【公報種別】特許法第１７条の２の規定による補正の掲載
【部門区分】第７部門第２区分
【発行日】平成23年8月18日(2011.8.18)

【公表番号】特表2002-526908(P2002-526908A)
【公表日】平成14年8月20日(2002.8.20)
【出願番号】特願2000-572780(P2000-572780)
【国際特許分類】
Ｈ０１Ｌ　21/82　(2006.01)
Ｇ０６Ｆ　17/50　(2006.01)
【ＦＩ】
Ｈ０１Ｌ　21/82　Ｂ
Ｇ０６Ｆ　17/50　６５４Ｋ
Ｈ０１Ｌ　21/82　Ｃ

【誤訳訂正書】
【提出日】平成23年6月27日(2011.6.27)
【誤訳訂正１】
【訂正対象書類名】明細書
【訂正対象項目名】００１１
【訂正方法】変更
【訂正の内容】
【００１１】
第１に、現在の方法は、異なる設計考察を有する複数のソースから供給される複数の設計ブロックを統合しながら、厳格な時間対市場の制約の中で階層的検査及び短いアセンブリ時間を提供するという両立性を包括的に評価し、保証するトップダウンアプローチを提供しない。
【誤訳訂正２】
【訂正対象書類名】明細書
【訂正対象項目名】０１３７
【訂正方法】変更
【訂正の内容】
【０１３７】
証明には２つの局面がある。
ａ）完全性の証明－全ＦＥＡ計量は提供されたパラメータ化方式を介して測定可能でなければならない。
ｂ）精度の証明－設計者に対する経験尺度と、収集データの精度を保証するプロセスの定義とを含む。
グルー論理
本発明は改善されたグルー論理分布及び低減方法論を更に開示する。３つの代替的なグルー論理分布機構の組み合わせは本発明の好適実施例を含む。第１として、予備設計されたブロック内に組み込まれていないグルー論理は現行ブロックへの分布用に多数コピーに複製され得る。第２として、ブロックにトップ・レベルで親和性を有さない論理は小数ブロックとして残留させられ得て、最適には、効果的なゲート独占化、配線密集度、並びに、仕入れ資金融資方法衝撃を最小化すべく据えられる。第３として、ブロック数がブロック場所及び経路指定限定を超えた場合、グルー論理がブロック・カウントが許容出きるレベルまで低減されるまでグルークラスタ・ブロック内にクラスタ化され得る。
【誤訳訂正３】
【訂正対象書類名】明細書
【訂正対象項目名】０１３８

【訂正方法】変更
【訂正の内容】
【０１３８】
図２９を参照すると、グルー論理２９１０が相互接続されたブロック間に不都合にも常駐している回路設計図が図示されており、それによってシリコン面積の大きな領域の使用が非効率化されて、著しい配線密集を作り出している。
【誤訳訂正４】
【訂正対象書類名】明細書
【訂正対象項目名】０１３９
【訂正方法】変更
【訂正の内容】
【０１３９】
図３０を参照して、より大きなトップ・レベルのブロックへの分布のためのグルー論理の多数コピーを作り出す本発明の方法の説明を始める。要素３０１０が多数付加を駆動する出力正味を有すれば、この要素は、各々が出力に対する単一のみの負荷を有している多数の要素３０１２内へこぼされる。次いで、複製された要素を駆動する各入力「コーン（ｃｏｎｅ）」（不図示）も、全てのブロック出力が達成されるまでコピーされる。同様に大きな入力ゲートは、木の頂部にオリジナル関数の２入力ゲートを伴う非反転２入力ゲートの木にまで低減される。このようにして、実質的により多くの論理が先行する相当により小さなグルー論理関数に専用化される。しかしながら、より大きなブロック間の領域からグルー論理を除去することによって、それらより大きなブロックはより効率的に配置されて、正味効率増大となる。
【誤訳訂正５】
【訂正対象書類名】明細書
【訂正対象項目名】０１４０
【訂正方法】変更
【訂正の内容】
【０１４０】
分布のために効率的に複製できない任意のグルー論理要素は、次いで、配置された要素に最も緊密な親和性を有するより大きなブロック内に好ましくは合併される。グルー論理合併は多数の規準に基づいた様式で実行され、その内の最も重要なものは合併がトップ・レベルのピン・アウト数を低減するかどうかである。よって、多数コピーが作り出されると、結果として論理の殆どが２入力ゲートから構成されるので、そうしたゲートを１つのピンがそのブロックに接続されているブロック内へ合併することはピン・カウントを２だけ低減する。２つ或はそれ以上のブロックが合併のための同等候補である際、最低のピン密度を有するブロックが好ましくは選択される。最後に最低の優先権は、好ましくは、タイミング考慮へ行く。
【誤訳訂正６】
【訂正対象書類名】明細書
【訂正対象項目名】０１４２
【訂正方法】変更
【訂正の内容】
【０１４２】
本発明は予備設計された回路ブロックを標準化されたインタフェースを有する回路へ変換する方法を更に開示する。
【誤訳訂正７】
【訂正対象書類名】明細書
【訂正対象項目名】０１４３
【訂正方法】変更
【訂正の内容】

【０１４３】
図１におけるブロック設計段階１０６で実行されるタスクは：（１）選択された回路ブロックに対する任意の喪失アブストラクトを作り出し、（２）回路ブロックをそれらのカラー（ｃｏｌｌａｒ）として既知の各標準インタフェース内へ埋め込み、そして、（３）カラー付けされた回路ブロックに対するアブストラクトから成る完全セットを作り出す。
【誤訳訂正８】
【訂正対象書類名】明細書
【訂正対象項目名】０１４４
【訂正方法】変更
【訂正の内容】
【０１４４】
図３２を参照すると、回路ブロックをカラー内へ埋め込む、本発明に従ったカラー付けプロセスが図示されている。ＢＢＤ方法論において、選択された回路ブロックはチップ・レベルでの主要な入力構成要素である。カラー付けプロセスは回路ブロック各々の回りにカラーを据え付けて、回路ブロック境界回りに標準インタフェースを作り出す。カラー付けされたブロックをチップ・レベルへ首尾よく統合するために、アブストラクトから成る完全セットをそのカラー付けられたブロックに対して作り出さなければならない。そのアブストラクトから成る完全セットを作り出す前に本発明のシステムは、アブストラクトがブロックのモデル若しくは風景、チップ・レベルでの組立で要求されるカラー付けブロックの設計、或は、プランニング・ツールである場合、選択されたブロックに対する任意の喪失アブストラクトを先ず形成する。模範的なアブストラクトは：
（１）固定状態のタイミング・アブストラクト－ＴＬＦ
（２）レイアウト遮断ファイル－ＬＥＦ
（３）立証用モデル－ボルト締めバス・ブロック・モデル
（４）システムへのブロックレイアウト拘束
図３３を参照すると、本発明に従った回路ブロックのアブストラクトから成る完全セットの作成が図示されている一方で、図３４は図３２及び図３３に図示された特徴の組合せを図示している。
【誤訳訂正９】
【訂正対象書類名】明細書
【訂正対象項目名】０１４５
【訂正方法】変更
【訂正の内容】
【０１４５】
次にカラー付けプロセスの説明に移る。そこでは、標準インタフェースが設計に使用されるべきブロック・タイプ毎に規定されていると仮定している。ブロックの内の任意のものが完全ブロック・アブストラクトを具備しなければ、このプロセスはそのブロックに対する完全ブロック・アブストラクトを形成する。
【誤訳訂正１０】
【訂正対象書類名】明細書
【訂正対象項目名】０１４７
【訂正方法】変更
【訂正の内容】
【０１４７】
次にこのプロセスは、識別されたブロックをその対応するインタフェース標準と関連させる。
【誤訳訂正１１】
【訂正対象書類名】明細書
【訂正対象項目名】０１４９
【訂正方法】変更

【訂正の内容】
【０１４９】
次の段階で、このプロセスはその識別された回路ブロックに関連されたその標準インタフェースに準じた第２カラー部を作り出す。
【誤訳訂正１２】
【訂正対象書類名】明細書
【訂正対象項目名】０１５０
【訂正方法】変更
【訂正の内容】
【０１５０】
次いでこのプロセスはその特定のインタフェースをその標準インタフェースと接続可能なフォーマットに変換する構成要素を含む第３カラー部を作り出して、第１カラー部を第２カラー部に接続する。
【誤訳訂正１３】
【訂正対象書類名】明細書
【訂正対象項目名】０１５１
【訂正方法】変更
【訂正の内容】
【０１５１】
ブロック・カラーは多数の層から構成され得る。現行、２つのカラー層（ブロック標準カラー及びシステム特定カラー）がＢＢＤ及びＳＯＣ用にそれぞれ標準化されている。図３５を参照すると、２層を含むカラーが示され、その一方のカラーが特定ブロックに対する標準であり、他方がブロックが展開されることになっている特定システムに特有のものとなっている。ブロック標準カラーは、特定システム或は統合されることになる特定前後関係の知識無しで規定され得るようなインタフェース構成要素を含む。例えば、ＢＢＤの前後関係において、特定の設計グループはＪＴＡＧ標準試験インタフェースが設計で必要とされていると決定し得る。よって、設計されるシステムの内の任意のシステムにおいて使用されるべきブロックの全てに対して、ＪＴＡＧ試験インタフェースが標準となり、よってブロック標準カラーに属する。システム特定カラー（或は適合カラー）はブロックに属するインタフェース構成要素を含むが、システム或は前後関係特定ではない。例えばデータ線に対する標準セットはパリティ・ビットを必要とし得ないが、設計される特定システムに対してパリティ・ビットが全データ線に対して必要とされている。パリティ・ビットを生成する論理は、チップ・プランニング中、ブロックと関連させられ、システム特定カラー内に常駐すべきである。
【誤訳訂正１４】
【訂正対象書類名】明細書
【訂正対象項目名】０１５２
【訂正方法】変更
【訂正の内容】
【０１５２】
ＢＢＤにおける２つのカラー間の別の相違は、ブロック標準カラーがフロントエンド受容及びチップ・プランニング（チップ・プランニングは初期カラーが必要とされたチップ・プランニング関数をより良好に実行すべく沈下プロセスの一部として設計されることを要求し得る）に先行してつけられるが、システム特定カラーはチップ・プランニング後にだけ付加され得る。
【誤訳訂正１５】
【訂正対象書類名】明細書
【訂正対象項目名】０１５３
【訂正方法】変更
【訂正の内容】

【０１５３】
２つのカラータイプ間のより微細な相違は、ブロック標準カラーに対する標準セットがＳＯＣにおける標準セットよりも範囲において相当に狭くなり得ることである。例えば、特定のパワー・インタフェースはＢＢＤに対して標準となり得るが、特定の会社に対してだけであり、その他の会社はブロックに対するその標準パワー・インタフェースに一致する必要性はない。結果として、会社の外側からのブロックはシステム特定カラーを必要とし、それは標準パワー・インタフェースを会社のものに変換する。これはＳＯＣとは対照的であり、そこでは業界全体にわたるインタフェース標準が存在し、ブロック標準カラー内に常駐している。ＳＯＣにおける最終ゴールは業界全体にわたる標準である標準カラーを作り出すことである。そうしたカラーを有するブロックはソケットに入れられたブロックと呼称される。将来、もしカラーの外観的特徴の全てが業界全体にわたるものであれば、システム特定カラーの追加的な層状化の必要性はなく、よってブロックはプラグ・アンド・プレイの理想へより近づけられる。
【誤訳訂正１６】
【訂正対象書類名】明細書
【訂正対象項目名】０１５５
【訂正方法】変更
【訂正の内容】
【０１５５】
カラー及びブロックはソフト、ファーム、並び、ハードの様々な組み合わせであり得る。ブロックのハード性に関して長所及び短所があるのと同じように、カラーのソフト性、ファーム性、並びに、ハード性の長所及び短所がある。例えば、もしブロック自体がソフトであれば、そのブロック標準をソフトのままとして、システム特定カラーが追加される際、ブロック全体が合成化、据え付けられ、そして最終的なレイアウトへの変換用に平坦となるように経路指定され得るように為すことが適切である。しかるにもしブロックがハードであれば、小量のみの標準機能変換を伴う物理的インタフェース発行を際だって取り扱うべくハード・ブロック標準カラーを使用することが適切であり得て、それはシステム特定発行を取り扱うソフトのシステム特定カラーが殆ど機能変化を含むからである。
【誤訳訂正１７】
【訂正対象書類名】明細書
【訂正対象項目名】０１９８
【訂正方法】変更
【訂正の内容】
【０１９８】
データ転送マトリックス
データ転送マトリクスを作成するために、設計者は先ず１つのブロックから別のブロックまで転送されているデータ量をコミュニケーションマネージャーモデルに加える。次に、スプレッドシートツールを使用して、設計者は各パターンセットのためテーブルにこのデータを蓄積する。例えば、３つのブロック及びテストベンチを有するチップ用テーブルは、バイトの状態でテーブルの各項目に転送されたすべてのデータの合計を有するテーブルからテーブルまで４×４になるはずである。対角線方向はすべて０になるはずである。より多くの実用的なモデルでは、チップの中へ及びチップの外へ進むバスを考慮に入れると、テストベンチが各評価軸の項目より多くを有することになることに留意されたい。データ転送マトリクスの例を図４３のテーブルに示す。このマトリクスの背後にある設計には、テストベンチのための３つのブロックと３つのポートがある：外部メモリへのインタフェース、ＰＣＩインタフェース及び並列Ｉ／Ｏインタフェースである。テーブルに示すように、ブロック１からブロック２へ転送されたデータは１０，０００バイトであり、かつブロック２からブロック１へ転送されたデータは８，０００バイトである。このように、データ転送マトリクスを作成する第１ステップは模範的パターンセットＸのトランザクションを示す図４４に示すように、すべてのトランザクションの総計でテーブルを作成する

ことである。図４３及び図４４に示すテーブルを作成するには、設計者は次のようにスケジューラ擬似コードを修正してもよい：

列が空白ではない間に、
列から次のトランザクションを取得する；
トランザクションから送信ブロックを取得する；
トランザクションからターゲットブロックを取得する；
トランザクションバイトカウントを取得する；
トランザクションマトリクス（送信側、目標）＝トランザクションマトリクス（送信側、目標）＋１；
トランザクションマトリクス（送信側、目標）＝トランザクションマトリクス（送信側、目標）＋トランザクションバイトカウント；
目標ブロック（トランザクション）呼出す；Ｅｎｄ；
を行う。

非バスブロックからブロックへのワイヤにはある遅延（代表的には、１クロックサイクル）があるので、バストランザクションの他にタイミング列における分離したトランザクションとしてこれらを加えることが好ましい。

待ち時間マトリクス
各ブロックのクロックサイクルタイムがフロントエンドキャパシタンスですでに定義されているため、設計者は原パフォーマンスを次のようにサイクルカウントに翻訳することができる：
１．それらの仕様で定義されたサイクルに近い動作を反映するため、設計者はその既存挙動モデルに各ブロックの予測クロックサイクルを加える。このステップは送るブロックをブロックデザインタスクへ送信する前、しかし検査の後に実行することが好ましい。
２．設計者はブロックをチップモデルに一体化する。チップモデルは相互接続で同時に定義されたサイクルに近いブロックを有することになる。
３．設計者はスプレッドシートを使用して図４３及び４４に示すそれに類似のテーブルをセットアップする。転送されたバイトの数の代わりに、設計者はデータが利用可能になる時間からデータが次のブロックまたはテストブロックに到着する時間（待ち時間）までのサイクル数を各転送受取り量に指定する。
４．設計者は、新しいテーブルに示したパフォーマンス値を使用するため相互接続モデルを修正する。

図４５に模範的待ち時間マトリクスを示す。これらの修正の疑似コード例を次に
示す：
列が空白ではない間に、
列から次のトランザクションを取得する；
トランザクションから時間を取得する；
トランザクションからターゲットブロックを取得する；
ターゲットブロック（トランザクション、時間）を取得する；
Ｅｎｄ；を行う。

ここで、各ブロックは次を実行する：
ターゲットブロック（トランザクション、時間）；
ブロックの機能を実行する；
トランザクション時間を時間＋遅延＋待ち時間（このブロック目標）；
列に対する新しいトランザクションを並べ替える；

Ｅｎｄ。

図４４の「０」を読取る項目が、データが転送されずかつ待ち時間マトリクスに適用不可能であることを表示していることに留意されたい。

５．設計者はテストベンチを修正して設計データフローの知識を使用して予測されたれた相互接続サイクルカウント遅延を有するチップ待ち時間要件を含むようにする。

６．設計者はそれがサイクル要件を満たしているかどうかを確かめるように設計をシミュレートする。

７．チップのサイクル要件が満たされるまで、設計者は待ち時間マトリクスを修正して検証プロセスを繰り返す。

各タイプのバス転送に利用可能な最大サイクルカウントを有するテーブルを作成するため、より厳格な待ち時間要件がよりゲート制御集約型バス相互接続方式に翻訳するため要件が満たされるまで、設計者は大きいサイクルカウントを使用してそれらから始めかつそれらを低減しなければならない。

クラスタ測定の判定
次に、データの自然なクラスタを反映するため、設計者は対角線方向の中心に最も近接した最も大きいカウントを移動させることによってデータ転送マトリクスを再組織する。このプロセスを行う多数の方法がある；好ましい方法は本明細書においてピボット運動と呼ばれる。ピボット運動の目的は、必要なピン数を最小にするための最高転送速度でブロックを集合させることである。設計者は自動的に計算を行うためのスプレッドシートをセットアップしてもよい。

集合させることがどのくらい効果的であるか測定するため、データ転送マトリクスの各サイトを正確に重みを加えなければならない。この例では、サイトに重みを加えるため、図４６に示した距離マトリクスを使用する。図４６のテーブルで、各セルにはセルが対角線方向による距離の二乗が含まれる。データ転送マトリクスサイトに重みを加えるための他の方法を使用してもよいが、より高度の測定により制限される、システムの要素のある程度の移動し易さがある間急速に集中することが配置アルゴリズムで示されているため、距離の二乗が好ましい。

次に、設計者は距離マトリクスのその対応するセルでデータ転送マトリクスの各セルを乗算し、かつセルすべてのためすべての値をまとめて加える。結果がクラスタ測定値である。図４３のテーブルのマトリクスのクラスタ測定値は４２８，２００である。クラスタ測定値が低いほど、バス集合はより効果的である。

ピボット運動ブロック
より低いクラスタ測定値を取得するように試みるには、クラスタ測定値が向上しているかどうかを確かめるため設計者は、ひとつずつ列を交換しかつすべての交換の後クラスタ測定値を再計算することによりデータ転送マトリクスをピボット運動しなければならない。下記擬似コードに示したように、サイトが並べ替えられるべき一覧表の要素である場合の並べ替えを実行することにより、列の交換が行われる。

マトリクスの電流クラスタ測定値を取得する（Ｇｅｔ　Ｃｕｒｒｅｎｔ　ｃｌｕｓｔｅｒ　ｍｅａｓｕｒｅ　ｏｆ　ｍａｔｒｉｘ）；
現行サイトに対して＝マトリクスのサイト１に、をｎ－１を実行する（Ｄｏ　ｆｏｒ　Ｃ

urrent site － site 1 to n－1 in the matrix）；
次のサイトに対して＝現行サイト＋１をｎに、を実行する（Do for Next site － Current site ＋ 1 to n in the matrix）；
現行サイトと次のサイトを交換する（Swap Next site with Current site）；
マトリクスの次のクラスタ測定値を取得する（Get Next cluster measure of matrix）；
次のクラスタ測定値＞現行クラスタ測定値なら、現行サイトと次のサイトを交換し元の場所へ戻す（If Next cluster measure ＞
Current cluster measure
Then Swap Next site with Current site back to original location.）。

さもなければ（Else）、
現行クラスタ測定値＝次のクラスタ測定値（Current cluster me8sure ＝ Next cluster）；
End；
End。

接続でなく相互接続は帯域幅であるけれども、この並べ替えは二次方程式配置アルゴリズムに類似している。設計者はこのひとつの代わりに類似の結果をもたらす他の方法を使用することができる。

上記に示すようにピボット運動させると、１１７，０００の改良クラスタ測定値を有する図４７のマトリクスが生成される。この理想化された例では、構成要素が情報を作成しないことに留意されたい。構成要素はそれらが読取ることを書込むので、列及び行はブロック３及びＰＩＯを除いて、一致するものを合計する。
これは現場で使用するためのケースではないかもしれない。

設計者は図４７に示すようなテーブルを使用してバスクラスタを定義することができる。この例はブロック１、ブロック２、ＰＣＩ及びメモリ間の高速データ転送を示す。従って、これらの構成要素は高速バスでなければならない。ブロック３とＰＩＯ間には低データ転送速度があるため、これらの設計要素は低度バス上にある。

ＰＩＯは出力のみであるが、他の構成要素はすべて双方向性である。クラスタ内部及び外部の構成要素は連通していなければならないので、図４８に示すように、設計者は２つのバス間のブリッジを作成しなければならない。

クラスタ化に基づくバスの定義
初期のクラスタ化には、好ましくは、すべての予め定義されたバス信号ネットが含まれなければならない。設計者はクラスタ内でピボット運動させて本質の内部サブクラスタを示すことができるが、ひとつの以上のバス形式がこれらのシグナルのために定義されなければ、それらを次のタスクのひとつのクラスタとして取扱う必要がある。

プロセッサのシステム及び周辺機器バスが定義されている場合、クラスタはシステムバスと周辺機器バスまたはクラスタ化情報に基づくバスに分けられる。例えば図４７のテーブルのバスマトリクスが前もって定義されたバスシグナルネットから構成されている場合、初期クラスタ化にはマトリクス全体が含まれる。ひとつの以上のバスが定義されている場

合、高速バスにあるべきブロックがひとつのバスを、かつ残りの部分が別のバスを形成する。この分割はその後、次のタスクに渡される。予め定義されたバス接続がない場合、バスはクラスタ情報に基づいた方法で定義される。ピボット運動されたマトリクスは通常、他の隣接したブロックと比較されたそれらの間の比較的高いレベルの交信を有する隣接したブロックのグループを有する。図４９のテーブルに以前のピボット運動されたマトリクスに類似の、この種類のクラスタ化を示す。図４９は、クラスタ化プロセスをより明らかにするため以前に示したそれらからの異なる例に基づいている。「＃＃」が大きい数を表すことに留意されたい。

ここの例では、３つのブロック間に高速の交信があり、かつＨを介してこれらのブロックとブロックＤ間に交信がないため、ブロックＡ、Ｂと、及びＣが１つの独立したバスクラスタを形成する。３つのブロックすべての間に高速の交信があるため、ブロックＤ，Ｅ、及びＦが別のバスクラスタを形成する。さらに、ブロックＤ，Ｅ、及びＦが２つの別々のバス、即ち、Ｄ及びＥ間の二地点間バス及び別のＥ及びＦ間の二地点間バスを形成する。ブロックＧ及びＨが第３クラスタを形成する。ＥＦ対及びＧＨ対間により低い帯域幅接続が存在する。データ転送量に依存して、Ｅ、Ｆ、Ｇ及びＨはより低いレベルの交信のためひとつのバス上またはそれらの間に双方向性ブリッジを有する２つの別々のＥＦ及びＧＨバス上に存在してもよい。

多数の異なるクラスタからオプションを選択するために、次の指針に従う：
１．比較的低い交信領域から高い交信領域の、ブロック間のカットポイントを指定して可能なクラスタを決定する。図４９のマトリクスのＣとＤ間のカットにより図５０に示したダイアグラムが生成される。ＡＢＣとＤＥＦＧＨグループ間交信量を決定するため、下方左側及び上方右側のセルを合計する。この例の場合のこの合計がＯである場合、２つのグループにはそれらの間には交信がない。これらのグループは完全に分離したバスを形成する。カットを横切って生じる交信がＯである場合のピボット運動したマトリクスを切断する。

指定グループの各々内に、効果のあるカットを見つける。生じるグループ間の交信は各グループ内より非常に少なくなければならない。図５０にひとつのカットがＤ－Ｈグループに現れ、かつ図５１に示すようにいかなるカットもＡ－Ｃグループには現れない。ＧＨグループ間のデータ転送速度は２２であるが、他のグループ内のデータ転送速度は非常に大きい数（＃＃）である。これらのクラスタはそれらの間にブリッジで２つのバスを形成することができる。

クラスタ間またはクラスタ内の交信にすべてのブロックが含まれない場合、ユーザに最適化が必要な可能性がある。待ち時間マトリクスにある一定のブロック間交信に対する非常に異なった要求がある場合、最適化することのみが重要である。例えば、図５１はＧＨクラスタはＤＥと交信しないことを示している。ＤＥ及びＥＦは交信しているが、ＤとＦは交信しない。ＤＥに対する待ち時間要求が非常に厳密な場合、それ故、設計者はバスの残り部分からＤＥ　交信を分割する必要がある。図５２から、生じるマトリクスが確かめられる。この例ではＥがＥ及びＥ’に分割されて、分離したインタフェースが２つのバスのためＥに生成されることになるため、それは２つの分離したブロックであるように見える。ブロックに２つの以上のバス・インタフェースがある場合、分離したインタフェースを効果的に使用するためこの技術を使用してもよい。

この技術が図４３の例で使用される場合、図５３に示したそれらの間のブリッジ２つのバスを有するクラスタが生成される。ひとつのバスがデータ著しい量を転送する一方、その他は非常に少ない量を転送する。ブロック３とＰＩＯ間の別のカットがクラスタ間のより低い交信にさえも生ずることになる。しかし、それがクラスタにただひとつのブロックを

残すためこれは重要なカットではなく、従ってそれは作成されない。

４．すべてのカットが作成されると、生じるクラスタ情報は次のタスクへ渡される。

このクラスタ化技術には、チップのためバス構造を生成するシステム知識が必要とされる。設計者は現行ブロック・バス・インタフェース、追加プロセッサ要求、及びバスに関するマスタ数などのデータタイミング及び実行詳細を考慮しなければならない。これらの要因は、このクラスタ化方法を使用して得られた構造からはずれることがよりよい性能または処理手順に単に追従することにより得られたものよりより低いゲートカウントを有するバス構造を生成することを示唆している。そうであれば、設計者はこのタスクを繰り返えしてクラスタ化結果を修正することを欲する可能性がある。

バスの選択
設計者がクラスタ化方法を使用してバスを定義したら、バス形式及び性能階層構造を選択しなければならない。バス階層構造とは最高性能のバスからその最低性能のバスまで低下するバスの水準である。例えば、設計に高速システムバス及び２つのより低速の周辺機器バスが含まれている場合、階層構造はシステムバスから２つの周辺機器バスまでである。

バス分類参照ライブラリによるバス属性及びサイズは、各バスのバス形式を定義するため使用されることが好ましい。ライブラリにより利用可能なバス形式の各々のため１組のバス属性が一覧表にされる。適切なバスを選択するため、設計者は現行バス・インタフェースのためのクラスタで各ブロックを分析する。何もないかまたはほとんどない場合、バスは最も類似する属性を有するバス分類参照ライブラリのバス形式が選択される。この選択プロセスの結果がバス設計を指定する、次のタスクで使用される定義されたひと組のバス及び階層構造である。

バス分類参照ライブラリ及び設計におけるブロックのインタフェースをチェックインして、バスを以下の通りに選択する必要がある：
ラスタの帯域幅及び待ち時間を満足しないバスを除去する；
２．バスがすでに定義されている場合、そのバスを使用するが、さもなければ；
３．プロセッサが存在れば、それがすでに接続しているシステムバスを使用する、さもなければ；
４．ほとんどのブロックがすでに接続しているバスを選択する；
５．それが接続されるものに、ほとんどのブロックのｅｎｄｉａｎ－ｎｅｓｓを扱うことが可能なバスを使用する（大きいｅｎｄｉａｎにはデータユニットの左端にＯビットがある；小さなｅｎｄｉａｎは右側にはない）　；
６．バスの負荷が過大な場合、多数のバスを使用する；
７．より低帯域幅のデバイスを周辺機器バスまたは（複数の）バスに分離する；
８．既存ブリッジを有する周辺機器バスを使用してシステムバスを選択する；
９．選択プロセスの終了後、ひとつの以上の選択がある場合、それが最良ツール及びモデルサポートを有することになるため、ＯＣＢ属性一覧表を最もよく満足するバス形式を選択する。

バスサイズの計算
バス待ち時間テーブルは、このステップの出発点として使用される。クラスタ化を使用して指定バスコンフィギレーションが識別されると、情報をバスのサイズを決定するのに使用可能な形態に翻訳しれなければならない。以前のタスクの例からのマトリクスでは、最初の４つのエントリはひとつのグループにクラスタ化され、かつ後の２つは２つのグループにクラスタ化される。

バスサイズの計算には、目標帯域幅が可能な限り近くに接近するまで別のバス幅値を代入して、転送されるデータ量のためかつ帯域幅を計算するため必要な帯域幅の決定が必要とされる。

目標帯域幅の決定
パターンセットのバスに必要なターゲット帯域幅の決定には次のステップが必要とされる；
１．ピボット運動されたデータ転送マトリクスの各クラスタで生ずるトランザクションすべてを加える。同様の例について続行すると、大きいクラスタで６２，６００、小さなクラスタで１００、及びクラスタ間で１，２００である。従って、図５５のマトリクスは図５４の４つのグループの各々にエントリを加えることによって生成される。
２．このパターンセットが取り込むことが予期される時間を決定する。フロントエンド受け入れタスクがこの情報を提供する。この例については、パターンセットは１ミリ秒で転送されなければならない、即ち、高速クラスタは６３，８００バイトのデーター１，２００バイトをブリッジへかつ内部の６２，６００バイトをバスへ－１１ミリ秒で転送しなければならない。帯域幅は１秒で転送可能な、ビットでの、データ量として定義される。この例では、１ミリ秒に５１０　Ｋビットが転送され、帯域幅はほぼ５１０ＭＨｚである。

バス幅の計算
帯域幅はデータが転送されているクロック周波数倍のバス（バス幅）のワイヤ数により構成される。
帯域幅はデータが転送されているクロック周波数倍のバス（バス幅）のワイヤ数により構成される。
計算は次のように実行される：
（ｕｎｔｉｌ　／　ｃｌｏｃｋ＿ｃｙｃｌｅ）　×　ｂｕｓ＿ｗｉｄｔｈ　＝ｂａｎｄｗｉｄｔｈ
ここで、
ｕｎｔｉｌは、選択されたバス形式の最小バス利用百分率である（図５９を参照）；
ｃｌｏｃｋ＿ｃｙｃｌｅは、設計のためのクロックサイクルであり；かつ
ｂｕｓ＿ｗｉｄｔｈは、バスのワイヤ数である。この値は２のべき数でなければならない。

計算のため、ｂｕｓ＿　ｗｉｄｔｈに対して２１から始め、かつ生じる帯域幅値が目標帯域幅より大きくなるまでより高い値（２２、２３、　．．．）を代入し続ける。例えば、クロックサイクルが２０ナノ秒であり、かつバス利用が２５％である場合、２に最も近いべき数に端数をまるめられたワイヤの数は、６４ビットである。ここで、
（２５％／２０　ｎｓ）＊２６　＝８００　ＭＨｚ＞５１０ＭＨｚ。

この例では、図５９のテーブルから形式４または５が選択される場合、高速クラスタのためのバスでは少なくとも６４ビットが必要とされることになる。同様に、２０　ナノ秒のサイクル時間は、より遅いクラスタのためただ８ビットが必要とされることになる。

バスの増大された利用が待ち時間を増大させるため、待ち時間情報は部分的利用機能である。例を簡単にしておくため、このような複雑さは含まれていない；それは、利用数の部分的な説明になっている。しかし一般的に、ひとつの帯域幅計算のために最小バス利用数を使用する場合、待ち時間も同様に最小に向う傾向がある。この効果を説明するために、設計者は、クラスタから最も悪いケース（最も少ない）待ち時間要求を選択する必要がある。

従って、設計者はシミュレーションで使用した待ち時間マトリクスからトランザクション

全体の待ち時間を導出すことができるが、図５９のテーブルは分離した数としてバス待ち時間データ及び転送値を示す。図５９に形式４のバスの１０通りの最大転送待ち時間を示す。最小データ待ち時間は単独データのため必要なサイクル数に近い。従って、設計者は、いずれが正味の転送待ち時間であるかを、下記に示す待ち時間マトリクス数からデータ転送時間を減算することにより計算する必要がある；
ｄｅｔａ＿ｔｒａｎｓｆｅｒ＿ｔｉｍｅ ＊ ｍｉｎ＿ｃｙｃｌｅｓ ／ ｎｕｍ＿ｗｏｒｄｓ ＊ ａｖｇ＿ｔｒａｎｓ
ここで、
ｍｉｎ＿ｃｙｃｌｅｓは、このバス形式のデータ待ち時間サイクルの最小数であり；
ｎｕｍ＿ｗｏｒｄｓは、バスにおけるワード数であり；かつ
ａｖｇ＿ｔｒａｎｓは、平均トランザクションサイズ；トランザクションマトリクス（図４４）におけるトランザクション数で分割されたデータ転送マトリクス（図４３）からのデータのバイト数である。

テーブルからの待ち時間を比較するために、設計者はトランザクションのデータ待ち時間をマイナスシミュレーションマトリクスから引いた待ち時間値を使用する新しい待ち時間マトリクスを作成しなければならない。上記例では、このテーブルは図５３に示すようになるはずである。このマトリクスの各要素は次のように計算される：［Ｒｅｓｕｌｔｉｎｇ Ｌａｔｅｎｃｙ（ｘ，ｙ） － Ｍｉｎ Ｂｕｓ Ｌａｔｅｎｃｙ ｄａｔａ （ｔｙｐｅ）］ ＊（Ｄａｔａ Ｔｒａｎｓｆｅｒ（ｘ，ｙ） ／ Ｔｒａｎｓａｃｔｉｏｎ（ｘ，ｙ） ｔ ｂｕｓ ｓｉｚｅ））。

システムバスクラスタの最少数は２５である。この値は帯域幅のために必要なバスの形式の最大転送待ち時間より大きくなければならない。その数はバス形式４の転送待ち時間の図５９のテーブルでは１０であり、従って、設計者は高速クラスタのためバス形式４または更に良好なものを選択することができる。

バス階層の生成
設計者が一旦バスとそれらの負荷を確認すると、いずれが高速バスであるか、いずれが低速バスであるか、及びどのブリッジ及びアービタが必要とされているかを判定するステップを含むバスパ性能階層が指定されなければならない。２つのバスが低減バスマトリクス（セルから／セルへのそれらはゼロでない値を有する）で接続される場合、それらの間でブリッジが生成される。図５４の例を使用して、ピボット運動したデータマトリクス及び低減バスマトリクスからの次のバスモデルが生成される：
下記に接続された６４ビットのシステムバス（形式４または５）：
Ｂｌｏｃｋ １（ＲＮＶ）
Ｂｌｏｃｋ ２ （ＲＮＶ）
Ｍｅｍｏｒｙ （ＲＮＶ）
ＰＣＩ （ＲＮＶ）。
【誤訳訂正１８】
【訂正対象書類名】明細書
【訂正対象項目名】０１９９
【訂正方法】変更
【訂正の内容】
【０１９９】
下記に接続された８ビットの周辺機器バス（形式３または更に良好なもの）へのブリッジ（ＲＮＶ）：
Ｂｌｏｃｋ ３ （読込み／書込み） ，
ＰＩＯ （書込みのみ）。
注記：それからの到来データがないため、ＰＩＯは書込みのみである。バス１と２間の両

対角線方向がゼロではないため、ブリッジは読込み／書込みである。

このマップは次のタスクに渡されバス設計が指定される。

バス設計の指定
バス設計を指定するため、設計者は作成されたバスを元のブロック、ブリッジ及びアービタなどのひと組の新しいブロック、及びひと組のグルー論理回路のためのひと組のインタフェース仕様に拡張する。新旧のロック仕様はブロック設計タスクへ渡される。グルー論理回路はミニブロックとしてブロック設計を通ってチップ組立体タスクへ転送される。バスがＯＣＢ属性仕様を満足する場合、それにはアービタ及びブリッジなどの他のバス対象と同様に、マスタスレーブデバイスのためのモデルを有している。バスを選択する定義されたマップを使用して、設計者は詳細バス構造を作成する。

詳細バス構造
詳細バス構造を作成するため、設計者は下記を実行しなければならない：
１．　単一負荷及びブリッジですべてのバスを除去することによってバスを最適化する。システムバスのプロトコルとただひとつの周辺機器のバスとの間の翻訳がゲートの観点から緩慢でかつ高価であるため、負荷をブリッジの反対側に置く必要がある。設計者はブリッジ論理回路を完全に除去できない可能性があるが、バスが二地点間交信に縮小するのでトリステートインターフェースは除去される。さらに、２つの端部を一緒に置くことができるため、８ビットを多くの不利点なしに１６ビットに変えることができる。
２．バス・マスタ（bus master）及びスレーブを様々なロードに割り当てる。設計者は、ブリッジでまる。それは、より遅いサイド（side）でのマスタ及びより速いサイドでのスレーブである。周辺バス上のすべての装置は、スレーブ装置である。システムバス上では、マスタ及びスレーブは、どの装置が、前記バスを制御する必要があるかによって定義される。設計の知識は、この決定を手助けすることができる。プロセッサが、前記バスに接続されている場合、そのインタフェースはマスタである。そうでなければ、明白なマスタがない場合、ＰＣＩ等の外部インタフェースがマスタである。メモリ・インタフェースは、ほぼ常に、スレーブ・インタフェースである。どのブロックがマスタ・インタフェースを要求するかを決定するために、設計者は、前記バスへの相互接続条件を参照しなければならない。
３．プロセッサ又は他のブロックが、メモリ・インタフェースも有するバスに接続されている場合、及び前記ブロックが特定的にそれを必要とする場合、設計者は、前記バス上に、一つ以上の直接メモリ・アクセス（direct memory access）（ＤＭＡ）装置を含むべきである。これらの装置は、バス・マスタとして動作する。
４．最後に、一つのバス上の二つ以上の装置がバス・マスタである場合、アービタ（arbiter）を追加する。

詳細なバス設計
　バス構造が定義されると、ブロック・バス・インタフェースが検査される。ブロックがすでにバス・インタフェースを有する場合、前記インタフェースは、前記バスに合わせるために、ソフト（soft）、ファーム（firm）、又はパラメータ表示された形式でなければならない。この場合には、存在するバス・インターフェース・ロジックが使用されるべきであり、そうでなければ、前記バスを備えたモデルが容認される。異なるバス・インタフェースが前記ブロック上にある場合、それは可能であれば排除されるべきである。
【誤訳訂正１９】
【訂正対象書類名】明細書
【訂正対象項目名】０２１２
【訂正方法】変更
【訂正の内容】

【０２１２】
この例に関して、図６０のテーブルは、例示的なブロックＡ一貫性検査を示す。

結合性
結合性は、ある流れにおけるテスト方法が、互いに関与する程度である。５つの密接な関係を有するテスト方法パラメータがある；各々は、他を変更することができる。例えば、前記テスト・アクセス方法は、テスト・プロトコルの活動化条件を定義し、前記テスト・プロトコルは、テスト・データがどのように順番をつけられるかを定義し、及びテスト・データは、特定のテスタ・タイムセットを有する、一組のパターンに分解される。そして、埋め込みブロックへのテスト・アクセスは、チップＩ／Ｏ制限及びコントローラ設計の影響を受けやすいので、これらのパラメータの結合性は、テスト・データの保全性を維持するために、独自の検査スタイルを必要とする。それゆえに、前記５つのテスト方法パラメータは、テスト・アクセス、テスト・プロトコル、テスト・データ、テスタ・タイムセット、及びテスト時間である。

アーキテクチャ規則
図６１は、ＤＦＴ透視図からの、チップの最高レベルの階層を示す。設計者がＤＦＴ処理を始める前に、前記設計者は、機能的ブロックの集まりとしてよりも、図６１に記載のとおり、前記チップを視覚化するであろう。図６２は、マージ非可能ブロックがソケット化されるＳＡＰ及びＤＡＰを伴う、機能的ブロックで作られた設計を示す。
【誤訳訂正２０】
【訂正対象書類名】明細書
【訂正対象項目名】０２３７
【訂正方法】変更
【訂正の内容】
【０２３７】
テスト生成メカニズムを作成する
テスト生成に関するＢＢＤ戦略は、マニュアル・ベクトル（manual vector）、ＡＴＰＧ、又は混合（mixed）を具備しうる。トランスレーション及び連結メカニズムは、最高レベル・テスト・ロジック及び個別のブロックのテスト・メカニズムに適合するように定義される。ＢＢＤにおいて、テスト開発は、二つの独立した処理を具備する。
１．各仮想ソケットに関するブロック・レベル・テスト開発である。ほとんどの場合、この処理は以下のタスクを含む：
ａ．ＳＡＰ宣言：ＳＡＰを、行動モデル・インタフェースに追加し、及び前記ブロックを、その仮想ソケットで再インスタンス生成をする。
テスト・ロジック挿入：対象となるブロック周辺のテスト・カラーを形成するために、テスト・アクセス、分離、相互接続テスト及びテスト制御ロジックを追加する。最善の結果を得るために、前記テスト・カラーを、合成可能ＲＴＬフォーマットで記述する。
テスト・データ変換：テスト・データをＳＡＰポートに拡張し、及びマッピングする。新しいテスト・データ・フォーマットを受容するために、ブロック・レベル・テスト・ベンチを変更しなければならない。テスト・フローを能率化するためには、１つのソケットにつきテスト・セットアップ時間を最短化し、及び同時に複数のブロック・テストをランさせるために、いくつかのブロック上でテスタ・タイミングを変更してもよい。
テスト検査：テスト・ロジックを検査するために、ブロック・レベル・テスト・ベンチを変更する。対象となるブロックを、前のステップの前後に、テスト・データ保全性を確実にするために、完全なブロック・レベル・テスト・ベクトル・セットのサブセットで検査する。