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

特許第5595635号
(P5595635)

(45)発行日　平成26年9月24日(2014.9.24)　(24)登録日　平成26年8月15日(2014.8.15)

(51)Int.Cl.　　F I
G06F 13/14 (2006.01)　　G06F 13/14 310E
G06F 11/18 (2006.01)　　G06F 11/18 310G
G06F 13/00 (2006.01)　　G06F 13/00 301K

請求項の数 9　(全 14 頁)

(21)出願番号　特願2007-163944 (P2007-163944)
(22)出願日　平成19年6月21日 (2007.6.21)
(65)公開番号　特開2008-9980 (P2008-9980A)
(43)公開日　平成20年1月17日 (2008.1.17)
審査請求日　平成22年3月23日 (2010.3.23)
審判番号　不服2013-6821 (P2013-6821/J1)
審判請求日　平成25年4月15日 (2013.4.15)
(31)優先権主張番号　11/426592
(32)優先日　平成18年6月27日 (2006.6.27)
(33)優先権主張国　米国 (US)

(73)特許権者　390009531
インターナショナル・ビジネス・マシーンズ・コーポレーション
ＩＮＴＥＲＮＡＴＩＯＮＡＬ　ＢＵＳＩＮＥＳＳ　ＭＡＣＨＩＮＥＳ　ＣＯＲＰＯＲＡＴＩＯＮ
アメリカ合衆国１０５０４　ニューヨーク州　アーモンク　ニュー　オーチャード　ロード
(74)代理人　100108501
弁理士　上野　剛史
(74)代理人　100112690
弁理士　太佐　種一

最終頁に続く

(54)【発明の名称】入出力（Ｉ／Ｏ）ファブリック内のキューをクリアする方法、Ｉ／Ｏファブリック・エラーを処理する方法及びコンピュータ・プログラム製品（エラー回復のためにＩ／Ｏファブリックのロッ

(57)【特許請求の範囲】
【請求項１】
入出力（Ｉ／Ｏ）ファブリック内のキューをクリアする方法であって、
前記Ｉ／Ｏファブリックは、各々がＣＰＵおよびメモリを備える複数のデータ処理ノードと、複数のＩ／Ｏアダプタとの間に設けられ、前記複数のデータ処理ノードの少なくとも１つから受け取るダイレクト・メモリー・アクセス（ＤＭＡ）コマンドを前記複数のＩ／Ｏアダプタの少なくとも１つに送ることにより、前記複数のデータ処理ノードと前記複数のＩ／Ｏアダプタとの間の通信を可能にし、
前記Ｉ／Ｏファブリック内の前記キューがデッドロックされていることを検出するステップと、
前記複数のデータ処理ノードによる前記Ｉ／Ｏファブリック内の前記キューへのアクセスを使用禁止にするステップと、
前記キュー内のエントリーをクリアするステップと、
前記キュー内のエントリーのクリア後に前記複数のデータ処理ノードによる前記キューへのアクセスを再び使用可能にするステップと、を含み、
前記キューがデッドロックされていることを検出する前記ステップは、
少なくとも１つのクレジットのカウントを維持するステップと、
前記少なくとも１つのクレジットのうちのいずれか１つの前記カウントがゼロであるか否か判定し、該判定の結果が肯定的である場合タイマーを開始させるステップと、
前記判定の結果が肯定的である間に、前記タイマーの開始の前に前記タイマーにロー

ドされた初期タイマー値がゼロまでディクリメントされたときに、前記キューがデッドロックされているものと検出するステップと、を含む方法。

【請求項２】

少なくとも１つのクレジットのカウントを維持する前記ステップは、

ポステッド・リクエスト・クレジットのカウントを維持するステップと、

ノンポステッド・リクエスト・クレジットのカウントを維持するステップと、

コンプリーション・クレジットのカウントを維持するステップと、

を含む、請求項１に記載の方法。

【請求項３】

前記タイマーはプログラマブルである、請求項１または２に記載の方法。

【請求項４】

前記タイマーが開始された後、前記ポステッド・リクエスト・クレジットのカウント、前記ノンポステッド・リクエスト・クレジットのカウント、及び前記コンプリーション・クレジットのカウントのうちのいずれか１つがゼロであるか否かに関して別の判定が行われ、該別の判定の結果が肯定的である場合に前記タイマーがディクリメントされる、請求項１に記載の方法。

【請求項５】

前記別の判定の結果が肯定的でない場合に前記タイマーがストップされる、請求項４に記載の方法。

【請求項６】

前記Ｉ／ＯファブリックはＰＣＩエクスプレスＩ／Ｏファブリックである、請求項１～５のいずれか１項に記載の方法。

【請求項７】

ルート・コンプレックスによってＩ／Ｏファブリックのエラーを処理する方法であって、

前記Ｉ／Ｏファブリックは、各々がＣＰＵ、メモリおよびルート・コンプレックスを備える複数のデータ処理ノードと、複数のＩ／Ｏアダプタとの間に設けられ、前記複数のデータ処理ノードの少なくとも１つから受け取るダイレクト・メモリー・アクセス（ＤＭＡ）コマンドを前記複数のＩ／Ｏアダプタの少なくとも１つに送ることにより、前記複数のデータ処理ノードと前記複数のＩ／Ｏアダプタとの間の通信を可能にし、

前記Ｉ／Ｏファブリックにおけるエラーを検出するステップと、

前記Ｉ／Ｏファブリック内の故障箇所に関連付けられた少なくとも１つのＩ／Ｏアダプタに関連付けられた少なくとも１つのデバイス・ドライバを使用禁止にするステップと、

メモリー・マップドＩ／Ｏ オペレーションを使用可能にするステップと、

前記使用禁止にされた少なくとも１つのデバイス・ドライバを再使用可能にするステップと、を含み、

前記Ｉ／Ｏファブリックにおけるエラーを検出するステップは、前記Ｉ／Ｏファブリック内のキューがデッドロックされていることを検出するステップを含み、当該デッドロックされていることを検出するステップは、

少なくとも１つのクレジットのカウントを維持するステップと、

前記少なくとも１つのクレジットのうちのいずれか１つの前記カウントがゼロであるか否か判定し、該判定の結果が肯定的である場合タイマーを開始させるステップと、

前記判定の結果が肯定的である間に、前記タイマーの開始の前に前記タイマーにロードされた初期タイマー値がゼロまでディクリメントされたときに、前記キューがデッドロックされているものと検出するステップと、を含む方法。

【請求項８】

Ｉ／Ｏファブリック内のエラーを処理する方法であって、

前記Ｉ／Ｏファブリックは、各々がＣＰＵおよびメモリを備える複数のデータ処理ノードと、複数のＩ／Ｏアダプタとの間に設けられ、前記複数のデータ処理ノードの少なくとも１つから受け取るダイレクト・メモリー・アクセス（ＤＭＡ）コマンドを前記複数のＩ

／Ｏアダプタの少なくとも１つに送ることにより、前記複数のデータ処理ノードと前記複数のＩ／Ｏアダプタとの間の通信を可能にし、
前記Ｉ／Ｏファブリック内の一箇所のデッドロック故障を検出するステップと、
前記Ｉ／Ｏファブリックをパワー・ダウンすること無く、前記Ｉ／Ｏファブリック内の一箇所のデッドロック故障から回復するステップを含み、
前記Ｉ／Ｏファブリック内の一箇所のデッドロック故障を検出するステップは、前記Ｉ／Ｏファブリック内のキューがデッドロックされていることを検出するステップを含み、当該デッドロックされていることを検出するステップは、
少なくとも１つのクレジットのカウントを維持するステップと、
前記少なくとも１つのクレジットのうちのいずれか１つの前記カウントがゼロであるか否か判定し、該判定の結果が肯定的である場合タイマーを開始させるステップと、
前記判定の結果が肯定的である間に、前記タイマーの開始の前に前記タイマーにロードされた初期タイマー値がゼロまでディクリメントされたときに、前記キューがデッドロックされているものと検出するステップと、を含む方法。
【請求項９】
Ｉ／Ｏファブリック内のエラーを処理するためのコンピュータ・プログラムであって、
前記Ｉ／Ｏファブリックは、各々がＣＰＵおよびメモリを備える複数のデータ処理ノードと、複数のＩ／Ｏアダプタとの間に設けられ、前記複数のデータ処理ノードの少なくとも１つから受け取るダイレクト・メモリー・アクセス（ＤＭＡ）コマンドを前記複数のＩ／Ｏアダプタの少なくとも１つに送ることにより、前記複数のデータ処理ノードと前記複数のＩ／Ｏアダプタとの間の通信を可能にし、コンピュータに、
前記Ｉ／Ｏファブリック内の一箇所のデッドロック故障を検出するステップと、
前記Ｉ／Ｏファブリックをパワー・ダウンすること無く、前記Ｉ／Ｏファブリック内の一箇所のデッドロック故障から回復するステップと、を実行させることにより前記エラーを処理し、
前記Ｉ／Ｏファブリック内の一箇所のデッドロック故障を検出するステップは、前記Ｉ／Ｏファブリック内のキューがデッドロックされていることを検出するステップを含み、当該デッドロックされていることを検出するステップは、
少なくとも１つのクレジットのカウントを維持するステップと、
前記少なくとも１つのクレジットのうちのいずれか１つの前記カウントがゼロであるか否か判定し、該判定の結果が肯定的である場合タイマーを開始させるステップと、
前記判定の結果が肯定的である間に、前記タイマーの開始の前に前記タイマーにロードされた初期タイマー値がゼロまでディクリメントされたときに、前記キューがデッドロックされているものと検出するステップと、を実行させるコンピュータ・プログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、Ｉ／Ｏファブリックを通してのホスト・コンピュータと入出力（Ｉ／Ｏ）アダプタとの間の通信に関する。より具体的には、本発明は、Ｉ／Ｏファブリックが該ファブリック内の一箇所での故障の故に輻輳し或いはデッドロックした状態に関する。特に、本発明は、ＰＣＩエクスプレス・ファブリック内の一箇所がクレジットを戻すことができなくて該ファブリックがロックアップ又はデッドロックし、最早それを通してＩ／Ｏオペレーションを動かせない場合のＰＣＩエクスプレス・プロトコルに関する。
【背景技術】
【０００２】
（オレゴン州ビーバートンのＰＣＩ－ＳＩＧにより定められた）ＰＣＩエクスプレス仕様は、空のバッファーに関連するクレジットがリンクの他端に与えられるリンク動作を詳述している。例えばバッファーが全くクリアされないことに起因してリンクの他端がクレジットを戻せなければ、オペレーションに関する順序付け要件に起因して、バッファーは全てのコンポーネントにおいてルート・コンプレックスまでいっぱいになって、ルート・

コンプレックスがそれらのＩ／Ｏサブ・システムにアクセスできなくなる可能性がある。ＰＣＩエクスプレス仕様は、この状態でハードウェアについて何が期待されるかを詳述していない。この様な状態では、ファブリックと、そのファブリックに取り付けられた１つ又は複数のルート・コンプレックスとはエラーをクリアするためにパワーダウンされ再びバックアップする必要があると期待される。

【発明の開示】

【発明が解決しようとする課題】

【０００３】

本発明は、Ｉ／Ｏファブリック又は該ファブリックに取り付けられたどのルート・コンプレックスをもパワーダウンせずに該Ｉ／Ｏファブリックを回復させることを可能にするコンピュータにより実行される方法及びメカニズムを提供することを目的とする。

【課題を解決するための手段】

【０００４】

実施例は、Ｉ／Ｏファブリック又は該ファブリックに取り付けられたどのルート・コンプレックスをもパワーダウンせずに該Ｉ／Ｏファブリックを回復させることを可能にするコンピュータにより実行される方法及びメカニズムを詳述する。具体的には、実施例はＰＣＩエクスプレスＩ／Ｏファブリックに関連するが、これが他の類似Ｉ／Ｏファブリックに応用され得ることを当業者は認めるであろう。

【０００５】

バッファー／キューを満ちさせることによりルート・コンプレックスにそのＩ／Ｏサブ・システムへのアクセスを失わせる故障に起因して末期的に輻輳し又はデッドロックしたＩ／Ｏファブリックの回復のためのコンピュータにより実行される方法及びメカニズムが提供される。末期的に輻輳又はデッドロックした送信キューが検出されると、そのキューの中の各アイテムが調べられて適宜に処理される間、他のルート・コンプレックスによるその様なキューへのアクセスは保留される。該キュー中の格納リクエストとＤＭＡ読み出し応答パケットとは捨てられ、該キュー中のロード・リクエストは、特別の完了パッケージを戻すことによって処理される。その後、ルート・コンプレックスによる該キューへのアクセスが再開される。

【０００６】

実施例を特徴付けると考えられる新規特徴は特許請求の範囲の欄に記載されている。しかし、実施例自体は、好ましい使用モード、その更なる目的及び利点と同じく、添付図面と関連させて実施例に関する以下の詳細な説明を参照することにより最善に理解されるであろう。

【発明を実施するための最良の形態】

【０００７】

本書に記載される実施例は、Ｉ／Ｏファブリックがリンクの他方の端でのリソース・アベイラビリティーを宣伝するためにクレジットのようなメッセージを使用する任意の汎用又は特別目的計算システムに適用される。より具体的には、本書で以下に記載されている好ましい実施態様は、ＰＣＩエクスプレスＩ／Ｏリンクを使用するインプリメンテーションを提供する。

【０００８】

図面の図１を特に参照すると、実施例による分散型計算システム１００の図が描かれている。図１に示されている分散型計算システムは、Ｉ／Ｏリンク１１０，１２０，１３０，１４２、及び１４３を通してＩ／Ｏファブリック１４４に取り付けられ、またルート・ノード（ＲＮ）１６０－１６３のメモリー・コントローラ１０４，１１４，１２４、及び１３４に取り付けられた１つ以上のルート・コンプレックス（ＲＣ）１０８，１１８，１２８，１３８、及び１３９の形をとっている。該Ｉ／Ｏファブリックは、リンク１５１－１５８を通してＩ／Ｏアダプタ（ＩＯＡ）１４５－１５０に取り付けられている。該ＩＯＡは、１４５－１４６及び１４９のような単機能ＩＯＡ又は１４７－１４８及び１５０のような多機能ＩＯＡであり得る。更に、該ＩＯＡは、１４５－１４８のように単一のリン

クを介して、或いは１４９－１５０のように冗長性を目的として複数のリンクで、Ｉ／Ｏファブリックに接続され得る。

【０００９】

ＲＣ１０８，１１８，１２８，１３８、及び１３９の各々は、夫々のＲＮ１６０－１６３の一部である。ＲＮ１６３のように１つのＲＮあたりに２つ以上のＲＣがあり得る。ＲＣの他に、各ＲＮは、１つ以上の中央処理装置（ＣＰＵ）１０１－１０２，１１１－１１２，１２１－１２２，１３１－１３２と、メモリー１０３，１１３，１２３、及び１３３と、メモリー・コントローラ１０４，１１４，１２４、及び１３４とから成っており、該メモリー・コントローラは該ＣＰＵ、メモリー、及びＩ／Ｏ　ＲＣを接続して、メモリーのためにコヒーレンシー・トラフィックを処理するような機能を実行する。

【００１０】

複数のＲＮが、夫々のメモリー・コントローラ１０４及び１１４を介して１５９で互いに接続されて１つのコヒーレンシー・ドメインを形成して単一の対称多重処理（ＳＭＰ）システムとして動作することができ、或いはＲＮ１６２－１６３のように別々のコヒーレンシー・ドメインを有する独立のノードであり得る。

【００１１】

コンフィギュレーション・マネージャ１６４は、単独にＩ／Ｏファブリック１４４に取り付けられ得（図１に示されているように）、或いはＲＮ１６０－１６３のうちの１つの一部分であり得る。該コンフィギュレーション・マネージャは、Ｉ／Ｏファブリックの共有されるリソースを設定し、リソースをＲＮに割り当てる。

【００１２】

分散型計算システム１００は、種々の商業的に入手し得るコンピュータ・システムを用いて実現され得る。例えば、分散型計算システム１００はＩＢＭ社から入手し得るＩＢＭ　ｅＳｅｒｖｅｒ　ｉＳｅｒｉｅｓ（ＩＢＭ社の商標）モデル８４０システムを用いて実現され得る。その様なシステムは、ＩＢＭ社から入手し得るＯＳ／４００（ＩＢＭ社の登録商標）オペレーティング・システムを用いて論理分割化をサポートすることができる。

【００１３】

当業者は、図１に描かれているハードウェアが変化し得ることを理解するであろう。例えば、光ディスク・ドライブなどのような他の周辺装置が、描かれているハードウェアに加えて又はその代わりとして、使用され得る。図示されている例は、実施例に関する構造上の制限を示唆するように意図されてはいない。

【００１４】

図２を参照すると、実施例を実現し得る代表的な論理分割プラットフォームのブロック図が描かれている。論理分割プラットフォーム２００のハードウェアは、例えば、図１の分散型計算システム１００として実現され得る。論理分割プラットフォーム２００は、分割ハードウェア２３０と、オペレーティング・システム（ＯＳ）２０２，２０４，２０６，２０８と、プラットフォーム・ファームウェア２１０とを含む。オペレーティング・システム２０２，２０４，２０６、及び２０８は、単一のオペレーティング・システムの複数のコピー、又は論理分割プラットフォーム２００上で同時に動作する複数の異種オペレーティング・システムであり得る。これらのオペレーティング・システムはＯＳ／４００（登録商標）オペレーティング・システムを用いて実現され得、それらはハイパーバイザー（Ｈｙｐｅｒｖｉｓｏｒ）のようなプラットフォーム又は分割管理ファームウェアとインターフェースするように設計されている。ＯＳ／４００（ＩＢＭ社の登録商標）オペレーティング・システムは、これらの実施例における単なる例として用いられるに過ぎない。具体的インプリメンテーションによりＡＩＸ（登録商標）オペレーティング・システム及びＬｉｎｕｘ（登録商標）オペレーティング・システムのような他のタイプのオペレーティング・システムも使用され得る（ＡＩＸは米国及び他の国におけるＩＢＭ社の登録商標であり、Ｌｉｎｕｘは米国及び他の国におけるリーナス・トーバルズの登録商標である）。オペレーティング・システム２０２，２０４，２０６、及び２０８はパーティション２０３，２０５，２０７、及び２０９に夫々置かれている。ハイパーバイザー・ソフトウ

ェアは、プラットフォーム・ファームウェア２１０を実現するために使用され得るソフトウェアの例であって、ＩＢＭ社から入手可能である。ファームウェアは、例えば、読み出し専用メモリー（ＲＯＭ）、プログラマブルＲＯＭ（ＰＲＯＭ）、消去可能プログラマブルＲＯＭ（ＥＰＲＯＭ）、電気的消去可能ＰＲＯＭ（ＥＥＰＲＯＭ）、及び不揮発性ランダム・アクセス・メモリー（不揮発性ＲＡＭ）のような、電力無しにその内容を保持するメモリー・チップに格納された“ソフトウェア”である。

【００１５】

更に、パーティション２０３，２０５，２０７、及び２０９はパーティション・ファームウェア２１１，２１３，２１５、及び２１７も夫々含んでいる。パーティション・ファームウェア２１１，２１３，２１５、及び２１７は、イニシャル・ブート・ストラップ・コード、ＩＥＥＥ－１２７５標準規格オープン・ファームウェア及びＩＢＭ社から入手可能なランタイム・アブストラクション・ソフトウェア（ＲＴＡＳ）を用いて実現され得る。パーティション２０３，２０５，２０７、及び２０９がインスタンシエートされるとき、プラットフォーム・ファームウェア２１０によってブート・ストラップ・コードのコピーがパーティション２０３，２０５，２０７、及び２０９にロードされる。その後、コントロールは該ブート・ストラップ・コードに移され、該ブート・ストラップ・コードはその後に該オープン・ファームウェア及びＲＴＡＳをロードする。その後、パーティションに関連付けられ又は割り当てられているプロセッサは、パーティション・ファームウェアを実行するべくパーティションのメモリーにディスパッチされる。

【００１６】

分割ハードウェア２３０は、複数のプロセッサ２３２－２３８と、複数のシステム・メモリー・ユニット２４０－２４６と、複数のＩＯＡ２４８－２６２と、ＮＶＲＡＭ記憶装置２９８と、記憶ユニット２７０とを含む。プロセッサ２３２－２３８と、メモリー・ユニット２４０－２４６と、ＮＶＲＡＭ記憶装置２９８と、ＩＯＡ２４８－２６２との各々、又はそれらの部分は、論理分割プラットフォーム２００内の複数のパーティションのうちの１つに割り当てられ得、その各々はオペレーティング・システム２０２，２０４，２０６、及び２０８のうちの１つに対応する。

【００１７】

プラットフォーム・ファームウェア２１０は、論理分割プラットフォーム２００の分割を生じさせ実施するべくパーティション２０３，２０５，２０７、及び２０９のために幾つかの機能及びサービスを実行する。プラットフォーム・ファームウェア２１０は、基礎をなすハードウェアと同一の、ファームウェアにより実現される仮想マシンである。従って、プラットフォーム・ファームウェア２１０は、論理分割プラットフォーム２００のハードウェア・リソースをバーチャル化することによって独立のＯＳイメージ２０２，２０４，２０６、及び２０８の同時実行を可能にする。

【００１８】

サービス・プロセッサ２９０は、パーティションにおけるプラットフォーム・エラーの処理のような種々のサービスを提供するために使用され得る。それらのサービスは、ＩＢＭ社のようなベンダーにエラーを報告するサービス・エージェントとしても動作することができる。複数の異なるパーティションの動作は、ハードウェア管理コンソール２８０のようなハードウェア管理コンソールを通して制御され得る。ハードウェア管理コンソール２８０は、異なるパーティションへのリソースの再配分を含む種々の機能をシステム管理機能がそれから実行し得る独立の分散型計算システムである。

【００１９】

論理分割（ＬＰＡＲ）環境においては、一パーティションのリソース又はプログラムが他のパーティションの動作に影響を及ぼすことは許されない。更に、有益であるために、リソースの割り当てはきめ細かくなければならない。例えば、１つの特定のＰＣＩホスト・ブリッジ（ＰＨＢ）のもとの全てのＩＯＡを同じパーティションに割り当てることはしばしば許容され得ない。なぜならば、それは、複数のパーティション間でリソースを動的に移す能力を含むシステムのコンフィギュレーション能力を制限するからである。従って

、個々のＩＯＡ又はＩＯＡの部分のようなリソースを別々のパーティションに割り当て得るようにＩＯＡをルート・ノードに接続し、同時に、割り当てられたリソースが他のパーティションのリソースへのアクセスを得るなどして他のパーティションに影響を及ぼすのを防止する何らかの機能がＩ／Ｏファブリック及びルート・コンプレックスにおいて必要とされる。

【００２０】

図３は、Ｉ／ＯパケットをＩ／Ｏファブリック３１４に送るために使用される自分自身の送信キュー３０６及び３０８を各々有する２つのＲＣ３０２－３０４を示す。ＲＣ３０２は実線３２８及び３３０及び点線３３２でＩ／Ｏアダプタ３２４及び３２６と通信するように示され（実線は通信の初期セットを示し、点線はその後の通信を示す）、ＲＣ３０４は破線３３４でＩ／Ｏアダプタ３２６と通信するように示されている。もしＩ／Ｏアダプタ３２４が送信キュー３１６からパケットを受信するのをやめれば（すなわち、送信キュー３１６のための制御論理へクレジットを戻すのをやめれば）、送信キュー３１６が満ちて送信キュー３０６を満ちさせてＩ／Ｏアダプタ３２６への通信３３０を妨げ得る。従って、Ｉ／Ｏアダプタ３２４の破損は、ＲＣ３０２にとってＩ／Ｏアダプタ３２６をも役に立たなくし得る。

【００２１】

同様に、もしＩ／Ｏアダプタ３２６が送信キュー３１８からパケットを受信するのをやめれば（すなわち、送信キュー３１８のための制御論理へクレジットを戻すのをやめれば）、送信キュー３１８が満ちて送信キュー３０６及び３０８を満ちさせ、該Ｉ／Ｏアダプタと通信する全てのＲＣからの３３２及び３３４のような通信を妨げ得る。従って、Ｉ／Ｏアダプタ３２６の破損は、該Ｉ／Ｏアダプタと通信する全てのＲＣからＩ／Ｏファブリックをロックアップし得、他のＩ／Ｏアダプタへの Ｉ／Ｏ動作も影響を受け得る。これらの実施例が防止しようとしているのはこの破損である。

【００２２】

図４は、送信キュー４０４を制御するキュー制御論理４１１を示す。送信キュー４０４からのパケット４０６の送信は、図３のＩ／Ｏアダプタ３２４又は３２６によるなどの、送信クレジット４０８を戻すリンクの他端に依存する。これらのクレジットは、ポステッド・リクエスト・クレジット・レジスタ４１０、ノンポステッド・リクエスト・クレジット・レジスタ４１２、及びコンプリーション・クレジット・レジスタ４１４によって追跡される。もしこれら３つのレジスタのいずれかが４１６で検出されたようにゼロになれば、ゼロ・クレジット・タイマー４１８は、ゼロ・クレジット・タイマー・イニシャル・レジスタ４２０に格納されている初期値をロードされ、その後、レジスタ４１０－４１４のうちの１つがゼロである間カウント・ダウンし続ける。もしレジスタ４１０－４１４の全てがゼロで無くなれば、ゼロ・クレジット・タイマー４１８はカウント動作を停止する。ゼロ・クレジット・タイマー・イニシャル・レジスタ４２０は固定値であり得或いはシステム・ファームウェア又はソフトウェアを介してプログラマブルであり得るが、プログラマブルであるのが好ましい実施態様である。

【００２３】

ゼロ・クレジット・タイマーがゼロまでカウントダウンしたとき、これはロックアップ状態が検出されたことを示し、それはクリアされなければならない。すなわち、ゼロ・クレジット・タイマーがゼロまでカウントしたとき、これはゼロ・クレジット・タイムアウト制御論理４２２内の停止状態レジスタ４２４のメモリー・マップドＩ／Ｏ（ＭＭＩＯ）ビット４２６及びダイレクト・メモリー・アクセス（ＤＭＡ）ビット４２８をセットする。これが起きると、全ての影響を受けるルート・コンプレックスがエラー・メッセージで信号され、例えばエラー・メッセージ４３０がＩ／Ｏファブリック４０２の主要なバス４３２のうちの１つで信号される。更に、詳しく後述されるように、ロックアップがクリアされる。

【００２４】

図５は、ゼロ・クレジット・タイムアウトが検出されたときのハードウェアによる処理

の流れを示す。流れは、エラーの検出で５０２から始まる。５０４において、カウント動作のための初期値がゼロ・クレジット・タイマー・イニシャル・レジスタからゼロ・クレジット・タイマーにロードされる。５０６において、ゼロ・クレジット・タイマーがゼロであるか否かが調べられ、もし否であれば、処理は５０８に続き、ここでゼロ・クレジット状態がなお存在するか否かが判定される。もし否であれば、プロセスは５１０から退去する。５０８でゼロ・クレジット状態がなお存在するならば、ゼロ・クレジット・タイマー・レジスタが５１２でディクリメントされ、その後５０６で再びゼロについて調べられる。もしゼロ・クレジット・タイマー・レジスタがゼロになれば、５１４でファブリック・ロックアップ処理が開始される。

【００２５】

図６はファブリック・ロックアップ処理を示し、これは６０２から始まる。停止状態レジスタのＭＭＩＯビット及びＤＭＡビットが６０４でハードウェアによりセットされる。ハードウェアは次にルート・コンプレックス６０６に、これらがエラー処理を開始できるように、エラーメッセージを送る。ロックアップ処理の最後のステップ６０８は、問題を検出した送信キューをクリアすることである。これを行うために、送信キュー内の各アイテムが検査されて適宜処理される。すなわち、ＭＭＩＯ格納リクエストは捨てられ、ＭＭＩＯロード・リクエストは、全て強制的に１にされたデータを有する完了パケットを戻すことによって処理され（例えば、該パケットの全ビットがバイナリー“１”の値にセットされる）、ＤＭＡ読み出し応答パケットは捨てられる。これを行うことにより、該送信キューは一時的にクリアされ、エントリーの処理は６１０で完了する。しかし、ファブリック輻輳の原因となっているトランザクションが上流側にあるかも知れず、それらは送信キューに流れ下るので、もしそれが起これば図７に示されているように処理が続く。

【００２６】

図７に、新しいエントリーの処理が示されている。停止状態レジスタのＭＭＩＯビット及びＤＭＡビットをセットすることの目的は（図６のステップ６０４）、ソフトウェアがエラー処理を開始して万事を管理された状態にできるまで送信キューをクリアされた状態に保つことである。これは次のように示される。新アイテムが受信７０２され、それがＭＭＩＯロード・オペレーションであるか否かが判定７０４される。もし判定の結果が肯定的であってＭＭＩＯビットが７０６で判定されるように０であるならば、ＭＭＩＯロード・オペレーションは７０８で標準的に処理され、オペレーションは７２６で完了する。もしＭＭＩＯビットが７０６で１にセットされていれば、７１０でロードのために全て１のデータが戻され、オペレーションは７２６で完了する。この全て１のデータは、Ｉ／Ｏサブシステムを検査してエラーが発生しているか否か調べるようにオペレーティング・システム、デバイス・ドライバ、又は他のソフトウェアに信号することができる。

【００２７】

もしこれが７０４で判定されたようにＭＭＩＯロード・オペレーションでなければ、該オペレーションは７１２でＭＭＩＯ格納オペレーションであるか否か調べられる。もし判定の結果が肯定的であって、ＭＭＩＯビットが７１４で０と判定されると、ＭＭＩＯ格納オペレーションが７１６で標準的に処理され、オペレーションは７２６で完了する。もしＭＭＩＯビットが７１４で１にセットされていれば、該格納は７１８で捨てられ、オペレーションは７２６で完了する。

【００２８】

もしこれが７０４、７１２でＭＭＩＯオペレーションでないと判定されたならば、それはＤＭＡ読み出し応答オペレーションでなければならない。この場合、停止状態レジスタのＤＭＡビットが７２０で調べられ、もし０ならば、該ＤＭＡオペレーションは７２２で標準的に処理され、オペレーションは７２６で完了する。最後に、もし７２０で該ＤＭＡビットが１と判定されたならば、ＤＭＡ読み出し完了は７２４で捨てられ、オペレーションは７２６で完了する。

【００２９】

ＤＭＡビットがセットされている間、入ってくる新しいＤＭＡリクエストがあれば、そ

れは適宜処理されなければならない。図８は、それがどの様に行われるかを示す。新ＤＭＡリクエストが８０２で受信され、８０４で停止状態レジスタ８０４のＤＭＡビットが０であるか否かが判定される。もし判定の結果が肯定的ならば、該ＤＭＡは８０６で標準的に処理され、オペレーションは８１０で完了する。

【００３０】

もし８０４でＤＭＡビットが０でなければ、ハードウェアはコンプリーター・アボート（ｃｏｍｐｌｅｔｅｒ　ａｂｏｒｔ）又はサポート無しリクエストを要求側に戻し８０８、オペレーションは８１０で完了する。

【００３１】

ＲＣにおけるファブリック・エラーの処理はプラットフォームに或る程度依存するが、図９は一般的な流れを示す。処理は９０２から始まり、９０４で、送信されたエラー・メッセージの検出により、又は全て１のデータが突然受信されたために、エラーが検出される。９０６で、オペレーティング・システム又はＲＣハードウェアは、Ｉ／Ｏファブリック内の、エラーが検出された箇所より下位であるＩ／Ｏにデバイス・ドライバがそれ以上のオペレーションを発するのをやめさせる。例えば、図３を参照すると、Ｉ／Ｏアダプタ３２４は送信キュー３１６（この例におけるエラーの箇所）の下にあり、もしこの箇所３１６がエラー状態であるか又はデッドロックされていることが検出されたならばＲＣハードウェアはデバイス・ドライバがそれ以上のオペレーションをこのＩ／Ｏアダプタ３２４に発するのをやめさせる。同様に、Ｉ／Ｏアダプタ３２６は送信キュー３１８（この例でのエラーの箇所）の下にあり、もしこの箇所３１８がエラー状態であるか又はデッドロックされていることが検出されたならばＲＣハードウェアはデバイス・ドライバがそれ以上のオペレーションをこのＩ／Ｏアダプタ３２６に発するのをやめさせる。ソフトウェア又はファームウェアはファブリックからエラー情報を読み出し、将来あり得る評価のためにその情報を記録する９０８。プラットフォームはその後プラットフォーム特有のエラー回復を９１０で実行し、もし可能ならばその箇所より下位であるＭＭＩＯオペレーションが続行され得るように９１２で停止状態レジスタのＭＭＩＯビットがクリアされる。９１４で、通信が続行され得るか否かが判定され、もし判定の結果が肯定的ならば９１８でＤＭＡビットがリセットされる。デバイス・ドライバが再開始され、デバイス特有のエラー回復が９２０で実行される。該回復は９２２で完了する。もし９１４で故障箇所より下での通信が再確立され得ないと判定されたならば、９１６で故障箇所より下位であるＩ／Ｏファブリックがリセットされ、９２０でデバイス・ドライバが再開始され、デバイス特有のエラー回復が実行される。回復は９２２で完了する。

【００３２】

本発明は、完全にハードウェアの実施態様、完全にソフトウェアの実施態様、又はハードウェア要素及びソフトウェア要素の両方を含む実施態様の形をとることができる。好ましい実施態様では本発明はソフトウェアで実現され、それはファームウェア、常駐ソフトウェア、マイクロコードなどを含むが、これらに限定はされない。

【００３３】

更に、本発明は、コンピュータ又は命令実行システムにより又はこれらと関連して使用されるプログラム・コードを提供するコンピュータにより使用され得る又はコンピュータにより読み出され得る媒体からアクセス可能なコンピュータ・プログラム製品の形をとることができる。この明細書の目的上、コンピュータにより使用され得る又はコンピュータにより読み出され得る媒体は、該命令実行システム、装置、又はデバイスにより又はこれらと関連して使用されるプログラムを含み、記憶し、伝え、伝播し、又はトランスポートすることのできる任意の有形の装置であり得る。

【００３４】

媒体は、電子、磁性、光学、電磁、赤外線、又は半導体システム（又は装置若しくはデバイス）又は伝播媒体であり得る。コンピュータにより読み出され得る媒体の例は、半導体若しくは固体メモリー、磁気テープ、取り外し可能なコンピュータ・ディスケット、ランダム・アクセス・メモリー（ＲＡＭ）、読み出し専用メモリー（ＲＯＭ）、堅い磁気デ

ィスク及び光ディスクを含む。光ディスクの現在の例は、コンパクト・ディスク－読み出し専用メモリー（ＣＤ－ＲＯＭ）、コンパクトディスク－読み書き（ＣＤ－Ｒ／Ｗ）及びＤＶＤを含む。

【００３５】

プログラム・コードを格納し且つ／又は実行するために適するデータ処理システムは、直接に又はシステム・バスを通して間接的に記憶素子に結合される少なくとも１つのプロセッサを含む。記憶素子は、プログラム・コードの実際の実行時に使用されるローカル・メモリーと、大容量記憶装置と、実行時に大容量記憶装置からコードを検索しなければならない回数を減らすために少なくとも何らかのプログラム・コードの一時記憶を提供するキャッシュ・メモリーとを含み得る。

【００３６】

入出力すなわちＩ／Ｏ装置（キーボード、表示装置、ポインティング・デバイスなどを含むが、これらに限定はされない）は、直接に、又は介在するＩ／Ｏコントローラを通してシステムに結合され得る。

【００３７】

介在する構内ネットワーク又は公衆網を通してデータ処理システムが他のデータ処理システム又は遠隔プリンタ若しくは記憶装置に結合され得るようにネットワーク・アダプタもシステムに結合され得る。モデム、ケーブル・モデム及びイーサネット（登録商標）・カードは、現在利用し得るタイプのネットワーク・アダプタのうちのほんの数例に過ぎない。

【００３８】

本発明の記述は例証及び説明の目的で呈示されているのであって、網羅的であることや、開示された形の発明に限定されることは意図されていない。多くの改変や変形が当業者にとっては明らかであろう。実施態様は、本発明の原理、実用的アプリケーションを最善に説明すると共に、熟考されている特定の用途に適する種々の改変を伴なう種々の実施態様を求めて他の当業者が本発明を理解できるようにするために、選択され記述された。

【図面の簡単な説明】

【００３９】

【図１】実施例に従って描かれた分散型コンピュータ・システムの図である。

【図２】実施例を実現することのできる代表的論理分割プラットフォームのブロック図である。

【図３】１ルート・コンプレックス及び数個のＩ／Ｏアダプタの間、並びに数個のルート・コンプレックス及び１つのＩ／Ｏアダプタの間の通信を示す高レベル図であり、ここでバッファー・ブロッケージは実施例に従って解決される。

【図４】代表的側面が具体化されるキュー制御を示す。

【図５】実施例に従ってロックアップ状態がどのように検出されるかを示すフローチャートである。

【図６】実施例に従うハードウェアによるファブリック・ロックアップ処理を示すフローチャートである。

【図７】実施例に従ってエラーのファームウェア又はソフトウェア処理の間、ファブリックが再びロックアップされるのをハードウェアがどの様に阻止するのかを示すフローチャートである。

【図８】実施例に従ってＩ／Ｏファブリックが回復される過程にある間のＤＭＡ処理を示すフローチャートである。

【図９】実施例によるファブリック・ロックアップ・エラーのルート・コンプレックス処理の高レベル・フローである。

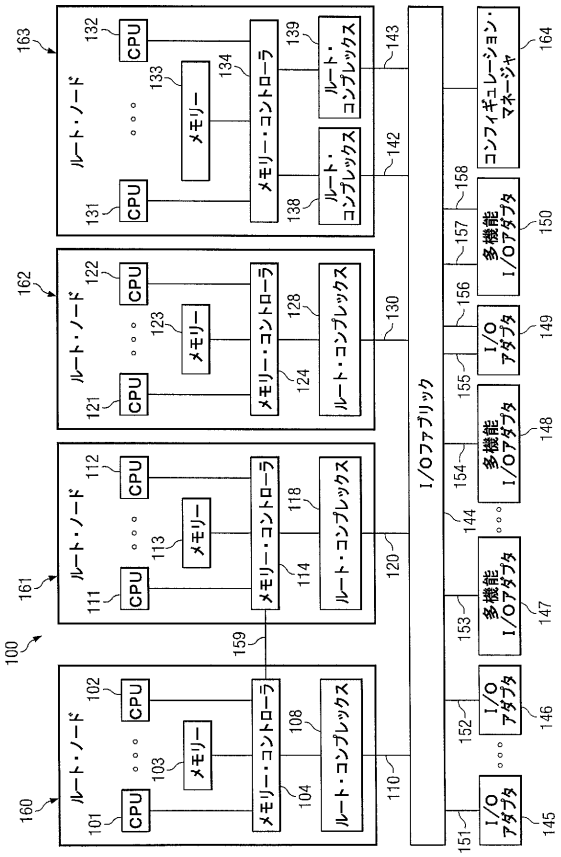
【符号の説明】

【００４０】

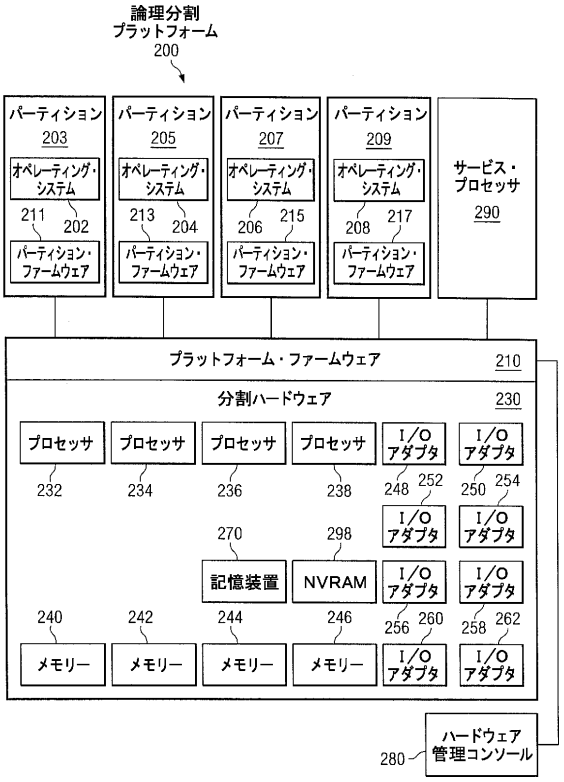
１０８，１１８，１２８，１３８，１３９，３０２，３０４　ルート・コンプレックス
３０６，３０８，３１６，３１８　送信キュー

３１４　Ｉ／Ｏファブリック
１４５－１５０，３２４　Ｉ／Ｏアダプタ
３２６　多機能Ｉ／Ｏアダプタ
３３０，３３２，３３４　通信
１００　分散型計算システム
１１０，１２０，１３０，１４２，１４３　Ｉ／Ｏリンク
１４４　Ｉ／Ｏファブリック
１６０－１６３　ルート・ノード
１０４，１１４，１２４，１３４　メモリーコントローラ
１５０－１５８　リンク
１０１－１０２，１１１－１１２，１２１－１２２，１３１－１３２　中央処理装置
１０３－１１３，１２３，１３３　メモリー
１６４　コンフィギュレーション・マネージャ
２００　論理分割プラットフォーム
２３０　分割ハードウェア
２０２，２０４，２０６，２０８　オペレーティング・システム
２１０　プラットフォーム・ファームウェア
２０３，２０５，２０７，２０９　パーティション
２１１，２１３，２１５，２１７　パーティション・ファームウェア
２３２－２３８　プロセッサ
２４０－２４６　システム・メモリ・ユニット
２４８－２６２　ＩＯＡ
２９８　ＮＶＲＡＭ記憶装置
２７０　記憶ユニット
２９０　サービス・プロセッサ
２８０　ハードウェア管理コンソール
４０４　送信キュー
４１１　キュー制御論理
４０６　パケット
４０８　送信クレジット
４１０　ポステッド・リクエスト・クレジット・レジスタ
４１２　ノンポステッド・リクエスト・クレジット・レジスタ
４１４　コンプリーション・クレジット・レジスタ
４１６　ゼロ・クレジット検出
４１８　ゼロ・クレジット・タイマー
４２０　ゼロ・クレジット・タイマー・イニシャル・レジスタ
４２２　ゼロ・クレジット・タイムアウト制御論理
４２４　停止状態レジスタ
４２６　メモリーマップドＩ／Ｏ
４２８　ダイレクト・メモリー・アクセスビット４２８
４３０　エラー・メッセージ
４０２　Ｉ／Ｏファブリック
４３２　バス

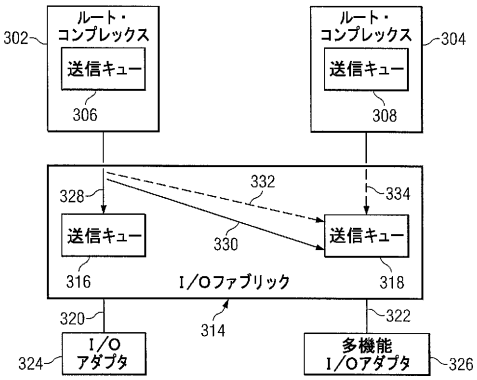
【図１】


ルート・ノード
CPU
メモリー
メモリー・コントローラ
ルート・コンプレックス
I/Oファブリック
コンフィギュレーション・マネージャ
多機能I/Oアダプタ
I/Oアダプタ

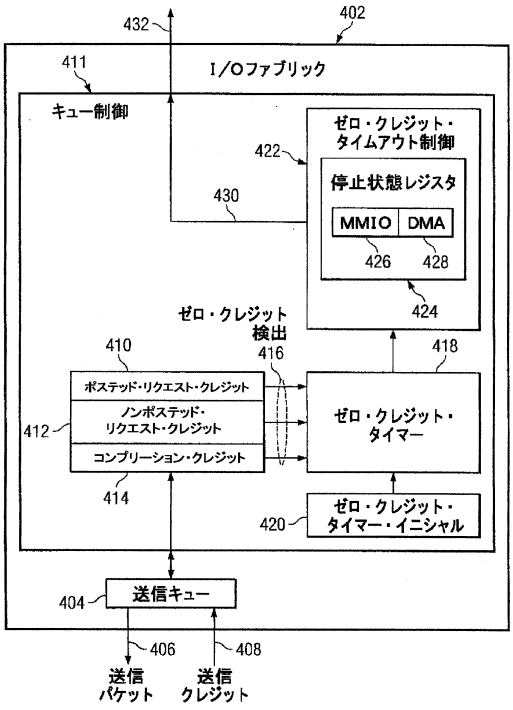
【図２】


論理分割プラットフォーム
200
パーティション
203
205
207
209
オペレーティング・システム
パーティション・ファームウェア
サービス・プロセッサ
290
プラットフォーム・ファームウェア
210
分割ハードウェア
230
プロセッサ
I/Oアダプタ
記憶装置
NVRAM
メモリー
ハードウェア管理コンソール
280

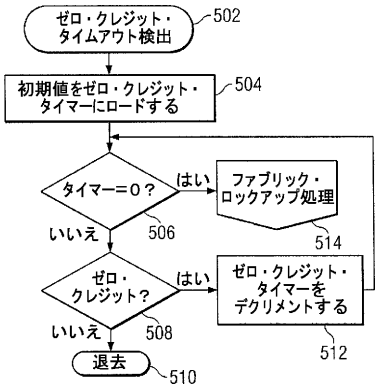
【図３】


ルート・コンプレックス
送信キュー
I/Oファブリック
多機能I/Oアダプタ
I/Oアダプタ

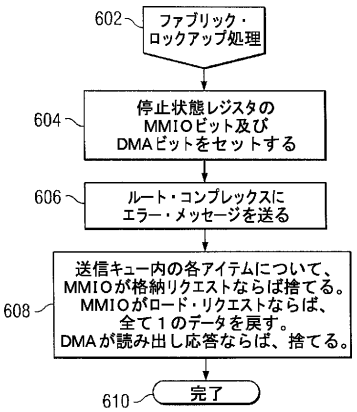
【図４】


I/Oファブリック
キュー制御
ゼロ・クレジット・タイムアウト制御
停止状態レジスタ
MMIO
DMA
ゼロ・クレジット検出
ポステッド・リクエスト・クレジット
ノンポステッド・リクエスト・クレジット
コンプリーション・クレジット
ゼロ・クレジット・タイマー
ゼロ・クレジット・タイマー・イニシャル
送信キュー
送信パケット
送信クレジット

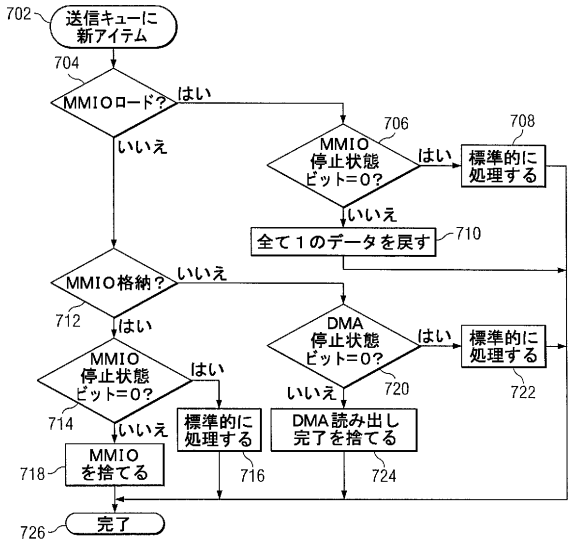
【図５】


ゼロ・クレジット・タイムアウト検出
502
初期値をゼロ・クレジット・タイマーにロードする
504
タイマー＝０？
はい
ファブリック・ロックアップ処理
514
いいえ
506
ゼロ・クレジット？
はい
ゼロ・クレジット・タイマーをデクリメントする
512
いいえ
508
退去
510

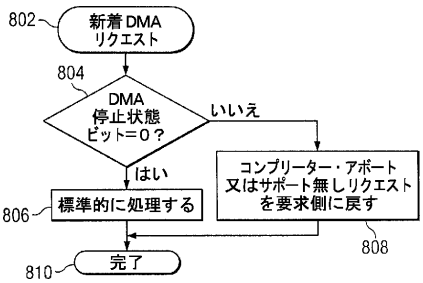
【図６】


602
ファブリック・ロックアップ処理
604
停止状態レジスタのMMIOビット及びDMAビットをセットする
606
ルート・コンプレックスにエラー・メッセージを送る
608
送信キュー内の各アイテムについて、MMIOが格納リクエストならば捨てる。MMIOがロード・リクエストならば、全て１のデータを戻す。DMAが読み出し応答ならば、捨てる。
610
完了

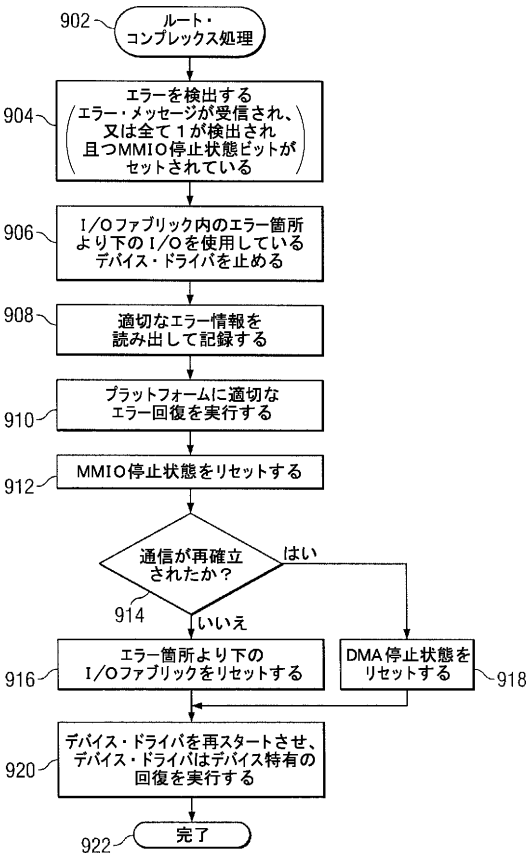
【図７】


702
送信キューに新アイテム
704
MMIOロード？
はい
いいえ
706
MMIO停止状態ビット＝0？
はい
708
標準的に処理する
いいえ
710
全て１のデータを戻す
MMIO格納？
いいえ
712
はい
MMIO停止状態ビット＝0？
はい
714
いいえ
標準的に処理する
716
MMIOを捨てる
718
DMA停止状態ビット＝0？
はい
720
標準的に処理する
722
いいえ
DMA読み出し完了を捨てる
724
完了
726

【図８】


802
新着DMAリクエスト
804
DMA停止状態ビット＝0？
いいえ
はい
コンプリーター・アボート又はサポート無しリクエストを要求側に戻す
808
標準的に処理する
806
完了
810

【図９】


902
ルート・コンプレックス処理
904
エラーを検出する（エラー・メッセージが受信され、又は全て１が検出され且つMMIO停止状態ビットがセットされている）
906
I/Oファブリック内のエラー箇所より下のI/Oを使用しているデバイス・ドライバを止める
908
適切なエラー情報を読み出して記録する
910
プラットフォームに適切なエラー回復を実行する
912
MMIO停止状態をリセットする
通信が再確立されたか？
はい
914
いいえ
916
エラー箇所より下のI/Oファブリックをリセットする
DMA停止状態をリセットする
918
920
デバイス・ドライバを再スタートさせ、デバイス・ドライバはデバイス特有の回復を実行する
922
完了

フロントページの続き

(72)発明者 スティーブン・マーク・サーバー
アメリカ合衆国 ７８７１７ テキサス州 オースティン エフライム・ロード ８３０８
(72)発明者 リチャード・ルイス・アーント
アメリカ合衆国 ７８７４６ テキサス州 オースティン バーン・スワロー・ドライブ １６０７
(72)発明者 トーマス・シュリップ
ドイツ連邦共和国 バーデン－ヴルッテンベルク ホルツガーリンゲン Ｄ－７１０８８ ゾンネンラインヴェーク ４３

合議体
審判長 小曳 満昭
審判官 和田 志郎
審判官 山田 正文

(56)参考文献 特開平６－２２７１００（ＪＰ，Ａ）
特開２００５－２９３２８３（ＪＰ，Ａ）
特開平８－３２０８３６（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
G06F 13/36
G06F 13/38
G06F 13/00

(54)【発明の名称】入出力（Ｉ／Ｏ）ファブリック内のキューをクリアする方法、Ｉ／Ｏファブリック・エラーを処理する方法及びコンピュータ・プログラム製品（エラー回復のためにＩ／Ｏファブリックのロックアップ状態を検出しクリアするためのメカニズム）