(19)日本国特許庁(JP)

(12)特許公報(B2)

(11)特許番号
特許第7065794号
(P7065794)

(45)発行日　令和4年5月12日(2022.5.12)

(24)登録日　令和4年4月28日(2022.4.28)

(51)国際特許分類

		FI	
Ｇ０６Ｆ	*30/347(2020.01)*	Ｇ０６Ｆ	30/347
Ｇ０６Ｆ	*16/903(2019.01)*	Ｇ０６Ｆ	16/903

請求項の数　16　外国語出願　（全17頁）

(21)出願番号　特願2019-1636(P2019-1636)
(22)出願日　平成31年1月9日(2019.1.9)
(65)公開番号　特開2019-121404(P2019-121404A)
(43)公開日　令和1年7月22日(2019.7.22)
審査請求日　令和3年11月11日(2021.11.11)
(31)優先権主張番号　10 2018 100 423.0
(32)優先日　平成30年1月10日(2018.1.10)
(33)優先権主張国・地域又は機関
ドイツ(DE)
早期審査対象出願

(73)特許権者　506012213
ディスペース ゲー・エム・ベー・ハー
ｄＳＰＡＣＥ ＧｍｂＨ
ドイツ連邦共和国 パデルボルン ラーテナウシュトラーセ ２６
Ｒａｔｈｅｎａｕｓｔｒ．２６，Ｄ－３３１０２ Ｐａｄｅｒｂｏｒｎ，Ｇｅｒｍａｎｙ
(74)代理人　100114890
弁理士　アインゼル・フェリックス＝ラインハルト
(74)代理人　100098501
弁理士　森田 拓
(74)代理人　100116403
弁理士　前川 純一

最終頁に続く

(54)【発明の名称】　グラフベースの類似度検索を用いたＦＰＧＡ実装のインクリメンタル方式による生成

(57)【特許請求の範囲】
【請求項１】
ＦＰＧＡモデルであるＦＰＧＡデザイン（１０）および／またはハードウェア記述に基づき、ＦＰＧＡ実装（１４）を生成する方法であって、前記方法は、
前記ＦＰＧＡデザイン（１０）からネットリスト（１２）を合成するステップと、
前記ＦＰＧＡデザイン（１０）に基づきグラフベース表現を生成するステップと、
類似ＦＰＧＡ実装（１４）を検索するステップと、
前記類似ＦＰＧＡ実装（１４）を使用して、前記ネットリスト（１２）から前記ＦＰＧＡ実装（１４）を生成するステップと、
を含み、
類似ＦＰＧＡ実装（１４）を検索する前記ステップは、前記ＦＰＧＡデザイン（１０）の前記グラフベース表現と、前記類似ＦＰＧＡ実装（１４）のグラフベース表現と、を比較するステップを含む
ことを特徴とする方法。
【請求項２】
前記ＦＰＧＡデザイン（１０）は、部分実装を含み、
類似ＦＰＧＡ実装（１４）を検索する前記ステップは、類似部分実装の検索を含み、
前記類似部分実装を使用して、前記ＦＰＧＡ実装（１４）を生成する前記ステップを行う、
請求項１記載の方法。
【請求項３】

類似ＦＰＧＡ実装（１４）を検索する前記ステップは、以前のＦＰＧＡ実装（１４）のデータベース（２２）内の類似ＦＰＧＡ実装（１４）の検索を含む、
請求項１または２記載の方法。
【請求項４】
前記方法は、前記ＦＰＧＡデザイン（１０）の変更された部分領域を識別するステップをさらに含み、
前記ネットリスト（１２）を合成する前記ステップは、前記ＦＰＧＡデザイン（１０）の前記変更された部分領域に対する前記ネットリスト（１２）の合成を含み、
ＦＰＧＡ実装（１４）を生成する前記ステップは、前記ＦＰＧＡデザイン（１０）の前記変更された部分領域の前記ネットリスト（１２）に対する前記ＦＰＧＡ実装（１４）の生成を含む、
請求項１から３までのいずれか１項記載の方法。
【請求項５】
前記方法は、
前記ＦＰＧＡデザイン（１０）の前記グラフベース表現と、少なくとも１つの前記類似ＦＰＧＡ実装（１４）の前記グラフベース表現と、の間の類似度の大きさを特定するステップと、
前記類似度の大きさが限界値を下回っているならば、類似ＦＰＧＡ実装（１４）を使用せずに、前記ネットリスト（１２）から前記ＦＰＧＡ実装（１４）を生成するステップと、をさらに含む、
請求項１から４までのいずれか１項記載の方法。
【請求項６】
前記ＦＰＧＡデザイン（１０）の前記グラフベース表現と、前記類似ＦＰＧＡ実装（１４）の前記グラフベース表現と、を比較する前記ステップは、線形最適化問題の解決を含み、前記解決にあたり、１つのノードを二重に割り当てることなく、一方のグラフ内の各ノードに対し他方のグラフ内の最類似ノードを識別する、
請求項１から５までのいずれか１項記載の方法。
【請求項７】
前記方法は、前記ＦＰＧＡデザイン（１０）の前記グラフベース表現のノードと、少なくとも１つの前記類似ＦＰＧＡ実装（１４）の前記グラフベース表現のノードと、の類似度を、細分化点の一致の検出によって特定するステップをさらに含む、
請求項１から６までのいずれか１項記載の方法。
【請求項８】
細分化点の一致を検出する前記ステップは、前記細分化点の重み付けを含む、
請求項７記載の方法。
【請求項９】
前記方法は、
データベース（２２）内に含まれている複数のグラフ（２４）の類似度分析を実施する付加的なステップと、
算出された類似度を類似度グラフ（３０）または類似度マトリックスに記憶するステップと、
をさらに含む、
請求項１から８までのいずれか１項記載の方法。
【請求項１０】
前記方法は、前記データベース（２２）内に含まれる前記複数のグラフ（２４）間の関係性類似度を算出するステップをさらに含む、
請求項９記載の方法。
【請求項１１】
前記方法は、すでに特定された類似度に基づき、未特定の関係性について予期される高い類似度を事前に求めるステップをさらに含む、

請求項１から１０までのいずれか１項記載の方法。
【請求項１２】
前記ＦＰＧＡデザイン（１０）の前記グラフベース表現と、前記類似ＦＰＧＡ実装（１４）のグラフベース表現と、を比較する前記ステップは、近傍関係を考慮する近傍マッチングを含む、
請求項１から１１までのいずれか１項記載の方法。
【請求項１３】
前記ＦＰＧＡデザイン（１０）に基づきグラフベース表現を生成する前記ステップは、仮想接続を考慮しながら前記ＦＰＧＡデザイン（１０）を非階層化するステップと、前記ＦＰＧＡデザイン（１０）の前記非階層化に基づき変換関数を適用するステップと、を含む、
請求項１から１２までのいずれか１項記載の方法。
【請求項１４】
前記ＦＰＧＡデザイン（１０）から前記ネットリスト（１２）を合成する前記ステップと、前記ＦＰＧＡデザイン（１０）に基づき前記グラフベース表現を生成する前記ステップと、を並列に実行する、
請求項１から１３までのいずれか１項記載の方法。
【請求項１５】
前記ＦＰＧＡデザイン（１０）の前記グラフベース表現と、少なくとも１つの前記類似ＦＰＧＡ実装（１４）のグラフベース表現と、を比較するステップは、グラフを類似度分析するアルゴリズムに基づき行われる、
請求項１から１４までのいずれか１項記載の方法。
【請求項１６】
ＦＰＧＡモデルであるＦＰＧＡデザイン（１０）および／またはハードウェア記述に基づきビットストリーム（１６）を生成する方法であって、前記方法は、
請求項１から１５までのいずれか１項に従い、ＦＰＧＡモデルであるＦＰＧＡデザイン（１０）および／またはハードウェア記述に基づきＦＰＧＡ実装（１４）を生成するステップと、
生成された前記ＦＰＧＡ実装（１４）からビットストリーム（１６）を生成するステップと、
を含む方法。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、ＦＰＧＡモデルであるＦＰＧＡデザインおよび／またはハードウェア記述に基づき、ＦＰＧＡ実装を生成する方法に関する。この方法は、ＦＰＧＡデザインからネットリストを合成するステップと、このネットリストからＦＰＧＡ実装を生成するステップと、を含む。ただしこの方法は類似ＦＰＧＡ実装の検索を含み、ネットリストからＦＰＧＡ実装を生成するステップは、この類似ＦＰＧＡ実装を使用しながら行われる。
【０００２】
さらに本発明は、ＦＰＧＡモデルであるＦＰＧＡデザインおよび／またはハードウェア記述に基づきビットストリームを生成する方法に関する。この方法は、ＦＰＧＡモデルであるＦＰＧＡデザインおよび／またはハードウェア記述に基づきＦＰＧＡ実装を生成する上述のステップと、生成されたＦＰＧＡ実装からビットストリームを生成することとを含む。
【背景技術】
【０００３】
かかる方法は、ＦＰＧＡ（Field Programmable Gate Array）のためのコードを生成する目的で使用される。ＦＰＧＡは、論理回路をロード可能な集積回路である。その際にプログラミングには、公知のプログラミング言語による古典的なプログラミングとは対照的に時系列の設定は僅かであり、それとは異なり所望の回路構造の定義が含まれている。
【０００４】

この場合、所望の回路構造を定義するためのＦＰＧＡデザインは、一般にグラフィックＦＰＧＡモデルとして生成され、これが例えばＶＨＤＬなどのようなハードウェア記述言語に書き換えられる。基本的にはＦＰＧＡデザインを直接、ハードウェア記述言語で生成することもできる。

【０００５】

ＦＰＧＡデザインは、次いで合成ステップにおいて相応のソフトウェアにより最初に、ＦＰＧＡにおいて個々の論理素子をどのように結線すべきかを設定するネットリストに書き換えられる。したがってこれは、ＦＰＧＡのコンフィギュレーションとも呼ばれる。論理素子およびそれらの結線を含むネットリストは、あとで利用するために記憶される。

【０００６】

ＦＰＧＡ実装を生成するステップがこれに続く。このステップは、一般的に「配置配線」（"Place and Route"）と呼ばれる。この場合、ＦＰＧＡのためにネットリストに記憶されている論理素子が配置され（Place）、ネットリストに従って互いに配線される（Route）。その成果物がＦＰＧＡ実装である。

【０００７】

次いでＦＰＧＡ実装からビットストリームが生成され、ＦＰＧＡに所望の機能を実装する目的でこのＦＰＧＡに伝送される。

【０００８】

これまでに挙げたステップは、しばしば一括りに「ビルド」（"Build"）と呼ばれる。

【０００９】

その際に今日では簡単に取り扱えるツールが存在しており、それらのツールによって例えば、抽象化を行うＦＰＧＡ開発環境が顧客に提供され、その結果として顧客は、ＦＰＧＡおよびツールフローに関する詳細な知識がなくても、自身のハードウェアを開発できるようになる。これによって顧客は、ＦＰＧＡデザインを作成して実装し、相応のＦＰＧＡハードウェアであるＦＰＧＡにおいて駆動することができる。例えばラピッドコントロールプロトタイピングなどの場合であると、ＦＰＧＡモデルにおける規則的な変更およびそれに続くリビルドは、かかる開発プロセスの通常の構成要素である。

【００１０】

その際に問題となるのは、ビルド時間がしばしば著しく長くなることであり、このように著しく長いビルド時間は、ＦＰＧＡモデルをごく僅かにしか変更しなくても、ＦＰＧＡのリビルドのために必要とされる場合もある。この場合、ビルド時間はしばしば数時間という範囲に及んでしまう。ＦＰＧＡはいっそう複雑になり続け、レジスタ、ロジック、ＤＳＰブロックまたはメモリの形態でいっそう多くのリソースを提供するので、ビルド時間はさらにいっそう長くなる。このことは、アルゴリズムを改善しビルドを実施するコンピュータを高速化しても、部分的にしか相殺できない。その際に特に時間のかかるステップは、たいていは配置配線である。

【００１１】

目下のＦＰＧＡツールには、ビルド時間を短縮するために２つのアプローチがある。両方のアプローチとも、すでに生成に成功しているＦＰＧＡ実装の非変更の領域／構成要素を再利用する、ということに基づいている。１つめのアプローチによれば、各構成要素にＦＰＧＡ内の１つの（矩形）領域が割り当てられる。ある構成要素が変更されると、その構成要素を含む領域が新たに実装される一方、ＦＰＧＡ実装の残りの部分は変更されないまま維持されて、再利用することができる。ここではインクリメンタルビルドと称する２つめのアプローチによれば、事前定義された領域は必要とされない。実装にあたり、すでに実装されたＦＰＧＡ実装が参照として指定される。一致部分が大きければ、多くの配置配線を自動的に引き継ぐことができ、ＦＰＧＡ実装の生成にかかる時間が短縮される。

【００１２】

この場合、当初からすでに、個々のモジュールもしくは部分実装を再利用できるようにＦＰＧＡ実装を生成すると、有利なものとなり得る。このアプローチは、ＦＰＧＡモデルの自動モジュール化を追求するものである。このようなモジュールベースにおいて、フロア

プラニングすなわち複数の領域への分割を行うことができる。個々のモジュールについてさらにリザーブも設けられているならば、ＦＰＧＡモデルの変更を個々のモジュールに限定したままにすることができる。

【００１３】

さらに、再プラニングのための方法が知られている。顧客が自身のＦＰＧＡモデルに対し変更を加えた場合に、直前のビルド結果が使用される。これに基づき、変更されたモジュールが個別にビルドされ、ついでそれらが全体的なビルドにおいて取り替えられる。複数のモジュールが該当するケースであれば、変更されたそれらのモジュールのためのビルドをそれぞれ異なるコンピュータにおいて行えば、それによってビルド時間を短縮することができる。つまり、変更された複数のモジュールの並列なビルドが実施される。この場合、再利用を簡単にする目的で、様々なバージョンのモジュールのバージョニングを設けることができる。

【００１４】

これらの方法によってビルド時間をすでに短縮できるけれども、さらに改善の余地がある。つまり例えば細粒度のモジュール化の場合には、無駄な部分が大きくなり、また、タイミングが損なわれるリスクが高まる。なぜならば、モジュール境界を越えて最適化することはできず、それらのモジュールのためにそれ相応のリザーブを含む広めの領域をそれぞれ設けなければならないからである。しかもこの方法は、製品開発において多大な実装コストをかけることによってしか実現できない。

【００１５】

２つのデザインの類似度を突き止める目的で、目下の合成ツールによれば、参照ネットリストが新たに配置配線すべきネットリストと比較される。その際にこの比較はもっぱら、ネットリストパス名の名前の比較に基づいている。このことは問題を孕むものであり、その理由は、Xilinx Vivado System Generator, Xilinx Vivado HLS, Xilinx Vivado SDSoCといった抽象化を行うハイレベルＦＰＧＡデザインツールの場合には、例えばサブシステムなどにおいて、同一のＦＰＧＡ構成要素が即座に名称変更および再編成されるのが一般的だからである。これによって、本来は同一である構成要素についても新たなネットリストパスが自動的に発生する。このような名称変更を、目下の合成ツールでは識別することができない。

【００１６】

したがって上述の従来技術から出発して本発明の基礎とする課題は、ＦＰＧＡモデルであるＦＰＧＡデザインおよび／またはハードウェア記述に基づきＦＰＧＡ実装を生成する方法、ならびにＦＰＧＡモデルであるＦＰＧＡデザインおよび／または上述の形式のハードウェア記述に基づきビットストリームを生成する方法において、特にすでに存在している類似ＦＰＧＡ実装を使用しながら、ＦＰＧＡ実装の簡単かつ効率的な生成を可能にすることである。

【発明の概要】

【発明が解決しようとする課題】

【００１７】

本発明によればこの課題は、独立請求項記載の構成要件により解決される。従属請求項には本発明の有利な構成が記載されている。

【課題を解決するための手段】

【００１８】

したがって本発明によれば、ＦＰＧＡモデルであるＦＰＧＡデザインおよび／またはハードウェア記述に基づき、ＦＰＧＡ実装を生成する方法が設けられており、この方法は、ＦＰＧＡデザインからネットリストを合成するステップと、このネットリストからＦＰＧＡ実装を生成するステップと、を含み、ただしこの方法は類似ＦＰＧＡ実装の検索を含み、この類似ＦＰＧＡ実装を使用して、ネットリストからＦＰＧＡ実装を生成するステップを行う。さらにこの方法は、ＦＰＧＡデザインに基づきグラフベース表現を生成するステップを含み、類似ＦＰＧＡ実装を検索するステップは、ＦＰＧＡデザインのグラフベース表

現を、少なくとも１つの類似ＦＰＧＡ実装のグラフベース表現と比較することを含む。
【００１９】
本発明によればさらに、ＦＰＧＡモデルであるＦＰＧＡデザインおよび／またはハードウェア記述に基づきビットストリームを生成する方法が設けられており、この方法は、ＦＰＧＡモデルであるＦＰＧＡデザインおよび／またはハードウェア記述に基づきＦＰＧＡ実装を生成する上述のステップと、生成されたＦＰＧＡ実装からビットストリームを生成することとを含む。
【００２０】
つまり本発明の基本的な着想は、目下のＦＰＧＡモデルと他のＦＰＧＡモデル例えば以前のＦＰＧＡモデルとの類似性を見つけ出し、ＦＰＧＡ実装の効率的な生成を実施するために使用する、ということである。これに基づき、少なくとも１つの存在する類似ＦＰＧＡ実装をベースとする目下のＦＰＧＡ実装のインクリメンタル方式による生成が実施される。次いで、目下のＦＰＧＡ実装を生成するために、類似ＦＰＧＡ実装またはそれらのうち少なくとも類似部分が使用される。一般的に、ＦＰＧＡ実装を生成するためのＦＰＧＡロジックの配置配線にＦＰＧＡビルドの大半の時間が費やされるので、これによってＦＰＧＡビルドの生成にかかる時間の短縮を達成することができる。
【００２１】
ＦＰＧＡデザインは、一般にグラフィックＦＰＧＡモデルとして生成され、このモデルがハードウェア記述言語に書き換えられる。基本的にはＦＰＧＡデザインを直接、ハードウェア記述言語で生成することもできる。ハードウェア記述言語として、例えばＶＨＤＬが知られている。ＶＨＤＬは、超高速集積回路ハードウェア記述言語として知られている（ＶＨＳＩＣハードウェア記述言語とも称する）。
【００２２】
ＦＰＧＡデザインは、次いで合成ステップにおいて相応のソフトウェアにより最初に、ＦＰＧＡにおいて個々の論理素子をどのように結線すべきかを設定するネットリストに書き換えられる。したがってこれは、ＦＰＧＡのコンフィギュレーションとも呼ばれる。論理素子およびそれらの結線を含むネットリストは、あとで利用するために記憶される。
【００２３】
ＦＰＧＡ実装を生成するステップがこれに続く。このステップは、一般的に「配置配線」（"Place and Route"）と呼ばれる。この場合、ＦＰＧＡのためにネットリストに記憶されている論理素子が配置され（Place）、ネットリストに従い互いに配線される（Route）。その成果物がＦＰＧＡ実装である。
【００２４】
次いでＦＰＧＡ実装からビットストリームが生成され、ＦＰＧＡに所望の機能を実装する目的でこのＦＰＧＡに伝送される。
【００２５】
これまでに挙げたステップは、しばしば一括りに「ビルド」（"Build"）と呼ばれる。
【００２６】
好ましくは、合成されたネットリストが個々のＦＰＧＡモデルのアーチファクトとしてデータベース内に保管され、その目的は、以降のビルドにあたり対応するＦＰＧＡ実装を、必要に応じて類似ＦＰＧＡ実装を検索するための参照として、利用できるようにするためである。すでに合成中に、参照としてできるかぎり類似したＦＰＧＡ実装を検索することもできる。この目的で最初に、ＦＰＧＡデザインに基づきグラフベース表現が作成され、すでに事前に実装されたＦＰＧＡデザインのグラフと比較される。
【００２７】
その際、少なくとも１つの類似ＦＰＧＡ実装のグラフの作成は一般に、類似ＦＰＧＡ実装の対応するビルド中に行われる。つまりこのことは、類似ＦＰＧＡ実装は慣用のように生成され、または対応するＦＰＧＡデザインから出発して本明細書に記載された方法に従っても生成される、ということを意味する。次いでこれらのＦＰＧＡ実装のグラフが、やはり個々のＦＰＧＡデザインに基づき生成される。

【００２８】
全体として、この方法を簡単に具現化することができ、その際に様々なステップを並列に実行することができる。したがってリソースを効率的に活用することができる。その際、類似ＦＰＧＡ実装をＦＰＧＡデザインのためのベースとして使用する目的で、最類似ＦＰＧＡ実装を選択することができ、または最小類似度が前提とされる。十分な類似性があるならば、対応するＦＰＧＡ実装を参照として使用することができる。まだ類似度について調査されていないさらに別のＦＰＧＡデザインが存在するならば、それらのＦＰＧＡデザインについて類似ＦＰＧＡ実装または最類似ＦＰＧＡ実装の検索を続けることができる。
【００２９】
ＦＰＧＡデザインからグラフを作成する際に、ノードおよび信号ラインに対するＦＰＧＡモデルのブロックがエッジに変換される。この方法を階層的に適用することができる。グラフィックのサブシステムを、最初の高速な推定についてブロックとして解釈することができ、または完全に行われた状態に至るまで任意の深さで非階層化することができる。変換関数によって例えば、ＦＰＧＡモデルをとりわけそれらのベースブロックに至るまで、仮想接続（Goto/From, DataStoreRead/DataStoreMemory/DataStoreWrite, Triggerport, Actionport等）を考慮しながら非階層化し、そこから有向グラフを生成することができる。その際にＦＰＧＡモデルは、１つのつながったグラフから成るものとすることができるだけでなく、複数の別個のグラフを含むことができる。かかる別個のグラフが発生するのは例えば、１つのＦＰＧＡデザイン内で互いに独立した複数の部分機能が実装される場合である。
【００３０】
ＦＰＧＡデザインのグラフベース表現と少なくとも１つの類似ＦＰＧＡ実装のグラフベース表現との比較は、グラフを類似度分析するアルゴリズムに基づき行われる。２つのグラフのそれぞれ２つのノード間の類似度の大きさに基づき、公知のアルゴリズムを適用することができる。２つのノードの類似度の大きさが、ブロックパス／ネットリストパス（部分的にも）、ブロックタイプ／ネットリストタイプ（例えば加算、乗算）およびブロックパラメータ／ネットリストパラメータ／ネットリスト属性（例えばビット幅、２進小数点位置）の一致の程度に基づき特定される。
【００３１】
本発明の有利な実施形態によれば、ＦＰＧＡデザインは部分実装を含み、類似ＦＰＧＡ実装の検索は類似部分実装の検索を含み、この類似部分実装を使用してＦＰＧＡ実装を生成するステップが行われる。したがって類似度検索の問題を、これらの部分実装に関する類似度を求めて比較することによって、種々の部分問題に分けることができる。
【００３２】
本発明の有利な実施形態によれば、類似ＦＰＧＡ実装の検索は、以前のＦＰＧＡ実装のデータベース内の類似ＦＰＧＡ実装の検索を含む。データベース内に、例えばすべてのＦＰＧＡ実装を記憶させることができ、その目的は、これらのＦＰＧＡ実装をあとで使用するために用意しておくことである。その際にデータベースは好ましくは、１つのプロジェクトの枠内で生成されたＦＰＧＡ実装を含んでおり、それというのもこのようにすれば、類似ＦＰＧＡ実装について高められた確率から出発できるからである。よって、類似ＦＰＧＡ実装の検索をその範囲に限定することができる。
【００３３】
本発明の有利な実施形態によればこの方法は、ＦＰＧＡデザインの変更された部分領域を識別するステップを含み、ネットリストの合成は、ＦＰＧＡデザインの変更された部分領域からのネットリストの合成を含み、ＦＰＧＡ実装の生成は、ＦＰＧＡデザインの変更された部分領域のネットリストに対するＦＰＧＡ実装の生成を含む。したがって新たなＦＰＧＡ実装の作成を、最初からＦＰＧＡデザインの変更された部分領域に限定することができる。好ましくは、ＦＰＧＡ実装を個別に新たに実施可能なＦＰＧＡデザインのブロックが考察される。
【００３４】

本発明の有利な実施形態によればこの方法は、ＦＰＧＡデザインのグラフベース表現と、少なくとも１つの類似ＦＰＧＡ実装のグラフベース表現との間の類似度の大きさを特定するステップを含み、さらにこの方法は、類似度の大きさが限界値を下回っているならば、類似ＦＰＧＡ実装を使用せずに、ネットリストからＦＰＧＡ実装を生成する慣用のステップを含む。つまり適切な類似ＦＰＧＡ実装が参照として見つけられたならば、目下のＦＰＧＡ実装の生成中にそれを利用することができる。そうでなければ、ＦＰＧＡモデルをまったく新たに実装する必要がある。限界値よりも大きい類似度の大きさをもつ有用なＦＰＧＡ実装が参照として存在しないことが、いっそう早く事前に認識されればされるほど、類似ＦＰＧＡ実装を検索するためにいっそう僅かなリソースしか必要とされない。このことから、ＦＰＧＡモデルが著しく異なっているのであれば、つまり類似度の大きさが限界値を下回っている場合には、インクリメンタルビルドよりも通常のビルドを用いることで、ビルドにかかる時間を短縮することができる。好ましくは限界値は７５％の類似度のところにある。さらに好ましくは限界値は８５％の類似度のところにあり、さらにいっそう好ましくは９０％のところにある。

【００３５】

本発明の有利な実施形態によれば、ＦＰＧＡデザインのグラフベース表現を少なくとも１つの類似ＦＰＧＡ実装のグラフベース表現と比較するステップは、線形最適化問題の解決を含み、この解決にあたり、１つのノードを二重に割り当てることなく、一方のグラフ内の各ノードに対し他方のグラフ内の最類似ノードが識別される。この場合、Kuhnによる「ハンガリー法」として知られた方法が適用される。ハンガリー法は線形最適化問題の解決にあたり、１つのノードに二重に割り当てることなく、一方のグラフの各ノードに対し他方のグラフにおける最類似ノードを見つけるために用いられる。この方法によって総類似度が最大化される。ただし近傍関係を、２つのブロックの類似度に関する大きさとして一緒に算入することができる。この方法は基本的には、近傍関係を考慮せずに実施される。

【００３６】

本発明の有利な実施形態によればこの方法は、ＦＰＧＡデザインのグラフベース表現のノードと、少なくとも１つの類似ＦＰＧＡ実装のグラフベース表現のノードと、の類似度を、細分化点の一致の検出によって特定するステップを含む。細分化点には例えば、ブロックパス／ネットリストパス（部分的にも）、ブロックタイプ／ネットリストタイプ例えば加算／乗算、ブロックパラメータ／ネットリストパラメータ例えばビット幅、２進小数点位置が含まれる。

【００３７】

本発明の有利な実施形態によれば、細分化点の一致を検出するステップは細分化点の重み付けを含む。この重み付けによって、細分化点を異なる強さで考慮することができる。

【００３８】

本発明の有利な実施形態によればこの方法は、データベース内に含まれている複数のグラフの類似度分析を実施する付加的なステップと、算出された類似度を類似度グラフまたは類似度マトリックスに記憶することを含む。したがって類似度分析において見つけ出されるかかる類似度を記憶することができ、その目的は、類似ＦＰＧＡ実装の以降の検索を、ＦＰＧＡデザインのグラフベース表現と少なくとも１つの類似ＦＰＧＡ実装のグラフベース表現との比較によって簡単にすることである。算出された類似度はここでは、類似ＦＰＧＡ実装の検索において求められる類似度に該当する。

【００３９】

本発明の有利な実施形態によればこの方法は、データベース内に含まれる複数のグラフ間の関係性類似度を算出する付加的なステップを含む。したがって例えば、グラフまたは部分グラフを事前に分析し、それらの構造を捕捉して記憶することで、一種の「ハッシング」を実施することができる。この構造を例えば、いわゆるハッシュ値として記憶することができる。したがってグラフの類似度を以降で比較する際に、算出された関係性類似度に基づきこの比較を実施することができ、これによって、ＦＰＧＡデザインのグラフベース表現と少なくとも１つの類似ＦＰＧＡ実装のグラフベース表現との比較による類似ＦＰＧ

Ａ実装の検索が、この構造を通して簡単にされる。データベース内に含まれるグラフ間の関係性類似度を算出するステップには多大な計算コストが付随し、空きリソースを利用できるフェーズ中に行うのが望ましく、例えば僅かな計算負荷でＦＰＧＡデザインが生成される開発者のモデリングフェーズ中に行うのが望ましい。この場合に好ましくは、複数のＦＰＧＡモデルから１つの類似度グラフが作成され、このグラフによって、互いに比較されたすべてのＦＰＧＡモデル間の類似度分析の結果が記憶される。この場合、２つのＦＰＧＡモデル間で調査された類似度が、対応する２つのノード間のエッジとして、例えば類似度の大きさである重みと共に表されている。完全にメッシュ化された類似度グラフの場合には、ＦＰＧＡモデル各々と別のＦＰＧＡモデル各々との類似度が表されている。完全な類似度グラフによれば、インクリメンタルＦＰＧＡビルドにとって理想的な開始点を見つけることができる。様々な手法によって、そのように完全なものとすることができる。第１のバリエーションとして総当たり方式に従い、類似度グラフを完全にメッシュ化するために欠けているエッジを、さらなる類似度分析を通して補うことができる。この結果として、すでに述べたように多大な計算能力が要求されることとなり、したがって例えばＦＰＧＡデザイン作成中、ＣＰＵ負荷が僅かなときに例えばバックグラウンドで行うことができる。第２のバリエーションは、存在するＦＰＧＡ実装の効率的な計算順序に基づくものである。これによれば、場合によっては高い類似度が生じる可能性をもって存在しているエッジに基づき、２つのノード間の類似度が推定され、その目的はそれらについて優先的に類似度分析を実施するためである。これによって、類似度グラフのメッシュ化をさらに完全なものとすることができる。その際に例えば確率のアプローチを適用することができる。これによれば確率と同様に類似度が、存在する複数のエッジを経由する１つの経路に沿って乗算される：経路２－１－４＝４５％×８５％＝３８％。第３のバリエーションによれば２つのノード間の類似度が、存在するエッジに基づきバリエーション２のアプローチによって推定される。ＦＰＧＡデザインの正確な類似度は、このアプローチの場合には分析されない。

【００４０】

本発明の有利な実施形態によればこの方法は、すでに特定された類似度に基づき、未特定の関係性について予期される高い類似度を事前に求める付加的なステップを含む。いずれの類似ＦＰＧＡ実装を参照として利用できるのか、または類似ＦＰＧＡを参照として利用できるのか否かについて、いっそう早く事前に識別できればできるほど、この方法をいっそう効率的に実施することができる。

【００４１】

本発明の有利な実施形態によれば、ＦＰＧＡデザインのグラフベース表現と少なくとも１つの類似ＦＰＧＡ実装のグラフベース表現とを比較するステップは、近傍関係を考慮する近傍マッチングを含む。かかる方法は例えば、Nikolicによる"Measuring Similarity of Graph Nodes by Neighbor Matching"として知られている。これによれば類似度が２つのグラフ間の類似度として求められ、これは最適化の最適なノード対応関係についてのノード類似度の総計をノード数で除算することによって得られる。

【００４２】

本発明の有利な実施形態によれば、ＦＰＧＡデザインに基づきグラフベース表現を生成するステップは、仮想接続を考慮しながらＦＰＧＡデザインを非階層化すること、およびＦＰＧＡデザインの非階層化に基づき変換関数を適用することを含む。階層型のサブシステムの場合にはそれらのサブシステムを、最初の高速な推定についてブロックとして解釈することができ、または完全に行われた状態に至るまで任意の深さで非階層化することができる。仮想接続および階層型のブロックの依存性を捕捉し、かかるブロックをＦＰＧＡ実装を作成するためのベースとして適用するときに正しく使用する目的で、変換が必要である。仮想接続は例えば、Goto/From, DataStoreRead/DataStoreMemory/DataStoreWrite, Triggerport, Actionport等といった接続を含むことができる。

【００４３】

次に、添付の図面を参照しながら好ましい実施形態に基づき、本発明について詳しく説明

する。
【図面の簡単な説明】
【００４４】
【図１】第１の好ましい実施形態に従いＦＰＧＡデザインに基づき、ＦＰＧＡ実装およびこのＦＰＧＡ実装からのビットストリームを生成するための方法を実施するフローチャートである。
【図２】第１の実施形態に従いＦＰＧＡ実装およびビットストリームを生成するための図１のフローチャートの詳細図である。
【図３】第１の実施形態に従いデータベースへのエントリを行うための図１のフローチャートの詳細図である。
【図４】第１の実施形態に従い類似ＦＰＧＡデザインを求めるための図１のフローチャートの詳細図である。
【図５】第１の実施形態による第１の例として、類似度グラフをそのノード３４間の複数のエッジと共に概略的に示す図である。
【図６】第１の実施形態による第２の例として、すべてのノード間に１つのエッジが形成されている類似度グラフを概略的に示す図である。
【図７】第１の実施形態による第３の例として、すべてのノード間に１つのエッジが形成されている類似度グラフを概略的に示す図であって、ただしノード間において明示的には求められない類似度が、ノード間において特定された類似度に基づき求められることを示す図である。
【図８】第１の実施形態による例示的なモデルグラフを概略的に示す図である。
【図９】第１の実施形態による２つのモデルグラフのノード間の２つの類似度マトリックスを概略的に示す図であって、左側には「近傍マッチング」なしのイメージを、右側には「近傍マッチング」ありのイメージを示す図である。
【発明を実施するための形態】
【００４５】
図１～図４には、ＦＰＧＡモデルであるＦＰＧＡデザインおよび／またはハードウェア記述に基づき、ＦＰＧＡ実装およびこのＦＰＧＡ実装からのビットストリームを生成するための方法が示されている。
【００４６】
この方法の出発点として、ここではＦＰＧＡデザイン１０がグラフィックＦＰＧＡモデルとして生成されて準備される。次いでこのＦＰＧＡモデルが、ここでは例えばＶＨＤＬ（Very High Speed Integrated Circuit Hardware Description Language 超高速集積回路ハードウェア記述言語）であるハードウェア記述言語に書き換えられる。
【００４７】
ここから出発してビルドをスタートさせることができ、ビルドはまずは以下で詳しく説明するステップＳ１００，Ｓ１１０およびＳ１２０を含む。
【００４８】
ステップＳ１００において、ＦＰＧＡデザイン１０からネットリスト１２の合成が行われる。ネットリスト１２は、ＦＰＧＡデザイン１０に記述されているような論理素子およびそれらの結線を含む。
【００４９】
ステップＳ１１０において、ネットリスト１２からＦＰＧＡ実装１４が生成される。このステップは、一般的に「配置配線」（"Place and Route"）と呼ばれる。この場合、ＦＰＧＡのためにネットリスト１２に記憶されている論理素子が配置され（Place）、ネットリスト１２に従って互いに配線される（Route）。その成果物がＦＰＧＡ実装１４である。ステップＳ１１０の詳細についてはあとで説明する。
【００５０】
ステップＳ１２０において、生成されたＦＰＧＡ実装１４から、ＦＰＧＡへ伝送するための、およびＦＰＧＡと共に使用するためのビットストリーム１６が生成される。

【００５１】
図４に詳しく示されているように、類似ＦＰＧＡデザインを求めるステップＳ１４０～Ｓ１８０は、先行するステップに並列に、特にステップＳ１００に並列に、実施される。
【００５２】
したがってステップＳ１４０においてＦＰＧＡデザイン１０から、グラフベース表現としてモデルグラフ２４がＦＰＧＡデザイン１０に基づき生成される。ＦＰＧＡデザイン１０からモデルグラフ２４を作成する際に、ノードおよび信号ラインに対するＦＰＧＡモデル１０のブロックがエッジに変換される。図８にはモデルグラフ２４が例示的に描かれており、その際にこの図には、複数の独立したモデルグラフ２４が共に実装されて描かれている。この場合、グラフィックのサブシステムを、最初の高速な推定についてブロックとして解釈することができ、またはそれらのサブシステムを完全に行われた状態に至るまで任意の深さで非階層化することができる。変換関数によって例えば、ＦＰＧＡモデルをとりわけそれらのベースブロックに至るまで、仮想接続（Goto/From, DataStoreRead/DataStoreMemory/DataStoreWrite, Triggerport, Actionport等）を考慮しながら非階層化し、そこから有向グラフを生成することができる。ＦＰＧＡデザイン１０の非階層化に際して仮想接続が考慮され、ＦＰＧＡデザイン１０の非階層化に基づき変換関数が適用される。
【００５３】
ステップＳ１５０において、モデルグラフ２４の類似度がデータベース２２からのモデルグラフ２４と比較される。データベース２２からのモデルグラフ２４は、先行のＦＰＧＡ実装１４のグラフベース表現である。
【００５４】
その際に最初に、目下のＦＰＧＡデザイン１０のモデルグラフ２４と、データベース２２からの先行のＦＰＧＡ実装１４のモデルグラフ２４と、の比較が、グラフの類似度分析のためのアルゴリズムに基づき実施される。２つのモデルグラフ２４のそれぞれ２つのノード３４間の類似度の大きさをベースにして、ノード３４の類似度の大きさが、ブロックパス／ネットリストパス（部分的にも）、ブロックタイプ／ネットリストタイプ（例えば加算、乗算）およびブロックパラメータ／ネットリストパラメータ／ネットリスト属性（例えばビット幅、２進小数点位置）の一致の程度に基づき特定される。
【００５５】
さらにこの比較には線形最適化問題の解決も含まれ、これによれば１つのノード３４が二重に割り当てられることなく、一方のモデルグラフ２４内のノード３４各々について、他方のモデルグラフ２４内の最類似ノード３４が識別される。これはKuhnによる「ハンガリー法」として基本的に知られている。
【００５６】
目下のＦＰＧＡデザイン１０のモデルグラフ２４と、データベース２２からのＦＰＧＡ実装１４のモデルグラフ２４と、の比較にはさらに、近傍関係を考慮するための近傍マッチングが含まれる。かかる方法は例えば、Nikolicによる"Measuring Similarity of Graph Nodes by Neighbor Matching"として知られている。これに関連して図９には、２つのモデルグラフ２４のノード３４間の２つの類似度マトリックスが示されている。図９の左側のイメージには「近傍マッチング」なしの類似度マトリックスが、右側のイメージには「近傍マッチング」ありの類似度マトリックスが示されている。
【００５７】
この場合、目下のＦＰＧＡデザイン１０のモデルグラフ２４のノード３４と、データベース２２からのＦＰＧＡ実装１４のモデルグラフ２４のノード３４と、の類似度は、細分化点の一致に基づき求められる。細分化点には例えば、ブロックパス／ネットリストパス、ブロックタイプ／ネットリストタイプ例えば加算／乗算、ブロックパラメータ／ネットリストパラメータ例えばビット幅、２進小数点位置が含まれる。ここでは細分化点の重み付けが行われる。
【００５８】

さてここで、両方のモデルグラフ２４の類似度についてこのようにして特定された大きさから出発して、目下のＦＰＧＡデザイン１０のモデルグラフ２４とデータベース２２からのＦＰＧＡ実装１４のモデルグラフ２４との類似度が、目下のＦＰＧＡデザイン１０のモデルグラフ２４とデータベース２２からの最良モデルグラフ２４との類似度よりも高いか否か、についてチェックされる。データベース２２からのモデルグラフ２４が、目下のＦＰＧＡデザイン１０のモデルグラフ２４との類似度を求めた最初のモデルグラフ２４であるならば、データベース２２からのモデルグラフ２４が最良モデルグラフ２４または参照２６として引き継がれる。これとは異なり、データベース２２からのモデルグラフ２４がこれまでの参照２６よりも高い類似度を有している場合のみ、参照２６が変更される。このことは、適用可能であれば、ステップＳ１６０において行われる。データベース２２内に比較のためにさらに別のモデルグラフ２４が含まれているのであれば、この方法はステップＳ１５０に戻る。ステップＳ１００による合成がすでに終了しているか否かについても、チェックされる。ステップＳ１００による合成がまだ終了していない場合のみ、ステップＳ１５０において比較が続けられる。

【００５９】

つまりステップＳ１００による合成が終了しているならば、またはデータベース２２内に比較のためにさらに別のモデルグラフ２４が含まれていなければ、目下の参照２６の類似度の大きさが限界値と比較される。類似度の大きさが限界値よりも大きいならば、ステップＳ１１０においてＦＰＧＡ実装１４を生成するための参照２６として、相応のモデルグラフ２４がステップＳ１７０において準備される。そうでなければ、この方法のこの部分は、ステップＳ１８０において結果を伴わずに中止される。ここで限界値は例えば、９０％の類似度のところにある。

【００６０】

ステップＳ１１０におけるＦＰＧＡ実装１４の生成は、以下のようにして実施される。最初に、ステップＳ１７０により類似ＦＰＧＡ実装１４がモデルグラフ２４に基づき求められて準備されたか否か、についてチェックされる。これが当てはまらなければ、ＦＰＧＡ実装１４を作成する目的で、ステップＳ２００において通常の慣用の配置配線が実施される。そうでなければ、この方法はステップＳ２１０と共に続けられる。

【００６１】

ステップＳ２１０において、参照のためにデータベース２２からＦＰＧＡ実装１４がロードされる。次いでステップＳ２２０において、データベース２２からのＦＰＧＡ実装１４に基づき、インクリメンタル方式で配置配線が行われる。

【００６２】

図３に示されているように、データベース２２へのエントリが行われる。ステップＳ１３０において、それぞれ目下のＦＰＧＡデザイン１０のネットリスト１２、それに属するモデルグラフ２４および生成されたＦＰＧＡ実装１４が、ＦＰＧＡモデル１０のアーチファクト２０としてデータベース２２に記憶される。このように各ＦＰＧＡデザイン１０は、その後の新たなＦＰＧＡデザイン１０についての以降の比較のために、データベース２２内にアーチファクト２０を自動的に形成する。

【００６３】

ここには示されていないバックグラウンドステップとして付加的に、データベース２２内に含まれているモデルグラフ２４間の関係性類似度が計算される。この場合、データベース２２内に含まれているアーチファクト２０のモデルグラフ２４間の関係性類似度を算出するステップは、空きリソースを利用できるフェーズ中に行われる。その際に複数のＦＰＧＡモデル１０から、１つの類似度グラフ３０が作成される。図５～図７には、種々の類似度グラフ３０が示されている。この場合、２つのＦＰＧＡモデル１０間で調査された類似度が、ＦＰＧＡモデル１０を表現する対応する２つのノード３４間のエッジ３２として、類似度の大きさである重み３６と共に記載されている。類似度グラフ３０は、類似度線図とも呼ばれる。

【００６４】

この場合、図５の類似度グラフ３０は、少数のノード３４間のエッジ３２だけしか示していないのに対し、図６および図７の類似度グラフ３０は、完全な類似度グラフ３０である。ノード３４間においてエッジ３２として直線で表されているすでに求められた類似度から出発して、破線として表されている付加的なエッジ３８の重み３６が求められる。様々な手法によって、このように完全なものとすることができる。この目的で総当たり方式に従い類似度グラフ３０を完全なものとするために、付加的なエッジ３８の重み３６がさらなる類似度分析を通して逐次、補われる。

【００６５】

図７には、ＦＰＧＡデザイン１０に存在するエッジ３２の効率的な計算順序に基づき、そこにある類似度グラフ３０を完全なものとするための１つのバリエーションが示されている。これによれば、場合によって高い類似度が生じる可能性をもって存在しているエッジ３２に基づき、２つのノード３４間の類似度が推定され、その目的はそれらについて優先的に類似度分析を実施するためである。このようにすれば付加的なエッジ３８を、存在しているエッジ３２の重み３６の乗算によって推定することができる。これは図７によれば例えば、「２」および「４」で表されたノード３４間の付加的なエッジ３８について、これら双方のノード３４間の類似度を推定する目的で、点線矢印４０に沿って類似度を乗算することによって行われる。このことから出発して、求められた高い類似度を有するエッジ３２に対する重み３６を第一に特定することができる。

【００６６】

類似度グラフ３０に基づき、ステップＳ１５０における比較を簡単にすることができ、かつ加速することができる。

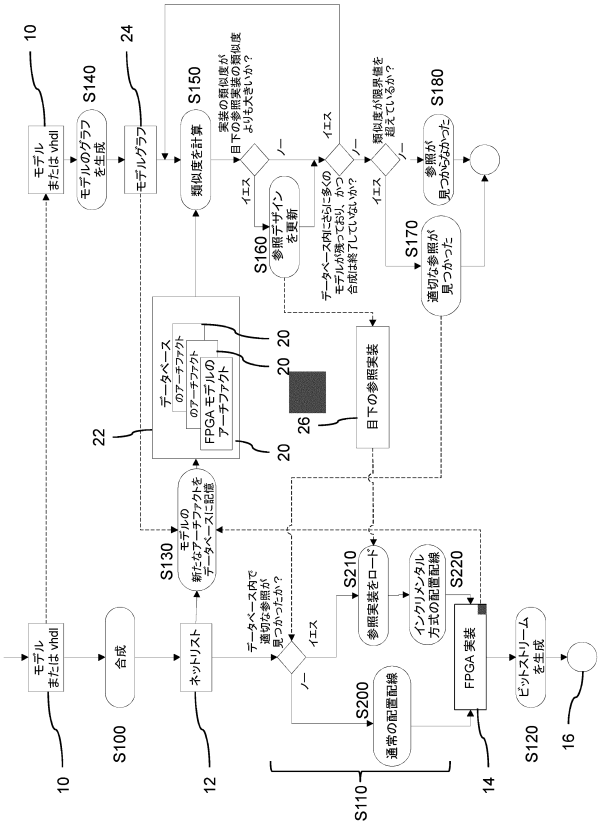
【符号の説明】

【００６７】

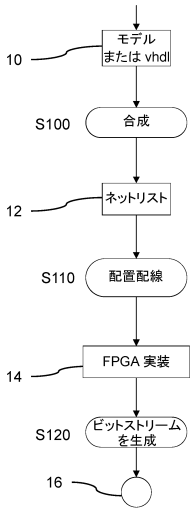
１０　ＦＰＧＡデザイン
１２　ネットリスト
１４　ＦＰＧＡ実装
１６　ビットストリーム
２０　アーチファクト
２２　データベース
２４　モデルグラフ
２６　参照
３０　類似度グラフ、類似度線図
３２　エッジ
３４　ノード
３６　重み
３８　付加的なエッジ
４０　点線矢印

【図面】

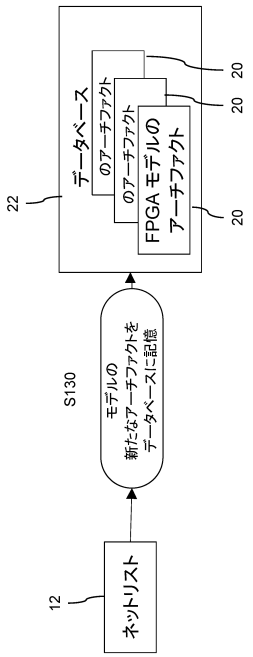
【図１】


モデル
または vhdl
モデルのグラフ
を生成
モデルグラフ
類似度を計算
実装の類似度が
目下の参照実装の類似度
より高く大きいか？
イエス
ノー
参照デザイン
を更新
データベース内にさらに多くの
モデルが残っており、かつ
合成は終了していないか？
類似度が限界値を
超えているか？
参照が
見つからなかった
適切な参照が
見つかった
データベース
のアーチファクト
FPGA モデルの
アーチファクト
目下の参照実装
新たなモデルの
アーチファクトを
データベースに記憶
データベース内で
適切な参照が
見つかったか？
参照実装をロード
インクリメンタル
方式の配置配線
通常の配置配線
FPGA 実装
ビットストリーム
を生成
合成
ネットリスト
S100
S110
S120
S130
S140
S150
S160
S170
S180
S200
S210
S220
10
12
14
16
20
22
24
26

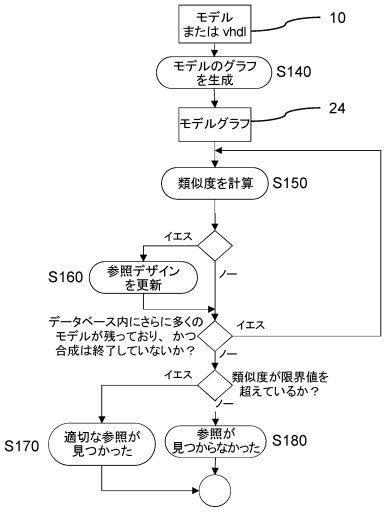
【図２】


モデル
または vhdl
合成
ネットリスト
配置配線
FPGA 実装
ビットストリーム
を生成
S100
S110
S120
10
12
14
16

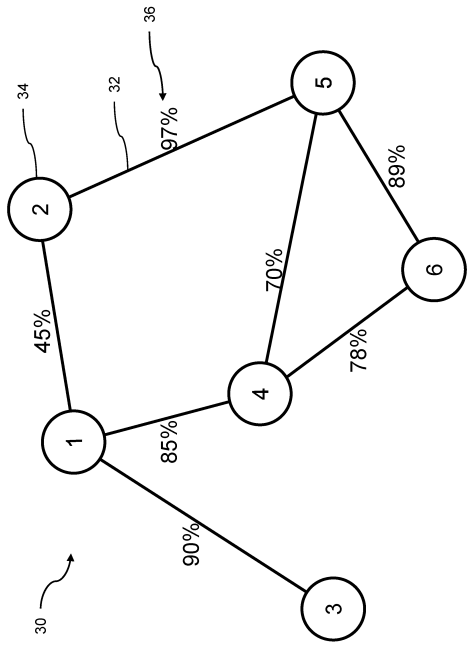
【図３】


データベース
のアーチファクト
FPGA モデルの
アーチファクト
モデルの
新たなアーチファクトを
データベースに記憶
ネットリスト
S130
12
20
22

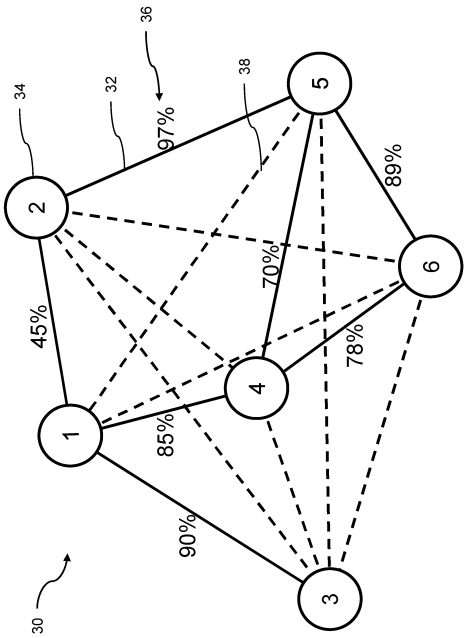
【図４】


モデル
または vhdl
モデルのグラフ
を生成
モデルグラフ
類似度を計算
イエス
ノー
参照デザイン
を更新
データベース内にさらに多くの
モデルが残っており、かつ
合成は終了していないか？
類似度が限界値を
超えているか？
適切な参照が
見つかった
参照が
見つからなかった
S140
S150
S160
S170
S180
10
24

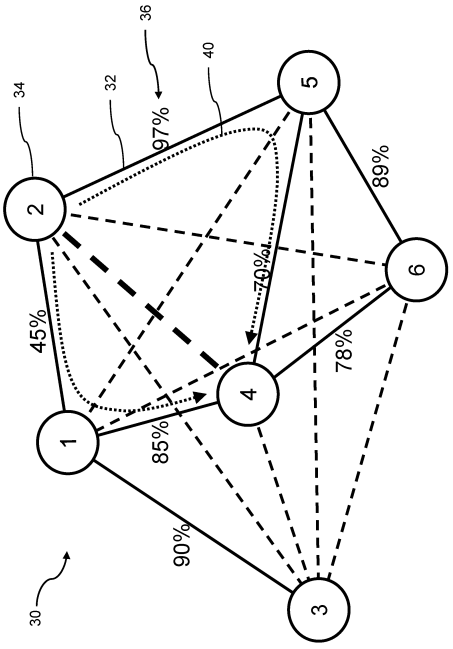
【図５】


36
34
32
97%
89%
70%
45%
78%
85%
90%
30
1
2
3
4
5
6

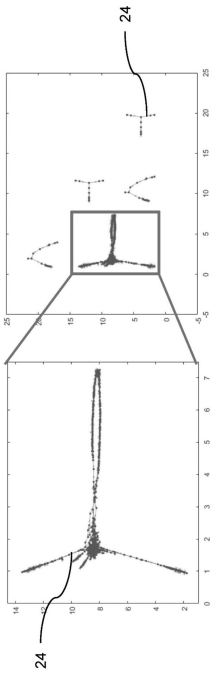
【図６】


36
38
34
32
97%
89%
70%
45%
78%
85%
90%
30
1
2
3
4
5
6

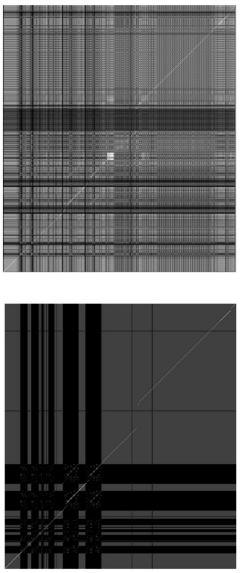
【図７】


36
40
34
32
97%
89%
70%
45%
78%
85%
90%
30
1
2
3
4
5
6

【図８】


24
24

【図９】



フロントページの続き

(74)代理人　100135633
弁理士　二宮 浩康
(74)代理人　100162880
弁理士　上島 類
(72)発明者　ドミニク ルーベライ
ドイツ連邦共和国 パデルボルン ラーテナウシュトラーセ ２６ ケア・オブ ディスペース ゲゼルシャフト ミット ベシュレンクテル ハフツング
(72)発明者　ハイコ カルテ
ドイツ連邦共和国 パデルボルン ラーテナウシュトラーセ ２６ ケア・オブ ディスペース ゲゼルシャフト ミット ベシュレンクテル ハフツング
審査官　甲斐 哲雄
(56)参考文献　特開２０１７－１９１５５６（ＪＰ，Ａ）
特開平０７－１５２８０１（ＪＰ，Ａ）
特表２００１－５１５２４７（ＪＰ，Ａ）
特開平１１－２１９３７９（ＪＰ，Ａ）
(58)調査した分野　(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　３０／００－３０／３９８
Ｇ０６Ｆ　１６／００－１６／９５８