(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第5595633号
(P5595633)
(45)発行日　平成26年9月24日(2014.9.24)　(24)登録日　平成26年8月15日(2014.8.15)

(51)Int.Cl.　F I
*G06F 11/28 (2006.01)*　G06F 11/28 340C

請求項の数 9　(全 13 頁)

(21)出願番号　特願2007-45577(P2007-45577)
(22)出願日　平成19年2月26日(2007.2.26)
(65)公開番号　特開2008-210107(P2008-210107A)
(43)公開日　平成20年9月11日(2008.9.11)
審査請求日　平成21年10月9日(2009.10.9)
審判番号　不服2012-3814(P2012-3814/J1)
審判請求日　平成24年2月28日(2012.2.28)

(73)特許権者　504378124
スパンション　エルエルシー
アメリカ合衆国　カリフォルニア州　９４０８８－３４５３　サニーベイル　デグウィン　ドライブ　９１５
(74)代理人　100079108
弁理士　稲葉　良幸
(74)代理人　100109346
弁理士　大貫　敏史
(72)発明者　立岡　真人
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内
(72)発明者　池　敦
神奈川県川崎市中原区上小田中４丁目１番１号　富士通株式会社内
最終頁に続く

(54)【発明の名称】シミュレーション方法及びシミュレーション装置

(57)【特許請求の範囲】
【請求項１】
復元ポイント設定手段が、並列処理を行う複数のコアモデルのいずれかにおいてブレイク命令に連動して復元ポイントが設定されると、ブレイク命令が実行されていない他のコアモデルにおいても復元ポイントを設定することで、前記複数のコアモデル間で、前記復元ポイントを、シミュレーション装置上での同一命令数で設定し、
前記復元ポイントにおける前記複数のコアモデルの各コアモデルの内部情報データまたは前記各コアモデルがアクセスしたメモリデータを含み、前記復元ポイントにおける前記複数のコアモデルの状態を再現するための情報を記憶手段に記憶することを特徴とするシミュレーション方法。
【請求項２】
実行した一定の命令数をもとに、前記複数のコアモデル間で同時に定期的に前記復元ポイントを設定することを特徴とする請求項１記載のシミュレーション方法。
【請求項３】
前記記憶手段は、所定数の前記復元ポイントにおける前記情報を記憶すると、過去の前記情報を新しい前記復元ポイントにおける情報で順次上書きしていくことを特徴とする請求項１または２に記載のシミュレーション方法。
【請求項４】
ブレイク命令と連動して前記復元ポイントを設定することを特徴とする請求項１乃至３のいずれか一項に記載のシミュレーション方法。

【請求項５】
ユーザの設定により、任意時点で前記復元ポイントを設定することを特徴とする請求項１乃至４のいずれか一項に記載のシミュレーション方法。
【請求項６】
ユーザの設定により、前記複数のコアモデル間で同時に任意のタイミングで前記復元ポイントを設定することを特徴とする請求項１乃至５のいずれか一項に記載のシミュレーション方法。
【請求項７】
前記情報を圧縮して前記記憶手段に記憶することを特徴とする請求項１乃至６のいずれか一項に記載のシミュレーション方法。
【請求項８】
前記複数のコアモデルは、マルチプロセッサにおけるプロセッサ、マルチコアプロセッサにおけるプロセッサコアまたは、マルチマスタスレーブにおけるマスタのシミュレーションモデルであることを特徴とする請求項１乃至７のいずれか一項に記載のシミュレーション方法。
【請求項９】
並列処理を行う複数のコアモデルのいずれかにおいてブレイク命令に連動して復元ポイントが設定されると、ブレイク命令が実行されていない他のコアモデルにおいても復元ポイントを設定することで、前記複数のコアモデル間で、前記復元ポイントを、シミュレーション装置上での同一命令数で設定する復元ポイント設定手段と、
前記復元ポイントにおける前記複数のコアモデルの各コアモデルの内部情報データまたは前記各コアモデルがアクセスしたメモリデータを含み、前記復元ポイントにおける前記複数のコアモデルの状態を再現するための情報を記憶する記憶手段と、
を有することを特徴とするシミュレーション装置。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明はシミュレーション方法及びシミュレーション装置に関し、特にマルチコアプロセッサ、マルチプロセッサまたはマルチマスタスレーブに適したソフトウェア開発用のシミュレーション方法及びシミュレーション装置に関する。
【背景技術】
【０００２】
従来、ＳｏＣ（System on a Chip）のソフトウェア開発などにシミュレータが用いられている。シミュレーション時にプロセッサの状態を再現するには再実行すればよいが、大規模なソフトウェアをシミュレータで開発するには再実行時間が長くなり、開発時間が膨大なものになってしまう。
【０００３】
そこで、ある時点で復元ポイントを設定して、プロセッサの状態の再現に必要な情報（メモリ、レジスタ、バッファ、その他プロセッサ内部の情報、周辺回路内のデータやステータス情報など）を保存し、後から保存した情報を読み出してプロセッサの動作を再現する技術が知られている（例えば、特許文献１参照）。
【０００４】
復元ポイントで情報を保存する技術は、近年登場したマルチプロセッサ、または複数のＣＰＵ（Central Processing Unit）コアを含むマルチコアプロセッサ用のソフトウェアを開発する際にも重要となってきている。
【０００５】
図８は、復元ポイントを用いた従来のシミュレーション方法をシングルスレッドで動作するマルチコアプロセッサに適用した場合の処理の流れを示す図である。
以下では、２つのプロセッサコアに対応するコアモデルＰＥ０，ＰＥ１を有するマルチコアモデルのシミュレーションを例にして説明する。

【０００６】
なお、シミュレーションには、命令セットシミュレータ（ＩＳＳ：Instruction Set Simulator）を用いる。
シミュレーションを開始すると、実行判定を行う（ステップＳ８０）。実行する場合には、まず、コアモデルＰＥ０，ＰＥ１のローカルな処理を実行する前に、コアモデルＰＥ０，ＰＥ１の両方で共有する周辺機器などにおける復元ポイント（共有復元ポイント）を設定してその状態（周辺機器が使用しているメモリの情報など）を保存する（ステップＳ８１）。
【０００７】
次に、コアモデルＰＥ０で復元ポイントを設定してコアモデルＰＥ０の状態を保存した後（ステップＳ８２）、コアモデルＰＥ０の命令実行を行う（ステップＳ８３）。
ＩＳＳにおいて命令実行はｅｘｅ＿ｌｏｏｐと呼ばれる以下のような関数で行われる。
【０００８】
図９は、ｅｘｅ＿ｌｏｏｐ関数の処理を示すフローチャートである。
ここでは、ｅｘｅ＿ｌｏｏｐ関数に実行命令数（実行ステップ）を指定して１命令毎実行している例を示している。
【０００９】
なお、ｅｘｅ＿ｌｏｏｐ内では命令数はアーキテクチャに依存するので１命令でなく複数命令を同時実行してもかまわない。ここでは簡略化のため、１命令を１ループとしている。
【００１０】
まず、ローカルな実行ステップ（ＬＳＥ）を０とする（ステップＳ８２１）。次に、ＬＳＥが、ユーザが設定したステップ数（ＵＳ）と等しいか否かを判定する（ステップＳ８２２）。判定の結果、等しければコアモデルＰＥ０の命令実行を終了し、等しくなければ、命令フェッチを行い（ステップＳ８２３）、命令をデコードして実行する（ステップＳ８２４）。その後、ＬＳＥをインクリメントし（ステップＳ８２５）、例外処理を行った後（ステップＳ８２６）、ステップＳ８２２からの処理を繰り返す。
【００１１】
図８に戻り、コアモデルＰＥ０の命令実行が終了すると、続いてコアモデルＰＥ１で復元ポイントを設定してコアモデルＰＥ１の状態を保存し（ステップＳ８４）、図９と同様にコアモデルＰＥ１の命令実行を行う（ステップＳ８５）。その後ステップＳ８０に戻り、上記の各処理を繰り返す。
【００１２】
図１０は、マルチコアプロセッサのシングルスレッド動作の例を示す図である。
シングルスレッド動作の場合、コアモデルＰＥ０，ＰＥ１は処理単位（図９のＵＳ）毎の命令実行を、例えば交互に繰り返している。また、コアモデルＰＥ０，ＰＥ１の各処理の前には復元ポイントＡ０，Ｂ０，Ｃ０，Ａ１，Ｂ１，Ｃ１が設定されており、そこでの各コアＰＥ０，ＰＥ１の状態の保存が行われている。
【特許文献１】特開平８－１８５３４１号公報
【発明の開示】
【発明が解決しようとする課題】
【００１３】
しかし、従来のシミュレーション方法は、シングルコアを対象にしているものが主であり、マルチコアプロセッサやマルチプロセッサまたはマルチマスタスレーブに適したものではなかった。例えば、図８～図１０で示したようなシングルスレッドに対応した復元ポイントでの状態の保存の仕方では、複数のコアモデルが並列処理したときに発生するデッドロックや、Ｉ／Ｏ競合、共有変数の多重アクセスなどを発見することができなかった。
【００１４】
本発明はこのような点に鑑みてなされたものであり、並列処理を行うマルチコアプロセッサまたはマルチプロセッサまたはマルチマスタスレーブに適用するソフトウェア開発に

適したシミュレーション方法及びシミュレーション装置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】
【００１５】
本発明者らは、復元ポイント設定手段が、並列処理によりスレッドを実行する複数のコアモデル間で、復元ポイントを同時に設定し、前記復元ポイントにおける前記コアモデルの状態を再現するための情報を記憶手段に記憶することを特徴とするシミュレーション方法を提案する。
【発明の効果】
【００１６】
本発明によれば、複数のコアモデル間で復元ポイントを同時に設定し、各コアモデルの状態を保存できるので、マルチコアプロセッサ（または、マルチプロセッサもしくはマルチマスタスレーブ）用の並列処理を行うソフトウェアの開発において、デバッグで重要なデッドロックやＩＯ競合、共有変数の多重アクセスをシミュレーションで容易に発見することができる。
【発明を実施するための最良の形態】
【００１７】
以下、本発明の実施の形態を図面を参照して詳細に説明する。
図１は、本実施の形態のシミュレーション方法の概要を説明する図である。
本実施の形態のシミュレーション方法は、マルチコアプロセッサ用の並列処理を行うソフトウェアを実機実装する前に、シミュレータにより検証するものである。
【００１８】
コアモデルＰＥ０，ＰＥ１は、実装するマルチコアプロセッサの各ＣＰＵコアに対応するシミュレーションモデルである。以下では、２つのコアモデルＰＥ０，ＰＥ１を用いた場合について説明するが、この数に限定されるものではない。
【００１９】
シミュレーションの際には、図１のように、コアモデルＰＥ０，ＰＥ１に対して、ＩＳＳを用いて、それぞれスレッドを並列処理させる。横軸は実行ステップ数である。
シミュレータは、所定の実行ステップ数（またはシミュレータサイクル数）毎定期的に、コアモデルＰＥ０，ＰＥ１に対して両方同時に復元ポイントを設定する。図１の例では、復元ポイントＡ，Ｂ，Ｃを順番に設定している。そして、この復元ポイントＡ，Ｂ，Ｃでの各コアモデルＰＥ０，ＰＥ１の状態を再現するために必要な情報を、記憶手段に記憶する（以下、単にコアモデルの状態を保存するという場合もある）。
【００２０】
ここで保存する情報は、各コアモデルＰＥ０，ＰＥ１が、実行する前のメモリデータや、レジスタとコアモデルＰＥ０，ＰＥ１の内部情報データなどである。なお、メモリデータは、プログラムデータ領域の全領域など、その時点でコアモデルＰＥ０，ＰＥ１がアクセスした領域を全て保存対象とする。また、コアモデルＰＥ０，ＰＥ１の内部情報データは、コアモデルＰＥ０，ＰＥ１内のレジスタやデータキャッシュ、命令キャッシュ、バッファデータなどである。
【００２１】
復元ポイントＡ，Ｂ，Ｃでは、それぞれ異なる記憶領域が使用される。復元ポイントＣまでの状態の保存が終わると、次は再び復元ポイントＡが設定され、過去の復元ポイントＡの状態が上書きされる。これにより、図１に示すように、右端の復元ポイントＣの状態を保存した時点を現在とすると、現在復帰できる復帰可能域は、復元ポイントＡまでとなる。しかし、このように記憶領域を限定することで、保存する情報量を削減でき、コンピュータのメモリを節約できる。なお、定期的な復元ポイントの数は３つに限定されるものではない。
【００２２】
ところで、復元ポイントは、上記のような定期的なもの以外に、あるコアモデル（例えばＰＥ０）の命令実行中に図１のようにブレイク命令が設定されている場合、そのブレイ

ク命令と連動した復元ポイントを、他のコアモデル（例えばＰＥ１）にも同時に設定して、そのときの状態を保存するようにしてもよい。

【００２３】

また、復元ポイントは、ユーザが任意の位置に設定するようにしてもよい。

上記のように復元ポイントを複数のコアモデルで同時に設定して、そのときの状態を保存することで、並列処理を行うソフトウェア開発の際のデバッグ時に、デッドロックやＩ／Ｏ共有変数の多重アクセスを見つけることが可能となるほか、並列プログラムのタイミング性能の解析や最適化に役立つ。

【００２４】

以下、本実施の形態のシミュレーション方法の詳細を説明する。

図２は、シミュレーションを実施するシミュレーション装置のハードウェア構成例である。

【００２５】

シミュレーション装置１０は例えばコンピュータであり、マルチコアＣＰＵ１１、ＲＯＭ（Read Only Memory）１２、ＳＤＲＡＭ（Synchronous Dynamic Random Access Memory）１３、ＨＤＤ（Hard Disk Drive）１４、グラフィック処理部１５、入力Ｉ／Ｆ（Interface）１６などによって構成され、これらはバス１７を介して相互に接続されている。

【００２６】

ここで、マルチコアＣＰＵ１１は、ＲＯＭ１２や、ＨＤＤ１４に格納されているプログラムや、各種データに応じて各部を制御し、複数のコアモデル間で同時に復元ポイントを設定する処理や、状態を記憶する記憶領域（記憶デバイス）を選択する処理など、前述した本実施の形態のシミュレーション機能を実行する。

【００２７】

ＲＯＭ１２は、マルチコアＣＰＵ１１が実行する基本的なプログラムやデータを格納している。

ＳＤＲＡＭ１３は、マルチコアＣＰＵ１１が実行途中のプログラムや、演算途中のデータを格納している。また、復元ポイントにおける各コアモデルの状態を保存する。

【００２８】

ＨＤＤ１４は、マルチコアＣＰＵ１１が実行するＯＳ（Operation System）や、各種アプリケーションプログラム、各種データを格納する。

グラフィック処理部１５には、表示装置として例えば、ディスプレイ１５ａが接続されており、マルチコアＣＰＵ１１からの描画命令に従って、ディスプレイ１５ａ上に、シミュレーション実行時の各種情報をユーザに提示する。

【００２９】

入力Ｉ／Ｆ１６には、マウス１６ａやキーボード１６ｂなどの入力装置が接続されており、ユーザにより入力された情報を受信し、バス１７を介してマルチコアＣＰＵ１１に伝送する。

【００３０】

上記のようなシミュレーション装置１０を用いて、マルチコアシミュレータが以下のようにソフトウェアで実現される。

図３は、マルチコアシミュレータの構成例である。

【００３１】

マルチコアシミュレータは、ｍａｉｎ関数で表現され、コアモデルＰＥ０，ＰＥ１，…，ＰＥｎ、メモリの他、Ｉ／Ｏモデルである ＩＲＣ（InterRupt Controller）、ＵＡＲＴ（Universal Asynchronous Receiver Transmitter）、ＤＭＡ（Direct Memory Access）、タイマなどで構成される。

【００３２】

このようなマルチコアシミュレータを構成するために、図２のマルチコアＣＰＵ１１は、ｎ以上のコア数のＣＰＵコアを有していることが好ましい。

次に、上記のようなシミュレーション装置１０、マルチコアシミュレータを用いたシミ

ュレーション方法を説明する。

【００３３】

図４は、定期的な復元ポイントの設定処理を示すフローチャートである。

マルチコアＣＰＵ１１は、実行判定を行う（ステップＳ１０）。実行する場合には、まず、全てのコアモデルＰＥ０～ＰＥｎで共有するＩ／Ｏモデルなどにおける復元ポイント（共有復元ポイント）を設定して、その状態をＳＤＲＡＭ１３または、マルチコアＣＰＵ１１のキャッシュメモリ（図示せず）に保存する（ステップＳ１１）。

【００３４】

次に、コアモデルＰＥ０～ＰＥｎ毎に復元ポイントを設定して状態を保存する処理を行う（ステップＳ１２－０，Ｓ１２－１，…，Ｓ１２－ｎ）。ここでの処理は、ステップＳ１２－ｎの処理部分に詳細に図示している（ステップＳ１２－０，Ｓ１２－１なども同様である）。

【００３５】

まず、プログラムカウンタ値ＰＣが定期の復元ポイントの値Ｓと同一か否かを判断して（ステップＳ１２ａ）、同一であれば各コアモデルＰＥ０～ＰＥｎの状態を、ＳＤＲＡＭ１３または、マルチコアＣＰＵ１１のキャッシュメモリに保存する。そして、復元ポイントのシリアル番号ＧＣＰｎｕｍｂｅｒ〔ｎ〕〔ｘ〕にプログラムカウンタ値ＰＣを入力する（ステップＳ１２ｂ）。

【００３６】

なお、“〔ｎ〕〔ｘ〕”はコアモデルＰＥｎのｘ番目の復元ポイントであることを示している。保存した情報は、上記のようなコア番号（１～ｎ）を含む、復元ポイントのシリアル番号（＝復元ポイントのプログラムカウンタ値ＰＣ）と対応付けられて管理される。

【００３７】

続いて、定期の復元ポイントの値Ｓに、次の復元ポイントの値Ｓｎｅｘｔを入力するとともに、シリアル番号ＧＣＰｎｕｍｂｅｒ〔ｎ〕〔ｘ〕の変数ｘをプラス１する（ステップＳ１２ｃ）。Ｓｎｅｘｔは、例えば配列で与えられており、所定の実行ステップ間隔毎の値（例えば、１００，２００，３００，…）が設定されている。

【００３８】

ステップＳ１２ａで、プログラムカウンタ値ＰＣが復元ポイントの値Ｓと同一でない場合、あるいはステップＳ１２ｃの後、各コアモデルＰＥ０～ＰＥｎによる命令実行が行われる（ステップＳ１３－０～Ｓ１３－ｎ）。ここでの処理は、図９で示した処理と同一である。

【００３９】

ステップＳ１３－０～Ｓ１３－ｎの処理が終了すると、ステップＳ１０の処理に戻り上記の処理を繰り返す。

このような処理により、定期的な復元ポイントをコアモデルＰＥ０～ＰＥｎで同時に設定することができ、復元ポイントでの状態を保存することができる。復元時には、シリアル番号ＧＣＰｎｕｍｂｅｒ〔ｎ〕〔ｘ〕を指定して、その状態をＳＤＲＡＭ１３などから読み出すことで、復元したい復元ポイントを特定してコアモデルＰＥ０～ＰＥｎの状態を再現できる。

【００４０】

次に、コアモデルＰＥ０～ＰＥｎの命令実行中にブレイクが設定されたときに連動して復元ポイントを設定する処理を説明する。

図５は、ブレイクが設定された際に、復元ポイントを設定して状態を保存する処理を示すフローチャートである。

【００４１】

ＩＳＳによる命令実行では、ｅｘｅ＿ｌｏｏｐ関数に実行命令数（実行ステップ）を指定して例えば１命令毎実行する。

なお、ｅｘｅ＿ｌｏｏｐ内では命令数はアーキテクチャに依存するので１命令でなく複数命令を同時実行してもかまわない。ここでは簡略化のため、１命令を１ループとしてい

る。
【００４２】
ここでは、コアモデルＰＥ０の命令実行中の処理を示している。他のコアモデルＰＥ１～ＰＥｎの動作も同様である。
まず、ローカルな実行ステップ（ＬＳＥ）を０とする（ステップＳ２０）。次に、ＬＳＥが、ユーザが設定したステップ数（ＵＳ）と等しいか否かを判定する（ステップＳ２１）。判定の結果、等しければコアモデルＰＥ０の命令実行を終了し、等しくなければ、命令フェッチを行い（ステップＳ２２）、命令をデコードする（ステップＳ２３）。
【００４３】
次に、ブレイク命令が設定されているか否かを判定する（ステップＳ２４）。コアモデルＰＥ０に対してブレイク命令が設定されている場合には、ステップＳ２５の処理に進み、ブレイク命令が設定されていない場合にはステップＳ２６の処理に進む。
【００４４】
ステップＳ２５の処理では、ブレイク命令と連動して復元ポイントを設定するか否かを示すフラグＣＰＦの状態を判定し、オンであればフラグＣＰＦを“ａｌｌ”とする（ステップＳ２７）。このフラグＣＰＦは、全てのコアモデルＰＥ０～ＰＥｎで参照可能なフラグ変数であり、ここでのステップＳ２７での変更は、他のコアモデルＰＥ１～ＰＥｎに通知される。
【００４５】
続いてコアモデルＰＥ０の状態を保存し、シリアル番号ＧＣＰｎｕｍｂｅｒ〔０〕〔ｘ〕にプログラムカウンタ値ＰＣを入力して（ステップＳ２８）、ステップＳ２９の処理に進む。ここで“〔０〕〔ｘ〕”は、コアモデルＰＥ０のｘ番目の復元ポイントであることを示している。
【００４６】
一方、ブレイク命令が設定されていない場合、ステップＳ２６の処理では、フラグＣＰＦが“ａｌｌ”か否かを判定し、“ａｌｌ”である場合には、何れかのコアモデルＰＥ１～ＰＥｎで復元ポイントが設定されたことを示しているので、前述のステップＳ２８の処理に進み、モデルコアＰＥ０においても状態を保存する。
【００４７】
なお、フラグＣＰＦの設定は、初期設定で決めたり、ソフトウェア的に別のプログラムで設定する場合や、ブレイク命令の際にユーザがＧＵＩ（Graphical User Interface）などで設定する場合、ブレイク命令に含める場合などがある。
【００４８】
ステップＳ２６の処理で、フラグＣＰＦが“ａｌｌ”でない場合、またはステップＳ２５の処理でフラグＣＰＦがオンでないと判定された場合には、ステップＳ２９の処理に進み、デコードされた命令を実行する。その後、ＬＳＥをインクリメントし（ステップＳ３０）、例外処理を行った後（ステップＳ３１）、ステップＳ２１からの処理を繰り返す。
【００４９】
上記のようなフラグＣＰＦを用いることで、複数のコアモデルＰＥ０～ＰＥｎのいずれかで復元ポイントが設定された場合、それと同時に他のコアモデルＰＥ０～ＰＥｎでも復元ポイントを設定して状態を保存することができる。
【００５０】
なお、上記では、ブレイク命令と連動して復元ポイントを設定した場合について説明したが、命令のステップ実行中に、ユーザがＧＵＩなどで任意に復元ポイントを設定するようにしてもよい。例えば、任意に復元ポイントが設定された場合、上記のフラグＣＰＦを用いて、全てのコアモデルＰＥ０～ＰＥｎで同時に復元ポイントを設定し、状態を保存することができる。
【００５１】
また、フラグＣＰＦの代わりに、あるコアモデルＰＥ０～ＰＥｎで復元ポイントを設定した実行ステップ数やサイクル数を、他のコアモデルＰＥ０～ＰＥｎに通知して同時に復

元ポイントを設定して状態を保存するようにしてもよい。

【００５２】

また、任意の復元ポイントは、各コアモデルＰＥ０～ＰＥｎの命令実行の前に設定してもよい。

図６は、任意の復元ポイントを設定する際の処理を示すフローチャートである。

【００５３】

ステップＳ４０、Ｓ４１は、図４のステップＳ１０、Ｓ１１の処理と同様である。ステップＳ４１の後、ユーザからの任意の復元ポイントの設定があったか否かを判定し（ステップＳ４２）、復元ポイントが設定された場合には、コアモデルＰＥ０～ＰＥｎの状態を保存し（ステップＳ４３）、ステップＳ４４－０，Ｓ４４－１，…，Ｓ４４－ｎの処理に進む。任意の復元ポイントが設定されない場合には、そのままステップＳ４４－０～Ｓ４４－ｎの処理に進む。ステップＳ４４－０～Ｓ４４－ｎは定期的な復元ポイントの設定及び状態保存処理で、図４のステップＳ１２－０～Ｓ１２－ｎの処理と同じである。その後、各コアモデルＰＥ０～ＰＥｎの命令実行処理を行い（ステップＳ４５－０，Ｓ４５－１，…，Ｓ４５－ｎ）、ステップＳ４０からの処理を繰り返す。

【００５４】

図６のように、任意の復元ポイントは、定期的な復元ポイントの設定と組み合わせて設定することができる。図示は省略するが、同様に、図５で示したようなブレイク命令と連動する復元ポイントの設定と、定期的な復元ポイントまたは任意の復元ポイントの設定を組み合わせるようにしてもよい。

【００５５】

上記のように設定した各種の復元ポイントで、コアモデルＰＥ０～ＰＥｎの状態を再現したい場合には、図３で示したようなシミュレータ内のコアモデルＰＥ０～ＰＥｎの内部データやメモリまたはＩ／Ｏモデル内の情報を全て破棄して、復帰したい復元ポイントの情報を、ＳＤＲＡＭ１３などから読み出し、シミュレータ内の情報を上書きする。

【００５６】

なお、上記の各種の復元ポイントで保存する情報を、圧縮してから保存するようにしてもよい。

また、上記では、アクセス速度の面から復元ポイントでは情報をＳＤＲＡＭ１３やキャッシュに保存するとして説明したが、メモリ消費が大きいとマルチコアＣＰＵ１１が判断すれば、ＨＤＤ１４に保存することも可能である。

【００５７】

また、本実施の形態のシミュレーション方法では、全てのコアモデルＰＥ０～ＰＥｎに同時に復元ポイントを設定することを特徴としているが、ユーザの所望により、コアモデルＰＥ０～ＰＥｎに個別に復元ポイントを設定して、状態を保存するようにしてもよい。

【００５８】

また、コアモデルＰＥ０～ＰＥｎの例として、マルチコアプロセッサのプロセッサコアを例にして説明したが、マルチプロセッサのプロセッサまたはマルチマスタスレーブとマスタ（ＣＰＵやＤＭＡ）としてもよい。

【００５９】

例として、マルチマスタスレーブに関して、定期の復元ポイントを保存する際の処理を説明する。

図７は、マルチマスタスレーブのマスタをコアモデルで示した際の定期の復元ポイントの設定処理を示す図である。

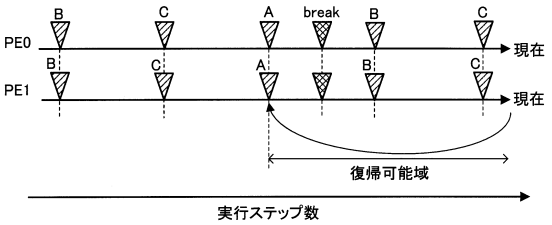
【００６０】

まず実行判定を行い（ステップＳ５０）、実行する場合には、まず、マスタ（ＣＰＵまたはＤＭＡ）のシミュレーションモデルであるコアモデルＰＥ０～ＰＥｎで共有するＩ／Ｏモデル（スレーブ）などにおける復元ポイント（共有復元ポイント）を設定して、その状態をＳＤＲＡＭ１３または、マルチコアＣＰＵ１１のキャッシュメモリ（図示せず）に保存する（ステップＳ５１）。

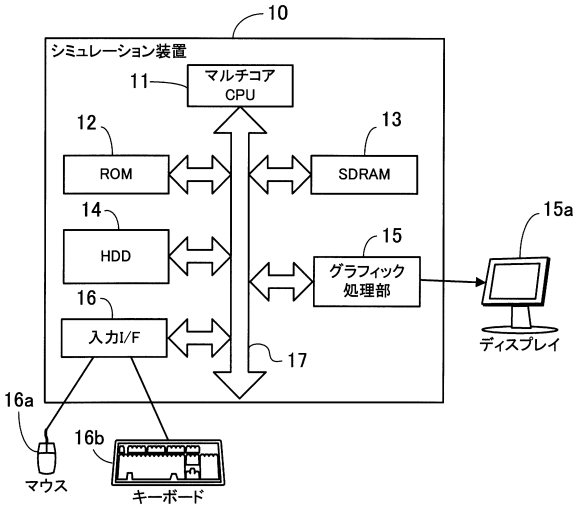
【００６１】
次に、図４で示したようなコアモデルＰＥ０～ＰＥｎ毎に定期の復元ポイントを設定してＰＥ０の状態を保存する処理を行う（ステップＳ５２－０，Ｓ５２－１，…，Ｓ５２－ｎ）。
【００６２】
続いて、各コアモデルＰＥ０～ＰＥｎの命令実行を行う（ステップＳ５３－０，Ｓ５３－１，…，Ｓ５３－ｎ）。このとき、例えば、コアモデルＰＥｎのスレーブとなっているＩ／Ｏモデルがあった場合、コアモデルＰＥｎはスレーブをコールし、スレーブを実行させる。スレーブの命令実行の終了後には、スレーブを起動したマスタにバック（リターン）する（ステップＳ５４）。
【００６３】
マルチマスタスレーブに対応したコアモデルを用いた、ブレイク命令に連動した復元ポイントの設定や、ユーザの任意で復元ポイントの設定も、マルチコアプロセッサの場合と同様である。
【００６４】
（付記１）　復元ポイント設定手段が、並列処理によりスレッドを実行する複数のコアモデル間で、復元ポイントを同時に設定し、
前記復元ポイントにおける前記コアモデルの状態を再現するための情報を記憶手段に記憶することを特徴とするシミュレーション方法。
【００６５】
（付記２）　実行した一定の命令数あるいはサイクル数をもとに、複数の前記コアモデル間で同時に定期的に前記復元ポイントを設定することを特徴とする付記１記載のシミュレーション方法。
【００６６】
（付記３）　前記記憶手段は、所定数の前記復元ポイントにおける前記情報を記憶すると、過去の前記情報を新しい前記復元ポイントにおける情報で順次上書きしていくことを特徴とする付記１または２に記載のシミュレーション方法。
【００６７】
（付記４）　複数の前記コアモデルのいずれかにおいて、命令の実行中に復元ポイントが設定された場合、連動して他の前記コアモデルにおいても前記復元ポイントを設定することを特徴とする付記１乃至３のいずれか一項に記載のシミュレーション方法。
【００６８】
（付記５）　複数の前記コアモデルで参照可能なフラグ変数によって、ある前記コアモデルで前記復元ポイントが設定されたことを他の前記コアモデルに通知することを特徴とする付記４記載のシミュレーション方法。
【００６９】
（付記６）　ブレイク命令と連動して前記復元ポイントを設定することを特徴とする付記４または５記載のシミュレーション方法。
（付記７）　ユーザの設定により、任意時点で前記復元ポイントを設定することを特徴とする付記４乃至６のいずれか一項に記載のシミュレーション方法。
【００７０】
（付記８）　ユーザの設定により、複数の前記コアモデル間で同時に任意のタイミングで前記復元ポイントを設定することを特徴とする付記１乃至７のいずれか一項に記載のシミュレーション方法。
【００７１】
（付記９）　前記情報を圧縮して前記記憶手段に記憶することを特徴とする付記１乃至８のいずれか一項に記載のシミュレーション方法。
（付記１０）　前記複数のコアモデルは、マルチプロセッサにおけるプロセッサ、マルチコアプロセッサにおけるプロセッサコアまたは、マルチマスタスレーブにおけるマスタのシミュレーションモデルであることを特徴とする付記１乃至９のいずれか一項に記載の

シミュレーション方法。
【００７２】
（付記１１） 記憶した前記情報は、前記コアモデルのコア番号及び前記復元ポイントが設定されたときのプログラムカウンタ値と対応付けられて管理されていることを特徴とする付記１乃至１０のいずれか一項に記載のシミュレーション方法。
【００７３】
（付記１２） 並列処理によりスレッドを実行する複数のコアモデル間で、復元ポイントを同時に設定する復元ポイント設定手段と、
前記復元ポイントにおける前記コアモデルの状態を再現するための情報を記憶する記憶手段と、
を有することを特徴とするシミュレーション装置。
【図面の簡単な説明】
【００７４】
【図１】本実施の形態のシミュレーション方法の概要を説明する図である。
【図２】シミュレーションを実施するシミュレーション装置のハードウェア構成例である。
【図３】マルチコアシミュレータの構成例である。
【図４】定期的な復元ポイントの設定処理を示すフローチャートである。
【図５】ブレイクが設定された際に、復元ポイントを設定して状態を保存する処理を示すフローチャートである。
【図６】任意の復元ポイントを設定する際の処理を示すフローチャートである。
【図７】マルチマスタスレーブのマスタをコアモデルで示した際の定期の復元ポイントの設定処理を示す図である。
【図８】復元ポイントを用いた従来のシミュレーション方法をシングルスレッドで動作するマルチコアプロセッサに適用した場合の処理の流れを示す図である。
【図９】ｅｘｅ＿ｌｏｏｐ関数の処理を示すフローチャートである。
【図１０】マルチコアプロセッサのシングルスレッド動作の例を示す図である。
【符号の説明】
【００７５】
ＰＥ０，ＰＥ１ コアモデル
Ａ，Ｂ，Ｃ 復元ポイント

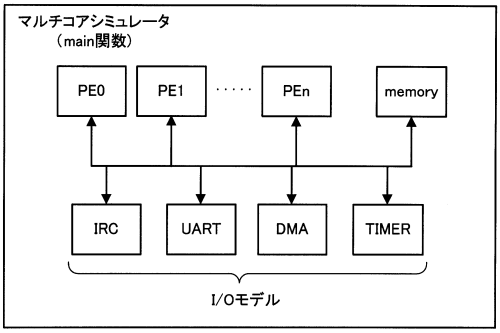
【図１】


PE0
PE1
B
C
A
break
B
C
現在
現在
復帰可能域
実行ステップ数

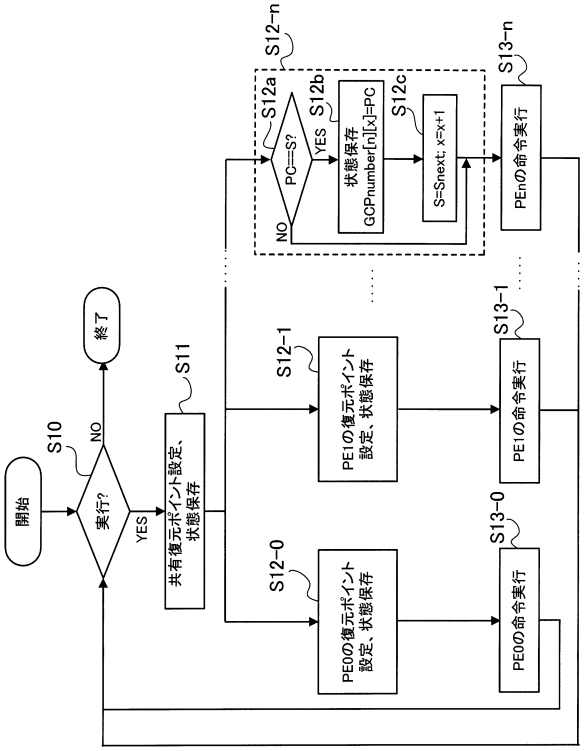
【図２】


10
シミュレーション装置
11
マルチコア CPU
12
ROM
13
SDRAM
14
HDD
15
グラフィック 処理部
15a
ディスプレイ
16
入力I/F
17
16a
マウス
16b
キーボード

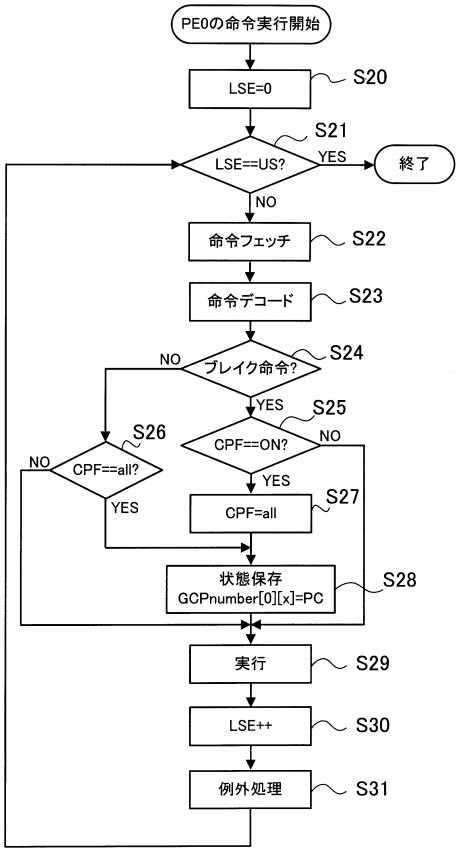
【図３】


マルチコアシミュレータ
（main関数）
PE0
PE1
PEn
memory
IRC
UART
DMA
TIMER
I/Oモデル

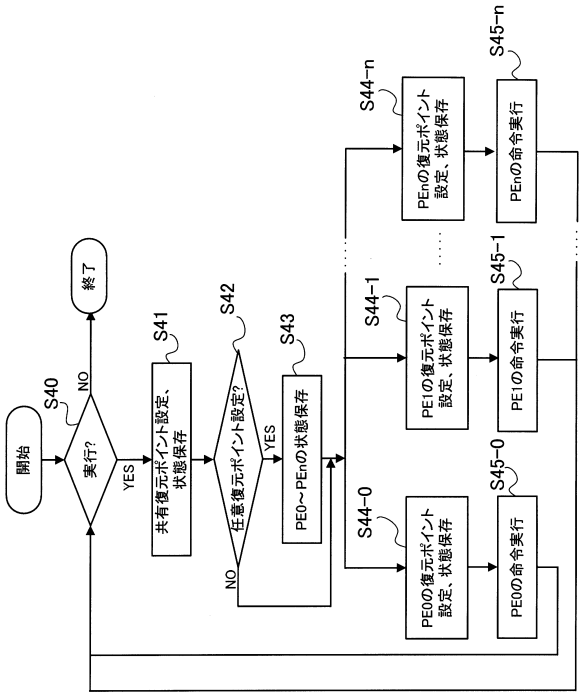
【図４】


開始
S10
実行?
NO
YES
終了
S11
共有復元ポイント設定、状態保存
S12-n
S12a
PC==S?
YES
NO
S12b
状態保存 GCPnumber[n][x]=PC
S12c
S=Snext; x=x+1
S13-n
PEnの命令実行
S12-1
PE1の復元ポイント設定、状態保存
S13-1
PE1の命令実行
S12-0
PE0の復元ポイント設定、状態保存
S13-0
PE0の命令実行

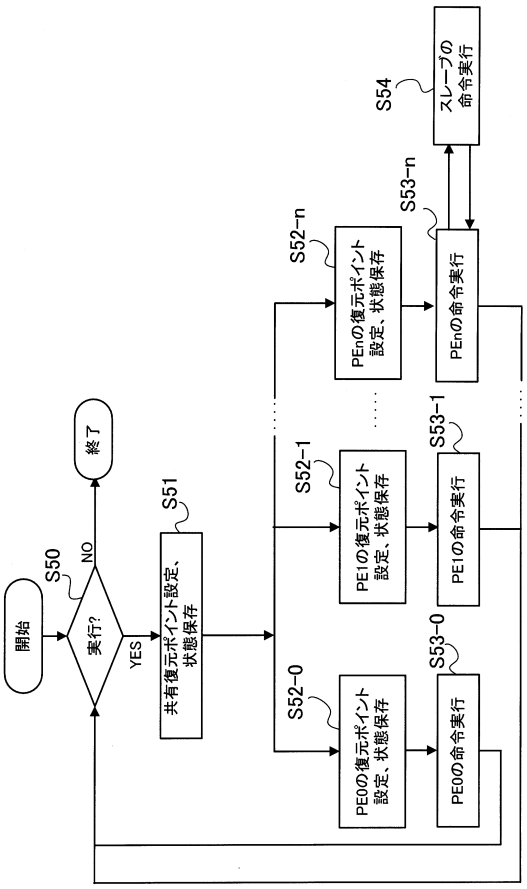
【図５】


PE0の命令実行開始
LSE=0
S20
S21
LSE==US?
YES
終了
NO
命令フェッチ
S22
命令デコード
S23
NO
ブレイク命令?
S24
YES
S25
CPF==ON?
NO
YES
S26
NO
CPF==all?
YES
CPF=all
S27
状態保存 GCPnumber[0][x]=PC
S28
実行
S29
LSE++
S30
例外処理
S31

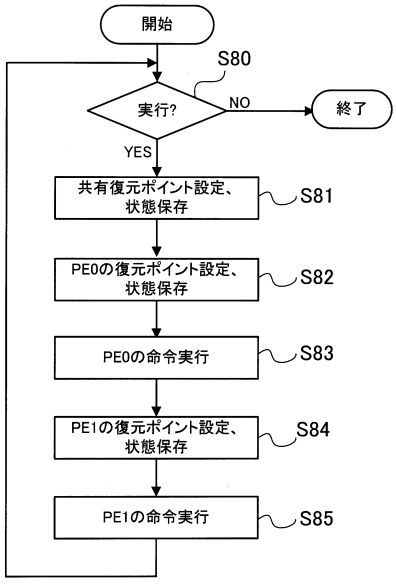
【図６】



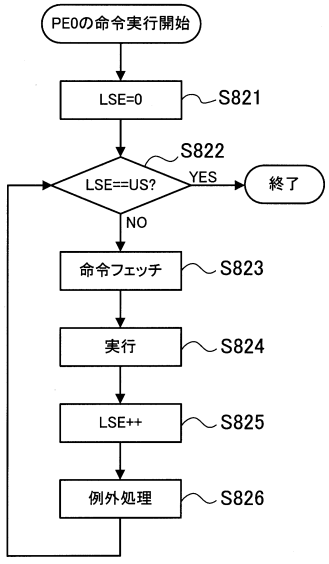

【図７】



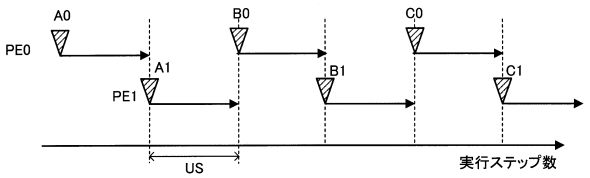

【図８】




【図９】




【図１０】



フロントページの続き

合議体
審判長　金子　幸一
審判官　飯田　清司
審判官　仲間　晃

(56)参考文献　特開２００６－２９３７５９（ＪＰ，Ａ）
特開２００５－３５３０２０（ＪＰ，Ａ）
特開２０００－２０３４９（ＪＰ，Ａ）
特開平９－１９８２７６（ＪＰ，Ａ）
特開平１０－１６１９０６（ＪＰ，Ａ）
特開平５－２７６６１（ＪＰ，Ａ）
立岡真人、外３名、組み込み型マルチプロセッサの高速シミュレーションの開発、ＤＡシンポジウム　２００５、日本、社団法人情報処理学会、２００５年８月２４日、第２００５巻、第９号、１６３－１６８頁

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F 11/28