(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2007-18313
(P2007-18313A)
(43)公開日　平成19年1月25日(2007.1.25)

(51)Int.Cl.		F I			テーマコード（参考）
G06F 17/50	(2006.01)	G06F	17/50	654M	5B046
		G06F	17/50	664A	

審査請求 有　請求項の数 5　OL　（全 18 頁）

(21)出願番号　特願2005-199822 (P2005-199822)
(22)出願日　平成17年7月8日 (2005.7.8)

(71)出願人　000005223
富士通株式会社
神奈川県川崎市中原区上小田中４丁目１番１号
(74)代理人　100101856
弁理士　赤澤　日出夫
(72)発明者　中村　義彦
栃木県小山市城東３丁目２８番１号　富士通ＳＣＭシステムズ株式会社内
(72)発明者　平本　覚
栃木県小山市城東３丁目２８番１号　富士通ＳＣＭシステムズ株式会社内
(72)発明者　峰　正高
栃木県小山市城東３丁目２８番１号　富士通ＳＣＭシステムズ株式会社内

Ｆターム(参考)　5B046 AA08　BA03　JA05

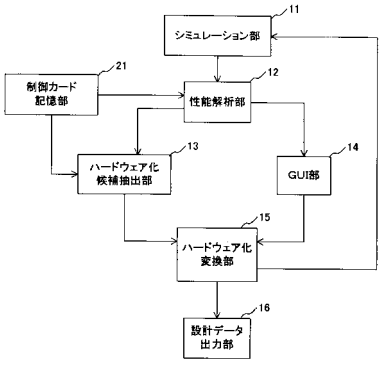
(54)【発明の名称】回路設計プログラム、回路設計装置、回路設計方法

(57)【要約】

【課題】　回路設計において、ハードウェアとソフトウェアの適切な分割を行う回路設計プログラム、回路設計装置、回路設計方法を提供する。

【解決手段】　回路の動作を表す第１のファームウェアのシミュレーションを行い、実行中の関数の識別子と時刻を含むログを出力するシミュレーションステップと、ログとファームウェア記述ルールとに基づいて、第１のファームウェアの関数毎の処理時間を算出すると共に、並列動作可能な関数を抽出する性能解析ステップと、並列動作可能な関数のうち、実装時にハードウェアに変換する関数であるハードウェア化関数とハードウェア化関数の処理時間の指定を行うハードウェア化指定ステップと、第１のファームウェアを、ハードウェア化関数に対応するハードウェアブロックに接続する第２のファームウェアに変換するハードウェア化変換ステップとをコンピュータに実行させる。

【選択図】　図３



【特許請求の範囲】
【請求項１】
ハードウェアと該ハードウェア上で実行されるファームウェアで構成される回路について、システム記述言語を用いた設計をコンピュータに実行させる回路設計プログラムであって、
前記回路の動作を表す第１のファームウェアのシミュレーションを行い、該シミュレーション実行中の関数の識別子と時刻を含むログを出力するシミュレーションステップと、
前記ログと並列動作可能な関数の記述のルールを定めたファームウェア記述ルールとに基づいて、前記第１のファームウェアの関数毎の処理時間を算出すると共に、前記第１のファームウェアから並列動作可能な関数を抽出する性能解析ステップと、
前記並列動作可能な関数のうち、実装時にハードウェアに変換する関数であるハードウェア化関数と該ハードウェア化関数の処理時間の指定を行うハードウェア化指定ステップと、
前記第１のファームウェアを、前記ハードウェア化関数に対応するハードウェアブロックに接続する第２のファームウェアに変換するハードウェア化変換ステップと、
をコンピュータに実行させる回路設計プログラム。
【請求項２】
請求項１に記載の回路設計プログラムにおいて、
前記ハードウェア化指定ステップは、アクティビティ図上に前記関数毎の処理時間と前記並列動作可能な関数を表示することを特徴とする回路設計プログラム。
【請求項３】
請求項１または請求項２に記載の回路設計プログラムにおいて、
前記ハードウェア化変換ステップは、前記第１のファームウェアから前記ハードウェア化関数の呼び出しを削除すると共に、前記ハードウェアブロックとの接続を追加することにより、前記第２のファームウェアに変換することを特徴とする回路設計プログラム。
【請求項４】
ハードウェアと該ハードウェア上で実行されるファームウェアで構成される回路について、システム記述言語を用いた設計を行う回路設計装置であって、
前記回路の動作を表す第１のファームウェアのシミュレーションを行い、該シミュレーション実行中の関数の識別子と時刻を含むログを出力するシミュレーション部と、
前記ログと並列動作可能な関数の記述のルールを定めたファームウェア記述ルールとに基づいて、前記第１のファームウェアの関数毎の処理時間を算出すると共に、前記第１のファームウェアから並列動作可能な関数を抽出する性能解析部と、
前記並列動作可能な関数のうち、実装時にハードウェアに変換する関数であるハードウェア化関数と該ハードウェア化関数の処理時間の指定を行うハードウェア化指定部と、
前記第１のファームウェアを、前記ハードウェア化関数に対応するハードウェアブロックに接続する第２のファームウェアに変換するハードウェア化変換部と、
を備えてなる回路設計装置。
【請求項５】
ハードウェアと該ハードウェア上で実行されるファームウェアで構成される回路について、システム記述言語を用いた設計を実行する回路設計方法であって、
前記回路の動作を表す第１のファームウェアのシミュレーションを行い、該シミュレーション実行中の関数の識別子と時刻を含むログを出力するシミュレーションステップと、
前記ログと並列動作可能な関数の記述のルールを定めたファームウェア記述ルールとに基づいて、前記第１のファームウェアの関数毎の処理時間を算出すると共に、前記第１のファームウェアから並列動作可能な関数を抽出する性能解析ステップと、
前記並列動作可能な関数のうち、実装時にハードウェアに変換する関数であるハードウェア化関数と該ハードウェア化関数の処理時間の指定を行うハードウェア化指定ステップと、
前記第１のファームウェアを、前記ハードウェア化関数に対応するハードウェアブロッ

クに接続する第２のファームウェアに変換するハードウェア化変換ステップと、
を実行する回路設計方法。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、システム記述言語を用いて、ハードウェアとソフトウェアで構成される回路を設計する際、ハードウェアとソフトウェアの分割を行う回路設計プログラム、回路設計装置、回路設計方法に関するものである。
【背景技術】
【０００２】
現在、ＬＳＩ（Large-scale Integrated Circuit）開発では、設計の上流においてシステム記述言語で記述された機能モデルを作成することにより、ハードウェア記述言語で記述される実際のハードの機能や性能の検証を行うことが要求されている。現在、システム記述言語としては、例えばＣ言語を用いるものが提供されており、併せて設計環境や検証環境が提供されている。しかし、実際には、工数や開発期間の問題から、単純な機能モデルである期待値生成プログラムを作成するのが限界である。更に、上述したシステム記述言語によるリファレンスモデルは、ＬＳＩの機能をハードウェアとソフトウェアに分割する場合の検討にも用いられてきている。
【０００３】
なお、本発明の関連ある従来技術として、例えば、下記に示す特許文献１が知られている。この半導体集積回路の設計装置は、ソフトウェア記述のうちハードウェア化する部分のアルゴリズムが指定されるに応じて、指定された部分をハードウェアの制御のアルゴリズムに置き換え、その度に、所望の性能を達成しているか否かの判断を行うものである。
【特許文献１】特開２００５－６３１３６号公報
【発明の開示】
【発明が解決しようとする課題】
【０００４】
最近では、システム記述言語の検証環境において定量的に性能を計測し、ハードウェアとソフトウェアの分割を行う方法が確立されてきているが、性能を計測するために必要な処理時間は経験則で設定する必要があり、エキスパートが経験を元に分割を行うしかなかった。それでも、最適な分割を導くために手間が掛かる、最適でない分割になる、等の問題が発生していた。
【０００５】
本発明は上述した問題点を解決するためになされたものであり、回路設計において、ハードウェアとソフトウェアの適切な分割を行う回路設計プログラム、回路設計装置、回路設計方法を提供することを目的とする。
【課題を解決するための手段】
【０００６】
上述した課題を解決するため、本発明は、ハードウェアと該ハードウェア上で実行されるファームウェアで構成される回路について、システム記述言語を用いた設計をコンピュータに実行させる回路設計プログラムであって、前記回路の動作を表す第１のファームウェアのシミュレーションを行い、該シミュレーション実行中の関数の識別子と時刻を含むログを出力するシミュレーションステップと、前記ログと並列動作可能な関数の記述のルールを定めたファームウェア記述ルールとに基づいて、前記第１のファームウェアの関数毎の処理時間を算出すると共に、前記第１のファームウェアから並列動作可能な関数を抽出する性能解析ステップと、前記並列動作可能な関数のうち、実装時にハードウェアに変換する関数であるハードウェア化関数と該ハードウェア化関数の処理時間の指定を行うハードウェア化指定ステップと、前記第１のファームウェアを、前記ハードウェア化関数に対応するハードウェアブロックに接続する第２のファームウェアに変換するハードウェア化変換ステップとをコンピュータに実行させるものである。

【０００７】
また、本発明に係る回路設計プログラムにおいて、前記ハードウェア化指定ステップは、アクティビティ図上に前記関数毎の処理時間と前記並列動作可能な関数を表示することを特徴とするものである。
【０００８】
また、本発明に係る回路設計プログラムにおいて、前記ハードウェア化変換ステップは、前記第１のファームウェアから前記ハードウェア化関数の呼び出しを削除すると共に、前記ハードウェアブロックとの接続を追加することにより、前記第２のファームウェアに変換することを特徴とするものである。
【０００９】
また、本発明に係る回路設計プログラムにおいて、前記第１のファームウェアは、実行中の関数の識別子と時刻を含むログを出力する機能を含むことを特徴とするものである。
【００１０】
また、本発明に係る回路設計プログラムにおいて、前記ハードウェア化指定ステップは、前記並列動作可能な関数のうち予め設定された処理時間の基準に適するものを前記ハードウェア化関数として指定し、該ハードウェア化関数の処理時間を予め設定された値に指定することを特徴とするものである。
【００１１】
また、本発明に係る回路設計プログラムにおいて、前記ハードウェア化指定ステップは、前記並列動作可能な関数のうち予め設定された処理時間の基準に適するものを前記ハードウェア化関数の候補として表示し、ユーザの操作に従ってハードウェア化関数と該ハードウェア化関数の処理時間を指定することを特徴とするものである。
【００１２】
また、本発明に係る回路設計プログラムにおいて、前記ハードウェア化指定ステップは、前記並列動作可能な関数を表示し、前記表示ユーザの操作に従ってハードウェア化関数と該ハードウェア化関数の処理時間を指定することを特徴とするものである。
【００１３】
また、本発明は、ハードウェアと該ハードウェア上で実行されるファームウェアで構成される回路について、システム記述言語を用いた設計を行う回路設計装置であって、前記回路の動作を表す第１のファームウェアのシミュレーションを行い、該シミュレーション実行中の関数の識別子と時刻を含むログを出力するシミュレーション部と、前記ログと並列動作可能な関数の記述のルールを定めたファームウェア記述ルールとに基づいて、前記第１のファームウェアの関数毎の処理時間を算出すると共に、前記第１のファームウェアから並列動作可能な関数を抽出する性能解析部と、前記並列動作可能な関数のうち、実装時にハードウェアに変換する関数であるハードウェア化関数と該ハードウェア化関数の処理時間の指定を行うハードウェア化指定部と、前記第１のファームウェアを、前記ハードウェア化関数に対応するハードウェアブロックに接続する第２のファームウェアに変換するハードウェア化変換部とを備えたものである。
【００１４】
また、本発明に係る回路設計装置において、前記ハードウェア化指定部は、アクティビティ図上に前記関数毎の処理時間と前記並列動作可能な関数を表示することを特徴とするものである。
【００１５】
また、本発明に係る回路設計装置において、前記ハードウェア化変換部は、前記第１のファームウェアから前記ハードウェア化関数の呼び出しを削除すると共に、前記ハードウェアブロックとの接続を追加することにより、前記第２のファームウェアに変換することを特徴とするものである。
【００１６】
また、本発明に係る回路設計装置において、前記第１のファームウェアは、実行中の関数の識別子と時刻を含むログを出力する機能を含むことを特徴とするものである。

【００１７】
また、本発明に係る回路設計装置において、前記ハードウェア化指定部は、前記並列動作可能な関数のうち予め設定された処理時間の基準に適するものを前記ハードウェア化関数として指定し、該ハードウェア化関数の処理時間を予め設定された値に指定することを特徴とするものである。
【００１８】
また、本発明に係る回路設計装置において、前記ハードウェア化指定部は、前記並列動作可能な関数のうち予め設定された処理時間の基準に適するものを前記ハードウェア化関数の候補として表示し、ユーザの操作に従ってハードウェア化関数と該ハードウェア化関数の処理時間を指定することを特徴とするものである。
【００１９】
また、本発明に係る回路設計装置において、前記ハードウェア化指定部は、前記並列動作可能な関数を表示し、前記表示ユーザの操作に従ってハードウェア化関数と該ハードウェア化関数の処理時間を指定することを特徴とするものである。
【００２０】
また、本発明は、ハードウェアと該ハードウェア上で実行されるファームウェアで構成される回路について、システム記述言語を用いた設計を実行する回路設計方法であって、前記回路の動作を表す第１のファームウェアのシミュレーションを行い、該シミュレーション実行中の関数の識別子と時刻を含むログを出力するシミュレーションステップと、前記ログと並列動作可能な関数の記述のルールを定めたファームウェア記述ルールとに基づいて、前記第１のファームウェアの関数毎の処理時間を算出すると共に、前記第１のファームウェアから並列動作可能な関数を抽出する性能解析ステップと、前記並列動作可能な関数のうち、実装時にハードウェアに変換する関数であるハードウェア化関数と該ハードウェア化関数の処理時間の指定を行うハードウェア化指定ステップと、前記第１のファームウェアを、前記ハードウェア化関数に対応するハードウェアブロックに接続する第２のファームウェアに変換するハードウェア化変換ステップとを実行するものである。
【発明の効果】
【００２１】
本発明によれば、予め制御カードで設定された基準に従って、ハードウェア化関数を指定することにより、適切なハードウェアとソフトウェアの分割を行うことができる。また、アクティビティ図上に関数毎の処理サイクル時間と並列動作可能な関数を表示し、更にハードウェア化を行った場合の処理サイクル時間を表示することにより、適切なハードウェアとソフトウェアの分割を行うことができる。また、シミュレーションを行ったリファレンスモデルをそのままファームウェアとして利用することができることから、性能測定、ハードウェアとソフトウェアの分割、実装を効率的に行うことができる。
【発明を実施するための最良の形態】
【００２２】
以下、本発明の実施の形態について図面を参照しつつ説明する。
【００２３】
本実施の形態では、設計する回路をＬＳＩとし、リファレンスモデルはＣ言語で記述されるものとする。
【００２４】
まず、本発明に係る回路設計装置が対象とするＬＳＩのアーキテクチャについて説明する。
【００２５】
図１は、本発明に係る回路設計装置が対象とするＬＳＩのアーキテクチャの一例を示すブロック図である。ＬＳＩ１００は、ＣＰＵ（Central Processing Unit）１０１、データパス入力バッファ１０２、データパス出力バッファ１０３、ハードウェアブロック１０４、バス１０５、メモリ１０６、外部メモリＩＦ（インタフェイス）１０７を備え、外部メモリＩＦ１０７は、ＬＳＩ１００の外部に備えられた外部メモリ１０８に接続される。

【００２６】
次に、ＬＳＩの検証のためにシステム記述言語で記述されるリファレンスモデルについて説明する。
【００２７】
図２は、本発明に係る回路設計装置が対象とするＬＳＩのリファレンスモデルの一例を示すブロック図である。ＬＳＩ１００をモデル化したリファレンスモデル１１０は、ＣＰＵ１０１とその上で動作するファームウェア１２０をモデル化したＣＰＵモデル１１１、データパス入力バッファ１０２をモデル化したデータパス入力バッファモデル１１２、データパス出力バッファ１０３をモデル化したデータパス出力バッファモデル１１３、ハードウェアブロック１０４をモデル化したハードウェアブロックモデル１１４、バス１０５をモデル化したバスモデル１１５を備える。
【００２８】
次に、本発明に係る回路設計装置の構成について説明する。
【００２９】
図３は、本発明に係る回路設計装置の構成の一例を示すブロック図である。この回路設計装置は、シミュレーション部１１、性能解析部１２、ハードウェア化候補抽出部１３、ＧＵＩ部１４、ハードウェア化変換部１５、設計データ出力部１６、制御カード記憶部２１を備える。
【００３０】
次に、本発明に係る回路設計装置の動作について説明する。
【００３１】
図４は、本発明に係る回路設計装置の動作の一例を示すフローチャートである。まず、シミュレーション部１１は、ファームウェアとテストベンチを読み込み（Ｓ１１）、コンパイルを行った後、シミュレーションを実行し、実行中にログを出力する（Ｓ１２）。ここで、ファームウェアは、性能測定用ファームウェアであり、ハードウェアブロックを用いずに全ての機能をソフトウェアで実現すると共に、性能測定機能を持つファームウェアである。性能測定機能により、実行中の関数の情報を含むログが出力される。また、この性能測定用ファームウェアは、ハードウェアとソフトウェアの分割の後、最終的には性能測定用の機能が省かれ、実装用ファームウェアとなるものである。次に、性能解析部１２とハードウェア化候補抽出部１３とＧＵＩ部１４は、ハードウェアとソフトウェアの分割処理を行う（Ｓ１３）。次に、設計データ出力部１６は、ファームウェアの性能が合格であるか否かの判断を行う（Ｓ１４）。
【００３２】
性能が合格でなければ（Ｓ１４，Ｎ）、ユーザの指示により性能測定用ファームウェアが変更され（Ｓ１５）、処理Ｓ１２へ戻る。一方、性能が合格であれば（Ｓ１４，Ｙ）、設計データ出力部１６は、性能測定用ファームウェアの変更が行われたか否かの判断を行う（Ｓ１７）。性能測定用ファームウェアの変更が行われていない場合（Ｓ１７，Ｎ）、このフローを終了する。一方、性能測定用ファームウェアの変更が行われた場合（Ｓ１７，Ｙ）、設計データ出力部１６は、性能測定用ファームウェアを実装用のファームウェアに変換して出力すると共に、ハードウェアブロックモデル、アーキテクチャモデルを出力し（Ｓ１８）、このフローを終了する。
【００３３】
ここで、実装用ファームウェアでは、性能測定用ファームウェアに含まれるログの出力など性能測定の機能が省かれる。また、実装用ファームウェアとは、ファームウェア１２０に対応し、ハードウェアブロックモデルとは、ハードウェアブロックモデル１１４に対応し、アーキテクチャモデルは、その他のＣＰＵモデル１１１、データパス入力バッファモデル１１２、データパス出力バッファモデル１１３、バスモデル１１５に対応する。
【００３４】
次に、ハードウェアとソフトウェアの分割処理について説明する。
【００３５】

図５は、本発明に係るハードウェアとソフトウェアの分割処理の動作の一例を示すフローチャートである。まず、性能解析部１２は、ログを読み込み、性能解析処理を行う（Ｓ２０）。次に、性能解析部１２は、制御カード記憶部２１に予め設定された制御カードに従って、自動ハードウェア候補抽出を行うか否かの判断を行う（Ｓ２１）。自動ハードウェア候補抽出を行わない場合（Ｓ２１，Ｎ）、処理Ｓ３１へ移行する。一方、自動ハードウェア候補抽出を行う場合（Ｓ２１，Ｙ）、ハードウェア化候補抽出部１３は、制御カードに従って自動ハードウェア候補抽出を行う（Ｓ２２）。次に、性能解析部１２は、制御カードに従ってＧＵＩ表示を行うか否かの判断を行う（Ｓ２３）。

【００３６】

ＧＵＩ表示を行う場合（Ｓ２３，Ｙ）、ＧＵＩ部１４は、ＧＵＩ表示としてアクティビティ図等の表示を行い、アクティビティ図上におけるハードウェア化の操作に伴って、表示の変更を行う（Ｓ３１）。次に、ＧＵＩ部１４は、ユーザの操作に従ってハードウェア化指定を行うか否かの判断を行う（Ｓ３２）。ハードウェア化指定を行わない場合（Ｓ３２，Ｎ）、このフローを終了する。一方、ハードウェア化指定を行う場合（Ｓ３２，Ｙ）、ＧＵＩ部１４は、ハードウェア化指定として、アクティビティ図上におけるユーザの操作に従ってハードウェア化関数の指定とハードウェア化関数の処理サイクル時間の指定を行う。

【００３７】

一方、ＧＵＩ表示を行わない場合（Ｓ２３，Ｎ）、性能解析部１２は、ハードウェア化指定として、ハードウェア化候補をハードウェア化関数として指定し、ハードウェア化関数に対して予め設定された処理サイクル時間を指定する（Ｓ３５）。次に、ハードウェア化変換部１５は、指定されたハードウェア化関数についてハードウェア化変換処理を行い（Ｓ３６）、このフローを終了する。

【００３８】

次に、性能測定用ファームウェアの具体例について説明する。

【００３９】

図６は、本発明に係るハードウェアとソフトウェアの分割前の性能測定用ファームウェアと共通領域の構成の一例を示す図である。この性能測定用ファームウェアは、計測関数定義、ドライバ関数定義、ファームウェアメイン関数（ｆｉｒｍ＿ｍａｉｎ）定義で構成される。ファームウェアメイン関数は、データ処理のための関数Ａ、関数Ｂを呼び出す。また、関数Ａは、関数Ｃを呼び出し、関数Ｂは、関数Ｄ、関数Ｅを呼び出し、関数Ｄは、関数Ｆを呼び出す。共通領域は、フラグは単位データが入力されたことを示す単位データ入力フラグ、関数Ａの単位データ処理が終わったことを示す関数Ａ単位データ処理終了フラグ、関数Ｂの単位データ処理が終わったことを示す関数Ｂ単位データ処理終了フラグ、各関数がアクセスするレジスタａ，ｂ，ｃ，ｄ，ｅを有する。

【００４０】

計測関数として、関数の開始時と終了時に呼び出す処理サイクル時刻ロギング関数が定義される。処理サイクル時刻ロギング関数は、現在のＣＰＵ時間から処理サイクル時刻を求め、引数として入力されたキーワード、呼び出した関数の識別子、処理サイクル時刻をログに出力する。ここで、関数の開始時には処理開始サイクル時刻関数として、キーワード“ＳＴＡＲＴ”を引数として処理サイクル時刻ロギング関数を呼び出し、関数の終了時には処理終了サイクル時刻関数として、キーワード“ＥＮＤ”を引数として処理サイクル時刻ロギング関数を呼び出す。また、処理サイクル時刻ロギング関数は、性能測定用ファームウェアでだけ実行され、実装用ファームウェアでは実行されない。

【００４１】

ドライバ関数の定義においては、データパス入力バッファ１０２から読み出すデバイスドライバとデータパス出力バッファ１０３へ書き込むデバイスドライバが定義される。書き込みデバイスドライバは、キーワード“ＷＲＩＴＥ”を引数として処理サイクル時刻ロギング関数を呼び出し、バスが占有されている間待機し、書き込み中フラグをｔｒｕｅとし、仮想空間エリアに対してアドレスを割り付け、アドレスを割り付けたエリアの部分に

データをコピーし、書き込み中フラグをｆａｌｓｅとし、終了する。読み出しデバイスドライバは、キーワード“ＲＥＡＤ”を引数として処理サイクル時刻ロギング関数を呼び出し、バスが占有されている間待機し、読み出し中フラグをｔｒｕｅとし、仮想空間エリアからアドレスに割り付けられたポインタを抽出し、ポインタから値を取得し、取得した値をデータに代入し、読み出し中フラグをｆａｌｓｅとし、終了する。

【００４２】

ファームウェアメイン関数は、入力データと出力データの定義を有する。また、ファームウェアメイン関数は、読み込みデバイスドライバの呼び出し、データ処理メイン関数（ｐｒｏｃ＿ｍａｉｎ）の呼び出し、書き込みデバイスドライバの呼び出しを有する。

【００４３】

データ処理メイン関数は、処理開始サイクル時刻計測関数の呼び出しを行い、単位データ入力フラグが１であれば、単位データ入力フラグを０とし、以下のデータ処理を実行する。全ての関数の単位データ処理終了フラグが１でなければ、各関数の単位データ処理終了フラグが０であれば対応する関数を呼び出す動作を繰り返す。全ての関数の単位データ処理終了フラグが１になれば、単位データ処理終了フラグを０として処理終了サイクル時刻計測関数を呼び出し、終了する。

【００４４】

データ処理のための関数Ａは、処理開始サイクル時刻計測関数を呼び出し、読み出しデバイスドライバを呼び出し、何らかのデータ処理を行い、書き込みデバイスドライバを呼び出し、関数Ｃを呼び出し、単位データ処理が終了すれば関数Ａの単位データ処理終了フラグを１とし、処理終了サイクル時刻計測関数を呼び出し、終了する。

【００４５】

データ処理のための関数Ｂは、処理開始サイクル時刻計測関数を呼び出し、読み出しデバイスドライバを呼び出し、何らかのデータ処理を行い、書き込みデバイスドライバを呼び出し、関数Ｄを呼び出し、何らかのデータ処理を行い、関数Ｅを呼び出し、単位データ処理が終了すれば関数Ｂの単位データ処理終了フラグを１とし、処理終了サイクル時刻計測関数を呼び出し、終了する。

【００４６】

データ処理のための関数Ｃは、処理開始サイクル時刻計測関数を呼び出し、読み出しデバイスドライバを呼び出し、何らかのデータ処理を行い、書き込みデバイスドライバを呼び出し、処理終了サイクル時刻計測関数を呼び出し、終了する。

【００４７】

データ処理のための関数Ｄは、処理開始サイクル時刻計測関数を呼び出し、読み出しデバイスドライバを呼び出し、何らかのデータ処理を行い、書き込みデバイスドライバを呼び出し、関数Ｆを呼び出し、処理終了サイクル時刻計測関数を呼び出し、終了する。

【００４８】

データ処理のための関数Ｅは、処理開始サイクル時刻計測関数を呼び出し、読み出しデバイスドライバを呼び出し、何らかのデータ処理を行い、書き込みデバイスドライバを呼び出し、処理終了サイクル時刻計測関数を呼び出し、終了する。

【００４９】

データ処理のための関数Ｆは、処理開始サイクル時刻計測関数を呼び出し、読み出しデバイスドライバを呼び出し、何らかのデータ処理を行い、書き込みデバイスドライバを呼び出し、処理終了サイクル時刻計測関数を呼び出し、終了する。

【００５０】

次に、ファームウェア記述ルールについて説明する。ユーザは、以下のファームウェア記述ルールに従ってファームウェアを記述する。

【００５１】

図７は、本発明に係るファームウェアにおける並列動作が可能な関数の一例を示すソースコードである。このように、同じ関数Ａ内で関数Ｂと関数Ｃが呼び出される記述がある場合、性能解析部１２は、関数Ｂと関数Ｃは並列に動作することが可能と判断する。図８

は、本発明に係るファームウェアにおける並列動作が不可能な関数の一例を示すソースコードである。このように、関数Ａ内で関数Ｂが呼び出され、関数Ｂ内で処理の後に関数Ｃが呼び出される場合、関数Ｂ内の処理の後に関数Ｃの処理を行うものとする。

【００５２】

このようなファームウェア記述ルールを設けることにより、ユーザは容易に並列動作可能な関数を記述することができると共に、性能解析部１２は容易にログから構成を解析することができる。

【００５３】

次に、性能解析処理について詳細に説明する。性能解析処理として、構成解析と時間解析が行われる。

【００５４】

まず、性能解析部１２は、シミュレーション部１１によるシミュレーションの実行中に処理サイクル時刻ロギング関数で出力されたログを読み込み、構成解析を行う。関数が呼び出されるごとにｃｏｕｎｔｅｒを用意し、同関数内でキーワード“ＳＴＡＲＴ”があれば、ｃｏｕｎｔｅｒに＋１を行い、キーワード“ＥＮＤ”があればｃｏｕｎｔｅｒに－１を行う。また、ｃｏｕｎｔｅｒ＝０のとき、次のキーワード“ＳＴＡＲＴ”を持つ関数は並列化できる。また、キーワード“ＳＴＡＲＴ”の時のｃｏｕｎｔｅｒが１である関数は直接呼出し関数となる。この動作を全ての関数について行う。

【００５５】

図９は、本発明に係るログと構成解析の結果の一例を示す図である。ここで、例えば、ファームウェアメイン関数のｃｏｕｎｔｅｒにおいて、関数Ａの終了時にｃｏｕｎｔｅｒが０となって関数Ｂが開始することから、関数Ａと関数Ｂは並列動作が可能であると判断される。また、関数Ｂのｃｏｕｎｔｅｒにおいて、関数Ｄの終了時にｃｏｕｎｔｅｒが０となって関数Ｅが開始することから、関数Ｄと関数Ｅは並列動作が可能であると判断される。また、ファームウェアメイン関数のｃｏｕｎｔｅｒにおいて、関数Ａの開始時と関数Ｂの開始時にｃｏｕｎｔｅｒが１となっていることから、関数Ａと関数Ｂは直接呼出し関数であると判断される。

【００５６】

次に、性能解析部１２は、時間解析を行う。同じ関数について、キーワード“ＥＮＤ”の処理サイクル時刻からキーワード“ＳＴＡＲＴ”の処理サイクル時刻を引いたものが、その関数に掛かる時間となる。また、ある関数について、“ＥＮＤ”が出現する前に直接呼出し関数Ｘの“ＳＴＡＲＴ”が出現し、かつ直接呼出し関数Ｙの“ＥＮＤ”が無い場合、関数Ｘの“ＳＴＡＲＴ”の処理サイクル時刻から呼び出し元の“ＳＴＡＲＴ”の処理サイクル時刻を引いたものが、関数Ｘを呼び出すまでに掛かる時間となる。また、ある関数について、“ＥＮＤ”が出現する前に直接呼出し関数Ｘの“ＳＴＡＲＴ”が出現し、かつ直接呼出し関数Ｙの“ＥＮＤ”がある場合、関数Ｘの“ＳＴＡＲＴ”の処理サイクル時刻から関数Ｙの“ＥＮＤ”の処理サイクル時刻を引いたものが、関数Ｘを呼び出すまでに掛かる時間となる。また、“ＥＮＤ”の処理サイクル時刻から直接呼出し関数の“ＥＮＤ”の処理サイクル時刻を引いたものが後処理サイクル時間となる。また、直接呼出し関数以外の関数の“ＳＴＡＲＴ”が重なる場合、下位の関数において時間を求める。この動作を全ての関数について行う。

【００５７】

図１０は、本発明に係るログと時間解析の結果の一例を示す図である。ここで、例えば、関数Ｆの処理サイクル時間は、７５と算出される。また、ファームウェアメイン関数において、関数Ａを呼び出すまでの処理サイクル時間は２１、関数Ａの“ＥＮＤ”から関数Ｂを呼び出すまでの処理サイクル時間は６となる。また、ファームウェアメイン関数において、関数Ｂの後処理サイクル時間は３２となる。

【００５８】

次に、制御カードについて説明する。制御カードは、ハードウェア化候補の抽出やハードウェア化関数の指定に関する設定が記述されたものであり、予め設定される。

【００５９】
図１１は、本発明に係る制御カードの一例を示す図である。ここで、制御カードはテキストで記述されており、そのフォーマットについて説明する。一行にキーワードと設定値が記述されており、キーワードと設定値の間は空白文字列が記載されている。また、キーワードと設定値からなる行は、複数記述することができる。また、“ＶＡＬＵＥ”以外の同じキーワードの行が複数行記述されている場合、同じキーワードの行のうち一番最後の行が有効になる。
【００６０】
図１２は、本発明に係る制御カード中のキーワードと設定値の説明の一例を示す図である。“ＡＵＴＯ＿ＳＥＬＥＣＴ”は自動実行の設定を表し、設定値が０であれば、この制御カードは利用しない。設定値が１であれば、ハードウェア化候補の抽出のみを自動で行い、ＧＵＩを表示する。設定値が２であれば、ハードウェア化指定までを全て自動で行い、ＧＵＩを表示しない。“ＭＥＴＨＯＤ＿ＴＹＰＥ”はハードウェア化候補の抽出方法を表し、設定値が０であれば、サイクル数の平均値に対する割合を基準としてハードウェア化候補を抽出する。設定値が１であれば、サイクル数の絶対値を基準としてハードウェア化候補を抽出する。
【００６１】
“ＡＶＥ＿ＴＨＲＥＳＨＯＬＤ”はハードウェア化候補の関数を抽出する閾値を表し、関数毎のサイクル数の平均値に対する割合を、サイクル数の平均値に対するパーセントで指定する。自動ハードウェア化候補抽出においては、この閾値以上の処理サイクル数を持つ関数が、ハードウェア化候補として抽出される。“ＡＢＳ＿ＴＨＲＥＳＨＯＬＤ”はハードウェア化候補の関数を抽出する閾値を表し、関数毎のサイクル数の絶対値で指定する。自動ハードウェア化候補抽出においては、この閾値以上の処理サイクル数を持つ関数が、ハードウェア化候補として抽出される。“ＮＯＴ＿ＴＡＲＧＥＴＥＤ”はハードウェア化対象外の関数を、関数名で指定する。カンマ区切りを用いて複数の関数名を指定することができる。但し、ＡＵＴＯ＿ＳＥＬＥＣＴ＝２のときにここで指定された関数がハードウェア化候補になってしまった場合は、ＡＵＴＯ＿ＳＥＬＥＣＴ＝１相当の動作を強制的に行う。設定値としては、関数名を指定する。
【００６２】
“ＶＡＬＵＥ”はＡＵＴＯ＿ＳＥＬＥＣＴ＝１以上の場合に利用する。ハードウェア化候補になった場合に指定するサイクル数を関数毎に指定する。ここで、値が定義されていない関数がハードウェア化候補になってしまった場合、ＡＵＴＯ＿ＳＥＬＥＣＴ＝１相当の動作を強制的に行う。
【００６３】
ＡＵＴＯ＿ＳＥＬＥＣＴ＝１である場合、“ＶＡＬＵＥ”で指定されたサイクル数が、ハードウェア化候補のサイクル数としてＧＵＩ部１４により表示される。また、ＡＵＴＯ＿ＳＥＬＥＣＴ＝１である場合、“ＶＡＬＵＥ”で指定されたサイクル数がハードウェア化関数のサイクル数としてハードウェア化変換部１５に渡される。
【００６４】
このような制御カードを用いることにより、回路設計装置がハードウェア化指定を全て自動で行うこともでき、回路設計装置が候補として抽出したハードウェア化関数と処理サイクル時間を表示し、ユーザがＧＵＩ上で確認し、指定することもでき、ユーザがＧＵＩ上で全てのハードウェア化指定を行うこともできる。
【００６５】
次に、ＧＵＩ部１４によるＧＵＩ表示について詳細に説明する。
【００６６】
ＧＵＩ部１４は、自動ハードウェア候補抽出を行わない場合、性能解析部１２による構造解析と時間解析の結果に基づいて、第１のアクティビティ図を生成し、表示する。アクティビティ図とは、ＵＭＬ（Unified Modeling Language）で用いられており、並列処理を記述することができる図である。図１３は、本発明に係るＧＵＩ部により表示される第

１のアクティビティ図の一例を示す図である。この例では、上述したように、関数Ａと関数Ｂは、並列動作が可能であることを表す。また、関数Ｄと関数Ｅは、並列動作が可能であることを表す。また、上述した時間解析で算出された処理サイクル時間が表示される。更に、アクティビティ図の最上部には、図中の処理全体の処理サイクル時間が表示される。

【００６７】

ＧＵＩ表示を行う場合、ユーザは、この第１のアクティビティ図上でハードウェア化指定を行うことができる。例えば、ユーザは、並列動作が可能と表示されている部分の中で処理サイクル時間が大きい部分をハードウェア化関数として指定する。

【００６８】

次に、ＧＵＩ部１４は、ハードウェア化変換部１５による自動ハードウェア化候補抽出の後、またはユーザの指示に従ってハードウェア化指定を行った後、第２のアクティビティ図を生成し、表示する。図１４は、本発明に係るＧＵＩ部により表示される第２のアクティビティ図の一例を示す図である。この例では、関数Ｄと関数Ｆがハードウェア化候補として抽出された、またはハードウェア化関数として指定された場合を表している。ハードウェア化による関数Ｄの処理サイクル時間は１００に指定されている。制御カードまたはユーザによる処理サイクル時間の指定は、予測値に基づいても良いし、仕様に基づいても良い。更に、第２のアクティビティ図の最上部には、第１のアクティビティ図と同様に図全体の処理サイクル時間が表示されても良いし、第１のアクティビティ図の処理全体の処理サイクル時間と第２のアクティビティ図の処理全体の処理サイクル時間の差が表示されても良い。

【００６９】

なお、本実施の形態において、ＧＵＩ部１４は、アクティビティ図を表示するとしたが、並列処理を表すことができる他の図を表示しても良い。

【００７０】

アクティビティ図上に関数毎の処理サイクル時間と並列動作可能な関数を表示し、更にハードウェア化を行った場合の処理サイクル時間を表示することにより、適切なハードウェアとソフトウェアの分割を行うことができる。

【００７１】

次に、ハードウェア化変換処理について説明する。

【００７２】

ハードウェア化変換処理とは、ハードウェア化関数をハードウェアブロックに変換し、これに伴って、ファームウェアからハードウェア化関数を削除し、ファームウェアにハードウェアブロックとの接続を追加するものである。具体的には、モジュール定義、接続追加、呼び出し元関数変更を行う。

【００７３】

モジュール定義とは、予め用意されたスケルトンを用いて、ハードウェア化のためのラッパーを生成するものであり、予めスケルトンが用意される。このスケルトンには、関数定義のヘッダファイル名、モジュール宣言のためのハードウェア化関数名、ハードウェア化指定において指定された処理サイクル数、呼び出しのためのハードウェア化関数名、等の可変部分が存在し、これらの可変部分がハードウェア化指定に従って実際の値に置き換えられる。このモジュールは、カウンタを初期化し、ハードウェア化関数のステータスレジスタにｂｕｓｙになるとハードウェア化関数を実行し、クロック変化の度にカウンタを増加させ、カウンタが指定された処理サイクル数になるまでウェイトを行い、カウンタが指定された処理サイクル数になったらハードウェア化関数のステータスレジスタをｒｅａｄｙにするものである。

【００７４】

接続追加の関数には、上述したモジュールを宣言するためのモジュール宣言部分と上述したモジュールを接続するための接続部分が存在する。モジュール宣言部分に同名のモジュールがない場合、モジュールの宣言として“モジュール名　関数名”を追加する。また

、接続部分に同名の関数がない場合、接続のための関数名とポート名を追加する。

【００７５】

呼び出し元関数変更では、まず、ハードウェア化関数の呼び出し元関数において、ハードウェア化関数の呼び出しをコメントアウトすることにより、無効化する。また、ハードウェア化関数の呼び出し元関数において、ハードウェア化関数のステータスレジスタにｂｕｓｙを書き込み、ハードウェア化関数に対応するハードウェアブロックによりｒｅａｄｙになるまで待つポーリングの動作を追加する。このポーリングの動作により、ハードウェア化関数の呼び出し元関数は、ハードウェアブロックの処理中に待機するようになる。

【００７６】

図１５は、本発明に係るハードウェアとソフトウェアの分割後のファームウェアと共通領域の構成の一例を示す図である。この例のファームウェアは、性能測定用ファームウェアにおける関数Ｄと関数Ｆをハードウェア化したものであり、ハードウェア化のためのラッパーからハードウェア化された関数Ｄと関数Ｆを呼び出す形に変換されている。

【００７７】

更に、回路設計装置を構成するコンピュータにおいて上述した各ステップを実行させるプログラムを、回路設計プログラムとして提供することができる。上述したプログラムは、コンピュータにより読取り可能な記録媒体に記憶させることによって、回路設計装置を構成するコンピュータに実行させることが可能となる。ここで、上記コンピュータにより読取り可能な記録媒体としては、ＲＯＭやＲＡＭ等のコンピュータに内部実装される内部記憶装置、ＣＤ－ＲＯＭやフレキシブルディスク、ＤＶＤディスク、光磁気ディスク、ＩＣカード等の可搬型記憶媒体や、コンピュータプログラムを保持するデータベース、或いは、他のコンピュータ並びにそのデータベースや、更に回線上の伝送媒体をも含むものである。

【００７８】

なお、ハードウェア化指定部とは、実施の形態におけるハードウェア化候補抽出部、ＧＵＩ部に対応する。ハードウェア化指定ステップとは、実施の形態におけるハードウェア化候補抽出部による処理、ＧＵＩ部による処理に対応する。

【００７９】

（付記１）　ハードウェアと該ハードウェア上で実行されるファームウェアで構成される回路について、システム記述言語を用いた設計をコンピュータに実行させる回路設計プログラムであって、

前記回路の動作を表す第１のファームウェアのシミュレーションを行い、該シミュレーション実行中の関数の識別子と時刻を含むログを出力するシミュレーションステップと、

前記ログと並列動作可能な関数の記述のルールを定めたファームウェア記述ルールとに基づいて、前記第１のファームウェアの関数毎の処理時間を算出すると共に、前記第１のファームウェアから並列動作可能な関数を抽出する性能解析ステップと、

前記並列動作可能な関数のうち、実装時にハードウェアに変換する関数であるハードウェア化関数と該ハードウェア化関数の処理時間の指定を行うハードウェア化指定ステップと、

前記第１のファームウェアを、前記ハードウェア化関数に対応するハードウェアブロックに接続する第２のファームウェアに変換するハードウェア化変換ステップと、

をコンピュータに実行させる回路設計プログラム。

（付記２）　付記１に記載の回路設計プログラムにおいて、

前記ハードウェア化指定ステップは、アクティビティ図上に前記関数毎の処理時間と前記並列動作可能な関数を表示することを特徴とする回路設計プログラム。

（付記３）　付記１または付記２に記載の回路設計プログラムにおいて、

前記ハードウェア化変換ステップは、前記第１のファームウェアから前記ハードウェア化関数の呼び出しを削除すると共に、前記ハードウェアブロックとの接続を追加することにより、前記第２のファームウェアに変換することを特徴とする回路設計プログラム。

（付記４）　付記１乃至付記３のいずれかに記載の回路設計プログラムにおいて、

前記第１のファームウェアは、実行中の関数の識別子と時刻を含むログを出力する機能を含むことを特徴とする回路設計プログラム。

（付記５）　付記１乃至付記４のいずれかに記載の回路設計プログラムにおいて、

前記ハードウェア化指定ステップは、前記並列動作可能な関数のうち予め設定された処理時間の基準に適するものを前記ハードウェア化関数として指定し、該ハードウェア化関数の処理時間を予め設定された値に指定することを特徴とする回路設計プログラム。

（付記６）　付記１乃至付記４のいずれかに記載の回路設計プログラムにおいて、

前記ハードウェア化指定ステップは、前記並列動作可能な関数のうち予め設定された処理時間の基準に適するものを前記ハードウェア化関数の候補として表示し、ユーザの操作に従ってハードウェア化関数と該ハードウェア化関数の処理時間を指定することを特徴とする回路設計プログラム。

（付記７）　付記１乃至付記４のいずれかに記載の回路設計プログラムにおいて、

前記ハードウェア化指定ステップは、前記並列動作可能な関数を表示し、前記表示ユーザの操作に従ってハードウェア化関数と該ハードウェア化関数の処理時間を指定することを特徴とする回路設計プログラム。

（付記８）　ハードウェアと該ハードウェア上で実行されるファームウェアで構成される回路について、システム記述言語を用いた設計を行う回路設計装置であって、

前記回路の動作を表す第１のファームウェアのシミュレーションを行い、該シミュレーション実行中の関数の識別子と時刻を含むログを出力するシミュレーション部と、

前記ログと並列動作可能な関数の記述のルールを定めたファームウェア記述ルールとに基づいて、前記第１のファームウェアの関数毎の処理時間を算出すると共に、前記第１のファームウェアから並列動作可能な関数を抽出する性能解析部と、

前記並列動作可能な関数のうち、実装時にハードウェアに変換する関数であるハードウェア化関数と該ハードウェア化関数の処理時間の指定を行うハードウェア化指定部と、

前記第１のファームウェアを、前記ハードウェア化関数に対応するハードウェアブロックに接続する第２のファームウェアに変換するハードウェア化変換部と、

を備えてなる回路設計装置。

（付記９）　付記８に記載の回路設計装置において、

前記ハードウェア化指定部は、アクティビティ図上に前記関数毎の処理時間と前記並列動作可能な関数を表示することを特徴とする回路設計装置。

（付記１０）　付記８または付記９に記載の回路設計装置において、

前記ハードウェア化変換部は、前記第１のファームウェアから前記ハードウェア化関数の呼び出しを削除すると共に、前記ハードウェアブロックとの接続を追加することにより、前記第２のファームウェアに変換することを特徴とする回路設計装置。

（付記１１）　付記８乃至付記１０のいずれかに記載の回路設計装置において、

前記第１のファームウェアは、実行中の関数の識別子と時刻を含むログを出力する機能を含むことを特徴とする回路設計装置。

（付記１２）　付記８乃至付記１１のいずれかに記載の回路設計装置において、

前記ハードウェア化指定部は、前記並列動作可能な関数のうち予め設定された処理時間の基準に適するものを前記ハードウェア化関数として指定し、該ハードウェア化関数の処理時間を予め設定された値に指定することを特徴とする回路設計装置。

（付記１３）　付記８乃至付記１１のいずれかに記載の回路設計装置において、

前記ハードウェア化指定部は、前記並列動作可能な関数のうち予め設定された処理時間の基準に適するものを前記ハードウェア化関数の候補として表示し、ユーザの操作に従ってハードウェア化関数と該ハードウェア化関数の処理時間を指定することを特徴とする回路設計装置。

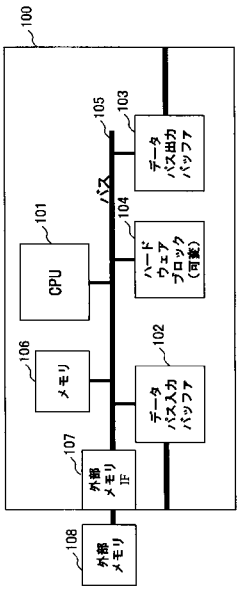
（付記１４）　付記８乃至付記１１のいずれかに記載の回路設計装置において、

前記ハードウェア化指定部は、前記並列動作可能な関数を表示し、前記表示ユーザの操作に従ってハードウェア化関数と該ハードウェア化関数の処理時間を指定することを特徴とする回路設計装置。

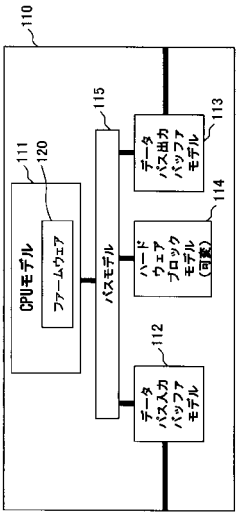
（付記１５）　ハードウェアと該ハードウェア上で実行されるファームウェアで構成される回路について、システム記述言語を用いた設計を実行する回路設計方法であって、
　前記回路の動作を表す第１のファームウェアのシミュレーションを行い、該シミュレーション実行中の関数の識別子と時刻を含むログを出力するシミュレーションステップと、
　前記ログと並列動作可能な関数の記述のルールを定めたファームウェア記述ルールとに基づいて、前記第１のファームウェアの関数毎の処理時間を算出すると共に、前記第１のファームウェアから並列動作可能な関数を抽出する性能解析ステップと、
　前記並列動作可能な関数のうち、実装時にハードウェアに変換する関数であるハードウェア化関数と該ハードウェア化関数の処理時間の指定を行うハードウェア化指定ステップと、
　前記第１のファームウェアを、前記ハードウェア化関数に対応するハードウェアブロックに接続する第２のファームウェアに変換するハードウェア化変換ステップと、
　を実行する回路設計方法。
（付記１６）　付記１５に記載の回路設計方法において、
　前記ハードウェア化指定ステップは、アクティビティ図上に前記関数毎の処理時間と前記並列動作可能な関数を表示することを特徴とする回路設計方法。
（付記１７）　付記１５または付記１６に記載の回路設計方法において、
　前記ハードウェア化変換ステップは、前記第１のファームウェアから前記ハードウェア化関数の呼び出しを削除すると共に、前記ハードウェアブロックとの接続を追加することにより、前記第２のファームウェアに変換することを特徴とする回路設計方法。
（付記１８）　付記１５乃至付記１７のいずれかに記載の回路設計方法において、
　前記第１のファームウェアは、実行中の関数の識別子と時刻を含むログを出力する機能を含むことを特徴とする回路設計方法。
（付記１９）　付記１５乃至付記１８のいずれかに記載の回路設計方法において、
　前記ハードウェア化指定ステップは、前記並列動作可能な関数のうち予め設定された処理時間の基準に適するものを前記ハードウェア化関数として指定し、該ハードウェア化関数の処理時間を予め設定された値に指定することを特徴とする回路設計方法。
（付記２０）　付記１５乃至付記１８のいずれかに記載の回路設計方法において、
　前記ハードウェア化指定ステップは、前記並列動作可能な関数のうち予め設定された処理時間の基準に適するものを前記ハードウェア化関数の候補として表示し、ユーザの操作に従ってハードウェア化関数と該ハードウェア化関数の処理時間を指定することを特徴とする回路設計方法。
【図面の簡単な説明】
【００８０】
【図１】本発明に係る回路設計装置が対象とするＬＳＩのアーキテクチャの一例を示すブロック図である。
【図２】本発明に係る回路設計装置が対象とするＬＳＩのリファレンスモデルの一例を示すブロック図である。
【図３】本発明に係る回路設計装置の構成の一例を示すブロック図である。
【図４】本発明に係る回路設計装置の動作の一例を示すフローチャートである。
【図５】本発明に係るハードウェアとソフトウェアの分割処理の動作の一例を示すフローチャートである。
【図６】本発明に係るハードウェアとソフトウェアの分割前のファームウェアと共通領域の構成の一例を示す図である。
【図７】本発明に係るファームウェアにおける並列動作が可能な関数の一例を示すソースコードである。
【図８】本発明に係るファームウェアにおける並列動作が不可能な関数の一例を示すソースコードである。
【図９】本発明に係るログと構成解析の結果の一例を示す図である。
【図１０】本発明に係るログと時間解析の結果の一例を示す図である。

【図１１】本発明に係る制御カードの一例を示す図である。
【図１２】本発明に係る制御カード中のキーワードと設定値の説明の一例を示す図である。
【図１３】本発明に係るＧＵＩ部により表示される第１のアクティビティ図の一例を示す図である。
【図１４】本発明に係るＧＵＩ部により表示される第２のアクティビティ図の一例を示す図である。
【図１５】本発明に係るハードウェアとソフトウェアの分割後のファームウェアと共通領域の構成の一例を示す図である。
【符号の説明】
【００８１】
１１　シミュレーション部、１２　性能解析部、１３　ハードウェア化候補抽出部、１４　ＧＵＩ部、１５　ハードウェア化変換部、１６　設計データ出力部、２１　制御カード記憶部。

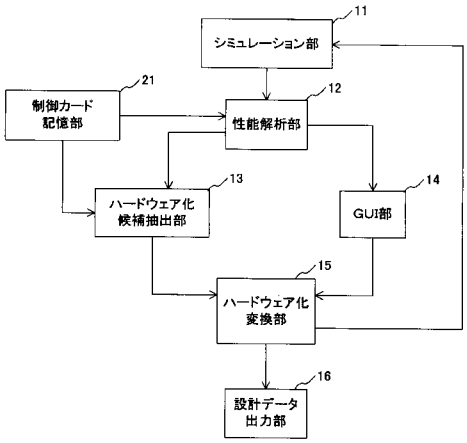
【図１】



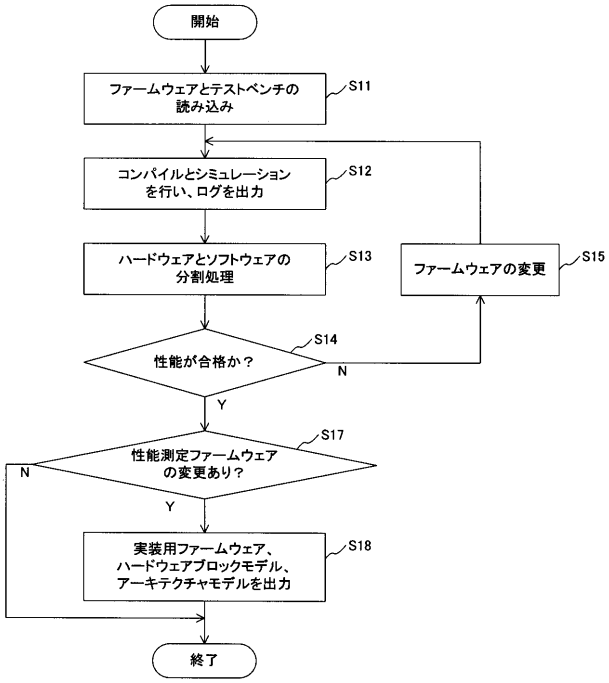
【図２】



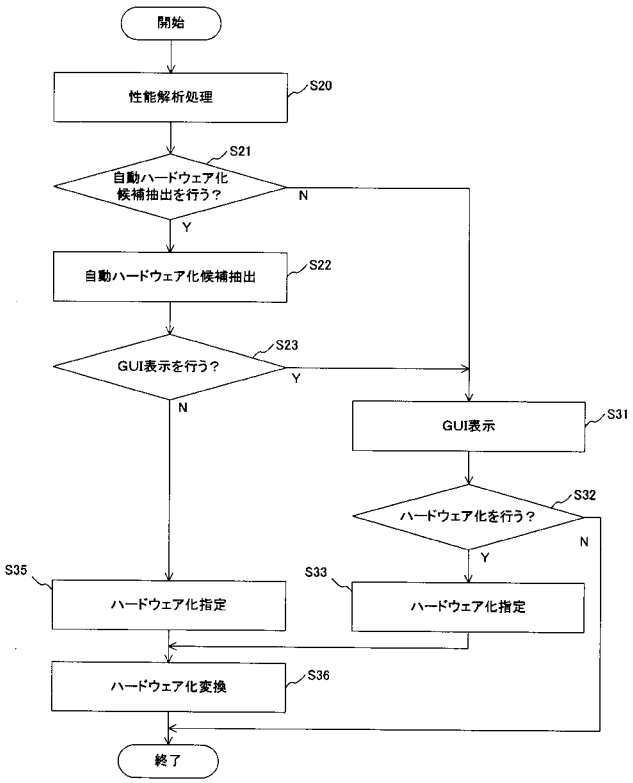
【図３】


11
シミュレーション部
21
制御カード記憶部
12
性能解析部
13
ハードウェア化候補抽出部
14
GUI部
15
ハードウェア化変換部
16
設計データ出力部

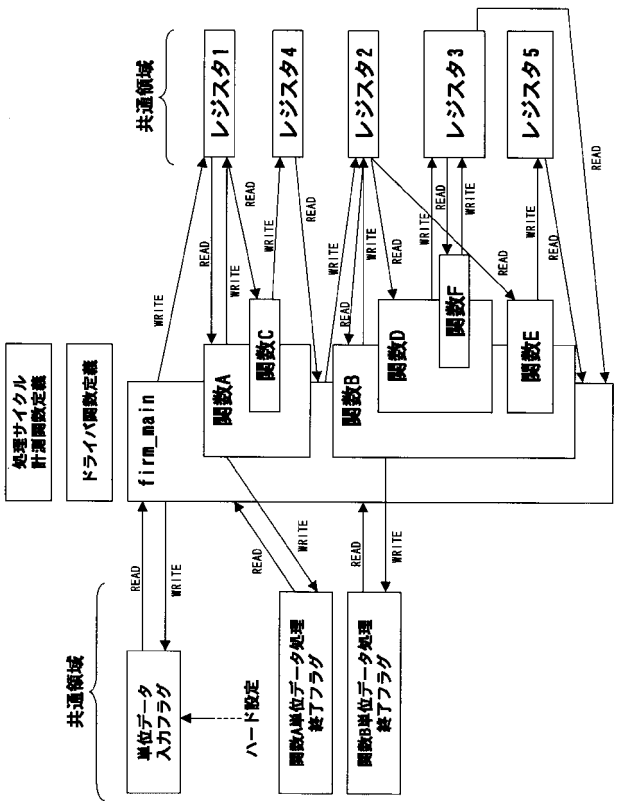
【図４】


開始
ファームウェアとテストベンチの読み込み
S11
コンパイルとシミュレーションを行い、ログを出力
S12
ハードウェアとソフトウェアの分割処理
S13
ファームウェアの変更
S15
性能が合格か？
S14
N
Y
性能測定ファームウェアの変更あり？
S17
N
Y
実装用ファームウェア、ハードウェアブロックモデル、アーキテクチャモデルを出力
S18
終了

【図５】


開始
性能解析処理
S20
自動ハードウェア化候補抽出を行う？
S21
N
Y
自動ハードウェア化候補抽出
S22
GUI表示を行う？
S23
Y
N
GUI表示
S31
ハードウェア化を行う？
S32
N
Y
S35
ハードウェア化指定
S33
ハードウェア化指定
ハードウェア化変換
S36
終了

【図６】


共通領域
レジスタ1
レジスタ4
レジスタ2
レジスタ3
レジスタ5
処理サイクル計測関数定義
ドライバ関数定義
firm_main
関数A
関数C
関数B
関数D
関数F
関数E
READ
WRITE
共通領域
単位データ入力フラグ
ハード設定
関数A単位データ処理終了フラグ
関数B単位データ処理終了フラグ

【図７】

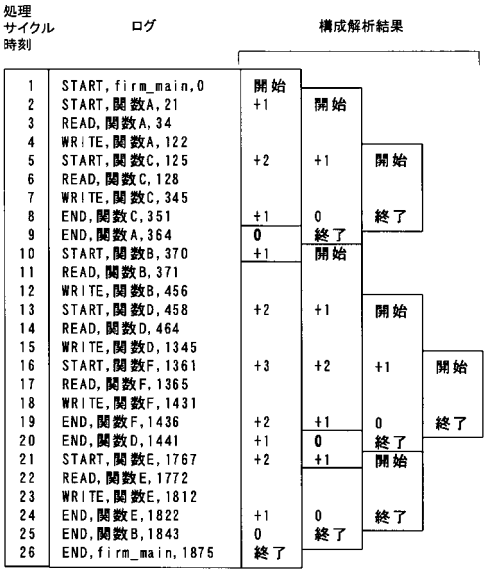
```
関数A() {
  関数B();
    :
  関数C();
}
```

【図８】

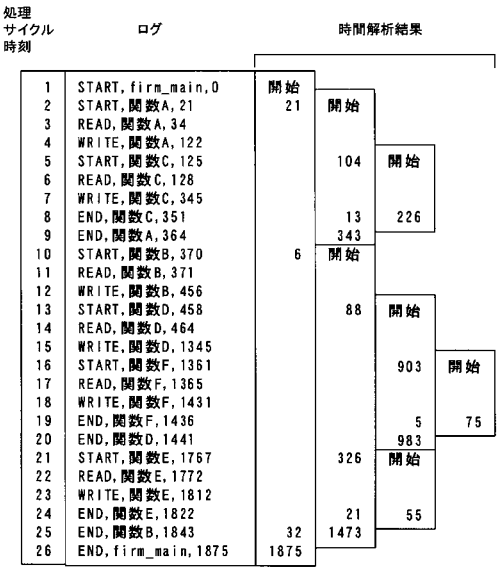
```
関数A() {
  関数B();
    :
}

関数B() {
    :
  関数C();
}
```

【図９】



【図１０】



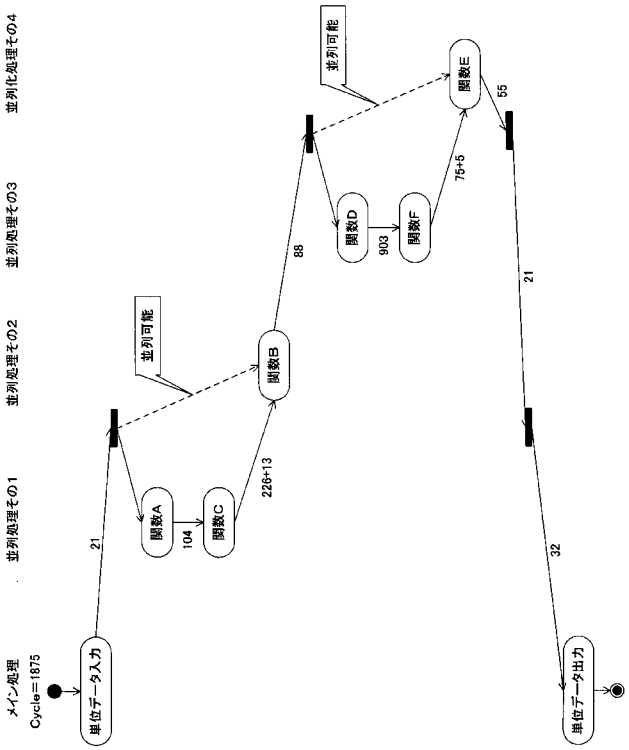
【図１１】

```
AUTO_SELECT 1
METHOD_TYPE 1
AVE_THRESHOLD 200
ABS_THRESHOLD 150
NOT_TARGETED 関数A,関数B
VALUE 関数A,100
VALUE 関数B,200
```

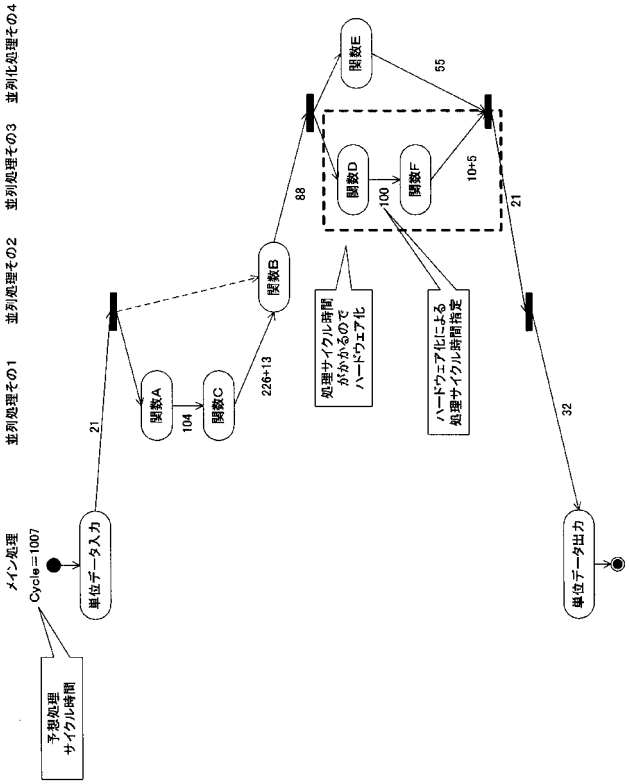
【図１２】

キーワード	内容	値の説明	
AUTO_SELECT	自動実行設定	0	この制御カードは利用しない
		1	ハード選択のみ行い GUI を表示する
		2	GUI を表示せずハード化まで自動で行う
METHOD_TYPE	抽出方法設定	0	平均値でハード化候補を抽出する
		1	絶対値でハード化候補を抽出する
AVE_THRESHOLD	平均値における ハード化閾値	平均値に対する割合をパーセントで指定。指定されたサイクル数以上の関数がハード化候補	
ABS_THRESHOLD	絶対値のハード化閾値	ハード化候補にするサイクル数を指定。 指定されたサイクル数以上の関数がハード化候補	
NOT_TARGETED	ハード化対象外の関数を指定。 AUTO_SELECT=2 の場合で、ここで定義されている関数がハード化候補になってしまった場合、AUTO_SELECT=1 相当の動作を強制的に行う。	関数名を指定。カンマ区切りで複数指定可能	
VALUE	AUTO_SELECT1 以上の場合に利用する。ハード化候補になった場合に指定するサイクル数を指定。 ここで値が定義されていない関数がハード化候補になってしまった場合は、AUTO_SELECT=1 相当の動作を強制的に行う。		

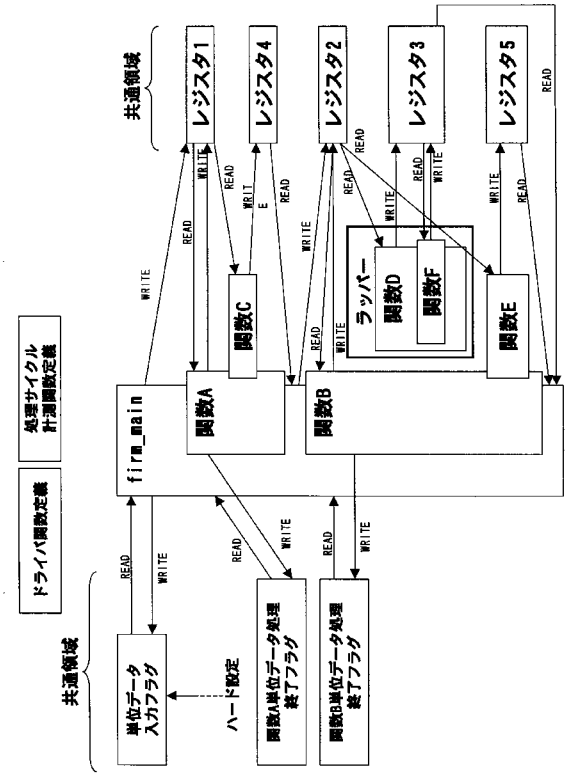
【図１３】


並列処理その1
並列処理その2
並列処理その3
並列化処理その4
メイン処理
Cycle＝1875
単位データ入力
関数A
関数C
関数B
関数D
関数F
関数E
並列可能
並列可能
21
104
226+13
88
903
75+5
55
21
32
単位データ出力

【図１４】


並列処理その1
並列処理その2
並列処理その3
並列化処理その4
メイン処理
Cycle＝1007
単位データ入力
関数A
関数C
関数B
関数D
関数F
関数E
予想処理
サイクル時間
21
104
226+13
88
100
10+5
55
21
32
処理サイクル時間
がかかるので
ハードウェア化
ハードウェア化による
処理サイクル時間指定
単位データ出力

【図１５】


共通領域
レジスタ1
レジスタ4
レジスタ2
レジスタ3
レジスタ5
ドライバ関数定義
処理サイクル
計測関数定義
fir_main
関数A
関数C
関数B
ラッパー
関数D
関数F
関数E
READ
WRITE
単位データ
入力フラグ
ハード割定
関数A単位データ処理
終了フラグ
関数B単位データ処理
終了フラグ
共通領域