
x	x	x	x	x	x		x	x	x	x	x	x	x

FIG. 5b

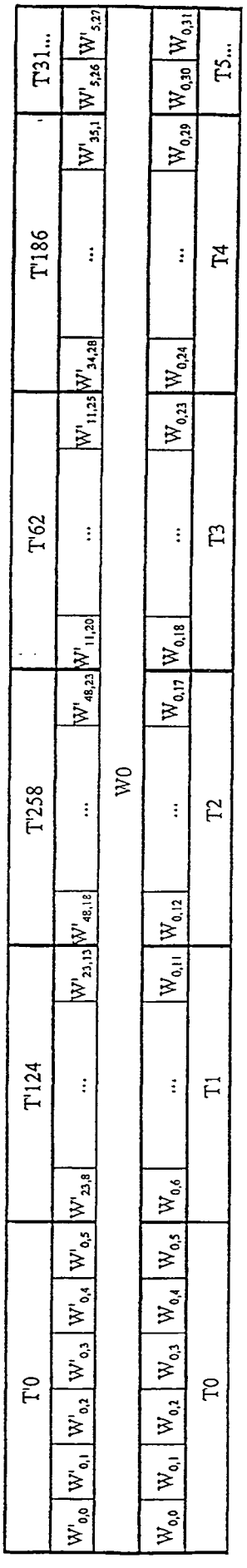


FIG. 6

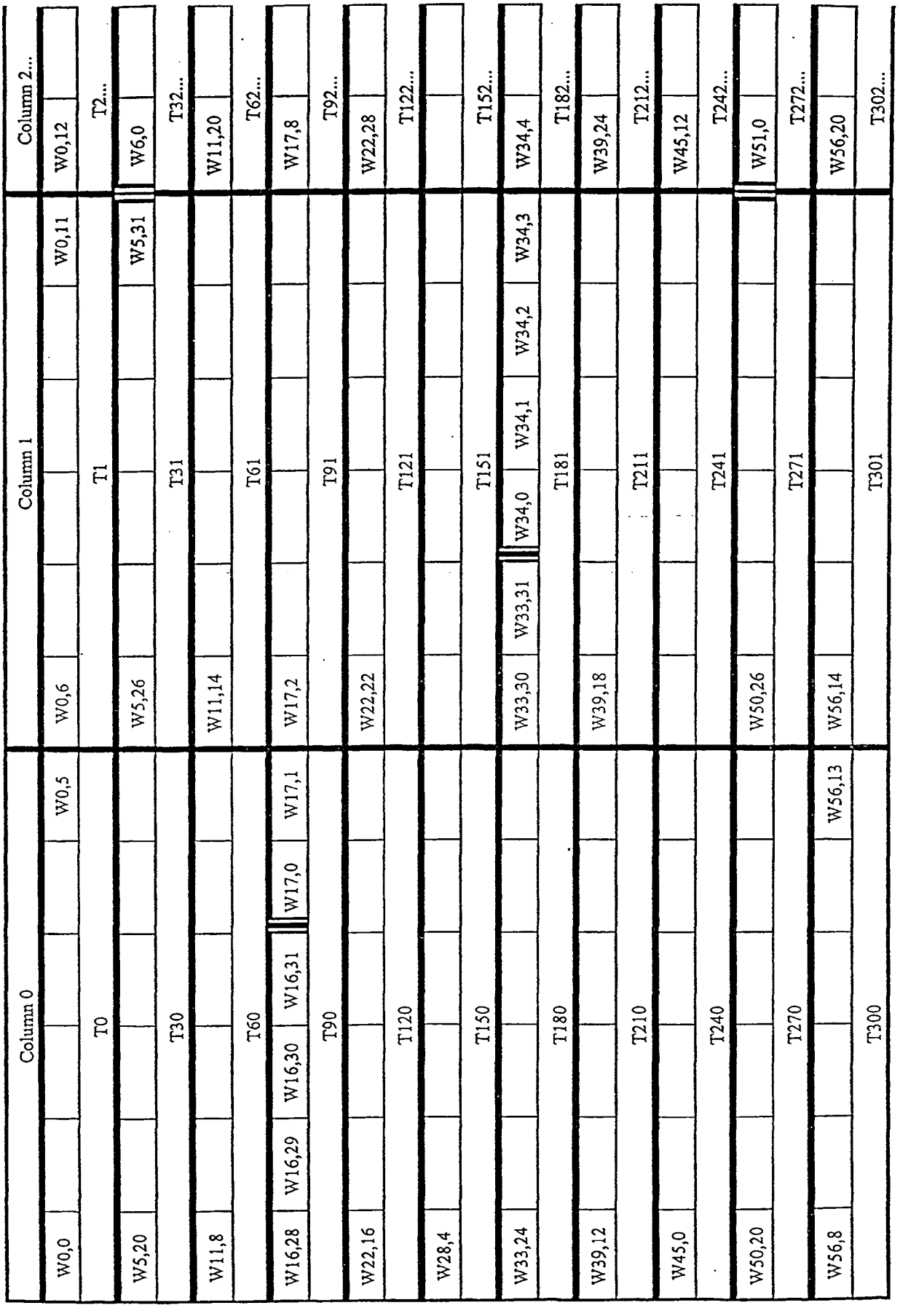

Column 0
Column 1
Column 2...
W0,0
W0,5
W0,6
W0,11
W0,12
T0
T1
T2...
W5,20
W5,26
W5,31
W6,0
T30
T31
T32...
W11,8
W11,14
W11,20
T60
T61
T62...
W16,28
W16,29
W16,30
W16,31
W17,0
W17,1
W17,2
W17,8
T90
T91
T92...
W22,16
W22,22
W22,28
T120
T121
T122...
W28,4
T150
T151
T152...
W33,24
W33,30
W33,31
W34,0
W34,1
W34,2
W34,3
W34,4
T180
T181
T182...
W39,12
W39,18
W39,24
T210
T211
T212...
W45,0
W45,12
T240
T241
T242...
W50,20
W50,26
W51,0
T270
T271
T272...
W56,8
W56,13
W56,14
W56,20
T300
T301
T302...

FIG. 7a

...Column 27		Column 28						Column 29					
W5,6		W5,8					W5,13	W5,14					W5,19
...T27		T28						T29					
W10,26		W10,28			W10,31	W11,0	W11,1	W11,2					W11,7
...T57		T58						T59					
W16,4		W15,6						W16,12					W16,29
...T87		T88						T89					
W22,2		W22,4						W22,10					W22,15
...T117		T118						T119					
W27,22		W27,24						W27,30	W27,31	W28,0			W28,3
...T147		T148						T149					
W33,10		W33,12						W33,18					W33,23
...T177		T178						T179					
W38,30	W38,31	W39,0						W39,6					W39,11
...T207		T208						T209					
W44,18		W44,20						W44,26					W44,31
...T237		T238						T239					
W50,6		W50,8						W50,14					W50,19
...T267		T268						T269					
W55,26		W55,28			W55,31	W56,0	W56,1	W56,2					W56,7
...T297		T298						T299					
x	x	x	x	x	x	x	x	x	x	x	x	x	x

FIG. 7b

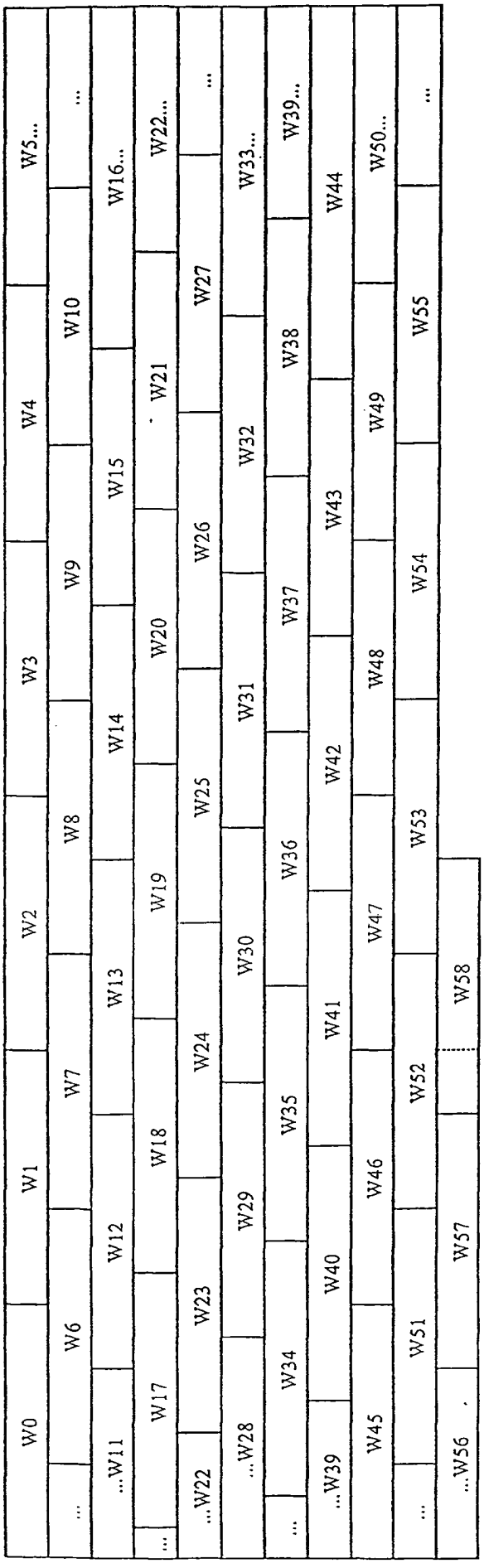

W0 W1 W2 W3 W4 W5...
... W6 W7 W8 W9 W10 ...
...W11 W12 W13 W14 W15 W16...
... W17 W18 W19 W20 W21 W22...
...W22 W23 W24 W25 W26 W27 ...
...W28 W29 W30 W31 W32 W33...
... W34 W35 W36 W37 W38 W39...
...W39 W40 W41 W42 W43 W44
W45 W46 W47 W48 W49 W50...
... W51 W52 W53 W54 W55 ...
...W56 W57 W58

FIG. 7c

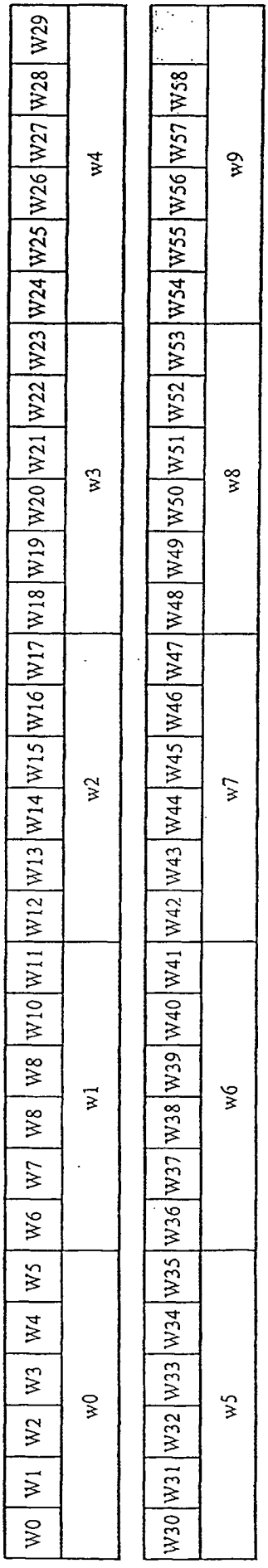
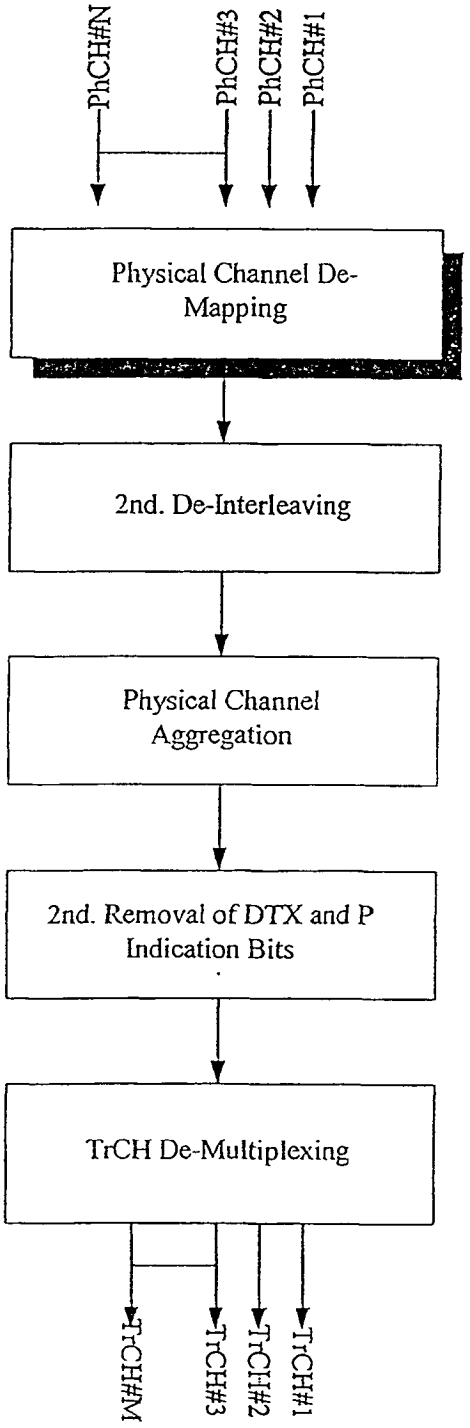

W0 W1 W2 W3 W4 W5 W6 W7 W8 W9 W10 W11 W12 W13 W14 W15 W16 W17 W18 W19 W20 W21 W22 W23 W24 W25 W26 W27 W28 W29
w0 w1 w2 w3 w4
W30 W31 W32 W33 W34 W35 W36 W37 W38 W39 W40 W41 W42 W43 W44 W45 W46 W47 W48 W49 W50 W51 W52 W53 W54 W55 W56 W57 W58
w5 w6 w7 w8 w9

FIG. 8

FIG. 9


PhCH#N
PhCH#3
PhCH#2
PhCH#1
Physical Channel De-Mapping
2nd. De-Interleaving
Physical Channel Aggregation
2nd. Removal of DTX and P Indication Bits
TrCH De-Multiplexing
TrCH#M
TrCH#3
TrCH#2
TrCH#1

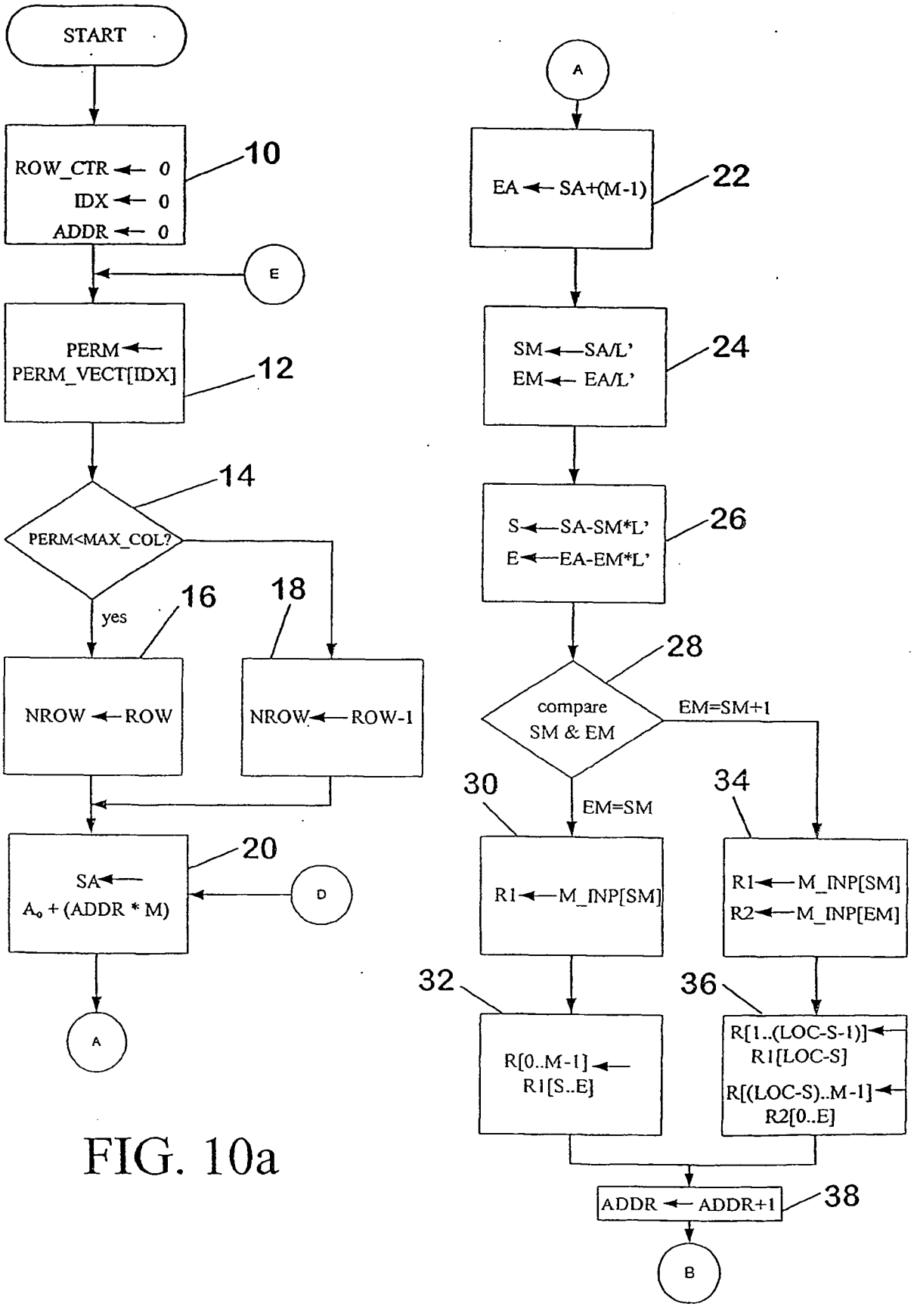

START
ROW_CTR ← 0
IDX ← 0
ADDR ← 0
10
E
PERM ← PERM_VECT[IDX]
12
14
PERM<MAX_COL?
yes
16
18
NROW ← ROW
NROW ← ROW-1
SA ← A0 + (ADDR * M)
20
D
A
A
EA ← SA+(M-1)
22
SM ← SA/L'
EM ← EA/L'
24
S ← SA-SM*L'
E ← EA-EM*L'
26
28
compare SM & EM
EM=SM+1
30
EM=SM
34
R1 ← M_INP[SM]
R1 ← M_INP[SM]
R2 ← M_INP[EM]
32
36
R[0..M-1] ← R1[S..E]
R[1..(LOC-S-1)] ← R1[LOC-S]
R[(LOC-S)..M-1] ← R2[0..E]
ADDR ← ADDR+1
38
B

FIG. 10a

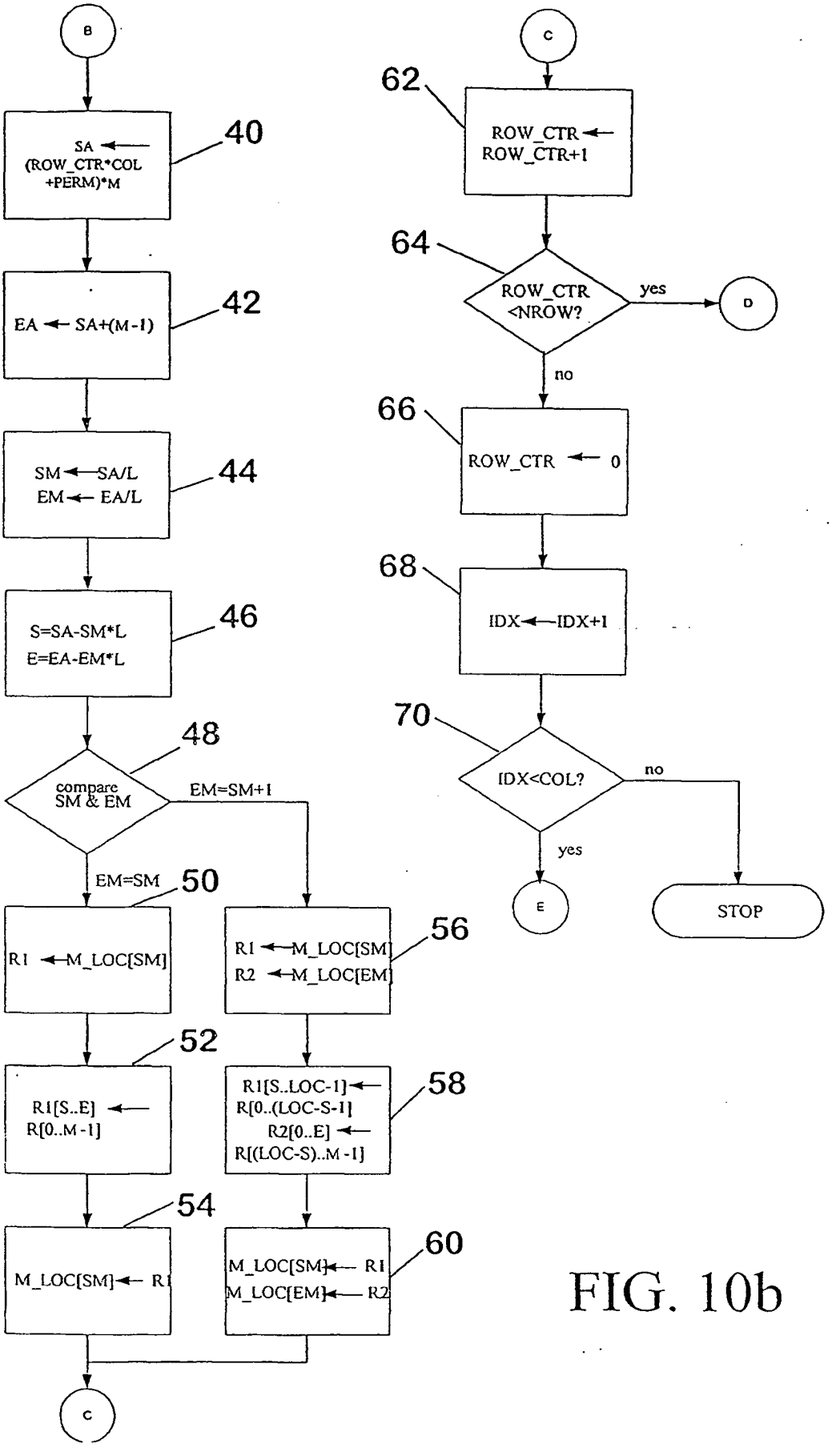

B
SA ← (ROW_CTR*COL +PERM)*M
40
EA ← SA+(M-1)
42
SM ← SA/L
EM ← EA/L
44
S=SA-SM*L
E=EA-EM*L
46
compare SM & EM
48
EM=SM+1
EM=SM
50
R1 ← M_LOC[SM]
R1 ← M_LOC[SM]
R2 ← M_LOC[EM]
56
52
R1[S..E] ← R[0..M-1]
R1[S..LOC-1] ← R[0..(LOC-S-1]
R2[0..E] ← R[(LOC-S)..M-1]
58
54
M_LOC[SM] ← R1
M_LOC[SM] ← R1
M_LOC[EM] ← R2
60
C
C
62
ROW_CTR ← ROW_CTR+1
64
ROW_CTR <NROW?
yes
D
no
66
ROW_CTR ← 0
68
IDX ← IDX+1
70
IDX<COL?
no
yes
E
STOP

FIG. 10b

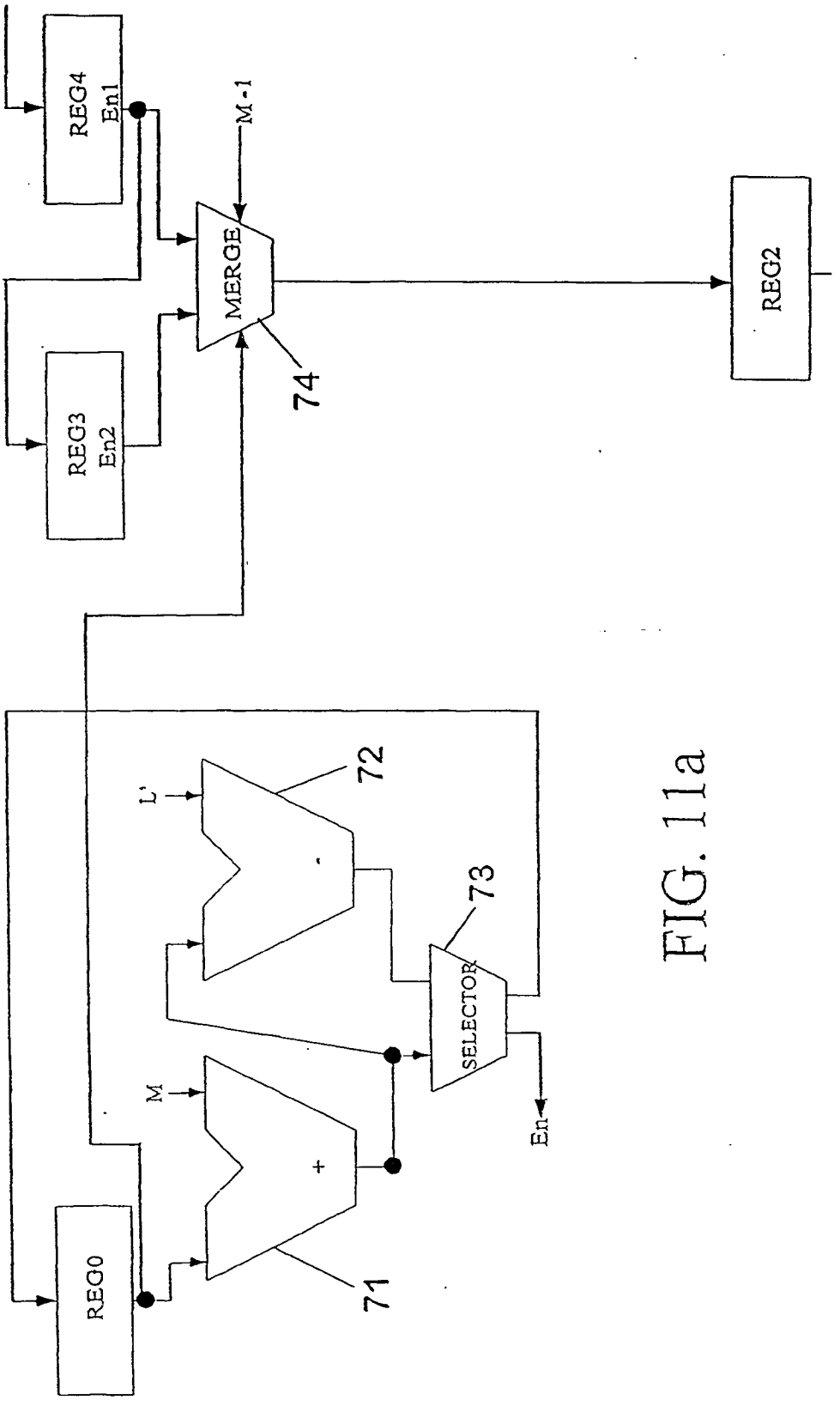


FIG. 11a

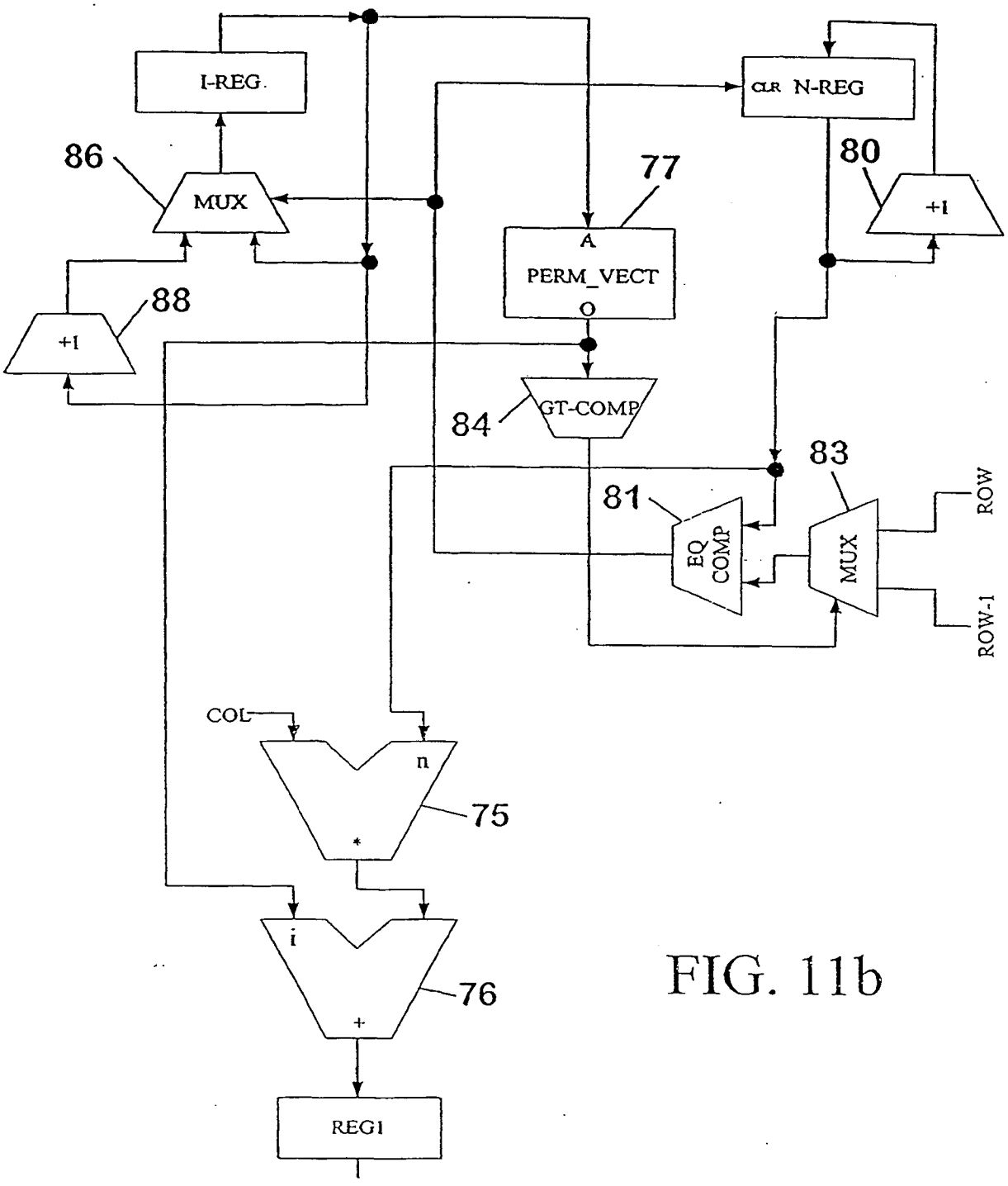

I-REG.
86
MUX
88
+1
77
A
PERM_VECT
O
CLR N-REG
80
+1
84
GT-COMP
83
81
EQ COMP
MUX
ROW
ROW-1
COL
n
*
75
i
+
76
REG1

FIG. 11b

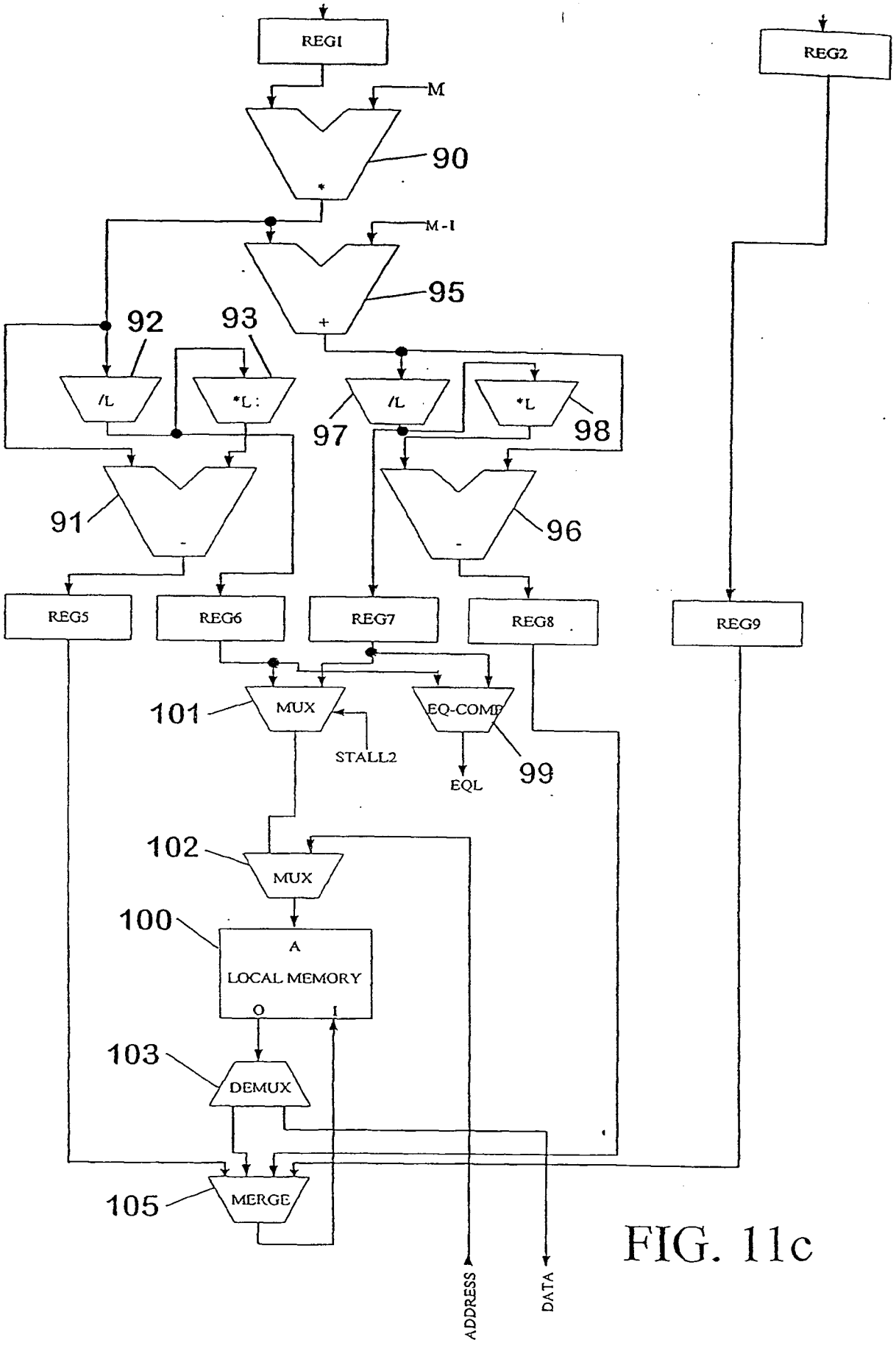

REG1
REG2
M
90
*
M-1
95
+
92
93
/L
*L:
/L
*L
97
98
91
96
-
-
REG5
REG6
REG7
REG8
REG9
101
MUX
EQ-COMP
STALL2
EQL
99
102
MUX
100
A
LOCAL MEMORY
O
I
103
DEMUX
105
MERGE
ADDRESS
DATA

FIG. 11c

INTERNATIONAL SEARCH REPORT

Int al Application No
PCT/US 01/22807

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 H03M13/27 H04L1/00

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 H03M H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	3GPP: "3G TS 25.212 version 3.0.0" 3RD GENERATION PARTNERSHIP PROJECT (3GPP), October 1999 (1999-10), pages 1-54, XP002149187 paragraphs '4.2.11!-'4.2.12!	1,15,22

☐ Further documents are listed in the continuation of box C.

☐ Patent family members are listed in annex.

° Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
21 January 2002	28/01/2002

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Van Staveren, M

1

[19] 中华人民共和国国家知识产权局

[51] Int. Cl[7]
H03M 13/27
H04L 1/00

[12] 发明专利申请公开说明书

[21] 申请号 01817961.4

[43] 公开日 2004 年 1 月 28 日

[11] 公开号 CN 1471762A

[22] 申请日 2001.7.19 [21] 申请号 01817961.4
[30] 优先权
[32] 2000. 9.13 [33] US [31] 60/232,224
[32] 2001. 1.11 [33] US [31] 60/260,930
[86] 国际申请 PCT/US01/22807 2001.7.19
[87] 国际公布 WO02/23740 英 2002.3.21
[85] 进入国家阶段日期 2003.4.25
[71] 申请人 交互数字技术公司
地址 美国特拉华州
[72] 发明人 S·M·沙赫里尔

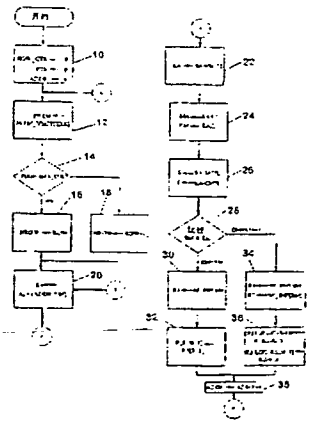
[74] 专利代理机构 北京纪凯知识产权代理有限公司
代理人 戈 泊 程 伟

权利要求书 7 页 说明书 14 页 附图 16 页

[54] 发明名称 第三代频分双工调制解调交织器

[57] 摘要

发明一种方法及装置来对扩展交织数据块进行解交织处理，尤其适用于无线通信系统中，如由第三代合作伙伴项目(3G)标准采用的系统中。 数据基于顺序元素被处理，其中每个元素具有一个预先确定数目的比特位数 M，比特位包含在连续数据字 W' 的一个块中。 元素以顺序地方式从字组 W' 块中抽取出来，每个被抽取的元素形成一个或者两个在字组 W' 中连续交织的字。 元素被保存在解交织器存储器中字组 W 中选定的位置，从而当完成抽取和写入所有元素时，从解交织器存储器中可以连续地读出字组 W，其对应于初始数据位块(从其中产生了交织元素块)。 采用其他传统的处理对解交织扩展字进行收缩，以在接收器中重新产生出数据位块，该数据位块与在发送器中最初指定发送的数据相一致。



ISSN1008-4274

知识产权出版社出版

1．一种在通信系统中处理数据的方法，其通过有选择地对 T 个连续扩展的位长度为 M 的数据元素块（包含在位长度为 L'的连续数据字组 W'中）进行重新排序，以产生一组连续的具有位长度为 L 的数据字，该字中以选定的顺序包含有 T 个扩展的数据元素，这个顺序基于对一个具有 C 元素列和 N 行的矩阵的交织器映射，其中 L 和 L'为大于 M 的整数，C 不等于 L，并且矩阵的后 r 列中具有 N－1 行，其中 r<C，且 r = (C * N) – T，该方法包括：

从连续数据字组 W'中连续地抽取 T 数据元素；

为预先确定的交织器列序列中起始列的第一行的最先抽取的元素确定矩阵映射位置；

对每个随后抽取的数据元素，在紧接前一数据元素相同列的下一行（行 n 列 i）确定矩阵映射位置，或者，如果该列没有下一行，则取预先确定交织器列序列中下一列的第一行位置；

对应于 C*N 矩阵，确定在数据字组 W 的本地存储器中 M 位连续地址的逐行连续映射；

对每个数据元素，对应于元素确定的矩阵映射位置，在字组 W 中的一个字中或者跨越两个连续字确定一个连续位地址；并且

在其确定的地址中存储每个数据元素。

2．如权利要求 1 所述方法，其中在接收器调制解调中执行的步骤作为第二解交织处理，其进一步包括：

在保存了后 T 个连续数据元素之后，从本地存储器中连续读出数据字 W，从而该 T 个元素在解交织序列中也具有连续顺序，对应于一个具有 T 个数据位的块，从其中在发送器调制解调器中可产生 T 个连续扩展交织的数据元素块。

3．如权利要求 1 所述方法，其中每个元素的连续位地址被确定在数据字组 W'中的一个字中或者跨越两个连续字中，该元素抽取包括确定一个起始位地址和一个结束位地址，其对应于将被抽取的数据元素，将位于确定的位地址之间和该地址本身中的数据保存在一个寄

存器中。

4. 如权利要求 3 所述方法，其中字组位于存储器中的连续地址中，起始于地址 A_0，在该处第一个被抽取的元素的位地址是 A_0，且每个随后抽取的元素的起始位地址是前一抽取元素的地址加上 M。

5. 如权利要求 1 所述方法，其中：

连续数据字组 W'被顺序地读入到第一和第二寄存器，从寄存器中将每个元素抽取并保存在第一流水线寄存器中；

字组 W'中的第一和第二顺序字最初被分别存储在第一和第二寄存器中；

从第一寄存器中的第一个字的前 M 位中抽取出第一个数据元素；

从第一寄存器中字的比特位开始抽取每个随后的数据元素；并且

当第一寄存器中字 W'的所有位被抽取之后，将在第二寄存器中的字 W'传送到第一寄存器中，并且下一个连续的字 W'被存储在第二寄存器中。

6. 如权利要求 5 所述方法，其中：

第一寄存器具有地址从 A_0 到 $A_0+(L'-1)$，第二寄存器具有地址从 A_0+L' 到 $A_0+(2L'-1)$；并且

第一寄存器中的每个随后元素的起始地址被确定为地址 A_0+SA，它基于前一个元素的起始地址 A_0+SA'，从而

当 $SA'+M<L'$, $SA=SA'+M$，且

当 $SA'+M \geq L'$, $SA=(SA'+M)-L$，并且在抽取之前，把在第二寄存器中存储的字保存到第一存储器中，并且把字组 W'中的下一个连续的字保存到第二寄存器中。

7. 如权利要求 5 所述方法，其中对每个元素的矩阵映射位置行和列的确定与在第一流水线寄存器中元素的存储是并行的，从而定义了第一级处理的一个周期。

8. 如权利要求 7 所述方法，其中把在第一流水线寄存器中的元素保存到第二流水线寄存器中，并且并行地确定本地存储器的地址信

息，从而定义了第二级处理的一个周期。

9. 如权利要求 8 所述方法，其中本地存储器中包括位地址 A_0' 到 A_0' + (T * M)-1，其中字组 W 中的每个连续字分配有 L 个连续位地址，对每个数据元素在本地存储器的起始地址 LSA，由 LSA = A_0' + ((n * C) + i) * M)确定，其中对于第一个数据元素，LSA = A_0'。

10. 如权利要求 9 所述方法，其中：

第一寄存器中具有地址 A_0 到 A_0+(L' – 1)，第二寄存器具有地址从 A_0 + L'到 A_0 + (2L' –1)；

第一数据元素的起始地址定义为 A_0 + 0；并且

每个随后元素的起始地址确定为地址 A_0 + SA，它基于前一个元素的起始地址 A_0 + SA'，从而

当 SA' + M < L', SA = SA' + M，且

当 SA' + M ≥ L', SA =(SA' + M) - L，并且在抽取之前，把在第二寄存器中存储的字保存到第一存储器中，并且把字组 W'中的下一个连续的字保存到第二寄存器中。

11. 如权利要求 8 所述方法，其中在第二流水线寄存器中的元素至少有一段保存到本地存储器中字组 W 的一个字中，从而定义了第三级处理的一个周期。

12. 如权利要求 11 所述方法，其中在第二级处理周期中为每个数据元素确定本地存储器起始地址 LSA 和本地存储器结束地址 LEA，并且当 LSA 和 LEA 是在本地存储器字组 W 中的两个连续字时，在第三级处理周期中把第二流水线寄存器中元素的第一段保存在两个字的一个之中，而在第一和第二处理级中停止一个周期。

13. 如权利要求 12 所述方法，其中，除了前后的元素外，每当第二级处理周期开始，第一级处理周期和第三级处理周期也开始。

14. 如权利要求 13 所述方法，其中当在第二寄存器中的字传送到第一寄存器并且将字组 W'中下一连续字保存到第二寄存器中时，

第一、第二、第三级处理都停止。

15．一种交织器，其通过有选择地对 T 个连续的位长度为 M 的数据元素块（包含在位长度为 L'的连续数据字组 W'中）进行重新排序，以产生一组连续的具有位长度为 L 的数据字 W，该字中以选定的顺序包含有 T 个数据元素，这个顺序基于对一个具有 C 元素列和 N 行的矩阵的交织器映射，其中 L 和 L'为大于 M 的整数，C 不等于 L，并且矩阵的后 r 列中具有 N－1 行，其中 r<C，且 r＝(C＊N)－T，该交织器包括：

一个存储器件，从其中可访问连续字 W'；

第一流水线寄存器，用于接收从存储器件中抽取出来的数据元素；

一个矩阵位置寄存器器件，用于接收矩阵位置数据，该数据与正在第一流水线寄存器中存储的元素有关；

第二流水线寄存器，用于从第一流水线寄存器中顺序地接收数据元素；

一个本地存储器；

一个本地地址寄存器器件，用于接收本地存储地址数据，该数据与正在第二流水线寄存器中存储的元素有关；

第一级处理电路，其包括

电路单元，其从存储器件中顺序抽取数据元素并将每个顺序抽取的元素保存在第一流水寄存器中，并且

矩阵映射电路，用来在矩阵位置寄存器器件中产生并保存相应的矩阵位置数据；

第二级处理电路，其根据来自保存在矩阵位置寄存器器件中的矩阵位置数据，在本地地址寄存器器件中产生并保存本地存储地址数据，该地址对应于一个正在从第一流水线寄存器传送并保存在第二流水线寄存器中的元素；并且

第三级处理电路，用于接收保存在第二流水寄存器中的数据元素，并有选择地将每个数据元素保存在本地存储器中字组 W 中的顺序字中，该过程基于相应的保存在本地地址器件中的地址数据。

16. 根据权利要求 15 中所述交织器，其中 L'不能被 M 整除，存储器件中包含有第一和第二 L'位寄存器，从字组 W'中来的字被顺序地传送到其中，并且第一级抽取电路从 M 顺序位地址中抽取元素，该位地址起始于选择性确定的第一 L'位寄存器的位地址。

17. 根据权利要求 15 中所述交织器，其中矩阵位置寄存器器件是单个寄存器，且矩阵映射电路中包括一个列索引寄存器、一个行索引寄存器和一个交织器矢量存储器，它们有选择地组合起来，从而行索引寄存器被增加或者被复位，同时为每个元素产生矩阵位置数据，索引寄存器被增加，并且复位行寄存器，根据存储在其中的交织器矢量，索引寄存器可用来增加交织器矢量存储器的输出。

18. 根据权利要求 17 中所述交织器，其中对每个数据元素，行寄存器输出一个值 n，且交织器矢量存储器输出一个值 i，值(n * C) + i 被存储在矩阵位置寄存器中，其与存储在第一流水线寄存器中的一个元素有关。

19. 根据权利要求 18 中所述交织器，其中 L' = 32，C=30，M =6，存储器件中包含有第一和第二 L 位寄存器，从字组 W'中来的字被顺序地传送到其中，并且第一级抽取电路从 M 顺序位地址中抽取元素，该位地址起始于选择性确定的第一 L'位寄存器的位地址。

20. 根据权利要求 15 中所述交织器，其中 L 不能被 M 整除，本地地址寄存器器件中包括一个起始位地址寄存器、一个结束位地址寄存器、一个起始字地址寄存器和一个结束字地址寄存器，第二级处理电路为本地地址寄存器器件中的各个寄存器计算起始位地址数据、结束位地址数据、起始字地址数据和结束字地址数据，该计算基于存储在矩阵位置寄存器器件矩阵位置数据（MPD）、数据元素位长度 M 和数据字位长度 L，同时将第一流水线寄存器中来的元素传送到第二流水线寄存器中。

21. 根据权利要求 20 中所述交织器，其中第二级处理电路计算起始位地址数据，将 MPD 对 L 取模；计算结束位地址数据，将（MPD

+M）对 L 取模；计算起始字地址数据，取 MPD/L 的最大整数；以及结束字地址数据，为（MPD+M）/L 的最大整数。

22. 根据权利要求 21 中所述交织器，其中当第一级处理电路从存储器器件中抽取一个新的单元并将其保存到第一流水线寄存器时，定义为第一级周期，当一个元素从第一流水线寄存器传送到第二流水线寄存器时，定义为第二级周期，当第三级电路有选择地将至少一段来自于第二级流水线寄存器中的元素存储到本地存储器中字组 W 中的一个字时，定义为第三级周期，其中第二级处理电路与第一级处理电路和第三级处理电路相关联，从而除了前后 T 个连续元素以外，每次当第二级周期开始时，第一级周期和第三级也开始。

23. 根据权利要求 22 中所述交织器，其中第三级处理电路中包括一个比较器，用于比较起始字地址寄存器中内容与结束字地址寄存器中的内容，它们与数据存储电路和第一和第二级处理电路相关，从而：

当起始字地址数据与结束字地址数据相等时，在第二流水线寄存器中的元素被保存到字组 W 中的一个字中，对应在该字中比特位位置中的起始字地址数据，该字对应于保存在起始地址寄存器中的起始位地址数据，一直到对应于保存在结束位地址寄存器中的结束位地址数据中地址；并且

当起始字地址数据与结束字地址数据不相等时，第一和第二级处理电路停止一个周期，同时保存在第二流水线寄存器中的元素的第一段将被保存在字组 W 中的一个字中，该字对应于在一个比特位位置中的起始位地址数据，从对应于保存在起始位地址寄存器中的起始位地址数据的位地址开始，一直到该字的一个结束位，从而完成一个第三级周期，并且在随后的一个第三级周期中，第一和第二级处理电路没有停止，保存在第二流水线寄存器中的该元素的第二段将被保存在字组 W 中的下一顺序字中，对应于结束字地址数据，其起始于该字的第一位一直到对应于结束位地址数据中的比特位位置，结束位地址数据保存在结束位地址寄存器中。

24．一种交织器，其通过有选择地对 T 个连续的位长度为 M 的数据元素块（包含在位长度为 L'的连续数据字组 W'中）进行重新排序，以产生一组连续的具有位长度为 L 的数据字 W，该字中以选定的顺序包含有 T 个数据元素，这个顺序基于对一个具有 C 元素列和 N 行的矩阵的交织器映射，其中 L 和 L'为大于 M 的整数，C 不等于 L，并且矩阵的后 r 列中具有 N－1 行，其中 r<C，且 r = (C * N) – T，该交织器包括：

一个存储器件，从其中可访问字组 W'中的连续字 W'；

一个流水线寄存器，用于接收从存储器件中抽取出来的数据元素；

一个本地存储器；

一个本地地址寄存器器件，用于接收本地存储器地址数据，该数据与正在流水线寄存器中存储的元素有关；

元素抽取电路，用于从存储器件中顺序地抽取数据元素，并将每个顺序抽取的元素保存到流水线寄存器器件中；

矩阵映射电路，用于为每个被抽取的元素产生矩阵位置数据；

地址处理电路，用于从来自矩阵位置数据（对应于保存在流水线寄存器器件中的一个元素）在本地地址寄存器器件中产生并保存本地存储地址数据；以及

存储处理电路，用于连续地接收保存在流水线寄存器器件中的数据元素，并根据保存在本地地址寄存器器件中相应的地址数据，将每个数据元素保存在本地存储器中字组 W 中连续的字中。

第三代频分双工调制解调交织器

技术领域

本应用有关于在电信系统中对数据进行交织处理。特别是，用于数据解交织的方法和装置。

背景技术

人们已经知道，在无线电通讯技术中，需要通过所谓的交织过程将数据加扰，以便将数据从一个通信基站发送到另一个通信基站。然后在接收基站通过解交织过程将数据解扰。

在第三代合作伙伴项目（Third Generation Partnership Project，3G）无线系统中，为频分双工（FDD）调制解调物理信道数据指定了具体的数据交织类型。在3G系统中物理信道数据是以具有预先规定比特位长的字进行处理的，目前每个字长指定为为32位。

为了在FDD物理信道中进行通信，需要指定连续的数据字，该字中包含有任意数目的连续数据位所组成的数据块。在准备每个数据块以用于在信道中传送时，该数据被逐行地映射到一个矩阵中，该矩阵具有预先规定的列数目。较好情况下列数比一个字中的位数要少。目前在3G中指定了30列用于对包含在32位字中的数据位块的物理信道进行交织。

举例说，将处于10个32位字w0-w9中的310位数据位块（包含于位$w_{0,0}$-$w_{9,21}$）映射到一个30列的矩阵中，显示如图1。310位数据位块映射到30列矩阵中具有11行。因为数据块总共有310位，所以后20列，列10—29分别比前10列少一个数据位。

是否矩阵列所有的位都具有映射到它们中的数据位取决于数据块的位数。比如，一个300位的数据位块映射到一个30×10矩阵中，将完全填充所有的列，因为300恰好被30整除。一般情况下，对于具有T个元素的块映射，C列N行矩阵的后r列只在N—1行中具有数据，其中r=(C*N)-T，且r<C。

在将数据位映射到交织矩阵之后，按照预先规定的顺序将列顺序重排，然后按逐列的顺序将数据位写入新的字组 w'，以得到一个由连续字组 w'中的连续位 $w'_{\#,\#}$所组成的交织数据块。

举例说，在图 1 中，将包含在字 w0-w9 中的 310 位数据块有选择地存储在字 w'0-w'9 中，并符合在图 2a，2b 中所希望的交织器列顺序。对于字组 w0-w9，在相应的 10 个字 w'0-w'9 形成的交织块中，以一种经过充分重排/加扰的顺序包含有所有初始字 w0-w9 中的 310 位数。如图 2a 所示，交织字 w'0 是由图 1 中的列 0、20 和 10 中的数据位所形成。初始字 w0-w9 中的位 $w_{\#,\#}$与交织字 w'0 中的位 $w'_{0,0}$-$w'_{0,31}$ 的对应参见图 2b。

在 3G 中，在数据被发送到接收基站之前，交织数据要经过多种处理。为了提高信噪比，将数据位长度结构扩展 M 倍。目前的位扩展指定为 6 倍。因此在 3G 系统中，物理信道数据块的每个交织数据位可被扩展为一个 6 位元素。

举个例子，在图 2a 和 2b 示例中的 10 个交织数据字 w'0-w'9 被扩展为具有 59 个字的数据块 W'0-W'58，以便于发送，显示如图 3。图 4a-4f 显示了将字 w'0 的交织位 $w'_{0,0}$-$w'_{0,31}$ 与扩展后的字 W'0-W'5 中交织的具有 6 比特位元素的 T'0-T'31 的对应示例。

因为元素的位长度不能整除字的位长度，所以有些元素会跨越两个连续字。比如，在图 4a 和 4b 中，元素 T'5 部分包含在字 W'0 中，部分包含在下一个字 W'1 中。

在接收基站中，在经过接收和处理之后，所接收的扩展交织元素块，比如处于 59 个字 W'0-W'58 中的比特位 $W'_{0,0}$-$W'_{58,3}$，必须经过解交织，也就是解扰，将数据解扰成其初始顺序形式。所以提供一种方法和装置来对扩展列交织数据块进行快速而有效的解交织是极为有利的。

发明内容

发明一种方法及装置来对扩展的交织数据块进行解交织，尤其是用于无线电通讯系统，如采用第三代合作伙伴项目（3G）标准的系统中。数据基于连续元素形式被处理，其中每个元素具有一个预先确

定的比特位数目 M，该比特位包含在连续数据字块 W'中。元素从字块 W'中被顺序地抽取出来，每个元素从字 W'集合中的单个或两个连续的交织字中抽取。在解交织器存储器内，元素被存储在字组 W 中选定的位置，从而当所有元素抽取和写入结束后，能够读取出解交织器存储器中内的字组 W，以对应于创建交织元素块的初始数据比特位块。通过其他传统的处理方式，将对解交织的扩展字进行收缩，在接收器中再生出与在发送器指定发送相一致的的数据位块。

尽管本方法和装置明确设计用于实现 3G FDD 接收器调制解调的第二解交织功能，但是该发明可在其他应用中方便地对扩展数据块进行加扰或解扰。

较好地，结合计算解交织器存储地址和对其中数据元素的选择性存储，可采用一个多级流水线配置来处理元素。建议采用三级流水线处理，可实现数据流量高达每秒 60 兆位。而且，多个解交织器可并行使用来处理多个数据块，比如，每个解交织器可用于一组不同的物理信道，从而解交织过程不会对整个通信系统的速度有负面影响。但是，由于对每个信道的物理信道处理目前确定为每秒 380 千位，结合所建议的结构，单个解交织器的速度已经完全足够处理 3G FDD 接收器调制解调中所有物理信道中的数据元素块了。

对于本领域中一般的技术人员而言，通过附图和以下详细的描述可以了解本发明的其他目标和优点。

附图说明

图 1 显示将包含在 10 个 32 位字 w 中的 310 位数据比特位映射到一个具有 30 列的矩阵中。

图 2a 显示按照目前 3G 交织器列顺序规范将图 1 中的数据比特位块映射到字组 w'的交织比特位 $w_{\#,\#}$ 组成的块中。

图 2b 显示从图 1 中的数据字 w 的比特位映射到一个交织字 w'中。

图 3 显示将图 2a 中交织比特位块字组 w'扩展映射到扩展的交织的 6 比特位元素字组 W'中。

图 4a-4f 显示将图 2a 中的一个交织比特位块字 w'位映射到 6 倍

扩展的元素交织字组 W'上。

图 5a 和 5b 显示将图 3 中字 W'的扩展交织元素块的比特位映射到具有 30 个 6 比特位元素列的交织器矩阵中。

图 6 显示在将数据解交织元素块的一个字 W 按比特位和元素映射到图 5a 和图 5b 中的矩阵。

图 7a 和 7b 显示图 5a 和 5b 中矩阵的相应的解交织扩展元素和比特位映射。

图 8 显示了图 7 中解交织的扩展元素字 W 与图 1 中初始数据比特位块字 w 的对应。

图 9 是采用本发明中通信系统的接收器处理元件方框图。

图 10a 和 10b 是根据本发明所进行解交织的一般方法的流程图。

图 11a－11c 是根据本发明的一个三级流水线交织器原理图。

具体实施方式

作为目前 3G 规范的一部分，扩展的交织数据块，比如，在 FDD 接收器物理信道中的数据，要被接收下来并且必须经过解交织以进一步处理。FDD 接收器被分成多个子块。其中之一称之为接收器复合信道（Receiver Composite Channel, RCC）。RCC 框图显示如图 9，它由物理信道反映射、第二解交织、物理信道聚合、第二 DTX 和 P 指示位分离以及传输信道（TrCH）解复用。有效情况下，接收器合成信道的操作与发送合成信道中发送器调制解调所执行的功能相反。

本发明特别适用于 FDD 接收器第二解交织器的结构。每个物理信道（PyCH）中待发送比特位序列通过交织过程被加扰，然后扩展成等长的数据包；每个数据包中含有数目很少的比特位。这些比特位组中的每一组命名为数据元素。目前，在建议的实施例中，3G FDD 物理信道数据元素长度被指定为 6 位，即 M＝6。图 1－4 中显示一个示例，关于发送器调制解调交织和将 310 位数据位块扩展为 310 个交织的 6 比特位元素 T'的数据块。

扩展的交织数据元素以其交织顺序被发送出去。接收器从空中接收数据元素，并将它们以连续的 32 位数据字组 W'方式存储下来。在图 1－4 中显示的示例中，310 位数据块最初在发送端以 32 位字 w0-w9

的形式存储，在接收端作为数据元素 T'0-T'309 以 32 位字 W'0-W'58 的形式被接收并存储。

第二交织器是一个带有列内置换的块交织器，其对交织数据元素进行重排。交织矩阵具有 30 个元素列，从左到右编号为 0、1、2、…、29。行序号由用户提供作为外部参数 N，对于一个具有 T 个元素的数据块可计算出 N 值，即取 N*30≥T 的最小整数 N。

用于 3G FDD 调制解调的第二解交织器中的列内置换方式如下：

列序号	列内置换方式
30	{0, 20, 10, 5, 15, 3, 13, 23, 8, 18, 28, 1, 11, 21, 6, 16, 26, 4, 14, 24, 19, 9, 29, 12, 2, 7, 22, 27, 17}

表 1—解交织器的列内置换方式

第二解交织器的输出是一个位序列，它从映射到列内置换过的 N×30 矩阵中逐行读出。在整个 N×30 矩阵被输出的位置，通过删除在输入数据元素位序列中不存在的数据位而将输出数据进行剪切。

图 5a 和 5b 显示了示例接收数据元素 T'0-T'309 一个位映射，图中显示了 11 行×30 个元素列交织矩阵的左边和右边部分。在图 5a 和 5b 中，比如，第 0 列表示元素 T'0-T'10 比特位的位映射，它们包含在字 W'0, W'1 和 W'2 中。元素 T'5 的位是从两个字中抽取出来的，即 W'0 和 W'1；元素 T'10 的位是从两个中抽取出来的，即 W'1 和 W'2。在图 5b 中，最底行没有元素，因为只有前 10 列是完全由数据元素填充。

图 6 和图 7 表示基于交织器矩阵映射，元素 T'在整个对字组 W 中元素选择存储中是如何被记录的。T'0, T'124, T'258, T'186 以及 T'31 的前两位存储在 W0 的 32 位中，其相应地对应于记录元素 T0 到 T4，以及元素 T5 的前两位。其结果是，元素 T'0 到 T'309 的选择性存储基于交织器矩阵映射，形成一个 32 位字 W 的序列，即 W0 到 W58，其中包含有记录元素 T0 到 T309，显示如图 7a-7c。图 8 表示

初始字 w0-w9 如何对应于字 W0－W58，显示了记录元素 T0－T309 与 310 个初始数据块位 w0,0-w9,21（如图 1 显示）的对应关系。

为了在矩阵中正确地放置元素 T'0-T'309，以便于元素 T'0-T'309 能够被逐行读出成为连续字 W0－W58，对每个元素 T'要进行选择处理，可用流程图 10a 和 10b 来表示。

在 3G FDD 调制解调接收器中，扩展的交织数据被分配到不同的物理信道，并且被存储在名为 M_INP 的随机存储器（RAM）中，以便于由解交织器处理。位流被分割为 32 位字，并且这些字被存放在 M_INP 中连续的位置。在图 1－4 中所显示的示例中，包含在字 W'0－W'58 的元素 T0－T309 的位流将被存储在 M_INP 的连续地址中。在图 10a-10b 中的流程图说明了解交织器如何从 M_INP 中读取数据，进行解交织并将它们写入到本地存储器 M_LOC 中。整个过程包括，从 M_INP 中逐个元素读出数据，执行地址翻译，并将元素写入到 M_LOC 中某位置，这个位置对应于发送器中执行交织处理之前的元素在存储器中的初始位置。图 5－8 显示了元素 T'0-T'309 的交织器映射与重排的位于字 W0－W59 中的元素 T0-T309 中的对应关系，并且在图 8 中，显示了与在发送端中包含在字 w0-w9 中的初始位序列的对应关系。

表 2 中提供了在流程图 10a 和 10b 中所用到的参数列表。

开始时，在处理过程中用到的变量在块 10 中初始化。地址增长器 ADDR 和行计数器 ROW_CTR 和列索引指针 IDX 被置为 0。将预先规定的排列顺序存储在一个名为 PERM_VECT 的矢量中。在 PERM_VECT 中置换过的列顺序较好地显示在表 1 中，以用于 FDD 调制解调接收器第二解交织器。在步骤 12，基于 IDX 的值从 PERM_VECT 中输出一个 PERM 值，IDX 的值指示当前正在处理元素的列位置。

在接下的几步操作 14,16,18 中确定在列号 PERM 中的行号，并将变量 NROW 设置成该值。设置一个恒定参数 MAX_COL，从而列 0,1，2，…，MAX_COL－1 具有其中行的“行”序号，并且列 MAX_COL，…，C－1 在其中具有“行数－1”行。根据这一结果以

及当前的 PERM 值，可相应地设置变量 NROW。

参数	描述
ADDR	在 M_INP 中的字地址增长器，字 W'起始于地址 A_0
T	在数据块中的元素总数
ROW_CTR(或 n)	在列 PERM 中对行计数的计数器
PERM_VECT	列置换矢量
COL（或 C）	在置换矩阵中的列序号
ROW（或 N）	在置换矩阵中的行序号
PERM（或 i）	由 IDX 指向的 PERM_VECT 元素
IDX	PERM_VECT 元素指针
MAX_COL	常数值，等于 T-(C*(N-1))
NROW	在列序号 PERM 中的行数
SA	元素的起始位地址
EA	元素的结束位地址
SM	元素的起始字地址
EM	元素的结束字地址
S	在 SM 中元素的起始位位置
E	在 EM 中元素的结束位位置
M	在每个元素 T'#或 T#中的比特位数
R，R1，R2	存储寄存器
L'	在字组 W'中每个字的位数
L	在字组 W 中每个字的位数

表 2—流程图参数列表

在步骤 20、22 中，采用起始地址 A0、当前 ADDR 值和元素长度 M、起始和结束位地址，在 M_INP 中的当前数据元素的 SA 和 EA 可分别被确定。用字的比特位长度 L'去除 SA 和 EA，并舍弃任何余数（或者相当于右移 5 位），在字集合 W'中产生相应的字地址。这

些字地址分别是 SM 和 EM。然后在第 26 步，在存储字中数据元素（由 SM 和 EM 标识）的起始和结束比特位位置可分别计算出为 S 和 E。S 和 E 可能会包含在字 W'集合的单个存储字中，或者分别位于两个连续的存储字中。以下的操作 28,30,32,34,36 显示这两种情况是如何处理的。

在流程图中的下一个操作 28 是比较 SM 和 EM 的位置。如果元素位于字组 W'中的单个字中，即 EM=SM，然后进入第 30 步，位于 SM 中的字可从 M_INP 中取出。在第 32 步，从元素的比特位位置抽取出元素，位置由 S 和 E 指明，并且将值赋给寄存器 R。在另一方面，如果元素包含在字组 W'的两个字中，即 EM=SM+1，则从 M_INP 中要访问两个字。相应地，取出 SM 中的字分配给寄存器 R1，取出 EM 中的字分配给寄存器 R2，显示如步骤 34。然后在步骤 36 中，将元素的比特位从 R1 和 R2 中抽取出来并分配给寄存器 R。因此，在两种情况下，包含在字 W'（存储在 M_INP 中）的交织元素的所有比特位被抽取。最后，地址计数器 ADDR 增加，以初始化对下一个元素的抽取。

在随后的步骤 40－60 中，如图 10b 所示，用来确定 M_LOC 中字和位的位置，以便从该处抽取要存储的元素，然后访问该字，将元素放置在字中正确的比特位位置，并且将字写回到 M_LOC 中。这些步骤可作为单次读－修改－写操作。

在步骤 40－42 中将确定起始和结束映射位地址 SA 和 EA，在该处抽取的元素（在步骤 32 或 36 中存储在 R 中）将被存储到 M_LOC 中。根据对在步骤 30,32 或 34,36 中抽取元素的行和元素列映射，在步骤 40 中对起始地址进行计算。矩阵位置的计算是由行数（由 ROW_CTR 给出）乘以矩阵的列数 COL，加上当前的列数目 PERM（得自 PERM_VECT 矢量），也就是（ROW_CTR*COL）+PERM。因为每个元素具有 M 位，所以将结果乘以 M 可得到 SA。

在步骤 46 中，用集合 W 中字的位长 L 去除 SA 和 EA，并舍弃余数，将产生相应的字地址。这些字地址分别是 SM 和 EM。最后，起始和结束比特位位置（在寄存器 R 中抽取的元素存放在该处）分别

被计算为 S 和 E。在 L 不能被 M、S 和 E 整除的位置可能被包含在单个存储字中，或者跨越字组 W 中两个连续的存储字。在随后的步骤 48、60 中将描述这两种情况是如何被处理的。

在步骤 48 中，对地址 SM 和 EM 进行比较。如果被抽取的元素将被存放在单个字中，即 EM=SM，然后在第 50 步中将位于 SM 中的字从 M_LOC 中取出并存放在寄存器 R1 中。在第 52 步，在 R 中被抽取的元素值被写入到比特位位置，该位置由在 R1 中的 S 和 E 指明。最后，在第 54 步，将 R1 写回到 M_LOC 的存储位置 SM。

在另一方面，如果被抽取元素存放在具有地址 SM 和 SM+1 的两个连续字中时，在步骤 56 中从 M_LOC 中取出那些字，并分别存放在寄存器 R1 和 R2 中。然后，在步骤 58，根据 S 和 E，将在 R 中被抽取元素的比特位分别放置到寄存器 R1 和 R2 中正确的位置。最后，在步骤 60 中将寄存器 R1 和 R2 的内容分别写回到存储位置 SM 和 SM+1。

在步骤 62 中的下一操作是将行计数器 ROW_CTR 增 1，以指明下一个被抽取的元素 T'#将被存储在同一列的下一行。在步骤 64 中进行检查，以确定行计数器是小于或者等于当前列的行数 NROW。如果是这样，则过程继续到步骤 20 处理列成员 PERM 中的下一个元素。

如果 ROW_CTR 不小于 NROW，在步骤 64 中，则下一个被抽取的元素将被存储在与下列（由矢量 PERM_VECT 指定）的首行（0 行）相关的一个地址中。相应地，如果情况如此，则在步骤 66,68 中将 ROW_CTR 复位到 0 并且将 PERM_VECT 索引增 1。如果，在步骤 70 中，IDX 小于 COL，则解交织过程从步骤 12 重新开始，分配一个新 PERM 值，否则处理过程结束，因为所有数据块的 T 元素都已经过处理。

按照流程图 10a 和 10b，对一般的处理方法进行了描述，同时在硬件上对该过程的一个较好的实现显示在图 11a-11c 中。所建议的设计中包括一个三级流水线处理，其带有相关联的存储器 LOCAL MEMORY，用于存储数据的解交织位。第一级的并行处理元件显示

在图 11a 和 11b；第二和第三级处理显示在图 11c 中。

第一级操作首先从一个 2L'位矢量(由两个寄存器 REG3 和 REG4 的内容所确定）中抽取数据元素。寄存器 REG3 和 REG4 中从物理信道（PyCH）存储器中存储两个连续的 L'位字。对于所建议的 32 位字长，这两个寄存器组成一个 64 位长的位矢量。

配置一个寄存器 REG0、一个加法器 71、一个减法器 72 和一个选择器 73 来结合合并元件 74 以顺序的方式从寄存器 REG3 和 REG4 中取出具有长度为 M 位的元素，并将该元素存储在寄存器 REG2 中。为了初始化交织器，序列字 W'中的第一及第二个字起初分别存放在寄存器 REG3 和 REG4 中，而寄存器 REG0 初始化为 0。合并元件 74 从寄存器 REG0 中接收 0 值，从地址 0 开始到地址 M－1 抽取 M 比特位。从而，在 REG3 中初始字的前 M 位被抽取出来，该 M 位数据对应于第一个元素 T'0。合并元件 74 然后在流水线寄存器中存储所抽取的 M 个比特位。

基于选择器 73 的操作，寄存器 REG0 的值可经过加法器 71 增加 M，或者经过加法器 71 和减法器 72 增加 M－L'。如果将寄存器 REG0 中的值增加 M 后不超过 L'，则选择器 73 将寄存器 REG0 增加 M。否则，选择器 73 将寄存器 REG0 的值增加 M－L'。这可作为一个有效的模 L'函数，从而 REG0 的值总是小于 L'，以确保由合并元件 74 所抽取的元素的起始地址总是处于寄存器 REG3 的位地址 0－L'－1 之内。

在选择器 73 选择将寄存器 REG0 增加 M－L'的位置，用一个信号 EN 来触发 REG4 中的内容传送到 REG3，以及从外部存储器中的字组 W'中取出下一个连续字以存储在 REG4 中。在取字的过程中，整个流水线停止工作。对寄存器 REG0 中值的增加以及减去 L'与将寄存器 REG4 中的字 W'传送到 REG3 中有关，从而对元素的顺序抽取至少在元素（正从寄存器 REG3 中内容中抽取）的第一位中是持续进行的。

参考图 11b，交织器的定位值的计算与被抽取元素的抽取过程是并行的。通过从寄存器 N_REG 中获取一个当前的行值 n，并在乘法

器 75 中将它乘以交织矩阵的元素列数，可计算出矩阵的映射信息。在加法器 76 中，然后加上当前的列值 i，该值是从寄存器组 78 中输出的，其中包含有以矢量 PERM_VECT 表示的交织器列序列。寄存器组 78 的输出由索引寄存器 I_REG 控制，其根据矢量 PERM_VECT 将寄存器组 78 的输出值增加。

该矩阵映射电路中也包括一些元素，用来选择性地增加行索引寄存器 N_REG 以及列索引寄存器 I_REG。电路能够有效地保持相同的列，直到每个顺序行的值都被使用，然后将列增加到交织器矢量中的下一列，该矢量起始于该列的初始行。通过利用一个单元增加器 80 来实现这一任务，该增加器与行寄存器 N_REG 相关，用于在第一级处理中的每个周期将行值增 1。寄存器 N_REG 的输出还要在比较器 81 中与由乘法器 83 所确定的最大行值进行比较。对于该特定的列，最大行值是整个矩阵的最大行值 ROW 或者是 ROW－1。响应比较器 84，乘法器 83 产生一个输出。该比较器将目前正在由寄存器组 78 输出的列值与具有最大行值 ROW 的最大列值进行比较。

如果比较器 81 确定最大行数已经达到寄存器 N_REG 的最大输出值，则比较器 81 发出一个信号来将 N_REG 复位到 0，并且操作乘法器（MUX）86，连同索引寄存器 I_REG。单元增加器 88 也与索引寄存器 I_REG 有关，并且当从比较器 81 中收到一个信号时，MUX86 通过增加器 88 对 I_REG 的值增 1。否则，乘法器 86 在第一级周期中只简单地将相同的值保存在寄存器 I_REG 中。

参考图 11c，流水线交织器的第二级中包含一个处理周期，其中被抽取并保存在第一流水线寄存器 REG2 中的元素被传送并保存在第二数据流水线寄存器 REG9 中。在第二级处理中以并行的方式，相应的保存在寄存器 REG1 中的矩阵映射数据被用于计算相关的起始位地址数据（该数据存储在寄存器 REG5 中）、结束位地址数据（保存在寄存器 REG8 中）、起始字地址数据（保存在寄存器 REG6 中）和结束字地址数据（保存在寄存器 REG7）中。在第二级处理周期中，REG1 中的矩阵映射数据最初在乘法器 90 中乘以元素的位长度 M。然后在减法器 91 中通过从结果值中减去一个值来产生与模 L 相等的

数，以计算起始位地址数据，其中 L 是本地存储器 100 中数据字的位长度，本地存储器可有选择地存储被抽取的元素。在减法器 91 中被减去的值可这样计算得到，在除法器 92 中将乘法器 90 的输出用 L 去除，不保留余数，并在乘法器 93 中将该值乘以 L。除法器 92 的输出也提供了相应字的起始字地址，在其中至少有寄存器 REG9 中元素的第一部分被存储在本地存储器 100 中。

结束位地址数据这样计算，在加法器 95 中将乘法器 90 的结果加上 M－1，然后在减法器 96 中将该值减去一个计算得到的值，以产生一个模 L 值，然后存储在寄存器 REG8 中。该被减去的值在除法器 97 中由加法器 95 的输出除以 L，不保留余数，然后在乘法器 98 中将该结果乘以 L 而得到。除法器 97 的输出也提供了结束字地址数据，其保存在寄存器 REG7 中。

在流水线交织器的第三级中，基于在寄存器 REG5、REG6、REG7 和 REG8 中的数据，执行一个读－修改－写操作，以有选择地在本地存储器的寄存器 REG9 中保存元素值。开始，将寄存器 REG6 和 REG7 中的内容在比较器 99 中进行比较。如果其值相等，则在寄存器 REG9 中的元素将被保存在本地存储器 100 的单个字中。在这种情况下，寄存器 REG6 的值从复用器 101 传送到复用器 102，在那里它可能要结合一个基地址，可用来在系统中分配整个存储资源。

乘法器 102 的输出指出了字 W 的地址，在其中可写入寄存器 REG9 中的元素。该字可输出到一个解复用器 103，从其中一个合并元件可在连续地址中创建一个新字（在连续地址中，该字以寄存器 REG5 的值开始并以寄存器 REG8 中的值结束），该字是由 REG9 中元素的位值组成的新字，该字的其余位从解复用器 103 中的字的值得到复制。然后将在合并元件 105 中新形成的字存储回到该地址，在那里初始字被输出到解复用器 103。

其中寄存器 REG6 和 REG7 的内容不同，第一和第二级流水线延迟一个周期，以便于第三级能够执行一个有关该字的读－修改－写周期，该字由在寄存器 REG6 中的数据标识，然后重新开始所有各级的流水线周期来执行一个关于本地存储字的读－修改－写操作，该本地

存储字对应于存储在寄存器 REG7 中的结束字数据。在这种情况下，在有关于该字读一修改一写周期（对应于寄存器 REG6 中的起始字地址数据）中，第三级将保存在寄存器 REG9 中的元素的初始部分存储到本地存储器字中的最后几位，该存储字起始于由保存在寄存器 REG5 中的值所指明的比特位位置。在第三级的第二个周期中，此处第一和第二级时钟周期已经重新启动，将在寄存器 REG9 中元素的其余部分保存在该字中，它对应于在寄存器 REG7 中的结束字地址数据（从该字的初始位开始到在寄存器 REG8 中值所指明的位地址）。

在下一个数据位块中的所有 T 元素被处理完之后，经过解复用器 103，本地存储器中的连续字被读出，在系统中用于进一步处理。对 310 个元素数据块的处理之后，本地存储器的输出显示在图 5—8 的示例中，对应于由图 7c 表示的字序列。在 3G 系统中的进一步处理中，扩展的 6 位元素被收缩成单个比特位，例如，再生出初始的 310 个位数据块，其与初始产生于发送单元的数据具有相同的顺序。

可采用两种不同的技术来测试这个第二交织器的 3 级流水线。这些方法中的第一个是手工方法，称为回归（regression）。回归测试可这样进行，从 PyCH 存储器中抽取 30 个 32 位字，从它们中取出 6 位长的元素，并将它们传送到流水线中。测试周期基于手工节拍式仿真，寄存器和内部存储中所预期的内容由手工确定。这些值被拿来与从仿真中获得的实际值作比较。对大量的测试情况以及对所有的流水阻碍情况进行仿真。在手工设置的所有测试情况下，发现交织器功能正常无误。

接下来，用 C 语言独立地实现交织器。将一组测试矢量运用到 C 程序块中，对输出进行监视并写入到一个结果文件中。将相同的输入测试矢量组运用于 VHDL 模型。将两组输入矢量应用于测试中：

一组 201 个元素输入矢量和一组 540 个元素输入矢量。两个不同组的输入用来创建两个不同的交织器矩阵。201 个元素矩阵具有两个不同的行数，其中一个比另一个小。540 个元素矩阵只具有一个行尺寸。从而，测试中包括有两种不同类型的交织器矩阵可能的结构。测试结果显示，从 VHDL 模型和 C 语言模型中的输出矢量与两种输入

情况相匹配。

硬件由 Synopsys 逻辑综合器进行综合，采用德州仪器公司的 0.18 μm 标准单元库。采用门数如下所示。

标准单元数目（TI/GS30/标准单元）	1034
时序门	1844
组合门	3348
门数总计	5192

表 3 为交织器估计的总门数

该流水线式结构能够确保高速率流量，由于所用门数较少所以面积小而紧凑。虽然建议采用三级流水线，但是从所建议系统中（显示如图 11a-11c）舍去寄存器 REG1 和 REG2 将很容易实现两级设计。

在本领域中的一般技术人员可以在本发明的范围内采用其他变化和修改。

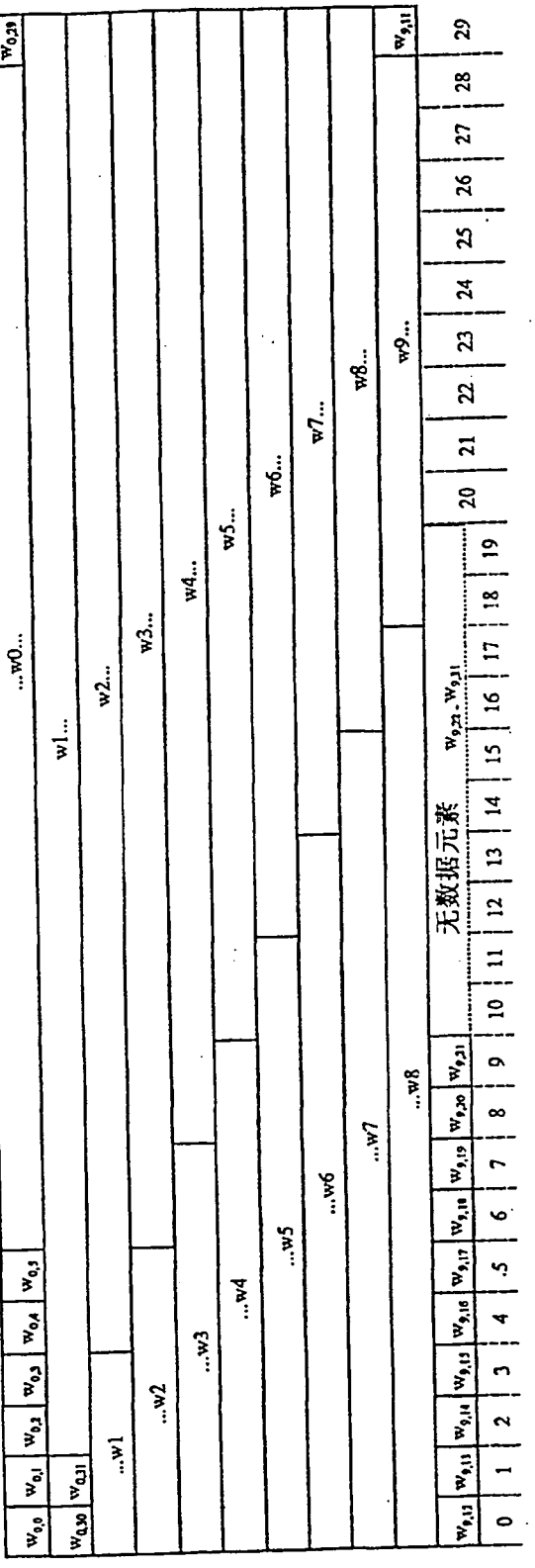


FIG. 1 (现有技术)

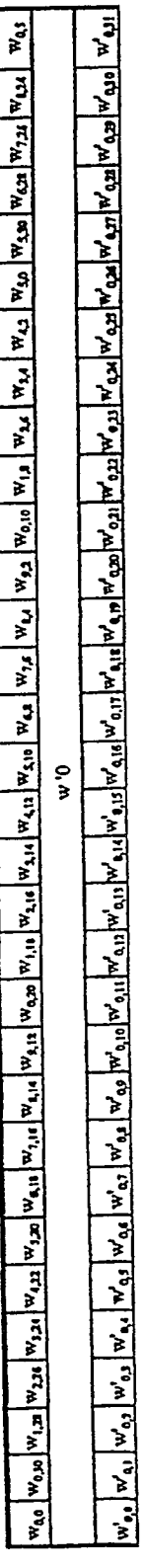


FIG. 2b (现有技术)

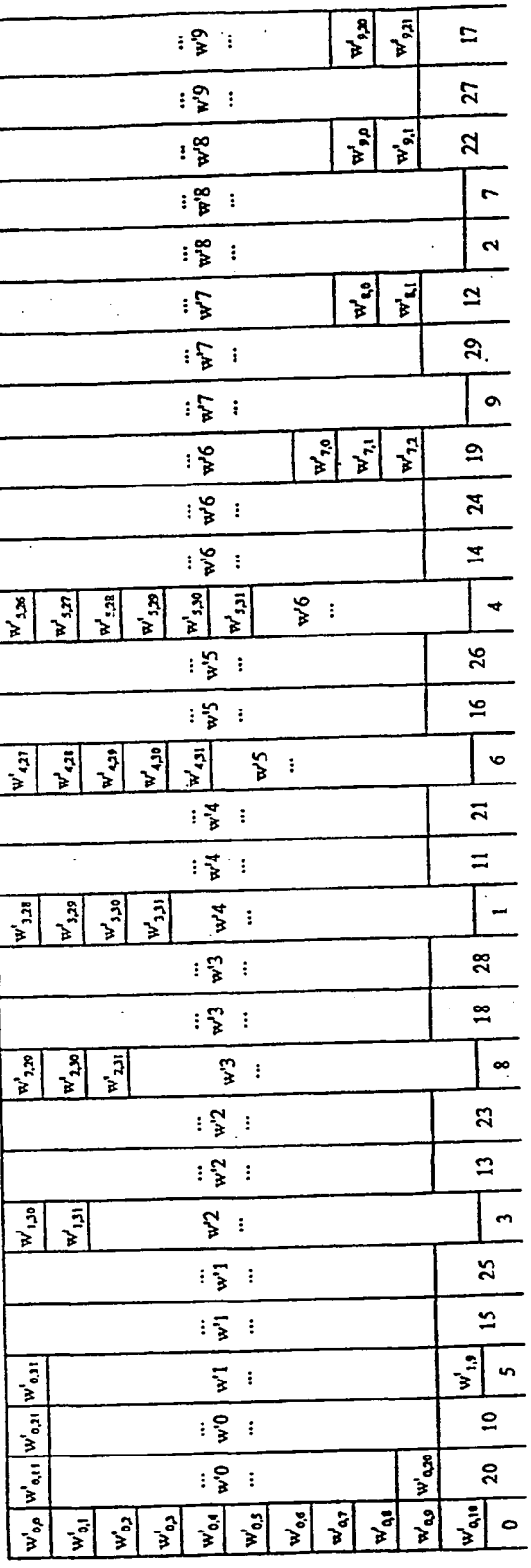


FIG. 2a (现有技术)

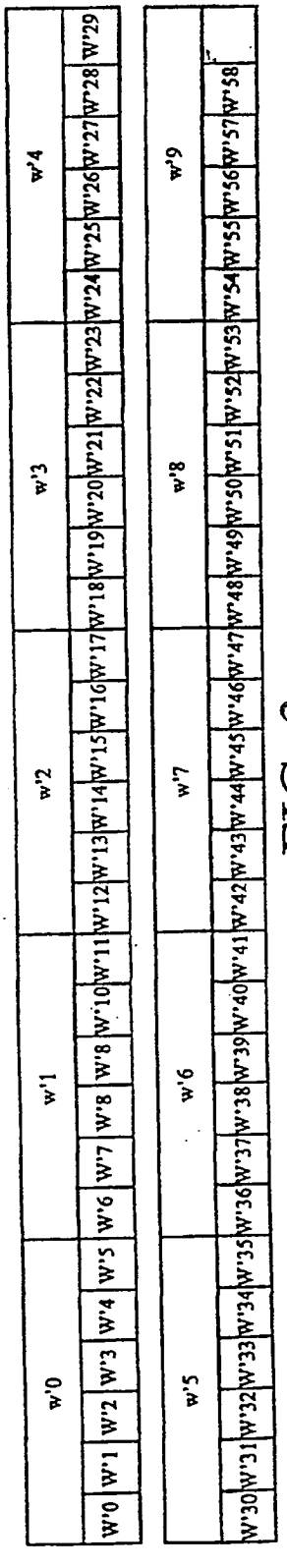


FIG. 3

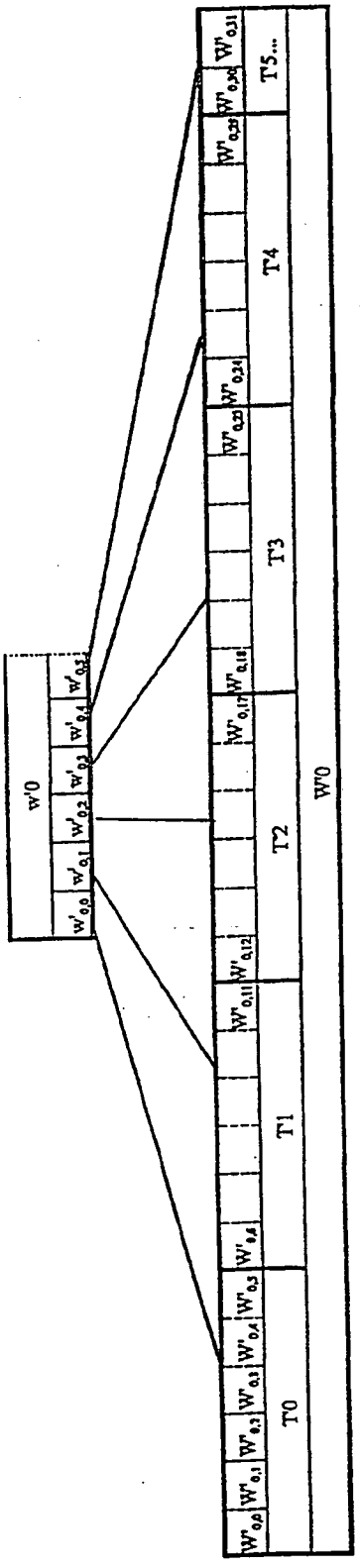


FIG. 4a (现有技术)

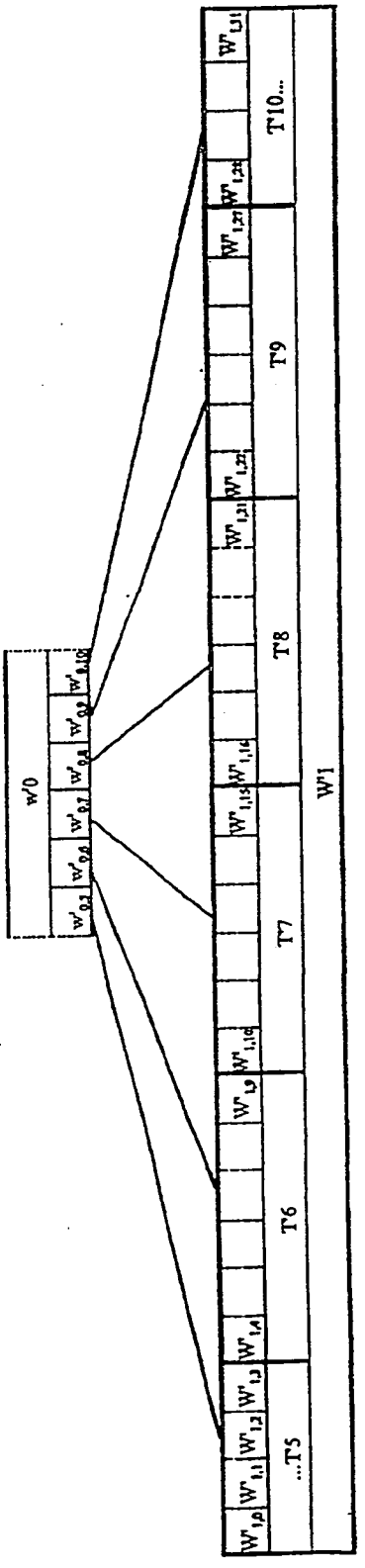


FIG. 4b (现有技术)

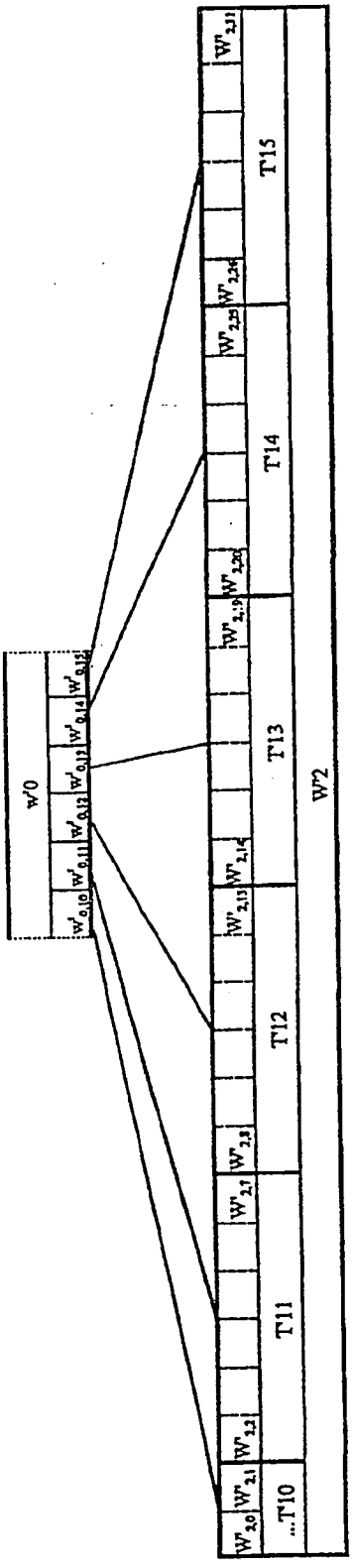


FIG. 4c（现有技术）

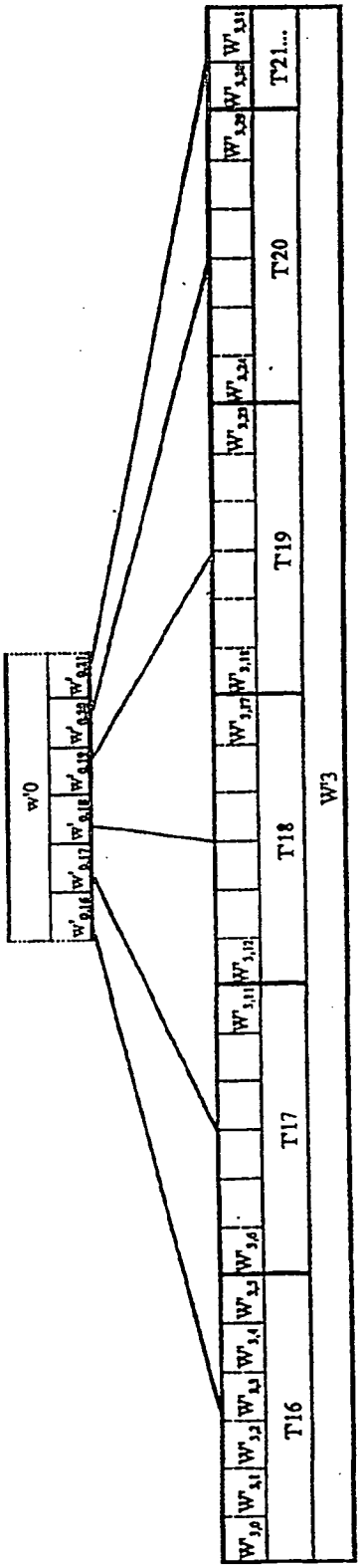


FIG. 4d（现有技术）

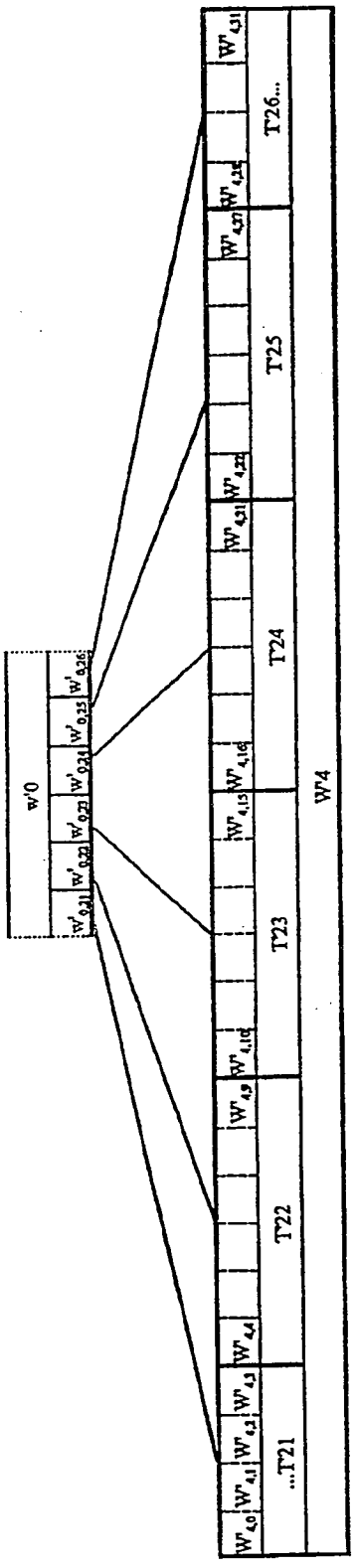


FIG. 4e (现有技术)

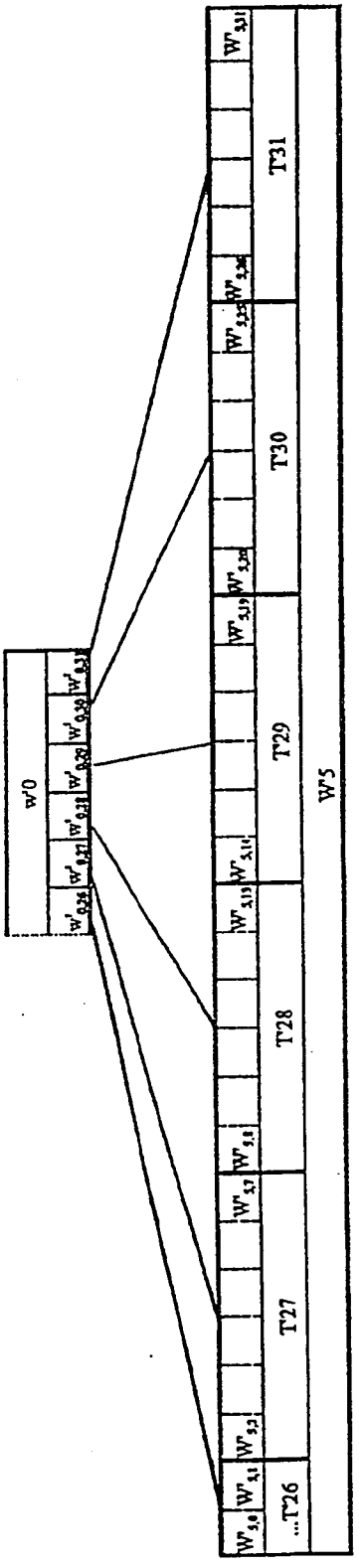
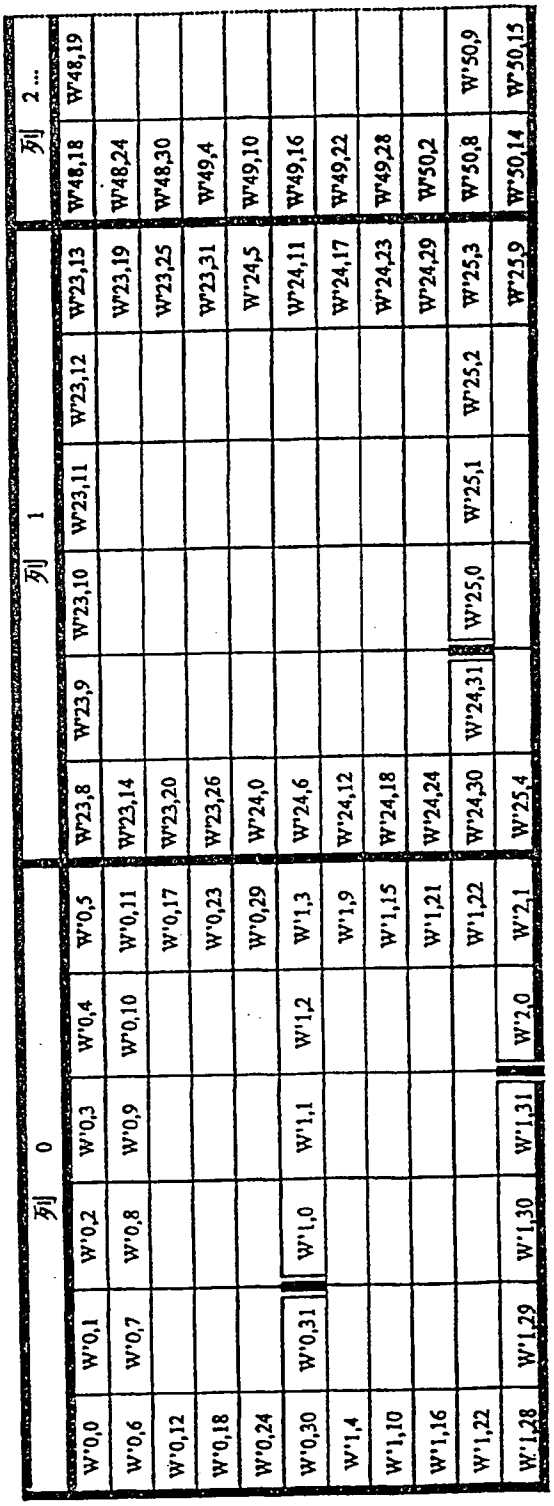
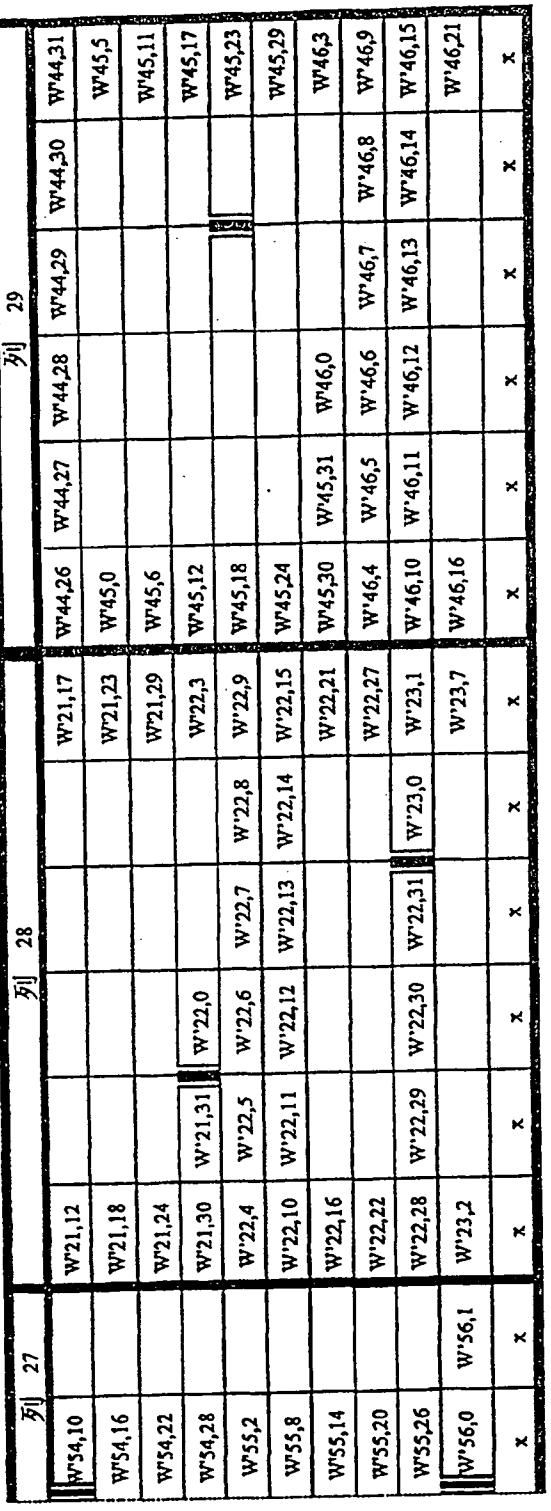


FIG. 4f (现有技术)



列 0						列 1						列 2...	
W'0,0	W'0,1	W'0,2	W'0,3	W'0,4	W'0,5	W'23,8	W'23,9	W'23,10	W'23,11	W'23,12	W'23,13	W'48,18	W'48,19
W'0,6	W'0,7	W'0,8	W'0,9	W'0,10	W'0,11	W'23,14					W'23,19	W'48,24	
W'0,12					W'0,17	W'23,20					W'23,25	W'48,30	
W'0,18					W'0,23	W'23,26					W'23,31	W'49,4	
W'0,24					W'0,29	W'24,0					W'24,5	W'49,10	
W'0,30	W'0,31	W'1,0	W'1,1	W'1,2	W'1,3	W'24,6					W'24,11	W'49,16	
W'1,4					W'1,9	W'24,12					W'24,17	W'49,22	
W'1,10					W'1,15	W'24,18					W'24,23	W'49,28	
W'1,16					W'1,21	W'24,24					W'24,29	W'50,2	
W'1,22					W'1,22	W'24,30	W'24,31	W'25,0	W'25,1	W'25,2	W'25,3	W'50,8	W'50,9
W'1,28	W'1,29	W'1,30	W'1,31	W'2,0	W'2,1	W'25,4					W'25,9	W'50,14	W'50,15

FIG. 5a



列 27	列 28	列 29
W'54,10	W'21,12	W'21,17	W'44,26	W'44,27	W'44,28	W'44,29	W'44,30	W'44,31
W'54,16	W'21,18	W'21,23	W'45,0					W'45,5
W'54,22	W'21,24	W'21,29	W'45,6					W'45,11
W'54,28	W'21,30	W'21,31	W'22,0	W'22,3	W'45,12			W'45,17
W'55,2	W'22,4	W'22,5	W'22,6	W'22,7	W'22,8	W'22,9	W'45,18					W'45,23
W'55,8	W'22,10	W'22,11	W'22,12	W'22,13	W'22,14	W'22,15	W'45,24					W'45,29
W'55,14	W'22,16	W'22,21	W'45,30	W'45,31	W'46,0			W'46,3
W'55,20	W'22,22	W'22,27	W'46,4	W'46,5	W'46,6	W'46,7	W'46,8	W'46,9
W'55,26	W'22,28	W'22,29	W'22,30	W'22,31	W'23,0	W'23,1	W'46,10	W'46,11	W'46,12	W'46,13	W'46,14	W'46,15
W'56,0	W'56,1	W'23,2	W'23,7	W'46,16				W'46,21
x	x	x	x	x	x	x	x	x	x	x	x	x

FIG. 5b

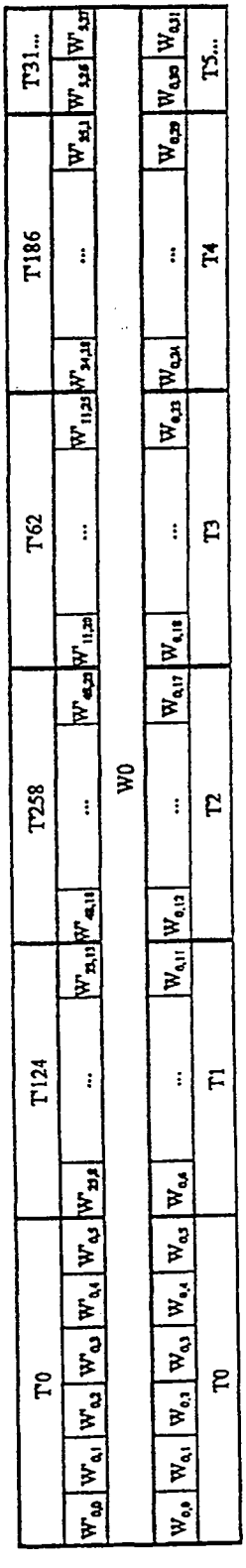


FIG. 6

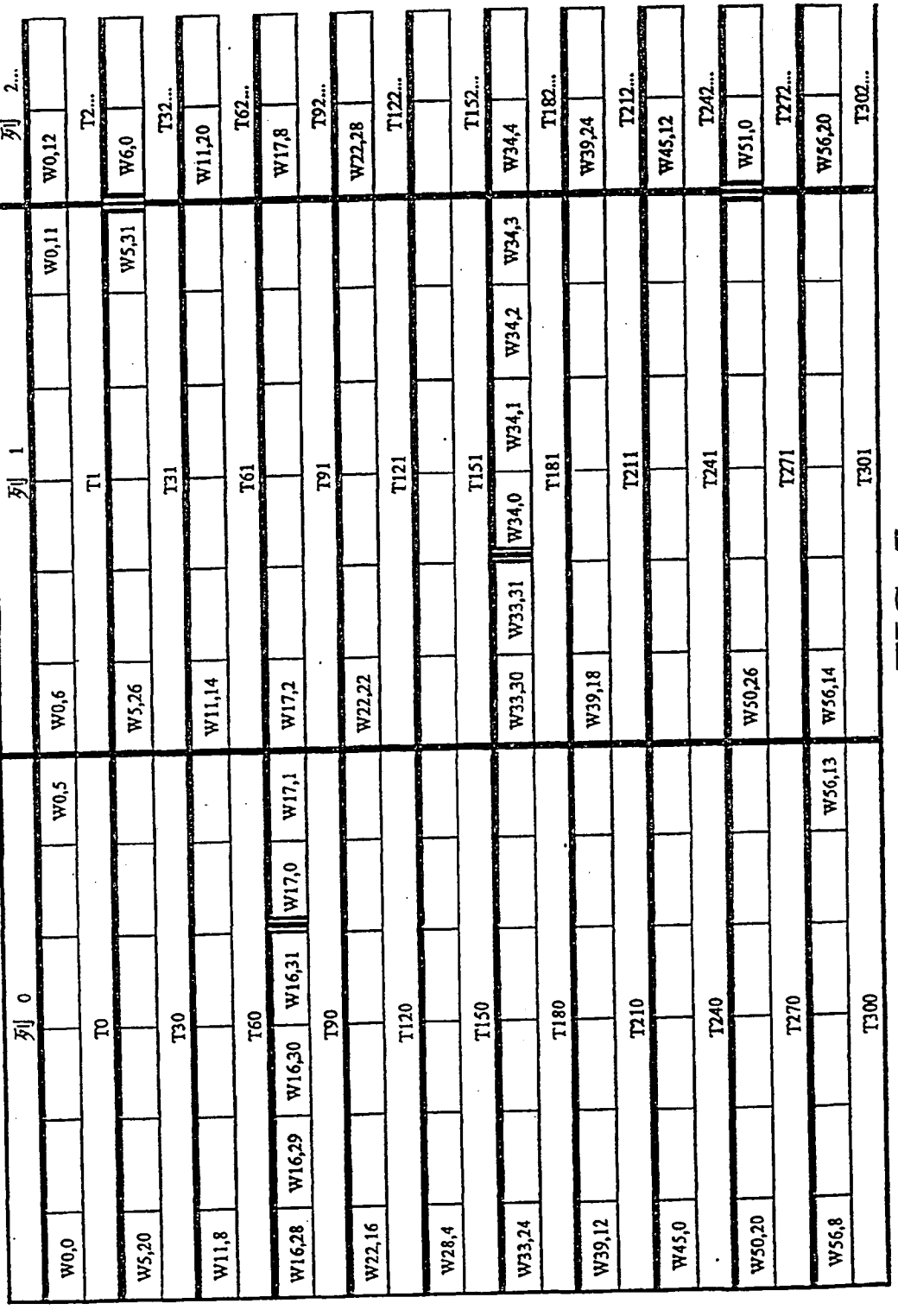

列 0
列 1
列 2...
W0,0
W0,5
W0,6
W0,11
W0,12
T0
T1
T2...
W5,20
W5,26
W5,31
W6,0
T30
T31
T32...
W11,8
W11,14
W11,20
T60
T61
T62...
W16,28
W16,29
W16,30
W16,31
W17,0
W17,1
W17,2
W17,8
T90
T91
T92...
W22,16
W22,22
W22,28
T120
T121
T122...
W28,4
T150
T151
T152...
W33,24
W33,30
W33,31
W34,0
W34,1
W34,2
W34,3
W34,4
T180
T181
T182...
W39,12
W39,18
W39,24
T210
T211
T212...
W45,0
W45,12
T240
T241
T242...
W50,20
W50,26
W51,0
T270
T271
T272...
W56,8
W56,13
W56,14
W56,20
T300
T301
T302...

FIG. 7a

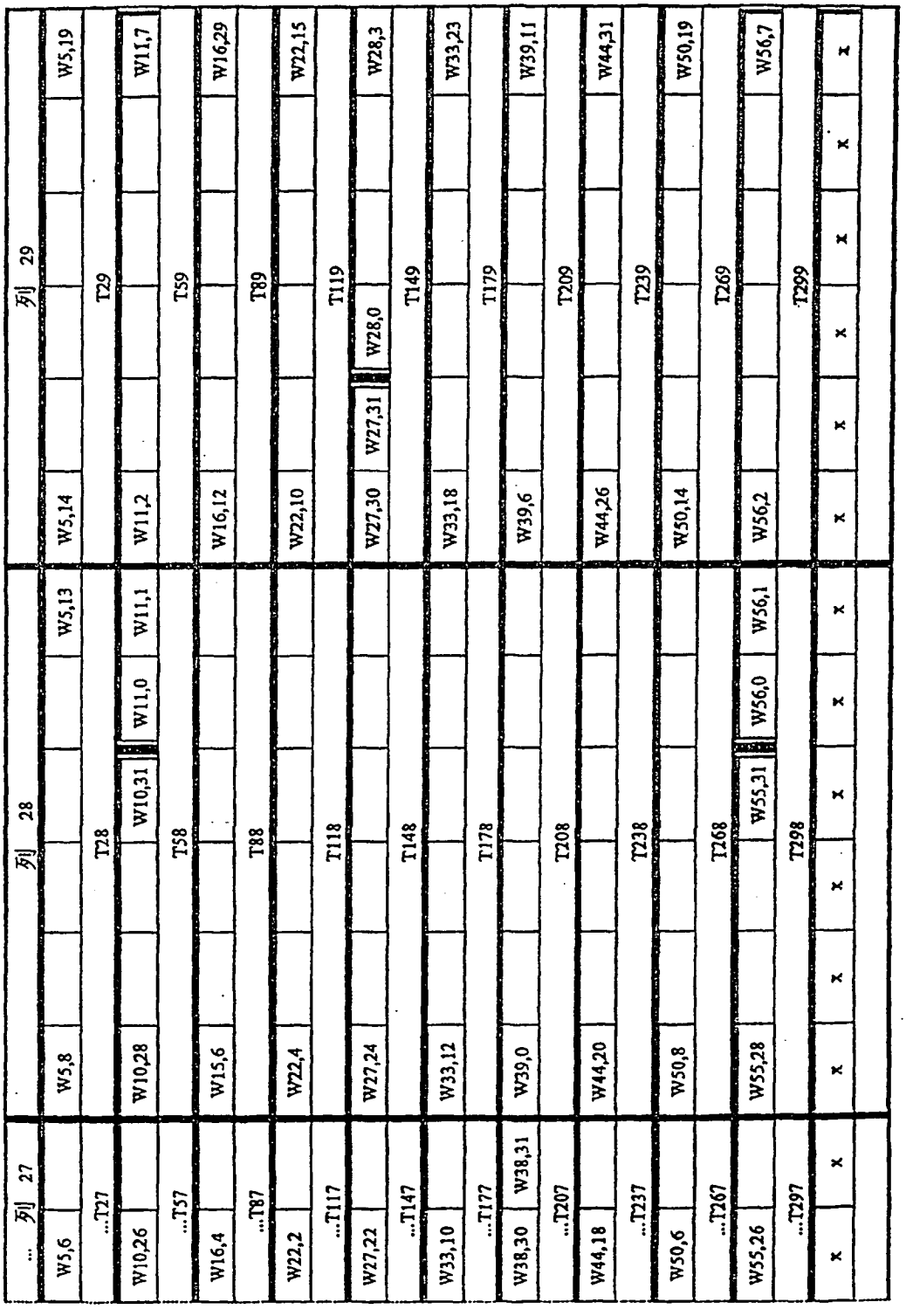

... 列 27
W5,6
...T27
W10,26
...T57
W16,4
...T87
W22,2
...T117
W27,22
...T147
W33,10
...T177
W38,30
W38,31
...T207
W44,18
...T237
W50,6
...T267
W55,26
...T297
x
x
列 28
W5,8
T28
W10,28
W10,31
T58
W15,6
T88
W22,4
T118
W27,24
T148
W33,12
T178
W39,0
T208
W44,20
T238
W50,8
T268
W55,28
T298
x
x
x
x
W5,13
W11,0
W11,1
T29
T59
T89
T119
T149
T179
T209
T239
T269
T299
x
x
x
x
列 29
W5,14
W5,19
W11,2
W11,7
W16,12
W16,29
W22,10
W22,15
W27,30
W27,31
W28,0
W28,3
W33,18
W33,23
W39,6
W39,11
W44,26
W44,31
W50,14
W50,19
W56,2
W56,7
x
x
x
x
x
x

FIG. 7b

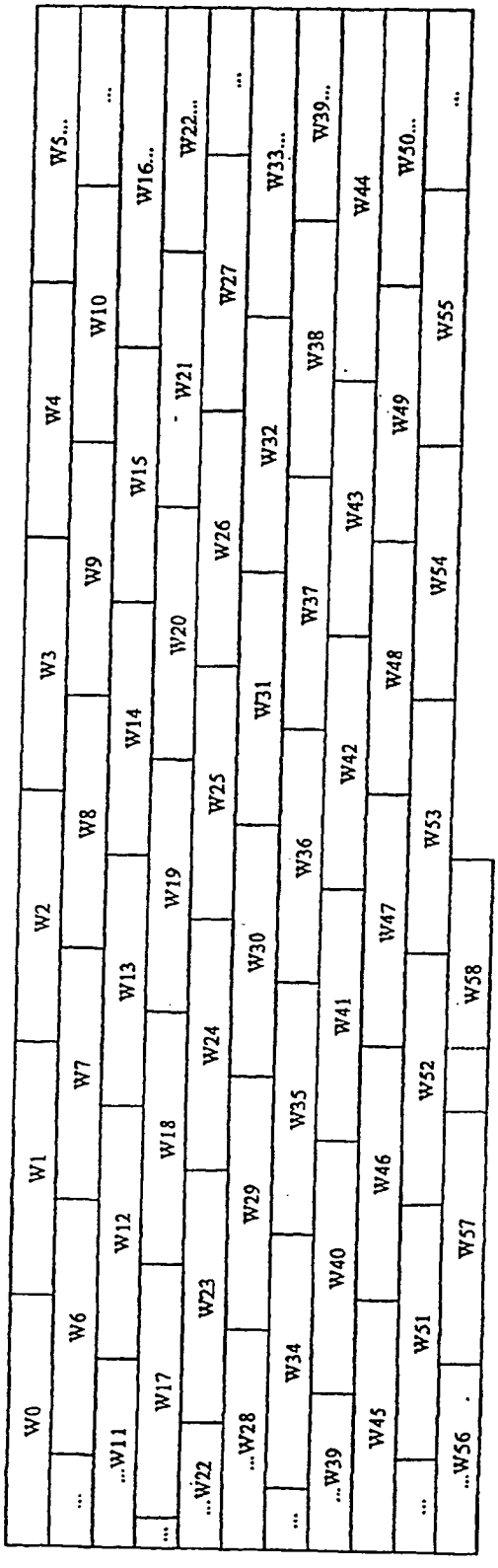

W0 W1 W2 W3 W4 W5...
... W6 W7 W8 W9 W10 ...
...W11 W12 W13 W14 W15 W16...
... W17 W18 W19 W20 W21 W22...
...W22 W23 W24 W25 W26 W27 ...
...W28 W29 W30 W31 W32 W33...
... W34 W35 W36 W37 W38 W39...
...W39 W40 W41 W42 W43 W44
W45 W46 W47 W48 W49 W50...
... W51 W52 W53 W54 W55 ...
...W56 W57 W58

FIG. 7c

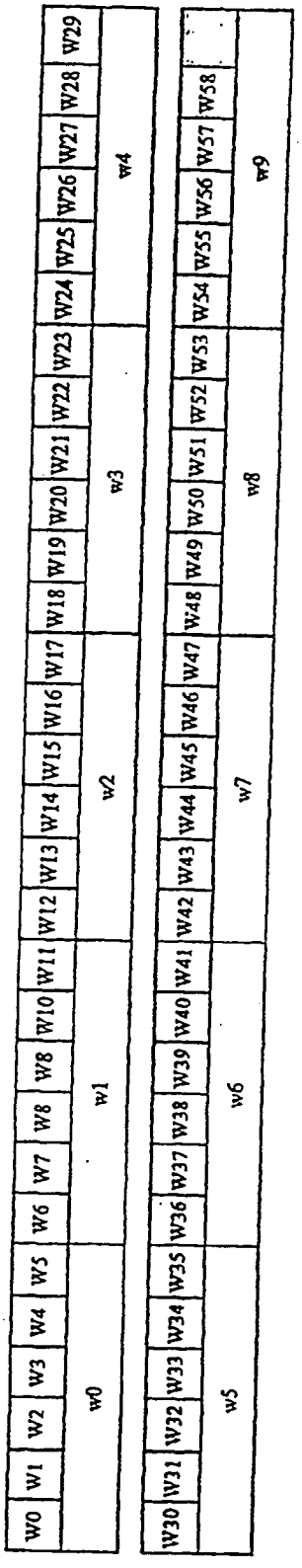
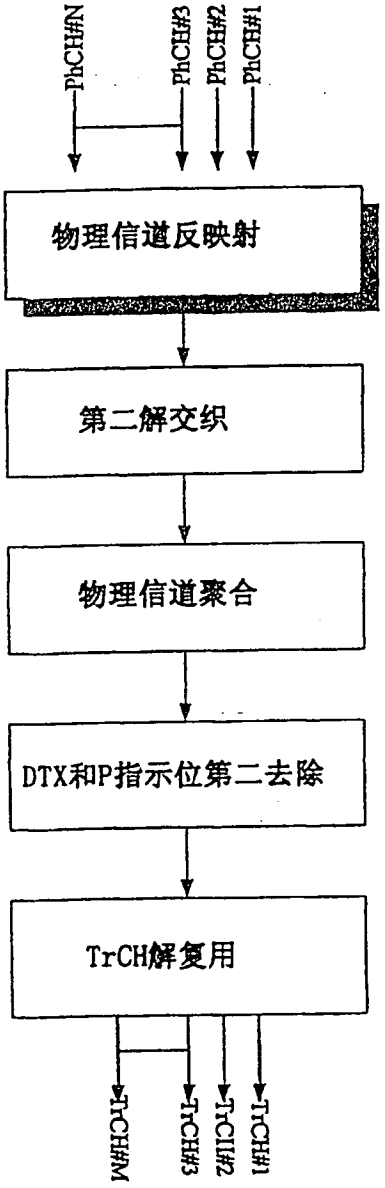

W0 W1 W2 W3 W4 W5 W6 W7 W8 W9 W10 W11 W12 W13 W14 W15 W16 W17 W18 W19 W20 W21 W22 W23 W24 W25 W26 W27 W28 W29
w0 w1 w2 w3 w4
W30 W31 W32 W33 W34 W35 W36 W37 W38 W39 W40 W41 W42 W43 W44 W45 W46 W47 W48 W49 W50 W51 W52 W53 W54 W55 W56 W57 W58
w5 w6 w7 w8 w9

FIG. 8

FIG. 9


PhCH#N
PhCH#3
PhCH#2
PhCH#1
物理信道反映射
第二解交织
物理信道聚合
DTX和P指示位第二去除
TrCH解复用
TrCH#M
TrCH#3
TrCH#2
TrCH#1

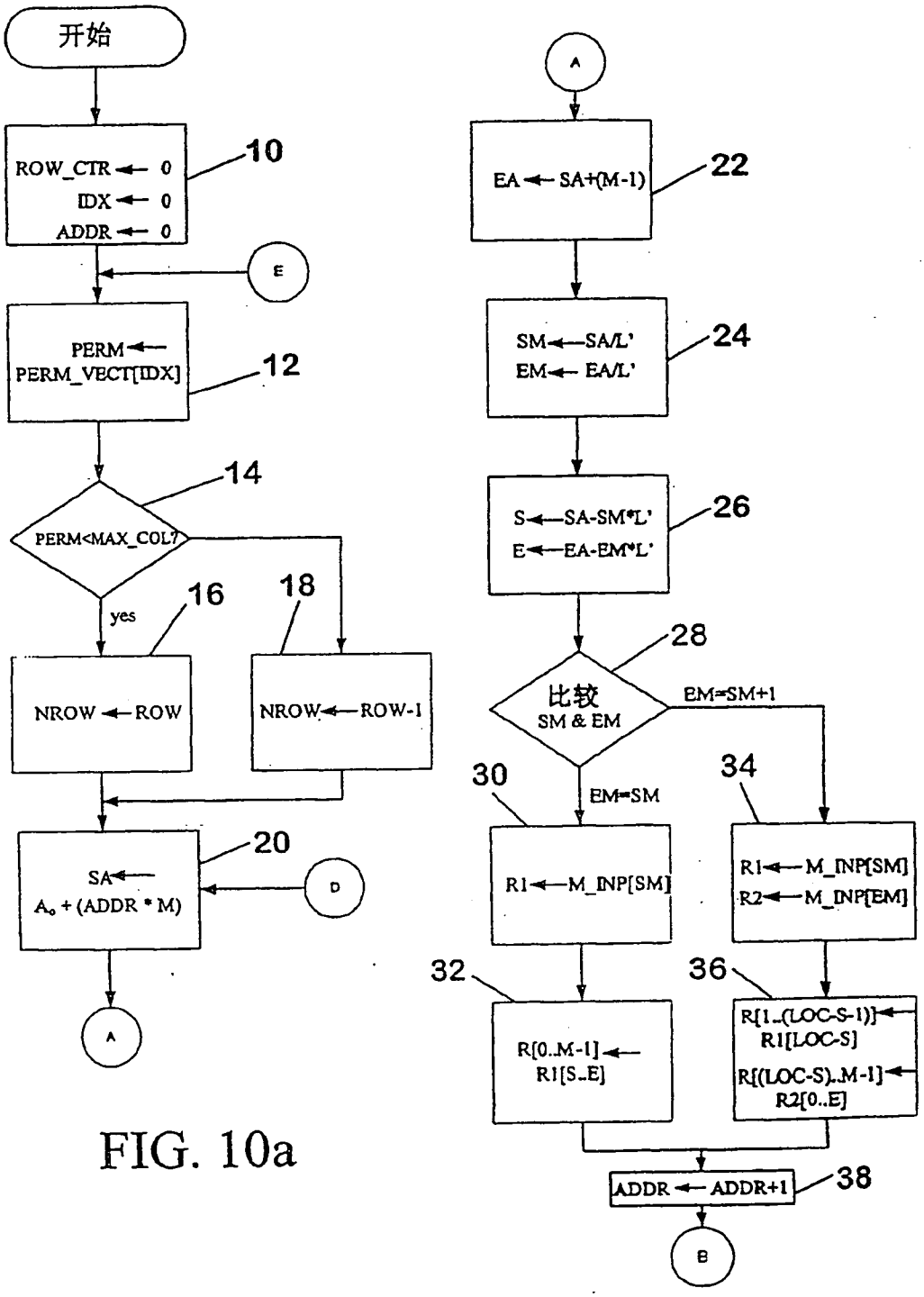

开始
ROW_CTR ← 0
IDX ← 0
ADDR ← 0
10
E
PERM ← PERM_VECT[IDX]
12
14
PERM<MAX_COL?
yes
16
18
NROW ← ROW
NROW ← ROW-1
SA ← A0 + (ADDR * M)
20
D
A
A
EA ← SA+(M-1)
22
SM ← SA/L'
EM ← EA/L'
24
S ← SA-SM*L'
E ← EA-EM*L'
26
28
比较
SM & EM
EM=SM+1
EM=SM
30
34
R1 ← M_INP[SM]
R1 ← M_INP[SM]
R2 ← M_INP[EM]
32
36
R[0..M-1] ← R1[S..E]
R[1..(LOC-S-1)] ← R1[LOC-S]
R[(LOC-S)..M-1] ← R2[0..E]
ADDR ← ADDR+1
38
B

FIG. 10a

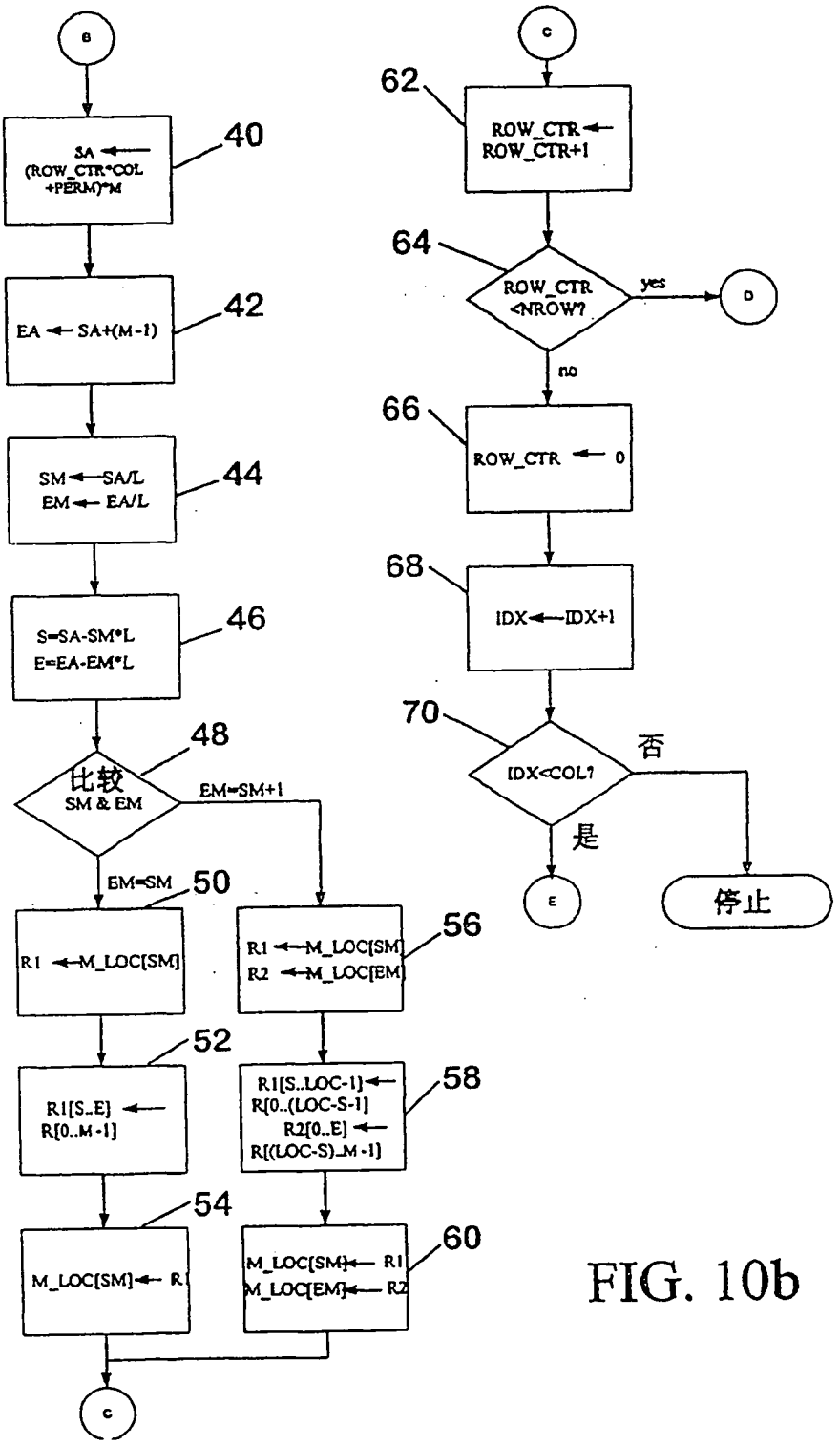

B
SA ← (ROW_CTR*COL +PERM)*M
40
EA ← SA+(M-1)
42
SM ← SA/L
EM ← EA/L
44
S=SA-SM*L
E=EA-EM*L
46
比较 SM & EM
48
EM=SM+1
EM=SM
50
R1 ← M_LOC[SM]
R1 ← M_LOC[SM]
R2 ← M_LOC[EM]
56
52
R1[S..E] ← R[0..M-1]
R1[S..LOC-1] ← R[0..(LOC-S-1]
R2[0..E] ← R[(LOC-S)..M-1]
58
54
M_LOC[SM] ← R1
M_LOC[SM] ← R1
M_LOC[EM] ← R2
60
C
C
62
ROW_CTR ← ROW_CTR+1
64
ROW_CTR <NROW?
yes
D
no
66
ROW_CTR ← 0
68
IDX ← IDX+1
70
IDX<COL?
否
是
E
停止

FIG. 10b

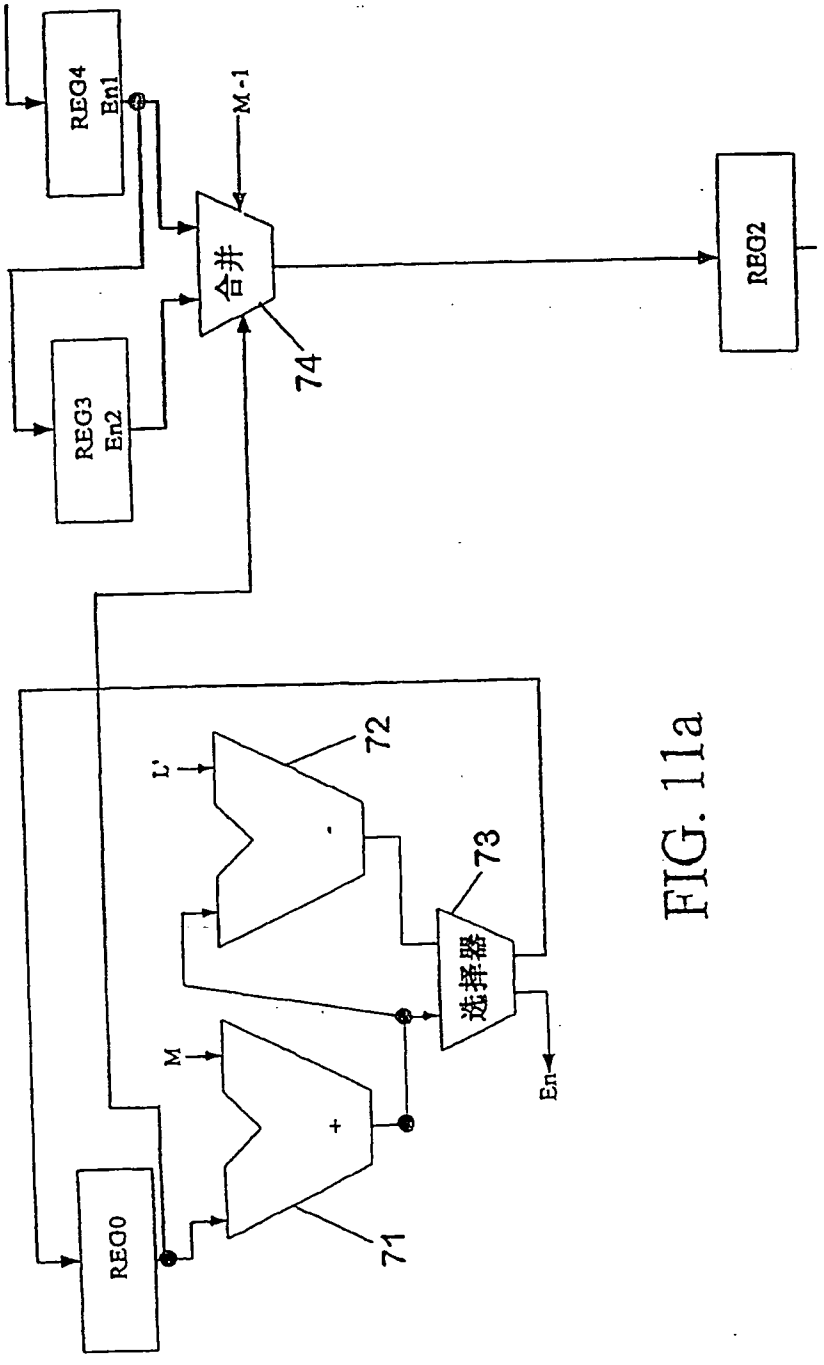


FIG. 11a

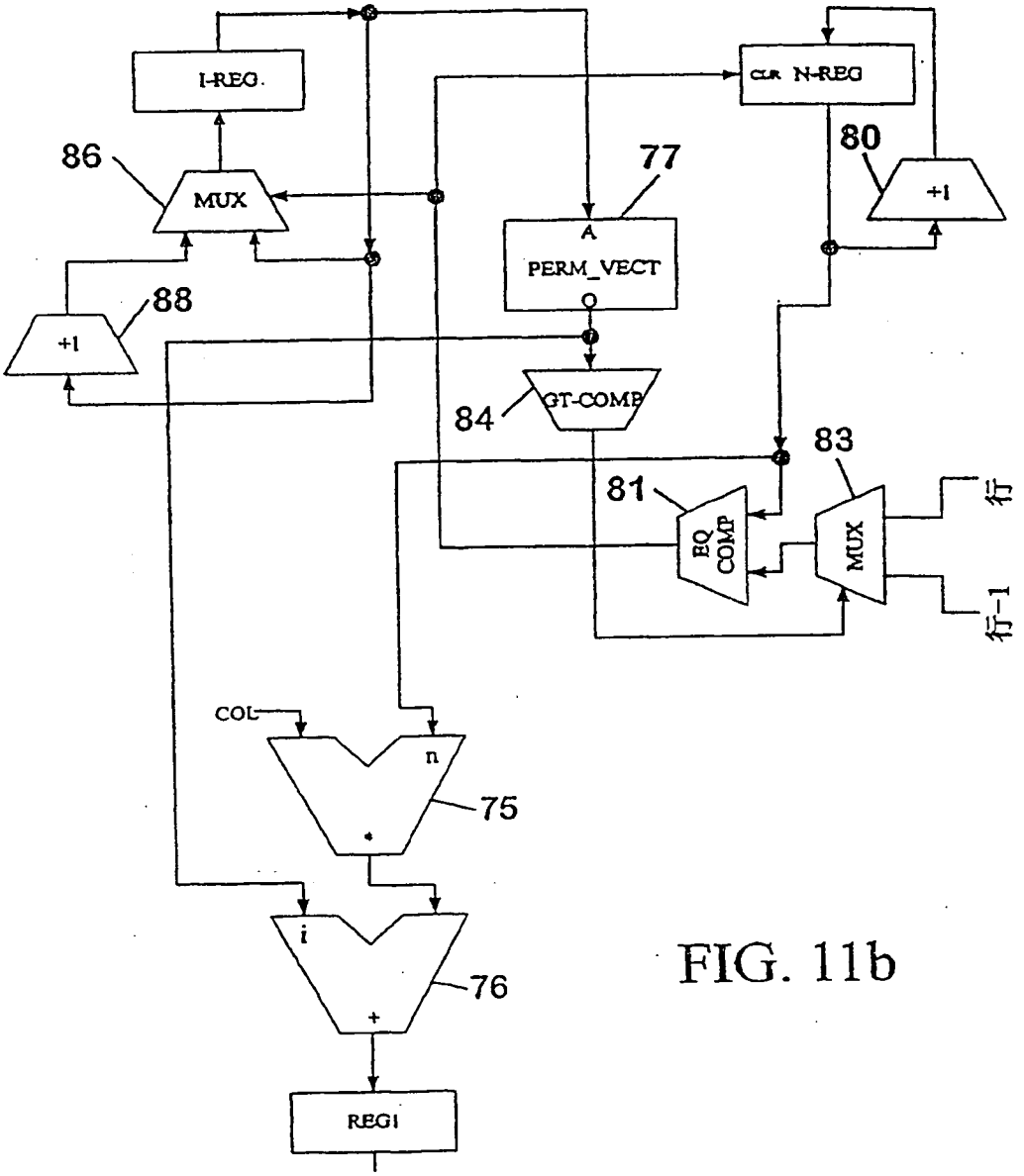

I-REG
CLR N-REG
86
MUX
77
A
PERM_VECT
O
80
+1
88
+1
84
GT-COMP
83
81
EQ COMP
MUX
行
行-1
COL
n
75
i
76
+
REG1

FIG. 11b

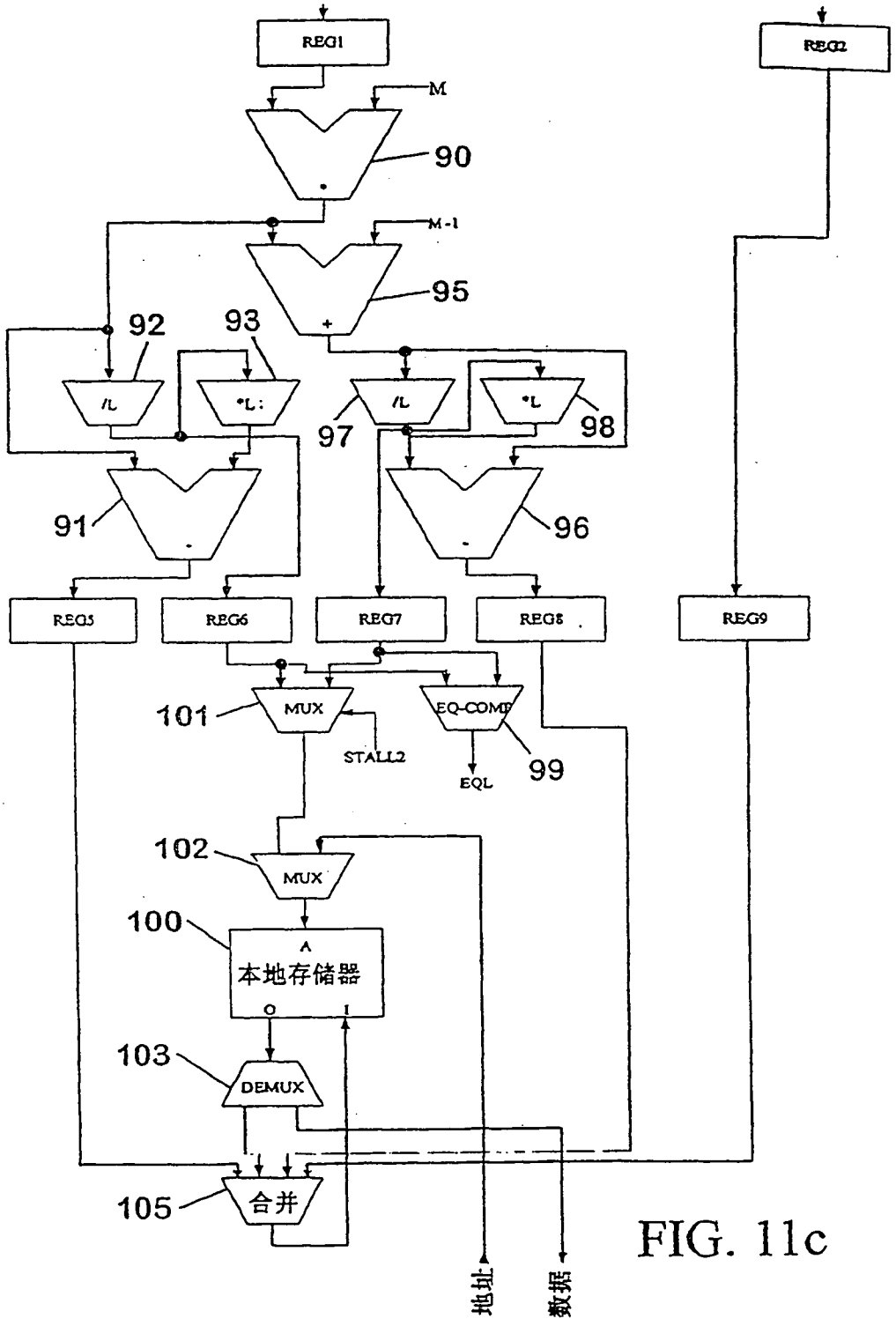

REG1
REG2
M
90
M-1
95
92
93
97
98
91
96
REG5
REG6
REG7
REG8
REG9
101
MUX
EQ-COMP
STALL2
EQL
99
102
MUX
100
A
本地存储器
O
I
103
DEMUX
105
合并
地址
数据

FIG. 11c

第三代频分双工调制解调交织器

摘要

发明一种方法及装置来对扩展交织数据块进行解交织处理，尤其适用于无线通信系统中，如由第三代合作伙伴项目(3G)标准采用的系统中。 数据基于顺序元素被处理，其中每个元素具有一个预先确定数目的比特位数M，比特位包含在连续数据字 W'的一个块中。 元素以顺序地方式从字组W'块中抽取出来，每个被抽取的元素形成一个或者两个在字组 W'中连续交织的字。 元素被保存在解交织器存储器中字组 W 中选定的位置，从而当完成抽取和写入所有元素时，从解交织器存储器中可以连续地读出字组W，其对应于初始数据位块（从其中产生了交织元素块）。 采用其他传统的处理对解交织扩展字进行收缩，以在接收器中重新产生出数据位块，该数据位块与在发送器中最初指定发送的数据相一致。

Third Generation FDD Modem Interleaver

ABSTRACT

A method and apparatus are disclosed for deinterleaving expanded interleaved data blocks, particularly for use in a wireless telecommunications systems such as provided by the Third Generation Partnership Project (3G) standard. The data is processed on a sequential element basis where each element has a pre-determined number of bits M which bits are contained in a block of sequential data world W '. The elements are extracted from the block of words W ' in sequential order, each element being extracted form either a single or two sequential interleaved words within the set of words W '. The elements are stored in selective location within a set of words W of a deinterleaver memory such that upon completion of the extraction and writing of all the elements, the words W from the deinterleaver memory can be sequentially read out to correspond to an original data block of bits from which the block of interleaved elements was created. Additional conventional processing results in the contraction of the deinterleaved expanded words to reproduce the data block of bits in a receiver as originally designated for transmission in a transmitter.