



(12) 发明专利

(10) 授权公告号 CN 110020456 B

(45) 授权公告日 2024. 08. 27

(21) 申请号 201811616236.2

(22) 申请日 2018.12.28

(65) 同一申请的已公布的文献号
申请公布号 CN 110020456 A

(43) 申请公布日 2019.07.16

(30) 优先权数据
102018100423.0 2018.01.10 DE

(73) 专利权人 德斯拜思有限公司
地址 德国帕德博恩

(72) 发明人 D·卢贝雷 H·卡尔特

(74) 专利代理机构 中国贸促会专利商标事务所
有限公司 11038
专利代理师 刘盈

(51) Int.Cl.

G06F 30/34 (2020.01)

G06F 16/22 (2019.01)

(56) 对比文件

CN 101196945 A, 2008.06.11

CN 101620643 A, 2010.01.06

审查员 张琳琳

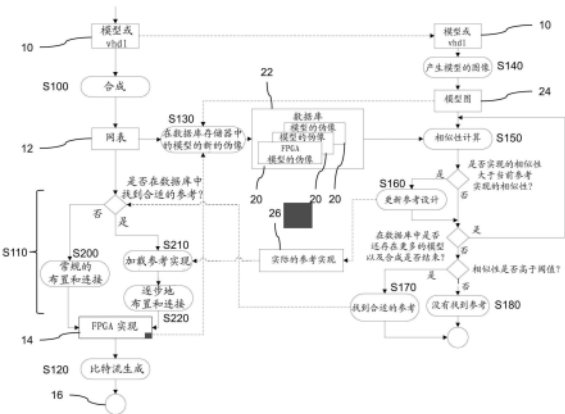
权利要求书2页 说明书9页 附图9页

(54) 发明名称

利用基于图的相似性搜索逐步生成FPGA实现的方法

(57) 摘要

本发明涉及一种基于作为FPGA模型的FPGA设计和/或硬件描述来生成FPGA实现的方法,该方法包括如下步骤:从FPGA设计合成网表,和从网表产生FPGA实现,其中,该方法包括:搜索相似的FPGA实现,和在使用相似的FPGA实现的情况下实现从网表产生FPGA实现的步骤,该方法包括用于基于FPGA设计来产生基于图的表示的步骤,并且搜索相似的FPGA实现的步骤包括将FPGA设计的基于图的表示与至少一个相似的FPGA实现的基于图的表示进行比较。此外,本发明涉及一种基于作为FPGA模型的FPGA设计和/或硬件描述来生成比特流的方法,该方法包括用于按照上述方法的基于作为FPGA模型的FPGA设计和/或硬件描述来生成FPGA实现的步骤,并且从生成的FPGA实现中产生比特流。



1. 一种用于基于作为FPGA模型的FPGA设计(10)和/或硬件描述来生成FPGA实现(14)的方法,该方法包括如下步骤:

从FPGA设计(10)合成网表(12),和
从网表(12)产生FPGA实现(14),其中,
该方法包括:搜索相似的FPGA实现(14),和
在使用所述相似的FPGA实现(14)的情况下实现从网表(12)产生FPGA实现(14)的步骤,
其特征在于,
该方法包括用于基于FPGA设计(10)来产生基于图的表示的步骤,并且
搜索相似的FPGA实现(14)的步骤包括将FPGA设计(10)的基于图的表示与至少一个相似的FPGA实现(14)的基于图的表示进行比较。

2. 根据权利要求1所述的方法,其特征在于:
所述FPGA设计(10)包括部分实现,
搜索相似的FPGA实现(14)包括搜索相似的部分实现,和
在使用所述相似的部分实现的情况下进行产生FPGA实现(14)的步骤。
3. 根据权利要求1或2所述的方法,其特征在于:
搜索相似的FPGA实现(14)包括在先前的FPGA实现(14)的数据库(22)中搜索相似的FPGA实现(14)。

4. 根据权利要求1或2所述的方法,其特征在于:
该方法包括用于识别FPGA设计(10)的经改变的部分区域的步骤,
网表(12)的合成包括合成针对FPGA设计(10)的经改变的部分区域的网表(12),以及
FPGA实现(14)的产生包括产生针对FPGA设计(10)的经改变的部分区域的网表(12)的FPGA实现(14)。

5. 根据权利要求1或2所述的方法,其特征在于:
该方法包括用于确定在FPGA设计(10)的基于图的表示与所述至少一个相似的FPGA实现(14)的基于图的表示之间的相似性程度的步骤,以及
当相似性程度低于阈值时,该方法包括用于在不使用相似的FPGA实现(14)的情况下从网表(12)产生FPGA实现(14)的传统步骤。

6. 根据权利要求1或2所述的方法,其特征在于:
将FPGA设计(10)的基于图的表示与所述至少一个相似的FPGA实现(14)的基于图的表示进行比较的步骤包括线性优化问题的求解,其中,对于图中的每个节点识别出另一图中最相似的节点,而不加倍地配设该节点。

7. 根据权利要求1或2所述的方法,其特征在于:
所述方法包括用于确定FPGA设计(10)的基于图的表示的节点与所述至少一个相似的FPGA实现(14)的基于图的表示的节点的相似性连同确定区分点的一致性的步骤。

8. 根据上述权利要求7所述的方法,其特征在于:
确定区分点的一致性的步骤包括对区分点的加权。

9. 根据权利要求1或2所述的方法,其特征在于:
该方法包括用于执行多个包含在数据库(22)中的图(24)的相似性分析以及将所计算的相似性存储在相似性线图(30)或相似性矩阵中的附加步骤。

10. 根据权利要求9所述的方法,其特征在于:

该方法包括用于计算在包含于数据库 (22) 中的图 (24) 之间的关系的相似性的附加步骤。

11. 根据权利要求10所述的方法,其特征在于:

该方法包括用于借助已经确定的相似性来预先确定尚未确定的关系的期望的高相似性的附加步骤。

12. 根据权利要求1或2所述的方法,其特征在于:

将FPGA设计 (10) 的基于图的表示与所述至少一个相似的FPGA实现 (14) 的基于图的表示进行比较的步骤包括邻近匹配,以用于考虑邻近关系。

13. 根据权利要求1或2所述的方法,其特征在于:

基于FPGA设计 (10) 来产生基于图的表示的步骤包括:在考虑到虚拟连接的情况下FPGA设计 (10) 的去层次化,以及基于所述FPGA设计 (10) 的去层次化,转换功能的应用。

14. 一种用于基于作为FPGA模型的FPGA设计 (10) 和/或硬件描述来生成比特流 (16) 的方法,该方法包括如下步骤:

根据权利要求1至13之一地基于作为FPGA模型的FPGA设计 (10) 和/或硬件描述来生成FPGA实现 (14),和

从生成的FPGA实现 (14) 中产生比特流 (16)。

15. 一种数据处理装置,其包括:

存储器,在其中存储有可执行指令;以及

处理器,其被配置为用于执行所述可执行指令,

其特征在于:所述处理器在执行所述指令时实施根据权利要求1至14之一所述的方法。

16. 计算机可读存储介质,所述计算机可读存储介质具有可执行指令,其特征在于:所述可执行指令促使计算机在执行所述可执行指令时实施根据权利要求1至14之一所述的方法。

利用基于图的相似性搜索逐步生成FPGA实现的方法

技术领域

[0001] 本发明涉及一种用于基于作为FPGA模型的FPGA设计和/或硬件描述来生成FPGA实现的方法,该方法包括如下步骤:从FPGA设计合成网表,和从网表产生FPGA实现,其中,该方法包括:搜索相似的FPGA实现,和在使用相似的FPGA实现的情况下实现从网表产生FPGA实现的步骤。

[0002] 此外,本发明涉及一种用于基于作为FPGA模型的FPGA设计和/或硬件描述来生成比特流的方法,该方法包括用于基于作为FPGA模型的FPGA设计和/或硬件描述来生成FPGA实现的上述步骤,并且从生成的FPGA实现中生成比特流。

背景技术

[0003] 这样的方法被用于生成针对FPGA(现场可编程门阵列)的代码。FPGA是可以在其中加载逻辑电路的集成电路。与已知编程语言中的传统的编程相比,编程在此包括较少的时间序列规定,而是包括所期望的电路结构的定义。

[0004] 在此,用于定义所期望的电路结构的FPGA设计典型地生成为图形FPGA模型,该FPGA模型被转换为硬件描述语言,例如VHDL。原则上,FPGA设计也可以直接在硬件描述语言中生成。

[0005] 然后,FPGA设计在合成步骤中由相应的软件先转换成网表,该网表规定了FPGA中单个的逻辑元件应如何电路连接。因此,也谈到了FPGA的配置。具有逻辑元件及其电路连接的网表被保存用于进一步使用。

[0006] 接下来是用于生成FPGA实现的步骤。该步骤通常被称为“布局 and 布线”。在此,存储在网表中的用于FPGA的逻辑元件被布置(Place),并根据该网表布线(Route)。结果是FPGA实现。

[0007] 随后,从FPGA实现产生比特流,该比特流被传输到FPGA上,以便在该FPGA中实现期望的功能。

[0008] 上述步骤通常被概括为“构建”。

[0009] 现在,有一些易于操纵的为客户提供例如抽象FPGA开发环境的工具,从而客户即使在没有详细了解FPGA和工具流程的情况下也可以开发自己的硬件。由此,客户可以创建、实现FPGA设计,并且在相应的FPGA硬件上运行该FPGA。通过随后的重新构建对FPGA模型的定期更改是这样的开发过程例如在快速控制原型中的自然组成部分。

[0010] 在此,一个问题在于通常非常长的构建时间,所述构建时间对于重新构建FPGA而言即使在仅细微地改变FPGA模型时也可能是必需的。在此,构建时间通常在数小时的范围内。因为FPGA变得越来越复杂并且提供越来越多的以寄存器、逻辑电路、DSP模块或存储器形式的资源,所以构建时间不断增加。这仅可以部分地通过算法改进和用于执行构建的更快的计算机来补偿。特别耗时的步骤在此通常是布置和连接。

[0011] 在当前的FPGA工具中存在两种方案,以便减少构建时间。两种方案基于:再次使用已经成功生成的FPGA实现的未改变的区域/组成部分。在第一种方案中,为每个组成部分在

FPGA中配设一个(矩形的)区域。当组成部分被改变时,重新实现具有该组成部分的该区域,而FPGA实现的其余部分可以保持不变并且可以被再次使用。在第二种方案(在此称为逐步构建)中,不需要预定义区域。为了实现,给出已经实现的FPGA实现作为参考。当一致性大时,则可以自动承担很多布置和布线,并且减少用于生成FPGA实现的时间。

[0012] 在此有利的可以是,从一开始就已经生成FPGA实现,从而可以再次使用一些模块或者说部分实现。该方案遵循FPGA模型的自动模块化。可以在该模块基础上进行布图规划,亦即划分区域。当还为各个模块提供储备时,FPGA模型的改变可以保持局限于各个模块。

[0013] 此外,用于重新规划的方法是已知的。当客户改变了其FPGA模型时,使用最后的构建结果。基于此,单独构建改变的模块,并且接着在整体构建中进行替换。对于涉及多个模块的情况,可以通过在不同的计算机上进行修改模块的构建的方式来减少构建时间。因此,实施了改变的模块的并行构建。在此,可以提供不同版本的模块的版本化,以便于再次使用。

[0014] 即使这些方法已经可以减少构建时间,仍存在进一步的改善潜力。因此,例如在细粒度模块化的情况下,出现大的错误裁切和计时故障增加的风险,因为不可以越过模块边界进行优化并且必须为模块分别提供一个具有相应储备的宽大区域。此外,该方法仅能以产品开发中的大的实现耗费来实施。

[0015] 为了确定两种设计之间的相似性,通过当前的合成工具将参考网表与要重新布置和布线的网表进行比较。该比较在此仅基于网表路径名称的名称比较。这是有问题的,因为在Xilinx Vivado系统生成器、Xilinx Vivado HLS、Xilinx Vivado SDSoc等抽象的高级FPGA设计工具中常见的是,实施相同的FPGA组成部分的快速重命名和重新组织,例如在子系统中。由此,即使对于实际相同的组成部分,仍自动得到新的网表路径。当前的合成工具无法识别重命名。

发明内容

[0016] 从上述现有技术出发,本发明的任务因此在于提供一种用于基于作为FPGA模型的FPGA设计和/或硬件描述而生成FPGA实现的方法以及一种用于基于作为上述类型的FPGA模型的FPGA设计和/或硬件描述而生成比特流的方法,所述方法能实现简单且有效地生成FPGA实现,特别是在使用已经存在的相似FPGA实现的情况下。

[0017] 该任务根据本发明通过有利的实施方案的特征解决。

[0018] 因此,按照本发明提供了一种用于基于作为FPGA模型的FPGA设计和/或硬件描述来生成FPGA实现的方法,该方法包括如下步骤:从FPGA设计合成网表,和从网表产生FPGA实现,其中,该方法包括:搜索相似的FPGA实现,和在使用相似的FPGA实现的情况下实现从网表产生FPGA实现的步骤,该方法包括用于基于FPGA设计来产生基于图的表示的步骤,并且搜索相似的FPGA实现的步骤包括将FPGA设计的基于图的表示与至少一个相似的FPGA实现的基于图的表示进行比较。

[0019] 根据本发明,还提供一种用于基于作为FPGA模型的FPGA设计和/或硬件描述来生成比特流的方法,该方法包括用于基于作为FPGA模型的FPGA设计和/或硬件描述来生成FPGA实现的上述步骤,并且从生成的FPGA实现中产生比特流。

[0020] 因此,本发明的基本思想是找到当前FPGA模型与其他的、例如先前的FPGA模型的

相似性,并且被使用来执行FPGA实现的高效产生。基于此,执行以至少一个现有的相似的FPGA实现为基础的当前FPGA实现的逐步产生。然后,使用相似的FPGA实现或其至少相似的部分来产生当前的FPGA实现。因为布置和布线FPGA逻辑来产生FPGA实现通常要求FPGA构建的大部分时间,由此总体上减少了用于产生FPGA构建的持续时间。

[0021] FPGA设计通常作为图形化的FPGA模型产生,并且其转换为硬件描述语言。原则上,FPGA设计也可以直接在硬件描述语言中产生。例如VHDL作为硬件描述语言已知。VHDL作为超高速集成电路硬件描述语言(也称为VHSIC硬件描述语言)已知。

[0022] 然后,FPGA设计在合成步骤中由相应的软件先转换成网表,该网络规定了FPGA中的单个的逻辑元件应如何电路连接。因此,也涉及到FPGA的配置。具有逻辑元件及其电路连接的网表被存储用于进一步使用。

[0023] 接着是用于产生FPGA实现的步骤。该步骤通常称为“布局 and 布线”。在此,针对FPGA的存储在网表中的逻辑元件被布置(Place),并将所述逻辑元件根据网表布线(Route)。结果是FPGA实现。

[0024] 随后,从FPGA实现产生比特流,该比特流被传输到FPGA,以便在该FPGA中实现期望的功能。

[0025] 上述步骤通常概括为“构建”。

[0026] 优选地,合成的网表作为相应的FPGA模型的伪像(Artefakt)存档在数据库中,以便能够在后续的构建中使用相应的FPGA实现,必要时,作为用于搜索相似FPGA实现的参考。也可以在合成期间就已经搜索尽可能相似的FPGA实现作为参考。对此,首先创建基于FPGA设计的基于图的表示,并将其与之前已经实现的FPGA设计的图进行比较。

[0027] 在此,通常在相似的FPGA实现的相应构建期间完成所述至少一个相似的FPGA实现的图的创建。这意味着:相似的FPGA实现是按照传统方式产生的,或者也根据在这里说明的方法从相应的FPGA设计出发产生。然后,这些FPGA实现的图同样基于相应的FPGA设计产生。

[0028] 总体上,该方法可以容易地实现,其中,不同的步骤可以并行地执行。因此,可以高效地使用资源。在此,可以选择相似的FPGA实现,或者假设最小相似性,以便将相似的FPGA实现用作为FPGA设计的基础。如果给出足够的相似性,则可以使用相应的FPGA实现作为参考。如果还存在其他的尚未进行相似性调查的FPGA设计,则可以继续搜索具有这些FPGA设计的相似的或最相似的FPGA实现。

[0029] 在从FPGA设计创建图时,FPGA模型的块转换为节点,信号线转换为指引线。该方法是分层适用的。图形子系统可以要么被解释为用于第一快速估计的块,要么任意深地直至完全地被去层次化。转换功能例如可以允许:在考虑到虚拟连接(Goto/From、数据存储读取/数据存储内存/数据存储写入、触发端口、操作端口、...)的情况下对除了FPGA模型的基本块之外的FPGA模型进行去层次化,并且由此产生定向图。在此,FPGA模型不仅可以由连续的图组成,还可以包含多个分离的图。例如,当在FPGA设计中实现彼此独立的子功能时,得到这样的分开的图。

[0030] 基于用于图的相似性分析的算法,将FPGA设计的基于图的表示与所述至少一个相似的FPGA实现的基于图的表示进行比较。基于两个模型图的每两个节点之间的相似性程度,可以使用已知的算法。两个节点的相似性的程度借助于块/网表路径(也局部地),块/网表类型(例如,加法,乘法),和块/网表参数/属性(例如位宽,二进制点位置)的一致性程度

确定。

[0031] 在本发明的有利实施方式中,所述FPGA设计包括部分实现,搜索相似的FPGA实现包括搜索相似的部分实现,和在使用相似的部分实现的情况下进行产生FPGA实现的步骤。因此,通过确定并且比较这些部分实现的相似性,可以将相似性搜索问题分解为不同的子问题。

[0032] 在本发明的有利实施方式中,搜索相似的FPGA实现包括在先前的FPGA实现的数据库中搜索相似的FPGA实现。例如,所有FPGA实现都可以存储在数据库中,以便提供这些FPGA实现用于稍后的使用。在此,数据库优选地包括在项目的框架中产生的FPGA实现,因为因此可以从针对相似的FPGA实现的增加的概率出发。因此,对相似FPGA实现的搜索可能在范围上受到限制。

[0033] 在本发明的有利实施方式中,该方法包括用于识别FPGA设计的改变的部分区域的步骤,网表的合成包括从FPGA设计的改变的部分区域合成网表,以及产生FPGA实现包括产生针对FPGA设计的改变的部分区域的网表的FPGA实现。因此,新的FPGA实现的创建可以从一开始就限制于FPGA设计的改变的部分区域。优选地,考虑FPGA设计的如下块,对于所述块可以单独地重新执行FPGA实现。

[0034] 在本发明的有利的实施方案中,该方法包括用于确定在FPGA设计的基于图的表示与所述至少一个相似的FPGA实现的基于图的表示之间的相似性程度的步骤,以及该方法包括当相似性程度低于阈值时在不使用相似的FPGA实现的情况下从网表产生FPGA实现的传统步骤。因此,如果找到合适的相似的FPGA实现作为参考,则可以在产生当前的FPGA实现期间使用它。在其他情况下,必须完全重新实现FPGA模型。越早在准备阶段识别出不存在有利的FPGA实现作为参考(其相似性程度高于阈值),则搜索相似的FPGA实现所需的资源越少。由此,常规构建相对于具有极其不同的FPGA模型的逐步构建,当相似性程度低于阈值时,可以减少用于构建的时间。阈值优选地为75%相似性。进一步优选地,阈值为85%相似性,甚至进一步优选为90%相似性。

[0035] 在本发明的有利的实施方案中,将FPGA设计的基于图的表示与所述至少一个相似的FPGA实现的基于图的表示进行比较的步骤包括线性优化问题的求解,其中,对于图中的每个节点,识别出另一图中最相似的节点,而不加倍地配设该节点。使用根据Kuhn的作为“匈牙利方法”已知的方法。“匈牙利方法”用于线性优化问题求解,即在不加倍地配设节点的情况下对于每个节点在另一个图中找到最相似的节点。该方法使整体相似性最大化。不过,可以引入邻近关系作为两个块的相似性的尺度。该方法原则上在不考虑邻近关系的情况下进行。

[0036] 在本发明的有利的实施方案中,所述方法包括用于确定FPGA设计的基于图的表示的节点与所述至少一个相似的FPGA实现的基于图的表示的节点的相似性连同确定区分点(Differenzierungspunkt)的一致性的步骤。区分点例如包括块/网表路径(也局部地),块/网表类型(例如,加法,乘法),和块/网表参数/属性(例如位宽,二进制点位置)。

[0037] 在本发明的有利的实施方案中,确定区分点的一致性的步骤包括对区分点的加权。通过加权,可以极为不同地考虑各区分点。

[0038] 在本发明的有利的实施方案中,该方法包括用于执行多个包含在数据库中的图的相似性分析并且将所计算的相似性存储在相似性线图或矩阵中的附加步骤。因此,可以存

储在相似性分析中发现的这样的相似性,以便通过将FPGA设计的基于图的表示与所述至少一个相似的FPGA实现的基于图的表示进行比较来便于将来搜索相似的FPGA实现。计算出的相似性在这里涉及在搜索相似的FPGA实现中所确定的相似性。

[0039] 在本发明的有利的实施方案中,该方法包括用于计算包含在数据库中的图之间的关系的相似性的附加步骤。

[0040] 因此,例如可以通过预先分析图或部分图,并由此获取和存储结构的方式来执行一种“散列法”。该结构例如可以存储为所谓的散列值。在图的相似性的后续比较中,因此可以基于关系的计算出的相似性来执行该比较,由此通过FPGA设计的基于图的表示与所述至少一个相似的FPGA实现的基于图的表示的比较来搜索相似的FPGA实现借助所述结构而得到便利化。计算在包含在数据库中的图之间的关系的相似性的步骤伴随有大的计算耗费,并且应该可以在其中使用空闲的资源阶段、例如在开发者的建模阶段进行,在该建模阶段中在低的计算负载的情况下产生FPGA实现。优选地,从多个FPGA模型中创建一个相似性图,该相似性图存储所有相互比较的FPGA模型之间的相似性分析结果。两个FPGA模型之间的查得的相似性被给出为在相应两个节点之间的指引线例如带有作为相似性的程度的权重。在完全互联的相似图中,给出每个FPGA模型与其他的每个FPGA模型的相似性。完全的相似性图能实现:可以为逐步的FPGA构建找到理想的起点。完成可以通过不同的方式实现。作为第一变型方案,可以根据暴力法通过进一步相似性分析来补充用于相似性图的完全互联的缺失的指引线。如已经所说明的那样,这导致对计算能力的较大需求,并且因此可以例如在具有低的CPU负载的FPGA设计的创建期间例如在后台进行。第二种变型方案基于现有FPGA实现的有效计算顺序。在此,可以估计基于可能存在潜在的高相似性的现有指引线的两个节点之间的相似性,以便优先为它们执行相似性分析。由此,可以进一步完成相似性图的互联。例如,可以使用概率方案。在此,相似性类似于概率那样沿着现有指引线上的路径相乘:路径2-1-4=45%×85%=38%。根据第三变型方案,使用来自变型方案2的方案来估计基于现有指引线的两个节点之间的相似性。在这种方案中没有分析FPGA设计的准确的相似性。

[0041] 在本发明的有利的实施方案中,该方法包括用于借助已经确定的相似性来预先确定尚未确定的关系的期望的高相似性的附加步骤。越早在准备阶段识别哪个相似的FPGA实现可以用作参考,或者相似的FPGA实现是否可以用作参考,则可以更有效地执行该方法。

[0042] 在本发明的有利的实施方案中,将FPGA设计的基于图的表示与所述至少一个相似的FPGA实现的基于图的表示进行比较的步骤包括邻近匹配,以用于考虑邻近关系。这样的方法例如作为根据Nikolic的“Measuring Similarity of Graph Nodes by Neighbor Matching”已知。在这里,相似性被确定为两个图之间的相似性,并且由优化的最佳的节点对应的节点相似性的总和除以节点的数量得到。

[0043] 在本发明的有利的实施方案中,基于FPGA设计来产生基于图的表示的步骤包括:在考虑到虚拟连接的情况下FPGA设计的去层次化和基于所述FPGA设计的去层次化的转换功能的应用。在分层次的子系统中,这些子系统可以被解释为用于第一快速估计的块,或者任意深地直至完全地被去层次化。当使用这些块作为用于创建FPGA实现的基础时,需要转换来检测并且正确应用虚拟连接和分层块相关性。虚拟连接例如可以包括诸如Goto/From,数据存储读取/数据存储内存/数据存储写入,触发端口,操作端口等的连接。

[0044] 本发明还涉及一种数据处理装置,其包括:存储器,在其中存储有可执行指令;以及处理器,其被配置为用于执行所述可执行指令,其中,所述处理器在执行所述指令时实施根据本发明所述的方法。

[0045] 此外,本发明还涉及一种计算机可读存储介质,所述计算机可读存储介质具有可执行指令,其中,所述可执行指令促使计算机在执行所述可执行指令时实施根据本发明所述的方法。

附图说明

[0046] 下面在参考附图的情况下借助于优选实施方式更详细地阐述本发明。

[0047] 在此,

[0048] 图1根据第一优选实施方式示出用于实施从基于FPGA设计的FPGA实现生成FPGA实现和比特流的方法的流程图;

[0049] 图2根据第一实施方式示出图1的流程图的用于产生FPGA实现和比特流的细节视图;

[0050] 图3根据第一实施方式示出图1的流程图的用于用伪像填充数据库的细节视图。

[0051] 图4根据第一实施方式示出图1的流程图的用于确定相似的FPGA设计的细节视图;

[0052] 图5根据作为第一实施方式的第一实施例示出在其节点34之间具有多个指引线的相似性线图的示意图;

[0053] 图6根据作为第一实施方式的第二实施例示出在所有节点之间构成有指引线的相似性线图的示意图;

[0054] 图7根据作为第一实施方式的第三实施例示出在所有节点之间构成有指引线的相似性线图的示意图,其中,基于节点之间的确定的相似性来确定节点之间的非明确确定的相似性,

[0055] 图8根据第一实施方式示出示例性模型线图的示意图;以及

[0056] 图9根据第一实施方式示出两个模型线图的节点之间的两个相似性矩阵的示意图,其中左侧的图没有“邻近匹配”,右侧的图具有“邻近匹配”。

具体实施方式

[0057] 图1至图4示出了一种用于从基于作为FPGA模型的FPGA设计和/或硬件描述来的FPGA实现中生成FPGA实现和比特流的方法。

[0058] 作为该方法的起始点,在此产生并且提供作为图形FPGA模型的FPGA设计10。然后将FPGA模型转换成硬件描述语言,该硬件描述语言在这里例如是VHDL(超高速集成电路硬件描述语言)。

[0059] 由此出发,可以开始首先包括以下详细描述的步骤S100,S110和S120的构建。

[0060] 在步骤S100中,从FPGA设计10合成网表12。网表12包括逻辑元件及其电路连接,如在FPGA设计10中规定那样。

[0061] 在步骤S110中,从网表12产生FPGA实现14。该步骤通常被称为“布局 and Route”)”。在此,存储在网表12中的用于FPGA的逻辑元件被布置(布局)并且根据网表12彼此连接(布线)。结果是FPGA实现14。下面阐述步骤S110的细节。

[0062] 在步骤S120中,从生成的FPGA实现14产生比特流16,该比特流用于传输到FPGA并且与FPGA一起使用。

[0063] 如图4中详细示出地,用于确定相似的FPGA设计的步骤S140至S180与先前的步骤并行执行,特别是与步骤S100并行执行。

[0064] 因此,从FPGA设计10在步骤S140中基于FPGA设计10产生作为基于图的表示的模型图24。在从FPGA设计10创建模型图24时,FPGA模型10的块被转换为节点并且信号线被转换为指引线(Kante)。示例性地在图8中示出了模型图24,其中,在那里多个独立的模型图24被一起实现并且示出。在此,用于第一快速估算的图形子系统可以被解释为块,或者它们可以任意深度地直至完全地去层次化。转换功能例如可以允许:在考虑到虚拟连接(Goto/From、数据存储读取/数据存储内存/数据存储写入、触发端口、操作端口、...)的情况下对除了FPGA模型的基本块之外的FPGA模型进行去层次化,并且由此产生定向图。在FPGA设计10的去层次化的过程中,考虑虚拟连接,并且应用基于FPGA设计10的去层次化的转换功能。

[0065] 在步骤S150中,将模型图24与来自数据库22的模型图24的相似性进行比较。来自数据库22的模型图24是先前的FPGA实现14的基于图的表示。

[0066] 在此,首先,基于用于图的相似性分析算法来执行当前FPGA设计10的模型图24与来自数据库22的先前的FPGA实现14的模型图24的比较。基于两个模型图24的每两个节点34之间的相似性程度,节点34的相似性程度借助于块/网表路径(也局部地)、块/网表类型(例如,加法、乘法)和块/网表参数/属性(例如位宽,二进制点位置)的一致性程度来确定。

[0067] 该比较还包括线性优化问题的求解,其中,对于模型图24中的每个节点34识别另一模型图24中的最相似的节点34,而不加倍地配设该节点34。这作为按照Kuhn的“匈牙利方法”原则上已知。

[0068] 当前FPGA设计10的模型图24与来自数据库22的FPGA实现14的模型图24的比较还包括邻近匹配,以用于考虑邻近关系。这样的方法例如作为根据Nikolic的“Measuring Similarity of Graph Nodes by Neighbor Matching(通过邻近匹配测量图节点的相似性)”已知。在图9中,在这方面示出了在两个模型图24的节点34之间的两个相似性矩阵。在图9的左侧图片中示出没有“邻近匹配”的相似性矩阵并且在右侧图片中示出具有“邻近匹配”的相似性矩阵。

[0069] 在此,基于区分点的一致性来确定来自数据库22的FPGA实现14的模型图24的节点34与当前的FPGA设计10的模型图24的节点34的相似性。区分点例如包括块/网表路径,块/网表类型例如相加/相乘,块/网表参数例如位宽、二进制点位置。在当前情况下,进行区分点的加权。

[0070] 从针对这两个模型图24的相似性的如此确定的程度出发,现在检查:当前FPGA设计10的模型图24与来自数据库22的FPGA实现14的模型图24的相似性是否高于当前FPGA设计10的模型图24与来自数据库22的最佳模型图24的相似性。来自数据库22的模型图24是第一模型图24,对于该第一模型图已确定与当前的FPGA设计10的模型图24的相似性时,则来自数据库22的模型图24被接受为最佳模型图24或参考26。在其他情况下,仅当来自数据库22的模型图24具有比到目前为止的参考26更高的相似性时才改变参考26。这(如果适用的话)在步骤S160中完成。然后,当在数据库22中还包含另外的用于比较的模型图24时,该方法返回到步骤S150。还检查:来自步骤S100的合成是否已经结束。仅当来自步骤S100的合成

尚未结束时,才在步骤S150中继续比较。

[0071] 因此,当来自步骤S100的合成结束时或者在数据库22中不包含用于比较的另外的模型图24时,将当前的参考26的相似性程度与阈值进行比较。当相似性程度大于阈值时,在步骤S170中提供相应的模型图24作为用于在步骤S110中产生FPGA实现14的参考26。在其他情况下,在步骤S180中没有结果地中止该方法的该部分。阈值在这里示例性地在于90%的相似性。

[0072] 步骤S110中的FPGA实现14的生成如下执行。首先,检查:是否已经由步骤S170确定并提供了基于模型图24的相似FPGA实现14。如果不是这种情况,则在步骤S200中执行常规的传统布置和,以便创建FPGA实现14。否则,该方法以步骤S210继续。

[0073] 在步骤S210中,为所述参考加载来自数据库22的FPGA实现14。然后,在步骤S220中,基于来自数据库22的FPGA实现14进行逐步布置和连接。

[0074] 如图3中所示那样地填充数据库22。在步骤S130中,当前的FPGA设计10的网表12,所属的模型图24和产生的FPGA实现14分别作为FPGA模型10的伪像 (Artefakt) 20存储在数据库22中。因此,每个FPGA设计10自动在数据库22中形成伪像20,用于针对后来新的FPGA设计10的稍后的比较。

[0075] 作为在这里未示出的后台步骤,附加地,计算包含在数据库22中的模型图24之间的关系的相似性。计算包含在数据库22中的伪像20的模型图24之间的关系的相似性的步骤在此可以在使用空闲资源的阶段中完成。从多个FPGA模型10创建相似性图30。不同的相似性线图30在图5至图7中示出。在此,在两个FPGA模型10之间的被查得的相似性作为相应的两个表示FPGA模型10的节点34之间的指引线32给出,并且其中,权重36作为相似性线图程度被给出。相似性图30也称为相似性线图。

[0076] 在此,在图5中的相似性图30仅示出了在几个节点34之间的指引线32,而图6和图7中的相似性图30是完整的相似性图30。基于在节点34之间的已经确定的用实线示出为指引线32的相似性,确定示出为虚线的附加指引线38的权重36。可以以多种方式得到完善。为此,根据暴力法,逐步地通过进一步的相似性分析来补充用于相似性图30的完全互联的附加指引线38的权重36。

[0077] 图7示出一种用于基于FPGA设计10的现有指引线32的有效计算顺序来完善在那里的相似性图30的方案。在此,基于现有指引线32来估计可能存在潜在的高的相似性的两个节点34之间的相似性,以便优先为这些节点执行相似性分析。因此,可以通过将现有指引线32的权重36相乘来估计附加指引线38。这在图7中示例性地通过沿着虚线箭头40将相似性相乘的方式来为在用“2”和“4”表征的节点34之间的附加指引线38进行实现,以便估计这两个节点34之间的相似性。基于此,可以首要确定具有确定的高相似性的指引线32的权重36。

[0078] 基于相似性图30,可以简化和加速步骤S150中的比较。

[0079] 附图标记列表

[0080] FPGA设计 10

[0081] 网表 12

[0082] FPGA实现 14

[0083] 比特流 16

[0084] 伪像 20

- [0085] 数据库 22
- [0086] 模型图 24
- [0087] 参考 26
- [0088] 相似性图,相似性线图 30
- [0089] 指引线 32
- [0090] 节点 34
- [0091] 权重 36
- [0092] 附加指引线 38
- [0093] 虚线箭头 40

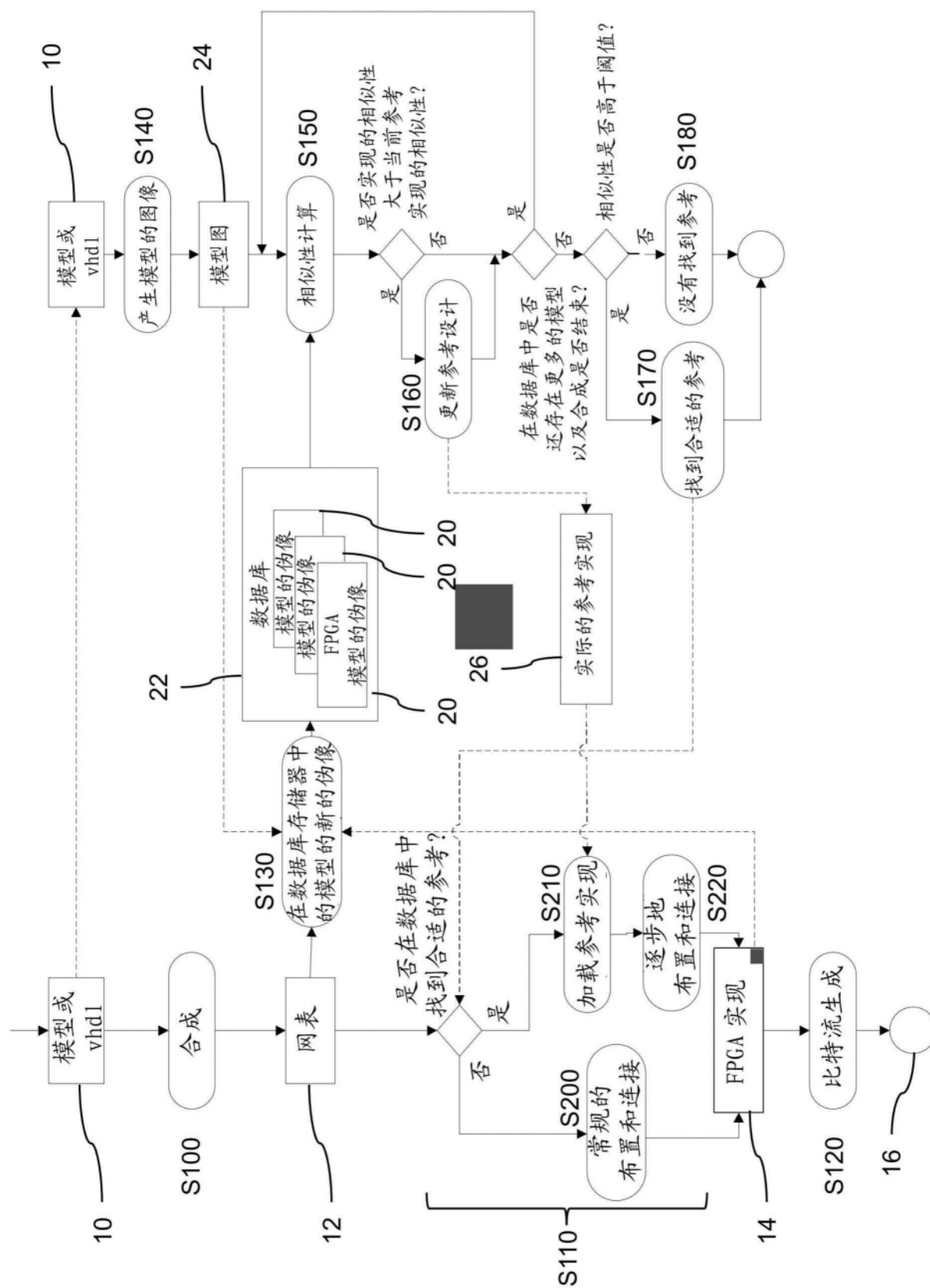


图1

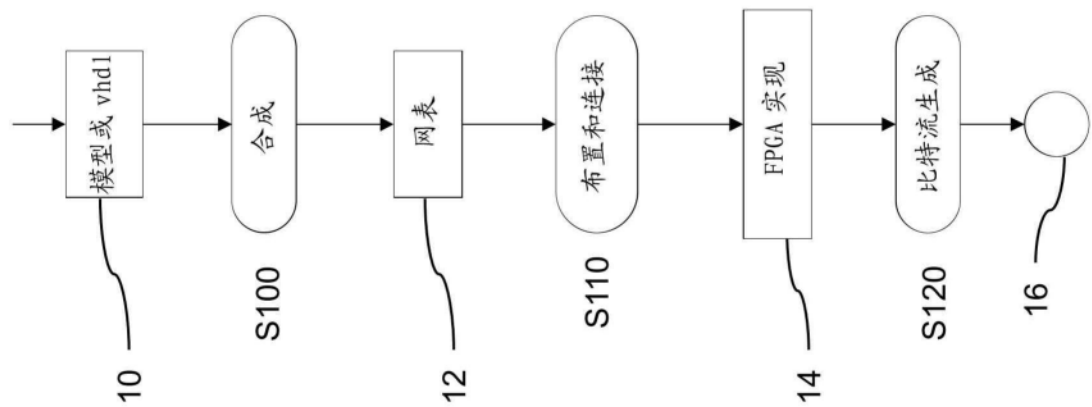


图2

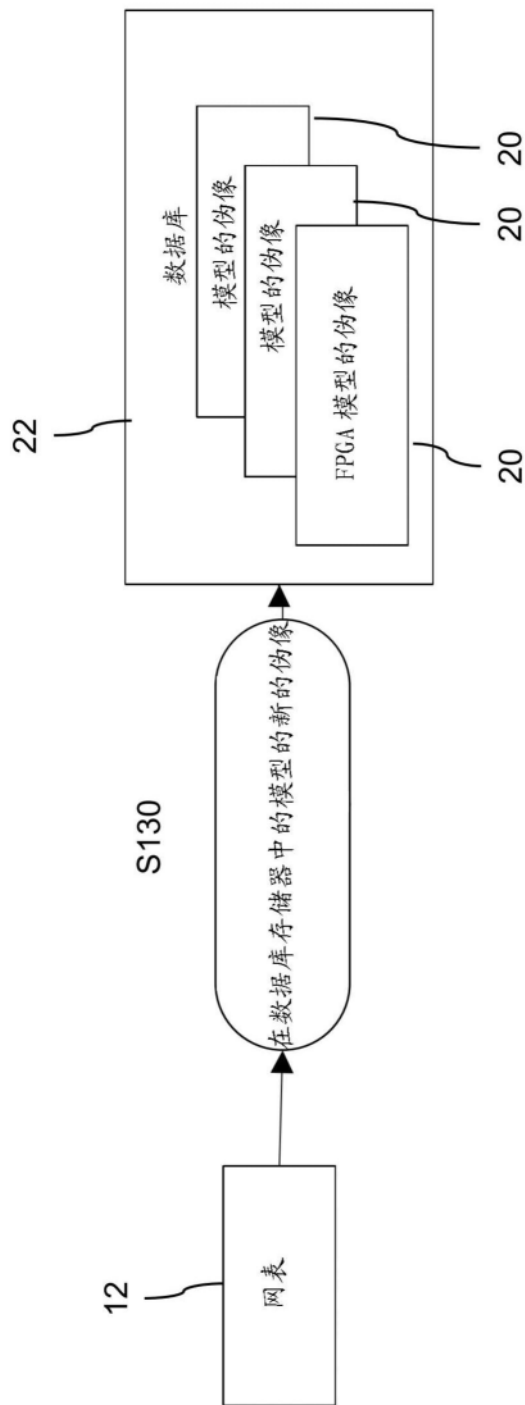


图3

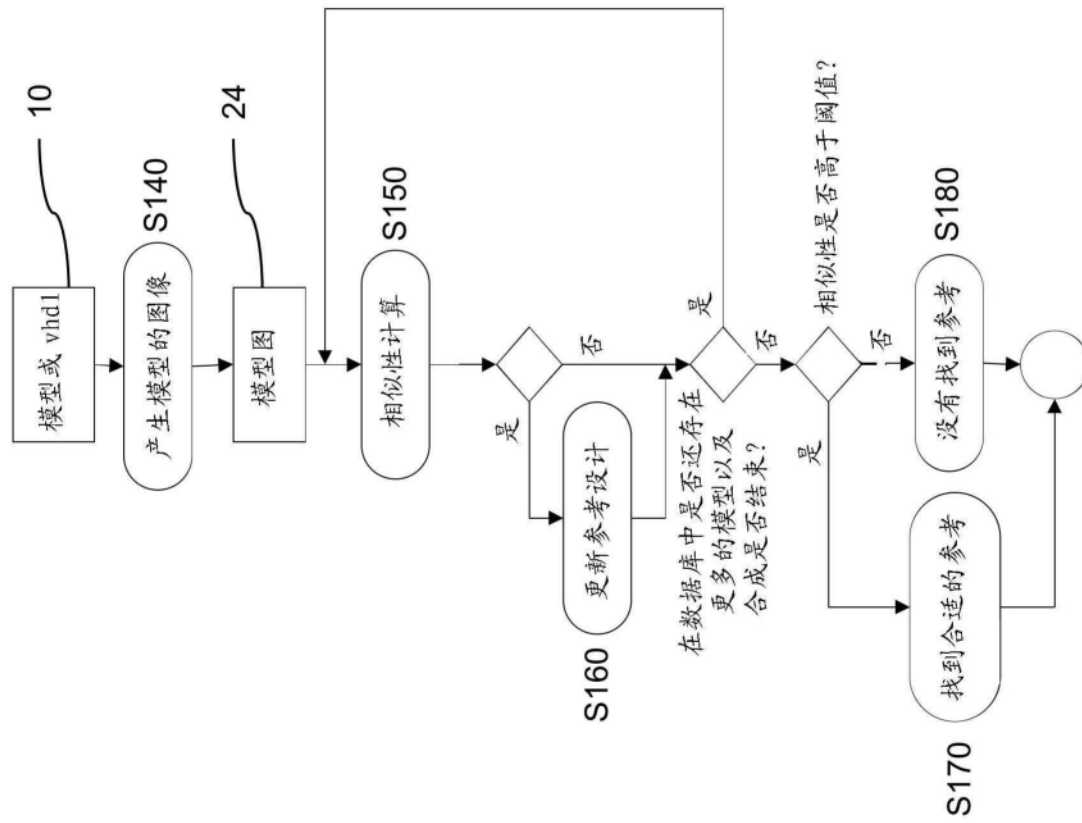


图4

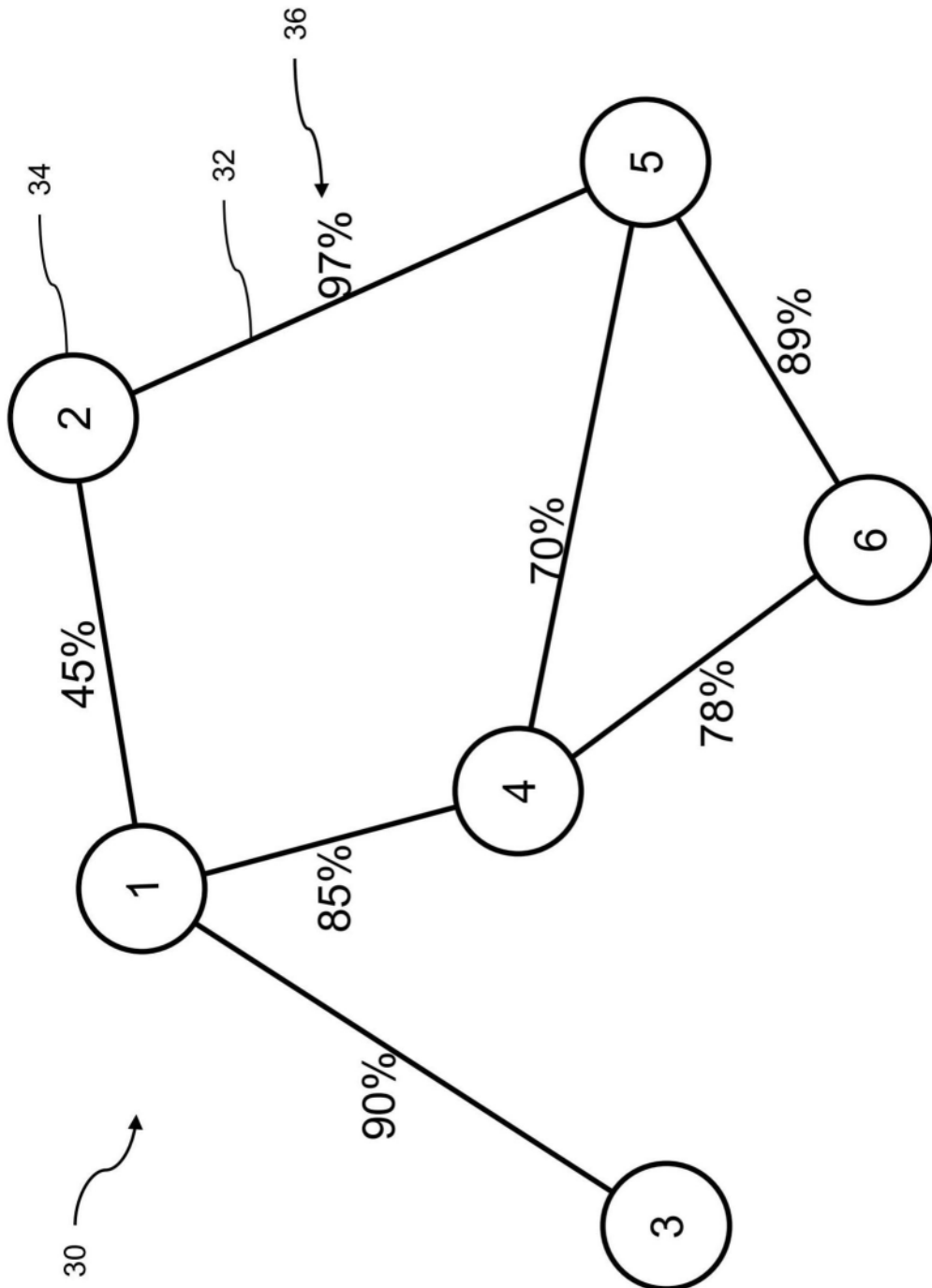


图5

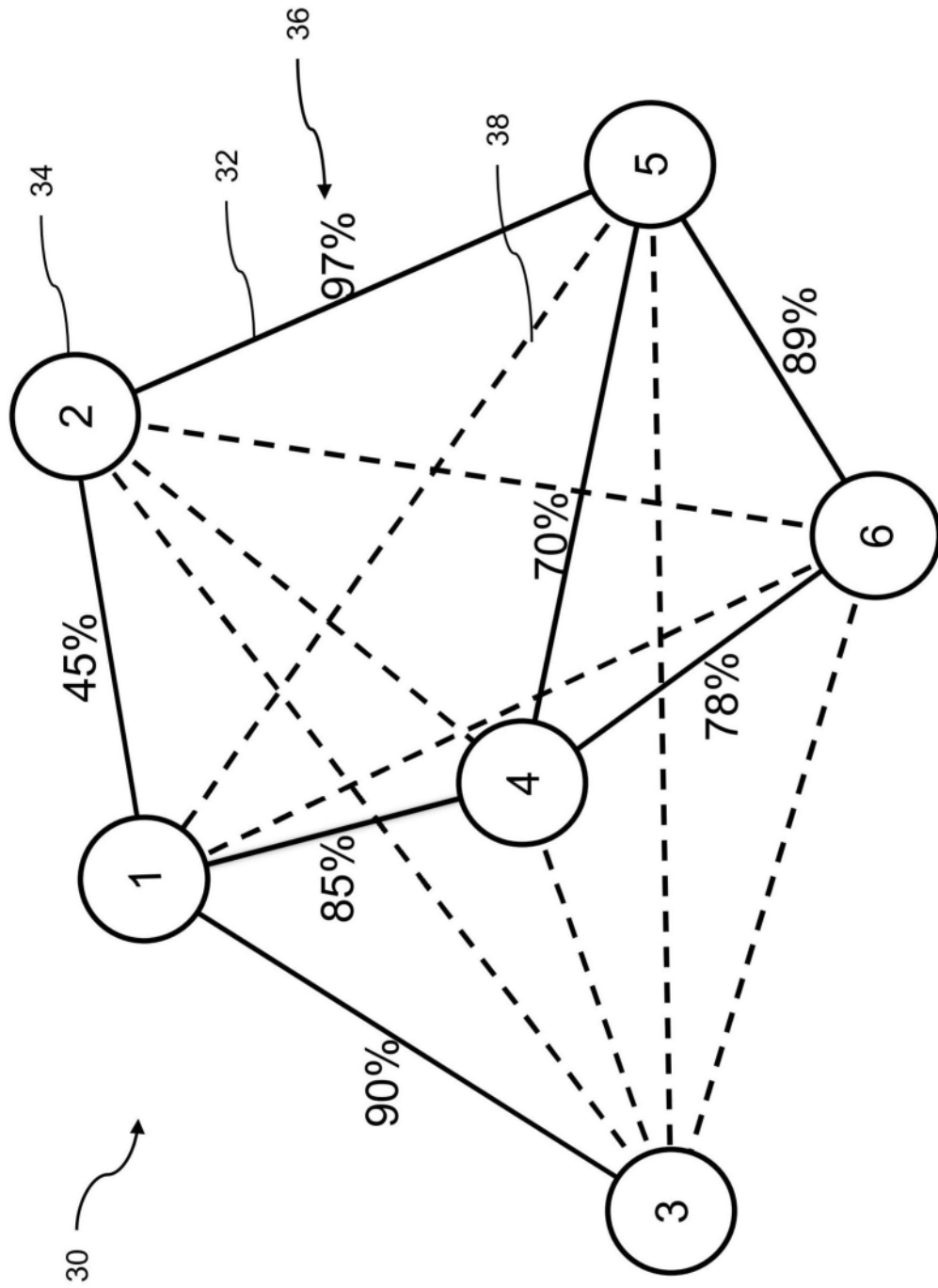


图6

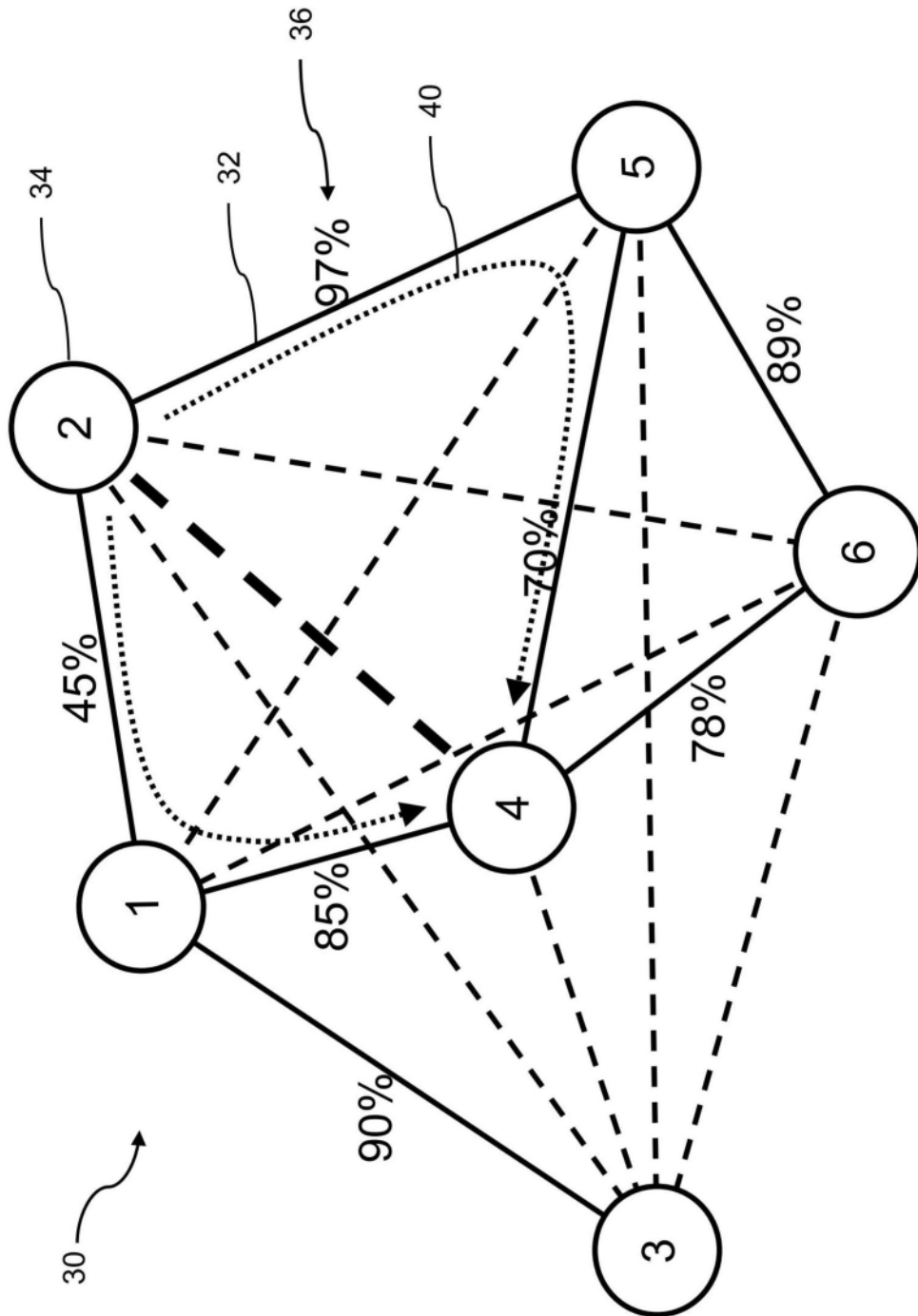


图7

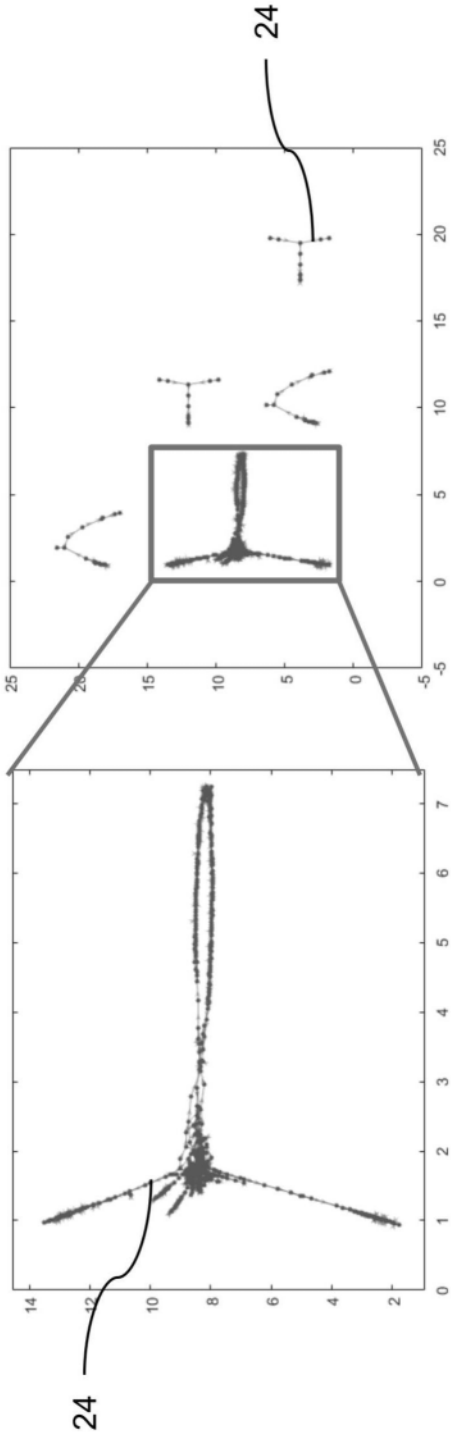


图8

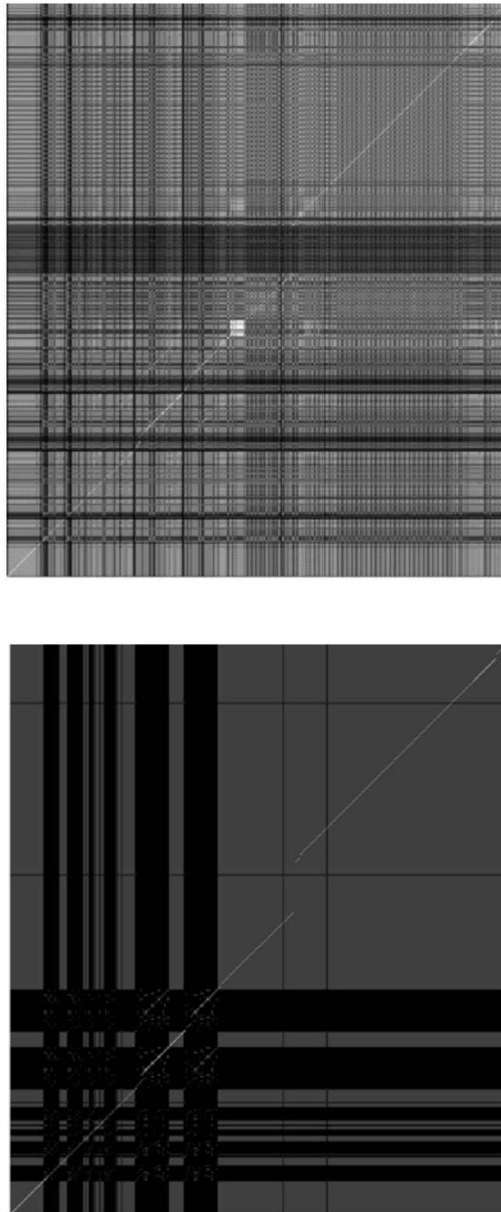


图9