(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第5365847号
(P5365847)
(45)発行日　平成25年12月11日(2013.12.11)　(24)登録日　平成25年9月20日(2013.9.20)

(51)Int.Cl.　　F I
G06F 13/10 (2006.01)　　G06F 13/10 330C
G06F 13/14 (2006.01)　　G06F 13/14 320B

請求項の数 4　(全 15 頁)

(21)出願番号　特願2009-51861(P2009-51861)
(22)出願日　平成21年3月5日(2009.3.5)
(65)公開番号　特開2010-205124(P2010-205124A)
(43)公開日　平成22年9月16日(2010.9.16)
審査請求日　平成23年3月10日(2011.3.10)

(73)特許権者　000004237
日本電気株式会社
東京都港区芝五丁目７番１号
(74)代理人　100130029
弁理士　永井　道雄
(74)代理人　100166338
弁理士　関口　正夫
(74)代理人　100152054
弁理士　仲野　孝雅
(72)発明者　小池　毅
東京都港区芝五丁目７番１号　日本電気株式会社内

審査官　坂東　博司

最終頁に続く

(54)【発明の名称】仮想化装置における物理デバイスのコンフィグレーション処理方法及びコンピュータシステム

(57)【特許請求の範囲】
【請求項１】
オペレーティングシステムと、
前記オペレーティングシステムから、物理デバイスへのコンフィグレーションアクセスを捕捉し、前記オペレーティングシステムがアクセスした前記物理デバイスのデバイスアドレスから、前記物理デバイスが所属するパーティションを特定し、
前記物理デバイスが自パーティションに所属するデバイスであった場合には、前記オペレーティングシステムが要求した前記物理デバイスへのコンフィグレーションレジスタリードまたはライトをそのまま実行し、
前記物理デバイスが自パーティションに所属していないデバイスであった場合には、前記物理デバイスを特定するコンフィグレーションレジスタの内容を、実在しないダミーデバイスの情報に置き換えて前記オペレーティングシステムに通知する仮想化手段と、
を備え、
前記オペレーティングシステムは前記ダミーデバイスに対応したダミーデバイスドライバを有し、該ダミーデバイスドライバによって、前記自パーティションに所属しない物理デバイスを、名称の付与された有効なデバイスとして認識し、
前記ダミーデバイスドライバは、前記オペレーティングシステムから要求される全物理デバイスに共通な処理のうち、他パーティションに所属している物理デバイスの動作に擾乱を与えるものについては、その実行を拒絶すること、を特徴とするコンピュータシステム。

【請求項２】
請求項１に記載のコンピュータシステムにおいて、前記仮想化手段は、前記物理デバイスが自パーティションに所属していないデバイスであった場合には、前記物理デバイスの製造元および種類を特定するコンフィグレーションレジスタの内容を、実在しないダミーデバイスの製造元または種類に置き換えて前記オペレーティングシステムに通知するコンピュータシステム。
【請求項３】
仮想化装置における物理デバイスのコンフィグレーション処理方法において、
仮想化ソフトウェア上で動作するオペレーティングシステムから、物理デバイスへのコンフィグレーションアクセスを捕捉し、
前記オペレーティングシステムがアクセスした前記物理デバイスのデバイスアドレスから、前記物理デバイスが所属するパーティションを特定し、
前記物理デバイスが自パーティションに所属するデバイスであった場合には、前記オペレーティングシステムが要求した前記物理デバイスへのコンフィグレーションレジスタリードまたはライトを実行し、前記物理デバイスが自パーティションに所属していないデバイスであった場合には、前記物理デバイスを特定するコンフィグレーションレジスタの内容を、実在しないダミーデバイスの情報に置き換えて前記オペレーティングシステムに通知し、
前記ダミーデバイスに対応したダミーデバイスドライバによって、前記自パーティションに所属しない物理デバイスを、名称の付与された有効なデバイスとして前記オペレーティングシステムに認識させ、
前記オペレーティングシステムから要求される全物理デバイスに共通な処理のうち、他パーティションに所属している物理デバイスの動作に擾乱を与えるものについては、その実行を前記ダミーデバイスドライバにて拒絶するコンフィグレーション処理方法。
【請求項４】
請求項３に記載のコンフィグレーション処理方法において、前記物理デバイスが自パーティションに所属していないデバイスであった場合には、前記物理デバイスの製造元および種類を特定するコンフィグレーションレジスタの内容を、実在しないダミーデバイスの製造元または種類に置き換えて前記オペレーティングシステムに通知するコンフィグレーション処理方法。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、仮想化装置における物理デバイスのコンフィグレーション処理方法及びコンピュータシステムに係わる。
【背景技術】
【０００２】
仮想化技術は、ハードウェアとソフトウェアの結びつきを抽象化し、ＯＳ（オペレーティングシステム）などのソフトウェアがハードウェアを直接制御するのではなく、抽象化レイヤを介して動作できるようにする技術である。
【０００３】
仮想化の技術自体はメインフレームが全盛の時代から存在し、１つのハードウェア上で複数の基幹システムを動作させることで、高価なハードウェアを、より効率的に使用することを目的としていた。
【０００４】
しかし現在は、テクノロジの進化とオープンソースソフトウェアの浸透により、安価なサーバ上でも容易に仮想化環境が構築できるようになった。
【０００５】
本願発明に関連する技術として、特許文献１には、ＰＣＩデバイスにおいて、メモリマップドＩＯレジスタのアドレス空間内にコンフィグレーションレジスタのアドレス空間を

規定し、メモリマップドＩＯレジスタのアドレス空間内に規定されたコンフィグレーションレジスタのアドレスを指定して、コンフィグレーションレジスタに対しアクセスすることの記載がある。

【０００６】

また、特許文献２には、ＰＣＩデバイスと非ＰＣＩデバイスとを実装するコンピュータシステムにおいて、非ＰＣＩデバイスに対応させた仮想的なＰＣＩデバイスの所定の識別情報を少なくとも記憶した記憶領域をシステムＢＩＯＳ側に用意しておき、システムＢＩＯＳが参照可能な媒体上に、仮想的なＰＣＩデバイスのコンフィギュレーション空間のヘッダ領域に仮想的なＰＣＩデバイスの所定の識別情報が格納されることの記載がある。

【先行技術文献】

【特許文献】

【０００７】

【特許文献１】特開２００２－３１８７７９号公報

【特許文献２】特開２００５－２５０９７５号公報

【発明の概要】

【発明が解決しようとする課題】

【０００８】

近年、仮想化の目的も、ハードウェアの使用効率の追求から、ある程度の使用効率の低下は許容しながらも、より機構が単純で、一度開発したソフトウェア資産を長期に渡って使用し続けられる仮想化のインフラストラクチャが、重要視されるようになってきた。

【０００９】

本願発明は、前述の仮想化技術の変遷を鑑み、Ｉ／Ｏデバイスの仮想化の分割粒度をＰＣＩデバイス単位に制限することで、仮想化ソフトウェアの構造の単純化と、パフォーマンスの向上を目的とするものである。

【課題を解決するための手段】

【００１０】

本発明に係わるコンピュータシステムは、オペレーティングシステムと、

前記オペレーティングシステムから、物理デバイスへのコンフィグレーションアクセスを捕捉し、前記オペレーティングシステムがアクセスした物理デバイスのデバイスアドレスから、前記物理デバイスが所属するパーティションを特定し、

前記物理デバイスが自パーティションに所属するデバイスであった場合には、前記オペレーティングシステムが要求した前記物理デバイスへのコンフィグレーションレジスタリードまたはライトをそのまま実行し、

前記物理デバイスが自パーティションに所属していないデバイスであった場合には、前記物理デバイスを特定するコンフィグレーションレジスタの内容を、実在しないダミーデバイスの情報に置き換えて前記オペレーティングシステムに通知する仮想化手段と、

を備え、

前記オペレーティングシステムは前記ダミーデバイスに対応したダミーデバイスドライバを有し、該ダミーデバイスドライバによって、前記自パーティションに所属しない物理デバイスを、名称の付与された有効なデバイスとして認識し、

前記ダミーデバイスドライバは、前記オペレーティングシステムから要求される全物理デバイスに共通な処理のうち、他パーティションに所属している物理デバイスの動作に擾乱を与えるものについては、その実行を拒絶すること、を特徴とするコンピュータシステムである。

【００１１】

本発明に係わる物理デバイスのコンフィグレーション処理方法は、仮想化装置における物理デバイスのコンフィグレーション処理方法において、

仮想化ソフトウェア上で動作するオペレーティングシステムから、物理デバイスへのコンフィグレーションアクセスを捕捉し、

前記オペレーティングシステムがアクセスした物理デバイスのデバイスアドレスから、

前記物理デバイスが所属するパーティションを特定し、
前記物理デバイスが自パーティションに所属するデバイスであった場合には、前記オペレーティングシステムが要求した前記物理デバイスへのコンフィグレーションレジスタリードまたはライトを実行し、前記物理デバイスが自パーティションに所属していないデバイスであった場合には、前記物理デバイスを特定するコンフィグレーションレジスタの内容を、実在しないダミーデバイスの情報に置き換えて前記オペレーティングシステムに通知し、
前記ダミーデバイスに対応したダミーデバイスドライバによって、前記自パーティションに所属しない物理デバイスを、名称の付与された有効なデバイスとして前記オペレーティングシステムに認識させ、
前記オペレーティングシステムから要求される全物理デバイスに共通な処理のうち、他パーティションに所属している物理デバイスの動作に擾乱を与えるものについては、その実行を前記ダミーデバイスドライバにて拒絶するコンフィグレーション処理方法である。

【発明の効果】

【００１２】

本発明によれば、仮想化ソフトウェア上で動作するＯＳからは、仮想メモリマップドＩ／Ｏ空間と実メモリマップドＩ／Ｏ空間のＰＣＩデバイスのレジスタ割り付けが同一に見えるため、ＯＳが標準で装備するＰＣＩデバイスドライバを、そのまま利用することができ、仮想化ソフトウェアのプログラム規模の縮小と、再利用性を向上させる効果がある。

【００１３】

また、ＯＳがＰＣＩデバイスのＩ／Ｏレジスタにアクセスする際に、仮想メモリマップドＩ／Ｏ空間から実メモリマップドＩ／Ｏ空間へのアドレス変換を行う必要がないため、仮想化によるＩ／Ｏスループットの低下を防ぐ効果がある。

【図面の簡単な説明】

【００１４】

【図１】図１は本発明の一実施形態に係わるコンピュータシステムの構成図である。
【図２】準仮想化方式の構成を示すブロック図である。
【図３】完全仮想化方式の構成を示すブロック図である。
【図４】ホストバスに複数のＣＰＵが接続されているマルチプロセッサシステムを示す図である。
【図５】ＰＣＩバスモデルのシステムを、仮想化によって２つのパーティションに分割した場合を示す図である。
【図６】仮想化を行っていないシステムでコンフィグレーション処理を行った場合の、実アドレス空間のメモリマップドＩ／Ｏのレジスタ割り付けと、仮想化されたパーティション１およびパーティション２のメモリマップドＩ／Ｏのレジスタ割り付けを示す図である。
【図７】デバイス管理テーブルを示す図である。
【図８】ＰＣＩ仕様２．０で規定されたコンフィグレーション・サイクルの発生メカニズムを表す図である。
【図９】割り付けが行われた、実アドレスとパーティション１および２のメモリマップドＩ／Ｏ空間のコンフィグレーション結果を表す図である。
【図１０】本実施形態の仮想化ソフトウェアの処理フローを示すフローチャートである。

【発明を実施するための形態】

【００１５】

以下、本発明の実施の形態について図面を用いて詳細に説明する。

【００１６】

まず、本発明の実施形態の理解を容易にするために、Ｉ／Ｏを仮想化する技術について説明する。Ｉ／Ｏを仮想化する技術には、準仮想化方式と完全仮想化方式がある。

【００１７】

図２は、準仮想化方式の構成を示すブロック図である。ここでは、２つのゲストＯＳ（

オペレーティングシステム）１１－１、１１－２とホスト１２を有する場合について説明する。準仮想化方式では、ゲストＯＳ１１－１、１１－２に仮想化ソフトウェア専用のデバイスドライバを組み込む。このドライバをフロントエンドドライバ４３－１、４３－２という。ゲストＯＳ１１－１、１１－２が実行したＩ／Ｏ処理は、それぞれフロントエンドドライバ４３－１、４３－２から仮想化ソフトウェア２０の内部通信機構６０を経由して、ホストＯＳ１２（サービス・パーティション、あるいは管理ドメインなどとも言う）のバックエンドドライバ４４に通知される。ホストＯＳ１２からは、全ての物理ハードウェア３０が参照できるため、仮想化ソフトウェア２０は、ゲストＯＳ１１－１、１１－２から依頼されたＩ／Ｏ処理を、ホストＯＳ１２上の標準デバイスドライバ４１で実行可能な形式に変換して、物理デバイス５１－１、５１－２にアクセスする。

【００１８】

準仮想化方式は、物理デバイス５１－１、５１－２にアクセスする際に、ホストＯＳ１２に付属の標準デバイスドライバ４１を使用するため、システムの置き換え等によりハードウェアが更新された場合でも、フロントエンドドライバ４３－１、４３－２やバックエンドドライバ４４を修正しなくて済むという利点がある。その反面、ゲストＯＳ１１－１、１１－２が実行したＩ／Ｏ処理は、必ず仮想化ソフトウェア２０の内部通信機構６０を経由してホストＯＳ１２に集約されるため、処理のオーバーヘッドが発生するという問題がある。

【００１９】

図３に示す完全仮想化方式は、準仮想化による処理オーバーヘッドとホストＯＳ（オペレーティングシステム）１２のようなサービス・パーティションを無くすよう改善した方式である。完全仮想化方式は、準仮想化方式のフロントエンドドライバ４３－１、４３－２の替わりに、仮想化ソフトウェア専用の仮想デバイスドライバ６１－１、６１－２を、ＯＳの標準デバイスドライバ４１－１、４１－２の配下に組み込む方式である。

【００２０】

仮想デバイスドライバ６１－１、６１－２は、その組み込み位置によって、仮想デバイス方式と直接割り当て方式に大別される。仮想デバイス方式は、仮想化ソフトウェア２０が用意するデバイスドライバをＯＳ１０－１、１０－２内に組み込む方式である。このデバイスドライバを通じて、ＯＳ１０－１、ＯＳ１０－２が実行したＩ／Ｏ処理が仮想化ソフトウェア２０に通知され、仮想化ソフトウェア２０が適切な形でＩ／Ｏを再現する。もう一つの直接割り当て方式は、仮想デバイス方式のデバイスドライバ部分を仮想化ソフトウェア２０内に取り込み、ＯＳ１０－１、１０－２には直接仮想化されたハードウェアを見せる方式である。ＯＳ１０－１、１０－２は標準デバイスドライバ４１－１、４１－２を使用して、仮想化ソフトウェア２０内の仮想デバイスドライバ６１－１、６１－２にアクセスする。図３は直接割り当て方式を例示したものである。

【００２１】

完全仮想化方式は、準仮想化方式と比べてゲストＯＳ１１－１、１１－２からホストＯＳ１２への内部通信機構６０を使用しない分、仮想化による処理オーバーヘッドは軽減される。但し、仮想デバイス方式・直接割り当て方式を問わず、完全仮想化方式は、ＯＳ１０－１、１０－２に対応した仮想デバイスドライバ６１－１、６１－２を、ハードウェアが変更されるたびに手直ししなければならないという問題がある。

【００２２】

仮想化技術において、物理デバイスと仮想デバイスの対応付けのために、専用の仮想デバイスドライバを必要とする理由に、オープンＯＳが採用しているＰＣＩデバイス仕様に則ったコンフィグレーション処理がある。以下、物理ハードウェアとして図４に示すＰＣＩバスモデルを使用して、オープンＯＳが採用しているコンフィグレーション処理について説明する。

【００２３】

図４のシステムは、ホストバス５にＣＰＵ１ａ～１ｄが接続されているマルチプロセッサシステムである。ホストバス５にはＩ／Ｏデバイスにアクセスするためのホストブリッ

ジ２が接続されている。本モデルは２つのＰＣＩバス６ａと６ｂを有する。ＰＣＩバス６ａにはバス番号０（Ｂ＝０）、ＰＣＩバス６ｂにはバス番号１（Ｂ＝１）が付与されている。ＰＣＩバス６ａとＰＣＩバス６ｂはＰＣＩ－ＰＣＩブリッジ４によって接続されている。ＰＣＩバス６ａには１つのＰＣＩデバイス３ａが接続されており、ＰＣＩバス６ｂには３つのＰＣＩデバイス３ｂ～３ｄが接続されている。

【００２４】

ＰＣＩデバイスは、システムの起動時やホットプラグ時に、ＯＳによるコンフィグレーション処理によってバス番号とデバイス番号の割り付けが決定される。

【００２５】

コンフィグレーション処理におけるバス番号の割り付け手順は以下の通りである。

（１）処理Ｐ１

ホストブリッジの二次側バスにバス番号０を付与し、このバス番号を記憶する。ホストブリッジの二次側バスを検査しＰＣＩ－ＰＣＩブリッジの存在を確認する。

（２）処理Ｐ２

見つかったＰＣＩ－ＰＣＩブリッジの二次側バスに、記憶してあったバス番号に１を加算したバス番号を付与し、このバス番号を記憶する。

（３）処理Ｐ３

上記（２）の処理Ｐ２で見つけたＰＣＩ－ＰＣＩブリッジの二次側バスを検査し、ＰＣＩ－ＰＣＩブリッジの配下に、更にＰＣＩ－ＰＣＩブリッジが存在しないか確認する。新たなＰＣＩ－ＰＣＩブリッジが見つかったら上記処理Ｐ２の処理を行う。

（４）処理Ｐ４

ＰＣＩ－ＰＣＩブリッジの配下に、新たなＰＣＩ－ＰＣＩブリッジが見つからなくなったら、ホストブリッジに接続されている次のＰＣＩ－ＰＣＩブリッジについて上記処理Ｐ２を行う。

【００２６】

コンフィグレーション処理におけるデバイス番号の割り付け手順は以下の通りである。

（１）処理Ｐ１１

ホストブリッジまたはＰＣＩ－ＰＣＩブリッジの二次側バスを検査する前に、デバイス番号を０に初期化し、このデバイス番号を記憶する。

（２）処理Ｐ１２

バスに接続されているＰＣＩデバイスまたはＰＣＩ－ＰＣＩブリッジを検査する。見つかったＰＣＩデバイスまたはＰＣＩ－ＰＣＩブリッジに、記憶してあったデバイス番号を割り当てて、デバイス番号に１を加算して記憶する。

（３）処理Ｐ１３

バスに接続されているＰＣＩデバイスまたはＰＣＩ－ＰＣＩブリッジを見つけるたびに、上記処理Ｐ１２を繰り返す。

（４）処理Ｐ１４

バスの配下に、新たなＰＣＩデバイスまたはＰＣＩ－ＰＣＩブリッジが見つからなくなったら、次のＰＣＩ－ＰＣＩブリッジの二次側バスについて上記処理Ｐ１１を行う。

【００２７】

図４の括弧内の数字は、本手順に従って割り付けられたＰＣＩデバイスまたはＰＣＩ－ＰＣＩブリッジの、一次側のバス番号（Ｂ）とデバイス番号（Ｄ）を表す。

【００２８】

ここで、本ＰＣＩバスモデルのシステムを、仮想化によって２つのパーティションに分割した場合を考える。図５は、ＰＣＩデバイス３ａと３ｂをパーティション１に、ＰＣＩデバイス３ｃと３ｄをパーティション２に割り当てた構成である。

【００２９】

仮想化ソフトウェアは、ＯＳからのコンフィグレーション処理に対して、自パーティションに所属しないデバイスは未実装状態に見せてＯＳに割り付けを行わせる。

【００３０】

図５のＰＣＩバスモデルの括弧内の数字は、パーティショニングされた２つのＯＳ１０ａ、１０ｂのそれぞれから、前記コンフィグレーション処理に従ってバス番号とデバイス番号を付与した状態を表している。

【００３１】

図５において、パーティション１に所属するＰＣＩデバイス３ｂと、パーティション２に所属するＰＣＩデバイス３ｃは、共にバス番号１／デバイス番号０（Ｂ＝１／Ｄ＝０）を有する。その結果物理デバイスにこのバス番号／デバイス番号を付与するとバス競合が発生する。

【００３２】

この場合、バス番号／デバイス番号のバス競合を回避するためには、仮想化ソフトウェアにて仮想デバイスと物理デバイスのバス番号／デバイス番号の変換を行う必要がある。

【００３３】

次に、図６を使用して、仮想化を行っていないシステムでコンフィグレーション処理を行った場合の、実アドレス空間のメモリマップドＩ／Ｏのレジスタ割り付けと、仮想化されたパーティション１およびパーティション２のメモリマップドＩ／Ｏのレジスタ割り付けを示す。

【００３４】

オープンＯＳにおいては、前記コンフィグレーション処理で検出したＰＣＩデバイスがレジスタ領域を必要とする場合、メモリマップドＩ／Ｏ空間の最上位アドレスから順番に、検出したＰＣＩデバイスのレジスタを並べて配置するように動作する。

【００３５】

同様の手順で、仮想化されたパーティションのＯＳがコンフィグレーション処理を行うと、自パーティションに所属していないデバイスは、仮想化ソフトウェアによって隠蔽されて未検出状態となるため、自パーティションの仮想メモリマップドＩ／Ｏ上にはマッピングされず、仮想メモリマップドＩ／Ｏと実メモリマップドＩ／Ｏのアドレス割り付けに差分が生じる。その結果、仮想メモリマップドＩ／Ｏと実メモリマップドＩ／Ｏを同一視してＰＣＩデバイスにアクセスすると、Ｉ／Ｏレジスタのアドレス競合が発生する。

【００３６】

仮想化によるＰＣＩデバイスのＩ／Ｏレジスタのアドレス競合を回避するためには、仮想化ソフトウェアにて仮想メモリマップドＩ／Ｏと実メモリマップドＩ／Ｏのアドレス変換を行う必要があった。

【００３７】

また、ゲストＯＳやＰＣＩデバイスに付属するデバイスドライバの中には、仮想メモリマップドＩ／Ｏに対応していないデバイスドライバもあるため、これらについては、仮想化ソフトウェア専用のデバイスドライバをＯＳの配下に組み込む必要があった。

【００３８】

次に、本実施形態について説明する。

【００３９】

図１は本発明の一実施形態に係わるコンピュータシステムの構成図である。

【００４０】

図１の実施形態のシステムの構成は、物理ハードウェア３０を物理デバイス５１－１、５１－２の単位で、バーティション１とパーティション２の２つの仮想システムに分割したものである。

【００４１】

パーティション１の物理デバイス５１－１は、パーティション１に所属する物理ＣＰＵ５０－１上で動作するＯＳ（オペレーティングシステム）の制御下に置かれ、パーティション１のＯＳ１０－１に付属する標準デバイスドライバ４１－１を使用してアクセスされる。

【００４２】

同様に、パーティション２の物理デバイス５１－２は、パーティション２に所属する物

理ＣＰＵ５０－２上で動作するＯＳの制御下に置かれ、パーティション２のＯＳ１０－２に付属する標準デバイスドライバ４１－２を使用してアクセスされる。

【００４３】

仮想化ソフトウェア２０は、物理ＣＰＵ５０－１、５０－２に対応する仮想ＣＰＵ４０－１、４０－２からのコンフィグレーションアクセスを捕捉し、例えば、自パーティションに所属していない物理デバイス５１－２へ仮想ＣＰＵ４０－１がアクセスした場合に、物理デバイス５１－２のベンダＩＤおよびデバイスＩＤを、実在しないダミーデバイスのＩＤに置き換えて仮想ＣＰＵ４０－１に通知する機能を有する。仮想化ソフトウェアは仮想化手段となる。

【００４４】

ダミーデバイスドライバ４２－１、４２－２は、ダミーデバイスに対応したデバイスドライバであって、自パーティションに所属しない物理デバイスを、名称の付与された有効なデバイスとしてＯＳに認識させる機能を有する。例えば、ダミーデバイスドライバ４２－１は、パーティション１に所属しない物理デバイス５１－２を、名称の付与された有効なデバイスとしてＯＳ１０－１に認識させる。また、ダミーデバイスドライバ４２－１、４２－２は、ＯＳ１０－１、１０－２から要求される全ＰＣＩデバイスに共通な処理のうち、他パーティションに所属しているＰＣＩデバイスの動作に擾乱を与えるものについては、その実行を拒絶する機能を有する。

【００４５】

なお、本実施形態の構成では、物理ＣＰＵ５０－１、５０－２を仮想化する手段として仮想ＣＰＵ４０－１、４０－２を定義したが、ＣＰＵに関する仮想化技術については、ここでは説明を省略する。

【００４６】

物理デバイス５１－１、５１－２と各パーティションの対応付けは、図７に示すデバイス管理テーブルにて行う。デバイス管理テーブルは、仮想化システムを構築する際に予め決定しておくべきテーブルであって、仮想化ソフトウェア２０内に保持される。

【００４７】

デバイス管理テーブルは、物理デバイス５１－１、５１－２のバス番号、デバイス番号、機能番号を入力に、当該物理デバイス５１－１、５１－２がどのパーティションに所属するＰＣＩデバイスであるのかを規定する。

【００４８】

本実施形態では、説明を容易化するため、物理デバイス５１－１、５１－２に対応した実デバイスのベンダＩＤ／デバイスＩＤと、これに対応するダミーデバイスのベンダＩＤ／デバイスＩＤを、デバイス管理テーブル内に保持する。但し、実デバイスのベンダＩＤ／デバイスＩＤは、物理デバイス５１－１、５１－２のコンフィグレーションレジスタをリードすることで、またダミーデバイスのベンダＩＤ／デバイスＩＤは、固定値を使用することで代用可能なため、これらの情報は必須の構成要素ではない。

【００４９】

本実施形態の動作について、図１と図４および図７から図９を使用して説明する。

【００５０】

図８はＰＣＩ仕様２．０で規定されたコンフィグレーション・サイクルの発生メカニズムを表す図である。ＡＴ互換機系のシステムでは、ＰＣＩデバイス３ａ～３ｄおよびＰＣＩ－ＰＣＩブリッジ４のコンフィグレーションレジスタにアクセスするためのＩ／Ｏポートを、ホストブリッジ２に内蔵する。ホストブリッジ２のコンフィグレーション・サイクル発生回路内には、ＣＯＮＦＩＧ＿ＡＤＤＲレジスタと呼ばれる４バイト幅のレジスタが存在し、ＣＰＵ１ａ～１ｄがＩ／ＯアドレスＣＦ８ｈに対してＩ／Ｏリードまたはライトを実行すると、これをＣＯＮＦＩＧ＿ＡＤＤＲレジスタに対するリード／ライトとして扱う。ＣＯＮＦＩＧ＿ＡＤＤＲレジスタは、ホストブリッジ２がアクセスすべきＰＣＩバスのバス番号、デバイス番号、機能番号、およびＰＣＩデバイスのレジスタアドレスを指し示すためのものであって、その内容はＣＯＮＦＩＧ＿ＡＤＤＲレジスタにライトした最後

の値を保持する。

【００５１】

ホストブリッジ２は、もう１つＣＯＮＦＩＧ＿ＤＡＴＡレジスタと呼ばれる４バイト幅のレジスタを内蔵する。ＣＯＮＦＩＧ＿ＤＡＴＡレジスタは、ＣＰＵ１ａ～１ｄがＩ／ＯアドレスＣＦＣｈからＣＦＦｈまでの４バイト以内のＩ／Ｏリードまたはライトを実行すると、これをＣＯＮＦＩＧ＿ＤＡＴＡレジスタに対するリード／ライトとして扱い、ＣＯＮＦＩＧ＿ＡＤＤＲレジスタで指定したＰＣＩデバイスに対して、コンフィグレーション・サイクルを発生させる。

【００５２】

本実施形態の仮想化ソフトウェア２０は、図１における仮想ＣＰＵ４０－１、４０－２からのＣＯＮＦＩＧ＿ＤＡＴＡレジスタへのアクセスを捕捉し、このときのＣＯＮＦＩＧ＿ＡＤＤＲレジスタの値から、各パーティションのＯＳ１０－１、１０－２が、どの物理デバイスのコンフィグレーションレジスタにアクセスしようとしたのかを判断する。

【００５３】

次に、仮想化ソフトウェア２０は、図７に示すデバイス管理テーブルを索引し、物理デバイスがどのパーティションに所属するデバイスなのかを判断する。

【００５４】

仮想化ソフトウェア２０は、仮想ＣＰＵが発行したコンフィグレーションアクセスが、自パーティションに所属する物理デバイスに対するものであった場合は、そのまま仮想ＣＰＵが発行したコンフィグレーションアクセスを実行する。例えば、仮想ＣＰＵ４０－１が発行したコンフィグレーションアクセスが、自パーティションに所属する物理デバイス５０－１に対するものであった場合は、そのまま仮想ＣＰＵ４０－１が発行したコンフィグレーションアクセスを実行する。

【００５５】

また、仮想ＣＰＵが発行したコンフィグレーションアクセスが、自パーティションに所属しない物理デバイスに対するものであった場合は、ＣＯＮＦＩＧ＿ＡＤＤＲレジスタの内容から、アクセス先の物理デバイスのレジスタアドレスを判定し、ベンダＩＤあるいはデバイスＩＤへのリード要求の場合は、実在しないダミーデバイスのベンダＩＤおよびデバイスＩＤに差し替えて、リードしたデータを仮想ＣＰＵに通知する。例えば、仮想ＣＰＵ４０－１が発行したコンフィグレーションアクセスが、自パーティションに所属しない物理デバイス５１－２に対するものであった場合は、ＣＯＮＦＩＧ＿ＡＤＤＲレジスタの内容から、アクセス先の物理デバイス５１－２のレジスタアドレスを判定し、ベンダＩＤあるいはデバイスＩＤへのリード要求の場合は、実在しないダミーデバイスのベンダＩＤおよびデバイスＩＤに差し替えて、リードしたデータを仮想ＣＰＵ４０－１に通知する。

【００５６】

本実施形態では、ダミーデバイスが使用するベンダＩＤとして５８５３ｈを、またデバイスＩＤとして９９９９ｈ（ＰＣＩデバイス用）と８８８８ｈ（ＰＣＩ－ＰＣＩブリッジ用）を定義する。

【００５７】

図７に、図４で示したＰＣＩバスモデルの、全ＰＣＩデバイス３ａ～３ｄとＰＣＩ－ＰＣＩブリッジ４のベンダＩＤ／デバイスＩＤについて、実デバイスとダミーデバイスのそれぞれのＩＤを記述する。

【００５８】

本実施形態の仮想化ソフトウェア２０上で動作するＯＳ１０－１、１０－２は、前述の機構によって、自パーティションに所属しない物理デバイス５１－２、５１－１を、物理的／論理的に実装されているものと認識しながらも、使用することのない未知のデバイス（ダミーデバイス）として扱う。

【００５９】

その結果、ＯＳ１０－１、１０－２の起動時に実施されるコンフィグレーション処理においても、何れのパーティションのＯＳからも、同じＰＣＩバス構成が参照できるため、

仮想化を行わなかったときと同じ図４に示すバス番号／デバイス番号が、各物理デバイス５１－１、５１－２に付与されることとなる。

【００６０】

図９は、割り付けが行われた、実アドレスとパーティション１および２のメモリマップドＩ／Ｏ空間のコンフィグレーション結果を表す。

【００６１】

図７および図９において、ＰＣＩデバイス＃１はベンダＩＤ：１０１４ｈ／デバイスＩＤ：１０８０ｈを有する。本デバイスはパーティション１に所属しているため、パーティション１には実デバイスと同じベンダＩＤ／デバイスＩＤを持つＰＣＩデバイスが、実アドレス空間と同じメモリマップドＩ／Ｏの位置に割り付けられる。

【００６２】

一方、パーティション２からは、ＰＣＩデバイス＃１はベンダＩＤ：５８５３ｈ／デバイスＩＤ：９９９９ｈを有する、実在しない未知のデバイス（ダミーデバイス）として認識されるが、本ダミーデバイスはＰＣＩデバイス＃１と同じベースアドレスレジスタのサイズを有しているため、実アドレス空間と同じメモリマップドＩ／Ｏの位置に、ＰＣＩデバイス＃１と同じＩ／Ｏレジスタサイズを持つダミーデバイスとして割り付けられる。

【００６３】

ＰＣＩデバイス＃２から＃４についても、同様の手順でメモリマップドＩ／Ｏへの割り付けが行われる。

【００６４】

また、本実施形態では、ＰＣＩ－ＰＣＩブリッジを何れのパーティションにも所属していない共通のデバイスとして定義している。このようなデバイスは、図７に示すデバイス管理テーブル上はパーティション番号０を付与され、ベンダＩＤ：５８５３ｈ／デバイスＩＤ：８８８８ｈを有するダミーブリッジとして、パーティション１および２のＯＳに認識される。

【００６５】

図９のメモリマップドＩ／Ｏ空間のデバイス割り付け状態を参照すると、パーティション１および２からは、自パーティションに所属するＰＣＩデバイスは、実アドレス空間のメモリマップドＩ／Ｏと全く同じアドレスに割り付けられていることが分かる。

【００６６】

また、自パーティションに所属しないＰＣＩデバイスあるいはＰＣＩ－ＰＣＩブリッジは、ダミーデバイスまたはダミーブリッジとして割り付けられていることが分かる。

【００６７】

本機構によって、ＯＳは標準のデバイスドライバを使用して、自パーティションに所属するＰＣＩデバイスを制御することが可能となる。

【００６８】

なお、ＰＣＩ仕様では、図８に示したＰＣＩデバイス用のコンフィグレーションレジスタ配置（タイプ００ｈ）以外に、ＰＣＩ－ＰＣＩブリッジ用のレジスタ配置（タイプ０１ｈ）や、更に後位のＰＣＩ仕様に準拠したレジスタ配置などがある。

【００６９】

これらの中には、ベンダＩＤやデバイスＩＤに類似したサブシステム・ベンダＩＤ／サブシステムＩＤ、あるいはベースアドレスレジスタに類似した拡張ＲＯＭベースアドレスレジスタなどがあるが、これらについても本発明は同様の効果を奏するため、本実施形態では説明を省略する。

【００７０】

次に、図１に示すダミーデバイスドライバ（またはダミーブリッジドライバ）の役割について説明する。

【００７１】

ＯＳが仮想化ソフトウェアを使用して、自パーティションに所属しないＰＣＩデバイスを参照した場合、そのＰＣＩデバイスは未知のデバイスとしてＯＳに認識されるため、自

パーティションに所属しないＰＣＩデバイスの機能をＯＳが使用することはない。

【００７２】

しかし、全ＰＣＩデバイスに共通のＯＳ操作については、ＯＳはこれを実行する可能性がある。特に、このような処理が、他パーティションに所属しているＰＣＩデバイスに対して実行された場合には、他パーティションのシステム運用に影響を与える場合もある。例えばＰＣＩデバイスのホットプラグに先立って実施される、デバイスの切り離し可否の確認がこれに該当する。

【００７３】

本実施形態のデバイスドライバは、全ＰＣＩデバイスに共通のＯＳ操作を防止するため、自パーティションに所属しないＰＣＩデバイスを、名称の付与された有効なデバイスとしてＯＳに認識させ、他パーティションのシステム運用に影響を与えるようなＯＳ操作が、ダミーデバイスドライバに要求された場合には、その実行を拒絶する役割を果たす。

【００７４】

図１０に、本実施形態の仮想化ソフトウェアの処理フローを示す。

【００７５】

仮想化ソフトウェアは、ステップＳ１で、仮想ＣＰＵからのＣＯＮＦＩＧ＿ＤＡＴＡレジスタへのアクセスを捕捉すると、ステップＳ２で、ＣＯＮＦＩＧ＿ＡＤＤＲレジスタの設定値から、アクセスすべきＰＣＩデバイスのバス番号、デバイス番号、機能番号を取得し、ステップＳ３で本情報を基にデバイス管理テーブルを索引し、ＰＣＩデバイスの所属するパーティション番号を特定する。

【００７６】

ステップＳ４で、もしＰＣＩデバイスが自パーティションに所属するデバイスであった場合は、仮想ＣＰＵからのＰＣＩデバイスへのレジスタリードまたはレジスタライトをそのまま実行する。ステップＳ４にて、ＰＣＩデバイスが自パーティションに所属しないデバイスであった場合は、ステップＳ６でＣＯＮＦＩＧ＿ＡＤＤＲレジスタの設定値から、アクセスされたコンフィグレーションレジスタのアドレスを特定し、ベンダＩＤまたはデバイスＩＤへのリードかどうかを判断する。もしこれに該当する場合は、仮想ＣＰＵに、デバイス管理テーブルで設定されているダミーデバイスのベンダＩＤ／デバイスＩＤを返却する。

【００７７】

ステップＳ６において、それ以外のコンフィグレーションレジスタへのアクセスの場合は、ステップＳ８においてＰＣＩデバイスがシステムの起動時やホットプラグ時に既に初期化されていたかどうかを判断し、初期化済みの場合は、ステップＳ９においてＰＣＩデバイスの動作に擾乱を与えないよう、仮想ＣＰＵからのコンフィグレーションレジスタのリードのみを実行し、そうでない場合は、ステップＳ１０においてＰＣＩデバイスのコンフィグレーションレジスタへのリードおよびライトを実行する。

【００７８】

なお、ステップＳ８、Ｓ９、Ｓ１０については、コンフィグレーションレジスタの種類によって、個々に異なるレジスタアクセスの制御を必要とするが、本発明の目的は仮想マップドＩ／Ｏ空間と実マップドＩ／Ｏ空間の、ＰＣＩデバイスのレジスタ割り付けを一致させることにあるため、本実施形態の説明では簡略化したフローを示す。

【００７９】

以上説明した、本実施形態では、サーバ仮想化（バーチャル・マシン）技術における、ＰＣＩデバイスのコンフィグレーション処理において、自パーティションに所属しないＰＣＩデバイスに対して、ＯＳがコンフィグレーションアクセスした場合に、ＰＣＩデバイスのベンダＩＤおよびデバイスＩＤを、実在しないダミーデバイスのＩＤに置き換えてＯＳに通知することによって、ＯＳからは自パーティションに所属していない物理デバイスの種類を隠蔽しながらも、あたかもダミーデバイスが実装されているかのようにコンフィグレーションさせることで、仮想マップドＩ／Ｏ空間と実マップドＩ／Ｏ空間のＰＣＩデバイスのレジスタ割り付けを一致させる機能を提供する。

【産業上の利用可能性】

【００８０】

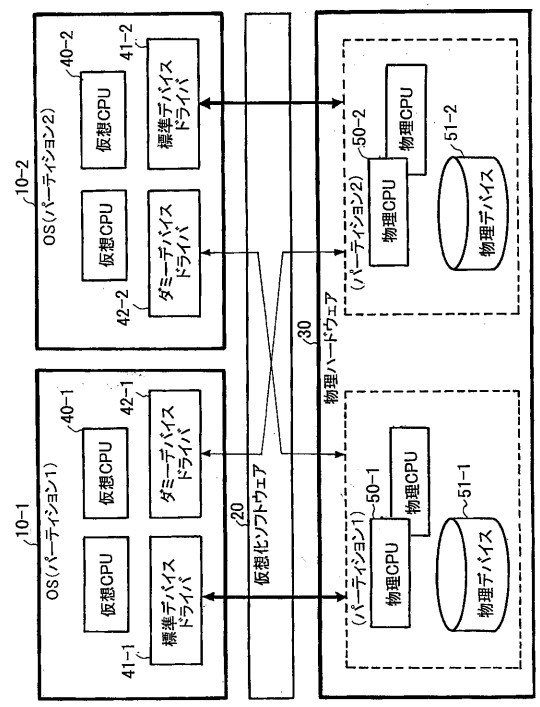
本発明は、ＰＣＩデバイスを含むコンピュータシステムに適用される。

【符号の説明】

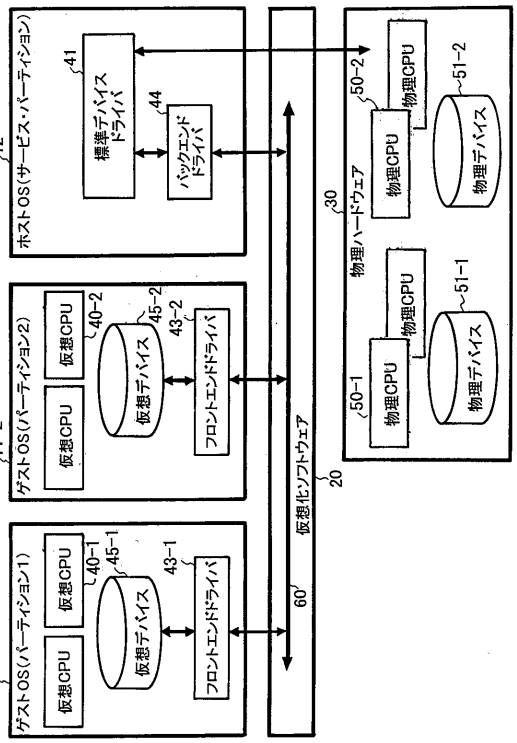
【００８１】

１０－１、１０－２　ＯＳ
２０　仮想化ソフトウェア
３０　物理ハードウェア
４１－１、４１－２　標準デバイスドライバ
４２－１、４２－２ダミーデバイスドライバ
５０－１、５０－２　物理ＣＰＵ
５１－１、５１－２　物理デバイス

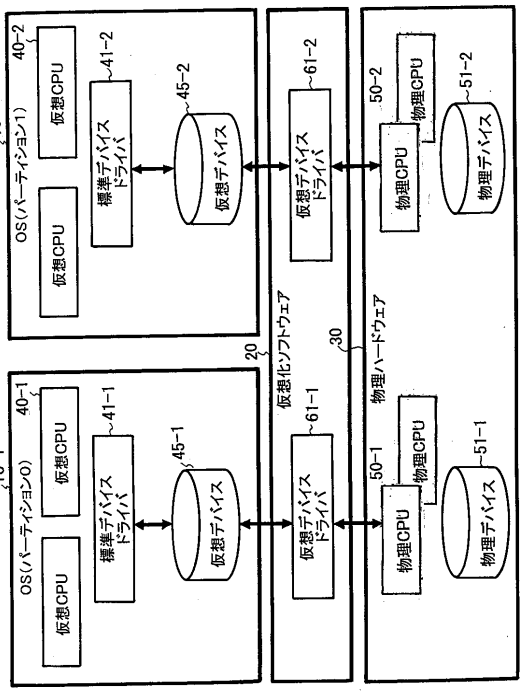
【図１】



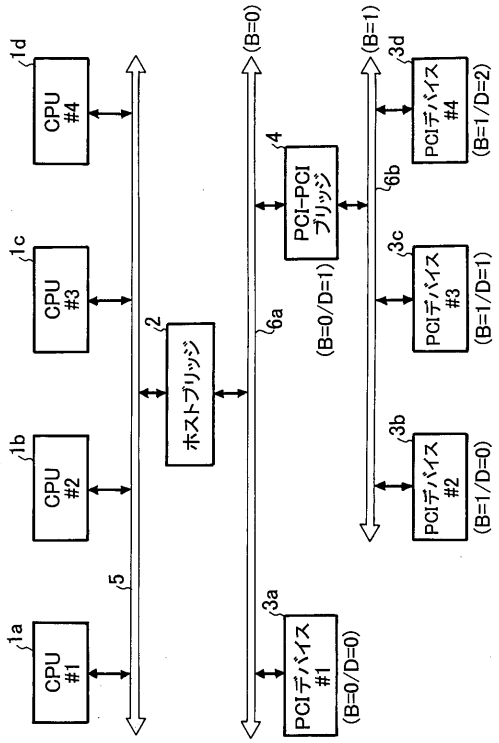
【図２】



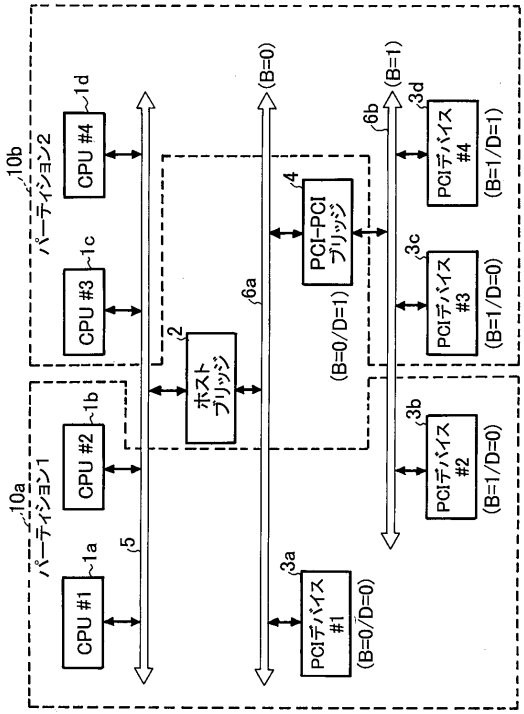
【図３】


10-2
OS(パーティション1)
40-2
仮想CPU
仮想CPU
標準デバイスドライバ
41-2
45-2
仮想デバイス
10-1
OS(パーティション0)
40-1
仮想CPU
仮想CPU
標準デバイスドライバ
41-1
45-1
仮想デバイス
20
仮想化ソフトウェア
61-2
仮想デバイスドライバ
61-1
仮想デバイスドライバ
30
物理ハードウェア
50-2
物理CPU
物理CPU
51-2
物理デバイス
50-1
物理CPU
物理CPU
51-1
物理デバイス

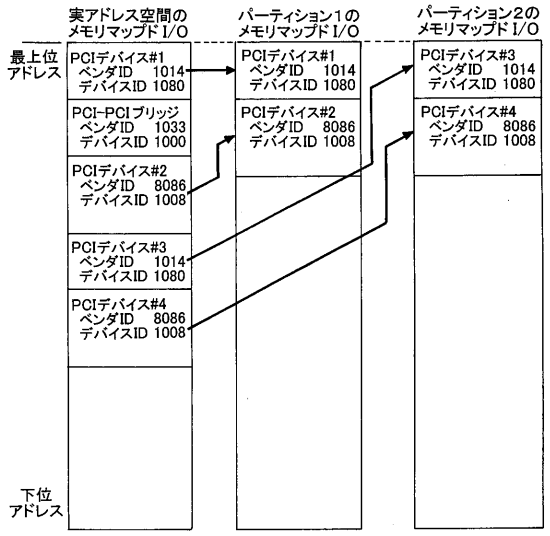
【図４】


1d
CPU #4
1c
CPU #3
1b
CPU #2
1a
CPU #1
5
2
ホストブリッジ
(B=0)
6a
3a
PCIデバイス #1
(B=0/D=0)
(B=0/D=1)
4
PCI-PCI ブリッジ
(B=1)
6b
3c
PCIデバイス #3
(B=1/D=1)
3b
PCIデバイス #2
(B=1/D=0)
3d
PCIデバイス #4
(B=1/D=2)

【図５】


10b
パーティション2
CPU #4
1d
CPU #3
1c
10a
パーティション1
CPU #2
1b
CPU #1
1a
5
2
ホストブリッジ
6a
(B=0)
4
PCI-PCI ブリッジ
(B=0/D=1)
6b
(B=1)
3a
PCIデバイス #1
(B=0/D=0)
3c
PCIデバイス #3
(B=1/D=1)
3b
PCIデバイス #2
(B=1/D=0)
3d
PCIデバイス #4
(B=1/D=1)

【図６】

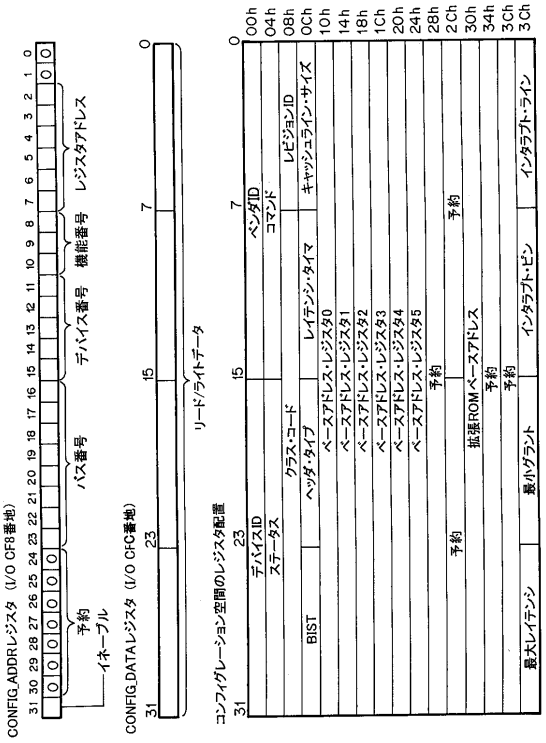

実アドレス空間のメモリマップドI/O
パーティション1のメモリマップドI/O
パーティション2のメモリマップドI/O
最上位アドレス
下位アドレス
PCIデバイス#1 ベンダID 1014 デバイスID 1080
PCI-PCIブリッジ ベンダID 1033 デバイスID 1000
PCIデバイス#2 ベンダID 8086 デバイスID 1008
PCIデバイス#3 ベンダID 1014 デバイスID 1080
PCIデバイス#4 ベンダID 8086 デバイスID 1008
PCIデバイス#1 ベンダID 1014 デバイスID 1080
PCIデバイス#2 ベンダID 8086 デバイスID 1008
PCIデバイス#3 ベンダID 1014 デバイスID 1080
PCIデバイス#4 ベンダID 8086 デバイスID 1008

【図７】

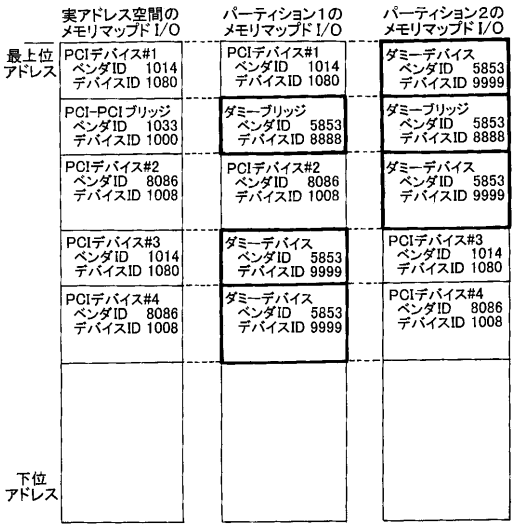
デバイス管理テーブル

CONFIG_ADDRの設定値			パーティション番号	実デバイス		ダミーデバイス	
バス番号	デバイス番号	機能番号		ベンダID	デバイスID	ベンダID	デバイスID
00	00	00	1	1014	1080	5853	9999
00	01	00	0	1033	1000	5853	8888
01	00	00	1	8086	1008	5853	9999
01	01	00	2	1014	1080	5853	9999
01	02	00	2	8086	1008	5853	9999

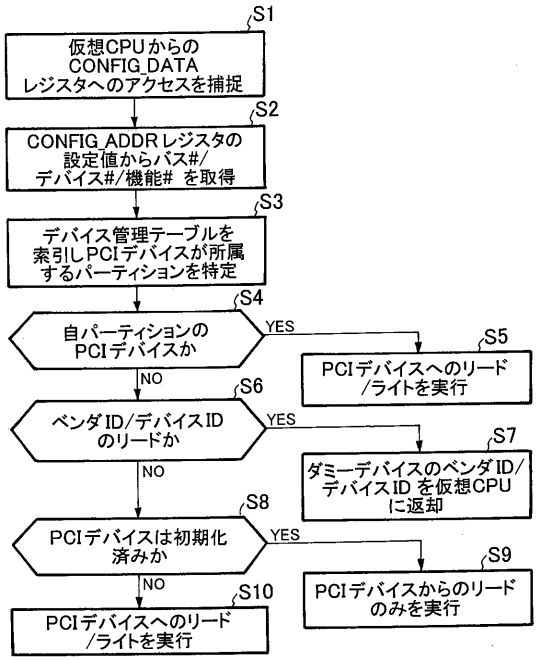
【図８】



【図９】



【図１０】



フロントページの続き

(56)参考文献 米国特許出願公開第２００８／０１６２８６５（ＵＳ，Ａ１）
米国特許出願公開第２００７／００４４１０８（ＵＳ，Ａ１）
特開平１０－２７５２２３（ＪＰ，Ａ）
特開平１１－１６１５９１（ＪＰ，Ａ）
特開平１１－１４３８０６（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　１３／１０
Ｇ０６Ｆ　１３／１４