Office de la Propriété Intellectuelle du Canada
Un organisme d'Industrie Canada

Canadian Intellectual Property Office
An agency of Industry Canada

CA 2441728 A1 2002/10/10

(21) 2 441 728

(12) DEMANDE DE BREVET CANADIEN
CANADIAN PATENT APPLICATION

(13) A1

(86) Date de dépôt PCT/PCT Filing Date: 2001/12/18
(87) Date publication PCT/PCT Publication Date: 2002/10/10
(85) Entrée phase nationale/National Entry: 2003/09/23
(86) N° demande PCT/PCT Application No.: US 2001/049160
(87) N° publication PCT/PCT Publication No.: 2002/080044
(30) Priorité/Priority: 2001/03/29 (09/822,904) US

(51) Cl.Int.^{7}/Int.Cl.^{7} G06F 17/50

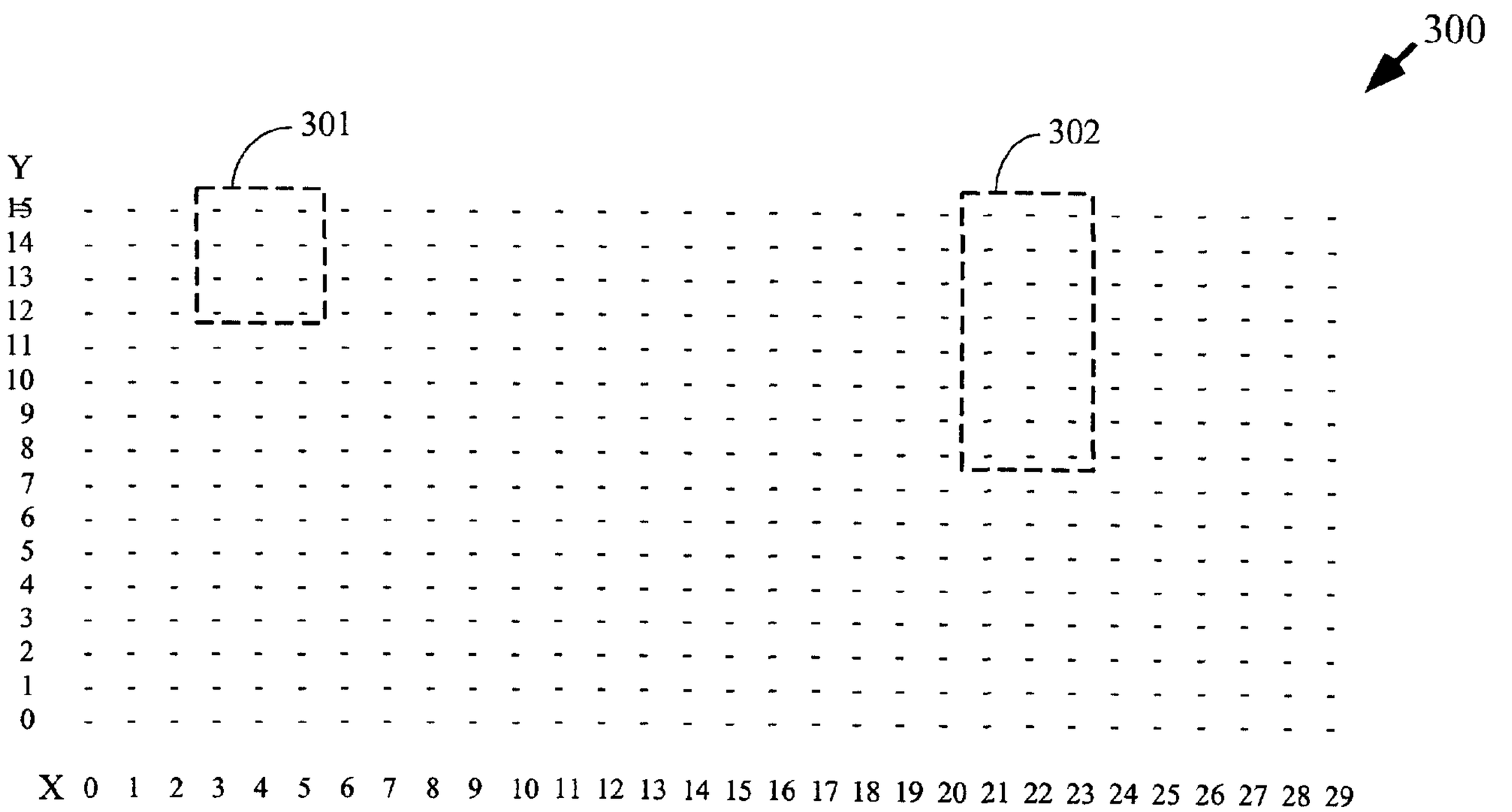
(71) Demandeur/Applicant:
XILINX, INC., US

(72) Inventeur/Inventor:
KRUSE, JAMES W., US

(74) Agent: SMART & BIGGAR

(54) Titre : PROCEDE DE LIMITATION D'IMPLANTATIONS NON UNIFORMES DE FPGA AU MOYEN D'UN SYSTEME DE COORDONNEES UNIFORMES

(54) Title: METHOD OF CONSTRAINING NON-UNIFORM FPGA LAYOUTS USING A UNIFORM COORDINATE SYSTEM



(57) Abrégé/Abstract:
A method of designating circuit element positions using uniform coordinate systems that can be applied to non-uniform logic arrays. A "site map" is constructed comprising a uniform array of "sites". A uniform coordinate system is applied to the site map. The various logic blocks, which may be of different types and sizes, are mapped to the site array. The result is the imposition of a uniform coordinate system on a non-uniform logic array, using the intervening abstraction of a site array. Because the site array is uniform, a relative location constraint applied to a site within the site array retains its validity regardless of the location of the site within the site array, even when the relative location constraints are normalized.

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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
10 October 2002 (10.10.2002)

PCT

(10) International Publication Number
WO 02/080044 A2

(51) International Patent Classification[7]: G06F 17/50

(21) International Application Number: PCT/US01/49160

(22) International Filing Date:
18 December 2001 (18.12.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/822,904 29 March 2001 (29.03.2001) US

(71) Applicant: **XILINX, INC.** [US/US]; 2100 Logic Drive, San Jose, CA 95124 (US).

(72) Inventor: **KRUSE, James, W.**; 4482 Clay Street, Boulder, CO 80301 (US).

(74) Agents: **CHANROO, Keith, A.** et al.; Xilinx, Inc., 2100 Logic Drive, San Jose, CA 95124 (US).

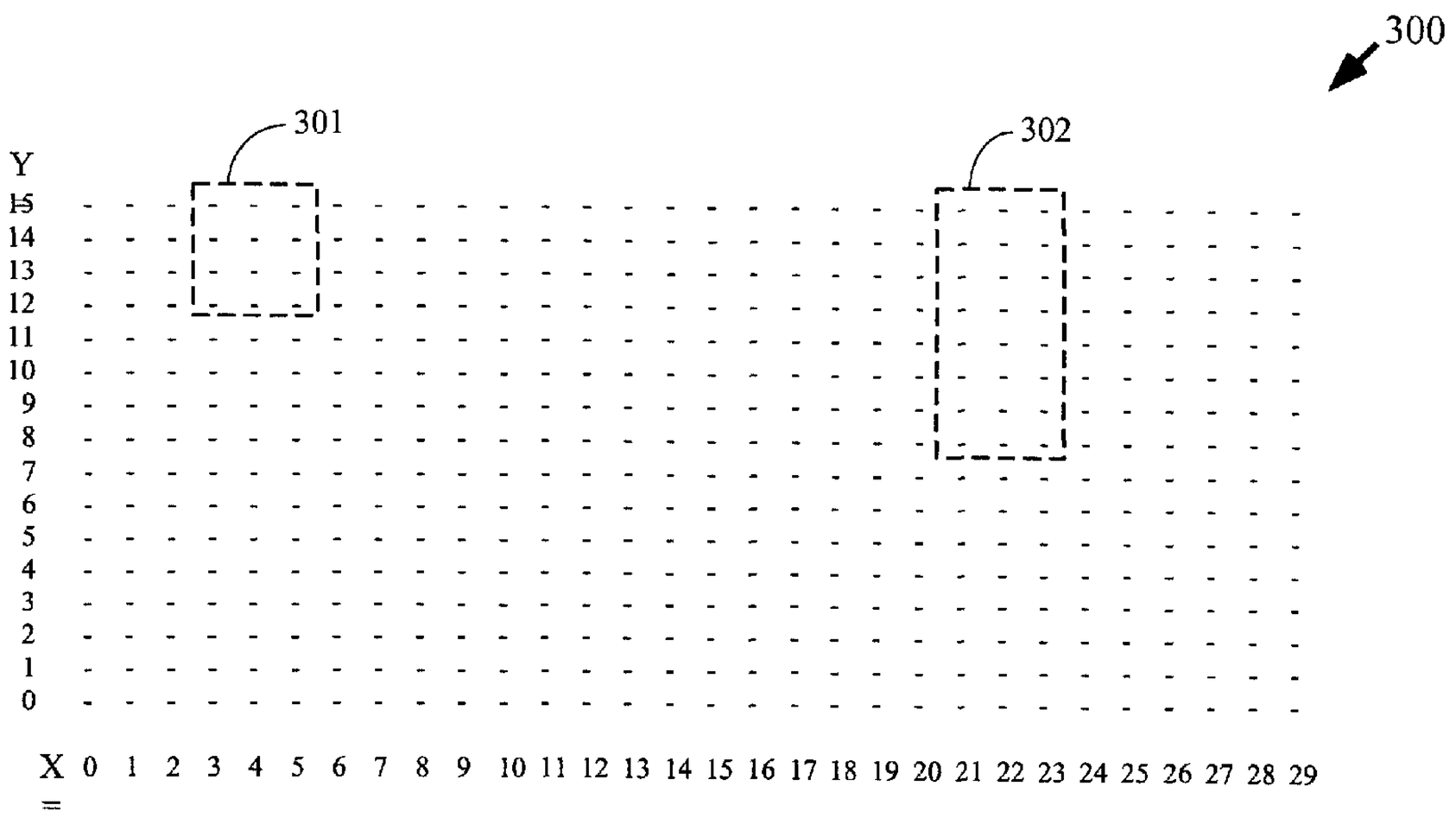
(81) Designated State *(national)*: CA.

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

Published:
— *without international search report and to be republished upon receipt of that report*

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.

(54) Title: METHOD OF CONSTRAINING NON-UNIFORM FPGA LAYOUTS USING A UNIFORM COORDINATE SYSTEM



(57) Abstract: A method of designating circuit element positions using uniform coordinate systems that can be applied to non-uniform logic arrays. A "site map" is constructed comprising a uniform array of "sites". A uniform coordinate system is applied to the site map. The various logic blocks, which may be of different types and sizes, are mapped to the site array. The result is the imposition of a uniform coordinate system on a non-uniform logic array, using the intervening abstraction of a site array. Because the site array is uniform, a relative location constraint applied to a site within the site array retains its validity regardless of the location of the site within the site array, even when the relative location constraints are normalized.

WO 02/080044 A2

METHOD OF CONSTRAINING NON-UNIFORM FPGA LAYOUTS USING A UNIFORM COORDINATE SYSTEM

FIELD OF THE INVENTION

The invention relates to Field Programmable Gate Arrays (FPGAs). More particularly, the invention relates to methods for constraining circuit element positions in structured FPGA layouts.

BACKGROUND OF THE INVENTION

Programmable ICs are a well-known type of digital integrated circuit that may be programmed by a user to perform specified logic functions. One type of programmable IC, the field programmable gate array (FPGA), typically includes an array of configurable logic blocks (CLBs) surrounded by a ring of programmable input/output blocks (IOBs). The CLBs and IOBs are interconnected by a programmable interconnect structure. The CLBs, IOBs, and interconnect structure are typically programmed by loading a stream of configuration data (bitstream) into internal configuration memory cells that define how the CLBs, IOBs, and interconnect structure are configured. The configuration data may be read from memory (e.g., an external PROM) or written into the FPGA by an external device. The collective states of the individual memory cells then determine the function of the FPGA.

One type of FPGA, the Xilinx XC4000™ Series FPGA, is described in detail in pages 4-5 through 4-69 of the Xilinx 1998 Data Book entitled "The Programmable Logic Data Book 1998", published in 1998 and available from Xilinx, Inc., 2100 Logic Drive, San Jose, California 95124. (Xilinx, Inc., owner of the copyright, has no objection to copying these and other pages referenced herein but otherwise reserves all copyright rights whatsoever.)

Fig. 1A shows a simplified diagram of an XC4000-Series FPGA 100. The FPGA includes a uniform array of CLBs surrounded by a ring of IOBs, as described above. (The

exemplary arrays herein are generally shown smaller than actual logic arrays in order to facilitate clear and simple drawings, but actual logic arrays typically include many more rows and columns of elements.) The CLB array typically includes variations in routing, for example at the outer edges and/or the center of the array, but the CLB array is considered uniform for purposes of this discussion because all of the logic blocks are similar. Each CLB includes two 4-input function generators, one 3-input function generator, and two flip-flops, in addition to other logic.

A more advanced FPGA is the Xilinx Virtex® FPGA, which in addition to the CLBs includes blocks of Random Access Memory (RAM). The Xilinx Virtex-II FPGA is described in detail in pages 3-75 through 3-96 of the Xilinx 2000 Data Book entitled "The Programmable Logic Data Book 2000", published April, 2000, available from Xilinx, Inc., 2100 Logic Drive, San Jose, California 95124.

Fig. 1B shows a simplified diagram of a Virtex FPGA 110. The FPGA includes a uniform array of CLBs, flanked by two columns of RAM blocks, with a ring of IOBs around the CLBs and RAM blocks. The Virtex CLB is organized into two functionally similar blocks called "slices", each of which includes two 4-input function generators and two flip-flops, in addition to other logic. The RAM blocks, of course, include different elements and are larger than the CLBs.

An even more advanced FPGA is the Xilinx Virtex®-II FPGA, which in addition to the CLBs includes blocks of Random Access Memory (RAM) and blocks implementing multiplier functions. (The Xilinx Virtex-II FPGA is described in detail in pages 33-75 of the "Virtex-II Platform FPGA Handbook", published January, 2001, available from Xilinx, Inc., 2100 Logic Drive, San Jose, California 95124.) The RAM and multiplier blocks are interspersed within the CLB array, forming a non-uniform array of logic blocks, as shown in Fig. 1C.

Fig. 1C shows a simplified diagram of a Virtex-II FPGA 120. The FPGA includes an array of CLBs, with columns of RAM blocks and multiplier blocks inserted within the CLB array. The resulting logic array is therefore non-uniform in nature, including three different kinds of logic blocks. The logic array is surrounded by a ring of IOBs, the IOB ring also including Digital Clock Manager (DCM) logic blocks. The Virtex-II FPGA also includes other types of logic blocks, but these are omitted from Fig. 1C for the sake of clarity. The Virtex-II CLB is organized into four slices, each of which includes two 4-input function generators and two flip-flops, in addition to other logic. Each CLB also includes two tristate buffers. The RAM blocks and multiplier blocks include different elements from the CLBs. The RAM and multiplier blocks are designed such that the height of each block is a multiple of the CLB height.

As FPGA designs increase in complexity, they reach a point at which the designer cannot deal with the entire design at the gate level. Where once a typical FPGA design comprised perhaps 5,000 gates, FPGA designs with over 100,000 gates are now common. To deal with this complexity, circuits are typically partitioned into smaller circuits that are more easily handled. Often, these smaller circuits are divided into yet smaller circuits, imposing on the design a multi-level hierarchy of logical blocks.

Libraries of predeveloped blocks of logic have been developed that can be included in an FPGA design. Such library modules include, for example, adders, multipliers, filters, and other arithmetic and DSP functions from which complex designs can be readily constructed. The use of predeveloped logic blocks permits faster design cycles, by eliminating the redesign of duplicated circuits. Further, such blocks are typically well tested, thereby making it easier to develop a reliable complex design.

To offer the best possible performance, some library modules have a fixed size and shape, with relative location

restrictions on each element. One type of module having a fixed size and shape is the Relationally Placed Macro (RPM) from Xilinx, Inc. RPMs are described in pages 4-96 and 4-97 of the "Libraries Guide" (hereinafter referred to as the "Xilinx Libraries Guide"), published October 1995 and available from Xilinx, Inc. An RPM is a schematic that includes constraints defining the order and structure of the underlying circuits. The location of each element within the RPM is defined relative to other elements in the RPM, regardless of the eventual placement of the RPM in the overall design. For example, an RPM might contain 8 flip-flops constrained to be placed into four XC4000 CLBs in a vertical column. The column of four CLBs can then be placed anywhere in any XC4000 Series FPGA.

Relative CLB locations in an RPM are specified using a Relative Location Constraint called "RLOC". RLOC constraints are described in detail in pages 4-71 through 4-95 of the Xilinx Libraries Guide. Fig. 2A shows how RLOC constraints are related to physical location in an array of XC4000 Series logic blocks, and how they are associated with assigned coordinates. Elements having an RLOC value of R0C0 are located in a given CLB corresponding to the (0,0) coordinate location. The next CLB "below" the (0,0) CLB is designated as R1C0, corresponding to the (0,1) coordinate location. When the FPGA design is mapped and placed by the FPGA implementation tools (prior to the final routing step), these RLOC constraints are referenced and, in effect, make the RPM a "jigsaw puzzle piece" to be fitted into the FPGA along with other elements and/or modules. Although the RPM has a rigid size and shape, other logic can be placed within the borders of the RPM.

Logic can also be mapped to elements within the logic block. For example, the function generators and flip-flops inside a CLB can be directly addressed using RLOC constraints. For example, in an XC4000 Series FPGA, the "F" function generator within the (0,0) CLB is addressed by assigning the constraint "RLOC=R0C0.F" to the logic element assigned to that location. The "Y" flip-flop in the same

CLB is addressed by assigning the constraint "RLOC=R0C0.FFY", and so forth.

The exact numbers assigned using RLOC constraints are unimportant in an FPGA with a uniform array of logic blocks. Only the relative values of the RLOC constraints are important. For example, if a user generates an RPM using two vertically adjacent CLBs at a first location, and then generates an identical RPM using two vertically adjacent CLBs at a second location, the resulting RPMs will be exactly the same, due to a process called "normalization". Normalization is the process whereby the RPM creation software (provided by the FPGA manufacturer) scans the selected logic blocks, looking for the RLOC with the lowest row and column number. The RPM creation software then subtracts the lowest row and column numbers from each RLOC, resulting in an RPM where the upper left corner has assigned relative location RLOC=R0C0. Normalization allows a user to create an RPM from any logic implemented anywhere in a logic array. An example of normalization is now described using the CLB array shown in Fig. 2A.

Fig. 2A shows a uniform CLB array 200, such as those in an XC4000 Series FPGA. A first circuit is implemented using two CLBs at coordinates (0,0) and (0,1), forming area 201. A second, identical circuit is implemented using two CLBs at coordinates (2,2) and (2,3), forming area 202. When the second circuit is selected and the RPM creation software is invoked, the software scans the selected logic blocks and determines that the lowest R,C coordinates are (2,2). The value of 2 is then subtracted from each column number and each row number of the selected logic blocks. The resulting RPM thus includes CLBs with relative locations (0,0) and (0,1), exactly as does the RPM resulting from the first circuit.

Although this method of specifying relative locations is a useful way to represent positions in a uniform array, when the array of logic blocks is non-uniform the representation is less straight-forward. In a non-uniform

array where all logic blocks are the same size regardless of their differing functions, the above-described scheme still works as desired, provided the placement software knows what logic elements can be placed in which locations. However, in a non-uniform scheme including blocks of different sizes, locations must be specified in some other manner. For the Virtex FPGA, the locations of CLBs are specified using a first numerical coordinate scheme, while RAM blocks are specified using a second numerical coordinate scheme independent from the first one, as shown in Fig. 2B.

Fig. 2B shows an array 210 of logic blocks in a Virtex FPGA. A centrally-located array of CLBs has assigned locations the same as those shown in Fig. 2A. The flanking columns of RAM blocks have locations specified independently from those of the CLB array, but in a similar manner. (The numbers and relative sizes of CLBs and RAM blocks in the figures herein are not representative of actual Virtex and Virtex-II FPGAs.)

In the logic array of Fig. 2B, an RPM implemented using only CLBs can be defined in the same manner as the example in Fig. 2A, at any point in the array, and is normalized by the RPM creation software. An RPM including only RAM blocks can be defined similarly, and normalization operates correctly. However, an RPM including both types of logic blocks does not normalize correctly. Referring to Fig. 2B, an RPM created in area 212 of array 210 would ideally result in the same RPM as if it were created in area 211. However, this is not the case, as shown by the following example.

A first circuit is implemented in area 211 of array 210, using one RAM block at coordinates (0,0) and two CLBs at coordinates (0,0) and (0,1). When converted to an RPM, this first RPM has the same shape as area 211. A second, identical circuit is implemented in area 212, using one RAM block at coordinates (0,1) and two CLBs at coordinates (0,2) and (0,3). When the second circuit is selected and the RPM creation software is invoked, the software scans

the selected logic blocks and determines that the lowest R,C coordinates are (0,1). The value of 1 is then subtracted from each row number of the selected logic blocks, while the column numbers remain unchanged. The resulting RPM thus includes one RAM block with relative coordinates (0,0) and two CLBs with relative locations (0,1) and (0,2). While logically correct, this second RPM does not have the shape of the original area 212. Instead, it includes, for example, the logic blocks in area 213. Therefore, creating RPMs including logic blocks of more than one type is a problematic process.

This difficulty is augmented in the Virtex-II FPGA, which has three types of logic blocks in the logic array. Using the previously-described method of designating relative locations, each type of logic block would have a separate coordinate system, as shown in Fig. 2C. The situation is further complicated if the logic array is defined to include the ring of IOBs shown in Fig. 1C, which also includes the Digital Clock Manager (DCM) blocks and other types of blocks not shown in Fig. 1C.

An RPM can be moved from place to place in a non-uniform logic array, provided that the new location includes the correct type of logic block in the correct positions for the RPM logic. However, the processes of defining and modifying an RPM in a non-uniform logic array are subject to the limitations described above. It is therefore desirable to provide methods for specifying circuit element positions in a uniform manner that can be applied to any logic block in a non-uniform logic array.

SUMMARY OF THE INVENTION

The invention provides methods and software for designating circuit element positions using uniform coordinate systems that can be applied to non-uniform logic arrays. A "site map" is constructed comprising a uniform array of "sites". A uniform coordinate system is then applied to the site map. The various logic blocks--for example, the slices, tristate buffers, RAM blocks, and

multiplier blocks of the Xilinx Virtex-II FPGA--are mapped to the site array. The result is the imposition of a uniform coordinate system on a non-uniform logic array, using the intervening abstraction of a site array. The logic blocks in the logic array may be of different types and sizes.

Because the site array is uniform, a relative location constraint applied to a site within the site array retains its validity regardless of the location of the site within the site array, even when the relative location constraints are normalized. However, during the subsequent placement operation the implementation software must take into account which sites support which logic blocks, so that logic blocks are only mapped to sites that support the implemented functions.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example, and not by way of limitation, in the following figures, in which like reference numerals refer to similar elements.

Fig. 1A shows a simplified diagram of an XC4000-Series FPGA.

Fig. 1B shows a simplified diagram of a Virtex FPGA.

Fig. 1C shows a simplified diagram of a Virtex-II FPGA.

Fig. 2A shows a uniform CLB array, such as those in an XC4000 Series FPGA, and an associated coordinate system.

Fig. 2B shows an array of CLBs and RAM blocks in a Virtex FPGA, and an associated coordinate system.

Fig. 2C shows an array of CLBs, RAM blocks, and multiplier blocks in a Virtex-II FPGA, with associated coordinates applied according to the system of Fig. 2B.

Fig. 3A shows a site map created in accordance with the present invention, with an associated set of uniform X,Y coordinates.

Fig. 3B shows the logic array of CLBs, RAM blocks, and multiplier blocks from Fig. 2C, assigned to the site map of Fig. 3A and its associated set of uniform coordinates.

Fig. 3C shows site area 301 of Fig. 3B, and examples of ways in which the associated CLB could be mapped to sites in different site arrays.

Fig. 3D shows site area 302 of Fig. 3B, and examples of ways in which the associated RAM and multiplier blocks could be mapped to sites in different site arrays.

Fig. 4 shows a Xilinx XC2V40 FPGA device mapped to a site map according to a method of the present invention.

Fig. 5 is a flow diagram showing a method of designating positions in a non-uniform array using a uniform coordinate system according to one aspect of the present invention.

DETAILED DESCRIPTION OF THE DRAWINGS

The present invention has been found to be particularly applicable and beneficial when applied to field programmable gate arrays (FPGAs) including non-uniform arrays of logic blocks. While the present invention is not so limited, an appreciation of the present invention is presented by way of specific examples showing the application of the invention to the Virtex-II FPGA. However, it will be apparent to one skilled in the art that the present invention can be applied to other programmable logic devices including logic arrays. It will further be apparent to one skilled in the art that the present invention can be applied to any type of non-uniform array, including those not included in a programmable logic device.

To map a non-uniform array in a uniform fashion, a "site map" is created that corresponds in a desired fashion to the non-uniform logic array. Generally, a logic block in the logic array corresponds to an array of sites in the site map. For example, Fig. 3A shows a site map 300 that can be used to represent the non-uniform logic array 220 shown in Fig. 2C. When site map 300 is applied to logic

array 220, each CLB corresponds to a 4x3 array of sites. For example, site area 301 in site array 300 represents the CLB at location (1,0) in logic array 220. However, in this embodiment of the invention the RAM and multiplier blocks share a site area, e.g., area 302. The reason for this shared site area is that in this example the width of the combined RAM and multiplier blocks is the same as the width of one CLB block. Therefore, combining the RAM and multiplier blocks into one site area simplifies the FPGA implementation software. In this example, the RAM and multiplier blocks together are the width of one CLB and the height of two CLBs. Therefore, the site area 302 for one RAM block and one multiplier block includes 8x3 sites.

The site map has an associated uniform coordinate system. In the example of Fig. 3A, the uniform coordinate system is the well-known X,Y coordinate system. However, any other uniform coordinate system could be used. For example, the R,C (row, column) coordinate system used in Figs. 2A-2C could be used. Regardless of the coordinate system used, each site in the site array has a specific, uniformly-numbered location. For example, using the X,Y coordinate system as shown in Fig. 3A, site area 301 includes sites (3,12), (3,13), (3,14), (3,15), (4,12), (4,13), (4,14), (4,15), (5,12), (5,13), (5,14), and (5,15).

The number of sites in the site area and the aspect ratio of the site area for each logic block are at the discretion of the person assigning the coordinate system. They are preferably selected to simplify the implementation software without creating an inordinately large site map.

Fig. 3B shows the site map of Fig. 3A after the logic blocks of Fig. 2C have been assigned to the sites in the site map. Note that the CLBs have been "broken down" into smaller logic blocks within the CLB (e.g., see site area 301). These logic blocks are individually addressable elements within the CLB. For example, a Virtex-II CLB includes four "slices", as described above. These slices (S0-S3) are assigned to individual sites. Similarly, the two tristate buffers in each CLB also have assigned sites.

Assigning specific sites to these elements makes it easy to specify locations for logic targeted to these elements, simply by specifying the coordinates of the site. For example, to assign a circuit element to Slice 0 of the CLB in site area 301, the relative location constraint "RLOC=SLICE_X3Y12" is attached to the element. If an absolute location is desired, rather than a relative location, the absolute location constraint "LOC=SLICE_X3Y12" is used.

Because the coordinate system is uniform, the normalization process when an RPM is created does not change the shape of the original circuit implementation. In other words, an RPM can be created at any location in the non-uniform array, and the relative positions of the blocks in the RPM are the same, unlike in the example described with reference to Fig. 2B.

Of course, an RPM can only be placed in locations that support that RPM. For example, an RPM that includes CLBs and a RAM block cannot be placed in a portion of the logic array that does not include a RAM block in the relative location specified in the RPM. In one embodiment, the implementation software scans the mapped site array prior to placing any RPM, and locates potential site areas having the correct types of logic blocks available to match the relative placements in the RPM. Only these valid site areas are considered during the placement process.

The number of sites assigned to each logic block, the aspect ratio of the site area, and the location of each element within the site area can vary at the discretion of the person creating the site map. For example, site area 301 could have taken on any of the aspects 301a-c shown in Fig. 3C. The location of the element site within the site area need not correspond to the actual physical location of the element within the logic block. The selection is generally made based on what will most simplify the FPGA implementation software. For example, the slice element sites were assigned as shown in site area 301 because the shift register in the Virtex-II CLB flows from slice S3

through slices S2 and S1 to slice S0. Site area 301a shows another possible implementation, which more closely corresponds to the carry chain logic in the Virtex-II CLB (two carry chains, S1-S0 and S3-S2). Site areas 301b and 301c show other possible implementations.

Site area 302 includes the RAM and multiplier blocks. The tristate buffer element sites have been retained, because the tristate buffers are still available in these blocks. However, note that in this embodiment the slice element sites are also retained, although these resources are not available in the RAM and multiplier blocks. Again, this is a choice that was made for the sake of the FPGA implementation software. In other embodiments, only elements actually available in the logic block are mapped to the site map.

Fig. 3D shows various aspects 302a-e that could have been assigned to site area 302.

Fig. 4 shows an actual site map 400 for an XC2V40 Virtex-II FGPA available from Xilinx, Inc., with the various logic blocks mapped to the site map. The site map uses an X,Y coordinate system, with X-axis values varying from 0 to 29, and Y-axis values varying from 0 to 39. Table 1 shows the various elements that are mapped to sites in site map 400. In the actual Virtex-II device, as shown in Fig. 4, the RAM and multiplier blocks together are 4 CLBs high and the same width as one CLB. Note that when the methods of the invention are used, the site map can be extended to cover not just the non-uniform logic array in the center of the IOB ring, but also the IOB ring itself. Thus, an entire FPGA can be mapped to a single uniform coordinate system.

Table 1

Symbol	Element	Location
S0,S1,S2,S3	Slices S0,S1,S2,S3	CLB
CA	Capture FF Data for Readback	IOB ring
DC	Digital Clock Manager	IOB ring
G	Global Clock Buffer	IOB ring
I	IOB	I/O block
IC	Capture I/O Data for Readback	IOB ring
B	Boundary Scan Control	IOB ring
M	Multiplier	Multiplier block
P	PCI Logic	IOB ring
R	Block RAM	RAM block
S	Startup Logic	IOB ring
T	Tristate Buffer	CLB, RAM block, Multiplier block

Fig. 5 is a flow diagram illustrating a method of designating positions in a non-uniform array using a uniform coordinate system according to a first aspect of the present invention. In step 501, a uniform site map is created. In step 2, a uniform coordinate system is assigned to the site map, such that each site in the site map has a unique set of coordinates. Fig. 3A shows an example of a uniform site map having a uniform coordinate system. In step 503, a non-uniform array of logic blocks is mapped to the site array, such that each logic block is mapped to one site. In one embodiment, the logic blocks include CLBs, RAM blocks, and multiplier blocks. In another embodiment, the logic blocks include slices and tristate buffers. As a result of the mapping step, each logic block can be addressed using the uniform coordinate system.

In optional step 504, relative location constraints are assigned to two or more logic blocks in the process of creating an RPM from the constrained logic blocks. In further optional step 505, an RPM is created from the constrained logic blocks. The shape of the resulting RPM is independent of the location in the array of the constrained logic blocks, even if the RPM creation process includes a normalization of the specified coordinates.

To simplify the location specification process for the user, a graphical tool is preferably provided that informs the user of the coordinates of each logic block. For example, the FPGA implementation tools from Xilinx, Inc., include a graphical editor. Preferably, when a user views this graphical editor and places a cursor on one element (or logic block) in the display, the uniform coordinates of the element are displayed on the screen. For example, placing the cursor on the lower tristate buffer of the CLB associated with site 301 (see Fig. 3B) results in the value "TBUF_X4_Y12" being displayed. Thus, the uniform coordinates of each logic block are easily determined by the user.

Those having skill in the relevant arts of the invention will now perceive various modifications and additions that may be made as a result of the disclosure herein. All such modifications and additions are deemed to be within the scope of the invention, which is to be limited only by the appended claims and their equivalents.

CLAIMS

What is claimed is:

1. A method of designating circuit element positions for a non-uniform array of logic blocks, the method comprising:

creating a uniform site map;

assigning a uniform coordinate system to the uniform site map; and

mapping the non-uniform array of logic blocks to the uniform site map.

2. The method of Claim 1, wherein the non-uniform array of logic blocks forms at least a portion of a Field Programmable Gate Array.

3. The method of Claim 1, further comprising:

assigning relative location constraints to a plurality of logic blocks in the non-uniform array of logic blocks using the uniform coordinate system.

4. The method of Claim 3, wherein the plurality of logic blocks includes logic blocks of at least two different types.

5. The method of Claim 3, wherein the plurality of logic blocks includes logic blocks of at least two different sizes.

6. The method of Claim 3, further comprising:

generating a relationally placed macro (RPM) from the plurality of logic blocks.

7. The method of Claim 6, wherein generating an RPM includes normalizing the relative location constraints.

8. Computer-readable media comprising computer software for designating circuit element positions for a non-uniform array of logic blocks, the computer software comprising:

means for creating a uniform site map;

means for assigning a uniform coordinate system to the uniform site map; and

means for mapping the non-uniform array of logic blocks to the uniform site map.

9. The computer-readable media of Claim 8, the computer software further comprising:

means for assigning relative location constraints to a plurality of logic blocks in the non-uniform array of logic blocks using the uniform coordinate system.

10. The computer-readable media of Claim 9, the computer software further comprising:

means for generating a relationally placed macro (RPM) from the plurality of logic blocks.

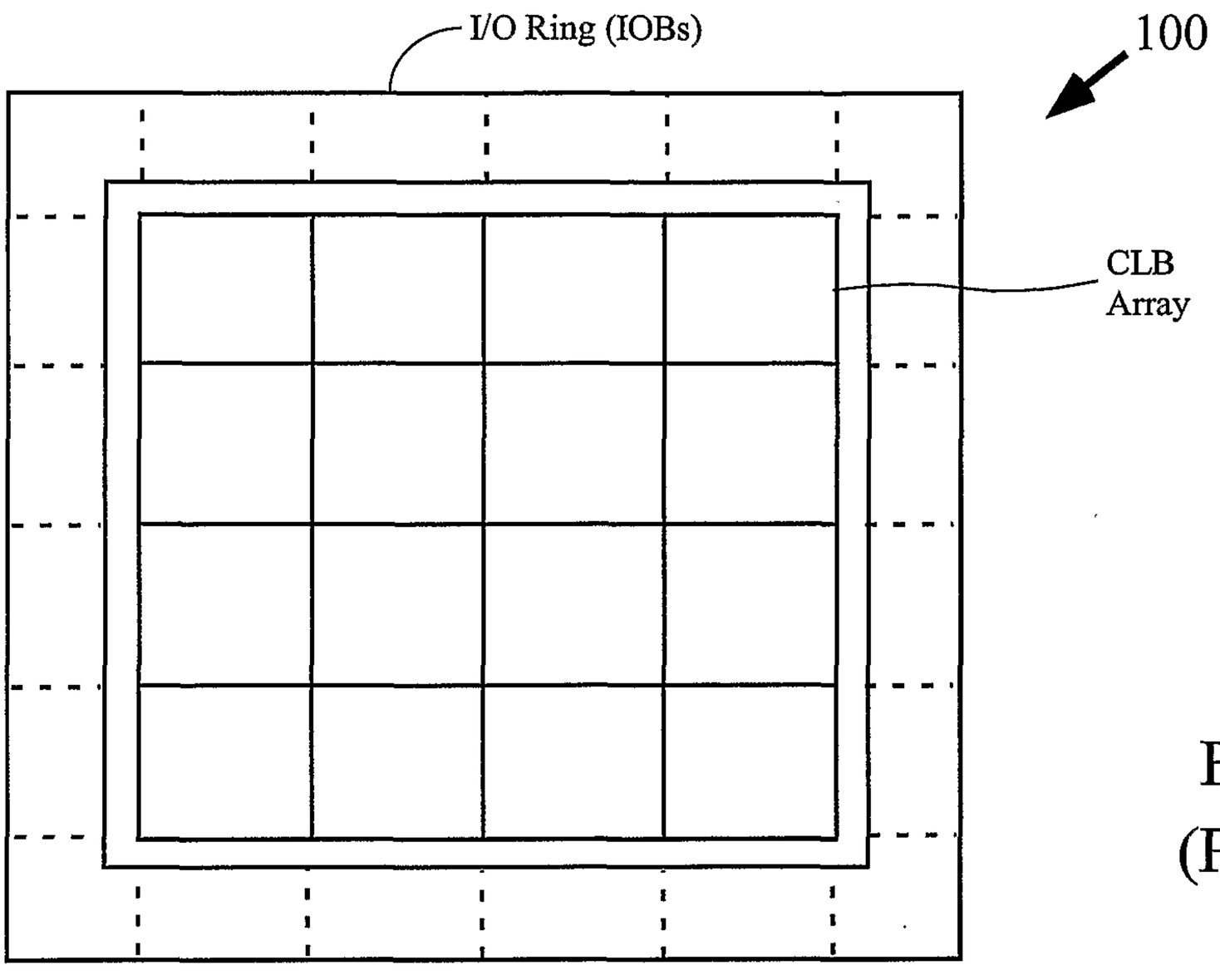


FIG. 1A
(Prior Art)

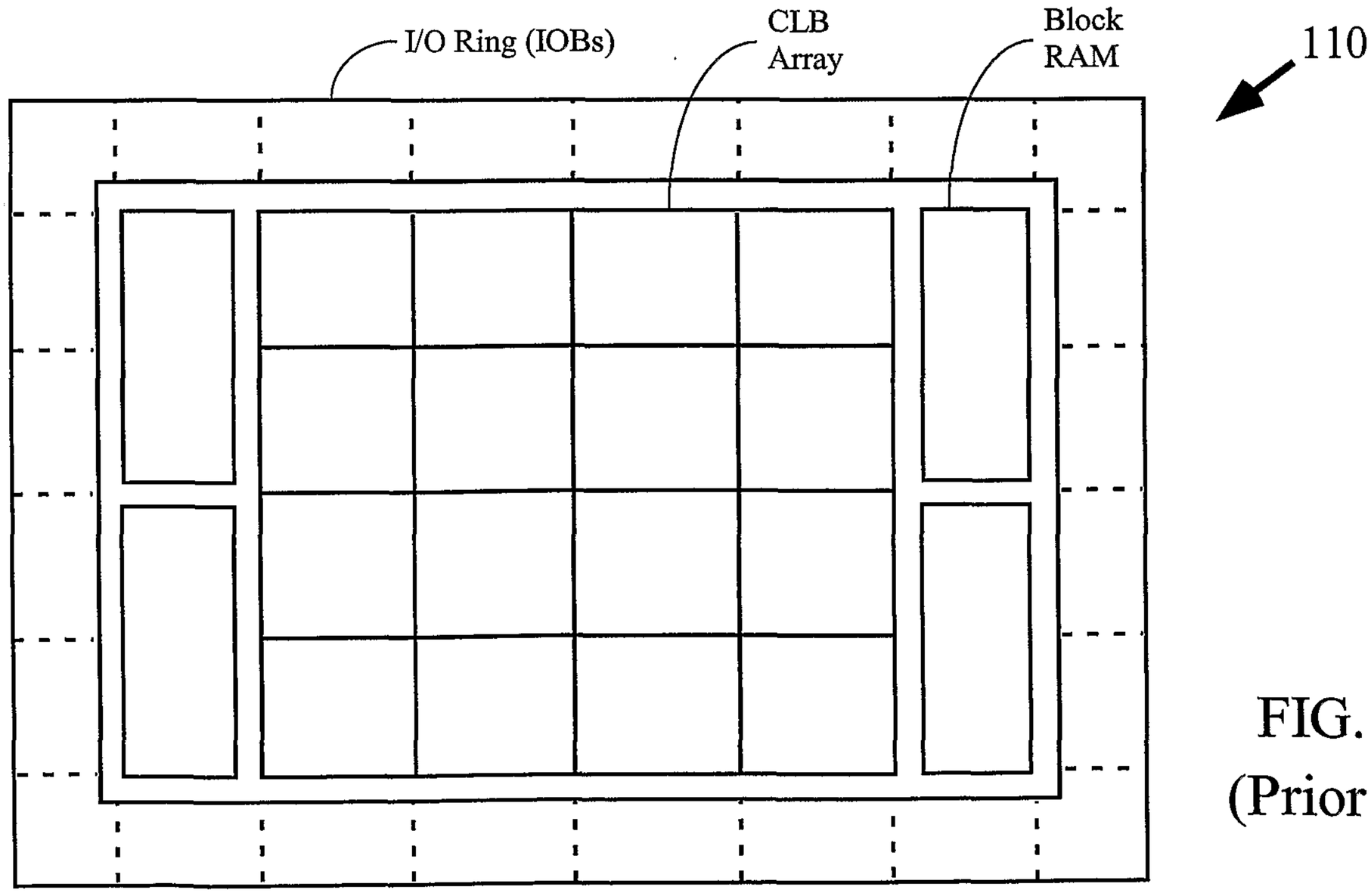


FIG. 1B
(Prior Art)

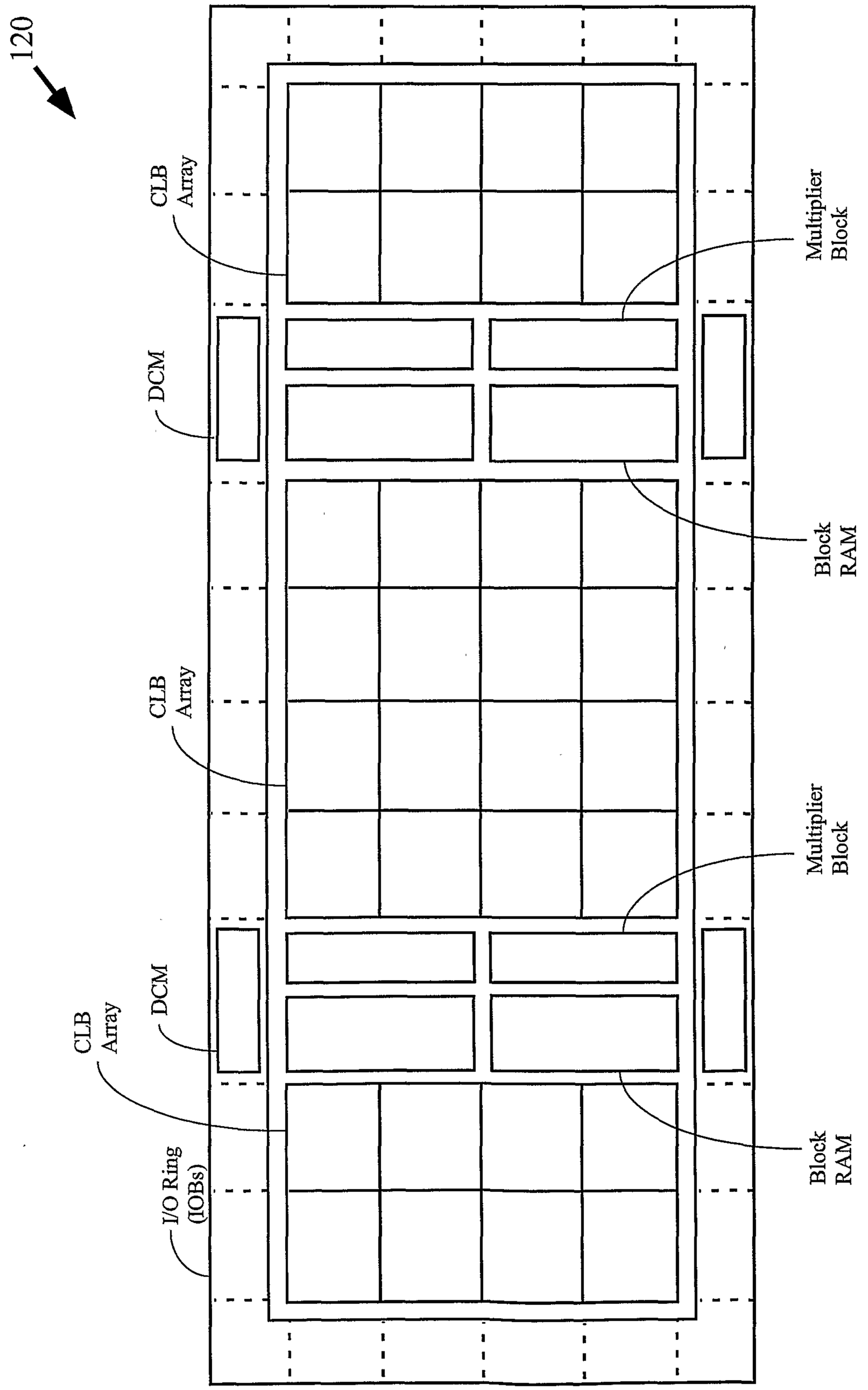

120
I/O Ring
(IOBs)
CLB
Array
DCM
Multiplier
Block
Block
RAM
CLB
Array
DCM
Multiplier
Block
Block
RAM
CLB
Array
DCM
I/O Ring
(IOBs)
Multiplier
Block
Block
RAM

FIG. 1C

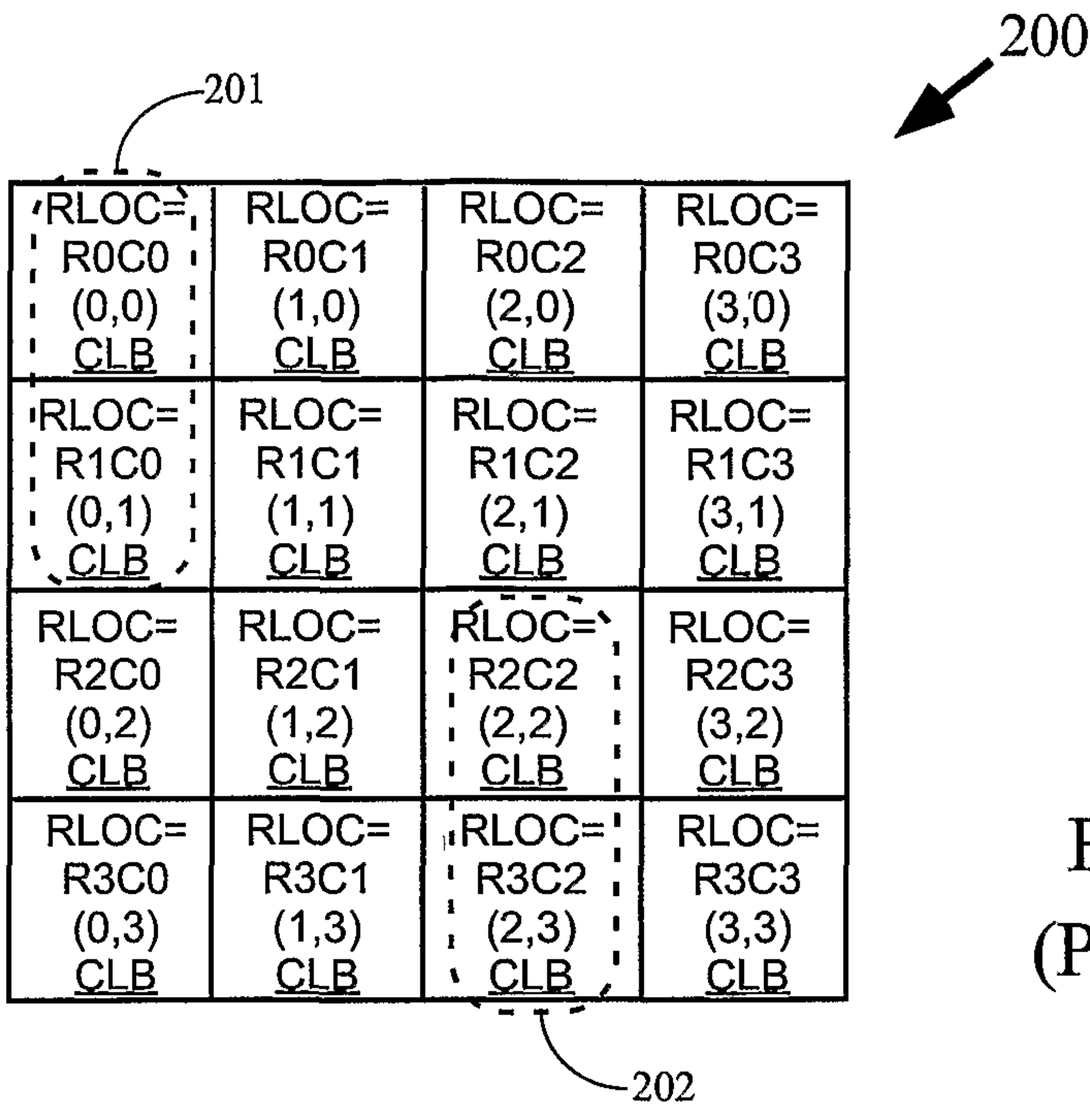


FIG. 2A
(Prior Art)

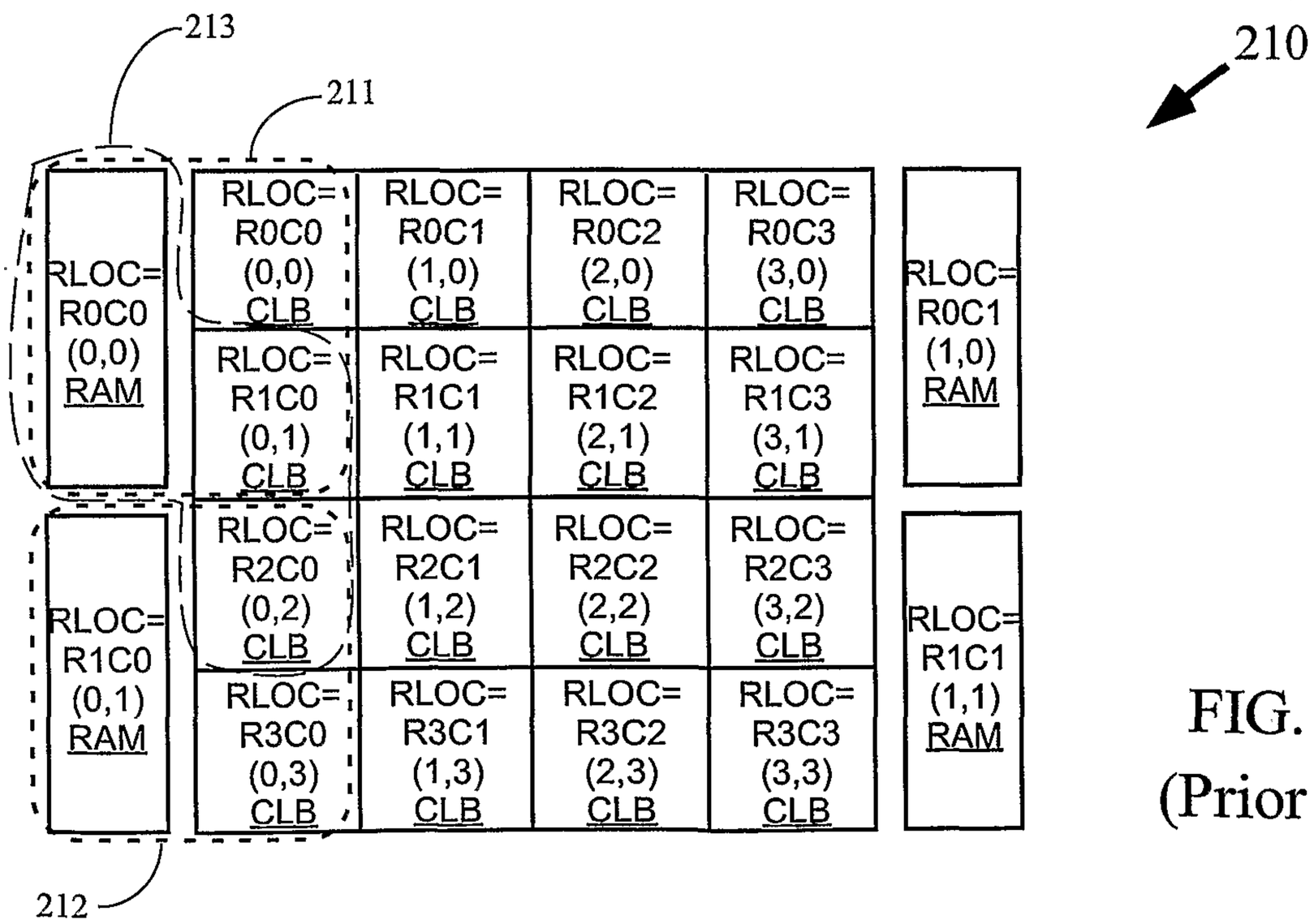


FIG. 2B
(Prior Art)

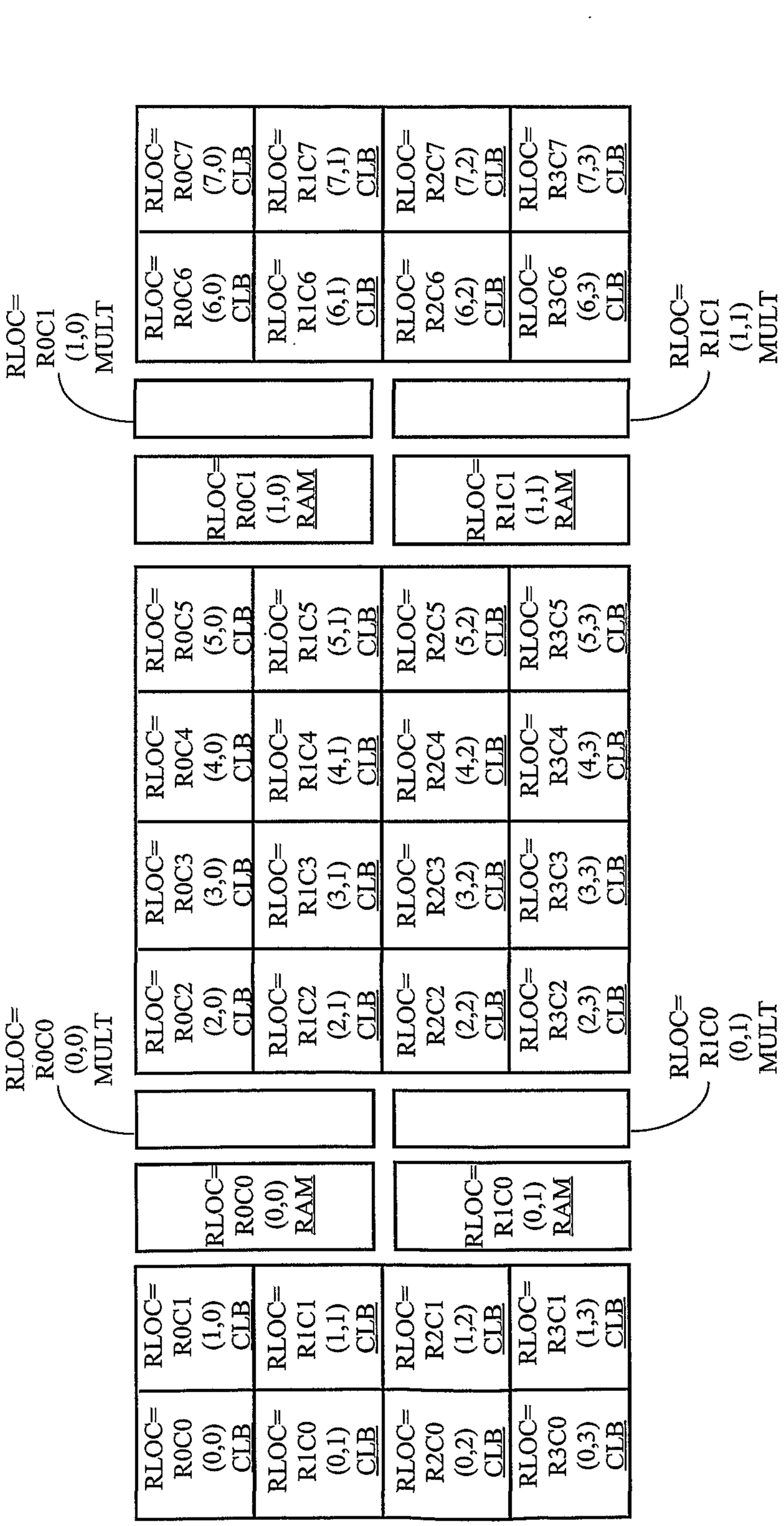

220
RLOC= R0C1 (1,0) MULT
RLOC= R1C1 (1,1) MULT
RLOC= R0C0 (0,0) MULT
RLOC= R1C0 (0,1) MULT
RLOC= R0C6 (6,0) CLB
RLOC= R1C6 (6,1) CLB
RLOC= R2C6 (6,2) CLB
RLOC= R3C6 (6,3) CLB
RLOC= R0C7 (7,0) CLB
RLOC= R1C7 (7,1) CLB
RLOC= R2C7 (7,2) CLB
RLOC= R3C7 (7,3) CLB
RLOC= R0C1 (1,0) RAM
RLOC= R1C1 (1,1) RAM
RLOC= R0C5 (5,0) CLB
RLOC= R1C5 (5,1) CLB
RLOC= R2C5 (5,2) CLB
RLOC= R3C5 (5,3) CLB
RLOC= R0C4 (4,0) CLB
RLOC= R1C4 (4,1) CLB
RLOC= R2C4 (4,2) CLB
RLOC= R3C4 (4,3) CLB
RLOC= R0C3 (3,0) CLB
RLOC= R1C3 (3,1) CLB
RLOC= R2C3 (3,2) CLB
RLOC= R3C3 (3,3) CLB
RLOC= R0C2 (2,0) CLB
RLOC= R1C2 (2,1) CLB
RLOC= R2C2 (2,2) CLB
RLOC= R3C2 (2,3) CLB
RLOC= R0C0 (0,0) RAM
RLOC= R1C0 (0,1) RAM
RLOC= R0C1 (1,0) CLB
RLOC= R1C1 (1,1) CLB
RLOC= R2C1 (1,2) CLB
RLOC= R3C1 (1,3) CLB
RLOC= R0C0 (0,0) CLB
RLOC= R1C0 (0,1) CLB
RLOC= R2C0 (0,2) CLB
RLOC= R3C0 (0,3) CLB

FIG. 2C

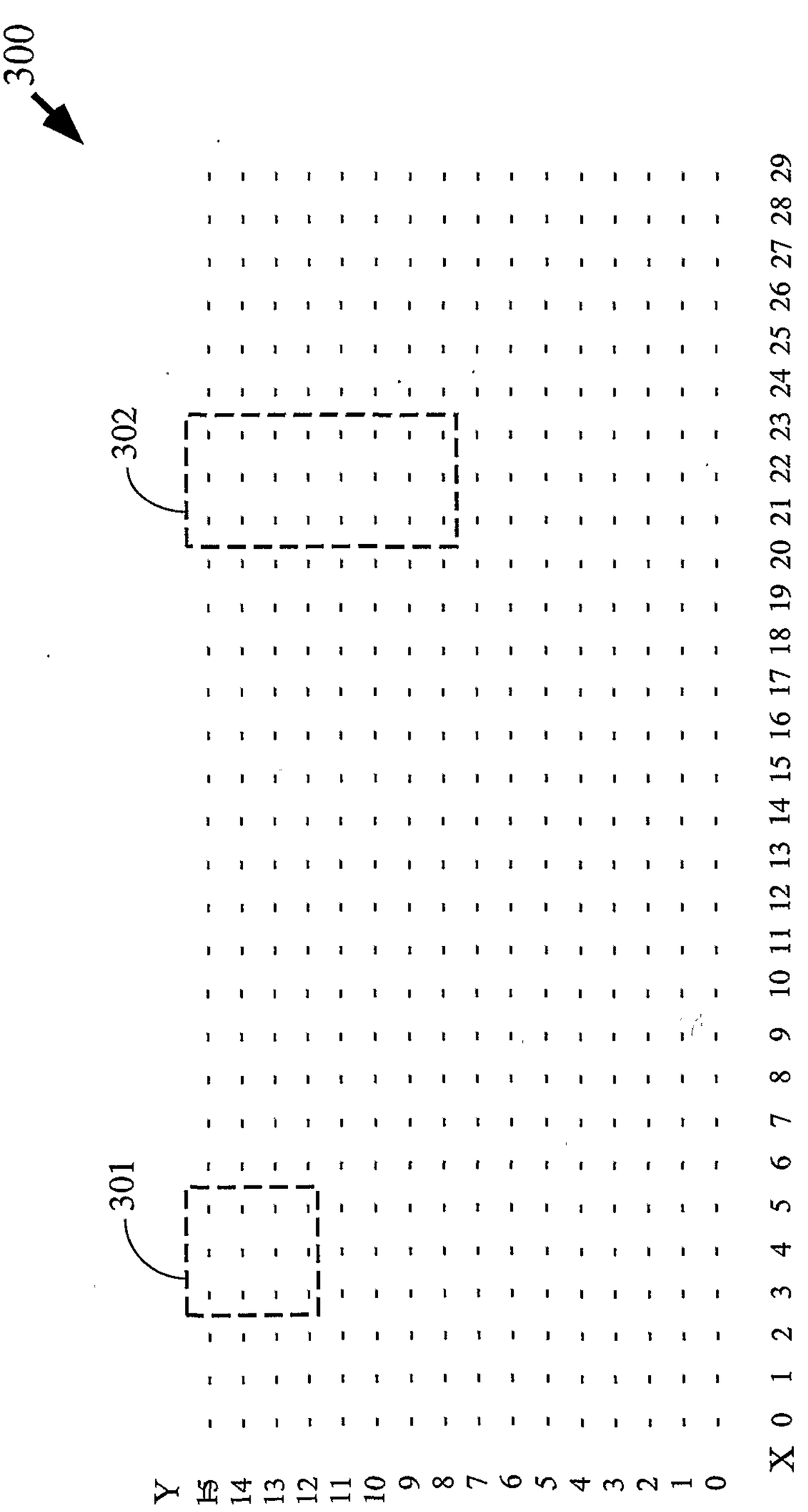

300
302
301
Y
X = 0 1 2
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29

FIG. 3A

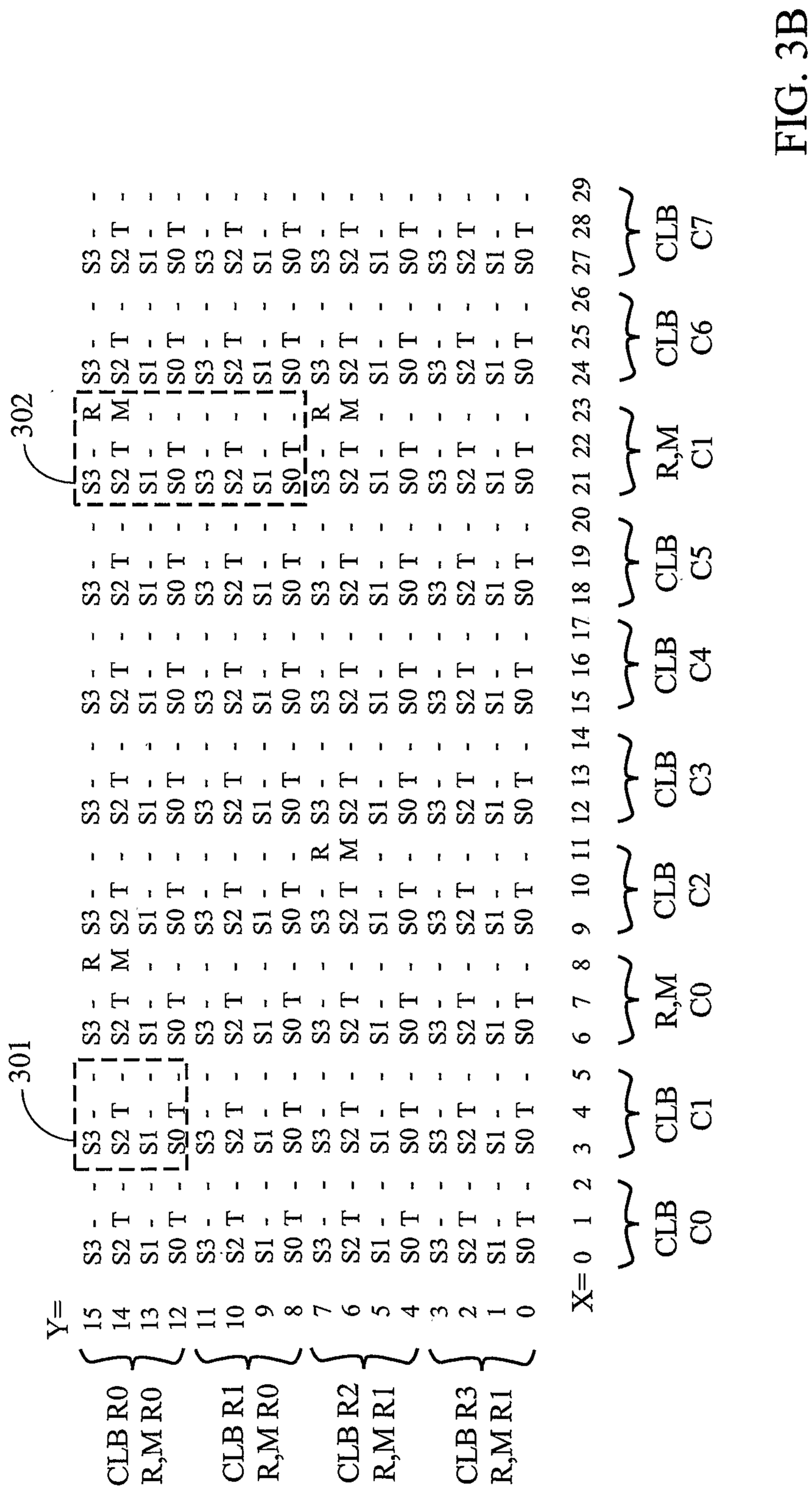

302
301
CLB R0
R,M R0
CLB R1
R,M R0
CLB R2
R,M R1
CLB R3
R,M R1
Y=
X= 0 1 2
CLB C0
CLB C1 R,M C0
CLB C2 CLB C3 CLB C4
CLB C5 R,M C1 CLB C6 CLB C7

FIG. 3B

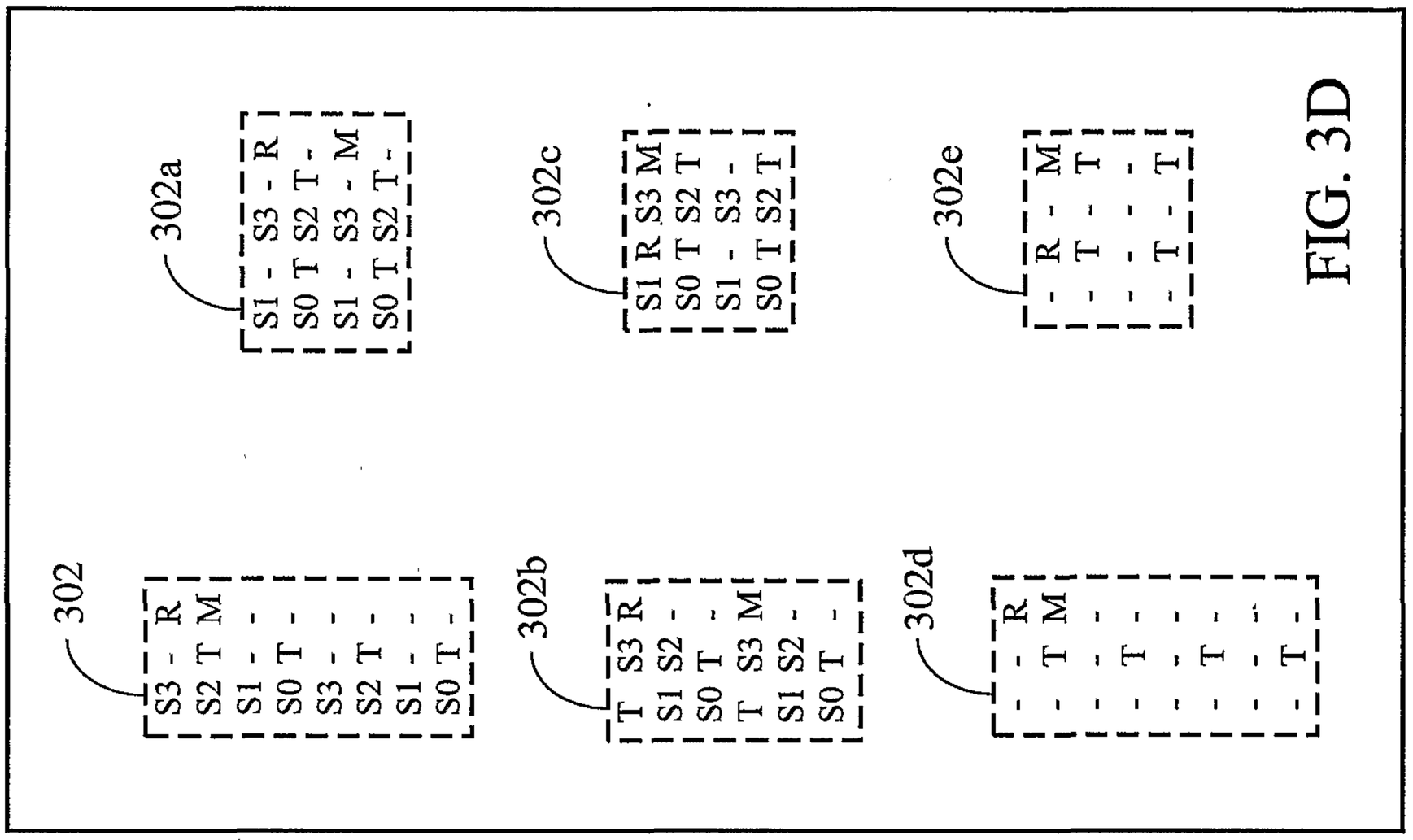

302
S3 - R
S2 T M
S1 - -
S0 T -
S3 - -
S2 T -
S1 - -
S0 T -
302a
S1 - S3 - R
S0 T S2 T -
S1 - S3 - M
S0 T S2 T -
302b
T S3 R
S1 S2 -
S0 T -
T S3 M
S1 S2 -
S0 T -
302c
S1 R S3 M
S0 T S2 T
S1 - S3 -
S0 T S2 T
302d
- R
- T M
- - -
- T -
- - -
- T -
- - -
- T -
302e
- R - M
- T - T
- - - -
- T - T

FIG. 3D

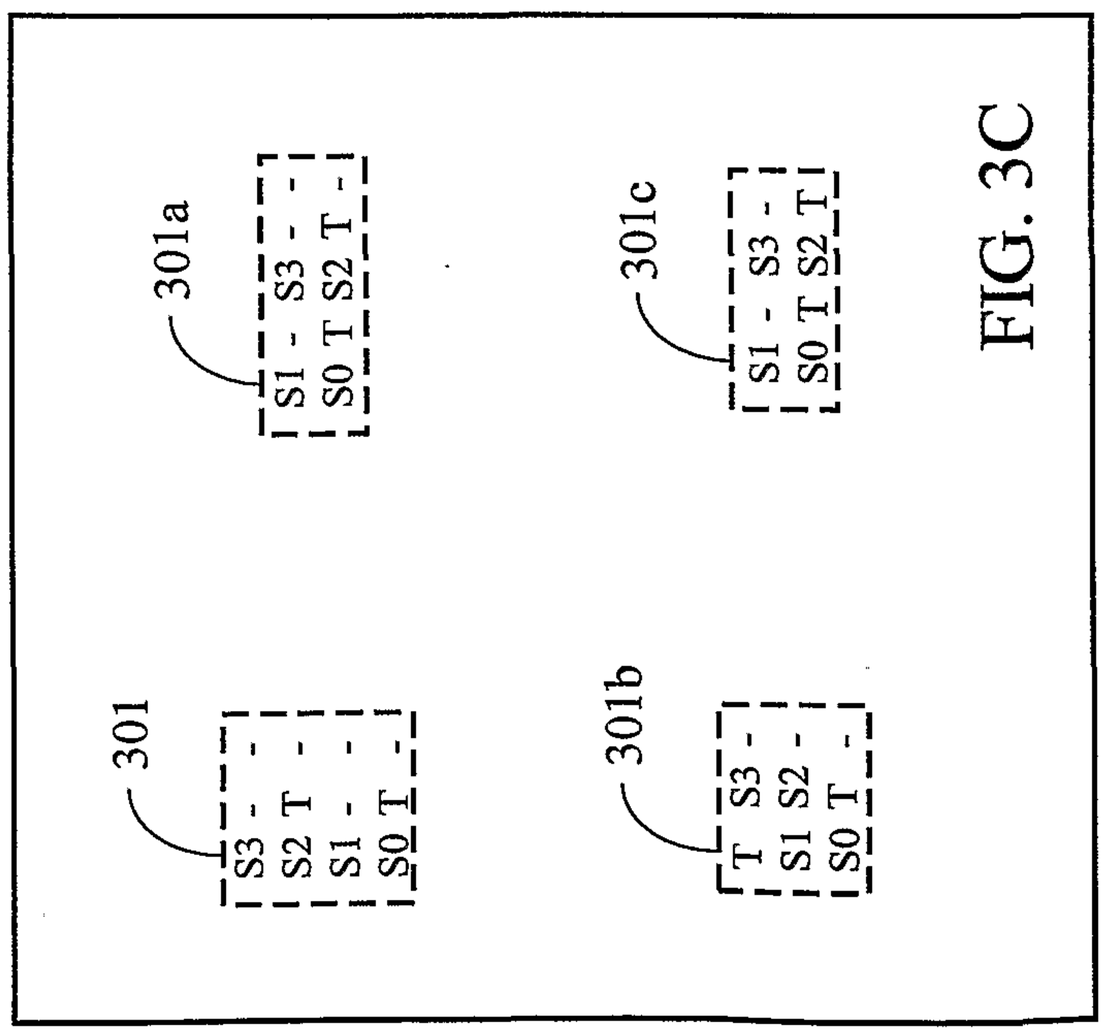

301
S3 - - -
S2 T - -
S1 - - -
S0 T - -
301a
S1 - S3 - -
S0 T S2 T -
301b
T S3 - -
S1 S2 - -
S0 T - -
301c
S1 - S3 - -
S0 T S2 T -

FIG. 3C

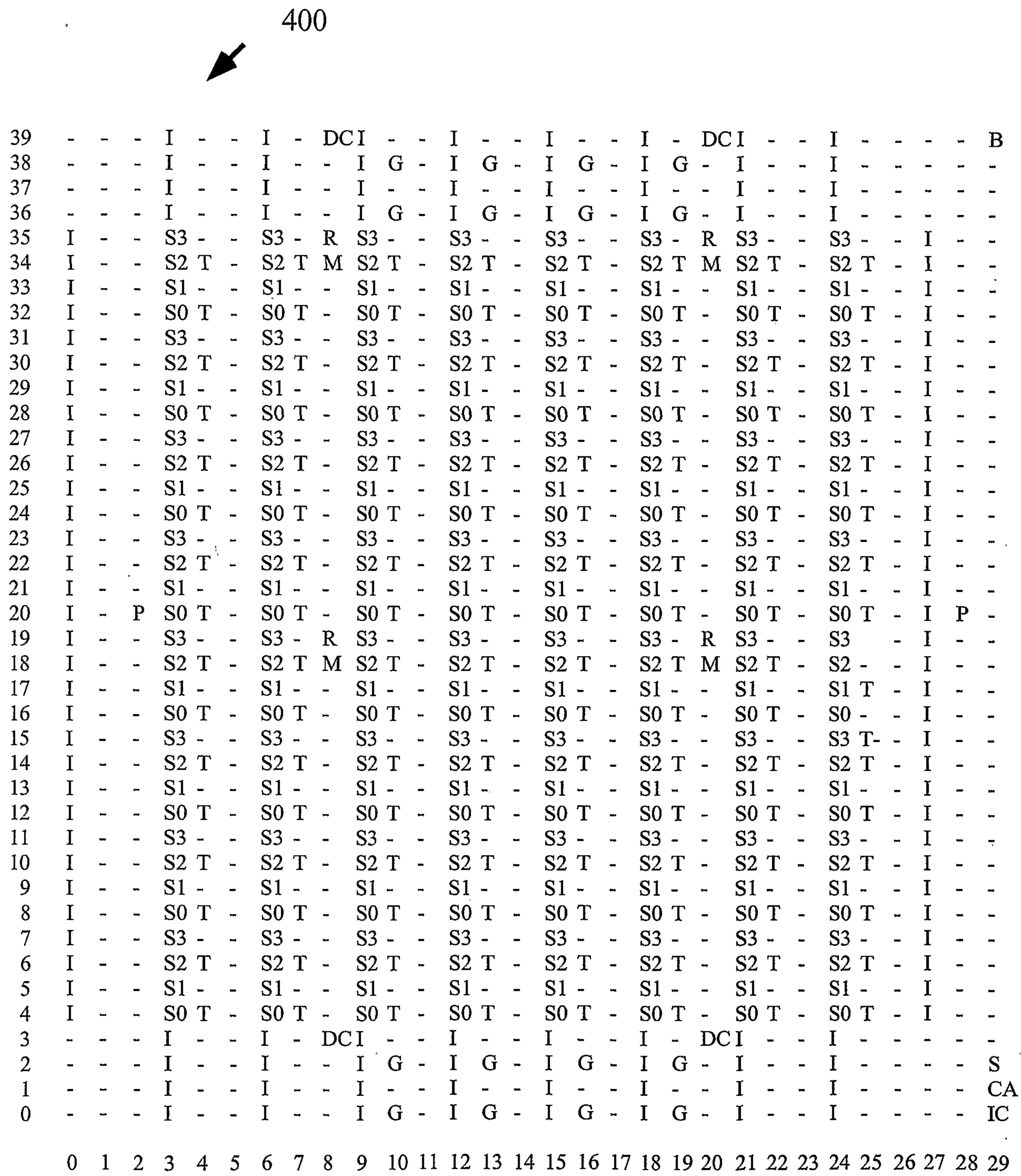

400
39 - - - I - - I - DC I - - I - - I - - I - DC I - - I - - - - - B
38 - - - I - - I - - I G - I G - I G - I G - I - - I - - - - - -
37 - - - I - - I - - I - - I - - I - - I - - I - - I - - - - - -
36 - - - I - - I - - I G - I G - I G - I G - I - - I - - - - - -
35 I - - S3 - - S3 - R S3 - - S3 - - S3 - - S3 - R S3 - - S3 - - I - -
34 I - - S2 T - S2 T M S2 T - S2 T - S2 T - S2 T M S2 T - S2 T - I - -
33 I - - S1 - - S1 - - S1 - - S1 - - S1 - - S1 - - S1 - - S1 - - I - -
32 I - - S0 T - S0 T - S0 T - S0 T - S0 T - S0 T - S0 T - S0 T - I - -
31 I - - S3 - - S3 - - S3 - - S3 - - S3 - - S3 - - S3 - - S3 - - I - -
30 I - - S2 T - S2 T - S2 T - S2 T - S2 T - S2 T - S2 T - S2 T - I - -
29 I - - S1 - - S1 - - S1 - - S1 - - S1 - - S1 - - S1 - - S1 - - I - -
28 I - - S0 T - S0 T - S0 T - S0 T - S0 T - S0 T - S0 T - S0 T - I - -
27 I - - S3 - - S3 - - S3 - - S3 - - S3 - - S3 - - S3 - - S3 - - I - -
26 I - - S2 T - S2 T - S2 T - S2 T - S2 T - S2 T - S2 T - S2 T - I - -
25 I - - S1 - - S1 - - S1 - - S1 - - S1 - - S1 - - S1 - - S1 - - I - -
24 I - - S0 T - S0 T - S0 T - S0 T - S0 T - S0 T - S0 T - S0 T - I - -
23 I - - S3 - - S3 - - S3 - - S3 - - S3 - - S3 - - S3 - - S3 - - I - -
22 I - - S2 T - S2 T - S2 T - S2 T - S2 T - S2 T - S2 T - S2 T - I - -
21 I - - S1 - - S1 - - S1 - - S1 - - S1 - - S1 - - S1 - - S1 - - I - -
20 I - P S0 T - S0 T - S0 T - S0 T - S0 T - S0 T - S0 T - S0 T - I P -
19 I - - S3 - - S3 - R S3 - - S3 - - S3 - - S3 - R S3 - - S3 - - I - -
18 I - - S2 T - S2 T M S2 T - S2 T - S2 T - S2 T M S2 T - S2 - - I - -
17 I - - S1 - - S1 - - S1 - - S1 - - S1 - - S1 - - S1 - - S1 T - I - -
16 I - - S0 T - S0 T - S0 T - S0 T - S0 T - S0 T - S0 T - S0 - - I - -
15 I - - S3 - - S3 - - S3 - - S3 - - S3 - - S3 - - S3 - - S3 T- - I - -
14 I - - S2 T - S2 T - S2 T - S2 T - S2 T - S2 T - S2 T - S2 T - I - -
13 I - - S1 - - S1 - - S1 - - S1 - - S1 - - S1 - - S1 - - S1 - - I - -
12 I - - S0 T - S0 T - S0 T - S0 T - S0 T - S0 T - S0 T - S0 T - I - -
11 I - - S3 - - S3 - - S3 - - S3 - - S3 - - S3 - - S3 - - S3 - - I - -
10 I - - S2 T - S2 T - S2 T - S2 T - S2 T - S2 T - S2 T - S2 T - I - -
9 I - - S1 - - S1 - - S1 - - S1 - - S1 - - S1 - - S1 - - S1 - - I - -
8 I - - S0 T - S0 T - S0 T - S0 T - S0 T - S0 T - S0 T - S0 T - I - -
7 I - - S3 - - S3 - - S3 - - S3 - - S3 - - S3 - - S3 - - S3 - - I - -
6 I - - S2 T - S2 T - S2 T - S2 T - S2 T - S2 T - S2 T - S2 T - I - -
5 I - - S1 - - S1 - - S1 - - S1 - - S1 - - S1 - - S1 - - S1 - - I - -
4 I - - S0 T - S0 T - S0 T - S0 T - S0 T - S0 T - S0 T - S0 T - I - -
3 - - - I - - I - DC I - - I - - I - - I - DC I - - I - - - - - -
2 - - - I - - I - - I G - I G - I G - I G - I - - I - - - - - S
1 - - - I - - I - - I - - I - - I - - I - - I - - I - - - - - CA
0 - - - I - - I - - I G - I G - I G - I G - I - - I - - - - - IC
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29

FIG. 4

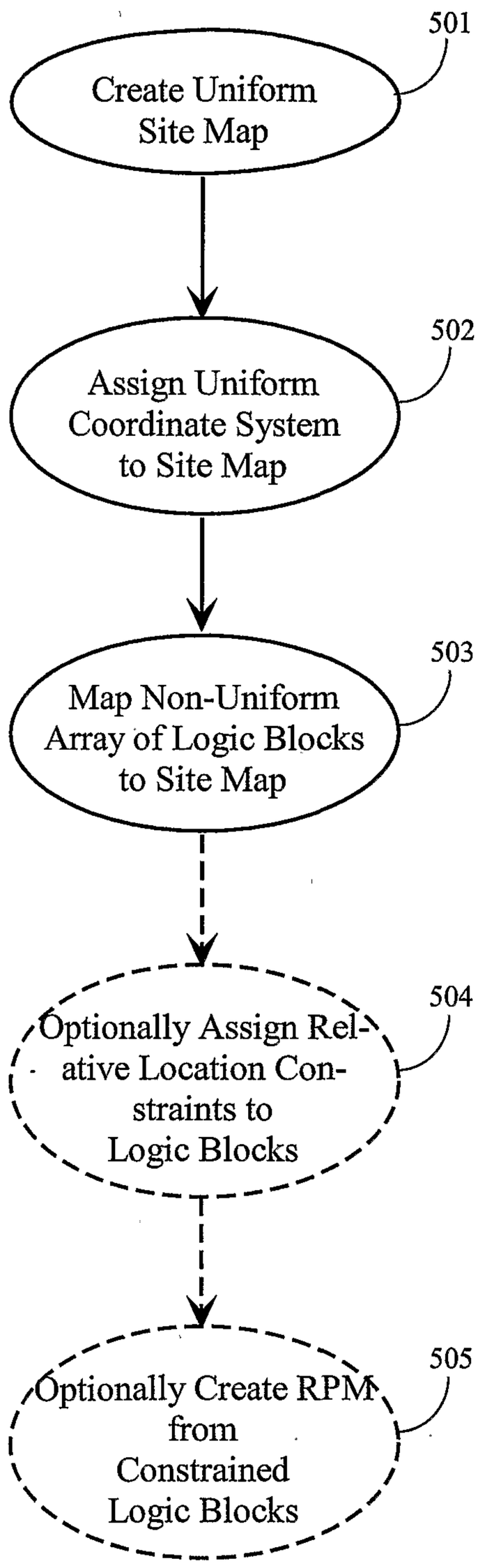
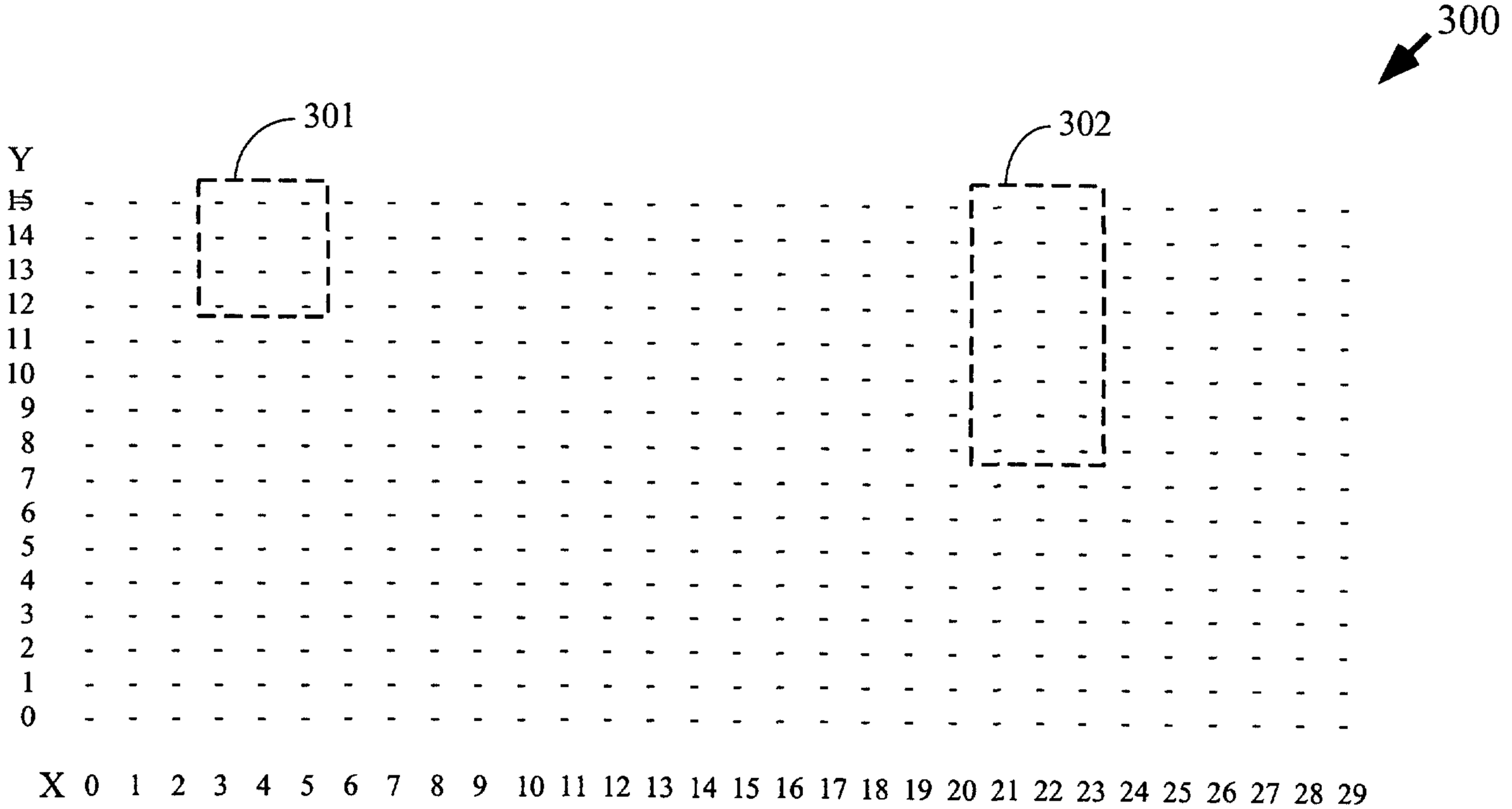

Create Uniform Site Map
501
Assign Uniform Coordinate System to Site Map
502
Map Non-Uniform Array of Logic Blocks to Site Map
503
Optionally Assign Relative Location Constraints to Logic Blocks
504
Optionally Create RPM from Constrained Logic Blocks
505

FIG. 5


300
301
302
Y
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
X = 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29