(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 101828120 B
(45)授权公告日 2013.01.30

(21)申请号 200880112144.9

(22)申请日 2008.07.18

(30)优先权数据
11/873,800 2007.10.17 US

(85)PCT申请进入国家阶段日
2010.04.19

(86)PCT申请的申请数据
PCT/US2008/070563 2008.07.18

(87)PCT申请的公布数据
WO2009/051871 EN 2009.04.23

(73)专利权人 新思科技有限公司
地址 美国加利福尼亚州

(72)发明人 约格什·潘迪 维贾伊·桑卡尔
马尼什·简因

(74)专利代理机构 北京康信知识产权代理有限责任公司 11240
代理人 余刚 吴孟秋

(51)Int.Cl.
G01R 31/28(2006.01)

(56)对比文件
US 2005229123 A1,2005.10.13, 全文.
US 6895372 B1,2005.05.17, 全文.
JP 11015860 A,1999.01.22, 第【0011】、【0030】-【0033】段.
CN 1347503 A,2002.05.01, 全文.
CN 1989417 A,2007.06.27, 全文.
JP 5073638 A,1993.03.26, 说明书第【0001】、【0011】-【0026】段,图 1,4.
US 2005149805 A1,2005.07.07, 全文.

审查员 刘俊杰

权利要求书 2 页 说明书 12 页 附图 9 页

(54)发明名称

提高 IC 设计仿真速度并测试扫描电路

(57)摘要

一种计算机被编程为配置计算机程序,该计算机程序用于仿真集成电路(IC)芯片的操作,以测试其中的扫描电路。该计算机被编程为追踪穿过 IC 芯片的设计中的组合逻辑的路径,该路径从第一扫描单元的输出端口开始并在第二扫描单元的输入端口结束。如果第一扫描单元和第二扫描单元接收共同的扫描使能信号,则计算机生成计算机程序的至少一部分,即,例如当共同的扫描使能信号未激活时,有条件地执行仿真穿过路径的信号传播,并且当共同的扫描使能信号激活时,跳过执行仿真的软件。计算机在存储器中存储计算机程序的一部分,用于计算机程序的其他这种部分。

CN 101828120 B

1. 一种用于配置计算机程序的计算机实施方法，所述计算机程序用于仿真集成电路 IC 芯片的操作，以测试包括在所述 IC 芯片中的扫描单元，所述方法包括 ：

追踪穿过所述 IC 芯片的设计中的组合逻辑的路径，所述路径从所述设计中的第一扫描单元的输出端口开始，并且所述路径在所述设计中的第二扫描单元的输入端口结束 ；

创建第一指令集，以仿真穿过所述路径的信号传播 ；

修改所述第一指令集以创建第二指令集，所述第二指令集要求满足预定条件来执行所述第一指令集 ；

将所述第一指令集和所述第二指令集作为所述计算机程序的一部分存储在存储器中 ；

其中，修改所述第一指令集包括 ：当共同的扫描使能信号被提供给所述第一扫描单元和所述第二扫描单元中的每一个时，修改所述第一指令集。

2. 根据权利要求 1 所述的方法，其中 ：

存储所述第一指令集和所述第二指令集包括 ：当所述第一扫描单元和所述第二扫描单元中的每一个接收不同的扫描使能信号时，存储所述第一指令集和所述第二指令集。

3. 根据权利要求 1 所述的方法，其中 ：

存储所述第一指令集和所述第二指令集包括 ：当共同的扫描使能信号被提供给所述第一扫描单元和所述第二扫描单元中的每一个时，存储所述第二指令集。

4. 根据权利要求 3 所述的方法，其中 ：

所述预定条件包括所述共同的扫描使能信号未激活。

5. 根据权利要求 1 所述的方法，进一步包括，在所述追踪之前 ：

根据所述设计确定至少包括所述第一扫描单元和所述第二扫描单元的多个扫描单元。

6. 根据权利要求 5 所述的方法，其中，所述确定包括 ：

对于所述多个扫描单元中的每个扫描单元，识别触发器和在所述触发器之前的复用器。

7. 根据权利要求 6 所述的方法，其中，所述确定进一步包括 ：

基于所述设计，识别用于实现所述触发器和所述复用器的至少一个模块。

8. 根据权利要求 7 所述的方法，其中，所述确定包括 ：

识别由所述模块使用以表示所述复用器的用户定义原语。

9. 一种用于配置计算机程序的设备，所述计算机程序用于仿真集成电路 IC 芯片的操作，以测试包括在所述 IC 芯片中的扫描单元，所述设备包括 ：

存储器，编码有描述所述 IC 芯片的设计 ；

追踪装置，用于追踪穿过所述设计中的组合逻辑的路径，所述路径从所述设计中的第一扫描单元的输出端口开始，并且所述路径在所述设计中的第二扫描单元的输入端口结束 ；

检查装置，用于检查所述第一扫描单元和所述第二扫描单元是否接收到共同的扫描使能信号 ；

生成装置，用于生成所述计算机程序的至少一部分，以在所述共同的扫描使能信号未激活时有条件地穿过所述路径传播信号，并在所述共同的扫描使能信号激活时不穿过所述路径传播所述信号 ；以及

存储装置，用于将所述计算机程序的至少一部分存储在所述存储器中。

10. 根据权利要求 9 所述的设备，进一步包括：

确定装置，用于根据所述设计确定至少包括所述第一扫描单元和所述第二扫描单元的多个扫描单元。

11. 根据权利要求 10 所述的设备，其中，所述确定装置包括：

识别装置，对于所述多个扫描单元中的每个扫描单元，所述识别装置用于识别触发器和在所述触发器之前的复用器。

12. 根据权利要求 10 所述的设备，其中，所述确定装置包括：

用于基于所述设计识别用于实现所述触发器和所述复用器的至少一个模块的装置。

13. 根据权利要求 12 所述的设备，其中，所述确定装置包括：

用于识别由所述至少一个模块使用以表示所述复用器的用户定义原语的装置。

提高 IC 设计仿真速度并测试扫描电路

技术领域

[0001] 本发明的实施例涉及用于对测试图样进行测试的集成电路 (IC) 芯片的仿真，该测试图样由自动测试图样生成 (ATPG) 创建，用于物理 IC 芯片中的扫描电路。

背景技术

[0002] 今天的电子装置包含数百万的单片的电路或“单元”。为了使这种装置的设计和制造自动化，已经开发了电子设计自动化 (EDA) 系统。EDA 系统包括一个或多个被编程的计算机，由芯片设计者使用以设计可以包括一个或多个 IC 芯片的电子装置。EDA 系统通常容纳将被置入 IC 芯片的电路的一个或多个高级动作描述（例如，以如 VHDL、Verilog 等的硬件描述语言 (HDL)），并将这种动作描述转换成抽象的多级网表 (netlist)。网表通常存储在 EDA 系统中的计算机可读介质中，并使用多种公知技术进行处理和验证。EDA 系统使用网表以便最终产生掩模形式的物理装置布局，用于在制造物理 IC 芯片时使用。

[0003] 测试用设计 (DFT) 流程可以（例如）以网表形式设计 IC 芯片（其实现期望的行为，例如数字信号处理 (DSP)），并且用称为“扫描单元”21 ～ 22（图 1B）的特殊单元代替一个或多个触发器 11 ～ 12（图 1A），扫描单元被设计为从 IC 芯片 10（图 1B）的主输入端 31 向一个或多个部分 13 提供测试向量。最初的 IC 芯片设计的部分 13 通常包括连接触发器 11 和 12 的组合逻辑。在刚刚描述的用扫描单元代替触发器期间，部分 13 通常保持不变。这种修改的设计具有两种操作模式，一种是任务模式，其执行 IC 芯片 10 被设计用于的预定功能（例如，DSP），一种是测试模式，其测试 IC 芯片 10 中的电路元件是否被正确地制造。

[0004] 通常，IC 芯片 10 的这种修改设计中的扫描单元 21（图 1B）包括由复用器 21M 驱动的触发器 21F；如果以测试模式运行（在此期间扫描使能信号 SE 是激活的），则复用器 21M 向触发器 21F 的数据输入 (D 输入）引脚提供信号 SI，并且可选地，如果以任务模式运行（在此期间信号 SE 是未激活的），则提供另一信号 MI。在图 1B 中示出了输入至触发器 21F 的信号作为复用器的输出信号 MO。在扫描设计期间，芯片设计者可以识别出扫描单元 21 和 22，由于它们想连接至扫描链，这涉及通过连接扫描单元 21 和 22（例如，单元 22 的输入引脚 SI 连接至单元 21 中的触发器 21F 的输出引脚 Q) 来创建扫描路径 23（见图 1B）。扫描路径 23 对于通过部分 13 的任务路径 13P 而言是可选的，并且复用器 22M 基于其扫描使能信号选择来自这两条路径之一的信号。芯片设计者可以指定共同的扫描使能信号 SE 或指定不同的扫描使能信号，以运行复用器 21M 和 22M。

[0005] 开发 IC 芯片设计中的附加步骤包括生成应用于 IC 芯片 10 的测试图样。用 ATGP 软件编程的计算机可以以网表的形式分析一个或多个 IC 设计的表述，并且可以自动生成测试图样。这种测试图样（也称为测试向量）由硬件装置（称为“测试器”）应用于物理 IC 芯片中的扫描单元，以测试（例如）是否正确地制造了电路的特定选择部分。

[0006] 更具体地，测试器（未示出）通过在转移 (shifting) 操作（也称为“加载操作”）期间，从 IC 芯片 10 的主输入端 31 连续向一个或多个扫描单元 21（也称为“输入扫描单元”）中加载一个或多个测试图样，同时激活扫描使能信号，来测试 IC 芯片 10。IC 芯片 10

的主输入端 31 和主输出端 32 是可以从芯片 10 的外部接入（例如至任意测试器）的外部引脚。在这种转移操作后，测试器可以去激活 (deactivate) 扫描使能信号，并用应用于部分 13（处于“测试操作”）的测试图样运行 IC 芯片 10 一个时钟周期。

[0007] 测试操作后有处于另一转移操作（也称为“卸载操作”）的激活的（多个）扫描使能信号的一个或多个周期，其中，由输出扫描单元 22 锁存的测试操作的结果被转移到 IC 芯片 10 的主输出端 32。本发明人注意到，现有技术中，在加载操作和卸载操作期间，源扫描单元 21 和宿扫描单元 22 之间的电路的选择部分 13 继续正常运行，即，这些部分中的所有门均会被评估。

[0008] 在制造物理 IC 芯片之前，测试图样通常应用于 IC 芯片的门级计算机模型。例如，通过将以 HDL 表述的 IC 设计转换成在计算机中执行（编译之后）或解释（不编译）的软件源代码（例如，以编程语言 C 或 C++) 来获得计算机指令 40（图 1C)。如在图 1C 中所示，计算机指令 40 包括三种函数，第一函数“Evaluate_Flipflop”仿真在扫描单元 21（图 1B) 中的触发器 21F 的输出引脚 Q 处的信号，第二函数“propagate”仿真这个信号穿过组合逻辑 13 经由任务路径 13P 到扫描单元 22 的 MI 输入引脚的传播。最后，第三函数“Evaluate_Multiplexer”仿真由复用器 22M 向触发器 22F 的输入引脚 D 提供的信号。编译后的计算机指令 40 的执行比解释 (interpreted) 的执行更快，因此，通常将这种软件源代码编译成编译代码。

[0009] 在上述段落中描述的函数“propagate”可以根据配置仿真或不仿真扫描路径 23 上的信号传播。例如，触发器通常具有称为 Q- 引脚（除 Q 引脚之外）的另一输出引脚，并且在一些配置中，Q- 引脚用在扫描链中，在这种情况下，函数“propagate”不进行任何额外的仿真。在其他配置中，不使用 Q- 引脚，相反地，在单元实例中产生路径分叉。在这种配置中，可以仿真 Q- 引脚，以驱动扫描路径 23 上的信号。

[0010] 例如，在 1994 年 3 月由密歇根州大学出版的 Michael A. Riepe 等人的“Ravel-XL：A Hardware Accelerator for Assigned-DelayCompiled-Code Logic Gate Simulation”中描述了基于编译代码的仿真，其全部内容通过引用结合于此作为背景技术。此外，在 2001 年 4 月 24 日授予 Ashar 的第 6,223,141 号美国专利中也描述了现有技术的一些编译代码仿真器，该专利的全部内容也通过引用结合于此作为背景技术。Ashar 描述了使用网表转换来加快平准化编译代码仿真。特别地，具有平准化编译代码仿真的同步数字电路的基于时延独立周期的逻辑仿真能极大地提高速度。清除、消除、和要素减少了文字数量。特别地，消除功能去除了由于其存在而增加了文字数量的门的网表，即，将这些门分解为其及时扇出 (immediatefanout)，减小了文字数量。在将门分解为其扇出之前，该功能估计新开端 (onset) 的大小。如果估计的大小大于预设界限，则不执行该分解。大多数的文字量减少是通过消除功能进行的。

[0011] 本发明人相信，编译代码仿真器能够变得过于缓慢。特别地，实现高故障覆盖率 (high fault coverage) 所需的测试图样数量随电路大小增加。此外，深亚微米技术在更多失败机制和更多缺陷类型的可能性方面对现有故障模型提出了挑战。更多故障模型又要求对于同一故障覆盖率和质量级别的更多的测试图样，这增加了仿真测试图样的测试所需的时间。因此，本发明人相信需要进一步提高编译代码仿真的速度。

发明内容

[0012] 本文公开的本发明的实施例提供了一种配置计算机程序的计算机实施方法、设备及计算机可读介质，该计算机程序用于仿真集成电路 (IC) 芯片的操作，以测试其中的扫描电路。

[0013] 本发明的示例性实施例提供了一种用于配置计算机程序的计算机实施方法，该计算机程序用于仿真 IC 芯片的操作，以测试扫描电路。该方法追踪穿过 IC 芯片设计中的组合逻辑的路径，创建第一指令集以仿真穿过该路径的信号传播；修改第一指令集以创建第二指令集，第二指令集要求满足预定条件以执行第一指令集；以及将第一指令集和第二指令集存储在存储器中。

[0014] 本发明的示例性实施例提供了一种用于配置计算机程序的设备，该计算机程序用于仿真 IC 芯片的操作，以测试扫描电路。该设备包括：存储器，编码有描述 IC 芯片的设计；追踪装置，用于追踪穿过设计中的组合逻辑的路径；检查装置，用于检查第一扫描单元和第二扫描单元是否接收到共同的扫描使能信号；生成装置，用于生成计算机程序的至少一部分，以在共同的扫描使能信号未激活时有条件地穿过路径传播信号，并在共同的扫描使能信号激活时不穿过路径传播信号；以及存储装置，用于将计算机程序的一部分存储在存储器中。

[0015] 本发明的示例性实施例提供了一种用于配置计算机程序的计算机可读介质，该计算机程序用于仿真集成电路 (IC) 芯片的操作，以测试扫描电路。该计算机可读介质包括：追踪指令，用于追踪穿过 IC 芯片的设计中的组合逻辑的路径；创建指令，用于创建第一指令集，以仿真穿过该路径的信号传播；修改指令，用于修改第一指令集以获得第二指令集，第二指令集要求满足预定条件以执行第一指令集；以及存储指令，用于将第一指令集和第二指令集作为计算机程序的一部分存储在计算机的存储器中。

附图说明

[0016] 图 1A 和图 1B 示出了插入扫描电路之前和之后的现有技术 IC 芯片设计。

[0017] 图 1C 示出了用于仿真图 1B 的设计以测试扫描电路的现有技术的计算机程序的一部分。

[0018] 图 2A 以流程图的形式示出了根据本发明的实施例的用于配置提高仿真速度的计算机程序的方法。

[0019] 图 2B 示出了根据本发明的实施例生成的、通过执行图 2A 的方法来包括条件语句的计算机程序的一部分。

[0020] 图 3 以流程图的形式示出了在本发明的示例性实施例中执行的、用于实施图 2A 的方法的动作。

[0021] 图 4A ～图 4C 以流程图的形式示出了在本发明的实施例的实施中执行的动作。

[0022] 图 5 在框图中示出了根据本发明的实施例编程的计算机。

[0023] 图 6 示出了根据本发明的实施例的示例性数字专用 IC(ASIC) 设计流程的简化表述。

具体实施方式

[0024] 计算机 150(图 5)根据本发明实施例编程来执行方法 200(图 2A),以创建提高 IC 芯片的仿真速度从而测试扫描电路的计算机程序。特别地,本专利申请的发明人注意到,当扫描使能信号 SE(图 1B)激活时,组合逻辑 13 的输出端的复用器 22M 选择经由扫描路径 23 传输的信号。相应地,发明人设想(当扫描使能信号激活时)这种复用器的操作使得没有必要仿真经由任务路径 13P 穿过部分 13 的信号传播。基于这个设想,如同在下一段中所描述的,发明人构想了方法 200,其通过当扫描使能信号激活时避免路径 13P 上的不必要仿真来提高仿真速度。根据实施例,扫描使能信号可以是激活的高信号(即,当信号为高或者值为“1”时激活),或者备选地是激活的低信号(即,当信号为低或者值为“0”时激活),这对本领域技术人员是显而易见的。

[0025] 如图 2A 中所示,在第一操作 201 中,计算机 150 首先确定 IC 芯片的哪个部件形成扫描单元。在该操作中,对于每个扫描单元,计算机 150 还确定其引脚中的哪一个分别承载 (1) 扫描数据信号,(2) 从任务模式的操作产生的数据信号,(3) 扫描使能信号,以及 (4) 时钟信号。计算机 150 进一步被编程以执行追踪操作 202,例如,以识别穿过组合逻辑 13 的路径 13P。如图 1B 中所示,任务路径 13P 从源扫描单元 21 中的触发器 21F 的输出引脚 Q 开始,并在宿扫描单元 22 的复用器 22M 的输入引脚 MI 结束。计算机 150 还被编程为执行操作 203,以创建软件指令来仿真穿过任务路径 13P 的信号传播。

[0026] 注意,通过图 1C 中的现有技术计算机指令 40 示出了操作 203 的结果。还要注意,计算机 150 可被编程为以对本领域技术人员显而易见的任何方式实施操作 201 ～ 203。因此,计算机 150 执行操作 201、202 和 203 的方式的特定细节对于实践本发明实施例而言不重要。

[0027] 计算机 150 被进一步编程为检查一个或多个条件(在操作 204 中),并且如果条件满足了,则计算机 150 执行操作 205(该操作在条件不满足时跳过)。在操作 204 中使用的条件 997(图 5)是预定的,并存储在计算机 150 的存储器中。操作 204 的特定条件用于确保路径 13P 的非仿真不会改变对自动测试图样生成 (ATGP) 创建的用于扫描电路的一个或多个测试图样进行测试的结果。如果条件满足了,则确定路径 13 是“可优化的”,从而将其作为非仿真的候选。

[0028] 例如,本发明的一些实施例支持使用多个扫描使能信号。相应地,这种实施例检查操作 204 中的如下预定条件 :源扫描单元 21 提供给路径 13P 的信号和宿扫描单元 22 从路径 13P 接收的信号是否被同步地使用(或不使用),即,各扫描单元 21 和 22 中的复用器 21M 和 22M 是否由相同的扫描使能信号操作。如果结果为真,则确定路径 13P 是可优化的。在本发明一些实施例的操作 204 中检查的另一个这种预定条件是,路径 13P 是否包含任意时序元件 (sequential element),并且仅当结果为否时,计算机 150 将路径 13P 标记为“可优化的”。注意,在本发明的一些实施例中,如果路径从扫描单元的数据引脚开始并最终在扫描单元的数据引脚结束,则认为该路径是可优化的。在追踪这种路径时,一个示例性实施例穿过组合元件而不是穿过其他电路元件进行追踪。组合元件的输出状态从其输入端的状态可以立即确定。如果如本文中所描述的路径追踪期间遇到了不同于组合元件的任何电路元件,则示例性实施例将该路径标记为不可优化的。

[0029] 如上面提到的,如果操作 204 发现路径 13P 不是可优化的,则计算机 150 仅仅转到操作 206,在该操作中,操作 203 创建的计算机指令 40(图 1C)作为计算机程序的一部分被

存储到存储器中，用于其他这种部分（例如，由操作 203 重复生成）。注意，指令 40（即，软件）包括语句 42，从而路径 13P 上的信号传播被无条件地仿真。如果操作 204 发现路径 13P 是可优化的，则如接下来所描述的，计算机 150 执行可选操作 205。

[0030] 在操作 205 中，计算机 150 通过向操作 203 中创建的计算机指令 40 中增加待检查的一个或多个条件来修该指令，以获得修改的计算机指令，该修改的计算机指令在没有必要时避免了沿优化路径 13P 的信号传播的仿真。例如，如图 2B 中示出的修改的计算机指令 250 中的语句 252 示出的，检查扫描使能信号，并且如果该信号是激活的，则不执行函数“propagate”，除非路径 13P 不是可优化的。特别地，软件语句 252 检查路径 13P 是否不是可优化的，并且如果不是可优化的，则执行函数“propagate”。另一方面，不管路径 13P 是不是可优化的，如果扫描使能信号是非激活的（例如，当仿真任务模式时），则再次执行函数 propagate。注意，指令 250 包括语句 252，从而有条件地执行信号传播的仿真。更特别地，语句 252 以扫描使能信号的状态以及路径 13P 是否是优化的为条件。

[0031] 相应地，通过检查这种修改的计算机指令 250 中的一个或多个条件，消除了穿过任务路径 13P 的信号传播的仿真，这又加快了加载和卸载操作，也就是从主输入端移入测试图样或从主输出端移出测试图样的操作，这对本公开的技术人员是显而易见的。因此，通过如图 2B 中示出的修改的计算机指令 250，加快了在对 IC 设计中的扫描电路进行测试期间的 IC 设计的仿真。因此，在操作 205 之后，计算机 150 执行操作 206，其中，修改的计算机指令 250 存储在存储器中作为用于其他这种部分的计算机程序部分（即，软件）。在操作 206 之后，计算机 150 转到操作 207，并检查是否已经追踪了从 IC 芯片设计（例如，门级网表的形式，见图 5）中的所有扫描单元开始的所有路径。如果没有，则计算机 150 返回到操作 202（如上所述）。如果操作 207 中发现已经追踪了所有路径，则计算机 150 已经完成了该方法，并因此退出（见操作 208）。

[0032] 如在图 1C 中所示，从操作 203 得到的计算机指令已被无条件地执行（与扫描使能信号有关）。根据本发明，如在语句 252（图 2B）中所示，操作 205（图 2A）对这些计算机指令进行修改，以使其可以有条件地执行。虽然语句 252 中示出了条件的特定实例，但在其他实施例中可以检查其他条件，这对了解了本公开的技术人员是显而易见的。

[0033] 在本发明的一些实施例中，基于图 2A 的方法 200 中的操作 201，计算机 150 实施图 3 中示出的类型的处理。特别地，在动作 301 中，计算机 150 识别 IC 芯片 10 设计中的一个或多个用户定义原语 (UDP) 作为触发器。特定 UDP 根据多种因素来使用，例如，由制造厂 (fabrication facility) 提供的单元技术库。接下来，在动作 302 中，计算机 150 识别设计中的附加 UDP 作为复用器。注意，可以以对了解了本公开的技术人员是显而易见的任意方式实施动作 301 和 302。

[0034] 之后，在动作 303 中，计算机 150 从 IC 芯片设计的数据模型中获得对在动作 301 中识别的触发器进行实现 (instantiate) 的所有模块的列表。接下来，在动作 304 中，计算机 150 从数据模型获得在动作 303 中识别的每个模块（当单独处理时，在下文中称为“当前模块”）的所有端口的列表。在动作 304 中，计算机 150 还获得至数据模型中的每个触发器的输入引脚的所有连接。然后，在动作 305 中，计算机 150 从数据模型获得将数据信号驱动到各触发器的所有驱动器的列表。在动作 306 中，计算机 150 检查在动作 305 中获得的列表中的任意驱动器，是否已在动作 302 中被识别为复用器。如果是，则计算机 150 转到动作

307，以进一步处理复用器（称为“当前”复用器），或者转到动作 310。在动作 310 中，计算机 150 将至触发器的数据引脚 D 的路径标记为不可优化的，并前进至动作 311。

[0035] 在动作 307 中，计算机 150 识别当前复用器的哪个引脚接收扫描数据（即，识别 SI 引脚），以及哪个引脚接收任务数据（即，识别 MI 引脚）。接下来，在动作 308 中，计算机 150 将来自当前复用器的这两个输入引脚（即，SI 引脚和 MI 引脚）的信号向回追踪到当前模块的输入端口。然后，在动作 309 中，计算机 150 将来自当前触发器的 Q 引脚的信号向前追踪到当前模块的输出端口。接下来，计算机 150 转到动作 311，其中，例如，如果在扫描单元间存在未被访问且被标记为可优化或不可优化的路径，则重复上述的一个或多个动作。如果不存在未访问的路径，则计算机 150 在动作 312 中退出该方法。

[0036] 根据本发明的一些示例性实施例执行如下描述的图 4A 中示出的动作。特别地，一些实施例进入执行动作 401 ～ 404，其中，动作 401 进行“for”循环，其中，计算机 150 分别选择表示 IC 设计的“网表”中的每个模块“m”。在动作 402 中，计算机 150 检查在模块“m”中是否存在扫描单元。如果答案为“是”，则计算机 150 转到动作 403，并存储关于扫描单元的信息，诸如其标识和其中的诸如复用器和触发器的部件。在动作 403 之后，计算机 150 转到动作 404。如果动作 402 中的答案为否，则计算机 150 也转到动作 404。动作 404 通过检查是否已经访问了网表中的所有模块来执行动作 401 的循环终止，在该情况下，计算机 150 转到操作 405，而如果否，则返回到动作 401。注意，虽然以上参照图 3 进行了描述，一些实施例基于 UDP 的识别，但是根据实施例，对扫描单元（及其一个或多个部件，诸如复用器和触发器）进行识别的特定方式不同。

[0037] 在操作 405 中，计算机 150 检查每对扫描单元实例（例如，在动作 403 中识别的），以确定一对中的两个实例是否由相同的扫描使能信号驱动，如果是，则这种对的标识存储在数据结构中（例如，根据实施例，可以使用二维表）。如接下来描述的，在操作 405 之后，计算机 150 转到动作 406。

[0038] 动作 406 进行另一“for”循环，其中，计算机 150 分别选择在动作 403 中识别的每个扫描单元实例，并转到动作 407。在动作 407 中，计算机 150 例如通过追踪扇出来检查来自当前扫描单元实例的所有路径是否都是可优化的。如果答案为“是”，则计算机 150 转到动作 408，并将所有这种路径标记为可优化的。在动作 408 之后，计算机 150 转到动作 409。如果动作 407 中的答案为否，则计算机 150 也转到动作 409。动作 409 通过检查是否已经访问了在动作 403 中识别的所有扫描单元实例来进行动作 406 的循环终止，如果是，则转到操作 410，否则返回到动作 406。

[0039] 在操作 410 中，计算机 150 生成软件指令，以仿真通过组合逻辑的信号传播，该组合逻辑包括条件（具有图 2B 的语句 252 中示出的类型）或者是无条件的。如上面提到的，在软件指令中使用的条件基于扫描使能信号。此外，所生成的软件指令是否包含这种条件取决于路径的可优化性。如果路径是可优化的，则软件指令是有条件的。如果路径是不可优化的，则软件指令是无条件的。

[0040] 图 4A 的操作 405 可以以对了解了本公开的技术人员显而易见的任意方式来执行，操作 405 的详细实施并不是本发明的重要方面。尽管如此，为了说明的目的，应注意，一些实施例执行图 4B 中示出的动作 411 ～ 418，以实施操作 405。特别地，在动作 411 中，计算机 150 通过分别选择在动作 403（图 4A）中识别的每个扫描单元实例来实施“for”循环，并转

到动作 412。在动作 412 中，计算机 150 向回追踪以识别扫描使能信号的根网 (root net)，并保存所识别的当前单元实例的根网。然后，计算机 150 转到动作 413，其中，其检查是否已经访问了所有单元实例，并且如果否，则返回到动作 411。如果已经访问了所有单元实例，则计算机 150 转到接下来描述的动作 414。

[0041] 在动作 414 中，计算机 150 通过分别选择一对扫描单元实例来实施另一“for”循环，并转到动作 415。在动作 415 中，计算机 150 检查当前选择的对中的每个扫描单元实例的扫描使能信号的根网是否一致。如果动作 415 中的答案为“是”，则计算机 150 转到动作 416，否则，转到动作 417。在动作 416 和 417 中，计算机 150 存储为真或假的标志，以分别指示扫描使能信号一致或不一致。在动作 416 和 417 之后，计算机 150 转到动作 418，该动作通过检查是否已经访问了所有对的扫描单元实例来执行动作 414 的循环终止，如果否，则返回到动作 414。

[0042] 图 4A 的动作 407 还可以以对了解了本公开的技术人员显而易见的任意方式来执行，并且操作 407 的详细实施不是本发明的重要方面。尽管如此，为了说明的目的，应注意，一些实施例执行图 4C 中示出的动作 421 ～ 427 来执行动作 407。特别地，在动作 421 中，计算机 150 通过分别选择其扇出将被追踪的扫描单元的 Q 引脚的每个扇出 f 来实施“for”循环。接下来，在动作 422 中，计算机 150 检查该扇出 f 是否是单向的简单组合元件，诸如 AND 门或 OR 门或反向器。如果动作 422 中的答案为“否”，则计算机 150 转到动作 424，并检查扇出 f 是否是推断的扫描单元实例，如果不是，则返回“假”，表示该路径不是可优化的。如果动作 424 中的答案为“是”，则计算机 150 转到动作 425，检查扇出 f 和扫描单元是否具有相同的扫描使能信号，如果否，则再次返回“假”，表示该路径不是可优化的。如果动作 425 中的答案为“是”，则计算机 150 转到动作 426，以检查扇出 f 是否与扫描单元 dataNet 相同，如果否，则再次返回“假”，即，路径是不可优化的。如果动作 426 中的答案为“是”，则计算机 150 返回“真”，表示该路径是可优化的。

[0043] 在动作 422 中，如果答案为“是”，则计算机 150 转到动作 423，并进行递归调用以返回到动作 422，但其具有新“f”，该新“f”是通过其输入了动作 423 的旧“f”的扇出。当在动作 423 中无法到达其他扇出时，例如，如果到达了主输出端，则计算机 150 转到动作 427，通过检查是否已经访问了所有对的扫描单元来进行动作 421 的循环终止，而如果否，则返回到动作 421。如果已经访问了所有对的扫描单元，则计算机 150 从该方法返回，即，完成了动作 407(图 4A)。

[0044] 应注意，在数字 ASIC 设计流程 (在图 6 中以简单地示例性表示而示出) 中，可以使用执行方法 200 以如上所述 (例如，参照图 2A) 实现仿真速度提高的任何适当编程的计算机 (下文中的“编译代码仿真器”)。在较高水平上，设计芯片的过程以产品构思 (900) 开始，并在 EDA 软件设计处理 (910) 中实现。当设计完成时，其可以试产 (taped-out)(事件 940)。在试产之后，进行制造处理 (950) 以及包装和装配处理 (960)，并最终产生完成的芯片 (结果 990)。

[0045] EDA 软件设计处理 (910) 实际上有多个阶段 912 ～ 930 组成，为了简化，以线性方式示出。在实际 ASIC 设计处理中，特定设计可能必须通过步骤返回，直到通过特定测试。类似地，在任何实际设计处理中，这些步骤可以以不同顺序及组合进行。因此，本说明书通过上下文和一般解释的方式来提供，而不是特定的或推荐的用于特殊 ASIC 的设计流程。现在

将提供 EDA 软件设计处理（阶段 910）的组成部分的主要描述。

[0046] 系统设计（阶段 912）：电路设计者描述他们想要实现的功能，他们可以执行“如果 ..... 怎样”(what-if) 计划，以改进功能、检查成本等。在这个阶段可出现硬件 - 软件结构划分。可以在该阶段使用的Synopsys®公司的示例性 EDA 软件产品，包括 Model Architect、Saber、System Studio 和 Design Ware®产品。

[0047] 逻辑设计和功能验证（阶段 914）：在该阶段，写入用于系统模块的 VHDL 或 Verilog 代码，并出于功能的准确性对设计（可以具有混合时钟域）进行检查。可以在该阶段使用的来自Synopsys®公司的示例性 EDA 软件产品，包括 VCS、VERA、Design Ware®、Magellan、Formality、ESP 和 LEDA 产品。

[0048] 测试的综合和设计（阶段 916）：这里，VHDL/Verilog 被翻译成网表。该网表对于目标技术而言可以是最优的。另外，测试的设计和实施允许对完成的芯片进行检查。可以在该阶段使用的来自Synopsys®公司的示例性 EDA 软件产品包括 Design Compiler®、Physical Compiler、Test Compiler、Power Compiler、FPGA Compiler、Tetramax 和 Design Ware®产品。

[0049] 设计计划（阶段 918）：这里，构造并分析芯片的总体平面规划，用于定时和顶级定线。可以在该阶段使用的来自Synopsys®公司的示例性 EDA 软件产品包括 Jupiter 和 Floorplan Compiler 产品。

[0050] 网表验证（阶段 920）：在该阶段，为了符合时间约束并与 VHDL/Verilog 源代码一致，检查网表。可以在该阶段使用的来自Synopsys®公司的示例性 EDA 软件产品，包括 VCS、VERA、Formality 以及 Prime Time 产品。

[0051] 注意，如图 6 所示，在该阶段 920 期间可以使用编译代码仿真器 999（具有上述的执行图 2A 的方法的类型）。如果显示的结果不满足，则芯片设计者可以返回到阶段 916 以对如图 5 所示的 IC 设计进行修改。

[0052] 物理实施（阶段 922）：在该阶段进行放置（定位电路元件，诸如上述的连续单元和组合单元）和定线（连接电路元件，诸如上述的连续单元和组合单元）。可以在该阶段使用的来自Synopsys®公司的示例性 EDA 软件产品包括 Astro 产品。虽然在该阶段可以认为其电路和部分（例如矩形）存在于现实世界，但是应当理解，在该阶段，在计算机 150 中仅存在布局。在下述的阶段之后生成现实世界中的实际电路。

[0053] 分析和提取（阶段 924）：在该步骤，在晶体管级验证电路功能，这又允许对“如果 ... 怎样”进行改进。可以在该阶段使用的来自Synopsys®公司的示例性 EDA 软件产品包括 Star RC/XT、Raphael 以及 Aurora 产品。

[0054] 物理验证（阶段 926）：在该阶段，执行各种检查功能，以确保制造、电力问题、平版问题以及电路的准确。可以在该阶段使用的来自Synopsys®公司的示例性 EDA 软件产品包括 Hercules 产品。

[0055] 分辨率增强（阶段 928）：这涉及布局的几何操作以改善设计的可制造性。可以在该阶段使用的来自Synopsys®公司的示例性 EDA 软件产品，包括 iN-Phase、Proteus 和 AFGen 产品。

[0056] 掩模数据配置（阶段 930）：这提供了用于掩模制造的“试产”数据，掩模用于平版

用途，以制造完成的芯片。可以在该阶段使用的来自Synopsys®公司的示例性 EDA 软件产品包括 CATS(R) 系列产品。在该阶段之后，在晶片制造厂（也称为“工厂”）中创建现实世界的实际电路。

[0057] 用于实施在这个详细的描述中（例如，图 2A、图 3、图 4A ～图 4C 和 / 或下面的子部分 A) 的一个或多个动作的数据结构和软件代码可以编码到计算机可读介质中，计算机可读介质可以是能够保存计算机使用的代码和 / 或数据的任意存储介质和 / 或任意传输介质。存储介质包括但不限于 :诸如磁盘驱动、磁带、CD（光盘）、和 DVD（数码影像光碟）的磁和光存储装置。传输介质（具有或不具有在其上调制信号的载波）包括但不限于诸如因特网的有线或无线通信网络。在一个实施例中，传输介质使用包括计算机指令信号的载波，该计算机指令信号用于完成由图 2A 中示出的方法执行的一个或多个步骤。另一实施例使用包括执行图 2A 示出的方法的指令的载波。

[0058] 注意，在一些实施例中使用以实施在本文中所描述的类型的仿真速度增加器的计算机系统，使用一个或多个linux®操作系统工作站（基于IBM®可兼容的 PC) 和 / 或unix®操作系统工作站（例如，SUN Ultrasparc、HP PA-RISC 或等同产品），其每个都包含经由局域网（以太网）互连的 2GHz 的 CPU 和 1GB 的存储器。

[0059] 在下文中，这个详细描述部分的位于权利要求之前的子部分 A 是这个详细描述的不可分割的一部分，其全部内容通过引用结合于此。子部分 A 包括用于实施根据本发明的仿真速度增加器的一个示例性实施例的伪码和相关信息，例如，用于通过使用可从Synopsys®公司购买的称为“VCS”的软件产品来实施在图 4A ～图 4C 中示出的动作。

[0060] 本文描述的实施例的多种修改和变形对了解了本公开的技术人员将显而易见。因此，本文描述的实施例的多种修改和变形也涵盖于本发明的范围之内。

[0061] 子部分 A

[0062] /* 用于本发明的示例性实施例的伪码如下 */

[0063] /* 顶级入口 */

[0064] doScanOpt(netlist)

[0065] {

[0066] /*

[0067] * 推断哪个 HDL 模块匹配 Mux-DFF 扫描单元的模板

[0068] * 如果成功推断，则相关 D/SI/SE/Q 网络位于该模块中。

[0069] */

[0070] foreach modules “m” in the ‘netlist’

[0071] {

[0072] If(isScanCell(m,&DataNet,&ScanDataNet,&ScanEnableNet,

[0073] &Qnet) == true)

[0074] {

[0075] scanCellModuleTable.append({m,DataNet,ScanDataNet,

[0076] ScanEnableNetQnet}) ;

[0077] }

[0078] }

```
[0079]          /*在充分扩展的HDL描述中收集扫描单元的实例*/
[0080]          scanCellInstanceTable＝{instances of all scan cell modules in
[0081]  ‘scanCellModuleTable’}；
[0082]          /*
[0083]           *创建SE等价表以答复扫描单元实例对是否
[0084]           *取决于相同的扫描使能(ScanEnable)根信号。
[0085]           */
[0086]           SEEquivTable ＝ createSEEquivTabel(scanCellInstanceTable,ne
[0087]  tlist)；
[0088]           /*识别可优化扫描单元输出(Q)信号并且对他们作标记用于在代码产
生时进行特定处理*/
[0089]          foreach instance ‘fi’in ‘scanCellInstanceTable’
[0090]          {
[0091]              if(allPathsFormQAreOptimizable(fi,netlist))
[0092]               {
[0093]                    markOutputAsOptimized(fi)；
[0094]               }
[0095]          }
[0096]      }
[0097]      /*程序至创建SE根网等价表*/
[0098]      Table createSEEquivTable(cellInstTable,netlist)
[0099]      {
[0100]      foreach instance ‘if’in ‘cellInstTable’
[0101]      {
[0102]      fi.rootSENet ＝ traceBackAndFindRootNet(fi.ScanEnableNet)；
[0103]      }
[0104]      foreach pair<fi1,fi2>
[0105]      {
[0106]         if(fi1.rootSENet ！ ＝ fi2.rootSENet)
[0107]             SEEquivTable[<fi 1,fi2>] ＝ false；
[0108]          else
[0109]             SEEquivTable[<fi 1,fi2>] ＝ true
[0110]      }
[0111]      reture SEEquivTable；
[0112]  }
[0113]  /*程序至检查这个实例是否可具有其优化的传播输出*/
[0114]  ScanCellInstance currentSourceInst；
[0115]  bool traceFanouts(signal,netlist)
[0116]  {
```

```
[0117]        foreach fanout ‘f’of signal
[0118]        {
[0119]           if( ‘f’is a simple combination gate){
[0120]               return traceFanouts(f- > fanOut) ;/* 扇出的递归调用 */
[0121]            }else if( ‘f’is an inferred scanCellInstance){
[0122]               if(SQEquivTabel[<currentSourceInst,f>] == false)
[0123]                   return false ;
[0124]              else if(signal == f.DataNet)
[0125]                  return true ;
[0126]              else
[0127]                 return false ;
[0128]        }else{
[0129]           return false ;
[0130]       }
[0131]   }
[0132]        }
[0133]   bool allPathsFormQAreOptimizable(SourceScanCellInstance,netlist)
[0134]        {
[0135]            currentSourceInst = SourceScanCellInstance ;
[0136]            /* 向前追踪 scanCellInstance.Q 的扇出 */
[0137]            If(traceFanouts(Q,netlist) == true){
[0138]                return ture ;
[0139]            }else{
[0140]               return false ;
[0141]        }
[0142]        /* 扫描单元模板匹配的程序 */
[0143]        bool isScanCell(m,pD,pSI,pSE,pQ)
[0144]        {
[0145]             if(m- > hasOneSequentialUDP() == false)
[0146]                return false ;
[0147]             pQ = udp.Q ;
[0148]             /* 向回追踪穿过简单门（如果有的话）的 UDP 的数据
[0149]   pport*/
[0150]       if((muxFound = traceBackTillMux(udp.D)) == false)
[0151]           return false ;
[0152]       else{
[0153]          D = mux.A ;SI = mux.B ;SE = mux.C ;
[0154]       }
[0155]       /* 向回追踪 D/SI/SE 信号直到模块端口边界。如果在路径中发现任何循环、
```

```
复杂门则返回假 */
[0156]      if((traceBackTillPort(D,pD,SI,pSI,SE,dSE) == false)
[0157]         Return false;
[0158]      /* 模板匹配成功则返回真 ;*/
[0159]      return true;
[0160]  }
[0161]      /* 改变至代码产生程序 */
[0162]      doCodeGen(netlist)
[0163]      {
[0164]       /*
[0165]        * 当产生了 cellInstance.Q 的传播程序,检查其是否
[0166]    * 由 doScanOpt() 标记为可优化。如果是,则产生监视代码。
[0167]    */
[0168]    if(isMarkedAsOptimized(cellInstance.Q))
[0169]    {
[0170]        codeGenIfCheck(“if(cellInstance.SE == 0)”);
[0171]    }
[0172]    codeGenPropagate(“propagate(Q);”);
[0173]  }
```

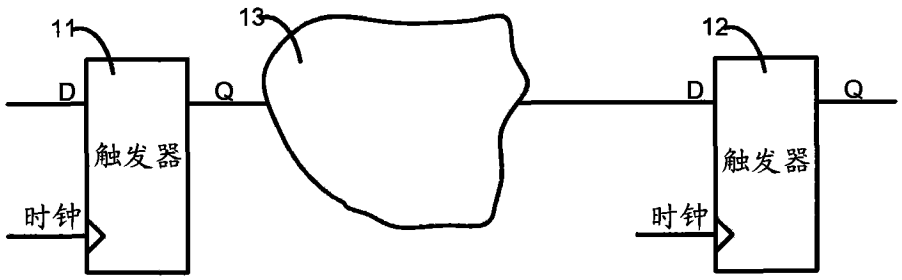

图 1A（现有技术）

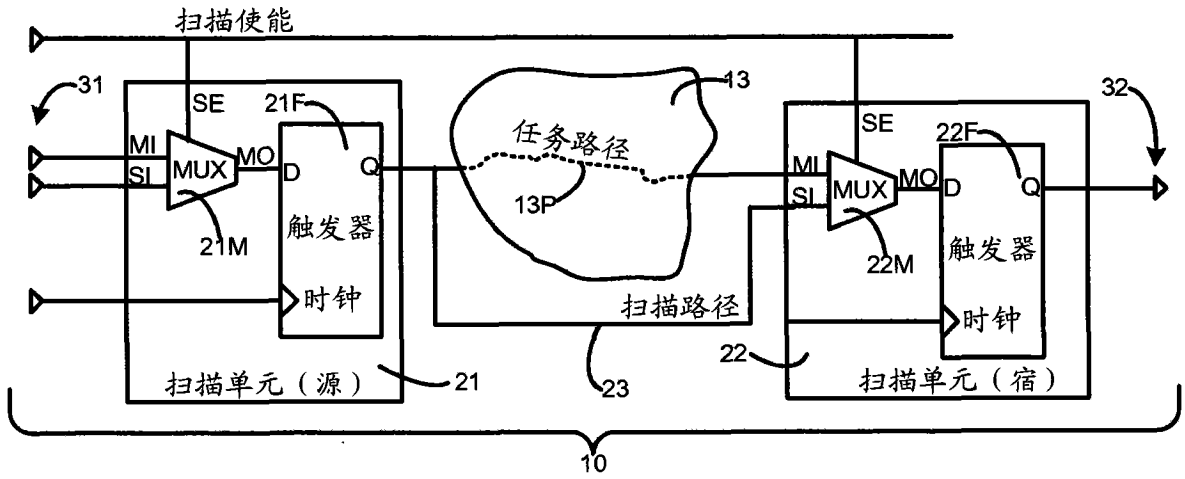

图 1B（现有技术）

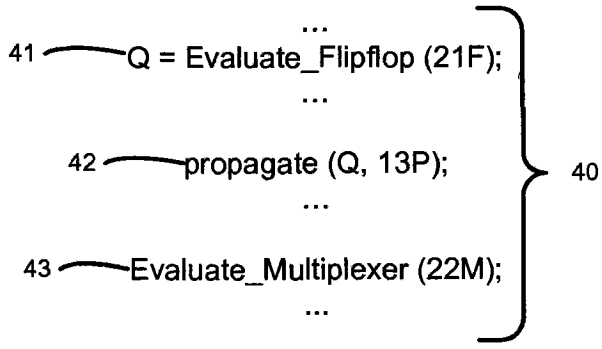

图 1C（现有技术）

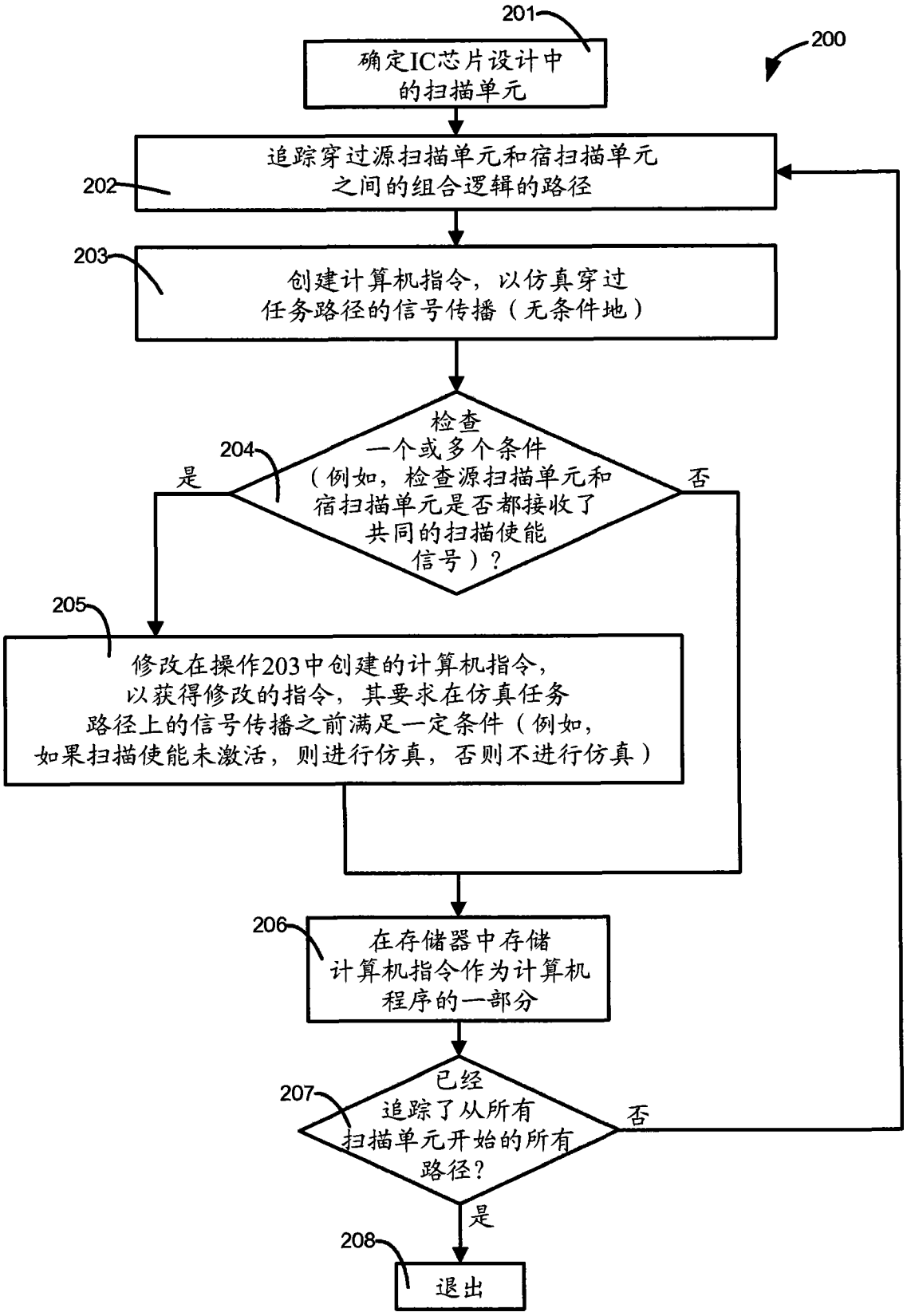
201
确定IC芯片设计中的扫描单元
200
追踪穿过源扫描单元和宿扫描单元之间的组合逻辑的路径
202
203
创建计算机指令，以仿真穿过任务路径的信号传播（无条件地）
检查一个或多个条件（例如，检查源扫描单元和宿扫描单元是否都接收了共同的扫描使能信号）？
204
是
否
205
修改在操作203中创建的计算机指令，以获得修改的指令，其要求在仿真任务路径上的信号传播之前满足一定条件（例如，如果扫描使能未激活，则进行仿真，否则不进行仿真）
206
在存储器中存储计算机指令作为计算机程序的一部分
已经追踪了从所有扫描单元开始的所有路径？
207
否
是
208
退出

图 2A

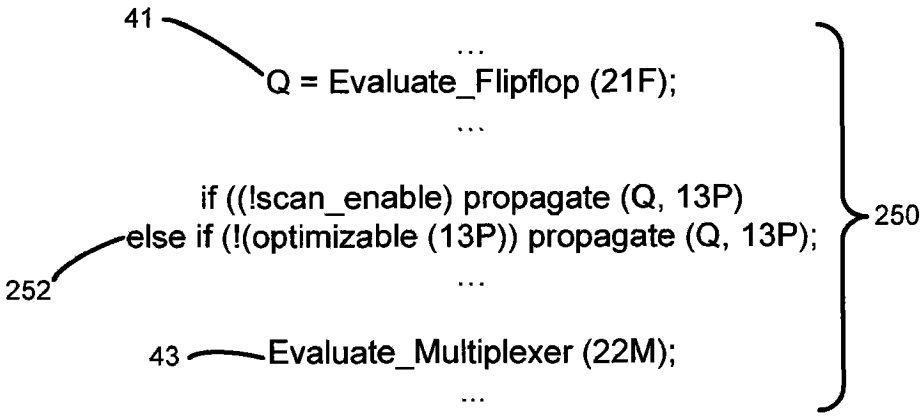
41
...
Q = Evaluate_Flipflop (21F);
...
if ((!scan_enable) propagate (Q, 13P)
else if (!(optimizable (13P)) propagate (Q, 13P);
252
...
43
Evaluate_Multiplexer (22M);
...
250

图 2B

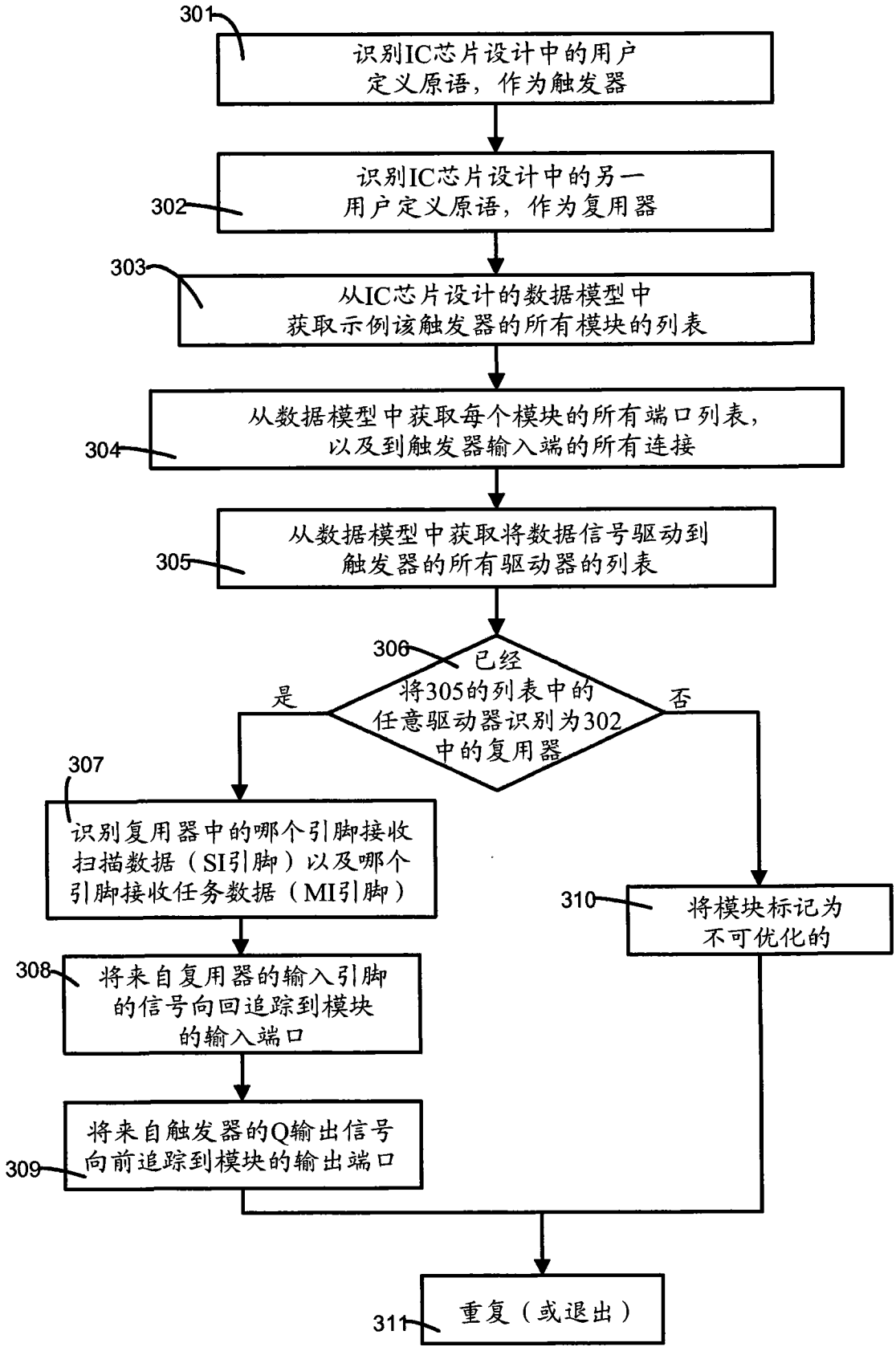
301
识别IC芯片设计中的用户定义原语，作为触发器
302
识别IC芯片设计中的另一用户定义原语，作为复用器
303
从IC芯片设计的数据模型中获取示例该触发器的所有模块的列表
304
从数据模型中获取每个模块的所有端口列表，以及到触发器输入端的所有连接
305
从数据模型中获取将数据信号驱动到触发器的所有驱动器的列表
306
已经将305的列表中的任意驱动器识别为302中的复用器
是
否
307
识别复用器中的哪个引脚接收扫描数据（SI引脚）以及哪个引脚接收任务数据（MI引脚）
308
将来自复用器的输入引脚的信号向回追踪到模块的输入端口
309
将来自触发器的Q输出信号向前追踪到模块的输出端口
310
将模块标记为不可优化的
311
重复（或退出）

图 3

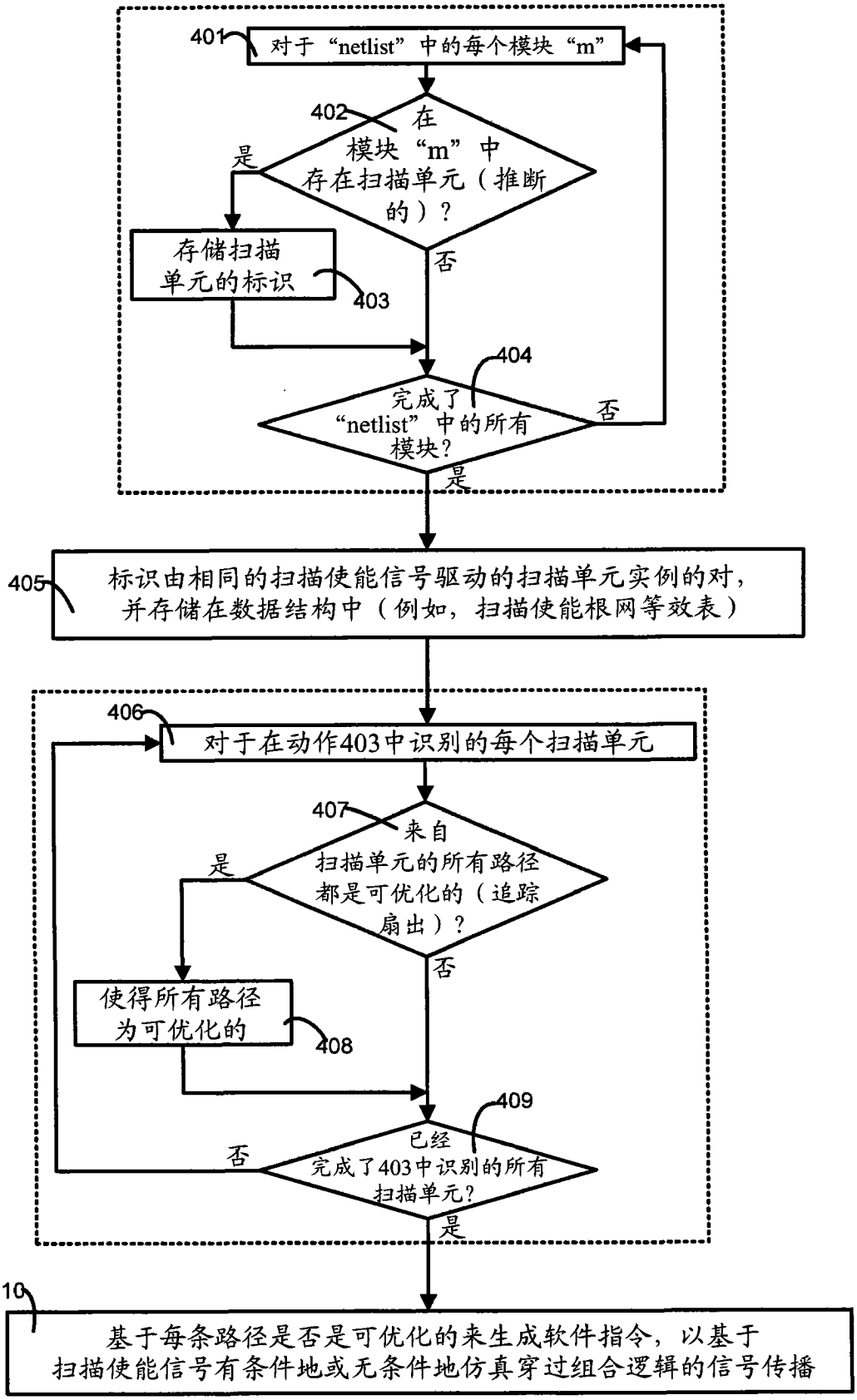
401
对于“netlist”中的每个模块“m”
402
在模块“m”中存在扫描单元（推断的）？
是
否
存储扫描单元的标识
403
404
完成了“netlist”中的所有模块？
否
是
405
标识由相同的扫描使能信号驱动的扫描单元实例的对，并存储在数据结构中（例如，扫描使能根网等效表）
406
对于在动作403中识别的每个扫描单元
407
来自扫描单元的所有路径都是可优化的（追踪扇出）？
是
否
使得所有路径为可优化的
408
409
已经完成了403中识别的所有扫描单元？
否
是
410
基于每条路径是否是可优化的来生成软件指令，以基于扫描使能信号有条件地或无条件地仿真穿过组合逻辑的信号传播

图 4A

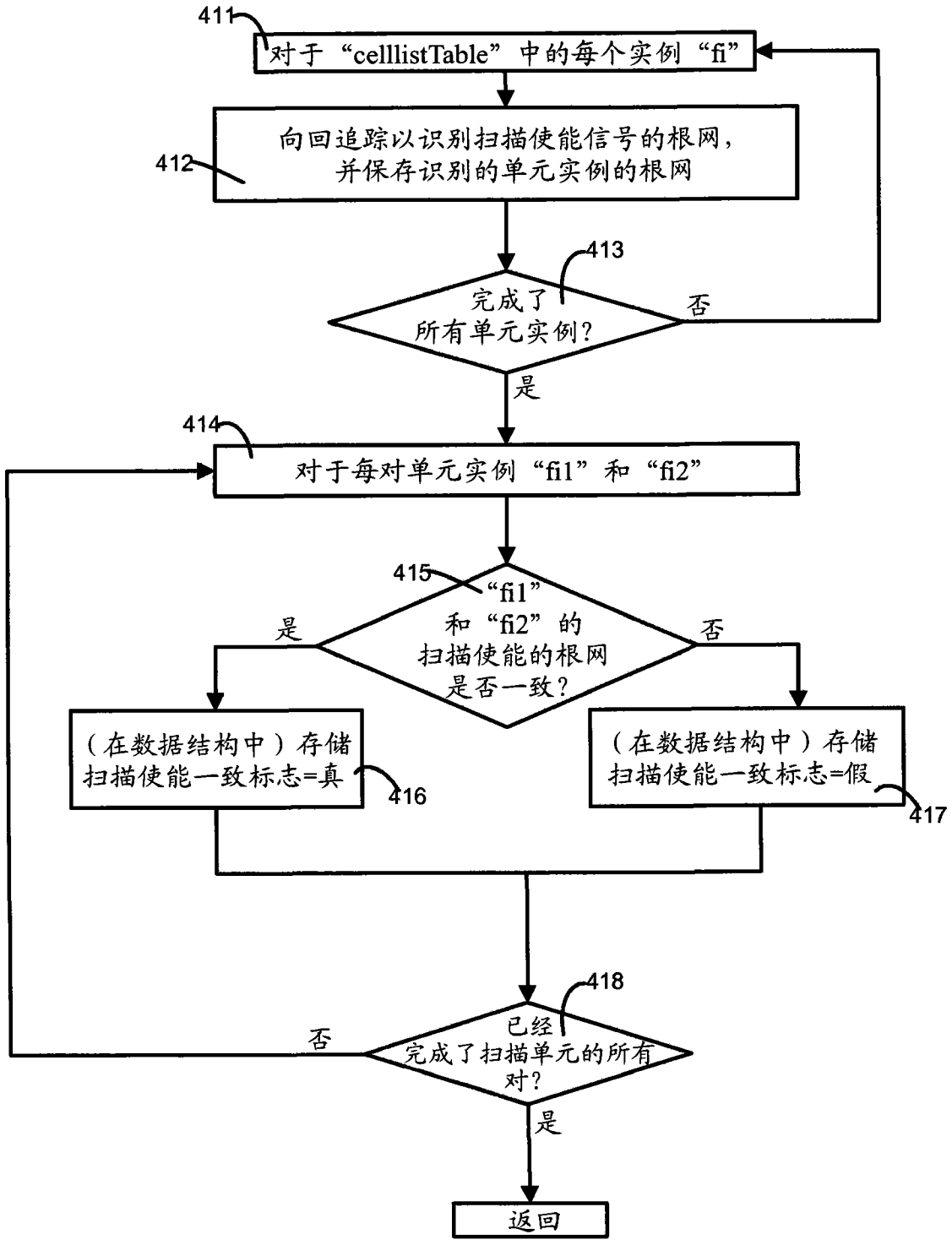
411
对于“celllistTable”中的每个实例“fi”
412
向回追踪以识别扫描使能信号的根网，并保存识别的单元实例的根网
413
完成了所有单元实例？
否
是
414
对于每对单元实例“fi1”和“fi2”
415
“fi1”和“fi2”的扫描使能的根网是否一致？
是
否
（在数据结构中）存储扫描使能一致标志=真
416
（在数据结构中）存储扫描使能一致标志=假
417
418
已经完成了扫描单元的所有对？
否
是
返回

图 4B

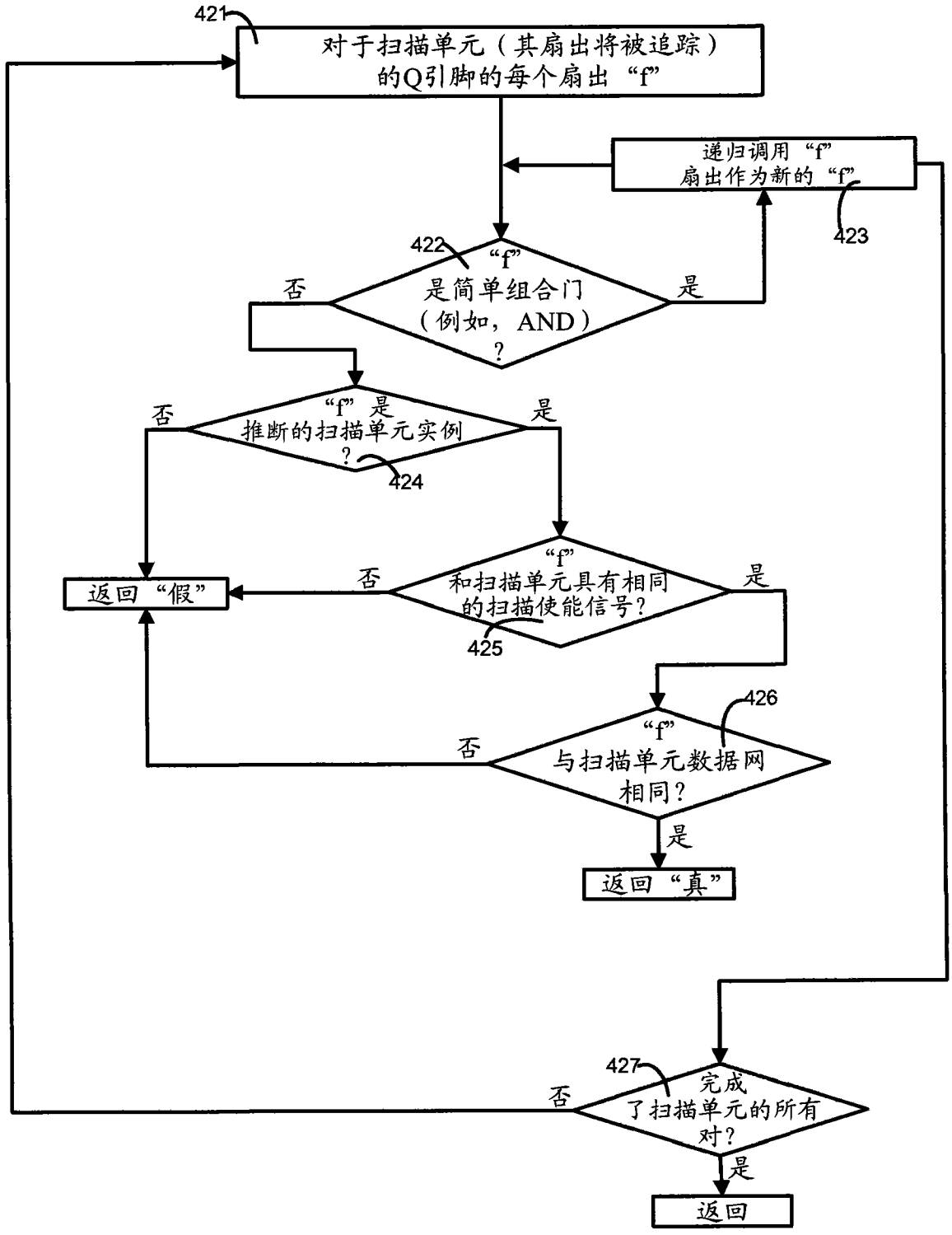
421
对于扫描单元（其扇出将被追踪）的Q引脚的每个扇出“f”
递归调用“f”扇出作为新的“f”
423
422
“f”是简单组合门（例如，AND）？
否
是
“f”是推断的扫描单元实例？
424
否
是
返回“假”
“f”和扫描单元具有相同的扫描使能信号？
425
否
是
426
“f”与扫描单元数据网相同？
否
是
返回“真”
427
完成了扫描单元的所有对？
否
是
返回

图 4C

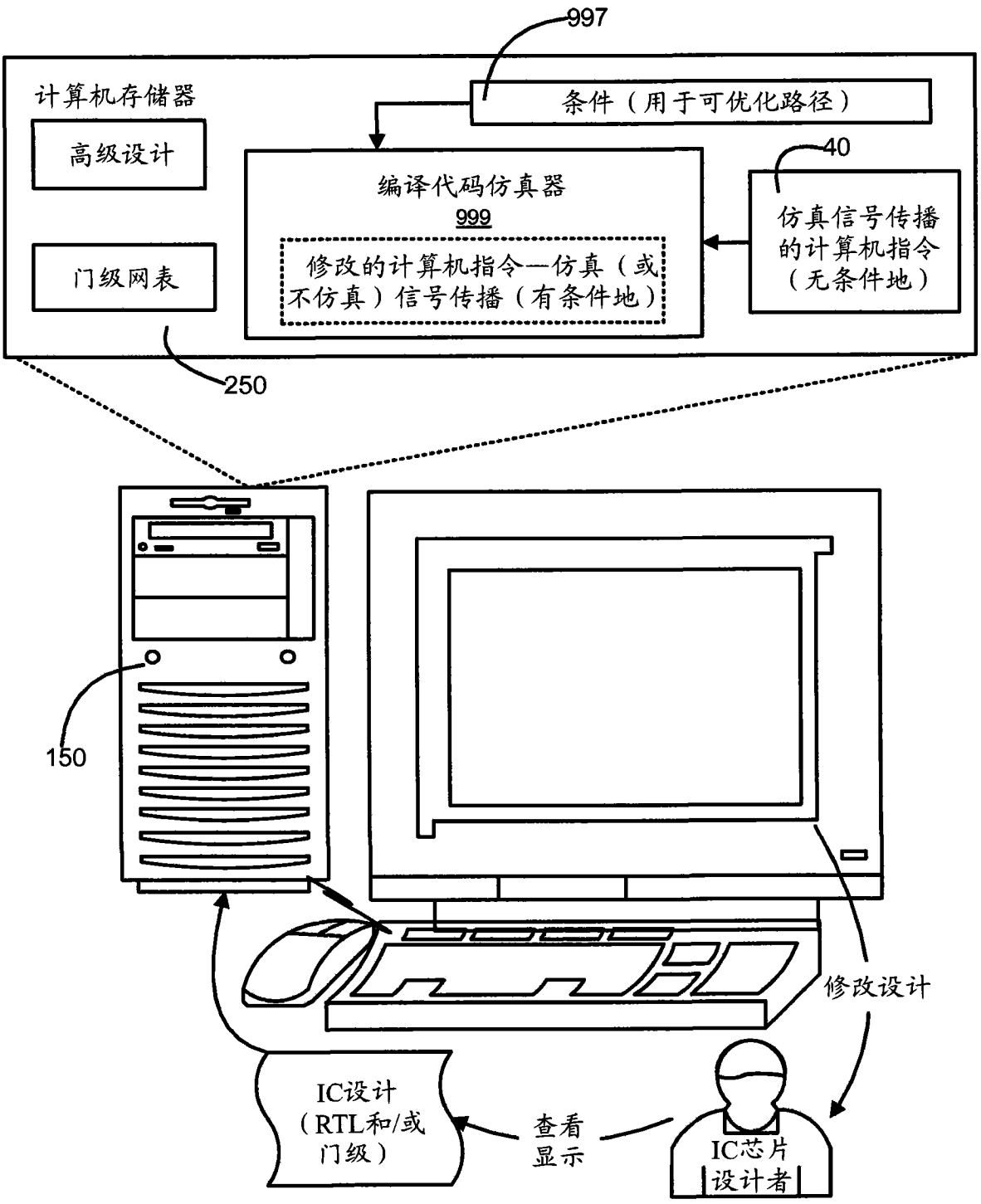
997
计算机存储器
条件（用于可优化路径）
高级设计
编译代码仿真器
999
40
仿真信号传播的计算机指令（无条件地）
门级网表
修改的计算机指令—仿真（或不仿真）信号传播（有条件地）
250
150
修改设计
IC设计（RTL和/或门级）
查看显示
IC芯片设计者

图 5

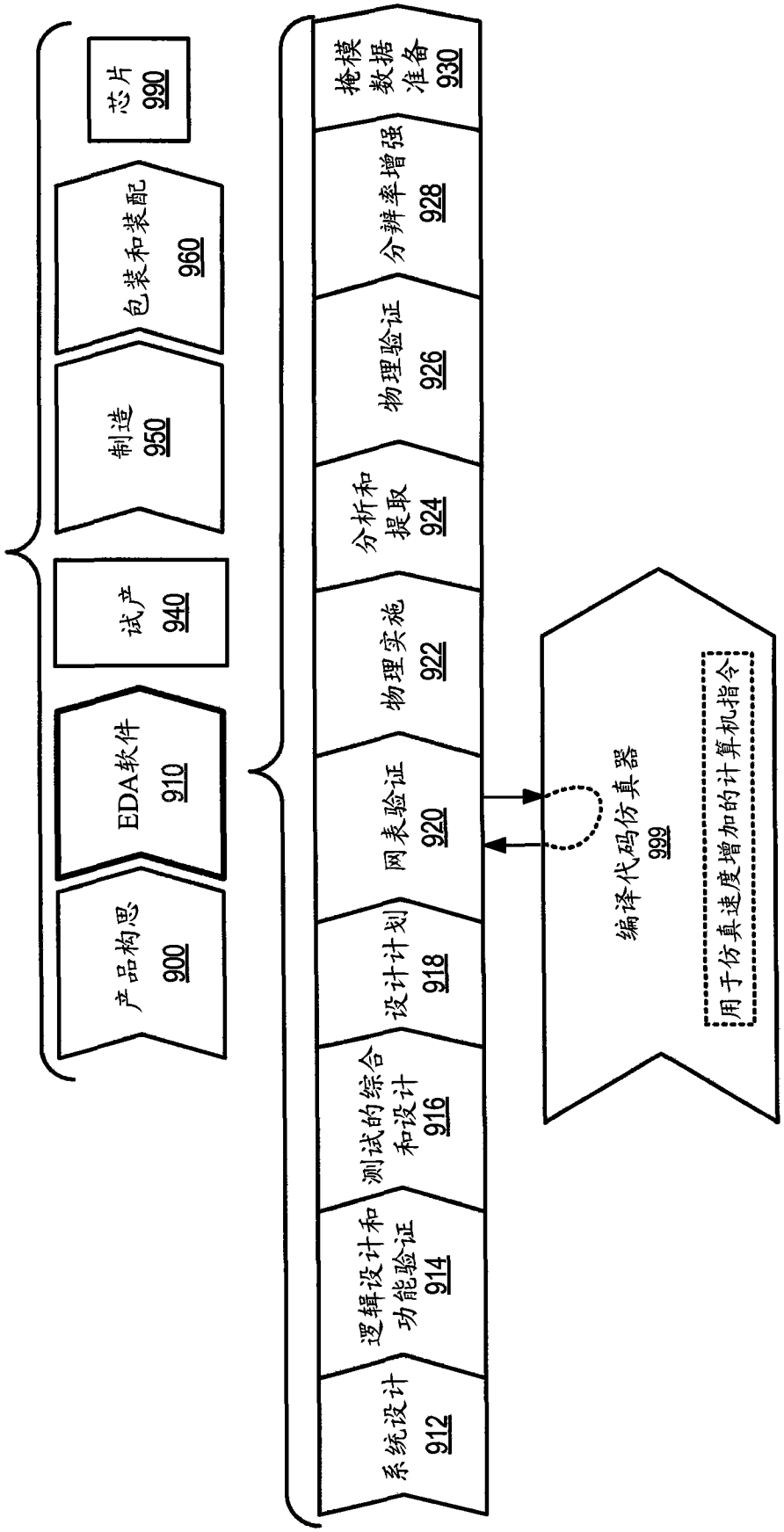
产品构思
900
EDA软件
910
试产
940
制造
950
包装和装配
960
芯片
990
系统设计
912
逻辑设计和
功能验证
914
测试的综合
和设计
916
设计计划
918
网表验证
920
物理实施
922
分析和
提取
924
物理验证
926
分辨率增强
928
掩模
数据
准备
930
编译代码仿真器
999
用于仿真速度增加的计算机指令

图 6