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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
12 February 2004 (12.02.2004)

PCT

(10) International Publication Number
WO 2004/013967 A2

(51) International Patent Classification[7]: H03K 19/00

(21) International Application Number:
PCT/US2003/024349

(22) International Filing Date: 4 August 2003 (04.08.2003)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
10/214,025 6 August 2002 (06.08.2002) US

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

(72) Inventors: **LAI, Andrew, W.**; 340 Lower Vintners Circle, Fremont, CA 94539 (US). **SIMMONS, Randy, J.**; 1940 Huxley Court, San Jose, CA 95125 (US). **MANSOUR, Teymour, M.**; 1372 Bedford Avenue, Sunnyvale, CA 94087 (US). **TONG, Vincent, L.**; 45641 Montclaire Terrace, Fremont, CA 94539 (US). **LINDHOLM, Jeffrey, V.**; 1008 Sparrow Hawk Drive, Longmont, CO 80501 (US). **YOUNG, Jay, T.**; 359 Troon Court, Louisville, CO 80027 (US). **TROXEL, William, R.**; 2328 Eagleview Circle, Longmont, CO 80504 (US). **KRISHNAMURTHY, Sridhar**; 4321 Verdigris Circle, San Jose, CA 95134 (US).

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

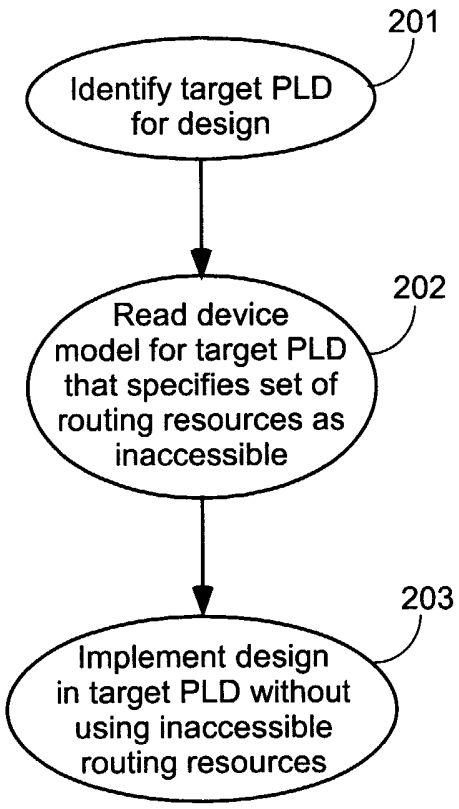
(81) Designated States *(national)*: CA, JP.

(84) Designated States *(regional)*: European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK, 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: METHODS OF RESOURCE OPTIMIZATION IN PROGRAMMABLE LOGIC DEVICES TO REDUCE TEST TIME



(57) Abstract: Methods of optimizing the use of routing resources in programmable logic devices (PLDs) to minimize test time. A set of routing resources is identified that are not used in most designs, and a device model is provided to the user that prevents the use of these resources. Because the routing resources will never be used, they need not be tested by the PLD manufacturer, significantly reducing the test time. For example, each PLD within a PLD family is typically designed using a different number of similar tiles. Thus, smaller PLDs in the family include an unnecessarily large number of routing resources. These excessive routing resources can be disabled during implementation of a design. In another example, each tile along the edges of an array includes routing resources designed primarily to provide access to tiles that are not present. These redundant routing resources can be disabled during implementation of a design.

WO 2004/013967 A2

METHODS OF RESOURCE OPTIMIZATION IN PROGRAMMABLE LOGIC DEVICES TO REDUCE TEST TIME

FIELD OF THE INVENTION

The invention relates to Programmable Logic Devices (PLDs). More particularly, the invention relates to methods of reducing test time in PLDs by eliminating the necessity for testing redundant resources.

BACKGROUND OF THE INVENTION

Programmable logic devices (PLDs) are a well-known type of digital integrated circuit that can be programmed to perform specified logic functions. One type of PLD, the field programmable gate array (FPGA), typically includes an array of configurable logic elements (CLEs) accessed from off-chip via programmable input/output blocks (IOBs). The CLEs and IOBs are interconnected by a programmable interconnect structure. Some FPGAs also include additional logic blocks with special purposes (e.g., DLLs, RAM, and so forth).

The interconnect structure, CLEs, IOBs, and other logic blocks are typically programmed by loading a stream of configuration data (bitstream) into internal configuration memory cells that define how the CLEs, IOBs, and interconnect structure are configured. The configuration data can 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.

The interconnect structure typically includes a large number of routing resources such as interconnect lines (e.g., metal wires) running generally horizontally and vertically between the various logic blocks, and programmable interconnect points (PIPs) that selectively couple the interconnect lines to each other and to input and output pins of the logic blocks. Interconnect lines within the CLE array can span, for example, one CLE, two CLEs, six CLEs, half of the chip, and so forth. By

enabling selected PIPs, two signals in two logic blocks can be interconnected via one or more of the interconnect lines.

In order to test a PLD, typically every logic block, every interconnect line, and every PIP must be tested. This exhaustive testing process is required because, being a programmable device, the manufacturer cannot predict which of the literally millions of resources will be used by a customer's design. For some PLDs, the process of testing every resource on the device can be so time-consuming that testing becomes the largest expense in producing the PLD-- larger than the cost of fabricating the die. Therefore, it is desirable to reduce PLD testing time as much as possible.

PLD providers typically offer a "family" of PLD products, that is, a set of PLDs that are closely related but include different numbers of similar logic blocks. A rectangular area (i.e., a "tile") is designed and laid out that includes, for example, a CLE and the associated routing resources. The CLE can include, for example, one or more lookup tables and one or more memory elements paired with the lookup tables. The routing resources can include, for example, interconnect lines that connect by abutment with interconnect lines in adjacent tiles. The routing resources also include PIPs that allow signals access to and from the interconnect lines and CLEs.

Typically, a single tile is designed (e.g., including a CLE and the associated routing) and this single tile is used in the CLE arrays of all members of a PLD family. For example, the smallest member of a PLD family can include an 8x8 array of tiles, while the largest member of the same family can include a 128x120 array of the same tiles. Each of these tiles is similar, i.e., has the same number of resources located in about the same positions within the tile, although minor variations can occur in edge tiles, for example, to improve the layout area of the PLD or for other reasons. The number of interconnect resources provided within the tile is typically sized to accommodate

the number of signals required by the largest array of tiles.

It is well known that the required number of interconnect resources grows at a rate larger than the number of tiles. Therefore, for example, if N interconnect lines per tile are sufficient to route a typical design in an 8x8 array, many more than N interconnect lines per tile are required to route a typical design in a 128x120 array. Similarly, for example, if M interconnect lines per tile are sufficient to route 90% of user designs in the 128x120 array, probably all user designs will route quickly within the smaller array having M interconnect lines per tile. Large numbers of interconnect lines and PIPs typically go unused in the smaller arrays.

A similar condition applies at the edges of each array, and particularly at the corners, regardless of the size of the array. Because of the uniformity of the tiles, the same number of interconnect lines per tile are available at the outer edges of the array as in the center of the array. However, signals are typically much more congested in the center of the array. Therefore, large numbers of interconnect lines and PIPs typically go unused in the outer tiles, and especially in the edge tiles.

Further, certain types of interconnect lines and PIPs are always unused in edge tiles, such as those used to connect directly to an adjacent tile that is not present because the edge of the array has been reached. Other types of interconnect lines and PIPs are very rarely used at the edges of an array, such as those generally used to route signals long distances in the direction of the edge.

Using similar tiles (and especially using uniform tiles) in the design of PLDs reduces the PLD design time, increases product yield due to the uniformity of the circuit structure, simplifies the implementation software for the PLD, and allows for many consistent timing specifications across the PLD family. However, this uniformity of design also leads to uniformity of testing.

In other words, each tile in the smaller members of the PLD family is typically tested just as exhaustively as tiles in larger members of the same PLD family, and each tile at the edges of the tile array is equally exhaustively tested.

Testing time for PLDs is an important issue not only in larger PLDs, but also in smaller PLDs, which typically include excessive routing resources and in which a larger percentage of the tiles are edge tiles and corner tiles. Therefore, it is desirable to reduce testing time for all PLDs in which arrays of similar tiles are used.

SUMMARY OF THE INVENTION

The invention provides methods of optimizing the use of routing resources in programmable logic devices (PLDs) to minimize test time. A set of routing resources is identified that are not used in most designs, and a device model is provided to the user that prevents the use of these resources. Because the routing resources will never be used, they need not be tested by the PLD manufacturer. The end result is a significantly faster test time, which allows the PLD manufacturer to reduce the price of the PLDs. The methods are also useful for eliminating routing resources that are difficult to test, in order to improve test coverage.

For example, each PLD within a PLD family is typically designed using a similar tile, with the various PLDs including different numbers of the similar tiles. Thus, smaller PLDs in the family include an unnecessarily large number of routing resources. Sometimes, even the largest member of a PLD family proves to include excessive routing resources. According to one aspect of the invention, these excessive routing resources are disabled during implementation of a user design.

In another example, each tile includes routing resources designed to access tiles (particularly adjacent tiles) in each of the four compass directions. Some PLDs also provide "direct connect" routing resources to

diagonally adjacent or nearby tiles. In either case, tiles along the edges of a tile array include routing resources designed primarily to provide access to tiles that are not present. According to one aspect of the invention, these redundant routing resources are disabled during implementation of a user design.

According to some embodiments, a method of reducing test time includes identifying a first set of routing resources that are not to be used in a target PLD, supplying a device model to users that specifies the first set of routing resources as inaccessible, and generating a set of test patterns for the target PLD that do not test the first set of routing resources. Some embodiments include the further step of testing the target PLD using the set of test patterns.

According to other embodiments, a method of implementing a design in a PLD includes identifying a target PLD for the design, reading a device model for the target PLD, where the device model specifies a first set of routing resources that are not to be used in the target PLD, and implementing the design in the target PLD using the device model to ensure that the first set of routing resources are not used.

According to yet other embodiments, a method of enabling the implementation of user designs applies to first and second PLDs within a PLD family that includes variously-sized arrays of similar tiles. The method includes identifying first and second sets of routing resources that are not to be used in the first and second PLDs, storing in first and second device models data specifying the first and second sets of routing resources as inaccessible, and supplying the first and second device models to a user. In one embodiment, the first PLD is smaller than the second PLD and the first set of routing resources is larger than the second set. Thus, excess routing resources included in the tile but not needed in the smaller PLD are inaccessible to the user and need not be tested.

According to some embodiments, a method of enabling the implementation of user designs includes identifying, for each tile located on an edge of the array, a set of routing resources that are designed to provide access to tiles not present for the instant tile, storing in a device model data specifying the identified sets of routing resources as inaccessible, and supplying the device model to a user.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example, and not by way of limitation, in the following figures.

Fig. 1 illustrates a method of reducing test time for a PLD according to one embodiment of the invention.

Fig. 2 illustrates a method of implementing a design in a PLD according to one embodiment of the invention.

Fig. 3 illustrates a method of enabling the implementation of user designs in a family of PLDs according to one embodiment of the invention.

Fig. 4 illustrates a method of enabling the implementation of user design in a PLD according to one embodiment of the invention.

DETAILED DESCRIPTION OF THE DRAWINGS

The present invention provides a variety of methods applicable to the design and testing of programmable logic devices (PLDs). The present invention has been found to be particularly applicable and beneficial for field programmable gate arrays (FPGAs). However, the present invention is not so limited.

Further, the methods of the present invention can be performed in either hardware, software, or any combination thereof, as those terms are currently known in the art. In particular, the present methods can be carried out by software, firmware, or microcode operating on a computer or computers of any type. Additionally, software embodying the present invention may comprise computer instructions in

any form (e.g., source code, object code, interpreted code, etc.) stored in any computer-readable medium (e.g., ROM, RAM, magnetic media, punched tape or card, compact disc (CD) in any form, DVD, etc.). Further, such software can also be in the form of a computer data signal embodied in a carrier wave, such as that found within the well-known Web pages transferred among computers connected to the Internet. Accordingly, the present invention is not limited to any particular platform.

Fig. 1 shows the steps of a method for reducing test time in PLDs. In step 101, a first set of routing resources is identified in the target PLD that will not be used in implementing user designs. The first set of routing resources can include, for example, programmable interconnect points (PIPs) and/or interconnect lines that are rarely or never used in user designs. These routing resources can sometimes be identified by logical inference, as in the case of PIPs in edge tiles that are primarily designed to interface with adjacent tiles not present. (An edge tile can be a tile along the outer edge of the complete tile array, or a tile adjacent to a different logic block inserted into the array, such as a block RAM.) When logical inference is used to identify redundant routing resources, it is desirable (but not mandatory) to test the logical inference by ensuring that most or all user designs still route without these identified resources.

Alternatively or additionally, an analysis can be made of a suite of previously implemented user designs that identifies resources rarely or never used by the implementation software.

The first set of routing resources can also (or alternatively) include excess resources that are not needed in a particular PLD. For example, a small PLD that uses a tile designed for a large PLD will include an unnecessarily large number of routing resources, as described in the Background section, above. For the small PLD, some of these routing resources can be included in the first set of

routing resources, thereby identifying them as resources that will not be used. For the largest member of the PLD family, the tile is presumably appropriately sized to include the correct number of routing resources. Therefore, for the largest member of the PLD family, these routing resources are preferably not included in the first set of routing resources. When excess routing resources are included in the first set of routing resources, it is desirable (but not mandatory) to test a large number of designs and ensure that most or all of the designs still route, given the reduced number of available routing resources.

In step 102, a device model (e.g., a computer file including data regarding the programmable resources in the target PLD) is supplied to one or more users, where the device model specifies as inaccessible the first set of routing resources. Because they are marked as inaccessible in the device model, the implementation software treats all routing resources in the first set of resources as if they do not exist.

In step 103, a set of test patterns is generated that is smaller than a full set of test patterns required to test every routing resource in the PLD. The number or size of test patterns is reduced because the resources specified as inaccessible in the device model are not tested.

In step 104, which can be performed, for example, by the PLD manufacturer or at a separate testing facility, the PLD is tested using the set of test patterns provided in step 103.

Fig. 2 shows the steps of a method of implementing a design in a PLD. In step 201, a target PLD is identified in which the design will be implemented. For example, based on an estimate of resources required to implement the design, a PLD can be selected from a family of available PLDs that is sufficiently large to hold the design but not larger than necessary. In step 202, a device model for the target PLD is read. The device model specifies, possibly among other data, a set of routing resources that is marked

as inaccessible, i.e., not to be used in implementing the design. The routing resources specified as inaccessible can be selected, for example, using any of the methods described above with relation to Fig. 1.

In step 203, the design is implemented in the target PLD using the device model. The routing resources specified as inaccessible in the device model are not used to implement the design.

Fig. 3 shows the steps of a method of enabling the implementation of user designs in a family of PLDs. The family of PLDs includes at least two PLDs, each including a differently-sized array of a similar tile. In step 301, a first set of routing resources is identified in the first PLD that will not be used when implementing user designs in the first PLD. The first set of routing resources can be identified, for example, using any of the methods described above with relation to Fig. 1. In step 302, data identifying the first set of routing resources is stored in a first device model for the first PLD.

In step 303, a second set of routing resources is identified in the second PLD that will not be used when implementing user designs in the second PLD. The second set of routing resources can be identified, for example, using any of the methods described above with relation to Fig. 1. The first and second sets of routing resources are not identical, because the first and second PLDs include differently-sized arrays of tiles. In step 304, data identifying the second set of routing resources is stored in a second device model for the second PLD.

Clearly, steps 301-304 can be performed in an order other than that shown. For example, step 303 (or steps 303 and 304) can be performed prior to step 302.

In step 305, the first and second device models are supplied to one or more users. For example, the first and second device models can be included with a software package that implements a user design, e.g., place and route software that supports the family of PLDs. As another example, one or more of the device models can be

supplied to a user as a software enhancement, e.g., as a file transferred over the Internet.

In step 306, a user implements a design in one or both of the first and second PLDs, where the implementation software uses the first and/or second device models to ensure that the routing resources specified as inaccessible are not used.

Fig. 4 shows the steps of a method of enabling the implementation of user design in a PLD. The PLD includes an array of similar tiles. In step 401, for each tile around the edge of the array, routing resources are identified that are primarily intended to provide access to tiles that are not present, i.e., tiles that would have been beyond the edge of the array.

In step 402, data is stored in a device model that specifies as inaccessible the routing resources identified in step 401. In step 403, the device model is supplied to a user. For example, the device model can be included with a software package, or supplied as a single file. In step 404, the user implements the design without using the routing resources specified as inaccessible in the device model.

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. Accordingly, 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 implementing a design in a programmable logic device (PLD), the PLD including an array of similar tiles each including similar routing resources, the method comprising:

identifying a target PLD in which the design will be implemented;

reading a device model for the target PLD, the device model specifying as inaccessible a set of routing resources present in the PLD that are not to be used in implementing the design; and

implementing the design in the target PLD using the device model, wherein the routing resources specified as inaccessible in the device model are not used.

2. The method of Claim 1, wherein the device model specifies routing resources within the target PLD that are infrequently used in user designs.

3. The method of Claim 1, wherein the device model specifies routing resources designed to provide access to tiles not present along edges of the array.

4. The method of Claim 1, wherein:

the target PLD is a member of a family of PLDs including differently-sized arrays of the tiles;

the target PLD is not the largest member of the family of PLDs; and

the device model specifies routing resources not specified in a corresponding device model for the largest member of the family of PLDs.

5. The method of Claim 1, wherein the routing resources comprise interconnect lines and programmable interconnect points (PIPs).

6. The method of Claim 1, wherein the target PLD is a field programmable gate array (FPGA).

7. A system for implementing a design in a programmable logic device (PLD), the PLD including an array of similar tiles each including similar routing resources, the system comprising:

means for identifying a target PLD in which the design will be implemented;

means for reading a device model for the target PLD, the device model specifying as inaccessible a set of routing resources present in the PLD that are not to be used in implementing the design; and

means for implementing the design in the target PLD using the device model, wherein the routing resources specified as inaccessible in the device model are not used.

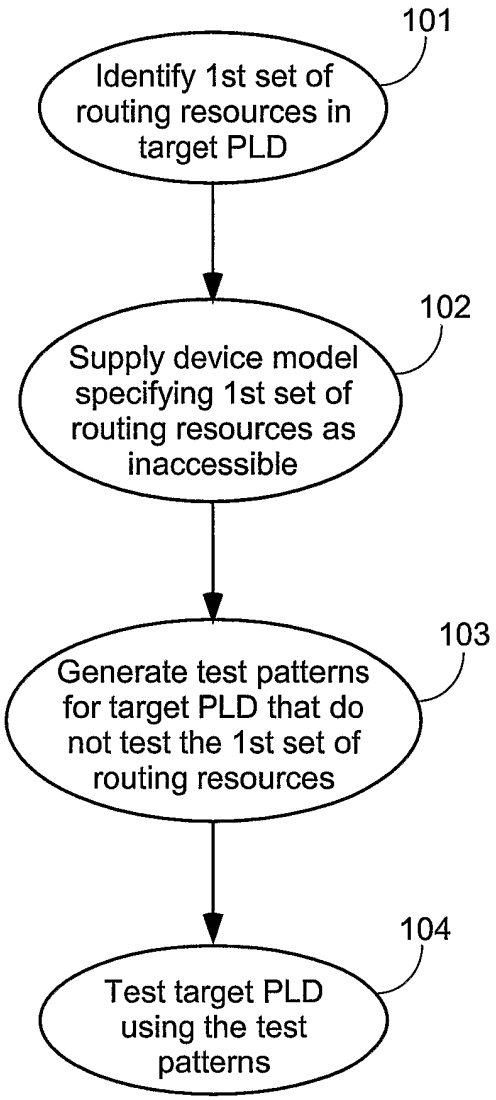

101
Identify 1st set of routing resources in target PLD
102
Supply device model specifying 1st set of routing resources as inaccessible
103
Generate test patterns for target PLD that do not test the 1st set of routing resources
104
Test target PLD using the test patterns

FIG. 1

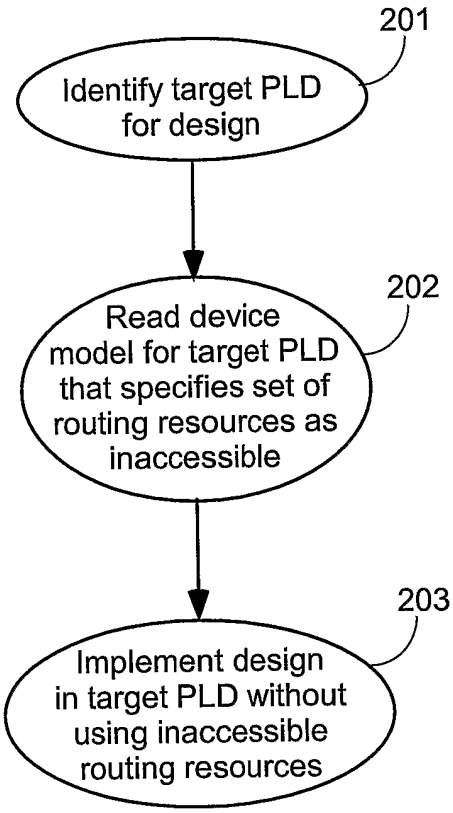

201
Identify target PLD for design
202
Read device model for target PLD that specifies set of routing resources as inaccessible
203
Implement design in target PLD without using inaccessible routing resources

FIG. 2

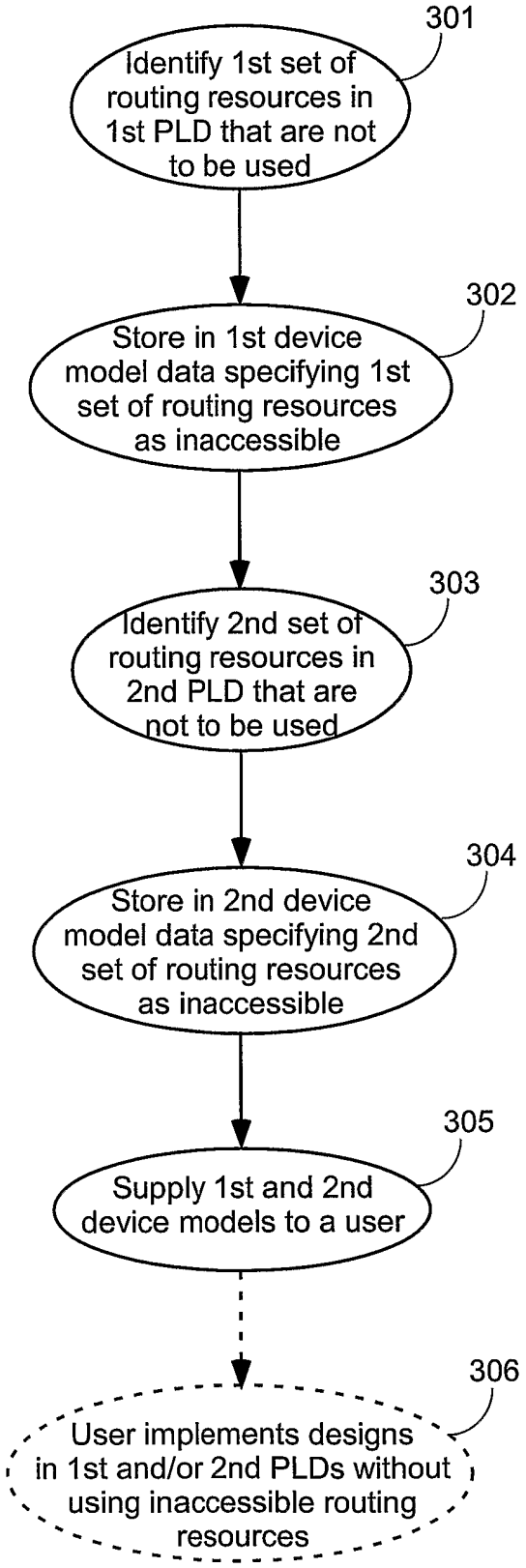

301
Identify 1st set of routing resources in 1st PLD that are not to be used
302
Store in 1st device model data specifying 1st set of routing resources as inaccessible
303
Identify 2nd set of routing resources in 2nd PLD that are not to be used
304
Store in 2nd device model data specifying 2nd set of routing resources as inaccessible
305
Supply 1st and 2nd device models to a user
306
User implements designs in 1st and/or 2nd PLDs without using inaccessible routing resources

FIG. 3

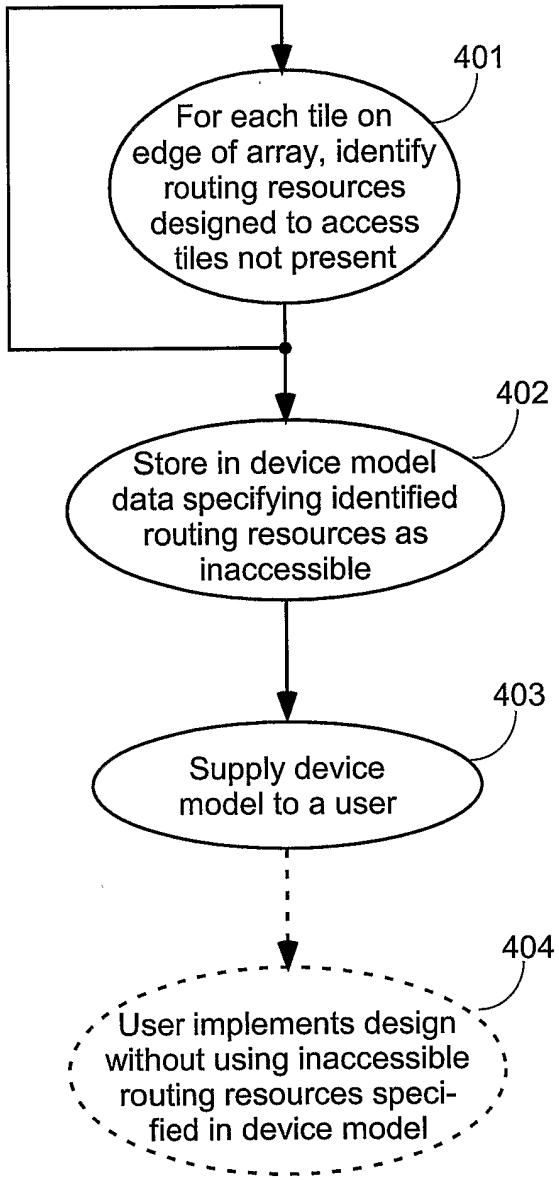

401
For each tile on edge of array, identify routing resources designed to access tiles not present
402
Store in device model data specifying identified routing resources as inaccessible
403
Supply device model to a user
404
User implements design without using inaccessible routing resources specified in device model

FIG. 4