OPIC
Office de la propriété intellectuelle du Canada

CIPO
Canadian Intellectual Property Office

(12) (19) (CA) **Demande-Application**

(21) (A1) **2,341,014**

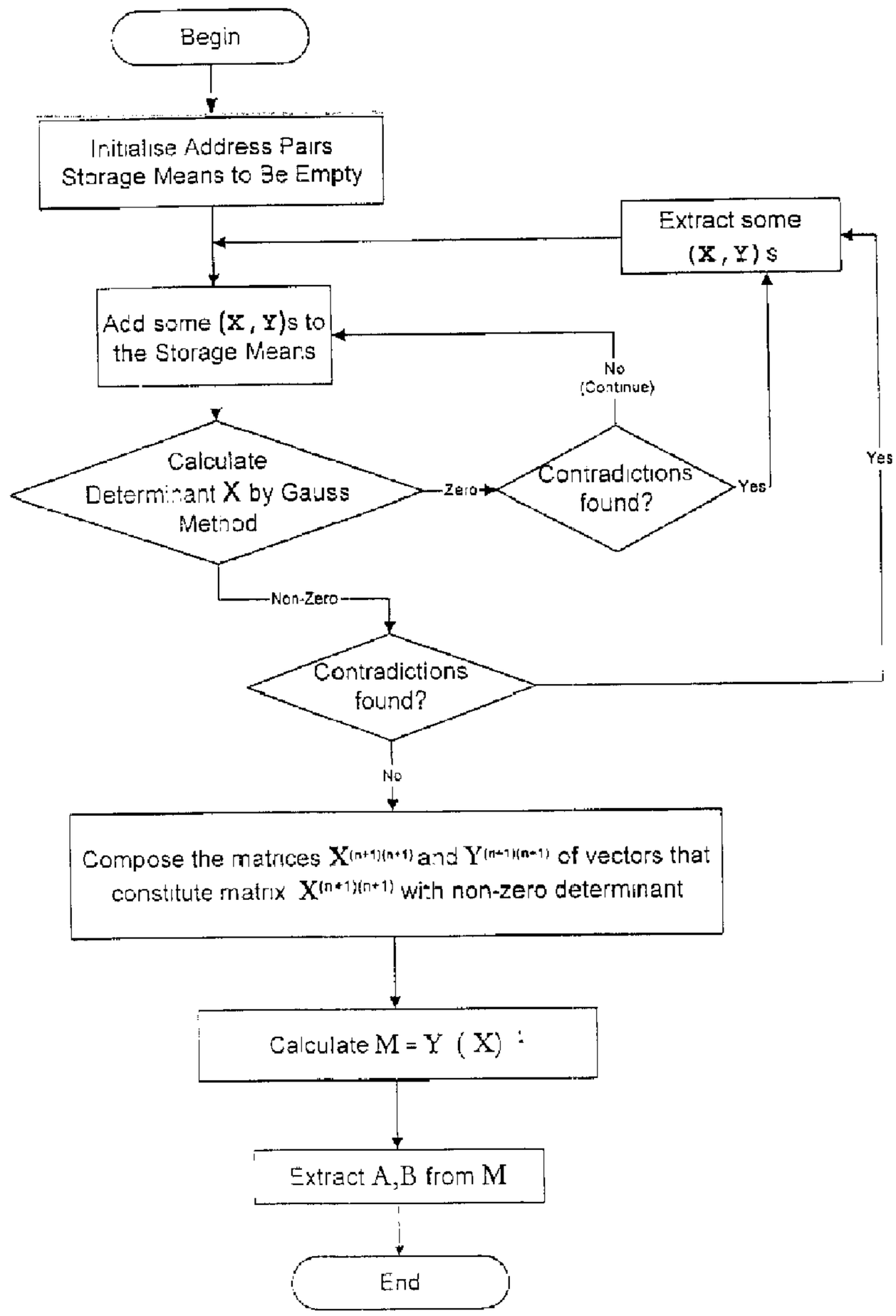
(86) 1998/08/19
(87) 2000/03/02

(72) VILKOV, BORIS NIKOLAEVICH, GB
(72) DEAS, ALEXANDER ROGER, GB
(71) VILKOV, BORIS NIKOLAEVICH, GB

(51) Int.Cl.^{7} G06F 11/20

(54) **SYSTEME ET PROCEDE D'IDENTIFICATION DE TRANSFORMATION D'ADRESSES DE DISPOSITIFS A MEMOIRE**

(54) **A SYSTEM AND METHOD FOR IDENTIFICATION OF TRANSFORMATION OF MEMORY DEVICE ADDRESSES**



OPIC
Office de la propriété intellectuelle du Canada

CIPO
Canadian Intellectual Property Office

(21) (A1) 2,341,014

(86) 1998/08/19
(87) 2000/03/02

(57) A transformation identification system for identification of transformation of cell addresses between different memory device topologies providing the use of minimum memory space and time required for storage and computing defect data and also the flexibility of approach offering a user friendly interface and simplification of the transformation procedure.

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 11/20	A1	(11) International Publication Number: WO 00/11554
		(43) International Publication Date: 2 March 2000 (02.03.00)

(21) International Application Number: PCT/RU98/00275

(22) International Filing Date: 19 August 1998 (19.08.98)

(71)(72) Applicant and Inventor: VILKOV, Boris Nikolaevich [RU/RU]; ul. Aprelskaya, 3–29, St.Petersburg, 195176 (RU).

(72) Inventor: DEAS, Alexander, Roger; 8 Eskview Grove, Dalkeith, Edinburgh EH22 1JW (GB).

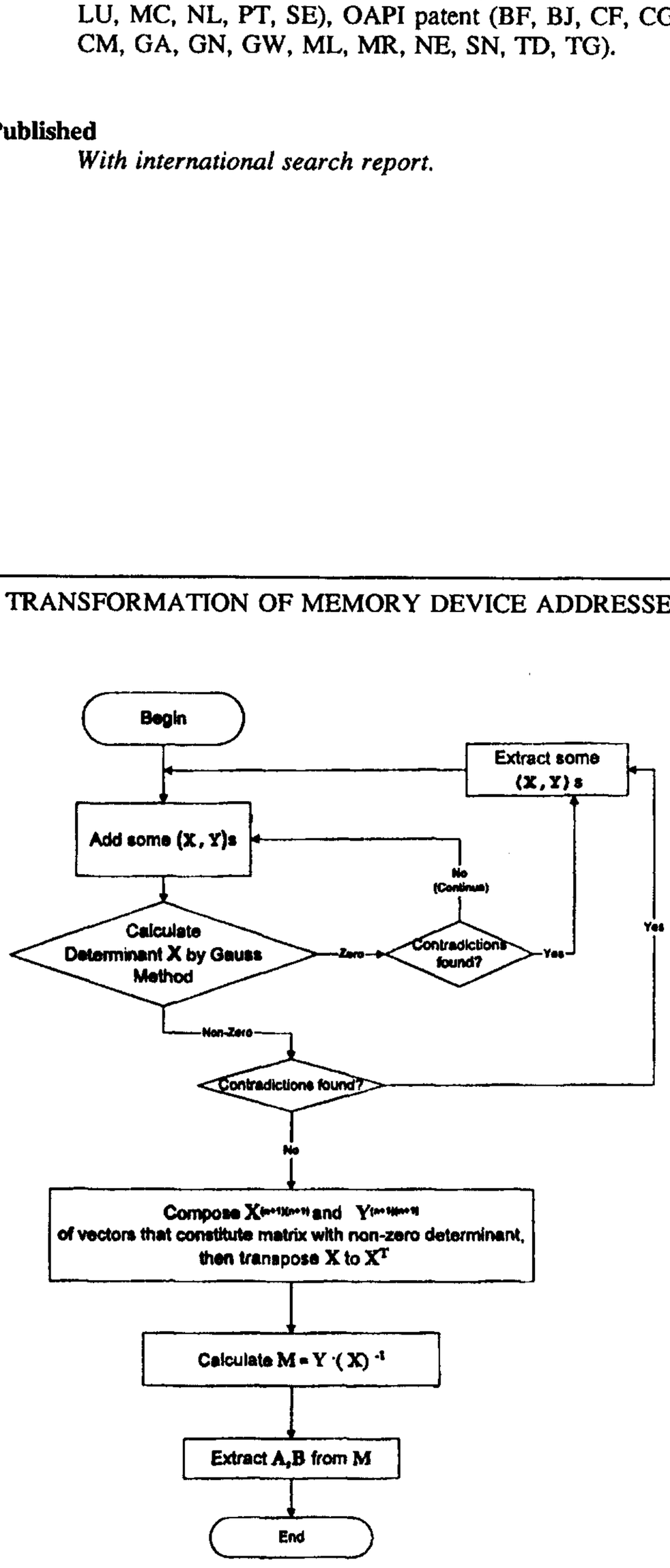
(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, GM, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SZ, 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), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
With international search report.

(54) Title: A SYSTEM AND METHOD FOR IDENTIFICATION OF TRANSFORMATION OF MEMORY DEVICE ADDRESSES

(57) Abstract

A transformation identification system for identification of transformation of cell addresses between different memory device topologies providing the use of minimum memory space and time required for storage and computing defect data and also the flexibility of approach offering a user friendly interface and simplification of the transformation procedure.



A SYSTEM AND A METHOD FOR DEFINING TRANSFORMS OF MEMORY DEVICE ADDRESSES

Technical Field

The present invention relates to a memory unit address transform definition system and a method of defining transforms and may be used, for example, for identification of address transformation from a logical address space into a topological address space in solid state memory devices, including semiconductor, ferro-electric, optical, holographic, molecular and crystalline atomic memories.

The present invention is applicable in particular, though not exclusively, in test systems for engineering test analysis, for example, for processing and representation of defect data, or in memory redundancy allocation systems for establishing a relationship between memory unit addresses in different memory device topologies for the purposes of distribution of spare resources.

Background of the invention

In the memory industry, large electronic systems are produced having hundreds of integrated circuits called devices designed to implement a large number of logical functions. These functions are implemented by the logical design of the system. However, the actual physical structure of the system which specifies the actual physical locations of the electronic components necessary to implement the logical, i.e. electrical, functions, differs from the logical design.

At present, the size and density of memory products is increasing exponentially over time: from 2^{10} bits in 1971 to more than 2^{28} bits being sampled by manufacturers today. As the density of memory devices increases, the number of defects in them increases as well. To properly test a memory device, a detailed description of the internal topology and address mapping of the device is required in order to run

complex redundancy schemes and optimize testing procedures.

To test memory products after fabrication, different test methods are used, some of them being independent of the physical location of the memory cell, but most requiring knowledge of the placement of every cell. The address presented to the memory device is called the logical address; this may not be the same as the address used to access the physical memory cell or cells, which is called the topological address. (See A.J. van de Goor "Testing Semiconductor Memories: Theory and Practice", publ. by John Wiley & Sons, 1996, pp. 429-436). The translation of logical addresses into topological addresses is called address transformation, or mapping, or scrambling. When addresses are transformed, successive logical addresses may transform into non-successive topological addresses. One reason for this difference is that decoders are restricted in size in order to fit the topology of rows and columns of memory cells. A second reason is that, to maintain acceptable production yields, redundant cells are added during manufacture that can be used to replace faulty cells. Spare rows and columns cause a difference in the logical and topological address sequence. Lastly, different device designs result in device layouts in which on-device address pads do not correspond to the standard pin numbers.

There are several transformation procedures described in *An Interactive Descrambler Program for RAMs with Redundancy, Kirschner, N. In Proc. IEEE Int. Test Conference, pp. 252-257, 1982.* The known transformation means can scramble the address lines of a 64 Kbit memory device using an 8-bit row address and an 8-bit column address. The equations describing the transformation operation for the row-select lines r_0 through r_7 are given below. To identify the mapping, each address must be calculated in accordance with the equations; there is no

simple scheme provided for identifying the transformation.

$r_0 = a_0$ XOR a_2 XOR A_7

$r_1 = a_1$ XOR a_2 XOR A_7

$r_2 = a_2$ XOR A_7

.....

$r_6 = a_6$ XOR a_7

$r_7 = a_7$

Using formulas for describing address transformation requires a tremendous amount of calculation and, taking into account the trend of continuously increasing numbers of units to be addressed in a memory, known procedures are becoming too bulky to enable fast and intelligent mapping from logical into topological space and reverse mapping. Moreover, these procedures cannot provide identification of mapping schemes in cases where formulas are unknown or the memory architecture is too complex to make possible fast and effective calculations.

There are numerous mapping schemes described in the literature where transformation tables are used. For example, US 4,774,652 describes a memory mapping scheme designed to simplify the access of pages in a cache memory system. However, these systems often make mapping definition very difficult, for example requiring a lot of routine machine work to create a large table with 2^n entries. Storing the address transformation table requires too much space. Besides, reverse transformation requires the same memory space as direct transformation and is not possible where the available memory is restricted.

A computer design system for mapping a logical hierarchy into a physical hierarchy has been proposed in US 5,455,775. The logical hierarchy contains several levels of logical entities connected by signals. The mapping is accomplished by physically allocating each of the logical entities to a specific physical component and storing lists

of these logical entities and signal classifications. The known system simplifies mapping in that it permits the automatic generation of physical designs for an electronic layout and does not require the creation of large transformation tables. However, this technique cannot be used for reverse mapping, nor for defining the mapping where the transformation formulas are unknown.

Moreover, in some applications it is desirable to display an enlarged view of the die containing the semiconductor device and the locations that produced error data (see US 5,720,031). Typically, complex algorithms are required to perform logical-to-physical and then physical-to-spatial mapping in order to display error data properly.

A simple mapping means and procedure has been proposed in WO 99/37083 (priority UK 9725066.6). The known means are capable of a configurable mapping represented as an affine transformation in P^n space, where n is the total number of bits in an address, and P is the modulo 2 field. The transformation map is stored as an n x n matrix of bits and an n x 1 translation vector. The known means permits direct and reverse mapping and provides a fast and cost-effective mapping procedure. To perform address transformation, the known means also uses transformation formulas.

However, in some applications transformation formulas are unknown, for example, when they are not given explicitly by the memory manufacturer, that represents serious problem at testing or incoming inspection by the end user or buyer. Thus, the problem of defining transformation remains when the transformation formulas are unknown.

Summary of the Invention

It is an object of the present invention to overcome at least one of the above drawbacks of the prior art and provide a user-friendly means capable of fast and easy identification of address transformation of memory unit addresses between different memory device topologies when

transformation formulas are unknown and a method for defining transforms of memory device addresses, with the advantages of reducing the required memory space and the time required for transformation. According to one aspect of the invention, a transform definition system is provided for the identification of transformations of memory device addresses between different memory device topologies, each topology having a corresponding address space, the system comprising

a receiving means for receiving a representative plurality of pairs of addresses, each pair consisting of one memory cell address in the first address space and one address in the second address space,

an address pairs storage means for storing said pairs of addresses, and

a computing means for computing the transformation formulas.

Preferably, the system according to the present invention further comprises a means for collecting and storing information about transformation, i.e. the transformation map, as an n x n matrix of bits and an n x **1** translation vector, where n is the total number of bits in an addresses.

The transformation formulas obtained according to the present invention may be used in any affine transformation means for address transformation, preferably, in an affine transformation means capable of representing mapping as an affine transformation in $\mathbf{P}^{\mathbf{n}}$ space, where **n** denotes the total number of bits in an address, and **P** is the modulo 2 field, which is described in PCT/RU98/00403.

Still another aspect of the present invention is a method of defining transforms of memory device addresses between different memory device topologies, each topology having a corresponding address space, including

a step of receiving a representative plurality of pairs of addresses, each pair consisting of one memory cell address in the first address space and one address in the second address space,

a step of storing said pairs, and

a step of computing the transformation formulas.

Preferably, the obtained transformation formulas are used to initialise the affine transformation.

The proposed method of defining transforms may also be computer-implemented permitting fast and extremely easy address identification without the necessity of performing complex intellectual work and routine machine calculations.

According to the invention, a computer program product comprises:

a computer usable medium having computer readable program code means embodied in said medium for defining transforms of memory device addresses between different memory device topologies, each topology having a corresponding address space, said computer readable program code means comprising:

a computer readable program code means for causing a computer to receive a representative plurality of pairs of addresses, each pair consisting of one memory cell address in the first address space and one address in the second address space,

a computer readable program code means for causing a computer to store said pairs,

a computer readable program code means for causing a computer to generate transformation formulas.

Preferably, the proposed computer program product further comprises a computer readable program code means for causing a computer to transfer the transformation formulas to initialise an affine transformation means.

Still another aspect of the present invention is a

computer readable memory operable in a computer, comprising a computer program for performing the above method.

According to the invention, a system and a method for defining transforms may be used for different address transformations, for example, from logical into topological address space for engineering purposes, from logical into an address space appropriate for the purposes of allocating spare resources and laser repair procedures, or into an address space suitable for displaying errors stored in the form of a fault bit map, the transformation being a configurable mapping represented as affine transformation.

It shall also be mentioned that an important feature of the present invention is that it is applicable for the identification of transformations independently of its direction, thereby providing identification of both direct and reverse transforms within the same procedure.

The term "a memory cell" as used herein is an example of a memory addressable unit and shall not be interpreted as a limiting feature. In general, any addressable memory device falls within the scope of the present invention and may be treated in accordance with the proposed procedures, including a memory tile, memory cell, or any other addressable unit within the memory device.

The terms "a transform definition system", or "a system for defining transforms" are used herein to denote a system capable of defining, or identifying, transforms, i.e. transformations, of memory device addresses between different memory device topologies.

The proposed system may also comprise a means for collecting and storing information about transformations in the form of a transformation map. The transformation map may be stored as an n x n matrix of elements (e.g., bits) and an n x **1** translation vector, where n is the total number of bits in an address.

For a better understanding of the present invention and to show how the same may be carried into effect, reference will now be made, by way of example, without loss of generality, to the accompanying drawings in which:

Brief Description of Drawings

Fig. 1 is a block scheme of the means for defining transforms in accordance with the present invention.

Fig. 2 shows an example flow chart of the method for defining transforms in accordance with the present invention.

Fig. 3 illustrates an example procedure of defining transformation formulas.

Detailed Description of the Invention

As shown in Fig.1, the proposed transform definition system comprises a sample pairs generator 1 for generating pairs of addresses, each pair comprising one address from the logical, i.e. electrical, address space, including a row address, column address, and DQ address, and one address from, for example, the topological (physical) address space. The total size of a topological address is defined by the sum of a column address size, row address size and, possibly, DQ address size. The sample pairs generator 1 is operated by the user and may be implemented in a computer interface. An address pairs receiving means 2 receives information about the memory device expressed in the form of pairs of addresses which are stored in an address pairs storing means 3, wherefrom these pairs are sequentially fed into a computing means 4 for computing transformation formulas. The defined transformation formulas may be transferred then to initialise the affine transformation means 5 representing mapping between these two address spaces. Any suitable affine transformation means, for example the means described in WO 99/37083, publ.22.07.1999, may be used for representing mapping.

The flow chart of the method of defining transforms in accordance with the present invention is shown in Fig.2. The size of each address vector is distributed between vector components, i.e. column, row and DQ components. Where the DQ coordinate is absent from the address, the total address size will be constituted by a column address size and row address size only. The transformation identification means starts to operate upon receipt of the following information in bit form about the memory device: a) the address of the first memory unit in the first address space, comprising row address, R1, column address, C1, and, possibly, DQ address, and b) any address in the second address space, comprising row address, R2, column address, C2, and, possibly, DQ address. On the basis of this information, the transform definition means combines bits of row addresses into a bit vector X and bits of column addresses into a bit vector Y.

Transformation formulas in the case of affine address transformation may be represented in general by the linear function shown in Fig.3(a):

$$Y = AX + B,$$

where **X** is a vector combined of bits of row, column and, possibly, DQ addresses of a memory unit in the first address space,

Y is a vector combined of bits of row, column and, possibly, DQ addresses of a memory unit in the second address space,

A is a mapping matrix to be identified, the matrix containing **n** x **n** elements of P (where P is the modulo 2 field), and **B** is a translation vector, **n** x **1** bits, to be identified. Conventionally, mapping matrices and translation vectors are obtained by calculations from formulas supplied by the manufacturer. Thus, to calculate, e.g. the physical address of a cell, both the logical address and the mapping formulas are needed. However, if formulas are unknown, or a

device architecture is too complex, the prior art methods cannot be used and the problem remains unresolved.

An example of the transform definition procedure in accordance with the proposed invention will now be explained in detail and illustrated in Figs.2 and 3.

The first step of the proposed procedure is taking at random a pair of addresses, one address, X1, being from the first address space and the other address, Y1, being from the second address space, and placing the address X1 in the corresponding matrix X, and the address Y1 in the corresponding matrix Y. Then, the next pairs of addresses X2 and Y2, X3 and Y3, ... , Xn+1 and Yn+1, are taken sequentially and arranged in the corresponding matrixes X,Y. To simplify the calculations and increase the speed of processing, the above formulas are represented as shown in Fig.3(b), i.e. at the bottom of each X vector a "1" is added, whereas a matrix M is composed out of matrix A and translation vector B.

After each pair of X,Y has been added, the determinant of matrix X is calculated and the next pair of X,Y addresses is added until a non-zero determinant of matrix X is obtained.

If the determinant of matrix X is Zero, the transformation may be unidentifiable. In this case some pairs of X,Y are extracted and the determinant is calculated until non-zero determinant is achieved.

The next step is reordering columns of matrix X so that the upper lines of matrix X give a non-zero determinant. After the calculations are performed, the error is checked to make the procedure time- and memory-effective, as shown in Fig.3. To provide this, each pair of addresses X, Y is checked to ensure it is non-contradictory, i.e. if Y is O, then X should be zero too.

Once the non-zero determinant is achieved, the matrix A

and the transformation vector B may be easily extracted and the transformation completely defined as follows:

$$M = Y \cdot X^{-1}$$, the X being invertible.

Where matrix X is uininvertible, the procedure is continued until a non-zero determinant is achieved.

A computer program for implementing functions of the above system and/or performing the above method of transform definition may be created in any suitable computer language, e.g. C, C++, any Assembler, etc. in a manner evident for a person skilled in the art.

It will be appreciated that the above is an example embodiment only and that various modifications may be made to the embodiment described above within the scope of the present invention.

AMENDED CLAIMS:

1. A system for defining transforms of memory device addresses between different memory device topologies, each topology having a corresponding address space, the system comprising

a receiving means (2) for receiving a representative plurality of pairs of addresses, each pair consisting of one memory cell address Xi, where Xi is a vector combined of n bits of row, column and, possibly, DQ addresses of a memory unit in the first address space, and one address Yi, where Yi is a vector combined of n bits of row, column and, possibly, DQ addresses of a memory unit in the second address space,

an address pairs storing means (3) for storing said pairs of addresses in a matrix form, wherein each address Xi of the pair of addresses is placed in a matrix X, and each address Yi of the pair of addresses is placed in a matrix Y, so that all the addresses Xi, Yi are arranged in the corresponding matrixes X, Y in such a way that a bit containing a value 1 is added to the bottom of each Xi vector, thereby matrix X is $(n+1)*(n+1)$ matrix, whereas Y is $(n+1)*n$ matrix; and

a computing means (4) for computing transformation formulas by multiplying matrix Y by inverted matrix X and extracting from the product matrix M which is a $(n+1)*n$ matrix defined as $M = Y*(X)^{-1}$ a transformation matrix A as the left n vectors of matrix M, and a translation vector B as the $(n+1)^{th}$ vector of matrix M, the transformation matrix and translation vector being a matrix and a vector used to transform memory addresses from one address space into another.

2. A system according to claim 1, further comprising a computing means operable to calculate the determinant of

matrix X after each pair of Xi, Yi is added and reorder columns of matrix X so that the upper lines of matrix X give a non-zero determinant.

3. A system according to claim 1 or 2, wherein the address spaces are selected from the group including logical address space, topological address space, redundancy allocation address space, spatial address space or any other memory device address space.

4. A system according to any one of claims 1-3, further comprising a transformation map storing means for storing the transformation map as an n x n matrix of bits and a n x 1 translation vector, where n is the total number of bits in an address.

5. A system according to any one of claims 1-4, wherein the system is used for initialisation of an address transformation means (5).

6. A system according to any one of claims 1-5, wherein the address transformation means (5) are capable of representing mapping as an affine transformation in $\mathbf{P}^n$ space, where **n** denotes the total number of bits in an address, and **P** is the modulo 2 field.

7. A method of defining transforms of memory device addresses between different memory device topologies, each topology having a corresponding address space, the method including:

a step of receiving a representative plurality of pairs of addresses, each pair consisting of one memory cell address Xi, where Xi is a vector combined of n bits of row, column and, possibly, DQ addresses of a memory unit in the first address space and one memory cell address Yi, where Yi is a vector combined of n bits of row, column and, possibly, DQ addresses of the memory unit address in the second address space,

a step of storing said pairs in a matrix form, wherein each address Xi of the pair of addresses is placed in a matrix X, and each address Yi of the pair of addresses is placed in a matrix Y, so that all the addresses Xi, Yi are arranged in the corresponding matrixes X, Y in such a way that a bit containing a value 1 is added to the bottom of each Xi vector, thereby matrix X is (n+1)*(n+1) matrix, whereas Y is (n+1)*n matrix; and

a step of computing transformation formulas by multiplying matrix Y by inverted matrix X and extracting from the product matrix M which is a (n+1)*n matrix defined as $M = Y*(X)^{-1}$ the transformation matrix A as the left n vectors of matrix M and transformation vector B as the $(n+1)^{th}$ vector of matrix M.

8. A method according to claim 7, further comprising a step of calculating the determinant of matrix X after each pair of Xi, Yi is added.

9. A method according to claim 8, wherein if the determinant of matrix X is Zero, some pairs of X,Y are extracted and the determinant is calculated until non-zero determinant is achieved.

10. A method according to any one of claims 7-9, further comprising the step of reordering matrix X so that the upper lines of matrix X give a non-zero determinant.

11. A method according to any one of claims 7-10, wherein the obtained transformation formulas are used to initialise an affine transformation means.

12. A method according to any one of claims 7-10, wherein the transformation formulas are transferred to an address transformation means capable of representing mapping as an affine transformation in $\mathbf{P}^{\mathbf{n}}$ space, where **n** denotes the total number of bits in an address, and **P** is the modulo 2 field.

13. A computer program product comprising:

a computer usable medium having computer readable program code means embodied in said medium for defining transforms of memory device addresses between different memory device topologies, each topology having a corresponding address space, said computer readable program code means comprising:

a computer readable program code means for causing a computer to receive a representative plurality of pairs of addresses, each pair consisting of one memory cell address Xi, where Xi is a vector combined of n bits of row, column and, possibly, DQ addresses of a memory unit in the first address space and one address Yi, where Yi is a vector combined of n bits of row, column and, possibly, DQ addresses of a memory unit in the second address space,

a computer readable program code means for causing a computer to store said pairs in a matrix form, wherein each address Xi of the pair of addresses is placed in a matrix X, and each address Yi of the pair of addresses is placed in a matrix Y, so that all the addresses Xi, Yi are arranged in the corresponding matrixes X, Y in such a way that a bit containing a value 1 is added to the bottom of each Xi vector, thereby matrix X is (n+1)*(n+1) matrix, whereas Y is (n+1)*n matrix;

a computer readable program code means for causing a computer to generate transformation formulas by multiplying matrix Y by inverted matrix X and extracting from the product matrix M which is a (n+1)*n matrix defined as $M = Y*(X)^{-1}$ the transformation matrix A as the left n vectors of matrix M and transformation vector B as the $(n+1)^{th}$ vector of matrix M.

14. The computer program product as claimed in claim 13, further comprising a computer readable program code means for causing a computer to transfer the transformation formulas to initialise an affine transformation means (5).

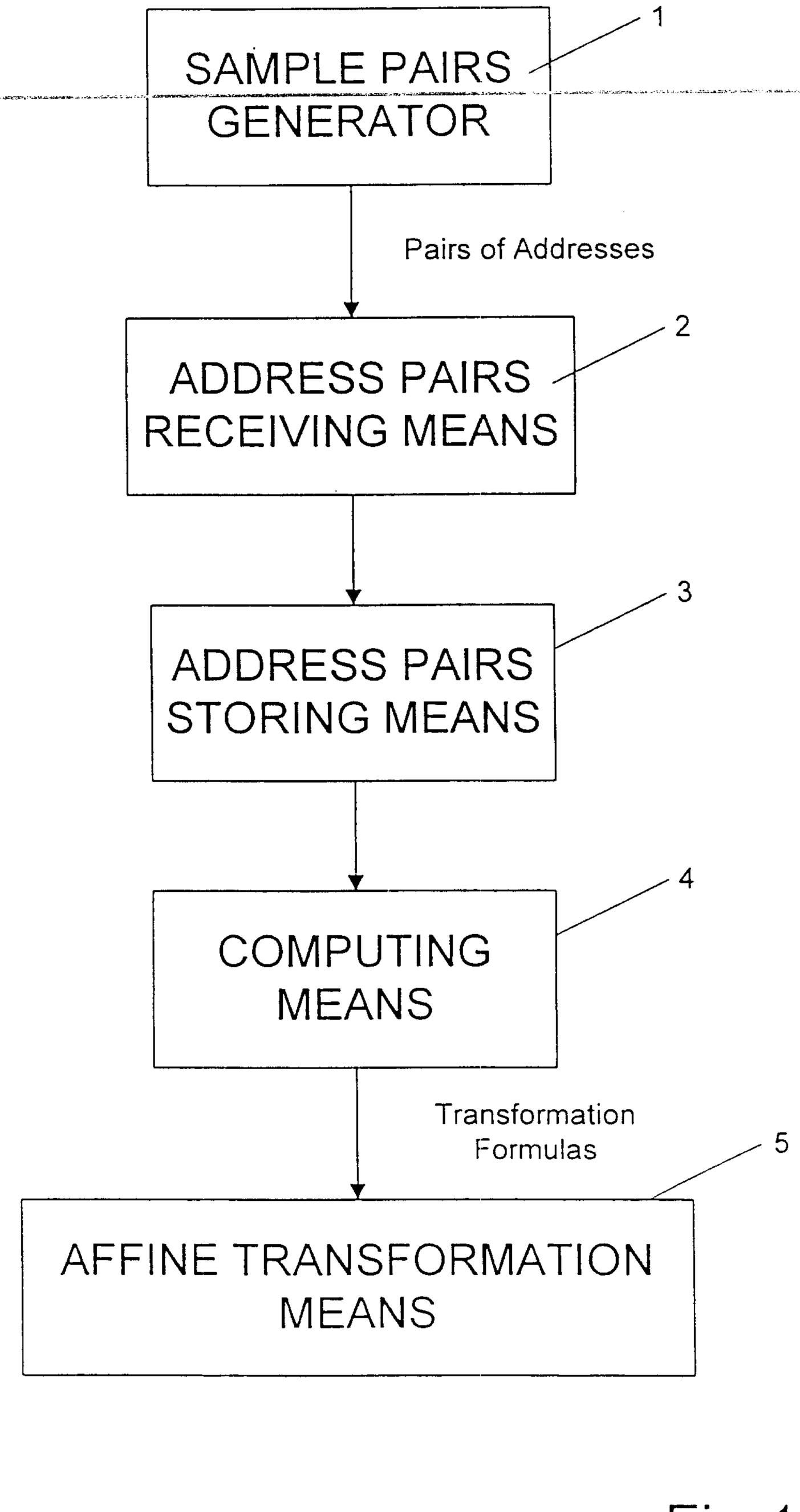


Fig.1

AMENDED SHEET

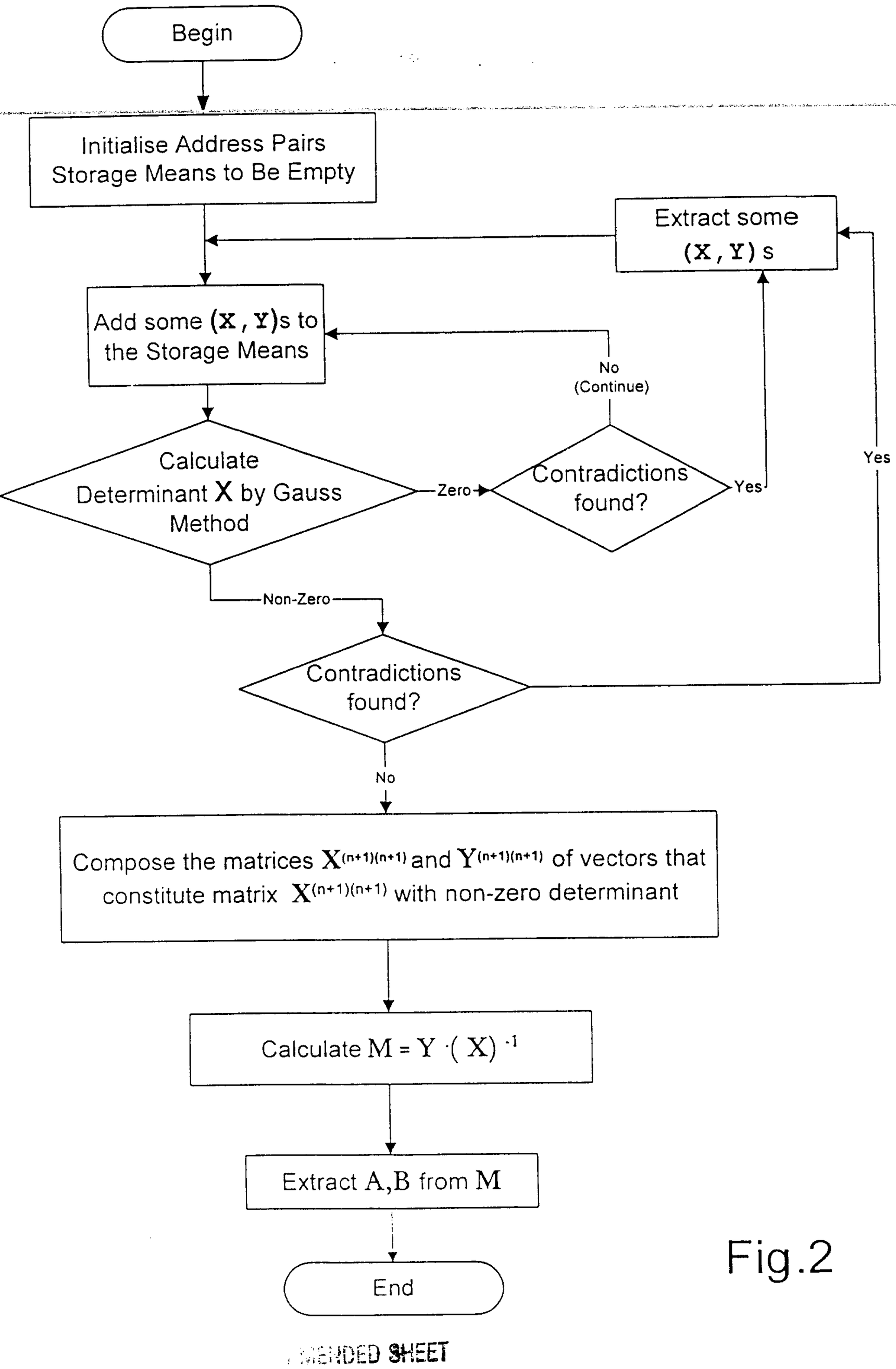
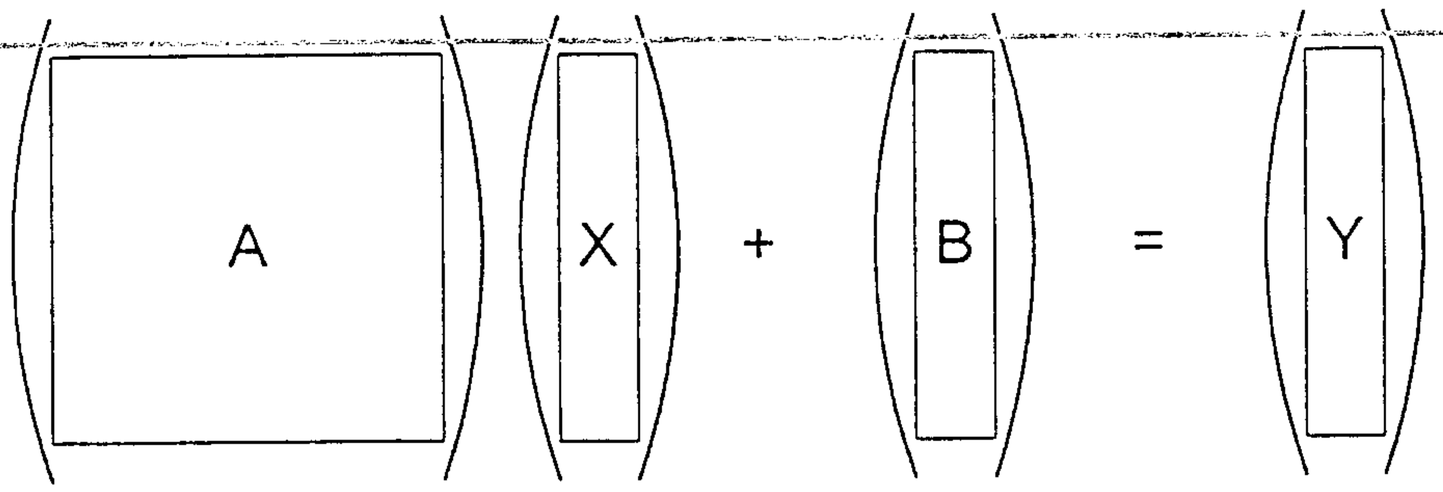
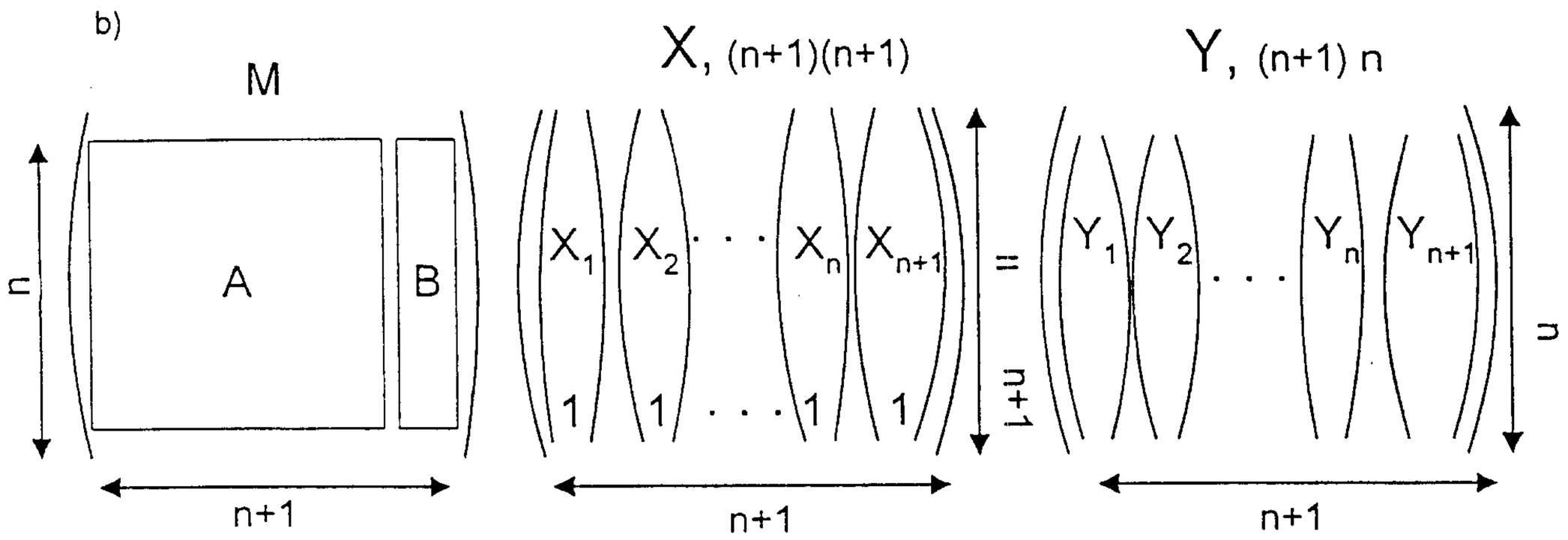


Fig.2

a)



b)



c)

When X invertible,
$M = Y \cdot (X)^{-1}$

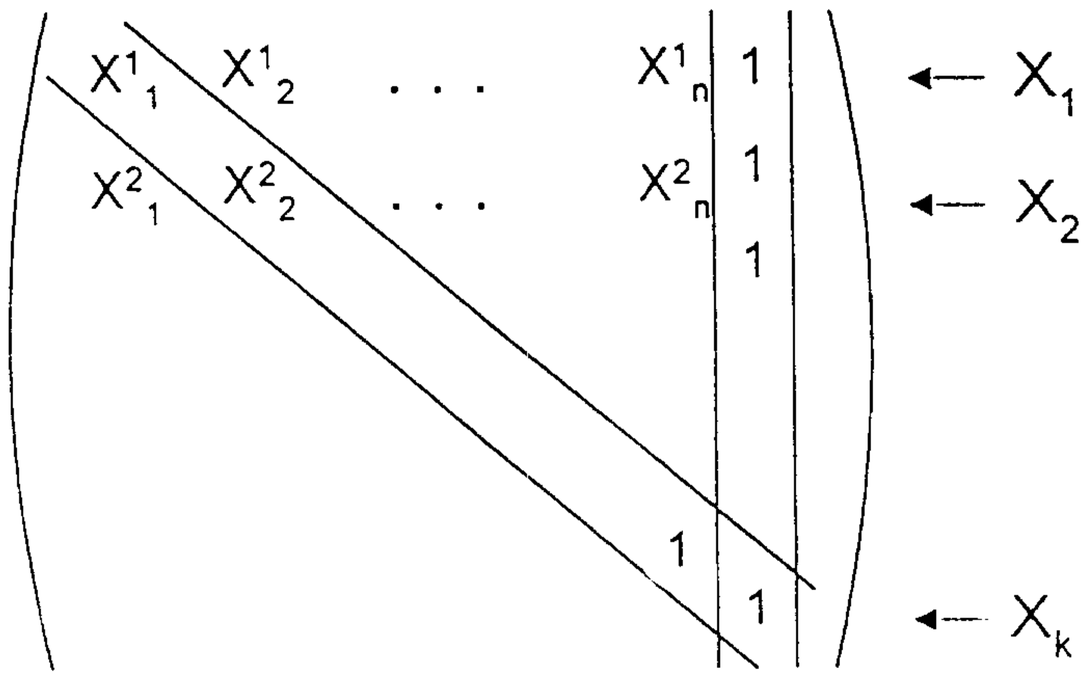
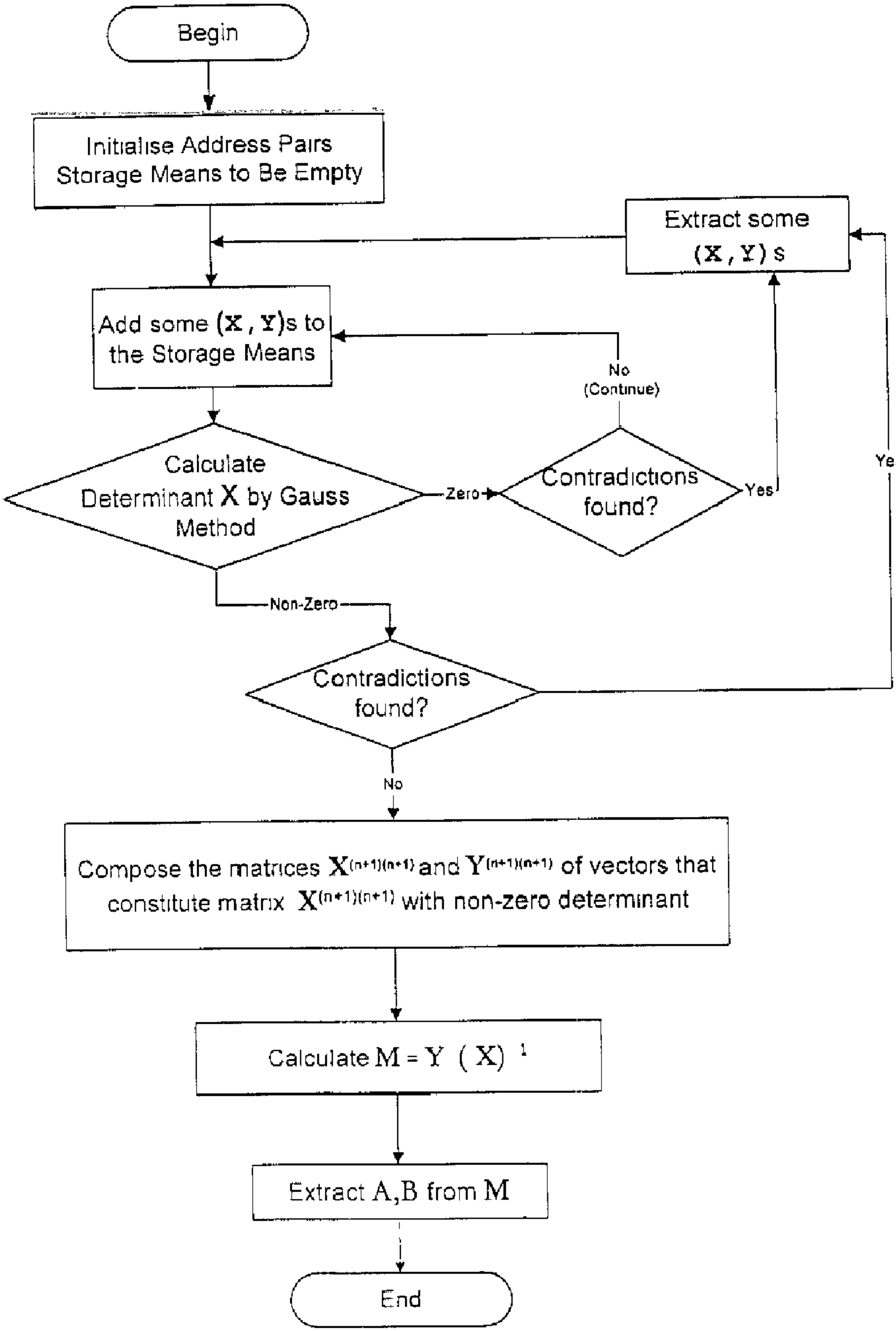


Fig.3


Begin
Initialise Address Pairs Storage Means to Be Empty
Extract some (X, Y) s
Add some (X, Y)s to the Storage Means
No (Continue)
Calculate Determinant X by Gauss Method
Zero
Contradictions found?
Yes
Yes
Non-Zero
Contradictions found?
Yes
No
Compose the matrices X(n+1)(n+1) and Y(n+1)(n+1) of vectors that constitute matrix X(n+1)(n+1) with non-zero determinant
Calculate M = Y (X) 1
Extract A,B from M
End