(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第3953243号
(P3953243)
(45)発行日　平成19年8月8日(2007.8.8)　(24)登録日　平成19年5月11日(2007.5.11)

(51)Int.Cl.　F I
G06F 13/00 (2006.01)　G06F 13/00 301M

請求項の数 16 (全 33 頁)

(21)出願番号　特願平11-352380
(22)出願日　平成11年12月10日(1999.12.10)
(65)公開番号　特開2000-194610(P2000-194610A)
(43)公開日　平成12年7月14日(2000.7.14)
審査請求日　平成11年12月13日(1999.12.13)
審判番号　不服2003-16255(P2003-16255/J1)
審判請求日　平成15年8月22日(2003.8.22)
(31)優先権主張番号　98480098.7
(32)優先日　平成10年12月29日(1998.12.29)
(33)優先権主張国　欧州特許庁(EP)

(73)特許権者　390009531
インターナショナル・ビジネス・マシーンズ・コーポレーション
ＩＮＴＥＲＮＡＴＩＯＮＡＬ　ＢＵＳＩＮＥＳＳ　ＭＡＳＣＨＩＮＥＳ　ＣＯＲＰＯＲＡＴＩＯＮ
アメリカ合衆国１０５０４　ニューヨーク州　アーモンク　ニュー　オーチャード　ロード
(74)代理人　100086243
弁理士　坂口　博
(74)代理人　100091568
弁理士　市位　嘉宏

最終頁に続く

(54)【発明の名称】システム分析のためにバス・アービトレーション制御を使用する同期方法及び装置

(57)【特許請求の範囲】
【請求項１】
（Ａ）各々が共通バスへの少なくとも１つの経路を有する、並列動作の複数のエージェントを有するシステムを実行するステップと、
（Ｂ）バス・アービトレーション機構を用い、所定の技法に従い前記複数のエージェント間を同期させるステップにおいて、
（ａ）前記技法が、前記エージェントの少なくとも１つが保留のトランザクションを実行しようとする時期を検出し、該エージェントの少なくとも１つが活動化されないことを示す場合、前記保留のトランザクションの実行を阻止するため、前記共通バスへの該少なくとも１つのエージェントの前記少なくとも１つの経路を遮断するステップと、
（ｂ）一旦前記技法が、前記エージェントの少なくとも１つが活動化され得ることを示すと、前記保留のトランザクションの実行を可能にするため、前記共通バスへの該少なくとも１つのエージェントの前記少なくとも１つの経路を回復するステップと、
を含む、同期させるステップと、
を含む、システム分析方法。
【請求項２】
前記複数のエージェントが複数のプロセッサを含む、請求項１記載の方法。
【請求項３】
前記複数のエージェントが少なくとも１つのプロセッサ及び少なくとも１つのＩ／Ｏユニットを含む、請求項１記載の方法。

【請求項４】
前記複数のエージェントが、少なくとも１つの中間バスを介して、前記共通バスへの経路を有する少なくとも１つのエージェントを含む、請求項１記載の方法。
【請求項５】
前記複数のエージェントが、１つのアドレスをほぼ同時にアクセスする少なくとも２つのエージェントを含み、前記同期させるステップが、前記２つのエージェントの所定の１つが、他のエージェントが前記アドレスをアクセスする以前に、該アドレスをアクセスするように強制する、請求項１記載の方法。
【請求項６】
前記複数のエージェントがメモリ制御装置の少なくとも１つのバッファを充填するように動作し、前記同期させるステップが、前記メモリ制御装置が前記共通バスをアクセスするのを阻止するステップを含む、請求項１記載の方法。
【請求項７】
前記複数のエージェントがバス・ブリッジの少なくとも１つのバッファを充填するように動作し、前記同期させるステップが、前記バス・ブリッジが前記共通バスをアクセスするのを阻止するステップを含む、請求項１記載の方法。
【請求項８】
（Ａ）各々が共通バスへの少なくとも１つの経路を有する、並列動作の複数のエージェントを有するシステムを実行するシステム・マニピュレータと、
（Ｂ）所定の技法に従い、前記複数のエージェント間を同期させるバス・アービトレーション機構であって、
（ａ）前記エージェントの少なくとも１つが保留のトランザクションを実行しようとする時期を検出し、該エージェントの少なくとも１つが活動化されないことを示す場合、前記保留のトランザクションの実行を阻止するため、前記共通バスへの該少なくとも１つのエージェントの経路を遮断するステップと、
（ｂ）一旦前記技法が、前記エージェントの少なくとも１つが活動化され得ることを示すと、前記保留のトランザクションの実行を可能にするため、前記共通バスへの該少なくとも１つのエージェントの前記少なくとも１つの経路を回復するステップと、
を含む、バス・アービトレーション機構と
を含む、装置。
【請求項９】
前記システム・マニピュレータが、前記システムをシミュレートするシステム・シミュレータを含む、請求項８記載の装置。
【請求項１０】
前記システム・マニピュレータが、前記システムの性能分析を実行する性能アナライザを含む、請求項８記載の装置。
【請求項１１】
前記システム・マニピュレータが、前記システムを検証するシステム・ベリファイヤを含む、請求項８記載の装置。
【請求項１２】
前記システム・マニピュレータが、前記システムをデバッグするシステム・デバッガを含む、請求項８記載の装置。
【請求項１３】
前記実行するステップが、前記システムをシミュレートするステップを含む、請求項１記載の方法。
【請求項１４】
前記実行するステップが、前記システムの性能分析を実行するステップを含む、請求項１記載の方法。
【請求項１５】
前記実行するステップが、前記システムを検証するステップを含む、請求項１記載の方

法。
【請求項１６】
前記実行するステップが、前記システムをデバッグするステップを含む、請求項１記載の方法。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】
本発明は、コンピュータ・システムを分析する装置及び方法に関する。
【０００２】
【従来の技術】
本出願人の知るところによれば、システム検証及び関連技術の最新技術が、次の文献等で述べられている。
【０００３】
［１］A. Aharon、D. Goodman、M. Levinger、Y. Lichtenstein、Y. Malka、C. Metzger、M. Molcho、及びG. Shurekによる"Test Program generation for functional verification of PowerPC processors in IBM"、in 32nd Design Automation Conference、DAC 95、pages 279-285、1995。
［２］Y. Lichtenstein、Y. Malka、及びA. Aharonによる"Model-based test generation for processor design verification"、in Innovative Applications of Artificial Intelligence（IAAI）、AAAI Press、1994。［３］ＩＢＭによる欧州特許出願第９４３０３１８１．５号（公開番号０　６２８　９１１　Ａ２）。
［４］D. Marr、S. Thakkar、及びR. Zuckerによる"Multiprocessor Validation of the Pentium pro Microprocessor"、Proceedings of the COMPCON'96。［５］J. Walterらによる"Hierarchical Random Simulation Approach for the Verification of S/390 CMOS Multiprocessors"、Proceedings of the DAC'97、pages 89-94。
［６］K. Hines及びG. Borrielloによる"Dynamic Communication Models in Embedded System Co-Simulation"、Proceedings of the DAC'97、pages 395-400。
［７］M. Bauer及びW. Eckerによる"Hardware/Software Co-Simulation in a VHDL-based Test Bench Approach"、Proceedings of the DAC'97、pages 774-779。
［８］Kreulenらによる米国特許第５７４０３５３号。
【０００４】
本発明の概要：
本発明はシステム分析のための、改善された装置及び方法を提供することを目的とする。
【０００５】
分析は検証、テスト、デバッグ及び性能分析を含むものと理解される。システム検証における同期の適用について述べることにする。
【０００６】
ハードウェア・システム検証は、あらゆるコンピュータ・システムの開発において、不可欠で重要な活動である。システム検証の目的は、製造の開始前に、バグと呼ばれる機能誤りから設計を浄化することである。システム設計誤りは異なる原因に起因し、それらには、別々の設計労力の結果生じ得るシステム・コンポーネント間の矛盾するインタフェース、システム・コンポーネントの検証プロセスからもれてしまう誤り、及びその他の理由などがある。全体的に、システム検証の目的は、システムの設計がその使用に準ずるか否かを検証することである。システムが製造される前に、システム検証により、こうした誤りを露呈することが重要である。なぜなら、製造後にバグを修正するコストは、非常に高額になり得るからである。
【０００７】
ハードウェア・システム検証のために使用される１方法は、システム・モデルのシミュレーションである。各システム・モデルは、システム・コンポーネントの設計から構成される。更に、周辺装置などのシステム・コンポーネントの振舞いをシミュレートするために

、ビヘイビュラル・ユニットがモデルに追加され得る。システム振舞いの検証は、テスト・プログラムのシミュレーションにより達成される。システム・テスト・プログラムは、システム・コンポーネントの各々に対する実行スレッドにより構成される。これらのシステム・テストは自動テスト生成ルールにより、自動的に生成され得る（従来の技術のセクションのところで掲げた文献１及び２は、シングル・プロセッサのためのテスト生成ツールについて述べている）。

【０００８】

システム・シミュレーション・モデルは、非常に大規模となり得る。すなわち、複数のプロセッサ、複数のバス・ブリッジ、メモリ制御装置、及びＩ／Ｏバスビヘイビュラルを含むようになる。こうしたモデルに対するテストのシミュレーションは、時間を消費する。多くのランダム・テストの単なる大規模シミュレーションでは、良好な検証効果は得られない。システム・テストの質を向上させる方法は、異なるシステム・コンポーネントを同期させることにより、興味深い事象シーケンスを強制することである。更に、良好な効果が、１つの事象セットの全ての可能な順序をテストすることにより獲得され得る。これは本発明で述べる同期法を使用することにより達成され得る。

【０００９】

システム検証はかなり複雑なテスト・プログラムにもとづく。プロセッサ検証のために使用されるテスト・プログラムに比較して、システム・テスト・プログラムは異なる独立のシステム・コンポーネントの同期を要求する。システム活動は、分散されたプロトコルのセットにより決定され、従って中央制御が欠如する。システム・コンポーネントが対話するシーケンスは、テストにより異なる。検証プランの良好な効果は、異なるシーケンスの組合せにより獲得される。従って、テストの間にシステム・コンポーネントの同期を制御する能力が、効率的なシステム検証にとって重要となる。

【００１０】

システム検証は自動テスト・プログラム生成にもとづく。自動テスト・プログラム生成ツールへの入力は、テスト生成命令のリストを含み得、出力は全てのシステム・コンポーネントへの初期化及び実行命令を含むテスト・プログラム、並びに期待結果を含み得る。同期は、設計自動化ツールの生産性を利用するために、テスト生成方法に統合されるべきである。更に、テスト同期は、同期されるシステム・コンポーネント間の遅延を制御するために、柔軟性を提供すべきである。短い遅延がシステム活動間の競合を増加させ、最終的に設計誤りの検出をもたらす。

【００１１】

エージェント間の同期を実現する幾つかの方法が存在する。次のフィーチャが、好適な同期方法にとって重要である。

【００１２】

高速性：

シグナリング事象と保留トランザクション間のできる限り少ないサイクル数。経験的に同期遅延が小さいほど、資源競合の機会が高いことが知れている。換言すると、シグナリング事象後、待機エージェントが保留トランザクションの実行を開始するために要する時間が、逼迫した対話を達成するのに重要である。

【００１３】

総称的な適用可能性：

例えばメモリ制御装置などの非始動システム・コンポーネントを含む、異なるタイプのエージェントによる使用可能性。

【００１４】

従来のテスト・システムは、それらがターゲットとするコンポーネントのタイプに依存した。プロセッサのために開発された同期機構、異なるビヘイビュラルの各々の詳細に依存して、それらに適用可能な技術が存在する。例えば、システム検証［上記文献４及び５］；ハンドシェーク同期［文献６］；マスタ同期［文献７］など。

【００１５】

本発明は、シグナリング事象と保留トランザクション間で可能な最短時間を一般に達成する同期方法を提供する。また、本発明は、プロセッサ、Ｉ／Ｏ装置、及びバスへのアクセスを有する非始動システム・コンポーネントと共に使用され得る。これはバス・ベースのシステムで使用される効率的で強力な同期方法である。

【００１６】

本発明の好適な実施例は、バス・アービトレーションを制御することにより、同期方法を提供する。この実施例は好適には、ＳＣ（同期制御装置）プログラムを含む。

【００１７】

本発明のシステムは、正規の複数コンポーネント・システム・オペレーションの間に使用され得る。本発明の好適な１実施例は、検証品質を改善する技術を含む。検証は通常、新たなシステムの設計労力の約５０％を占め、何十億ドルものコストがかかる。ここで述べられる同期技術は、設計バグを含むことで有名であり、ランダム検証ではシミュレーションが非常に困難な領域である資源競合をターゲットとする。従って、本発明の好適な実施例の１つの目的は、問題領域をターゲットとすることにより、限られたシミュレーション時間内で、システム検証労力をより効率化することである。

【００１８】

本発明の好適な実施例の特定のフィーチャは、ここで提供される同期はアーキテクチャ・レベルで述べられるが、任意のインプリメンテーション・レベルの同期を提供することである。これは一般に、バス・アービトレーションによりバス・トランザクション順序を制御する任意のバス・プロトコルと共に使用され得る。

【００１９】

一般に、本発明は、外部コードに対するＡＰＩを提供するシミュレータを使用し、これがシミュレーションの間にシミュレーション信号をポーリングし、それを変更可能にする。モデル・テクノロジ社及びシノプシス社からそれぞれ販売される、ModelSim/VHDL及びSynopsis VHDL System Simulation（ＶＳＳ）などの最新のシミュレータは、この機能を有する。この方法は、今日のほとんどのバス・プロトコルがそうであるように、バス・プロトコルがアービトレーションを使用する広範囲のシステム検証問題に適用するのに一般に十分である。

【００２０】

本発明のアービトレーション同期機構は、例えばＡＳ／４００及びＲＳ／６０００のシステム検証、並びにＬ２キャッシュ制御装置などのバス上に載置する他の装置の検証に好適である。

【００２１】

バス・アービトレーション同期は、少なくとも一部のアプリケーションでは、バスに接続されるエージェントを解放するための最速の非割込み方法（non-intrusive way）である。バス・アービトレーション同期の可能なアプリケーションには、次のものが含まれる。

ａ．システム検証プロセスの自動化において使用される同期制御装置。

ｂ．設計内のバグを検出するための手動テスト・プロセス。

ｃ．設計内の誤動作原因を識別するための手動デバッグ・プロセス。

ｄ．ハードウェア・システムの所与の設計の期待性能を評価するための全ての方法を含む性能分析。例えば、本発明のバス・アービトレーション同期は、評価システム内の所与のレベルのバス利用を確立するために使用され得る。

【００２２】

【発明が解決しようとする課題】

ここで示され、述べられる同期方法は、シーケンスの制御を達成するためのシステム・テストにおいて、非常に有用である。設計されるシステムの複雑度は、コンポーネントの数及び機能と共に増大するので、益々システム検証が重要視されるようになり、システム・テストにおける高度に洗練された方法として、アービトレーション同期の有用性が向上する。

【００２３】

用語説明：
エージェント（または装置）：
バスに接続されるシステム・コンポーネント。このコンポーネントはバス（アドレスまたはデータ・バス）のアービトレーションを要求し、バス保有期間のバス・マスタとなることができる。例えば、プロセッサは分割アドレス／データ・バス上でのアドレス保有期間のための、及び書込みデータ保有期間のためのバス・マスタとなり得る。他方、メモリ制御装置は、読出しデータ保有期間だけのバス・マスタとなり得る。
【００２４】
エージェント・タイプ：
例えば、PowerPC603プロセッサ、ＰＣＩ装置。
【００２５】
アービトレーション：
バスに接続される複数のシステム・コンポーネントが存在する場合、２つのシステム・コンポーネントが同時にバスに書込まないように保証するために使用される機構。この機構はアービトレーションとして知られ、同様にバスに接続されるアービタ・システム・コンポーネントの利用により達成される。しばしば、バス上の各システム・コンポーネントは要求ラインを介してバスへのアクセスを要求し、グラント・ラインを介してバスへのアクセスを与えられる。
【００２６】
ステーションへの到来（Arrival at the Station）：
保留のトランザクション以前に待機エージェントがトランザクションを完了した、従って、今にもその保留のトランザクションを実行しようとしている時点。
【００２７】
ビヘイビュラル（Behavioral）：
ビヘイビュラルはシステム検証のために、周辺（Ｉ／Ｏ）装置の振舞いをシミュレートするために使用される、または実際の設計ではない余分なエージェントの役割をする。
【００２８】
バス：
バスはシステム・コンポーネント間の共用通信リンクとして作用し、その上をアドレス及びデータが転送される。
【００２９】
初期エージェント：
アドレスをその関連バス上に出力するエージェント。
【００３０】
非始動エージェント：
アドレスをその関連バス上に決して出力しないが、バス上へのデータの提供者であり得るエージェント。
【００３１】
参照モデル：
システムの参照モデルは、特定の入力がシステム状態に対して有する効果を予測する。テスト生成の間の参照モデルの使用により、ジェネレータはテストにとって関心となるシステム資源の現状態を推定する。
【００３２】
事象（Event）：
シミュレーション実行時間中の意義深いバス／エージェント状態の特定の発生（例えば、ＰＣＩバスにおけるアドレス・フェーズの発生）。
【００３３】
シグナリング・エージェント／装置：
別のエージェント（"待機エージェント"または"保留エージェント"と呼ばれる）のオペレーションのための条件として作用する事象（"シグナリング事象"）が発生するエージェン

ト。シグナリング・エージェントにおいてシグナリング事象が発生するまで、待機／保留エージェントのオペレーションは保留にされる。

【００３４】

同期：

同期は事象シーケンスの順序を保存する。例えば同期は、あるエージェントがアドレスＡへのそのアクセスを完了後にだけ、別のエージェントにＡへの書込みを強制するために使用される。待機エージェントは待機（または保留）エージェント／装置と呼ばれる。待機エージェントは、シグナリング事象と呼ばれる所定の事象が発生するまで、その実行を継続しない。一旦シグナリング事象が発生すると、待機エージェントの保留トランザクションが実行される。

【００３５】

同期制御装置（ＳＣ）：

動機制御装置の目的は、所与の事象シーケンスを保証することである。シミュレーションの間、シミュレーションと並行して実行されるシミュレーション制御プログラムの存在が仮定される。このプログラムすなわち同期制御装置（ＳＣ）は、現システム状態をモニタでき、またエージェント間の同期を強要できる。ＳＣはシステム信号をモニタする能力を有し、またシステム信号を新たな特定の値に変化させることができる。

【００３６】

テスト・ジェネレータ：

検証の目的のためにシミュレートされるテストを生成するプロセス。

【００３７】

タイト・スヌーピング（tight snooping）：

これは２つのエージェントが同一のシステム資源、例えばメモり位置を、時間的に接近してアクセスする任意の事象シーケンスである。

【００３８】

時間：

本開示では、時間はシミュレーション・サイクルまたはクロック・サイクルに関して測定されるものとする。

【００３９】

トランザクション：

本開示の目的上、トランザクションは、システム・モデルの残りのものとのエージェントの外部インタフェースに反映される、そのエージェントの任意のオペレーションとして定義される。

【００４０】

保有期間（tenure）：

バス保有期間は、あるタイプの情報がバス上に出力される時間間隔である。例えば、メモリからのプロセッサ読出しの間、２保有期間すなわちアドレス保有期間及びデータ保有期間が存在する。アドレス保有期間の間、プロセッサはアドレスをバス上に出力する。データ保有期間の間、メモリ制御装置が要求データをバス上に出力する。

【００４１】

非アサート（de-asserted）：

非活動化を意味する。

【００４２】

エイエイオ（eieio）：

PowerPCアーキテクチャ使用は、エイエイオ命令を含む（"PowerPC 601 RISC Microprocessor User's Manual"、1993参照）。この命令はバス上でエイエイオ・トランザクションを生成する。エイエイオは本明細書では、単に、テスト・シミュレーションの間に、他の目的のためにバス上に現れないバス・トランザクションを生成する命令の例として使用される。同期制御装置が、保留トランザクションに対して準備完了状態の待機エージェントからの信号と解釈できる、専用のバス・トランザクションが要求される。

【００４３】
【課題を解決するための手段】
従って、本発明の好適な実施例によれば、並列動作の複数のエージェントを有するシステムを実行し、少なくとも１介入時点に、複数のエージェント間の同期順序を強制する同期方法が提供される。
【００４４】
更に、本発明の好適な実施例によれば、複数のエージェントが、ほとんど同時に１つのアドレスをアクセスする少なくとも２つのエージェントを含み、同期順序を強制するステップが、２つのエージェントの内の所定の１つが、他方がそのアドレスをアクセスするよりも先に、それをアクセスするように強制するステップを含む。
【００４５】
更に、本発明の好適な実施例によれば、複数のエージェントが第１及び第２のエージェントを含み、同期順序を強制するステップが、第１の介入時点に、第１及び第２のエージェント間の順序を強制するステップを含む。また、複数のエージェントが第３及び第４のエージェントを含み、それらの少なくとも一方が第１及び第２のエージェントと異なり、同期順序を強制するステップが、第１の介入時点と異なる第２の介入時点に、第３及び第４のエージェント間の順序を強制するステップを含む。
【００４６】
更に、本発明の好適な実施例によれば、複数のエージェントが個々のプロセッサ上に複数のスレッドを含む。
【００４７】
更に、本発明の好適な実施例によれば、複数のエージェントが、少なくとも１介入時点において、並列動作の少なくとも３つのエージェントを含む。
【００４８】
更に、本発明の好適な実施例によれば、複数のエージェントが、少なくとも１シグナリング事象を実行する少なくとも１シグナリング・エージェントと、少なくとも１保留トランザクションを実行する少なくとも１待機エージェントとを含み、同期順序を強制するステップが、少なくとも１シグナリング・エージェントが少なくとも１シグナリング事象を実行するまで、少なくとも１待機エージェントのオペレーションを保留にするステップを含む。
【００４９】
更に、本発明の好適な実施例によれば、少なくとも１シグナリング事象が複数のシグナリング事象を含み、同期順序を強制するステップが、複数のシグナリング事象が実行されるまで、少なくとも１待機エージェントのオペレーションを保留にするステップを含む。
【００５０】
更に、本発明の好適な実施例によれば、少なくとも１保留トランザクションが複数の保留トランザクションを含み、同期順序を強制するステップが、少なくとも１シグナリング事象が実行されるまで、複数の保留トランザクションを遅延するステップを含む。
【００５１】
更に、本発明の好適な実施例によれば、３つのエージェントを含み、第１のエージェントによるあるアドレスに対するアクセス・オペレーションが、第２及び第３のエージェントが同一のキャッシュ・ラインを一般に同時にアクセスする前に、発生しなければならない。
【００５２】
更に、本発明の好適な実施例によれば、並列動作の複数のエージェントが、バス・ブリッジの少なくとも１つのバッファを充填するように作用する。
【００５３】
更に、本発明の好適な実施例によれば、並列動作の複数のエージェントが、メモリ制御装置の少なくとも１つのバッファを充填するように作用する。
【００５４】

更に、本発明の好適な実施例によれば、並列動作の複数のエージェントを有するシステムを実行するシステム・マニピュレータと、少なくとも１介入時点において、複数のエージェント間の同期順序を強制するように作用する順序インポーザとを含む同期システムが提供される。

【００５５】

更に、本発明の好適な実施例によれば、複数のエージェントが複数の事象に参加し、同期順序を強制するステップが、複数の事象が全ての可能な順序で発生するように強制するステップを含む。

【００５６】

更に、本発明の好適な実施例によれば、少なくとも１シグナリング・エージェントが、少なくとも１シグナリング事象を実行する複数のシグナリング・エージェントを含み、同期順序を強制するステップが、複数のシグナリング・エージェントが少なくとも１シグナリング事象を実行するまで、少なくとも１待機エージェントのオペレーションを保留にするステップを含む。

【００５７】

更に、本発明の好適な実施例によれば、少なくとも１待機エージェントが複数の待機エージェントを含み、同期順序を強制するステップが、少なくとも１シグナリング・エージェントが少なくとも１シグナリング事象を実行するまで、複数の待機エージェントのオペレーションを保留にするステップを含む。

【００５８】

更に、本発明の好適な実施例によれば、各々が共通バスへの少なくとも１つの経路を有する、並列動作の複数のエージェントを有するシステムを実行するステップと、バス・アービトレーション機構により所定の技法に従い、複数のエージェント間を同期させるステップとを含む、同期方法が提供される。前記同期させるステップは、前記技法が個々のエージェントが活動化されないことを示す場合、共通バスへの前記エージェントの少なくとも１つの経路を遮断するステップと、一旦前記技法が個々のエージェントが活動化され得ることを示すと、共通バスへの個々のエージェントの少なくとも１つの経路を回復するステップとを含む。

【００５９】

更に、本発明の好適な実施例によれば、複数のエージェントが複数のプロセッサを含む。

【００６０】

更に、１本発明の好適な実施例によれば、複数のエージェントが少なくとも１つのプロセッサ及び少なくとも１つのＩ／Ｏユニットを含む。

【００６１】

更に、本発明の好適な実施例によれば、複数のエージェントが、少なくとも１つの中間バスを介する共通バスへの経路を有する、少なくとも１つのエージェントを含む。

【００６２】

更に、本発明の好適な実施例によれば、前記同期させるステップが外部的に、従ってエージェントの各々の内部構造とは無関係に実行される。

【００６３】

更に、本発明の好適な実施例によれば、前記同期させるステップが、体系的信号を強制するステップを含む。

【００６４】

更に、本発明の好適な実施例によれば、複数のエージェントが、ほとんど同時に１つのアドレスをアクセスする少なくとも２つのエージェントを含み、前記同期させるステップが、２つのエージェントの内の所定の１つが、他方がそのアドレスをアクセスするよりも先に、それをアクセスするように強制するステップを含む。

【００６５】

更に、本発明の好適な実施例によれば、複数のエージェントが第１及び第２のエージェントを含み、前記同期させるステップが、第１の介入時点に、第１及び第２のエージェント

間の順序を強制するステップを含む。また、複数のエージェントが第３及び第４のエージェントを含み、それらの少なくとも一方が第１及び第２のエージェントと異なり、前記同期させるステップが、第１の介入時点と異なる第２の介入時点に、第３及び第４のエージェント間の順序を強制するステップを含む。

【００６６】

更に、本発明の好適な実施例によれば、複数のエージェントが、メモリ制御装置の少なくとも１つのバッファを充填するように動作し、前記同期させるステップが、メモリ制御装置が共通バスをアクセスするのを阻止するステップを含む。

【００６７】

更に、本発明の好適な実施例によれば、複数のエージェントが、バス・ブリッジの少なくとも１つのバッファを充填するように作用し、前記同期させるステップが、バス・ブリッジが共通バスをアクセスするのを阻止するステップを含む。

【００６８】

更に、本発明の好適な実施例によれば、各々が共通バスへの少なくとも１つの経路を有する、並列動作の複数のエージェントを有するシステムを実行するシステム・マニピュレータと、所定の技法に従い、複数のエージェント間を同期させるバス・アービトレーション機構とを含む、同期装置が提供される。前記同期させるステップは、前記技法が個々のエージェントが活動化されないことを示す場合、共通バスへの個々のエージェントの少なくとも１つの経路を遮断するステップと、一旦前記技法が個々のエージェントが活動化され得ることを示すと、共通バスへの個々のエージェントの少なくとも１つの経路を回復するステップとを含む。

【００６９】

更に、本発明の好適な実施例によれば、システム・マニピュレータが、システムをシミュレートするように作用するシステム・シミュレータを含む。

【００７０】

更に、本発明の好適な実施例によれば、システム・マニピュレータが、システムの性能分析を行うように動作する性能アナライザを含む。

【００７１】

更に、本発明の好適な実施例によれば、システム・マニピュレータが、システムを検証するように作用するシステム・ベリファイヤを含む。

【００７２】

更に、本発明の好適な実施例によれば、システム・マニピュレータが、システムをデバッグするように作用するシステム・デバッガを含む。

【００７３】

更に、本発明の好適な実施例によれば、実行するステップが、システムをシミュレートするステップを含む。

【００７４】

更に、本発明の好適な実施例によれば、システムの性能分析を行うステップを含む。

【００７５】

更に、本発明の好適な実施例によれば、実行するステップが、システムを検証するステップを含む。

【００７６】

更に、本発明の好適な実施例によれば、実行するステップが、システムをデバッグするステップを含む。

【００７７】

【発明の実施の形態】

本文献の開示の一部では、著作権保護を受けた資料を含む。著作権の所有者は、特許文献または特許開示のファクシミリ複製に意義を唱えるものではない。なぜなら、それは特許及び商標局（Patent and Trademark Office）のファイルまたは記録に現れるからである。しかしながら、それ以外では、全ての著作権が保護される。

【００７８】
前述の説明は順序付け機構に関連し、自動システム・テスト・ジェネレータの例の一部を成す機構のレビューを含む。ここで示され述べられる本発明の適用性は、ここで述べられるテスト・システムに限られるものでないことが理解されよう。
【００７９】
本発明で提供される最小の同期機構は、図１に示される次のステップを含む。
ステップ２０：
同期制御装置または待機エージェントが、待機エージェントがその保留のトランザクションを実行しようとする時期を検出する。すなわち、検出は、待機エージェントの前のトランザクションの完了と、バス上での保留のトランザクションの出現との間に発生する。このステージは、"ステーションへの到来（arrival in the station）"と呼ばれる。
ステップ３０：
待機エージェントが、その保留のトランザクションの実行を阻止される。これは同期制御装置により、または待機エージェント自身により行われる。このステップはこれ以後、"ステーションにおける待機"または待機フェーズと呼ばれる。
ステップ４０：
同期制御装置または待機エージェントが、シグナリング事象が発生する時を検出する。これはシグナリング条件と呼ばれる。
ステップ５０：
待機エージェントが、その保留のトランザクションの実行を許可される。これは同期制御装置自身により、または待機エージェント自身により行われる。このステップは、シグナリング・フェーズと呼ばれる。
【００８０】
上述の同期機構は、システム・テスト・ジェネレータ内で実現され得る。
【００８１】
用語"ステーション"は、保留の装置の準備完了状態を検出するステップを指し示す。従って、ステップ２０及び４０は活動をチェックし、ステップ３０及び５０はアクションである。
【００８２】
図１は、上述の事象の順序を示す。待機エージェントがステーションに到来し、待機フェーズに入る。シグナリング事象が検出されるとき、待機エージェントを解放するための信号が送信される。
【００８３】
勿論、シグナリング条件が既に満足されたときに、待機エージェントがステーションに到来する場合、待機エージェントはそこに０サイクル留まる。このような場合、退化待機フェーズ（degenerated wait phase）と呼ばれる。こうしたケースでは、信号がステーションへの到来前に、若しくは到来直後に送信されるか、または決して送信されない。
【００８４】
本発明の好適な実施例に従い構成され動作する同期制御装置の好適な実施例の４つのフェーズが、図２の仮想フロー図に示される。このフロー図は、どの単一コンポーネントもその全てを実行せず、実行の順序が必ずしも図示の順序である必要がない点で、仮想的である。代わりに、その実行は通常分散され、並列に実行される。
【００８５】
同期は次の現象を生成するために使用され得る。
１．トランザクション順序付け：２つの事象が所与の順序で発生しなければならない。
２．ランデブ：複数の事象が同時に開始しなければならない。
３．相互排他：複数の事象が同時に発生すべきでない。
【００８６】
同期はこれらの任意の機能を生成するために使用され得る。
【００８７】

図３は、３つのエージェント間のランデブを示す。各エージェントはそのステーションに到来し、シグナリング条件が満足されるまで待機する。シグナリング条件は、３つの全てエージェントのそれらのステーションへの到来である。全てのエージェントに対して１つの条件が存在するが、３つの同期すなわち各エージェントに対して１つの同期が存在する。

【００８８】

図４は、２つのエージェントの相互実行を示す。４つの同期が使用される。なぜなら、各エージェントのクリティカル・ゾーンの開始及び終わりが別々のステーションであるからである。エージェントＡのその第１のステーション（ｓｔＡ）におけるシグナリング条件は、エージェントＢがまだｓｔＢに達していないか、或いは既にｓｔＢ'に達していることである。ｓｔＢにおけるエージェントＢのシグナリング条件は、エージェントＡがｓｔＡに達していないか、既にｓｔＡ'に達していることである。ｓｔＡ'（ｓｔＢ'）におけるＡ（Ｂ）のシグナリング条件は、待機及びシグナリング・フェーズのように退化される。図５乃至図１０は、様々な同期の時間的振舞いを示す。図５及び図６はトランザクション順序付けを示し、図７はランデブを示し、図８及び図９は相互実行を示す。

【００８９】

上述の同期は様々な態様で結合され得る。例えば、タイト・スヌーピングを達成するために、２つのエージェント間のランデブが、トランザクション順序付けの直前に発生すべきである。これは保留のトランザクションがシグナリング事象の後で、時間的に接近して発生することを保証する。

【００９０】

図１３は、本発明の好適な実施例に従い構成され動作する同期制御装置の動作の好適な方法を示す、単純化されたフロー図である。図１３のフロー図では、入力パラメータは次のようである。

ＡＧ：テスト中のエージェント

Ｓ＿ｃｏｎｄ：ステーションへの到来を検出する条件（本発明のステップ１）

Ｇ＿ｃｏｎｄ：シグナリング事象を検出する条件（本発明のステップ３）

【００９１】

ＳＴＯＰ（）及びＧＯ（）は、図１のステップ３０及び５０にそれぞれ対応する。Ｓ＿ｃｏｎｄ及びＧ＿ｃｏｎｄは好適には、シミュレーション実行中にシミュレートされるモデル上で発生する事象に関する論理関数である。

【００９２】

内部構造及び変数：

好適には、入力として受信される各タプル（ＡＧ、Ｓ＿ｃｏｎｄ、Ｇ＿ｃｏｎｄ）に対して、図１３のフロー図にもとづく同期制御装置がＯＣ（順序制御）オブジェクトを生成し、それをＯＣのリストに記憶する。このリストはタイプＯＣのｎｂＯＣ個のオブジェクトを含む。ＯＣ（ｉ）は、リスト内の位置ｉ（ここでｉ＝１、．．、ｎｂＯＣ）に記憶されるオブジェクトＯＣである。

【００９３】

ＳＣプログラムにより使用される各ＯＣは、次のものを含む。

ＡＧ：エージェント

ＡＧ＿ｓｔａｔｅ：エージェントの状態＝｛アクティブ／待機｝、アクティブに初期化

Ｓ＿ｃｏｎｄ

Ｇ＿ｃｏｎｄ

ＣＯＮＤ＝｛Ｓ＿ｃｏｎｄ、Ｇ＿ｃｏｎｄ、ＦＡＬＳＥ｝は、評価されるそれぞれの状態を指し示す変数。ＣＯＮＤはＳ＿ｃｏｎｄ、Ｇ＿ｃｏｎｄまたはＦＡＬＳＥを指し示し、Ｓ＿ｃｏｎｄに初期化される。

【００９４】

図２と図１３を比較すると、図１３は多重サイクル同期を示し、そこでは（ステップ１３０、１４０、１６０）複数のエージェントが複数の介入時点において同期される。図１３

のステップ１２０、１５０、１７０、１９０及び２００は一般に、図２のステップ６０、１１０、７０、８０、１００にそれぞれ対応する。

【００９５】

上述の同期機構の次に示すような多くの変形が使用され得る。

ステーション検出（図１のステップ２０）の変形：

ステップ１に対して、少なくとも次のオプションが存在する。そこでは同期制御装置が、待機エージェントがその保留のトランザクションを実行しようとする時期に注意を払う。

ａ．待機プロセッサが非ゼロ値を所定のレジスタに書込む。同期制御装置が次にこのレジスタを読出し、その値に従い、待機エージェントがその保留のトランザクションを実行するのを阻止する時期を知る（下記の例１（タイト・スヌーピング）のTrigReg参照）。

ｂ．同期制御装置は、待機エージェントの前のトランザクションの完了をチェックする。これはプロセッサの場合、非プログラム順序の実行により影響され得る。ビヘイビュラルの場合、これは単にビヘイビュラルの現トランザクション番号のチェックである。

ｃ．待機エージェントは、例えばＰｏｗｅｒＰＣのバス上のエイエイオなどの、同期命令を実行できる。この命令は単に、同期制御装置に、エージェントが待機準備完了状態であることを知らせるために実行されるべきである。同期制御装置はこの時、エイエイオ・トランザクションの成功終了のために、適切なバスをモニタできる。

【００９６】

シグナリング事象の検出（図１のステップ４０）の変形：

ステップ３すなわちシグナリング・エージェントの事象を検出する１つの可能なインプリメンテーションは、所与のバス上での所与のアドレスの出現である。別の例は、メモリ制御装置の読出しバッファがフルになることである。

【００９７】

待機及びシグナリング方法（図１のステップ３０及び５０）の変形：

信号がＳＣにより、またはモデル化装置により駆動され得る。ＳＣにより駆動される信号は、シミュレーション信号（またはシミュレーション信号のセット）の値の変化である。シミュレーション信号はプロセッサ・レジスタであり、イネーブル・ビット（またはイネーブル・アクション）、またはアービトレーション信号である。テスト・システムはまた、他の装置により送信される信号、すなわち（装置：プロセッサ信号：セマフォ値の変更）及び（装置：プロセッサ信号：構成サイクル）を使用し得る。

【００９８】

送信される信号は、装置により使用される待機方法に対応する。

【００９９】

図１１は、本発明の５つの異なる実施例に従い構成され動作する同期制御方法を比較するテーブルであり、これらは図１の"待機"及び"シグナリング"・ステップのインプリメンテーションと主に異なる。図１１の方法は、次の方法を含む。

ａ）レジスタ法（図１６）

初期化：各プロセッサが、同期制御装置のためだけに使用される専用のTrigRegレジスタを有する。初期には、その値は０である。

待機：待機プロセッサがループし、そのTrigRegレジスタが０でない特定の値にセットされるのを待機する。

シグナリング：同期制御装置が待機プロセッサのTrigRegレジスタを、それが待機している値に変更する。

制限：プロセッサ専用

ｂ）バス・アービトレーション方法（図１４）

待機：同期制御装置が待機エージェントのアービトレーション信号をアサート解除することにより、待機エージェントがバス上でアービトレーションを受信するのを阻止する。

シグナリング：同期制御装置が待機エージェントのアービトレーション信号のアサート解除を中止し、待機エージェントがその保留のバス・トランザクションを実行することを可能にする。

制限：技術的困難
ｃ）イネーブル・ビット方法（図１７）
初期化：各トランザクションに対するイネーブル・ビットを有するビヘイビュラルに対して使用される。すなわち、０の場合、トランザクションは実行され得ず、１の場合、実行され得る。初期には、保留のトランザクションのイネーブル・ビットは、０にセットされる。
待機：保留のトランザクションに対する待機ビヘイビュラルのイネーブル・ビットは、初期に０にセットされるので、ビヘイビュラルは実行を継続できない。
シグナリング：同期制御装置は、ビヘイビュラルの保留のトランザクションのイネーブル・ビットを１にセットし、ビヘイビュラルが実行を継続することを可能にする。
制限：１トランザクションにつき１イネーブル・ビットを有するビヘイビュラルに限られる。
ｄ）構成方法（図１８）
待機：この方法は、自身に向けられる構成トランザクションを介してコマンドを受信するビヘイビュラルに対して使用される。構成トランザクションを受信することなく、ビヘイビュラルは実行を継続する。
シグナリング：シグナリング・エージェントが、待機エージェントをイネーブルにする構成トランザクションを実行する。
制限：構成トランザクションをサポートするビヘイビュラル/装置に限られる。非常に低速。
ｅ）セマフォ方法（図１９）
初期化：初期に、セマフォが－１にセットされる。
待機：プロセッサがセマフォにおいて待機する。すなわち、特定のメモリ位置が０になるのを待機してループする。
シグナリング：シグナリング・エージェントがセマフォを解除する。すなわち、０を特定のメモリ位置に書込むことにより、待機エージェントをそのループから解放する。
制限：プロセッサ専用
【０１００】
好適なアービトレーション同期機構は、図１５に示される次のステップを含む。
ステップ３００：同期制御装置がバス上において、待機エージェントの保留トランザクションの保留の出現を検出する。
ステップ３１０：同期制御装置が待機エージェントのアービトレーション信号をアサート解除することにより、待機エージェントがバス上でアービトレーションを受信するのを阻止する。これは待機エージェントがバス上で更に別のトランザクションを実行するのを阻止する。
ステップ３１５乃至３４０：シグナリング事象が発生すると、同期制御装置は、待機エージェントのアービトレーション信号のアサート解除を中止する。これにより待機エージェントはバスへのアクセスを許可され、その保留のバス・トランザクションの実行が可能になる。
【０１０１】
図１４及び図１５を比較すると、図１５のステップ３００、３１０及び３１５は一般に、図１４のステップ２２０、２３０及び２４０にそれぞれ対応する。図１５のステップ３２０は一般に、図１４のステップ２５０、２６０及び２７０に対応する。ステップ３３０及び３４０は、同期制御装置により実行されるステップではなく、従って図１４では対応するステップを有さない。
【０１０２】
アービトレーション同期はデッドロックのリスクの結果生じ得る。デッドロックは、一旦待機エージェントがそのバスへのアービトレーションの獲得を阻止されると、何らかの理由によりバスをアクセスできないことから発生し得る。こうした理由には、実際の同期に接続されていないなどが含まれる。デッドロックの阻止は一般に、テスト・ライタまたは

自動テスト・ジェネレータの責任である。
【０１０３】
次に、３つのアプリケーション例について述べる。
例１：２つのプロセッサ間のタイト・スヌーピング
テスト・プログラムの第１の例は、２つのプロセッサ間のタイト・スヌーピングである。この状況の目的は、次の事象が順次的に、時間的に非常に接近して発生することである。
１．プロセッサＡがレジスタＲａからアドレスＸに書込む。
２．プロセッサＢがアドレスＸからレジスタＲｂに読出す。
【０１０４】
前記の２つの事象が要求順序に従い発生する場合、Ｒｂの最終値は、プロセッサＡにより書込まれた値である。しばしば、メモリ・アドレス上での競合が設計バグを暴露する。この例はまた、単一プロセッサ上の２つのスレッドにも当てはまる。この場合、図１６に示されるようなレジスタ待機／シグナリング方法が使用される。
【０１０５】
例２：読出しバッファの充填
この例では、テスト・プログラムがメモり制御装置の読出しバッファを充填する。図１２の部分的システムは、例えばメモり制御装置が８個の読出しバッファを含む例において使用される。
【０１０６】
この状況では、８個の読出しバッファを充填することが好ましい。これを達成するために、少なくとも８個のエージェント（プロセッサ及びＩ／Ｏ装置）が、メモリに対する読出し要求を実行するために使用され、その間、メモリ制御装置が任意の要求を満足するのを阻止する。一旦バッファがフルになると、メモリ制御装置は応答を許可される。バッファをしばしば充填することは、設計バグを暴露する。この例は、バス・ブリッジのバッファを充填するためにも、同様に使用され得る。
【０１０７】
例３：３つのプロセッサ間のスヌーピング
この状況の目的は、２つの他のプロセッサがあるアドレスから読出す前に、１プロセッサがその同一のアドレスに書込むことである。すなわち、後述のトランザクション１が、任意の順序で発生し得るトランザクション２及び３の両方より先に発生すべきである。Ｒｂ及びＲｃの最終値は、プロセッサＡにより書込まれた値であるべきである。
１．プロセッサＡがレジスタＲａからアドレスＸに書込む。
２．プロセッサＢがアドレスＸからレジスタＲｂに読出す。
３．プロセッサＣがアドレスＸからレジスタＲｃに読出す。
【０１０８】
上述の３つの例について詳述する。一般に、テスト・コンポーネントは、同期制御装置を介して同期されない限り互いに独立に実行される。
【０１０９】
例１：２つのプロセッサすなわちプロセッサＡ及びプロセッサＢ間のタイト・スヌーピング（Ｂの読出し前にＡの書込みが発生）
以下のテスト説明では、TrigRegは残りのテストにおいて使用される任意の定義済みのレジスタである。同期制御装置は、プロセッサＡのTrigReg（ProcA. TrigReg）及びプロセッサＢのTrigReg（ProcB. TrigReg）の値をそれぞれモニタすることにより、プロセッサＡ及びＢのアービトレーション・ラインをアサート解除する時期を知る。従って、プロセッサＡ及びＢは同期制御装置に、それらが待機準備完了状態であることを知らせる。
【０１１０】
テスト記述：
プロセッサＡ：初期には、ProcA. TrigReg＝＝０
１．ProcA. TrigReg＝１にセット
２．アドレスＸにストア命令を実行

プロセッサＢ：初期には、ProcB. TrigReg＝＝０
１．ProcB. TrigRegを１にセット
２．アドレスＸにロード命令を実行
同期制御装置：
１．ProcA. TrigReg＝＝１及びProcB. TrigReg＝＝０であれば、プロセッサＡのアービトレーション・ラインをアサート解除。
２．ProcB. TrigReg＝＝１であれば、プロセッＢのアービトレーション・ラインをアサート解除し、プロセッサＡのアービトレーション・ラインのアサート解除を中止。
３．システム・バス上で、アドレス＝Ｘ及びマスタ＝プロセッサＡなるトランザクションが発生する場合、プロセッサＢのアービトレーション・ラインのアサート解除を中止。

【０１１１】
分析：
このタイト・スヌーピングの例は、実際には、２つの単純な同期を結合し、ランデブ（同時に発生しなければならない複数の事象）の後に、トランザクション順序付け（２つの事象が所与の順序で発生）を伴う。ランデブのためのシグナリング事象は、ProcB. TrigReg＝＝１である。ランデブのための保留トランザクションは、プロセッサＡがアドレスＸをシステム・バス上に出力することである。トランザクション順序付けのためのシグナリング事象もまた、プロセッサＡがアドレスＸをシステム・バス上に出力することである。トランザクション順序付けのための保留トランザクションは、プロセッサＢがアドレスＸをシステム・バス上に出力することである。

【０１１２】
例２：読出しバッファの充填
この例では、メモリ制御装置は、そのデータ・バス・アービトレーション・ラインをアサートできないので、自身に転送される多くの読出し要求に応えることを阻止される。一旦同期制御装置が、読出しバッファがフルであることを検出すると、同期制御装置はデータ・バス・アービトレーション・ラインのアサート解除を中止する。

【０１１３】
テスト記述：
全てのプロセッサ及びＩ／Ｏエージェント（少なくとも８個）：
１．読出しトランザクションを実行
同期制御装置：
１．メモリ制御装置のデータ・バス・アービトレーション・ラインをアサート解除
２．読出しバッファがフルの場合、メモリ制御装置のデータ・バス・アービトレーション・ラインのアサート解除を中止

【０１１４】
分析：
この例では、シグナリング事象は読出しバッファがフルになることである。実際には、複数の保留トランザクションが存在する。換言すると、メモリ制御装置が、それに向けられる全ての読出しトランザクションのためのデータを提供する。実際には、一緒にシグナリング事象を発生させる複数のシグナリング・エージェントが存在する。

【０１１５】
例３：３つのプロセッサ間のスヌーピング
次のテスト記述では、ＴｒｉｇＲｅｇが残りのテストにおいて使用される任意の定義済みのレジスタである。同期制御装置は、プロセッサＢのTrigReg（ProcB. TrigReg）及びプロセッサＣのTrigReg（ProcC. TrigReg）の値をそれぞれモニタすることにより、プロセッサＢ及びＣのアービトレーション・ラインをアサート解除する時期を知る。従って、プロセッサＢ及びＣは同期制御装置に、それらが待機準備完了状態であることを知らせる。

【０１１６】
テスト記述：
プロセッサＡ：

１．アドレスＸにストア命令を実行
プロセッサＢ：初期には、ProcB. TrigReg＝＝０
１．ProcB. TrigRegを１にセット
２．アドレスＸにロード命令を実行
プロセッサＣ：初期には、ProcC. TrigReg＝＝０
１．ProcC. TrigRegを１にセット
２．アドレスＸにロード命令を実行
同期制御装置：
１．ProcB. TrigReg＝＝１であれば、プロセッＢのアービトレーション・ラインをアサート解除。
２．ProcC. TrigReg＝＝１であれば、プロセッＣのアービトレーション・ラインをアサート解除。
３．システム・バス上で、アドレス＝Ｘ及びマスタ＝プロセッサＡなるトランザクションが発生する場合、プロセッサＢ及びプロセッサＣのアービトレーション・ラインのアサート解除を中止。

【０１１７】

分析：
この例では、実際２つの待機エージェント及び２つの保留トランザクションが存在する。シグナリング事象は、プロセッサＡがシステム・バス上にアドレスＸを出力することである。保留のトランザクションは、プロセッサＢがシステム・バス上にアドレスＸを出力し、プロセッサＣがシステム・バス上にアドレスＸを出力することである。

【０１１８】

出願人により実施された実験において、ここで提案されるアービトレーション同期方法が、本発明の概要のセクションで上述したプロセッサ・レジスタ・ポーリング方法と比較された。シミュレートされた各テストは、少なくとも１０個のプロセッサ・ロード／ストア命令を含み、２つのプロセッサ間の少なくとも１つのタイト・スヌーピング状況を含んだ。テストはインハウス自動システム・テスト・ジェネレータを用いて生成された。使用されたＡＳ／４００シミュレーション・モデルは、２つのプロセッサを含んだ。１０のテストが各方法を用いて実行された。各テストにおける２つの同期事象間のサイクル数が表１に示される。

【表１】

同期方法タイミング比較：

テスト番号	アービトレーション同期事象間のサイクル数	レジスタ・ポーリング同期事象間のサイクル数
1	3	23
2	3	19
3	4	26
4	49（*）	62（*）
5	5	61（*）
6	3	30
7	49（*）	24
8	4	29
9	3	23
10	3	29

【０１１９】
マーク（*）付きのテーブル・エントリは、シグナリング・エージェントのバス・トランザクションと、待機エージェントの次のバス・トランザクション間に、命令フェッチが存在したテストを表す。ランダム・テスト生成の間、非常に多数のテストが生成される。これらのテストの大多数が要求された振舞いを果たす限り、要求された振舞いを果たさないテストは無視され得る。これらの値が無視される場合、アービトレーション同期方法は平均３．５サイクルの差を与えたのに対して、レジスタ・ポーリング方法は２５．４サイクルの平均を与えた。換言すると、平均してアービトレーション方法は、レジスタ・ポーリング方法の７倍の向上を達成した。更に、アービトレーション方法はしばしば、３サイクルの差を与えた。これは使用されたPowerPC60ｘ方式のバス・プロトコルにおいて、トランザクション間で可能な最小バス・サイクル数に相当する。
【０１２０】
レジスタ・ポーリング同期機構では、プロセッサは保留のバス・トランザクションを開始準備できる前に、複数の命令を実行する。しかしながら、アービトレーション同期機構では、遅延は、バス・アービタがプロセッサのバス要求に応答するのに要する時間だけである。待機エージェント、特にプロセッサは、待機に続く命令のための全ての準備を完了できる。プロセッサの場合、トリガされる命令は、プロセッサがバスの調停を許可されるまでに、既にプロセッサのバス・インタフェース・ユニット（ＢＩＵ）内に存在する。この理由から、レジスタ・ポーリング同期機構は、アービトレーション同期機構により達成されるような、最小を達成することができない。
【０１２１】
本発明の好適な実施例の特定のフィーチャは、それがエージェント・タイプ及びそれが使用されるシステムに依存しない点である。
【０１２２】
各異なるタイプ対の同期装置が、異なる同期機構を使用する必要がある従来のシステムに

比較して、この同期方法はプロセッサ、ビヘイビュラル、Ｉ／Ｏ装置、及び非始動システム・コンポーネントにより使用され得る。

【０１２３】

本発明の方法は好適には、アービトレーション機構を有する任意のシステムにおいて使用され得る。

【０１２４】

他のテスト方法は、後述の好適なフィーチャの全てでは無しに、幾つかを有し得る。期待結果を予測可能なランダム・テスト・ジェネレータをサポートするために、次のフィーチャの全てを提供することが好ましい。

【０１２５】

本発明の好適なフィーチャ：

１．本方法は好適には、インプリメンテーションには依存しない。本方法は、プロトコル・バス・インタフェースのアーキテクチャすなわち体系記述だけに依存する。異なるコンポーネント（及びこれらのコンポーネントのインプリメンテーション）が、同一のトリガを使用できる（唯一の変更は、体系的信号名への設計信号名のマッピングである）。更に、設計ライフ・サイクル内で頻繁に発生するインプリメンテーションの内部変更は、ジェネレータ内の変更を課さない。

２．本方法は、シミュレーションの間にランタイム同期制御装置を使用するので、好適にはシミュレーション実行タイミングに依存しない。従って、本方法は、いかなる人の介入も必要としないテストをランダムに生成するために使用され得る。

３．本方法は好適には、テストされるインプリメンテーション（ＩＵＴ）の振舞いを変更しない。本方法は、テスト結果を導くために、ＩＵＴの外部のアービトレーション機構を使用する。

４．本方法は好適には"高速"である。本方法は、２つのバス事象間の"最小可能"時間遅延を有するテストを生成する。例えば、実験で使用されたバス上の２つの異なるバス・トランザクション間の最小可能時間遅延は、３バス・クロックである。この方法によれば、テストはランダムに生成され、シミュレーションにおいて、この実際の最小を達成した。これは非常に強力な利点である。なぜなら、他の同期方法は単にそれを有さないからである。例えば、プロセッサにおいて、同期がプロセッサ命令を介して達成される場合、PowerPC６ｘｘバス上における２つの異なるプロセッサ・トランザクションの実際の出現は、バス・プロトコルにより許可される最小よりも、常に多大に長い。

５．本方法は好適には"強力"である。この同期方法を使用することにより、シミュレーションにおいて事象の実際の順序を予測することが可能になる。これはランダムに生成されるテストにおいてさえ、期待結果の生成を可能にする。

６．本方法は好適には、メモリ制御装置及びＩ／Ｏ制御装置などの、命令ストリームを有さないシステム・コンポーネント上で使用され得る。

７．好適には、本方法は、命令ストリームを用いる同期から要求されるもの以外に、バス・プロトコルの理解にもとづく。これは命令ストリーム・レベルの同期にとっては無関係であるが、バス・レベルの同期にとってはそうでない、非プログラム順序の実行を行うプロセッサにとっても、好ましい。更に、デッドロックが好適には回避される。これらのフィーチャを提供する利点は、開発される同期が、広範囲のシステム及びコンポーネントに渡って使用され得ることである。

８．事象間の短時間遅延が決定論的に達成され得る。なぜなら、命令フェッチ及び他のこうした事象は一般に、実際のタイミングを予測する能力を阻み、順序の制御だけを可能にするからである。しかしながら、多くのランダムに生成されるテストのシミュレーションは、一部のテストにおいて、短い遅延を獲得する確率を増加させる。

【０１２６】

図１４は、本発明の同期制御装置が単一の同期を実行するための好適な方法の単純化されたフロー図を示し、この方法は通常、各同期に対して各サイクル繰り返される。同期制御装置のコードは、付録Ａに示される。要求される各同期に対して、すなわち各ＯＣ（順序

制御）に対して、次の整数変数が好適には維持される。
ステーション（station）：どのくらい長く待機エージェントが待機しているか？
条件（condition）：シグナリング・エージェントが発生したか？
保留ユニット（pending_unit）：待機エージェントのユニット番号
保留トランザクション番号（pending_trans_num）：保留トランザクションのトランザクション番号
シグナリング事象ユニット（signal_event_unit）：シグナリング・エージェントのユニット番号
シグナリング事象トランザクション番号（signal_event_trans_num）：シグナリング・トランザクションのトランザクション番号

【０１２７】
図１４のフロー図は、１サイクルにつき１つの同期のフローを示し、従って、フロー図は、毎同期ごとに毎サイクル繰り返される。

【０１２８】
トランザクション番号は、各エージェントの現トランザクションをモニタするために使用される専用のシミュレーション変数を記録する。例えば、PowerPCプロセッサにおいて現在使用されるインプリメンテーションは、ＣＴＲレジスタである。各プロセッサがトランザクションを実行する前に、プロセッサは通常、そのＣＴＲレジスタを増分する。ｈｏｌｄ（）ルーチンは、保留ユニットのアドレス及びデータ・バス要求、及びグラント・ラインを非活動化する。同様に、ｓｉｇｎａｌ（）ルーチンは、同一のラインの非活動化を中止する。現トランザクション番号及びアービトレーション信号として使用する実際のシミュレーション変数は、テスト・ファイル内で定義され、これが総称的な同期制御装置により解析される。

【０１２９】
図１４の方法は、テスト・シミュレーションにおいて、２つの事象が、それらの間の経過時間が最も少ない状況で発生することを自動的に保証する。この方法により克服される難題は、ハードウェア・コードがどのように振る舞うかを予測可能でないことである。なぜなら、設計者がコードを開発するとき、コードは常に変化するからである。ここで述べられ、示されるテストは、ハードウェア・インプリメンテーションとは無関係である。なぜなら、ここで述べられるツールは、シミュレーションの間に、所望の事象シーケンスが発生するための条件を設定する作用をするからである。利点はツールがハードウェア・コード振舞いを予測する必要がないことである。なぜなら、ツールはシミュレーション時に、ハードウェア振舞いからのフィードバックにより作用するからである。ここで述べられ、示される方法は、２対以上の事象及び異なる研究課題に適用されるのに十分に汎用的である。本発明の方法は、手操作により実行されることの単なる自動化ではない。なぜなら、手動方法は、異なるテストによる複数のシミュレーションを含むのに対して、本発明の方法は一般に、同一の結果を得るために、単一のテスト・シミュレーションを要求するだけであるからである。

【０１３０】
図１６は、図１１に示される同期のレジスタ方法の単純化されたフロー図である。

【０１３１】
図１７は、図１１に示される同期のイネーブル・ビット方法の単純化されたフロー図である。

【０１３２】
図１８は、図１１に示される同期の構成方法の単純化されたフロー図である。

【０１３３】
図１９は、図１１に示される同期のセマフォ方法の単純化されたフロー図である。

【０１３４】
図２０は、複数の事象を並列動作で実行する複数のエージェントを有するシステムが繰り返し実行され、複数の事象が全ての可能な順序で発生するように強制される同期方法の、

単純化されたフロー図である。
【０１３５】
付録Ａは、本発明の好適な実施例に従い構成され、図１４の同期制御方法を実行する同期制御装置の好適なソフトウェア・インプリメンテーションのコンピュータ・リストである。
【０１３６】
付録Ａの同期制御装置を、Ｃコードとインタフェースする能力を有する任意のシミュレータＸと共に使用するために、次のステップが実行され得る。
１）テストに関わるエージェントが、０から始まる実行番号を割当てられる。
２）テストに関わるエージェントがシミュレーション信号を提供し、それがサンプリングされて、エージェントの現トランザクション番号を決定する。例えば、プロセッサのレジスタの１つが、このタスクに割当てられる。この場合、プロセッサの命令ストリームが変更され、割当てられたレジスタが各トランザクション前に増分される。
３）次のアレイが、テストに関わる各エージェントに対する適切なシミュレーション信号を含むように、初期化される。
トランザクション番号：上記（２）参照
グラント・アレイ：バスへのアービトレーション要求及びグラントを制御するシミュレーション信号
４）順序制御（ＯＣ）テーブル・アレイは、要求される各同期に対して、順序制御タイプのコンポーネントを含むように初期化される。１つの可能な方法は、シミュレーションの開始前に呼び出されるＩｎｉｔ（）関数内で、アレイを初期化することである。
５）HandleOrderControl（）関数は、シミュレーション・サイクル毎に１度呼び出される。イベント・ドリブン型シミュレータでは、シミュレーション・サイクルがシステム・クロックの使用によりシミュレートされ得る。
６）シミュレータＸとインタフェースするために、３つの関数、すなわちGetSimSignal（）、StickSimSignal（）、及びUnStickSimSignal（）が提供される。これらの関数の作用はそれぞれ、シミュレーション信号の現在値を返却する；シミュレーション信号に特定値を取らせる；及び、シミュレーション信号が特定値を取るのを中止するである。
【０１３７】
或いは、付録Ａの同期制御装置は、ModelSim/VHDLシミュレータに動的にリンクされ得る。そのコンパイルは、それに従い実行される。リンケージ・フェーズの間、実行可能な同期制御装置に対して、エントリ・ポイントが定義される。エントリ・ポイントは好適には、次の内容を実行する初期化関数である。
ａ）前記ステップ（５）のセットアップ、すなわち毎クロック・サイクル後に、シミュレータにHandleOrderControl（）関数を呼び出すように指示する。
HandleOrderControl（）関数のためのＭｏｄｅｌＳｉｍ／ＶＨＤＬプロセスを生成する。
プロセスＩＤ　process＝mti_CreateProcess（name、HandleOrderControl、0）；
毎シミュレーション・サイクルごとに、シミュレータにこのプロセスを実行するように指示する。シミュレーション・サイクルはＣＬＫ信号の支援により定義され得る。プロセスは、ＣＬＫ信号がアクティブになる度に、実行され得る。
mti_Sensitize（process、mti_FindSignal（"CLK"）、MTI_EVENT）；
前記ステップ（３）及び（４）をセットアップ、すなわち要求されるシミュレーション信号及び順序制御アレイを初期化する。
HandleOrderControl（）関数で使用されるドライバ信号を生成する。
ドライバＩＤ drive_this_signal=mti_CreateDriver（signalID signal）；
HandleOrderControl（）関数で使用される全てのアレイを、この特定のシミュレーション実行のために定義される値により初期化する。
【０１３８】
HandleOrderControl（）関数が呼び出される度に、それはモデルの現状態を読出すための関数、及びそれを変更する関数を使用する。次の項目では、前記のステップ（６）をいか

に実行するかについて説明する。
GetSimSignal（）。ModelSim/VHDLでは、駆動する必要がある信号のタイプに応じて、次の関数が使用される。
mti_GetSignalValue（Signal）、または
mti_GetArraySignalValue（Signal）
StickSimSignal（）は信号を特定の値に強制する。論理に応じて、この効果を得るために、異なる使用が必要とされる。
mti_SetSignalValue（signal、value）
mti_ScheduleDriver（signal、．．．）
UnStickSimSignal（）は、信号を特定の値に強制するのを中止する。論理に応じて、この効果を得るために、異なる使用が必要とされる。使用される関数には次のものがある。
mti_SetSignalValue（signal、HIGH_IMPEDANCE/UNDEFINED/Z）
mti_ScheduleDriver（signal、．．．）
関数及びそれらの使用に関する詳細については、モデル・テクノロジ社から販売されるModelSim/VHDLユーザ・マニュアルで述べられている。

【０１３９】

本発明のソフトウェア・コンポーネントは、要求に応じて、ＲＯＭ（読出し専用メモリ）形式で実現され得る。ソフトウェア・コンポーネントは一般に、要求に応じて、従来技術の使用により、ハードウェアにより実現され得る。

【０１４０】

［表２］
本発明の好適な実施例に従い構成され、図１４の同期制御方法を実行する同期制御装置の好適なソフトウェア・インプリメンテーションのコンピュータ・リストである。

```
/*************************************************************
*  (C) Copyright IBM Corp.1990,1999                          *
*************************************************************/

/*************************************************************************/
/* 同期制御装置                                                          */
/*************************************************************************/

/*順序制御状態変数　事象後は非ゼロ*/
typedef struct
{
  int station; /*待機エージェントが待機したサイクル数*/
  int condition; /*シグナリング事象発生*/
  int signal;/*待機エージェント解放*/

/*待機エージェント・パラメータ*/
  int pending_trans_num;
  int pending_unit;

/*シグナリング事象パラメータ*/
  int signal_event_trans_num;
  int signal_event_unit;
} OrderControl;

/*シミュレーション変数*/
typedef struct
{
  char*symbol;
  int index,offset,size;
} SimSignal;
```

```
/*次のテーブルはシミュレーションの開始時にテスト・ファイルから初期化され
る*/

/*順序制御同期テーブル*/
static OrderControl oc_table[MAX_OC];
static int num_of_ocs=0;

/*順序制御の間に使用されるシミュレーション信号*/
static SimSignal trans_num[MAX_UNITS];
static SimSignal grant_dbg[MAX_UNITS];
static SimSignal grant_dbr[MAX_UNITS];
static SimSignal grant_abg[MAX_UNITS];
static SimSignal grant_abr[MAX_UNITS];
・・・

/*****************************************************************/
/****                    順序制御処理関数                       ****/
/*****************************************************************/
```

```
/*この関数はシミュレーション・サイクル毎に実行される*/
oid HandleOrderControl()
..
/*どの待機エージェントがそれらの保留トランザクションを実行準備完了かをチ
ェックする*/
for(i=0;i<num_of_ocs;i++)
  if(!oc_table[i].station)
/*この順序制御のための保留ユニットが、その保留トランザクションにまだ達し
ていない*/
{
  if(station(oc_table[i].pending_unit,oc_table[i].pending_trans_num))
  {
   oc_table[i].station=1;
   hold(&oc_table[i]);
  }
  else oc_table[i].station++;
```

```
/*シグナリング事象のチェック*/
for(i=0;i<num_of_ocs;i++)
  if(!oc_table[i].condition &&
    station(oc_table[i].signal_event_unit,oc_table[i].signal_event_trans
_num))
    /*シグナリング事象が丁度発生*/
   oc_table[i].condition=1;

/*待機中で、それらのシグナリング事象が発生したものに知らせる*/
for(i=0;i<num_of_ocs;i++)
  if(!oc_table[i].signal && oc_table[i].condition && oc_table[i].station
)
/*保留ユニットが待機中で、シグナリング事象が発生*/
  {
    oc_table[i].signal=1;
    signal(&oc_table[i]);
  }
}
int station(int unit,int transaction)
{
..
/*各ユニット内に、現トランザクション番号に割り当てられるシミュレーション
変数が存在する
ＣＰＵにおけるこのインプリメンテーションでは、１レジスタがこの目的に割り
当てられる*/
  return GetSimSignal(&trans_num[unit])=transaction;
}
void hold(OrderControl* oc)
{
```

```
/*アービトレーション要求及びグラント・ラインを非活動化*/
  StickSimSignal(&grant_abr[oc->pending_unit],1);
  StickSimSignal(&grant_abg[oc->pending_unit],1);
  StickSimSignal(&grant_dbr[oc->pending_unit],1);
  StickSimSignal(&grant_dbg[oc->pending_unit],1);
}
void signal(OrderControl* oc)
{
/*アービトレーション要求及びグラント・ラインのホールドを解除*/
  UnStickSimSignal(&grant_abr[oc->pending_unit]);
  UnStickSimSignal(&grant-abg[oc->pending_unit]);
  UnStickSimSignal(&grant_dbr[oc->pending_unit]);
  UnStickSimSignal(&grant_dbg[oc->pending_unit]);
}
```

付録に記載される特定の実施例は、本発明の極めて詳細な開示を提供することだけを意図するものであり、本発明を制限するものではない。明瞭化のために、別々の実施例の状況において述べられた本発明の様々なフィーチャは、組み合わされて、単一の実施例としても提供され得る。逆に、簡潔のために、単一の実施例の状況において述べられた本発明のフィーチャは、別々に、または任意の好適なサブ組み合わせによっても提供され得る。

【０１４１】

まとめとして、本発明の構成に関して以下の事項を開示する。

【０１４２】

（１）各々が共通バスへの少なくとも１つの経路を有する、並列動作の複数のエージェントを有するシステムを実行するステップと、
バス・アービトレーション機構を用い、所定の技法に従い前記複数のエージェント間を同期させるステップにおいて、
前記技法が、個々の前記エージェントが活動化されないことを示す場合、前記共通バスへの前記エージェントの前記少なくとも１つの経路を遮断するステップと、
一旦前記技法が、前記個々のエージェントが活動化され得ることを示すと、前記共通バスへの前記エージェントの前記少なくとも１つの経路を回復するステップと
を含む、同期させるステップと、
を含む、同期方法。

（２）前記複数のエージェントが複数のプロセッサを含む、前記（１)記載の方法。

（３）前記複数のエージェントが少なくとも１つのプロセッサ及び少なくとも１つのＩ／Ｏユニットを含む、前記（１)記載の方法。

（４）前記複数のエージェントが、少なくとも１つの中間バスを介して、前記共通バスへの経路を有する少なくとも１つのエージェントを含む、前記（１)記載の方法。

（５）前記同期させるステップが外部的に実行され、従って、各前記エージェントの内部構造とは無関係に実行される、前記（１)記載の方法。

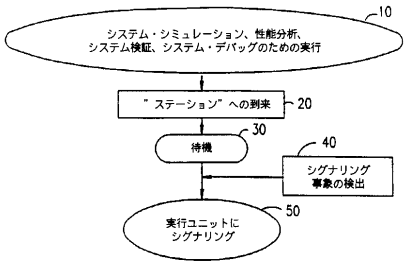
（６）前記同期させるステップが、体系的信号を強制するステップを含む、前記（１)記載の方法。

（７）前記複数のエージェントが、１つのアドレスをほぼ同時にアクセスする少なくとも２つのエージェントを含み、前記同期させるステップが、前記２つのエージェントの所定

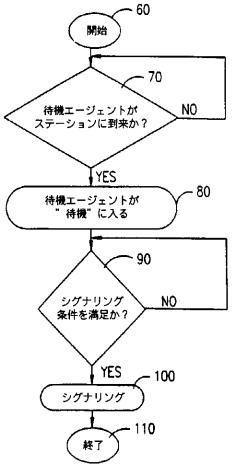
の１つが、他のエージェントが前記アドレスをアクセスする以前に、該アドレスをアクセスするように強制する、前記（１)記載の方法。
（８）前記複数のエージェントが第１及び第２のエージェントを含み、前記同期させるステップが、第１の介入時点に、前記第１及び第２のエージェント間の順序を強制するステップを含み、
前記複数のエージェントが第３及び第４のエージェントを含み、それらの少なくとも一方が前記第１及び第２のエージェントと異なり、前記同期させるステップが、前記第１の介入時点と異なる第２の介入時点に、前記第３及び第４のエージェント間の順序を強制するステップを含む、前記（１)記載の方法。
（９）前記複数のエージェントがメモリ制御装置の少なくとも１つのバッファを充填するように動作し、前記同期させるステップが、前記メモリ制御装置が前記共通バスをアクセスするのを阻止するステップを含む、前記（１)記載の方法。
（１０）前記複数のエージェントがバス・ブリッジの少なくとも１つのバッファを充填するように動作し、前記同期させるステップが、前記バス・ブリッジが前記共通バスをアクセスするのを阻止するステップを含む、前記（１)記載の方法。
（１１）各々が共通バスへの少なくとも１つの経路を有する、並列動作の複数のエージェントを有するシステムを実行するシステム・マニピュレータと、
所定の技法に従い、前記複数のエージェント間を同期させるバス・アービトレーション機構であって、
前記技法が、個々の前記エージェントが活動化されないことを示す場合、前記共通バスへの前記エージェントの前記少なくとも１つの経路を遮断するステップと、
一旦前記技法が、前記個々のエージェントが活動化され得ることを示すと、前記共通バスへの前記エージェントの前記少なくとも１つの経路を回復するステップと
を含む、バス・アービトレーション機構と
を含む、同期装置。
（１２）前記システム・マニピュレータが、前記システムをシミュレートするシステム・シミュレータを含む、前記（１１)記載の装置。
（１３）前記システム・マニピュレータが、前記システムの性能分析を実行する性能アナライザを含む、前記（１１)記載の装置。
（１４）前記システム・マニピュレータが、前記システムを検証するシステム・ベリファイヤを含む、前記（１１)記載の装置。
（１５）前記システム・マニピュレータが、前記システムをデバッグするシステム・デバッガを含む、前記（１１)記載の装置。
（１６）前記実行するステップが、前記システムをシミュレートするステップを含む、前記（１)記載の方法。
（１７）前記実行するステップが、前記システムの性能分析を実行するステップを含む、前記（１)記載の方法。
（１８）前記実行するステップが、前記システムを検証するステップを含む、前記（１）記載の方法。
（１９）前記実行するステップが、前記システムをデバッグするステップを含む、前記（１)記載の方法。
【図面の簡単な説明】
【図１】本発明の好適な実施例に従い構成され動作する、保留装置の実行を示すフロー図である。
【図２】本発明の好適な実施例に従い構成され動作する、好適な同期制御装置の単純化された仮想フロー図である。
【図３】本発明の好適な実施例に従い動作する、３つのエージェント間のランデブの単純化された図である。
【図４】本発明の好適な実施例に従い動作する、２つのエージェント間で発生する相互排他状況の単純化された図である。

【図５】ステーションへの到来が関連シグナリング事象前に発生する、トランザクション順序付けの時間体系を示す図である。
【図６】ステーションへの到来が関連シグナリング事象後に発生する、トランザクション順序付けの時間体系を示す図である。
【図７】ランデブの時間体系を示す図である。
【図８】２つの競合エージェントの同時クリティカル・アクセスが阻止される相互排他状況の時間体系を示す図である。
【図９】２つの競合エージェントに別々のアクセスが提供される相互排他状況の時間体系を示す図である。
【図１０】図５乃至図８で使用される主要解釈図式表記法を示す図である。
【図１１】図１の"待機"及び"信号"ステップに関して主に異なる、本発明の５つの異なる実施例に従い構成され動作する同期制御方法を比較した表である。
【図１２】テストされるシステムの単純化されたブロック図である。
【図１３】本発明の好適な実施例に従い構成され動作する、同期制御装置の動作の好適な方法を示す単純化されたフロー図である。
【図１４】図１１に示される同期のバス・アービトレーション方法の１サイクルの単純化されたフロー図である。
【図１５】本発明の好適な実施例に従い構成され動作する同期制御装置により実行される、アービトレーション同期のための好適な方法を示す図である。
【図１６】図１１に示される同期のレジスタ方法の単純化されたフロー図である。
【図１７】図１１に示される同期のイネーブル・ビット方法の単純化されたフロー図である。
【図１８】図１１に示される同期の構成方法の単純化されたフロー図である。
【図１９】図１１に示される同期のセマフォ方法の単純化されたフロー図である。
【図２０】複数の事象を並列動作で実行する複数のエージェントを有するシステムが繰り返し実行され、複数の事象が全ての可能な順序で発生するように強制される同期方法の、単純化されたフロー図である。

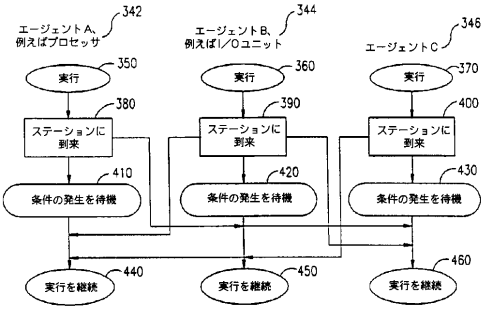
【図１】



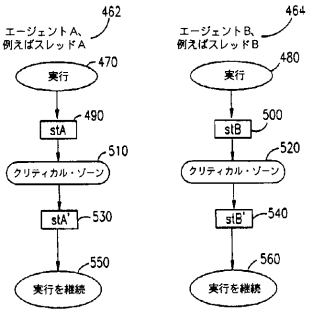
【図２】



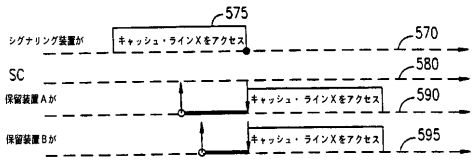
【図３】



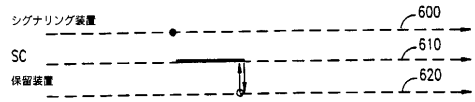
【図４】



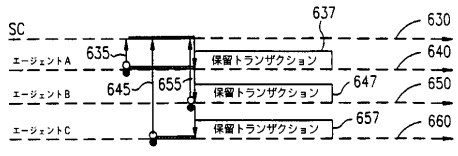
【図５】



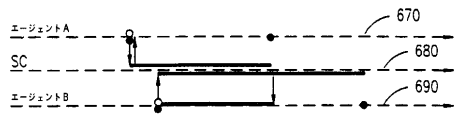
【図６】



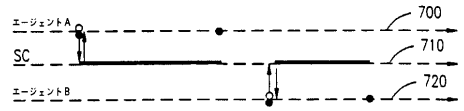
【図７】



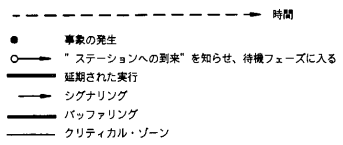
【図８】



【図９】



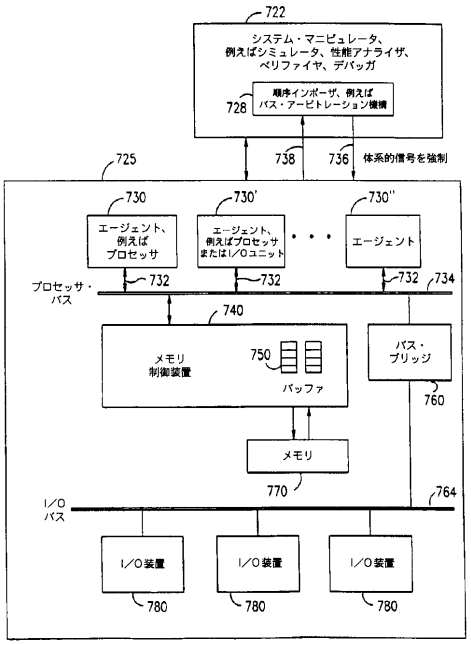
【図１０】



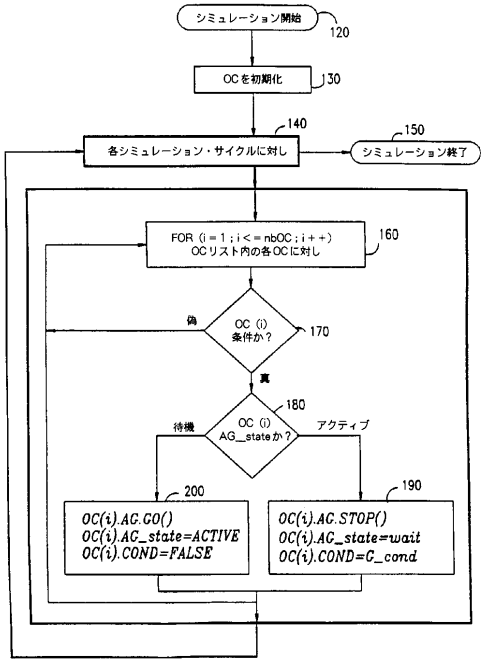
【図１１】

	待機	シグナリング	制限
レジスタ方法	CPUがレジスタ値のためにループ	SCがCPUレジスタの値を変更	プロセッサ専用
バス・アービトレーション方法	アービトレーション・グラントを待機	SCがアービトレーション制御の値を変更	技術的困難
イネーブル・ビット	イネーブル・ビット／アクションを待機	SCがイネーブル・ビット（またはアクション）の値を変更	ビヘイビュラル専用
構成	イネーブル・ビット／アクションを待機	プロセッサが装置をイネーブルにする構成サイクルを実行	構成サイクルを有するビヘイビュラル／装置
セマフォ	セマフォにおいて待機	先行プロセッサがセマフォを解除	プロセッサ専用

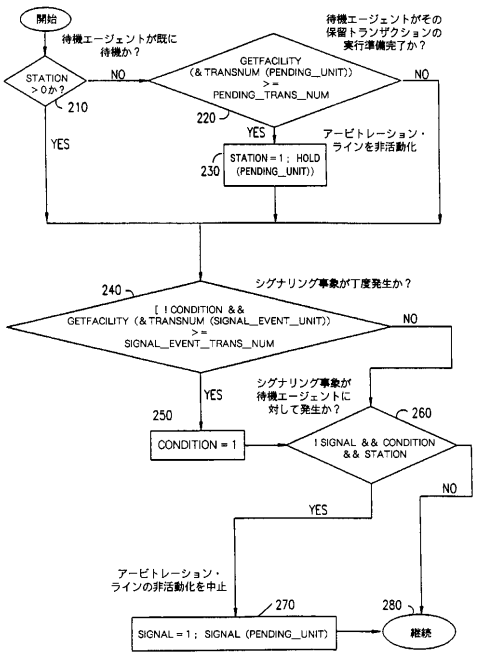
【図１２】



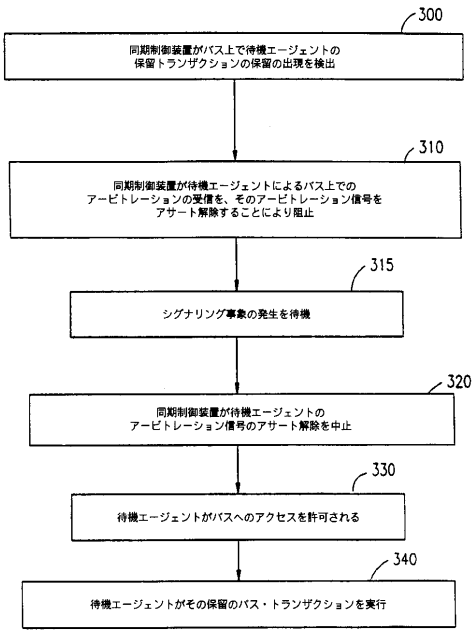
【図１３】



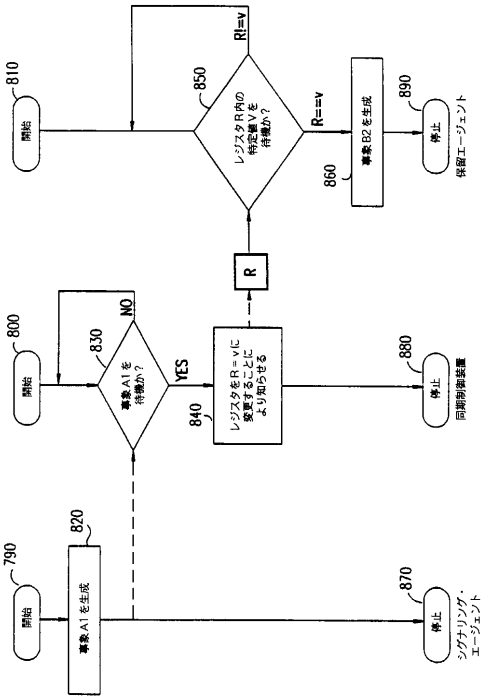
【図１４】



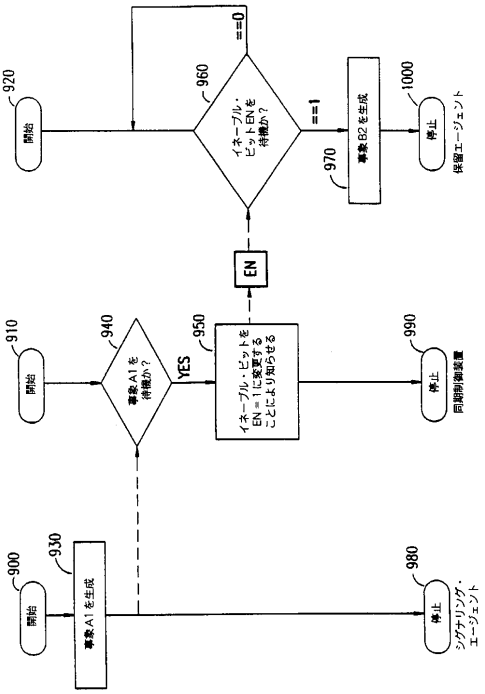
【図１５】



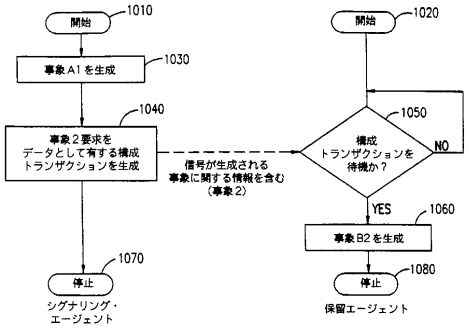
【図１６】



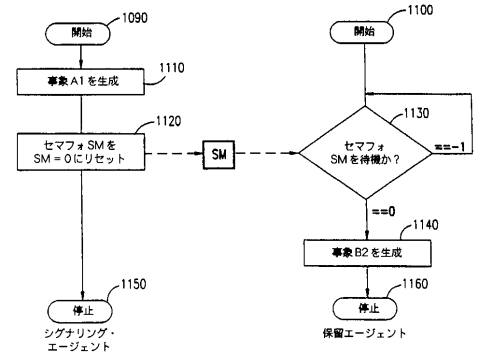
【図１７】



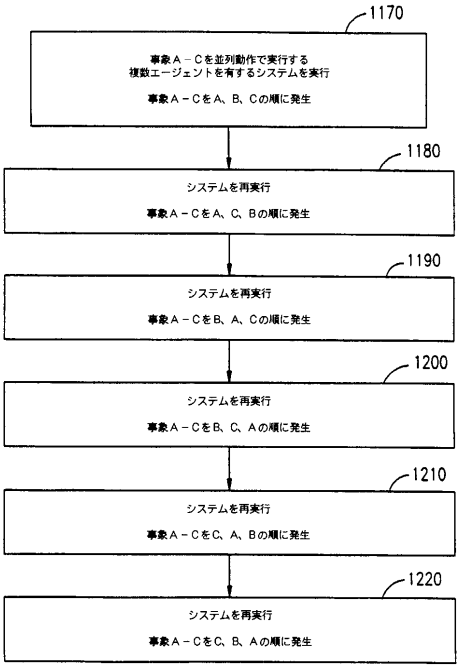
【図１８】



【図１９】



【図２０】



フロントページの続き

(72)発明者　モニカ・ファーカス
イスラエル、ハイファ、ハティシュビ・ストリート　１１２エイ
(72)発明者　ラーナン・ジェワーツマン
イスラエル、ハイファ、ソロカ・ストリート　３９
(72)発明者　ダニエル・ジスト
イスラエル、ハイファ、ペレル・ストリート　３
(72)発明者　カレン・ホルツ
イスラエル、ハイファ、ネーベ・シャーナン、コモイ・ストリート　１１

合議体
審判長　大日方　和幸
審判官　植松　伸二
審判官　小林　正明

(56)参考文献　特開平７－２１９８９２（ＪＰ，Ａ）
特開平１０－１４５４５４（ＪＰ，Ａ）
特開平１０－２１１５７（ＪＰ，Ａ）
特開平８－３１４８１７（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F13/00