(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号

特許第4507563号
(P4507563)

(45)発行日　平成22年7月21日(2010.7.21)　(24)登録日　平成22年5月14日(2010.5.14)

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

請求項の数 8　(全 18 頁)

(21)出願番号　特願2003-379294 (P2003-379294)
(22)出願日　平成15年11月10日 (2003.11.10)
(65)公開番号　特開2005-141606 (P2005-141606A)
(43)公開日　平成17年6月2日 (2005.6.2)
審査請求日　平成18年10月6日 (2006.10.6)

(73)特許権者　000005108
株式会社日立製作所
東京都千代田区丸の内一丁目６番６号
(74)代理人　100100310
弁理士　井上　学
(72)発明者　助川　直伸
東京都国分寺市東恋ケ窪一丁目２８０番地
株式会社日立製作所中央研究所内

審査官　▲高▼橋　正▲徳▼

最終頁に続く

(54)【発明の名称】マルチプロセッサシステム

(57)【特許請求の範囲】
【請求項１】
それぞれキャッシュメモリを備えた複数のプロセッサと、該複数のプロセッサに共有の主記憶とを有するマルチプロセッサシステムであって、
キャッシュコヒーレンス要求をバス経由で各プロセッサにブロードキャストすることでキャッシュコヒーレンス制御を実現する手段と、
前記バスでのブロードキャストの範囲をシステム全体ではなくシステムの一部になるように該バスを分割設定する手段とを有し、
前記主記憶に対応して該主記憶の各データブロック毎にそのデータブロックをキャッシュメモリに登録したプロセッサのＩＤを記録するディレクトリを有し、
前記ディレクトリに記録されたＩＤの情報を用いて各プロセッサの間でキャッシュコヒーレンス制御を行う手段を有し、
前記バスで結合されるプロセッサ間は前記バスを介したキャッシュコヒーレンス要求の伝達によるキャッシュコヒーレンス制御を行い、
該バスの分割設定により互いに分断されたプロセッサ間では前記ディレクトリを用いたキャッシュコヒーレンス制御を行うことを特徴とするマルチプロセッサシステム。
【請求項２】
前記ディレクトリには前記バスの分割設定により分断されたプロセッサのＩＤ情報とともに、バスで結合されているプロセッサのＩＤ情報も記録されることを特徴とする請求項１に記載のマルチプロセッサシステム。

【請求項３】
前記バスで結合されているプロセッサのＩＤ情報について、該ＩＤ情報に従ったキャッシュコヒーレンス要求の生成は行わないにもかかわらず、該ＩＤ情報については該バスを介したキャッシュコヒーレンス要求の伝達によるキャッシュコヒーレンス制御が実施されたと見なし、前記ディレクトリに記録された該ＩＤ情報を変更することを特徴とする請求項２に記載のマルチプロセッサシステム。
【請求項４】
前記バスの分割設定が動作途中で変更になったことにより該バスで元々結合されていたプロセッサ同士の結合が分断された場合に、該バスを介したキャッシュコヒーレンス要求の伝達によるキャッシュコヒーレンス制御から前記ディレクトリに記録されていたプロセッサＩＤ情報を使用したキャッシュコヒーレンス制御に切り替えることを特徴とする請求項３に記載のマルチプロセッサシステム。
【請求項５】
それぞれキャッシュメモリを備えた複数のプロセッサと、該複数のプロセッサに共有であってかつ該複数のプロセッサの各々もしくはプロセッサ群の各々に対応してそれぞれ設けられた複数の部分主記憶で構成される主記憶とを有するマルチプロセッサシステムであって、
キャッシュコヒーレンス要求をバス経由で各プロセッサにブロードキャストすることでキャッシュコヒーレンス制御を実現する手段と、
前記バスでのブロードキャストの範囲をシステム全体ではなくシステムの一部になるように該バスを分割設定する手段とを有し、
前記部分主記憶の各々に対応して設けれら、各部分主記憶のデータブロック毎にそのデータブロックをキャッシュメモリに登録したプロセッサのＩＤを記録するディレクトリを有し、
前記ディレクトリに記録されたＩＤの情報を用いて各プロセッサの間でキャッシュコヒーレンス制御を行う手段を有し、
分割設定された前記バスで相互に結合されたプロセッサに対応する範囲の部分主記憶に含まれるデータに対するキャッシュコヒーレンス制御で、かつ該相互に結合されたプロセッサ間のキャッシュコヒーレンス制御の場合に、前記バスを用いたキャッシュコヒーレンス制御を行い、
前記バスの分割設定で分断されたプロセッサ間、もしくは前記バスで結合されたプロセッサ間であってもキャッシュコヒーレンス制御対象のデータが前記結合されたプロセッサに対応する範囲の部分主記憶ではなく、該範囲から外れる部分主記憶に含まれる場合には該ディレクトリを用いたキャッシュコヒーレンス制御を行うことを特徴とするマルチプロセッサシステム。
【請求項６】
前記ディレクトリには前記バスの分割設定により分断されたプロセッサのＩＤ情報とともに、該バスで結合されているプロセッサのＩＤ情報も記録することを特徴とする請求項５に記載のマルチプロセッサシステム。
【請求項７】
前記バスを介したキャッシュコヒーレンス要求の伝達によるキャッシュコヒーレンス制御を行う場合についても、前記ＩＤ情報については該バスによるキャッシュコヒーレンス制御が実施されたと見なし、前記ディレクトリに記録された該ＩＤ情報を変更することを特徴とする請求項６に記載のマルチプロセッサシステム。
【請求項８】
前記バスの分割設定が動作途中で変更になったことにより該バスで元々結合されていたプロセッサ同士のバスが分割された場合に、該バスを介したキャッシュコヒーレンス要求の伝達によるキャッシュコヒーレンス制御から前記ディレクトリに記録されていたプロセッサＩＤ情報を使用したキャッシュコヒーレンス制御に切り替えることを特徴とする請求項６に記載のマルチプロセッサシステム。

【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、複数のＣＰＵを持ちＣＰＵ間で主記憶を共有する共有メモリ型並列計算機におけるＣＰＵ間のキャッシュメモリの一致制御、すなわちＣＰＵ間キャッシュコヒーレンス制御方式に属する。
【背景技術】
【０００２】
デバイス技術の進歩により、ＣＰＵの動作周波数の向上は急峻であり、これに対しＣＰＵから主記憶をアクセスする際のメモリアクセスレイテンシは、ＣＰＵから主記憶までの物理的な距離や主記憶素子の特性に制約されることから、絶対時間での改善は緩やかである。つまりＣＰＵの単位動作時間（＝１秒÷動作周波数）を単位として見た場合、アクセスレイテンシ時間はむしろ延びる方向にあり、アクセスレイテンシが性能向上のボトルネックとなる傾向にある。
【０００３】
この相対的なメモリアクセスレイテンシの悪化による性能低下を救う技術がキャッシュメモリである。キャッシュメモリは、高速小容量のバッファをＣＰＵに近いところに用意し、使用頻度の高いデータのコピーを登録することで、実効的なメモリアクセスレイテンシを削減する技術である。
【０００４】
現代の計算機では、前述のキャッシュメモリを実装するＣＰＵを複数個搭載し、ＣＰＵ全部もしくは一部の間で主記憶を共有する共有メモリ型並列計算機の構成、もしくは共有メモリ型並列計算機のクラスタ構成を採用することが多い。ＣＰＵを複数搭載するのは、（１）性能向上、（２）可用性の向上（１つのＣＰＵに障害が発生した場合にもシステムダウンを防ぐ）、を狙うことによる。計算機サービスに使用される「サーバ」と呼ばれる計算機では、最低でも２ＣＰＵ以上の共有メモリ型並列計算機の構成が必須とされる。
【０００５】
このように、キャッシュメモリを持った複数のＣＰＵが主記憶を共有する場合、キャッシュメモリの一致制御、いわゆるキャッシュコヒーレンス制御が問題になる。これはあるＣＰＵ（Ａ）がキャッシュメモリに登録しているデータを別なＣＰＵ（Ｂ）がストア命令で更新した場合、その更新結果はＣＰＵ（Ａ）のキャッシュメモリに反映される（キャッシュメモリ上のデータが更新される、もしくは無効化される）必要がある。
【０００６】
このようなキャッシュメモリ一致制御は、一般的にはバスを通じた制御で行われる。これは、あるプロセッサによるデータの更新はバスを通じて全ＣＰＵにブロードキャストされる仕組みと、各ＣＰＵがバスを常時チェックしブロードキャストされた更新情報をキャッシュメモリに登録されているデータに反映させる（スヌープ動作）仕組みとにより実現される。
【０００７】
上記バスによるキャッシュコヒーレンス制御は、バス上でキャッシュコヒーレンスリクエストが輻輳を起こし得ることから、ＣＰＵ数の多い大規模な共有メモリ型マルチプロセッサ構成を全てバスで実現すると各ＣＰＵの性能が低下する問題がある。この問題に対し、メモリ上のデータそれぞれに、どのプロセッサがキャッシュメモリにデータを登録しているかを覚えるディレクトリを設け、ディレクトリに登録された情報に基づき必要なプロセッサにのみキャッシュコヒーレンスリクエストを伝達することにより、バスに比べて輻輳の発生頻度を低下させることが出来る。本方式がいわゆるＮＵＭＡ(Non-Uniform Memory Architecture)型マルチプロセッサである。ＮＵＭＡ型マルチプロセッサについては、情報処理学会誌情報処理Vol.34 No.1(１９９３年１月)、９６～１０５頁「マルチプロセッサの記憶システム（１）」を参考にされたい。
【０００８】

ＮＵＭＡ型はＣＰＵ間バスのように全てのＣＰＵからのリクエストが集中する箇所がないことから、ＣＰＵの増加に従い性能をスケーラブルに高めることが出来るメリットがある。他方、バス型のマルチプロセッサであればリクエストをバスに送出した直後にコヒーレンス制御が低レイテンシで実行されるのに対し、ＮＵＭＡではリクエストが発生するとこれに対して他ＣＰＵに対するコヒーレンス制御を行うか行わないかの判定を行う回路を経て、更にそこからコヒーレンス制御要求が対象ＣＰＵに伝達されるという手順を踏むことから、一般にコヒーレンス制御の遅延時間が大きく、小規模システムではバス型マルチプロセッサに性能で劣るというデメリットがある。

【０００９】

特開平１０－２４０７０７号公報では、バス型のマルチプロセッサを単位としてＮＵＭＡでこれらを結合したシステムを構築する技術を開示している。更に特開平１０－２４０７０７では、ＮＵＭＡシステムをパーティション分割した場合にＮＵＭＡ制御を軽減できる技術を開示する。具体的には、主記憶をパーティション内でのみ使用する領域とパーティション間でも使用する領域とに分け、パーティション内で使用する領域へのアクセスに対してはパーティション内のバス型マルチプロセッサ群にのみブロードキャスト（マルチキャスト）することで、コヒーレンス制御オーバヘッドを低減する。例えば、このパーティションをバスでＣＰＵが結合される範囲一つのみに設定した場合には、前記パーティション内のみで使用する領域に対するオペレーションはバスで高速に処理され、パーティション間でも使用する領域に対するオペレーションのみがＮＵＭＡで制御されることになり、スケーラビリティと高速性とを両立させることが出来る。

【００１０】

【非特許文献１】情報処理学会誌情報処理Vol.34 No.1(１９９３年１月)、９６～１０５頁「マルチプロセッサの記憶システム（１）」

【００１１】

【特許文献１】特開平特開平１０－２４０７０７号公報

【発明の開示】

【発明が解決しようとする課題】

【００１２】

実際のシステムでは、特開平１０－２４０７０７で定義されるパーティションといったものは、例えばプログラム毎に変更する、もしくはプログラム実行途中で演算処理をマイグレーションさせるために変更するなど、動的に設定する要求が発生しうる。しかし、特開平１０－２４０７０７はパーティションの動的変更に関する技術は開示していない。

【００１３】

本発明は、パーティションの動的は設定に対応した、バス型の高速性とＮＵＭＡのスケーラビリティとを両立するマルチプロセッサシステムの実現を目的とする。

【課題を解決するための手段】

【００１４】

各ＣＰＵを分割可能なバスによるバス接続とＮＵＭＡを実現するネットワーク接続との両方で接続する。バスが分割されずＣＰＵ間を結合される範囲は、バスによるコヒーレンス制御が実行される。バス接続が如何なる形態であっても、ＮＵＭＡディレクトリには全ＣＰＵのアクセスを登録する。ＮＵＭＡディレクトリの制御回路にバスの分割状態を記憶するグループ設定レジスタを設け、バスで接続されているＣＰＵ間のコヒーレンス制御については、別途バスにより実現されていることから、ディレクトリによるコヒーレンス制御リクエストを省略する制御を行う。バスが分割されているＣＰＵ間のコヒーレンス制御については、ディレクトリによるコヒーレンス制御をネットワークを通じて実行する。

【００１５】

ジョブの実行形態に合わせてバスの分割形態を変更する場合には、同時にグループ設定レジスタも変更する。これにより、新たにバスが分割されたＣＰＵ間ではそれまでバスで実現されていたコヒーレンス制御をディレクトリによるコヒーレンス制御に切り替え、新たにバスが結合されたＣＰＵ間ではそれまでディレクトリにより実現されていたコヒーレ

ンス制御をバスによるコヒーレンス制御に切り替える。
【発明の効果】
【００１６】
本発明により、パーティションの動的は設定に対応しつつ、バス型の高速性とＮＵＭＡのスケーラビリティとを両立するマルチプロセッサシステムを実現できる。
【実施例１】
【００１７】
以下、最初に本発明のマルチプロセッサシステムの動作の概要、次に本発明のマルチプロセッサシステムの動作の詳細として、
（１）バスを通したコヒーレンス制御
（２）ＮＵＭＡネットワークを通したコヒーレンス制御
（３）バス結合の変更時の処理
を順に説明する。
[動作の概要]
本節では、図１を用いて、本発明のマルチプロセッサシステムの動作の概要を示す。なお、説明における初期設定として、バス分割結合回路５００、５１０、５２０は、５００と５２０とが結合状態、５１０は分割状態だとする。つまり、ＣＰＵ１００とＣＰＵ２００とはバスで結合されており、またＣＰＵ３００とＣＰＵ４００ともバスで結合されているが、ＣＰＵ１００＆２００とＣＰＵ３００＆４００の間は分離されているとする。
【００１８】
この状態では、例えばＣＰＵ１００発のキャッシュコヒーレンス要求はＣＰＵ２００には部分バス１４０→バス分割結合回路５００→部分バス２４０を通じて伝達することが出来る。他方、バス分割結合回路５１０は分割状態であるから、ＣＰＵ１００発のキャッシュコヒーレンス要求はバスを通じてＣＰＵ３００に伝達されることはない。
【００１９】
バスによるコヒーレンス制御とは別個に、ディレクトリによるコヒーレンス制御を実施する。ＣＰＵ１００からＣＰＵ３００へのキャッシュコヒーレンス要求も、ＮＵＭＡネットワーク１０００を通じてであれば実施することができる。
【００２０】
以上のような仕組みを生かして、次のコヒーレンス制御を実施する。
（Ａ）バスで結合されたＣＰＵ同士で、かつキャッシュコヒーレンス制御対象のデータのアドレスがバスで結合された範囲の部分主記憶に対するものである場合には、バスでのみキャッシュコヒーレンス制御を行う。
（Ｂ）（Ａ）以外のケースについては、ＮＵＭＡネットワークを使ってキャッシュコヒーレンス制御を行う。
【００２１】
以下実例を示す。例えば、ＣＰＵ１００発のキャッシュコヒーレンス制御で、対象アドレスが部分主記憶１８０に対するものであれば、まず部分バス１４０→バス分割結合回路５００→部分バス２４０を通じてＣＰＵ２００に対してキャッシュコヒーレンス要求が伝達される。
【００２２】
この要求に対して、部分主記憶１８０に対するアクセス情報をディレクトリ１６０に記憶するディレクトリ制御回路１５０は、ディレクトリ１６０に登録された情報から、もしキャッシュコヒーレンス制御がバスで結合されるＣＰＵ２００に対してのみ必要と判定できれば、バスを通じてコヒーレンス制御が全て実行されていると判定し、ＮＵＭＡネットワーク１０００を通したキャッシュコヒーレンス制御を実行しない。ディレクトリ１６０に登録された情報から、もしキャッシュコヒーレンス制御がバスでは結合されないＣＰＵ３００やＣＰＵ４００に対しても必要と判定されれば、ＮＵＭＡネットワークを通じてキャッシュコヒーレンス制御を実行する。なお、ディレクトリ制御回路１５０は、バスの分割結合状況をグループ設定レジスタ１７０に記憶している。また、ディレクトリ１６０に

は、バスで結合されていないＣＰＵ３００、ＣＰＵ４００のアクセス情報だけでなく、バスで結合されるＣＰＵ２００の情報も記憶している。部分主記憶２８０に対するアクセスに関しても、ＣＰＵ１００から見て部分主記憶２８０へはバスを通じてアクセスできることから、基本的には部分主記憶１８０（及びディレクトリ制御回路１５０）についてと同様の動作が部分主記憶２８０（及びディレクトリ制御回路２５０）について実行される。

【００２３】

上記の動作に対して、例えば、ＣＰＵ１００発のキャッシュコヒーレンス制御で、対象アドレスが部分主記憶３８０に対するものであれば、ＮＵＭＡ制御回路１２０は当該要求がバスで結合されていない部分主記憶３８０へのリクエストと判定し、コヒーレンス制御要求をＮＵＭＡネットワーク１０００を通してディレクトリ制御回路３５０へ伝播する。ディレクトリ制御回路３５０はディレクトリ３６０に登録された情報に基づき、通常のＮＵＭＡによるキャッシュコヒーレンス制御を実施する。

【００２４】

キャッシュコヒーレンス制御動作の詳細は、別途動作詳細にて説明する。

【００２５】

以上であるバス分割設定の時の定常的なキャッシュコヒーレンス制御方法の概要を説明した。次に、バス分割設定を変更する場合の動作概要を説明する。ここでは、バス分割結合回路５００が結合状態から分割状態に遷移した場合を説明する。

【００２６】

先に述べたとおり、ディレクトリ１６０には、元よりＣＰＵ２００のアクセス情報も登録されていることから、ディレクトリ制御回路１５０はグループ設定レジスタ１７０さえ変更すれば、ＣＰＵ２００に対してもＮＵＭＡネットワークを通じたキャッシュコヒーレンス制御を実行できる。故に、バス分割設定を変更する場合、バス分割結合回路５００の設定を変更するだけでなく、ＮＵＭＡ制御回路１２０、２２０、３２０、４２０中のバス設定レジスタ１３０、２３０、３３０、４３０、及びディレクトリ制御回路１５０、２５０、３５０、４５０中のグループ設定レジスタ１７０、２７０、３７０、４７０を同時に変更する。この変更後には、今までバスを通じて行われていた、ＣＰＵ１００発の部分主記憶１８０に関するコヒーレンス制御でＣＰＵ２００にのみ対する要求は、ディレクトリ制御回路１５０からＮＵＭＡネットワーク１０００を通じてＣＰＵ２００に対して実施されるようになり、また、主記憶２８０に関するコヒーレンス制御は、バスを使用せずＮＵＭＡ制御回路１２０よりＮＵＭＡネットワーク１０００を通じてディレクトリ２６０ベースで実施されるようになる。

【００２７】

以上、バス設定が結合から分割に変更される場合の動作変更を説明したが、逆に分割から結合に変更される場合にも、基本的には上記と同様にバス分割結合回路５００、５１０、５２０と同時にＮＵＭＡ制御回路１２０、２２０、３２０、４２０中のバス設定レジスタ１３０、２３０、３３０、４３０、及びディレクトリ制御回路１５０、２５０、３５０、４５０中のグループ設定レジスタ１７０、２７０、３７０、４７０を変更することで、従来ＮＵＭＡネットワーク１０００を通じて実施されていたキャッシュコヒーレンス制御動作が結合した範囲ではバスによる動作に変更される。変更手順詳細は別途動作詳細にて説明する。

【００２８】

以上で、本発明システムの動作概要を説明した。次に、各構成要素の中身を含め、本発明システムの動作詳細を説明する。

[動作の詳細]

動作の詳細に入る前に、本発明で前提としているキャッシュコヒーレンスプロトコルを説明する。本発明では、各ＣＰＵのキャッシュコヒーレンス制御はＭＥＳＩプロトコルに従うとする。ＭＥＳＩプロトコルでは、Ｃｌｅａｎなデータ（＝キャッシュメモリと主記憶との中身が一致しているデータ）は１ＣＰＵ（Ｅステータス：Ｅｘｃｌｕｓｉｖｅ）もしくは複数ＣＰＵ（Ｓステータス：Ｓｈａｒｅｄ）が所有できるが、Ｍｏｄｉｆｉｅｄデ

ータ（＝キャッシュメモリに更新された最新値があり主記憶には更新前の古い値が入っていることから中身が不一致を起こしているデータ）を所有できるのは１ＣＰＵのみ（Ｍステータス：Ｍｏｄｉｆｉｅｄ）というルールがある。故に、Ｓステータスの状態であるＣＰＵがデータ更新を行うと、更新するＣＰＵ以外の各ＣＰＵへキャッシュ無効化要求が発生し、更新するＣＰＵのみが更新後にＭステータスでデータをキャッシュメモリに持つようになる。また、Ｍステータスにあるデータを他のＣＰＵがアクセスした場合には、当該データはキャッシュメモリから主記憶に書き戻され、複数のＣＰＵがＣｌｅａｎなデータをキャッシュメモリにＳステータスで所有するようになる。

【００２９】

ＭＥＳＩプロトコルで最低限必要とされるトランザクション（＝ＣＰＵから発生するデータ操作の要求）は、

・フェッチ要求（キャッシュへの新規登録）

・キャッシュ無効化要求（キャッシュデータの更新）

・キャストアウト要求（キャッシュからメモリへの書戻し）

の３つであり、本発明でも上記３つの要求がＣＰＵから発生するとしている。

【００３０】

なお、図７中のＩステータス２０００はＩｎｖａｌｉｄステータス（キャッシュ中のデータが無効である状態）、Ｅステータス２０１０は上記で説明したＥｘｃｌｕｓｉｖｅステータス、Ｓステータス２０３０は上記で説明したＳｈａｒｅｄステータス、Ｍステータス２０２０は上記で説明したＭｏｄｉｆｉｅｄステータスである。load-miss(exclusive)とあるのは、フェッチ要求を出した結果キャッシュメモリにデータを登録したＣＰＵが他に無いことが分かった場合、load-miss(not exclusive)とあるのは、フェッチ要求を出した結果キャッシュメモリにデータを登録したＣＰＵが他にもあった場合、store-missはストア命令でキャッシュミスを起こしたために、一旦データをキャッシュメモリへのフェッチ要求を出し、フェッチ実行後にstoreを実行するためにキャッシュ無効化要求を出した場合を表す。

【００３１】

load-hit、store-hitはload命令、store命令を実行した時に結果としてキャッシュメモリがヒットした場合で、この場合もＳステータスでstore-hitを起こした場合には、他ＣＰＵのキャッシュメモリを無効化するために、キャッシュ無効化要求を通達する必要がある。

【００３２】

snoop-loadとあるのは他ＣＰＵからフェッチ要求を受けた場合、snoop-storeとあるのは他ＣＰＵからキャッシュ無効化要求を受けた場合である。Ｍステータスでsnoop-loadを受けると、主記憶にデータを書き戻すためのキャストアウトを実行する必要がある。

【００３３】

上記キャッシュコヒーレンスプロトコルをベースとして、以下、最初の２節でバス分離結合設定を変更しない状態での定常的なキャッシュコヒーレンス動作をバス経由とネットワーク経由とに分けて説明し、最後の節でバス分離結合設定を変更する場合の動作について説明する。

【００３４】

（１）バスを通したコヒーレンス制御

バスを通したコヒーレンス制御は、動作概要で説明したとおり、バスで結合されたＣＰＵ同士で、かつキャッシュコヒーレンス制御対象のデータのアドレスがバスで結合された範囲の部分主記憶に対するものである場合に行う。以下、バスを通したフェッチ要求の処理、キャッシュ無効化要求の処理、キャストアウト要求の処理を順次説明する。

【００３５】

（１）－１：フェッチ要求

本節では、バス分離結合回路５００、５１０、５２０がそれぞれ結合、分離、結合という設定だったということを前提に、ＣＰＵ１００からの部分主記憶１８０及び部分主記憶

２８０へのフェッチ要求がＣＰＵ２００との間でどう制御されるかを説明する。
【００３６】
ＣＰＵ１００はload命令やstore命令でキャッシュメモリ１１０がミスを起こすと、信号線Ｌ１００を通じてフェッチ要求パケットを出力する。最初に、この時のアドレスは、部分主記憶１８０へのリクエストであったとする。
【００３７】
アドレスのマップについては図８に本実施例のアドレスマップを示す。各部分主記憶１８０、２８０、３８０、４８０は、その半分がローカルメモリとなっており、ＣＰＵ１００専用のローカル領域は部分主記憶１８０に、ＣＰＵ２００専用のローカル領域は部分主記憶２８０に、ＣＰＵ３００専用のローカル領域は部分主記憶３８０に、ＣＰＵ４００専用のローカル領域は部分主記憶４８０に確保される。なお、図８は各部分主記憶の容量を５１２メガバイトであるという前提で記載してある。図８の共有メモリ（Ａ）４１００と示されるのが部分主記憶１８０のローカルメモリを除いた半分、共有メモリ（Ｂ）４２００と示されるのが部分主記憶２８０のローカルメモリを除いた半分、共有メモリ（Ｃ）４３００と示されるのが部分主記憶３８０のローカルメモリを除いた半分、共有メモリ（Ｄ）４４００と示されるのが部分主記憶４８０のローカルメモリを除いた半分である。故に、ＣＰＵ１００から見て、ローカルメモリ４０００へのアクセスは部分主記憶１８０へのアクセス、共有メモリ（Ａ）４１００へのアクセスも部分」主記憶１８０へのアクセス、共有メモリ（Ｂ）４２００へのアクセスが部分主記憶２８０へのアクセスとなる。
【００３８】
フェッチ要求パケットについて図９にそのフォーマットを示す。図９のうち先頭はコマンド５０００であり、中身００００はフェッチ要求であることを示す。更に要求元プロセッサＩＤ５０１０に例えばＣＰＵ１００によるキャッシュミスであれば００００、ＣＰＵ２００によるキャッシュミスであれば０００１、ＣＰＵ３００によるキャッシュミスであれば００１０、ＣＰＵ４００によるキャッシュミスであれば００１１の値が入る。アドレス５０２０はフェッチをするアドレスである。
【００３９】
以上、説明に戻るが、アドレスに対してＣＰＵ１００から信号線Ｌ１００に対してフェッチ要求パケットを出力したとする。ＮＵＭＡ制御回路は中身を図２に示すが、フェッチ要求はリクエストルータ６００に到達したところで、バス設定レジスタ１３０の設定値とのチェックにより行き先の決定が行われる。バス設定レジスタ１３０の中身（３２ビットとした）を図３に示すが、バス設定レジスタでは、下位４ビットを用いて（図中ビット１３２、１３４、１３６、１３８）、先程のメモリマップ上に見える共有メモリ（Ａ）４１００～共有メモリ（Ｄ）４４００が、当該ＮＵＭＡ制御回路から見て、それぞれバスで接続されているのか、バスは分割設定でバスではアクセス不可能なのかを表す。共有メモリ（Ａ）４１００に対応するのが最下位ビット１３８、共有メモリ（Ｂ）４２００に対応するのがビット１３６、共有メモリ（Ｃ）４３００に対応するのがビット１３４、共有メモリ（Ｄ）４４００に対応するのがビット１３２である。中身は１であればバスで結合されているし０であればバスは分割されていることを表す。以上の設定値がバス設定レジスタ１３０からルータへと信号線Ｌ６７０を通じて伝播していることから、ルータは今回のパケットは部分主記憶１８０へのリクエストであると判定し、そのリクエストを信号線Ｌ６１０、Ｌ１１０を通じて部分バス１４０に出力する。
【００４０】
先述の通り、バス分離結合回路５００は結合状態となっているので、同フェッチ要求はバス分離結合回路５００を経由して部分バス２４０にも通達される。これにより、まずＮＵＭＡ制御回路２２０は同リクエストを信号線Ｌ２１０を通じてスヌープすると、図２中（これはＮＵＭＡ制御回路１２０の中身を示しているが、内部構成はＮＵＭＡ制御回路２２０と同一）のセレクタ６１０を通してＣＰＵ２００およびそのキャッシュメモリ２１０にフェッチ要求を通達する。
【００４１】

ここでスヌープした結果がミス（つまりキャッシュメモリ２１０には当該データが登録されていない）、もしくはメモリクリーン（つまりメモリの値が最新であったことが保障されたので、メモリからのフェッチを許可する）であれば、この情報がＣＰＵ２００より信号線Ｌ２００を通じてリプライされ、その信号はＮＵＭＡ制御回路２２０内のリクエストルータ６００、信号線Ｌ２１０を通じて部分バス２４０に通達され、さらにこの信号はバス分離結合回路５００、部分バス１４０、信号線Ｌ１２０を通じて、当該アクセス対象となっている部分主記憶１８０に接続されるディレクトリ制御回路１５０に通知される。

【００４２】

ディレクトリ制御回路１５０の内部を図４に示す。リクエストセレクタ７００までは、元々フェッチ要求パケットが部分バス１４０に出た時点で部分主記憶１８０へのリクエストとして到達し、リクエストが保留されているが、このようにリクエストが保留されると同時に信号線Ｌ７２０を通じてディレクトリ１６０を検索する動作を行う。ディレクトリ１６０は、ＣＰＵ１００、２００、３００、４００が主記憶をアクセスする単位（キャッシュブロックと一般に言う）毎に図５に示すようなエントリを用意するものであり（つまりディレクトリ１６０の中には図５に示されるようなエントリが多数入っている）、図５のビット１６２が１であることは当該ブロックがＣＰＵ１００によりキャッシュメモリに登録された（ことが過去にあり、まだキャッシュメモリに残っている可能性があるということ。正確にはキャッシュメモリから既に消滅している可能性もあるが、ディレクトリではキャッシュメモリに登録されていることになる）ことを表し、同様にビット１６４はＣＰＵ２００に、ビット１６６はＣＰＵ３００に、ビット１６８はＣＰＵ４００に対応する。

【００４３】

図５の例ではＣＰＵ１００、２００、３００によりアクセスされていることになるが、本節の事例では、このパタンではなく、値が０１００であったとする。つまり、他ＣＰＵについては、ＣＰＵ２００のみ当該データをキャッシュに取り込んでいる可能性があるとする。この信号が信号線Ｌ７３０を通じてリクエスト生成回路７１０に入るとリクエスト生成回路７１０は別途信号線７８０と通じて入るグループ設定レジスタ１７０の値と比較する。

【００４４】

グループ設定レジスタ１７０の中身を図６に示す。グループ設定レジスタ１７０は３２ビットのレジスタで、上位４ビットにＣＰＵ１００、２００、３００、４００に対応したビット１７２、１７４、１７６、１７８を持つ。それぞれのＣＰＵが当該ディレクトリ制御回路から見てバスで結合されている場合は１、バスが分離されている場合は０が登録されている。

【００４５】

リクエスト生成回路は７１０は、図６のような情報から、ディレクトリ１６０でチェックされたＣＰＵ２００がバスで結合されていることを判定できる。この場合は信号線Ｌ７４０を通じてリクエストセレクタ７００にＣＰＵ２００からのリプライを待つよう通達する。なお、もしディレクトリの値が０１１０でＣＰＵ３００も当該データをキャッシュに登録していることになっている場合には、同時にリクエスト生成回路７１０は信号線７４０、リクエストセレクタ７００、信号線Ｌ７１０、Ｌ１５０を通じてＮＵＭＡネットワーク１０００経由でのキャッシュコヒーレンス制御も行うことになるが、これについては（２）節で詳しく説明するので、本節では前述の通りディレクトリの値は０１００であったとして説明を続ける。

【００４６】

先述の通り、当該キャッシュブロックはディレクトリ１６０によりＣＰＵ２００のキャッシュに登録された可能性があったが、ＣＰＵ２００よりキャッシュミスもしくはメモリクリーンのリプライがリクエストセレクタ７００まで到達することで、リクエストセレクタ８００は当該フェッチ要求に対しては部分主記憶１８０をアクセスすべきであることが判定できることから、信号線Ｌ７２０を通じてディレクトリ１６０へのＣＰＵ１００分の

登録を通知するとともに（これにより当該ディレクトリエントリの値は０１００から１１００に変更される）、信号線Ｌ７５０、Ｌ１３０を通じて部分主記憶１８０へフェッチリクエストを出力する。部分主記憶１８０はこれに対して図１１に示されるフェッチリプライパケットを返答する。フェッチリプライパケットは、パケットのコマンド５２００が値００１０で、要求元プロセッサＩＤ５２１０（＝リプライ送付先プロセッサＩＤ）を持ち、これとフェッチしたデータ本体５２２０（キャッシュブロック分のサイズを持つ）とから成る。

【００４７】

このフェッチリプライパケットは、信号線Ｌ１３０から図４中の信号線Ｌ８１０、リプライルータ７２０、信号線Ｌ７９０、Ｌ１２０、部分バス１４０に出力され、更に信号線Ｌ１１０から図２中信号線Ｌ６３０を通じてセレクタ６１０に渡り、ここから信号線Ｌ６８０、Ｌ１００を経由してキャッシュメモリ１１０及びＣＰＵ１００へとリプライデータが戻る。

【００４８】

以上は、ＣＰＵ２００の返答がミスもしくはメモリクリーンであり、主記憶からデータを読むケースであったが、これがＣＰＵ２００が図７中のＭステータス２０２０であり（つまり最新のデータはＣＰＵ２００のキャッシュメモリ２１０に存在する）、ＣＰＵ２００のキャッシュメモリ２１０に登録されているデータをＣＰＵ１００が読み出す必要があるケースを次に説明する。

【００４９】

この場合はミスもしくはメモリクリーンのステータスではなく、
ＣＰＵ２００からはキャストアウトパケットが出力される。キャストアウトパケットを図１０に示す。コマンド５１００は０００１であり、その他に要求元プロセッサＩＤ５１１０（＝書き込み元プロセッサＩＤであり今回の場合で言えばＣＰＵ２００）、書き戻すべきアドレス５１２０、キャッシュブロック分のデータ５１３０から成る。

【００５０】

キャストアウトパケットはミスもしくはメモリクリーンのステータスと同様にディレクトリ制御回路１５０のリクエストセレクタ７００に到達すると、信号線Ｌ７５０、Ｌ１３０を通じて部分主記憶１８０にデータを書き戻す。リクエストセレクタ７００は、この書き戻し動作を待ち、後は前述ミスもしくはメモリクリーンのステータスの時と同様に部分主記憶１８０からデータを読み出す。（この際にディレクトリ１６０の当該エントリを０１００から１１００に変更することも前述の通りである）

なお、本節では部分主記憶１８０に対するアクセスについて説明したが、同様にバスで結合される部分主記憶２８０へのアクセスの場合も、ディレクトリ制御回路１５０の代わりにディレクトリ制御回路２５０が主体となるだけで、その動作は基本的に同一となる。

【００５１】

（１）－２：キャッシュ無効化要求

本節では、ＣＰＵ１００が既にキャッシュメモリ１１０に登録しているデータ（他のＣＰＵとの共有があり、キャッシュステータスは図７のＳステータス２０３０である）に対してストアを実行する場合、他のＣＰＵのキャッシュメモリをキャンセルする必要が発生する。本節では、この動作について説明する。なお、本節ではキャッシュ登録データの共有はＣＰＵ１００とＣＰＵ２００とで行われているとし、対象となるデータは部分主記憶１８０中に存在するものとする。つまり、ディレクトリ１６０の、当該データに対するエントリの値（図５相当）は１１００であるとする。

【００５２】

前節同様にしてキャッシュ無効化要求パケットがまずＣＰＵ１００から信号線Ｌ１００を通して出力される。キャッシュ無効化要求パケットを図１２に示す。コマンド５３００は００１１、要求元プロセッサＩＤ５３１０は今回の場合はＣＰＵ１００を示し、要求先プロセッサＩＤ５３２０は、ＣＰＵ１００から出力された時はＮｕｌｌ（ここでは２進数でオール１とする）となる。要求先プロセッサＩＤ５３１０はＮＵＭＡネットワークを通

じたコヒーレンス制御の際に意味のある値が入るフィールドであり、本節ではＮｕｌｌのままである。キャッシュ無効化要求パケットは更に無効化するアドレス５３３０を持つ。

【００５３】

キャッシュ無効化要求パケットは、前節のフェッチ要求パケット同様にＣＰＵ１００からＣＰＵ２００及びディレクトリ制御回路１５０に伝達されるが、処理内容は次の３点が異なる。一つ目の違いは、ＣＰＵ２００に伝達された結果はキャッシュのミスやメモリクリーンステータスの代わりに、キャッシュ無効化成功のステータスが返されること。二つ目の違いは、ディレクトリ制御回路は、ステータスが帰っても部分主記憶１８０へのアクセスを行わず、単にディレクトリ１６０の値の再設定を行うのみであること（本例では、１１００から１０００に変更）。三つ目の違いはＣＰＵ１００へはフェッチデータが返る代わりにキャッシュ無効化完了のステータスが返ること。

【００５４】

なお、前節同様、部分主記憶１８０のデータに対するキャッシュ無効化も、同様にバスで結合される部分主記憶２８０のデータに対するキャッシュ無効化も、ディレクトリ制御回路１５０の代わりにディレクトリ制御回路２５０が主体となるだけで、その動作は基本的に同一となる。

【００５５】

（１）－３：キャストアウト要求

Ｍステータス２０２０で登録していた情報を、他の新しいデータをキャッシュメモリ１１０に登録するために主記憶に書き戻す必要が発生した場合の動作を説明する（ここでは前節までと同様に部分主記憶１８０に対する書き戻し要求であるとする）。なお、Ｍステータス２０２０でデータを所有しているということは、同一キャッシュブロックを登録している可能性のあるＣＰＵは他に無いことを表しているので、ディレクトリ１６０の当該エントリの値は１０００となる。

【００５６】

キャストアウトの場合も、最初にＣＰＵ１００がキャストアウト要求パケットを信号線Ｌ１００を通して出力する。キャストアウト要求パケットについては、（１）－１で図１０を用いて説明した通りのフォーマットであり、今回のパタンでは要求元プロセッサＩＤにはＣＰＵ１００が入る。

【００５７】

キャストアウトは基本的には主記憶に対してデータを書き戻すだけのアクションであり、ＣＰＵ間でのコヒーレンス制御は不要なので、キャストアウト要求パケットはフェッチ要求パケット同様にディレクトリ制御回路１５０に到達した後は、他のＣＰＵのコヒーレンス制御を待つことなく、速やかに部分主記憶１８０への書き戻しを行う。具体的には図４のリクエストセレクタ７００でコヒーレンス操作の完了を待つことなく、信号線Ｌ７５０、Ｌ１３０を通じて速やかに部分主記憶１８０へ書き戻す。

【００５８】

なお、本アクションにおけるディレクトリ１６０の操作だが、キャストアウトが実行される場合は基本的には当該ＣＰＵ１００のキャッシュメモリ１１０からは当該キャッシュデータブロックは消滅するので、ディレクトリ１６０のエントリの値は１０００から００００に変更しても良い。但し、特別の命令を使用することで、キャッシュメモリ１１０から消滅させずにキャストアウトが実行できるような場合には、ディレクトリ１６０のエントリの値は１０００のままとする必要がある。本実施例では後者を前提とし、ディレクトリ１６０は変更しないとする。

【００５９】

前節同様、部分主記憶１８０のデータに対するキャストアウトも、同様にバスで結合される部分主記憶２８０のデータに対するキャストアウトも、ディレクトリ制御回路１５０の代わりにディレクトリ制御回路２５０が主体となるだけで、その動作は基本的に同一となる。

【００６０】

なお、本節ではＣＰＵ１００が自分自身の都合でキャストアウトする場合を説明したが、他ＣＰＵからのフェッチ要求を受けてキャストアウトする場合の動作については、バスで結合された範囲については（１）－１で説明した通りである。バスが分離された範囲については次節で説明する。

【００６１】

（２）ＮＵＭＡネットワークを通したコヒーレンス制御

前述の通り、バスで結合されたＣＰＵ同士でない場合、もしくはバスで結合されたＣＰＵ同士でも、キャッシュコヒーレンス制御対象のデータのアドレスがバスで結合された範囲の部分主記憶に対するものでない場合は、ＮＵＭＡネットワーク１０００を通したキャッシュコヒーレンス制御となる。

【００６２】

本節では、ＮＵＭＡネットワーク１０００を通したコヒーレンス制御について、バスを通した制御との差分を中心に説明する。

【００６３】

（２）－１：フェッチ要求

ここでは、まずバスで結合されていない部分主記憶に対してＮＵＭＡネットワークを経由してフェッチ要求を発行する場合を説明する。この動作はＮＵＭＡ制御回路１２０、２２０、３２０、４２０にて、フェッチ要求リクエストのアドレスとバス設定レジスタ１３０、２３０、３３０、４３０の値との関係で、バスが結合されていないことをＮＵＭＡ制御回路内のリクエストルータ６００で判定した場合、従来バスに信号線Ｌ６１０、Ｌ１１０を経由して出力していたフェッチ要求パケットを信号線Ｌ６２０、Ｌ１４０を経由してＮＵＭＡネットワーク１０００へ出力する。ＮＵＭＡネットワークはパケットの要求先アドレス５０２０を用いて行き先を例えば部分主記憶３８０と判定するとその部分主記憶に対応するディレクトリ制御回路３５０へとパケットを伝達する。　ディレクトリ制御回路内のリクエストセレクタ７００に伝達された先は、基本的にディレクトリ１６０の情報を用いてコヒーレンス制御を行う基本概念は（１）－１節と同一だが、フェッチ要求パケットはバス経由で入ってきたのではないことから、ディレクトリを検索した結果、全てのコヒーレンス制御パケットをディレクトリ生成回路７１０で生成し、リプライルータ７２０、信号線Ｌ８２０、Ｌ１５０を経由して、コヒーレンス制御パケットもまたＮＵＭＡネットワーク１０００を用いて対象プロセッサに分配しなければならない。分配したパケットはＮＵＭＡ制御回路１２０、２２０、３２０、４２０に入り、セレクタ６１０を経由して各ＣＰＵ１００、２００、３００、４００に通達される。この結果例えばＭステータスのデータを持ったＣＰＵが存在し、キャッシュメモリ上のデータを部分主記憶に書き戻す必要が出た場合にも、ＮＵＭＡ制御回路１２０、２２０、３２０、４２０を経由してＮＵＭＡネットワーク１０００を通って書き戻す。

【００６４】

要は、バスで結合されていない部分主記憶に対してＮＵＭＡネットワークを経由してフェッチ要求を発行する場合については、前節でバスを経由して実行していた動作全てが、基本的にＮＵＭＡネットワーク１０００を経由して実行されることになる。

【００６５】

ＮＵＭＡネットワークを経由するフェッチ要求については、バスで結合される部分主記憶１８０、２８０、３８０、４８０に対するフェッチ要求だったが、ディレクトリ１６０、２６０、３６０、４６０を検索した結果、バスで結合されないＣＰＵ１００、２００、３００、４００もキャッシュメモリ１１０、２１０、３１０、４１０に登録しているケースがある。この場合には、（１）－１節でディレクトリ制御回路１５０内で、ディレクトリ１６０を検索した結果、ＣＰＵ２００のキャッシュメモリへの登録が分かり、リクエストセレクタ７００に、バス結合されたＣＰＵ２００についてのキャッシュコヒーレンス制御解消を待ってフェッチ要求パケットを保留していた部分の動作に次の変更が加わる。すなわち、ディレクトリ１６０を検索した結果、バスで結合されないＣＰＵ，例えばＣＰＵ３００がキャッシュメモリに登録していることが判明した時点で、リクエスト生成回路７

１０よりフェッチ要求パケットを生成し、リプライルータ７２０、信号線８２０、Ｌ１５０、ＮＵＭＡネットワーク１０００を経由して、ＮＵＭＡ制御回路３２０経由でＣＰＵ３００に伝達し、このＣＰＵ３００からのリプライが再びＮＵＭＡ制御回路３２０及びＮＵＭＡネットワーク１０００を経由してディレクトリ制御回路１５０に返答されるまでリクエストセレクタ７００にリクエストを保持することになる。本リクエストの結果、ＣＰＵ３００からキャストアウト要求が発生した場合にも、同様にＮＵＭＡ制御回路３２０とＮＵＭＡネットワーク１０００を経由して、ディレクトリ制御回路にそのキャストアウト要求が伝播し、これを部分主記憶１８０に書き戻してからフェッチ動作をディレクトリ制御回路１５０が実行することになる。

【００６６】

ディレクトリ１６０、２６０、３６０、４６０への設定内容については、（１）節と同様に、新たにフェッチを行ったＣＰＵを新たにディレクトリに登録することになる。

【００６７】

（２）－２：キャッシュ無効化要求

ＮＵＭＡネットワーク１０００経由のキャッシュ無効化要求には、（２）－１と同様に、バスで結合されていない部分主記憶のデータに対するキャッシュ無効化要求の場合と、バスで結合されている部分主記憶のデータに対するキャッシュ無効化要求がバスで結合されていないＣＰＵに対して発生するケースとがある。それぞれ基本的には（２）－１と同様であるが、但しフェッチリプライの代わりに、図１３で示されるキャッシュ無効化完了パケットが返答されてくることになる。

【００６８】

本動作を行った場合にも（１）－２同様に、当該ディレクトリエントリの値は、キャッシュ無効化要求を行った１ＣＰＵ以外は全て０に戻される。

【００６９】

（２）－３：キャストアウト要求

ＮＵＭＡネットワーク１０００経由のキャストアウトは、バスが分離されてる部分主記憶１８０、２８０、３８０、４８０への書き戻しの際に発生するが、これもＣＰＵ１００、２００、３００、４００から書き戻し要求がＮＵＭＡ制御回路１２０、２２０、３２０、４２０に伝達されると、バス設定レジスタ１３０、２３０、３３０、４３０の値に応じてリクエストルータ６００によりＮＵＭＡネットワーク１０００への出力が選択され、このキャストアウト要求パケットはＮＵＭＡネットワーク１０００からディレクトリ制御回路１５０、２５０、３５０、４５０を経由して部分主記憶１８０、２８０、３８０、４８０に書き戻される。本実施例ではキャストアウトによる書き戻しの際にディレクトリの設定値を変更しないとしたが（（１）－３参照）、ＮＵＭＡネットワーク１０００経由のキャストアウトでもこれは同じである。

【００７０】

（３）バス結合の変更時の処理

（１）（２）の動作により、ディレクトリ制御回路１５０、２５０、３５０、４５０内のディレクトリ１６０、２６０、３６０、４６０には、バス分離・結合に関わらず、当該データブロックをキャッシュメモリに登録してうる全てのＣＰＵが登録されている。これにより、バス結合の状態が変更されても、変更後のバスの結合・分離に従ってディレクトリ１６０、２６０、３６０、４６０に従ったキャッシュコヒーレンス制御（バスで接続されるＣＰＵの組についてはバスでの制御期待でＮＵＭＡネットワーク経由の制御なし）が実現できる。

【００７１】

本節では、バス接続形態を変更する際の動作を図１を用いて更に説明する。例えばＣＰＵ１００がバスの接続形態を変更したい場合には、まず信号線Ｌ１０を通してサービスプロセッサ１０に要求を通達する。サービスプロセッサは要求ＣＰＵ１００以外のＣＰＵ２００、３００、４００を信号線Ｌ３０、Ｌ５０、Ｌ７０を通じてストップさせ、これが完了するとバス分離結合回路５００、５１０、５２０の設定値を変えるとともに、その変更

の完了をＣＰＵ１００に通知する。

【００７２】

ＣＰＵ１００は、アドレス空間上に図８の通りマップされるバス設定レジスタ（Ａ）～（Ｄ）、グループ設定レジスタ（Ａ）～（Ｄ）（実体は、バス設定レジスタ１３０、２３０、３３０、４３０およびグループ設定レジスタ１７０、２７０、３７０、４７０をバス接続形態に応じた値に変更する。

【００７３】

なお、各レジスタへのアクセスは、（バスの接続形態がどう変化するか設定値により変わりうるために）設定値によらず、全てＮＵＭＡネットワーク１０００を経由して実行される。例えばＣＰＵ１００がバス設定レジスタ２３０を更新する場合には、ＮＵＭＡ制御回路１２０にて当該リクエストはＮＵＭＡネットワーク経由と判定され、ＮＵＭＡネットワーク経由でＮＵＭＡ制御回路２２０中のバス設定レジスタ２３０を設定することになる。

【００７４】

以上の通り、バス分割結合回路５００、５１０、５２０はサービスプロセッサ１０が、バス設定レジスタ１３０、２３０、３３０、４３０及びグループ設定レジスタ１７０、２７０、３７０、４７０はＣＰＵ１００、２００、３００、４００自身が変更することにより、バスの分離結合形態に関する全ての設定が変更できる。変更後にサービスプロセッサ１０を経由して全てのＣＰＵ１００、２００、３００、４００の動作を再開すると、以降変更後のバスの形態に従い、正しいキャッシュコヒーレンス制御が実行されることになる。

【実施例２】

【００７５】

実施例１では、ＮＵＭＡ制御用に専用のＮＵＭＡネットワーク１０００が存在することを前提としたが、以下にのべる実施例２では、ＮＵＭＡネットワーク１０００の代わりに、ＮＵＭＡプロトコル用のパケットもバスを経由して実行される。

【００７６】

図１４に実施例２のシステム構成を示す。図１に示される実施例１との違いは３点ある。

【００７７】

第１点目は、バス分割結合回路５００、５１０、５２０をバスフィルター回路５０５、５１５、５２５に置き換えたことである。バスフィルター回路５０５、５１５、５２５は、ＮＵＭＡ制御用以外のパケットに対してはバス分割結合回路５００、５１０、５２０と同様に機能する（つまりサービスプロセッサ１０による設定により、パケットを通したり通さなくなったりする）が、ＮＵＭＡ制御用パケットに対しては、常にこれを通す機能を持つ。

【００７８】

第２点目は、実施例１では例えばフェッチ要求パケットはバス結合のＣＰＵの間でもＮＵＭＡネットワークでの結合のＣＰＵの間でも、同じコマンド（００００）を使用していたが、これだとバスフィルター回路５０５、５１５、５２５がＮＵＭＡ制御用とそれ以外との分類が出来なくなることを考慮し、最上位１ビットを１とするように変更する。つまり、バス間のフェッチリクエストではコマンドは００００だが、ＮＵＭＡ接続のプロセッサの間でのフェッチ要求パケットはコマンドを１０００とすることになる。なお、本変更はＮＵＭＡ制御回路１２０内のリクエストルータ６００、及びディレクトリ制御回路１５０内のリクエスト生成回路７１０を変更することで実現する。

【００７９】

第３点目は、実施例１ではＮＵＭＡネットワーク１０００に対してＮＵＭＡ制御パケットをやり取りしていたＮＵＭＡ制御回路１２０、２２０、３２０、４２０及びＮＵＭＡディレクトリ制御回路１５０、２５０、３５０、４５０のＮＵＭＡネットワーク１０００への入出力の口を全て部分バス１４０、２４０、３４０、４４０に対する口と統合すること

である。

【００８０】

なお、構成上の上記３点の変更とは別だが、実施例１でＮＵＭＡ制御パケットがＮＵＭＡネットワーク１０００により１対１で実行されていたことに対し、実施例２の構成では、パケットの伝達という観点だけで見れば、部分バス１４０、２４０、３４０、４４０の全てにＮＵＭＡ制御パケットがブロードキャストされてしまうことがある。但し、実際にはアドレスやプロセッサＩＤにより各パケットは１対１で機能することから、実質的なパケットの処理については実施例１と差異はない。

【００８１】

ＮＵＭＡ制御パケットがブロードキャスト伝達されてしまうことで、バスの上での輻輳が増加する危険があるが、ジョブの実行形態（例えば、あるユーザＪＯＢをＣＰＵ１００とＣＰＵ２００とで並列実行する）に即したパーティションの設定（前述の例であればＣＰＵ１００とＣＰＵ２００とを同一のパーティションに設定する、つまり部分バス１４０と部分バス２４０との間は完全結合し、全てのリクエストを通す。但し部分バス２４０と部分バス３４０とはフィルターをかけ、ＮＵＭＡ制御パケットのみを通す設定とする）をすることで、実質的にＮＵＭＡ制御パケットの発生頻度は著しく低減されるので、ＮＵＭＡ制御パケットのブロードキャストによる性能低下は問題にならない。

【００８２】

上記の通り、ＮＵＭＡネットワーク１０００で制御していたパケットを部分バス１４０、２４０、３４０、４４０で伝達することにより、ＣＰＵ間のネットワーク接続無しに本発明の特徴である性能のスケーラビリティを確保しながら、パーティション設定にも自由度があるマルチプロセッサが実現することになる。

【図面の簡単な説明】

【００８３】

【図１】本発明の実施例１の並列計算機の全体構成を表すブロック図である。
【図２】上記実施例のＮＵＭＡ制御回路を表すブロック図である。
【図３】上記実施例のバス設定レジスタを表す図である。
【図４】上記実施例のディレクトリ制御回路を表すブロック図である。
【図５】上記実施例のディレクトリのエントリを表す図である。
【図６】上記実施例のグループ設定レジスタを表す図である。
【図７】上記実施例のキャッシュメモリの状態遷移を表す図である。
【図８】上記実施例の並列計算機のアドレスマップを表す図である。
【図９】上記実施例のフェッチ要求パケットを表す図である。
【図１０】上記実施例のキャストアウト要求パケットを表す図である。
【図１１】上記実施例本発明のフェッチリプライパケットを表す図である。
【図１２】上記実施例本発明のキャッシュ無効化要求パケットを表す図である。
【図１３】上記実施例本発明のキャッシュ無効化報告パケットを表す図である。
【図１４】本発明の実施例２の並列計算機の全体構成を表ブロック図である。

【符号の説明】

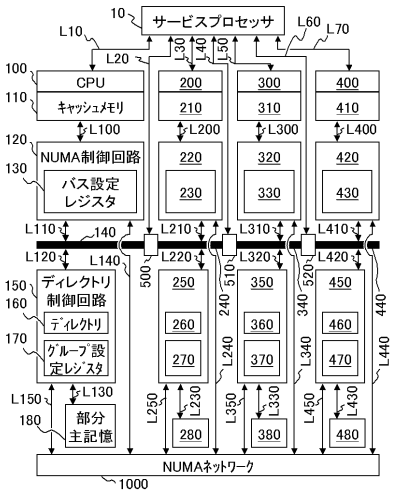
【００８４】

２００、３００、４００…ＣＰＵ
２１０、３１０、４１０…キャッシュメモリ
２２０、３２０、４２０…ＮＵＭＡ制御回路
２３０、３３０、４３０…バス設定レジスタ
１４０、２４０、３４０、４４０…部分バス
２５０、３５０、４５０…ディレクトリ制御回路
２６０、３６０、４６０…ディレクトリ
２７０、３７０、４７０…グループ設定レジスタ
２８０、３８０、４８０…部分主記憶
１３２、１３４、１３６、１３８…バス設定ビット

１６２、１６４、１６６、１６８…ディレクトリビット
１７２、１７４、１７６、１７８…グループ設定ビット
５００、５１０、５２０…バス分割結合回路
５０５、５１５、５２５…バスフィルター回路
５０００、５１００、５２００、５３００、５４００…コマンド
Ｌ１０～Ｌ８００…信号線。

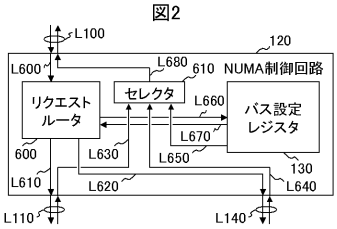
【図１】

図1



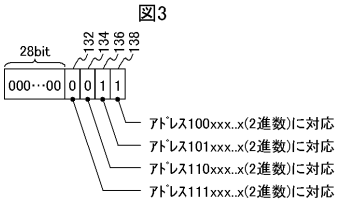
【図２】

図2



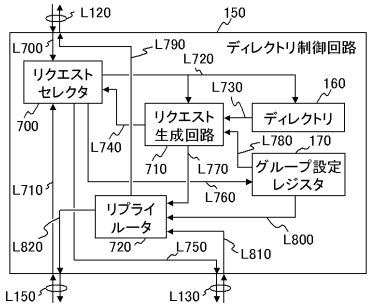
【図３】

図3



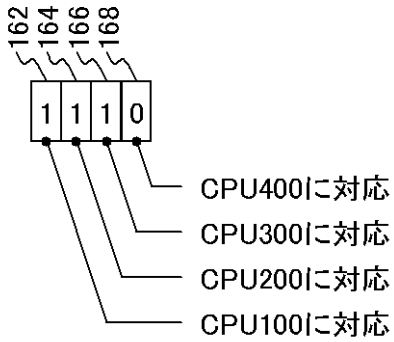
【図４】

図4



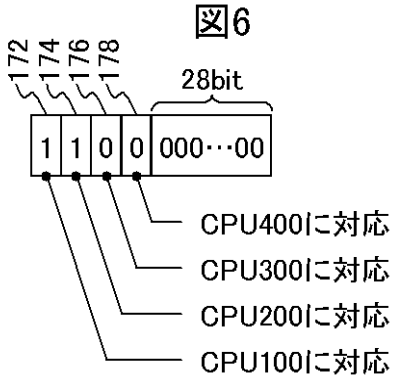
【図５】

図5



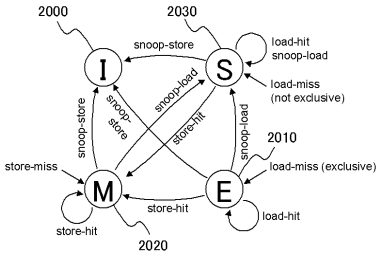
【図６】

図6



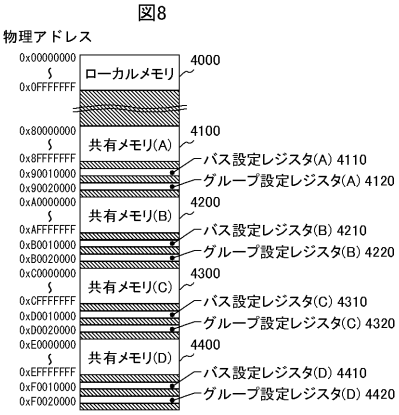
【図７】

図7



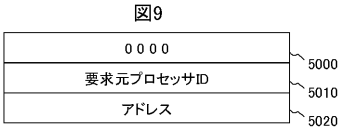
【図８】

図8



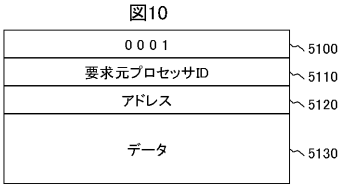
【図９】

図9



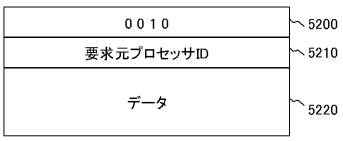
【図１０】

図10



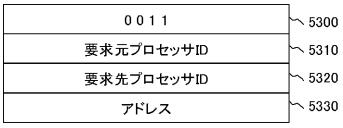
【図１１】

図11



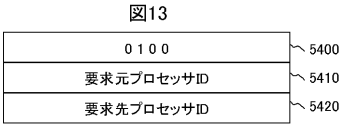
【図１２】

図12



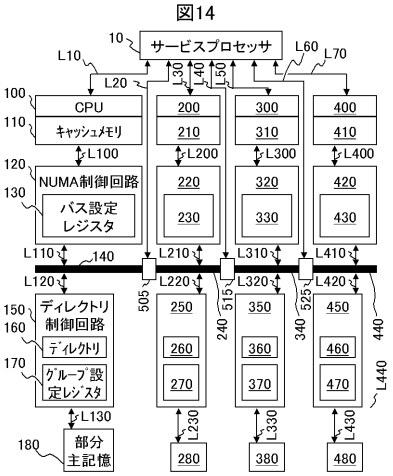
【図１３】

図13



【図１４】

図14



フロントページの続き

(56)参考文献 特開２００２－３０４３２８（ＪＰ，Ａ）
特開平０８－３２０８２７（ＪＰ，Ａ）
特開平０８－０１６４７４（ＪＰ，Ａ）
特開平０９－１９８３０９（ＪＰ，Ａ）
特開平０５－１０８５７８（ＪＰ，Ａ）
特開平１０－２４０７０７（ＪＰ，Ａ）
特開２０００－２５０８８２（ＪＰ，Ａ）
特開平０９－２０４４０５（ＪＰ，Ａ）
寺澤 卓也 Takuya TERASAWA, 計算機の記憶システムーＩＶ, 情報処理 第３４巻 第１号 Journal of Information Processing Society of Japan, 日本, 社団法人情報処理学会 Information Processing Society of Japan, １９９３年 1月, 第34巻, p.96-105
寺澤 卓也 Takuya TERASAWA, 計算機の記憶システムーＩＶ, 情報処理 第３４巻 第２号 Journal of Information Processing Society of Japan, 日本, 社団法人情報処理学会 Information Processing Society of Japan, １９９３年 2月, 第34巻, p.233-243

(58)調査した分野(Int.Cl., ＤＢ名)
Ｇ０６Ｆ １２／０８－１２／１２