(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4082612号
(P4082612)
(45)発行日　平成20年4月30日(2008.4.30)　(24)登録日　平成20年2月22日(2008.2.22)

(51)Int.Cl.　F I
G06F 12/08 (2006.01)　G06F 12/08 531B
G06F 12/08 565
G06F 12/08 551C
G06F 12/08 523Z

請求項の数 31　(全 16 頁)

(21)出願番号　特願2004-185622(P2004-185622)
(22)出願日　平成16年6月23日(2004.6.23)
(65)公開番号　特開2005-18772(P2005-18772A)
(43)公開日　平成17年1月20日(2005.1.20)
審査請求日　平成16年6月23日(2004.6.23)
(31)優先権主張番号　10/603252
(32)優先日　平成15年6月25日(2003.6.25)
(33)優先権主張国　米国(US)

(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（番地なし）
(74)代理人　100086243
弁理士　坂口　博
(74)代理人　100091568
弁理士　市位　嘉宏
(74)代理人　100108501
弁理士　上野　剛史

最終頁に続く

(54)【発明の名称】複数のコヒーレンシ領域およびキャッシュ・パージのないコヒーレンシ領域間ソフトウェア・プロセス移行を備えるマルチプロセッサ・コンピュータ・システム

(57)【特許請求の範囲】
【請求項１】
マルチプロセッサ・コンピュータ・システムであって、
複数の処理ノードと、
前記処理ノードに関連付けられたキャッシュを使用する複数の動的キャッシュ・コヒーレンシ領域と
を含み、
１つまたは複数の前記処理ノード中のキャッシュ内容の選択的パージを必要とせずに、前記複数のキャッシュ・コヒーレンシ領域間でのソフトウェア・プロセスの移動を開始するソフトウェアの動きを制御する、前記処理ノード中のキャッシュ・コントローラ論理を有し、
２つの別々のセットの処理ノード間でソフトウェア・プロセスを移動する場合に、キャッシュ・ラインを２つの別々のコヒーレンシ領域において共用とマーク付けすることができず、コヒーレンシ領域を処理ノードのあるセットから処理ノードの他のセットに移動するときに、旧処理ノード上のコヒーレンシ領域についてキャッシュ・エントリを実際に残したままにし、かつこれらの旧キャッシュ・エントリが新しい処理ノードからの着信記憶要求に見えるようにし、かつ旧エントリが無効化されるまでは同一の主記憶アドレスについてのキャッシュ・エントリが新しい処理ノード中で確立されないようにする、
前記マルチプロセッサ・コンピュータ・システム。
【請求項２】

要求された記憶アドレスのキャッシュ・エントリを、着信記憶要求を開始したプロセッサを含む処理ノード中のいずれかのキャッシュ上に存在させるようにするときには、該記憶アドレスについての前記着信記憶要求に応じて、処理ノード中の前記キャッシュ・コントローラ論理に、現在のコヒーレンシ領域モードによって指定される、前記プロセッサの現在のコヒーレンシ領域の外部に、該要求された記憶アドレスのコピーが存在しないようにする処理を実行させるためのスーパバイザ・ソフトウェアを含む、請求項１に記載のマルチプロセッサ・コンピュータ・システム。

【請求項３】

前記マルチプロセッサ・コンピュータ・システムが、前記スーパバイザ・ソフトウェアを実行して、それ自体のコヒーレンシ領域を有する各プロセスについて一意のコヒーレンシ領域ＩＤを生成する、請求項２に記載のマルチプロセッサ・コンピュータ・システム。

【請求項４】

前記マルチプロセッサ・コンピュータ・システムが、前記スーパバイザ・ソフトウェアを実行して、各処理ノード上に現在ディスパッチ可能なすべてのコヒーレンシ領域ＩＤについてエントリを有するテーブルをシステム内の前記各処理ノードごとに生成する、請求項２に記載のマルチプロセッサ・コンピュータ・システム。

【請求項５】

前記マルチプロセッサ・コンピュータ・システムが、前記スーパバイザ・ソフトウェアを実行して、それ自体のコヒーレンシ領域を有する各プロセスについて一意のコヒーレンシ領域ＩＤとマルチプロセッサ・コンピュータ・システム中の各プロセッサについて１つまたは複数のコヒーレンシ・モード・ビットとを生成し、そのプロセッサによって開始される各記憶トランザクションが、通信のために前記マルチプロセッサ・コンピュータ・システムの他のプロセッサに送信されるとき、このプロセッサに関連する前記コヒーレンシ・モード・ビットおよびコヒーレンシ領域ＩＤがそのトランザクションと共に送信される、請求項２に記載のマルチプロセッサ・コンピュータ・システム。

【請求項６】

前記マルチプロセッサ・コンピュータ・システムが、前記スーパバイザ・ソフトウェアを実行して、それ自体のコヒーレンシ領域を有する各プロセスについて一意のコヒーレンシ領域ＩＤとマルチプロセッサ・コンピュータ・システム中の各プロセッサについて１つまたは複数のコヒーレンシ・モード・ビットとを生成する、請求項２に記載のマルチプロセッサ・コンピュータ・システム。

【請求項７】

前記マルチプロセッサ・コンピュータ・システムが、前記スーパバイザ・ソフトウェアを実行して、それ自体のコヒーレンシ領域を有する各プロセスについて一意のコヒーレンシ領域ＩＤとマルチプロセッサ・コンピュータ・システム中の各プロセッサについて１つまたは複数のコヒーレンシ・モード・ビットとを生成し、前記マルチプロセッサ・コンピュータ・システムのプロセッサのいずれかから受信するいずれかの記憶トランザクションにどのキャッシュが関与しなければならないかを判断するために、各トランザクションに関連する前記モード・ビットが使用される、請求項２に記載のマルチプロセッサ・コンピュータ・システム。

【請求項８】

前記マルチプロセッサ・コンピュータ・システムが、前記スーパバイザ・ソフトウェアを実行して、それ自体のコヒーレンシ領域を有する各プロセスについて一意のコヒーレンシ領域ＩＤとマルチプロセッサ・コンピュータ・システム中の各プロセッサについて１つまたは複数のコヒーレンシ・モード・ビットとを生成し、かつコヒーレンシ領域間でソフトウェア・プロセスを移動させるいくつかの動作の間に、複数のキャッシュ・コヒーレンシ領域がキャッシュ・パージを使用せずに動作することを可能にする、請求項２に記載のマルチプロセッサ・コンピュータ・システム。

【請求項９】

前記マルチプロセッサ・コンピュータ・システムが、前記スーパバイザ・ソフトウェア

を実行して、ソフトウェア・プロセスを、もはや前記ソフトウェア・プロセスによって使用される予定のないあるコヒーレンシ領域から、前者と同じアドレス空間をカバーするように生成されているが、新しい処理ノードのセットを含むことになる他のコヒーレンシ領域に移動させる、請求項８に記載のマルチプロセッサ・コンピュータ・システム。

【請求項１０】

前記マルチプロセッサ・コンピュータ・システムが、前記スーパバイザ・ソフトウェアを実行して、それ自体のコヒーレンシ領域を有する各プロセスごとに一意のコヒーレンシ領域ＩＤを生成し、ある１組の処理ノードを含むあるコヒーレンシ領域から他の１組の処理ノードを含む他のコヒーレンシ領域に、どの処理ノード中のキャッシュについてもキャッシュ・パージを必要とせずに、ソフトウェア・プロセスを移動する、請求項２に記載のマルチプロセッサ・コンピュータ・システム。

【請求項１１】

前記他のコヒーレンシ領域が、元のコヒーレンシ領域より少数のハードウェア処理ノードを含む場合、前記処理ノードについてのコヒーレンシ領域のサイズが実際に縮小される、請求項１０に記載のマルチプロセッサ・コンピュータ・システム。

【請求項１２】

複数の前記処理ノードを有するマルチプロセッサ・コンピュータ・システムが、各処理ノードに関連するアクティブ・コヒーレンシ領域情報のテーブルを使用して、その処理ノードのキャッシュ状態遷移をいつ変更するかを判断する、請求項１に記載のマルチプロセッサ・コンピュータ・システム。

【請求項１３】

前記マルチプロセッサ・コンピュータ・システムが、スーパバイザ・ソフトウェアを実行して、各処理ノードに関連する前記テーブルを初期化し、その処理ノード上で使用される予定の各コヒーレンシ領域ごとに前記テーブル中のエントリが作成される、請求項１２に記載のマルチプロセッサ・コンピュータ・システム。

【請求項１４】

前記マルチプロセッサ・コンピュータ・システムが、スーパバイザ・ソフトウェアを実行して、コヒーレンシ領域によって含まれる記憶アドレスにアクセスするすべてのソフトウェア処理と関連づけることができる各コヒーレンシ領域ごとに、一意のコヒーレンシ領域ＩＤを割り当てる、請求項１に記載のマルチプロセッサ・コンピュータ・システム。

【請求項１５】

処理ノードが、もはやスーパバイザ・ソフトウェアによって現在そのノード上にディスパッチ可能であるどのソフトウェア・プロセスのアドレス空間の一部でもないラインをターゲットとする着信記憶要求を識別できる、請求項１に記載のマルチプロセッサ・コンピュータ・システム。

【請求項１６】

処理ノードが、コヒーレンシ領域外部からの記憶要求に応答して、もはやスーパバイザ・ソフトウェアによって現在そのノード上にディスパッチ可能であるいずれのソフトウェア・プロセスのアドレス空間の一部でもないラインをターゲットとする着信記憶要求を識別でき、それによって、その処理ノード上のいずれのソフトウェア・プロセスによってももはや実際に使用されていないキャッシュ・ラインを識別でき、その処理ノードについてキャッシュ・エントリを無効に変更できる、請求項１に記載のマルチプロセッサ・コンピュータ・システム。

【請求項１７】

処理ノードが、もはやスーパバイザ・ソフトウェアによって現在そのノード上にディスパッチ可能であるいずれのソフトウェア・プロセスのアドレス空間の一部でもないラインをターゲットとする着信記憶要求を識別でき、あるソフトウェア・プロセスについてコヒーレンシ境界が実際に変更されている間もマルチプロセッサ・コンピュータ・システム中のキャッシュのすべてがコヒーレンシ・トランザクションの処理を続けることができるようにする、請求項１に記載のマルチプロセッサ・コンピュータ・システム。

【請求項１８】
処理ノードが、もはやスーパバイザ・ソフトウェアによって現在そのノード上にディスパッチ可能であるいずれのソフトウェア・プロセスのアドレス空間の一部でもないラインをターゲットとする着信記憶要求を識別でき、その結果、もはや所与の処理ノード上に実際にディスパッチされていないソフトウェア・プロセスに属するキャッシュ・ラインが識別され、無効化され、それによって、それらの再利用が可能になる、請求項１に記載のマルチプロセッサ・コンピュータ・システム。
【請求項１９】
前記マルチプロセッサ・コンピュータ・システムが、スーパバイザ・ソフトウェアを実行して、プロセッサ状態情報を使用して、単一の発信元プロセッサの着信記憶要求によって生成されたコヒーレンシ・トランザクションを検査するのにマルチプロセッサ・コンピュータ・システム中のどのキャッシュが必要かを判断する、請求項１に記載のマルチプロセッサ・コンピュータ・システム。
【請求項２０】
マルチプロセッサ・コンピュータ・システム中の処理ノードが動的コヒーレンシ境界を有しており、その結果、そのマルチプロセッサ・コンピュータ・システムが単一のワークロードのために任意の特定の時点でシステム中の全プロセッサのサブセットだけを使用し、
前記マルチプロセッサ・コンピュータ・システムが、前記スーパバイザ・ソフトウェアを実行して、任意の単一のワークロードを実行させるために使用されるプロセッサの数を拡大および縮小する際にキャッシュ・コヒーレンシを最適化できる、請求項１９に記載のマルチプロセッサ・コンピュータ・システム。
【請求項２１】
大規模マルチプロセッサ・システムを作成するために、処理ノードの複数のインスタンスをセカンド・レベル・コントローラと接続することができる、請求項２０に記載のマルチプロセッサ・コンピュータ・システム。
【請求項２２】
前記マルチプロセッサ・コンピュータ・システムが、前記スーパバイザ・ソフトウェアを実行して、マルチプロセッサ・コンピュータ・システム中の各プロセッサについてそれ自体のコヒーレンシ領域と１つまたは複数のコヒーレンシ・モード・ビットを有する各プロセスについて一意のコヒーレンシ領域ＩＤを生成し、かつコヒーレンシ領域間でソフトウェア・プロセスを移動するいくつかの動作の間にキャッシュ・パージを使用せずに、複数のキャッシュ・コヒーレンシ領域が動作することを可能にし、ノード・コントローラが、前記コヒーレンシ・モード・ビットを使用して、どのプロセッサがノード・コントローラによって受信された任意の所与のトランザクションを受信しなければならないかを判断する、請求項２１に記載のマルチプロセッサ・コンピュータ・システム。
【請求項２３】
セカンド・レベル・コントローラが、モード・ビットを使用して、どの処理ノードがセカンド・レベル・コントローラによって受信された任意の所与のトランザクションを受信しなければならないかを判断する、請求項２２に記載のマルチプロセッサ・コンピュータ・システム。
【請求項２４】
前記マルチプロセッサ・コンピュータ・システムが、前記スーパバイザ・ソフトウェアを実行して、物理プロセッサにマップされた論理区分を使用し、ハイパーバイザを使用する各区分ごとに別個のキャッシュ・コヒーレンシ領域が定義できる、請求項２１に記載のマルチプロセッサ・コンピュータ・システム。
【請求項２５】
コヒーレンシ領域ＩＤが、キャッシュ・コヒーレンシ・モードの機能を実施するために使用され、ノード・コントローラが、どの物理処理ノードが特定のコヒーレンシ領域ＩＤに関連づけられるかを判断する、請求項２に記載のマルチプロセッサ・コンピュータ・シ

ステム。
【請求項２６】
発信元コヒーレンシ領域中のすべてのキャッシュをミスしたどの着信記憶要求も、モード・ビットの設定にかかわらず、次いで全体システム中のすべての処理ノードに転送される、請求項３に記載のマルチプロセッサ・コンピュータ・システム。
【請求項２７】
発信元のコヒーレンシ領域中でヒットするが、正しいキャッシュ状態を有していない要求であるどの着信記憶要求も、該コヒーレンシ領域の外部に送信する必要がない、請求項３に記載のマルチプロセッサ・コンピュータ・システム。
【請求項２８】
マルチプロセッサ・コンピュータ・システムであって、
複数の処理ノードと、
前記処理ノードに関連付けられたキャッシュを使用する複数の動的キャッシュ・コヒーレンシ領域と
を含み、
１つまたは複数の前記処理ノード中のキャッシュ内容の選択的パージを必要とせずに、前記複数のキャッシュ・コヒーレンシ領域間でのソフトウェア・プロセスの移動を開始するソフトウェアの動きを制御する、前記処理ノード中のキャッシュ・コントローラ論理を有し、
処理ノードが、コヒーレンシ領域外部からの記憶要求に応答して、もはやスーパバイザ・ソフトウェアによって現在そのノード上にディスパッチ可能であるいずれのソフトウェア・プロセスのアドレス空間の一部でもないラインをターゲットとする着信記憶要求を識別でき、それによって、その処理ノード上のいずれのソフトウェア・プロセスによってももはや実際に使用されていないキャッシュ・ラインを識別でき、その処理ノードについてキャッシュ・エントリを無効に変更できる、前記マルチプロセッサ・コンピュータ・システム。
【請求項２９】
マルチプロセッサ・コンピュータ・システムであって、
複数の処理ノードと、
前記処理ノードに関連付けられたキャッシュを使用する複数の動的キャッシュ・コヒーレンシ領域と
を含み、
１つまたは複数の前記処理ノード中のキャッシュ内容の選択的パージを必要とせずに、前記複数のキャッシュ・コヒーレンシ領域間でのソフトウェア・プロセスの移動を開始するソフトウェアの動きを制御する、前記処理ノード中のキャッシュ・コントローラ論理を有し、
処理ノードが、もはやスーパバイザ・ソフトウェアによって現在そのノード上にディスパッチ可能であるいずれのソフトウェア・プロセスのアドレス空間の一部でもないラインをターゲットとする着信記憶要求を識別でき、その結果、もはや所与の処理ノード上に実際にディスパッチされていないソフトウェア・プロセスに属するキャッシュ・ラインが識別され、無効化され、それによって、それらの再利用が可能になる、前記マルチプロセッサ・コンピュータ・システム。
【請求項３０】
複数の処理ノードと前記処理ノードに関連付けられたキャッシュを使用する複数の動的キャッシュ・コヒーレンシ領域とを含むマルチプロセッサ・コンピュータ・システム中で使用する方法であって、マルチプロセッサ・コンピュータ・システムに、
１つまたは複数の前記処理ノードにおけるキャッシュ内容の選択的パージを必要とせずに、前記複数のキャッシュ・コヒーレンシ領域間でソフトウェア・プロセスを移動するステップと、
２つの別々のセットの処理ノード間でソフトウェア・プロセスを移動する場合に、キャ

ッシュ・ラインを２つの別々のコヒーレンシ領域において共用とマーク付けすることができず、コヒーレンシ領域を処理ノードのあるセットから処理ノードの他のセットに移動するときに、旧処理ノード上のコヒーレンシ領域についてキャッシュ・エントリを実際に残したままにし、かつこれらの旧キャッシュ・エントリが新しい処理ノードからの着信記憶要求に見えるようにし、かつ旧エントリが無効化されるまでは同一の主記憶アドレスについてのキャッシュ・エントリが新しい処理ノード中で確立されないようにするステップとを実行させる、方法。

【請求項３１】

要求された記憶アドレスのキャッシュ・エントリを、着信記憶要求を開始したプロセッサを含む処理ノード中のいずれかのキャッシュ上に存在させるようにするときには、該記憶アドレスについての前記着信記憶要求に応じて、前記マルチプロセッサ・コンピュータ・システムに、現在のコヒーレンシ領域モードによって指定される、前記プロセッサの現在のコヒーレンシ領域の外部に、該要求された記憶アドレスのコピーが存在しないようにするステップをさらに実行させる、請求項３０に記載の方法。

【発明の詳細な説明】

【技術分野】

【０００１】

本発明は、複数のノードおよび動的キャッシュ・コヒーレンシ領域を有するマルチプロセッサ・コンピュータ・システムに関する。本発明は、詳細にはキャッシュ内容の選択的パージを必要とせずに、コヒーレンシ領域間でソフトウェア・プロセスを移動できるコンピュータ・システムに関するものである。

【０００２】

関連出願

本発明は、本明細書と同時に出願された、２００３年６月２３日出願の「Multiprocessor System with Dynamic Cache Coherency Regions」という名称の米国出願第１０／６０３，２５２号と関連するものである。

【０００３】

この同時係属出願および本出願は、同一の譲受人である、米国ニューヨーク州アーモンクのインターナショナル・ビジネス・マシーンズ・コーポレーションが所有している。

【０００４】

この同時係属出願中に示される記述内容は、この参照により本出願に組み込まれる。

【０００５】

登録商標

ＩＢＭ（登録商標）は、米国ニューヨーク州アーモンクのインターナショナル・ビジネス・マシーンズ・コーポレーションの登録商標である。その他の名称も、インターナショナル・ビジネス・マシーンズ・コーポレーションまたは他の企業の登録商標または製品名であることがある。

【背景技術】

【０００６】

メモリ参照が完了するのを待つ間にコンピュータ・プロセッサによって費やされるアイドル時間は、広範で重要な商業的および技術的なコンピューティング・ワークロードの全実行時間のうちの、はるかに大きな部分を占めるようになってきた。マルチプロセッサ・システム設計において、主記憶装置ロケーションへのアクセスが完了するまでにプロセッサが待たなければならない時間を最小限に抑えるために、多くの従来技術による手法が使用されてきた。これらの手法は、おおまかに２つのカテゴリに分類される。手法の第１のカテゴリでは、遅延を伴うメモリ参照を待つ間に、プロセッサが実行すべき追加の命令を見つけようと試みる。これらの手法には、順不同の実行やマルチスレッド処理などの、ソフトウェアおよびハードウェアの機構が含まれる。手法の第２のカテゴリでは、例えばＳＲＡＭキャッシュ、ＤＲＡＭキャッシュ、高速マルチプロセッサ・バス・アーキテクチャなど、メモリ参照自体の待ち時間を最小限に抑えることに焦点を合わせる。ＳＲＡＭおよ

びＤＲＡＭキャッシュは、メモリ参照の待ち時間短縮にきわめて有効であり、その片方または両方が現行のすべてのマルチプロセッサ設計によって使用されている。従来技術によるキャッシュ設計には、マルチプロセッサ・システムのキャッシュ・コヒーレンスを維持する専用のハードウェアおよびソフトウェアが含まれる。共用バスを介して複数のプロセッサを接続するシステムでは、一般にスヌープ・バス・プロトコルが使用される。共用バス上で実施される各コヒーレント・トランザクションは、バスに接続された他のすべてのデバイスのキャッシュ中のデータと突き合わせて検査される（すなわち「スヌープされる」）。影響を受けるデータのコピーが見つかった場合には、コヒーレント・トランザクションに応答してそのデータを含むキャッシュ・ラインの状態が更新されることがある。

【０００７】

キャッシュは、中程度の数のプロセッサを有するマルチプロセッサ・システムではうまく動作したが、多数のプロセッサに拡張されたとき、従来技術のマルチプロセッサ設計では、ＴＰＣ－Ｃベンチマークによってシミュレートされるトランザクションおよびデータベースのワークロードを含む多くの重要なワークロードに対してうまくスケール（scale）できない。

【０００８】

米国特許第４，８４３，５４１号に記載された論理区分化も、共用プロセッサを使用するとき、多数のプロセッサにまで拡張する場合、従来技術によるシステム設計に対してうまくスケールできない。米国特許第４，８４３，５４１号は、バーチャル・マシン・ハイパーバイザ・プログラムをどのように使用すれば、「データ処理システムの中央電子複合（central electronic complex）中のリソースを複数の論理区分に区分する」ことができるかを示している。論理区分化は、プライベート・データに対して同時に作用する多数のワークロードを実行させるために、大規模マルチプロセッサ・システム上で広く使用されている。論理区分化を使用する典型的なシステムでは、オペレーティング・システム・インスタンスが各論理区分内で初期化される。論理区分は、１～ｎ個の論理プロセッサを所有することができる。ハイパーバイザは、各論理プロセッサを物理プロセッサ上へディスパッチする役割を果たす。ある物理プロセッサが、長時間にわたり単一の論理プロセッサだけのホストである場合、それはこの論理プロセッサ区分に「専用」であると言われる。ある物理プロセッサが、複数の区分の論理プロセッサのホストである場合、それは「共用」プロセッサであると言われる。ハードウェア全体の利用率の点からは、大規模マルチプロセッサ・システムが、物理プロセッサの多くあるいは大部分を「共用」として定義するフレキシビリティを許容し、外部変化により物理プロセッサの使用状況が変動するとき、マルチプロセッサの物理プロセッサ間で論理プロセッサが移動できるようにすることが望ましい。従来技術によるマルチプロセッサ・キャッシュ設計では、これらの区分されたワークロードに対し、特に物理プロセッサが「共用」として定義されている場合にはうまくスケールできなかった。

【０００９】

大規模単一データベース・ワークロード（large single database workload）に対しても共用論理区分（shared logical partition）に対しても大規模マルチプロセッサのパフォーマンス・スケーリングが不十分である大きな要因は、増加するプロセッサの数とそれらの間での通信に必要な時間遅延との関係にある。スヌープ・バス・プロトコルでは、ローカル・キャッシュをミスしたメモリ参照を、要求された行のコピーを含む可能性があるすべてのキャッシュ、通常はシステム中の他のすべてのキャッシュにブロードキャストする必要がある。大規模マルチプロセッサ・システムのアドレスと応答を配送するために必要なバス帯域幅は非常に広い。必要な広帯域を提供する必要から、従来技術による設計では、多数のワイド・ポートを有するスイッチ・チップ（switch chips）、必要なピンを提供するための高価なチップ・キャリア、良好な電気的特性を、したがって高速バスを提供するための高価なカード技術、ワイド・バスを提供するための高価なカード・コネクタなどを使用せざるを得なかった。これらの要素すべてのコストが、大規模マルチプロセッサ・システムのコスト・パフォーマンスを向上させようとする際の重大な問題になってきた

。

【００１０】

従来技術の設計では、これら２つの問題、すなわちコヒーレンシ動作待ち時間およびアドレス帯域幅制限の問題を多くの異なる方法で解決しようと試みてきたが、それぞれ本発明が回避しようとしているシステム設計上の他のコストがかかるものであった。

【００１１】

例えばＩＢＭ　Ｓ／３９０　Ｇ４設計（「IBM Journal of Research and Development」、４１巻、４号および５号、１９９７年）に例示されるような大規模共用キャッシュが、この両方の問題に対処するために従来技術設計で使用されてきた。少数の大規模共用キャッシュの相互接続では、共用キャッシュ中でヒットした要求に対して良好な待ち時間が得られる。この包括的共用キャッシュ（inclusive shared cache）はまた、フィルタとしても働き、このフィルタにより、場合によってはシステム中のすべてのプロセッサにアドレスをブロードキャストする必要がなくなる。この設計も多数のプロセッサにうまくスケールできない。追加のプロセッサの使用により、この設計では、多数の配線層を有するマルチチップ・モジュールと、接続された各プロセッサにポートを提供するために必要な極めて多数の入出力（Ｉ／Ｏ）を有するＬ２キャッシュ・チップとを使用せざるを得なくなる。

【００１２】

例えばＳｅｑｕｅｎｔ　ＮＵＭＡ－Ｑ設計（「ＳＴｉＮＧ：A CC-NUMA Computer System for the Commercial Marketplace」、Proc. 23rd International Symposium of Computer Architecture、１９９６年５月）に例示される、ディレクトリを利用してリモート要求元によるローカル・メモリのアクセスを追跡するマルチプロセッサ・システムは、多数のプロセッサのために必要とされるアドレス帯域幅を狭めるように働く。これらのシステムは、大規模ＲＡＭディレクトリならびにプロトコルの複雑さとハードウェア・サポートの増大という犠牲を払ってそれを実施している。このタイプの設計はまた、特定のソフトウェア・プロセスによって参照される主記憶ラインの大部分が、そのワークロードを実行中のプロセッサが現在ディスパッチされているノード上と同じ物理ノード上に位置しているとの想定に依存している。リモート・ノードによって「チェック・アウト」され得るラインの数がＮＵＭＡディレクトリのサイズによって制限されるので、ワークロードが多数のリモート・ラインにアクセスしている場合には大きなパフォーマンス上の不利益がある。本発明の１つの目標は、主記憶の内容を移動する必要なしに、かつ大幅なパフォーマンスの低下なしに、多数のプロセッサ間でワークロードの実行を迅速かつ容易に移動できるようにすることである。

【００１３】

「ローカル」であり、大規模マルチプロセッサ中のプロセッサのサブセットに限定される場合、あるいは「グローバル」であり、したがってすべてのプロセッサにブロードキャストされる場合のキャッシュ・コヒーレント動作を定義するために、Hagersten他による米国特許第５，８５２，７１６号は、複数のアドレス区分の使用を記載している。Hagersten他におけるローカル・トランザクションは、記憶要求を発生するプロセッサが属する処理ノードのサブセットと同一のサブセットに割り振られた物理メモリを有するものとして定義される。米国特許第５，８５２，７１６号の第７欄６３行目以降の記載から、この従来技術による発明では、あるプロセスに関連する物理的な記憶域を移動せずに、あるいはアドレッシング・モードを「グローバル」に切り換えずに、いわゆる「ローカル・ドメイン」間でのプロセスの移動ができないことは明らかである。

【特許文献１】米国出願第１０／６０３，２５２号

【特許文献２】米国特許第４，８４３，５４１号

【特許文献３】米国特許第５，８５２，７１６号

【非特許文献１】「IBM Journal of Research and Development」、４１巻、４号および５号、１９９７年

【非特許文献２】「ＳＴｉＮＧ：A CC-NUMA Computer System for the Commercial Mark

etplace」、Proc. 23rd International Symposium of Computer Architecture、１９９６年５月

【発明の開示】

【発明が解決しようとする課題】

【００１４】

大量のＳＲＡＭディレクトリを使用せずに、また主記憶内容の移動を必要とせずに、マルチプロセッサ・コンピュータ・システム中の種々のプロセッサ間でのアドレス要求送信を減らす手法に対するニーズがあると判断した。この必要を満たす解決策を開発するにあたり、大規模マルチプロセッサ・システムにおけるすべての記憶参照トランザクションの待ち時間を短縮するという関連するニーズがあると判断した。

【課題を解決するための手段】

【００１５】

これらの判断されたニーズを実現するために、複数のキャッシュ・コヒーレンシ領域を使用するシステムが、コヒーレンシ領域間でソフトウェア・プロセスを移動させるいくつかの動作の間にキャッシュ・パージを使用せずに動作することを可能にするハードウェア・コヒーレンシ制御がある。本発明は、もはや使用される予定のないコヒーレンシ領域から、最初のアドレス空間と同一のアドレス空間をカバーするために生成されたものではあるが、新しい１組の処理ノードを含むことになる他のコヒーレンシ領域に、ソフトウェア・プロセスを移動させる場合にうまく働く。本発明の好ましい実施形態では、スーパバイザ・プログラムが、ソフトウェア・プロセスを、ある１組の処理ノードを含むあるコヒーレンシ領域から他の１組の処理ノードを含む他のコヒーレンシ領域に、どの処理ノードにおいてもキャッシュのキャッシュ・パージを必要とせずに移動させることができる。移動先のコヒーレンシ領域が、移動元のコヒーレンシ領域より少ないハードウェア処理ノードしか含まない場合は、コヒーレンシ領域のサイズが実際に削減される。

【００１６】

本発明の好ましい実施形態は、複数のノードを有するマルチプロセッサ・コンピュータ・システム中で実施され、この実施形態は、各処理ノードに関連するアクティブ・コヒーレンシ領域情報のテーブルを使用して従来技術のキャッシュ状態遷移をいつ変更するかを判断する。スーパバイザ・プログラムは、各処理ノードに関連するテーブルを初期化する。スーパバイザ・プログラムがその処理ノード上で使用しようとしている各コヒーレンシ領域ごとにテーブルのエントリが作成される。各コヒーレンシ領域には、一意のコヒーレンシ領域ＩＤが割り当てられ、スーパバイザは、そのＩＤをコヒーレンシ領域に含まれる記憶アドレスにアクセスするすべてのソフトウェア・プロセスと関連づけることができる。

【００１７】

本発明を使用する処理ノードは、もはやスーパバイザ・ソフトウェアがそのノード上に現在ディスパッチすることができるどのソフトウェア・プロセスのアドレス空間の一部でもないラインをターゲットとする着信記憶要求を識別することができる。この好ましい実施形態では、この情報により、処理ノードは、コヒーレンシ領域外部からの記憶要求に応答して、もはやそのノード上のどのソフトウェア・プロセスによっても実際に使用されていないキャッシュ・ラインを識別し、キャッシュ・エントリを無効に変更できる。

【００１８】

本発明の利点は非常に多い。本発明の１つの利点は、キャッシュ・コントロール・ハードウェアの必要性をなくすことである。このハードウェアは、本発明を使用しなければ、キャッシュ・コヒーレンシ領域間でソフトウェア・プロセスを移動する際に、キャッシュの選択的パージを実施するために必要になるはずのものである。第２の利点は、本発明により、システム中のキャッシュのすべてが、ソフトウェア・プロセスのコヒーレンシ境界が実際に変更されている間もコヒーレンシ・トランザクションの処理を続けることができることである。第３の利点は、もはや実際に所与のノード上にディスパッチされることがないソフトウェア・プロセスに属するキャッシュ・ラインを識別し、無効にできることで

あり、それによってそれらの再使用が可能になることである。

【００１９】

これらおよびその他の改良は、以下の詳細な説明の中で示される。本発明ならびにその利点と特徴のより良い理解のためには、以下の記述および図面を参照されたい。

【発明を実施するための最良の形態】

【００２０】

この詳細な説明では、図面を参照しながら例によって本発明の好ましい実施形態を、利点および特徴とともに説明する。

【００２１】

ここで図１を参照すると、動的コヒーレンシ境界を有するコンピュータを含む１つのノード（１０）の一実施形態のブロック図が示されている。図１は、ローカル・ノード・コントローラ（１１）に接続された、それぞれキャッシュを備える複数のプロセッサＰ０～Ｐ３を示す。ローカル・コントローラ（１１）は、複数のプロセッサをＤＲＡＭ主記憶要素（１２）に接続する。単一のプロセッサによって開始された記憶トランザクションは、ノード・コントローラ（１１）に送信され、ノード・コントローラ（１１）は、次いでそのトランザクションをそのノードに属する他のプロセッサのいずれかまたはすべてに送信する。ノード・コントローラ（１１）はまたバス（１３）上のトランザクションを、追加のプロセッサ（図示せず）を含むコンピューティング・システムの他の部分に送信することもできる。アクティブ・コヒーレンシ領域テーブル（１４）は、コンピューティング・システム（図示せず）の他の部分からバス（１３）上のノードに入ってくる記憶要求に応答して必要とされる適切なキャッシュ状態遷移を決定するために、ノード・コントローラ（１１）によって使用される。図２は、大規模マルチプロセッサ・システムを作成するために、図１のノード（１０）の複数のインスタンスにセカンド・レベル・コントローラ（１５）をどのように接続できるかを示している。図１には４個の処理要素の使用が示してあるが、任意の数の処理要素が使用できるものと理解されたい。図１には１つのメモリ要素だけが示してあるが、任意の数のメモリ要素が使用できるものと理解されたい。この好ましい実施形態では、図１および図２に示される階層的バス構成を使用しているが、本発明は、他のどのようなタイプの相互接続トポロジを使用するマルチプロセッサ・システムにも適用することができる。

【００２２】

図３は、図１の単一の処理要素を示す。本発明は、マルチプロセッサ・システム中の各プロセッサに対して１つまたは複数のコヒーレンシ・モード・ビット（１６）を使用する。本発明は、マルチプロセッサ・システム中の各プロセッサに対して１つのコヒーレンシ領域ＩＤを使用する。あるプロセッサに関連するコヒーレンシ・モード・ビットおよびコヒーレンシ領域ＩＤは、そのプロセッサによって開始される各記憶トランザクションが図３のバス（１７）を経由してノード・コントローラに送信されるとき、そのトランザクションと共に送信される。本実施形態ではノード・コントローラを使用しているが、他の実施形態では単純な物理バスで置き換えることができることを理解されたい。ノード・コントローラ（１１）およびセカンド・レベル・コントローラ（１５）中のキャッシュ・コヒーレンシ・ハードウェアは、各トランザクションに関連するモード・ビットを使用して、それらがいずれかのプロセッサから受信した任意の記憶トランザクションにどのキャッシュが関与しなければならないかを判断する。この好ましい実施形態では３つのモード・ビットを使用する。この３つのモード・ビットは、ノード・コントローラおよびセカンダリ・コントローラに対する以下の動作モードを識別するために共に使用される。コヒーレンシ・モード設定「０００」は、図１中の破線（１０’）によって示される単一のプロセッサだけのコヒーレンシ領域を定義するために使用される。他の３つのプロセッサのどれも、単一プロセッサ・コヒーレンシ領域でも使用できる。コヒーレンシ・モード設定「００１」は、図１の破線（１８）および（１９）によって示される、２つのプロセッサのコヒーレンシ領域を定義するために使用される。本実施形態では、ノード・コントローラ中で必要なハードウェア制御を単純化するために、ハイパーバイザは、（Ｐ０とＰ１）または

（Ｐ２とＰ３）をカバーする２プロセッサ・コヒーレンシ領域（two-processor coherency regions）を定義することができる。他の実施形態では、ノード１のＰ０とノード２のＰ１など他の組合せが可能になる。コヒーレンシ・モード設定「０１０」は、図１中の破線（２０）によって示される単一のノードのすべてのプロセッサを含むコヒーレンシ領域を定義するために使用される。設定「１０１」は、図２中の破線（２１）および（２２）によって示される２つのノードを含むコヒーレンシ領域を定義する。最後に、設定「１１１」のプロセッサは、生成されたすべての記憶トランザクションが全体システム中のすべてのキャッシュに送信される必要があることを示す。

【００２３】

コヒーレンシ・モード設定は、論理区分の状態の一部と見なされ、したがってその区分中で定義された論理プロセッサの状態の一部と見なされる。本実施形態では、ある単一の論理区分の論理プロセッサはすべて、ある単一の時点で同一のコヒーレンシ・モード設定を有する。分離した１組の記憶アドレスを使用しており、したがってディスパッチのために使用される、異なるコヒーレンシ・モード設定および異なる１組の許容物理プロセッサが提供され得るプロセスを単一の区分内で定義するために、追加のソフトウェアまたはファームウェアが使用できることを理解されたい。論理プロセッサが、物理的な単一のプロセッサ上にディスパッチされると、その物理プロセッサは、一時的にその論理プロセッサのコヒーレンシ・モード設定を引き継ぐ。コヒーレンシ・モード・ビットは、このプロセッサによって生成されたあらゆる記憶トランザクションが、ノード・コントローラ（１１）に送信されるとき、それらのトランザクションとともに送信される。一時に多数の論理区分を定義し使用できるので、同時に多数の異なる、オーバラップしたコヒーレンシ領域が使用される。本発明は、ノード・コントローラ（１１）およびセカンド・レベル・コントローラ（１５）中にハードウェアおよびファームウェアの制御を提供するものであり、それらのコントローラは、各バス・トランザクションに付随するコヒーレンシ・モード・ビットを使用して、システム中のプロセッサを相互接続するバスを介してトランザクションをどのような経路で送るべきかを判断する。

【００２４】

図４は、ノード・コントローラが、そのノード・コントローラが受信した任意の所与のトランザクションを、どのプロセッサが受信しなければならないかを判断するために、どのようにモード・ビットを使用するのかを記述したテーブルを示す。図５は、セカンド・レベル・コントローラが、そのセカンド・レベル・コントローラが受信した任意の所与のトランザクションを、どのノードが受信しなければならないかを判断するために、どのようにモード・ビットを使用するのかを記述したテーブルを示す。図６は、許容物理プロセッサへの論理区分の１つの可能なマッピングを示す。本実施形態では、ノード・コントローラは、セカンド・レベル・ノード・コントローラから受信したすべてのトランザクションを、そのノード・コントローラに接続されたすべてのプロセッサに転送することになる。ノード・コントローラが、セカンド・レベル・ノード・コントローラからきた要求を、そのセカンド・レベル・ノード・コントローラに接続されたプロセッサのサブセットだけに送信することを必要とする、他の多くのコヒーレンシ境界を確立できる可能性があることを理解されたい。

【００２５】

スーパバイザ・ソフトウェアは、それぞれコヒーレンシ領域を有する各プロセスに対し一意のコヒーレンシ領域ＩＤを生成する。スーパバイザ・ソフトウェアはシステム中の各処理ノードについてテーブルを生成する。このテーブルは、各コヒーレンシ領域ＩＤについて、現在その処理ノード上にディスパッチできるエントリを有する。

【００２６】

代替の手法は関連特許出願に記載されている。参照した関連特許出願は、ソフトウェア制御プログラムとハードウェア・モード・ビットの組合せを使用して、複数の処理ノードを使用するコンピューティング・システム中で動的コヒーレンシ境界を定義している。コヒーレンシ境界は、全体システム中の１つのノードからすべてのノードにいたる、任意の

数の処理ノードをカバーするコヒーレンシ領域を生成するように調整することができる。この関連出願はまた、それぞれ専用のアドレス空間で動作する、複数のコヒーレンシ領域をどのように定義することができるかを記載している。このコヒーレンシ領域は、システム動作中の任意の時間に追加のノードを含むように拡張できる。このコヒーレンシ領域はまた、必要な手順に従って、その領域から処理ノードを取り除くことによってサイズを縮小することもできる。これらの手順においては、コヒーレンシ領域から取り除こうとしている処理ノード中のいくつかのキャッシュ・エントリをパージする必要がある。取り除く必要があるキャッシュ・エントリは、サイズを縮小しようとしているコヒーレンシ領域の一部であり、主記憶ラインのキャッシュされたコピーを保持しているキャッシュ・エントリだけである。キャッシュされたラインを「所有する（owns）」コヒーレンシ領域のＩＤに基づいて、選択的パージを実施することのできないキャッシュは、完全にパージしなければならない。この選択的パージでは、従来技術による設計と比べて追加のキャッシュ・コントローラ・ハードウェアを必要とする。本明細書で例示した好ましい実施形態は、異なる２組の処理ノードの間でソフトウェア・プロセスを移動する際に、キャッシュ・パージの必要性をなくす機能をうまく利用する処理システムに適用可能である。

【００２７】

この関連特許出願は、これまでディスパッチしていたソフトウェア・プロセスとは異なるキャッシュ・コヒーレンシ領域を使用するソフトウェア・プロセスをディスパッチするときに、スーパバイザ・ソフトウェアが、プロセッサの「キャッシュ・コヒーレンス・モード」をどのように変更しなければならないかを記載している。本発明では、記憶要求に、その要求を発信したコヒーレンシ領域そのものに関するより詳細な情報のタグを付けることを必要とする。本発明は、このより詳細なタグとして、発信元プロセスのコヒーレンシ領域ＩＤを使用する。この好ましい実施形態では、記憶要求を検査するためにマルチプロセッサ・システム中のどのノードが必要であるかを判断するために、ノード制御論理によって、やはり「キャッシュ・コヒーレンシ・モード」が使用される。本発明のコヒーレンシ領域ＩＤは、正しいスヌープ応答を判断するためにスヌーピング・プロセッサによって使用される。

【００２８】

本発明の好ましい代替実施形態では、コヒーレンシ領域ＩＤを使用して、この好ましい実施形態で記述した機能に加えて、「キャッシュ・コヒーレンシ・モード」の機能が実施できることを理解されたい。好ましい代替実施形態では、「キャッシュ・コヒーレンシ・モード」の簡単なコード化が利用できれば、「キャッシュ・コヒーレンシ・モード」を読み取らなければならないノード・コントローラ論理を、より高速にかつより小型にできると想定している。本発明の好ましい代替実施形態は、ノード・コントローラ中に搭載されるプログラマブルな論理を提供し、スーパバイザ・プログラムは、この論理を使用して、どの物理ノードが特定のコヒーレンシ領域ＩＤに関連するのかをノード・コントローラが判断するのを助ける。

【００２９】

本発明では、関連特許出願と比べて、どの処理ノードが特定の記憶要求を検査しなければならないかを決定するために使用される、論理を変更している。参照した関連特許出願では、モード・ビットで表されるコヒーレンシ領域が、判断を行うために使用されていた。本発明のこの好ましい実施形態では、これを変更して、発信元のコヒーレンシ領域中のすべてのキャッシュをミスしたどの記憶要求も、モード・ビットの設定に関係なく、次に全体システムのすべての処理ノードに転送される。発信元のコヒーレンシ領域中でヒットしたが、正しいキャッシュ状態を有していない要求は、そのコヒーレンシ領域の外部に送信する必要はない。この後者の場合の例は、ある記憶ラインの変更を目的とする記憶要求であるが、記憶トランザクションの間に、そのコヒーレンシ領域内のキャッシュが共用とマークされたラインのコピーを有していることが判明した記憶要求である。本発明のキャッシュ状態遷移は、あるキャッシュ・ラインが、２つの別々のコヒーレンシ領域において共用とマークできないように設定される。スーパバイザ・プログラムは、コヒーレンシ領

域を、ある処理ノードのセットから他の処理ノードのセットに移動するとき、旧ノード上にそのコヒーレンシ領域のキャッシュ・エントリを実際に残したままにする。本発明は、これらの旧キャッシュ・エントリが、新しい処理ノードからの要求によって認識され、しかも旧エントリが無効化されるまでは、新しい処理ノード中で同じ主記憶アドレスについてキャッシュ・エントリが確立されないように働く。

【００３０】

本発明は、関連特許出願中に記載されたシステムに比べて、処理ノード間の追加の通信の使用を提供する。本発明は、大規模オンノード・キャッシュと共に使用されることを想定しており、それにより、このオンノード・キャッシュを完全にミスする記憶要求の数が最小限に抑えられることになる。本発明は、動的コヒーレンシ境界を使用しない従来技術の設計と比べて、処理ノード間の通信を全体としてさらに削減できるはずである。

【００３１】

本発明は、コヒーレンシ領域間でのソフトウェア・プロセスの容易な移動を提供するものである。本発明は、要求された記憶アドレスのキャッシュ・エントリが要求を開始したプロセッサを含む処理ノード中のいずれかのキャッシュ上に存在することが判明したときには、その処理ノード中のキャッシュ・コントローラが、スーパバイザ・ソフトウェアを使用して、現在のコヒーレンシ領域モードによって指定される、そのプロセッサの現在のコヒーレンシ領域の外部に、要求された記憶アドレスのコピーが存在しないようにすることを可能にする。

【００３２】

新しいコヒーレンシ領域ＩＤは、記憶要求にどのように応答するかを決定するためにスヌーピング・キャッシュによって使用される。着信要求に付いているコヒーレンシ領域ＩＤが、処理ノードのアクティブ・コヒーレンシ領域テーブル中のどのＩＤにも一致しない場合、この処理ノード中のキャッシュは、通常のＭＥＳＩプロトコル応答で応答し、次いでこのキャッシュを無効に設定する。無効への変更により、汚いデータ（dirty data）を主記憶に書き戻すことが必要になる場合は、この動作も同時に開始される。図７は、着信記憶トランザクションの処理中に処理ノードによって使用される、アクティブ・コヒーレンシ領域ＩＤテーブルの例を示す。このテーブルは、この処理ノード上でアクティブな一意のコヒーレンシ領域あたり１つずつ、予想される数のエントリを収容できるように任意に大きくすることができる。この好ましい実施形態ではハードウェア・テーブルを使用しているが、本発明に必要な機能を実施するためにソフトウェア、ファームウェア、およびハードウェアの任意の組合せが使用できることは明らかであろう。

【図面の簡単な説明】

【００３３】

【図１】動的コヒーレンシ境界を有するコンピュータを示すブロック図である。ノード・コントローラが、アクティブ・コヒーレンシ領域テーブルのハードウェア実装を含む。

【図２】大規模マルチプロセッサ・システムを作成するために、図１のコンピュータを含むノードの複数のインスタンスを、どのようにセカンド・レベル・コントローラに接続できるかを示す図である。

【図３】図１の単一の処理要素を示す図である。

【図４】ノード・コントローラによって受信された任意の所与のトランザクションを、どのプロセッサが受信しなければならないかを判断するために、ノード・コントローラがどのようにモード・ビットを使用するのかを記述するテーブルを示す図である。

【図５】セカンド・レベル・コントローラによって受信された任意の所与のトランザクションを、どのノードが受信しなければならないかを判断するために、セカンド・レベル・コントローラがどのようにモード・ビットを使用するのかを記述するテーブルを示す図である。

【図６】許容物理プロセッサへの論理区分の１つの可能なマッピングを示す図である。

【図７】アクティブ・コヒーレンシ領域テーブルのハードウェア実装の追加の詳細を示す図である。該テーブルは、ソフトウェア・プロセス・ディスパッチのために現在その処理

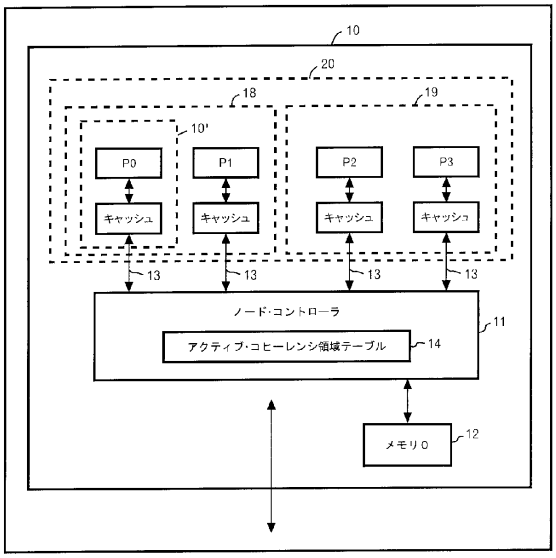
ノードを使用できる、スーパバイザ・ソフトウェアによって制御されるコヒーレンシ領域のリストである。

【符号の説明】

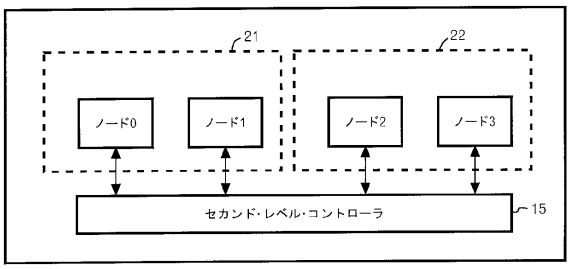
【００３４】

１０　ノード
１１　ローカル・ノード・コントローラ、ノード・コントローラ
１２　ＤＲＡＭ主記憶要素
１３　バス
１４　アクティブ・コヒーレンシ領域テーブル
１５　セカンド・レベル・コントローラ
１６　コヒーレンシ・モード・ビット
１７　バス

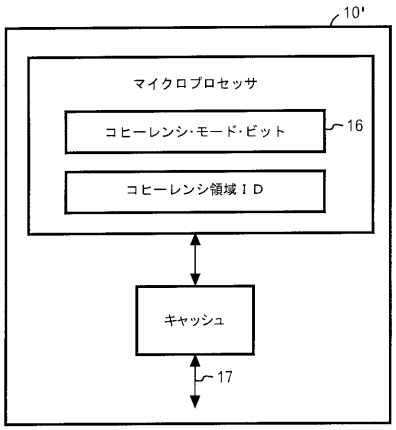
【図１】



【図２】



【図３】



【図４】

発信元プロセッサ	モード・ビット設定	ノード・コントローラ動作：下記のプロセッサまたはセカンド・レベル・コントローラあるいはその両方にトランザクションを送信する
P0	"000"	トランザクションを転送しない
P0	"001"	プロセッサP1
P0	"010"	発信元ノード中のすべてのプロセッサ
P0	"1xx"	発信元ノード中のすべてのプロセッサおよびセカンド・レベル・コントローラ
P1	"000"	トランザクションを転送しない
P1	"001"	プロセッサP0
P1	"010"	発信元ノード中のすべてのプロセッサ
P1	"1xx"	発信元ノード中のすべてのプロセッサおよびセカンド・レベル・コントローラ
P2	"000"	トランザクションを転送しない
P2	"001"	プロセッサP3
P2	"010"	発信元ノード中のすべてのプロセッサ
P2	"1xx"	発信元ノード中のすべてのプロセッサおよびセカンド・レベル・コントローラ
P3	"000"	トランザクションを転送しない
P3	"001"	プロセッサP2
P3	"010"	発信元ノード中のすべてのプロセッサ
P3	"1xx"	発信元ノード中のすべてのプロセッサおよびセカンド・レベル・コントローラ

「1xx」は、最初のビット位置の値が１で、第２および第３のビット位置の値が任意の値の組合せであることを示す

【図５】

発信元ノード	モード・ビット設定	セカンド・レベル・コントローラ動作：下記ノードにトランザクションを送信する
0	"0xx"	該当なし
0	"101"	ノード１
0	"111"	すべてのノード
1	"0xx"	該当なし
1	"101"	ノード０
1	"111"	すべてのノード
2	"0xx"	該当なし
2	"101"	ノード３
2	"111"	すべてのノード
3	"0xx"	該当なし
3	"101"	ノード２
3	"111"	すべてのノード

【図６】

区分ＩＤ	コヒーレンシ・モード	ディスパッチ可能なプロセッサ
0	"000"	(0:P2)
1	"000"	(1:P1)
2	"001"	(2:P0)(2:P1)
3	"010"	(3:P0)(3:P1)(3:P2)(3:P3)
4	"010"	(0:P0)(0:P1)(0:P2)(0:P3)
5	"101"	(2:P0)(2:P1)(2:P2)(2:P3) (3:P0)(3:P1)(3:P2)(3:P3)
6	"111"	４つのノードのいずれか上のいずれかのプロセッサ

【図７】

アクティブ・コヒーレンシ領域テーブル

エントリ＃	コヒーレンシ領域ＩＤ
0	4
1	7
2	23
N	6

フロントページの続き

(72)発明者　トーマス・ジェイ・ヘラー・ジュニア
アメリカ合衆国１２５７２　ニューヨーク州ラインベック　ノルウッド・ロード　２１５

審査官　清木　泰

(56)参考文献　特開昭６０－０７９４４６（ＪＰ，Ａ）
特開平０６－０９６０３９（ＪＰ，Ａ）
特開平０２－２９７６５６（ＪＰ，Ａ）
特表２００１－５１５６３３（ＪＰ，Ａ）
特開２０００－２３５５５８（ＪＰ，Ａ）
特開２０００－１７２６５５（ＪＰ，Ａ）
特開平１１－２５９３１８（ＪＰ，Ａ）
特開平１０－２４０７０７（ＪＰ，Ａ）
特開平０８－０３０５６７（ＪＰ，Ａ）
特開平０８－０３０５６２（ＪＰ，Ａ）
特開平０７－１６０５８１（ＪＰ，Ａ）
特開平０６－２７４４６１（ＪＰ，Ａ）
特開平０６－２６６６２０（ＪＰ，Ａ）
特開平０１－１０９４６４（ＪＰ，Ａ）
特開昭６２－１１５５６７（ＪＰ，Ａ）
特開２００２－１４０２３４（ＪＰ，Ａ）
特開２００１－１８４３２１（ＪＰ，Ａ）
特開平０９－２３１１８７（ＪＰ，Ａ）
特開平０２－１２２３６５（ＪＰ，Ａ）

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