(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号

特許第5706689号
(P5706689)

(45)発行日　平成27年4月22日(2015.4.22)　(24)登録日　平成27年3月6日(2015.3.6)

(51)Int.Cl.　F I
*G06F 17/50 (2006.01)*　G06F 17/50 654M
G06F 17/50 656D
G06F 17/50 658A
G06F 17/50 658T
G06F 17/50 658U

請求項の数 17　(全 29 頁)

(21)出願番号　特願2010-518222 (P2010-518222)
(86)(22)出願日　平成20年7月23日 (2008.7.23)
(65)公表番号　特表2010-534374 (P2010-534374A)
(43)公表日　平成22年11月4日 (2010.11.4)
(86)国際出願番号　PCT/US2008/008998
(87)国際公開番号　WO2009/014731
(87)国際公開日　平成21年1月29日 (2009.1.29)
審査請求日　平成23年7月22日 (2011.7.22)
(31)優先権主張番号　60/951,436
(32)優先日　平成19年7月23日 (2007.7.23)
(33)優先権主張国　米国 (US)
(31)優先権主張番号　12/177,867
(32)優先日　平成20年7月22日 (2008.7.22)
(33)優先権主張国　米国 (US)

前置審査

(73)特許権者　597035274
シノプシス，　インコーポレイテッド
ＳＹＮＯＰＳＹＳ，　ＩＮＣ．
アメリカ合衆国，　カリフォルニア州　９４０４３，　マウンテン　ビュー，　イースト　ミドルフィールド　ロード　６９０
(74)代理人　100092093
弁理士　辻居　幸一
(74)代理人　100082005
弁理士　熊倉　禎男
(74)代理人　100067013
弁理士　大塚　文昭
(74)代理人　100086771
弁理士　西島　孝喜

最終頁に続く

(54)【発明の名称】アーキテクチャー上の物理的合成

(57)【特許請求の範囲】
【請求項１】
複数の形式のリソースがあらかじめ決定された分布で配置されているチップにマッピングされる集積回路（ＩＣ）を設計する方法において、
選択された実装を有しない少なくとも１つのコンポーネントを含む高レベル回路表現において前記ＩＣの設計の一部分の配置変換に対する配置判断を決定するステップと、
前記配置変換に基づいて前記ＩＣの前記一部分に対するリソース位置を含む不完全な配置情報を決定するステップと、
前記配置変換によって変換された前記ＩＣの前記一部分に対する前記リソース位置に近い前記チップ上の位置に、前記リソースに対応する形式のリソースが十分に分布しているかを示すリソースの利用性についての情報に基づいて、前記高レベル回路表現で前記ＩＣの前記設計の前記一部分に対して１又は複数の合成変換を決定するステップと、
を備え、
前記配置判断を決定するステップ、前記不完全な配置情報を決定するステップ及び前記１又は複数の合成変換を決定するステップは、前記設計の各コンポーネントが前記選択された実装を有し、設計目的が満足されるまで繰り返され、プロセッサが、前記配置判断を決定するステップ、前記不完全な配置情報を決定するステップ及び前記１又は複数の合成変換を決定するステップの各々のステップを実行する、方法。
【請求項２】
前記選択された実装は、前記コンポーネントに対して選択されたチッププリミティブで

なされる、請求項１に記載の方法。
【請求項３】
以前に合成された高レベル回路表現の一部分に対して合成変換を繰り返すステップを更に備えた、請求項１に記載の方法。
【請求項４】
高レベル回路表現での前記合成変換がまだ完了していない間に前記配置判断が決定される、請求項１に記載の方法。
【請求項５】
前記合成変換は、
アダーツリー分解、
ＡＮＤ／ＯＲゲート分解、
ネットリストの平坦化、
電子的マルチプレクサ分解、
ロジック最適化、
ロジック分解、
アンドゥ／ドゥリソースシェアリング
ロジック複写、
ロジック因子分解、
シャノン展開、
ビットスプライシング、
ｍｕｘ／ｐｍｕｘコラプス及びタイミング駆動分解、及び
迂回除去、
の１つである、請求項１に記載の方法。
【請求項６】
前記合成変換は、高レベル回路表現からコンパイルされたＲＴＬコンポーネントをアッセンブルする、請求項１に記載の方法。
【請求項７】
前記合成変換は、前記ＩＣの前記設計に逐次変更を提供する、請求項１に記載の方法。
【請求項８】
前記合成変換は、回路設計の一部分に対する少数のオブジェクトの部分的合成である、請求項１に記載の方法。
【請求項９】
デジタル処理システムで実行されたときに、そのデジタル処理システムが、請求項１から８のいずれかに記載の集積回路（ＩＣ）を設計する方法を遂行するようにさせる複数の実行可能なインストラクションを含むマシン読み取り可能な記憶媒体。
【請求項１０】
複数の形式のリソースがあらかじめ決定された分布で配置されているチップにマッピングされる集積回路（ＩＣ）を設計するデータ処理システムであって、
選択された実装を有しない少なくとも１つのコンポーネントを含む高レベル回路表現において前記ＩＣの設計の一部分の配置変換に対する配置判断を決定する手段と、
前記配置変換に基づいて前記ＩＣの前記一部分に対するリソース位置を含む不完全な配置情報を決定する手段と、
前記配置変換によって変換された前記ＩＣの前記一部分に対する前記リソース位置に近い前記チップ上の位置に、前記リソースに対応する形式のリソースが十分に分布しているかを示すリソースの利用性についての情報に基づいて、高レベル回路表現で設計の１又は複数の合成変換を決定する手段と、
を備え、
前記配置判断を決定すること、前記不完全な配置情報を決定すること及び前記１又は複数の合成変換を決定することは、前記設計の各コンポーネントが前記選択された実装を有し、設計目的が満足されるまで繰り返される、データ処理システム。

【請求項１１】
前記選択された実装は、前記コンポーネントに対して選択されたチッププリミティブでなされる、請求項１０に記載のデータ処理システム。
【請求項１２】
以前に合成された高レベル回路表現の一部分に対して合成変換を繰り返す手段を更に備えた、請求項１０に記載のデータ処理システム。
【請求項１３】
高レベル回路表現での前記合成変換がまだ完了していない間に前記配置判断が決定される、請求項１０に記載のデータ処理システム。
【請求項１４】
前記合成変換は、
アダーツリー分解、
ＡＮＤ／ＯＲゲート分解、
ネットリストの平坦化、
電子的マルチプレクサ分解、
ロジック最適化、
ロジック分解、
アンドゥ／ドゥリソースシェアリング
ロジック複写、
ロジック因子分解、
シャノン展開、
ビットスプライシング、
ｍｕｘ／ｐｍｕｘコラプス及びタイミング駆動分解、及び
迂回除去、
の１つである、請求項１０に記載のデータ処理システム。
【請求項１５】
前記合成変換は、高レベル回路表現からコンパイルされたＲＴＬコンポーネントをアッセンブルする、請求項１０に記載のデータ処理システム。
【請求項１６】
前記合成変換は、前記ＩＣの前記設計に逐次変更を提供する、請求項１０に記載のデータ処理システム。
【請求項１７】
前記合成変換は、回路設計の一部分に対する少数のオブジェクトの部分的合成である、請求項１０に記載のデータ処理システム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、一般に、集積回路を設計する分野に係り、より詳細には、高レベル記述から合成プロセスを通して集積回路を設計することに係る。
【０００２】
関連出願：本出願は、参考としてここに援用する２００７年７月２３日に出願の米国プロビジョナル特許出願第６０／９５１，４３６号（管理番号０２９８６．Ｐ０５９Ｚ）の利益を主張する。又、この出願は、“Architectural Physical Synthesis”と題する２００８年７月２２日に出願の特許出願第１２／１７７，８６９号（管理番号０２９８６．Ｐ１１１７）にも関連し、それと同日に出願されたものである。
【背景技術】
【０００３】
ＶＬＳＩ（超大規模集積）技術の規模でデジタル回路を設計する場合、設計者は、コンピュータ支援技術をしばしば使用する。複雑なデジタル回路の設計及びシミュレーションを助けるために、デジタル回路を記述するハードウェア記述言語（ＨＤＬ）のような標準

言語が開発されている。ＶＨＤＬやＶｅｒｉｌｏｇのような多数のハードウェア記述言語は、工業標準として進化した。ＶＨＤＬやＶｅｒｉｌｏｇは、抽象的なデータ形式を使用してチッププリミティブレベル、レジスタ転送レベル（ＲＴＬ）又はビヘイビアレベルでハードウェアモデルを定義できるようにする汎用のハードウェア記述言語である。装置技術が進歩し続けているので、より新しい装置及び設計スタイルで使用するようにＨＤＬを適応させるために種々の製品設計ツールが開発されている。

【０００４】

ＨＤＬコードで集積回路を設計するときには、先ず、コードが書かれ、次いで、ＨＤＬコンパイラーによってコンパイルされる。ＨＤＬソースコードは、あるレベルで回路素子を記述し、そしてコンパイラーは、このコンピレーションからＲＴＬネットリストを形成する。ＲＴＬネットリストは、複数のＲＴＬオブジェクト、又はコンポーネントと、それらコンポーネント間の信号接続である複数のネットとで構成される。ＲＴＬネットリストは、フィールドプログラマブルゲートアレイ（ＦＰＧＡ）又は特定用途向け集積回路（ＡＳＩＣ）のような特定ベンダーの集積回路の技術又はアーキテクチャーとは独立しているという点で技術的に独立したネットリストである。ＲＴＬネットリストは、（ビヘイビア表現とは対照的に）回路素子の概略表現に対応する。次いで、マッピングオペレーションが遂行されて、技術的に独立したＲＴＬネットリストから、ベンダーの技術又はアーキテクチャーで回路を生成するのに使用できる技術的に特有のネットリストへと変換し、これは、インスタンスを配置し、相互接続を配線して、回路が所与のタイミング、スペース及び電力制約を満足するようにすることを含む。

【０００５】

初期の電子設計自動化（ＥＤＡ）は、図１に示すように、ＨＤＬ合成を配置／配線プロセスから完全に分離する。オペレーション１１において、ＨＤＬコードが準備される。オペレーション１３では、オペレーション１１で準備されたＨＤＬがネットリストを形成するようにコンパイル及び合成され、このネットリストは、典型的に、ロジック最適化を遂行することにより最適化される。その後、マッピングプロセスで、ネットリストを特定のターゲット技術／アーキテクチャーへマップする。オペレーション１３の終わりに、合成が完了し、ここで、ベンダーのＩＣに使用される技術／アーキテクチャーに特有のネットリストが与えられる。このネットリストは、実際上、ゲートレベルにあり、ファンアウトカウント又は接続されたコンポーネント形式及びサイズのような予め配置された情報に基づいて相互接続プロパティの統計学的モデルを使用することによりタイミング分析が推定される。合成後、オペレーション１５においてロジック回路で従来の配置オペレーションが遂行され、タイミング性能を満足するためにオペレーション１７においてネットリストへのローカルな変化（チッププリミティブ或いはセル又はゲートレベルのみにおける）がなされる。次いで、各ＩＣにおいて回路の設計を生成するためにオペレーション１９において従来の配線オペレーションが遂行される。満足されない制約がある場合には、プロセスは、ループバック繰り返しで変更を行う。

【０００６】

以前は、初期の合成ツールにおいてインスタンスの遅延が優勢であるときに、統計学的モデルに基づくタイミング推定が充分正確であり、合成と配置を分離しても、ＨＤＬ及び合成段階へ戻る繰り返しは比較的僅かに要求されるだけであった。

【０００７】

しかしながら、技術的な結節が収縮するにつれて、相互接続遅延が顕著になり、ゲート遅延を越える。その結果、合成オペレーションにおける遅延推定は、配置及び配線オペレーションに続く実際の遅延との相関性が益々なくなり、後分析結果と後レイアウト結果との間のタイミング予想の不足を招く。従って、多くのケースでは、配置及び配線プロセスの後に、回路の物理的レイアウトが回路設計基準を満足できなくなり、しばしば、設計者は、合成ステップからやり直して、合成／配置／配線プロセスを繰り返さなければならない。

【０００８】

合成を改善するために、合成プロセス中に設計（例えば、配置）に関連した物理的特性を考慮することが重要である。配置情報を合成プロセスへ持ち込むために、フロアプランニング、インプレース(in-place)最適化（ＩＰＯ）及び物理的合成のような一連の技術が採用されている。

【０００９】

フロアプランニング技術では、設計がチップ上の領域へと区画され、配置に基づく相互接続推定が領域間の相互接続に使用される一方、統計学的モデルを使用して領域内の相互接続が推定される。フロアプランは、初期のＲＴＬ段階で使用することもできるし、後で初期合成動作の後に使用することもできる。フロアプランニングは、ＲＴＬコンポーネントを領域へと区画化し、複写し及びスライスし、そしてＲＴＬレベルタイミング及びエリアモデルと結合するように拡張することができる。次いで、領域間タイミングからの改善されたタイミングを使用して、ＲＴＬレベル最適化をより正確に推進することができる。良質のフロアプランを手動で生成することは、挑戦であり、ユーザの熟練度を要求する。テラ(Tera)システム（米国特許第６，１４５，１１７号及び第６，３６０，３５６号）からのもののような自動フロアプランナーは、領域を生成し、それらにＲＴＬコンポーネントを指定することができる。合成は、減結合され、自動フロアプランニングに続くので、フロアプランニング中にはタイミング及びエリア情報の精度が悪い。

【００１０】

インプレース最適化（ＩＰＯ）と称される技術は、配置及び配線遅延のバックアノテーションを合成ドメインに与える。クリティカル経路は、最適化し直されるが、ディテール配置は更新されないので、変更されたネットに対する相互接続遅延は、統計学的モデルへ立ち返る。多くの変化がなされた場合には、それにより生じるネットリストをその後に合法化するのに、インスタンスをそれらの初期位置から離れて移動することが必要となり、大きな遅延推定エラーを招く。このために、タイミング閉鎖を達成するのに顕著な変化が要求されるときには、ＩＰＯが不安定に見える。

【００１１】

別の技術は、ＩＰＯ技術に勝る改善である物理的合成であり、この場合は、マップされたネットリストにおける少数の最適化が増分的な再合法化とインターリーブされて、遅延及びリソースメトリックの忠実度を維持する。この技術の制約は、個々の変化がリソースの控えめな増加に限定されること、又はＩＰＯ技術の不安定さの問題が再浮上することである。現在、物理的な合成に対して多数の異なるアルゴリズムがある。図２は、配置されたインスタンスの接近度に基づくタイミング推定を使用して物理的合成エンジンを与える１つのアルゴリズムを示す。オペレーション２３においてマップされたネットリストが最初に配置された後に、物理的合成オペレーションは、チッププリミティブレベルのみで遂行されるオペレーション２４において増分的最適化及び再配置のための回路の部分を選択する。

【発明の概要】

【発明が解決しようとする課題】

【００１２】

以上のことから、電子設計自動化のためのアルゴリズムの改良が必要とされていることが明らかである。

【００１３】

従来の特許も、チップ合成に係り、又はチップ合成を説明しており、それらの特許は、米国特許第６，５１９，７５４号、第６，７１１，７２９号、第７，０１０，７６９号、第６，１４５，１１７号、及び第６，３６０，３５６号を含む。配置アルゴリズムは、最近、論文：ブーヒュー氏、Timing-Driven Placement for Heterogeneous Field Programmable Gate Array、IEEE/ACM International Conference on Computer-Aided Design, November 2006 (ICCAD '06)、第３８３－３８８ページ（ＩＳＳＮ：１０９２－３１５２；ＩＳＢＮ１－５９５９３－３８９－１）に掲載されている。

【課題を解決するための手段】

【００１４】
本発明は、集積回路を設計するための方法及び装置を開示する。ここに例示する実施形態において、本発明の回路設計は、ＲＴＬ又はビヘイビアレベルで始まる合成及び配置の繰り返しプロセスを開示し、ここで、各繰り返しは、集積回路の設計の変換を通して増分的変化を与える。ある態様では、変換は、合成変換又は配置変換のいずれかである。合成変換は、ネットリスト内のオブジェクト、及び／又はオブジェクト間の接続を形成するネットを変更する。配置変換は、ネットリスト内の１つ以上のオブジェクトの位置を変更する。本発明の少なくともある実施形態の増分的繰り返し解決策は、現在回路ネットリスト、配置、タイミング、リソース利用性、及び電力のような設計メトリックにより決定される適切な合成及び配置変換を使用して連続的な進行を与える。ある態様では、各変換の後に、影響のある設計メトリックが更新され、将来の変換判断が、正確な設計統計情報に基づいてなされるようにする。このプロセスは、設計の最終的タイミングリソース及び電力エンクロージャーに向かって増分的に繰り返される。
【００１５】
本発明の少なくとも幾つかの実施形態の重要な観点は、高レベルコンポーネントに対して特定のリソース形式が識別される前に配置が行われることである。例えば、コンポーネントに対して望ましい重み及び関連リソース合計をもつ別の具現化がカタログ化され、そしてプレーサー(placer)は、望ましい具現化のためにリソース形式の付近へコンポーネントを移動するように配置を進行させる。
【００１６】
好ましい実施形態では、本発明は、ＲＴＬ又はビヘイビア設計（回路）を表現するグラフ及びチップリソースの物理的マップでスタートする。繰り返し変換が遂行され、各変換は、回路の最適化又は洗練化、或いは回路内のオブジェクトの配置を生じさせる。
【００１７】
一実施形態では、変換は、高レベル最適化より成る。この変換は、コンポーネント又は複数のコンポーネントを、ルール又は数学的変換を通して、タイミング、電力又はリソース消費のような優れた特性を有する機能的に同等の別の１組のコンポーネントへと最適化する。このような変換の一例は、ツリーの高さを減少して遅延を改善するように演算式を編成し直すことである。別の例は、リソースシェアリング又はアンシェアリングである。
【００１８】
別の実施形態では、高レベル最適化変換は、回路オブジェクト（１つ又は複数）のグループ（１つ又は複数）を、より抽象的な形態から、より具体的な形態へと洗練化する。洗練化変換の一例は、演算式をチップ上のＤＳＰリソースへとマッピングすることである。抽象的な形態が洗練化されるときには、通常、多数の具現化選択肢がある。例えば、演算式は、チップ上の特殊目的の演算機能（ＤＳＰブロック）により又はメモリ内のテーブルルックアップにより具現化することができ、或いはチップ上の低レベルロジックコンポーネント（ＬＵＴ又はゲート及びフリップ－フロップ）から構築することができる。ビヘイビア合成フローからのコンポーネントは、リソースの別のスケジュール及びシェアリングに基づいて登録される複数の具現化を有する。ビヘイビアコンポーネントに対するこのような代替物は、現在使用可能なリソース及び相互接続遅延に基づいて動的に発生することもできる。
【００１９】
又、別の実施形態では、洗練化変換は、別の具現化のクオリティに基づく緊急メトリックも有し、緊急性の順に選択される。具現化のクオリティは、エリア消費、電力消費又はタイミングのような設計目的に関して測定される。単一イベントアップセットの困難さのような他の、より秘儀的な目的を含むこともできる。例えば、設計が１つの大きなメモリ及び多数の小さなサイズのメモリを含み、且つ大きなメモリが、論理ファブリックにより具現化されたときに比較的悪い具現化クオリティをもつ場合には、大きなメモリを、設計上、中間サイズのメモリよりもチップ上の少ない特殊目的メモリリソースに関連付けることが非常に重要である。従って、大きなメモリに対する緊急性メトリックは、小さなメモ

リに対するメトリックよりも非常に高くなる。コンポーネントが特定の具現化へマップされ且つチップ上の特定のリソースに関連付けられると、それらコンポーネントへの接続が、回路の残部を配置するためのアンカーとして働き、タイミング及び使用可能なリソース推定のクオリティを改善する。

【００２０】

一実施形態では、配置変換は、インスタンスの混雑、配線可能性及び回路性能のような配置メトリックを改善するために１つ以上の配置可能なオブジェクトの位置を洗練化することができる。配置可能なオブジェクトは、ビヘイビア合成コンポーネント、非マップロジックのＲＴＬブロック、マップされたロジック、又はそれらの組合せより成る。

【００２１】

一実施形態では、配置変換は、異なる抽象レベルのオブジェクトを変更することができる。例えば、ある配置可能なオブジェクトは、ＲＴＬブロックであり、他のオブジェクトは、マップされたゲートである。

【００２２】

別の実施形態では、配置がローカルに充分進化して、使用可能なリソースを決定できると共に、配線遅延を推定できるときに、洗練化変換がトリガーされる。

【００２３】

本発明の別の態様によれば、集積回路を設計するためのここに例示する方法は、増分的変換の繰り返しを与え、合成及び配置変換は、いずれの順序でもなく、それらの機能に対して選択されるだけである。回路設計自動化は、選択機能に基づいて、合成又は配置のいずれかである次の変換を選択する。各繰り返しにおいて、変換の所定リストに対するコストが計算される。このコストは、他の変換のコストの変化に対して前を見ることを含む。例えば、演算オペレーションがＲＯＭへマップされる場合には、そのコストを上げる別の具現化に対してＲＯＭオプションを除去することができる。現在配置、ネットリスト、リソース、タイミング又は電力のようなコスト集中基準に基づき最良の変換が選択される。

【００２４】

次の変換は、配置更新、リソース指定、合成最適化、配置最適化、又は配線更新である。従って、ＩＣ設計の状態は、最終的な回路仕様及びレイアウトに向かって増分的に進行する。

【００２５】

別の実施形態では、配置変換は、クリティカル経路が整形を開始するまで又は所定の混雑スレッシュホールドに基づいてリソースが充分に拡散するまで繰り返し遂行される。繰り返し遂行に対する基準は、タイミング、リソースレイヤ当たりの混雑、エリア利用度及び電力である。

【００２６】

リソースレイヤ当たりの混雑は、リソースレイヤの使用により決定することができる。チップ上のリソースの個別のプリミティブ形式ごとにリソースレイヤがある。例えば、今日のＦＰＧＡ及び構造化ＡＳＩＣは、プリミティブチップリソースの不規則なレイアウトを導入する。これらのプリミティブな形式は、ロジック（ＬＵＴＳ）、フリップ－フロップ、高速シリアル相互接続のためのＳＥＲＤＥＳのような特殊なＩ／Ｏセル、異なる容量をもつ種々のメモリコンポーネント、及びＤＳＰアルゴリズムをスピードアップするための高速演算ブロックを含む。ロジック及びフリップ－フロップを除くと、典型的に、これらのリソースは、希薄に且つおそらく不規則な形態で含まれる。多くのＦＰＧＡは、限定された量のＲＡＭ、ＤＳＰ、及びチップ上で希薄な列に配置された他の専用ロジックブロックを有する。例えば、ＤＳＰ演算ブロックは、チップレイアウトにおいて２列しか使用できない。リソースレイヤは、プリミティブ形式ごとに生成される分布マップであり、その形式に対する使用可能なリソース位置と、その形式の各プリミティブの配置とを記録する。レイヤは、供給量より使用量が多いローカルな物理的領域が存在する場合に混雑していると言える。

【００２７】

この方法の典型的な例では、集積回路の設計の初期状態が、ＩＯピン、既存のフロアプラン、又は既存の配置のようなタイミング制約及び配置制約を伴う高レベル表現から発生される。高レベル表現は、ハードウェア記述言語（ＨＤＬ）コードであるか、又はハードウェア記述言語（ＨＤＬ）コードからコンパイルした後の技術的に独立したＲＴＬネットリストである。

【００２８】

一実施形態では、集積回路の設計の初期状態のネットリストが、先ず、タイミングに基づく一連のニュートラル最適化により最適化される。ニュートラル最適化は、リソースシェアリング又はアンシェアリング、好ましくはファンアウトテーブルタイミングに基づくアダーツリー分解、リソース指定、ハイアラーキーにわたり最適化を容易にするためのネットリストの平坦化、マルチプレクサ抽出又は再構築のような容易にアンドゥできるエリアの回復である。

【００２９】

一実施形態では、集積回路の設計の状態の一般的なフローは、ＲＴＬネットリストから分解及び因子分解へと進み、次いで、マップされ配線されたネットリストへと進む。配置変更、リソース指定、及びエリア又はタイミング最適化は、このフローを通して遂行される。

【００３０】

一実施形態では、配置及び回路アーキテクチャーを洗練化するプロセスは、全ての高レベルコンポーネントに特定の具現化及びリソース指定が与えられ、そして配置がチップに分散されて各コンポーネントが具現化のために充分近くにリソースを有するまで、繰り返される。この点から具現化を完了するために、より慣習的な物理的合成フローを使用することができる。

【００３１】

別の実施形態では、適用される変換及びそれらの潜在的な代替物が記録される。フローが繰り返され、そして別の変換を適用して、良好な結果を得ることができる。

【００３２】

又、本発明は、集積回路を設計するのに使用できるソフトウェア媒体を含む装置も開示する。例えば、本発明は、本発明に基づき集積回路を設計することのできるデジタル処理システムを含み、又、本発明は、コンピュータシステムのようなデジタル処理システムで実行されたときに、そのデジタル処理システムが集積回路設計方法を実行するようにさせるマシン読み取り可能な媒体も提供する。

【００３３】

本発明の他の特徴は、添付図面及び以下の詳細な説明から明らかとなろう。

【００３４】

本発明は、同様の要素が同じ参照番号で示された添付図面を参照して、一例として以下に説明するが、これに限定されない。

【図面の簡単な説明】

【００３５】

【図１】集積回路を設計するための従来の方法を示す。

【図２】物理的合成の従来の方法を示す。

【図３】本発明の一実施形態に基づき集積回路を設計するための方法のフローチャートである。

【図４】本発明の一実施形態に基づき集積回路を設計するための別の方法のフローチャートである。

【図５Ａ】本発明のある実施形態に基づき集積回路を設計する方法の細部を示す。

【図５Ｂ】本発明のある実施形態に基づき集積回路を設計する方法の細部を示す。

【図６】本発明の一実施形態に基づき集積回路を設計するための方法のフローチャートである。

【図７】形状及びリソースの推定を例示する。

【図８】リソース形式に対するマッピングを例示する。
【図９Ａ】メモリリソースのマッピングを例示する。
【図９Ｂ】メモリリソースのマッピングを例示する。
【図１０Ａ】リソースシェアリング具現化を例示する。
【図１０Ｂ】リソースシェアリング具現化を例示する。
【図１１】アダーツリー分解の一例を示す。
【図１２】ゲートツリー分解の一例を示す。
【図１３Ａ】スライス最適化の一例を示す。
【図１３Ｂ】スライス最適化の一例を示す。
【図１４】複写最適化の一例を示す。
【図１５】シャノン展開の一例を示す。
【図１６Ａ】ｍｕｘ／ｐｍｕｘコラプス及びタイミング駆動分解の一例を示す。
【図１６Ｂ】ｍｕｘ／ｐｍｕｘコラプス及びタイミング駆動分解の一例を示す。
【図１７】本発明に使用できるデータ処理システムのブロック図である。
【発明を実施するための形態】
【００３６】
集積回路又は複数の集積回路を設計するための方法及び装置をここに開示する。以下の説明において、説明上、本発明を完全に理解するために多数の特定の細部について述べる。しかしながら、当業者であれば、本発明は、これらの特定の細部を伴わずに実施できることが明らかであろう。他の点について、良く知られた構造、プロセス及び装置は、過度な細部を伴わずに説明をなすために、ブロック図で示すか又は概略を述べる。
【００３７】
本発明は、一実施形態において配置及び合成を単一パスに結合して集積回路を設計する方法及び装置を開示する。本発明の一実施形態は、合成と配置との間の相互作用がアーキテクチャーレベルで行われる「アーキテクチャー上の物理的合成」と称される物理的合成プロセスを開示する。これは、合成を、集積回路基板の表現上に実際に物理的に配置した状態で行えるようにし、使用可能なローカルリソースと、配置からの実際の回路タイミングに厳密に関連した遅延推定で合成を行い、従って、合成と配置との間の相互作用を同時に考慮することができる。更に、これは、高レベルのアーキテクチャー上の判断を行うか、高レベルコンポーネントをマッピングするか、或いは配置、混雑推定及びターゲットチップアーキテクチャーの特性を考慮する仕方で高レベル回路変換を行う自動的方法を提供することができ、前記特性は、種々のリソースの物理的分布、コンポーネント遅延、及び相互接続遅延を含むが、これに限定されない。本発明の１つの態様によれば、回路設計、又はＨＤＬコード表現が与えられると、特に、所与の分布リソースを伴う既存のフロアプランに対して合成及び配置を相互リンクする非常に多数の別々の具現化が存在する。最適な設計具現化を達成するために、配置を通して収集されたタイミング又は電力のような現在入手できる回路データに基づいて早期合成判断を後方追跡できることが重要である。
【００３８】
従って、本発明の態様では、種々の設計具現化の適切さを正確に評価できるようにするため、配置は、早期合成サイクル、例えば、回路アーキテクチャーレベル、高レベル設計又はビヘイビア表現において遂行される。これは、リソースがチップ上に均一に分布されないＦＰＧＡ及び構造化ＡＳＩＣのような前拡散型(prediffused)チップでは特に重要である。この前拡散型チップでは、リソースの位置及びリソースの形式が予め決定されて、希薄な状態で分布される。例えば、今日のＦＰＧＡ及び構造化ＡＳＩＣは、チップリソースの不規則なレイアウトを導入している。これらのコンポーネントは、ロジック、フリップ－フロップ、高速シリアル相互接続のためのＳＥＲＤＥＳのような特殊Ｉ／Ｏ、異なる容量をもつ種々のメモリコンポーネント、及びＤＳＰアルゴリズムをスピードアップするための高速演算ブロックを含む。多くのＦＰＧＡは、限定された量のＲＡＭ、ＤＳＰ、及びチップ上で希薄な列に配置された他の専用ロジックブロックを有する。例えば、ＤＳＰ演算ブロックは、チップレイアウトにおいて２列しか使用できない。

【００３９】
１つの態様において、本発明は、合成フローの始めに物理的配置及びアーキテクチャー選択を統合するためにチップアーキテクチャー進化のこの変化に向けられる。この要求は、ＲＴＬレベル又はビヘイビア合成レベルであり、ここで、異なる形式の所要リソースの数が決定される。
【００４０】
リソースレイアウト情報の現在の意識と、早期合成プロセスにおける配置及び合成の統合（例えば、設計の多くのコンポーネントは具現化を選択していないが）は、リソースの最適な利用を与える。例えば、リソースレイアウト情報を意識しないＲＴＬ合成プロセスは、あるリソース形式は過剰に使用するが、他のリソース形式はあまり使用されない中間ネットリストを生じる。更に、リソース形式判断は、リソースの物理的位置に適合しないことがある。例えば、チップのあるローカル部分では、利用できる以上のＤＳＰリソースが要求されることがある。本合成方法は、チップ上のリソース分布について知り、そして特定のリソースが充分あるかだけでなく、充分近くにあるかも知ることで、これらリソースの効率的な利用を与えることができる。従って、遠くに配置されたリソースへ信号を配線することによる大きな相互接続遅延を回避することができる。
【００４１】
本発明の態様によれば、種々の配置判断が決定されるが、合成は依然高レベル回路表現であり（例えば、設計における多数のコンポーネントはまだ具現化を選択しておらず）、又はゲートレベル記述を依然決定しなければならない。これらの配置判断は、タイミング遅延又は電力消費のような回路パラメータの正確な評価を可能にし、最適な設計具現化に向かう増分的経路を許す。一実施形態では、図３に示すように、プロセスは、オペレーション３０において、ＥＳＬ又はＨＤＬ言語、ビヘイビア抽象、或いはＲＴＬネットリストの高レベル抽象に対するコンパイルされたＨＤＬコード、並びにタイミング、フロアプラン、電力及び配置制約を含むＩＣ設計の初期状態でスタートする。オペレーション３１において、合成変換が遂行され、これは、プロセスの早期段階では、高レベル変換である。この合成変換は、設計の一部分に対するものである。オペレーション３２では、配置変換が既存の回路表現において遂行され、早期段階では、アーキテクチャーレベルでの配置となる。この配置変換は、設計の一部分に対するものである。このオペレーションにおける配置判断は、この早期段階では詳細な情報がおそらく欠落するので、種々の仮定及び推定を必要とすることがある。ＩＣ設計状態の容易さがオペレーション３４において評価され、そして設計及び法的目的を満足する場合に、オペレーション４８における慣習的な物理的合成へ移行する。この早期段階ではあり得るが、目的を満足しない場合には、合成の別の段階へループバックする。
【００４２】
合成の次の繰り返し（現在のオペレーション３１）は、特に、物理的配置情報が与えられた（手前のオペレーション３２）後に、設計表現を改善する。同様に、配置の次の繰り返し（現在のオペレーション３２）は、合成の改善が与えられた後に回路パラメータの推定を改善する。このような密接なループでは、合成及び配置が一緒に緊密に働き、顕著な見直しを伴わずに最適な設計表現への経路を与えることができる。
【００４３】
一実施形態では、合成オペレーションは、回路設計表現に対して種々の具現化を与え、そして配置オペレーションは、選択肢を狭める上で助けとなるように、回路パラメータの分析を遂行することができる。例えば、具現化＃１が明らかに優れている場合には、それが選択され、潜在的具現化の数が１に狭められる。或いは又、具現化＃２が明らかに設計制約の範囲外である場合には、それが排除され、潜在的具現化の数が１だけ狭められる。
【００４４】
本発明の１つの態様によれば、複数の集積回路を設計するためのここに例示する方法は、抽象的なマシン仕様から、統合された双方向及び繰り返し式の合成及び配置を与える。一実施形態では、集積回路を設計するここに例示する方法は、ＩＣ設計の状態を増分的に

変化させる。ＥＳＬ又はＨＤＬ言語、ビヘイビア抽象、或いはＲＴＬネットリストの高レベル抽象に対するコンパイルされたＨＤＬコード、並びにタイミング、フロアプラン、電力及び配置制約を含むＩＣ設計の初期状態でスタートして、ここに例示する方法は、最適な設計状態に到達するまでＩＣ設計状態を増分的に繰り返し変化させる。最適な状態は、タイミング及び配置制約を満足するチッププリミティブレベルネットリストであるのが好ましく、これは、次いで、広範囲な見直しを行わずに慣習的な配置及び配線プロセスへパスすることができる。

【００４５】

１つの態様によれば、本発明は、合成及び配置の繰り返しプロセスを開示し、各繰り返しは、集積回路の設計に対する増分的変化を与える。本発明の幾つかの実施形態についての一般的な例を、図４を参照して説明する。図４の方法は、ＩＣ設計の初期状態を発生するオペレーション４０で開始される。ＩＣ設計の初期状態は、ビヘイビア表現又は高レベルＲＴＬネットリストを含み、これは、回路及びロジックを記述するＨＤＬソースコードからコンパイルすることができる。

【００４６】

技術的に独立したＲＴＬネットリストは、典型的に、設計の高レベルビヘイビア表現である。これは、最終的なマッピングステップの前にプロセスによって使用するための抽象情報を保存する。これは、言語コンピレーションを行った直後に、設計を微細な低レベル（ゲート）表現へと断片化する慣習的な合成ツールとは異なる。高レベルのビヘイビア表現を保存することにより、合成ツールは、最適化、区画化、及びフロアプランニングを非常にグローバルなレベルで遂行して、典型的に、良い結果を与えることができる。抽象データに対して動作することで、合成ツールは、より迅速に動作し、より大きな設計を取り扱うことができる。高レベルのＲＴＬネットリストは、特定ベンダーの技術又はアーキテクチャーとは独立して、回路ブロック表現のような高レベル抽象を含む。

【００４７】

ＩＣ設計の初期状態は、更に、タイミング制約、電力制約及び配置制約、例えば、ＩＯピン位置、既存のフロアプラン又は既存の配置（例えば、ＩＣチップ、ＩＰブロックのサイズ及び形状）含む。オペレーション４２において、ＩＣ設計の状態は、増分的に変化される。集積回路の設計の状態は、一般的に、ネットリスト、タイミングデータ、リソース情報、配置情報、配線情報、及び電力データを含む。設計状態の増分的変化は、合成又は配置変更であり、以下に詳細に述べる。本発明の１つの態様において、変化は、増分的であり、これは、設計の最適化が、典型的にタイミング推定及び配置制約のような全ての現在情報と共に小さな変更で進行することを意味する。増分的変化は、着実に進行する完全な機密状態で設計を進行させることができる。１つの態様では、増分的な変化は、力指向方法のような増分的なグローバルな配置アルゴリズムを含む。別の態様では、増分的変化は、シミュレーションされるアニールのようなグローバルな最適化アルゴリズムを含む。オペレーション４４では、ＩＣ設計の状態が評価され、そしてオペレーション４６において、オペレーション４２へ戻ることにより更に繰り返しを続けるべきか、又はオペレーション４８で設計フローを完了すべきかの判断がなされる。

【００４８】

本回路設計方法は、合成及び物理的設計（例えば、配置及び配線）である集積回路設計の２つの基本的ステップ間に高度な統合及び繰り返しプロセスを与える。合成及び配置が強く相互依存する概念では、配置を伴わない合成において設計制約を正確に推定できず、又、合成を伴わずに配置を遂行することもできないので、本発明の設計方法は、合成及び配置を、増分的繰り返し解決策で、１ステッププロセスへ効果的に合体する。

【００４９】

一実施形態では、本方法は、合成／配置変換の繰り返しを与える。繰り返しプロセスの本体は、配置変換、合成変換、或いは合成及び配置変換の組合せである。いずれの場合にも、集積回路の設計の状態は、設計目的を満足するチッププリミティブレベルネットリストの合成又は配置に向かって増分的に繰り返し変化する。図５Ａ及び５Ｂは、ＩＣを設計

するためのフローの一部分の２つの例を示し、即ち図５Ａに示す方法のケースでは、最初に配置変換が生じ、その後、合成変換が生じるが、図５Ｂでは、その逆のことが生じる。合成、配置、又は合成／配置の増分的な繰り返し変換は、設計のいずれの状態においても合成と配置との間の連続的な相互作用を与える。合成及び配置の増分的な繰り返しの進行は、合成変換が、常に、最新の最も正確な設計状態情報を有するように保証し、この設計状態情報は、配置変換からの遅延情報及びローカルリソースの利用性を含み、そして配置変換は、常に、最新の合成ネットリストに基づいて物理的配置及び配線情報に対する最良の推定を与える。配置及び合成変換は、ネットリストがチップレベルプリミティブのみで構成され、設計目的が満足され、そしてディテールプレーサーが小さなローカル領域を独立して容易に合法化できるレベルに配置混雑が減少されるまで、続けられる。このフローの後に慣習的な物理的合成が続いて、具現化が完了となる。

【００５０】

図６は、ＩＣ設計状態の増分的変化のための本発明の実施形態を示す。本発明は、全ての抽象レベルを同時に配置することができる。早期の繰り返し中には、設計が主としてチッププリミティブより成るその後の繰り返しより、高い抽象レベルにあるオブジェクトがはびこっている。チッププリミティブインスタンスは、典型的に、最低レベルの表現である。合成変換は、ネットリストを徐々に変更し、より高い抽象レベルにあるオブジェクトを、より具体的なオブジェクトへと変化させる。これらの具体的なオブジェクトは、それに続く合成及び配置変換において考慮に入れられるより特定のリソース要求を有している。配置変換は、ＲＴＬインスタンス、非マップインスタンス、マップされたインスタンス、又はチッププリミティブレベルインスタンスのいずれかであるネットリストインスタンスの位置を決定し、これにより、回路におけるネットの長さ及び遅延をルータで決定する。配置変換は、合法的配置に向かって回路配置を徐々に繰り返すことができ、ここで、合法的配置とは、ＩＣチップのリソース使用を支配するルールを満足することを意味する。典型的に、早期の繰り返しでは、配置は、あまり合法的でない。配置変換は、オブジェクト位置の増分的変化をなすので、配置変換を一回繰り返しても、合法的配置とならない。配置変換を繰り返すことで、配置が合法的となる。この実施形態では、配置変換は、本電子設計自動化の中心である。

【００５１】

各繰り返しにおいて、繰り返しの基準は、タイミングデータ、リソースレイヤ当たりの混雑、エリア利用性、電力レベル、又はその組合せである。この方法は、更に、設計を最適化し、クリティカル経路を整形し、又はリソースを所定スレッシュホールドに対して拡散するために考えられる内部ループ繰り返しを含むことができる。

【００５２】

合成及び配置変換を増分的に繰り返す本発明の方法の実施形態では、設計の全ての段階で合成変換において物理的設計情報を常に得ることができる。従って、合成における最適化及び変換は、タイミング及びエリアに関して、又、配線可能性に対する影響に関して、常に最新である。合成においてなされる回路構造に関する判断は、配置と完全に一致している。

【００５３】

合成及び配置変換を増分的に繰り返す本発明の方法は、論理的構造及び回路の空間的配置を同時に最適化するように合成及び配置変換を効果的に結合する。この方法の典型的な例では、集積回路の設計の状態が、最終的な回路仕様及びレイアウトに向かって増分的に進行する。

【００５４】

繰り返し配置変換の進行は、ネットリストの成熟度の増加レベル又は配置構成である。設計の成熟度は、ネットリストがチップレベルプリミティブのみで構成される程度、設計目的が満足される程度、及びディテールプレーサーが小さなローカル領域を独立して容易に合法化できるレベルまで配置混雑が減少される程度によって測定される。

【００５５】

繰り返し合成変換の進行は、タイミング制約を満足するためのオブジェクト又はインスタンスの再構成又は複写のような合成最適化である。合成最適化は、回路最適化、抽象的コンポーネントの分解、演算マッピング、アンドゥ／ドゥリソースシェアリング、アダーツリー分解、配置に基づくアンド／オアゲート分解、経路複写、経路迂回除去、ＲＡＭ又はＤＳＰのような離散的リソースへの指定、論理的因子分解、マルチプレクサ再構成、或いはハイアラーキーにわたる最適化を容易にするためのネットリストの平坦化を含むが、これらに限定されない。

【００５６】

この方法の実施形態が図６に示されており、これは、ＩＣ設計の初期状態が発生されるオペレーション６１で開始される。ＩＣ設計の状態は、タイミングデータ、リソース情報、配置情報、配線情報、及び／又は電力データのような関連状態情報を伴うＲＴＬネットリストである。典型的に、ＩＣ設計の状態は、機能、タイミング、電力及びフロアプランのような回路要件を指定するのに充分な情報を含む。

【００５７】

高レベルＲＴＬネットリストは、ほとんどのオブジェクトが低レベルチッププリミティブの抽象であるところのネットリストである。関連プリミティブのグループは、ＲＴＬによってエンコードされた機能を表す高い表現レベルをもつオブジェクトとして表すことができる。集積回路設計の高レベル又は抽象的表現は、ＲＴＬコード又はその一部分を表す論理的オブジェクトである。各オブジェクトは、典型的に、多数のチッププリミティブを表し、例えば、アダー、マルチプライヤ、マルチプレクサ、及びシーケンスロジック、並びにＡＮＤ機能、ＯＲ機能のような、より複雑な機能を表す。又、高レベル表現のオブジェクトは、メモリブロック又は専有（知的プロパティブロック又はＩＰ）ブロックを含むこともできる。他の論理的オブジェクトは、グルーロジック（バッファ又はインターフェイス機能を与える）、タイミングロジック、制御ロジック、又はメモリロジックのようなサポート機能を与えるためのＲＴＬコードの各部分である。又、高レベルＲＴＬオブジェクトの幾つかは、チップレベルプリミティブでもよい。又、オブジェクトのネットリストは、配線及び配置のための各オブジェクトに関連した情報も含む。オブジェクトは、対応するＲＴＬコードへマップして戻るための情報を含むことができる。

【００５８】

更に、ＲＴＬコードは、機能が一緒にグループ編成されたハイアラーキーを含むことができる。ある状況では、タイミング、ルーティング、エリア、又は電力要件を最適化するために、コンポーネントを１つのハイアラーキーから別のハイアラーキーへグループ編成し直すことができる。他の状況では、増分的繰り返しプロセス中に、機能的ＲＴＬハイアラーキーを全体的又は部分的に平坦化することができる。

【００５９】

最初に、設計の初期状態は、タイミング制約、電力制約、及び／又は配置制約のような制約を含むことができる。例えば、配置制約は、ＩＯピンの位置、既存のフロアプラン、又は既存の配置データを含むことができる。

【００６０】

ここに例示する実施形態では、設計の初期状態は、最初に、タイミングに基づく一連のニュートラル最適化によって最適化される。このニュートラル最適化は、容易にアンドゥできるエリア回復、例えば、アンドゥ／ドゥリソースシェアリング、ファンアウトテーブルタイミングに基づくアダーツリー分解、明らかなリソース洗練化、例えば、設計上大きなＲＡＭがあり且つ１つのＲＡＭブロックリソースしか使用できない場合にＲＡＭがそこへ行かねばならないこと、ハイアラーキーを横切る最適化を容易にするためのネットリストの平坦化、並びにマルチプレクサ構造体の抽出及び再構築を含む。

【００６１】

オペレーション６２において、ＩＣ設計の状態を増分的に変化するために現在設計状態（現在配置、ネットリスト、リソース、タイミング、電力及び配線）に基づいて次の変換が選択される。オペレーション６３－７０は、本発明の一実施形態による典型的な変換で

あり、配置又は配置更新（６３）、リソース指定（６４）、因子分解（６５）、マップ（６６）、ロジック最適化（６７）、具現化の生成／洗練化（６８）、配線更新（６９）、及び他の合成（７０）を含む。これらの変換は、典型的に、配置及び合成のシームレスな統合を許す小さな増分的なオペレーションであり、従って、合成は、配置を知ることで遂行され、そして配置は、合成を知ることで遂行される。

【００６２】

従って、繰り返し及び増分的変換６３－７０は、アンドゥ／ドゥリソースシェアリング、アダーツリー分解、ＡＮＤ／ＯＲゲート分解、ロジック複写、ビットスプライシング、迂回除去、因子分解のような最適化変換、並びに離散的リソース（ＲＡＭ、ＤＳＰ、等）への指定、及び配線のような配置変換を含む配置及び合成オペレーションで構成される。

【００６３】

ここに例示する実施形態では、各繰り返しにおいて、オペレーション６２では、種々の潜在的な変換がコスト関数に基づいて評価される。コスト関数は、最初に動作すべき最良の変換を選択するように設計され、それ故、タイミング、配置混雑、配線混雑、エリア利用性及び電力のような設計状態情報を含む。評価の際に、最良の変換が遂行され、そして設計制約が満足されるまで繰り返しが続けられる。１つの態様において、設計は、次いで、慣習的なゲートレベル配置及び配線へと進む。

【００６４】

各繰り返しにおいて、この方法は、選択のリストを通して実行されて、コスト関数に基づいて最良の変換を選択する。例えば、配置変換と合成変換との間の選択は、タイミング収斂基準に基づく。クリティカル経路では、配置は、もし可能であれば、クリティカルネットを短縮するように試みることができる。クリティカルネットを短縮できない場合には、物理的合成最適化にネットを使用することができる。

【００６５】

本発明の別の態様によれば、集積回路設計するためのここに例示する方法は、変換の繰り返しを与え、合成及び配置変換は、いずれの順序でもなく、それらの機能に対して選択されるだけである。この方法は、合成と配置との間の良好な繰り返しを与え、その繰り返しの中で、タイミング及び配置制約を伴う最終的な構成に向かって進行するように、集積回路の設計の状態に基づいて次の変換が選択される。一実施形態では、この方法は、タイミング、リソースレイヤ当たりの混雑、エリア利用性、及び電力のような幾つかの基準に基づいて次の変換を選択する変換選択アルゴリズムを与える。次の変換は、リソース混雑の少ない現在ネットリストに対して配置変化を生じさせるか又は設計目的を良好に満足するために回路が繰り返しを受けるような配置の更新である。次の変換は、因子分解、最適化、又は分解のような合成最適化である。次の変換は、タイミング又は重要経路要件を満足するための分割、再構築又は複写のような合成最適化である。次の変換は、回路仕様及びレイアウトを完成させるか又は配線を更新するためのチッププリミティブレベルネットリストに向かって、現在ネットリストを低い抽象レベルへマップさせることのできる合成である。

【００６６】

次の変換は、フロアプラン区画化、リソース指定、タイミング又は重要経路要件を満足するためのロジック再構築又は複写、或いはインスタンス配置のための配線更新のような配置最適化である。次の変換は、回路仕様及びレイアウトを完成させるためのチッププリミティブレベルネットリストに向かって、現在ネットリストを低い抽象レベルへマップさせることのできる合成オペレーションである。

【００６７】

増分的変換では、タイミング及び電力のような設計状態情報が最新のものであり、それ故、目的に対する影響の正確なビューで最適化を遂行することができる。

【００６８】

別の実施形態では、多数の変換が選択される。各々の選択された変換が、次いで、設計状態に対する影響を測定するように適用され、そして復帰又はアンドゥされる。次いで、

最良の変換が選択され、適用される。

【００６９】

一実施形態では、本発明の重要なステップは、オペレーション６８であり、ネットリストの各ＲＴＬオブジェクトに対して、考えられる具現化選択肢を生成し又は洗練化する。その関連機能は、別の具現化の各々に必要とされる形状及びリソースの推定を遂行する。別の実施形態では、オペレーション６８は、好ましい具現化を指示する重みを各具現化に指定することもできる。アーキテクチャーレベルで合成及び配置を合体する本発明の１つの重要な効果は、異なるアーキテクチャー上の具現化を評価できるようにすることである。本発明のアーキテクチャー上の物理的合成がないと、ＲＴＬ合成段階において具現化が選択されたときに、ゲートレベル配置段階で、高レベル情報を回復することができない。その結果、他の具現化が好ましい場合には、準最適状態となる。それ故、具現化判断がＲＴＬレベルにおいて物理的情報でなされた場合には、非常に優れたタイミング結果を得ることができる。この変換は、配置及び配線段階に対して回路がマップされると、遂行することが非常に困難である。

【００７０】

繰り返しが進み、設計状態が洗練化するにつれて、オペレーション６８は、下位プロパティをもつ具現化選択を排除する。関数Ｆ、即ち具現化、Ｆ＝Ｓ＆（Ａ＊Ｃ）｜｜～Ｓ＆（Ｂ＊Ｃ）の例を使用して、オペレーション６８を説明する。選択信号Ｓが１である場合には、Ｆは、ＡとＣを乗算した結果であり、一方、Ｓが０である場合には、Ｆは、ＢとＣを乗算した結果である。オペレーション６８は、この関数に対して考えられる別の具現化を決定する。図１０Ａ及び１０Ｂは、この関数に対して生成／洗練化(Create/Refine)具現化オペレーションが生成する２つの考えられる別の具現化を示す。図１０Ａは、２つのマルチプライヤ及び１つのマルチプレクサを使用する具現化を示し、これは、出力Ｆがタイミング重要であり且つ選択信号Ｓが最新の到着時間である場合に望ましいものである。図１０Ｂは、単一のマルチプライヤ及びマルチプレクサを使用する具現化を示し、これは、入力Ｃが最新の到着信号であるか、或いは出力Ｆがタイミング重要でなく且つエリアの減少が望まれる場合に、より望ましいものである。これら２つの別の具現化は、リソースシェアリング／アンシェアリングを示す。関数のタイミング及び配置に関する具体的情報がないと、典型的な高レベル合成アルゴリズムは、典型的に、図１０Ａのような別の具現化を評価しない。というのは、２つの非常に高価なマルチプライヤに対してリソースを使用するからである。これは、慣習的なフローの配置が、出力重要で且つ選択信号ＳがＡ、Ｂ及びＣの後に到着する状態で、この関数を、専用の未使用マルチプライヤリソース付近に配置する場合であってもそうである。本発明では、オペレーション６８は、これら具現化の両方を生成し、そしておそらく他のものは、明らかにそれらが準同等であるときには別の具現化を排除する。例えば、繰り返しが進むにつれて、出力Ｆが重要でなくなることが明らかである。この場合に、オペレーション６８は、具現化選択を図１０Ｂのものだけに洗練化する。というのは、この別の具現化は、少数のリソースしか使用しないからである。或いは又、オペレーション６８は、Ｆ及び選択ラインＳが重要であり且つマルチプライヤを具現化するのに使用できるリソースが近くにある場合には図１０Ｂの具現化を排除する。

【００７１】

ＦＰＧＡチップは、典型的に、複数の前拡散型メモリリソース、例えば、フリップ－フロップと、ビットサイズの変化するブロック、例えば、５１２、４Ｋ、及びＭＲＡＭとを有する。設計により要求されるメモリコンポーネントも、サイズが変化する。典型的に、これらのメモリコンポーネントをどのように具現化すべきか明確でない。例えば、２ないし５１２ビットの適度なサイズのＲＡＭは、フリップ－フロップ、５１２リソース、又は４Ｋリソースで具現化することができる。更に、大きなメモリサイズに対するリソースのサイトは、典型的に、チップ上で希薄に得られるだけである。以前のＥＤＡツールでは、配置情報は、メモリ具現化段階では得られなかった。それ故、具現化の判断は、ローカル使用及び正確なタイミング情報なしに行われた。この制約は、重大な性能低下を生じさせ

る。適度なサイズのＲＡＭが５１２リソースとして具現化され、唯一使用可能な５１２サイトが、ＲＡＭが接続されるロジックから遠くに位置される場合には、ＲＡＭを５１２に強制すると、相互接続部が長くなると共に、フリップ－フロップの具現化に対して５１２サイトを使用する遅延の有利さを無効にしてしまう。フリップ－フロップを使用する具現化の遅延は、長くなるが、この具現化で、ＲＡＭのフリップ－フロップと、ＲＡＭが接続されるロジックとの間の相互接続部を短縮できる場合には、より高速の設計となる。或いは又、使用可能な４ＫリソースがＲＡＭ接続ロジックの付近にある場合には、４Ｋとして具現化するのが効果的である。従って、メモリ具現化判断は、種々の使用可能なメモリリソースと、メモリへ接続するコンポーネントの位置とを考慮して、行わねばならない。

【００７２】

図９Ａは、メモリ具現化判断の一例を示す。この図は、メモリリソースを頂面及び底面にもつチップを例示している。４ビットＲＡＭがチップの右側のパッド及びＡＮＤゲートに接続されている。ＲＡＭがメモリとして具現化されてチップの頂面に配置される場合には、そのパッド入力及びそれが駆動するＡＮＤゲートへの相互接続部が非常に長いものになる。図９Ｂは、同じロジックの交互のマッピングを示す。ＲＡＭは、近傍のロジックを使用して具現化され、その結果、相互接続部及び遅延が非常に短くなる。

【００７３】

オペレーション６８に密接に関連するのは、具現化に必要な形状及びリソースを推定する機能である。一実施形態では、この機能は、ＲＴＬコンポーネントに対するリソースを推定するためにマッピングを遂行する。別の実施形態では、このマッピングは、ターゲットとするチップアーキテクチャーに特有のものである。これらリソース推定は、特定コンポーネントのロジック要件及び入力／出力要件を推定して、ターゲットアーキテクチャーでモジュールを具現化するように設計された合成に基づいている。更に、一実施形態では、この機能は、コンポーネントに対するタイミング遷移も推定する。

【００７４】

図７は、２つのバスＡ［３１：０］及びＢ［３１：０］を加算して第３のバスＯ［３１：０］を形成するアダーの一例を示す。このアダーを具現化するのに必要なロジックエリアは、具現化を推定し、必要なリソースと、入力から出力への内部遷移遅延とを決定する変換を通して推定される。ある態様では、例えば、アダーは、各々１６個のルックアップテーブル（ＬＵＴ）より成る２つのロジックアレイブロック（ＬＡＢ）を使用して具現化することができる。

【００７５】

オペレーション６５－６７及びオペレーション７０は、ここに例示する合成変換で、例えば、ロジック因子分解（オペレーション６５）、ロジックマッピング（オペレーション６６）、ロジック最適化（オペレーション６７）、及び抽象化（オペレーション７０）であり、ＲＴＬネットリストによって表されたコンポーネント及び接続が変更されて、設計状態、例えば、タイミング、電力を改善する機能的等価回路を生じさせる。これらの変換は、コンポーネント及びそれらの相互接続部を追加又は除去することができる。変換の例は、コンポーネントの複写を遂行し、又は一体的ＲＴＬコンポーネントを分割することを含む。

【００７６】

ここに例示する実施形態は、Ｉ／Ｏ、異なるサイズのメモリ、ＣＰＵ、及びＤＳＰに対して存在する非常に大きなクラスの具現化選択の非常に簡単なケースを表す。異なる設計は、これらのリソースを異なる仕方で使用することを希望する。本発明の抽象化変換であるオペレーション７０は、タイミング情報、接続されたコンポーネントの位置、各リソース形式の利用性、及び配線の利用性に基づき、具現化を変化させることができる。抽象化変換は、オペレーション６８の生成／洗練化変換と同様である。オペレーション６８は、将来の繰り返しにおいて維持され評価される複数の交互の具現化を生成するが、抽象化オペレーションは、より詳細な具現化から抽象的コンポーネントへと抽象化する。抽象的コンポーネントの種々の具現化が考慮され、最良の具現化が選択されて、元の具現化に置き

換えられる。この能力は、考えられる全てのアーキテクチャー上のマッピング選択を列挙して、それらの全部を、マッピング、配置及び配線を通して実行する別の具現化を回避する。

【００７７】

抽象化変換の一例が、アダーツリー分解オペレーションを示す図１１に示されている。アダーツリー分解は、ｎ入力アダーをｍ入力アダーツリーへ分割する。配置から導出される遅延情報がないと、この最適化は、どこの入力がどこに位置するアダーへ入るかの情報をもたず、入力到着時間のおおよその推定に基づきツリーを形成することしかできない。この例では、全ての入力がレジスタから到来する場合に、それらは、ほぼ同じ到着時間を有する。分解は、リーフノードに対する（ａ、ｂ）、（ｃ、ｄ）及び（ｅ、ｆ）組合せを取り上げる。しかしながら、入力ｂ及びｄ、ａ及びｃは、互いに接近して配置されてもよい。配置情報と共に、リーフノードに対する（ａ、ｃ）、（ｂ、ｄ）、（ｅ、ｆ）組合せを取り上げるのがよい。これは、非常に良好なタイミングを出力に生じる。

【００７８】

ゲートツリー分解の別の抽象化例が図１２に示されている。合成フローのクリティカルステップは、３２入力ＡＮＤゲートのような多数の入力をもつ大きなゲートをツリー表現へ分解することである。この段階は、通常、フローにおいて早期に一度遂行され、ツリー分解に関する判断は、大きなゲートのドライバの位置に関する情報を含まない。本発明は、ゲートツリー分解及び再構成を、配置及びタイミングが分かった変換として含む。重要度の最も低い最も早期に到着する入力は、ツリーのリーフレベルで入れられ、そして近傍にある他の重要度の低い入力とグループ編成される。タイミングが因子でないときには、信号ドライバの位置によって入力信号がグループ編成される。

【００７９】

最適化ロジック変換であるオペレーション６７は、タイミング又は電力のような設計目的に対して最適化するようにネットリストを変化させる。最適化変換の一例が、図１３Ａに示すスライシングオペレーションである。広いプリミティブの入力又は出力が遠く離れている場合には、プリミティブを分割することが効果的である。この最適化は、配置情報に基づいて遂行するしかない。以下の例は、出力が非常に離れた２ビットメモリａ［１：０］に対するこのケースを示す。このメモリは、２つのフリップ－フロップに分割され、これを、次いで、それらの出力に非常に接近して配置することができる。

【００８０】

別の例では、コンポーネントがファンアウト又はファンイン信号の位置に基づいて分割される。例えば、図１３に示す例は、メモリのファンアウトの位置に基づいて３つのクラスターに分割されたメモリを示す。従って、単一のボックスとして示された元のコンポーネントは、対応する負荷に基づいてスライスされた３つの新たなコンポーネントを生成するように分割されている。コンポーネントの入力信号に基づいて同様の分割を適用することができる。この最適化は、一般的なもので、メモリに限定されない。

【００８１】

ここに例示する別のオペレーションは、図１４に示すロジック複写である。複写の条件は、分割に非常に類似している。入力又は出力が遠く離れたコンポーネントの場合、コンポーネントのコピーを取って、クリティカル負荷の近くに配置することが効果的である。この最適化は、配置情報に基づいて遂行するしかない。以下の例は、出力が非常に離れたコンポーネントａに対するこのケースを示す。これは、２つのインスタンスａ＿１及びａ＿２に分割され、これを、次いで、それらの出力に非常に接近して配置することができる。これは、ドライバのファンアウトが高であるときには、非常に一般的である。インスタンスの１つのコピーのみが所与の物理的範囲内に保存される。

【００８２】

ここに例示する別のオペレーションは、図１５に示すシャノン展開である。アダー又はマルチプライヤのような、大きな遅延を伴うＲＴＬ要素の入力コーンにおけるロジックの場合に、クリティカル入力ネットは、タイミングを改善するために「先に出す(pulled ah

ead)」ことができる。ロジックが複写され、クリティカルネットが一定入力０及び１と置き換えられ、そしてｍｕｘを使用して、どの演算子コピーが出力であるか選択するクリティカルネットで２つの演算子の出力を選択する。２つのロジックコピーは、一定の入力に基づいて更に簡単化することができる。この場合も、これは、ロジックの位置の知識と、ロジックを駆動するクリティカルネットのドライバとで最良に遂行される最適化である。

【００８３】

ここに例示する更に別のオペレーションは、図１６Ａ及び１６Ｂに示すＭｕｘ／ＰＭｕｘ（ＰＭｕｘは、１ホットエンコード選択をもつｍｕｘとして定義される）コラプシング及びタイミング駆動分解である。大きなＭｕｘは、商業的回路において非常に一般的である。ｍｕｘを分解することは、上述したアダーツリー及びＡＮＤ／ＯＲツリー分解に類似しているが、選択ロジックは、Ｍｕｘ分解をより困難にする。というのは、後で到着する入力をツリー内で移動すると、ツリーの構造に影響するだけでなく、選択ロジックにも影響するからである。他の分解と同様に、本発明は、適切な分解を決定するために配置及び配線に基づくタイミング情報を含む。

【００８４】

オペレーション６９は、配線更新である。本発明の増分的な繰り返し方法は、設計の性能、ノイズ感度、収率、面積及び電力を改善するために集積回路の良好な配線可能性を与える。増分的な繰り返しプロセスは、単位面積当たりに要求される配線リソースの密度であるチップ上の配線混雑を徐々に改善することができる。

【００８５】

ここに述べる多数の変換は、ＦＰＧＡにより消費される電力に影響する。例えば、メモリを分解する仕方（列形態・対・行）は、それが消費する電力に影響する。行分解は、あまり電力を使用しないが、付加的なマルチプレクシングを要求し、これは、付加的な遅延を導入する。電力消費を最適化するための行・対・列の分解の決定は、本発明において遂行することができる。というのは、合成と配置との間の本発明の密接な接続、即ち正確な遅延情報が得られるからである。

【００８６】

オペレーション６３は、配置変換、又は更新配置変換である。配置変換は、ＲＴＬオブジェクト、非マップインスタンス、又はチッププリミティブレベルインスタンスのようなネットリストインスタンスの位置を変更し、それにより、ルータオペレーションと共に、回路におけるネットの長さ及び遅延を決定する。

【００８７】

配置変換は、ネットリスト及び配置の成熟度に基づいて種々の配置方法を使用することができる。ここに例示する実施形態では、本発明のプレーサーは、増分的アルゴリズムを使用する。増分的アルゴリズムとは、入力の小さな変化に応答して、アルゴリズム出力に増分的変化を生じさせるものである。例えば、力指向(force directed)配置のようなグローバルな配置は、成熟度の低いネットリスト及び配置を配するために使用することができる。力指向配置（ＦＤＰ）方法は、本発明におけるグローバルな配置の好ましい選択肢の１つである。というのは、ＦＤＰの繰り返しで増分的配置変化が生じるような増分的方法だからである。典型的に、ＦＤＰは、ネットをモデリングすると共に、重畳するインスタンスをどのように拡散すべきか決定するために、二次プログラミング技術を使用する。

【００８８】

一実施形態では、第１ステップＦＤＰは、インスタンスを相互接続するネットしかモデリングしない非制約二次プログラミング問題を解決する。この初期の解決策は、通常、非常に高い混雑度を有する。次いで、ＦＤＰは、拡散力を繰り返し構成して、インスタンスを過剰混雑（高インスタンス使用）のエリアから過少混雑（高リソース利用性）のエリアへ移動させる。これら繰り返しステップの性質は、ＦＤＰを増分的なアルゴリズムにすることである。これら繰り返しと繰り返しとの間にネットリスト又は他の設計状態データに対する変化をなすことができる。これらの状態変化が増分的であるときには、それにより生じるＦＤＰの変化も、設計状態変化がなされなかった場合に生じるものに対して増分的

でなければならない。

【００８９】

ＦＤＰの種々のアルゴリズムがあるが、過剰混雑のエリアを解消するためにインスタンスを移動すべき方向を計算するという基本的概念を全てが共有している。所与の配置において、ネットで接続されるインスタンスは、インスタンスとインスタンスとの間の二次距離に比例して互いに吸引力を及ぼすものと仮定される。この以前の研究では、全てのインスタンスが互いに反発し、インスタンスにとってサイトが適切でなくても全ての配置サイトへ吸引される。次いで、インスタンスは、システムが最小エネルギー状態で平衡となるまで移動される。従って、ＦＤＰ方法は、インスタンスに作用する合計力の方向にインスタンスを移動することに基づく。

【００９０】

１つの態様において、本発明は、多数のモデム再プログラム可能なチップ及び幾つかのＡＳＩＣ設計フローからの異種リソースに対処するための新規な異種リソース配置を提供する。例えば、ほとんどのＦＰＧＡは、特定のサイトでしか使用できないＩＯ、ＤＳＰ、ＲＡＭ、ＬＵＴ、ＦＦ、等の種々の予め定義されたチップリソースを有している。これらの予め定義されたリソースは、ＦＰＧＡチップの前拡散特性の結果である。各リソースサイトは、サイトに配置できるインスタンスの数に制限がある。例えば、Ａｌｔｅｒａ Ｓｔｒａｔｉｘ－IIチップの場合、ＬＡＢサイトに配置できるＬＵＴ又はＦＦは１６個以下であり、又、５１２バイト、４Ｋバイト、及び６４Ｋバイトを保持する３個の個別のＲＡＭサイトがある。

【００９１】

ここに例示する実施形態では、本発明の増分的配置は、異種リソースの問題に対処する。ＦＰＧＡ、構造化ＡＳＩＣ、及びあるＡＳＣＩチップでは、配置エリアにわたってしばしば均一に分布されない幾つかのサイトのみにリソースが配置されることがある。以前の全てのＦＤＰを含むほとんどのグローバルなプレーサーは、仮定された異種リソースを有し、いずれのインスタンスも、その形式に関わらず、チップ境界内の有効エリアに配置することができる。この以前の解決策は、全てのインスタンスを簡単な直線的オブジェクトとして処理できるので、配置の問題を簡単化し、そしてこれらのオブジェクトが重畳せずにチップ境界内に配置される限り、配置は合法的である。この簡単な長方形モデルは、ある形式のインスタンスを不充分なリソースに隣接して配置できるようにする。この仮定は、異種リソースの場合に、各リソースが、インスタンスを配置しなければならない特定の１組のサイトを有することを無視している。この「結合」配置は、重畳をもたないことがあるが、実際のリソース形式を考慮したときに、配置が少しも合法的でないことがある。シミュレーテッドアニーリングプレーサーにおけるある以前の研究では、リソース情報が考慮されているが、これらのプレーサーは、静的にマップされたネットリストのみを配置し、ＲＴＬオブジェクトは配置しないように使用されている。更に、シミュレーテッドアニーリングは、非常に小さな設計に使用され、ランタイムのために大きな設計には困難である。

【００９２】

１つの態様において、本発明は、各個別のリソースサイトを別々にモデリングし、全ての配置変換において、リソース要件がプレーサーによって最適化されるようにする。１つの態様において、本発明は、「レイヤ」と称される、任意の数のサイト形式をモデリングする。これらのレイヤは、各インスタンスに対する拡散力を決定するのに使用される。一実施形態では、レイヤは、初期化段階において生成される。レイヤは、チップに存在する各リソース形式に対して生成される。レイヤのリソースサイトは、レイヤの供給分布においてそれらの位置に記録される。分布は、位置における値が、その位置における供給の値を与えるようなマトリクス状の二次元データ構造である。

【００９３】

各インスタンスは、それがリソースを消費するところのレイヤ（１つ又は複数）に指定される。単一リソース形式を消費するインスタンスは、プリミティブインスタンスと称さ

れ、多数のリソース、非プリミティブを消費するものである。非プリミティブの一例は、ＬＵＴ及びＦＦサイト形式の両方を消費する状態マシンである。レイヤに指定された各インスタンスによって使用されるリソースは、レイヤ使用分布に記録される。本発明の方法は、リソースを有している全レイヤ上のエリアを記録することにより取り扱われる非プリミティブを与える。これらの使用貢献は、次いで、非プリミティブレイヤの各々に対する力の計算に影響を及ぼす。

【００９４】

レイヤに対して、その使用と供給分布との間の差は、そのレイヤに対する混雑分布である。以前のＦＤＰ方法と同様に、この混雑分布を使用して、そのレイヤにおける各インスタンスの力を計算する。

【００９５】

非プリミティブインスタンスの力は、そのリソースレイヤの各々から重み付けされた力の平均を取り出すことにより、又はそれらリソースのローカルの混雑に基づいて計算される。各レイヤに適用される重みは、均一な重みであるか、又はレイヤのリソースの相対的な離散性に基づく重みである。リソースの離散性は、どれほど離れてリソースが配置されるか、リソースがどれほど希薄であるか、或いはどれほど均一に又は非均一にリソースが分布されるかによって特徴付けることができる。

【００９６】

一実施形態では、多数の具現化が考えられるコンポーネントの力は、非プリミティブインスタンスの場合と同様に計算される。この力は、具現化のリソースレイヤの各々から重み付けされた力の平均を取り出すことで計算される。各具現化のリソースに適用される重みは、均一な重みであるか、又は所与の具現化が選択される確率に基づく重みである。

【００９７】

本発明の効果は、インスタンスの力が、同じリソース形式及びその形式に対するリソース供給を使用する他のインスタンスのみに依存することである。例えば、インスタンスＡ及びＢの各々がリソースＣを使用する部分を有する場合には、インスタンスＡ（又はリソースＣを使用するインスタンスＡの部分）にかかる力は、リソースＣを使用するインスタンスＢの部分に依存すると共に、配置に対して使用可能なリソースＣにも依存する。異なるレイヤにおけるインスタンスは、互いの拡散力に影響しない。

【００９８】

１つの態様において、グローバルなプレーサーが終了するときに、各インスタンスは、その形式に適した有効サイト又はその付近にあり、僅かな移動で配置を合法化することができる。この解決策は、全てのインスタンスを単一形式としてモデリングし、全てのリソースエリアを結合し、次いで、その結合されたエリアにわたってインスタンスを拡散することを必要とした以前のＦＤＰに比して新規である。

【００９９】

ここに例示する実施形態では、本発明のアーキテクチャー上の物理的合成は、リソース利用の問題に対して改善を与えることができる。チップリソースが回路の要件を越えるケースがしばしばある。例えば、ＦＰＧＡ設計では、チップ又はそれが具現化される部分が２５６個のＬＵＴを有するときに、具現化されるべき回路が１５０個のＬＵＴを要求することがある。この問題は、リソース利用問題と称される。リソース利用問題が無視されるときには、リソースにわたり変化する密度での配置によってたとえ良い結果が得られたとしても、プレーサーは、典型的に、使用可能なリソースにわたり回路インスタンスを均一に拡散する。以前のプレーサーは、この問題を無視するか、又は特別な「フィラー」インスタンスを挿入している。フィラーインスタンスは、回路に接続される接続性をもたない特別なインスタンスである。「フィラー」インスタンスの使用も、問題である。というのは、それらのインスタンスに対して位置を決定しなければならないからである。

【０１００】

ここに例示する実施形態では、本発明は、エリア除去方法を使用してリソース利用問題を解決する。力の発生と同様に、各リソースレイヤは、別々に考えられる。エリア除去方

法では、リソースは、そのクオリティに基づいて利用され、クオリティの低いリソースが除去される。クオリティメトリックが最初に決定され、次いで、リソース供給が分析されて、リソースのランキングがそのクオリティに基づいて決定される。次いで、プレーサーにより配置サイトとして考慮することから低クオリティ部分が除去される。配置の変化はリソースのクオリティに影響するので、ランキング及び除去は、配置プロセス中に何回も遂行される。従って、このプロセスは、本発明による設計状態の繰り返し及び増分的改善に良く適している。

【０１０１】

一実施形態では、ランキングを形成するのに使用されるクオリティメトリックは、使用からのリソースの距離に基づく。力を計算する１つの方法の副産物は、グリーン関数でのレイヤの密度分布のコンボリューションである。このコンボリューションの結果は、より高いポイントがリソースの需要を指示し、より低いポイントが重要の欠乏を表すトポロジーマップとして見ることができる。分布は、個別ボックスで構成されるので、これらのボックスは、コンボリューション結果に基づいて分類することができる。次いで、コンボリューション分類順に最低の値をもつリソースでスタートして、要求されたリソースが除去されるまで、供給を考察してリソースを除去することにより、除去されるべきリソースを決定することができる。１つの態様では、この方法は、そのレイヤに対するインスタンス需要を満足するに足るリソースがあり且つチップを配線できるように充分なリソースを残すことができる。

【０１０２】

或いは又、ここに示す他の実施形態では、本発明は、力レンジ方法を使用してリソース利用問題を解決する。力レンジ方法では、各インスタンスに作用する力は、複数の力レンジからの重み付けされた力平均である。１つの態様では、ショートレンジの重み付けされたファクタは、高いローカル密度が大きな力を生じさせるショートレンジ領域におけるインスタンスの密度に比例する。従って、この比例は、重畳を減少するようにインスタンスの拡散ンを向上させることができる。

【０１０３】

力レンジ方法では、インスタンスに加えられる力は、インスタンスの近傍におけるインスタンス密度に依存する。一般的な考え方として、インスタンスの拡散力は、インスタンスをその近傍において合法化するのに必要なエリアに依存しなければならない。全てのインスタンスが小さな近傍域で重畳している最も極端に混雑したケースでは、各インスタンスの力は、全てのインスタンス及び全てのリソースの位置に基づいて計算される。インスタンスがその付近に他のインスタンスをもたずに、あるリソース上に直接載っている最も混雑度の低いケースでは、インスタンスは、何ら力をもたない。これら両極端間のケースでは、力は、インスタンスを合法化するのに必要なエリアにおけるインスタンス及びリソースに依存する。

【０１０４】

一実施形態では、力のレンジは、ローカル、中間、及びロングレンジの力に区分化することができる。他の実施形態では、より大きな又はより小さな力レンジを使用することができる。一般的に、近傍域に対する合法的エリア、及び各合法的レンジに対する力を決定することは、計算及びメモリリソースとの兼ね合いである。１つの態様では、グリーン関数のサイズを変えることで力が計算される。ロングレンジのグリーン関数は、全配置エリアをカバーし、小さなグリーン関数は、例えば、平均インスタンスエリアの５倍の半径をもつ円形エリアをカバーし、そして中間レンジのグリーン関数は、例えば、平均インスタンスエリアの１０倍の半径を有する。インスタンスの力は、インスタンスのローカル、中間及びロングレンジの力の重み付けされた和である。適用される重みは、インスタンスの隣接域における密度によって決定される。隣接域が非常に高密度である場合には、ロングレンジの力が非常に高い重みを有し、そしてローカルの重みが０である。低密度エリアにおけるインスタンスは、ゼロのロングレンジ重みと、高いローカル重みとを有する。

【０１０５】

本発明の方法の別の態様は、アーキテクチャー上の構成を具現化するのにどのリソースを使用すべきか決定する重要なアーキテクチャー上の判断を決定する能力である。アーキテクチャーレベルでは、ＦＰＧＡに関して、小さなＲＡＭを５１２ビットのＲＡＭリソースへマップすべきか又は４ＫビットのＲＡＭリソースへマップすべきか、といった多数の判断がある。他の例では、マルチプライヤ具現化の判断、及び上述したケース、例えば、アダーツリー分解が含まれる。本発明は、これらの特定例に限定されない。配置情報が得られる状態では、本発明は、設計目的を満足する重要なアーキテクチャー上の具現化判断を洗練化する。ここに示す例は、１ｋビットメモリを、２つの５１２ビットリソース又は単一の４ｋビットリソースのいずれかに指定できるケースである。この具現化は、１ｋビットメモリが接続されるロジックが、５１２ビット又は４ｋビットのいずれかのサイトに非常に接近して位置される場合には、首尾良く具現化するのが非常に厳しい。１ｋメモリの接続ロジックが５１２ビットリソースに非常に接近し、４ｋビットリソースがより遠くにあるケースでは、４ｋリソースへのマッピングが最適でないことにより実質的に低性能の回路を招く。この及び他のアーキテクチャー上の判断を行うのに、配置情報を使用することが重要である。

【０１０６】

ここに例示する実施形態では、具現化の洗練化は、インスタンスがマップされる各レイヤに対して柔軟なレイヤインスタンスのエリアの一部分を使用に含ませることにより取り扱われる。本発明の１ｋビットの例では、インスタンスのエリアが５１２レイヤ及び４ｋレイヤの両方に部分的に含まれる。インスタンスの力は、潜在的なレイヤに対する力の重み付けされた和をとるか、又は最小の大きさの力をとることにより決定される。最小の大きさの力をとることの背景にある合理性は、この力に関連したレイヤが低い近傍密度を有していなければならないことである。

【０１０７】

ここに例示する他の実施形態では、リソース具現化は、レイヤの使用において多数のあり得るリソース具現化を有するインスタンスを含まないことによりスタートする。エリア除去オペレーションが全てのレイヤに対して遂行された後、これらの柔軟な具現化インスタンスが考えられる。柔軟な具現化インスタンスの場合、そのあり得るレイヤ各々の潜在的な供給が考えられる。潜在的な供給は、エリア除去オペレーションによって全供給から除去されたエリアである。具現化レイヤの各々における潜在的な供給を検査して、除去されたエリアにインスタンスが配置された場合に最も破壊的でない除去されたエリアをどのレイヤが有するか決定する。次いで、その最も破壊的でないレイヤにインスタンスが指定される。

【０１０８】

指定リソース変換（オペレーション６４）は、特定チップリソースへのインスタンスの指定を決定するという役割を果たす。力指向配置、シミュレーテッドアニーリング、モングレル、ｍｉｎ－ｃｕｔ配置、数値最適化による配置、進化ベースの配置、及び他のディテール配置アルゴリズムを含む種々の配置アルゴリズムをこのオペレーションに使用することができる。

【０１０９】

本発明のほとんどの実施形態は、ＨＤＬ設計合成ソフトウェアプログラムに使用するように意図されるが、本発明は、必ずしも、そのような使用に限定されない。他の言語及びコンピュータプログラムの使用が考えられる（例えば、コンピュータプログラムは、ハードウェアを記述するように書かれ、従って、ＨＤＬでの表現が考えられ、そしてコンパイルされるか、或いは本発明は、ある実施形態では、ＨＤＬを使用せずに生成された例えばネットリストのようなロジック表現を割り当て及び再割り当てできる）が、本発明の実施形態は、ＨＤＬ合成システムでの使用に関して説明され、特に、ベンダー特有の技術／アーキテクチャーを有する集積回路に使用するように設計されたものについて説明する。良く知られたように、ターゲットアーキテクチャーは、典型的に、プログラマブルＩＣの供給者によって決定される。ターゲットアーキテクチャーの一例は、カリフォルニア州サン

ノセのＸｉｌｉｎｘ社からのフィールドプログラマブルゲートアレイである集積回路のプログラマブルルックアップテーブル（ＬＵＴ）及び関連ロジックである。ターゲットアーキテクチャー／技術の他の例は、Ａｌｔｅｒａ、Ｌｕｃｅｎｔ　Ｔｅｃｈｎｏｌｏｇｙ、Ａｄｖａｎｃｅｄ　Ｍｉｃｒｏ　Ｄｅｖｉｃｅｓ、及びＬａｔｔｉｃｅ　Ｓｅｍｉｃｏｎｄｕｃｔｏｒのようなベンダーからのフィールドプログラマブルゲートアレイ及び複雑なプログラマブルロジック装置におけるこれらの良く知られたアーキテクチャーを含む。又、ある実施形態の場合、本発明は、特定用途向け集積回路（ＡＳＩＣ）に使用することもできる。

【０１１０】

本発明の一実施形態は、ＣＤ　ＲＯＭ又は磁気ハードディスク又は光学ディスク或いは種々の他の記憶装置のようなマシン読み取り可能な媒体に記憶されるコンピュータプログラムとして具現化される回路設計及び合成コンピュータ支援設計ソフトウェアである。更に、本発明の方法の多くは、従来の汎用コンピュータシステムのようなデジタル処理システムで遂行することができる。１つの機能しか遂行しないように設計又はプログラムされた特殊目的のコンピュータも使用できる。

【０１１１】

図１７は、本発明に使用できる典型的なコンピュータシステムの一例を示す。このコンピュータシステムは、ＨＤＬコードで記述された設計のロジック合成を遂行するのに使用される。図１７は、コンピュータシステムの種々のコンポーネントを示しているが、それらコンポーネントを相互接続する特定のアーキテクチャー又は仕方を表すことを意図したものでないことに注意されたい。というのは、そのような細部は、本発明に密接に関係したものではないからである。図１７のアーキテクチャーは、例示のためのものに過ぎず、本発明に関連して使用されるコンピュータシステム又は他のデジタル処理システムは、この特定アーキテクチャーに限定されないことに注意されたい。又、より少数のコンポーネント、又はおそらく、より多数のコンポーネントを有するネットワークコンピュータ及び他のデータ処理システムも、本発明に使用できることが明らかであろう。図１７のコンピュータシステムは、例えば、アップル・マッキントッシュ・コンピュータである。

【０１１２】

図１７に示すように、データ処理システムの一形式であるコンピュータシステム１０１は、マイクロプロセッサ１０３、ＲＯＭ１０７、揮発性ＲＡＭ１０５、及び不揮発性メモリ１０６に結合されたバス１０２を備えている。インテル、又はモトローラ、又はＩＢＭからのマイクロプロセッサであるマイクロプロセッサ１０３は、キャッシュメモリ１０４に結合される。バス１０２は、これらの種々のコンポーネントを一緒に相互接続すると共に、これらのコンポーネント１０３、１０７、１０５及び１０６を、ディスプレイコントローラ及びディスプレイ装置１０８へ相互接続し、且つマウス、キーボード、モデム、ネットワークインターフェイス、プリンタ、スキャナ、ビデオカメラである入力／出力（Ｉ／Ｏ）装置、及びこの技術で良く知られた他の装置のような周辺装置にも相互接続する。典型的に、入力／出力装置１１０は、入力／出力コントローラ１０９を通してシステムに結合される。揮発性ＲＡＭ１０５は、典型的に、メモリのデータをリフレッシュ又は維持するために電力を常時必要とするダイナミックＲＡＭ（ＤＲＡＭ）として具現化される。不揮発性メモリ１０６は、典型的に、システムから電力が除去された後もデータを維持する磁気ハードドライブ、磁気光学ドライブ、光学ドライブ、ＤＶＤ　ＲＡＭ又は他の形式のメモリシステムである。典型的に、不揮発性メモリは、ランダムアクセスメモリであるが、これは、要求されない。図１７は、不揮発性メモリが、データ処理システムの残りのコンポーネントに直結されたローカル装置であることを示しているが、本発明は、システムから離れた不揮発性メモリ、例えば、モデム又はイーサネットインターフェイスのようなネットワークインターフェイスを通してデータ処理システムに結合されたネットワーク記憶装置を使用できることが明らかであろう。バス１０２は、この技術で良く知られたように、種々のブリッジ、コントローラ及び／又はアダプタを通して互いに接続された１つ以上のバスを含んでもよい。一実施形態では、Ｉ／Ｏコントローラ１０９は、ＵＳＢ（ユ

ニバーサルシリアルバス）周辺機器を制御するためのＵＳＢアダプタ、及び／又はＩＥＥＥ－１３９４周辺機器を制御するためのＩＥＥＥ－１３９４バスアダプタを備えている。

【０１１３】

以上の説明から、本発明の態様は、少なくとも一部分は、ソフトウェアで実施できることが明らかであろう。即ち、この技術は、コンピュータシステム又は他のデータ処理システムにおいて、ＲＯＭ１０７、揮発性ＲＡＭ１０５、不揮発性メモリ１０６、キャッシュ１０４、又はリモート記憶装置のようなメモリに含まれたインストラクションのシーケンスを実行するマイクロプロセッサのようなプロセッサに応答して実施することができる。種々の実施形態では、本発明を具現化するために、固定布線回路がソフトウェアインストラクションと組み合わされて使用される。従って、この技術は、ハードウェア回路及びソフトウェアの特定の組合せに限定されず、又、データ処理システムにより実行されるインストラクションのための特定のソースにも限定されない。加えて、この説明全体を通して、種々のファンクション及びオペレーションは、説明を簡単にするために、ソフトウェアコードにより遂行され又は引き起こされるものとして説明された。しかしながら、当業者であれば、それらファンクションは、マイクロプロセッサ１０３のようなプロセッサによりコードを実行することから生じるものであることが明らかであろう。

【０１１４】

データ処理システムにより実行されたときにそのシステムが本発明の種々の方法を遂行するようにさせるソフトウェア及びデータを記憶するためにマシン読み取り可能な媒体を使用することができる。この実行可能なソフトウェア及びデータは、例えば、ＲＯＭ１０７、揮発性ＲＡＭ１０５、不揮発性メモリ１０６及び／又はキャッシュ１０４を含む種々の場所に記憶することができる。このソフトウェア及び／又はデータの各部分は、これら記憶装置のいずれかに記憶することができる。

【０１１５】

従って、マシン読み取り可能な媒体は、マシン（例えば、コンピュータ、ネットワーク装置、パーソナルデジタルアシスタント、製造ツール、１組の１つ以上のプロセッサをもつ装置、等）によってアクセス可能な形態で情報を与える（即ち、記憶及び／又は送信する）メカニズムを含む。例えば、マシン読み取り可能な媒体は、記録可能／記録不能媒体（例えば、リードオンリメモリ（ＲＯＭ）、ランダムアクセスメモリ（ＲＡＭ）、磁気ディスク記憶媒体、光学的記憶媒体、フラッシュメモリ装置、等）、並びに電気的、光学的、音響的又は他の形式の伝播信号（例えば、搬送波、赤外線信号、デジタル信号、等）、等々を含む。

【０１１６】

以上、本発明は、その特定の実施形態を参照して説明された。特許請求の範囲に規定された本発明の広い精神及び範囲から逸脱せずに種々の変更がなされることが明らかであろう。従って、明細書及び添付図面は、例示のためのもので、それに限定するためのものではない。

【符号の説明】

【０１１７】

１０２：バス
１０３：プロセッサ
１０４：キャッシュ
１０５：メモリ（ＲＡＭ）
１０６：不揮発性記憶装置
１０７：リードオンリメモリ（ＲＯＭ）
１０８：ディスプレイ＆ディスプレイコントローラ
１０９：Ｉ／Ｏコントローラ
１１０：Ｉ／Ｏ装置

【図１】

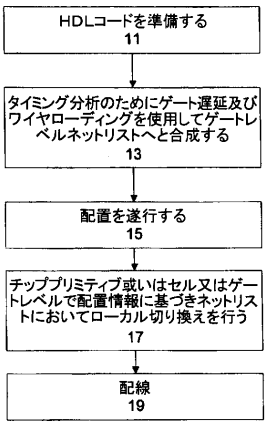


Fig. 1

【図２】

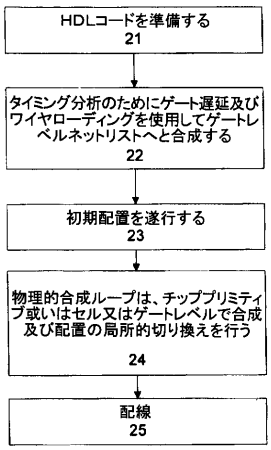


Fig. 2

【図３】

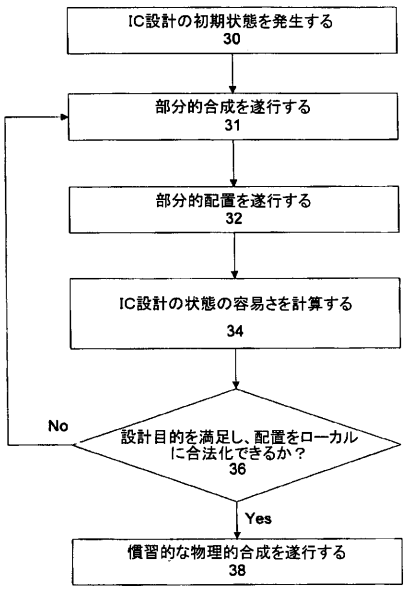


Fig. 3

【図４】

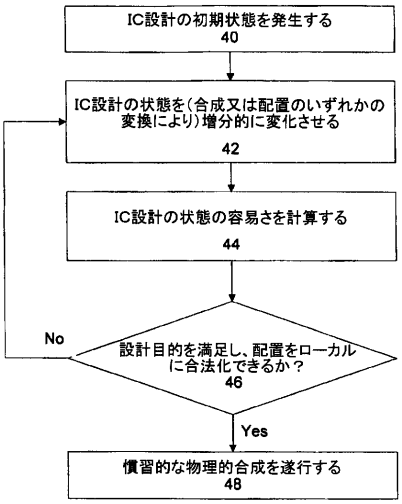


Fig. 4

【図５Ａ】

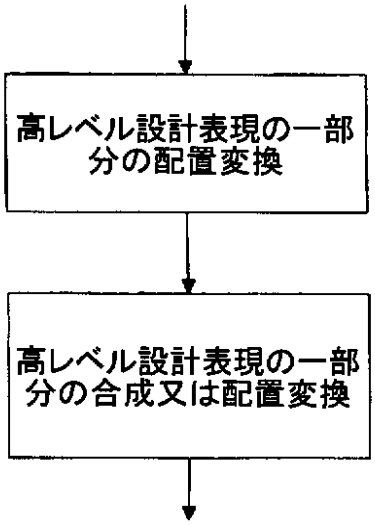


Fig. 5A

【図５Ｂ】

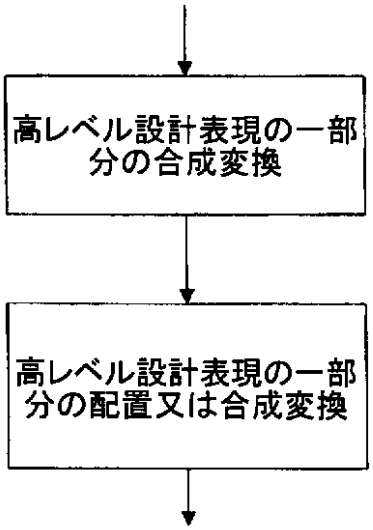


Fig. 5B

【図６】

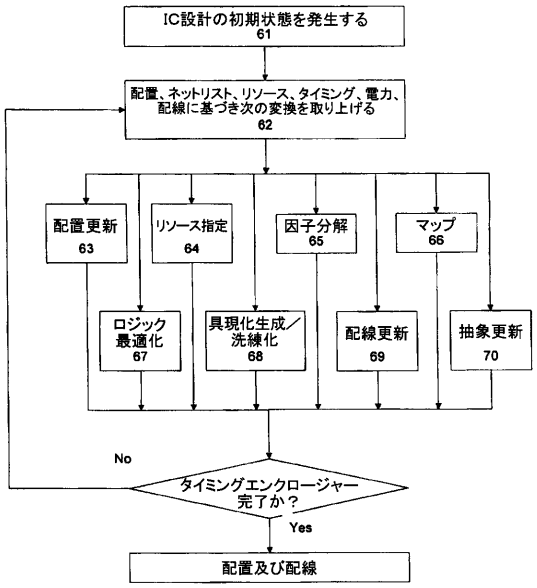


Fig. 6

【図７】

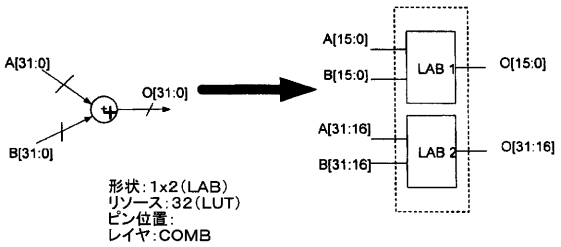


Fig. 7

【図８】

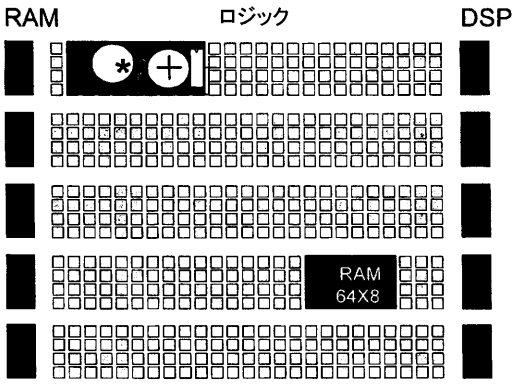


Fig. 8

【図９Ａ】

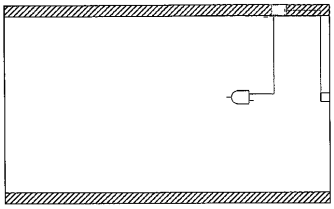

Fig. 9A

【図９Ｂ】

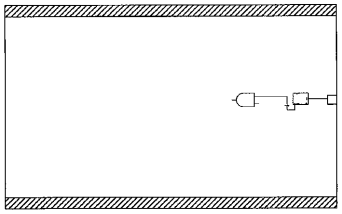

Fig. 9B

【図１０Ａ】

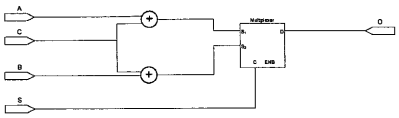

Fig. 10A

【図１０Ｂ】

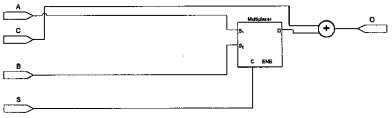

Fig. 10B

【図１１】

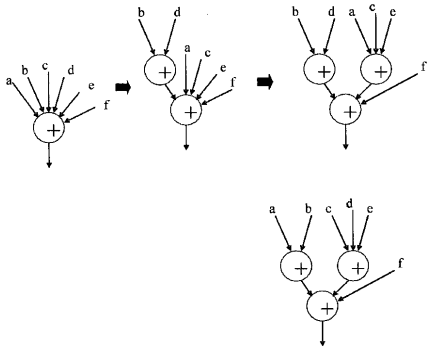

Fig. 11

【図１２】

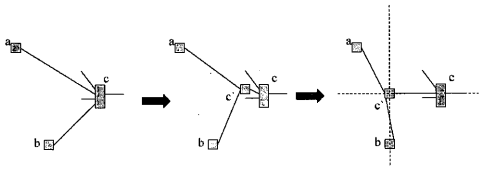

Fig. 12

【図１３Ａ】

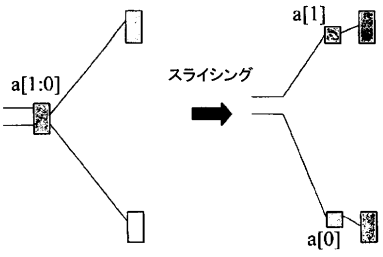


Fig. 13A

【図１３Ｂ】

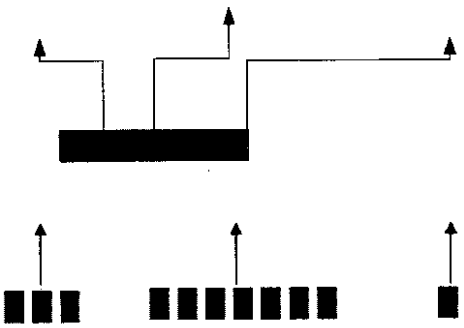

Fig. 13B

【図１４】

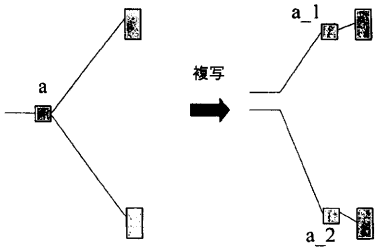


Fig. 14

【図１５】

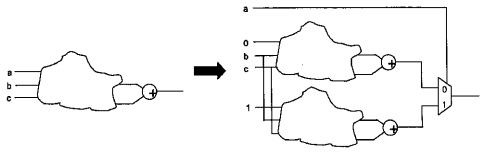


Fig. 15

【図１６Ａ】

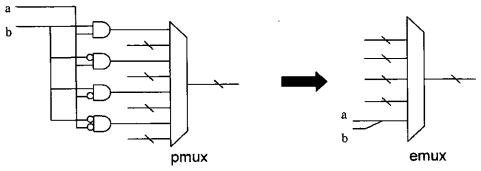


Fig. 16A

【図１６Ｂ】

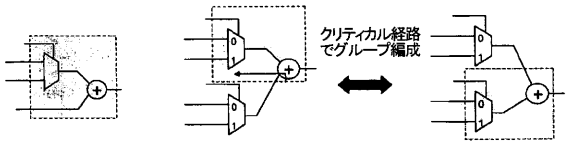


Fig. 16B

【図１７】

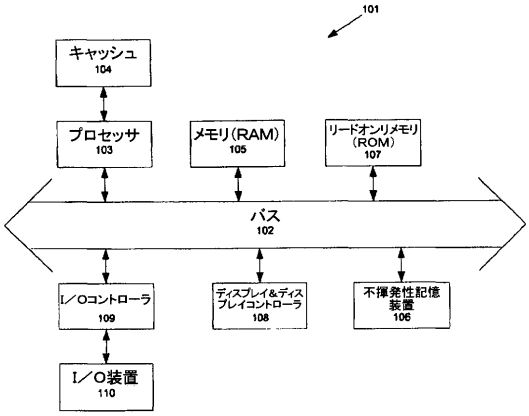


Fig. 17

フロントページの続き

(74)代理人 100109070
弁理士 須田 洋之
(74)代理人 100109335
弁理士 上杉 浩
(74)代理人 100120525
弁理士 近藤 直樹
(74)代理人 100141553
弁理士 鈴木 信彦
(72)発明者 マケルヴェイン ケニス エス
アメリカ合衆国 カリフォルニア州 ９４０２５ メンロ パーク メイ ブラウン アベニュー １１６０
(72)発明者 ルモニエ ベノワ
アメリカ合衆国 カリフォルニア州 ９４３０３ パロ アルト マリオン アベニュー ７５１
(72)発明者 ハルピン ビル
アメリカ合衆国 カリフォルニア州 ９５１３４ サン ホセ カミール サークル ４３１ ユニット １５

審査官 松浦 功

(56)参考文献 特開２００４－１６４６２７（ＪＰ，Ａ）
特開２００１－１４２９２２（ＪＰ，Ａ）
米国特許第０７３３７１００（ＵＳ，Ｂ１）
特開平０６－２６６８０１（ＪＰ，Ａ）
特開平０８－２０２７５８（ＪＰ，Ａ）
特開平１０－１７１８５７（ＪＰ，Ａ）
特開２００４－２６５２２４（ＪＰ，Ａ）
特開２００２－１２３５６３（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ １７／５０