(19)日本国特許庁(JP)　(12)公表特許公報(A)　(11)特許出願公表番号
特表2007-526540
(P2007-526540A)
(43)公表日　平成19年9月13日(2007.9.13)

(51)Int.Cl.		F I			テーマコード（参考）
G06F 15/177	(2006.01)	G06F	15/177	A	5B045
G06F 15/173	(2006.01)	G06F	15/173	640A	

審査請求 未請求　予備審査請求 未請求　（全 20 頁）

(21)出願番号	特願2006-517396 (P2006-517396)
(86)(22)出願日	平成16年6月18日 (2004.6.18)
(85)翻訳文提出日	平成18年2月23日 (2006.2.23)
(86)国際出願番号	PCT/US2004/019524
(87)国際公開番号	WO2005/003893
(87)国際公開日	平成17年1月13日 (2005.1.13)
(31)優先権主張番号	10/607,819
(32)優先日	平成15年6月27日 (2003.6.27)
(33)優先権主張国	米国 (US)

(71)出願人　504056174
ニューイシス・インコーポレーテッド
ＮＥＷＩＳＹＳ　ＩＮＣＯＲＰＯＲＡＴＥＤ
アメリカ合衆国　テキサス州７８７３０，オースティン、ブリッジポイント・パークウェイ、６２００、ビルディング　４，スイート　１００
(74)代理人　110000028
特許業務法人明成国際特許事務所
(72)発明者　コタ・ラジェッシュ
アメリカ合衆国　テキサス州７８７２７　オースティン，アラメダ・トレイス・サークル，１２３２０，アパートメント　＃１１０７

最終頁に続く

(54)【発明の名称】動的マルチクラスタシステムリコンフィギュレーション

(57)【要約】
【課題】動的なマルチクラスタシステムのコンフィギュレーションの変更を可能にする方法および装置を提供する。
【解決手段】ある例では、マルチクラスタシステムにおけるプロセッサ群は、仮想アドレス空間を共有する。動的にプロセッサ、Ｉ／Ｏリソース、およびクラスタを導入および除去するメカニズムが提供される。このメカニズムは、リセットのあいだ、またはシステムが動作中に実現されえる。リンクは動的にイネーブルまたはディセーブルされえる。
【選択図】図８

【特許請求の範囲】
【請求項１】
コンピュータシステムであって、
第１複数プロセッサおよび第１相互接続コントローラを含む第１クラスタであって、前記第１複数プロセッサおよび前記第１相互接続コントローラはポイントツーポイントアーキテクチャを用いて通信する、第１クラスタ、
第２複数プロセッサおよび第２相互接続コントローラを含む第２クラスタであって、前記第２複数プロセッサおよび前記第２相互接続コントローラはポイントツーポイントアーキテクチャを用いて通信する、第２クラスタ
を備え、
前記第１相互接続コントローラから前記第２相互接続コントローラへのリンクのためのポリングが、前記第１相互接続コントローラをコンフィギャすることによってイネーブルまたはディセーブルされえる
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であって、プロセッサの前記第１クラスタおよびプロセッサの前記第２クラスタは単一の仮想アドレス空間を共有するコンピュータシステム。
【請求項３】
請求項１～２のいずれかに記載のコンピュータシステムであって、前記第１相互接続コントローラは、物理レイヤイネーブル指示子を含むコンピュータシステム。
【請求項４】
請求項１～３のいずれかに記載のコンピュータシステムであって、前記第１相互接続コントローラは、前記第１相互接続コントローラおよび前記第２相互接続コントローラ間で論理パケットの送信を防ぐようコンフィギャ可能なフェンス指示子を含むコンピュータシステム。
【請求項５】
請求項１～４のいずれかに記載のコンピュータシステムであって、前記第１相互接続コントローラは、前記第１相互接続コントローラが前記リンクを再初期化するよう指示するようコンフィギャ可能な再初期化指示子を含むコンピュータシステム。
【請求項６】
請求項５に記載のコンピュータシステムであって、再初期化は、前記第１相互接続コントローラに関連付けられる送信機にトレーニングシーケンスを前記第２相互接続コントローラへ送信させることを含むコンピュータシステム。
【請求項７】
請求項６に記載のコンピュータシステムであって、前記送信機は、ポリングアクティブ状態が設定されるときに前記トレーニングシーケンスを送るコンピュータシステム。
【請求項８】
請求項７に記載のコンピュータシステムであって、前記送信機は、前記ポリングスリープ状態が設定されるときにトレーニングシーケンスを送らないコンピュータシステム。
【請求項９】
請求項５に記載のコンピュータシステムであって、再初期化は、前記第１相互接続コントローラに関連付けられる送信機に初期化シーケンスを前記第２相互接続コントローラへ送信させることを含むコンピュータシステム。
【請求項１０】
請求項１～９のいずれかに記載のコンピュータシステムであって、前記第１相互接続コントローラは、プロセッサのリモートクラスタを特定する値を保持するよう動作可能である複数のクラスタＩＤ指示子を含むコンピュータシステム。
【請求項１１】
請求項１～１０のいずれかに記載のコンピュータシステムであって、前記第１相互接続

コントローラは、物理レイヤイネーブル、フェンス、再初期化、およびクラスタＩＤビットを備えるコンフィギュレーションスペースレジスタを含むコンピュータシステム。
【請求項１２】
プロセッサのクラスタを導入する方法であって、
第２相互接続コントローラの存在を探してポリングするポイントツーポイントアーキテクチャを用いて通信する第１複数プロセッサを含む第１クラスタ中の第１相互接続コントローラをコンフィギャすること、
ポイントツーポイントアーキテクチャを用いて通信する第２複数プロセッサを含む第２クラスタ中で第２クラスタ中の第２相互接続コントローラ上のリセット信号をアサートすること、
リンクレイヤプロトコルを前記第１および第２相互接続コントローラ間の接続上で確立すること
を含む方法。
【請求項１３】
請求項１２に記載の方法であって、ポリングは連続的に実行される方法。
【請求項１４】
請求項１２～１３のいずれかに記載の方法であって、前記第１相互接続コントローラは、物理レイヤイネーブル指示子を含む方法。
【請求項１５】
請求項１２～１４のいずれかに記載の方法であって、前記第１相互接続コントローラは、前記第１相互接続コントローラおよび前記第２相互接続コントローラ間で論理パケットの送信を防ぐようコンフィギャ可能なフェンス指示子を含む方法。
【請求項１６】
請求項１２～１５のいずれかに記載の方法であって、前記第１相互接続コントローラは、前記第１相互接続コントローラが前記リンクを再初期化するよう指示するようコンフィギャ可能な再初期化指示子を含む方法。
【請求項１７】
請求項１２～１６に記載の方法であって、再初期化は、前記第１相互接続コントローラに関連付けられる送信機にトレーニングシーケンスを前記第２相互接続コントローラへ送信させることを含む方法。
【請求項１８】
請求項１７に記載の方法であって、前記送信機は、ポリングアクティブ状態が設定されるときに前記トレーニングシーケンスを送る方法。
【請求項１９】
請求項１８に記載の方法であって、前記送信機は、前記ポリングスリープ状態が設定されるときにトレーニングシーケンスを送らない方法。
【請求項２０】
請求項１６に記載の方法であって、再初期化は、前記第１相互接続コントローラに関連付けられる送信機に初期化シーケンスを前記第２相互接続コントローラへ送信させることを含む方法。
【請求項２１】
請求項１２～２０のいずれかに記載の方法であって、前記第１相互接続コントローラは、プロセッサのリモートクラスタを特定する値を保持するよう動作可能である複数のクラスタＩＤ指示子を含む方法。
【請求項２２】
請求項１２～２１のいずれかに記載の方法であって、前記第１相互接続コントローラは、物理レイヤイネーブル、フェンス、再初期化、およびクラスタＩＤビットを備えるコンフィギュレーションスペースレジスタを含む方法。
【請求項２３】
コンピュータシステムであって、

第２相互接続コントローラの存在を探してポリングするポイントツーポイントアーキテクチャを用いて通信する第１複数プロセッサを含む第１クラスタ中の第１相互接続コントローラをコンフィギャする手段、
ポイントツーポイントアーキテクチャを用いて通信する第２複数プロセッサを含む第２クラスタ中で第２クラスタ中の第２相互接続コントローラ上のリセット信号をアサートする手段、
リンクレイヤプロトコルを前記第１および第２相互接続コントローラ間の接続上で確立する手段
を備えるコンピュータシステム。
【請求項２４】
請求項２３に記載のコンピュータシステムであって、ポリングは連続的に実行されるコンピュータシステム。
【請求項２５】
請求項２３～２４のいずれかに記載のコンピュータシステムであって、前記第１相互接続コントローラは、物理レイヤイネーブル指示子を含むコンピュータシステム。
【請求項２６】
請求項２３～２５のいずれかに記載のコンピュータシステムであって、前記第１相互接続コントローラは、前記第１相互接続コントローラおよび前記第２相互接続コントローラ間で論理パケットの送信を防ぐようコンフィギャ可能なフェンス指示子を含むコンピュータシステム。
【請求項２７】
請求項２３～２６のいずれかに記載のコンピュータシステムであって、前記第１相互接続コントローラは、前記第１相互接続コントローラが前記リンクを再初期化するよう指示するようコンフィギャ可能な再初期化指示子を含むコンピュータシステム。
【請求項２８】
請求項２３～２７に記載のコンピュータシステムであって、再初期化は、前記第１相互接続コントローラに関連付けられる送信機にトレーニングシーケンスを前記第２相互接続コントローラへ送信させることを含むコンピュータシステム。
【請求項２９】
請求項２８に記載のコンピュータシステムであって、前記送信機は、ポリングアクティブ状態が設定されるときに前記トレーニングシーケンスを送るコンピュータシステム。
【請求項３０】
請求項２９に記載のコンピュータシステムであって、前記送信機は、前記ポリングスリープ状態が設定されるときにトレーニングシーケンスを送らないコンピュータシステム。
【請求項３１】
請求項２７に記載のコンピュータシステムであって、再初期化は、前記第１相互接続コントローラに関連付けられる送信機に初期化シーケンスを前記第２相互接続コントローラへ送信させることを含むコンピュータシステム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、一般にマルチプロセッサシステムに関する。より具体的には、本発明は、マルチクラスタシステム中のリソースを効果的かつ効率的に管理する技術に関する。
【背景技術】
【０００２】
パフォーマンスの制限のために、単一のメモリ空間でシステム中のプロセッサ群を接続するポイントツーポイントアーキテクチャが開発されるようになっている。ある例では、個別プロセッサは、互いに複数のポイントツーポイントリンクを通して直接に接続されることによってプロセッサ群のクラスタを形成しえる。プロセッサ群の別個のクラスタも接続されえる。ポイントツーポイントリンクは、コプロセッシングおよびマルチプロセッシ

ング機能のための帯域幅を大きく増加させる。

【０００３】

マルチプロセッサクラスタを接続するのにさまざまなメカニズムが用いられえる。しかし、動的にマルチクラスタシステムのコンフィギュレーションを変化させるメカニズムは限られてきた。さまざまな困難のために、マルチクラスタシステムの物理レイヤプロトコルを含む通信プロトコルのその機能性が制限されてきた。

【発明の開示】

【発明が解決しようとする課題】

【０００４】

したがって、ポイントツーポイントリンクを用いて接続されるマルチプロセッサのマルチクラスタを有するシステム中のクラスタ群の間の通信のためのメカニズムおよび技術を改良する技術を提供することが望ましい。

【課題を解決するための手段】

【０００５】

本発明によれば、動的なマルチクラスタシステムのコンフィギュレーションの変更を可能にする方法および装置が提供される。ある例では、マルチクラスタシステムにおけるプロセッサ群は、仮想アドレス空間を共有する。動的にプロセッサ、Ｉ／Ｏリソース、およびクラスタを導入および除去するメカニズムが提供される。このメカニズムは、リセットのあいだ、またはシステムが動作中に実現されえる。リンクは動的にイネーブルまたはディセーブルされえる。

【０００６】

ある実施形態において、コンピュータシステムが提供される。このコンピュータシステムは、第１クラスタおよび第２クラスタを含む。第１クラスタは、第１複数プロセッサおよび第１相互接続コントローラを含む。前記第１複数プロセッサおよび前記第１相互接続コントローラはポイントツーポイントアーキテクチャを用いて通信する。第２クラスタは、第２複数プロセッサおよび第２相互接続コントローラを含む。前記第２複数プロセッサおよび前記第２相互接続コントローラはポイントツーポイントアーキテクチャを用いて通信する。前記第１相互接続コントローラから前記第２相互接続コントローラへのリンクのためのポリングが、前記第１相互接続コントローラをコンフィギャすることによってイネーブルまたはディセーブルされえる。

【０００７】

他の例では、プロセッサのクラスタを導入する方法が提供される。第２相互接続コントローラの存在を探してポリングするよう、ポイントツーポイントアーキテクチャを用いて通信する第１複数プロセッサを含む第１クラスタ中の第１相互接続コントローラがコンフィギャされる。ポイントツーポイントアーキテクチャを用いて通信する第２複数プロセッサを含む第２クラスタ中で第２クラスタ中の第２相互接続コントローラ上でリセット信号がアサートされる。リンクレイヤプロトコルが前記第１および第２相互接続コントローラ間の接続上で確立される。

【０００８】

本発明の性質および優位性のさらなる理解は、明細書および図面の残りの部分を参照することによって実現されよう。

【発明を実施するための最良の形態】

【０００９】

本発明は、以下の説明を添付の図面と併せて参照することによって最もよく理解されえ、図面は本発明の具体的な実施形態を図示するものである。

【００１０】

本発明を実施するための発明者によって考えられたベストモードを含む本発明のいくつかの具体的な実施形態が詳細にここで参照される。これら具体的な実施形態の例は、添付の図面に示される。本発明は、これら具体的な実施形態について説明されるが、本発明を記載された実施形態に限定するよう意図されてはいないことが理解されよう。むしろ、代

替物、改変、および等価物を、添付の特許請求の範囲によって規定される本発明の精神および範囲内に含まれるとしてカバーするよう意図される。そのプロセッサ間でポイントツーポイント通信を有するマルチプロセッサアーキテクチャは、本発明の具体的な実施形態を実現するのに適する。以下の記載において、本発明の完全な理解を促すために多くの具体的な詳細が述べられている。本発明は、これら具体的な詳細の一部または全てなしでも実施されえる。本発明の趣旨を不必要にぼかさないために、よく知られたプロセス操作は詳細には記載されていない。さらに本発明が単一のエンティティを参照する場合、本発明の方法および装置は、文脈上明らかにそうでないと表されない限り、１より多いエンティティを用いて実現可能である。

【００１１】

図１Ａは、本発明の技術を採用しえるマルチクラスタ、マルチプロセッサシステムの一例の概略図である。それぞれの処理クラスタ１０１、１０３、１０５、および１０７は、複数のプロセッサを含む。処理クラスタ１０１、１０３、１０５、および１０７は、互いにポイントツーポイントリンク１１１ａ～ｆを通して接続される。ある実施形態において、図１Ａに示されるマルチクラスタアーキテクチャ中のマルチプロセッサは、同じメモリ空間を共有する。この例では、ポイントツーポイントリンク１１１ａ～ｆは、マルチクラスタ１０１、１０３、１０５、および１０７内のマルチプロセッサを接続するために従来のフロントサイドバスに代わって用いられる内部システム接続である。このポイントツーポイントリンクは、任意のポイントツーポイントのコヒーレンスプロトコルをサポートしえる。

【００１２】

図１Ｂは、本発明の技術を採用しえるマルチクラスタ、マルチプロセッサシステムの他の例の概略図である。それぞれの処理クラスタ１２１、１２３、１２５、および１２７は、ポイントツーポイントリンク１４１ａ～ｄを通してスイッチ１３１に結合される。スイッチおよびポイントツーポイントリンクを用いることは、システム内でマルチクラスタを接続するとき、より少ないポイントツーポイントリンクで実現することを可能にすることに注意されたい。スイッチ１３１は、コヒーレンスプロトコルインタフェースを持つプロセッサを含みえる。さまざまな実現例によれば、図１Ａに示されるマルチクラスタシステムは、図１Ｂに示されるスイッチ１３１を用いて拡張されえる。

【００１３】

図２は、例えば図１Ａに示されるクラスタ１０１のようなマルチプロセッサクラスタの概略図である。クラスタ２００は、プロセッサ２０２ａ～２０２ｄ、１つ以上の基本Ｉ／Ｏシステム（ＢＩＯＳ）２０４、メモリバンク２０６ａ～２０６ｄを備えるメモリサブシステム、ポイントツーポイント通信リンク２０８ａ～２０８ｅ、およびサービスプロセッサ２１２を含む。ポイントツーポイント通信リンクは、プロセッサ２０２ａ～２０２ｄ、Ｉ／Ｏスイッチ２１０、および相互接続コントローラ２３０の間で相互接続を可能にするよう構成される。サービスプロセッサ２１２は、リンク２１４ａ～２１４ｆによって図２に表されるＪＴＡＧインタフェースを介してプロセッサ２０２ａ～２０２ｄ、Ｉ／Ｏスイッチ２１０、および相互接続コントローラ２３０と通信を可能にするよう構成される。他のインタフェースがサポートされることに注意されたい。ある実現例においては、サービスプロセッサは、マルチプロセッサクラスタに含まれないことにも注意されたい。Ｉ／Ｏスイッチ２１０は、システムの残りをＩ／Ｏアダプタ２１６および２２０に接続する。

【００１４】

具体的な実施形態によれば、本発明のサービスプロセッサは、システムリソースを以前に特定されたパーティションスキーマに従って分割する（partition）ためのインテリジェンスを有する。パーティショニングは、システムプロセッサに関連付けられたルーティングテーブルをサービスプロセッサによって直接に操作することによって達成されえ、これはポイントツーポイント通信インフラストラクチャによって可能になる。ルーティングテーブルは、さまざまなシステムリソースを制御および分離するために用いられ、リソース間の接続がテーブル中で定義される。

【００１５】
プロセッサ２０２ａ～ｄは、相互接続コントローラ２３０にもポイントツーポイントリンク２３２ａ～ｄを通して結合される。さまざまな実施形態によれば、また後で詳述されるように、相互接続コントローラ２３０は、システム中の相互接続されたプロセッサの数が、ノードＩＤスペースおよび複数のプロセッサクラスタのそれぞれに関連付けられたマッピングテーブル制限を越えることができるようにするさまざまな機能を実行する。ある実施形態によれば、相互接続コントローラ２３０は、クラスタにわたるキャッシュコヒーレンシの維持を含むさまざまな他の機能を実行する。相互接続コントローラ２３０は、他のマルチプロセッサクラスタ群に関連付けられた同様のコントローラ群に結合されえる。１つのクラスタにこのような相互接続コントローラが１つより多く存在しえることに注意されたい。相互接続コントローラ２３０は、リモートクラスタと共にプロセッサ２０２ａ～ｄともポイントツーポイントプロトコルを用いて通信する。

【００１６】
より一般には、図２に示される具体的なアーキテクチャは、単に例示的であって、本発明の実施形態は、異なる構成およびリソース相互結合、および示されたシステムリソースのそれぞれについてさまざまな代替物を有すると想定されることが理解されよう。しかし例示する目的のためにサーバ２００の特定の詳細が想定される。例えば、図２に示されるリソースのほとんどは、単一の電子アセンブリ上に常駐すると想定される。加えて、メモリバンク２０６ａ～２０６ｄは、デュアルインラインメモリモジュール（ＤＩＭＭ）として物理的に提供されるダブルデータレート（ＤＤＲ）メモリを備えうる。Ｉ／Ｏアダプタ２１６は、例えば、永久記憶デバイスにアクセスを提供するウルトラダイレクトメモリアクセス（ＵＤＭＡ）コントローラ、またはスモールコンピュータシステムインタフェース（ＳＣＳＩ）コントローラでありえる。Ｉ／Ｏアダプタ２２０は、例えばローカルエリアネットワーク（ＬＡＮ）またはインターネットのようなネットワークと通信を提供するよう構成されたイーサネットカード（イーサネットは登録商標である）でありえる。ＢＩＯＳ２０４は、フラッシュメモリのような任意の持続性メモリでありえる。

【００１７】
ある実施形態によれば、サービスプロセッサ２１２は、集積化されたチップセット機能を含むモトローラＭＰＣ８５５Ｔマイクロプロセッサである。相互接続コントローラ２３０は、ローカルポイントツーポイントコヒーレンスプロトコルをサポートする特定用途向け集積回路（ＡＳＩＣ）である。相互接続コントローラ２３０は、Ｉ／Ｏデバイスとの通信を可能にするために非コヒーレントプロトコルを扱うようにも構成されえる。ある実施形態において、相互接続コントローラ２３０は、プログラマブルロジックデバイスまたはフィールドプログラマブルゲートアレイのような特別に構成されたプログラマブルチップである。他の実施形態において、相互接続コントローラ２３０は、特定用途向け集積回路（ＡＳＩＣ）である。さらに他の実施形態において、相互接続コントローラ２３０は、相互接続パケットトラフィックにアクセスし処理する機能によって拡張された汎用プロセッサである。

【００１８】
図３は、相互接続コントローラ２３０の一例の概略図である。さまざまな実施形態によれば、相互接続コントローラは、マルチプロセッサシステムのさまざまなクラスタ中のプロセッサ群から受け取られたプローブおよびリクエストのようなパケットを扱うよう構成されたプロトコルエンジン３０５を含む。プロトコルエンジン３０５の機能は、パフォーマンスを改善するためにいくつかのエンジンにわたって分割されえる。ある例では、パーティショニングは、パケットタイプ（リクエスト、プローブおよびレスポンス）、方向（着信および発信）、またはトランザクションフロー（リクエストフロー、プローブフローなど）に基づいてなされえる。

【００１９】
プロトコルエンジン３０５は、最近のリクエストおよびプローブのようなトランザクションを相互接続コントローラが追跡し、これらトランザクションを特定のプロセッサ群と

関連付けることを可能にするペンディングバッファ３０９へのアクセスを有する。ペンディングバッファ３０９で維持されるトランザクション情報は、トランザクション宛先ノード、後で起こるコリジョン検出およびプロトコル最適化のためのリクエストのアドレス、レスポンス情報、タグ、および状態情報を含みえる。ヒストリバッファ３１１は、他のクラスタ中の相互接続コントローラによって成功して受け取られなかったパケットの効率的な再送信を可能にするためにも提供されえる。

【００２０】

相互接続コントローラは、外部プロセッサクラスタと共に、クラスタ中の他のプロセッサと相互接続コントローラが通信することを可能にするコヒーレントプロトコルインタフェース３０７を有する。相互接続コントローラは、Ｉ／Ｏデバイス（例えば図２においてリンク２０８ｃおよび２０８ｄによって表されるような）と通信するための非コヒーレントプロトコルインタフェース３１１のような他のインタフェースも含みえる。さまざまな実施形態によれば、それぞれのインタフェース３０７および３１１は、フルクロスバーとして、またはマルチプレクサおよびバッファのような要素を用いた別個の受信および送信ユニットとしてのいずれかで実現される。相互接続コントローラ２３０は、コヒーレントおよび非コヒーレントインタフェースの両方を提供する必要は必ずしもないことに注意されたい。またあるクラスタ内の相互接続コントローラ２３０は、他のクラスタ内の相互接続コントローラ２３０と通信しえることにも注意されたい。

【００２１】

本発明のさまざまな実施形態によれば、プロセッサ２０２ａ～２０２ｄは、実質的に同一である。図４は、複数のポート４０４ａ～４０４ｃを有するインタフェース４０２およびそれに関連付けられたルーティングテーブル４０６ａ～４０６ｃを含むそのようなプロセッサ２０２の簡略化されたブロック図である。それぞれのポート４０４は、例えば図２のリンク２０８ａ～２０８ｅのような関連するリンクを介してコンピュータシステム中の他のリソース、例えばプロセッサまたはＩ／Ｏデバイスと通信できる。

【００２２】

図４に示されるインタフェースは、さまざまなトポロジー、例えばリング、メッシュなどの任意のものによるシステムプロセッサを相互接続する複数のセグメントを備えるポイントツーポイント、分散化ルーティングメカニズムとして一般化されえる。これらセグメントのそれぞれのエンドポイントのそれぞれは、ユニークなノードＩＤおよびそれが「所有する」複数の関連付けられたリソース、例えばメモリおよびそれが接続されるＩ／Ｏを有する、接続されたプロセッサに関連付けられる。

【００２３】

分散ルーティングメカニズム中のノードのそれぞれに関連付けられたルーティングテーブルは、コンピュータシステムリソース内の相互接続の現在の状態を全体として表現する。任意の与えられたノード（例えばプロセッサ）によって所有されるリソース（例えば特定のメモリ範囲またはＩ／Ｏデバイス）のそれぞれは、ルーティングテーブル（群）中で、アドレスとしてそのノードと関連付けられる。リクエストがノードに到着すると、リクエストされたアドレスが、適切なノードおよびリンク、すなわちアドレスのある範囲内の与えられた特定のアドレスを特定するノードのルーティングテーブル中の２つのレベルのエントリと比較され、これらはノードｘに行け（go to node x）およびノードｘについてリンクｙを使え（for node x use link y）である。

【００２４】

図４に示されるように、プロセッサ２０２は、関連付けられたルーティングテーブル中の情報によって３つの他のプロセッサ群とポイントツーポイント通信を行いえる。具体的な実施形態によれば、ルーティングテーブル４０６ａ～４０６ｃは、２レベルテーブルを備え、第１レベルは、システムリソース（例えばメモリバンク）のユニークなアドレスを対応するノード（例えばプロセッサのうちの一つ）に関連付け、第２レベルは、現在のノードからそのノードへ行くのに用いられるように、それぞれのノードをリンク（例えば２０８ａ～２０８ｅ）に関連付ける。

【００２５】
プロセッサ２０２は、ＪＴＡＧハンドシェークレジスタ４０８のセットも有し、これはとりわけサービスプロセッサ（例えば図２のサービスプロセッサ２１２）およびプロセッサ２０２の間の通信を促進する。すなわち、サービスプロセッサは、ルーティングテーブル４０６ａ～４０６ｃでの最終的な記憶のために、ルーティングテーブルエントリをハンドシェークレジスタ４０８に書き込める。図４に図示されたプロセッサアーキテクチャは、本発明の具体的な実施形態を記載する目的のための単なる例示であることが理解されよう。例えば、本発明の他の実施形態を実現するために、より少ないまたはより多い数のポートおよび／またはルーティングテーブルが用いられえる。
【００２６】
上述のように、本発明の具体的な実施形態が基づく基本プロトコルは、特定の実現例によれば、３ビット空間である制限されたノードＩＤスペースを提供し、したがって８ノードのみユニークな特定が可能になる。すなわち、もしこの基本プロトコルが、本発明によって示される技術革新なしで採用されるなら、単一のクラスタ中ではポイントツーポイントインフラストラクチャを介して８ノードしか相互接続できない。この制限を回避するために、本発明は、特定のクラスタ内で単一レイヤ識別スキームを保持しつつ、一方で他の同様に配置されるクラスタおよび処理ノードとの相互接続およびそれらの間での通信を可能にする階層メカニズムを導入する。
【００２７】
具体的な実施形態によれば、それぞれのマルチプロセッサクラスタ内のノード群のうちの一つは、相互接続コントローラ、例えば図２の相互接続コントローラ２３０であり、これが情報の階層マッピングを管理し、それによってマルチクラスタが単一のアドレス空間を共有することを可能にし、一方で同時にそのクラスタ内のプロセッサが、それら自身のクラスタの外のものの「知識」なしに、任意のクラスタ内の任意のプロセッサと動作し、かつ対話することを可能にする。相互接続コントローラは、その関連付けられたプロセッサには、クラスタ内のプロセッサまたはノードのうちの単なる別の一つであるように見える。
【００２８】
基本プロトコルにおいて、クラスタ内の特定のプロセッサがリクエストを生成するとき、アドレスマッピングテーブルのセットは、そのリクエストをクラスタ内の他のノードのうちの一つにマッピングするために採用される。すなわち、クラスタ内のそれぞれのノードは、それが関連付けられる共有されたメモリ空間の一部を有する。異なるタイプのアドレスマッピングテーブルがメインメモリ、メモリマッピングされたＩ／Ｏ、異なるタイプのＩ／Ｏ空間などのために存在する。これらアドレスマッピングテーブルは、クラスタ中の特定のノードへのリクエスト中で特定されたアドレスをマッピングする。
【００２９】
それからルーティングテーブルのセットは、リクエストするノードからアドレスマッピングテーブルから特定されたノードへどのようにたどり着くかを決定するために採用される。すなわち、上述のように、それぞれのプロセッサ（すなわちクラスタノード）は、リクエストを現在のノードからアドレスマッピングテーブルから特定されたノードへ送信するために用いられえる、ポイントツーポイントインフラストラクチャ内の特定のリンクを特定するルーティングテーブルを関連付ける。一般に、ノードは、１つまたは複数のリソース（例えばプロセッサを含む）に対応しえるが、ノードおよびプロセッサという語はここではしばしば交換できるように用いられる。特定の実現例によれば、ノードは、複数のサブユニット、例えばＣＰＵ、メモリコントローラ、Ｉ／Ｏブリッジなどを備え、これらのそれぞれはユニットＩＤを有する。
【００３０】
加えて、個別のトランザクションは、非連続のパケット（non-consecutive packets）中にセグメント化されえるので、それぞれのパケットは、パケットがトランザクションを開始するノードに対して関連付けられるトランザクションを特定するために、ユニークな

トランザクションタグを含む。具体的な実現例によれば、トランザクションタグは、ソースノード（３ビットフィールド）、ソースノードユニット（２ビットフィールド）、およびトランザクションＩＤ（５ビットフィールド）を特定する。

【００３１】

よって、トランザクションが特定のノードにおいて開始されるとき、アドレスマッピングテーブルは、それからパケットにアペンドされ、ルーティングテーブルによってそのパケットをルーティングする適切なリンク（群）を特定するよう用いられる宛先ノード（およびユニット）を特定するために採用される。ソース情報は、リクエストに適切に応答するためにリクエストでプローブされる宛先ノードおよび任意の他のノードによって用いられる。

【００３２】

具体的な実施形態によれば、かつ上述のように、それぞれのクラスタ中の相互接続コントローラは、そのクラスタ中の他のプロセッサには、クラスタ中の他のプロセッサであるように単に見える。しかし、相互接続コントローラに関連付けられる共有されたメモリ空間の一部は、実際にはグローバルに共有されたメモリ空間の残り、すなわちシステム中の他のクラスタに関連付けられるメモリを含む。すなわち、特定のクラスタ中のローカルプロセッサの視点からは、システム中の他のマルチプロセッサクラスタの全てに関連付けられるメモリ空間は、それら自身のクラスタ中の相互接続コントローラ（群）によって表される。

【００３３】

図５を参照して説明されるさらにより具体的な実施形態によれば、それぞれのクラスタは、４つのプロセッサ２０２ａ～ｄおよび相互接続コントローラ２３０を含む５つのノード（例えば図２示されるように）を有し、それらのそれぞれは、クラスタ内でユニークな３ビットノードＩＤによって表される。上述のように、それぞれのプロセッサ（すなわちクラスタノード）は、例えばＣＰＵ、メモリコントローラなどを含む多くのサブユニットを表しえる。

【００３４】

本発明によって設計され、そのようなクラスタコンフィギュレーションを仮定する例示的アドレスマッピングスキームの図示は、図５に示される。図示された例において、グローバルメモリ空間は、ここではクワッドとも呼ばれる（４つのローカルプロセッサをそれぞれ含むので）４つのそのようなクラスタによって共有されるとも仮定される。理解されるように、それぞれのクラスタ内のクラスタおよびノードの個数は、異なる実施形態によって変わりえる。

【００３５】

アドレスマッピング機能を単一のクラスタを越えて拡張するために、それぞれのクラスタは、そのローカルメモリ空間、すなわちそのクラスタ中のプロセッサに関連付けられるグローバルメモリ空間の一部を、連続した領域内にマッピングし、一方、この領域の上および下のグローバルメモリ空間の残りの部分は、ローカル相互接続コントローラ（群）にマッピングされる。それぞれのクラスタ中の相互接続コントローラは、２つのマッピングテーブルを維持する。すなわち、グローバルマップおよびローカルマップである。ローカルマップは、リモートクラスタからローカルクラスタ内の特定のノードへの着信リクエストリクエストをマッピングする。

【００３６】

図５をここで参照して、それぞれのローカルクラスタは、ローカルメモリマップ（５０１～５０４）を有し、これらはローカルメモリ空間（すなわちローカルプロセッサに関連付けられるグローバルメモリ空間の連続部分）をそれぞれのノードにマッピングし、全てのリモートメモリ空間（すなわちグローバルメモリ空間の残り）を、ローカル相互接続コントローラ（群）、例えばクワッド３のノード４に関連付けられる１つまたは２つのマップエントリにマッピングする。ローカルクラスタのそれぞれのノードは、ローカルマップのコピーを有する。それぞれのクラスタ中の相互接続コントローラは、システム中の他の

クラスタのそれぞれと共にこれらリモートメモリ空間に関するグローバルマップ（５０５～５０８）も保持する。それぞれの相互接続コントローラは、リモートクラスタからそのクラスタ内の個々のノードへ受け取られるリクエストをマッピングするためにローカルマップ（５０９～５１１）のそのコピーを用いる。

【００３７】

図５を参照して説明される例としてのトランザクションは、例示的でありえる。この例では、クワッド３のノード２は、ローカル相互接続コントローラ（すなわちノード４）へマッピングする（マップ５０１を介して）リクエストを生成する。相互接続コントローラがこのリクエストを受け取るとき、そのグローバルマップ５０５は、アドレスをクワッド２にマッピングする。それから相互接続コントローラは、そのリクエストをクワッド２に転送する。クワッド２における相互接続コントローラは、そのローカルメモリマップを用いて、リクエストに対するターゲットへの適切なノード、この例ではノード１、を決定する。

【００３８】

具体的な実現例において、それぞれのプロセッサまたはクラスタノードは、８つのメモリマップレジスタに限定される。図５を参照して説明されるスキームは、ローカルメモリ空間について４つのエントリを、リモート空間について多くとも２つのレジスタを必要とする。したがって、具体的な実施形態によれば、２つの残りのエントリが領域を再分割するために用いられえる。８つのマッピングレジスタの制限は、クワッドにローカルな全てのメモリが連続なブロック内で割り当てられることを要求する。そのような実施形態における相互接続コントローラのローカルメモリも８つのエントリである。しかし相互接続コントローラのグローバルマップのサイズは、システム中のクラスタの個数によって決定される。さまざまな実施形態によれば、メモリマッピングされたＩ／Ｏ空間は、マッピングレジスタの同一のセットによってマッピングされる。

【００３９】

上述のように、ローカルクラスタレベルで、アドレスマッピングテーブルからの情報は、クラスタ内の宛先ノードへ情報を送信する適切なリンクを特定するために用いられる。上述のグローバルマッピングを用いてクラスタ間の送信を行うためには、同様のメカニズムが必要とされる。したがって、さまざまな実施形態によれば、クラスタ中のそれぞれのノードに関連付けられるローカルルーティングテーブルに加えて、相互接続コントローラは、システム中の他のクラスタを、クラスタを相互接続するさまざまなポイントツーポイント送信リンク（例えば図１Ａのリンク１１１）にマッピングするグローバルルーティング情報を保持する。

【００４０】

本発明の具体的な実施形態によれば、２つのタイプのローカルルーティングテーブルが採用される。一つはディレクテッドパケットのためであり、一つはブロードキャストパケットのためである。それぞれのテーブル（例えば図４のテーブル４０６）は、ターゲットノードおよびリンク間のマッピングを保持する。ディレクテッドパケットについて、別個のテーブルがリクエストについておよび応答について用いられる。これは、応答がリクエスタにリクエストと同じパスを通って元に戻ることを可能にする。同じルートを維持することは、デバッギングを容易にし、正確さのためには要求されない。ブロードキャストパケットについては、対応するテーブルは、どのリンクにブロードキャストパケットが転送されるかを示す。ブロードキャストパケットは、よって複数のリンクへとルーティングされえる。

【００４１】

本発明の相互接続コントローラの特定の実現例において、そのローカルテーブルは、ローカル宛先ノードをディレクテッドパケットについては４つのリンクのうちの１つにマッピングし、ブロードキャストパケットについては任意の個数のリンクにマッピングする。相互接続コントローラはまた、リモート宛先クラスタを特定のリモートリンクにマッピングするグローバルルーティングテーブルを維持する。特定の実施形態によれば、相互接続

コントローラは、グローバルルーティングレベルにおいてパケットのマルチキャストもサポートする。

【００４２】

本発明によって設計されたルーティングメカニズムの具体的な実施形態が図６Ａおよび６Ｂを参照してこれから説明される。図６Ａのシステム６００は、ノードN_0およびN_1を含む複数のローカルノードをそれぞれ有する４つのクラスタを含む。図６Ｂのテーブルは、例示的目的のためにシステムの全てのローカルおよびグローバルルーティングテーブルを結合する。

【００４３】

例示的トランザクションの一部として、クラスタ０中のノードN_0におけるＣＰＵ６０２は、クラスタ３中のノードN_0におけるＣＰＵ６０４に向けられたパケットを生成する。このパケットは、例えば、そのノードにおけるメモリコントローラにマッピングするメモリリクエストでありえる。ＣＰＵ６０２が、そのクラスタの外の何ものについても知識を有しないので、それは宛先としてクラスタ０中のノードN_1（すなわちローカル相互接続コントローラ６０６）にターゲットするパケットを生成する。上述のように、これは、ノードN_0によって所有されるローカルメモリマップ（図６Ｂのテーブルの関連部分を参照）が、ノードN_1をリモートクラスタによって所有された全てのメモリに対応するものとして特定するという事実による。相互接続コントローラ６０６は、パケットを受け取り、そのグローバルアドレスマップを用いて（例えば上述のもの）、パケットの最終宛先がクラスタ３であることを決定し、クラスタ３をターゲットするリモートパケットを生成する。それから、そのグローバルルーティングテーブル（図６Ｂのテーブルの関連部分を参照）を用いて、相互接続コントローラ６０６は、このパケットがリンクL_1上に送られなければならないことを決定する。上述のローカルルーティングメカニズムに同様に、ソースおよび宛先クラスタを特定する情報がパケットにアペンドされる。

【００４４】

クラスタ１における相互接続コントローラ６０８がパケットを受け取ると、それはパケットがクラスタ３に宛先が設定されていることも決定し、そのグローバルルーティングテーブル（図６Ｂ）からリンクL_2がそのパケットを送るために用いられなければならないことを決定する。クラスタ３における相互接続コントローラ６１０は、パケットを受け取り、パケットがローカルクラスタをターゲットとしていることを決定して、そのローカルルーティングテーブル（図６Ｂ）を用いて、そのパケットをその宛先に送るためにはローカルリンクL_0が用いられなければならないことを決定する。それからノードN_0におけるＣＰＵ６０４は、パケットをリンクL_0を介して受け取る。ノードＩＤ空間が３ビットＩＤ空間である具体的な実施形態によれば、このマルチレベルルーティングメカニズムは、クラスタの数に具体的な制限がない８つのローカルノードに拡張されえる。

【００４５】

マルチレベルルーティングメカニズムを有することは、システムが比較的束縛されない個数のプロセッサおよび処理クラスタを有することを可能にする。しかし、システムサイズおよび複雑さが増すと共に、システムパフォーマンスを低下させることなくプロセッサおよび処理クラスタを動的に追加および削除することが急激に重要になる。より多くの個数のプロセッサおよび処理クラスタを持つさらなるコンフィギュレーションも可能であり、シームレスにコンフィギュレーション変更を可能にすることが有利になる。コンフィギュレーション変更の一つのタイプは、プロセッサ、処理クラスタ、リンクなどのような欠陥のある要素の除去または動的置換である。

【００４６】

従来のメカニズムによってリンクを開いたり閉じたりすることは制限がある。既存のメカニズムは、例えば、リセット中に見つからないリンクを動的に開いたり、リセット中に見つかったリンクを閉じたりすることを提供できない。したがって、ユーザが動的にリンクをイネーブルまたはディセーブルすることを可能にし、マルチプロセッサ、マルチクラスタシステムのコンフィギュレーションを変更する改良点を提供することが望ましい。あ

る例では、ユーザは、既存のシステムにプロセッサおよび処理クラスタのようなリソースを動的に追加できる。他の例では、ユーザは、クラスタ間の相互接続コントローラまたはハイパートランスポートケーブルのような欠陥のある要素を置換できる。

【００４７】

本発明のさまざまな実施形態によれば、相互接続コントローラがリンクの動的なイネーブリングおよびディセーブリングを可能にするメカニズムが提供される。ある実施形態において、コンフィギュレーションスペースレジスタは、相互接続コントローラと関連付けられる。コンフィギュレーションスペースレジスタは、物理レイヤおよびリンクレイヤ通信をイネーブリングする情報、リンクを再初期化する指示子、および相互接続コントローラに結合された処理クラスタをトラッキングする識別子を含む。

【００４８】

図７は、コンフィギュレーションスペースレジスタを示す概略図である。コンフィギュレーションスペースレジスタ７０１は、プロセッサおよびクラスタ間の物理およびリンクレイヤ通信をイネーブルおよびディセーブルするために提供される。コンフィギュレーションスペースレジスタ７０１は、相互クラスタ通信を管理する相互接続コントローラ中に含まれえる。コンフィギュレーションスペースレジスタ７０１は、他のリソース中に含まれることに注意されたい。物理レイヤおよびリンクレイヤトグル指示子と共に複数のプロセッサクラスタＩＤを維持するメカニズムは、ここではコンフィギュレーションスペースレジスタと呼ばれる。ある例では、コンフィギュレーションスペースレジスタ７０１は、物理レイヤ通信をイネーブルおよびディセーブルするための物理レイヤ指示子７１１を含む。コンフィギュレーションスペースレジスタへワイヤリングすることによって物理レイヤ通信をディセーブルすることは、プロセッサおよびクラスタ間のリンクをシャットダウンする。ある例では、物理マクロがオフにされ、ディセーブルされた物理マクロから発する電気的アクティビティは存在しなくなる。ポリングは行われず、関連付けられる相互接続コントローラは、ポリングに応答せず、リンクを維持しない。さまざまな実施形態によれば、コンフィギュレーションスペースレジスタ７０１は、リンクレイヤ指示子７１３も含む。ある例では、リンクレイヤ指示子７１３は、フェンスビットである。コンフィギュレーションスペースレジスタへのワイヤリングによってリンクレイヤをディセーブルすることは、リンクが物理レイヤ通信以外の任意の通信のために用いられることを防ぐ。

【００４９】

しかし、相互接続コントローラは、ポリングに依然として応答し、既存の接続を維持する。物理レイヤ通信以外のものは何も許されないので、ハイパートランポートコマンドまたはデータパケットは、ディセーブルされたリンクレイヤを持つリンクにわたっては送られえない。ポリング状態は、コンフィギュレーションスペースレジスタを用いて維持されえる。ある例では、システムがリセットから出てくるときにもしリンクが見つからないなら、物理レイヤプロトコルは、何かのデバイスが取り付けられていないかを決定するためにリンクを連続的にポリングする能力を有する。このポリングの特徴は、コンフィギュレーションスペースレジスタ中の適切なトグルを設定することによってイネーブルまたはディセーブルされえる。ポリングすることは、自動認識および通知によってリソースの既存のクラスタへのホットプラギングを可能にする。ある例では、ポリングは、新しいマルチ処理クラスタの既存のマルチ処理クラスタへのホットプラギングを可能にする。

【００５０】

システムコンフィギュレーションを維持するプロセッサは、新しいクラスタが取り付けられるときにリンクをイネーブルし、システムを拡張するか、または単にユーザに最近なされた接続がシステムによって認識されたことを通知するよう選択しえる。さまざまな実施形態によれば、コンフィギュレーションスペースレジスタは、新しい初期化シーケンスが実行されるべきかを示すための再初期化指示子７１５も含む。コンフィギュレーションスペースレジスタ７０１が接続されたクラスタのＩＤを維持することを可能にするために、クラスタ識別子７１７、７１９、および７２１を保持するレジスタも提供される。コンフィギュレーションスペースレジスタ７０１中の全ての値が必要なわけではないことに注

意されたい。例えば、２つのクラスタしかサポートしないシステムにおいては、クラスタＩＤを維持するためには単一のレジスタしか提供されない。同じように、コンフィギュレーションスペースレジスタ７０１は、特定のポリング状態情報のような他の情報も維持しえる。

【００５１】

図８は、システムのリセット中にコンフィギュレーションスペースレジスタを用いるある技術を示すフロープロセス図である。８０１において、物理レイヤ通信がリセットからイネーブルされる。物理レイヤをイネーブルすると同時に標準物理レイヤ初期化シーケンスが実行されえる。ポリング状態も維持されえる。ある例では、ポリング状態がアクティブに設定され、相互接続コントローラは、他のリソースの存在のためにリンクをモニタするだけでなく、ビットのシーケンスをアクティブに送る。他の例では、ポリング状態がパッシブに設定され、相互接続コントローラは単に他のリソースの存在だけをモニタする。８０３において、フェンスビットは、リンクレイヤ通信をディセーブルするように設定される。リンクレイヤ通信をディセーブルすることは、データローカル送信からの干渉なしに、物理レイヤ通信が確立されることを可能にする。しかし、リンクレイヤ通信は必ずしもディセーブルされる必要はないことを注意されたい。８０５において、初期化シーケンスまたはトレーニングシーケンスが送信される。

【００５２】

送信は、シリアライゼーションおよびデシリアライゼーション（ＳＥＲＤＥＳ）メカニズムおよび８ｂ／１０ｂエンコーディングを用いて典型的には実行される。８ｂ／１０ｂエンコーディングは、１または０の連続したシーケンスを混合されたシーケンスで置換し、さまざまな理由のために用いられる。８ｂ／１０ｂは、正しいＰＬＬ動作を確実にするために送信されたデータ中に充分な信号遷移を発生するのを助ける。この８ｂ／１０ｂエンコーディングスキームなしでは、データ中の１または０のストリングが、クロックがドリフトまたは同期を失い、データの損失をきたすことを起こしえる。８ｂ／１０ｂは、信号がＤＣ的にバランスがとれていることも確実にし、これはＤＣオフセットが時間と共にリンク内で大きくなっていかないことを意味する。８ｂ／１０ｂエンコードされた文字は、多くの信号の誤りをすぐに検出する特定の規則に従う。

【００５３】

用いられえる初期化またはトレーニングシーケンスの例は、Infiniband Trade Associationから入手可能なInfiniband Architecture Specification Volumes I and IIにおいて記載され、その全体が全ての目的のために参照によって援用される。Infinibandは必ずしも用いられなければならないのではないことに注意されたい。物理レイヤ通信の初期化のあいだ、情報は相互接続コントローラ群間で交換される。８１１において、リンク幅パラメータが交換される。８１３において、リンクスピードパラメータが交換される。８１５において、オプションの誤り訂正情報が交換される。８１７において、フェンスビットをトグルオフすることによってリンクレイヤがそれからイネーブルされる。リンクレイヤがイネーブルされると、データ通信が進められる。８１９において、クラスタＩＤ情報が交換される。８２１において、クラスタＩＤがコンフィギュレーションスペースレジスタにおいて設定される。８２３において、接続されたリソースを反映するように相互接続コントローラに関連付けられるルーティングテーブルが必要に応じてアップデートされる。

【００５４】

他の例において、複数のクラスタにわたってのリセットの同期化については仮定はなされていない。システムリセット信号の任意の時間的順序のアサーションおよびデアサーションが有効である。ある例では、それぞれのクラスタ中のそれぞれのリモートリンクレイヤは、リセットのデアサーションの後、フェンスビットがクリアにされて立ち上がる。それぞれのリモートリンクは、ディセーブルされて立ち上がり、リモート物理レイヤは、接続の他端上に他のクラスタがないかポリングする。リモート接続の両端上のクラスタがリセットから出た後、リモート物理レイヤはポーリングと共に互いを検出する。さまざまな実施形態によれば、リモート物理レイヤは、物理リンク接続の信頼性のある確立のために

必要とされる任意の他の情報と共に、リンク幅パラメータ、リンクスピードパラメータ、誤り訂正情報を交換することを伴う初期化／トレーニングシーケンスをここで実行しえる。

【００５５】

初期化／トレーニングシーケンスの後、クラスタＩＤフィールドが交換されえる。クラスタはリセットから出てきたばかりなので、クラスタＩＤは、所定の固定値に、例えば全てゼロに初期化されえる。それからリモートクラスタＩＤは、ＣＳＲ中に維持される。物理レイヤはイネーブルされ、リンクレイヤはまだディセーブルされている。この時点で、サービスプロセッサは、主クラスタ中のリモートリンクのＣＳＲを読み、クラスタＩＤを割り当て、かつリモート接続の物理レイヤを再初期化するためにＣＳＲに書き込むことによって、主クラスタから始まってシステム全体の探索をしている。

【００５６】

さまざまな実施形態によれば、主クラスタ中のサービスプロセッサは、どのクラスタがどの他のクラスタにそれぞれのクラスタ上のリモートリンクを通して接続されているかについての情報をここで得る。この情報は、ルーティングテーブルを生成するのに用いられる。この情報は、ＢＩＯＳへも渡される。ＢＩＯＳは、クラスタのうちの一つの中のＣＳＲ中の「リンクレイヤを初期化」ビットをセットする。これは、リンクレイヤスタートパケットで片方のリンクレイヤがリンクレイヤ初期化を開始するようにする。他端のリンクレイヤがリンクレイヤスタートパケットを受け取るとき、それはリンクレイヤスタートパケットで応答し、リンクレイヤが確立される。

【００５７】

リンクレイヤが立ち上がると、リモートクラスタ中のプロセッサおよびコントローラをプログラム／初期化するためにパケットがリモートリンクをわたって主クラスタから送られえる。これでシステムが動作している。ある例では、もし任意のリモート接続が用いられないなら、サービスプロセッサまたはＢＩＯＳは、これらリモート接続についてリンクレイヤを分離し、物理レイヤをディセーブルすることを選びえる。

【００５８】

図９は、ホットプラギングのあいだのスペースレジスタの使用を示すフロープロセス図である。システムがまだ動作しているあいだのリソースのシステムへの任意の動的付加は、ここではホットプラギングと呼ばれる。９０１において、ポリング状態が相互接続コントローラにおいて維持される。９０３において、そのリンクレイヤをディセーブルするためにフェンスビットが維持される。しかしフェンスビットは必ずしもディセーブルされる必要はないことに注意されたい。新しい処理クラスタが検出されるとき、再初期化が９０５においてトリガされる。ある例では、再初期化は、コンフィギュレーションスペースレジスタの再初期化指示子であるビットに書き込むことによってトリガされる。９０７において、初期化シーケンスまたはトレーニングシーケンスが送られる。９１１において、リンク幅およびリンクスピードのような情報が交換される。９１３においてそれからリンクレイヤがイネーブルされる。９１５において、プロセッサの新しく追加されたグループのクラスタＩＤがコンフィギュレーションスペースレジスタに書き込まれる。９１７において上述のルーティングテーブルは、複数のクラスタの通信を可能にするために同様にアップデートされる。

【００５９】

他の例において、１つ以上のクラスタが動作し、新しいクラスタがシステムに追加される必要がある。さまざまな実施形態によれば、追加される必要がある新しいクラスタ上でリセットがアサートおよびディアサートされる。もし新しいクラスタへのリモート接続についての物理レイヤがディセーブルされるなら、物理レイヤはポリング状態に置かれる。リモート接続は分離されない（unfenced）。クラスタＩＤが交換され、物理レイヤはイネーブルされる。サービスプロセッサは、上述のように同様のやり方で新しいクラスタおよび主クラスタをもプログラムするために必要とされる情報を生成する。新しいリモート接続のリンクレイヤがここで初期化される。

【００６０】
図１０は、システムからリソースを動的に除去する技術を示すフロープロセス図である。システムがアクティブであるときに、プロセッサまたは処理クラスタのようなリソースを動的にアンプラグすることは一般的に難しいが、これはプロセッサおよびそれらに関連付けられたキャッシュを除去することがシステム動作を非常に大きく左右するからである。ある例では、特定のアプリケーションによって処理されているデータは、システムから除去されるためにプロセッサキャッシュ中において中間状態で保持されえる。同様に、オペレーティングシステムプロセスは、除去が設定された１つ以上のプロセッサ上で走っているかもしれない。さまざまな実施形態によれば、クラスタの動的除去をサポートするオペレーティングシステムが用いられる。ある例では、アプリケーションは、マルチプロセッサクラスタの動的除去をサポートするオペレーティングシステム環境中で１００１において終了される。クラスタが１００３においてディセーブルされるが、まだアンプラグされない。１００５において、除去されるべきそのクラスタに関連付けられるキャッシュがフラッシュされる。１００７において、処理クラスタの除去を反映するようにルーティングテーブルが変更される。１００９において、フェンスビットがコンフィギュレーションスペースレジスタ中に書き込まれる。フェンスビットは、さまざまなエンティティによって書き込まれえる。ある例では、フェンスビットは、サービスプロセッサによって、またはプロセッサに関連付けられるＪＴＡＧインタフェースによって書き込まれる。１０１１において、マルチプロセッサクラスタが物理に除去されえる。１０１３において、プロセッサの新しいまたは置換クラスタが導入されることを可能にするために、まだシステム中に存在する相互接続コントローラにおける物理レイヤが維持される。

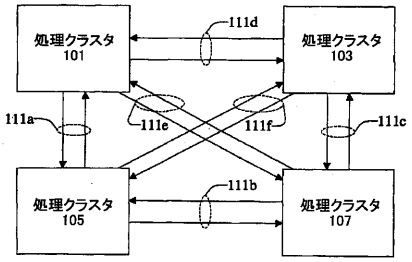
【００６１】
本発明は、その具体的な実施形態を参照して示され記述されてきたが、記載された実施形態の形態および詳細の変更が本発明の精神または範囲から逸脱することなくなされえることが当業者には理解されよう。例えば本発明の実施形態は、ポイントツーポイント、スイッチ、またはバスアーキテクチャを通して接続されたマルチプロセッサクラスタと共に採用されえる。他の例では、プロセッサのマルチクラスタは、単一の相互接続コントローラを共有しえ、または複数の相互接続コントローラが単一のクラスタ中で用いられえる。従って本発明の範囲は添付の特許請求の範囲を参照して決定されるべきである。

【図面の簡単な説明】

【００６２】
【図１Ａ】複数のクラスタを有するシステムを示す概略図である。
【図１Ｂ】複数のクラスタを有するシステムを示す概略図である。
【図２】複数のプロセッサを有する例示的クラスタを示す概略図である。
【図３】本発明のさまざまな実施形態を利用する例示的相互接続コントローラの概略図である。
【図４】ローカルプロセッサの概略図である。
【図５】本発明の特定の実施形態によるメモリマッピングスキームの概略図である。
【図６Ａ】本発明の特定の実施形態を示す４つのクラスタシステムの簡略化されたブロック図である。
【図６Ｂ】図６Ａの４つのクラスタシステムのためのルーティング情報を含む結合されたルーティングテーブルの図である。
【図７】相互接続コントローラに関連付けられるコンフィギュレーションスペースレジスタを示す概略図である。
【図８】システムのリスタートを示すフロープロセス図である。
【図９】新しいリンクの動的導入を示すフロープロセス図である。
【図１０】既存リンクの動的除去を示すフロープロセス図である。

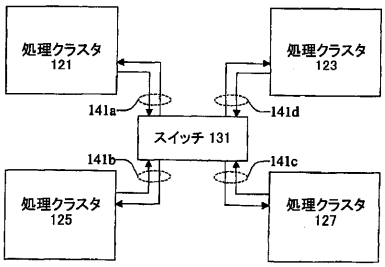
【図１Ａ】

Fig. 1A



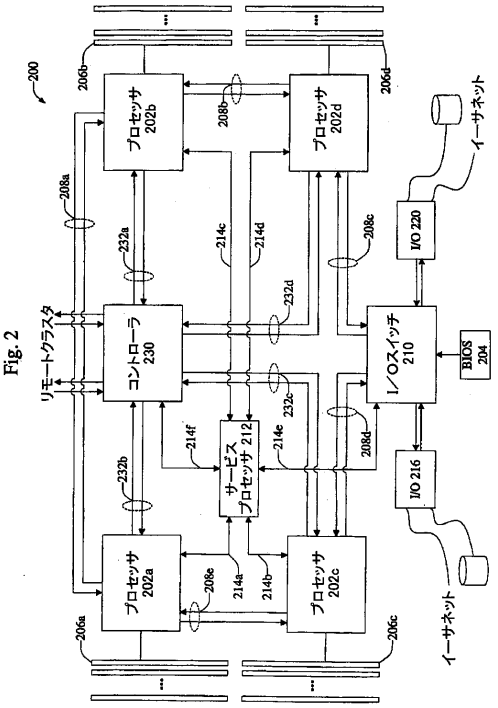
【図１Ｂ】

Fig. 1B



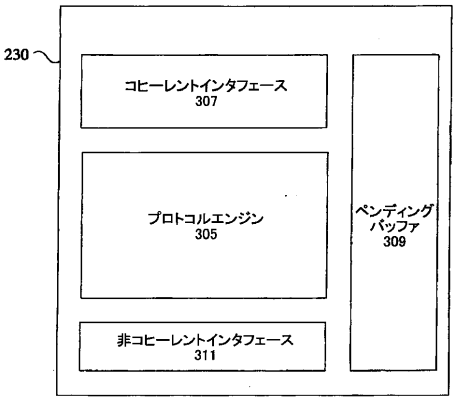
【図２】

Fig. 2



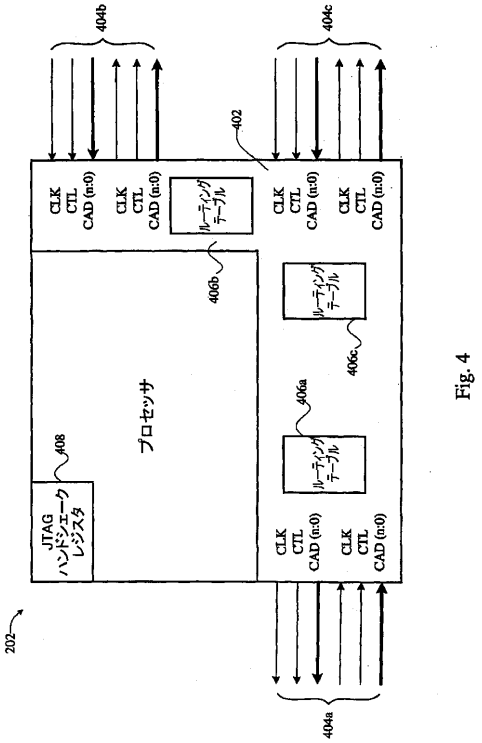
【図３】

Fig. 3



【図４】

Fig. 4



【図５】

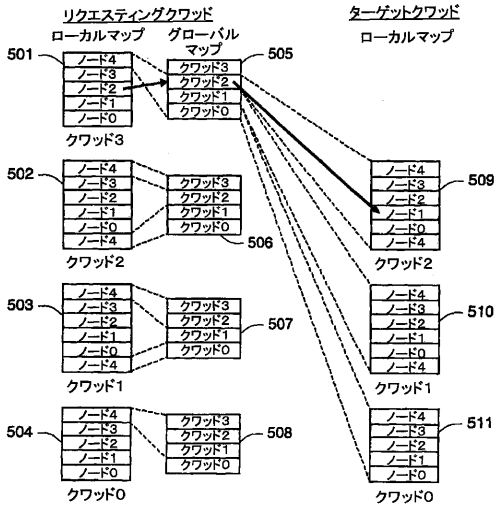


Fig. 5

【図６Ａ】

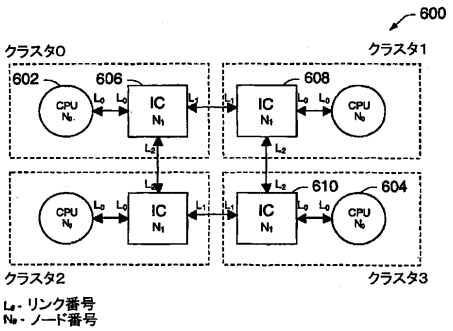


Fig. 6A

【図６Ｂ】

		ローカルテーブル		グローバルテーブル			
		宛先ノード		宛先クラスタ			
ソース		N_0	N_1	C_0	C_1	C_2	C_3
クラスタ0	ノード0	X	L_0	NA	NA	NA	NA
	ノード1	L_0	X	X	L_1	L_2	L_1
クラスタ1	ノード0	X	L_0	NA	NA	NA	NA
	ノード1	L_0	X	L_1	X	L_2	L_2
クラスタ2	ノード0	X	L_0	NA	NA	NA	NA
	ノード1	L_0	X	L_2	L_2	X	L_1
クラスタ3	ノード0	X	L_0	NA	NA	NA	NA
	ノード1	L_0	X	L_2	L_2	L_1	X

Fig. 6B

【図７】

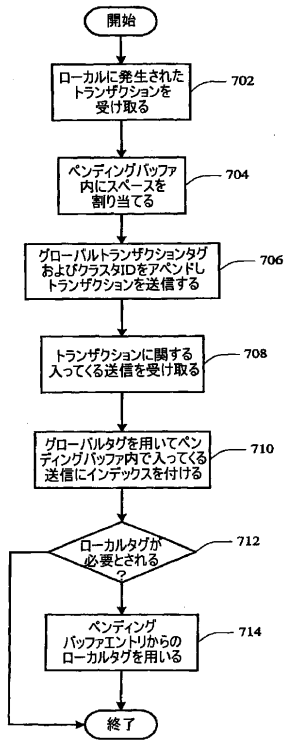


Fig. 7

【図８】

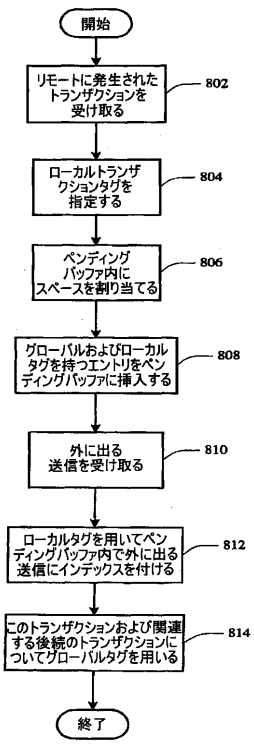
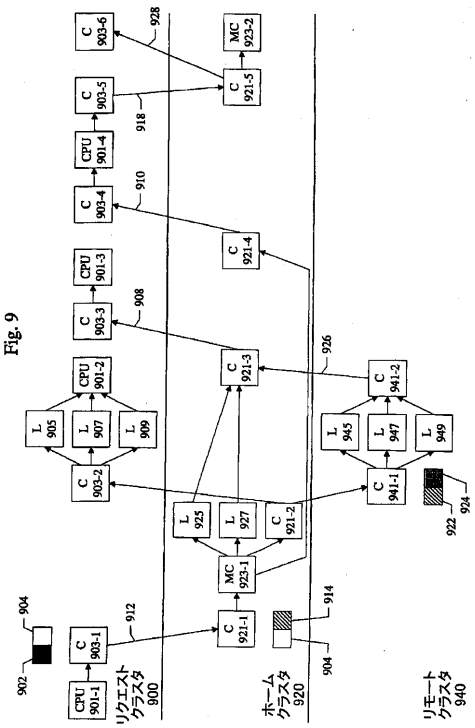


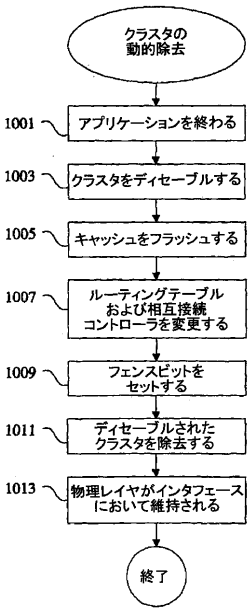
Fig. 8

【図９】



【図１０】

Figure 10



フロントページの続き

(81)指定国 AP(BW,GH,GM,KE,LS,MW,MZ,NA,SD,SL,SZ,TZ,UG,ZM,ZW),EA(AM,AZ,BY,KG,KZ,MD,RU,TJ,TM),EP(AT,BE,BG,CH,CY,CZ,DE,DK,EE,ES,FI,FR,GB,GR,HU,IE,IT,LU,MC,NL,PL,PT,RO,SE,SI,SK,TR),OA(BF,BJ,CF,CG,CI,CM,GA,GN,GQ,GW,ML,MR,NE,SN,TD,TG),AE,AG,AL,AM,AT,AU,AZ,BA,BB,BG,BR,BW,BY,BZ,CA,CH,CN,CO,CR,CU,CZ,DE,DK,DM,DZ,EC,EE,EG,ES,FI,GB,GD,GE,GH,GM,HR,HU,ID,IL,IN,IS,JP,KE,KG,KP,KR,KZ,LC,LK,LR,LS,LT,LU,LV,MA,MD,MG,MK,MN,MW,MX,MZ,NA,NI,NO,NZ,OM,PG,PH,PL,PT,RO,RU,SC,SD,SE,SG,SK,SL,SY,TJ,TM,TN,TR,TT,TZ,UA,UG,US,UZ,VC,VN,YU,ZA,ZM,ZW

(72)発明者 ネワワーカー・シャシャンク
アメリカ合衆国 テキサス州７８７１７ オースティン，コルデロ・ドライブ，１５００８
(72)発明者 プラサドゥ・グル
アメリカ合衆国 テキサス州７８７５０ オースティン，カラニッシュ・パーク・ドライブ，１１１１１
(72)発明者 ツァイトラー・カール
アメリカ合衆国 テキサス州７７３７５ トムボール，ブラッシュ・キャニオン・ドライブ，１１８３５
(72)発明者 グラスコ・デイビッド・ビー.
アメリカ合衆国 テキサス州７８７２６ オースティン，エンバー・グレン・ドライブ，１０３３７

Ｆターム(参考) 5B045 BB02 BB15 BB28 DD01 HH05 HH06 JJ38