(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第5475996号
(P5475996)
(45)発行日　平成26年4月16日(2014.4.16)　(24)登録日　平成26年2月14日(2014.2.14)

(51)Int.Cl.　　F I
G06F 17/50 (2006.01)　　G06F 17/50 664K
G06F 9/455 (2006.01)　　G06F 17/50 664L
　　G06F 9/44 310D

請求項の数 18　(全 30 頁)

(21)出願番号　特願2008-550846 (P2008-550846)
(86)(22)出願日　平成19年1月18日 (2007.1.18)
(65)公表番号　特表2009-524138 (P2009-524138A)
(43)公表日　平成21年6月25日 (2009.6.25)
(86)国際出願番号　PCT/GB2007/000168
(87)国際公開番号　WO2007/083134
(87)国際公開日　平成19年7月26日 (2007.7.26)
審査請求日　平成22年1月18日 (2010.1.18)
(31)優先権主張番号　0601135.7
(32)優先日　平成18年1月20日 (2006.1.20)
(33)優先権主張国　英国 (GB)

前置審査

(73)特許権者　505330125
メンター　グラフィックス　コーポレイション
アメリカ合衆国，オレゴン州　９７０７０－７７７７，ウィルソンヴィル，サウスウエスト　ベックマン　ロード　８００５番地
(74)代理人　100087701
弁理士　稲岡　耕作
(74)代理人　100101328
弁理士　川崎　実夫
(74)代理人　100110799
弁理士　丸山　温道

最終頁に続く

(54)【発明の名称】モデル化およびシミュレーション方法

(57)【特許請求の範囲】
【請求項１】
プログラム・メモリ内に記憶されているプロセッサ読取可能命令を読み取って実行するように構成したプロセッサを用いて、システムをモデル化およびシミュレーションする方法であって、前記システムはコンポーネントを含み、
（ａ）前記プロセッサが、前記コンポーネントの挙動を、第１コンピュータ・プログラミング言語で書かれた第１コンピュータ・プログラムに記述された仕様を使用してモデル化する工程と、
（ｂ）前記プロセッサが、前記第１コンピュータ・プログラムを処理して第２コンピュータ・プログラムを生成する工程と、
（ｃ）前記プロセッサが、前記第２コンピュータ・プログラムに基づいて、前記コンポーネントの実装を生成する工程と、
（ｄ）前記プロセッサが、エンティティの階層内の第１エンティティをインスタンス化することによって前記生成した実装を使用して、前記コンポーネントの挙動をシミュレーションする工程とを含み、
前記第２コンピュータ・プログラムが、前記シミュレーションに応答して、前記エンティティの階層内のより高いレベルのエンティティおよびより低いレベルのエンティティ双方をインスタンス化するように構成した有限状態機械の実装を含むことを特徴とする方法。
【請求項２】

前記システムが、それぞれの仕様が機能仕様および関連シミュレーション・エレメントを含む、相関のある少なくとも第１および第２コンポーネントを含み、前記シミュレーション・エレメントが互いに通信してシミュレーション・システムを提供することを特徴とする、請求項１に記載の方法。

【請求項３】

前記シミュレーション・システムが前記有限状態機械を実装することを特徴とする、請求項２に記載の方法。

【請求項４】

前記シミュレーション・システムが、前記第１コンポーネントおよび前記第２コンポーネントを表す前記第１仕様と前記第２仕様との通信を管理することを特徴とする、請求項２または請求項３に記載の方法。

【請求項５】

前記シミュレーション・システムが、前記第１コンポーネントおよび前記第２コンポーネントがイベントを処理する順序を決定することを特徴とする、請求項２～請求項４のいずれか１項に記載の方法。

【請求項６】

前記少なくとも１つのインスタンス化した第１エンティティに応答して、少なくとも１つのさらなるエンティティをインスタンス化する工程をさらに含み、前記少なくとも１つのさらなるエンティティは第１機能仕様および第２機能仕様の少なくとも１つによって定義され、前記シミュレーションに応答してインスタンス化される前記少なくとも１つのさらなるエンティティは、前記第１エンティティとの階層関係に基づいて、前記シミュレーション・システムによって選択される、請求項２～請求項５のいずれか１項に記載の方法。

【請求項７】

前記第１コンポーネントおよび前記第２コンポーネントを、前記相関エンティティの階層内におけるそれぞれのエンティティによって表し、前記それぞれのエンティティのインスタンスを作成することを特徴とする、請求項６に記載の方法。

【請求項８】

前記少なくとも１つのさらなるエンティティが、前記相関エンティティの階層において、前記少なくとも１つの第１エンティティの親であることを特徴とする、請求項６または請求項７に記載の方法。

【請求項９】

前記少なくとも１つのさらなるエンティティが、前記相関エンティティの階層において、前記少なくとも１つの第１エンティティの子であることを特徴とする、請求項６～請求項８のいずれか１項に記載の方法。

【請求項１０】

前記所定の階層内のエンティティは、前記第１コンポーネントおよび前記第２コンポーネントを表す前記第１仕様と前記第２仕様との間で送信されるデータを表すようにインスタンス化され、前記第１コンポーネントと前記第２コンポーネントとの通信をモデル化することを特徴とする、請求項６～請求項９のいずれか１項に記載の方法。

【請求項１１】

前記第１コンポーネントを前記第２コンポーネントよりも低い抽象化レベルでモデル化し、当該方法が前記第１コンポーネントと前記第２コンポーネントとの関係の詳細を、前記階層を使って提供する工程を含むことを特徴とする、請求項６～請求項１０のいずれか１項に記載の方法。

【請求項１２】

前記エンティティ間の階層関係を、前記第１機能仕様および前記第２機能仕様のうちの少なくとも１つから導き出すことを特徴とする、請求項６～請求項１１のいずれか１項に記載の方法。

【請求項１３】

前記処理する工程が、前記第１コンピュータ・プログラムを、前記第１コンピュータ・プログラミング言語から第２コンピュータ・プログラミング言語に変換することを特徴とする、請求項１～請求項１２のいずれか１項に記載の方法。

【請求項１４】

前記第２コンピュータ・プログラミング言語がＶＨＤＬまたはＶｅｒｉｌｏｇであることを特徴とする請求項１３に記載の方法。

【請求項１５】

前記第２コンピュータ・プログラムに基づいてハードウェアに前記第１コンポーネントおよび前記第２コンポーネントの実装を生成する工程をさらに含む、請求項１～請求項１４のいずれか１項に記載の方法。

【請求項１６】

請求項２に従属することを前提として、前記処理する工程が、前記第２コンピュータ・プログラムに、追加のコンピュータ・コードを挿入する工程を含み、前記追加のコンピュータ・コードが前記少なくとも１つのシミュレーション・エレメントを実装することを特徴とする、請求項１３～請求項１５のいずれか１項に記載の方法。

【請求項１７】

プロセッサ読取可能命令を記憶するプログラム・メモリと、前記プログラム・メモリ内に記憶されている命令を読み取って実行するように構成したプロセッサとを含み、

前記プロセッサ読取可能命令が、前記プロセッサを請求項１～請求項１６のいずれか１項の方法を行うように制御する命令を含むことを特徴とする、システムをモデル化およびシミュレーションするコンピュータ装置。

【請求項１８】

コンポーネントを含むシステムをモデル化およびシミュレーションする装置であって、当該装置が、

前記コンポーネントの挙動を、第１コンピュータ・プログラミング言語で書かれた第１コンピュータ・プログラムに記述された仕様を使用して、モデル化するモデル化手段と、第２コンピュータ・プログラムを生成するように前記第１コンピュータ・プログラムを処理する処理手段と、

前記第２コンピュータ・プログラムに基づいて、前記コンポーネントの実装を生成する生成手段と、

前記コンポーネントの挙動を、エンティティの階層内の第１エンティティをインスタンス化することによって、前記生成した実装を使って、シミュレーションするシミュレーション手段とを含み、

前記第２コンピュータ・プログラムが、前記シミュレーションに応答して、前記エンティティの階層内のより高いレベルのエンティティおよびより低いレベルのエンティティ双方をインスタンス化するように構成した有限状態機械の実装を含むことを特徴とする、装置。

【発明の詳細な説明】

【技術分野】

【０００１】

本発明は、モデル化およびシミュレーション方法に関係する。

【背景技術】

【０００２】

システムのモデルを構築して、予め定義された条件下で該モデルの挙動を実証するためにシミュレーション・プロセスを使用することは周知である。システムは２以上の相関コンポーネントを含んでもよい。多くのモデル化の方法がよく知られており、選択された特定の方法は、通常モデル化するシステムの要件によって判断する。

モデル化およびシミュレーション方法のある具体的な用途は、プロトタイプシステムを製作する前に、コンピュータのハードウェア・システムをモデル化およびシミュレートすることである。これは、プロトタイプ・コンピュータのハードウェア・システムの開発に

関わる時間と費用の点で有利である。このようなモデル化およびシミュレーションは、最終設計に到る前に、必要な設計イタレーション（繰り返し）の回数を実質的に減らすことができる。さらに、モデル化およびシミュレーションは、設計期間の初期に問題を突き止められるため、最終システムが意図したとおりに動作する確率を高めることができる。モデル化およびシミュレーション・プロセスで、モデル化したシステムが必要な挙動を示すことが一度実証されたら、コンピュータのハードウェア設計者は、検証可能な形でモデルと同じシステムの実装を構築すれば所望の結果が得られるという確信がもてる。

【０００３】

コンピュータのハードウェア・システムはその設計中に様々な抽象化レベルで考えられることは周知である。さらに、コンピュータのハードウェア・システムは、様々な抽象化レベルでモデル化およびシミュレーションをすることができる。例えば、典型的にはモデル化はシステムの仕様を英語などの自然言語で書くことから始める。それから、この仕様に基づいて、アーキテクチャ・モデルおよびパフォーマンス・モデルを作成する。このようなモデルはＣ＋＋などの高レベルのコンピュータ・プログラミング言語で作成されることが多い。アーキテクチャ・モデルおよびパフォーマンス・モデルを使用して、システムのアーキテクチャ設計の妥当性を確認し、様々な設計特徴のパフォーマンスを評価する。このモデル化の段階で、アーキテクチャおよびパフォーマンスのモデル化の結果を考慮してシステム仕様を修正してもよい。このプロセスが完了したら、システムの個々のコンポーネントのユニット仕様を自然言語で作成し、さらにＶＨＤＬまたはＶｅｒｉｌｏｇなどのハードウェア記述言語（ＨＤＬ）で書かれた仕様に変換すると、レジスタ転送レベル（ＲＴＬ）など、より低い抽象化レベルでモデル化できる。

【０００４】

前述したシステム仕様は、システム仕様から作成する適切なシステムテストを同時に行って、システムの挙動を検証する。同じくシステム仕様から作成するユニットテストを使用して、個々のコンポーネントの挙動をシミュレーション・プロセスを使って検証する。このようなシミュレーションは従来、例えばＲＴＬなど単一の抽象化レベルで行ってきた。すべてのコンポーネントに詳細ＲＴＬ設計を行ってからでないと、どのコンポーネントもシミュレーションできないため、この点は明らかに不利である。前述のシステム仕様からのユニット仕様の作成と、適切なテストの作成とは手作業で行う。このような手作業によるユニット仕様の作成は時間がかかるとともに、間違いやすい。さらに、ＲＴＬでのシステム全体のテストは、テストデータの作成の点においてもシミュレーションの点においても非常に時間がかかる。

【０００５】

上記の問題を解決しようとして、ＶＨＤＬおよびＣ＋＋などのプログラミング言語が備えるより高レベルの特徴を使って、より高い抽象化レベルでモデル化を試みてきた。しかし、より高レベルのモデルは直接ＲＴＬモデルに接続できないなど、大きな問題がでてきた。異なる抽象化レベルでモデル化したコンポーネント間でトランスレータを手作業で作成すると、多くの場合、より低いレベルのモデル自体を作成するのと少なくとも同じくらい複雑で間違いやすい作業になることがわかっている。

【０００６】

システムのコンポーネントを異なる抽象化レベルで（例えば、システムレベルおよびＲＴＬ）記述するシステムのシミュレーション・プロセスを自動化することが周知である。下記特許文献１は、システム内の相関のある第１および第２コンポーネントの挙動のシミュレーション方法を記載している。ここで記載される方法は、コンポーネントの挙動を第１機能仕様および第２機能仕様を使用してモデル化する工程と、予め定義した状況下でコンポーネントをシミュレーションする工程とを含む。モデル化するコンポーネントは異なる抽象化レベルにすることができる。

【０００７】

下記特許文献１は、コンポーネント全部をソフトウェアでモデル化するシステムを記載している。しかし、マルチレベルのモデル化では、シミュレーションの速度が最も低いレ

ベルのコンポーネントの速度に制限されてしまう。単一のコンポーネントをより高次の挙動（ビヘイビア）レベルではなく、ＲＴＬなどの低い抽象化レベルでシミュレーションするには、低レベルのモデル内に含まれている詳細の量が多くなるため、計算上の負荷がはるかに多いためである。さらに、下記特許文献１の中で記載されるような先行技術のシミュレーション・システムは、モデル化するコンポーネントすべてのシミュレーション・プロセスを管理する個別のシミュレータが必要である。このことは計算上の大きな負荷を意味するが、計算リソースに対してモデル化する個々のコンポーネントの矛盾する要求を管理するために、個別のシミュレータが必要である。

【０００８】

単一コンポーネントの低レベルＨＤＬモデルを使用して、そのモデルをハードウェアに実装するプログラマブル・コンピューティング・デバイスをプログラムすることが周知である。「プログラマブル・コンピューティング・デバイス」とは、本明細書で使用する場合、単一のプログラマブル・チップであってもよい。あるいは、プログラマブル・コンピューティング・デバイスは相互接続した複数のプログラマブル・チップを含んでもよい。さらに、プログラマブル・コンピューティング・デバイスは、１または複数のプログラマブル・チップの代わりに、またはそれに追加して、プロセッシング・エレメントを含むエミュレータ・ボックスを含んでもよい。ＨＤＬモデルは、プログラマブル・コンピューティング・デバイスの挙動を決定（判定）する。プログラマブル・コンピューティング・デバイスはシミュレーション・プロセス内で使用して、システム全体のシミュレーションを迅速化できる。しかし、これは、モデル化するシステムのコンポーネント全部が同じ低レベルでモデル化する場合に限って、効率的なプロセスであることがわかっている。それは、高レベルのソフトウェア・モデルを低レベルのハードウェア・モデルにインターフェースする際に問題が生じるためである。

【０００９】

モデル化するシステム内の一部のコンポーネントを低い抽象化レベルでハードウェアに実装し、一部のコンポーネントを高い抽象化レベルでソフトウェアに実装すると、高レベルのソフトウェア・モデルと低レベルのハードウェア・モデルとの間のトランスレータを開発する必要がある。このようなトランスレータはソフトウェアに実装してもよい。しかし、これが通常シミュレーションするシステムにおいて障害となる。障害の一因は、低い抽象化レベルでモデル化するハードウェア・コンポーネントに情報を渡す場合、ソフトウェアからハードウェアに大量の情報を渡さなければならないことである。さらに、ソフトウェアとハードウェアとが通信するときに、多くの非連続通信が発生することも障害の原因になるおそれがある。各トランザクションの開始および終了時点で、非連続通信が発生する。各トランザクションは多サイクルに渡ることがある。高レベルのソフトウェア・コンポーネントと低レベルのハードウェア・コンポーネントとの間で伝送する情報の総量が、両コンポーネントをソフトウェアに実装した場合と同じであったとしても、ソフトウェア・コンポーネントとハードウェア・コンポーネントとを同期させる負荷によって、情報交換できる速度が遅くなる。

【００１０】

ソフトウェア・コンポーネント・モデルとハードウェア・コンポーネント・モデルとのトランスレータをハードウェアに実装することが周知である。このようなトランスレータは前記または各ハードウェア・コンポーネント・モデルと同じプログラマブル・コンピューティング・デバイス内に実装する。これは、ソフトウェアとハードウェアとの間のデータ伝送を高い抽象化レベルでできるため、伝送する必要のあるデータの総量を低減できる点が有利である。また、ハードウェア・コンポーネントとソフトウェア・コンポーネントととのデータ交換の同期により生じる障害が少なくなる点も有利である。

【特許文献１】欧州特許出願公開第１，５１７，２５４号明細書

【発明の開示】

【発明が解決しようとする課題】

【００１１】

ハードウェア・トランスレータは、データを低レベルのハードウェア・モデルが必要とするより低い抽象化レベルに変換する。しかし、今までのところ、このようなハードウェア・トランスレータは手作業で実装している。これは手間がかかり、かつ間違いやすいプロセスである。システムの一部を高レベル（例、システムレベル）でソフトウェアに記述およびモデル化するとともに、システムの一部を低レベル（例、ＲＴＬ）でハードウェアに記述およびモデル化する場合に、システムを自動的にかつ正確にシミュレーションする方法は知られていない。

【００１２】

本発明の実施形態の目的は、本明細書または他で特定されるものかどうかを問わず、先行技術の問題の１または複数を排除または軽減することである。

【課題を解決するための手段】

【００１３】

本発明の第１の局面によると、相関のある第１および第２コンポーネントを含むシステムをモデル化およびシミュレーションする方法を提供し、当該方法は、前記第１コンポーネントおよび前記第２コンポーネントの挙動を、第１仕様および第２仕様を使用してモデル化する工程であって、前記第１仕様および前記第２仕様のそれぞれが機能仕様と関連シミュレーション・エレメントとを含む、挙動をモデル化する工程と、前記第１コンポーネントおよび前記第２コンポーネントの挙動を前記第１仕様および前記第２仕様を使用してシミュレーションする工程とを含み、前記シミュレーション・エレメントが互いに通信して、シミュレーション・システムを提供する。

【００１４】

本発明の第１の局面の利点は、シミュレーション・エレメントがまとまって分散シミュレーション・システムを形成し、これがハードウェア実装コンポーネントのシミュレーションを制御することである。本発明のある実施形態において、コンポーネント内にエレメントを含む分散シミュレーション・システムをシミュレーションさせることによって、ハードウェア実装エレメントのシミュレーションを制御する役割を担う個別のエンティティを有する必要がない。

【００１５】

前記シミュレーション・システムは、前記第１コンポーネントおよび前記第２コンポーネントを表す前記第１仕様と前記第２仕様との通信を管理してもよい。

前記シミュレーション・システムは、前記第１コンポーネントおよび前記第２コンポーネントを表す前記第１仕様および前記第２仕様がイベントを処理する順序を決定してもよい。

【００１６】

前記シミュレーションする工程はさらに、相関エンティティの階層内の少なくとも１つの第１エンティティをインスタンス化する工程を含んでもよい。前記少なくとも１つの第１エンティティは前記第１機能仕様および前記第２機能仕様のうちの少なくとも１つによって定義される。

前記方法はさらに、前記または各インスタンス化した第１エンティティに応答して、少なくとも１つのさらなるエンティティをインスタンス化する工程を含んでもよく、前記または各さらなるエンティティは前記第１機能仕様および前記第２機能仕様のうちの少なくとも１つによって定義され、前記または各さらなるエンティティは前記少なくとも１つの第１エンティティとの階層関係に基づいてシミュレーション・システムによって選択される。

【００１７】

前記第１コンポーネントおよび前記第２コンポーネントを、前記相関エンティティの階層内のそれぞれのエンティティによって表してもよく、前記それぞれのエンティティのインスタンスが作成される。

前記少なくとも１つのさらなるエンティティは、前記相関エンティティの階層において前記少なくとも１つの第１エンティティの親であってもよい。あるいは、前記少なくとも

１つの別のエンティティは、前記相関エンティティの階層において前記少なくとも１つの第１エンティティの子であってもよい。

【００１８】

前記階層内のエンティティは、第１コンポーネントと第２コンポーネントとの間で送信されるデータを表すインスタンスを生成してもよい。前記第１コンポーネントおよび前記第２コンポーネントを表す第１仕様と第２仕様との間の通信をモデル化する。

前記第１コンポーネントは前記第２コンポーネントよりも高い抽象化レベルでモデル化してもよく、前記方法は前記第１コンポーネントと前記第２コンポーネントとの関係の詳細を前記階層を使って規定する工程を含む。

【００１９】

前記エンティティ間の階層関係は、前記第１機能仕様および前記第２機能仕様のうちの少なくとも１つから導き出してもよい。

前記第１仕様および前記第２仕様は、第１コンピュータ・プログラミング言語で書かれた第１コンピュータ・プログラムとして記述してもよい。前記方法はさらに、前記第１仕様と前記第２仕様との関係を前記第１コンピュータ・プログラムに記述する工程を含んでもよい。

【００２０】

前記方法はさらに、第２コンピュータ・プログラムを生成するように前記第１コンピュータ・プログラムを処理する工程を含んでもよい。前記処理する工程は、前記第１コンピュータ・プログラムを前記第１コンピュータ・プログラミング言語から第２コンピュータ・プログラミング言語に変換（翻訳）してもよい。前記第２コンピュータ・プログラミング言語はＶＨＤＬまたはＶｅｒｉｌｏｇであってもよい。

【００２１】

前記方法はさらに、前記第２コンピュータ・プログラムに基づいて、ハードウェアに前記第１コンポーネントおよび前記第２コンポーネントの実装（実動化）を生成する工程を含んでもよい。

前記処理する工程は、前記第２コンピュータ・プログラムに追加のコンピュータ・コードを挿入する工程を含んでもよい。前記追加のコンピュータ・コードは前記または各シミュレーション・エレメントを実装する。

【００２２】

前記方法はさらに、第２コンピュータ・プログラムおよび第３コンピュータ・プログラムを生成するように前記第１コンピュータ・プログラムを処理する工程を含んでもよい。前記処理する工程は、前記第１コンピュータ・プログラムを前記第１コンピュータ・プログラミング言語から、それぞれ第２コンピュータ・プログラミング言語および第３コンピュータ・プログラミング言語に変換してもよい。前記第２コンピュータ・プログラミング言語はＶＨＤＬまたはＶｅｒｉｌｏｇであってもよく、前記第３コンピュータ・プログラミング言語はＣまたはＣ＋＋であってもよい。前記第１コンポーネントはハードウェアに実装してもよく、前記第２コンポーネントはソフトウェアに実装してもよい。

【００２３】

前記方法はさらに、少なくとも１つのさらなるコンポーネントの挙動を少なくとも１つのさらなる機能仕様を使用してモデル化する工程と、前記少なくとも１つのさらなる機能仕様に基づいて、ソフトウェアに前記少なくとも１つのさらなるコンポーネントの実装を生成する工程と、前記少なくとも１つのさらなるコンポーネントの挙動を前記少なくとも１つのさらなる機能仕様を使用してシミュレーションする工程とを含んでもよい。

【００２４】

前記方法はさらに、前記第１コンポーネントおよび前記第２コンポーネントと、前記少なくとも１つの別のコンポーネントとの通信を管理するために、ソフトウェア実装シミュレーション・システムを提供する工程を含んでもよい。

前記方法はさらに、前記第１コンポーネントおよび前記第２コンポーネントを表す前記第１仕様および前記第２仕様が、ソフトウェア実装シミュレーション・システムと、ハー

ドウェアに実装されている他のコンポーネントを表す他の仕様とを介して、同時にさらなるコンポーネントとの通信を試みる工程を含んでもよい。

【００２５】

前記第２コンピュータ・プログラムは、前記シミュレーションに応答して、より低い階層レベルの少なくとも１つのエンティティに基づいて、より高い階層レベルのエンティティをインスタンス化するように構成した有限状態機械の実装を含んでもよい。

前記第２コンピュータ・プログラムは、前記シミュレーションに応答して、より高い階層レベルの少なくとも１つのエンティティに基づいて、より低い階層レベルのエンティティをインスタンス化するように構成した有限状態機械の実装を含んでもよい。

【００２６】

前記第２コンピュータ・プログラムは、前記シミュレーションに応答して、それぞれより低い階層レベルまたはより高い階層レベルの少なくとも１つのエンティティに基づいて、より高い階層レベルのエンティティおよびより低い階層レベルのエンティティ双方をインスタンス化するように構成した有限状態機械の実装を含んでもよい。

本発明の第２の局面により、第１仕様および第２仕様を記憶する記憶手段であって、前記第１仕様および前記第２仕様がそれぞれ第１コンポーネントおよび第２コンポーネントの挙動をモデル化し、前記第１仕様および前記第２仕様のそれぞれが機能仕様とシミュレーション・エレメントとを含む、記憶手段と、前記第１コンポーネントおよび前記第２コンポーネントの挙動を前記第１仕様および前記第２仕様を使用してシミュレーションするように構成した処理手段とを含むプログラマブル・コンピューティング・デバイスを提供する。前記シミュレーション・エレメントは互いに通信してシミュレーション・システムを提供するように構成する。

【００２７】

本発明の第３の局面により、相関のある第１および第２コンポーネントとを含むシステムをモデル化およびシミュレーションする装置を提供する。前記装置は、前記第１コンポーネントおよび前記第２コンポーネントの挙動を第１仕様および第２仕様を使用してモデル化するモデル化手段であって、前記第１仕様および前記第２仕様のそれぞれが機能仕様と関連シミュレーション・エレメントとを含む、モデル化手段と、前記第１コンポーネントおよび前記第２コンポーネントの挙動を前記第１仕様および前記第２仕様を使用してシミュレーションするシミュレーション手段とを含み、前記シミュレーション・エレメントは互いに通信してシミュレーション・システムを提供する。

【００２８】

本発明の第４の局面により、コンポーネントを含むシステムをモデル化およびシミュレーションする方法を提供する。前記方法は、第１コンピュータ・プログラミング言語で書かれた第１コンピュータ・プログラムに記述された仕様を使用して、コンポーネントの挙動をモデル化する工程と、第２コンピュータ・プログラムを生成するように前記第１コンピュータ・プログラムを処理する工程と、前記第２コンピュータ・プログラムに基づいてコンポーネントの実装を生成する工程と、エンティティの階層内の第１エンティティをインスタンス化することによって、前記生成した実装を使用してコンポーネントの挙動をシミュレーションする工程とを含み、前記第２コンピュータ・プログラムは、前記シミュレーションに応答して、前記エンティティの階層内のより高いレベルのエンティティおよびより低いレベルのエンティティ双方をインスタンス化するように構成した有限状態機械の実装を含む。

【００２９】

本発明の第４の局面の利点は、エンティティの階層内のより高いレベルとより低いレベルのエンティティとをインスタンス化するように構成されている単一のコンポーネントを有することによって、システムのモデル化およびシミュレーションの複雑さを低減できることである。

前記システムは、各仕様が機能仕様と関連シミュレーション・エレメントとを含む相関のある少なくとも第１および第２コンポーネントとを含んでもよい。前記シミュレーショ

ン・エレメントは互いに通信してシミュレーション・システムを提供する。
【００３０】
前記シミュレーション・システムは前記有限状態機械を実装してもよい。
前記シミュレーション・システムは、前記第１コンポーネントおよび前記第２コンポーネントを表す前記第１仕様と前記第２仕様との通信を管理してもよい。
前記シミュレーション・システムは、前記第１コンポーネントおよび前記第２コンポーネントがイベントを処理する順序を決定してもよい。
【００３１】
前記方法はさらに、前記または各インスタンス化した第１エンティティに応答して、少なくとも１つのさらなるエンティティをインスタンス化する工程を含んでもよく、前記または各さらなるエンティティは第１機能仕様および第２機能仕様のうちの少なくとも１つによって定義され、前記シミュレーションに応答して、インスタンス化される前記または各さらなるエンティティは、前記第１エンティティとの階層関係に基づいて前記シミュレーション・システムによって選択される。
【００３２】
前記第１コンポーネントおよび前記第２コンポーネントは、前記相関エンティティの階層内のそれぞれのエンティティで表してもよい。前記それぞれのエンティティのインスタンスを作成する。前記少なくとも１つのさらなるエンティティは、前記相関エンティティの階層において前記少なくとも１つの第１エンティティの親であってもよい。前記少なくとも１つのさらなるエンティティは、前記相関エンティティの階層において前記少なくとも１つの第１エンティティの子であってもよい。
【００３３】
前記所定の階層内のエンティティは、前記第１コンポーネントおよび前記第２コンポーネントを表す第１仕様と第２仕様との間で送信されるデータを表すようにインスタンス化され、前記第１コンポーネントと前記第２コンポーネントとの通信をモデル化してもよい。
前記第１コンポーネントは、前記第２コンポーネントよりも高い抽象化レベルでモデル化してもよい。前記方法は前記第１コンポーネントと前記第２コンポーネントとの関係の詳細を前記階層を使用して規定する工程を含む。
【００３４】
前記エンティティ間の階層関係は、前記第１機能仕様および前記第２機能仕様のうちの少なくとも１つから導き出してもよい。
前記処理する工程は、前記第１コンピュータ・プログラムを前記第１コンピュータ・プログラミング言語から第２コンピュータ・プログラミング言語に変換してもよい。前記第２コンピュータ・プログラミング言語はＶＨＤＬまたはＶｅｒｉｌｏｇであってもよい。
【００３５】
前記方法はさらに、前記第２コンピュータ・プログラムに基づいて、ハードウェアに前記第１コンポーネントおよび前記第２コンポーネントの実装を生成する工程を含んでもよい。
前記処理する工程は、前記第２コンピュータ・プログラムに追加のコンピュータ・コードを挿入する工程を含んでもよい。前記追加のコンピュータ・コードは、前記または各シミュレーション・エレメントを実装する。
【００３６】
本発明の第５の局面により、仕様を記憶する記憶手段であって、前記仕様がコンポーネントの挙動をモデル化し、前記仕様を第１コンピュータ・プログラムに記述する記憶手段と、第２コンピュータ・プログラムを生成するように前記第１コンピュータ・プログラムを処理するように構成した第１処理手段と、前記第２コンピュータ・プログラムに基づいて前記コンポーネントの実装を生成するように構成した生成手段と、エンティティの階層内の第１エンティティをインスタンス化することによって前記生成した実装を使用して前記コンポーネントの挙動をシミュレーションするように構成した第２処理手段とを含むプ

ログラマブル・コンピューティング・デバイスを提供し、前記第２コンピュータ・プログラムが、前記シミュレーションに応答して前記エンティティの階層内のより高いレベルのエンティティとより低いレベルのエンティティの双方をインスタンス化するように構成した有限状態機械の実装を含む。

【００３７】

本発明の第６の局面により、コンポーネントを含むシステムをモデル化およびシミュレーションする装置を提供し、前記方法は、第１コンピュータ・プログラミング言語で書かれた第１コンピュータ・プログラムに記述される仕様を使用して、前記コンポーネントの挙動をモデル化するモデル化手段と、第２コンピュータ・プログラムを生成するように前記第１コンピュータ・プログラムを処理する処理手段と、前記第２コンピュータ・プログラムに基づいて前記コンポーネントの実装を生成する生成手段と、エンティティの階層内の第１エンティティをインスタンス化することによって、前記生成した実装を使用して前記コンポーネントの挙動をシミュレーションするシミュレーション手段とを含み、前記第２コンピュータ・プログラムは、前記シミュレーションに応答して、前記エンティティの階層内のより高いレベルのエンティティとより低いレベルのエンティティの双方をインスタンス化するように構成した有限状態機械の実装を含む。

【００３８】

本発明の第７の局面により、相関のある第１および第２コンポーネントとを含むシステムをモデル化およびシミュレーションする方法を提供し、前記方法は、第１コンピュータ・プログラミング言語で書かれたコンピュータ・プログラムに記述された第１仕様および第２仕様を使用して、前記第１コンポーネントおよび前記第２コンポーネントの挙動をモデル化する工程と、それぞれ第２コンピュータ・プログラミング言語および第３コンピュータ・プログラミング言語で書かれた第２コンピュータ・プログラムおよび第３コンピュータ・プログラムを生成するように、前記コンピュータ・プログラムを処理する工程と、前記第２コンピュータ・プログラムに基づいて、ハードウェアに前記第１コンポーネントの実装を生成する工程と、前記第３コンピュータ・プログラムに基づいてソフトウェアに前記第２コンポーネントの実装を生成する工程と、前記生成した第１コンポーネントの実装と前記生成した第２コンポーネントの実装とをそれぞれ使用して、前記第１コンポーネントおよび前記第２コンポーネントの挙動をシミュレーションする工程とを含む。

【００３９】

本発明の第７の局面の利点は、ハードウェアおよびソフトウェアにコンポーネントの実装を生成することによって、コンポーネントは第１プログラミング言語で書かれた単一のコンピュータ・プログラムに記述され、第１コンピュータ・プログラムのシミュレーションの速度を高めることができることである。システムのあるコンポーネントをハードウェアに実装することによって、これらのコンポーネントはソフトウェアに実装されているシステムの残りとは個別に記述する必要がなくなるため、シミュレーションの速度が高まるからである。

【００４０】

前記第１仕様は、機能仕様と関連シミュレーション・エレメントとを含んでもよい。前記シミュレーション・エレメントは他のシミュレーション・エレメントと通信してシミュレーション・システムを提供するように配列し、前記方法はさらに、前記第２コンポーネントおよび前記少なくとも１つのさらなるコンポーネントを表す仕様間の通信を管理するソフトウェア実装シミュレーション・システムを提供する工程を含む。

【００４１】

前記シミュレーション・システムは、前記第１コンポーネントおよび前記第２コンポーネントを表す前記第１仕様と前記第２仕様との通信を管理してもよい。

前記シミュレーション・システムは、前記第１コンポーネントおよび前記第２コンポーネントを表す前記第１仕様および前記第２仕様がイベントを処理する順序を決定してもよい。

【００４２】

前記シミュレーションする工程はさらに、相関エンティティの階層内の少なくとも１つの第１エンティティをインスタンス化する工程を含んでもよく、前記少なくとも１つの第１エンティティは前記第１機能仕様および前記第２機能仕様のうちの少なくとも１つによって定義される。

前記方法はさらに、前記または各インスタンス化した第１エンティティに応答して、少なくとも１つのさらなるエンティティをインスタンス化する工程を含んでもよく、前記または各さらなるエンティティは前記第１機能仕様および前記第２機能仕様のうちの少なくとも１つによって定義され、前記または各さらなるエンティティは前記少なくとも１つの第１エンティティとの階層関係に基づいて前記シミュレーション・システムによって選択される。

【００４３】

前記第１コンポーネントおよび前記第２コンポーネントは前記相関エンティティの階層内のそれぞれのエンティティによって表してもよく、前記それぞれのエンティティのインスタンスを作成する。

前記少なくとも１つのさらなるエンティティは、前記相関エンティティの階層内の前記少なくとも１つの第１エンティティの親であってもよい。あるいは、前記少なくとも１つのさらなるエンティティは、前記相関エンティティの階層内の前記少なくとも１つの第１エンティティの子であってもよい。

【００４４】

前記階層内のエンティティは、前記第１コンポーネントおよび前記第２コンポーネントを表す前記第１仕様と前記第２仕様との間で送信されるデータを表すようにインスタンス化されてもよく、前記第１コンポーネントと前記第２コンポーネントとの通信をモデル化する。

前記第１コンポーネントは、前記第２コンポーネントよりも高い抽象化レベルでモデル化してもよい。前記方法は、前記第１コンポーネントと前記第２コンポーネントとの関係の詳細を前記階層を使用して提供する工程を含む。

【００４５】

前記エンティティ間の階層関係は、前記第１仕様および前記第２仕様のうちの少なくとも１つから導き出してもよい。

前記処理する工程は、前記第２コンピュータ・プログラムおよび前記第３コンピュータ・プログラムに追加のコンピュータ・コードを挿入する工程を含んでもよい。前記追加のコンピュータ・コードは、前記または各シミュレーション・エレメントを実装する。

【００４６】

前記第２プログラミング言語はＶＨＤＬまたはＶｅｒｉｌｏｇであってもよく、前記第３プログラミング言語はＣまたはＣ＋＋であってもよい。

本発明の第８の局面により、第１仕様および第２仕様を記憶する記憶手段であって、前記第１仕様および前記第２仕様がそれぞれ第１コンポーネントおよび第２コンポーネントの挙動をモデル化し、前記第１仕様および前記第２仕様は、第１コンピュータ・プログラミング言語で書かれたコンピュータ・プログラムに記述されている記憶手段と、それぞれ第２コンピュータ・プログラミング言語および第３コンピュータ・プログラミング言語で書かれた第２コンピュータ・プログラムおよび第３コンピュータ・プログラムを生成するように、前記コンピュータ・プログラムを処理するために構成した処理手段と、前記第２コンピュータ・プログラムに基づいてハードウェアに前記第１コンポーネントの実装を生成するために構成した第１生成手段と、前記第３コンピュータ・プログラムに基づいてソフトウェアに前記第２コンポーネントの実装を生成するために構成した第２生成手段と、前記生成した第１コンポーネントの実装および前記生成した第２コンポーネントの実装をそれぞれ使用して、前記第１コンポーネントおよび前記第２コンポーネントの挙動をシミュレーションするために構成したシミュレーション手段とを含むプログラマブル・コンピューティング・デバイスを提供する。

【００４７】

本発明の第９の局面により、コンポーネントを含むシステムをモデル化およびシミュレーションする装置を提供する。前記装置は、第１コンピュータ・プログラミング言語でコンピュータ・プログラムに記述されている第１仕様および第２仕様を使用して、前記第１コンポーネントおよび第２コンポーネントの挙動をモデル化するモデル化手段と、第２コンピュータ・プログラムおよび第３コンピュータ・プログラムを、それぞれ第２コンピュータ・プログラミング言語および第３コンピュータ・プログラミング言語で生成するように前記コンピュータ・プログラムを処理する処理手段と、前記第２コンピュータ・プログラムに基づいてハードウェアに前記第１コンポーネントの実装を生成する第１生成手段と、前記第３コンピュータ・プログラムに基づいてソフトウェアに前記第２コンポーネントの実装を生成する第２生成手段と、前記生成した第１コンポーネントの実装と前記生成した第２コンポーネントの実装とをそれぞれ使用して、前記第１コンポーネントおよび前記第２コンポーネントの挙動を、シミュレーションするシミュレーション手段とを含む。

【００４８】

本発明の第１０の局面により、上記方法のいずれか１つを行うようにコンピュータを制御するコンピュータ読取可能コードを伝達する伝達媒体を提供する。

本発明の第１１の局面により、システムをモデル化およびシミュレーションするコンピュータ装置を提供する前記装置は、プロセッサ読取可能命令を記憶するプログラム・メモリと、前記プログラム・メモリに記憶した命令を読み取って実行するように構成したプロセッサとを含み、前記プロセッサ読取可能命令は、上記方法のいずれか１つを行うように、前記プロセッサを制御する命令を含む。

【発明を実施するための最良の形態】

【００４９】

本明細書で使用する「ハードウェアに実装」という表現は、プログラマブル・コンピューティング・デバイスをプログラムするのにコンポーネントのＨＤＬモデルを使用することを意味するものである。

本明細書で使用する「ソフトウェアに実装」という表現は、コンピュータ装置でランするソフトウェアに、コンピュータ・プログラミング言語で書かれたコンポーネントのモデルを実装することを意味するものである。

【００５０】

ここで単なる例示として、添付の図面を参照しながら、本発明の実施形態を説明する。

図１を参照すると、先行技術から周知のモデル化プロセスが示されており、様々なシステム・コンポーネントが様々な抽象化レベルで記述されている。システム・モデル１はＣＹとして知られるプログラミング言語で相対的に高いレベルで記述されている。先行技術と本発明の実施形態はＣＹを使って記述されているが、他のプログラミング言語を同様に使用してもよいことは当業者には容易に明らかであろう。システムの他のコンポーネントは、より低い抽象化レベルで記述されている。例えば、２つのコンポーネントはそれぞれ挙動（ビヘイビア）モデル２，３としてシステムＣおよびＶＨＤＬを使って記述されている。同様に、さらなる２つのコンポーネント４，５は、それぞれＶＨＤＬおよびＶｅｒｉｌｏｇを使って、ＲＴＬレベルで記述されている。各プログラミング言語は他のプログラミング言語で代用できることは理解されよう。

【００５１】

そのため、図１は様々な抽象化レベルの多数のモデルを含んでいる。これらのモデルは同時にシミュレーションされる。モデル間のインタラクションはインターフェース６，７，８で定義され、同じくコンピュータ言語ＣＹで記述されている。

図１に示す先行技術のモデル化およびシミュレーション方法では、複数のレベルのモデルがＣＹで記述されている。そして、このＣＹモデルをシミュレーションできる形にコンパイルする。コンパイラがＣＹコードをシミュレータに渡すことのできる形のＣ＋＋に変換する。シミュレーション・プロセスを使用して、コンポーネントを様々な抽象化レベルでモデル化できる。図１のモデル化およびシミュレーション方法は、全部ソフトウェアに実装されている（ソフトウェアで実行される）。

【００５２】
前述したように、一部のコンポーネントはソフトウェアにモデル化および実装され、一部のコンポーネントはハードウェアにモデル化および実装されているため、システムの様々なコンポーネントを様々な抽象化レベルでシミュレーションできることが望ましい。
プロトコル・ツリーは、様々な抽象化レベルのオブジェクトの関わり合い方に関係する。本発明の実施形態により使用に適したプロトコル・ツリーを、プロトコル・ツリーに記述される関係も含めて図２に図示している。各オブジェクトＡ，Ｂ，Ｗ，Ｘ，Ｙ，Ｚはモデル化されているシステムのコンポーネント、コンポーネント間で渡されるデータ、コンポーネントが行う活動、または他の種類の情報を表すことができる。オブジェクトＡは２つの子、ＷおよびＸを有するように示されている。逆に、ＡはＷおよびＸの親である。オブジェクトＡは相対的に高い抽象化レベルのコンポーネントのモデルである。ＷおよびＸは相対的に低い抽象化レベルのコンポーネントのモデルである。同時に、ＷおよびＸは、コンポーネントＡを完全に定義する。すなわち、実行中において、ＷおよびＸはその親の機能の全部を備えている。同様に、コンポーネントＢは２つの子、ＹおよびＺを有する（また、ＢはＹおよびＺ双方の親である）。
【００５３】
ここで、図３を参照して、本発明の実施形態によるモデル化およびシミュレーション方法のコンパイル・プロセスを説明する。モデルの仕様３０は、プログラミング言語ＣＹで作成する。この仕様３０をコンパイラ３１に通し、コンパイラ３１はＣＹコードを少なくとも１つのさらなるプログラミング言語に変換する。さらなるプログラミング言語は、個々のコンポーネントがハードウェアまたはソフトウェアにモデル化および実装されるかどうかによって変わる。コンパイルしたコードは、本質的に５つのコンポーネントを含む。
【００５４】
第１コンポーネント３２は機能アルゴリズムを含み、モデル化するコンポーネントをどのように動作させるかを定義する。機能アルゴリズム３２はモデル化するシステム内のモデルの機能性を定義する。この機能性はＣＹソースコードで記述することができ、あるいは、ＣＹソースコードは機能性のある場所を示すヘッダーだけ規定してもよい。機能アルゴリズム３２はユニット仕様内に規定されている定義から作成する。
【００５５】
モデル形状の第２コンポーネント３３は、モデル化するシステムの形状を、モデル化するコンポーネントと、それらが互いにどのような関係にあるかとについて定義する。モデル形状３３は、どのコンポーネントが他のどのコンポーネントと、どのインターフェースによって通信するかを記述する。これはコンパイラ３１によってＣＹコードから導き出すことができる。
【００５６】
第３コンポーネント３４はプロトコル・ツリーであり、様々な抽象化レベルのコンポーネント間の関係を記述する。図２に戻ると、これがプロトコル・ツリーの一実施例である。
第４コンポーネント３５は、生成アルゴリズムを含み、高いレベルのオブジェクトから相対的に低いレベルのオブジェクトを生成するのに使う。
【００５７】
最後の第５コンポーネント３６は認識アルゴリズムを含み、１または複数の低いレベルのオブジェクトから相対的に高いレベルのオブジェクトを認識するのに使う。
システムのコンポーネントのすべてを共通言語ＣＹを使ってモデル化すると仮定すると、それをコンパイラ３１が使って、さらなるコンピュータ・プログラムを適切なコンピューティング言語で生成することになり、図３に示しているモデル化およびシミュレーション・プロセスを使用して、システム内のコンポーネントを様々な抽象化レベルでモデル化できる。
【００５８】
本発明のある実施形態によると、ＣＹで書かれた単一のモデル仕様３０は、コンパイラ

３１内で、２以上のコンピューティング言語で書かれた２以上のコンピュータ・プログラムにコンパイルされる。具体的には、本発明のある実施形態では、モデル化するシステム内の相関コンポーネントを、ソフトウェアとハードウェアと別々に実装する。コンパイラは、どのコンポーネントをハードウェアに実装し、どれをソフトウェアに実装するかを決定する役割を担う。コンパイラ３１はＣＹソースコードを受け取って、ソフトウェアに実装するコンポーネントを記述するＣＹコードを、ソフトウェアの実装に適したコンピュータ・コード（Ｃ＋＋など）に変換する役割を担う。コンパイラはまた、ハードウェアに実装するコンポーネントを記述するＣＹソースコードも受け取って、そのＣＹコードをハードウェアの実装に適したコンピュータ・コード（ＶＨＤＬまたはＶｅｒｉｌｏｇなど）に変換する。

【００５９】

生成アルゴリズムおよび認識アルゴリズムは、ＣＹソースコード３０からコンパイルする。本発明の実施形態によると、ハードウェアに実装してモデル化したシステムのコンポーネントの場合、生成アルゴリズムおよび認識アルゴリズムは、ＣＹ言語を、例えば、Ｖｅｒｉｌｏｇにコンパイルした単一ブロックのコードで実装してもよい。この１つに結合した生成および認識アルゴリズムは状況に依存する。すなわち、生成または認識が起こるかは、生成／認識アルゴリズムに渡される入力次第である。

【００６０】

図２を参照して、生成／認識アルゴリズムへの入力がコンポーネントＡなどの単一コンポーネントならば、生成／認識アルゴリズムはこのコンポーネントが２つの子を有することを記憶しているため、コンポーネントＷおよびＸのインスタンスを生成する。すなわち、Ａを生成する場合、より低レベルのＷおよびＸも生成されなければならないことが導き出せる。逆に、生成／認識アルゴリズムへの入力がコンポーネントＷおよびＸならば、生成／認識アルゴリズムはこれら２つのコンポーネントの親がコンポーネントＡであることを認識し、そのためインスタンスＡを生成する。

【００６１】

このように、コンポーネントＡおよびＢはコンポーネントＷ，Ｘ，ＹおよびＺよりも抽象化レベルが高い。上記したように、コンポーネントはハードウェアかソフトウェアのいずれかに実装できる。考えられるある分割状態を図２に示しており、より高いレベルのコンポーネント（ＡおよびＢ）はソフトウェアに実装され、より低いレベルのコンポーネント（Ｗ，Ｘ，ＹおよびＺ）はハードウェアに実装されている。しかし、本発明はこれに限定されない。本発明のある実施形態では、どのコンポーネントも、またはコンポーネントのどのグループも、ハードウェアまたはソフトウェアのいずれにも実装できる。本発明の他の実施形態では、あるコンポーネントをハードウェアに実装することが不可能または実用的でないことがあり得る。このような実施形態では、コンパイラ３１がどのコンポーネントをハードウェアに実装するかを決定するときに、これらの問題を考慮する。

【００６２】

ソフトウェアにモデル化するコンポーネントの場合、各コンポーネントはどの抽象化レベルでもモデル化できる。ソフトウェアにモデル化するシステム内では、任意の数のコンポーネントを様々なレベルでモデル化することができる。しかし、本発明の実施形態によるモデル化およびシミュレーション・システムでソフトウェアにモデル化および実装するコンポーネントの場合、抽象化レベルの数を制限することが有利である。

【００６３】

ここで図４を参照すると、この図はシステムのコンポーネントをハードウェアにモデル化および実装するための使用に適した、標準化された一連の抽象化レベルを模式的に表している。モデル化システムの様々なコンポーネントに単独で取り組む開発者が、モデル化するコンポーネントを標準化された抽象化レベルで生成しやすくなるため、抽象化レベルの数を標準化することが望ましい。しかし、この抽象化レベルの正確な数が異なっていてもよいことは、当業者には容易に明らかとなろう。

【００６４】

図４は、個々のコンポーネントをモデル化できる４つの抽象化レベルを示している。加えて、かつ完全にするために、本発明はこれらのレベルのコンポーネントのモデル化には関係ないが、アルゴリズムレベルとゲートレベルを示している。各抽象化レベルを模式的に表しており、隣同士の抽象化レベルの関係を括弧で示して、ある抽象化レベル、またはレベルのある部分を次のレベルに関係付けている。

【００６５】

アルゴリズムレベル（ＡＬ）では、システムは一連の機能によって表されている。個々の機能間に通信はなく、むしろこれらは順次に実行されることが模式的に示されている。各機能の実行にかかる時間は考慮していない。

第１モデル化抽象化レベルはプログラマーズ・ビュー（ＰＶ）である。ＰＶレベルは一連の時間情報をもたない（非時限の）機能を含むが、異なる機能間の通信を考慮する。各機能は不連続なイベントと考えられる。また、ＰＶレベルでは、機能または機能のグループが並行して動作してもよい。ＡＬレベルおよびＰＶレベルでは、（図４に図示するように）機能間に１対１のマッピングがなくてもよい。ＰＶレベルは、プログラマがＣ＋＋などの高レベルのプログラミング言語でプログラムを作成するときにイベントのシーケンスをどのように理解するかに近いものである。

【００６６】

第２モデル化抽象化レベルは時間調節（タイミング）を伴うプログラマーズ・ビュー（ＰＶＴ）である。ＰＶＴレベルは、ＰＶレベルからの各機能にタイミングの検討を導入する。例えば、機能４０は、機能４０と並行して逐次実行する機能４１および４２と同じ時間がかかることが示されている。ＰＶＴレベルでのタイミングは概算である。

第３モデル化抽象化レベルはサイクル・コーラブル（ＣＣ）である。ＣＣレベルでは、タイミングは機能ごとに要するクロックサイクル数に基づいている。すなわち、各機能について、開始時間と終了時間がわかる。各機能はｎクロックサイクルを要する。図４は、モデル化しているＰＶＴレベルの機能４０を、ＣＣレベルの一連の機能４３として模式的に示している。ＣＣレベルでは、各機能中に出力が生成される時刻までは分からない。

【００６７】

第４モデル化抽象化レベルはサイクル精度／レジスタ転送レベル（ＣＡ／ＲＴＬ）である。ＣＡ／ＲＴＬレベルでは、ＣＣレベルからの各機能または機能のサブグループが、逐次実行される一連の機能で表わされ、各機能が１クロックサイクルを要する。このように、各機能からの出力の正確なタイミングが、もっとも近いクロックサイクルまでわかる。

最後に、図４はゲートレベル（ＧＬ）を示している。ＧＬレベルは、ＣＡ／ＲＴＬからの各機能または機能のサブグループを、ハードウェアに実装されるときに論理ゲートまたは一連の論理ゲートとして表す。本発明はＧＬレベルでのシステムのモデル化には関係していない。

【００６８】

ここで図５を参照すると、この図はモデル化およびシミュレーション・システムを模式的に示している。システムは２つの相関コンポーネント５０および５１として示され、それぞれＰＶレベルおよびＣＡレベルでソフトウェアにモデル化される。ソフトウェアに実装されているトランスレータ５２も、コンポーネント５０とコンポーネント５１との間で渡されるデータをＰＶレベルとＣＡ／ＲＴＬレベルとに変換する。第１方向（矢印５３で示す）で渡される場合、トランスレータは高レベルのトランザクションを低レベルの刺激に変換して、これをコンポーネント５１に渡す。第２方向（矢印５４で示す）に渡す場合、トランスレータはコンポーネント５１からの低レベルの結果を、コンポーネント５０用の高レベルのトランザクションに変換する。トランスレータ５２は当初プログラミング言語ＣＹで書かれている。コンポーネント５０および５１もＣＹで書いてもよい。あるいは、コンポーネント５０および５１は、１または複数の他のコンピュータ・プログラミング言語で書いてもよい。

【００６９】

図６は、本発明の実施形態によるモデル化およびシミュレーション・システムを示して

いる。前と同様、システムは２つの相関コンポーネント６０および６１として示しており、それぞれＰＶレベルおよびＣＡレベルでモデル化している。ただし、コンポーネント６０はソフトウェアにモデル化および実装し、コンポーネント６１はハードウェアにモデル化して実装している。

【００７０】

上記導入部で述べたように、システムの一部を高レベル（例、システムレベル）でソフトウェアに記述およびモデル化し、システムの一部を低レベル（例、ＲＴＬ）でハードウェアに記述およびモデル化する場合、システムを自動的にかつ正確にシミュレーションする方法は知られていない。本発明のある実施形態の目的は、ソフトウェアに実装したＲＴＬモデルおよびテストを、ハードウェアに実装したモデルと効率的に通信できるようにすることである。ソフトウェア・モデルとハードウェア・モデルとの通信の場合、ソフトウェア・コンポーネントとハードウェア・コンポーネントとの間でデータを伝送するために周知の標準協調エミュレーション・モデリング・インターフェース（ＳＣＥ－ＭＩ）が使用されている。

【００７１】

ハードウェア・コンポーネントとソフトウェア・コンポーネントとの間でデータを渡すトランスレータを２つの半体部に実装して、シミュレーション・システムのソフトウェア部分とシミュレーション・システムのハードウェア・エミュレータ部分との間に分割する。トランスレータ６２の第１半体部はソフトウェアに実装されている。ソフトウェアからハードウェアにＰＶレベルで情報が（ＳＣＥ－ＭＩインターフェースを介して）渡されるため、トランスレータのソフトウェア半体部６２がソフトウェアの情報をＰＶレベルより低い抽象化レベルまで変換する必要はない。そのため、トランスレータのソフトウェア半体部６２だけがＰＶレベルを含む。トランスレータの第２半体部６３はハードウェアに実装されている。トランスレータのハードウェア半体部６３はコンポーネント６０からの高レベルのトランザクションを、コンポーネント６１に渡す低レベルの刺激に変換する役割を担うため、トランスレータのハードウェア半体部はＰＶからＣＡ／ＲＴＬまですべてのレベルで示されている。トランスレータのハードウェア半体部６３はコンポーネント６１からの低レベルの結果をコンポーネント６０用の高レベルのトランザクションに変換する。

【００７２】

図５の場合と同様、トランスレータ６２，６３は当初ＣＹで書かれており、その後各半体部が適切な別のプログラミング言語に変換される（例えば、ソフトウェア実装の半体部にはＣ＋＋、ハードウェア実装の半体部にはＶｅｒｉｌｏｇ）。

トランスレータの部分間の通信はＰＶレベルで起こるように示されているが、同様に他のどのレベルであっても起こってもよい。トランスレータのソフトウェア部分６２とトランスレータのハードウェア部分６３との間の通信は、伝送するデータの総量を最小限にするとともに、ハードウェア・コンポーネントとソフトウェア・コンポーネントとの間のデータ交換の同期による障害を最小限にするために、できるだけ高い抽象化レベルで起こることが望ましい。コンポーネントをトランスレータ６２，６３を介して接続すれば、どの抽象化レベルでもモデル化および実装できる。本発明のある実施形態では、コンポーネント６１はＣＡ／ＲＴＬレベルで実装されている。ＣＡ／ＲＴＬレベルでモデル化したコンポーネントをハードウェアに実装することによって、システムのシミュレーションの速度が高まる。

【００７３】

図７は図６と同様なシミュレーション・システムを示しているが、システムを２つの相関コンポーネント７０および７１として示しており、それぞれＣＣレベルおよびＣＡレベルでモデル化している。しかし、コンポーネント７０はソフトウェアにモデル化および実装され、コンポーネント７１はハードウェアにモデル化および実装されている。図６と同様に、情報はＣＣレベルで（ＳＣＥ－ＭＩインターフェースを介して）ソフトウェアからハードウェアに渡されるため、トランスレータのソフトウェア半体部７２がソフトウェア

の情報をＣＣレベルより低い抽象化レベルまで変換する必要はない。しかし、トランスレータが渡すコンポーネント（メッセージおよびデータなど）をプロトコル・ツリーにおいてより高いコンポーネントに認識させるには、トランスレータのソフトウェア半体部７２ではＣＣレベルより上の抽象化レベルが必要である。そのため、トランスレータのソフトウェア半体部７２だけがＣＣレベルより上の抽象化レベルを含む。トランスレータの第２半体部７３はハードウェアに実装されている。トランスレータのハードウェア半体部はＰＶからＣＡ／ＲＴＬまでのすべてのレベルで示されている。トランスレータのソフトウェア半体部７２の場合と同様、認識するには、データを交換するＣＣレベルより上の抽象化レベルが必要である。トランスレータのハードウェア半体部６３は、コンポーネント６０からの高レベルのトランザクションを、コンポーネント６１に渡される低レベルの刺激に変換する役割を担う。トランスレータのハードウェア半体部７３はコンポーネント６１からの低レベルの結果を、コンポーネント７２用の高レベルのトランザクションに変換する。

【００７４】

トランスレータの第１半体部７２はソフトウェアに実装されている。トランスレータの第２半体部７３はハードウェアに実装されている。ＳＣＥ－ＭＩインターフェースを使用して、トランスレータの２つの半体部７２，７３の間はＣＣレベルで通信する。

トランスレータの部分間の通信はＣＣレベルで起こるように示されている。コンポーネント７０はＣＣレベルで実装されているため、データをトランスレータのハードウェア部分７３に渡す前にトランスレータのソフトウェア部分７２はデータを異なる抽象化レベルに変換する必要はない。トランスレータのハードウェア部分７３は、（矢印７４で示すとおり）ＣＣレベルのデータをＣＡ／ＲＴＬレベルのデータに、またその逆も同様に変換する役割を担う。

【００７５】

ここで図８を参照すると、この図は本発明のある実施形態により、ソフトウェアに実装してモデル化するコンポーネントとハードウェアに実装してモデル化するコンポーネントとの違いを示している。

コンポーネントＡはソフトウェアに実装されている。コンポーネントＡは有限状態機械（ＦＳＭ）を含む。コンポーネントＡ内のＦＳＭは入力８０と出力８１とを有し、ＦＳＭが他のコンポーネントとデータを送受信できるようにしている。ソフトウェアに実装されるコンポーネントの場合、シミュレーションの管理の役割を担う個別のシミュレータ・エンティティ（図８には図示せず）がある。ソフトウェア実装コンポーネントは互いに直接通信せず、シミュレータを介して通信する。イベントが発生する順序およびコンポーネントがシミュレーションされた作業（アクティビティ）を引き受ける順序は、シミュレータが決定する。ランタイム時のコンポーネントＡはシミュレータに要求を送るサービスが必要であり、タスクをプロセッサに割り当てる。

【００７６】

コンポーネントＸはハードウェアに実装されている。ＦＳＭを有することに加えて、コンポーネントＸはシミュレーション・エンティティ８２を有する。ハードウェアに実装された各モデル化コンポーネントのシミュレーション・エンティティは互いに通信して、まとまって分散シミュレーション・システム（ＤＳＳ）になる。ＤＳＳはイベントが発生する順序およびコンポーネントがシミュレーションされる作業を引き受ける順序を決定する役割を担う。

【００７７】

ＤＳＳはＣＹ（図３のＣＹソースコード）で書かれるそのコンポーネントのモデル仕様によって定義される。各コンポーネントのＣＹコードはコンポーネント仕様を含む。各コンポーネントの仕様は、機能仕様と関連シミュレーション・エレメントとを含む。機能仕様はコンポーネントの動作を定義し、そのコンポーネントのＦＳＭを定義し、シミュレーション・システム内でコンポーネントの動作をシミュレーションするのに使われる。シミュレーション・エレメントはＤＳＳの動作を定義し、そのコンポーネントが他のコンポー

ネントとどのように通信するかを定義する。

【００７８】

本発明のある実施形態により、各コンポーネントのＤＳＳエレメントのＣＹソースコードは、コンパイラ内のテンプレートコードが供給する。コンパイラは、モデル化されたシステムをコンパイルする時、ＤＳＳエレメントに適したＣＹコードを挿入するよう配列されている。

コンポーネントＸのＤＳＳエレメント８２は、ＦＳＭからの他のコンポーネントとのすべての通信を操作する（扱う）役割を担う。入力８３および出力８４はＤＳＳ８２に接続して、これがそれぞれＦＳＭ入力８５およびＦＳＭ出力８６を介してデータをＦＳＭに渡す。各ＤＳＳエレメントはそのコンポーネント内のリソース管理と通信、および他のコンポーネントとの通信を担う。ハードウェアに本質的な並列処理により、各ハードウェア実装コンポーネントは他のコンポーネントとは独立して作業を処理できる。

【００７９】

分散シミュレーション・システムは６つのコンポーネントを含む。ＤＳＳの第１コンポーネントは生成トリガを扱う。ＤＳＳは双方向のマルチプレクサを含み、他のコンポーネントから出入りする複数の出入力をより少ない数の通信リンクでＦＳＭに多重送信する。生成トリガが受信されると、生成トリガはＦＳＭに渡されて、より高い抽象化レベルの親コンポーネントの生成に応答して、またはソフトウェア実装コンポーネントからのトリガに応答して、そのコンポーネントのインスタンスの生成をトリガするのに使われる。生成トリガを第１コンポーネントのＤＳＳエレメントで受信するとすぐに生成イベントが起こる。生成が完了すると、完了通知が開始したアイテムに渡される。複数の生成トリガが受信された場合、これらは適切なリソースに分散される。利用できるリソースが十分にない場合、生成トリガはリソースが利用できるようになるまで待機状態になる。

【００８０】

ＤＳＳの第２コンポーネントは認識トリガを扱う。あるコンポーネントの子全部が生成または認識されると、ＤＳＳは認識トリガを生じて、親コンポーネントのインスタンスが要求されていることを知らせる。あるいは、あるコンポーネントの子の一部が生成または認識されたときに、認識トリガを生じてもよい。認識イベントを発生させるために、一連のイベントが発生してしまうまで、すなわち、子コンポーネント全部が生成または認識されるまで待つ必要があってもよい。そのため、認識イベントがトリガされる時点を識別できるようにするために、ＦＳＭと連動したＤＳＳは発生したイベントのシーケンスを追跡する。さらに、より低レベルのコンポーネントを認識しようとする場合、この識別がコンポーネントの階層の上まで渡されて、モデル化するシステムの状態を維持する。しかし、より低レベルのコンポーネントがインスタンス化（すなわち、生成または自己認識）されなければ、より高レベルのコンポーネントは実際に認識できない。

【００８１】

ＤＳＳの第３コンポーネントはパラメータ・ブロックである。これは生成および認識されたコンポーネントの動作中に、これらコンポーネント間で渡されるパラメータを扱う役割を担う。パラメータ・ブロックはパラメータがコンポーネント内に正確に記憶されることを確実にする役割も担う。生成イベント中、生成される子コンポーネントのパラメータ・ブロックはすぐにパラメータを取り込む（ポピュレートする）ことができる。しかし、認識イベントの場合、認識された親ブロックのパラメータ・ブロックは一定時間かけて移植され、その間に子コンポーネントが生成または自己認識される。あるコンポーネントが終了すると、そのパラメータのコピーがそれを生成した親または認識する子コンポーネントに渡されて、例えば、そのリソースを使って別のコンポーネントを生成または認識することによって、そのリソースがすぐに再利用してもよい。

【００８２】

ＤＳＳの第４コンポーネントは他のオブジェクトに対するオブジェクトのタイミングを担う。コンポーネントの開始時間と終了時間とは、そのコンポーネントのすぐ下のエレメントの階層によって決定する。開始時間と終了時間とは、階層レベルがより高いハードウ

ェア実装コンポーネントに渡される。コンポーネントの階層はプロトコル・ツリー３４で定義される。次に開始時間と終了時間とはソフトウェア実装コンポーネントに渡されて、コンポーネントのタイミングがハードウェア実装コンポーネントとソフトウェア実装コンポーネントとの間で一致する。

【００８３】

ＤＳＳの第５コンポーネントは、ハードウェア・コンポーネントが動作を完了したときにソフトウェア実装コンポーネントに通知する役割を担う。

ＤＳＳの第６コンポーネントは、ハードウェアに実装するシミュレータの部分（すなわち、ＤＳＳ）とソフトウェアに実装するシミュレータの部分とのシミュレーション時間の同期を担う。これはシミュレータのソフトウェア部分からメッセージを受信して、シミュレーションのハードウェア部分を進行させ、ハードウェア実装コンポーネントに適用されるクロックを制御する。ＤＳＳの第６コンポーネントは、ハードウェア・コンポーネントがそのタスクの処理を完了したら、または他のあるメッセージを送る必要があれば（例えば、別のコンポーネントが完了してそれをソフトウェアに通知する必要がある）、シミュレータのソフトウェア部分にメッセージを送り返す。ＤＳＳの第６コンポーネントは、ハードウェア実装コンポーネントとソフトウェア実装コンポーネントとのシミュレーションのタイミングを同期させる役割を担うため、このインスタンスは１つしか存在しない。このようにして、ＤＳＳの第６エレメントは各ハードウェア実装コンポーネントの外部に置かれている。個々のハードウェア実装コンポーネントは、内部のタイミングのために未制御クロックを含んでいてもよい。ＤＳＳの第６コンポーネントは、図９ないし図１１には明示的には示されていない。

【００８４】

ソフトウェア実装コンポーネントおよびハードウェア実装コンポーネントともに、有限状態機械（ＦＳＭ）がそのコンポーネントの機能性を実装する役割を担う。さらに、ＦＳＭは、任意のプロトコル・エレメント（すなわち、アイテムはあるインスタンス化で２つの子からなり、別のインスタンス化で３つの子からなってもよい）を含めて、プロトコル・ツリーの階層構造およびエレメント間のタイミングの関係を定義する。

【００８５】

図９を参照すると、２つのコンポーネントＸおよびＹを模式的に示しており、その両方ともハードウェアに実装されている。ＸおよびＹは両方ともＤＳＳエレメント８２を内蔵している。ＤＳＳエレメント８２は、複数の双方向の出入力を有するように示されている。これらの通信リンクは、（ソフトウェア・シミュレータを介した）ソフトウェア・コンポーネントとの通信用のものと、他のハードウェア実装コンポーネントとの通信用とに分割されている。図９に示すように、ハードウェア実装コンポーネントは、分散シミュレーション・システムを介して他のハードウェア実装コンポーネントと直接通信できる。

【００８６】

ここで図１０を参照すると、これは本発明のある実施形態によるモデル化およびシミュレーション・システムを示している。このシステムは４つの相関コンポーネントを有するように示されている。コンポーネントＡおよびＢの２つはソフトウェアに実装されている。ＡおよびＢはソフトウェア・シミュレータ１００を介して互いにおよび他のコンポーネントと通信する。残り２つのコンポーネントＸおよびＹはハードウェアに実装されている。ＸおよびＹは線１０１で示すように、互いに直接通信できる。コンポーネントＡおよびＢと通信するために、コンポーネントＸおよびＹはソフトウェア・シミュレータ１００と通信する。そのため、ＸおよびＹの通信形態は、通信しようとする他のコンポーネントがハードウェア実装コンポーネントであるか、またはソフトウェア実装コンポーネントであるかによって変わる。逆に、ＡおよびＢの場合、ＡおよびＢがシミュレータを介してしかデータを送れないため、通信モードは常に同じである。

【００８７】

第１ハードウェア・コンポーネントのモデルは、第１コンポーネントが他のどのコンポーネントを認識しているかを記述する。例えば、図１０のシステムの場合、コンポーネン

トＸのモデルはコンポーネントＹに関する情報を含んでいる。コンポーネントＸはコンポーネントＹがハードウェアに実装されていることを識別している。このようにして、コンポーネントＸがコンポーネントＹと通信する必要がある場合、リンク１０１を介してしかデータを送らない。同様に、コンポーネントＸはコンポーネントＡがソフトウェアに実装されていることを識別している。そのため、コンポーネントＡと通信するために、コンポーネントＸはリンクを介してしかソフトウェア・シミュレータ１０２にデータを送らない。しかし、コンポーネントＸのモデルはコンポーネントＢの情報を含んでいない。そのため、コンポーネントＸがコンポーネントＢと通信する必要がある場合、要求はリンク１０２を介してソフトウェア・シミュレータに送るとともに、リンク１０１を介して他のハードウェア・コンポーネント（この場合はコンポーネントＹだけ）を送る。次に、コンポーネントＸは１つのリンクからしか応答を受け取らない。コンポーネントＸはコンポーネントＢとの通信を試みると、ソフトウェア・シミュレータおよび他のハードウェア・コンポーネント双方と通信を続ける。本発明の別の実施形態によると、代わりに、コンポーネントＸは他のハードウェア・コンポーネントとのリンクだけを介してコンポーネントＢとの通信を試みることができよう。

【００８８】

図１１は、ハードウェア実装コンポーネントが４つある以外は図１０と同様のシステムを示している。図１１は、ハードウェア実装コンポーネントのどの対の間にも直接通信リンクがあることを示している。

添付の特許請求の範囲から逸脱することなく、本発明の他の変型および利点は当業者には、容易に明らかとなろう。

本発明の精神および範囲は、添付する特許請求の範囲の中に存在するが、本願の出願時に存在し、その一部は補正により削除された、以下の［予備的な特許請求の範囲］の中にも潜在する。この［予備的な特許請求の範囲］の記載事項は、本願明細書の開示に含まれるものとする。

［予備的な特許請求の範囲］

［請求項１］

相関のある第１および第２コンポーネントを含むシステムをモデル化およびシミュレーションする方法であって、当該方法が、

前記第１コンポーネントおよび前記第２コンポーネントの挙動を、第１仕様および第２仕様を使用してモデル化する工程であって、前記第１仕様および前記第２仕様のそれぞれが機能仕様と関連シミュレーション・エレメントとを含む工程と、

前記第１コンポーネントおよび前記第２コンポーネントの挙動を前記第１仕様および前記第２仕様を使用してシミュレーションする工程とを含む方法。

［請求項２］

前記シミュレーション・システムが、前記第１コンポーネントおよび前記第２コンポーネントを表す前記第１仕様と前記第２仕様との通信を管理することを特徴とする、請求項１に記載の方法。

［請求項３］

前記シミュレーション・システムが、前記第１コンポーネントおよび前記第２コンポーネントを表す前記第１仕様および前記第２仕様がイベントを処理する順序を決定することを特徴とする、請求項１または２に記載の方法。

［請求項４］

前記シミュレーションする工程がさらに、相関エンティティの階層内の少なくとも１つの第１エンティティをインスタンス化する工程を含み、前記少なくとも１つの第１エンティティは前記第１機能仕様および前記第２機能仕様のうちの少なくとも１つによって定義されることを特徴とする、請求項１～３のいずれか１項に記載の方法。

［請求項５］

前記または各インスタンス化した第１エンティティに応答して、少なくとも１つのさらなるエンティティをインスタンス化する工程を含み、前記または各さらなるエンティティ

は前記第１機能仕様および前記第２機能仕様のうちの少なくとも１つによって定義され、前記または各さらなるエンティティは前記少なくとも１つの第１エンティティとの階層関係に基づいて前記シミュレーション・システムによって選択される、請求項４に記載の方法。

［請求項６］

前記第１コンポーネントおよび前記第２コンポーネントを、前記相関エンティティの階層内のそれぞれのエンティティによって表し、前記それぞれのエンティティのインスタンスを作成することを特徴とする、請求項５に記載の方法。

［請求項７］

前記少なくとも１つのさらなるエンティティが、前記相関エンティティの階層において前記少なくとも１つの第１エンティティの親であることを特徴とする、請求項５または６に記載の方法。

［請求項８］

前記少なくとも１つの別のエンティティが、前記相関エンティティの階層において前記少なくとも１つの第１エンティティの子であることを特徴とする、請求項５または６に記載の方法。

［請求項９］

前記階層内のエンティティは、前記第１コンポーネントと前記第２コンポーネントとの間で送信されるデータ、ならびに前記第１コンポーネントおよび前記第２コンポーネントを表す前記第１仕様と前記第２仕様との間の通信を表すインスタンスを生成することを特徴とする、請求項５～８のいずれか１項に記載の方法。

［請求項１０］

前記第１コンポーネントを前記第２コンポーネントよりも高い抽象化レベルでモデル化し、前記方法が前記第１コンポーネントと前記第２コンポーネントとの関係の詳細を前記階層を使って規定する工程を含むことを特徴とする、請求項５～９のいずれか１項に記載の方法。

［請求項１１］

前記エンティティ間の階層関係を、前記第１機能仕様および前記第２機能仕様のうちの少なくとも１つから導き出すことを特徴とする、請求項５～１０のいずれか１項に記載の方法。

［請求項１２］

前記第１仕様および前記第２仕様を、第１コンピュータ・プログラミング言語で書かれた第１コンピュータ・プログラムとして記述することを特徴とする、請求項１～１１のいずれか１項に記載の方法。

［請求項１３］

前記第１仕様と前記第２仕様との関係を前記第１コンピュータ・プログラム内に記述する工程をさらに含む、請求項１２に記載の方法。

［請求項１４］

第２コンピュータ・プログラムを生成するように前記第１コンピュータ・プログラムを処理する工程をさらに含む、請求項１２または１３に記載の方法。

［請求項１５］

前記処理する工程が、前記第１コンピュータ・プログラムを、前記第１コンピュータ・プログラミング言語から、第２コンピュータ・プログラミング言語に変換することを特徴とする、請求項１４に記載の方法。

［請求項１６］

前記第２コンピュータ・プログラミング言語がＶＨＤＬまたはＶｅｒｉｌｏｇであることを特徴とする、請求項１５に記載の方法。

［請求項１７］

前記第２コンピュータ・プログラムに基づいてハードウェアに前記第１コンポーネントおよび前記第２コンポーネントの実装を生成する工程をさらに含む請求項１４に記載の方

法。
［請求項１８］
前記処理する工程が、前記第２コンピュータ・プログラムに追加のコンピュータ・コードを挿入する工程を含み、前記追加のコンピュータ・コードが前記または各シミュレーション・エレメントを実装することを特徴とする、請求項１４～１７のいずれか１項に記載の方法。
［請求項１９］
第２コンピュータ・プログラムおよび第３コンピュータ・プログラムを生成するように前記第１コンピュータ・プログラムを処理する工程をさらに含む、請求項１２または１３に記載の方法。
［請求項２０］
前記処理する工程が、前記第１コンピュータ・プログラムを、前記第１コンピュータ・プログラミング言語から、それぞれ第２コンピュータ・プログラミング言語および第３コンピュータ・プログラミング言語に変換することを特徴とする、請求項１９に記載の方法。
［請求項２１］
前記第２コンピュータ・プログラミング言語がＶＨＤＬまたはＶｅｒｉｌｏｇであり、前記第３コンピュータ・プログラミング言語がＣまたはＣ＋＋であることを特徴とする、請求項２０に記載の方法。
［請求項２２］
前記第１コンポーネントをハードウェアに実装し、前記第２コンポーネントをソフトウェアに実装することを特徴とする、請求項２１に記載の方法。
［請求項２３］
少なくとも１つのさらなるコンポーネントの挙動を、少なくとも１つのさらなる機能仕様を使用してモデル化する工程と、
前記少なくとも１つのさらなる機能仕様に基づいて、ソフトウェアに前記少なくとも１つのさらなるコンポーネントの実装を生成する工程と、
前記少なくとも１つのさらなるコンポーネントの挙動を、前記少なくとも１つのさらなる機能仕様を使用してシミュレーションする工程とをさらに含む請求項１７もしくはそれに従属する請求項のいずれか１項のまたは請求項２２に記載の方法。
［請求項２４］
前記第１コンポーネントおよび前記第２コンポーネントと、前記少なくとも１つのさらなるコンポーネントとの通信を管理するために、ソフトウェア実装シミュレーション・システムを提供する工程をさらに含む、請求項２３に記載の方法。
［請求項２５］
前記第１コンポーネントおよび前記第２コンポーネントを表す前記第１仕様および前記第２仕様が、前記ソフトウェア実装シミュレーション・システムと、ハードウェアに実装されている他のコンポーネントを表す他の仕様とを介して、同時にさらなるコンポーネントとの通信を試みる工程をさらに含む、請求項２４に記載の方法。
［請求項２６］
前記第２コンピュータ・プログラムが、前記シミュレーションに応答して、より高い階層レベルのエンティティを、より低い階層レベルの少なくとも１つのエンティティに基づいて、インスタンス化するように構成した有限状態機械の実装を含むことを特徴とする、請求項１４または請求項１９もしくはそれに従属する請求項のいずれか１項に記載の方法。
［請求項２７］
前記第２コンピュータ・プログラムが、前記シミュレーションに応答して、より低い階層レベルのエンティティを、より高い階層レベルの少なくとも１つのエンティティに基づいて、インスタンス化するように構成した有限状態機械の実装を含むことを特徴とする、請求項１４、１９、または請求項１４もしくは請求項１９に従属する請求項のいずれか１

項に記載の方法。
［請求項２８］
前記第２コンピュータ・プログラムが、前記シミュレーションに応答して、より高い階層レベルおよびより低い階層レベル双方のエンティティを、それぞれより低い階層レベルまたはより高い階層レベルの少なくとも１つのエンティティに基づいて、インスタンス化するように構成した有限状態機械の実装を含むことを特徴とする、請求項１４または請求項１９もしくはそれに従属する請求項のいずれか１項に記載の方法。
［請求項２９］
請求項１～２８のいずれか１項の方法を行うようにコンピュータを制御するコンピュータ読取可能コードを伝達する伝達媒体。
［請求項３０］
プロセッサ読取可能命令を記憶するプログラム・メモリと、
前記プログラム・メモリに記憶されている命令を読み取って実行するように構成したプロセッサとを含み、
前記プロセッサ読み取り可能命令が、前記プロセッサを請求項１～２８のいずれか１項の方法を行うように制御する命令を含むことを特徴とする、システムをモデル化およびシミュレーションするコンピュータ装置。
［請求項３１］
第１仕様および第２仕様を記憶する記憶手段であって、前記第１仕様および前記第２仕様がそれぞれ第１コンポーネントおよび第２コンポーネントの挙動をモデル化し、前記第１仕様および前記第２仕様のそれぞれが機能仕様とシミュレーション・エレメントとを含む、記憶手段と、
前記第１コンポーネントおよび前記第２コンポーネントの挙動を前記第１仕様および前記第２仕様を使用してシミュレーションするように構成した処理手段とを含み、
前記シミュレーション・エレメントが、互いに通信してシミュレーション・システムを提供するように構成されていることを特徴とする、プログラマブル・コンピューティング・デバイス。
［請求項３２］
相関のある第１および第２コンポーネントを含むシステムをモデル化およびシミュレーションする装置であって、当該装置が、
前記第１コンポーネントおよび前記第２コンポーネントの挙動を、第１仕様および第２仕様を使用してモデル化するモデル化手段であって、前記第１仕様および前記第２仕様のそれぞれが機能仕様と関連シミュレーション・エレメントとを含む、モデル化手段と、
前記第１コンポーネントおよび前記第２コンポーネントの挙動を、前記第１仕様および前記第２仕様を使用してシミュレーションするシミュレーション手段とを含み、
前記シミュレーション・エレメントが互いに通信してシミュレーション・システムを提供することを特徴とする、装置。
［請求項５３］
相関のある第１および第２コンポーネントを含むシステムをモデル化およびシミュレーションする方法であって、当該方法が、
前記第１コンポーネントおよび前記第２コンポーネントの挙動を、第１コンピュータ・プログラミング言語で書かれたコンピュータ・プログラムに記述された第１仕様および第２仕様を使用してモデル化する工程と、
それぞれ第２コンピュータ・プログラミング言語および第３コンピュータ・プログラミング言語で書かれた第２コンピュータ・プログラムおよび第３コンピュータ・プログラムを生成するように、前記コンピュータ・プログラムを処理する工程と、
前記第２コンピュータ・プログラムに基づいて、ハードウェアに前記第１コンポーネントの実装を生成する工程と、
前記第３コンピュータ・プログラムに基づいて、ソフトウェアに前記第２コンポーネントの実装を生成する工程と、

前記第１コンポーネントおよび前記第２コンポーネントの挙動を、それぞれ前記第１コンポーネントの前記生成した実装と前記第２コンポーネントの前記生成した実装とを使用してシミュレーションする工程とを含む方法。

［請求項５４］

前記第１仕様は、機能仕様と関連シミュレーション・エレメントとを含み、前記シミュレーション・エレメントは、他のシミュレーション・エレメントと通信してシミュレーション・システムを提供するように配列し、当該方法が、前記第２コンポーネントおよび前記少なくとも１つのさらなるコンポーネントを表す前記仕様間の通信を管理するソフトウェア実装シミュレーション・システムを提供する工程を含むことを特徴とする、請求項５３に記載の方法。

［請求項５５］

前記シミュレーション・システムが、前記第１コンポーネントおよび前記第２コンポーネントを表す記第１仕様と前記第２仕様との通信を管理することを特徴とする、請求項５３または請求項５４に記載の方法。

［請求項５６］

前記シミュレーション・システムが、前記第１コンポーネントおよび第２コンポーネントを表す前記第１仕様および前記第２仕様がイベントを処理する順序を決定することを特徴とする、請求項５３～５５のいずれか１項に記載の方法。

［請求項５７］

前記シミュレーションする工程が、相関エンティティの階層内の少なくとも１つの第１エンティティをインスタンス化する工程をさらに含み、前記少なくとも１つの第１エンティティは、前記第１機能仕様および第２機能仕様のうちの少なくとも１つによって定義されることを特徴とする、請求項５３～５６のいずれか１項に記載の方法。

［請求項５８］

さらに、前記または各インスタンス化した第１エンティティに応答して、少なくとも１つのさらなるエンティティをインスタンス化する工程を含み、前記または各さらなるエンティティは前記第１機能仕様および前記第２機能仕様のうちの少なくとも１つによって定義され、前記または各さらなるエンティティは、前記少なくとも１つの第１エンティティとの階層関係に基づいて、前記シミュレーション・システムによって選択される、請求項５７に記載の方法。

［請求項５９］

前記第１コンポーネントおよび前記第２コンポーネントを前記相関エンティティの階層におけるそれぞれのエンティティによって表し、前記それぞれのエンティティのインスタンスを作成することを特徴とする、請求項５８に記載の方法。

［請求項６０］

前記少なくとも１つのさらなるエンティティが、前記相関エンティティの階層内の前記少なくとも１つの第１エンティティの親であることを特徴とする、請求項５８または請求項５９に記載の方法。

［請求項６１］

前記少なくとも１つのさらなるエンティティが、前記相関エンティティの階層内の前記少なくとも１つの第１エンティティの子であることを特徴とする、請求項５８または請求項５９に記載の方法。

［請求項６２］

前記階層内のエンティティは、前記第１コンポーネントおよび前記第２コンポーネントを表す前記第１仕様と前記第２仕様との間で送信されるデータを表すようにインスタンス化され、前記第１コンポーネントと前記第２コンポーネントとの通信をモデル化することを特徴とする、請求項５８～６１のいずれか１項に記載の方法。

［請求項６３］

前記第１コンポーネントを前記第２コンポーネントよりも高い抽象化レベルでモデル化し、当該方法が、前記第１コンポーネントと前記第２コンポーネントとの関係の詳細を、

前記階層を使って提供する工程を含むことを特徴とする、請求項５８～６２のいずれか１項に記載の方法。
［請求項６４］
前記エンティティ間の階層関係を、前記第１仕様および前記第２仕様のうちの少なくとも１つから導き出すことを特徴とする、請求項５８～６３のいずれか１項に記載の方法。
［請求項６５］
前記処理する工程が、前記第２コンピュータ・プログラムおよび前記第３コンピュータ・プログラムに追加のコンピュータ・コードを挿入する工程を含み、前記追加のコンピュータ・コードが、前記または各シミュレーション・エレメントを実装することを特徴とする、請求項５４～６４のいずれか１項に記載の方法。
［請求項６６］
前記第２プログラミング言語がＶＨＤＬまたはＶｅｒｉｌｏｇであり、前記第３プログラミング言語がＣまたはＣ＋＋であることを特徴とする、請求項５３～６５のいずれか１項に記載の方法。
［請求項６７］
請求項５３～６６のいずれか１項の方法を行うようにコンピュータを制御するコンピュータ読取可能コードを伝達する伝達媒体。
［請求項６８］
プロセッサ読取可能命令を記憶するプログラム・メモリと、
前記プログラム・メモリ内に記憶されている命令を読み出して実行するように構成したプロセッサとを含み、
前記プロセッサ読取可能命令が、請求項５３～６６のいずれか１項の方法を行うようにプロセッサを制御する命令を含むことを特徴とする、システムをモデル化およびシミュレーションするコンピュータ装置。
［請求項６９］
第１仕様および第２仕様を記憶する記憶手段であって、前記第１仕様および前記第２仕様がそれぞれ第１コンポーネントおよび第２コンポーネントの挙動をモデル化し、前記第１仕様および前記第２仕様は、第１コンピュータ・プログラミング言語で書かれたコンピュータ・プログラムに記述されている、記憶手段と、
それぞれ第２コンピュータ・プログラミング言語および第３コンピュータ・プログラミング言語で書かれた第２コンピュータ・プログラムおよび第３コンピュータ・プログラムを生成するように、前記コンピュータ・プログラムを処理するために構成した処理手段と、
前記第２コンピュータ・プログラムに基づいてハードウェアに前記第１コンポーネントの実装を生成するために構成した第１生成手段と、
前記第３コンピュータ・プログラムに基づいてソフトウェアに前記第２コンポーネントの実装を生成するために構成した第２生成手段と、
前記第１コンポーネントおよび前記第２コンポーネントの挙動を、それぞれ前記第１コンポーネントの前記生成した実装と前記第２コンポーネントの前記生成した実装とを使用して、シミュレーションするために構成したシミュレーション手段とを含むプログラマブル・コンピューティング・デバイス。
［請求項７０］
コンポーネントを含むシステムをモデル化およびシミュレーションする装置であって、当該装置が、
第１コンポーネントおよび第２コンポーネントの挙動を、第１コンピュータ・プログラミング言語でコンピュータ・プログラムに記述されている第１仕様および第２仕様を使用してモデル化するモデル化手段と、
それぞれ第２コンピュータ・プログラミング言語および第３コンピュータ・プログラミング言語で第２コンピュータ・プログラムおよび第３コンピュータ・プログラムを生成するように、前記コンピュータ・プログラムを処理する処理手段と、
前記第２コンピュータ・プログラムに基づいて、ハードウェアに前記第１コンポーネン

トの実装を生成する第１生成手段と、
前記第３コンピュータ・プログラムに基づいて、ソフトウェアに前記第２コンポーネントの実装を生成する第２生成手段と、
前記第１コンポーネントおよび前記第２コンポーネントの挙動を、それぞれ前記第１コンポーネントの前記生成した実装と前記第２コンポーネントの前記生成した実装とを使用してシミュレーションするシミュレーション手段とを含む、装置。

【図面の簡単な説明】

【００８９】

【図１】システムのコンポーネントを様々な抽象化レベルでモデル化する先行技術のモデル化プロセスを模式的に示している。

【図２】本発明の実施形態での使用に適したプロトコル・ツリーを模式的に示している。

【図３】本発明の実施形態によるモデル化およびシミュレーション方法のコンパイル・プロセスを模式的に示している。

【図４】本発明の実施形態によるモデル化およびシミュレーション・システムでの使用に適した抽象化レベルの縮少化した階層を模式的に示している。

【図５】本発明の実施形態により相関コンポーネントを異なる抽象化レベルでモデル化するのに適したモデル化およびシミュレーション・システムを模式的に示している。

【図６】本発明の実施形態によりハードウェアおよびソフトウェアに実装されている相関コンポーネントをモデル化するのに適したモデル化およびシミュレーション・システムを模式的に示している。

【図７】本発明のさらなる実施形態によりハードウェアおよびソフトウェアに実装した相関コンポーネントをモデル化するのに適したモデル化およびシミュレーション・システムを模式的に示している。

【図８】本発明の実施形態によりハードウェアおよびソフトウェアに実装したモデル化したコンポーネント間の違いを模式的に示している。

【図９】本発明の実施形態によりハードウェアに実装した２つの相関コンポーネントを模式的に示している。

【図１０】本発明の実施形態によるモデル化およびシミュレーション・システムを模式的に示している。

【図１１】本発明のさらなる実施形態によるモデル化およびシミュレーション・システムを模式的に示している。

【図１】

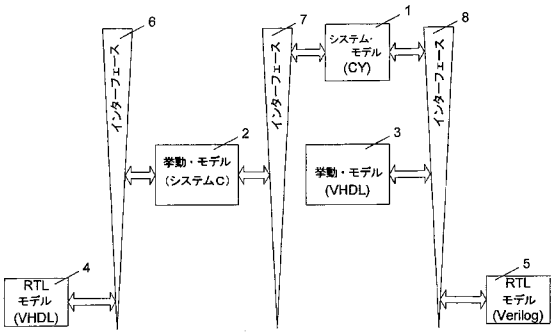

図１

【図２】

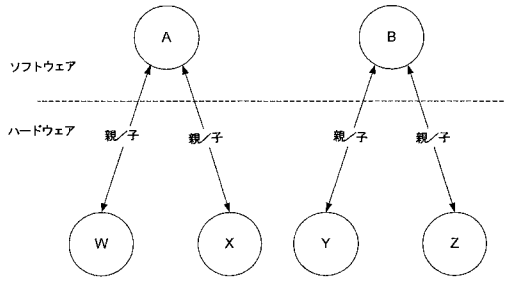

図２

【図３】

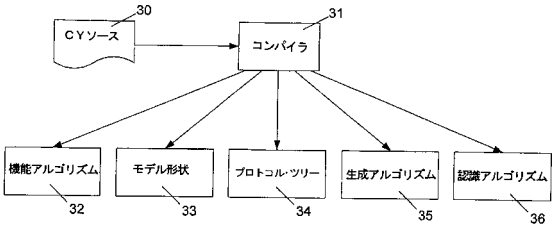

図３

【図４】

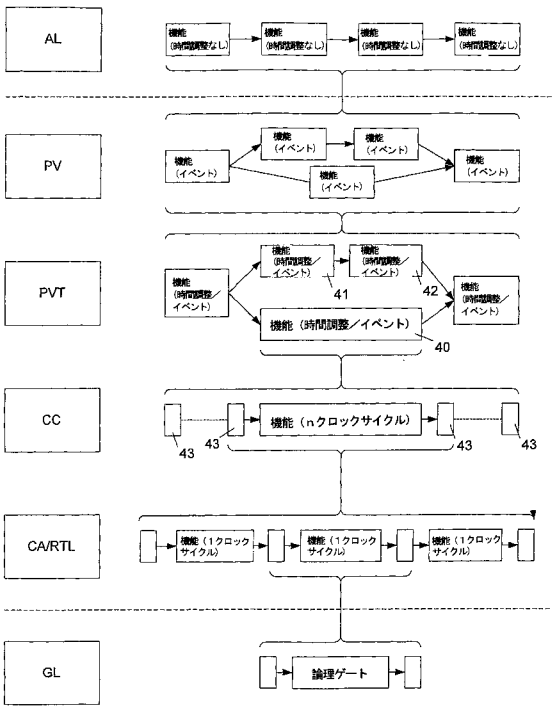

図４

【図５】

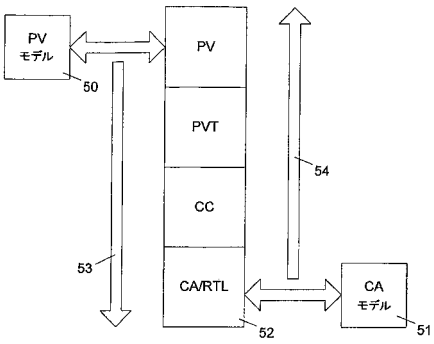

図５

【図６】

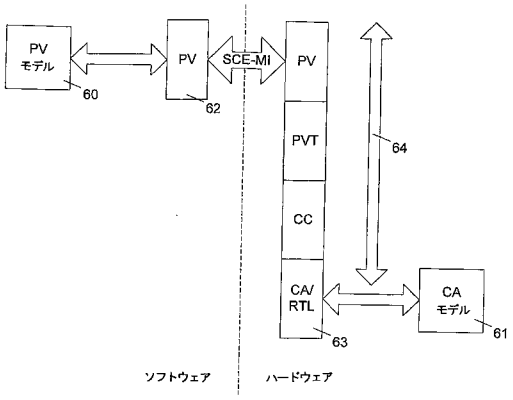

図６

【図７】

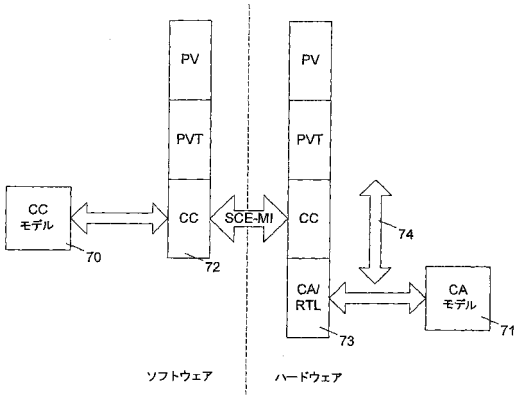

図７

【図８】

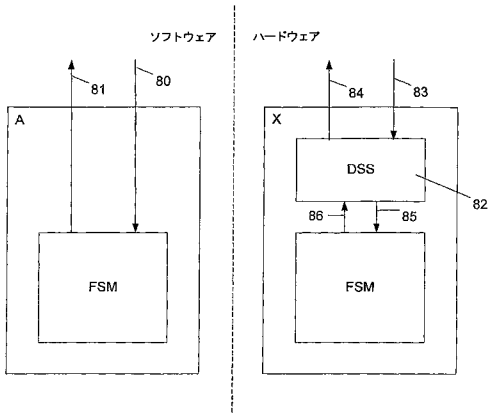

図８

【図９】

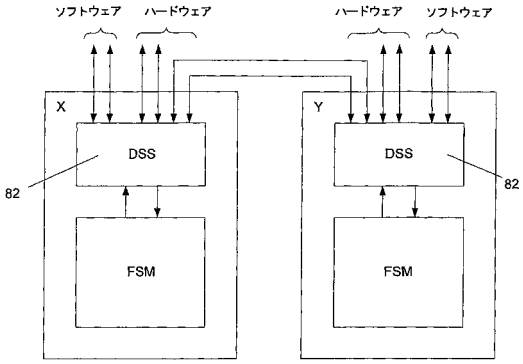

図９

【図１０】

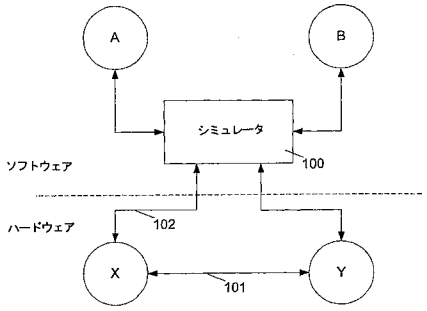

図１０

【図１１】

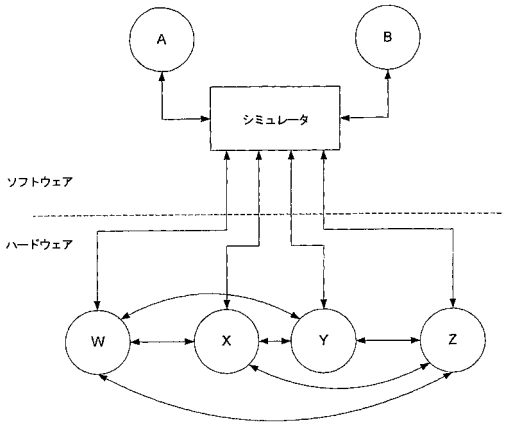

図１１

フロントページの続き

(72)発明者 パーカー，ダニエル，ロビン
イギリス，ハイ ピーク エス・ケイ・２３ ７・ディー・ワイ，ワーレイ ブリッジ，タクサル，ザ コーチ ハウス
(72)発明者 ジョーンズ，クリストファー
イギリス，マンチェスター エム・２６ ４・キュー・イー，ラドクリフ，イーストフィールズ ３４番地
(72)発明者 ポリクロノポウロス，ジェイソン，ソティリス
イギリス，マンチェスター エム・４１ ５・エイ・ジー，ウルムストン，グロスヴェナー ロード ５３番地

審査官 合田 幸裕

(56)参考文献 特開２００５－０８４９５６（ＪＰ，Ａ）
特開２００３－３３０９９２（ＪＰ，Ａ）
特開２００３－１１４９１４（ＪＰ，Ａ）
特開２００２－１７５３４４（ＪＰ，Ａ）
特開２００５－３３２１６２（ＪＰ，Ａ）
特開２０００－３１５２２２（ＪＰ，Ａ）
米国特許出願公開第２００５／００９１０２６（ＵＳ，Ａ１）
特開２０００－２１５２２６（ＪＰ，Ａ）
大西 充久，西田 浩一，岡田 和久，ハードウェアコンパイラＢａｃｈを用いたハードウェア／ソフトウェア協調設計環境について，ＤＡシンポジウム 2000，日本，社団法人情報処理学会，２０００年 7月１７日，第2000巻／第8号，第13-18頁
湯山 洋一，高井 幸輔，小林 和淑，小野寺 秀俊，ＳｙｓｔｅｍＣを用いたハードウェア・ソフトウェア協調設計，第１４回 回路とシステム（軽井沢）ワークショップ 論文集，日本，電子情報通信学会 システムと信号処理サブソサイエティ，電子情報通信学会 非線形問題研究専門委員会，２００１年 4月２３日，第399-404頁
Young-Il Kim, Ki-Young Ahn, Heejun Shim, Wooseung Yang, Young-Su Kwon, Ando Ki and Chong-Min Kyung, Automatic Generation of Software/Hardware Co-Emulation Interface for Transaction-Level Communication, VLSI Design, Automation and Test, 2005. (VLSI-TSA-DAT). 2005 IEEE VLSI-TSA International Symposium on, ２００５年 4月２７日，第196-199頁

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ １７／５０
Ｇ０６Ｆ ９／４５５
ＩＥＥＥ Ｘｐｌｏｒｅ
ＣｉＮｉｉ
ＪＳＴＰｌｕｓ（ＪＤｒｅａｍＩＩＩ）