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

特許第4495865号
(P4495865)

(45)発行日　平成22年7月7日(2010.7.7)　(24)登録日　平成22年4月16日(2010.4.16)

(51)Int.Cl.		F I		
G06F 17/50	(2006.01)	G06F	17/50	652Z
G06Q 30/00	(2006.01)	G06F	17/50	662D
H01L 21/82	(2006.01)	G06F	17/50	601Z
		G06F	17/50	ZEC
		G06F	17/60	302E

請求項の数 30　(全 29 頁)　最終頁に続く

(21)出願番号	特願2000-620508 (P2000-620508)
(86)(22)出願日	平成12年5月26日 (2000.5.26)
(65)公表番号	特表2003-500745 (P2003-500745A)
(43)公表日	平成15年1月7日 (2003.1.7)
(86)国際出願番号	PCT/US2000/014617
(87)国際公開番号	WO2000/072185
(87)国際公開日	平成12年11月30日 (2000.11.30)
審査請求日	平成19年5月15日 (2007.5.15)
(31)優先権主張番号	60/136,127
(32)優先日	平成11年5月26日 (1999.5.26)
(33)優先権主張国	米国(US)
(31)優先権主張番号	60/135,902
(32)優先日	平成11年5月26日 (1999.5.26)
(33)優先権主張国	米国(US)

(73)特許権者　501456401
ゲット２チップ
アメリカ合衆国　カリフォルニア州　９５１３１　サンノゼ　エヌ　ファースト　ストリート　２１０７　スイート　３５０
(74)代理人　100147485
弁理士　杉村　憲司
(74)代理人　100072051
弁理士　杉村　興作
(72)発明者　エロフ　フランク
アメリカ合衆国　カリフォルニア州　９４０８６　サニーヴェイル　アメリカ　アヴェニュー　３７８

最終頁に続く

(54)【発明の名称】業者間アプリケーションサービスプロバイダ

(57)【特許請求の範囲】

【請求項１】

ソフトウェアツール環境が、独自のインテレクチュアルプロパティ（ＩＰ）を創成すべき、システム・オン・チップ設計者にたいし、従量料金制で提供される、業者間アプリケーションサービスプロバイダー装置であって、

少なくとも一人のシステム・オン・チップ設計者にとってアクセス可能であり、かつ、ハードウェア記述言語（ＨＤＬ）で表わされた電子的設計のアップロードを受容可能な、インターネットウェブサイト；

前記インターネットウェブサイトで提供され、フロントエンド電子的設計自動化（ＥＤＡ）ツールを供給し、かつ、前記ＨＤＬを受容するように接続される、ＥＤＡオンデマンド解決手段；および、

シミュレートされ、実証された、前記電子的設計の派生物を、インターネットを通じて、返送ダウンロードすることを条件として、前記システム・オン・チップ設計者に請求書を供給する、申し込みコントローラーを含み、

前記ＥＤＡオンデマンド解決手段のユーザーに、独自のインテレクチュアルプロパティ（ＩＰ）を創成するために、従量制料金を請求する、業者間アプリケーションサービスプロバイダー装置において、

前記ＥＤＡオンデマンド解決手段は、電子的設計自動化（ＥＤＡ）コンピュータプログラムを含み、

該プログラムは、

元の回路設計を、一組の対応ロジックツリーに分割すること；
前記ロジックツリーの各々を、内部ノードを持たない、簡単化等価ツリーと交換すること；
前記元の回路において、各経路を、ツリーのリーフ（葉）からルート（根）まで分析すること；
前記各経路について、伝達遅延を計算すること；および、
計算された遅延を、前記簡単化ツリーの対応リーフ（葉）に注記すること、
のためであることを特徴とする、業者間アプリケーションサービスプロバイダー装置。
【請求項２】
前記電子的設計自動化（ＥＤＡ）コンピュータプログラムは、
電子回路設計を生成すること；
前記電子回路設計をその成分ブロックおよびプロトコール設計に分割すること；
前記成分ブロックとプロトコール設計をハードウェア記述言語（ＨＤＬ）にてコード化すること；
前記成分ブロックとプロトコール設計の操作スケジューリングとリソース割り当てのために高レベル合成（ＨＬＳ）を用いること；
操作スケジューリングとリソース割り当ての後で、前記成分ブロックを最適化して、中間設計を生成する、技術独立性ブールロジック；
前記電子回路設計のハードウェア実行用の特定デバイスを選択する、技術マッピングの前記中間設計；
前記特定デバイスを、半導体チップの部位に位置付けること；および、
前記特定デバイスの一組の相互接続の経路指定を行うこと、のためにあって、
技術マッピング行程は、
元の回路設計を、一組の、対応ロジックツリーに分割すること；
前記一組の対応ロジックツリーを順序付けし、それを、別順のツリーを駆動する各ツリーＴは、その別順ツリーに先行する、かつ、前記ツリーＴを駆動する、各順序ツリーは、前記ツリーＴに先行する、ように順序付けして、順序化直線性リストを構成すること；
前記順序化直線性リストを前方掃引し、一方、複数のネットノードそれぞれについて、技術ライブラリー要素に適合する、一組のパレート最適負荷／到着曲線を計算すること；
前記順序化直線性リストを後方掃引し、一方、前記ネットノードそれぞれの、前記一組のパレート最適負荷／到着曲線と、容量負荷とを用いて、前記技術ライブラリー要素の内から、最短信号到着時間を持つ、最善要素を選択すること；
のサブ行程を含み、かつ、
ゲート入力に対応するネットノードのみが考慮され、かつ、容量負荷は全てあらかじめ指定されることを特徴とする、請求項１の業者間アプリケーションサービスプロバイダー装置。
【請求項３】
前記電子的設計自動化（ＥＤＡ）コンピュータプログラムが、
前記元の回路設計の伝達遅延が、入力信号の変位速度に依存する場合、それがいずれのものであっても、前記簡単化ツリーの対応リーフ（葉）に注記すること、
をさらに供給することを特徴とする、請求項１の業者間アプリケーションサービスプロバイダー装置。
【請求項４】
前記電子的設計自動化（ＥＤＡ）コンピュータプログラムが、
前記ロジックツリーの全てのリーフにおいてそこに容量負荷値があれば、それがいずれのものであっても、前記簡単化ツリーの対応リーフにコピーすること、
をさらに供給することを特徴とする、請求項１の業者間アプリケーションサービスプロバイダー装置。
【請求項５】
前記電子的設計自動化（ＥＤＡ）コンピュータプログラムは、

前記ロジックツリー頂上の出力ゲートの負荷／遅延反応曲線を、前記簡単化ツリーのルートにコピーすること、
をさらに供給することを特徴とする、請求項１の業者間アプリケーションサービスプロバイダー装置。
【請求項６】
前記電子的設計自動化（ＥＤＡ）コンピュータプログラムは、
全遅延計算を取り潰して、リソースの抽象的タイミングモデルの内部にたいする、単純な、辺縁重み付け最長経路横断への変更すること、
をさらに供給することを特徴とする、請求項１の業者間アプリケーションサービスプロバイダー装置。
【請求項７】
前記電子的設計自動化（ＥＤＡ）コンピュータプログラムは、
回路境界に接触する複雑モデルツリーと、内部的で、回路境界に接触しない単純モデルとの併用によって、電子設計のタイミング遅延を計算すること、
をさらに供給することを特徴とする、請求項１の業者間アプリケーションサービスプロバイダー装置。
【請求項８】
前記電子的設計自動化（ＥＤＡ）コンピュータプログラムは、技術選択行程をさらに供給し、同技術選択行程は、
元の回路設計を、一組の対応ロジックツリーに分割すること；
前記一組の対応ロジックツリーを順序付けし、それを、別順のツリーを駆動する各ツリーＴは、その別順ツリーに先行する、かつ、前記ツリーＴを駆動する、各順序ツリーは、前記ツリーＴに先行する、ように順序付けして、順序化直線性リストを構成すること；
前記順序化直線性リストを前方掃引し、一方、複数のネットノードそれぞれについて、技術ライブラリー要素に適合する、一組のパレート最適負荷／到着曲線を計算すること；および、
前記順序化直線性リストを後方掃引し、一方、前記ネットノードそれぞれの、前記一組のパレート最適負荷／到着曲線と、容量負荷とを用いて、前記技術ライブラリー要素の内から、最短信号到着時間を持つ、最善要素を選択すること；
のステップを含み、かつ、ここに、
ゲート入力に対応するネットノードのみが考慮され、かつ、容量負荷は全てあらかじめ指定される、
ことを特徴とする、請求項１の業者間アプリケーションサービスプロバイダー装置。
【請求項９】
前記電子的設計自動化（ＥＤＡ）コンピュータプログラムは、
電子回路設計を生成すること；
前記電子回路設計をその成分ブロックおよびプロトコール設計に分割すること；
前記成分ブロックとプロトコール設計をハードウェア記述言語（ＨＤＬ）にてコード化すること；
前記成分ブロックとプロトコール設計の操作スケジューリングとリソース割り当てのために高レベル合成（ＨＬＳ）を用いること；
操作スケジューリングとリソース割り当ての後で、前記成分ブロックを最適化して、中間設計を生成する、技術独立性ブールロジック；
前記電子回路設計のハードウェア実行用の特定デバイスを選択する、技術マッピングの前記中間設計；
前記特定デバイスを、半導体チップの部位に位置付けること；および、
前記特定デバイスの一組の相互接続の経路指定を行うこと、
をさらに供給し、ここに、
技術マッピング行程は、
元の回路設計を、一組の、対応ロジックツリーに分割すること；

前記一組の対応ロジックツリーを順序付けし、それを、別順のツリーを駆動する各ツリーＴは、その別順ツリーに先行する、かつ、前記ツリーＴを駆動する、各順序ツリーは、前記ツリーＴに先行する、ように順序付けして、順序化直線性リストを構成すること；

前記順序化直線性リストを前方掃引し、一方、複数のネットノードそれぞれについて、技術ライブラリー要素に適合する、一組のパレート最適負荷／到着曲線を計算すること；

前記順序化直線性リストを後方掃引し、一方、前記ネットノードそれぞれの、前記一組のパレート最適負荷／到着曲線と、容量負荷とを用いて、前記技術ライブラリー要素の内から、最短信号到着時間を持つ、最善要素を選択すること；

のサブ行程を含み、かつ、

ゲート入力に対応するネットノードのみが考慮され、かつ、容量負荷は全てあらかじめ指定されることを特徴とする、請求項１の業者間アプリケーションサービスプロバイダー装置。

【請求項１０】

前記高レベル合成を用いるステップは、タイミング分析が、個別の操作がスケジュールされる度毎に適用され、かつ、単一の操作がスケジュールされるために多数回呼び出されてもよいように行われる、

ことを特徴とする、請求項９の業者間アプリケーションサービスプロバイダー装置。

【請求項１１】

前記技術マッピングステップは、前記電子回路設計のブールロジックゲートを、技術ライブラリー由来の標準セルにマップする、

ことを特徴とする、請求項９の業者間アプリケーションサービスプロバイダー装置。

【請求項１２】

前記電子的設計自動化（ＥＤＡ）コンピュータプログラムは、

時系列プログラムを表わすハードウェア記述言語テキストを、その後の操作スケジューリングと技術割り当てに備えて、制御フローグラフに変換することをさらに供給し、同変換は、

時系列プログラムを表わすハードウェア記述言語テキストを減少させて制御フローグラフに変換する減少ステップと；

前記制御フローグラフからワン・ホット・ビット有限状態装置を構築すること；および、

電子的設計自動化システムにおける操作スケジューリングの前に、前記ワン・ホット・バイト有限状態装置の操作タイミングを予測すること、

によって実行され、ここに、

前記ハードウェア記述言語テキストと、最終合成設計の間に、サイクル毎のタイミング一致が維持される、

ことを特徴とする、請求項１の業者間アプリケーションサービスプロバイダー装置。

【請求項１３】

前記減少ステップは、解析ツリーのステップ毎減少を含み、ここに、特定の解析ツリー構造が認識され、かつ、対応サブグラフが構築されることを特徴とする、請求項１２の業者間アプリケーションサービスプロバイダー装置。

【請求項１４】

前記減少ステップは、１個の僅少自己帰還ループを含む、１個のリセットノードと、１個のジョインノードの構築で始まることを特徴とする、請求項１３の業者間アプリケーションサービスプロバイダー装置。

【請求項１５】

前記減少ステップは、アークに注記された全ての解析ツリー宣言文について操作を適用することによって、前記単純グラフを、さらに洗練された制御フローグラフに変換することによって続行し、ここに、

１個の宣言文を持つアークは除去され、少なくとも２個の新規アークと、少なくとも１個の新規ノードによって交換されることを特徴とする、請求項１４の業者間アプリケーシ

ョンサービスプロバイダー装置。
【請求項１６】
前記減少ステップは、前記全てのアークに注記された前記全ての解析ツリー宣言文にたいして、前記操作を再帰的に適用することによって続行し、ここに、
全ての分解可能な宣言文が処理されることを特徴とする、請求項１５の業者間アプリケーションサービスプロバイダー装置。
【請求項１７】
前記減少ステップは、全ての表示ブロックの名前を、そのような名前を"ｅｎｄ"ノードにマップするテーブルに保存することによって続行し、ここに、
前記"ｅｎｄ"ノードは、全てのＶｅｒｉｌｏｇ"ｄｉｓａｂｌｅ"宣言文にたいし終着点を供給することを特徴とする、請求項１６の業者間アプリケーションサービスプロバイダー装置。
【請求項１８】
前記減少ステップは、"ｒｅｐｅａｔ"ループの場合、繰り返しカウンター変数を導入することによって続行し、ここに、
前記繰り返しカウンター変数は、前記ループに進入する前に初期化され、かつ、前記ループが一周する度毎に繰り上げられることを特徴とする、請求項１６の業者間アプリケーションサービスプロバイダー装置。
【請求項１９】
前記減少ステップは、"ｒｅｐｅａｔ"ループ、"ｗｈｉｌｅ"ループ、または、"ｆｏｒ"ループにたいする、新規"ｉｔｅｒ"ノードの２個のアウト-アークのそれぞれに条件を付着させることによって続行することを特徴とする、請求項１６の業者間アプリケーションサービスプロバイダー装置。
【請求項２０】
前記減少ステップは、"ｒｅｓｅｔ"ノードからの前方横断によって到達不能な全てのアークとノードを除去することによって続行することを特徴とする、請求項１６の業者間アプリケーションサービスプロバイダー装置。
【請求項２１】
前記減少ステップは、さらにそれ以上グラフ構造を含まない分枝から派生する全てのアーク組をまとめて取り潰し、かつ、全ての条件解析ツリーを、前記制御フローグラフに再注記することによって続行することを特徴とする、請求項１６の業者間アプリケーションサービスプロバイダー装置。
【請求項２２】
前記減少ステップは、いずれの条件文も、余分分枝の創設以外に、前記制御フローグラフにたいして何の作用も及ぼすことはないかどうかを検出し、かつ、もしそうなら、減少を適用せず、前記条件文をそのまま注記することによって続行することを特徴とする、請求項１７の業者間アプリケーションサービスプロバイダー装置。
【請求項２３】
前記減少ステップは、その条件が真となることはあり得ない全ての死んだ分枝を除去することによって続行することを特徴とする、請求項１６の業者間アプリケーションサービスプロバイダー装置。
【請求項２４】
前記減少ステップは、状態としてマークされていない、１個のイン-アークと１個のアウト-アークを含む単純ノードのイン-アーク同士、および、アウト-アーク同士を合流させることによって続行することを特徴とする、請求項１６の業者間アプリケーションサービスプロバイダー装置。
【請求項２５】
前記減少ステップ後に、前記制御フローグラフの剪定を行って、前記ハードウェア記述言語テキストの中にＶｅｒｉｌｏｇ"ｄｉｓａｂｌｅ"宣言文を受け入れることをさらに含む、請求項１６の業者間アプリケーションサービスプロバイダー装置。

【請求項２６】
前記減少ステップ後に、前記制御フローグラフの剪定を行って、前記ハードウェア記述言語テキストの中に"ｇｏｔｏ"宣言文を受け入れることをさらに含む、請求項１６の業者間アプリケーションサービスプロバイダー装置。
【請求項２７】
前記ワン・ホット・ビット有限状態装置を構築するステップは、前記制御フローグラフの各状態ノードを、対応する単一状態フリップフロップにマップすることを含むことを特徴とする、請求項１２の業者間アプリケーションサービスプロバイダー装置。
【請求項２８】
前記ワン・ホット・ビット有限状態装置を構築するステップは、テーブルＭＡＰを構築することによって続行し、かつ、
前記制御フローグラフの全てのアークが、前記有限状態装置の対応出力ポートにマップされる、
ことを特徴とする、請求項１５の業者間アプリケーションサービスプロバイダー装置。
【請求項２９】
前記ワン・ホット・ビット有限状態装置を構築するステップは、一組の一次入力と状態フリップフロップによって駆動され、かつ、ＭＡＰ（Ｃ）を駆動する、回路を建設することによって続行することを特徴とする、請求項２８の業者間アプリケーションサービスプロバイダー装置。
【請求項３０】
前記ワン・ホット・ビット有限状態装置を構築するステップは、近似操作を用いることを含むことを特徴とする、請求項２９の業者間アプリケーションサービスプロバイダー装置。
【発明の詳細な説明】
【０００１】
（発明の分野）
本発明は、電子設計自動化に関し、さらに詳細には、インターネット上で従量料金制に基づいて供給される行動準拠合成ツールに関する。
【０００２】
（発明の背景）
近年シリコンチップの使用は飛躍的に増加している。数百万ゲートの特定用途向け集積回路（ＡＳＩＣ）が、あらゆるタイプのアプリケーションに、例えば、携帯電話、ネットワーク装置、ＤＶＤプレーヤー等に現在使われている。しかしながら、このような複雑な装置の多くの設計者は、チップサイズが５０Ｋから１００Ｋゲートの範囲にあった、１９８０年代半ば以来の、時代遅れの電子設計自動化（ＥＤＡ）ツールや技術を用いている。現在、設計チームは、互いに協力するために、多数の点状ツールを手に入れようと不規則な時間を費やしている。これらの問題は、人力の不足や短い製品サイクルと相俟って、設計要求と、従来技術の能力との間のギャップを広げつつある。
【０００３】
このような無効なツールの使用は、市場においては設計時間がさらに長くなり、製品寿命がどんどん短くなることを意味するから、やがて、非常な供給不足を招く。
【０００４】
従来のＥＤＡツールが現在、システム-オン-チップ（ＳｏＣ）設計への移行を妨げている。既存のツールよりも確かに早く、かつ、大きな容量を持つ、新しい、構造的、次世代ロジック合成技術が求められている。設計者は、より高レベルの抽象化へと移動し、百万ゲートＲＴＬや構造レベル設計を取り扱う際の、複雑性・実証性問題を克服しなければならない。
【０００５】
ＥＤＡツールに関する、従来の販売・流通にも問題がある。高度のＥＤＡツールを、設計者が必要とする時に、設計者の前に置き、かつ、そのＥＤＡツールが実際に使用された

時にのみ、その顧客に料金を請求するには、新しいやり方が必要である。現在、インターネットは、ＥＤＡツールを販売・流通させるための新しい方法を提供する。

【０００６】

全てのフロントエンド設計相、例えば、構造、ＲＴＬ、データパス、ロジックの諸相にたいしては、一つのＥＤＡツールを提供する必要がある。構造的合成が実現されるならば、真のシステムレベルの設計が、１０倍も大きい容量で実現可能となろう。運転時間が速くなれば、より優れた品質信頼性（ＱｏＲ）が得られるだろう。なぜなら、普遍的最適化が可能とならからである。コードと設計時間において、２対５の時間短縮をもたらすために新しいＥＤＡツールが求められている。

【０００７】

（発明の概要）

簡単に言うと、本発明の、業者間アプリケーションサービスプロバイダ実施態様は、システム-オン-チップ設計者にたいして、ＥＤＡ-オン-デマンド解決法を持つインターネットウェブサイトおよびウェブサーバーを含む。このウェブサイトは、ハードウェア記述言語で表わされた電子設計が、フロントエンドＥＤＡ設計環境にアップロードされることを可能にする。ウェブサーバーにおいて私的に宿される、行動モデルシミュレーションツールが、その設計を試験し、その有効性を保証する。このツールは、業者間アプリケーションサービスプロバイダの安全な環境においてのみ実行する。次に、この有効保証された設計解決策はさらに、その他の設計の交換として、他者にダウンロード可能であり、また、技術ライブラリーにおいて入手可能となる。このようにして創成されたインテレクチュアルプロパティ（ＩＰ）は、利潤追求の中心同盟センターから、効率的にかつ簡単に、再使用され、販売され、共有され、交換され、かつ、その他のやり方で配給されることが可能である。

【０００８】

（詳細な説明）

図１は、本発明のインターネットシステム実施態様を表わすが、本明細書では、一般参照番号１００と呼ぶことにする。このシステム１００は、高レベル合成（ＨＬＳ）サービスとインテレクチュアルプロパティ（ＩＰ）を販売する、業者間アプリケーションサービスプロバイダ１０４のインターネット接続１０２を含む。このアプリケーションサービスプロバイダ１０４は、クライアントのウェブブラザー訪問を迎えるために、例えば、Ｍｉｃｒｏｓｏｆｔ社から販売されるソフトウェア、ＷＩＮＤＯＷＳ－ＮＴ、ＩＩＳおよびＡＳＰを搭載するウェブサーバー１０６を有する。ペイパーユース即ち従量料金制電子設計自動化（ＥＤＡ）ツール１０８が、ソフトウェアアプリケーションとして、ウェブサーバー１０６に搭載されている。いずれの数であってもよい複数のユーザーおよび顧客が、ウェブクライアント１１０およびブラウザー１１２で表わされる。ハードウェア記述言語（ＨＤＬ）で記述された構造電子設計は、ＥＤＡツール１０８によるシミュレーションと設計証明を求めて、インターネット１０２を通じてアップロードされる。一旦完了したならば、この証明結果は、場所バックエンド工程１１４に、また、ルートバックエンド工程１１６にダウンロードして戻される。

【０００９】

ＥＤＡツール１０８は、ユーザーに従量制料金を請求し、かつ、設計のアップロード・ダウンロードを許可する、入会モジュール１１８によってサポートされる。ＨＤＬ変換モジュール１２０は、ＨＤＬを翻訳して、制御フロー（ＣＦ）グラフに変える。操作スケジュールモジュール１２２は、各ＨＤＬステートメントを、適当なＣＦグラフノードにマップする。リソース割り当てモジュール１２４は、スケジュールで要求されるハードウェアを最適化する。一固まりのユーザーツール１２６が、ユーザーが、このウェブサイトをナビゲートし、理解し、かつ、使用するのを助けるために含まれている。

【００１０】

この業者間アプリケーションサービスプロバイダ１０４は、好ましくは、高レベル合成のスケジュール相において、ディジタル設計の浮動時間分析を含む。抽象時間モデルは、

従来のフルタイミング分析に見られる複雑性という罰をもたらすことなく、成分のビットレベル・タイミングを表出する。各スケジュール決定のタイミング結果について、速やかで、正確な見積もりが供給される。次に、この見積もりを用いて、どれかのスケジュール決定を拒否すべきかどうかを決めることが可能である。

【００１１】

高レベル合成（ＨＬＳ）は、電子設計自動化（ＥＤＡ）システムにおけるディジタルシステム設計の幾つかのサブタスクを自動化する。システム構築は、導入されるべき全体アルゴリズムを、例えば、Ｃ、Ｃ＋＋、特殊言語、または、捕捉言語を使って、設計し、有効保証することから始まる。得られた構築仕様を、ボード、チップ、および、ブロックに区分する。各ブロックは、それ自身の制御フローを有する単一工程である。近代の、大規模チップ設計では、通常、数十から数百のこのようなブロックがある。典型的なブロックは、全体フィルター、待ち行列、および、パイプラインのステージを表わす。一旦チップが、その構成ブロックに区分されたならば、必要とされる、何らかの交信プロトコルを構築しなければならない。このようなプロトコルは、ブロック間のサイクル毎交信を用いる。

【００１２】

いわゆる「スケジューリング」と「割り当て」は、１時に１ブロックに適用される。スケジューリング工程１２２は、加算や乗算のような演算を、有限状態装置（ＦＳＭ）の状態（ステータス）に割り当てる。このＦＳＭは、合成されるブロックによって実行される制御フローをアルゴリズムに記述する。幾つかの演算は、特定の状態でロックされて、他のブロックとの交信を表わす。これらの入出力操作は、ある状態から別の状態へ動かしたり、スケジュール変更したりすることはできない。なぜなら、そんなことをすれば、ブロック対ブロックの交信プロトコルを混乱させることになるからである。

【００１３】

しかしながら、他の幾つかの演算は、ある状態から別の状態への移動が可能である。多数の演算を有する状態から、ほとんど演算を持たない状態へ操作を移動させることは、ハードウェアリソース（資源）が、操作間でより均一に共有されることを可能とする。タイミング問題は、時として、操作を、操作遅延がタイミング問題を引き起こしている状態から、そのような問題の存在しない状態へ移動することによって解決が可能なことがある。

【００１４】

割り当て工程１２４は、あるスケジュールされたＦＳＭの操作を、特定のハードウェアリソースにマップする。例えば、３回の加算操作を、単一の加算器のみを必要とするようにスケジュールすることが可能である。適当な加算器を構築し、演算をその加算器に割り当てる。しかしながら、あるビット幅と機能を持つ１個を越えるハードウェアリソースが必要とされる場合、厄介な問題が発生し得る。従って、各演算について、どのリソースを使用すべきかを決めなければならない。考慮しなければならないこととして、多重化コスト、偽似タイミングパスの創成、レジスタ割り当て、さらには、小規模演算用として大きなリソースの使用等が挙げられる。ハードウェアリソースは、多数機能のために使用することが可能である。ある全体工程のために必要な、最小の一組のリソースを計算することは困難ではあるが、困難の価値ある有意味な作業である。時として、別様導入が可能な場合がある。全体タイミング制限に合致し、ゲートカウントを極小にする構築を選ぶことが可能な場合がしばしばある。リソース割り当ては、リソース（抽象機能）を、ゲートレベルの設置成分にマップすることを含む。

【００１５】

割り当ては、レジスタセットを計算し、データを、後の状態での使用のためにレジスタに割り当てることを含む。例えば、一時的変数を使用して、さらに大規模な計算の中間結果を保存する。しかしながら、そのような一時的変数の内容は、異なる状態において、共通レジスタを共有することも可能である。この内容は、１状態について一つだけ必要とされる。従って、保存する必要のあるデータを、そのような保存要素に割り当てることによって、ハードウェア上にセーブすることが可能になる。しかしながら、データ値が、互い

に排除し合うセットを形成したり、または、保存を共有することが可能な場合は、レジスタおよび保存割り当てが複雑になる可能性がある。データ値はしばしば機能的リソースを駆動するが、逆に、機能的リソースによって生産されることもしばしばある。データを保存に上手に割り当てることは、多重化コストと遅延の縮小をもたらす。割り当てはまた、レジスタと機能的ハードウェアが相互作用を持つ場合、より複雑化される。

【００１６】

スケジューリングと割り当ての後には、技術から独立した、または、ブール的最適化が続く。回路設計は、ネットリストに接続された、一般的ＡＮＤとＯＲゲートを含む。技術独立的最適化は、ネットリストの中のリテラルの数を最小化する。ブールゲートの抽象化そのものが、ブール代数に基づく高度に数学的な処理となる。例えば、ブールの等式ＡＢ＋ＡＣ＝Ａ（Ｂ＋Ｃ）を用いて、対応するゲートネットワークを減少させることが可能である。

【００１７】

ブール最適化の後には技術マッピングが続く。回路の抽象ブールゲートは、技術ライブラリーから得た標準セルにマップされる。標準ライブラリーセルは、単純なＡＮＤ、ＯＲまたはＮＯＴ機能や、それより遥かに複雑な機能を含む。例えば、全加算器、ａｎｄ－ｏｒ反転ゲート、および、多重化器である。様々な駆動強度、遅延、インプット負荷等を持つ、技術-ライブラリーゲートが入手可能である。一つの個別ブールゲートをマップするのに多数のやり方があり、かつ、各々のやり方が、それ独自の利点を持っているという事実から、技術マッピングはさらに複雑化される。

【００１８】

時として、技術マッピングを回避し、あらかじめ構築され、特性付与されたセルのライブラリーからセルを選択する代わりに、回路のゲートのために、特注ゲートレイアウトを構築することが可能である。しかしながら、この方法は、自動化合成とは普通は関連しない。

【００１９】

技術マッピングの次には、セル設置とネット配線というレイアウト作業が続く。チップ上における各セルの物理的位置が定められ（設置）、かつ、セル同士を相互接続するのに必要なネットがレイアウトされる（経路配置）。アプリケーションサービスプロバイダ１０４においては、設置や経路配置に関して、設計インテレクチュアルプロパティ（ＩＰ）がユーザーにダウンロードされる。

【００２０】

図２は、本発明の電子設計自動化法（ＥＤＡ）の実施態様を表わすが、本明細書では、これを符号２００で示す。ＥＤＡ法は、アルゴリズム設計工程２０２から始まる。このシステム設計は、工程２０４において、複数のブロックとプロトコル設計に区分される。工程２０６において、Ｖｅｒｉｌｏｇ、または、その他のハードウェア記述言語（ＨＤＬ）によるコーディングを実行する。高レベル合成（ＨＬＳ）工程２０８は、演算スケジューリング工程２１０とリソース割り当て工程２１２とを含む。個々の演算がスケジュールされる度毎に、タイミング分析が適用されるので、単一の演算をスケジュールさせるのに何回も呼び出されることがある。技術-独立（ブール的）最適化工程２１４が次に続く。技術マッピング工程２１６は、回路の抽象ブールゲートを、例えば、技術ライブラリーから得た標準セルにマップする。設置工程２１８は、ゲートを、チップ不動産上に配置し、経路配置工程２２０は、ゲート同士を配線で相互接続する。

【００２１】

タイミング分析は、状態ダイアグラム、一揃いのリソース、技術ライブラリー、および、少なくとも部分的にスケジュール化され、割り当てられた演算をもって開始する。分析は、設計全体が、そのタイミング要求に合致しているかどうかを判断する。この回路の遅延合計は、有効データが、各クロックサイクルの終了時において、行き先レジスタへのラッチが可能なようになっていなければならない。スケジューリングシステムを用いて、現実スケジュールを構築し、それによって、レイアウト後タイミング合致の確率を良好にす

る割り当て回路を得ることも可能である。

【００２２】

タイミング疑問には、素早く、かつ、効率的に答えなければならない。なぜなら、そのような疑問は、単一設計をスケジュールする工程で度々尋ねられるからである。例えば、リスト-スケジューリングアルゴリスムでは、設計の転移（即ち変更）は個別に、順番に考慮される。各転移について、一組の「スタンバイ」演算が構築される。このスタンバイ演算は、転移のソース状態において利用可能な入力データを含む演算である。これらの演算の内から一つが、何らかの基準―例えば、もっとも緊急なものをもっとも先に―を用いて選択され、スタンバイリストから取り出され、現在の転移ではこうしたことがなければ未使用とされるリソースに割り当てられる。次に、得られたデータは、利用可能なデータセットに加算される。次に、その結果に依存する他の演算が、スタンバイリストに加算されてもよい。この工程は、もうこれ以上の演算がスタンバイリストに無くなるまで、または、スタンバイリストの演算を実行するためのリソースがもはや無くなるまで、または、スタンバイリストの演算が、現在の転移ではスケジューリングできなくなるまで、続く。これは、全てのアーク即ち円弧が考慮され、全ての演算がスケジュールされるまで、繰り返される。このタイミング工程は、与えられた一組のリソースを使っては、その設計がスケジュールできない場合は、停止する。

【００２３】

タイミング分析は、個別の演算がスケジュールされる度毎に、実行しなければならない。最初の、候補の、演算-転移-リソーススケジューリング組が受け入れられなかった場合、タイミング分析工程は、一つが受け入れられるまで、繰り返し呼び出されなければならない。タイミング分析工程は、一つの演算のスケジューリング組が考慮される度毎に、設計リソースの全てについてタイミングを評価することができなくてはならない。

【００２４】

さらに、タイミング分析は、ビット塊状ではなく、ビット忠実でなくてはならない。各種リソースの遅延を明らかにする場合、各リソースに関連する、単一数、または、単一負荷／機能では不充分である。一つの遅延または負荷／遅延機能は、リソースの各アウトプットビットと関連していなければならない。

【００２５】

併合リソースについて、高速で、正確なビットレベルのタイミングモデルを構築することは可能である。なぜなら、併合ロジックを表わすグラフは、必ず、一揃いのツリー（ツリー）に区分けすることが可能だからである。ロジックのツリーは、そのノードを使って、ゲートと端子を表わし、その円弧を使って、接点を表わす。電気的ドライバーは、ロジックツリーにおいて、必ず、ドライブするゲートの下に来る、例えば、ルート（ツリーの根）から遠い方に来る、あるいは、そうでなければ、ルートと、ドライバーが駆動するゲートは別のツリーに存在する。ツリーの終末点は、ロジックツリーの外側のゲートにたいする接点を表わす。ネットワークのゲートが、２個以上の扇形広がりを持つ場合、最大ロジックツリーのルートは、そのゲートの出力端子である。他の全てのツリーは、潅ツリー（サブツリー）となる。

【００２６】

図３は、本発明のタイミング分析法を表わし、本明細書では、一般参照番号３００と呼ばれる。この方法３００は、回路設計を、対応するロジックツリー（ツリー）に区分する工程３０２から始まる。一旦回路がロジックツリーに区分されたならば、工程３０４において、その回路の小型モデルを構築することが可能になる。ロジックツリーは、例えば、工程３０６において、内部的ノードを持たない等価的ツリーに交換される。この等価ツリーは、しばしば、もとのツリーよりも実質的により単純になっている。工程３０８において、もとの回路のタイミングが、ツリーの葉からそのルートまで、各経路にそって分析される。工程３１２において、もとの回路の伝達遅延が、入力信号の変位速度に依存していることが認められた場合、それがいずれものであっても、簡単化ツリーの対応葉の上に注記される。工程３１４は、ロジックツリーの葉から、容量性負荷を簡単化ツリーの葉にコ

ピーする。例えば、ロジックツリーの頂上における出力ゲートの、ロード／遅延反応曲線は、工程３１６において、簡単化ツリーのルートにコピーされる。

【００２７】

変位や負荷依存性のような、幾つかの辺縁効果は、本発明の実施態様では、経路遅延の中に取り込まれてもよい。これらの追加項は、末端が、全体リソースの境界にあるツリーにたいしてのみ計算する必要があるだけである。従って、工程３１８において、全体遅延計算は、リソースの抽象タイミングモデルの内部における、単純な、辺縁重み付け最長横断経路に縮小される。これによって、回路の各セルについて変位速度や遅延を計算するのに比べて、タイミング分析がずっと速くなる。

【００２８】

図４Ａ、４Ｂおよび４Ｃは、回路４００からロジックツリー４１０、および、簡単化ツリー４２０への転移を表わす。図４Ｂにおいて、ロジックゲートは、ツリー中のノードとして表わされる。ＡＮＤゲートは、上を向いたシェブロン記号を持ち、ＮＯＲゲートは、下を向いたシェブロン記号を持つ。図４Ｃでは、ノードは完全に取り除かれる。本発明の実施態様におけるその後の工程では、簡単化ツリー４２０は、回路４００（図４Ａ）における遅延問題について速やかな解答を与える注釈で飾られる。点線による囲みは、サブツリーを示す。

【００２９】

図５は、インプット境界に一組の複雑モデルアーク５０２、内部に一組の簡単化モデルアーク５０４、および、アウトプット境界に、別の一組の複雑モデルアーク５０６を含む設計５００を表わす。モデル内部の伝達遅延について、このような簡単化モデルを用いているために、正確性において若干の損失があるが、実際には、そのような不正確は本質的ではないようである。正確性の損失が問題なら、新たな内部アークにたいしてさらに複雑なモデルを使用することも可能である。この場合、正確度は向上するであろうが、解法運転時間も増大するであろう。回路境界に接触するアークのみが、真に、複雑モデルを必要とする。簡単化モデルを有するアークを破線矢印で示す。

【００３０】

回路を表わすロジックツリーの全てが、簡単化ツリーにマップされたなら、もとのリソース回路図は廃棄してもよい。関係タイミング情報の全ては、簡単化ツリーのネットワークに保存される。次に、簡単化ツリーのネットワークに取り込まれたリソースのモデルを、到着時間を計算する、グラフ横断アルゴリスムと一緒に用いることによって、連鎖リソースのタイミングを分析することが可能となる。この簡単化モデルタイミング分析用グラフ横断によって、従来の方法に比べて一桁違う高速分析が可能となる。

【００３１】

図６は、二つのグラフ、パターングラフ６０２　Ｇ１＝（ｖ１，ｅ１）と標的グラフ６０４　Ｇ２＝（Ｖ２，ｅ２）がある場合の、一般的グラフ適合問題６００に関わる。目的は、パターングラフ要素から、標的グラフ要素にたいする、１対１マッピングを見つけることである。適合されたノードによって定義されるサブグラフＧ２は、Ｇ１と同形でなければならない。そのような適合の例がここに示され、適合されたノードとアークが、囲み６０８の中に示される。

【００３２】

図７は、回路７０２と、それに対応する、二つの部分から成るグラフ表示７０４を示す。大体、ゲートはノードに変換され、相互接続はアークに変換される。特定の、技術マッピング適合問題は、比較的一般的な適合問題とは異なる。回路のために抽出されたグラフは、方向性を持つ、２部分グラフ、例えば、Ｇ＝（ｖ１，ｖ２，ｅ）である。ここに、"ｅ"の辺縁は、ｖ１の要素をｖ２の要素に接続するが、ｖ１の要素をｖ１に、または、ｖ２の要素をｖ２に接続することは決してない。辺縁は、順序を持つペアで、例えば、方向性を持つ。ｖ２のノードはゲートを表わし、ｖ１のノードは回路ネットを表わす。２部分方向性グラフにおけるグラフ適合は、ネットノードはネットノードにのみマップされ、ゲートノードはゲートノードにのみマップされ、かつ、同形性を用いて辺縁の方向性を保存

するように、行われる。２部分グラフ表示７０４において、ゲートを表わすノードは、タイプ区分を有する、例えば、ＡＮＤ（∧）、ＯＲ（∨）、およびＮＯＴ（！）である。これらは、同形構築の一部を形成する。Ｇ１におけるタイプＸのノードは、Ｇ２におけるタイプＸのノードにマップされる。

【００３３】

従来技術は通常ツリーを適合させる。方向性非サイクルグラフ（ＤＡＧ）を用いて、幾つかの型の複数出力ゲートを表わすことが可能であるから、併合ロジック回路を、一つのＤＡＧで表わすことも可能である。すなわち、それら併合ロジック回路がサイクルを含まない限りは可能である。ＤＡＧはまた、形良く形成され、偽似サイクル経路を含まないようになっていなければならない。

【００３４】

図８は、技術マッピングにおける最初の工程、すなわち、ネットワークグラフ８０２を区分して、一揃いのツリー群８０４－８０８、すなわち、ＤＡＧに変換する工程を示す。次に、各ツリー８０４－３０８を、個別のマッピング問題として、処理する。もっとも簡単な処方はツリーを使用することであるが、ＤＡＧへの拡張を行ったとしても難しくはないだろう。一般に、ツリーは、そのルートや葉は全て、ゲートノードではなくして、ネットノードであるというように定義される。ルートやリーフは、必要なだけ多数重複される。でなければ、ルートやリーフは、ツリー間で共有されなければならない。ツリー８０４－８０８は、できるだけ大きく、例えば、最大ツリーでなければならない。ＤＡＧに拡張する場合、いずれの拡張であっても、一つのＤＡＧのアウトプット端子の数は、２または、その他の小さい数に限定されなければならない。でないと、適合計算が複雑になりすぎる。

【００３５】

典型的な技術ライブラリーは、原始的要素を表わす幾つかのセルを含む。ライブラリーの併合セルは、ブール関数を持つ。各セルについて、選択された一組の２部分方向性グラフ表示が構築される。これらのグラフはそれぞれ、小さな原始的タイプのアルファベットで表現された、セルのブール関数と関連する。ライブラリーの各セルは、ネットノード、２入力ＮＡＮＤノード、および、ＮＯＴノードのみを含むツリー（またはＤＡＧ）によって記述される。選ばれた正確なアルファベットは、それが比較的単純なものであり、ロジック的に完全である限り、決定的に重要ではない。ブール関数は全て、アルファベットのユニットのみを含むネットワークとして表わすことが可能である。

【００３６】

このような分解されたライブラリーを図９に示す。セル名は左のコラムに挙げてある。真中のコラムは、対応するブール関数を挙げる。各セルは、右のコラムに示すような１個以上のパターンツリーで表わされる。

【００３７】

技術マッピング工程に委ねられた回路設計は、いずれのものであっても、通常、単純なゲート、例えば、ＮＡＮＤ、ＡＮＤ、ＮＯＲ、ＯＲ、ＸＯＲ、および、ＮＯＴのネットワークとして表わされる。各ネットワークは、ライブラリーツリーのゲートタイプおよび扇形インのみを用いて、機能的に等価なネットワークに変換される。図９の例示ライブラリーについて言えば、回路は、インバーターと、２入力ＮＡＮＤゲートのみから成る等価回路に変換される。次に、その等価回路は、２部分方向性グラフ表示に、例えば、ライブラリーパターングラフと同じスタイルのグラフ表示にマップされる。その後、グラフ適合を実行することが可能になる。ライブラリーセルおよびマップされる回路の両方とも、同じグラフ形式を用いて表わされる。

【００３８】

回路のツリーは、個別的適合問題となる。好ましくは、適合結果は全て、各ネットノードＮに、適合パターンツリーのリストを付着させることによってコードされる。このリストは、そのルートがＮに適合する、一組のパターンツリーを表わす。下記の偽似コードはそのようなリストを設け、かつ、そのリストを適合（Ｎ）と呼ぶ。

Ｒを回路ツリーＴのルートとせよ。
Ｓを、最初はＲのみから成る一組とせよ。
（Ｓが非空虚である限り）｛
ＮをＳの一要素とせよ。ＳからＮを取り出せ。
（ライブラリーの、パターンツリー組の全ての数Ｐ）について
（適合（ルート（Ｐ）、Ｎ））ならば｛
Ｍを適合（Ｎ）に加えよ。
｝
｝
（ＮがＴのリーフノードでない）ならば｛
ＧをＮのドライバーとせよ。
Ｇの全てのドライバーをＳに加えよ。
｝
｝

【外１】

```
Let R be the root of a circuit tree T.
Let S be a set comprising initially only of R.
While (S is nonempty) {
        Let N be an element of S. Remove N from S.
        For (all members P of the set of pattern trees in the library)
                If (matches(root(P), N)) {
                        Add M to matchings(N).
                }
        }
if (N is not a leaf node of T) {
                Let G be the driver of N.
                Add all drivers of G to S.
        }
}
```

【００３９】
パターン適合アルゴリズムを用いてツリー適合テストを実行するには幾つかの方法がある。例えば、再帰的アルゴリズムは、ある回路ツリーがあるパターンツリーにたいし、ルート同士が適合場合、適合することを認識することが可能である。サブツリーマッピングは、回路ツリーとパターンサブツリーが１対１対応でなければならない。すなわち、回路ツリーの各サブツリーは、正確に、パターンツリーの一つのサブツリーにマップしなければならない。パターンツリーの全てのサブツリーは、回路ツリーの何れかのサブツリーに、例えば、"ｏｎｔｏ"マッピングを通じて、マップされなければならない。これが行われないと、回路ツリーの１を越えるサブツリーが、パターンツリーの単一サブツリーにマップされたり、あるいは、パターンツリーのいずれのサブツリーにもマップされなくなる可能性がある。

【００４０】
あるサブツリーとサブツリーの適合が失敗しても、別の適合が成功するかも知れない。このツリーは非対称であってもよい。パターンツリーのサブツリーの順番リストの、全ての順列が、回路ツリーのサブツリーの順番リストにたいしてテストされる。全ての順列が失敗した場合、適合試行全体が放棄される。いずれかの順列が成功した場合、適合が見出されたことになる。

【００４１】
下記の偽似コードは、適合アルゴリズムを実行する。名前の付けられたノードは全てネットノードである。ゲートノードは、ネットノードのドライバーである。ｉｆ文の中に挙げられているリスト"Ｕ"は、ネットＮのドライバーリスト、例えば、ネットノードＮを駆動するゲートＧのドライバーリストである。

（Ｍがリーフノードであるならば）｛
真を返せ。
｛でなければ（Ｎがリーフノードであるならば）｛
偽を返せ。
｝でなければ（Ｍのドライバーのタイプがドライバーのタイプと同じであるならば）｛
（Ｍのドライバーのドライバーリストの全ての順列Ｐについて）｛
Ｔｅｍｐ＝真；
Ｕを、Ｎのドライバーのドライバーリストとせよ。
（Ｐの全ての要素ｐ、および、Ｕの全ての要素ｕについてこの順番で）｛
（（ｐ、Ｕ）適合が偽であるなら）｛
Ｔｅｍｐ＝偽
Ｃｏｎｔｉｎｕｅ（続行）へ行け。
｝
｝
Ｃｏｎｔｉｎｕｅ（続行）：
Ｉｆ　（Ｔｅｍｐ）｛
（Ｐ、Ｕ）が、成功的１対１の、ｏｎｔｏマッピングであるなら、
真を返せ。
｝
｝
この地点まで達した場合、１対１の、ｏｎｔｏマッピングはない。
偽を返せ。
｝でなければ｛
偽を返せ。
｝

【外２】

```
If(M is a leaf node) {
        Return true.
{ else if (N is a leaf node) {
        Return false.
}else if (the type of the driver of M is the same as the type of the driver of N) {
        For (all permutations P of the list of drivers of the driver of M) {
                Temp = true;
                Let U be the list of drivers of the driver of N.
                For (all elements p of P, and u of U, in order) {
                        If (matches(p, U) is false) {
                                Temp = false;
                                Go to Continue.
                        }
                }
        Continue:
                If (Temp) {
                        (P, U) is a successful one-to-one and onto mapping.
                        Return true.
                }
        }
        If you get to this point there is no one-to-one and onto mapping.
        Return false.
}else{
        Return false.
}
```

【００４２】

この形式では、アルゴリスムは、技術ライブラリーセルの扇形インが増加するにつれて、次第に高価になる。複雑性は指数関数的である。このアルゴリスムはスピードアップが可能であるが、現在の、大抵の技術ライブラリーにたいしては十分に高速である。

【００４３】

この適合工程によって、回路のネットノードから、パターンツリーのルートノードにたいして１対１マッピングが生成される。このようなマッピング候補は、ネットノードを引き数とし、一組のパターンツリーを返す関数である。回路の導入は、一組のネットノードとなるが、そのために、前記候補組の内の一員が選ばれる。この選ばれたネットノードの組は、通常、ネットノードの全体組よりも小さい。なぜなら、幾つかのネットノードは、内部ネットノードを持つパターンの内側に「埋められる」からである。

【００４４】

図１０は、右に回路ツリー１００２を示すが、この回路ツリーの全ての部分と適合するのに必要とされるのは、左の、たった２個のパターンツリー１００４と１００６だけである。１個の、回路ツリーネットノードにたいする、パターンツリーの可能な適合を、破線で示す。従って、全体回路は、６個の適合から成る適当なサブセットが選ばれるならば、僅か４個のゲートで「カバー」することが可能である。言いかえれば、この回路ツリーは、４個の構成ゲートタイプに分解することが可能である。図示の６個の適合から成るサブセットは、冗長なまたは不充分なカバーリングを生成する可能性がある。解決すべき問題は、遅延や冗長を最小にし、かつ、カバーができる適合サブセットを選択することである。

【００４５】

図１１は、本発明のカバリング選択法実施態様を表わし、本明細書では一般参照番号１１００と呼ばれる。工程１１０２で、回路は、ツリーに区分される。工程１１０４で、それらツリーは、立体位置仕分けアルゴリスムによって順序づけられる。深さ順グラフ横断アルゴリスムを用いることも可能である。順序付けの規則にはこうある－ツリー”Ｔ”は、その葉（リーフ）が、ツリー”Ｔ”によって駆動される、全てのツリーに先行し、かつ、ツリー”Ｔ”のいずれかの葉”Ｌ”を駆動する全てのツリーに追随する。このようにして順序付けされたツリーリストは"０"と呼ばれる。

【００４６】

工程１１０６で、順序付け直線リストにおいて前進掃引を実行する一方で、技術ライブラリー要素と適合する複数のネットノードの各々について、一組のパレート至適負荷／到着曲線を計算する。工程１１０８では、順序付け直線リストにおいて後退掃引を実行する一方で、ネットノードの各々と容量性負荷について、一組のパレート至適負荷／到着曲線を用い、技術ライブラリー要素の内、最短の信号到達時間を持つ最善要素を選択する。ゲート入力に対応するネットノードのみを考慮の対象とし、容量性負荷があれば、それらは必ずあらかじめ指定する。

【００４７】

回路と順序リストを表わす、そのようなツリー"Ｔ"を、図１２に示す。"０"とされる第１ツリー１２０１は、任意に"Ａ"と表示した。他のツリー１２０２－１２０９は、"Ｂ"から"Ｊ"にて表示した。前記規則を満足する、ツリーの立体的順序付けは、Ａ、Ｈ、Ｇ、Ｊ、Ｂ、Ｃ、Ｆ、Ｅ、Ｄとなる。ツリーＡ、ＪおよびＨは、インプット境界にあるから、他のツリーは、ツリーＡ、ＪまたはＨのいずれをも駆動できない。従って、前記規則に基づき、ツリーＡを、順序リスト"０"の頭に置くのは許される。ツリーＡ、Ｈ、Ｇ、Ｊ、Ｂ、Ｃ、Ｆ、Ｅ、Ｄのこの順序づけによって、各ツリーが、まだ評価されないインプットツリーを持つツリーがないようなやり方で、順番に、評価されることが可能となる。

【００４８】

インプット境界において、ツリーＡのリーフにたいする信号到着時間は想定可能である。その時間は、回路の併合部分の一次入力を表わすから、信号到着時間は、回路環境やユーザー条件から入手することが可能である。ツリーＡの各リーフＬは、候補適合を持たない。負荷／遅延曲線を、ツリーＡの各リーフに付属させることも可能である。この負荷／

遅延曲線は、Ｌが表わす一次インプットの想定ドライバーに関連する、いずれの負荷／遅延曲線であってもよい。

【００４９】

ツリーＡのルートは、アウトプットネットである。技術ライブラリー由来の少なくとも一つの候補が適合しなければならない。でないと、工程は停止しなければならない。ツリーＡのルートについて、例えば、下記のような再帰工程を含むアルゴリスムを用いて、集合負荷／遅延曲線を計算することが可能である。

候補適合セットＣを持つネットノードＮが与えられたとせよ。
Ｎの集合負荷／遅延を初期化せよ。
（Ｃの各要素ｃについて）
（ｃの各リーフノードＸについて）
ＹをＸに対応するＴのネットノードとせよ。
Ｙにおける集合負荷／遅延曲線を再帰的に計算せよ。
Ｘに関連する負荷を、ライブラリー中に求め、
かつ、Ｘにおける信号到着時間を、
Ｘにおける負荷と、負荷／遅延曲線に基づいて計算せよ。
}
ｃの各リーフにおける到着時間が既知となった。
Ｎにおける到着時間を、負荷の関数として、
かつ、ｃの表わすライブラリー要素の電気的特性を、計算せよ。
計算された曲線を、Ｎの集合不可／到着曲線に加えよ。
}

【外３】

```
Given net node N with candidate matching set C.
Initialize the aggregate load/delay curve of N.
For (each element c of C) {
        For (each leaf node X of c) {
                Let Y be the net node of T corresponding to X.
                Recursively compute the aggregate load/delay curve at Y.
                Look up the load associated with X in the library,
                        and compute the signal arrival time at X,
                        based on the load and the load/delay curve at X.
        }
        The arrival times at each of the leaves of c are now known.
        Compute the arrival time at N as a function of load
                and the electrical characteristics of the library element that c
represents.
        Add the curve so computed to the aggregate load/arrival curve of N.
}               .
```

【００５０】

結局、このアルゴリスムは、Ｎの適合候補を走査し、各候補"ｃ"について、"ｃ"が選ばれるであろうという仮定の下に、"ｃ"のインプットにおける到着時間を計算する。次に、"ｃ"の既知の特性から、Ｎにおける負荷／到着曲線を、従って、"ｃ"のインプットにおける信号到着時間を計算する。Ｎにたいする可能な適合を持つセルのそれぞれを表わすように、負荷／到着曲線の組が生成される。

【００５１】

この集合負荷／遅延曲線表示は、効率的なものとすることが可能である。いずれの負荷にたいしても、ある特定のセル"ｃ"にとって、唯一の至適遅延がある。その負荷にたいしては、"ｃ"の他のいずれの成員も、せいぜい同じ到着時間に適合させることができるだけである。従って、Ｎにおける負荷／到着曲線は、好ましくは、負荷を、"ｃ"の最適成員にマップする関数である。この関数は、集合負荷／遅延曲線の、断片的最小値を取ることに

よって生成が可能である。

【００５２】

ある特定の容量負荷という観点から、最適ゲート選択の実行が必要である。例えば、ゲートＧ２が最善である負荷範囲は、ゼロ負荷から破断点までである。最善は、最短信号到達時間（最小遅延）を有すると定義される。ゲートＧ１が最善な負荷範囲は、破断点から無限までである。容量負荷が与えられているならば、Ｎにおける集合曲線を用いて、到着時間と、Ｎにマップされる最善ゲートの両方を見つけることが可能である。

【００５３】

ツリーの順序リストからスタートした場合、考慮すべき最初のツリーは、一次入力によってのみ駆動されるものである。その後のツリーはいずれも、一次入力によって駆動されるか、または、既に処理されたツリーによって駆動されるかのいずれかである。このようにして、順番に考慮される全てのツリーＴは、一次入力か、前に考慮されたツリーか、そのいずれかによってのみ駆動されることが保証される。ツリーＴのインプットにおける全てのドライバーは、ツリーＴが考慮される時点において既知のパレート最適負荷／到着曲線を持つことになる。パレート最適負荷／到着曲線は、回路の一次出力のそれぞれについて計算される。

【００５４】

最終工程は、各出力－Ｏについて負荷を選択する。最初、その負荷は、それが、出力－Ｏにたいする集合負荷／到着曲線の範囲内に入る限り、任意に選択することが可能である。出力－Ｏにおける集合曲線は、到着時間と最適ゲートＧの両方をマップし、出力－Ｏを駆動する。負荷を設定することは、出力－Ｏを駆動するのに、どのライブラリーゲート－Ｇを選択すべきかを宣告することになる。

【００５５】

そのルートが出力－Ｏであるツリーは、ライブラリーゲート－Ｇから始めて内部を動き回ることが可能である。入力容量は既知である。なぜなら、ライブラリー－Ｇは、技術ライブラリーにおいて既知の要素であるから。これは、その入力の各々を駆動するのに最適なゲートに関する簡単な探索を可能にする。ツリー－Ｔが完全にカバーされるまで、これは繰り返される。

【００５６】

このカバー工程を、ツリーの順序リストにたいして逆順に適用する場合、その駆動するツリーが全て既にカバーされているのでない限り、いかなるツリーも考慮されない。そのツリーのルートゲートへの負荷は考慮される。このカバー工程は、全てのツリーがカバーされるまで進行する。このようにして、技術マッピングが終了する。

【００５７】

要約すると、選択工程は、ツリーの順序リストを掃引する。最初は前方に向かって、次ぎは後方に向かって。前方通過の際、ライブラリー要素に適合する各ネットノードについて、一組のパレート最適負荷／到着曲線が計算される。後方通過では、負荷／到着曲線と負荷値を用いて、考慮される各ノードを駆動するのに最善のゲートを選択する。ゲート入力に対応するノードのみが考慮される。このため、負荷は常に既知であり、従ってこの工程も終了が可能である。

【００５８】

理解と実行を助けることを可能とする、背景ソースが幾つかある。適合ツリーと動的プログラミングを用いる基本概念が、Ｋｅｕｔｚｅｒ，Ｔｅｃｈｎｏｌｏｇｙ Ｂｉｎｄｉｎｇ Ａｎｄ Ｌｏｃａｌ Ｏｐｔｉｍｉｚａｔｉｏｎ Ｂｙ ＤＡＧ Ｍａｔｃｈｉｎｇ，Ｐｒｏｃｅｅｄｉｎｇｓ ｏｆ ｔｈｅ ２４ｔｈ Ｄｅｓｉｇｎ Ａｕｔｏｍａｔｉｏｎ Ｃｏｎｆｅｒｅｎｃｅ，ＩＥＥＥ，１９８７，３４１ページに詳細に記述されている。この論文は、適合ツリーの使用と、選択問題における面積最適化処方をカバーしている。この方法は、面積を最適化するが、負荷／到着曲線は用いない。さらに、これは、２回通過ではなく、前方通過しか使わない。遅延／面積曲線を用いるツリー適合の、また別の洗練された方法がＣｈａｕｄｈａｒｙ，Ａ Ｎｅａｒ Ｏｐｔｉｍａｌ Ｍａｔｃ

hing Algorithm for Technology Mapping Minimizing Area Under Delay Constrains, 29th Design Automation Conference, IEEE, 1994に記載される。しかしながら、この研究で、Ｐｅｄｒａｍは、面積／遅延曲線を用いているので、異なる負荷の作用を計算することができなかった。これは、極めて重要な違いである。

【００５９】

図１３は、どのようにして制御信号が、最重要タイミング経路を支配するかを示す実施例である。制御信号は破線で示され、最重要経路は、太実線で示される。回路１３００は、司令ＦＳＭ１３０２を含む。最重要信号伝達タイミング経路は、一組の入力ポート１３０４と、一組の出力ポート１３０６の間に存在する。回路１３００は、司令ＦＳＭ１３０２に制御される、幾つかの多重化器１３０８－１３１２、幾つかの加算器１３１４－１３１６、および、幾つかのラッチ１３１８－１３２１を含む。状態信号１３２２が、司令ＦＳＭ１３０２によって入力される。多重化器１３０８－１３１２とラッチ１３１８－１３２１は、司令ＦＳＭ１３０２によって出力される一組の制御信号１３２４－１３３３に依存する。従って、入力から出力までの信号の波及状態は、何時制御信号１３２４－１３３３が発せられ、定着するかに、極めて大きく依存する。

【００６０】

制御フローグラフは、ソースコードＨＤＬで表わされる制御フローを記述する、方向性グラフである。一つのＨＤＬ記述から、一意の制御グラフにたいして、直接的マッピングが存在する。そのような一意の制御フローグラフから、例えば、「バブルグラフ」表示で表わされた有限状態装置にたいしても直接的マッピングが存在する。直接マッピングはさらに、一意の制御フローグラフから、「ワン・ホット」ＦＳＭ回路にたいして形成することも可能である。従って、司令ＦＳＭは、その後に構築される、いずれのスケジュールにも無関係に、２工程で生成が可能である。大事なことは、司令ＦＳＭのタイミングは、スケジューリング時に確定することが可能である。

【００６１】

一般に、制御グラフ－Ｇは、方向性アーク”Ｅ”によって接続されるノード”Ｖ”を含む。図１４において、制御グラフ－Ｇの一つのノードＶは、"ｒｅｓｅｔ"と表示される。このようなノードは、制御フローグラフ－Ｇの起点を表わす。定義により、"ｒｅｓｅｔ"ノードは、イン-アークを持たず、ただ一つのアウト-アーク（外に向かうアーク）を持つだけである。制御フローグラフ－Ｇにおける、他のノードについては、いずれの数であっても、状態ノードと表示されてよいし、または、表示無しとしてよい。これらのノードは、少なくとも一つのイン-アーク（内に向かうアーク）と、少なくとも一つのアウト-アークを持たなければならない。

【００６２】

図１５に示すように、アークは、状態および動作による表示が可能である。これらは、解析ツリー、または、解析ツリーのリストであり、ＦＳＭの状態および動作と近似する。状態表示は、例えば、"ｉｆ"で、制御分枝がどちらの道に行くかを明らかにする。動作表示は、制御フローが、動作で表示されるアークにそって移動する場合、生ずる操作を記述する。

【００６３】

"Ｊｏｉｎ　ｎｏｄｅｓ"は、１個を越えるイン-アークを持つノード、例えば、図１５の"ｉｆ"ノードである。"Ｆｏｒｋ　ｎｏｄｅｓ"は、１個を越えるアウト-アークを持つノード、例えば、図１５の"ｆｉ"ノードである。一般のプログラミング構築体に一致する、その他の名前や表示、例えば、"ｂｅｇｉｎ，" "ｌｏｏｐ" "ｅｎｄ"等を用いることも可能である。

【００６４】

制御フローグラフ－Ｇは、典型的には、解析ツリーの段階的減少において分解されたＨＤＬテキストから構築される。特定の解析ツリー構造が認識され、次に、Ｇの対応サブグ

ラフが構築される。本明細書では、Ｖｅｒｉｌｏｇが、各種実例ＨＤＬ・Ｖｅｒｉｌｏｇ構築名に使われていが、マッピングは、ＶＨＤＬや、その他の、辺縁事象を可能とする必須のシミュレーション準拠ＨＤＬにたいして実施することも可能である。

【００６５】

ここで図１４を参照すると、解析ツリーＰ１４００から、制御フローグラフ－Ｇ１４０２への、単一工程の移行は、リセットノード１４０６とジョインノード１４０８、および、小さな自己還元ループ１４１０を持つ単純グラフ１４０４から始まる。Ｖｅｒｉｌｏｇでは、工程は、"ａｌｗａｙｓ"キーボードを用いて創成され、その後、単純または複合"ｓｔａｔｅｍｅｎｔ"が続く。"ｓｔａｔｅｍｅｎｔ"という言葉は、そのノードの動作または操作として、自己ループ１４１０に一時的に注記される。分枝がないところでは、"ｃｏｎｄｉｔｉｏｎ"表示もない。

【００６６】

単純グラフ－Ｇ１４０４は、アークに注記された宣言に工程を適用することによって、さらに洗練された制御フローグラフ－Ｇ１４０２に変換が可能である。例えば、一つの宣言を持つアークは、２個以上のアークと１個以上のノードに交換される。次に、この新しいアークは、さらに簡単な宣言、および／または、状態によって装飾される。新しいノードも同様に表示される。この工程が、分解可能な宣言が残らなくなるまで、再帰的に続く。新しいアークに表示される宣言と状態は、宣言の解析ツリーの、特定サブツリーとなる。

【００６７】

例えば、Ｖｅｒｉｌｏｇにおける連続複文は、Ｂａｃｋｕｓ－Ｎａｕｒ形（ＢＮＦ）語法定義を持つ。

連続文：：＝始め＜宣言文＞終わり

｜｜始め-表示：＜宣言文＞終わり

【外４】

```
sequential-statement ::= begin <statement>* end
                      II begin label: <statement>* end
```

【００６８】

従って、典型的なＶｅｒｉｌｏｇ工程は、図１４の宣言文１４００にあるように"ｓｔａｔｅｍｅｎｔ１"と"ｓｔａｔｅｍｅｎｔ２"で表わされることがある。

【００６９】

図１４において、Ｖｅｒｉｌｏｇの連続ブロック"ｂｅｇｉｎ．．．ｅｎｄ"宣言文は、ソースノード－Ｓとシンクノード－Ｔの間の、制御フローアーク－Ａに変換することが可能である。アーク－Ａは、シンクノード－Ｔから外され、連続ブロック解析ツリー－Ｐは、アーク－Ａ１４１２から取り除かれる。二つの、新しいノード１４１４と１４１６"ｂｅｇｉｎ"と"ｅｎｄ"が構築される。アーク－Ａ１４１２の矢印の先端は、"ｂｅｇｉｎ"ノード１４１４に接続される。新規アーク－Ｂ１４１８が構築され、その矢羽ルート端が、"ｂｅｇｉｎ"ノード１４１４に接続され、かつ、その矢の先端が、"ｅｎｄ"ノード１４１６に接続される。新規アーク－Ｃ１４２０が構築され、その矢羽ルート端が、"ｅｎｄ"ノード１４１６に接続され、その矢の先端が、"ｌｏｏｐ"ノード１４２２に接続される。連続ブロック（例えば、Ｐのサブツリー）の宣言文は全て、解析ツリー１４２４の順序リストとして、アーク－Ｂ１４１８に付着される。

【００７０】

表示されるブロックについては、ブロック名を、その名前を、"ｅｎｄ"ノードにマップするテーブルの中にセーブしなければならない。このノードは、Ｖｅｒｉｌｏｇの"ｄｉｓａｂｌｅ"宣言文の、ジャンプ目的地となることもある。

【００７１】

Ｖｅｒｉｌｏｇ条件文"ｉｆ．．．ｅｌｓｅ"や"ｃａｓｅ．．．ｅｎｄｃａｓｅ"は、ほぼ同じように処理される。Ｖｅｒｉｌｏｇの"ｉｆ．．．ｅｌｓｅ"文は、キーワード"ｉ

ｆ"で始まり、その後に、その条件が真であるなら、実行されるべき発言Ｅや宣言Ｓ１が、さらに、要すればあってもよいキーワード"ｅｌｓｅ"、および、条件が偽であるなら、実行されるべき宣言Ｓ２、が続く。全てのオプションをつけた"ｉｆ"宣言と、その制御フローグラフ減少の例を、図１５に示す。

【００７２】

"ｉｆ"宣言文は、フォークノード１５０６とジョインノード１５０８、および、それらを結ぶアーク１５１０と１５１２、および、結合を、Ｔに接続する新しいアーク１５１４を導入することによって、除去される。条件"ｃｏｎｄ"とそのブール否定"！ｃｏｎｄ"が、条件アーク１５１０と１５１２に注記される。これらアークは、各分枝が通過される条件を指定する。真または偽の分枝のいずれとも関連しない宣言文はないと考えられる。従って、このアークには何も注記されない。真（Ｓ１）の分枝と偽（Ｓ２）の分枝の宣言は、それぞれ、真のアーク１５１０と偽のアーク１５１２に注記される。場合（ｃａｓｅ）宣言も同様に処理される。違いは、条件はデフォールトを含むことがあること、また、分枝の数が、２よりも大きいことがあること、である。デフォールトは、他の全ての条件のロジック合計のロジック否定である。

【００７３】

Ｖｅｒｉｌｏｇにおけるループは、幾つかの形の内のいずれを取ってもよい。それを、対応する制御フローグラフ減少と一緒に、図１６Ａ－１６Ｄに示す。ループを適正に動作させるためには、さらに新たに解析ツリーを構築する必要があるかも知れない。繰り返しループでは、新たな変数、すなわち、繰り返しカウンターが導入される。ループに導入する前に、その値を初期化しなければならない、また、それは、ループが実行する度毎に繰り上げられる。"ｒｅｐｅａｔ．．．ｗｈｉｌｅ"または"ｆｏｒ．．．ｌｏｏｐ"では、条件を、"ｌｔｅｒ"と表示されるノードの、２個のアウト-アークに付着させなければならない。

【００７４】

ｆｏｒｅｖｅｒ（無限）ループへの減少操作は、イン-アークを持たないＴを設けることになる。これは、その制御フローグラフは、永久ループの減少操作後、語法的に不適切となることを意味する。しかしながら、これは、リセットノードから前方横断によって到達不能な、全てのアーク・ノードを除去することによって救済可能な一時的状態である。このような刈り込み救済措置は、制御フローグラフが完全に減少された後に、例えば、ＨＤＬテキストの制御フロー構築体の全てが減少された後に、実行される。

【００７５】

刈り込み措置を減少操作終了時に行わなければならない理由は、Ｖｅｒｉｌｏｇのｄｉｓａｂｌｅ（無効）宣言は、永久ループへの出口を強制的に設けることが可能だからである。このような出口は（Ｖｅｒｉｌｏｇでは）、Ｔのある後続ノードへ行く。Ｖｅｒｉｌｏｇのｄｉｓａｂｌｅは、結局、ｂｅｇｉｎ-ｅｎｄ表示ブロックのエンドノードへの跳躍である。ｄｉｓａｂｌｅが、ノードＳからノードＴへ向かうアークＡにたいして注記されている場合、このｄｉｓａｂｌｅは、Ａの接続をＴから外すことによって、除去される。無効とされた（ｄｉｓａｂｌｅｄ）ブロックに対応するエンドノードＥが、ブロック名を、対応エンドノードにマップするテーブルにおいて求められる。このテーブルは、表示ブロックの減少操作時に構築される。Ａの矢の先端がＥに接続される。"（ｐｏｓｅｄｇｅ　ｃｌｏｃｋ）"という形の事象制御宣言文は、単一ノードを付加し、かつ、それを状態と表示することによって、減少される。制御フローにたいして直接作用を持たない宣言文は除去されず、それぞれのアークに注記される。

【００７６】

高レベル合成に適当な、簡単なＶｅｒｉｌｏｇＨＤＬテキスト１７０２の例を、図１７の左に示す。そのテキストに対応する、完全減少制御フローグラフ１７０４を右に示す。一旦グラフ１７０４が完全に構築されたならば、グラフの意味を変えることなく後続ステップの効率の向上を可能とするために、若干の基本的刈り込みが必要になる。リセットノードから到達不能なノードやアークはいずれのものであっても除去が可能である。分枝か

ら派生し、さらなるグラフ構造を持たないアークの組も、まとめて取り潰しが可能であり、また、条件解析ツリーは、制御フローグラフに再度注記し直すことが可能である。グラフ構造は、ループ、状態、および、無効を含む。別態様として、これらのものが派生する条件文を、余剰分枝の創成以外に、制御フローグラフにたいしては何の作用も及ぼさないものとして、検出することが可能である。そのような場合、減少操作は単純には適用されず、条件文は、そのまま注記される。その条件が決して真になることはあり得ない、死んだ分枝、例えば、フォークノードのアウト-アークは、いずれのものであっても除去が可能である。単純ノードは除去が可能であり、その複数のイン-アークおよびアウト-アークは合流が可能である。例えば、状態とマークされない、１個のイン-アークと１個のアウト-アークを持つノードのようである。

【００７７】

ワン・ホットＦＳＭは、その状態を、単項コードでコードされる。全ての状態は、１ビットを除く全てのビット組を偽とする２進数で表わされる。ワン・ホットコードの例として、０００１、００１０、０１００、１０００が挙げられる。一つを除く全てのビットがゼロであり、単一"ｈｏｔ　ｂｉｔ"（「ホットビット」）は１である。このようなコードは、ワン・ホット性を失うことなく反転が可能である。例えば、１１１０、１１０１、１０１１、０１１１である。

【００７８】

ワン・ホットＦＳＭは、制御フローグラフの各状態ノードは、１対１的に、単一状態フリップフロップにマップすることが可能である、ということに注目して、抽出される。ワン・ホットＦＳＭは、制御フローグラフに見られる状態ノードと同じ数のフリップフロップによって構築される。各フリップフロップ"Ｆ"が、一つの状態ノードに割り当てられる。Ｆの出力が１の場合、対応状態にあると予想する。

【００７９】

制御フローグラフのアークを、ＦＳＭの出力ポートにマップし、しかも、それが１対１マッピングである、テーブルＭＡＰが構築される。制御フローグラフに見られるアークと同じ数の出力ポートがＦＳＭに構築される。テーブルＭＡＰは、アークを、ＦＳＭの出力ポートにマップすることによって機能する。位置ＭＡＰ（Ａ）は、制御フローグラフアークＡを、ＦＳＭの出力ピンにマップする。逆関数ＰＡＭは、ＦＳＭの出力ピンを、対応する制御フローグラフアークにマップするのに用いられる。機能ＦＬＯＰは、状態ノードを、フリップフロップにマップする。

【００８０】

図１８は、機能ＭＡＰを構築する工程を例示するのに役立つ。リセットアークをＦＳＭ１８０４にマップした後、制御フローグラフ１８０２が示される。リセットノードのアウト-アークは、それがなければ割り当てられないままだった出力ピンＰに割り当てられる。"ｒｅｓｅｔ"と命名される入力ピンが構築され、直接Ｐに接続される。ＭＡＰ（Ａ）からＰが設定される。次に、制御フローグラフ１８０２の状態ノード全てが考慮される。各ノードは、一つのイン-アークと一つのアウト-アークを持つ。例えば、状態ノード”Ｎ”における”Ｃ”と”Ｄ”である。ノードＮは、割り当てされないフリップフロップＦに、ＭＡＰ（Ｃ）は、ＦのＤ－ピンに、そして、ＭＡＰ（Ｄ）は、ＦのＱ－ピンに、割り当てられる。ＦＬＯＰ（Ｎ）は、Ｆに設定される。これが、全ての状態ノードが割り当てられるまで繰り返される。こうして、状態ノードの全てのイン-アークとアウト-アーク、およびリセットがＭＡＰに収められる。

【００８１】

次ぎのステップは、状態ノードに入るアークを観察することである。Ｃをそのようなアークとしよう。ＭＡＰ（Ｃ）は、状態フリップフロップのＤ－ピンに接続される。一次入力と状態フリップフロップによって駆動され、かつ、ＭＡＰ（Ｃ）を駆動する回路が構築される。下の表のような再帰工程が、このような回路を構築する。

【００８２】

【外５】

TABLE I Recursive Procedure

```
Procedure cct(c)

T = PAM(c);
        N = { the node at the feather end of C }
        If (N is a state node) {
                Connect T to the Q pin of FLOP(N).
        } else if (N is a join node with K in-arcs) {
                G = a new k-input OR gate
                Connect the output of G to T.
                for (each in-arc A of N) {
                        Let P = MAP(A).
                        If (P is null) {
                                construct a new output pin named P.
                                set MAP(A) = P.
                                call cct(A).
                        }
                        connect P to an unconnected input pin of G.

                }
        } else if (N is a fork node) {
                Construct a new 2-input AND gate G.
                Connect the output of G to T.
                Let A be the in-arc of N.
                Let P = MAP(A).
                If (P is null) {
                        Construct a new output pin named P.
                        Set map(A) = P
                        Call cct(A).
                }
                Connect one input of G to P
                Make  the other input of G a primary status input
        Corresponding to the branch condition
                        That is annotated onto A.
        } else if (N is the reset node) {
                Connect P to the reset input.
        }else{
                Let A be the in-arc of N.
                Let P = MAP(A).
                If (P is null) {
                        Construct a new output pin named P.
                        Set map(A) = P
                        Call cct(A).
                }
                Connect P to T.
        }
        end
```

【００８３】

図１９Ａ－１９Ｄは、完全なワン・ホットＦＳＭを、表の工程によって構築可能とする、各種断片構築体を示す。ＯＲ－ゲートとフリップフロップは、その関連するノードに因んで命名された。ＡＮＤ－ゲートは、その関連するアークに因んで命名された。分枝アークＡとＢに関連する状態は、一次入力ｃｏｎｄ（Ａ）とｃｏｎｄ（Ｂ）として取り扱った。全ての分枝条件は、他の場所で計算されたものと仮定する。

【００８４】

図２０は、Ｖｅｒｉｌｏｇテキストサンプル２００２を、その対応する制御フローグラフ２００４とワン・ホットＦＳＭ２００６と共に示す。単純、非状態、非フォーク、非結合、非リセット、ノードによって接続されるアークに関連する冗長出力は取り潰された。簡単明瞭とするため、幾つかの制御出力も取り除いた。

【００８５】

本発明の実施態様を構築するのに役立つと思われる、優良な一般的参考書が幾つか出版されている。例えば、Ｄｅ Ｍｉｃｈｅｌｉ, Ｓｙｎｔｈｅｓｉｓ ａｎｄ Ｏｐｔｉｍｉｚａｔｉｏｎ ｏｆ Ｄｉｇｉｔａｌ Ｃｉｒｃｕｉｔｓ, ＭｃＧｒａｗ-Ｈｉｌ

l，１９９４年がある。これは、スケジューリング、割り当て、および、高レベル、ロジック合成のその他の局面をカバーする。Ｂ．　Ｇｒｅｇｏｒｙ　ｅｔ　ａｌ．米国特許第５，６６１，６６１号、「レジスタおよびラッチ想定」。これは、ＨＤＬの使用法、および、制御フローグラフと同様の内部表示をカバーする。Ｄ．　Ｋｎａｐｐ　ａｎｄ　Ｍ．　Ｗｉｎｓｌｅｔｔ，　Ａ　Ｐｒｅｓｃｒｉｔｉｖｅ　Ｆｏｒｍａｔ　Ｍｏｄｅｌ　ｆｏｒ　Ｄａｔａ-Ｐａｔｈ　ｈａｒｄｗａｒｅ，ＩＥＥＥ　Ｔｒａｎｓａｃｔｉｏｎｓ　ｏｎ　ＣＡＤ，１９９２年2月、１５８－１８４ページ。これは、制御フローグラフと構造と機能が類似の、データパスと制御ハードウェアの表示を記述する。この表示は、レジスタ予想、スケジューリングや割り当てに使用が可能なもので、再現の主要能力の必要サブセットである。これは、ユーザーまたはその他の代行者が、手動編集によって設計を「破壊」した場合、前記全ての補正に必要な情報を供給するものである。Ｄ．　Ｋｎａｐｐ，　Ｓｙｎｔｈｅｓｉｓ　ｆｒｏｍ　Ｐａｒｔｉａｌ　Ｓｔｒｕｃｔｕｒｅ　ｉｎ　Ｄｅｓｉｇｎ　Ｍｅｔｈｏｄｏｌｏｇｉｅｓ　ｆｏｒ　ＶＬＳＩ　ａｎｄ　Ｃｏｍｐｕｔｅｒ　Ａｒｃｈｉｔｅｃｔｕｒｅ，　Ｄ．Ａ．　Ｅｄｗａｒｄｓ編、Ｅｌｓｅｖｉｅｒ，１９８９年がある。これは、本明細書に記載したものと類似の制御フローグラフを構築するために使用されるハードウェア記述言語について説明する。これも、レジスタ想定、および、他の構造的、行動的情報の想定に使用される。Ｄ．　Ｋｎａｐｐ　ａｎｄ　Ａ．Ｃ．　Ｐａｒｋｅｒ，　Ａ　Ｕｎｉｆｉｅｄ　Ｒｅｐｒｅｓｅｎｔａｔｉｏｎ　ｆｏｒ　Ｄｅｓｉｇｎ　Ｉｎｆｏｒｍａｔｉｏｎ　ｉｎ　Ｐｒｏｃｅｅｄｉｎｇｓ　ｏｆ　ＣＨＤＬ-８５，　Ｅｌｓｅｖｉｅｒ（１９８５年）は、本明細書で使用される制御フローグラフの早期版について、構造や行動の、他の再現法、および、設計の物理的レイアウトと一緒に、説明する。

【００８６】

本明細書では、本発明は、好ましい実施態様に関連して説明されたが、当業者であれば、本明細書に記載されているものの代わりに、他の運用を、本発明の精神や範囲を逸脱することなく、置換することが可能であることを直ちに認めるであろう。従って、本発明は、上述の請求項のみによって限定されるべきものとする。

【図面の簡単な説明】

【図１】　インターネットウェブサイト、および、システム・オン・チップ設計者のためのＥＤＡオンデマンド解決法を有するウェブサーバーを含む、本発明の、業者間アプリケーションサービスプロバイダ実施態様の機能的ブロックダイグラムである。

【図２】　本発明の、電子的設計自動化法実施態様のフローチャート図である。

【図３】　本発明の、タイミング分析法実施態様のフローチャート図である。

【図４】　４Ａ、４Ｂおよび４Ｃは、回路から、ロジックツリー、および、簡単化ツリーへの移行を表わす模式図である。

【図５】　入力境界に一組の複雑モデルアーク、内部に一組の簡単化モデルアーク、さらに出力境界に別の一組の複雑モデルアークを含む設計をあらわす模式図である。

【図６】　電子的設計自動化における、一般的グラフ適合問題を示す模式図である。

【図７】　回路と、対応する、２部分グラフ表示の模式図である。

【図８】　ネットワークグラフと、一揃いのツリーに分割する、技術マッピングにおける最初のステップの模式図である。

【図９】　分解された技術ライブラリーを表わす模式図である。

【図１０】　右に回路ツリーを示し、左に、その回路ツリーの全ての部分に適合させるのに必要とされる、ただ２個のパターンツリーを示す模式図。

【図１１】　本発明のカバリング選択法実施態様の模式図である。

【図１２】　回路のツリーへの分割、および、それらのツリーを順序付けて、基本規則に合致するリストに構成する模式図である。

【図１３】　制御信号が、どのようにして最重要タイミング経路を支配することが可能なのかを示す実例の模式図である。

【図１４】　ソースノードＳとシンクノードＴの間で、Ｖｅｒｉｌｏｇの時系列ブロック

"ｂｅｇｉｎ．．．ｅｎｄ"が、制御フローグラフのアークＡに変換されるところを示す模式図である。

【図１５】　全てのオプションを含む"ｉｆ"宣言文と、その制御フローグラフ減少の実例の模式図である。

【図１６】　１６Ａ－１６Ｄは、Ｖｅｒｉｌｏｇにおける各種ループと、その対応制御フローグラフ減少を表わす模式図である。

【図１７】　左に、高レベル合成に好適な、簡単なＶｅｒｉｌｏｇ ＨＤＬテキストを、右に、そのテキストに対応する、完全に減少された制御フローグラフを掲げる模式図である。

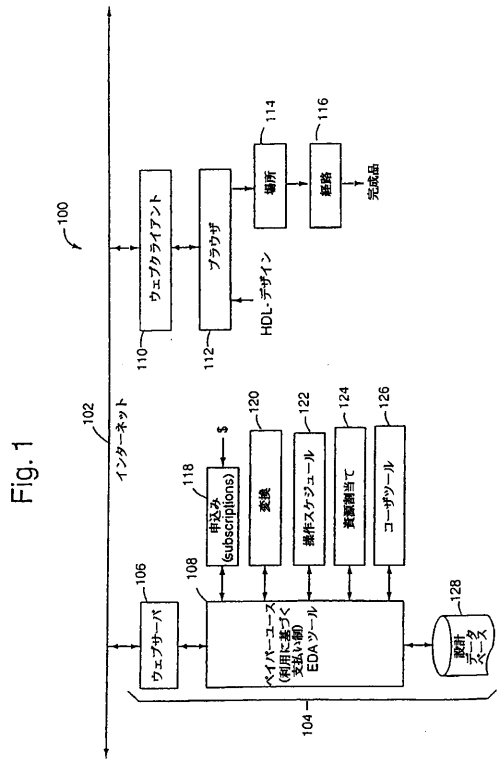
【図１８】　機能的ＭＡＰを構築する手順を示す模式図である。

【図１９】　１９Ａ－１９Ｄは、それによって、完全なワン・ホットＦＳＭが、表の手順によって構築可能となる、各種断片構築を表わす模式図である。

【図２０】　Ｖｅｒｉｌｏｇのテキスト見本、その制御フローグラフ、および、最終的ワン・ホットＦＳＭ間の一致を示す模式図である。

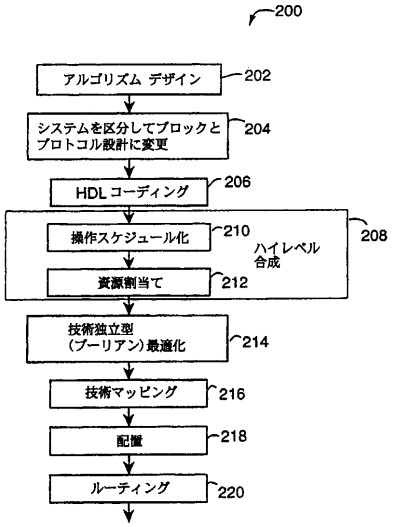
【図１】

Fig. 1



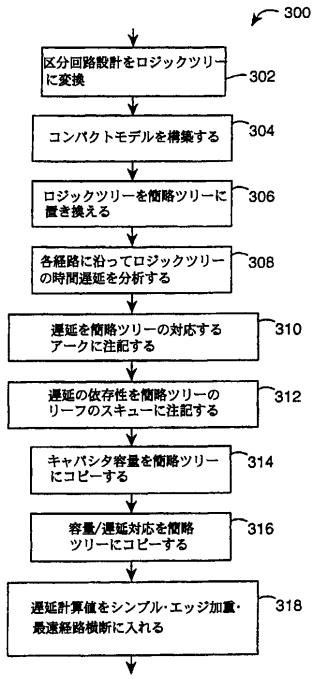
【図２】

Fig. 2

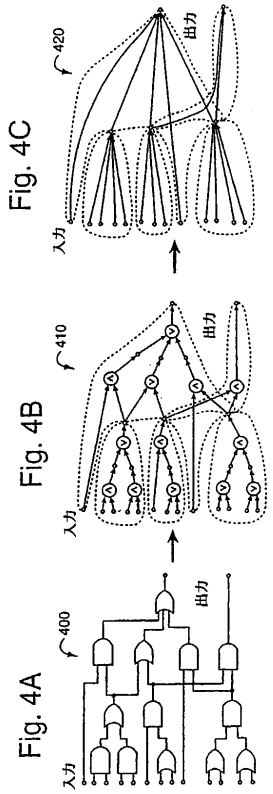


【図３】

Fig. 3

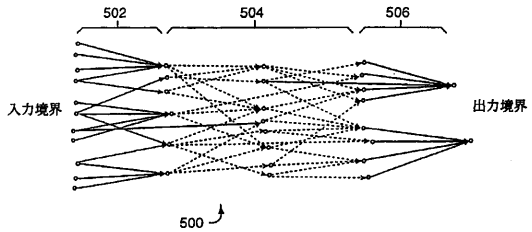


【図４】



【図５】

Fig. 5



【図６】

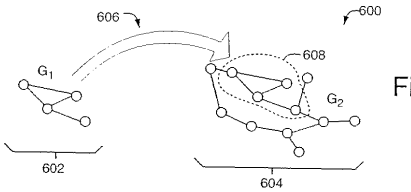
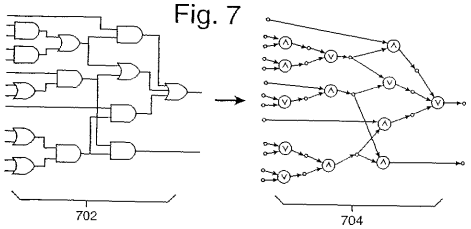


Fig. 6

【図７】

Fig. 7



【図８】

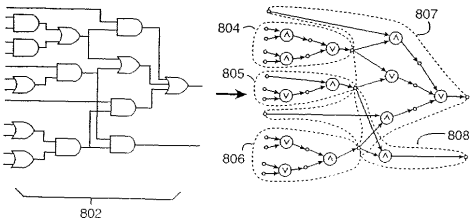


Fig. 8

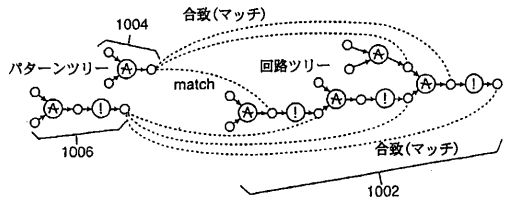
【図９】

Fig. 9

inv	! a	
and2	a b	
nand2	! (a b)	
and4	a b c d	

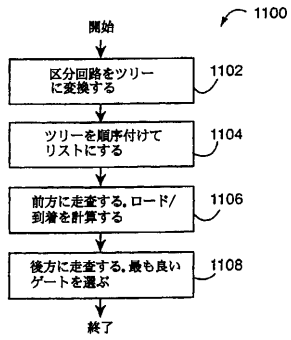
【図１０】

Fig. 10



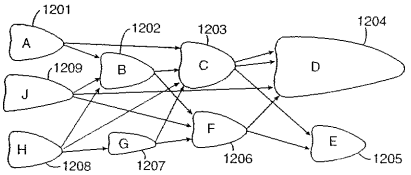
【図１１】

Fig. 11



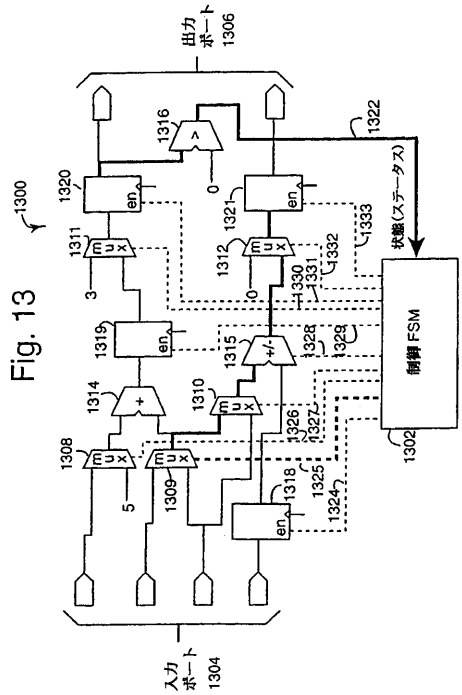
【図１２】

Fig. 12



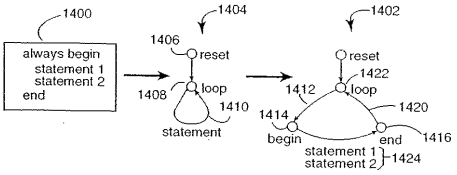
【図１３】

Fig. 13



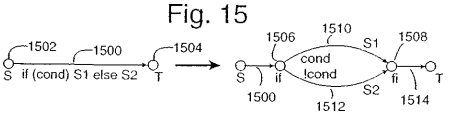
【図１４】

Fig. 14



【図１５】

Fig. 15

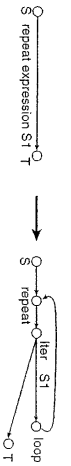


【図１６Ａ】

Fig. 16A

【図１６Ｂ】

Fig. 16B

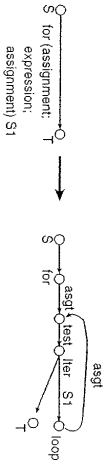


【図１６Ｃ】

Fig. 16C

【図１６Ｄ】

Fig. 16D



【図１７】

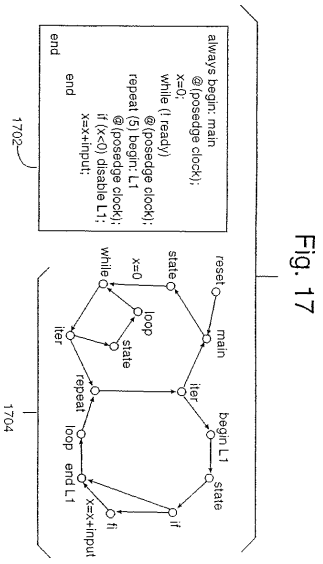


Fig. 17

【図１８】

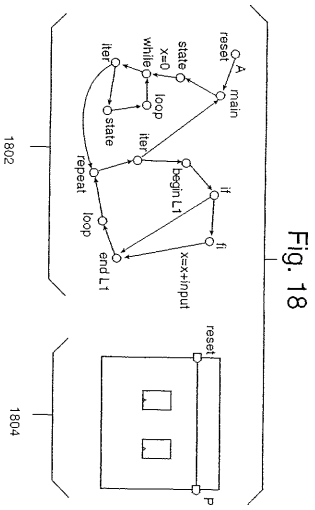


Fig. 18

【図１９Ａ】

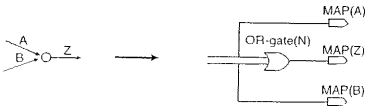
Fig. 19A

【図１９Ｂ】

Fig. 19B

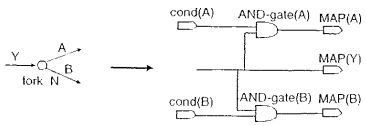
【図１９Ｃ】

Fig. 19C



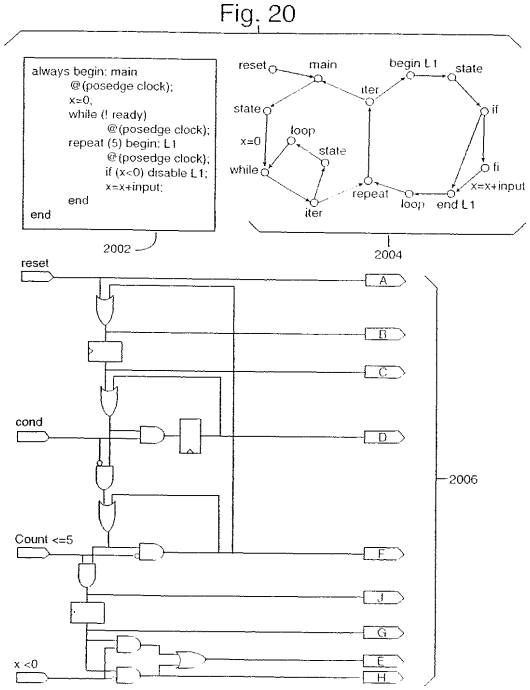
【図１９Ｄ】

Fig. 19D



【図２０】

Fig. 20



フロントページの続き

(51)Int.Cl. F I
G06F 17/60 332
H01L 21/82 C

(31)優先権主張番号 60/136,126
(32)優先日 平成11年5月26日(1999.5.26)
(33)優先権主張国 米国(US)
(31)優先権主張番号 09/574,572
(32)優先日 平成12年5月17日(2000.5.17)
(33)優先権主張国 米国(US)
(31)優先権主張番号 09/574,693
(32)優先日 平成12年5月17日(2000.5.17)
(33)優先権主張国 米国(US)
(31)優先権主張番号 09/577,426
(32)優先日 平成12年5月22日(2000.5.22)
(33)優先権主張国 米国(US)
(31)優先権主張番号 09/579,825
(32)優先日 平成12年5月25日(2000.5.25)
(33)優先権主張国 米国(US)

(72)発明者 ベルント ブラウン
アメリカ合衆国 カリフォルニア州 ９４０２５ メンロ パーク フレモント アヴェニュー ６２４
(72)発明者 デヴィッド ナップ
アメリカ合衆国 カリフォルニア州 ９５１３１ サンノゼ グース ポイント コモン １２８１
(72)発明者 ブラディープ フェルナンデス
アメリカ合衆国 カリフォルニア州 ９５１３２ サンノゼ ミリンダ ドライヴ ３９１８
(72)発明者 ハンスーヨーアヒム シュミット
ドイツ国 ８１９２５ ミュンヘン ヴァーンフリートアレー １ベー

審査官 早川 学

(56)参考文献 特開平０９－２６９９５７（ＪＰ，Ａ）
特開昭６２－２０２２６８（ＪＰ，Ａ）
特開平１１－２８２８８４（ＪＰ，Ａ）
特開昭６３－０７６０６５（ＪＰ，Ａ）
Chakrabarti, D.R. et al., WADE: a Web-based Automated Parallel CAD Environment, PROCEEDINGS Fifth International Conference on High Performance Computing, IEEE, １９９８年１２月２０日, pp.473-480
長谷川拓己, 外４名, 大規模回路向けタイミング解析システムＨＥＡＲＴ（１）高速化の手法, 情報処理学会全国大会講演論文集, 情報処理学会, １９８７年, Vol.35, No.3, p.2275-2276

(58)調査した分野(Int.Cl., ＤＢ名)
G06F 17/50
G06Q 30/00
H01L 21/82