(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第3745108号
(P3745108)
(45)発行日　平成18年2月15日(2006.2.15)　(24)登録日　平成17年12月2日(2005.12.2)

(51)Int.Cl.　F I
G06F 12/08 (2006.01)　G06F 12/08 531B
G06F 12/08 531E
G06F 12/08 507F
G06F 12/08 541Z
G06F 12/08 551C

請求項の数 10　(全 12 頁)

(21)出願番号　特願平10-15359
(22)出願日　平成10年1月28日(1998.1.28)
(65)公開番号　特開平10-293719
(43)公開日　平成10年11月4日(1998.11.4)
審査請求日　平成14年12月20日(2002.12.20)
(31)優先権主張番号　792077
(32)優先日　平成9年1月31日(1997.1.31)
(33)優先権主張国　米国(US)

(73)特許権者　398038580
ヒューレット・パッカード・カンパニー
HEWLETT-PACKARD COMPANY
アメリカ合衆国カリフォルニア州パロアルト　ハノーバー・ストリート　3000
(74)代理人　100063897
弁理士　古谷　馨
(74)代理人　100076680
弁理士　溝部　孝彦
(74)代理人　100087642
弁理士　古谷　聡

最終頁に続く

(54)【発明の名称】共用リストを備えたメモリの書き込み動作を行う多ノードシステム及び方法

(57)【特許請求の範囲】
【請求項１】
多ノードシステム内で共用リストを備えたメモリの書き込み動作を行う方法であって、
前記共用リストに新たなノードを連結するＤＭＡ書き込みリンク・コマンドを発行し、該共用リストが、各ノード内のキャッシュメモリ中にコピーされた共通のメモリ行を有するノードのリストであり、
前記新たなノードを前記共用リストに連結している間に前記メモリと前記共用リストとの間の接続を維持し、該維持ステップが前記ＤＭＡ書き込みリンク・コマンドにより実行可能となる、
という各ステップを含む方法。
【請求項２】
前記リンク・コマンドが前記新たなノードにより発行される、請求項１に記載の方法。
【請求項３】
前記共用リスト、前記新たなノード、及び前記メモリが、複数のポインタによって接続される、請求項１に記載の方法。
【請求項４】
前記新たなノードが、前記メモリと前記共用リストとの間に接続される、請求項１に記載の方法。
【請求項５】
前記共用リスト中の各ノードにパージ・コマンドを発行し、

各ノードに対する該パージ・コマンドの発行が完了した後に前記メモリから前記共用リストを切り離すためのコラプス・コマンドを発行する、
という各ステップを更に含む、請求項１に記載の方法。
【請求項６】
データパケットを用いて前記システム全体にわたり前記書き込み動作のためのコマンドを分配するステップを更に含む、請求項１に記載の方法。
【請求項７】
共用リストを備えたメモリのＤＭＡ書き込み動作を用いる多ノードシステムであって、
前記共用リストに新たなノードを連結するＤＭＡ書き込みリンク・コマンドであって、前記共用リストが、各ノード内のキャッシュメモリ中にコピーされた共通のメモリ行を有するノードのリストである、ＤＭＡ書き込みリンク・コマンドと、
前記新たなノードを前記共用リストに連結している間に前記メモリと前記共用リストとの間の接続を維持する手段であって、前記ＤＭＡ書き込みリンク・コマンドにより実行可能となる手段と
を備えているシステム。
【請求項８】
前記共用リスト、前記新たなノード、及び前記メモリが、複数のポインタにより接続され、
前記接続が、前記新たなノードと前記メモリとの間で行なわれる、
請求項７に記載のシステム。
【請求項９】
前記共用リスト中の各ノードをパージする手段と、
各ノードのパージが完了した後に前記メモリから前記共用リストを切り離す手段と
を更に含む、請求項７に記載のシステム。
【請求項１０】
前記共用リスト中の各ノードを互いに切り離す手段を更に含む、請求項９に記載のシステム。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】
本発明は、一般的には、多ノード、多重プロセッサ、キャッシュコヒーレント、不均一アクセスシステムにおけるメモリアクセスに関し、特定的には、かかるシステムにおいて書き込みパージ動作を実行するシステムおよび方法に関する。
【０００２】
【従来の技術】
標準スケーラブル・コヒーレント・インターフェース（SCI)仕様でのSCI ダイレクト・メモリ・アクセス（DMA)書き込み動作は、キャッシュ行を完全に上書きするものである。DMA 装置が行の一部のバイトの更新のみを行なっている場合、他のバイトのメモリに含まれる情報が、完全な上書きによって失われる。
【０００３】
【発明が解決しようとする課題】
DMA 書き込み動作は又、キャッシュのコヒーレンシーを維持するのに、ソフトウエアに頼っている。２つの装置が同じ行に書き込みを行なっている場合、第２の装置は、行が第１の装置のためにパージされている場合でさえも、古いデータを含むすべてのキャッシュのパージを完了したものと見なす可能性がある。そうならば、この第２の装置が、他のソフトウエアに陳腐化したデータを読み出させる可能性がある。ソフトウエアが陳腐化したデータを使用すると、プログラムエラーが生じることになる。
【０００４】
部分行を書き込む他の従来技術の方法では、指定されたバイトを更新する前に、行を局所キャッシュに読み込むことに頼っている。この結果、記憶された読み出しデータとの衝突

のために、他の所望のデータがキャッシュからスワップアウトされる。最後には、この結果、現在実行中の他の処理の性能が低下する。これは、余分なメモリ動作と影響を受けたデータの再取り出しに関連した待ち時間に起因する。

【０００５】

したがって、当該技術分野において、ソフトウエアに陳腐化したデータを使用させない書き込みコマンドを有する方法およびシステムが必要とされている。

【０００６】

さらに、当該技術分野において、メモリバイトの更新前に、メモリ行を局所キャッシュに読み込む必要のない方法及びシステムが必要とされている。

【０００７】

【課題を解決するための手段】

以上の及び他の目的及び特徴は、SCI 仕様に記載されるDMA 書き込みと同じ概略フローにしたがうシステムにおいて達成されるが、本発明のシステム及び方法では、メモリからキャッシュ共用リストは消去されない。その代わりに、共用リストが結合される。これによって、メモリ行がキャッシュに依然として入れられている間、他のノードからの他の書き込みパージが、その動作を終了したものと見なすことが防止される。共用リストが存在しない場合、コマンドによって供給されるマスクを用いて、新たなデータがメモリに併合される。本システム及び方法は、遠隔キャッシュ内の陳腐化したデータを突き止め、該データを破棄する代わりに、マスクを用いて該データをメモリ行に併合する。

【０００８】

本発明の１つの技術的利点は、メモリと共用リストとの間の接続を維持しながら、新たなノードを共用リストに結合する書き込みパージ・コマンドを発行することにある。

【０００９】

本発明の他の技術的利点は、メモリと共用リストとの間の接続を維持しながら、新たなノードに、共用リスト中の各ノードへとパージ・コマンドを発行させることにある。

【００１０】

本発明の他の技術的利点は、各ノードにパージ・コマンドが発行された後、新たなノードに、メモリから共用リストを切り離すためのコラプス・コマンドを送らせることにある。このコラプス・コマンドは、共用リストの破壊を完了させるものである。

【００１１】

本発明の他の技術的利点は、書き込みパージ・コマンドと共に書き込みマスクを用いることにある。

【００１２】

以上は、以下の本発明の詳細な説明を更に良く理解できるように、本発明の特徴及び技術的利点をかなり大まかに概説したものである。本発明の更なる特徴及び利点についても以下に説明するが、それらは、本願の特許請求の範囲の内容を形成するものである。当業者には明らかであろうが、本明細書に開示される概念及び特定の実施形態は、本発明の同じ目的を達成するために、変更あるいは他の構造の設計の基礎として容易に利用することができる。また、当業者には理解されるであろうが、かかる等価な構成は、特許請求の範囲に記載の本発明の精神及び範囲から逸脱するものではない。

【００１３】

【発明の実施の形態】

図１から図３は、従来技術のSCI 仕様により、共用リストを用いて、フレッシュな行へのDMA 書き込みが如何に行われるかを示す。このリストは、フレッシュ状態にあるメモリ101 のポインタが、ノード０ 102を指すことで始まる。ノード０ 102は、head_freshであり、mid_valid であるノード１ 103への順方向ポインタを有する。ノード１ 103は、先頭すなわちノード０ 102に戻る逆方向ポインタ、及びノード２ 104への順方向ポインタを有する。ノード２ 104は、tail_validであり、中間すなわちノード１ 103への逆方向ポインタを有する。

【００１４】

標準SCI フローによって実行される第１のステップは、このメモリ行の書き込みを他のノードが希望するか否かを判定することであり、希望する場合、MWRITE_FRESH_TO_HOMEが発行される。図１に示すように、ノード３ 105が、メモリ行への書き込みを希望しており、MWRITEコマンドを発行して書き込み動作を始める。図２は、リストfresh_to_home が、実際にメモリ101 から切り離されるので、ノード０ 102、ノード１ 103及びノード２ 104の共用リストが、もはやメモリ101 に接続されないことを示す。したがって、メモリはホームとしてマークされ、任意の新たなアクセスが、メモリ101 から直接メモリ行を検索することができる。

【００１５】

次に、ノード３ 105が入り、共用リストを片付けるすなわちパージする。ノード３ 105は、ノード３からノード０に、CREAD00_FRESH_TO_INVALIDを送る。ノード３ 105は、応答を受け取ると、その応答を用いて順方向ポインタを得る。ノード３ 105は、この順方向ポインタを用いて、ノード３ 105からノード１ 103に、CREAD00_VALID_TO_INVALIDを送る。ノード３ 105が応答を受け取ると、この応答からの順方向ポインタを用いて、ノード３ 105からノード２ 104に、CREAD00_VALID_TO_INVALIDを送る。この動作が完了する又は行われると、ホーム状態のメモリにのみデータのコピーが保持される。

【００１６】

SCI 仕様は、リストが完全にパージされるまで、フレッシュなメモリにアクセスしないようなソフトウエア・インターロックを頼りとしている。これは、同じメモリ行に対して、多数の書き込みパージがほとんど同時に発生する場合、非常に困難となり得る。というのは、２番目の書き込みパージは、その行がHOMEとマークされていることをただちに知り、ソフトウエアに、陳腐化したコピーを依然として持っている可能性のあるこの行をアクセスさせるためである。

【００１７】

図４から図６は、本発明の動作が図１に示す動作と同様に開始する、すなわちメモリ101 がフレッシュとマークされ、そのポインタがノード０ 102を指している初期状態で始まることを示す。ノード０ 102は、head_freshとマークされ、mid_valid であるノード１ 103への順方向ポインタを有する。mid_valid は、先頭への逆方向ポインタ、及びtail_validであるノード２ 104への順方向ポインタを有する。この場合も、ノード３ 105は、write_purge 又は新たなデータによるupdate_memory を希望している。したがって、ノード３ 105は、ノード３ 105からメモリ101 に、MWRITE_purgeコマンドである、MWRITE_PU_ATTACH_TO_LISTを送る。図５に示すように、メモリ101及びノード３ 105は、write_purge_partialコマンドからのマスク500 を用いて新たなデータに併合し、図２のようにリストを切り離さない。

【００１８】

したがってここで、リストは、フレッシュとマークされたメモリ101 に接続されたままである。メモリ101 の順方向ポインタがノード３ 105を指す。ノード３ 105がノード０ 102を指す。ノード０ 102は、ノード３ 105から何の通信も受け取っていないためhead_fresh状態にとどまる。ノード０ 102は、mid_valid ノード１ 103を指す順方向ポインタを有し、このmid_valid は、tail_validノード２ 104への順方向ポインタを有する。

【００１９】

ノード３ 105がmwrite_purgeコマンドからの応答を受け取ると、CREAD ではなくPURGE を実行する。というのは、この場合この行はFRESH であり、新たなデータを必要としないためである。したがって、mwrite_purge応答の後、ノード３ 105は、ノード３ 105からノード０ 102へのCPURGE_FRESH_TO_INVALID で、このリストのパージングを開始する。次に、ノード３ 105は、次の順方向ポインタを含むノード０からの応答を受け取り、ノード３ 105からノード１ 103に、CPURGE_VALID_TO_INVALID を送る。

【００２０】

ノード３ 105が、次の順方向ポインタを有するノード１ 103からの応答を受け取ると、ノード３ 105は、ノード３ 105からノード２ 104に、CPURGE_VALID_TO_INVALID を送る。ノ

ード３ 105は、tail_validノードであるノード２ 104に達したことを知ると、ノード３ 105は、ノード３ 105からメモリ101に、MUPDATE_LIST_TO_HOME を送り、メモリは図６に示すようにホーム状態にとどまる。かくして、本発明の方法によって、図３に示すものと同じ結果が得られる。

【００２１】

図７から図６は、SCI コントローラ内部の各種の状態マシーンがwrite_purge_partial フローを実行するために処理する動作を記述している。

【００２２】

図７は、メモリアクセスコントローラの要求状態マシーン論理に関する動作を記述している。この状態マシーンは、メモリアクセスコントローラからの要求を受け取り、メモリアクセスコントローラによって渡された状態とフローにしたがって、SCI コントローラ内の第１の状態を決定する。たとえば、write_purge の場合、メモリアクセスコントローラはhead_freshで要求し、SCI コントローラはこの要求をCS_HF_MODS_HD状態に初期化するか、別の場合には、head_fresh_MODS_only_head_dirty状態に初期化する。

【００２３】

図８は、その状態がリング上で要求に如何に変更されるかを記述している。状態がCS_HF_MODS_HD である前述の場合、この状態マシーンは次いで、mupdate_list_to_goneのリング要求を行なう。

【００２４】

図９から図１４は、応答状態マシーンに関する動作を記述している。この状態マシーンは、遠隔メモリ又はキャッシュから応答を受けたときに、何を行うかを記述している。この状態マシーンは、リングへの他の要求を行なうべきか、又はメモリアクセスコントローラへの応答を行なうべきかの決定を行い、場合によってはその両方を行う。やはり前述のCS_HF_MODS_HD 状態の場合に従うと、応答が無効にならずFRESH である場合、状態マシーンはOS_HD_INVAL_OD状態に遷移し、次に、図４の状態マシーンがこの状態をとり、新たな要求を生成する。

【００２５】

図１５から図１７の状態マシーンは、書き込みパージフローが活動状態にある間、他のノードから同じ行に対する要求を受け取った場合に、何が起きるかを記述している。やはり前述のOS_HF_MODS_HD 状態の場合に従うと、この状態マシーンは、cupdate_prevmid 又はcupdate_prevtailを完了させる。他の要求はすべて無効となる。prev-TAIL は、順方向ポインタを更新して、状態を変化させる。

【００２６】

図１８は、SCI リング上の送出要求パケット400 中の記号を示す。この特定パケット内の有意フィールドは第２の記号であり、バイトマスク401 と称するフィールドを含む。バイトマスク・フィールド401 は、write_purge_partial のためのバイトマスク情報の保持に用いられる。

【００２７】

図１９は、マスク500をより詳細に示す。このマスクは、それぞれが５ビットの開始部501と終了部502からなる。開始部がすべて０であり、終了部がすべて１である場合、これは０で始まり、31で終わったことを意味し、その行全体をwrite_purge で書き込んだことを意味する。他の値の使用すること、たとえば開始部を５、終了部を20とすることも可能であるが、サブフィールドは常に連続するバイトでなければならない。

【００２８】

本発明及びその利点を詳細に説明したが、特許請求の範囲に規定される本発明の精神及び範囲から逸脱することなく、各種の修正、置換及び変更が可能であることを理解されたい。

【００２９】

以下に、本発明の実施態様を列挙する。

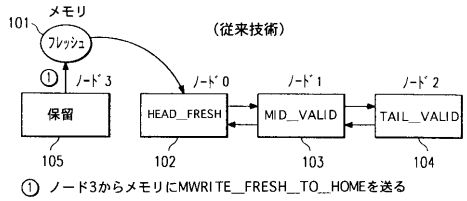
【００３０】

１．多ノードシステム内で共用リストを備えたメモリの書き込み動作を行う方法において、
前記共用リストに新たなノードを連結する書き込みリンク・コマンドを発行するステップと、
前記新たなノードを前記共用リストに連結している間、前記メモリと前記共用リストとの間の接続を維持するステップと、
を含むことを特徴とする方法。
【００３１】
２．前記リンク・コマンドは、前記新たなノードによって発行されることを特徴とする、前項１記載の方法。
【００３２】
３．前記共用リスト、前記新たなノード、及び前記メモリは、複数のポインタによって接続されることを特徴とする、前項１記載の方法。
【００３３】
４．前記新たなノードは、前記メモリと前記共用リストとの間に接続されることを特徴とする、前項１記載の方法。
【００３４】
５．前記方法は更に、
前記共用リスト中の各ノードに、パージ・コマンドを発行するステップと、
前記パージ・コマンドが各ノードに発行された後、前記メモリから前記共用リストを分離するためのコラプス・コマンドを発行するステップとを含み、
前記パージ・コマンドは、各ノードに記憶されたデータを無効にし、
前記パージ・コマンドは、前記新たなノードによって発行され、
前記コラプス・コマンドを発行するステップは又、前記共用リスト中の各ノードも互いに分離し、
前記コラプス・コマンドを発行するステップは又、前記新たなコマンドも前記メモリから分離し、
前記コラプス・コマンドは、前記新たなノードから発行されることを特徴とする前項１記載の方法。
【００３５】
６．前記方法は更に、
データパケットを用いて、前記システム全体に渡って、前記書き込み動作のためのコマンドを分配することを特徴とする、前項１記載の方法。
【００３６】
７．共用リストを備えたメモリの書き込み動作を用いる多ノードシステムにおいて、
共用リストに新たなノードを連結する手段と、
前記新たなノードを前記共用リストに連結している間、前記メモリと前記共用リストとの間の接続を維持する手段と、
からなることを特徴とするシステム。
【００３７】
８．前記共用リスト、前記新たなノード、及び前記メモリは、複数のポインタによって接続され、
前記接続は、前記新たなノードと前記メモリとの間で行なわれることを特徴とする、前項７記載のシステム。
【００３８】
９．前記システムは更に、
前記共用リスト中の各ノードをパージする手段と、
各ノードがパージされた後、前記メモリから前記共用リストを分離する手段とからなることを特徴とする、前項７記載の方法。
【００３９】

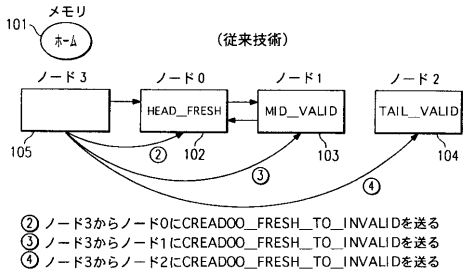
10．前記パージ手段は、各ノードに記憶されたデータを無効にし、前記システムは更に、前記共用リスト中の各ノードを互いに分離する手段を含み、
前記システムは、キャッシュコヒーレント不均一メモリアクセスを有するSCI システムであることを特徴とする、前項９記載のシステム。
【００４０】
【発明の効果】
本発明は上述のように、メモリからキャッシュ共用リストを消去せに結合することによって、メモリ行がキャッシュに依然として入れられている間、他のノードからの他の書き込みパージが、その動作を終了したものと見なすことが防止される。共用リストが存在しない場合、コマンドによって供給されるマスクを用いて、新たなデータをメモリに併合する。遠隔キャッシュ内の陳腐化したデータを突き止め、該データを破棄する代わりに、マスクを用いて該データをメモリ行に併合する。従って、他のソフトウエアに陳腐化したデータを読み出させることがないので、プログラムエラーの発生を防止することが可能になる。また、メモリバイトの更新前に、メモリ行を局所キャッシュに読み込む必要がないので、他の所望のデータがキャッシュからスワップアウトされて現在実行中の他の処理性能が低下する可能性が排除される。
【図面の簡単な説明】
【図１】　SCI 仕様の書き込みパージ動作の一例を示す。
【図２】　SCI 仕様の書き込みパージ動作の一例を示す。
【図３】　SCI 仕様の書き込みパージ動作の一例を示す。
【図４】本発明の書き込みパージ部分動作の一例を示す。
【図５】本発明の書き込みパージ部分動作の一例を示す。
【図６】本発明の書き込みパージ部分動作の一例を示す。
【図７】書き込みパージ動作を実行するために、初期化状態マシーンにより行われる動作を記述する。
【図８】書き込みパージ動作を実行するために、要求状態マシーンにより行われる動作を記述する。
【図９】書き込みパージ動作を実行するために、応答状態マシーンにより行われる動作を記述する。
【図１０】書き込みパージ動作を実行するために、応答状態マシーンにより行われる動作を記述する。
【図１１】書き込みパージ動作を実行するために、応答状態マシーンにより行われる動作を記述する。
【図１２】書き込みパージ動作を実行するために、応答状態マシーンにより行われる動作を記述する。
【図１３】書き込みパージ動作を実行するために、応答状態マシーンにより行われる動作を記述する。
【図１４】書き込みパージ動作を実行するために、応答状態マシーンにより行われる動作を記述する。
【図１５】書き込みパージ動作を実行するために、衝突状態マシーンにより行われる動作を記述する。
【図１６】書き込みパージ動作を実行するために、衝突状態マシーンにより行われる動作を記述する。
【図１７】書き込みパージ動作を実行するために、衝突状態マシーンにより行われる動作を記述する。
【図１８】バイトマスクに書き込みパージコマンドを持たせるための送出要求データパケットを示す。
【図１９】書き込みマスクを示す。
【符号の説明】
101　メモリ

102　共有リストのノード０
103　共有リストのノード１
104　共有リストのノード２
105　ノード３
400　送出要求パケット
401　バイトマスク

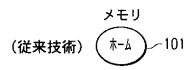
【図１】



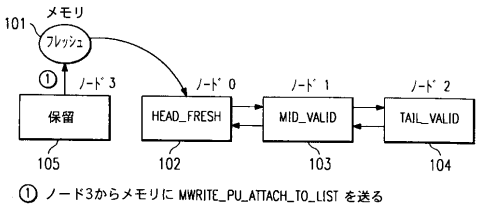
【図２】



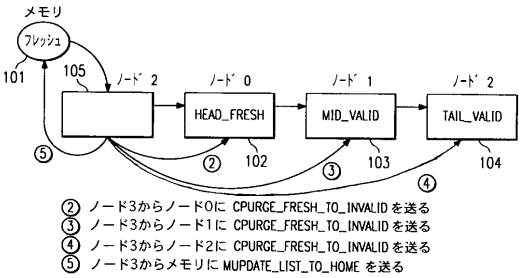
【図３】



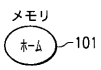
【図４】



【図５】



【図６】



【図７】

```
// wr_purge eri_table 初期化状態マシーン
// 現在の定常状態を第1の過渡状態
// にする。
switch (cstate) {
case ERI_CS_INVALID:
        eri_write(CS_PENDING,NULL,NULL) ;
        break;
case CS_ONLY_FRESH:
        eri_write(CS_OF_MODS_OD,NULL,NULL) ;
        break;
case CS_ONLY_DIRTY:
        eri_write(CS_OD_RETN_IN,NULL,NULL) ;
        break;
case CS_HEAD_FRESH:
        eri_write(CS_HF_MODS_HD,forw,NULL) ;
        break;
case CS_MID_VALID:
        eri_write(CS_MV_forw_MV,forw,back) ;
        break;
case CS_TAIL_VALID:
        eri_write(CI_TV_back_PEND,NULL,back) ;
        break;
case CS_HEAD_DIRTY:
        eri_write(CS_HD_INVAL_OD,forw,NULL) ;
        break;
default:
// 他の全てを局所的に処理可能な場合
        error () ;
        break;
}
```

【図８】

```
// wr_purge 要求状態マシーン
switch (cstate) {
case CS_PENDING:
        eri_request(MWRITE_PU,mem_addr) ;
        break;
case CS_QUEUED_JUNK:
        eri_request(CREAD,forw) ;
        break;
case CS_QUEUED_FRESH:
        eri_request(CPURGE,CC_FRESH_INVALID,forw) ;
        break;
case CS_HD_INVAL_OD:
        eri_request(CPURGE,CC_VALID_INVALID,forw) ;
        break;
case CS_HX_INVAL_XX:
        eri_request(CPURGE,CC_VALID_INVALID,forw) ;
        break;
case CS_OD_RETN_IN:
        eri_request(MWRITE,mem_addr) ;
        break;
case CS_OD_spin_IN:
        eri_request (MUPDATE,MC_LIST_TO_HOME,mem_addr) ;
        break;
case CS_TV_back_IN:
case CI_TV_back_PEND:
        eri_request(CUPDATE,CC_PREV_VTAIL,back) ;
        break;
case CS_MV_back_IN:
        eri_request(CUPDATE,CC_PREV_VMID,back) ;
        break;
case CS_MV_forw_MV:
        eri_request(CUPDATE,CC_NEXT_VMID,forw) ;
        break;
case CS_OF_MODS_OD:
        eri_request(MUPDATE,MC_LIST_TO_GONE,mem_addr) ;
        break;
case CS_HF_MODS_HD:
        eri_request(MUPDATE,MC_LIST_TO_GONE,mem_addr) ;
        break;
case CS_OF_retn_IN:
case CI_OF_retn_PEND:
        eri_request(MUPDATE,MC_LIST_TO_HOME,mem_addr) ;
        break;
case CI_HF_FORW_HX:
        eri_request(CUPDATE,CC_NEXT_FHEAD,forw) ;
        break;
case CS_HX_retn_IN:
        eri_request(MUPDATE,MC_PASS_HEAD,mem_addr) ;
        break;
}
```

【図９】

```
// wr_purge 応答状態マシーン
// レスポンダの前の状態を用いて、当方のキャッシュの
// 次の状態を判定する。
switch (cstate) {
case CS_PENDING:
    if (resp_nullified)
        eri_retry() ;
    else {
        switch (resp_rnstate) {
        case MS_HOME:
            mac_pass_d(NULL,NULL,NULL,data) ;
            free_eri_entry() ;
            break;
        case MS_FRESH:
            eri_write(CS_QUEUED_FRESH,resp_forw,NULL) ;
            break;
        case MS_GONE:
            eri_write(CS_QUEUED_JUNK,resp_forw,NULL) ;
            break;
        default:
            error() ;
            break;
        }
    }
    break;
case CS_QUEUED_FRESH:
    if (resp_nullified)
        eri_retry() ;
    else {
        switch (resp_cstate) {
        case CS_ONLY_FRESH:
        case CS_OF_retn_IN:
        case CI_OF_retn_PEND:
            eri_write(CS_OF_retn_IN,NULL,NULL) ;
            break;
        case CS_HEAD_FRESH:
            eri_write(CI_HX_INVAL_XX,resp_forw,NULL) ;
            break;
        case CS_HX_retn_IN:
            eri_write(CS_QUEUED_FRESH,resp_forw,NULL) ;
            break;
        default:
            error() ;
            break;
        }
    }
    break;
```

【図１０】

```
case CI_HX_INVAL_XX:
    if (resp_nullified)
        eri_retry() ;
    else {
        switch (resp_cstate) {
        case CS_TAIL_VALID:
        case CS_TV_back_IN:
        case CI_TV_back_PEND:
        case CS_ONLY_DIRTY:
        case CS_OD_spin_IN:
            eri_write(CS_OF_retn_IN,NULL,NULL);
            break;
        case CS_MID_VALID:
        case CS_MV_back_IN:
        case CS_MV_forw_MV:
        case CS_HEAD_DIRTY:
        case CS_HX_retn_IN:
            eri_write(CI_HX_INVAL_XX,resp_forw,NULL) ;
            break;
        default:
            error() ;
            break;
        }
    }
    break;
case CS_QUEUED_JUNK:
    if (resp_nullified)
        eri_retry() ;
    else {
        switch (resp_cstate)
        case CS_ONLY_DIRTY:
        case CS_HEAD_DIRTY:
        case CS_OD_spin_IN:
            eri_write(CS_HD_INVAL_OD,forw,NULL) ;
            break;
        case CS_HX_retn_IN:
            eri_write(CS_QUEUED_JUNK,resp_forw,NULL) ;
            break;
        default:
            error() ;
            break;
        }
    }
    break;
case CS_HD_INVAL_OD:
    if (resp_nullified)
        eri_retry() ;
    else {
        switch (resp_cstate) {
        case CS_TAIL_VALID:
        case CS_TV_back_IN:
        case CI_TV_back_PEND:
```

【図１１】

```
// この時点からまさに ONLY_DIRTY ロールアウトのように見える
                    eri_write(CS_OD_RETN_IN,forw,NULL) ;
                    break;
          case CS_MID_VALID:
          case CS_MV_back_IN:
          case CS_MV_forw_MV:
                    eri_write(CS_HD_INVAL_OD,resp_forw,NULL) ;
                    break;
          default:
                    error() :
                    break;
          }
     }
     break;
case CS_OD_RETN_IN:
     if (resp_nullified)
          eri_write(CS_OD_spin_IN,NULL,NULL) ;
     else {
          switch (resp_mstate) {
          case MS_GONE:
                    mac_write(ERI_CS_INVALID,NULL,NULL) ;
                    free_eri_entry() ;
                    break;
          default:
                    error() :
                    break;
          }
     }
     break;
case CS_OD_spin_IN:
     if (resp_nullified)
          eri_retry() ;
     else {
          switch (resp_mstate) {
          case MS_GONE:
                    eri_write(CS_OD_RETN_IN,NULL,NULL) ;
                    break;
          default:
                    error() :
                    break;
          }
     }
     break;
case CS_OF_MODS_OD:
     if (resp_nullified)
          eri_write(CS_OF_retn_PEND,NULL,NULL) ;
     else {
          switch (resp_mstate) {
          case MS_FRESH:
```

【図１２】

```
// この時点からまさに ONLY_DIRTYロールアウトのように見える
                    eri_write(CS_OD_RETN_IN,forw,NULL) ;
                    break;
          default:
                    error() :
                    break;
          }
     }
     break;
caseCI_OF_retn_PEND:
     if (resp_nullified)
          eri_retry() ;
     else {
          switch (resp_mstate) {
          case MS_FRESH:
                    eri_write(CI_PENDING,NULL,NULL) ;
                    break;
          default:
                    error() :
                    break;
          }
     }
     break;
case CI_TV_back_PEND:
     if (resp_nullified)
          eri_retry() ;
     else {
          switch (resp_cstate) {
          case CS_HEAD_DIRTY:
          case CS_HEAD_FRESH:
          case CS_MID_VALID:
          case CS_MV_forw_MV:
          case CI_HD_FORW_HX:
          case CI_HF_FORW_HX:
          case CS_HF_MODS_HD:
          case CS_HD_INVAL_OD:
          case CI_HX_INVAL_XX:
          case CI_HD_UPDATE_HD:
                    eri_write(CS_PENDING,NULL,NULL) ;
                    break;
          default:
                    error() ;
                    break;
          }
     }
     break;
case CI_TO_PENDING:
     eri_write(CS_PENDING,NULL,NULL) ;
     break;
```

【図１３】

```
          case CS_HF_MODS_HD:
               if (resp_nullified)
                    eri_write(CI_HF_FORW_HX,forw,NULL) ;
               else {
                    switch (resp_mstate) {
                    case MS_FRESH:
                         eri_write(CS_HD_INVAL_OD,forw,NULL) ;
                         break;
                    default:
                         error() :
                         break;
                    }
               }
               break;
          case CI_HF_FORW_HX:
               if (resp_nullified)
                    eri_retry() ;
               else {
                    switch (resp_cstate) {
                    case CS_MID_VALID:
                    case CS_TAIL_VALID:
                         eri_write(CS_HX_retn_IN,NULL,NULL) ;
                         break;
                    default:
                         error() ;
                         break;
                    }
               }
               break;
          case CI_HF_FORW_OF:
               eri_write(CI_OF_retn_PEND,NULL,NULL) ;
               break;
          case CS_HX_retn_IN:
               if (resp_nullified)
                    eri_retry() ;
               else {
                    switch (resp_mstate) {
                    case MS_FRESH:
                    case MS_GONE:
                         eri_write(CS_PENDING,NULL,NULL) ;
                         break;
                    default:
                         error() :
                         break;
                    }
               }
               break;
          case CS_MV_back_IN:
               if (resp_nullified)
                    eri_retry() ;
```

【図１４】

```
               else {
                    switch (resp_cstate) {
                    case CS_HEAD_DIRTY:
                    case CS_HEAD_FRESH:
                    case CS_MID_VALID:
                    case CS_MV_forw_MV:
                    case CI_HD_FORW_HX:
                    case CI_HF_FORW_HX:
                    case CS_HF_MODS_HD:
                    case CS_HD_INVAL_OD:
                    case CI_HX_INVAL_XX:
                    case CI_HD_UPDATE_HD:
                         eri_write(CS_PENDING,NULL,NULL) ;
                         break;
                    default:
                         error() ;
                         break;
                    }
               }
               break;
          case CS_MV_forw_MV:
               if (resp_nullified)
                    eri_retry() ;
               else {
                    switch (resp_cstate) {
                    case CS_MID_VALID:
                    case CS_TAIL_VALID:
                         eri_write(CS_MV_back_IN,NULL,back) ;
                         break;
                    default:
                         error() ;
                         break;
                    }
               }
               break;
          case CS_TV_back_IN:
               if (resp_nullified)
                    eri_retry() ;
               else {
                    switch (resp_cstate) {
                    case CS_HEAD_DIRTY:
                    case CS_HEAD_FRESH:
                    case CS_MID_VALID:
                    case CS_MV_forw_MV:
                    case CI_HD_FORW_HX:
                    case CI_HF_FORW_HX:
                    case CS_HF_MODS_HD:
                    case CS_HD_INVAL_OD:
                    case CI_HX_INVAL_XX:
                    case CI_HD_UPDATE_HD:
                         mac_write(ERI_CS_INVALID,NULL,NULL) ;
                         free_eri_entry;
                         break;
```

【図１５】

```
// wr_purge 衝突要求状態マシーン
// テーブル内のキャッシュ行の現在の状態と到来要求を
// 用いて、新しいキャッシュ状態を判定する。　以下の文に
// リストされないものを無効にする。

switch (cstate)
case CS_MV_back_IN:
        switch (req_in)
        case CPURGE:
                eri_write(CS_TO_INVALID, forw, back) ;
                break;
        case CINC:
                // no state change, but don't nullify
                break;
        break;
case CS_MV_forw_MV:
        switch (req_in)
        case CPURGE:
                eri_write(CS_TO_PENDING, forw, back) ;
                break;
        case CUPDATE, PREV_VMID:
                // no state change, but change forw ptr to passed ptr
                eri_write(cstate, req_ptr, back) ;
                break;
        case CUPDATE, PREV_VTAIL:
                eri_write(CI_TO_TV_back_PEND, forw, back_)
                break;
        break;
case CS_OD_spin_IN:
        switch (req_in)
        case CUPDATE, PEND_VALID:
        case CREAD:
                eri_write(CS_TO_TV_back_IN, forw, back) ;
                break;
case CPURGE, VALID_INVALID:
                eri_write(CS_TO_INVALID, forw, back) ;
                break;
break;
```

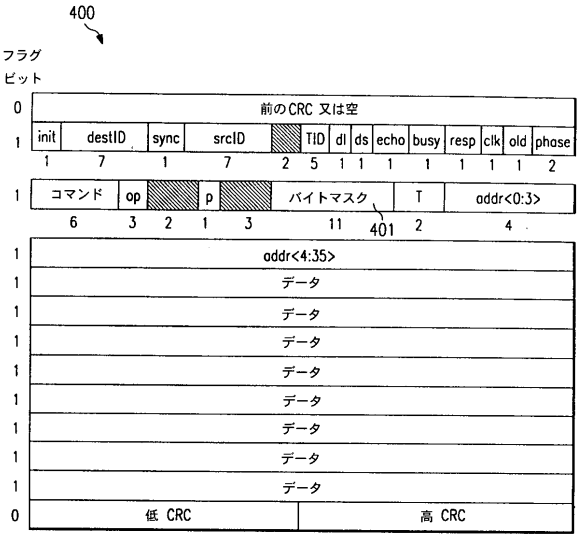
【図１６】

```
case CS_HF_MODS_HD:
        switch (req_in)
        case CUPDATE, PREV_VMID:
                //状態変化はないが、順方向ポインタを通過ポインタに変える
                eri_write(cstate, req_ptr, back) ;
                break;
        case CUPDATE, PREV_VTAIL:
                eri_write(CS_OF_MODS_OD, forw, back) ;
                break;
        break;
case CS_OF_retn_IN:
        switch (req_in)
        case CUPDATE, PEND_VALID:
        case CREAD:
                eri_write(CS_TO_TV_back_IN, forw, back) ;
                break;
        case CPURGE, FRESH_INVALID:
                eri_write(CS_TO_INVALID, forw, back) ;
                break;
        break;
case CS_TV_back_IN:
        switch (req_in)
        case CPURGE, VALID_INVALID:
                eri_write(CS_TO_INVALID, forw, back) ;
                break;
        break;
case CI_HF_FORW_HX:
        switch (req_in)
        case CUPDATE, PREV_VMID:
                // 状態変化はないが、順方向ポインタを通過ポインタに変える
                eri_write(cstate, req_ptr, back) ;
                break;
        case CUPDATE, PREV_VTAIL:
                eri_write(CI_HF_FORW_OF, forw, back) ;
                break;
        break;
```

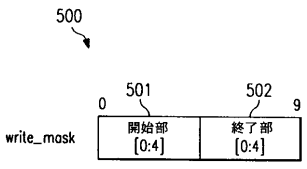
【図１７】

```
case CI_HD_fORW_HX:
        switch (req_in)
        case CUPDATE, PREV_VMID:
                eri_write(cstate, req_ptr, back) ;
                //状態変化はないが、順方向ポインタを通過ポインタに変える
                break;
        case CUPDATE, PREV_VTAIL:
                eri_write(CI_HD_FORW_OD, forw, back) ;
                break;
        break;
case CS_HX_retn_IN:
        switch (req_in)
        case CUPDATE, PEND_VALID:
        case CREAD:
        case CPURGE, FRESH_INVALID:
        case CPURGE, VALID_INVALID:
                eri_write(CI_TO_PENDING, forw, back) ;
                break;
        break;
case CI_TV_back_PEND:
        switch (req_in)
        case CPURGE, VALID_INVALID:
                eri_write(CI_TO_PENDING, forw, back) ;
                break;
        break;
case CI_HD_UPDATE_HD:
        // この状態に変化なし（到達していない場合）
        break;
case CI_OF_retn_PEND:
        switch (req_in)
        case CUPDATE, PEND_VALID:
        case CREAD:
                eri_write(CI_TO_TV_back_PEND, forw, back) ;
                break;
        case CPURGE, FRESH_INVALID:
                eri_write(CI_TO_PENDING, forw, back) ;
                break:
        break;
default:
        break;
```

【図１８】



【図１９】



フロントページの続き

(72)発明者 ブライアン・ホーナング
アメリカ合衆国テキサス州７５０７５，プラノ，クリークフィールド・ドライヴ・１１０８
(72)発明者 ブライアン・マリエッタ
アメリカ合衆国テキサス州７８７２９，オースティン，ブーマー・レーン・１３１１７

審査官 清木 泰

(56)参考文献 特開平３－３１９４７（ＪＰ，Ａ）
特開平４－３４８４５０（ＪＰ，Ａ）
特開平９－２５１４４２（ＪＰ，Ａ）
特開平１０－２３２８３５（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F12/08-12/12
G06F15/16-15/177