(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第3884073号
(P3884073)
(45)発行日　平成19年2月21日(2007.2.21)　(24)登録日　平成18年11月24日(2006.11.24)

(51)Int.Cl.　　F I
G06F 13/10 (2006.01)　　G06F 13/10 330A
G06F 13/38 (2006.01)　　G06F 13/38 310A

請求項の数 1　(全 24 頁)

(21)出願番号　特願平9-503399
(86)(22)出願日　平成8年6月17日(1996.6.17)
(65)公表番号　特表平11-514113
(43)公表日　平成11年11月30日(1999.11.30)
(86)国際出願番号　PCT/US1996/010466
(87)国際公開番号　WO1997/000533
(87)国際公開日　平成9年1月3日(1997.1.3)
審査請求日　平成15年5月21日(2003.5.21)
(31)優先権主張番号　08/490,651
(32)優先日　平成7年6月15日(1995.6.15)
(33)優先権主張国　米国(US)

(73)特許権者
インテル・コーポレーション
アメリカ合衆国　９５０５２　カリフォルニア州・サンタ　クララ・ミッション　カレッジ　ブーレバード・２２００
(74)代理人
弁理士　山川　政樹
(74)代理人
弁理士　黒川　弘朗
(74)代理人
弁理士　紺野　正幸
(74)代理人
弁理士　西山　修
(74)代理人
弁理士　鈴木　二郎

最終頁に続く

(54)【発明の名称】複数プロセッサ・システム内のプロセッサ間でメッセージを送信する装置

(57)【特許請求の範囲】
【請求項１】
ローカル・バスを介してローカル・プロセッサとメモリとに結合され、かつ第２のバスを介してバス・エージェントに結合されたメッセージ通信装置において：
ａ）インバウンド・フリー待ち行列であって、前記ローカル・プロセッサによりこの待ち行列の頭部ポインタによって示された該待ち行列の頭部から書き込まれ、かつ前記バス・エージェントによって使用されるために該待ち行列の尾部ポインタによって示された該待ち行列の尾部から読み出される空のメッセージ・バッファのハンドル情報であってバス・エージェントからローカル・プロセッサにメッセージを渡すために、該バス・エージェントによって使用され、かつ前記ローカル・プロセッサによって解放されるハンドル情報を記憶するインバウンド・フリー待ち行列と；
ｂ）前記インバウンド・フリー待ち行列内のハンドル情報を利用するためにこのインバウンド・フリー待ち行列に結合されたインバウンド・フリー回路手段であって、
前記インバウンド・フリー待ち行列の頭部ポインタと尾部ポインタとに基づき、前記インバウンド・フリー待ち行列が「空」であるか否かを判定する判定手段と、
前記インバウンド・フリー待ち行列の尾部から読み出したハンドル情報を保持するインバウンド・フリー・レジスタと
を備え、前記インバウンド・フリー回路手段によるハンドル情報に対する操作が、
（１）前記インバウンド・フリー待ち行列が空でない場合には、該インバウンド・フリー待ち行列の尾部からハンドル情報をプリフェッチして前記インバウンド・フリー・レジス

タに格納し、
（２）前記インバウンド・フリー待ち行列が空の場合には、該インバウンド・フリー・レジスタに該インバウンド・フリー待ち行列が空であることを示す「空」指示情報をロードし、かつ
（３）一回のバス・トランザクションで、前記バス・エージェントが前記インバウンド・フリー・レジスタ内の情報を読み取ることを可能にするインバウンド・フリー回路手段と；
ｃ）インバウンド・ポスト待ち行列であって、前記バス・エージェントから読み出された後でこの待ち行列の頭部ポインタによって示された該待ち行列の頭部から書き込まれ、かつ前記ローカル・プロセッサによって使用されるために該待ち行列の尾部ポインタによって示された該待ち行列の尾部から読み出される、前記バス・エージェントからローカル・プロセッサに渡されるメッセージのハンドル情報であって前記インバウンド・フリー待ち行列から先に読み出されたハンドル情報を記憶するインバウンド・ポスト待ち行列と；
ｄ）前記インバウンド・ポスト待ち行列内のハンドル情報を利用するためにこのインバウンド・ポスト待ち行列に結合されたインバウンド・ポスト回路手段であって、
前記インバウンド・ポスト待ち行列の頭部ポインタと尾部ポインタとに基づき、前記インバウンド・ポスト待ち行列が「フル」であるか否かを判定する判定手段と、
前記バス・エージェントからの書き込むべきメッセージのハンドル情報を保持するインバウンド・ポスト・レジスタと
を備え、前記インバウンド・ポスト回路手段によるハンドル情報に対する操作が、
（１）前記インバウンド・ポスト待ち行列がフルである場合には、再試行信号を前記バス・エージェントに返し、
（２）前記インバウンド・ポスト待ち行列がフルでない場合には、前記インバウンド・ポスト・レジスタに保持されたハンドル情報を前記インバウンド・ポスト待ち行列の頭部に書込み、さらに
（３）一回のバス・トランザクションで、前記バス・エージェントが前記インバウンド・ポスト・レジスタにハンドル情報を書き込むことを可能にするインバウンド・ポスト回路手段と
を備えたことを特徴とするメッセージ通信装置。
【発明の詳細な説明】
発明の背景
１．発明の分野
本発明は、複数プロセッサ・システム内の分野に関する。さらに詳細には、本発明は、複数プロセッサ・システム内のプロセッサ間でメッセージを送信する方法および装置に関する。
２．従来技術の説明
メッセージは、単にオペレーショナル・パラメータおよびデータを伝達するデータ構造である。メッセージは、１つまたは複数のプラットフォーム上で実行されている１つまたは複数のプロセス（すなわち、アプリケーション）によって生成される。プラットフォームは、メモリに関連するプロセッサまたはプロセッサのクラスタ、ローカル・メモリ・バス、およびメモリ入出力バスを含んでいる。プラットフォーム内のこれらの要素は、動作環境を構成する。
さらに、プラットフォームは、オペレーティング・システムの単一のインスタンスを実行する。言い換えれば、コンピュータ・システムは、単一のオペレーティング・システムが複数のプロセッサをサポートする分配処理システムである。メッセージは、ある特定のプラットフォーム上のプロセスの１つによって生成された後、処理のために他のプロセッサ・プラットフォームに送られる。
メッセージは、メモリ内に常駐し、命令および追加の情報のデータ・ブロックに対する他のポインタを含んでいる制御ブロックにポインタによって示される。例えば、制御ブロックは、特定の周辺装置（例えば、ハード・ディスク・ドライブ）を指定し、装置の指定さ

れたセクタからデータを読み取るよう要求する。
メッセージ送信は、プロセッサが「密に」結合された（すなわち、プロセッサが単一のキャッシュを共用する）対称複数プロセッサ・システム（ＳＭＰ）内、およびプロセッサが共通のバス構造によって「粗に」結合された非対称複数プロセッサ・システム内のプロセッサ間で使用される。
メッセージを第１のプラットフォーム内のあるプロセッサから第２のプラットフォーム内の第２のプロセッサに送るとき、そのメッセージを向けられたプロセッサが自身の資源が自由なときにメッセージを処理することができるようにメッセージを待ち行列化しなければならない。
メッセージを待ち行列化する従来技術の方法は、主としてソフトウェア技法を使用して実施される。これらの方法は、共用された待ち行列構造への複数のアトミック・プロセッサ・アクセスを必要とする。例えば、単一のプロセッサ上で動作している複数のプロセスは、プロセッサによって共用されたメモリ内に配置されたメッセージの１つの待ち行列を共用する。プロセスの１つに対してアトミック・アクセスを行うために、オペレーティング・システムは、待ち行列へのアクセスを要求しているプロセスに、待ち行列に対するそのプロセス独占権（例えば、アトミック・アクセス）を与えるセマフォを付与する。セマフォは、単に共用されたデータ構造（すなわち、オペレーティング・システム・コンテキストの一部）に対するプロセス独占アクセス権を与えるオペレーティング・システム変数である。その場合、プロセスは、待ち行列に対してメッセージを追加したり、削除することができる。特定のプロセスは、セマフォを制御するとき、その待ち行列に対してアクセスを要求している他のプロセスをロックアウトする。他のプロセスは、共用された構造が使用できるようになるまで、第１のプロセスがセマフォを解放するのを待たなければならない。
複数プロセッサ・システムでは、複数のプロセッサがセマフォへのアクセス権を同時に得ようと試みることができる。したがって、同期のためにバス・ロック（すなわち、アトミック・アクセス）が必要となる。１つのプロセッサがバスをロックしている間、他のプロセッサは、第１のプロセッサがそのバスをロック解除するまでメモリ内の同じ共用された構造（すなわち、メモリ・ブロック）にアクセスすることはできない。セマフォはシステム・メモリ内にあるので、他のプロセッサは、セマフォを求めて争っていなくてもロックアウトされる。したがって、サスペンドすることができるソフトウェア・モジュール（すなわち、マルチタスク・オペレーティング・システム）内では、バス・ロックは使用できない。代わりに、これらのアプリケーション内でセマフォを獲得し、解放するとき、オペレーティング・システム・カーネルへのコールが必要となる。
上述の動作は、セマフォを待っている間またはバス・アクセスを待っている間、各プロセスがアイドルである時間のために非常に非効率的である。さらに、オペレーティング・システム・カーネルへの上述のコールは、高価なコンテキスト切換えを行う。
コンテキストは、単にアプリケーション（すなわち、アプリケーション・コードおよびデータ）に充てられるメモリ領域である。アプリケーション・コンテキストは、フラグ、変数、現在プロセスの状態を含んでいる。セマフォはアプリケーション・コンテキストと異なるコンテキスト（すなわち、オペレーティング・システム・コンテキスト）内のオペレーティング・システム変数であるので、システム資源はアプリケーション・コンテキストを切り換える必要がある。例えば、コンテキスト切換え中に、データ・ポインタを変更し、ポインタをスタック上に押し上げ、プロセス制御パラメータも変更する。
バス・ロック能力を有しない従来技術のコンピュータ・システムは、高度に複雑なアルゴリズムを使用して、プロセッサ間の同期を実施する。これらのシステムでは、性能がさらに低下する。
したがって、セマフォを使用せずに待ち行列への直接アクセスを効率的に可能にする方法および装置が必要である。

発明の概要

複数プロセッサ・システム内のプロセッサ間でメッセージを送信する方法および装置。本

発明の方法および装置は、非対称複数プロセッサ・システム内のプロセッサ間でメッセージの通信を可能にする。非対称複数プロセッサ・システムは、単にプロセッサが様々なオペレーティング・システムを同時に実行しているシステムである。例えば、アプリケーション・プラットフォーム上のアプリケーション・プロセッサは、ＷｉｎｄｏｗｓＮＴ™など標準のアプリケーション・オペレーティング・システム・ソフトウェア上で動作している。しかしながら、入出力プラットフォーム上のプロセッサは、入出力動作（例えば、実時間オペレーティング・システム：ＲＴＯＳ）に適した特定のオペレーティング・システムを動作させている。特に、本発明は、１つまたは複数のプロセッサ・プラットフォーム上で実行されている１つまたは複数のプロセスからローカル・プロセッサを含んでいるプラットフォームへのメッセージを待ち行列化する迅速かつ直接的な機構を提供する。

本発明は、他のプラットフォームにメッセージ・バッファを割り当てるインバウンド・フリー待ち行列、および入出力プラットフォームの外部のプロセッサおよびバス・エージェントからメッセージをポストするインバウンド・ワーク待ち行列を提供する。さらに、本発明は、ローカル・プロセッサ（すなわち、入出力プラットフォーム用のプロセッサ）からのメッセージを、他のプラットフォーム上のプロセッサがこれらのメッセージを検索することができるように他のプロセッサ・プラットフォーム（すなわち、ホスト・プロセッサ）にポストするアウトバウンド・ワーク待ち行列を提供する。本発明はまた、ホスト・プロセッサがそれに対してメッセージ・バッファを解放するアウトバウンド・フリー待ち行列を提供する。この待ち行列は、ホスト・プロセッサがメッセージを処理した後、メッセージ・バッファをローカル・プロセッサに解放する。

本発明はホスト・プラットフォームと入出力プラットフォームとの間の非常に速くかつ効率的なハードウェア待ち行列インタフェースを提供するメッセージ・ユニットを使用してこれらの待ち行列を管理する。本発明は、単一のＰＣＩバス・トランザクション・サイクル中に自由なメッセージ・バッファ、すなわち「エンプティ」インジケータを準備（すなわち、メッセージ・ユニット内のレジスタの読取り）することができる。さらに、本発明は、単一のＰＣＩバス・トランザクション中にメッセージまたは「フル」インジケータのポストまたは検索（すなわち、メッセージ・ユニット内のレジスタの書込み）を可能にする。

本発明は、待ち行列をハードウェア・インタフェースを使用して管理するので、従来技術のソフトウェア待ち行列管理に勝るいくつかの利点を提供する。第１に、本発明はプロセスがフル待ち行列またはエンプティ待ち行列に対して待ち行列操作を実施しようとしたときデッドロックまたはロックアップを回避する。本発明のメッセージ・ユニットは、エンプティ・リストまたは待ち行列からフェッチしようとする試みが検出されたときエンプティ指示を迅速に戻す。同様に、本発明は、フル待ち行列にポストしようとする試みが検出されたとき特定の待ち行列がフルであるという指示を迅速に戻す。本発明は、最小のハードウェア資源を使用して効率的に実施することができる。

さらに、本発明は単一のバス・トランザクション内で待ち行列アクセスを実行するので、同期（すなわち、セマフォの獲得および解放）の必要がなく、またシステムの性能が大幅に改善される。待ち行列アクセスは、単に要素を待ち行列に追加すること、または要素を待ち行列から削除することである。待ち行列アクセスは、次の要素を見つけ出し、その要素を変更し、次の待ち行列アクセスのために次の要素を示す待ち行列記述子を修正する特定のタスクを含んでいる。これらのタスクは、本発明によって自動的に実施される。これらのタスクが完了する時間中、待ち行列は、他のプロセスが同じメッセージ・バッファを獲得したり、または他のメッセージを上書きしないようにロックされる。本発明は、単一のＰＣＩバス・トランザクションが本来アトミックであることを利用するために１つのバス・トランザクション内で待ち行列アクセス（すなわち、トランザクションを実行しているバス・エージェントによる独占アクセス）を提供する。さらに、本発明は、読取り信号および再試行信号を介して自動的に同期を処理する。

さらに、本発明ではセマフォが不要であるので、システム資源をタイアップするコンテキスト切換えが不要である。メッセージ・ユニット内のレジストに対する単一の読取りまた

は書込みだけで、特定の待ち行列にアクセスすることができ、また１つのバス・トランザクション内で読取りまたは書込みを行うことができるので、セマフォは不要である。
本発明について、添付の図面において例を挙げて非限定的に説明する。図面中、同じ参照番号は同じ要素を示す。

【図面の簡単な説明】

第１図は、本発明を実施する非対称複数プロセッサ・コンピュータ・システムのブロック図である。
第２図は、本発明を含む入出力プラットフォームを示す図である。
第３図は、本発明の一実施形態を示す図である。
第４図は、本発明の円形待ち行列を示す図である。
第５図は、本発明の円形待ち行列動作をさらに示す図である。
第６Ａ図は、本発明のインバウンド・フリー状態機械を示す図である。第６Ｂ図は、インバウンド・フリー状態機械の状態図である。
第７Ａ図は、本発明のインバウンド・ポスト状態機械を示す図である。第７Ｂ図は、インバウンド・ポスト状態機械の状態図である。
第８Ａ図は、本発明のアウトバウンド検索状態機械を示す図である。第８Ｂ図は、アウトバウンド検索状態機械の状態図である。
第９Ａ図は、本発明のアウトバウンド解放状態機械を示す図である。第９Ｂ図は、アウトバウンド解放状態機械の状態図である。

好ましい実施形態の詳細な説明

第１図は、本発明を実施する複数プロセッサ・コンピュータ・システムのブロック図である。マルチプロセッサ・システム１００は、ホスト・プロセッサ１０２を含んでいる。ホスト・プロセッサ１０２は、複数のプロセッサ（すなわち、密に結合されたプロセッサのクラスタ）を含んでいる。ホスト・プロセッサ１０２は、ホスト・バス１０３を介してホスト・メモリ１０４に結合される。メモリ・バス１０３はまた、ホスト・プロセッサ１０２およびメモリ１０４をホスト・チップ・セット１０５に結合する。ホスト・チップ・セット１０５は、メモリ・コントローラ、キャッシュ・コントローラ、およびメモリ・バス１０３と入出力（Ｉ／Ｏ）バス１０６（例えばＰＣＩバス）とのインタフェースを提供するブリッジを含んでいる。
ホスト・チップ・セット１０５は、当技術分野において周知である。例えば、ホスト・プロセッサ１０２はＩｎｔｅｌ社製のＰｅｎｔｉｕｍ™プロセッサである場合、適切なホスト・チップ・セット１０５は、これもＩｎｔｅｌ社製のＴｒｉｄｅｎｔ™である。同様に、Ｐ６™プロセッサが使用される場合、適切なホスト・チップ・セット１０５は、これもＩｎｔｅｌ社製のＯｒｉｏｎ™チップ・セットである。ホスト・プロセッサ１０２、メモリ・バス１０３、ホスト・メモリ１０４、ホスト・チップ・セット１０５は、このマルチプロセッサ・システム１００内ではホスト・プラットフォームと呼ばれる。
複数プロセッサ・システム１００はさらに、第１のＰＣＩバス１０６に結合された入出力プラットフォーム１０８を含んでいる。さらに、入出力プラットフォーム１０８は、第１のＰＣＩバス１０６のアドレス空間と、入出力プラットフォーム１０８内に含まれているプロセッサのアドレス空間とのインタフェースを提供する。入出力プラットフォーム１０８はさらに、第１のＰＣＩバス１０６を第２のＰＣＩバス（図示せず）に結合するブリッジを含んでいる。
入出力プラットフォーム１０８は、ホスト・プロセッサ用の入出力サポート、および第１のＰＣＩバス１０６および第２のＰＣＩバスに結合されたデバイス（図示せず）を提供する。第２図は、本発明を含んでいる（前に第１図において要素１０８と呼んでいた）入出力プラットフォーム２００を詳細に示す。入出力プラットフォーム２００は、メモリ・コントローラ２０５からローカル・バス２０４を介してローカル・メモリ２０６に結合されたローカル・プロセッサ２０２を含んでいる。ローカル・プロセッサ２０２は、Ｉｎｔｅｌ８０９６０ＪＦプロセッサである。
アドレス・トランザクション・ユニット（ＡＴＵ）２１８は、ローカル・バス２０４と（

前に第１図において要素１０６と呼んでいた）第１のＰＣＩバス２０８とに結合される。アドレス・トランザクション・ユニット（ＡＴＵ）２１８は、ＰＣＩバス２０８のアドレス空間内のアドレスをプロセッサ２０２アドレス空間内のアドレスに変換し、その逆も行う。したがって、ＰＣＩアドレス空間内のアドレスを有するＰＣＩバス２０８に対するトランザクションは、メモリ・コントローラ２０５がローカル・メモリ２０６内の正確な位置またはＭＵ２１０内の正確なレジスタ２１２にアクセスすることができるようにローカル・バス２０４アドレス空間に変換されなければならない。

ＡＴＵ２１８は、ローカル・バス・トランザクションをＰＣＩバス・トランザクションに変換するアウトバウンド・モジュール、ＰＣＩバス・トランザクションをローカル・バス・トランザクションに変換するインバウンド・モジュール、およびこのアドレス・トランザクションを管理する制御状態機械を含んでいる。本発明では、ＡＴＵ２１８は、特定のＰＣＩバス・トランザクションがＭＵ２１０内のレジスタ２１２の１つにアクセスすることを検出するアドレス・デコーダと考えられる。ＡＴＵ２１８は、トランザクションがＭＵ２１０内のレジスタ２１２の１つへのアクセスであることを検出した後、以下で説明するＭＵ２１０内の制御状態機械２１４を開始するためにデータ・パス２２１を介して信号を送る。制御状態機械２１４は、ＭＵ２１０がトランザクションを受け取る準備ができていないことをＡＴＵ２１８に通知するため、または要求側プロセスに再試行を通知するようＡＴＵ２１８に指示するために、複数の信号をデータ・パス２２１を介してＡＴＵ２１８に送る。

ローカル・バス・アービタ２４０は、ローカル・バス・マスタ（すなわち、ＭＵ２１０、ＡＴＵ２１８のインバウンド・モジュール、およびローカル・プロセッサ２０２）のいずれかにローカル・バス２０４の制御権を付与する。アービトレーション回路２４０は、当技術分野において周知である。

メモリ・コントローラ２０５は、データ・パス２２４および２２５を介してローカル・メモリ２０６にアクセスするために備えられる。ローカル・バス２０４は、単一のデータ・パスとして示されているが、ローカル・バス２０４は、アドレス部分およびデータ部分から構成される。

バス・エージェント２０１は、ホスト・プロセッサまたは他の入出力プラットフォームである。さらに、バス・エージェント２０１は、第１図のホスト・メモリ１０４、ホスト・プロセッサ１０２、ホスト・チップ・セット１０５、メモリ・バス１０３を含んでいる。言い換えれば、バス・エージェント２０１は、それ自体サブシステムまたは任意のインテリジェント・バス・エージェントである。

メッセージ・ユニット（ＭＵ）２１０は、ローカル・バス２０４とＡＴＵ２１８とに結合される。ＭＵ２１０は、本発明の教示を実施し、複数のレジスタ２１２および複数の状態機械２１４を含んでいる。これらのレジスタ２１２および状態機械２１４について、第３図に関して詳細に説明する。

第３図は、ＭＵ２１０内で実施された本発明を示す。ＭＵ２１０は、制御パス３５０を介してＡＴＵ２１８に結合された複数の状態機械２１４を含んでいる。ＭＵ２１０はまた、複数のプリフェッチおよび一時レジスタ３３２を含んでいる。これらのレジスタ３３２は、データ・パス３３６を介してＡＴＵ２１８に結合される。プリフェッチおよび一時レジスタ３３２はまた、データ・パス３５２を介して制御状態機械２１４によって制御される。レジスタ３３２はまた、ローカル・メモリ２０６にアクセスするためにデータ・パス３３４を介してローカル・バス２０４に結合される。

この実施形態では、ＭＵ２１０は、４つの円形待ち行列を使用したメッセージ送信方式を含んでいる。この実施形態では、４つのプリフェッチおよび一時レジスタ３３２がある。ホスト・プロセッサが円形待ち行列にデータを書き込むことを可能にする２つのレジスタが備えられる。ホスト・プロセッサが円形待ち行列の１つからデータを読み取ることを可能にする２つのレジスタが備えられる。

ＭＵ２１０はまた、データ・パス３４２を介して制御状態機械２１４に結合された複数の待ち行列ポインタ・レジスタ３４０を含んでいる。これらのレジスタ３４０は、待ち行列

２０７の頭部ポインタおよび尾部ポインタを記憶する。これらの待ち行列について、第４図および第５図に関して詳細に説明する。

円形待ち行列

ＭＵ２１０は、４つの円形待ち行列２０７へのアクセスをバス・エージェント２０１に提供する。２つのインバウンド待ち行列および２つのアウトバウンド待ち行列がある。「インバウンド」および「アウトバウンド」は、活動メッセージの流れの方向を示す。「インバウンド」メッセージは、ローカル・プロセッサ２０２が処理すべきバス・エージェント２０１によってポストされた新しいメッセージであるか、またはバス・エージェント２０１が使用するために使用できるエンプティまたはフリーのメッセージ・バッファである。「アウトバウンド」メッセージは、ホスト・プロセッサ２０１が処理すべきローカル・プロセッサ２０２によってポストされた新しいメッセージであるか、またはローカル・プロセッサ２０２が使用するために使用できるフリーのメッセージ・バッファである。

一実施形態では、ホスト・プロセッサ／バス・エージェント２０１とローカル・プロセッサ２０２との間でメッセージを送るために使用される４つの円形待ち行列がある。インバウンド・メッセージを処理するために使用される２つのインバウンド待ち行列があり、アウトバウンド・メッセージを処理するために使用される２つのアウトバウンド待ち行列がある。インバウンド待ち行列の１つは、フリー待ち行列に指定され、インバウンド・フリー・メッセージ・ハンドルを含んでいる。メッセージ・ハンドルは、メッセージ・バッファの論理アドレスまたは物理アドレスである。他のインバウンド待ち行列は、ポスト待ち行列または作業待ち行列に指定され、インバウンド・ポスト・メッセージ・ハンドルを含んでいる。同様に、アウトバウンド待ち行列の１つはフリー待ち行列に指定され、他のアウトバウンド待ち行列はポスト待ち行列に指定される。

２つのアウトバウンド待ち行列は、ローカル・プロセッサ２０２がアウトバウンド・メッセージをポスト待ち行列内にポストし、外部ホスト・プロセッサ２０１からアウトバウンド・フリー待ち行列内に戻ったフリー・メッセージを受信することを可能にする。２つのインバウンド待ち行列は、バス・エージェント２０１がインバウンド・フリー待ち行列からフリー・メッセージ・バッファを獲得し、その後ローカル・プロセッサ２０２によって処理するためにそのバッファをインバウンド・フリー待ち行列にポストすることを可能にする。

円形待ち行列２０７のデータ記憶は、ローカル・メモリ２０６によって実施される。この特定の実施形態では、待ち行列内の各エントリは、メッセージ・ハンドルである３２ビット・データ値である。さらに、待ち行列の読取りまたは書込みは、１つの待ち行列エントリに正確にアクセスすることができる。

各円形待ち行列は、頭部ポインタおよび尾部ポインタを有する。待ち行列への書込みは、待ち行列の頭部から行われ、読取りは、尾部から行われる。頭部ポインタおよび尾部ポインタは、ローカル・プロセッサ２０２上で動作しているソフトウェアまたはメッセージ・ユニット２１０によって増分される。頭部ポインタおよび尾部ポインタがどのようにしてローカル・プロセッサ２０２およびＭＵ２１０によって増分されるかに関する詳細について、以下で説明する。

頭部ポインタおよび尾部ポインタは、それぞれの円形待ち行列内にオフセットされ、０から円形待ち行列サイズ－１に及ぶ（すなわち、ポインタを０からラベル付けし始める）。ポインタは、各待ち行列アクセスの後で増分される。頭部ポインタならびに尾部ポインタは、円形待ち行列サイズ（すなわち、待ち行列の終り）に達したときに０に丸められる。

メッセージ・ユニット２１０は、ある条件のもとで、ローカル・プロセッサ２０２に対して割込みを発生するか、またはＰＣＩバス割込み（すなわち、外部プロセッサに対する割込み）を発生する。一般に、ポスト待ち行列が書き込まれたとき、メッセージがポストされたターゲット・プロセッサを通知する割込みが発生する。

一実施形態では、各円形待ち行列のサイズは１６Ｋバイト（４０９６ハンドル）から２５６Ｋバイト（６５５３６ハンドル）までである。さらに、この実施形態では、４つのすべての待ち行列は、同じサイズであり、隣接している。したがって、円形待ち行列によって

必要とされるローカル・メモリの総量は、６４Ｋバイトから１Ｍバイトまでである。これらの待ち行列は、ローカル・メモリ２０６内に常駐し、待ち行列の頭部ポインタおよび尾部ポインタは、ＭＵ２１０内のレジスタ内に常駐する。待ち行列サイズは、メッセージ・ユニット構成レジスタ（ＭＵＣＲ）内の待ち行列サイズ・フィールドによって決定される。ＭＵＣＲの可能な１つのフォーマットを表１に示す。また、この実施形態では、４つのすべての待ち行列に対して１つのベース・アドレスがある。各待ち行列の開始アドレスは、待ち行列ベース・アドレスおよび待ち行列サイズ・フィールドに基づいている。ベース・アドレスは、これもＭＵ２１０内に常駐する待ち行列ベース・アドレス・レジスタ（ＱＢＡＲ）内に記憶される。ＱＢＡＲの可能な１つのフォーマットを表２に示す。第６図から第９図に示す実施形態は、各待ち行列ごとに別個のベース・アドレスを含んでいる。

表１

ＭＵ構成レジスタ－ＭＵＣＲ			
ビット	デフォルト	読取り／書込み	説明
３１：０５	００００００００Ｈ	読取り専用	予約
０４：００	00000_2	読取り／書込み	待ち行列サイズ－このフィールドは、各円形待ち行列のサイズを決定する。４つのすべての待ち行列は同じサイズである。

表２

待ち行列ベース・アドレス・レジスタ－ＱＢＡＲ			
ビット	デフォルト	読取り／書込み	説明
３１：２０	０００Ｈ	読取り／書込み	待ち行列ベース・アドレス－円形待ち行列のローカル・メモリ・アドレス。
１９：００	００００００Ｈ	読取り専用	予約

第４図は、本発明の４つの円形待ち行列を示す。ローカル・メモリ２０６内に常駐する２つのアウトバウンド待ち行列４１０および４２０と、２つのインバウンド待ち行列４３０および４４０がある。

ローカル・プロセッサ２０２は、アウトバウンド・ポスト待ち行列４２０の頭部に書込みを行うことによってアウトバウンド・メッセージ４２２をポストする。ホスト・プロセッサ２０１は、アウトバウンド・ポスト待ち行列４２０の尾部の読取りを行うことによってアウトバウンド・ポスト待ち行列４２０からポスト・メッセージを取り出す。

ホスト・プロセッサ２０１は、アウトバウンド・フリー待ち行列４１０の頭部に書込みを行うことによってアウトバウンド・メッセージ・バッファ４１２を解放する。ローカル・プロセッサ２０２は、アウトバウンド・フリー待ち行列４１０の尾部からフリー・メッセージ・バッファ４１４を読み取る。

ホスト・プロセッサまたはバス・エージェント２０１は、インバウンド・ポスト待ち行列４３０の頭部に書込みを行うことによってインバウンド・ポスト待ち行列４３０にインバウンド・メッセージ４３２をポストする。ローカル・プロセッサ２０２は、インバウンド・ポスト待ち行列４３０の尾部からこれらのポスト・メッセージを読み取る。ホスト・プ

ロセッサがインバウンド・ポスト待ち行列４３０に書込みを行ったとき、ローカル・プロセッサ２０２に対して割込み４３６が発生する。
ローカル・プロセッサ２０２によってアウトバウンド・ポスト待ち行列４２０に対してメッセージがポストされたとき、ホスト・プロセッサ２０１に対して割込み４２６が発生する。ここでは、ＰＣＩバス仕様改訂２．０によって指定されている割込みを使用する。
ローカル・プロセッサ２０２は、インバウンド・フリー待ち行列４４０の頭部に書込みを行うことによってこの待ち行列４４０にフリー・メッセージ・バッファ４４２を戻す。ホスト・プロセッサ／バス・エージェント２０１は、データ・パス４４４を介してインバウンド・フリー待ち行列４４０の尾部から読取りを行うことによってフリー・メッセージ・バッファを獲得する。
第５図は、アウトバウンド・フリー待ち行列５１０、アウトバウンド・ポスト待ち行列５２０、インバウンド・ポスト待ち行列５３０、インバウンド・フリー待ち行列５４０を示す。
アウトバウンド・フリー待ち行列
アウトバウンド・フリー待ち行列（ＯＦＱ）５１０は、ローカル・プロセッサ２０２が使用すべきバス・エージェント２０１によってそこに配置された（すなわち、解放された）エンプティ・メッセージ用のハンドルを保持する。ホスト・プロセッサ２０１は、アウトバウンド・ポート５１６内のレジスタに書込みを行うことによってＯＦＱ５１０に対してメッセージ・バッファを解放する。ＯＦＱ５１０は、ローカル・プロセッサ２０２によって待ち行列尾部から読み取られ、ホスト・プロセッサ２０１によって待ち行列頭部に書き込まれる。頭部ポインタ（ＯＦＨＰ）５１２は、メッセージ・ユニット２１０によって維持される。アウトバウンド・フリー待ち行列尾部ポインタ（ＯＦＴＰ）５１４は、ローカル・プロセッサ２０２上で動作しているソフトウェアによって維持される。
アウトバウンド待ち行列ポート５１６にアクセスするＰＣＩ書込みトランザクションの場合、ＭＵ２１０は、メッセージ・ハンドル（すなわち、フリー・メッセージ・バッファに対するアドレス）を、アウトバウンド・フリー頭部ポインタ・レジスタ（ＯＦＨＰＲ）９２６内に記憶された頭部ポインタ（ＯＦＨＰ）５１２によって示されたローカル・メモリ２０６内の位置に書き込む。ローカル・メモリ・アドレスは、待ち行列ベース・アドレス・レジスタ＋３＊待ち行列サイズ＋アウトバウンド・フリー頭部ポインタ・レジスタ（ＯＦＨＰＲ）９２６である。ＯＦＨＰＲの可能な１つのフォーマットを表３に示す。
アウトバウンド待ち行列ポート５１６に書き込まれたデータがローカル・メモリ２０６に書き込まれたとき、ＭＵ２１０はＯＦＨＰ５１２を増分する。
データがローカル・メモリ２０６に書き込まれるまでＰＣＩ書込みトランザクションがＭＵ２１０によって受け取られ、かつＯＦＨＰ５１２が増分されたときから、インバウンド待ち行列ポート５１６にアクセスしようと試みるＰＣＩトランザクションは、待ち状態を挿入することによって遅延される。待ち状態を挿入している間にＰＣＩ待ち時間違反が発生した場合、外部ＰＣＩエージェント２０１に再試行が通知される。
ローカル・プロセッサ２０２は、アウトバウンド・フリー待ち行列尾部ポインタ（ＯＦＴＰ）５１４によって示されたローカル・メモリ位置を読み取ることによってＯＦＱ５１０からメッセージ・バッファ・ハンドルを取り出す。ローカル・メモリ・アドレスは、待ち行列ベース・アドレス・レジスタ＋３＊待ち行列サイズ＋アウトバウンド・フリー尾部ポインタ・レジスタ（ＯＦＴＰＲ）９３８である。ＯＦＴＰＲの可能な１つのフォーマットを表４に示す。次いで、ローカル・プロセッサ２０２は、（第９Ａ図に示される）アウトバウンド・フリー尾部ポインタ・レジスタ（ＯＦＴＰＲ）９３８内のＯＦＴＰ５１４を増分する。

表３

アウトバウンド・フリー頭部ポインタ・レジスタ－ＯＦＨＰＲ			
ビット	デフォルト	アクセス	説明
３１：１９	０００００Ｈ	読取り専用	予約
１８：０２	０００００Ｈ	読取り／書込み	アウトバウンド・フリー頭部ポインタ・レジスターアウトバウンド・ポスト待ち行列に対する頭部ポインタのローカル・メモリ・オフセット
０１：００	００_{2}	読取り専用	予約

表４

アウトバウンド・フリー尾部ポインタ・レジスタ－ＯＦＴＰＲ			
ビット	デフォルト	アクセス	説明
３１：１９	０００００Ｈ	読取り専用	予約
１８：０２	０００００Ｈ	読取り／書込み	アウトバウンド・フリー尾部ポインタ・レジスターアウトバウンド・フリー待ち行列に対する尾部ポインタのローカル・メモリ・オフセット
０１：００	００_{2}	読取り専用	予約

アウトバウンド・ポスト待ち行列

アウトバウンド・ポスト待ち行列（ＯＰＱ）５２０は、ホスト・プロセッサ２０１が取り出し、処理するために、ローカル・プロセッサ２０２によってそこに配置されたポスト・メッセージのハンドルを格納している。ホスト・プロセッサ２０１は、アウトバウンド待ち行列ポート５１６内のレジスタの読取りを行うことによってＯＰＱ５２０からメッセージを取り出す。ローカル・プロセッサ２０２は、待ち行列頭部に書込みを行うことによってＯＰＱ５２０にメッセージを追加する。頭部ポインタ（ＯＰＨＰ）５２２は、ローカル・プロセッサ２０２によって維持される。尾部ポインタ（ＯＰＴＰ）５２４は、メッセージ・ユニット２１０によって維持される。

アウトバウンド待ち行列ポート５１６にアクセスするＰＣＩ読取りトランザクションの場合、ＭＵ２１０は、ＯＰＴＰ５２４によって示されたローカル・メモリ位置においてデータをプリフェッチする。ローカル・メモリ・アドレスは、待ち行列ベース・アドレス・レジスタ＋２＊待ち行列サイズ＋アウトバウンド・ポスト尾部ポインタ・レジスタ（ＯＰＴＰＲ）８２６（第８Ａ図に示す）である。ＯＰＱ５２０がエンプティでない（すなわち、頭部ポインタ５２２と尾部ポインタ５２４が等しくない）場合、メッセージ・ハンドルが要求側プロセッサ２０１に供給される。ＯＰＱ５２０がエンプティである（すなわち、頭部ポインタ５２２と尾部ポインタ５２４が等しい）場合、－１の値（ＦＦＦＦ．ＦＦＦＦＨ）が要求側プロセッサ２０１に供給される。ＯＰＱ５２０待ち行列がエンプティでなく、かつＭＵ２１０が尾部におけるデータのプリフェッチに成功した場合、ＭＵ２１０は、ＯＰＴＰＲ８２６内の尾部ポインタ（ＯＰＴＲ）５２４を増分する。

上述のように、プリフェッチ機構は、頭部ポインタ５２２と尾部ポインタ５２４が等しい（すなわち、ＯＰＱ５２０がエンプティである）場合、－１の値（ＦＦＦＦ．ＦＦＦＦＨ）を（以下で第８Ａ図に関して説明する）プリフェッチ・レジスタ８０６内にロードする。メッセージがＯＰＱ５２０に追加され、それがエンプティでなくなったときにＯＲＲ８

０６を更新するために、ＭＵ２１０内のプリフェッチ機構は、ＯＲＲ８０６が（ＦＦＦＦ．ＦＦＦＨ）を含んでいる場合、自動的にプリフェッチを開始し、アウトバウンド・ポスト頭部ポインタ・レジスタ（ＯＰＨＰＲ）４２２がローカル・プロセッサ２０２によって書き込まれる。ＯＰＨＰＲの可能な１つのフォーマットを表５に示す。ローカル・プロセッサ２０２は、ローカル・プロセッサ２０２がＯＰＱ５２０にメッセージを追加したときＯＰＨＰＲ４２２を更新する。

プリフェッチは、外部バス・エージェント２０１から見てアトミックに見えなければならない。プリフェッチが開始されたとき、アウトバウンド待ち行列ポート５１６内の（以下で第８Ａ図に関して説明する）アウトバウンド検索レジスタ８０６にアクセスしようと試みるＰＣＩトランザクションは、プリフェッチが完了するまで待ち状態を挿入することによって遅延される。待ち状態を挿入している間にバス待ち時間違反が発生した場合、外部バス・エージェント２０１に再試行信号が通知される。

ＯＰＨＰ５２２がＯＰＴＰ５２４に等しくない場合、ホスト・プロセッサ２０１に対してＰＣＩ割込みが発生する。ＯＰＨＰ５２２がＯＰＴＰ５２４に等しい場合、割込みは発生しない。アウトバウンド・ドーベル・レジスタ内のアウトバウンド・ポスト待ち行列割込みビットは、ＯＰＨＰＲ８３８の値とＯＰＴＰＲ８２８の値との比較の状態を示す。頭部ポインタ５２２と尾部ポインタ５２４が等しい場合、割込みはクリアされる。これは、ホスト・プロセッサ２０１がＯＰＱ５２０をエンプティにするために十分な待ち行列エントリを読み取る場合に起こる。割込みは、ソフトウェアによって制御されるアウトバウンド・ドーベル・マスク・レジスタによってマスクされる。

ローカル・プロセッサ２０２は、頭部ポインタ（ＯＰＨＰ）５２２によって示されたローカル・メモリ位置にデータを書き込むことによってメッセージをＯＰＱ５２０内に入れる。ローカル・メモリ・アドレスは、待ち行列ベース・アドレス・レジスタ＋アウトバウンド・ポスト頭部ポインタ・レジスタ８３８である。ＯＰＴＰＲの可能な１つのフォーマットを表６に示す。次いで、ローカル・プロセッサ２０２は、アウトバウンド・ポスト頭部ポインタ・レジスタ８３８内のＯＰＨＰ５２２を増分する。

表５

アウトバウンド・ポスト頭部ポインタ・レジスターＯＰＨＰＲ			
ビット	デフォルト	アクセス	説明
３１：１９	０００００Ｈ	読取り専用	予約
１８：０２	０００００Ｈ	読取り／書込み	アウトバウンド・ポスト頭部ポインタ・レジスターアウトバウンド・ポスト待ち行列に対する頭部ポインタのローカル・メモリ・オフセット
０１：００	００$_2$	読取り専用	予約

表６

アウトバウンド・ポスト尾部ポインタ・レジスタ－ＯＰＴＰＲ			
ビット	デフォルト	アクセス	説明
３１：１９	００００Ｈ	読取り専用	予約
１８：０２	００００Ｈ	読取り／書込み	アウトバウンド・ポスト尾部ポインタ・レジスターアウトバウンド・ポスト待ち行列に対する尾部ポインタのローカル・メモリ・オフセット
０１：００	００_2	読取り専用	予約

インバウンド・ポスト待ち行列

インバウンド・ポスト待ち行列（ＩＰＱ）５３０は、ローカル・プロセッサ２０２が処理するために、バス・エージェント２０１によってそこに配置されたポスト・メッセージのハンドルを保持する。ホスト・プロセッサ２０１またはバス・エージェントは、インバウンド待ち行列ポート５３６内のレジスタに書込みを行うことによってＩＰＱ５３０にメッセージをポストする。ＩＰＱ５３０は、ローカル・プロセッサ２０２によって待ち行列尾部から読み取られ、外部バス・エージェント２０１によって待ち行列頭部に書き込まれる。尾部ポインタ（ＩＰＴＰ）５３４は、ローカル・プロセッサ２０２上で動作しているソフトウェアによって維持される。頭部ポインタ（ＩＰＨＰ）５３２は、ＭＵ２１０によって維持される。

インバウンド待ち行列ポート（ＩＱＰ）５３６にアクセスするＰＣＩ書込みトランザクションの場合、ＭＵ２１０は、（第７Ａ図に示される）インバウンド・ポスト頭部ポインタ・レジスタ（ＩＰＨＰＲ）７２４内に記憶されたＩＰＨＰ５３２によって示されたローカル・メモリ位置にデータを書き込む。ローカル・メモリ・アドレスは、待ち行列ベース・アドレス・レジスタ＋待ち行列サイズ＋インバウンド・ポスト頭部ポインタ・レジスタ（ＩＰＨＰＲ）７２４である。ＩＰＨＰＲの可能な１つのフォーマットを表７に示す。ＩＰＴＰＲの可能な１つのフォーマットを表８に示す。

インバウンド待ち行列ポート５３６に書き込まれたデータがローカル・メモリ２０６に書き込まれたとき、ＭＵ２１０はＩＰＨＰＲ７２４を増分する。データがローカル・メモリ２０６に書き込まれ、ＩＰＨＰＲ７２４が増分されたとき、ＭＵ２１０は、ローカル・プロセッサ２０２に対して割込みを発生する。この割込みは、インバウンド・ドーベル・レジスタのインバウンド・ポスト待ち行列割込みビットを設定することによって記録される。割込みは、ソフトウェアによって制御されるインバウンド・ドーベル・マスク・レジスタによってマスクされる。

表７

インバウンド・ポスト頭部ポインタ・レジスタ－ＩＰＨＰＲ			
ビット	デフォルト	アクセス	説明
３１：１９	００００Ｈ	読取り専用	予約
１８：０２	００００Ｈ	読取り／書込み	インバウンド・ポスト頭部ポインタ・レジスタ－インバウンド・ポスト待ち行列に対する頭部ポインタのローカル・メモリ・オフセット
０１：００	００_2	読取り専用	予約

表８

インバウンド・ポスト尾部ポインタ・レジスターＩＰＴＰＲ			
ビット	デフォルト	アクセス	説明
３１：１９	００００Ｈ	読取り専用	予約
１８：０２	００００Ｈ	読取り／書込み	インバウンド・ポスト尾部ポインタ・レジスターインバウンド・ポスト待ち行列に対する尾部ポインタのローカル・メモリ・オフセット
０１：００	００_{2}	読取り専用	予約

インバウンド・フリー待ち行列

インバウンド・フリー待ち行列５４０は、バス・エージェント２０１が使用するために、ローカル・プロセッサ２０２によってそこに配置されたエンプティ・メッセージ・バッファのハンドルを保持する。ホスト・プロセッサ２０１は、インバウンド待ち行列ポート５３６内のレジスタの読取りを行うことによってＩＦＱ５４０からメッセージ・バッファを割り振られる。ＩＦＱ５４０は、外部バス・エージェント２０１によって待ち行列尾部から読み取られ、ローカル・プロセッサ２０２によって待ち行列頭部に書き込まれる。頭部ポインタ（ＩＰＨＰ）５４２は、ローカル・プロセッサ２０２上で動作しているソフトウェアによって維持される。尾部ポインタ（ＩＦＴＰ）５４４は、ＭＵ２１０によって維持される。

インバウンド待ち行列ポート（ＩＱＰ）５３６にアクセスするＰＣＩ読取りトランザクションの場合、ＭＵ２１０は、ＩＦＴＰ５４４によって示されたローカル・メモリ位置においてデータをプリフェッチする。ローカル・メモリ・アドレスは、尾部ポインタを記憶する待ち行列ベース・アドレス・レジスタ＋インバウンド・フリー尾部ポインタ・レジスタ（ＩＦＴＰＲ）６２６である。ＩＦＴＰＲの可能な１つのフォーマットを表１０に示す。ＩＦＱ５４０がエンプティでない（すなわち、頭部ポインタと尾部ポインタが等しくない）場合、ホスト・プロセッサまたはバス・エージェントによる次のアクセスのためにＩＦＴＰ５４４によって示されたデータが供給される。ＩＦＱ５４０がエンプティである（すなわち、頭部ポインタと尾部ポインタが等しい）場合、－１の値（ＦＦＦＦ．ＦＦＦＦＨ）が要求側ホスト・プロセッサまたはバス・エージェントに供給される。ＩＦＱ５４０がエンプティでなく、かつＭＵ２１０がＩＦＴＰ５４４によって示されたデータをプリフェッチした場合、ＭＵ２１０は、インバウンド・フリー尾部ポインタ・レジスタ（ＩＦＴＰＲ）６２６内のポインタの値を増分する（第６Ａ図に示す）。

ＰＣＩ読取りアクセスに対する待ち時間を短縮するために、ＭＵ２１０は、ＩＦＱ５４０へのアクセスを予測するプリフェッチ機構を実施する。ＭＵ２１０は、ＩＦＱ５４０の尾部からデータをプリフェッチし、それを内部プリフェッチ・レジスタ内にロードする。ＰＣＩ読取りアクセスが行われたとき、データは、プリフェッチ・レジスタから直接読み取られる。

プリフェッチ機構は、頭部ポインタと尾部ポインタが等しい（すなわち、ＩＦＱ５４０がエンプティである）場合、－１の値（ＦＦＦＦ．ＦＦＦＦＨ）をプリフェッチ・レジスタ８０６内にロードする。メッセージがＩＦＱ５４０に追加され、それがエンプティでなくなったときにプリフェッチ・レジスタを更新するために、プリフェッチ機構は、プリフェッチ・レジスタがＦＦＦＦ．ＦＦＦＦＨを含んでいる場合、自動的にプリフェッチを開始し、インバウンド・フリー頭部ポインタ・レジスタ（ＩＦＨＰＲ）６３８が書き込まれる。ＩＦＨＰＲの可能な１つのフォーマットを表９に示す。ローカル・プロセッサ２０２上で動作しているソフトウェアは、ＩＦＨＰ５４２がＩＦＱ５４０にメッセージを追加したときにＩＦＨＰ５４２を更新する。

プリフェッチは、外部バス・エージェント２０１から見てアトミックに見えなければならない。プリフェッチが開始されたとき、インバウンド待ち行列ポート５３６内のインバウンド・フリー・レジスタにアクセスしようと試みるＰＣＩトランザクションは、プリフェッチが完了するまで待ち状態を挿入することによって遅延される。待ち状態を挿入している間にＰＣＩ待ち時間違反が発生した場合、ＭＵ２１０によって外部バス・エージェント２０１に再試行が通知される。

ローカル・プロセッサ２０２は、頭部ポインタ（ＩＦＨＰ）５４２によって示されたローカル・メモリ位置にデータを書き込むことによってメッセージをＩＦＱ５４０内に入れる。ローカル・メモリ・アドレスは、待ち行列ベース・アドレス・レジスタ＋インバウンド・フリー頭部ポインタ・レジスタ（ＩＦＨＰＲ）６３８である。次いで、ローカル・プロセッサ２０２上で動作しているソフトウェアは、ＩＦＨＰＲ６３８を増分する。

インバウンド・フリー表

表９

インバウンド・フリー頭部ポインタ・レジスタ－ＩＦＨＰＲ			
ビット	デフォルト	アクセス	説明
３１：１９	００００Ｈ	読取り専用	予約
１８：０２	００００Ｈ	読取り／書込み	インバウンド・フリー頭部ポインタ・レジスタ－インバウンド・フリー待ち行列に対する頭部ポインタのローカル・メモリ・オフセット
０１：００	００_2	読取り専用	予約

表１０

インバウンド・フリー尾部ポインタ・レジスタ－ＩＦＴＰＲ			
ビット	デフォルト	アクセス	説明
３１：１９	００００Ｈ	読取り専用	予約
１８：０２	００００Ｈ	読取り／書込み	インバウンド・フリー尾部ポインタ・レジスタ－インバウンド・フリー待ち行列に対する尾部ポインタのローカル・メモリ・オフセット
０１：００	００_2	読取り専用	予約

第６Ａ図は、本発明がどのようにしてＰＣＩバス上のバス・エージェントにフリー・メッセージ・バッファを割り振るのかを示す。ＭＵ２１０内で実現される。データは、ローカル・メモリ２０６内に配置されたインバウンド・フリー待ち行列（ＩＦＱ）５４０からローカル・データ・バスを介してインバウンド・フリー・レジスタ（ＩＦＲ）６０６まで移動する。このコンテキストでは、データは、特にメッセージ・バッファのアドレス（すなわち、メッセージ・ハンドル）を示す。その後、データは、インバウンド・フリー・レジスタ６０６からデータパス６０８を介してＡＴＵ２１８まで移動し、その後データパス６１０を介してＰＣＩバス２０８上のバス・エージェントまで移動する。

ＭＵ２１０は、いくつかの制御信号を発生し、受信するフリー・メッセージ・バッファを割り振るインバウンド・フリー状態機械６１２を含んでいる。インバウンド・フリー状態機械６１２の状態図について、第６Ｂ図に関して詳細に説明する。

ＩＦＱ６０２にメッセージ・バッファを要求するために、バス・エージェントは、ＰＣＩ

バス２０８およびデータパス６１０を介して読取りトランザクションをＡＴＵ２１８に送る。インバウンド・フリー・レジスタ６０６のアドレスを指定する読取りトランザクションは、ＡＴＵ２１８によって検出される。ＡＴＵ２１８が、バス・エージェントがインバウンド・フリー・レジスタ６０６を読み取ることを望んでいることを検出した後、ＡＴＵは、ＩＦＲ＿Ｒｅａｄｙ信号６１４の状態をテストする。ＩＦＲ＿Ｒｅａｄｙ信号６１４がアサートされた場合、ＡＴＵは、パス６０８を介してＩＦＲ６０６内のデータをＡＴＵ２１８に供給するＰＣＩトランザクションを完了し、状態機械６１２に対してＲｅａｄ＿Ｉｎｂｏｏｕｎｄ＿Ｆｒｅｅ信号６１６を発生する。

ＩＦＲ＿Ｒｅａｄｙ信号６１４がアサート解除された場合（すなわち、状態機械６１２が準備できていない場合）、ＡＴＵ２１８は、待ち状態を挿入し、ＩＦＲ＿Ｒｅａｄｙ信号６１４がアサートされるまでＲｅａｄ＿ＩＦＲ信号６１６を送らない。ＩＦＲ＿Ｒｅａｄｙ信号６１４は、ＩＦＲ６０６内に状態データがある（すなわち、状態機械６１２がまだＩＦＲ６０６へのデータのプリフェッチを完了していない）場合、アサート解除される。状態機械６１２は、Ｒｅａｄ＿ＩＦＲ信号６１６を受信した後、ローカル・バス・アービタ２４０にＭｅｍｏｒｙ＿Ｒｅａｄ＿Ｒｅｑｕｅｓｔ信号６１８を送り、ＩＦＲ＿Ｒｅａｄｙ信号６１４をアサート解除する。許可信号６３２に基づいて、ＭＵ２１０は、単にＩＦＱ６０２の適切な尾部アドレスをローカル・アドレス・バス６３０にアサートする。次いで、データがローカル・データ・バス６０４を介してローカル・メモリ２０６からＩＦＲ６０６に転送される（すなわち、ＩＦＱ６０２の尾部において値を読み取る）。ＭＵ２１０は、ＩＦＱ６０２の適切な尾部アドレスを計算する加算器６２４を含んでいる。加算器６２４は、インバウンド・フリー尾部ポインタ・レジスタ（ＩＦＴＰＲ）６２６およびインバウンド・フリーベースレジスタ（ＩＦＢＲ）６２８の内容の合計を発生する。

ＩＦＱ５４０の尾部ポインタによって示されたデータがローカル・データ・バス６０４上にきた後、状態機械６１２は、ローカル・データ・バス６０４上のデータをＩＦＱ６０６内にラッチするためにラッチ信号６３４を送り、増分信号６４４をＩＦＴＰＲ６２６に送る。したがって、次の使用できるメッセージ・バッファのプリフェッチが行われた。

ＭＵ２１０はまた、インバウンド・フリー頭部ポインタ・レジスタ（ＩＦＨＰＲ）６３８内の値とインバウンド・フリー尾部ポインタ・レジスタ（ＩＦＴＰＲ）６２６内の値とを比較するコンパレータ６３６を含んでいる。これら２つの値が等しい場合、コンパレータ６３６は、エンプティ信号６４０を発生する（すなわち、待ち行列内にフリー・メッセージ・バッファがない）。このエンプティ信号６４０は、状態機械６１２に送られ、状態機械６１２にプリセット信号をアサートさせる。プリセット信号６４２により、ＩＦＲ６０６の内容がエンプティ指示に対して予約された（すなわち、有効なバッファアドレスでない）所定の値に設定される。バス・エージェントは、ＩＦＲ６０６を読み取る場合、ＩＦＲ６０６内に記憶されたプリフェッチされたデータか、またはＩＦＱ６０２がエンプティであることを示すプリセット値に直ちにアクセスする

第６Ｂ図は、インバウンド・フリー状態機械６１２の状態図を示す。状態機械６１２は、エンプティ状態６５０、プリフェッチ状態６５２、プライム状態６５６の３つの状態を有する。状態機械６１２は、エンプティ信号６５４がアサート解除されるまでエンプティ状態６５０にある。ｎｏｔ＿Ｅｍｐｔｙ信号は、状態機械６１２をエンプティ状態６５０からプリフェッチ状態６５２に遷移させ、状態機械６１２は、Ｍｅｍｏｒｙ＿Ｒｅａｄ＿Ｒｅｑｕｅｓｔ信号６１８を発生し、ＩＦＲ＿Ｒｅａｄｙ信号６１４をアサート解除する。状態機械６１２は、許可信号６３２に基づいてプリフェッチ状態６５２からプライム状態６５６に遷移する。許可信号６３２を受信すると、状態機械６１２は、Ｌａｔｃｈ＿ＩＦＲ信号６３４、Ｉｎｃｒｅｍｅｎｔ＿ＩＦＴＰＲ信号６４４を出力し、ＩＦＲ＿Ｒｅａｄｙ信号６１４をアサートする。状態機械６１２は、Ｒｅａｄ＿ＩＦＲ信号６１６が受信されたとき、プライム状態６５６からプリフェッチ状態６５２に遷移し、エンプティ信号６５４はアサートされない。この遷移はまた、Ｍｅｍｏｒｙ＿Ｒｅａｄ＿Ｒｅｑｕｅｓｔ信号６１８を発生し、ＩＦＲ＿Ｒｅａｄｙ信号６１４をアサートする。

状態機械６１２は、Ｒｅａｄ＿ＩＦＲ信号６１６が受信されたとき、プライム状態６５６

からエンプティ状態６５０に遷移し、エンプティ信号６４０はアサートされる。この遷移は、プリセット信号６４２を発生させる。
第７Ａ図は、本発明がどのようにしてバス・エージェントによって生成されたメッセージを、ローカル・メモリ２０６内に配置されたインバウンド・ポスト待ち行列（ＩＰＱ）５３０内にポストするかを示す。
バス・エージェントがインバウンド・ポスト・レジスタ（ＩＰＲ）７０６に書込みを行いたい場合、データは、ＰＣＩバス２０８からデータパス７０２を介してＡＴＵ２１８まで進み、次いでデータパス７０４を介してＩＰＲ７０６に進む。データは、ＩＰＲ７０６内にラッチされた後、ローカル・データ・バス６０４を介してローカル・メモリ２０６内のＩＰＱ５３０内に転送される。
ＡＴＵ２１８は、ＩＰＲ＿Ｒｅａｄｙ信号７１６の状態をテストする。ＩＰＲ＿Ｒｅａｄｙ信号７１６がアサートされた場合、ＡＴＵ２１８は、データをＩＰＲ７０６に供給し、状態機械７１２に対してＷｒｉｔｅ＿ＩＰＲ信号７１８を発生することによってＰＣＩトランザクションを完了する。
ＩＰＲ＿Ｒｅａｄｙ信号７１６がアサートされない場合、ＡＴＵ２１８は、待ち状態を挿入し、ＩＰＲ＿Ｒｅａｄｙ信号７１６がアサートされたときＰＣＩトランザクションを完了する。要求側プロセスは、バスの制御権を保持し、ＰＣＩ待ち時間ルールが違反されなければＰＣＩトランザクションは完了する。
ＡＴＵ２１８はまた、ＩＰＲ＿Ｒｅｔｒｙ信号７１４の状態をテストする。ＩＰＲ＿Ｒｅｔｒｙ信号７１４がアサートされた場合、ＰＣＩトランザクションは完了せず、要求側プロセスは、再試行を通知され、バスを解放し後で再度試行する。
ＭＵ２１０のインバウンド・ポスト状態機械７１２について、第７Ｂ図に示される状態図によって詳細に説明する。状態機械７１２は、アイドル状態７５０、ポスト状態７５２、フル状態７５４の３つの状態を有する。状態機械７１２は、Ｗｒｉｔｅ＿Ｉｎｂｏｕｎｄ＿Ｐｏｓｔ信号７１８がＡＴＵ２１８によってアサートされたときアイドル状態７５０からポスト状態７５２に遷移する。Ｗｒｉｔｅ＿Ｉｎｂｏｕｎｄ＿Ｐｏｓｔ信号が状態機械７１２によって受け取られたとき、状態機械７１２は、Ｍｅｍｏｒｙ＿Ｗｒｉｔｅ＿Ｒｅｑｕｅｓｔ信号７２０を発生し、ＩＰＲ＿Ｒｅａｄｙ信号７１６をアサート解除する。状態機械７１２は、状態機械７１２がローカル・バス・アービタ２４０から許可信号７２８を受信したときポスト状態７５２からアイドル状態７５０に戻る。許可信号７２８を受信し、ＩＰＲデータ６０４をメモリに書き込んだとき、状態機械７１２は、インバウンド・ポスト頭部ポインタ・レジスタ（ＩＰＨＰＲ）７２４に対して増分信号７４０を発生し、またＩＰＲ＿Ｒｅａｄｙ信号７１６をアサートする。
状態機械７１２は、コンパレータ７３４からフル信号７３８を受信したときアイドル状態７５０からフル状態７５４に遷移する。フル信号７３８は、インバウンド・ポスト尾部ポインタ・レジスタ（ＩＰＴＰＲ）７３０およびインバウンド・ポスト頭部ポインタ・レジスタ（ＩＰＨＰＲ）７２４の内容がインバウンド・ポスト待ち行列（ＩＰＱ）５３０がフルであることを示す場合、コンパレータ７３４によって生成される。フル信号７３８を受信したとき、状態機械７１２は、ＩＰＲ＿Ｒｅｔｒｙ信号７１４をＡＴＵ２１８にアサートする。
状態機械７１２は、フル信号７５６がフル信号７５６（すなわち、ｎｏｔ＿Ｆｕｌｌ）にアサート解除されたときフル状態７５４からアイドル状態７５０に遷移する。ｎｏｔ＿Ｆｕｌｌ信号を受信したとき、状態機械７１２は、ＩＰＲ＿Ｒｅｔｒｙ信号７１４をアサート解除する。
コンパレータ７３４はまた、入出力プロセッサに対してローカル割込みを発生する割込み発生論理（図示せず）に対してｎｏｔ＿Ｅｍｐｔｙ信号７３６を発生する。ｎｏｔ＿Ｅｍｐｔｙ信号７３６を受信したときにローカル割込みを発生する論理は、当技術分野において周知である。この論理はまた、割込みレジスタを含んでおり、またソフトウェアによって制御され、割込みを選択的にマスクするマスクレジスタを含んでいる。
増分信号７４０は、ＩＰＨＰＲ７２４に送られ、インバウンド・ポスト頭部ポインタを増

分する。加算器７２２は、ＩＰＨＰＲ７２４の値７２５およびＩＰＢＲ７２６の値７２７を使用して、新しいインバウンド・ポスト頭部ポインタ７２３を計算する。このアドレス７２３は、ローカル・バス（すなわち、ローカル・アドレス・バス６３０）を介してローカル・メモリにアクセスするためにメモリ・コントローラ２０５に送られる。

前に説明したように、ＭＵ２１０は、ローカル・アドレス・バス６３０上のアドレス７２３をアサートし、ＩＰＲ７０６内でＩＰＱ５３０の頭部にラッチされたデータ（すなわち、メッセージ・バッファのアドレス）の転送を可能にする。

第８Ａ図は、アウトバウンド検索状態機械８１２、および本発明がどのようにしてホスト・プロセッサまたはバス・エージェントがアウトバウンド・ポスト待ち行列５２０（ＯＰＱ）からポスト・メッセージを取り出すことを可能にするかを示す。ホスト・プロセッサまたはバス・エージェントがポスト・メッセージ・ハンドルを取り出したとき、データ（すなわち、メッセージ・バッファのアドレス）は、ローカル・データ・バス６０４を介してローカル・メモリ２０６内のＯＰＱ５２０からアウトバウンド検索レジスタ（ＯＲＲ）８０６に進む。次いで、データは、ＯＲＲ８０６からデータパス８０８を介してＡＴＵ２１８のアウトバウンド部分に送られる。次いで、データは、データパス８１０およびＰＣＩバス２０８を介してそれぞれホスト・プロセッサまたはバス・エージェントに送られる。状態機械８１２は、ＯＲＲ８０６内の状態データを示すためにＯＲＲ＿Ｒｅａｄｙ８１４をアサート解除する。ＯＲＲ＿Ｒｅａｄｙ信号８１４がアサート解除されたとき、ＡＴＵ２１８は、ＯＲＲ＿Ｒｅａｄｙ信号８１４がアサートされるまで待ち状態を挿入する。これは、ＯＲＲ８０６が有効なデータであることを示す。

ＭＵ２１０のアウトバウンド検索状態機械８１２について、第８Ｂ図に示される状態図によって詳細に説明する。アウトバウンド検索状態機械８１２は、エンプティ状態８５０、プリフェッチ状態８５２、プライム状態８５６の３つの状態を有する。アウトバウンド検索状態機械８１２は、エンプティ信号８４０がアサート解除されたときエンプティ状態８５０からプリフェッチ状態８５２に遷移する。それに応答して、アウトバウンド検索状態機械８１２は、Ｍｅｍｏｒｙ＿Ｒｅａｄ＿Ｒｅｑｕｅｓｔ８１８をローカル・バスアービトレーションユニット２４０に対してアサートし、許可信号８３２を待っている間、ＯＲＲ＿Ｒｅａｄｙ信号８１４をアサート解除する。許可信号８３２を待っている間、加算器８２４は、次のメッセージのアドレス（すなわち、尾部ポインタ）を計算し、このアドレスをローカル・アドレス・バス６３０上に配置する。

状態機械８１２は、許可信号８３２に基づいてプリフェッチ状態８５２からプライム状態８５６に遷移する。メモリ・コントローラ２０５は、アドレス８２５を使用し、ＯＰＱ５２０から適切なメッセージ・ハンドルを読み取る。このメッセージ・ハンドル（すなわち、ポインタ）は、ローカル・アドレス・バス６０４上に配置され、ＯＲＲ８０６に転送される。次いで、状態機械８１２は、ＯＰＱ５２０からデータをＯＲＲ８０６内にラッチするラッチ＿ＯＲＲ８３４を発生し、またＯＰＴＰＲ８２６内に記憶されたＯＰＱ５２０の尾部ポインタを増分するＩｎｃｒｅｍｅｎｔ＿ＯＦＴＰＲ信号８４４を発生する。このプリフェッチが完了し、新しいデータがＯＲＲ８０６内にラッチされた後、状態機械８１２は、ＰＣＩバス２０８からの他のトランザクションを完了する準備ができていることをＡＴＵ２１８に通知する前にＯＲＲ＿Ｒｅａｄｙ信号８１４をアサートする。

状態機械８１２は、Ｒｅａｄ＿ＯＲＰ信号８１６が発生したとき、プライム状態８５６からプリフェッチ状態８５２に遷移し、エンプティ信号８４０はアサートされない。それに応答して、状態機械８１２は、Ｍｅｍｏｒｙ＿Ｒｅａｄ＿Ｒｅｑｕｅｓｔ信号８１８をローカル・バス・アービタ２４０に対してアサートし、ＡＴＵ２１８に対してＯＲＲ＿Ｒｅａｄｙ信号８１４をアサート解除する。その結果、後のトランザクションは、プリフェッチが完了するまでＯＲＲ８０６の内容を読み取らない。

状態機械８１２は、エンプティ信号８４０がアサートされたときにアサートされたＲｅａｄ＿ＯＲＰ信号を検知したときにプライム状態８５６からエンプティ状態８５０に遷移する。それに応答して、状態機械８１２は、プリセット信号８４２をアサートする。プリセット信号８４２により、ＯＰＱ５２０に読取りを要求しているトランザクションにＯＰＱ

５２０がエンプティであることが通知されるようにＯＲＲ８０６の内容がエンプティ指示に対して予約された値に設定される。
コンパレータ８３６がＯＰＨＰＲ８３８とＯＰＴＰＲ８２６の内容を比較し、各値が等しい場合、エンプティ信号８４０はアサートされる。ｎｏｔ＿Ｅｍｐｔｙ　ＯＰＱ５２０（すなわち、ｎｏｔ＿Ｅｍｐｔｙ）は、ホスト・プロセッサ２０１が処理するために保留されているメッセージがあることを示す。本発明は、ＰＣＩバス仕様リリース２．０に指定されている割込み線を介してホスト・プロセッサ２０１に対して割込みを発生する論理（図示せず）を含んでいる。
第９Ａ図および第９Ｂ図は、アウトバウンド解放状態機械９１２を示す。ホスト・プロセッサ２０１は、メッセージを処理した後、データ・バス９０４を通ってＰＣＩバス２０８を介してフリー・メッセージ・バッファポインタをＡＴＵ２１８に戻し、アウトバウンド解放レジスタ（ＯＲＬＳＲ）９０６内にラッチされる。次いで、フリー・メッセージ・バッファ・ハンドルは、アウトバウンド解放レジスタＯＲＬＳＲ９０６からローカル・データ・バス６０４を介してアウトバウンド・フリー待ち行列（ＯＦＱ）５１０に送られる。フリー・メッセージ・バッファを解放するために、ホスト・プロセッサ２０１は、単に１つのバストランザクションサイクル中にそのフリー・メッセージ・バッファのアドレスをＯＲＬＳＲ９０６に書き込む。
ＡＴＵ２１８は、ＯＲＬＳＲ＿Ｒｅｔｒｙ信号９１６およびＯＲＬＳＲ＿Ｒｅｔｒｙ信号９１４の状態をテストする。ＯＲＬＳＲ＿Ｒｅｔｒｙ信号９１４がアサート解除された場合、ＰＣＩトランザクション（すなわち、ＯＲＬＳＲ９０６への書込み）は完了しない。要求側プロセスは再試行を通知され、要求側プロセスは、バスの制御権を解放し、後で再度試行する。ＯＲＬＳＲ＿Ｒｅｔｒｙ信号９１６がアサート解除された場合、ＡＴＵ２１８は、ＯＲＬＳＲ＿Ｒｅｔｒｙ信号９１６がアサートされるまで待ち状態を挿入する。ＯＲＬＳＲ＿Ｒｅｔｒｙ信号９１６がアサートされたとき、ＡＴＵ２１８は、状態機械９１２に対してＷｒｉｔｅ＿ＯＲＬＳＲ信号９１８を発生し、データをＯＲＬＳＲ９０６内にラッチする。
第９Ｂ図は、アウトバウンド解放状態機械９１２の状態図を示す。状態機械９１２は、フル状態９５４、アイドル状態９５０、ポスト状態９５２の３つの状態を有する。状態機械９１２は、フル信号９４０がコンパレータ９３６によってアサートされたときアイドル状態９５０からフル状態９５４に遷移する。このフル信号９４０に応答して、状態機械９１２は、ＡＴＵ２１８に対してＯＲＬＳＲ＿Ｒｅｔｒｙ信号９１４をアサートする。ＯＲＬＳＲ＿Ｒｅｔｒｙ信号９１４が発生したとき、ＯＲＬＳＲ９０６に対する書込みトランザクションを示すプロセスは、後で再度試行するよう通知される。
状態機械９１２は、フル信号９４０がアサート解除されたときフル状態９５４からアイドル状態９５０に遷移する。アウトバウンド・フリー待ち行列ＯＦＱ５１０がフルでない場合、状態機械９１２は、ＯＲＬＳＲ＿Ｒｅｔｒｙ信号９１４をアサート解除する（すなわち、ＯＦＱ５１０内に追加のフリー・メッセージ・ハンドル用の空きがある）。
状態機械９１２は、ＡＴＵ２１８からＷｒｉｔｅ＿ＯＲＬＳＲ信号９１８を受信したときアイドル状態９５０からポスト状態９５２に遷移する。Ｗｒｉｔｅ＿ＯＲＬＳＲ信号９１８はまた、フリー・メッセージ・ハンドルをＯＲＬＳＲ９０６内にラッチする役目をする。Ｗｒｉｔｅ＿ＯＲＬＳＲ信号９１８がアサートされたことに応答して、状態機械９１２は、ローカル・バス・アービタ２４０に対してＭｅｍｏｒｙ＿Ｗｒｉｔｅ＿Ｒｅｑｕｅｓｔ信号９１８をアサートする。アービタからの許可信号９３２を待つ。次のフリー・メッセージ・ハンドルを書き込むべきＯＦＱ５１０内の次の位置を計算する。状態機械９１２はまた、後のトランザクションがＯＲＬＳＲ９０６内にラッチされているデータを上書きするのを防ぐためにＯＲＬＳＲ＿Ｒｅｔｒｙ信号９１６をアサート解除する。
状態機械９１２は、ローカル・バス・アービタ２４０から許可信号９３２を受信したときポスト状態９５２からアイドル状態９５０に遷移する。それに応答して、アウトバウンド解放状態機械９１２は、Ｉｎｃｒｅｍｅｎｔ＿ＯＦＨＰＲ信号９４４を介してＯＦＨＰＲ９２６内の頭部ポインタを増分する。状態機械９１２はまた、すでにＯＲＬＳＲ９０６の

内容を記憶していること、および次のフリー・メッセージ・ハンドルを記憶すべきＯＦＱアドレスを計算したことをＡＴＵ２１８に示すＯＲＬＳＲ＿作業可能信号９１６をアサートし、ＯＲＬＳＲ９０６への次の書込みの準備ができる。

要約すると、ホスト・プロセッサは、そのハンドルをＯＲＬＳＲ９０６に書き込むことによってＯＦＱ５１０に対してフリー・メッセージ・バッファを解放する。ＯＦＱ５１０がフルである場合、要求側プロセスは、後で再試行するよう通知される。ＯＦＱ５１０がフルでない場合、フリー・メッセージ・バッファのハンドルはＯＲＬＳＲ９０６内にラッチされる。次いで、状態機械９１２は、ローカル・バスへのアクセス権を得るためにローカル・バス・アービタ２４０からの許可信号９３２を待つ。ローカル・バスの制御権が付与された後、状態機械９１２は、前に計算した頭部ポインタ／アドレスによって示された位置においてＯＲＬＳＲ９０６内にラッチされたデータをＯＦＱ５１０に転送する。

以上、遠隔プロセスがセマフォの使用またはバスのロックなしにメッセージ・バッファを割り振り、次いでメッセージ・バッファを作業待ち行列にポストすることを可能にする方法および装置について説明した。

さらに、作業待ち行列からメッセージを取り出し、メッセージがホスト・プロセッサによって処理された後でメッセージをフリー待ち行列に解放する方法および装置について説明した。

本発明はまた、スケーラビリティ、フレキシビリティ、および他のプラットフォームとの互換性を提供する。例えば、上述のように、インバウンドメッセージ待ち行列を含むすべてのプラットフォームは、プロセッサ間メッセージを容易に送ることができる。インバウンドメッセージ待ち行列を実施しない他のプラットフォームとの互換性については、プラットフォームのハードウェアを修正することなくアウトバウンド・メッセージ待ち行列がそのプラットフォームに同等の機能を供給する。さらに、本発明は、他のプロセッサがコンピュータシステム内に存在することを明確に知ることなく、他のプラットフォームが１つのプラットフォームのインバウンド待ち行列を同時に使用できるという抽象を可能にする。

したがって、本発明は、非対称マルチプロセッサ・システム内のプロセッサに対するハードウェア修正を必要とすることなくプロセッサ間で非常に効率的な形でメッセージを直接送る方法および装置を提供する。

以上、本発明について、その特定の例示的な実施形態に関して説明した。しかしながら、下記の請求の範囲に記載されている本発明のより広い精神および範囲から逸脱することなく本発明に様々な修正および変更を加えることができることが明らかであろう。したがって、明細書および図面は、限定的なものではなく、例示的なものと考えられたい。

【図１】

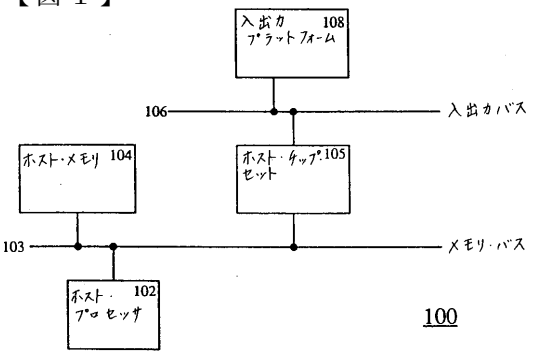


FIG. 1

【図２】

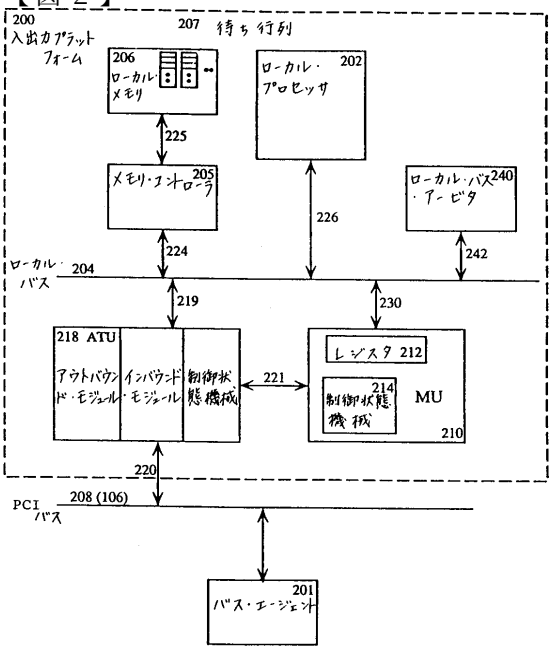


FIG. 2

【図３】

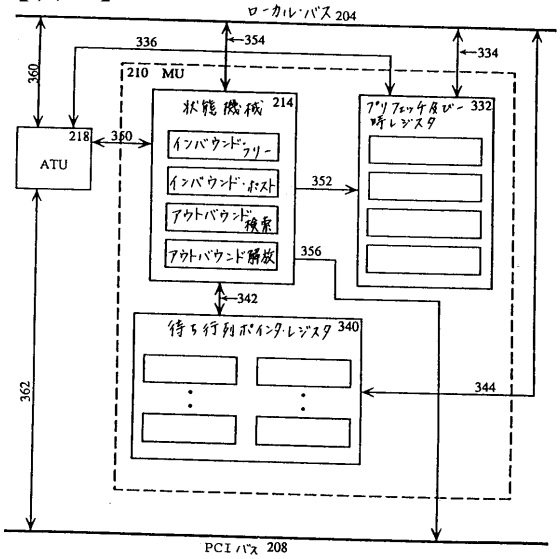


FIG. 3

【図４】

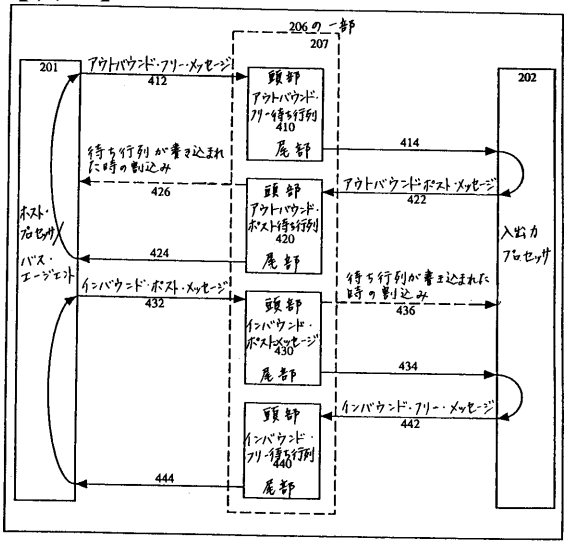


FIG. 4

【図５】

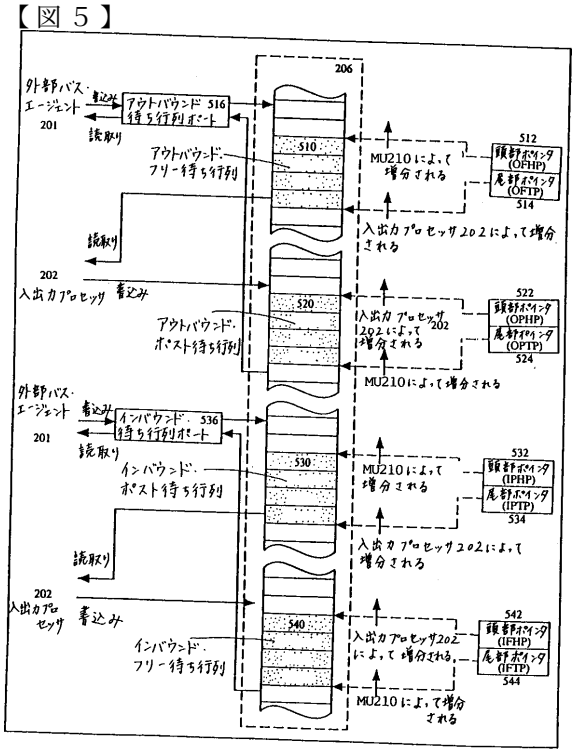


FIG. 5

【図６Ａ】

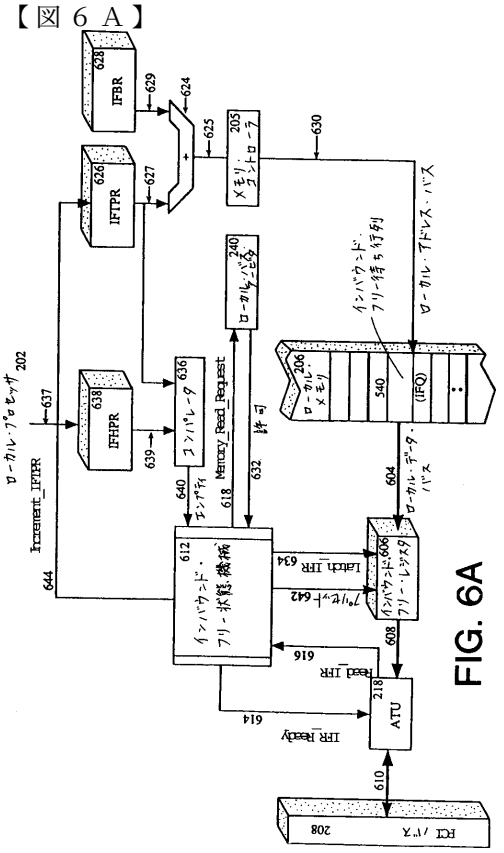


FIG. 6A

【図６Ｂ】

インバウンド・フリー状態機械の状態図

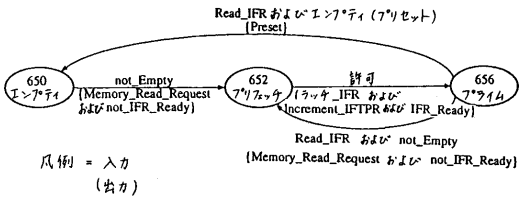


FIG. 6B

【図７Ａ】

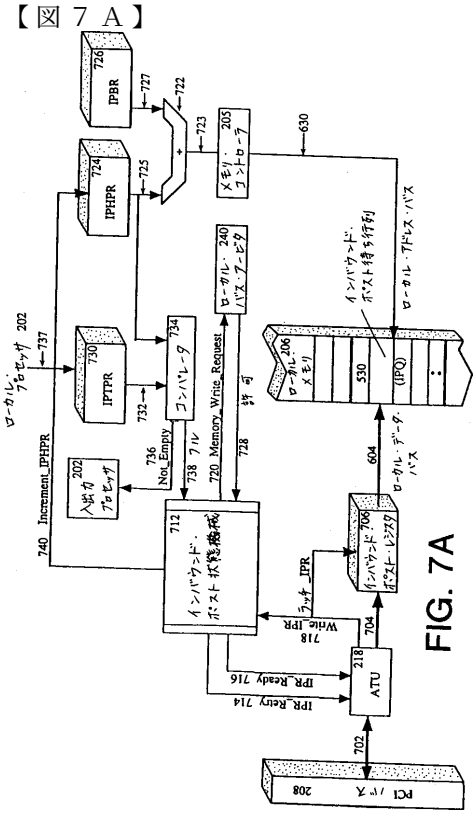
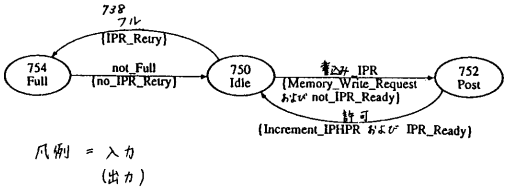


FIG. 7A

【図７Ｂ】
インバウンド・ポスト状態機械の状態図



凡例 = 入力
(出力)

FIG. 7B

【図８Ａ】

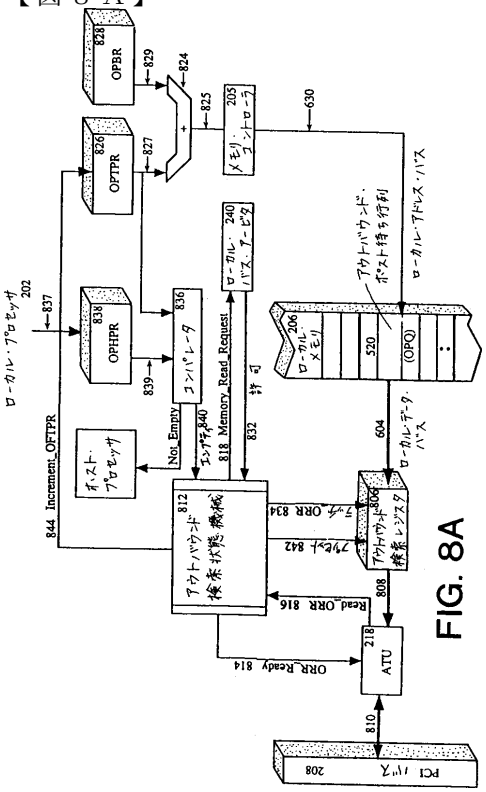
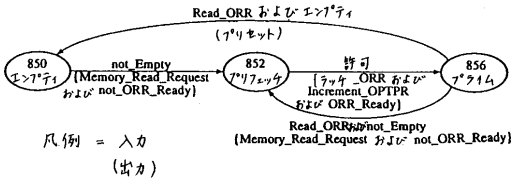


FIG. 8A

【図８Ｂ】
アウドバウンド検索状態機械の状態図



凡例 = 入力
(出力)

FIG. 8B

【図９Ａ】

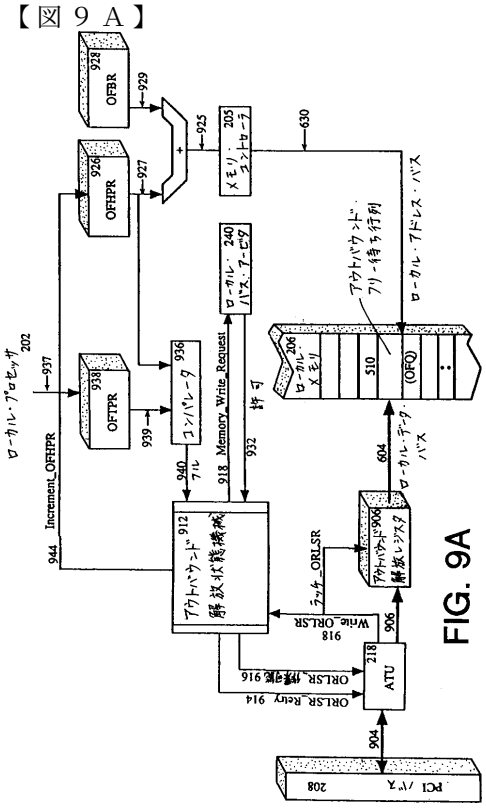
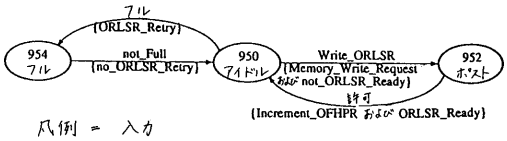


FIG. 9A

【図９Ｂ】
アウトバウンド解放状態機械の状態図



凡例 ＝ 入力
（出力）

FIG. 9B

フロントページの続き

(72)発明者　デイビス，バリー
アメリカ合衆国・８５２２４・アリゾナ州・チャンドラー・ウエスト　サラゴサ　ストリート・７４０
(72)発明者　フュトラル，ウイリアム・ティ
アメリカ合衆国・９７２２９・オレゴン州・ポートランド・ノースウエスト　エルク　ラン　ドライブ・１７７１５
(72)発明者　ガーバス，エリオット
アメリカ合衆国・８５２５７・アリゾナ州・スコッツデイル・ノース　ヘイデン　ロード・２７００・３１０６番

審査官　鳥居　稔

(56)参考文献　特開平０５－２８２１６６（ＪＰ，Ａ）
特開平０２－１５８８５８（ＪＰ，Ａ）
特開平０５－２６８２９１（ＪＰ，Ａ）
特表平１１－５１３１５０（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F 13/10
G06F 13/38
G06F 15/16 - 177