

(19) 日本国特許庁(JP)

(12) 特 許 公 報(B2)

(11) 特許番号

特許第4002337号

(P4002337)

(45) 発行日 平成19年10月31日(2007.10.31)

(24) 登録日 平成19年8月24日(2007.8.24)

(51) Int. Cl.	F I
HO 4 L 12/44 (2006.01)	HO 4 L 12/44 3 0 0
HO 4 L 12/66 (2006.01)	HO 4 L 12/66 Z
GO 6 F 13/00 (2006.01)	GO 6 F 13/00 3 5 5

請求項の数 4 (全 135 頁)

(21) 出願番号	特願平10-291	(73) 特許権者	591030868
(22) 出願日	平成10年1月5日(1998.1.5)		コンパック・コンピューター・コーポレーション
(65) 公開番号	特開平10-215275		COMPAG COMPUTER CORPORATION
(43) 公開日	平成10年8月11日(1998.8.11)		アメリカ合衆国テキサス州77070, ヒューストン, ステイト・ハイウェイ 249, 20555
審査請求日	平成16年12月24日(2004.12.24)		20555 State Highway 249, Houston, Texas
(31) 優先権主張番号	774605		77070, United States of America
(32) 優先日	平成8年12月30日(1996.12.30)	(74) 代理人	100089705
(33) 優先権主張国	米国 (US)		弁理士 社本 一夫

最終頁に続く

(54) 【発明の名称】 多重バス・アーキテクチャを有するネットワーク・スイッチ

(57) 【特許請求の範囲】

【請求項 1】

ネットワーク・スイッチにおいて、
 データを受信し送信するための複数のネットワーク・ポートであって、各々が、
 ネットワーク・インターフェース、
 データ・バス・インターフェース、及び
 プロセッサ・ポート・インターフェース
 を含むネットワーク・ポートと、
 上記複数のネットワーク・ポートの各々の上記データ・バス・インターフェースに接続されたデータ・バスと、
 上記複数のネットワーク・ポートそれぞれの上記プロセッサ・ポート・インターフェースに接続されているプロセッサ・バスに接続されたプロセッサと、
 メモリ・バス接続されたメモリと、
 上記データ・バス、上記プロセッサ・バス、上記複数のネットワーク・ポートそれぞれに接続されたプロセッサ・バス、及び上記メモリ・バスに接続され、上記複数のネットワーク・ポートと上記メモリとの間のデータの流れを制御し、且つ、上記プロセッサが上記複数のネットワーク・ポート及び上記メモリにアクセスできるようにするためのスイッチ・マネージャであって、
 上記データ・バスに接続され、上記複数のネットワーク・ポートそれぞれの状態を定期的にポーリングして、各々のポート状態を決定するためのポーリング・ロジックと、

10

20

上記メモリ・バス及び上記データ・バス・インターフェースに接続されたメモリ・バス・インターフェースと、

上記ポーリング・ロジックに接続され、上記複数のネットワーク・ポート、上記プロセッサ、及びメモリ・バス・インターフェース間のデータの流れを制御するための、コントロール・ロジックと、

を備えたスイッチ・マネージャと、

上記プロセッサ・バス、上記データ・バス・インターフェース及び上記メモリ・バス・インターフェースに接続されたプロセッサ・バス・インターフェースとを含むスイッチ・マネージャと

からなることを特徴とするネットワーク・スイッチ。

10

【請求項 2】

請求項 1 記載のネットワーク・スイッチにおいて、

上記プロセッサ・バス・インターフェースは、上記プロセッサ・バス、上記プロセッサ・インターフェース、及びポート・インターフェースを含み、

上記プロセッサ・バスが、上記スイッチ・マネージャと上記プロセッサとの間に接続されたプロセッサ部分、及び上記スイッチ・マネージャと上記複数のネットワーク・ポートの各々の上記プロセッサ・ポート・インターフェースとの間に接続されたポート部分を含み、

上記プロセッサ・インターフェースが、上記プロセッサ・バスの上記プロセッサ部分を介して上記プロセッサに接続されており、

20

上記ポート・インターフェースが、上記プロセッサ・インターフェースに接続され、且つ上記プロセッサ・バスの上記ポート部分を介して上記複数のネットワーク・ポートの各々に接続されている

ことを特徴とするネットワーク・スイッチ。

【請求項 3】

請求項 2 記載のネットワーク・スイッチにおいて、上記プロセッサ・バス・インターフェースが、上記プロセッサ・バスの上記プロセッサ部分と上記ポート部分との間でサイクルを変換するためのバス転送ロジックを含むことを特徴とするネットワーク・スイッチ。

【請求項 4】

請求項 3 記載のネットワーク・スイッチにおいて、上記プロセッサ・バス・インターフェースはさらに、

30

上記プロセッサ・インターフェース及び上記データ・バス・インターフェースに接続された第 1 の受信バッファと、

上記プロセッサ・インターフェース及び上記データ・バス・インターフェースに接続された第 1 の送信バッファと、

上記プロセッサ・バス・インターフェースと上記データ・バス・インターフェースとの間のデータの流れを制御するための、上記データ・バス・インターフェース、上記プロセッサ・インターフェース、上記第 1 の受信バッファ及び上記第 1 の送信バッファに接続された第 1 のコントローラ、

上記プロセッサ・インターフェース及び上記メモリ・バス・インターフェースに接続された第 2 の受信バッファと、

40

上記プロセッサ・インターフェース及び上記メモリ・バス・インターフェースに接続された第 2 の送信バッファと、

上記プロセッサ・バス・インターフェースと上記メモリ・バス・インターフェースとの間のデータの流れを制御するための、上記メモリ・バス・インターフェース、上記プロセッサ・インターフェース、上記第 2 の受信バッファ及び上記第 2 の送信バッファに接続された第 2 のコントローラと

を含むことを特徴とするネットワーク・スイッチ。

【発明の詳細な説明】

【0001】

50

【産業上の利用分野】

本発明は、ネットワーク・デバイスの分野に関し、より詳細には多重バス・アーキテクチャを含むネットワーク・スイッチに関する。

【0002】**【従来の技術】**

ファイル及び資料を分かち合うための、又は2つ又はそれ以上のコンピュータ間の通信を他の方法で可能にするためのネットワーク及びネットワーク・システムには多くの異なった型式がある。ネットワークは、種々の特徴及び機能、例えば、メッセージ容量、各ノードが分配される範囲、ノード又はコンピュータの型式、ノードの相関関係、位相幾何学的要因、即ち論理的及び／又は物理的配置、ケーブル型式及びデータ・パケット形式に基づくアーキテクチャすなわち構造、アクセス可能性、等に基づいて分類される。例えば、ネットワークの範囲は、各ノードが分配される距離、例えば、ビルの事務所又は床内のローカル・エリア・ネットワーク（LAN）、大学キャンパス、或いは都市又は州にわたるワイド・エリア・ネットワーク（WAN）、国境を隔てるグローバル・エリア・ネットワーク（GAN）、等のことを指している。

10

【0003】

ネットワークの構造は、ケーブル又は媒体、及び用いられる媒体アクセスだけでなく媒体を通して送信されるデータの packets 構造に関連している。種々の構造は共通であり、毎秒10メガビット（Mbps）において作動するための同軸ツイスト・ペア線又は光ファイバ・ケーブルを用いているイサernet（例えば、10Base-T, 10Base-F）或いは100Mbpsにおいて作動するための高速イサernetを含んでいる。ARCネット（Attached Resource Computer Network）は、2.5Mbpsにおいて作動するための同軸ツイスト・ペア線又は光ファイバを用いている比較的安価な通信網構造である。トークン・リング幾何学構造は1～16Mbpsにおいて作動するための特殊なIBMケーブル又は光ファイバを用いている。勿論、外にも多くの型式のネットワークが周知であり使用可能である。

20

【0004】

各ネットワークは一般的に、選択された媒体及び各ノード間のデータに対して遅延、送信、反復、変換、フィルタリング等を行うための種々の他のネットワーク・デバイスを通して結合される、しばしばノード或いはステーションと呼ばれる2つの又はそれ以上のコンピュータを含んでいる。「ネットワーク・デバイス」の用語は一般的に、コンピュータ及びそれらのネットワーク・インターフェースカード（NIS）だけでなくネットワーク上の種々の他のデバイス、例えば、リピータ（中継器）、ブリッジ、スイッチ、ルータ、ブルータのことを指している。与えられたネットワーク・プロトコルに従って作動するネットワークは、1つ又はそれ以上のリピータ、ブリッジまたはスイッチを用いて拡大し得る。リピータは、物理層において機能し且つ各々の受信パケットを他の各ポートに再送信するハードウェア・デバイスである。ブリッジはOSI基準モデルのデータ・リンク層において作動するが、パケットを濾波して各ネットワーク・セグメント上の不要パケットの伝播量を低減することにより、効率を増大する。

30

【0005】

ネットワーク・スイッチは、ネットワーク・トラフィックをネットワーク間で方向付けるために幾つかの類似のネットワークに結合するための複数のポートを含む多重ブリッジと機能的に類似しており、更にそれよりは効率的になっている。ネットワーク・スイッチは通常は、バス及びネットワーク・データを一時的に記憶するためのメモリを介して各ポートに結合されているスイッチング・マトリクス、例えば、イサernet・パケット等を含んでいる。トラフィックを方向付けるために、且つ他のタスク、例えば、初期化、構成、統計監視、ネットワーク管理を実行するために、有意な処理能力が通常必要となる。ネットワーク管理は、メモリ管理、IEEE（アメリカ電気電子学会）802.1規格によるスパンニング・ツリー・アルゴリズムの実施、管理情報ベース（MIB）又はMIB IIの保守及び管理、等を含んでいる。

40

50

【0006】

典型的なスイッチ・アーキテクチャは、全てのネットワーク並びにプロセッサのトラフィックのための1つの一次バスを有している。斯かるオーバーヘッド機能は各ポート、スイッチ・ファブリック及びメモリを監視し管理するために、バスに結合された少なくとも1つのプロセッサ等を必要としている。これらのオーバーヘッド機能は、通常のネットワーク・トラフィックに干渉して、これによりスイッチの処理を減速して低下せしめる有意なプロセッサ時間及びバス帯域を必要とする。斯かる処理低下は、しばしば、特に重負荷のときに外れパケットをもたらしてしまう。

【0007】

【発明が解決しようとする課題】

重負荷の期間中においてもネットワーク・ときにを扱う能力が改良された通信網スイッチを提供することが要求されている。斯くして、ネットワーク・オーバーヘッド機能、例えば、初期化、構成、監視及びネットワーク管理を実行しながらもネットワーク・トラフィックを扱うことが出来るネットワーク・スイッチを提供することが要求される。

【0008】

【課題を解決するための手段】

本発明に係るネットワーク・スイッチは、データを受信且つ送信するための1つ又はそれ以上のネットワーク・ポートであって、各々がネットワーク・インターフェース、データ・バス・インターフェース並びにプロセッサ・ポート・インターフェースを含むネットワーク・ポートを含んでいる。このネットワーク・スイッチは、ポートの各々のデータ・バス・インターフェースが結合されたデータバス、プロセッサに結合されかつポートの各々のプロセッサ・ポート・インターフェースに結合されたプロセッサ・バス、並びにメモリに結合されたメモリ・バスを含んでいる。このネットワーク・スイッチは更に、データ・バス、プロセッサ・バス及びメモリ・バスに結合されたスイッチ・マネージャであって、ポートとメモリ間のデータ流を制御するための、且つポート及びメモリへのプロセッサのアクセスを可能にするためのスイッチ・マネージャを含む。このようにして、プロセッサは、データ・バスの貴重なバンドを消費することなく、監視、決定状態、構成及び管理のためのネットワークへの直接且つ独立アクセスを行うことができる。

【0009】

スイッチ・マネージャは、データ・バス、メモリ・バス及びプロセッサ・バスにそれぞれ結合するためのデータ・バス・インターフェース、メモリ・バス・インターフェース及びプロセッサ・バス・インターフェースを含んでいる。データ・バス・インターフェースは、データを転送するための受信／送信バッファ、ポートを定期的にポーリングしてそれらの状態を決定するための少なくとも1つの状態マシン、並びにポートどうし間とポートとメモリ間のデータの流れを制御するためのコントロール・ロジックを含んでいる。メモリ・バス・インターフェースは、メモリのメモリ・サイクルを制御するためのメモリ・コントローラ、並びにメモリ・コントローラを介してのメモリへのアクセスを制御するためのアビータを含んでいる。メモリ・バス・インターフェースはまた、データ・バス・インターフェースからメモリへのデータ流を制御するための受信コントローラ並びにメモリからデータ・バス・インターフェースへのデータ流を制御するための送信コントローラを含んでいる。メモリ・バス・インターフェースは更に、メモリ・バスを介してのメモリの状態を保守して、これにより、リフレッシュ機能からプロセッサを解放するためのリフレッシュ・コントローラを含んでいる。

【0010】

本明細書に記載された実施例によると、プロセッサ・バスは、スイッチ・マネージャとプロセッサとの間に結合されたプロセッサ部分並びにスイッチ・マネージャとポートの各々の間に結合されたポート部分を含んでいる。スイッチ・マネージャのプロセッサ・バス・インターフェースはプロセッサ・バスのプロセッサ部分を介してプロセッサに結合されたプロセッサ・インターフェース並びにプロセッサ・バスのポート部分を介してプロセッサ・インターフェースに且つネットワーク・ポートの各々に結合されたポート・インター

10

20

30

40

50

フェースを含んでいる。これらのプロセッサ及びポート及びバス部分は同じ寸法であることが好ましい。しかしながら、図示された且つ本明細書に記載された実施例によると、プロセッサ・バス・プロセッサ及びポート部分は、異なったバンドは有しており、プロセッサ・インターフェースはプロセッサ・バス・プロセッサ部分とポート部分との間のサイクルを変換するための状態マシンを含んでいる。ネットワーク・ポートの各々は、その対応ポートの状態及び作動を追跡するための1つ又はそれ以上の統計カウンタを含んでおり、これらのカウンタは結合されており、斯くして、プロセッサ・バス・ポート部分で容易に使用可能となる。このようにして、プロセッサは、データバス上の活動を妨害することなく、作動中にオーバーヘッド機能を実行するためのポートの各々への独立した完全なアクセスをすることができる。

10

【0011】

更に、プロセッサ・バス・インターフェースによると、メモリ・バス・インターフェースを介してのデータ・バスへの且つメモリへのプロセッサ・アクセスが可能になる。特に、プロセッサ・バス・インターフェースは、適切な送信／受信バッファ並びにプロセッサ・バス・インターフェースとデータ・バス・インターフェースとの間のデータ流を制御するための第一のコントローラ及びプロセッサ・バス・インターフェースとメモリ・バス・インターフェースとの間のデータ流を制御するための第二のコントローラを含んでいる。

【0012】

本明細書に記載された本発明に係るネットワーク・スイッチの特定の実施例では、これら複数のネットワーク・ポートは第一のデータバスに結合された第一のプロトコルに従って作動する第1のグループのポート並びに第二のプロトコルに従って作動する第2のグループのポートを含んでいる。第2のグループのポートをインターフェースするための第二のデータ・バスが提供されており、第一のデータ・バスと第二のデータ・バスとの間にはブリッジ・デバイスが結合されている。図示の実施例では、第1のグループのポートは10Mbpsでイーサネット規格に従って作動し、一方、第2のグループは100Mbpsでイーサネット規格に従って作動するが、本発明は何ら特定のプロトコル或いはデータ転送速度に限定されないことが理解されるであろう。

20

【0013】

本発明に係るネットワーク・システムは、複数のネットワークであって、各々がデータパケットを送り且つ受けるための少なくとも1つのデータ・デバイスを含むネットワーク、並びにデータパケットを転送するためのネットワークに結合された上記のネットワーク・スイッチを含んでいる。

30

【0014】**【発明の実施の態様】**

図1を参照すると、本発明に従って構成されたネットワーク・スイッチ102を含むネットワーク・システム100の簡略図が示されている。ネットワーク・スイッチ102は、それぞれが適当なメディア・セグメント108を介して“A”ネットワーク106の1つと結合およびこれと交信する1つまたは複数の“A”ポートを含む。各メディア・セグメント108は、よった対のワイヤ・ケーブル、光ファイバ・ケーブルその他のような、ネットワーク・デバイスを接続するための任意のタイプの媒体である。ポート104は、ネットワーク・スイッチ102とネットワーク106の各々との間における双方向通信またはデータ・フローを可能ならしめる。このような双方向データ・フローは、例えば半二重モードあるいは全二重モードのような、いくつかのモードのいずれか1つのモードに従う。図1に示すように、“j”+1までのネットワーク106が存在し、それぞれにAネットワーク(A-NETWORK)0、Aネットワーク1、・・・、Aネットワークjという名称が付与されており、各ネットワーク106は、それぞれAポート(A-PORT)0、Aポート1、・・・、Aポートjという名称が付与されているj+1個のポート104のうち対応する1つを介してネットワーク・スイッチ102に結合する。ネットワーク・スイッチ102は、対応する数までのネットワーク106に結合すべく任意の数のポート104を含むことができる。本明細書で説明する実施例において、jは24までのネッ

40

50

トワーク 106 との結合のための全部で 24 のポートに対するために 23 に等しい整数である。本明細書においては、これらのポートを一括してポート 104 と呼ぶか、あるいは個別にポート (PORT) 0、ポート 1、ポート 2、・・・、ポート 23 と呼称する。

【0015】

同様に、ネットワーク・スイッチ 102 はさらに、それぞれが適当なメディア・セグメント 114 を介して “B” ネットワーク 112 に結合およびこれとインタフェースする 1 つまたは複数の “B” ポート 110 を含む。また、各メディア・セグメント 114 は、よった対のワイヤ・ケーブル、光ファイバ・ケーブルその他のような、ネットワーク・デバイスを接続するための任意のタイプの媒体である。ポート 110 もまた双方向型であり、ネットワーク・スイッチ 102 とネットワーク 112 との間におけるデータ・フローを、ポート 104 に関する上述の説明と同様に可能ならしめる。本明細書で説明する実施例において、それぞれに B ネットワーク (B-NETWORK) 0、B ネットワーク 1、・・・、B ネットワーク k という名称が付与されている “k” + 1 までのネットワーク 112 との結合に備えて “k” + 1 の数のポート 110 が存在し、個別に B ポート (B-PORT) 0、B ポート 1、・・・、B ポート k と呼称する。ネットワーク・スイッチ 102 は、対応する数までのネットワーク 112 に結合すべく任意の数のポート 110 を含むことができる。本明細書に示す特定のな実施例において、K は 4 つまでのネットワーク 112 との結合のための全部で 4 個のポート 110 のために、3 に等しい整数である。“A” タイプのポートおよびネットワークは、“B” タイプのポートおよびネットワークと異なるネットワーク・プロトコルおよび／または速度で動作する。本明細書に示す特定のな実施例において、ポート 104 およびネットワーク 106 はイーサネット (Ethernet) プロトコルに従い 10 メガビット／秒 (Mbps) で動作し、一方、ポート 110 およびネットワーク 112 はイーサネットのプロトコルに従って 100 Mbps で動作する。本明細書では、B ポート 0、B ポート 1、・・・、B ポート 3 を総称してポート 110 とし、個別にはそれぞれポート 24、ポート 25、・・・、ポート 27 と呼称する。

【0016】

ネットワーク 106 および 112 は、データの入力あるいは出力のために 1 つまたは複数のデータ・デバイスもしくはデータ端末装置 (DTE)、あるいは 1 つまたは複数のデータ・デバイスを接続するために任意のタイプのネットワーク・デバイスを含む。このように、A ネットワーク 0 や B ネットワーク・1 などのようないずれのネットワークも、それぞれ 1 つまたは複数のコンピュータ、ネットワーク・インタフェース・カード (NIC)、ワークステーション、ファイル・サーバ、モデム、プリンタ、あるいはリピータ、スイッチ、ルータ、ハブ、集信装置といったネットワーク内でのデータの受信や送信のための他のデバイスを含むことができる。例えば図 1 に示すように、いくつかのコンピュータ・システムあるいはワークステーション 120、122 および 124 は、A ネットワーク j の対応するセグメント 108 に結合されている。コンピュータ・システム 120、122 および 124 は相互に、あるいはネットワーク・スイッチ 102 を介して他のネットワークの他のデバイスと通信することができる。そこで各ネットワーク 106 および 112 は 1 つまたは複数のセグメントを介して結合された 1 つまたは複数のデータ・デバイスを表し、ネットワーク・スイッチ 102 がネットワーク 106 および 112 のいずれかの中の何れか 2 つまたはそれ以上のデータ・デバイスの間でデータの転送を行う。

【0017】

ネットワーク・スイッチ 102 は、ポート 104 および 110 の各々に結合されたデータ・デバイスから情報を受け取り、その情報を他のポート 104 および 110 のいずれかのものまたは複数のものへルーティングする (送る) 動作を一般的に行う。ネットワーク・スイッチ 102 はまた、同じネットワーク内のデータ・デバイスに対してのみと意図された、1 つのネットワーク 106 または 112 内の 1 つのデータ・デバイスから受信した情報をドロップ (落とす) するか、さもなくば無視することによって情報のフィルタリングを行う。データあるいは情報はパケットの形になっているが、各データ・パケットの形はそのネットワークがサポートしているプロトコルによって異なる。パケットは予め定義さ

10

20

30

40

50

れたバイトのブロックであり、通常ヘッダ、データ、およびトレーラから成り、特定のパケットの形式はそのパケットを生成したプロトコルによって決まる。ヘッダは、一般に、宛て先のデータ・デバイスを識別する宛先アドレス、およびパケットの発信元であるデータ・デバイスを識別するソース・アドレスを含み、普通これらのアドレスは業界内での一意性を保証するメディア・アクセス・コントロール (MAC) アドレスである。1つの宛て先デバイスに対して意図されたパケットを、ここではユニキャスト (unicast) ・パケットという。さらに、ヘッダはグループ (GROUP) ビットを含み、このビットは、そのパケットが複数の受信先デバイスに向けられたマルチキャスト (multicast) 又はブロードキャスト (BC) ・パケットであるかを表示する。もしグループ・ビットがロジック 1 (1) にセットされていれば、それはマルチキャスト・パケットであると考慮され、もし宛先アドレスのビットがすべてロジック 1 (1) にセットされていれば、そのパケットは BC パケットである。しかし、本発明の目的上、マルチキャストおよび BC パケットを同等に扱い、以降は BC パケットと呼称する。

10

【0018】

図2を参照すると、ネットワーク・スイッチ102のさらに詳細なブロック図が示されている。示した実施例において、ネットワーク・スイッチ102は、6つの類似のカッド・コントローラあるいはカッド・カスケード (QC) ・デバイス202を含み、それぞれが4つのポート104を組み込んでいる。QCデバイス202は、単一の特定用途向け IC (ASIC) パッケージへ統合して、あるいは示されているような個別の集積回路 (IC) チップとして、任意の所望の形で実施することができる。示した実施例において、各ポート104は半二重方式により10Mbpsで動作し、合計スループットが全二重で1ポートあたり20Mbpsとなる。その結果、6つのQCデバイス202がすべて全二重方式で動作すれば合計で480Mbpsとなる。各QCデバイス202は、好適には、QC/CPUバス204に結合したプロセッサ・インタフェース、および高速バス (HSB) 206に結合したバス・インタフェースを含む。HSB206は、データ部206aおよび各種の制御及び状態信号206bを含む。HSB206は、毎秒1ギガビット以上のデータを転送する32ビット、33メガヘルツ (MHz) のバスである。

20

【0019】

HSB206およびQC/CPUバス204はさらに、イーサネット・パケット・スイッチ・マネージャ (EPSM) 210に結合される。EPSM210の実施について、本発明はなんら特定の物理的または論理的制約を課していないが、示されている実施例ではASICとして実施される。EPSM210はさらに、データおよびアドレス部214aと制御信号214bを含む32ビットのメモリ・バス214を介してメモリ212に結合される。メモリ212は、好適には、特定の用途で必要に応じて任意に増設が可能ではあるが、4から16メガバイト (MB) のダイナミック・ランダム・アクセス・メモリ (DRAM) を含んでいる。EPSM210は、動作が約60ナノ秒 (ns) の高速ページ・モード (FPM) のシングル・インライン・メモリ・モジュール (SIMM)、拡張データ出力 (EDO) モードのDRAM SIMM、あるいは同期モードのDRAM SIMMを含む、メモリ212の実施のための少なくとも3つの異なったタイプのDRAMのうちのいずれか1つをサポートする。同期DRAMは、一般に、66MHzデータ速度又は1秒あたり266MBのバースト・データ速度を達成するために、66MHzのクロックを必要とする。EDO DRAMは、33又は66MHzのいずれかのクロックで動作できるが、いずれのクロック速度においても33MHz、または1秒あたり133MBの最大データ・バースト・データ速度を達成する。FPM DRAMもまた33又は66MHzのクロックで動作が可能であるが、33MHzクロックで16MHz又は1秒あたり64MBの最大バースト速度を達成し、66MHzクロックで22MHz又は1秒あたり88MBのバースト速度を達成する。

30

40

【0020】

メモリ・バス214は、メモリ・データ・バスMD [31:0]、データ・パリティ信号MD__PAR [3:0]、行および列 (カラム) アドレス信号MA [11:0]、ライト

50

(書き込み)・イネーブル信号MWE＊、FPM DRAM及びEDO DRAMの行信号又は同期DRAMのチップ選択のいずれかであるバンク選択信号RAS [3:0]＊/SD_CS＊ [3:0]、FPM及びEDOの列信号または同期DRAMのDQMであるメモリ・バイト制御信号CAS [3:0]＊/SD_DQM [3:0]、同期DRAMのみへの行信号SD_RAS＊、同期DRAMのみへの列信号SD_CAS＊、シリアル入力SIMM/DIMM存在検知信号PD_SERIAL_IN、およびパラレル入力SIMM/DIMM存在検知信号PD_LOAD＊を含む。

【0021】

HSB206は、サンダー(Thunder)LAN(TLAN)ポート・インタフェース(TPI)220に結合され、これがさらにデータ及びアドレス信号222aおよび関連の制御及び状態信号222bを含む周辺コンポーネント相互接続(PCI)バス222に結合される。PCIバス222は4つのTLAN226に結合され、これは任意の様式で実施される。TLAN226は、それぞれがポート110の1つを組み込んでいる、テキサス・インスツルメンツ社(Texas Instruments, Inc.)(TI)製のTNETE100 ThunderLANTM(サンダーLAN、登録商標)PCI EthernetTM(イーサネット、登録商標)コントローラが好適である。EPSM210に対して、TPI220は4つのポートをインタフェースするために、別のQCデバイス202と同様にHSB上で動作する。従って、EPSM210には實際上7つのカッド・ポート・デバイスが「見える」。PCIバス222に関しては、TPI220が、標準PCIバスのエミュレーションを、通常PCIのメモリ・デバイスとインタフェースするTLAN226の適切な動作に必要な程度まで、行う。従って、PCIバス222は完全にPCIに従順である必要がない。PCIバス222は、CPU230をローカルのRAM234、ローカルのフラッシュRAM236および必要であればシリアル・ポート・インタフェース238に結合するためのローカル・プロセッサ・バス232に結合されているプロセッサ又は中央処理装置(CPU)230に結合される。シリアル・ポート・インタフェース238は、UARTまたは同等のものが望ましい。示した実施例においては、CPUはインテル社(Intel)製の32ビット、33MHzのi960RP CPUであるが、CPU230は他の適切なプロセッサでも構わない。

【0022】

CPU230は、通常ネットワーク・スイッチ102のパワーアップでTPI220およびEPSM210の初期設定とコンフィギュレーションの処理を行う。また、CPU230は統計情報の監視及び収集を行い、さらに動作時にはネットワーク・スイッチ102の各種デバイスの機能を管理及び制御する。さらにまたCPU230は、メモリ212内のハッシュ・テーブル・データをEPSM210を通じて更新する。しかし、EPSM210は、メモリ212へのアクセスを制御し、DRAMのリフレッシュ・サイクルを実行し、それによってCPU230によるリフレッシュ動作が不要となる。このように設計されていなければ、CPU230は各リフレッシュ・サイクルの実行におよそ6～8バス・サイクルを要することになり、これは貴重なプロセッサ・リソースを消費することとなる。CPU230はまた、様々な目的のための付加的なネットワーク・ポートとして機能し、従って本明細書ではポート(PORT)28として言及する場合がある。このように、ポート104、110、およびCPU230は、それぞれポートポート0～ポート28を集合的に含むものである。

【0023】

CPU230はさらに、アドレス及びデータ部218aおよび関連の制御及び状態信号218bを含むCPUバス218を介してEPSM210に結合される。アドレス及びデータ部218aは、アドレスとデータ信号間で多重化されていることが望ましい。特定的には、CPUバス218は、アドレス/データ・バスCPU_AD [31:0]、CPU230からのアドレス・ストロブCPU_ADS＊、データ・バイト・イネーブルCPU_BE [3:0]、リード/ライト選択信号CPU_WR＊、バースト最終データ・ストロブCPU_BLAST＊、データ・レディ信号CPU_RDY＊、および少なくとも

10

20

30

40

50

1つのCPU割り込み信号CPU__INT*を含む。本開示において、データまたはアドレス信号の他の通常の信号名は正のロジックを表し、その信号はハイ又はロジック1のときアサートされるとみなされ、後尾にアステリスク(*)が付加された信号名は負のロジックを示し、その信号はロー又はロジック0のときにアサートされるとみなされる。各信号の機能的な定義は一般に直接的であって、普通はその信号名で判断され得る。

【0024】

図3は、4つのポート104の実施のための例示的なQCデバイス202のブロック図であり、このデバイスは24ポート、ポート0～ポート23を実施するために6つ複製される。特定のデバイスを1つ挙げれば、LSIロジック社(LSI Logic Corporation)(LSI)製のL64381カッド・カスケード・イーサネット(Quad Cascade Ethernet)・コントローラ・デバイスがある。これよりグレードの高いデバイスとして、やはりLSI製のQE110カッド・カスケード・イーサネット・コントローラ・デバイスがあり、これは本明細書で説明しているような付加的機能および能力を備えている。しかし留意すべきは、本発明はポート104の実施をなんら特定のデバイスに限定しているものではない。示した実施例において、各QCデバイス202はポート104のそれぞれに対してイーサネット・コア300を含み、イーサネット・コア300は完全な同期型であって、メディア・アクセス・コントローラ、マンチェスタ・エンコーダ/デコーダ、およびよった対/AUI(接続機構インタフェース(Attachment Unit Interface))トランシーバを含む。各イーサネット・コア300は、対応するセグメント108上の結合されているネットワーク106との双方向データ通信を可能とし、それぞれが対応する128ビット受信FIFO(先入れ先だし(First-In, First-Out))302および128ビット送信FIFO304と結合している。各イーサネット・コア300は、さらに統計カウンタ306のブロックと結合しており、統計カウンタ306の各ブロックは、オンチップ・メインテナンス用に25のカウンタを含む。統計カウンタ306の各ブロック内のカウンタは、シンプル・ネットワーク・マネジメント・プロトコル(Simple Network Management Protocol)(SNMP)の要件に見合うのが望ましい。FIFO302および304の各々は、さらに、各QCデバイス202とEPSM210の間での双方向データ・フローを可能とするためにHSB206に結合しているバス・インタフェース・ロジック308に結合される。各QCデバイス202は、ソース・アドレス挿入、フレーム・チェック・シーケンス(FCS)挿入、衝突時の即時再送信、バス転送サイズ、および送信バッファ・スレッシュールド・サイズといったコンフィギュレーションをプログラミング可能(プログラマブル)とするために、コンフィギュレーション及びコントロール(制御)・ロジック310を含む。

【0025】

コンフィギュレーション及びコントロール・ロジック310と、統計カウンタ306の各ブロックと、FIFO302、304はQC/CPUバス204に結合される。EPSM210は、CPUバス218とQC/CPUバス204との間に別のインタフェースを提供する。このようにして、CPU230は、各QCデバイス202の各々、従ってポート104の各々に対し、そのアクティビティを初期設定、構成(コンフィギュレーション)、監視(モニタ)、および修正すべく完全なアクセスを得る。QE110カッド・カスケード・イーサネット・コントローラ・デバイスは、もし背圧(バックプレッシャ(backpressure))指示の受信が間に合うならば、受信されていたパケットを終了するためのジャミング・シーケンス(jamming sequence)をアサートするために背圧指示を検知するために、コンフィギュレーション及びコントロール・ロジック310間に付加的な接続320を含む。背圧指示はHSB206上で実行される背圧サイクルが望ましいが、背圧指示を示すために別の信号又はそれと同様のものを用いるなど、いくつかの方法の任意のものを用いることができる。

【0026】

ここで、ジャミング・シーケンスは「早い」又は適時だと考えられるポートで受信中のデータ・パケットの最初の64バイトの間に送信すべきであるという点に留意されたい。最

10

20

30

40

50

初の16バイト(4つのWORD)は、後述するハッシュ・ルックアップ手順がE P S M 2 1 0によって実行される前に要求される。最初の16バイトがおおよそ13マイクロ秒(μs)で転送されるように、各データ・ビットはイーサネット10Base-Tを約100nsの速度で転送される。64バイトがおおよそ51 μs の間に受信され、それによって、ネットワーク・スイッチ102は、受信された最初の16バイトを転送し、ハッシュ手順を行い、背圧サイクルを実行し、最終的にジャミング・シーケンスをアサートするために、約38 μs 有する。ハッシュ・ルックアップは完了するのに約1~2 μs 要するので、ほとんど常に、適時(タイムリー)にジャミング・シーケンスを送信するために十分な時間がある。しかし、ジャミング・シーケンスをタイムリーにアサートできるという保証はない。そのため、スレッシュOLD違反条件に起因してパケットを落とす(ドロップする)可能性がある。もし背圧サイクル遅れて実行されると、そのポートは背圧サイクルを拒否し、ネットワーク・スイッチ102はそのパケットを受け取れなければそのパケットをドロップする。スレッシュOLD条件が早期の指示であり、従ってメモリがパケットを格納するために使用可能であり得るため、ネットワーク・スイッチ102はそのパケットを受け取れる。

10

【0027】

もし背圧サイクルがタイムリーに実行され、もしポートが半二重モードで動作していれば、コンフィギュレーション及びコントロール・ロジック310は示されたポート104のイーサネット・コア300の1つへ衝突コマンドを応答的にアサートする。衝突コマンドを受け取るイーサネット・コア300は、ジャミング・シーケンスをアサートし、そのポート104が受信しているパケットを終了させる。もし背圧サイクルが64バイト・ウィンドウ内に実行されるならば、ポートは、HSB206上でアボート信号ABORT_OUT*をアサートすることによって、そのポートに背圧サイクルが実行される旨をE P S M 2 1 0に示す。もし背圧サイクルが64バイト・ウィンドウの外側であり、従って時間内にアサートされなければ、ABORT_OUT*信号はアサートされず、E P S M 2 1 0はそのパケットをドロップする。背圧アサートの試行が失敗すれば、ほとんどの場合E P S M 2 1 0はそのパケットをドロップする。最高の能率を達成するためにはドロップされるパケットはできるだけ少ない方がよいが、ドロップされたパケットは最終的に送信側のデータ・デバイスにおける高いネットワーク・レベルで検知され、従ってネットワーク・システム100の全体的な動作には致命的なものとならない。送信側のデバイスはパケットのドロップを検知し、そのドロップされたパケットを含む1つ又はそれ以上の数のパケットを再送信する。

20

30

【0028】

バス・インタフェース・ロジック308は、後に詳述するように、HSB206上で同時のリード及びライト・サイクルを実現するために、リード・ラッチ324およびライト・ラッチ326を含んでいることが望ましい。これらのラッチは、第1のクロック(CLK_1)信号の特定のサイクルでHSB206上にアサートされたPORT_NO[1:0]信号をラッチする。CLK_1信号は、HSB206にとっての主クロックであり、示した実施例においては通常おおよそ30~33MHzで動作する。CLK_1信号は主クロックであるので、以降本明細書では単にCLK信号と呼称する。第2のクロック信号CLK_2もメモリ212とのインタフェースに使用され、CLK信号の周波数の2倍(2X)又は約60~66MHzで動作する。

40

【0029】

図4は、図3に示す特定のカード・カスケード・デバイス202の信号の図解である。これらの信号は、QCバス204と関連のプロセッサ・インタフェース信号、4つのポート104に関連のネットワーク・インタフェース信号、状態信号、クロック及びテスト信号、HSBバス206に関連のバス・インタフェース信号、およびその他種々の信号を含む、いくつかの機能およびバスのセクションに分けられる。

【0030】

QCバス204に関しては、E P S M 2 1 0は、データ信号PDATA[15:0]を通

50

じて、QCデバイス202のレジスタおよびカウンタ306、310とデータの読み書きを行う。READ*信号は書き込み動作に対してはハイにアサートされ、読み出し動作に対してはローにアサートされる。QCデバイス202内の特定のレジスタは、ADDRESS[5:0]信号にアサートされたアドレスによって決定される。アドレス・ストロブ信号ADDRESS__STROBE*がいくつかのチップ選択信号CHIP__SELECTm*の対応する1つとともにアサートされると、QCデバイス202はADDRESS信号をラッチする。信号名に付けられた小文字の“m”は、一般に1つの特定のタイプに属する複数の信号を意味する。例えば、6つの別々のCHIP__SELECT[5:0]*信号があり、その場合それぞれの信号は6つのQCデバイス202のそれぞれ1つに別個にアクセスするためのものである。信号READY*は、要求されたデータがラッチされるCLK信号の立ち上がり後のライト・サイクル中に、CLK信号の1サイクルに対してQCデバイス202によってローにアサートされる。リード・サイクルについては、QCデバイス202が、データをPDATAバス上に置いた後の1CLKサイクルに対してREADY*をローにアサートする。

10

【0031】

図5は、QCデバイス202のプロセッサ・リード・サイクルを図解する例示的なタイミング図であり、図6は、プロセッサ・ライト・サイクルを図解する例示的なタイミング図である。図7は、QCデバイス202のプロセッサ・バースト・リード・アクセス・サイクルを図解する例示的なタイミング図である。これらのタイミング図はいずれもあくまで例示であって、特定のタイミングや特定の信号特性などを示すものではなく、一般的な相

20

【0032】

図4に戻り、これを参照する。ネットワーク・インタフェース信号は、負および正の衝突スレッシュホールド信号、衝突参照信号、信号中のシリアル・データ、負および正のマンチェスタ符号化データ信号、正および負のデータ・スレッシュホールド信号、データ・スレッシュホールド参照信号、正および負のプリエンファシス（Pre-emphasis）信号、および各QCデバイス202の[3:0]で表される4ポートの各々に対するよった対/AUIモード選択信号を含む。各QCデバイスはCLK信号を受信し、ポート104が使用する80、20および10MHzの内部クロック信号を生成するための20MHzのクロック信号を受信するCLOCK__20MHZ入力を有する。各イーサネット・コア300は、対応するセグメント108で発生する衝突を検知し、イーサネットのCSMA/CD（キャリア検知多重アクセス/衝突検出（Carrier Sense Multiple Access/Collision Detect））法に従ってジャミング・シーケンスを送信する。

30

【0033】

HSB206に関連するバス・インタフェース信号については、QCデバイス202がABORT__OUT*信号をアサートして1つのパケット全体をアボートする。EPSM210は、アボート信号ABORT__IN*をアサートして現在のバス・サイクルをアボートする。1つの実施例においては、QCデバイス202は、EPSM210がHSB206上で背圧サイクルを実行することによって受信しているパケットをアボートできるように考案されたQE110デバイスである。この特定のタイプの背圧機能は、1つのポートで受信中の1つのパケットの拒否を可能とする「パケット毎（パケット・バイ・パケット）」あるいは動的な「ポートごと」の背圧である。L64381デバイスは、本明細書で後に詳述する自動挿入フレーム・チェック・シーケンス信号(AI__FCS__IN*)を含む。QE110デバイスはAI__FCS__IN*信号を信号FBPN*と置換する。この信号はAI__FCS__IN*信号と同じ機能を遂行するために使用されるが、背圧サイクルおよびエンハンスド・パケット・フラッシュ（enhanced packet flush）を示すためにも用いられる。本明細書で説明しているように、動的背圧を実施するために使用できる代替方法が多数存在することは言うまでもない。特に、EPSM210は、背圧要求サイクルを実行するためにリード・サイクル中にFBPN*信号をアサートする。もしABORT__OUT*信号がリード・サイクルのデータ・フェーズの間に対応するQCデバイス

40

50

202によってアサートされると、その背圧「要求」はそのQCデバイス202に認められたことになり、これがジャミング・シーケンスをアサートしてそのパケットをアポートする。もしABORT__OUT*信号がアサートされないと、E P S M 2 1 0はそのパケットをドロップする。

【0034】

E P S M 2 1 0は、QCデバイス202およびT P I 2 2 0のすべてに対して状態ストローブ信号S T R O B E *をアサートし、その各々は、S T R O B E *信号がC L K信号の立ち上がりでアサートされてサンプリングされるときに、信号P K T __A V A I L m*およびB U F __A V A I L m*上で多重化された様式でその4つのポート104又は110 (T P I 2 2 0の場合)の状態に回答する。或る動作に対しては別のポートとして働く、各QCデバイス202に対する別個の信号、T P I 2 2 0に対して1組及びC P U 2 3 0に対して類似の組、がある。特にP K T __A V A I L m*およびB U F __A V A I L m*信号は、QCデバイス202用に信号P K T __A V A I L [5:0]*およびB U F __A V A I L [5:0]*と、T P I 2 2 0用にそれぞれP K T __A V A I L [6]*およびB U F __A V A I L [6]*とも呼ばれる信号T P I __P K T __A V A I L*およびT P I __B U F __A V A I L*と、C P U 2 3 0に対応するP K T __A V A I L [7]*およびB U F __A V A I L [7]*ともそれぞれ呼ばれる信号P C B __P K T __A V A I L*およびP C B __B U F __A V A I L*との、1つの信号タイプについて全部で8つの信号を含む。

【0035】

このように、H S B 2 0 6は最初QCデバイス202が4つのポート、ポート0～ポート3にアクセスするための信号P K T __A V A I L [0]*およびB U F __A V A I L [0]*を含み、H S B 2 0 6は次のQCデバイス202が次の4つのポート、ポート4～ポート7にアクセスするための信号P K T __A V A I L [1]*およびB U F __A V A I L [1]*を含み、と以下同様で、T P I 2 2 0はポート、ポート24～ポート27にアクセスするための信号P K T __A V A I L [6]*およびB U F __A V A I L [6]*を含み、E P S M 2 1 0はC P U 2 3 0に対する内部信号P K T __A V A I L [7]*およびB U F __A V A I L [7]*を含む。C L K信号のそれぞれのサイクルで分離される4つのポートに対応する各信号に、最高4ビットが多重化される。

【0036】

S T R O B E *信号に回答して、バス・インタフェース・ロジック308は、それぞれのポートに対する対応する各送信F I F O 304にデータを格納するスペースが十分あるかどうかを表示するB U F __A V A I L [5:0]*信号のそれぞれのものに4つの状態ビットを多重化するためのポート状態ロジック303を含む。ポート状態ロジック303は、図示されている4つのポートのすべてに対して集中化するか、又はポート間に分散するかの何れかである。空きスペースの判定は、C P U 2 3 0によって16、32あるいは64バイトにコンフィギュレーションされるのが望ましい、バス転送フィールド・サイズ (T B U S) を格納するバス・インタフェース・ロジック308内のコンフィギュレーション・レジスタに従う。同様に、S T R O B E *信号に回答して、T P I 2 2 0は、後述するその内部の送信F I F Oのそれぞれに、ポート24～ポート27の各々に対するT L A N 2 2 6の対応するものに対するデータを格納するスペースが十分あるかどうかを示すために、B U F __A V A I L [6]*信号に4つの状態ビットを多重化するための、H S B 2 0 6に結合している類似したポート状態ロジック820 (図31)を含む。C P U 2 3 0あるいはポート28については、E P S M 2 1 0内のP C B 406 (図11)が、E P S M 2 1 0の内部の内部P C B送信F I F OにC P U 2 3 0に対するデータを格納するための使用可能なスペースがあるかどうかを表示するためにB U F __A V A I L [7]*信号に1つの状態ビットをアサートする。

【0037】

同様に、S T R O B E *信号に回答して、各QCデバイス202内のバス・インタフェース・ロジック308のポート状態ロジック303は、それぞれのポートに対するその受信

10

20

30

40

50

F I F O 3 0 2 の各々に、H S B 2 0 6 上におけるバス転送のための受信したデータを送信するために十分なデータがあるかどうかをT B U S の値によって表示するP K T _ A V A I L [5 : 0] * 信号のそれぞれのものに4つの状態ビットを多重化する。同様に、T P I 2 2 0 は、その内部の受信F I F O がH S B 2 0 6 上における転送のためにそれぞれのポート2 3 ~ ポート2 7 から十分なデータを受信したかどうかを表示するP K T _ A V A I L [6] * 信号に4つの状態ビットを多重化する。C P U 2 3 0 については、E P S M 2 1 0 内のP C B 4 0 6 が、E P S M 2 1 0 の内部P C B 受信F I F O がH S B 2 0 6 バス転送のためにC P U 2 3 0 から十分なデータを受信したかどうかを表示するP K T _ A V A I L [7] * 信号に1つの状態ビットをアサートする。

【0038】

図8は、Q C デバイス2 0 2 およびT P I 2 2 0 のバッファ状態問い合わせを図解する例示的なタイミング図であり、E P S M 2 1 0 によるS T R O B E * 信号のアサートと各Q C デバイス2 0 2 の応答、T P I 2 2 0 のアサートするそれぞれのP K T _ A V A I L m * およびB U F _ A V A I L m * 信号を含む。図8におけるポート0、ポート1、ポート2、およびポート3は、特定のQ C デバイス2 0 2 の4つのそれぞれのポートあるいはT P I 2 2 0 である。P C B 4 0 6 は、そのポートが4つのフェーズすべてでアクティブになっていることを除けば、その応答は同様である。S T R O B E * 信号はレベル・トリガされ、従ってC L K 信号の最初の立ち上がりでローにサンプリングされる。ここで、図8のタイミング図はあくまでも例示であって、特定のタイミングや特定の信号特性などではなく、一般的な相関性を図解するものであることに留意されたい。例えば、S T R O B E * 信号は周期的であり、示した実施例の動作においては典型的には1 C L K サイクルを超える間ローにアサートされる。

【0039】

図4に戻り、これを参照する。信号P O R T _ B U S Y * は、それぞれのポートが半二重モードで送信中であるか受信中であるか、あるいはそのポートがいつ全二重モードで送信しているかを表示するために使用される。リード・データ信号R E A D _ O U T _ P K T [5 : 0] * はE P S M 2 1 0 にアサートされて、それぞれのQ C デバイス2 0 2 に対し、それぞれの受信F I F O 3 0 2 からのデータをデータ信号D A T A [3 1 : 0] 上に置くことを通知する。同様にして、ライト・データ信号W R I T E _ I N _ P K T [5 : 0] * はE P S M 2 1 0 にアサートされて、それぞれのQ C デバイス2 0 2 に対し、データ信号D A T A [3 1 : 0] からそれぞれの送信F I F O 3 0 4 にデータを取り出すことを通知する。さらに、類似の信号P C B _ R D _ O U T _ P K T * 、P C B _ W R _ I N _ P K T * 、およびT P I _ R E A D _ O U T _ P K T * 、T P I _ W R I T E _ I N _ P K T * がそれぞれT P I 2 2 0 およびC P U 2 3 0 用に含まれる。すべてのリードおよびライト信号は、集散的にそれぞれR E A D _ O U T _ P K T m * およびW R I T E _ I N _ P K T m * 信号と呼称する。P O R T _ N O [1 : 0] ビットは、どの特定のポート1 0 4 がH S B 2 0 6 上で実行されるサイクルに対してアドレスされているかを表示する。

【0040】

信号S O P * は、パケットの先頭又はヘッダがH S B 2 0 6 上に転送されたときにパケットの開始 (Start Of Packet) を示す。A I _ F C S _ I N * 信号は、一般にS O P * およびW R I T E _ I N _ P K T m * 信号の1つとともに外部のデバイスにアサートされ、これにより、(Q C デバイス2 0 2 の1つの実施に対して) L 6 4 3 8 1 デバイスが自動的にパケット内のデータからC R C (巡回冗長検査 (Cyclic Redundancy Check)) 値を計算し、そのC R C をパケットのF C S フィールドに挿入するようにする。Q E 1 1 0 デバイスは付加的な機能のために、前述したように、A I _ F C S _ I N * 信号をF B P N * 信号と置換する。E O P * 信号は、H S B 2 0 6 上でデータ・パケットの最後のデータ転送が転送されたときにパケットの終了 (End Of Packet) を表示する。B Y T E _ V A L I D [3 : 0] * 信号は、D A T A (データ) 信号上の現在のワードにおいてどのバイトが有効であるかを表示する。通常1つのデータ・パケットはH S B 2 0 6 上での1回での転送には大き過ぎ、従って各バス・サイクルではT B U S 値に等しいか又はこれより少

10

20

30

40

50

ない量のデータが転送されることに留意されたい。

【0041】

各QCデバイス202が4つのポートのそれぞれを10Base-Tイーサネット・ポートとして動作させる点が理解できる。また、E P S M 2 1 0がQCバス204を介してQCデバイス202のすべてのレジスタに読み書きのアクセスができるということが理解できる。さらに、E P S M 2 1 0はH S B 2 0 6を介して受信F I F O 3 0 2のすべてからデータを読み取り、送信F I F O 3 0 4のすべてにデータを書き込む。

【0042】

図9は、H S B 2 0 6上での同時リード及びライト・サイクルを図解する例示的なタイミング図である。このタイミング図の一番上にサイクルのタイプを示しており、2つの同時
10
リード及びライト・サイクルが順次実行される。C L K、C L K_2、S T R O B E*、R E A D_ O U T_ P K T m*、W R I T E_ I N_ P K T m*、P O R T_ N O [1:0]、D A T A [31:0]、およびA B O R T_ O U T*信号をこのタイミング図のY軸（すなわち縦軸）に書いて示し、それに対して時間をX軸（すなわち横軸）に書いている。同時リード及びライト・サイクルには2種類があって、それらは特定の構成に依存して実行される。最初の一般的なタイプの同時サイクルについて、QCデバイス202がラッチ324および326を含むQ E 1 1 0デバイスで実施される場合は、なんら追加的な策を要せず同時リード及びライト・サイクルが実行される。これに代わって、もしQCデバイス202がL 6 4 3 8 1デバイスで実施される場合、外部のラッチおよび選択ロジック（示さず）が追加され、P O R T_ N O信号がH S B 2 0 6上でアサートされたとき、
20
これをラッチする。2番目の特殊なタイプの同時リード及びライト・サイクルは、何も補強せずL 6 4 3 8 1デバイスで実行される。ただし、それはP O R T_ N O信号が同じであるときのみ且つQCデバイス202が異なるときのみに限られる。

【0043】

E P S M 2 1 0は、例えばリード、ライト、同時リード及びライト、背圧などといった、実行すべきサイクルのタイプを決定する。リード・サイクルは一般にR E A D_ O U T_ P K T m*信号の1つのアサートによって指示され、ライト・サイクルは通常W R I T E_ I N_ P K T m*信号の1つのアサートによって指示される。同時リード及びライト・サイクルは、R E A D_ O U T_ P K T m*信号とW R I T E_ I N_ P K T m*信号の同時のアサートによって指示される。E P S M 2 1 0は、例えば、後に詳述するように両
30
ポートともカットスルー（C T）モードで動作すべくコンフィギュレーションされている場合のみのような、特定の条件の下で2つのポート間で同時リード及びライトを行う。

【0044】

同時サイクルの期間中、E P S M 2 1 0は3番目のC L Kサイクルの始まりでR E A D_ O U T_ P K T m*信号の1つをローにアサートしてQCデバイス202の1つまたはT P I 2 2 0を指示し、3番目のC L Kサイクル中にP O R T_ N O [1:0]信号上に当該のポート番号をアサートして、アサートされた特定のR E A D_ O U T_ P K T m*信号で識別されるQCデバイス202の4ポートのうちの1つを指示する。特定のR E A D_ O U T_ P K T m*信号で識別されるQCデバイス202は、3番目のC L KサイクルにおいてP O R T_ N O [1:0]信号をラッチし、読み出される特定のポートを判断する。
40
例えば、QCデバイス202を実施するQ E 1 1 0デバイスは、P O R T_ N O [1:0]信号をラッチするリード・ラッチ324を用いて構成される。また、T P I 2 2 0は同様のリード・ラッチ819b（図31）を含み、これは、もしR E A D_ O U T_ P K T [6]*信号で指示されていれば、3番目のC L KサイクルにおいてP O R T_ N O [1:0]信号をラッチする。あるいは、もしQCデバイス202の機能遂行に用いられるデバイスがL 6 4 3 8 1デバイスであれば、外部のラッチがこの目的に使用される。この時点で、識別されたポート0～ポート27の特定のポートがH S B 2 0 6上でリード・サイクルのソース・ポートとして指示されている。

【0045】

E P S M 2 1 0は、次に4番目のC L Kサイクルの始めでW R I T E_ I N_ P K T m*
50

信号の1つをローにアサートして、QCデバイス202の同じ又は他のものまたはTPI220を指示し、4番目のCLKサイクル中にPORT_NO[1:0]信号上に適当なポート番号をアサートし、アサートされた特定のWRITE_IN_PKTm*信号で示されるデバイスの4ポートのうち1つを指示する。特定のWRITE_IN_PKTm*信号で識別されるQCデバイス202は、4番目のCLKサイクルにおいてPORT_NO[1:0]信号をラッチし、書き込まれる特定のポートを判断する。例えば、QCデバイス202の機能を実施するQE110デバイスは、第4のCLKサイクルにおいてPORT_NO[1:0]信号をラッチするためのライト・ラッチ326を用いて構成される。また、TPI220は、もしWRITE_IN_PKT[6]*信号で指示されたならば、4番目のCLKサイクルにおいてPORT_NO[1:0]信号をラッチするための同様のライト・ラッチ819bを含む。このようにして、ポート0～ポート27の他のいずれかのポートがHSB206上のライト・サイクルの宛て先ポートとして指示され、そのライト・サイクルは指示されたばかりのリード・サイクルと同時に実行される。ソース・ポートと宛て先ポートは、同一のQCデバイス202上か、TPI220の2つのポート間か、異なるQCデバイス202間のいずれに存在し得る。しかし、示した実施例においては、同時リード及びライト・サイクルは、QCデバイス202のポート104の1つとTPI220のポート110の1つとの間では、データ転送の速度が違うために実行されない。

【0046】

CLK信号の次のサイクルで、パケット・データはHSB206を介して転送、あるいはソース・ポートから読み出され、直接に宛て先ポートに書き込まれ、その際EPSM210あるいはメモリ212には格納されない。データ転送は、実施例によって異なるが幾つかのバイトを転送するためにサイクル5、6、7、および8で実行される。例えば、L64381デバイスに関しては64バイトまでが転送され、QE110デバイスでは256バイトまでが転送される。データ転送に4つのCLKサイクルを示しているが、送るべきデータの量によっては1、2あるいは4のCLKサイクルで転送される場合もあり得る。新しパケットに関しては、最初に通常のリード・サイクルが実行されてソースおよび宛て先のMACアドレスがEPSM210に供給され、これが後に詳述するハッシュ手順を実行し、もし既知であれば、宛て先ポート番号を決定する。受信先（宛て先）ポート番号が分かり、そしてもし宛て先ポートが1つだけであれば、必要に応じてパケットの残存部分のいずれかの部分あるいは全部について、同時リード及びライト動作を実行することができる。

【0047】

もしPORT_NO信号が同じであるが、2つの異なったポート間であり、従って2つの異なったQCデバイス202間であるならば、特殊なタイプの同時リード及びライト・サイクルが実行される。図9ではこのケースも図解しているが、サイクル全体を通してPORT_NO信号が不変のままであるという点が例外である。PORT_NO信号が変わらないので、ラッチ324、326は不要である。従って、このタイプの同時サイクルは2つの異なるL64381デバイス間で、外部にラッチや選択ロジックを必要とせずに実行することができる。EPSM210は、送信元（ソース）と宛て先のポート間でPORT_NO信号が等しいこと、および2つの異なったQCデバイス202が用いられることを判断してから、説明したように同時サイクルを実行する。

【0048】

図9に図解されているように、2回目の同時リード及びライト転送は6番目のCLKサイクルで発生し、PORT_NO[1:0]信号が7番目、8番目および9番目のサイクルにおいて、それぞれ、リード・モード、リード・ポート番号、およびライト・ポート番号でアサートされる。それに応答して、READ_OUT_PKTm*信号は7番目のCLKサイクルに対してデアサート（de-assert）される。同様に、WRITE_IN_PKTm*信号は8番目のCLKサイクルに対してデアサートされる。この2回目の同時サイクルは、同一データ・パケットの続きのない連続したデータを供給するための最初の同時

10

20

30

40

50

サイクルの続きか、あるいはまったく異なったデータ・パケットの開始のいずれかである。同一パケットの連続したデータについては、ソースおよび宛て先のポートは同じである。しかし、ソース・ポートまたは宛て先ポートあるいはその両方は、異なるパケットのデータを転送する2回目の同時サイクルでは同一のものではないこともある。

【0049】

図10は、HSB206上で同時リード及びライト・サイクルを実行する手順を示すフローチャートである。最初のステップ330で、E P S M 2 1 0は、ソース・ポートと宛て先ポートの間でのHSB206上での同時リード及びライト・サイクルが実行可能かどうかを判断する。E P S M 2 1 0は、それから次のステップ332で、ソース・ポートを識別するための適当な信号をアサートする。これは、HSB206上でP O R T _ N O信号を用いてソースまたは「リード」ポートの番号をアサートすることによって、及び適当なR E A D _ O U T _ P K T m *信号をアサートすることによって行われる。次のステップ334では、識別されたソース・ポート・デバイスがその識別（アイデンティフィケーション）信号を検知もしくは格納する。ラッチを伴わない特殊な同時サイクルでは、Q C デバイス202がHSB206上でR E A D _ O U T _ P K T m *信号を検知し、続いてP O R T _ N O信号を検知して、リード・サイクルの準備を開始する。ラッチを用いる一般的な同時サイクルでは、指示されたQ C デバイス202あるいはT P I 2 2 0がステップ334でリード・ポート番号をラッチし、リード・サイクルの準備を開始する。

【0050】

次のステップ336では、E P S M 2 1 0は宛て先ポートを識別するための適当な信号をアサートする。特殊な同時サイクルでは、E P S M 2 1 0は適当なW R I T E _ I N _ P K T m *信号をアサートし、同じP O R T _ N O信号を維持する。一般の場合では、ステップ336において、E P S M 2 1 0はまた、HSB206上に宛て先または「ライト」ポート番号を適当なW R I T E _ I N _ P K T m *信号とともにアサートする。続くステップ338では、識別された宛て先ポート・デバイスがその識別信号を検知もしくは格納する。ラッチを伴わない特殊な同時サイクルでは、示されたQ C デバイス202がHSB206上でW R I T E _ I N _ P K T m *信号を検知し、続いてP O R T _ N O信号を検知して、ライト・サイクルの準備を開始する。一般的な場合では、指示されたQ C デバイス202あるいはT P I 2 2 0が、ステップ338、で宛て先またはライト・ポート番号をラッチする。最後に、同時リード及びライト・サイクルのステップ340で、ここで指示されたソース・ポートがHSB206上にデータを送出し、指示された宛て先ポートがHSB206からデータを読み取る。

【0051】

同時リード及びライト動作は、パケット・データの各転送にただ1つのバスしか必要としないため、最速タイプのデータ転送サイクルである。後に詳述するように、通常のC T モードの動作では少なくとも2回の転送が必要である。すなわち、1つはソース・ポートからE P S M 2 1 0へ、そしてもう1つはE P S M 2 1 0から宛て先ポートへの転送であって、これは同じデータに対してHSB206上で2つの別のサイクルが必要となる。同時リード及びライト・サイクルは、HSB206上で同一のデータについて1回で直接の転送を要し、それによりHSB206の帯域幅が増大する。その他、幾つかの暫定的なC T や蓄積転送（S n F）モードを含むより遅いモードもあり、その場合、パケット・データはメモリ212に書き込まれてから宛て先ポートに転送される。

【0052】

次に図11を参照すると、E P S M 2 1 0の簡略なブロック図で、データの流れとコンフィギュレーション・レジスタを図解している。E P S M 2 1 0は、HSBコントローラ・ブロック（H C B）402、メモリ・コントローラ・ブロック（M C B）404、およびプロセッサ制御ブロック（P C B）406という3つの主要セクションを含む。Q C インタフェース410はHSB206をE P S M 2 1 0のH C B 402に結合する。Q C インタフェース410の他側には1組のバッファ、すなわちF I F O 412が結合されており、これらのF I F O 412には受信F I F O、送信F I F O、および本明細書で後に詳述

10

20

30

40

50

するカットスルーFIFOが含まれる。FIFO412の他側（図12のCTバッファ528を除く）は、MCBインタフェース414を介してMCB404に結合されており、そのMCBインタフェース414は適当なバス420を介してMCB404内のHCBインタフェース418に結合されている。HCBインタフェース418はさらにメモリ・インタフェース422に結合され、メモリ・インタフェース422はメモリ・バス214を介してメモリ212に結合される。メモリ・インタフェース422はさらにPCBインタフェース424の一侧に結合されており、そのPCBインタフェース424の他側は適当なMCBバス428を介してPCB406内のMCBインタフェース426の一侧に結合されている。MCBインタフェース426の他側は1組のFIFO430の一侧に結合されており、FIFO430がさらにPCB406内のCPUインタフェース432に結合されている。CPUインタフェース432はQC/CPUバス204およびCPUバス218に結合される。CPUインタフェース432はさらにPCB406内の第2の組のFIFO434の一侧に結合されており、FIFO434の他側はQC/HCBインタフェース436に結合されている。QC/HCBインタフェース436の他側は適当なHCBバス438を介してQCインタフェース410に結合されている。

10

【0053】

PCB406とCPU230に関連するHCBバス438のPCB__BUF__AVAIL*、PCB__PKT__AVAIL*、PCB__RD__OUT__PKT*、およびPCB__WR__IN__PKT*信号は、それぞれ、BUF__AVAILm*、PKT__AVAILm*、READ__OUT__PKTm*、およびWRITE__IN__PKTm*信号に含まれていることに留意されたい。示した本実施例において、HCBバス438はHSB206と類似しており、本質的にはEPSM210内のHSB206の内部バージョンである。PCB406は、HCB402に対してポート104のそれぞれおよびTP1220と同様の働きをする。このようにして、CPU230はPCB406の動作を通じて、HCB402に対する追加的なポート（PORT28）として動作する。

20

【0054】

CPUインタフェース432はバス442を介してレジスタ・インタフェース440に結合され、レジスタ・インタフェース440はさらにレジスタ・バス444に結合される。レジスタ・バス444はHCB402内の1組のHCBコンフィギュレーション・レジスタ、およびMCB404内の1組のMCBコンフィギュレーション・レジスタ448に結合される。このようにして、CPU230は、CPUインタフェース432とレジスタ・インタフェース440を介し、HCBコンフィギュレーション・レジスタ446およびMCBコンフィギュレーション・レジスタ448の両方のレジスタの初期設定とプログラミングを行う。

30

【0055】

MCBコンフィギュレーション・レジスタ448は、ポートおよびメモリ212に関連する相当量のコンフィギュレーション情報の格納に使用される。例えば、MCBコンフィギュレーション・レジスタ448は、各ポートが学習（LRN）状態か転送（FWD）状態かブロック（閉じた）（BLK）状態か聴取（LST）状態か又はディスエーブル（DIS）状態かを示すポート状態情報、メモリ・セクタ情報、メモリ・バス214のバス使用情報、ドロップされたパケットの数、ハッシュ・テーブル定義、メモリ・スレッシュولد、BCスレッシュولد、もしあれば機密保護ポートのアイデンティフィケーション、メモリ制御情報、MCB割り込みソース・ビット、割り込みマスクビット、ポーリング・ソース・ビットなどを含む。

40

【0056】

EPSM210の説明では、CPU230はコンフィギュレーションおよび制御の目的で、QCデバイス202およびメモリ212にアクセスできることを述べている。EPSM210とのHSB206を用いる主たるデータ・フローはFIFO412とメモリ212を通じてのものであるが、HSB206とCPU230の間でも、HCBバス438およびEPSM210の関連するFIFO及びインタフェースを介したデータ・フローも発

50

生する。

【0057】

次に図12を参照すると、HCB402の詳細なブロック図が示されている。HCBバス438はPCB406にインタフェースするためのHSB206の内部バージョンであり、そこでバス206と438を一括してHSB206と呼称する。ポーリング・ロジック501は、HSB206、1組のローカル・レジスタ506、およびHCBコンフィギュレーション・レジスタ446に結合されている。ポーリング・ロジック501は、CLK信号を受信し、ポート104、110およびPCB406を問い合わせるべくQCデバイス202およびTPI220へのSTROBE*信号を周期的にアサートする。ポーリング・ロジック501は、QCデバイス202およびTPI220からの多重化されたPKT__AVAILm*およびBUF__AVAILm*信号をモニタする。ここで、各QCデバイス202およびTPI220は、前述したように、その4つのポート104、110の状態をそれぞれ供給する。TPI220はPKT__AVAIL[6]*およびBUF__AVAIL[6]*信号で応答し、PCB406はPKT__AVAIL[7]*およびBUF__AVAIL[7]*信号で応答する。

10

【0058】

ポーリング・ロジック501は受信(RX)ポーリング状態マシン502を含み、これでPKT__AVAILm*信号を見直し(レビューし、review)、レジスタ506内の受信リスト(RECEIVE LIST)509を更新する。同様に、ポーリング・ロジック501は送信(TX)ポーリング状態マシン503を含み、これでBUF__AVAILm*信号を見直し、レジスタ506内の送信リスト(TRANSMIT LIST)510を更新する。もしHCBコンフィギュレーション・レジスタ446におけるWTPRIORITYフラグがCPU230によってセットされれば、RXポーリング状態マシン502およびTXポーリング状態マシン503の両方は、HCBコンフィギュレーション・レジスタ446内の1組のウェイト・ファクタ(WEIGHT FACTORS)508を使用して、後に詳述するように、それぞれ受信リスト509および送信リスト510をプログラミングする。HCBコンフィギュレーション・レジスタ446はまた1組のCT__SNFレジスタ507を含んでおり、これがCPU230にプログラミングされ、対応するポートがソース・ポートあるいは宛て先ポートである場合、所望される動作モードをCTとSnFとの間で決定する。

20

【0059】

レジスタ506は、ラッチ、フリップ・フロップ、スタティックRAM(SRAM)、DRAMデバイスなどのような、EPSM210の実施に従っての任意の様式で実施され、複数の状態および制御(コントロール)のレジスタ又はバッファを含む。受信リスト509は、各ポートの相対的受信状態(ステータス)および優先度(優先順位)を示す複数のレジスタ値を含む。同様に、送信リスト510は、各ポートの相対的送信ステータスおよび優先度を示す複数のレジスタ値を含む。PRカウント(RPCOUNT)・レジスタ511aは、各ポートが外部のネットワーク・デバイスからパケット・データを受信したとき、その受信ポートに相対的な受信優先順位を割り当てるためにRXポーリング状態マシン502によって使用されるPRカウント(RPCOUNT)番号を格納している。もしくは、RXポーリング状態マシン502はウェイト・ファクタ508からの対応するウェイト(重み)・ファクタを使用する。同様に、TPカウント(TPCOUNT)・レジスタ511bは、ポートによって外部のネットワーク・デバイスへ送信できるパケット・データがあり、ポートが送信のためのデータを収容可能なとき、そのポートに相対的な送信優先順位を割り当てるためにTXポーリング状態マシン503によって使用されるTPカウント(TPCOUNT)番号を格納する。もしくは、TXポーリング状態マシン502はウェイト・ファクタ508からの対応するウェイト・ファクタを使用する。相対的アービトレーション・カウント番号RXニューカウント(RXNEWCNT)、RXACTカウント(RXACTCNT)、TXニューカウント(TXNEWCNT)およびTXCTカウント(TXCTCNT)は、それぞれ、レジスタRXニューカウント511c、RXACTカウント511d、TXニューカウント511eおよびTXCTカウント511fに格納される。

30

40

50

【0060】

HCB402は、レジスタ506および446内のデータを調べてHSB206上で実行されたサイクルのタイプを判断するために結合されたアービトレーション・ロジック504を含む。HSBコントローラ505は、EPSM210とHSB206との間のデータ・フローをコントロールするために、HSB206上で実行される各サイクルを実行及び制御する。HSBコントローラ505は、状態ビットを変更するためにレジスタ506に結合される。HSBコントローラ505は、各サイクルのタイプのアイデンティフィケーションをアービトレーション・ロジック504から受け取る。アービトレーション・ロジック504は、新パケット受信(RX NW)アービタ513、受信アクティブ(RX ACT)アービタ514、新パケット送信(TX NW)アービタ515および送信カットスルー(TX CT)アービタ516の全部で4つのデータ・アービタに結合されたメイン(MAIN)・アービタ512を含む。メイン・アービタ512は、一般に、RX NWアービタ513、RX ACTアービタ514、TX NWアービタ515、およびTX CTアービタ516の間で選択を行い、各アービタは調停(仲裁)を行って次のサイクルを決める。メイン・アービタ512は、必要に応じて条件に適ったいずれかの優先順位スキームを用いる。例えば、示した実施例においては、メイン・アービタ512はラウンドロビン優先順位スキームを採用する。

10

【0061】

FIFO412は任意の望ましい様式で実施される。示した実施例においては、2つの受信バッファ、RX BUF520および522でRX FIFOを実現しており、データは1つのバッファへの書き込み中に他のバッファから読み出され、また、その逆も行われる。また、2つの送信バッファ、TX BUF524および526が用意されており、RX BUF520および522と同様に動作する。FIFO412は、少なくとも1つのカットスルー・バッファ、CT BUF528も含む。RX BUF520および522は両方とも64バイトのバッファであり、それぞれが両方向のデータ・フローを実現するためにHSB206との双方向データ・インタフェース、およびRX MCBインタフェース530を介してMCB404にデータを送るための単向インタフェースを含む。TX BUF524および526は両方とも64バイトのバッファであり、HSB206とTX MCBインタフェース531との間に結合されている。TX BUF524および526は、TX MCBインタフェース531を介してMCB404からデータを受け取り、データをHSB206に送る。CT BUF528は64バイトのバッファであり、HSB206との双方向インタフェースを有する。FIFOコントロール・ブロック529は、FIFO520、522、524、および526のデータ・フローを制御するため、及びRX MCBインタフェースおよびTX MCBインタフェース530、531を介してアサートされた特定の状態信号を検知するため、及び後に詳述するように、レジスタ506内の特定のビットをセットするために、レジスタ506、HSBコントローラ505、RX BUF520と522、TX BUF524と526、CT BUF528、RX MCBインタフェース530およびTX MCBインタフェース531に結合している。

20

30

【0062】

バス420は、RX MCBインタフェース530、TX MCBインタフェース531、ハッシュ要求ロジック(ハッシュ・リクエスト・ロジック)及びMCBインタフェース(HASH REQ LOGICと呼ぶ)532、および送信アービタ要求ロジック(TX ARB リクエスト・ロジック)及びMCBインタフェース(TX ARB REQ LOGICと呼ぶ)533を介してHCB402をMCB404にインタフェースするための複数のデータおよび制御信号を含む。HSBコントローラ505は、ポート0~ポート28の1つからのそれぞれの新しいパケットのヘッダをRX BUF520と522の1つに、及びHASH REQ LOGIC532にコピーする。ヘッダは、サイズが少なくとも3つのDWORD(それぞれ32ビット)すなわち96ビットであり、ソースと宛て先の両方のMACアドレスを含む。HASH REQ LOGIC532は、MCB404によって実行されるハッシュ

40

50

ユの手順を要求し、適当なビットをレジスタ506にセットする。このハッシュ手順は、パケットに対して取られる適当な動作を決定するために行われる。

【0063】

示した実施例において、新しパケットのヘッダを受け取った後、HASH REQ LOGIC532はMCB404へ信号HASH_REQ*をアサートし、HASH_DASA[15:0]信号上に48ビットのMACの宛て先およびソース・アドレスおよび8ビットのソース・ポート番号を多重化する。MCB404はHASH_REQ*信号を検知し、ハッシュ手順を実行し、そしてHASH REQ LOGIC532へ信号HASH_DONE*をアサートする。MCB404は、状況が許せば、信号HASH_DSTPRT[4:0]、HASH_STATUS[1:0]、及びHASH_BP*もアサートする。HASH_STATUS[1:0]信号は次の4種類の結果の1つを表示する。すなわち、それらは、パケットをドロップするための00b=DROP_PKT(bは2進数を示す)、ブロードキャスト(同報通信)(BC)パケットに対しての01b=GROUP_BC、宛て先ポートが未知であり、従ってBCパケットであるという10b=MISSED_BC、および単一の宛て先ポートへのユニキャスト・パケットを示す11b=FORWARD_PKTである。もしHASH_STATUS[1:0]=FORWARD_PKTであれば、HASH_DSTPRT[4:0]信号が、そのパケットの宛て先ポートを指定する2進数のポート番号とともにアサートされる。HASH_BP*信号は、もし背圧がイネーブルとなっていて適用可能であれば、MCB404が判断したメモリ212におけるスレッシュールド・オーバーフロー状態に起因して、背圧を示すためにアサートされる。

10

20

【0064】

一定のしきい(スレッシュールド)値が、メモリ212全体に対して、特定のタイプのパケット(例えばBCパケット)に対して、及びポートごとに設定される。しきい値に達したとき、従ってメモリ212にもう1つのパケットを入れるとスレッシュールド条件を侵すことになる場合、そのパケットのドロップの如何はネットワーク・スイッチ102が決定する。送信側のデバイスは最終的にそのパケットがドロップされたことを検知し、そのパケットを再送信する。或るスレッシュールド条件に違反があった場合、もし背圧がイネーブルとなっていてソース・ポートが半二重モードで動作していれば、HASH_BP*信号がアサートされる。

30

【0065】

HASH REQ LOGIC532はHASH_BP*信号を検知して、例えばソース・ポートと宛て先ポートが同じかのように、HASH_STATUS[1:0]=DROP_PKTであるかどうかを判断する。もしHASH_STATUS[1:0]=DROP_PKTであれば、そのパケットはドロップされるべきものであるから、それ以上の動作は不要である。もしHASH_STATUS[1:0]とDROP_PKTが等しくなければ、HASH REQ LOGIC532はHASH_STATUS[1:0]=FORWARD_PKTであるかどうかを判断し、そのパケットはCT BUF528を介してCTモードで転送されることになり、可能性としてメモリ212が避けられる。もし宛て先ポートが使用中(ビジー)であるか、またはもしHASH_STATUS[1:0]がパケットのドロップあるいは転送を指示しなければ、HASH REQ LOGIC532が、データを受信するポートに対して背圧サイクルを実行するようHSBコントローラ505に指示する。

40

【0066】

SnF動作の間、EP SM210は、パケットのいずれかの部分を宛て先ポートへ送信する前に、パケット全体を受信してメモリ212に格納する。パケットの受信が完了後であって、もし宛て先ポートが既知であれば、そのパケットは、使用されている特定のアービトレーション・スキームに従って、可能なときに宛て先ポートに送られる。CT動作を適用する場合、両方のポートがCT_SNFレジスタ507内でCTモードにプリセットされ、両ポートが同一速度で動作し、そして宛て先ポートのTBUS設定がソース・ポート

50

のT B U S設定と比べて等しい又は大きい。1 0 0 M b p sのイーサネット・ポート、ポート2 4～ポート2 7の実現にT L A N 2 2 6を使用するここに示した特定の実施例において、T L A Nは送信に先立ってパケット全体のサイズが必要であるため、ポート2 4～ポート2 7についてC Tモードは実行されない。また、示した実施例ではT B U Sの値が等しいことが要件である。本発明は、これら様々な設計上の問題には制約されない。C Tモードでの動作中、E P S M 2 1 0は指示された宛て先ポートに対して、もしこれがビジーでなければ、データを送信するために、データを適当なQ Cデバイス2 0 2に供給する。パケット・データはメモリ2 1 2には転送されず、ソース・ポートと宛て先ポートの間のF I F O 4 1 2を通じて緩衝格納（バッファ記憶）される。

【0067】

もし受信したパケットの先頭で受信先のポートがビジーであれば、データは暫定的（interim）C T動作モードに従って、ソース・ポートと宛て先ポートの間のメモリ2 1 2内でバッファされる。しかし、パケット部は宛て先ポートによる送信に直ちに使用可能であって、宛て先ポートへの転送に、パケット全体の受け取り完了を待つ必要がない。安全対策として、暫定的C T動作モードを無効とし、その特定のパケットのための動作を次のパケットのS n Fモードに切り替えるメカニズムが適用できる。

【0068】

C Tモードでのパケット転送中に、例えば宛て先ポートの停止のような何らかの理由で宛て先ポートがそれ以上のデータの受信をできなくなった場合、動作はミッドパケット（mid-packet）暫定C Tモードに切り替えられる。ミッドパケット暫定C Tモードの間、F I F O 4 1 2内のパケット・データはメモリ2 1 2へ送られ、その後宛て先ポートがさらにデータを受信することができるときにパケット・データがそのポートに送られる。他の後続の受信されたパケットが、同じ停止したポートによる送信のために他のポートによって受信され、これら後続のパケットはそのポートに対する対応する送信チェーン内に入れられるので、ミッドパケット暫定C Tモードに切り替えられたパケットの残りの部分は順序の適性化を意図してその送信チェーンの先頭に置かれることに留意されたい。

【0069】

もう1つのモードは適応（アダプティブ）S n Fモードと呼ばれる。パケットがC T動作モードで転送されている間、C P U 2 3 0は、ポート1 0 4、1 1 0、およびP C B 4 0 6のいずれか1つまたはそれ以上に「ラント（runt）」、「オーバーラン」、「ジャバ（jabber）」、遅刻衝突（レイト・コリジョン、late collision）、F C Sエラーなどの誤りが相当回数発生するかどうかを判定するために、それらのアクティビティの監視及び追跡を行っている。ラントはデータが一定の最少量に満たないパケットで、示した本実施例におけるその最小サイズは6 4バイトである。オーバーランはデータが一定の最多量より多いパケットで、イーサネット標準に従って示されている本実施例におけるその最大サイズは1 5 1 8バイトである。ジャバはサイズが最大サイズ（イーサネットではの1 5 1 8バイト）を超えており、無効なC R C（巡回冗長検査、（Cyclic Redundancy Check））値が入っているパケットである。通常、このような誤りのあるパケットはドロップされ、システム内に伝播されることはない。適応S n Fモードについては、もしポート1 0 4がC Tモードで動作していて、このような誤りの発生がC P U 2 3 0が判断するところでは頻繁であると、C P U 2 3 0は誤りが訂正または除去されるまで、そのポートにプリセットされているモードをC T動作からS n F動作に切り替える。各T L A N 2 2 6のポート1 1 0の動作も同様であるが、パケット・データがT P I 2 2 0を通じてH S B 2 0 6を介してE P S M 2 1 0に入り、送信の前にメモリ2 1 2に格納されるという点が異なる。T P I 2 2 0は、實際上P C Iバス2 2 2とH S B 2 0 6との間のブリッジとして動作する。T L A N 2 2 6が外部のネットワークにパケットを送信する前にはパケット全体の長さが必要であり、従って、各パケットはT L A N 2 2 6の1つによって再送信される前に、そのパケットが受信されてその全体がメモリ2 1 2に格納される。さらに、Q Cデバイス2 0 2による送信用にT L A N 2 2 6が受け取るデータ、およびT L A N 2 2 6による送信のためにQ Cデバイス2 0 2が受け取るデータは、示した実施例におけるデバ

10

20

30

40

50

イス202と226の間の速度の差が大きいため、SnFモードで処理されてメモリ212に格納される。

【0070】

RX MCBインタフェース530は、パケット・データがRX BUF520、522の1つに入っていてメモリ212への転送準備完了状態にあるとき、MCB404へRX__PKT__AVAIL*信号をアサートする。パケット・データはHCB402から転送され、メモリ・データ出力バスMemDataOutあるいはMDO[31:0]を介してMCB404へ転送される。スタティック信号MEM_EDOは、メモリ212のタイプがEDOが同期DRAMであればアサートされ、FPM DRAMであればアサートされない。RX MCBインタフェース530は、RX__PKT__AVAIL*信号を適宜にアサートしている間、他のいくつかの信号もアサートする。特に、RX MCBインタフェース530は、1CLKサイクルに対してRX__SRC__DST[4:0]信号上にソース・ポート番号を多重化し、続いて、RX__PKT__AVAIL*信号をアサートしているときに、次のCLKサイクルの間に、もし既知であれば、宛て先ポート番号を多重化する。また、RX MCBインタフェース530は、選択されたRX BUF520、522内の、信号RX__CNT[5:0]上のDWORDの数（マイナス1DWORD）をアサートする。

10

【0071】

RX MCBインタフェース530は、もしデータがパケットの始まりであれば信号RX__SOP*をRX__PKT__AVAIL*信号とともにアサートし、もしデータがそのパケットの終わりであれば信号RX__EOP*をRX__PKT__AVAIL*信号とともにアサートする。RX MCBインタフェース530は、パケットがCTモードで転送中であるが、暫定CTやミッドパケットCTモードの場合のようにメモリ212で緩衝格納されていれば、信号RX__CUT__THRU__SOP*を信号RX__PKT__AVAIL*およびRX__SOP*とともにアサートする。特に、もし(!RX__CUT__THRU__SOP* & !RX__PKT__AVAIL* & !RX__SOP*)であれば暫定CT（全パケット）が指示され、もし(!RX__CUT__THRU__SOP* & !RX__PKT__AVAIL* & RX__SOP*)であれば暫定CTミッドパケットが指示される。RX MCBインタフェース530は、もし宛て先アドレスが未知であり、そしてパケットがBCパケットであれば、信号RX__MISS__BC*をRX__PKT__AVAIL*およびRX__SOP*信号とともにアサートする。RX MCBインタフェース530は、もしヘッダ内でGROUP（グループ）ビットがセットされていれば、従って、また。パケットがBCパケットであれば、信号RX__GROUP__BC*をRX__PKT__AVAIL*およびRX__SOP*信号とともにアサートする。RX MCBインタフェース530は、信号RX__END__BYTE[1:0]をRX__PKT__AVAIL*およびRX__EOP*信号とともにアサートし、パケット内の最終バイトのバイト・レーン(lane)を示す。

20

30

【0072】

RX MCBインタフェース530は、ソース・ポートが送信中にABORT__OUT*信号をアサートしてパケット内における誤りの検知と表示を行えば、信号RX__ERROR*をRX__PKT__AVAIL*およびRX__EOP*信号とともにアサートする。FIFOオーバーラン、ラント・パケット、オーバーサイズのパケット、フレーム・チェック・シーケンス(FCS)・エラー、あるいはフェーズ・ロック・ループ(PLL)エラーの検知のように、各種のエラー状況がポート104、110によってチェックされる。もしRX__ERROR*信号がアサートされると、ネットワーク・スイッチ102は、パケットがSnFモードで転送中であれば、そのパケットをドロップする。

40

【0073】

MCB404は、アサートされたRX__PKT__AVAIL*信号を検知した後、そして前述のようにRX__PKT__AVAIL*信号でアサートされた関連の信号をラッチした後に、HCB402へのRX__ACK*信号をアサートする。MCB404は、次のDW

50

ORDのデータを受け取れる状態に入ったときRX__STB*信号をアサートする。MCB404は、HCB402がデータを要求する可能性があると判断したとき、信号RX__PKT__COMPLETE*をアサートする。とりわけMCB404は、CTモードのパケットに対してHCB402によってアサートされたRX__SOP*信号を検知した後にRX__PKT__COMPLETE*信号をアサートする。またMCB404は、SnFモードのパケットに対してHCB402によってアサートされたRX__EOP*信号を検知した後にRX__PKT__COMPLETE*信号をアサートする。MCB404は、SnFパケットに関してRX__ERROR*信号がアサートされていた場合(RX__SOP*信号とともにアサートされていないRX__CUT__THRU*信号で示される状態)には、RX__PKT__COMPLETE*信号をアサートしない。MCB404は、MCB404が判断したメモリ212におけるオーバーフロー状態に起因してパケットがドロップされた場合、RX__PKT__COMPLETE*信号の代わりにHCB402へ信号RX__PKT__ABORTED*をアサートする。

10

【0074】

TX__ARB__REQ__LOGIC533は、使用可能な宛て先ポートによる送信のためのデータのメモリ212からの取り出し要求を、アービトレーション・ロジック504から受け取る。この要求は、一般にTX__NW__ARB__515から出される。TX__ARB__REQ__LOGIC533は、応答してMCB404へ送信要求信号TX__ARB__REQ*をアサートし、一方、信号TX__ARB__PORT[4:0]上に宛て先ポート番号を、及び信号TX__ARB__XSIZE[2:0]に各データ部の最大転送長をアサートする。TX__BUF524および526について、最大転送長は、000b=16バイト、001b=32バイト、010b=64バイト、011=128バイト、および100=256バイトとして定義される。MCB404はこれらの値をラッチし、TX__ARB__REQ__LOGIC533へ肯定応答(アクノレッジ)信号TX__ARB__ACK*をアサートする。MCB404は、要求されたデータをメモリ212から取り出し、そのデータをTX__BUF524、526の1つに書き込む。

20

【0075】

データはメモリ・データ入力バスMemDataInまたはMDI[31:0]を介してHCB402内のTX__BUF524、526へ転送される。TXMCBインタフェース531は、TX__BUF524および526のいずれかがMCB404からのデータの受け取りに使用可能であるとFIFO制御ブロック529が判断したとき、TX__BUF__AVAIL*をアサートする。MCB404は、使用可能なTX__BUF524あるいは526に格納すべくHCB402のTX__MCBインタフェース531によるサンプリングの対象となるデータが存在するとき、ストローブ信号TX__STB*をアサートする。MCB404は、データの特性を識別するためにTX__STB*と同時にいくつかの信号もアサートする。特に、MCB404はTX__STB*信号とともに信号TX__SOP*をアサートし、メモリ212からデータの始めを検出する。MCB404はTX__STB*信号とともにTX__AIFCS*信号をアサートし、送信元ポートがCPU230を指示しているPCB406であるかどうかを判断する。MCB404はTX__STB*信号とともに信号TX__CNT[5:0]上の2進数をアサートする。ここで、TX__CNT[5:0]は選択したTX__FIFOに書き込むDWORDの数(マイナス1 DWORD)を表す。MCB404はTX__STB*信号とともに信号TX__EOP*をアサートし、メモリ212からパケットの終わりを検出する。MCB404はTX__EOP*およびTX__STB*信号とともにバッファ・チェーン終結信号TX__EOBCもアサートし、メモリ212内に特定の受信先ポートに宛てたデータが無くなったかどうかを確認する。MCB404はTX__EOP*およびTX__STB*信号とともにエンド・バイト信号もアサートしてパケット内の最終バイトのバイト・レーンを示す。

30

40

【0076】

BCパケットについては、MCB404がMDI[31:0]信号上のBCビットマップをアサートしつつ信号BC__PORT__STB*をアサートする。FIFO制御ブロック

50

529は、BC_PORT_STB*信号がアサートされたことを検知し、MDI [31:0] 信号をラッチして結果を内部のBCBITMAP [28:0] レジスタ内に格納する。FIFO制御ブロック529は、TRANSMIT_LIST 510内のメモリ・ビット配列TXMEMCYC [28:0] のビットを設定するときにBCBITMAPレジスタ内の値を用いる。

【0077】

図13は、レジスタ506に属するいくつかのレジスタの図解である。CT_SNFレジスタ507は、プログラミング可能な送信元ポート・モードのビット配列SRC_CT_SNF [28:0] を含み、各ビットはそれぞれポートPORT28からPORT0の1つに対応しており、対応するポートが送信元ポートである場合にCTとSnF間における動作モードを指定するためCPU 230によってプログラミングされる。特に、特定のポートにSRC_CT_SNFビットがセットされている場合、そのポートが送信元ポートとして機能するときの動作モードとしてはCTモードが望ましい。SRC_CT_SNFビットがクリアされている場合は、そのポートが送信元ポートとして機能するときの動作モードとしてはSnFモードが望ましい。同様に、CT_SNFレジスタ507は、プログラミング可能な受信先ポート・モードのビット配列DEST_CT_SNF [28:0] を含み、各ビットはそれぞれポートPORT28からPORT0の1つに対応しており、対応するポートがユニキャスト用の受信先ポートである場合にCTとSnF間における動作モードを指定するためCPU 230によってプログラミングされる。CTモードは、送信元と受信先の両方のポートがCT_SNFレジスタ507でCTモードに指定されている場合にのみ望ましい。

【0078】

RECEIVE_LIST 509は、対応する受信優先権カウントを格納する複数のレジスタを含む。優先権カウントはRXPORTBUFx [4:0] カウントと呼ばれ、“x”はポート番号である。最高32のポートに優先権を割り当てるために、示した実施例においては各RXPORTBUFx [4:0] カウントは5ビットである。RECEIVE_LIST 509は対応するポート・マスクビット配列RXPRTMASK [28:0] を含み、それぞれのRXPRTMASKビットは初めにロジック0、すなわち優先権がまだ割り当てられていないとき、そしてそれぞれのPKT_AVALm*信号がその後アサートされたときにRXポーリング・ステート・マシン502によってセットされる。そのとき、RXポーリング状態マシン502は対応するRXPORTBUFxレジスタ内に優先権番号を割り当てる。割り当てられた優先権番号は、そのポートがサービスされるまで有効となっている。RXPRTMASKがセットされている間、RXポーリング状態マシン502は対応するPKT_AVALm*信号のその後のアサートをマスクして、それ以上の要求を無視する。HSBコントローラ505は、新しいパケットの最初の転送以外、それぞれのポートからそのパケットを転送するすべてのリード・サイクル期間中、RXPRTMASKビットをクリアする。HASH_REQ_LOGIC 532は、もしそのパケットがSnFの動作モードで転送すべきものであれば、最初のリード・サイクル転送期間中、RXPRTMASKビットをクリアする。HSBコントローラ505は、もしそのパケットがCTの動作モードで転送されるものであれば、受信先ポートへの最初のライト・サイクル転送期間中、RXPRTMASKビットをクリアする。

【0079】

RECEIVE_LIST 509はインキュー・ビットの配列RXINQUE [28:0] を含み、各ビットは対応するRXPRTMASKビットがセットされたときにセットされる。それぞれのRXINQUEビットは優先権の値が有効であるか否か、そしてもし有効であればその対応するポートがアービトレーション・ロジック504による調停に委ねられるべきものであるかどうかを表示する。RXINQUEビットは、それぞれのポートが新しいパケットあるいはSnFパケットの続きを転送するための次のポートとして指定されるべくMAINアービタ512に付託されたとき、アービトレーション・ロジック504内のアービタによってクリアされる。

10

20

30

40

50

【0080】

RECEIVE LIST 509は、それぞれのポートがメモリ212内にデータを受信すべきかどうかを示すメモリ・ビット配列RXMEMCYC[28:0]を含む。これは、SnFモード、暫定CTモード、および暫定ミッドパケットCTモードでの動作時に行われる指示である。HASH REQ LOGIC 532は、SnFモードまたは暫定CTモードが決定したときに対応するRXMEMCYCビットをセットする。MAINアービタ512は、ミッドパケット暫定CTモードのパケットについて、もし受信先ポートが通常のCTモードの動作中に使用可能なバッファのスペースを表示しなければRXMEMCYCビットをセットする。HSBコントローラ505は、それぞれのポートについて、転送データの最終リード・サイクルでRXMEMCYCビットをクリアする。

10

【0081】

RECEIVE LIST 509は、それぞれのポートが通常のCT動作モードでデータ・パケットを転送しているかどうかを表示するアクティブのCTのビット配列RXACTCYC[28:0]を含む。HASH REQ LOGIC 532は、CTモードのパケットについて対応するRXACTCYCビットをセットする。HSBコントローラ505は、対応するポートに関し、最終パケット・データのリード・サイクルでRXACTCYCビットをクリアする。MAINアービタ512は、ビットがCTモードを指示すべくセットされていて、MAINアービタ512がそのパケットをミッドパケット暫定CTモードのパケットに変更する場合にRXACTCYCビットをクリアする。

【0082】

20

TRANSMIT LIST 510は、対応する送信優先権カウンタを格納する複数のレジスタを含む。優先権カウンタはTXPORTBUFx[4:0]カウンタと呼ばれ、“x”はポート番号である。最高32のポートに優先権を割り当てるために、示した実施例においては各TXPORTBUFx[4:0]カウンタは5ビットである。TRANSMIT LIST 510は対応するポート・マスクビット配列TXPRTMASK[28:0]を含み、それぞれのTXPRTMASKビットは初めにロジック0、すなわち優先権がまだ割り当てられていないとき、そしてそれぞれのBUF__AVAILm*信号がその後アサートされたときにTXポーリング状態マシン503によってセットされる。そのとき、TXポーリング状態マシン503は対応するTXPORTBUFxレジスタ内に優先権番号を割り当てる。割り当てられた優先権番号は、そのポートがサービスされるまで有効となっている。TXPRTMASKがセットされている間、TXポーリング状態マシン503は対応するBUF__AVAILm*信号のその後のアサートをマスクして、それ以上の要求を無視する。HSBコントローラ505は、新しいパケットの最初の転送以外、それぞれのポートからそのパケットを転送するすべてのリード・サイクル期間中、TXPRTMASKビットをクリアする。HSBコントローラ505は、受信先ポートに対するパケット・データ転送のすべてのライト・サイクル期間中、TXPRTMASKビットをクリアする。

30

【0083】

TRANSMIT LIST 510は待ちインキュー・ビットの配列TXINQUE[28:0]を含み、各ビットは対応するTXPRTMASKビットがセットされたときにセットされる。それぞれのTXINQUEビットは優先権の値が有効であるか否か、そしてもし有効であればその対応するポートがアービトレーション・ロジック504による調停に委ねられるべきものであるかどうかを表示する。TXINQUEビットは、それぞれのポートが新しいパケット、あるいはSnFパケットの続きを転送するための次のポートとして指定されるべくMAINアービタ512に付託されたとき、アービトレーション・ロジック504内のアービタによってクリアされる。

40

【0084】

TRANSMIT LIST 510は、それぞれのポートがメモリ212から受け取ったデータを送信すべきかどうかを示すメモリ・ビット配列TXMEMCYC[28:0]を含む。これは、SnFモード、暫定CTモード、および暫定ミッドパケットCTモード

50

での動作時に行われる指示である。F I F O制御ブロック529は、H C B 402からデータを受け取った後M C B 404によるR X _ P K T _ C O M P L E T E * 信号のアサートに応答して、1つまたは複数のT X M E M C Y Cビットをセットする。ユニキャストのパケットについては、T X M E M C Y Cビットが1つのみセットされる。B Cパケットについては、F I F O制御ブロック529がそのB C B I T M A PレジスタによってセットすべきT X M E M C Y Cビットを決定する。S n Fモードのパケットに関しては、パケット全体がメモリ212内への格納のためにM C B 404に転送された後でT X M E M C Y Cビットがセットされる。C Tモードのパケットについては、ミッドパケット暫定モードC Tパケットを含め、M C B 404へのデータの最初のデータ転送中にT X M E M C Y Cビットがセットされる。H S Bコントローラ505は、それぞれのポートへの転送データの最終ライト・サイクルでT X M E M C Y Cビットをクリアする。これはM C B 404が、メモリ212内にはそのポートに対するデータが無くなった旨を示すT X _ E O B C * 信号をアサートした場合も同じで、T X M E M C Y Cビットがクリアされる。

10

【0085】

T R A N S M I T L I S T 510は、R X B U F 520、522の1つにC Tの動作モードでそれぞれの受信先ポートへ直接送信すべきデータがあるかどうかを表示するC Tのビット配列T X C T C Y C [28:0]を含む。H A S H R E Q L O G I C 532は、最初のパケット・データの転送で対応するT X C T C Y Cビットをセットする。H S Bコントローラ505は、対応する受信先のポートに対するデータ転送の最終のライト・サイクルでT X C T C Y Cビットをクリアする。

20

【0086】

T R A N S M I T L I S T 510は、それぞれのポートがC T動作モードでデータ・パケットを転送しているかどうかを表すアクティブのC Tのビット配列T X A C T C Y C [28:0]を含む。H A S H R E Q L O G I C 532は、そのパケットがC Tモードで転送すべきものであると判断すれば、対応するT X A C T C Y Cビットをセットする。F I F O制御ブロック529は、パケットがC Tモードからミッドパケット暫定C Tモードに変更されるとき、メモリ212への格納のためのM C B 404への最初のデータ転送の間にT X A C T C Y Cビットをクリアする。H S Bコントローラ505もパケットの最終転送でT X C T C Y Cビットをクリアする。

【0087】

W E I G H T F A C T O R S 508は、ポートP O R T 0 ~ P O R T 28のそれぞれについてポート・ウェイト・ファクタの配列P O R T W T x [4:0]を含む。“x”は特定のポート番号を表す。P O R T W T ウェイト・ファクタは一意であって、ポートの優先権をユーザがプログラミングできるように、ユーザによって予めプログラミングされていることが望ましい。受信および送信の動作にそれぞれの異なったウェイト・ファクタを定義することができるが、示した実施例においては、受信と送信の両方のケースについて、各ポートに同じウェイト・ファクタを割り当てている。

30

【0088】

図14は、R X受信ポーリング状態マシン502の受信ポーリング動作を表す状態図である。R X受信ポーリング状態マシン502の主たる機能は、P K T _ A V A I L m * 信号のモニタ、優先権カウンタR X P O R T B U F x の割り当て、およびR E C E I V E L I S T 509内のR X P R T M S K ビットの設定である。状態間の移り変わりは、C L K 信号の遷移もしくはサイクルおよびS T R O B E * 信号の状態に基づいている。最初に、パワーアップとコンフィギュレーションで受信優先権カウンタ番号R P C O U N T はゼロに設定され、R Xポーリング状態マシン502は初期アイドルリング状態550に入る。また、P K T _ A V A I L m * 信号に対応するR X I N C C N T B Y [7:0] 論理ビットがクリアされる。R Xポーリング状態マシン502は、S T R O B E * 信号がアサートされない間、すなわちS T R O B E * 信号がハイ、つまりロジック1のときは状態550に留まっている。S T R O B E * 信号がロウにアサートされたとき、動作は1 C L K 待ち状態 (R x P o l l W a i t) 552に移る。

40

50

【0089】

サンプリングで検知されたSTROBE*信号のアサートに応答して、QCデバイス202、TPI 220、およびPCB 406は、ひとつのCLKサイクル後にそれぞれPKT__AVAILm*信号、言い換えればPKT__AVAIL[7:0]*信号の対応する1つをアサートする。このようにして、動作はひとつのCLKサイクル後に状態554に進み、それぞれのPKT__AVAIL[7:0]*信号のポーリングを開始する。動作は状態554から状態556に入り、それから状態558さらに状態560へとCLK信号の経時的なサイクルに追従して移る。動作は状態560から状態554へ戻り、STROBE*信号がアサートされている間はこのループを継続する。しかし、STROBE*信号は周期的であり、1CLKサイクル間抑止され、そして次の3CLKサイクル間再アサートされるのが望ましい。こうして、もしSTROBE*信号がステップ560でディ

10

【0090】

もしステップ554において初期アービトレーション・カウント論理演算が真であれば、それぞれのQCデバイス202およびTPI 220の最初のポート、およびPCB 406について1~9と呼称する9回の論理演算が実行される。ここで、最初の8動作はPORT0、PORT4、PORT8、PORT12、PORT16、PORT20、PORT24、およびPORT28にそれぞれ対応する。8つのポート論理演算1~8の各々について、PKT__AVAILm*信号の対応する1つが対応するRXPRTMASKビットと比較されて要求を受容するかどうかが決定的される。RXPRTMASKビットが予めセットされていない場合にあり得る事象であるが、もし1つのポートについて要求が受け付けられると、そのポートにRXPORTBUFx優先権番号が割り当てられる。また、対応するRXPRTMASKビットがロジック1にセットされてポートからのそれ以上の要求をマスクし、そして対応するRXINCCNTBYビットがロジック1にセットされる。9番目の論理演算はRPCOUNTの増分に実行される。

20

【0091】

PORT0について、もしPKT__AVAIL[0]*信号がアサートされないか、あるいはもしRXPRTMASK[0]がロジック1に等しいと、優先権が既に設定されているのであって、それはPORT0がサービスされるまで変更されることはない。しかし、もしPKT__AVAIL[0]*信号がロウにアサートされ、かつRXPRTMASK[0]がロジック0であれば、対応する優先権カウントRXPORTBUF0はWTPRIORITYフラグがウエイト・ファクタに従って優先権を表示している場合、対応するウエイト・ファクタRXPORTWT0に等しく設定される。しかし、もしWTPRIORITYフラグが偽であれば、RXPORTBUF0はRPCOUNTに等しくセットされる。そして、RXPRTMASK[0]およびRXINCCNTBYビットが両方ともロジック1にセットされる。RXPRTMASK[0]マスクをセットすれば、PORT0のさらなるポーリング要求を受け付けることになる。RXINCCNTBYビットはPKT__AVAIL[0]*信号に対応しており、状態554における残りの論理演算に用いられてPORT0に優先権の値が設定されたことを表示する。

30

40

【0092】

PORT4に対応する2番目の論理演算において、もしPKT__AVAIL[1]*信号がアサートされないか、あるいはもしRXPRTMASK[4]がロジック1に等しいと、優先権が既に設定されているのであって、それはPORT4がサービスされるまで変更されることはない。しかし、もしPKT__AVAIL[1]*信号がロウにアサートされ、かつRXPRTMASK[4]がロジック0であれば、対応する優先権カウントRXPORTBUF4はWTPRIORITYフラグがウエイト・ファクタに従って優先権を表示している場合、対応するウエイト・ファクタRXPORTWT4に等しく設定される。しか

50

し、もしWTPRIORITYフラグが偽であれば、優先権カウンTRXPORTBUF4はRPCOUNTプラスRXINCCNTBY[0]にセットされる。このようにして、もしWTPRIORITYが偽であれば、RXPORTBUF4にはPORT0に優先権の値が設定されていない場合に優先権番号としてRPCOUNTが割り当てられ、PORT0に優先権番号が設定されている場合はRPCOUNT+1の優先権番号が与えられる。これによって、PORT0とPORT4に同じ優先権番号が割り当てられないことが保証される。RXPRTMASK[4]はそれからロジック1にセットされ、さらなるポーリング要求は無視される。このようにして、各ポートに割り当てられる優先権番号は、そのポートに予め決められたウェイト・ファクタであるか、もしくは優先権番号はRPCOUNTに加えてより小さいポート番号と同時に割り当てられた優先権番号を持っているポートの数である。

10

【0093】

次の6つの論理演算は2番目の論理演算と同様である。PCB406に対応する8番目の論理演算において、もしPKT_AVAAIL[7]*信号がロウにアサートされていないか、あるいはもしRXPRTMASK[28]がロジック1に等しいと優先権が既に設定されているのであって、それはPCB406がサービスされるまで変更されることはない。しかし、もしPKT_AVAAIL[1]*信号がロウにアサートされていて、かつRXPRTMASK[28]がロジック0であれば、対応するPCB406の優先権カウンTRXPORTBUF28はWTPRIORITYフラグがウェイト・ファクタに従って優先権を表示している場合、対応するウェイト・ファクタRXPORTWT28に等しく設定される。しかし、もしWTPRIORITYフラグが偽であれば、優先権カウンTRXPORTBUF28はRPCOUNTプラスRXINCCNTBY[6:0]の「ビット合計」に等しくセットされる。RXINCCNTBY[6:0]の「ビット合計」は、その前に7回のポート論理演算において割り当てられた優先権番号の値の数である。従って、PCB406に与えられる優先権番号は予め決められているウェイト・ファクタか、もしくはその優先権番号はRPCOUNTに加えてより小さいポート番号と同時に割り当てられた優先権番号を持っているポートの数である。9番目の論理演算は状態554で実行され、RPCOUNTを状態554において優先権が割り当てられたポートの数に等しいRXINCCNTBY[7:0]のビット合計だけ増分する。この演算により、状態556で実行される1組の論理演算のためにRPCOUNTが増分されることが保証される。

20

30

【0094】

例えば、PKT_AVAAIL[7]*信号の最初の多重化されたビットに関連するすべてのポート、すなわちPORT0、PORT4、PORT8、PORT12、PORT16、PORT20、PORT24、およびPORT28が状態554で同時に要求を出し、RPCOUNTが最初から0のままで、前に設定されていて対応するようなRXPRTMASKビットが存在せず、そしてWTPRIORITYが偽であれば、状態554において対応する優先権カウンTRXPORTBUF_x (x=0、4、8、12、16、20、24、および28) に対し、それぞれ優先権番号0、1、2、3、4、5、6、および7が割り当てられる。それからRPCOUNTが8に等しくセットされる。別の例として、サービスを要求しているポートがPORT4、PORT12、およびPORT20のみの場合、もしWTPRIORITYが偽でRPCOUNTが3に設定されていれば、優先権カウンTRXPORTBUF_x (x=4、12、20) にそれぞれ0、1、および2の優先権番号が割り当てられる。ビット合計の演算によって、複数のポートが同時にサービスを要求しているとき、各ポートに一意の番号を与えられることが保証される。このようにして、優先権番号は先着順、すなわちFCFS (First-Come, First-Served) の優先権スキームに従って割り当てられるが、同時割り当ての処理には特定の順序が予め決められる。

40

【0095】

状態556、558、および560における論理演算は、もし初期アービトレーション・

50

カウント論理演算が真で、それぞれのQCデバイス202およびTPI 220の2番目のポート、つまりポートPORT1、PORT5、PORT9、PORT13、PORT17、PORT21、およびPORT25に関連するPKT_AVAAIL[6:0]*信号に基づいた8つの論理演算が実行され、そして状態554の8番目の論理演算がCPU230へのポートPORT28について繰り返されれば、状態554での論理演算と同様である。状態558において、それぞれのQCデバイス202およびTPI220の3番目のポート、つまりポートPORT2、PORT6、PORT10、PORT14、PORT18、PORT22、およびPORT26に関連する7つの論理演算がPKT_AVAAIL[6:0]*信号に基づいて実行され、状態554の8番目の論理演算がCPU230へのポートPORT28について繰り返される。状態560において、それぞれのQCデバイス202およびTPI220の4番目のポート、つまりポートPORT3、PORT7、PORT11、PORT15、PORT19、PORT23、およびPORT27に関連する7つの論理演算がPKT_AVAAIL[6:0]*信号に基づいて実行され、状態554の8番目の論理演算がCPU230へのポートPORT28について繰り返される。状態556、558、および560のそれぞれにおいて、最後の論理演算が実行されて前述と同様にRPCOUNTがRXINCCNTBYビットのビット合計だけ増分される。

10

【0096】

図19は、TX送信ポーリング状態マシン503の送信ポーリング動作を表す状態図である。TX送信ポーリング状態マシン503の動作はRX受信ポーリング状態マシン502の動作と同様で、状態550、552、554、556、558、および560にそれぞれ相似の状態561、562、564、566、558、および570を含む。しかし、TPCOUNTがRPCOUNTに代わり、初期アービトレーション・カウント論理演算は実行すべき論理演算が残存するかどうかを判断するTPCOUNT番号との比較におけるTXNEWCNTおよびTXACTCNTの増分に基づいて実行される。BUF_AVAAILm*信号がPKT_AVAAILm*信号に代わり、TXPRTMSKビットがRXPRTMSKビットに代わる。また各ポートの等式では、TXPRTMSKビットとTXMEMCYC、TXCTACTCYC、およびTXCTCYCビット配列の対応するビットに基づいた論理項との論理積が求められる。特に、EPSM 210またはメモリ212内に受信先のポートが送信すべきデータがある場合にのみ当該ポートに優先権が割り当てられるよう、TXMEMCYC、TXCTACTCYC、およびTXCTCYCビット配列の論理和が求められる。さらに、TXPORTBUFx優先権カウントがRXPORTBUFx優先権カウントに代わり、TXPORTWTウエイト・ファクタがRXPORTWTウエイト・ファクタに代わり、そしてTXINCCNTBYビットがRXINCCNTBYビットに代わる。このようにして、各ポートおよびPCB 406の表示はSTROBE*信号に応答してBUF_AVAAIL*信号のそれぞれ1つによるものとなり、TXポーリング状態マシン503はTPCOUNTを用い、FCFSあるいはウエイト・ファクタに基づいて優先権番号を割り当て、それに応じて優先権を設定する。

20

30

【0097】

要求しているポートの各々に対する優先権の割り当て、および対応するポーリング・マスクビットの設定に備えてポーリング・ロジック501が周期的あるいは連続的にSTROBE*信号をトグルし、ポート104、110、およびPCB 406のそれぞれのPKT_AVAAILm*およびBUF_AVAAILm*信号を監視する機能は高い評価に値する。割り当てられる優先権は、もしWTPRIORITYが真であれば予めプログラミングされているウエイト・ファクタに基づくか、あるいはもしWTPRIORITYが偽であればFCFSに基づく。与えられた優先権は、そのポートがサービスされるまでそのままに留まっている。後述するように、最終的にそのポートはサービスを受け、そのマスクビットはクリアされる。

40

【0098】

アービタ513～516は、いくつかのアービトレーション・スキームの1つに基づいて

50

ポート104、110、およびPCB 406間における選択を行う。ここで、特定のアービトレーション・スキームをユーザがプログラミングすることも可能である。最初はラウンドロビン法であって、これによりポートがPORT1、PORT2、...、PORT28といったような任意の順序でチェックされるか、あるいはその順序はPORTWTxレジスタ内に予めプログラミングされているWEIGHT FACTORS 508で選択される。示した実施例においては、ラウンドロビン法による割り当てにWEIGHT FACTORSが用いられており、それぞれのRXPORTBUFxおよびTXPORTBUFxカウンタにプログラミングされている。RX NWアービタ513はRXNEWCNT優先権番号を用いてこれを増分し、RX ACTアービタ514はRXACTCNT優先権番号を用いてこれを増分し、TX NWアービタ515はTXNEWCNT優先権番号を用いてこれを増分し、TX CTアービタ516はTXCTCNT優先権番号を用いてこれを増分する。ラウンドロビン法では、RXアービタ513および514は、それぞれRXINQUE[]の値を調べてサービスを要求しているアクティブな受信ポートの存在如何を確認し、それからそのそれぞれの優先権番号(RXNEWCNT、RXACTCNT)をアクティブなポートのRXPORTBUFxカウンタ内の値と比較して次にサービスされるべきポートの有無を確認する。また、TXアービタ515、516は、それぞれTXINQUE[]の値を調べてサービスを要求しているアクティブな送信ポートの存在如何を確認し、それからそのそれぞれの優先権番号(TXNEWCNT、TXCTCNT)をアクティブなポートのTXPORTBUFxカウンタ内のカウンタ値と比較して次にサービスされるべきポートの有無を確認する。WEIGHT FACTORSは特定の順序を決めるので、ポートはラウンドロビンの方式で序列される。

【0099】

2番目のアービトレーション・スキームはFCFSであり、その場合WTPRIORITYは偽であって、ポートはRXPORTBUFxおよびTXPORTBUFx優先権番号で表されているサービスを要求した順序でサービスを受ける。FCFSにおける動作は、前述したようにRXPORTBUFxおよびTXPORTBUFxカウンタがRPCOUNTおよびTPCOUNTの値に従ってプログラミングされる点を除き、ラウンドロビンの動作と同様である。この場合、RXアービタ513および514は、それぞれRXINQUE[]の値を調べてサービスを要求しているアクティブな受信ポートの存在如何を確認し、それからそのそれぞれの優先権番号(RXNEWCNT、RXACTCNT)をアクティブなポートのRXPORTBUFxカウンタ内の値と比較して次にサービスされるべきポートの有無を確認する。また、TXアービタ515、516は、それぞれTXINQUE[]の値を調べてサービスを要求しているアクティブな送信ポートの存在如何を確認し、それからそのそれぞれの優先権番号(TXNEWCNT、TXCTCNT)をアクティブなポートのTXPORTBUFxカウンタ内のカウンタ値と比較して次にサービスされるべきポートの有無を確認する。RPCOUNTおよびTPCOUNTは特定の順序を決めるので、ポートはFCFSの方式で序列される。

【0100】

もう1つのスキームはウェイト優先権スキームであり、その場合WTPRIORITYは真であって、RXPORTWTxおよびTXPORTWTx番号がRXPORTBUFxおよびTXPORTBUFxレジスタの対応する1つにコピーされ、優先権の決定に使用される。しかし、RXアービタ513、514はRX HIGH PRIORITY番号から優先権を決め、TXアービタ515、516はTX HIGH PRIORITY番号から優先権を決定する。RX HIGH PRIORITY番号は、アクティブな受信ポートのRXPORTBUFxカウンタ内における最高の優先権番号(すなわち1番小さい数)を識別することによって決定される。ここで、アクティブな受信ポートはRXINQUEの値で判断される。同様に、TX HIGH PRIORITY番号は、アクティブな送信ポートのTXPORTBUFxカウンタ内における最高の優先権番号(すなわち1番小さい数)を識別することによって決定される。ここで、アクティブな送信ポートはTXINQUEの値で判断される。このようにして、ウェイト・ファクタが最高のアクテ

ィブな（すなわちサービスを要求している）ポートが毎回選択されて加重優先権割り当て法の機能が遂行される。

【0101】

RX NWアービタ513は、ポートPORT0～PORT28で受信されたすべての新しいパケット・ヘッダのデータおよびSnFモードのパケット・データの続きを処理し、そのデータはRX BUF 520、522のいずれか1つに転送される。RX NWアービタ513は、RXNEWCNT番号を更新し、RECEIVE LIST 509をチェックして受信決定基準に合致しているポートはPORT0～PORT28のいずれであるかを判断する。RX NWアービタ513の受信決定基準に適合するポートは、そのRX INQUEビットがアサートされていて、そのRXACTCYCビットがアサートされてい

10

【0102】

TX CTアービタ516は、RX BUF 520、522の中のデータを受信先のポートへ通常のCT動作モードで転送する。TX CTアービタ516は、TXNEWCNT番号を更新し、TRANSMIT LIST 510をチェックして送信決定基準に合致しているポートはPORT0～PORT28のいずれであるかを判断する。TX NWアービタ516の送信決定基準に適合するポートは、それぞれのTX INQUEおよびTXCTCYCビットがアサートされているポートである。TX CTアービタ516は、その送信決定基準を満たしている複数のポート間で、選択した前述のアービトレーション・スキームに従って調停を行う。1つのポートを選択してサイクルを定義した後、TX CTアービタ516は選択したRX BUF 520、または522からデータを選ばれた受信先ポートへ送信すべく、MAINアービタ512に対してライト・サイクルを1回

20

30

【0103】

RX ACTアービタ514は、（RX NWアービタ513が処理する）新しいパケットの1回目のリード・サイクルを除き、後続のパケット・データを通常のCT動作モードで動作している送信元のポートからCT BUF 528へ転送する。RX ACTアービタ514は、RXACTCNT番号を更新し、RECEIVE LIST 509をチェックしてその受信決定基準に合致しているポートはPORT0～PORT28のいずれであるかを判断する。RX ACTアービタ514の受信決定基準に適合するポートは、そのRX INQUEおよびRXACTCYCビットがアサートされており、そのRXMEMCYCビットがアサートされていないポートである。RX ACTアービタ514は、その受信決定基準を満たしている複数のポート間で、選択した前述のアービトレーション・スキームに従って調停を行う。1つのポートを選択してサイクルを定義した後、RX ACTアービタ514は選択された送信元ポートからCT BUF 528へデータを転送すべく、MAINアービタ512に対してリード・サイクルを1回実行するよう要求する。RX ACTアービタ514がMAINアービタ512によって次に選択されたとき、RX ACTアービタ514はサービスを受けるべく選択されたポートのRX INQUEビットをクリアする。このプロセスをRX ACTアービタ514は連続的に繰り返す

40

50

。

【0104】

MAINアービタ512は、CT BUF 528へのCTモードの各リード・サイクルに続いて、CT BUF 528内のデータをHASH REQ LOGIC 532に指示される受信先ポートへ転送するためのライト・サイクルを1回実行する。MAINアービタ512は、RX ACTアービタ514にCTデータをCT BUF 528へ転送させる前に受信先のポートが使用中かどうかをチェックする。MAINアービタ512は、もし受信先ポートが使用中であることを確認すれば、それぞれのRX MEMCYCビットをセットし、送信元ポートのそれぞれのRX ACTCYCビットをクリアし、送信元と受信先のポートの動作モードをミッドパケット暫定CTモードに変更する。

10

【0105】

TX NWアービタ515は、TX BUF 524および526のいずれかから、データをHSB 206へSnFの動作モードで転送する。TX NWアービタ515は、TX NEWCNT番号を更新し、TRANSMIT LIST 510をチェックしてその送信決定基準に合致しているポートはPORT0～PORT28のいずれであるかを判断する。TX NWアービタ515の送信決定基準に適合するポートは、それぞれのTX INQUEおよびTX MEMCYCビットがアサートされており、それぞれのTX ACTCYCビットがアサートされていないポートである。TX NWアービタ515は、その送信決定基準を満たしている複数のポート間で、選択したアービトレーション・スキームに従って調停を行う。1つのポートを選択して、TX BUF 524および526のいずれかから選択された受信先ポートへのライト・サイクルを定義した後、TX NWアービタ515はMAINアービタ512に対してライト・サイクルを実行するよう要求する。TX NWアービタ515がMAINアービタ512によって次に選択されたとき、TX NWアービタ515はサービスを受けるべく選択されたポートのTX INQUEビットをクリアする。このプロセスをTX NWアービタ515は連続的に繰り返す。

20

【0106】

次に図24を参照する。EPSM 210内のMCB 404の詳細ブロック図である。MCB構成レジスタ448は図24に示されていないが以下に説明されており、ここで解説する多数の機能ブロックにより、必要に応じて適切な理解が得られる。MCB 404は、バス420を介してMCBインタフェース414に結合されているハッシュ・コントローラ602を含む。ハッシュ・コントローラ602は、メモリ212から取り出されたデータを格納するハッシュ・キャッシュ・テーブル603をオプションとして含む。ハッシュ・キャッシュ603を使用すれば、メモリ212から最近取り出されたデータに対する速いアクセスが可能となり、最近アクセスされた情報を取り出す場合に、もう一度メモリ・サイクルを実行する必要がなくなる。ハッシュ・コントローラ602は、バス610を介して4入力アドレス・マルチプレクサ(mux)630の1つの複線入力に結合されたアドレス/長さ/状態、AD/LN/ST(Address/Length/Status)出力を含む。AD/LN/ST出力は、メモリ212のアドレス、バースト・サイクルを実行すべきか否かを決定するトランザクションの長さ、およびリード/ライト(R/W)信号、バイト・イネーブル、ページ・ヒット信号、ロック信号といった種々の状態信号を定義する。DRAM要求/許可/ストロブ/制御、DRAM RQ/GT/STB/CTL(DRAM Request/Grant/STrobe/Control)制御628は、DRAMメモリ・アービタ638およびハッシュ・コントローラ602のDRAM RQ/GT/STB/CTL入力に結合されている。mux 630の出力はDRAMメモリ・コントローラ636のAD/LN/ST入力に供給され、DRAMメモリ・コントローラ636はメモリ・バス214を介して、さらにメモリ212に結合されている。ハッシュ・コントローラ602は、DRAMメモリ・コントローラ636からデータ・バス618を介してデータを受け取るためのデータ入力(DIN)を持っている。

30

40

【0107】

RX HCBインタフェース601は、MDO[31:0]信号を含むバス420に結合

50

されており、4入力データ・マルチプレクサ (mux) 632の1番目の複線入力にバス620を介してデータを供給するためのデータ出力 (DOU T) を含む。ここでmux 632は、その出力をDRAMコントローラ636のMemDataOut入力に供給する。RX HCBインタフェース601は、DRAM RQ/GT/STB/CTL信号628のストロブおよび制御信号を受け取るためのSTB/CTL入力を含む。RXコントローラ604はバス420に結合されており、マルチプレクサ630の2番目の入力にバス612を介して結合されているAD/LN/ST出力を持っている。RXコントローラ604は、mux 632の2番目の入力にバス622を介して結合されているデータ出力DOU T、バス618に結合しているデータ入力DIN、静的RAM (SRAM) 650関連のSRAM RQ/GT/STB/CTL信号654を受け取るためのSRAM RQ/GT/STB/CTL入力、およびDRAM RQ/GT/STB/CTL信号628を受け取るためのDRAM RQ/GT/STB/CTL入力を持っている。

10

【0108】

TX HCBインタフェース605は、MDI [31:0] 信号を含むバス420に結合されており、バス618に結合されているデータ入力DINとDRAM RQ/GT/STB/CTL信号628のストロブおよび制御信号を受け取るSTB/CTL入力を持っている。TXコントローラ606はバス420に結合されており、mux 630の3番目の入力にバス614を介して供給されるAD/LN/ST出力、mux 632の3番目の入力にバス624を介して結合されているデータ出力DOU T、バス618に結合されているデータ入力DIN、SRAM RQ/GT/STB/CTL信号654を受け取るためのSRAM RQ/GT/STB/CTL入力、およびDRAM RQ/GT/STB/CTL信号628を受け取るためのDRAM RQ/GT/STB/CTL入力を持っている。PCBインタフェース424は、マルチプレクサ630の4番目の入力にバス616を介して結合されているAD/LN/ST出力、マルチプレクサ632の4番目の入力にバス626を介して結合されているデータ出力DOU T、バス618に結合されているデータ入力DIN、SRAM RQ/GT/STB/CTL信号654を受け取るためのSRAM RQ/GT/STB/CTL入力、およびDRAM RQ/GT/STB/CTL信号628を受け取るためのDRAM RQ/GT/STB/CTL入力を持っている。

20

【0109】

ハッシュ・コントローラ602、RXコントローラ604、TXコントローラ606、PCBインタフェース424、RX HCBインタフェース601、およびTX HCBインタフェース605は、それぞれSTB信号を用いてデータ・フローを同期させるが、STROBE*信号のアサートで、データがいつリード・サイクルに有効であるか、あるいはデータがいつライト・サイクルに取り出されるかを判断する。CTL信号は、例えばデータ・サイクルの完了時を表示する信号のような種々の制御信号である。

30

【0110】

DRAMアービタ638はさらに、メモリ制御信号 (MEMCTL) でDRAMコントローラ636に結合し、マルチプレクサ制御信号 (MUXCTL) をマルチプレクサ 630、632の選択入力に供給する。MEMCTL信号は、一般に各メモリ・サイクルの開始と終了を表示する。このように、ハッシュ・コントローラ602、RXコントローラ604、TXコントローラ606、およびPCBインタフェース424は、それぞれの要求信号をアサートすることによって、メモリ212に対してメモリ・サイクルを実行するためにDRAMコントローラ636へのアクセスの調停を行う。DRAMアービタ638は要求信号を受け取って、要求しているデバイス602、604、606、および424の1つに対応する許可 (GT) 信号をアサートすることにより、そのデバイスに対してアクセスを許可する。いったんアクセスが許可されると、DRAMアービタ638はマルチプレクサ630および632へのMUXCTL信号をアサートし、デバイス602、604、606、および424のうち選択された1つが必要に応じてメモリ・サイクルを実行すべくDRAMコントローラ636に対するアクセスを可能とし、そしてMEMCTL信号

40

50

の1つがアサートされてDRAMコントローラ636に対しサイクルの開始を示す。DRAMコントローラ636は、MEMCTL信号の1つをアサートまたは抑止してメモリ・サイクルの完了を示す。

【0111】

ハッシュ・コントローラ602は、HASH REQ LOGIC 532と交信してハッシュ手順を実行し、HASH REQ LOGIC 532に格納されている新しいパケット・ヘッダの処理方法を決定する。ハッシュ・コントローラ602は、アサートされたHASH__REQ*信号を検知し、HASH__DA__SA[15:0]信号から送信元および受信先のメディア・アクセス制御(MAC)信号を取り出し、HASH__STATUS[1:0]を判定するために、そしてもし受信先のポート番号がメモリ212内に予め格納されていれば、それをHASH__DSTPRT[4:0]上に供給するためにハッシュ手順を実行する。RXコントローラ604およびRX HCBインタフェース601は、RX BUF 520、522からのデータを制御し、メモリ212へ転送する。TXコントローラ606およびTX HCBインタフェース605は、主としてメモリ212からのデータを制御し、TX BUF 524、526へ転送する。PCBインタフェース424によって、CPU 230はメモリ212、およびSRAM 650のメモリ内のデータにより直接的にアクセスすることができる。

10

【0112】

トラローラ604およびRX HCBインタフェース601は、RX BUF 520、522からのデータを制御し、メモリ212へ転送する。TXコントローラ606およびTX HCBインタフェース605は、主としてメモリ212からのデータを制御し、TX BUF 524、526へ転送する。PCBインタフェース424によって、CPU 230はメモリ212、およびSRAM 650のメモリ内のデータにより直接的にアクセスすることができる。

20

【0113】

SRAM 650はSRAMコントローラ652に結合しており、SRAMコントローラ652はさらにRXコントローラ604、TXコントローラ606、およびPCBインタフェース424にバス653を介して結合している。SRAMアービタ651は、制御信号SCTLでSRAMコントローラ652に結合しており、さらにPCBインタフェース424によるSRAM 650へのアクセスを制御するためにSRAM RQ/GT/S TB/C TL信号654、およびDRAMアービタ638によるDRAMコントローラ636へのアクセス制御と同様に、TXコントローラ606およびRXコントローラ604にバス653を介して結合している。

30

【0114】

MCB 404は、本明細書で後に詳述するように、パケット制御レジスタおよびその他のデータを格納するSRAM 650を含む。パケット制御レジスタは、ポートごとのRECEIVE SECTOR CHAIN、ポートごとのTRANSMIT PACKET CHAIN、およびメモリ212の空きメモリ・セクタのFREEPOOL CHAINへの1組のポインタを含む。パケット制御レジスタは、さらにネットワーク102内におけるパケット・データの流れの制御を可能とする制御情報やパラメータを含む。メモリ212は、隣接した同一サイズの複数のセクタで編成されているパケット・メモリ・セクションを含む。これらのセクタは、初期にはアドレス・ポインタ、あるいは同様な手段で相互にリンクされてをFREEPOOL CHAIN形成している。ポートからパケット・データが受け取られると、これらのセクタはFREEPOOL CHAINから取り出され、そのポートのRECEIVE SECTOR CHAINに追加される。さらにそのパケットは、それが送信時に送られるべき1つまたは複数の受信先のポートの1つまたは複数のTRANSMIT PACKET CHAINにリンクされる。バス653によって、RXコントローラ604、TXコントローラ606、およびCPUインタフェース436はメモリ212内のデータのパケット・チェーンへのポインタを含んでいるパケット制御レジスタにアクセスすることができる。

40

50

【0115】

DRAMコントローラ636は、メモリ212内のデータを保持するためのメモリ・リフレッシュ・ロジック660を含む。リフレッシュ・ロジック660は、メモリ・バス214に結合されているFPM DRAM、EDO DRAM、あるいは同期DRAMのような各種のメモリのタイプに従って動作する順応性を備えている。このようにして、CPU 230はリフレッシュの機能が不要となり、動作能率およびパフォーマンスが向上する。MCB構成レジスタ448内にある10ビットのメモリ・リフレッシュ・カウンタ(MRC)は、リフレッシュ要求間のクロック・サイクルの数を定義する。その期間は15.26 μ sに等しいかそれより短いことが望ましい。既定値は208hであり、“h”は16進数を示すが、これによって30 nsのCLKサイクルでのリフレッシュ期間はおおよそ15.60 μ sとなる。MRCカウンタはタイムアウトでDRAMアービタ638への信号REFREQをアサートし、DRAMアービタ638はDRAMコントローラ636へのMEMCTL信号の1つをアサートし、メモリ・リフレッシュ・ロジック660に対しリフレッシュ・サイクルを実行するよう指示する。MCB構成レジスタ448は、メモリ212のメモリのタイプ、速度、および構成を定義するメモリ制御レジスタ(MCR)を含む。例えば、MCRの2ビットはメモリのタイプがFPM、EDO、および同期DRAMのいずれであるかを表す。別の1ビットは、メモリの速度が50および60 nsのいずれであるかを示す。その他のビットは、選択したタイプのDRAMの特定のモードを定義し、パリティ・エラーのような誤りも表示する。

10

【0116】

20

次に図25を参照する。PCB 406の詳細ブロック図である。CPUバス218がCPUインタフェース432の中のCPUインタフェース・ロジック700に結合しており、CPUインタフェース・ロジック700は、さらにQC/CPUバス204とインタフェースするためにバス701を経由してQC/CPUインタフェース702に結合している。CPUインタフェース・ロジック700は、FIFO 430内の16バイトの受信バッファRX BUF 706にデータを供給し、これがMCBバス428上のデータをアサートする。MCBバス428は、CPUインタフェース・ロジック700にデータを供給すべく、これもまたFIFO 430内にある16バイトの送信バッファTX BUF 708にデータを入れる。MCBインタフェース426はCPUインタフェース・ロジック700とMCBバス428との間のデータの流れを制御する。CPUインタフェース・ロジック700は、バス信号703でRX BUF 706、TX BUF 708、およびMCBインタフェース426と結合している。

30

【0117】

CPUインタフェース・ロジック700は、バス442を介してレジスタ・インタフェース440に結合されており、レジスタ・インタフェース440によってEPSM 210内の他の構成レジスタにアクセスが可能となる。CPUインタフェース・ロジック700は、割り込みレジスタ、構成レジスタ、パケット情報レジスタ、メモリ関連のレジスタ、設定/状態レジスタ、インタフェース/監視(モニタ)レジスタ、統計レジスタ、モード・レジスタ、アービトレーション・レジスタなどのような、CPU 230の入出力(I/O)空間を定義する。

40

【0118】

CPU 230は、パワーアップとコンフィギュレーションの間にPCBレジスタ704内の初期値ないしは既定値をプログラミングする。例えば、CPU 230はPCBレジスタ704内のPORT SPEED REGISTERのプログラミングを行うが、これは各ポートの速度を定義するビットマップである。示した実施例では、10または100 MHzである。また、PORT TYPE REGISTERも定義されるが、これはQCとTLAN間のポートのタイプを定義するビットマップである。普通、これらのレジスタは動作中に変更されることはないが、必要に応じて再プログラミングすることもできる。

【0119】

50

PCBレジスタ704のその他のレジスタは動作中に使用される。例えば、PCBレジスタはINTERRUPT SOURCEレジスタおよびPOLLING SOURCEレジスタを含む。INTERRUPT SOURCEレジスタは1組の割り込みビット、MCB__INT、MEM__RDY、PKT__AVAIL、BUF__AVAIL、ABORT__PKT、およびSTAT__RDYを含む。PKT__AVAILおよびBUF__AVAIL割り込みビットは、PCB__PKT__AVAIL*およびPCB__BUF__AVAIL*信号にそれぞれ対応する。少なくとも1つのCPU__INT*信号がCPU 230に用意され、このCPU__INT*信号がアサートされたときCPU 230がINTERRUPT SOURCEレジスタを読み取って割り込み元を特定する。MCB__INT割り込みビットは、割り込みがMCB 404内で発生したことをCPU 230に知らせる。MEM__RDY割り込みビットは、要求されたメモリ212のデータがFIFO 430内に存在することをCPU 230に通知する。PKT__AVAIL割り込みビットは、CPU 230が処理すべきパケット・データが存在することをCPU 230に通知する。BUF__AVAIL割り込みビットは、CPU 230がパケット・データを送るために使用するバッファ・スペースがあることをCPU 230に通知する。ABORT__PKTは、ABORT__IN*信号がアサートされたことをCPU 230に通知する。STAT__RDY割り込み信号は、要求されたQCデバイス202からの統計情報がFIFO 430内に存在することをCPU 230に通知する。POLLING SOURCEレジスタは、割り込みがマスクされてポーリング方式が適用されている場合の、各割り込みビットのコピーを含む。

10

20

【0120】

CPUインタフェース・ロジック700は、FIFO 434内の64バイトの受信バッファRX BUF 710にデータを供給し、これがHCBバス438上のデータをアサートする。FIFO 434内の送信バッファTX BUF 712は、CPUインタフェース・ロジック700にデータを供給すべく、HCBバス438から受信データを受け取る。CPUインタフェース・ロジック700は、バス信号705でRX BUF 710、TX BUF 712、およびQC/HCBインタフェース436と結合されている。QC/HCBインタフェース436は、CPUインタフェース・ロジック700、RXおよびTX BUF 710、712、およびHCBバス438と結合しており、HCB 402とPCB 406との間のデータ転送を制御する。

30

【0121】

図26は、CPUインタフェース700の詳細ブロック図である。CPU制御/状態信号218bは制御ロジック713にアサートされる。制御ロジック713は、CPUトラッカ状態マシン717およびオルターネット・メモリ・コントロール状態マシン718と結合している。CPUバス218のアドレス/データ・ポーション218aは多重化されたバスであり、PCB 406の他の部分からのデータがCPU 230へのCPUアドレス/データ・ポーション218a上でアサートされるべく、バス・イネーブル・ロジック716に供給される。CPU 230はアドレス復号/要求生成ロジック714をアサートし、そのロジック714は複数の要求信号をCPUトラッカ状態マシン717およびオルターネット・メモリ・コントロール状態マシン718を含むPCB 406の他の部分に供給する。1組のCPU情報ラッチ715はCPU 230からアドレスおよびデータを受け取り、本明細書で後に詳述するように、PCB 406の他の部分へのラッチされたアドレスおよびラッチされたデータをアサートする。CPUサイクルを監視し、制御するために、CPU制御信号がアドレス復号/要求生成ロジック714、CPUトラッカ状態マシン717、およびオルターネット・メモリ・コントロール状態マシン718間に供給される。

40

【0122】

図27は、QC/HCBインタフェース・ロジック702の詳細ブロック図である。QC/HCBインタフェース・ロジック702は、CPU 230とQCデバイス202との間で、例えばCPU 230の32ビットとQCデバイス202の16ビット間のフォー

50

マット変換のような、一般に比較的トランスペアレントなインタフェースを実現するように動作する。REGISTER REQUEST信号がアドレス復号／要求生成ロジック714からCPUトラッカ状態マシン717に供給され、CPUトラッカ状態マシン717は、16ビットと32ビット間のフォーマット変換のためにディスアセンブリ／アセンブリ状態マシン722に結合されている。ディスアセンブリ／アセンブリ状態マシン722は、バス701を介してCPUインタフェース700と、およびQC/CPUバス204を介してQCデバイス202とそれぞれインタフェースするために、1組のデータ、アドレス、制御信号ドライバ／レシーバ724に結合している。統計バッファ726は、QC/CPUバス204から統計データおよびその他の情報を受け取り、そのデータをバス701を介してCPUインタフェース700に供給する。STATISTICS REQUEST信号が、アドレス復号／要求生成ロジック714からディスアセンブリ／アセンブリ状態マシン722とQC/CPUバス状態マシン730に結合している、スタティスティクス・リクエスト状態マシン728に供給される。QC/CPUバス状態マシン730はさらに、ディスアセンブリ／アセンブリ状態マシン72および1組のデータ、アドレス、制御信号ドライバ／レシーバ724に結合している。このようにして、ポート104の統計およびその他の情報収集、さらにポート104の構成変更のために、CPU230はデータの流れやHSB 206の動作を妨げることなくQCデバイス202に対して比較的完全で独立したアクセスができるようになっている。

【0123】

CPU230は、PCBレジスタ704内のQC STATISTICS INFORMATIONレジスタに書き込むことによってEPSM 210に対しQCデバイス202から統計および状態情報を取り出すよう要求する。CPU 230は、QCデバイス202の1つに対応する番号、ポート番号、指定したポートに関する開始レジスタの番号、および指定したポートについて読み取るべきレジスタの数を供給して統計情報を要求する。図27に示されるように、QC STATISTICS INFORMATIONレジスタへの書き込みによってQC STATISTICS REQUEST信号がアサートされる。統計リクエスト状態マシン728は、1組のデータ、アドレス、制御信号ドライバ／レシーバ724を経由してQC/CPUバス204上で指示された要求を行う。CPUインタフェース700は、当該のCHIP SELECT_m*信号を用いて1つか複数の当該のQCデバイス202に対して必要なリード・サイクルを実行し、その情報を統計バッファ726に書き込む。

【0124】

要求されたデータがすべて取り出されて統計バッファ726に格納されると、CPUインタフェース700はPCBレジスタ内のPOLLING SOURCEレジスタのSTAT__RDYビットを更新し、INTERERRUPT SOURCEレジスタ内のSTAT__RDY割り込みビットをセットする。EPSM210はCPU230へのCPU__INT*信号をアサートし、CPU230はこれに応答してINTERERRUPT SOURCEレジスタを読み取り、割り込み元を特定する。もし割り込みがマスクされていれば、CPU230はポーリング・ルーチン中にPOLLING SOURCEレジスタのSTAT__RDYビットを検知する。このようにして、CPU230は要求が割り込みか、割り込みがマスクされていればポーリングのメカニズムによって完了したことを判断する。もしポーリング・メカニズムを適用するのであれば、プログラミングによってSTAT__RDY割り込みを必要に応じてマスクすることができる。CPU23は、応答方式により1つまたは連続した複数のプロセッサ・サイクルで、統計バッファ726から統計情報をすべて取り出す。CPUバス218上のプロセッサ・サイクルは標準どおりのプロセッサ・バス・サイクルでよいが、大量データの転送にはバーストのサイクルが望ましい。

【0125】

勿論いくつかの別の実施形態が企図される。第1の別の実施形態においては、CPU230がQCデバイス202のいずれかに対応する番号を供給するだけで、EPSM210が応答してQCデバイス202のすべてのポートのすべてのレジスタ306のデータを全部

10

20

30

40

50

収集する。第2の代案実施例においては、CPU 230がグローバルな統計情報の要求を出すだけで、すべてのQCデバイス202のすべてのレジスタ306の情報が収集される。しかし、CPU 230は一回につきポート104の1つだけの情報を必要とする点に留意する。

【0126】

CPU 230が、E P S M 210に対するただ1回の要求でポート104のいずれに関する統計情報をも、すべて取り出すことができることは高い評価に値する。特に、要求を出す場合はQC STATISTICS INFORMATIONレジスタが1つのコマンドでCPU 230によって書き込まれる。その後CPU 230は、QCデバイス202からの応答の待機に拘束されることなく、自由に他のタスクを実行に移ることができる。その代わりにE P S M 210がQC/CPUバス204を介して個々の統計読み取り要求を実行し、全部のデータを収集する。CPU 230に対する通知が割り込み信号、もしくはポーリング・メカニズムによって行われ、CPU 230は要求したすべての情報を取り出すことができる。その結果、CPU 230のプロセッサ時間の使用効率が向上する。

10

【0127】

図28は、CPUインタフェース700とMCB 404間のインタフェースの詳細ブロック図である。アドレス復号／要求生成ロジック714からのメモリ要求信号が、アドレス生成ロジック746およびFIFO状態／割り込み生成ロジック742に結合しているFIFOアクセス状態マシン740に供給される。RX BUF 706およびTX BUF 708を含むFIFOブロック748が、アドレス生成ロジック746とFIFO状態／割り込み生成ロジック742に結合している。アドレス生成ロジック746およびFIFO状態／割り込み生成ロジック742は、バス703を介してCPUインタフェース700と、およびMCBバス428を介してMCB 404とそれぞれインタフェースするために、両方とも1組のデータ、アドレス、制御信号ドライバ／レシーバ744に結合している。

20

【0128】

図29は、CPUインタフェース700とHCB 402間のインタフェースの詳細ブロック図である。アドレス復号／要求生成ロジック714からのパケット読み出し要求信号が、TX BUF 712を含む送信バッファ762に結合されている送信パケット状態マシン760に供給される。アドレス復号／要求生成ロジック714からのパケット書き込み要求信号が、RX BUF 710を含む受信バッファ770に結合されている受信パケット状態マシン768に供給される。送信バッファ762および受信バッファ770は、バス705を介してCPUインタフェース700と、およびHCBバス438を介してHCB 402とそれぞれインタフェースするために、両方とも1組のデータ、アドレス、制御信号ドライバ／レシーバ764に結合している。

30

【0129】

次に図30を参照する。TPI 220の簡略ブロック図であり、これの全体を示している。TPI 220は、HSB 206とPCIバス222との間に介在してデータ転送を行い、TLAN 226とE P S M 210との間でネットワーク・データの受け渡しを行う。TPI 220はHSB 206上でスレーブとして動作し、E P S M 210のポーリングに応答し、そしてQCデバイス202と同じようにE P S M 210とデータの受け渡しを行う。PCIバス222側では、TPI 220がPCIバス222を介して4つのTLAN 226 (PORT 24、PORT 25、PORT 26、およびPORT 27)のそれぞれとネットワーク・データの受け渡しを行う。

40

【0130】

TPI 220は、HSBコントローラ804、PCIバス・コントローラ802、およびメモリ806を含む。PCIバス・コントローラ802は、PCIバスの標準に従ってPCIバス222とインタフェースし、TPI 220とPCIバス222との間のデータ転送を簡便化する。PCIバス標準は、Intel Architecture Labとその業界のパートナー各社によって定義されているものである。HSBコントローラ80

50

4は、HSB 206の定義済み動作に従ってHSB 206とインタフェースし、TPI 220とEPSM 210との間のデータ転送を簡便化する。メモリ806は1個所に集中あるいは分散配置することができ、複数のデータ・バッファ807および1つの制御リスト・メモリ808を含む。データ・バッファ807は、PCIバス222とHSB 206との間のデータ転送を簡便化するための一時的なメモリとして機能する。制御リスト・メモリ808はPCIバス222上における各TLAN 226のバス・マスタ動作を簡便化する。

【0131】

次に図31を参照する。TPI 220の詳細ブロック図である。TPI 220は、PCIバス222とのインタフェースに用いられるバッファ、ドライバ、および関連の回路を含んだPCIバス・インタフェース・ロジック810を含む。本実施例のPCIバス222は、データ幅が32ビットで33 MHzのクロック周波数で動作する。しかし、PCIバス222のデータ幅は特にこれでもよく、また動作クロックも、例えば66 MHzといった任意の、あるいは使用可能ないずれの周波数でも構わないことはもとより理解されている。TPI 220はPCIアービタ811を含み、これがPCIバス222へのアクセスとこれの制御についてTLAN 226、TPI 220、およびCPU 230のそれぞれの間で調停を行う。特に、TLAN 226、TPI 220、およびCPU 230は、それぞれいくつかの要求信号REQmの1つをアサートしてPCIバス222の制御を要求する。REQm信号はPCIアービタ811に受け取られる。PCIアービタ811は、応答してそれぞれの許可信号GNTmをアサートすることによって要求しているデバイスの1つに制御を許可する。PCIアービタ811は必要に応じて他のアービトレーション・スキームを適用することもできるが、図解した実施形態においては、PCIアービタ811による調停はラウンドロビン法に基づいている。1つのTLAN 226にPCIバス222の制御を許可した後で、PCIアービタ811はTLAN選択信号(TSELm)をアサートしてその特定のTLAN 226を識別する。

【0132】

TPI 220は、TPI 220とHSB 206とのインタフェースに用いるバッファ、ドライバ、および関連の回路を含んだHSBデータ転送インタフェース・ロジック819を含む。HSBデータ送信インタフェース・ロジック819は、HSB 206上における同時リード／ライト・サイクルのためにリード・ラッチ819aおよびライト・ラッチ819bを含む。HSBデータ転送インタフェース・ロジック819は、EPSM 210のポーリングに응答し、HSB 206上で実行されているサイクルを監視するために、ポート状態ロジック820を含む。特にポート状態ロジック820は、STROBE*信号がEPSM 210によってアサートされればそれを検知し、応答してPKT__AVAIL[6]*およびBUF__AVAIL[6]*信号をTPI 220のデータ状態に基づいて多重化の方式でアサートする。ポート状態ロジック820は、READ__OUT__PKT[6]*およびWRITE__IN__PKT[6]*信号をそれぞれアサートし、TPI 220に意図されたHSB 206上でのリードおよびライト・サイクルも検知する。TPI 220からEPSM 210へのHSBバス206を介したパケット・データの転送中、ポート状態ロジック820は転送されているデータがパケットの始め、またはパケットの終わりであれば、それぞれSOP*またはEOP*信号をHSB 206のバス・サイクルの期間アサートする。EPSM 210からTPI 220へのHSBバス206を介したパケット・データの転送中、ポート状態ロジック820はSOP*またはEOP*信号を読み取って、受信されているデータがパケットの始めであるか、あるいはパケットの終わりであるかを判断する。

【0133】

データ・バッファ807はいくつかの双方向FIFOデータ・バッファ、807a、807b、807c、および807d(807a-d)を含み、それぞれは32ビット幅の送信バッファ(TPI TX FIFO)および32ビット幅の受信バッファ(TPI RX FIFO)を含む。示した実施形態において、データ・バッファ807a、807

10

20

30

40

50

b、807c、および807dは、それぞれポートPORT24、PORT25、PORT26、およびPORT27に対応する。各TPIRX FIFOは、PCIバス222を介してそれぞれのTLAN 226からデータを受け取り、そのデータはTPI220によりHSB206を介してEPSM210に送られる。各TPI TX FIFOは、HSB206を介してEPSM210からデータを受け取り、そのデータはTPI220によりPCIバス222を介してそれぞれのTLAN 226へ送られる。

【0134】

受信リスト復号ロジック812はPCIバス・インタフェース・ロジック810に結合されており、少なくとも1つの受信制御リストを制御リスト・メモリ808の一部である受信制御リスト・メモリ(RXCNTLLIST)808aに格納する。受信リスト復号ロジック812は、PCIバス222上のアドレスとしてアサートされたRECEIVE LIST MEMORY BASE ADDRESSに応答し、PCIバス222へのデータとしてRXCNTLLIST808aからの受信制御リストの書き込みを行う。示した実施形態においては、RXCNTLLIST808aは一時に1つの受信制御リストを保持する。特に、それぞれのTLAN 226はPCIバス222の制御権を得てRECEIVE LIST MEMORY BASE ADDRESSをPCIバス222上でアサートし、対応する受信制御リストをRXCNTLLIST808aから受け取る。受信制御リストは、TLAN 226が使用するPACKET DATA MEMORY BASE ADDRESSを含み、これは受信データを格納する場所を示すアドレスである。それぞれのポートからのデータ・パケットの受信に応答し、TLAN 226は再びPCIバス222の制御権を得て、受信データ・パケットからのデータを、予め取り出してある受信制御リスト内に格納されているアドレスを用いてTPI220へ転送する。本明細書で後に詳述するように、TLAN 226は調停を行ってPCIバス222の制御を許可され、PACKET DATA MEMORY BASE ADDRESSをPCIバス222上でライト・サイクル中にアサートする。

【0135】

受信データ復号ロジック813、PCI RX FIFO制御ロジック817、PCIアービタ811、およびFIFO同期ロジック818が、PCIバス・インタフェース・ロジック810から対応するTPIRX FIFOへの受信データの流れを制御する。PCI RX FIFO制御ロジック817は、PCIバス・インタフェース・ロジック810からのデータを受け取る入力、およびそれぞれが対応するTPIRX FIFOに結合されているいくつかの選択可能な出力を含む。PCIアービタ811はTSELM信号をFIFO同期ロジック818に供給し、これがPCI RX FIFO制御ロジック817への対応するPCIバッファ選択信号(PBSELM)をアサートし、PCIバス222へのアクセスが許可されている特定のTLAN 226に基づいて適当なTPIRX FIFOを選択する。受信データ復号ロジック813は、PCIバス222上でライト・サイクルを実行中のTLAN 226によってアサートされたPACKET DATA MEMORY BASE ADDRESSを受け取って復号化し、応答してPCI RX FIFO制御ロジック817への受信イネーブル信号(REN)をアサートして選択したTPIRX FIFOにデータを渡す。

【0136】

PCIバス222とHSB 206との間における双方向データ・フローは、データ・バッファ807を介して実現されることに留意する。1つの実施形態において、PCIバス222とHSB 206は33 MHzといった等しい速度で動作するが、代案の実施形態では異なったクロック周波数で動作することも考えられる。例えば別の実施形態において、HSB 206が33 MHzで動作し、一方PCIバス222で66 MHz動作する。クロックの速度が異なっても、TPI220の機能が遂行されてデータ・フローを処理して同期が実現される。データ・バッファ807a-dのそれぞれのTPIRX FIFOおよびTPI TX FIFOは、データの書き込みと読み出しのためにポインタを両端に保持したサーキュラ・バッファとしての機能の遂行が望ましい。FIFO同期

10

20

30

40

50

ロジック 818 は、一般に各 FIFO の両端のポインタの同期、保持、および更新のために動作し、適当な TPI FIFO との正しいデータの読み書きを保証する。

【0137】

前述したように、各 TPI RX FIFO はサーキュラ・バッファとして機能を遂行する。PCI RX FIFO 制御ロジック 817 はいくつかの PCI 受信ポインタ (PCI RX PTR) を含み、選択された TPI RX FIFO 内の 1DWORD (32 ビット) のデータを受け取る次のロケーションを指示あるいはアドレスするために、それぞれの TPI RX FIFO に 1 つのポインタが充てられている。同様に、HSB RX FIFO 制御ロジック 821 が各 TPI RX FIFO の他端にあり、いくつかの PCI 受信「シンクロナイズド」ポインタ (PCI RX SPTR) を含み、これらのポインタは、それぞれが対応する 1 つの PCI RX PTR のシンクロナイズされたコピーである。適当な TPI RX FIFO を選択するための PBSELM 信号とともに、FIFO 同期ロジック 818 も複数の PCI カウント信号 (PCNTM) の対応する 1 つをアサートし、PCI RX FIFO 制御ロジック 817 内の当該の PCI RX PTR の同期的な更新すなわち増分を行う。FIFO 同期ロジック 818 は、さらに複数の HSB カウント信号 (HCNTM) の対応する 1 つをアサートし、HSB RX FIFO 制御ロジック 821 内の当該の PCI RX SPTR の同期的な更新すなわち増分を行う。このように、それぞれの TPI RX FIFO の両端に 1 つずつ用意されたポインタによって、データを挿入すべき場所が指示される。

【0138】

PCI TX FIFO 制御ロジック 816 は、TPI TX FIFO のいずれかの中でデータを検出し、送信すべきデータを持っている TPI TX FIFO に対応する TLAN226 に対してコマンドを送るため、TPI220 に PCI バス 222 の制御を要求させ、その制御を得させる。PCI TX FIFO 制御ロジック 816 は、1 組の TPI 制御レジスタ 846 から当該の TLAN226 のアドレスにアクセスする。TPI220 は当該の TLAN226 にコマンドを書き込み、TRANSMIT LIST MEMORY BASE ADDRESS を用意し、TLAN226 にその TRANSMIT LIST MEMORY BASE ADDRESS を使用する TPI220 から送信制御リストを続いて要求させる。

【0139】

送信リスト・デコード・ロジック 814 は、PCI バス・インタフェース・ロジック 810 に結合されており、少なくとも 1 つの送信コントロール・リストをコントロール・リスト・メモリ 808 の一部である送信コントロール・リスト・メモリ (TX CNTL LIST) 808b に格納する。送信リスト・コントロール・ロジック 814 は、PCI バス 222 上のアドレスとしてアサートされた送信リスト・メモリ・ベース・アドレス (TRANSMIT LIST MEMORY BASE ADDRESS) に応答し、PCI バス 222 へのデータとして TX CNTL LIST 808b からの送信コントロール・リストの書き込みを行う。示した実施例においては、TX CNTL LIST 808b は、一時に 1 つの送信コントロール・リストを保持する。このようにして、それぞれの TLAN226 は PCI バス 222 の制御権を得て、TRANSMIT LIST MEMORY BASE ADDRESS を PCI バス 222 上でアサートし、対応する送信コントロール・リストを TX CNTL LIST 808b から受け取る。送信コントロール・リストを取り出した後、TLAN226 は PCI バス 222 を要求し、そのバスの制御権を得ることによってその送信コントロール・リストを実行し、リード・サイクルを 1 回実行して、TPI220 の対応する TPI TX FIFO からパケット・データ・メモリ・ベース・アドレス (PACKET DATA MEMORY BASE ADDRESS) を用いてデータを取り出す。

【0140】

送信データ・デコード・ロジック 815、PCI TX FIFO コントロール・ロジック 816、PCI アービタ 811、および FIFO 同期ロジック 818 が、データ・バッ

10

20

30

40

50

ファ807の各TPI TX FIFOからPCIバス222へのデータの流れを制御する。PCI TX FIFOコントロール・ロジック816は、PCIバス・インタフェース・ロジック810へデータを供給する出力、およびそれぞれがTPI TX FIFOの対応する1つに結合されているいくつかの選択可能な入力を含む。TLAN226がデータを読み取るべくPCIバス222上でリード・サイクルを実行するとき、PCIアービタ811はTSELM信号をFIFO同期ロジック818に供給し、これがPCI TX FIFOコントロール・ロジック816へのPBSELM信号をアサートし、PCIバス222の制御権を持っている特定のTLAN226に基づいて、対応するTPI TX FIFOを選択する。送信データ・デコード・ロジック815は、TLAN226によってアサートされたPACKET DATA MEMORY BASE ADDRESSを受け取って復号化し、それに応答して、PCI TX FIFOコントロール・ロジック816への送信イネーブル信号(TEN)をアサートすることによって、選択されたTPI TX FIFOへのデータの転送を可能とする。PBSELM信号がPCI RX FIFOコントロール・ロジック817とPCI TX FIFOコントロール・ロジック816の両方に供給されること、そしてTENおよびREN信号によるPCI RX FIFOコントロール・ロジック817とPCI TX FIFOコントロール・ロジック816との間の選択が、サイクルのタイプ、およびデータ・フローの方向に依存していることに留意する必要がある。

【0141】

示された本実施例において、各TPI TX FIFOはサーキュラ・バッファとして機能20を遂行する。PCI TX FIFOコントロール・ロジック816はいくつかのPCI送信ポインタ(PCI TX PTR)を含み、1つのデワード(DWORD)のデータを読み出すべき次のロケーションを指示あるいはアドレス指定するために、それぞれのTPI TX FIFOに1つのポインタが充てられている。同様に、TPI TX FIFOの他端にある、本明細書で後に詳述するHSB TX FIFOコントロール・ロジック822は、いくつかのPCI送信「同期(シンクロナイズド)」ポインタ(PCI TX SPTR)を含み、これらのポインタは、それぞれが対応する1つのPCI TX PTRの同期されたコピーである。FIFO同期ロジック818は、PCI TX FIFOコントロール・ロジック816から1つのDWORDのデータがPCIバス222に供給される度に、対応する1つのPCNTM信号をアサートして当該のPCI TX 30PTRを増分し、対応する1つのHCNTM信号をアサートして当該のPCI TX SPTRを増分する。このように、それぞれのTPI TX FIFOの両端に1つずつ用意されたポインタによって、データを読み出すべき場所が指示される。

【0142】

HSB RX FIFOコントロール・ロジック821は、それぞれがTPI RX FIFOの対応する1つの出力に結合された幾つかの選択可能な入力を持っている。HSB RX FIFOコントロール・ロジック821は、HSB206上でアサートされるべきデータをHSBデータ転送インタフェース・ロジック819に供給するための1つの出力を持っている。HSB TX FIFOコントロール・ロジック822は、それぞれがTPI TX FIFOの対応する1つの入力に結合された幾つかの選択可能な出力を持っ40ている。HSB TX FIFOコントロール・ロジック822は、HSBデータ転送インタフェース・ロジック819からHSB206を介してデータを受け取るための1つの入力を持っている。

【0143】

HSB RX FIFOコントロール・ロジック821、ポート状態ロジック820、およびFIFO同期ロジック818は、TPI220からEPSM210へのデータ転送中、データ・バッファ807a~807dのTPI RX FIFOとHSB206との間におけるデータの流れを制御する。ポート状態ロジック820は、HSB206上におけるリード・サイクルを示READ__OUT__PKT[6]*信号がアサートとされたときにそれを検知し、選択されているポートの対応するTPI RX FIFOを識別すべく50

PORT_NO [1:0] 信号をデコードする。特に、EP SM 210は、PORT_NO [1:0] 信号00、01、10、または11をアサートして、ポートPORT 24、PORT 25、PORT 26、またはPORT 27にそれぞれ対応するデータ・バッファ807a、807b、807c、または807dの1つのTPI_RX_FIFOを選択する。ポート状態ロジック820は、FIFO同期ロジック818へのポート選択信号(PSELm)をアサートして選択されたポートを表示し、FIFO同期ロジック818が応答して対応するHSB選択信号(HBSELm)をアサートし、対応するTPI_RX_FIFOに結合されているHSB_RX_FIFO制御ロジック821の1つの出力を選択する。また、ポート状態ロジック820がHSBイネーブル信号(HREN)をアサートすることにより、HSB_RX_FIFO制御ロジック821は、HSB206上

10

【0144】

HSB_RX_FIFOコントロール・ロジック821は、TPI_RX_FIFO内における特定のデータのロケーションを示すためのHSB受信ポインタ(HSB_RX_PTR)を、それぞれのTPI_RX_FIFOについて1つずつ含む。FIFO同期ロジック818は、HCNTm信号の対応する1つをアサートして、TPI_RX_FIFOからDWORDが1つ読み出される度に、選択されているTPI_RX_FIFOの対応するHSB_RX_PTRを更新すなわち減分する。また、PCI_RX_FIFOコントロール・ロジック817は、対応するHSB受信「同期」ポインタ(HSB_RX_SPTR)を含み、これはFIFO同期ロジック818がPCNTm信号の対応する1つをアサートすることによって減分される。このように、HSB_RX_FIFOコントロール・ロジック821は、TPI_RX_FIFOのそれぞれについて2つのポインタを含み、PCI_RX_SPTRはデータを書き込むべき場所を指示し、HSB_RX_PTRはデータを読み出すべき場所を指示する。ポート状態ロジック820もこれらのポインタにアクセスし、各TPI_RX_FIFO内の有効なデータの量あるいは有効なデータ・バイト数を引き出す。このカウントは(TBUSの値に対応している)対応するRBSIZEと比較され、HSB206が、STROBE*信号に応答して、PKT_AVAL [6]*信号をアサートする方法を決定する。

20

【0145】

HSB_TX_FIFOコントロール・ロジック822、ポート状態ロジック820、およびFIFO同期ロジックは、EP SM 210からTPI 220へのデータ転送中、TPI_TX_FIFOとHSB206との間におけるデータの流れを制御する。ポート状態ロジック820はWRITE_IN_PKT [6]*信号がアサートとされたときにそれを検知し、EP SM 210がHSB206上で実行しているライト・サイクルの間に、PORT_NO [1:0] 信号からポート番号を検出する。ポート状態ロジック820はそれに応答して、PSELm信号およびHSB送信イネーブル信号(HTEN)をアサートし、当該するTPI_TX_FIFOを示す。FIFO同期ロジック818はそれに応答して、HBSELm信号をアサートし、当該TPI_TX_FIFOに対してHSB_TX_FIFOコントロール・ロジック822の対応する入力を選択する。HTEN信号によってHSB_TX_FIFOコントロール・ロジック822がイネーブルされ、HSBデータ転送インタフェース・ロジック819から選択されたTPI_TX_FIFOにアサートすべきデータを受け取る。

30

40

【0146】

HSB_TX_FIFOコントロール・ロジック822は、それぞれのTPI_TX_FIFOについて1つのHSB送信ポインタ(HSB_TX_PTR)を含み、これによって、データを書き込むべきTPI_TX_FIFO内の特定のロケーションが指示される。FIFO同期ロジック818はHCNTm信号の対応する1つをアサートし、選択されたTPI_TX_FIFOに1つのDWORDが書き込まれる度に、その選択されたTPI_TX_FIFOの対応するHSB_TX_PTRを更新すなわち増分する。また、PC

50

I TX FIFOコントロール・ロジック816は、対応するHSB送信「同期」ポインタ(HSB TX SPTR)を含み、これは、FIFO同期ロジック818がPCNT m信号の対応する1つをアサートすることによって増分される。このように、HSB TX FIFOコントロール・ロジック822はTP I TX FIFOのそれぞれについて2つのカウンタを含み、PC I TX SPTRはデータを読み出すべき場所を指示し、HSB TX PTRはデータを書き込むべき場所を指示する。ポート状態ロジック820もこれらのポインタにアクセスし、各TP I TX FIFO内の使用可能なスペース量あるいは空のデータ・バイト数を取り出す。このカウンタは(TBUSの値に対応している)対応するXBSIZEと比較され、HSB 206がSTROBE*信号に応答して、BUF__AVAIL[6]*信号をアサートする方法を決定する。

10

【0147】

TP I 220内には1組のTP I PC Iコンフィギュレーション・レジスタ835が用意されており、PC Iバス222を介したアクセスのために、PC Iバス・インタフェース・ロジック810に結合されている。また、TP Iコントロール・レジスタ846が用意されており、TP I 220内の各種のデバイス、およびPC Iバス222を介したアクセスのために、PC Iバス・インタフェース・ロジック810に結合されている。これらのレジスタ846および835の内容や構造は、後に詳述する。HSBデータ転送インタフェース・ロジック819は、PACKET SIZEタグ・レジスタ819cも含む。HSBデータ転送インタフェース・ロジック819は、E PSM 210から送られる各パケット・データの最初のDWORDを捉え、パケット・サイズ(PACKET SIZE 20)タグ・レジスタ819cに格納し、該レジスタ819cの内容を送信リストデコード・ロジック814のTX CNTL LISTに書き込む。

【0148】

次に図32を参照する。各TLAN 226の構成と機能を示すブロック図である。TLAN 226は、イーサネット(Ethernet)・ポート110、PC Iバス・インタフェース824、およびイーサネット・ポート110とPC Iバス・インタフェース824との間に結合されたメモリ825を含む。

【0149】

イーサネット・ポート110は、対応するネットワーク112との間におけるパケット・データの送受信のために、100Mbのイーサネット・セグメント114の適合するコネクタを受容するための適宜のソケットを含む。イーサネット・ポート110は、受信したパケット・データをメモリ825内のデータ・バッファ826に供給する。イーサネット・ポート110はデータ・バッファ826からデータを取り出し、そのパケット・データをイーサネット・セグメント114に送信する。

30

【0150】

TLAN 226は、その動作を制御するための1組のレジスタ828をメモリ825内に含む。レジスタ828は、外部のデバイスがPC Iバス222を介してコマンドを挿入できるようにするために、コマンド・レジスタ828aを含む。レジスタ828は、外部のメモリからPC Iバス222を介してコマンド・リストをアクセスするためのアドレスを格納する、チャンネル・パラメータ・レジスタ828bをさらに含む。コマンド・レジスタ828aは、TLAN 226に対し、コマンド・リストを取り出して実行するように指示するための(示していないが)GOビットを含む。コマンド・レジスタ828aは、TLAN 226に対し、(RXの場合)受信コマンド・リストを、そして(TXの場合)送信コマンド・リストを取り出して実行するように指示するための(示していないが)RX/TXビットも含む。メモリ825は現在のコントロール・リストを格納するためのリスト・バッファ827を含み、さらにリスト・バッファ827は、現在の受信・コントロール・リストを格納するための受信コントロール・リスト・バッファ827a、およびカレントの送信コントロール・リストを格納するための送信コントロール・リスト・バッファ827bを含む。

40

【0151】

50

PCIバス・インタフェース824は、PCIバス222に結合し、データ転送中にPCIバス222のバス・マスタを動作させることによって、TPI220とTLAN226との間のデータ転送を制御するためのロジックを含む。TPI220やCPU230のような外部のデバイスは、チャンネル・パラメータ・レジスタ828bにアドレスを書き込み、コマンド・レジスタ828aにコマンドを書き込む。TLAN226はそれに応答してREQm信号をアサートし、PCIバス222を仲裁に委ねる。GNTm信号を受け取ると、TLAN226は指示されたコマンド・リストを受け取ってリスト・バッファ827に格納するため、PCIバス222上で1サイクルを実行する。コマンドは、RX/TXビットがTXにセットされていれば送信コマンドとみなされ、RX/TXビットがRXにセットされていれば受信コマンドとみなされる。

10

【0152】

受信動作を開始するために、CPU230は、受信リスト・メモリ・ベース・アドレス (RECEIVE LIST MEMORY BASE ADDRESS) をチャンネル・パラメータ・レジスタ828bに書き込み、受信コマンドを各TLAN226のコマンド・レジスタ828aに書き込む。TLAN226は、応答してRECEIVE LIST MEMORY BASE ADDRESSを用いて受信コントロール・リストを取り出すべく、PCIバス222を要求する。TPI220は受信コントロール・リストをTLAN226に供給し、そしてTLAN226は、データの受信を待ってから受信コントロール・リストを実行する。受信コントロール・リストは順方向ポインタを含み、TLAN226はそれを用いて次の受信コントロール・リストを取り出し、コントロール・リストのチェーン（連鎖）を形成する。しかし、望ましい実施例では、TPI220が各受信コントロール・リストの順方向ポインタを同一のRECEIVE LIST MEMORY BASE ADDRESSとともにロードする。ポート110からのデータがTPI220に受信される場合、PCIバス・インタフェース824は仲裁に委ねて、PCIバス222の制御権を得てから、その受信コントロール・リスト・バッファ827a内の受信コントロール・リストを実行して、データをPCIバス222を介してTPI220に転送する。データ・パケット全体の転送が完了したとき、TLAN226は、現在の受信コントロール・リストの順方向ポインタ内のRECEIVE LIST MEMORY BASE ADDRESSを使用し、新しく別の受信コントロール・リストを要求する。

20

【0153】

送信動作について説明する。TPI220がそのTPI TX FIFOのいずれかから送信すべきデータを検知し、仲裁に委ねてPCIバス222の制御権を獲得する。それからTPI RX FIFOは、送信リスト・メモリ・ベース・アドレス (TRANSMIT LIST MEMORY BASE ADDRESS) をそれぞれのTLAN226のチャンネル・パラメータ・レジスタ828bに、送信コマンドをコマンド・レジスタ828aに書き込む。TLAN226は、TRANSMIT LIST MEMORY BASE ADDRESSを用いて送信コントロール・リストを取り出すべく、PCIバス222を要求する。送信コントロール・リストが受け取られると、TLAN226はその送信コントロール・リストを送信コントロール・リスト・バッファ827bに格納し、そして、格納されている送信コントロール・リストを実行してパケット・データを受け取る。送信コントロール・リストも順方向ポインタを含み、通常はこれをTLAN226が次のアドレスとして用いることによって次の送信コントロール・リストを受け取り、コントロール・リストのチェーンを形成する。ただし、示した実施例では、TPI220は各送信コントロール・リストの順方向ポインタをヌル値とともにロードする。従って、その送信コントロール・リスト・バッファ827b内の送信コントロール・リストの実行後は、TLAN226はTPI220が新しく別の送信コマンドを書き込むまで、待機することになる。

30

40

【0154】

次に図33を参照する。該図はコントロール・リスト830を示している。これは受信と送信の両方のコントロール・リストの形式であり、さらにRX CNTL LIST 80

50

8 aおよびTX CNTL LIST 8 0 8 bの形式でもある。コントロール・リスト 8 3 0は、FORWARD_POINTERフィールド 8 3 1、PACKET_SIZEフィールド 8 3 2 a、CSTATフィールド 8 3 2 b、COUNTフィールド 8 3 3、およびDATA_POINTERフィールド 8 3 4を含む。各フィールドは32ビットであるが、PACKET_SIZEフィールド 8 3 2 aとCSTATフィールド 8 3 2 bは、16ビットのフィールドである。

【0155】

FORWARD_POINTERフィールドは、一般に複数のコントロール・リストをチェーン化するために使用される。受信動作については、FORWARD_POINTERフィールド 8 3 1がそれぞれのケースで同じRECEIVE_LIST_MEMORY_BASE_ADDRESSであるので、TPI 2 2 0が、RX CNTL LIST 8 0 8 aから何度も繰り返して供給する受信コントロール・リストをTLAN 2 2 6が実行する。このように、各TLAN 2 2 6は、そのカレントの受信コントロール・リストのFORWARD_POINTERフィールド 8 3 1内のRECEIVE_LIST_MEMORY_BASEADDRESSを使用して、ネットワーク 1 1 2から次のデータ・パケットが受信されたとき、次の受信コントロール・リストを要求する。従って、受信動作については、TLAN 2 2 6に対してTPI 2 2 0が動作開始コマンドを出す必要がない。送信動作については、TPI 2 2 0が毎回同一のTX CNTL LIST 8 0 8 bからの送信コントロール・リストを実行する。しかし、TPI 2 2 0はFORWARD_POINTERフィールド 8 3 1をヌル値(0000h)にセットし、従ってTPI 2 2 0によって開始されたときは、TPI 2 2 0およびそれぞれのTLAN 2 2 6は1つの送信動作を実行する。いずれかのTPIRX_FIFOの中でデータが検知されて、TPI 2 2 0がTPIRX_FIFOのそれぞれのTLANポート上で送信動作を行っていないとき、TPI 2 2 0は送信コマンドをそれぞれのTLAN 2 2 6に対して発生し、送信動作が開始される。それぞれのTLAN 2 2 6はTX CNTL LIST 8 0 8 bから送信コントロール・リストを取り出し、その送信コントロール・リストを実行し、そしてFORWARD_POINTERフィールド 8 3 1のヌル値を検知したとき、デフォルトの状態に戻る。

【0156】

PACKET_SIZEフィールド 8 3 2 aは、通常データ・パケットのサイズを表示する。受信動作については、TPI 2 2 0が最初にRX CNTL LIST 8 0 8 a内のPACKET_SIZEフィールド 8 3 2 aをゼロにセットする。TLAN 2 2 6がTPI 2 2 0に対して1つのデータ・パケット全体の送信を完了した後、TLAN 2 2 6は、RX CNTL LIST 8 0 8 aのPACKET_SIZEフィールド 8 3 2 aおよびCSTATフィールド 8 3 2 bに対して最後のDWORDの書き込みを実行する。PACKET_SIZEフィールド 8 3 2 aは実際のパケット・データのサイズでロードされ、CSTATフィールド 8 3 2 b内のフレーム完了ビットがセットされる。送信動作については、TX CNTL LIST 8 0 8 bのPACKET_SIZEフィールド 8 3 2 aは、TPI 2 2 0によってTLAN 2 2 6に送信されるべきデータ・パケットのサイズでロードされる。HSBデータ転送インタフェース・ロジック 8 1 9は、TX CNTL LIST 8 0 8 bのPACKET_SIZEレジスタ・タグ 8 1 9 c内のパケット・サイズDWORDを送信リスト・デコード・ロジック 8 1 4内のTX CNTL LIST 8 0 8 bに書き込む。そして、TPI 2 2 0が前述したように送信コマンドを対応するTLAN 2 2 6に書き込み、TX CNTL LIST 8 0 8 bの内容が送信コントロール・リストとしてTLAN 2 2 6に対して要求されたときに供給される。

【0157】

CSTATフィールド 8 3 2 bは、TPI 2 2 0とTLAN 2 2 6との間におけるコマンドおよび状態情報の受け渡しに使用される。TPI 2 2 0はRX CNTL LIST 8 0 8 aのCSTATフィールド 8 3 2 bを最初にゼロにセットする。TLAN 2 2 6からそれぞれのTPIRX_FIFOへのパケット・データの転送が完了したとき、TPI

10

20

30

40

50

220はRX CNTL LIST808a内のCSTATフィールド832bのフレーム完了ビット（ビット14）をセットすることによって、パケット・データ転送の完了を表示する。TPI220は、データ・パケットをHSB206を介してEPSM210へ転送を開始できる状態にあることをポート状態ロジック820に知らせる。そしてポート状態ロジック820は、それぞれのTPI RX FIFO内にEPSM210によるポーリングに応答して、EPSM210に対して送信可能なデータがあることを表示する。パケットの終わりは必ず転送しなければならないため、たとえパケットの終わりのデータ量がRBSIZEもしくはTBUSの値に適合しない場合でも、同様である。

【0158】

TPI220は、EPSM 10からのデータ・パケットの受信中におけるAL__FCS 10
__IN*（またはFBPN*）信号の状態に基づいて、TX CNTL LIST808
bのCSTATフィールド832b内のパス巡回冗長検査CRC（Cyclic Red
undancy Check）ビットをセットする。TPI220は、データ・パケット
がCRCに使用されるデータを含んでいるかどうかを示すCRCビットをセットする。C
RCを含むイーサネットのデータ・パケットには、パケット・データに加えて誤り検査に
用いられる4バイトのCRCデータが入っている。

【0159】

DATA__POINTERフィールド834は、データ転送動作中にTLAN226によ
ってアサートされるべきPCIアドレスを指定する。このアドレスは、パケット・デー
タ・メモリ・ベース・アドレス（PACKET DATA MEMORY BASE AD 20
DRESS）であって、送信および受信動作の両方に同じものであることが望ましい。デ
ータ受信動作中、TLAN226がPACKET DATA MEMORY BASE
ADDRESSをアサートし、受信データ復号ロジック813がPCIバス222上のア
ドレスおよびライト・サイクルをデコードし、そして、選択されているTPI RX F
IFO内へパケット・データが受容されるように、PCI RX FIFOコントロール
・ロジック817をイネーブルする。データ送信動作中、TLAN226がPACKET
DATA MEMORY BASE ADDRESSをアサートし、送信データ復号ロ
ジック815がアドレスおよび読み出し動作をデコードし、そしてTPI TX FIF
Oからのパケット・データの送信を促進するように、PCI TX FIFO制御ロジッ
ク816をイネーブルする。 30

【0160】

COUNTフィールド833は、存在するデータの量あるいはDATA__POINTER
フィールド834の現在値における使用可能なバッファ・スペースを示す。データ受信動
作中、受信リスト・デコード・ロジック812は、COUNTフィールド833をTPI
コントロール・レジスタ846のRCV__DATA__COUNTレジスタ847b（第8
F図）内に書き込まれる値に設定する。RCV__DATA__COUNTレジスタ847b
の値でTPI220が受信すべき最大パケット・サイズが決まる。既定値は1, 518バ
イトであって、これはCRCの4バイトを含むイーサネット・データ・パケットの最大サ
イズである。データ送信動作中、TPI220はCOUNTフィールド833をPACK
ET__SIZEフィールド832aと同じ値に設定する。 40

【0161】

次に図34を参照する。該図は、TPI220に使用されるTPI PCIコンフィギュ
レーション・レジスタ835の定義を示している。TPI PCIコンフィギュレー
ション・レジスタ835は、TPI220専用の追加的なレジスタ、およびすべてのPCIバ
スのアーキテクチャに共通のレジスタを含む。すべてのPCIバスに共通のレジスタは、
DEVICE__IDレジスタ836a、VENDOR__IDレジスタ836b、状態（S
TATUS）レジスタ837a、コマンド（COMMAND）レジスタ837b、CLAS
SS__CODEレジスタ838a、REV__IDレジスタ838b、BISTレジスタ8
39a、HDR__TYPEレジスタ839b、レイテンシすなわち待ち時間（LATEN
CY）レジスタ839c、CACHELSレジスタ839d、MAXLATレジスタ84 50

5 a、MINGNTレジスタ845b、INTPINレジスタ845c、およびINTLINEレジスタ845dである。

【0162】

TPI220専用のレジスタは、TPIコントロールIOベース・アドレス(CONTROL IO BASE ADDRESS)レジスタ840、TPIコントロール・メモリ・ベース・アドレス(CONTROL MEMORY BASE ADDRESS)レジスタ841、送信リスト・メモリ・ベース・アドレス(TRANSMIT LIST MEMORY BASE ADDRESS)レジスタ842、受信リスト・メモリ・ベース・アドレス(RECEIVE LIST MEMORY BASE ADDRESS)レジスタ843、およびパケット・データ・メモリ・ベース・アドレス(PACKET DATA MEMORY BASE ADDRESS)レジスタ844である。

10

【0163】

初期化後、TPIコントロールIOベース・アドレス・レジスタ840にはTPIコントロール・レジスタ846のためのTPI CONTROL IO BASE ADDRESSが入っている。TPIコントロール・メモリ・ベース・アドレス・レジスタ841にはTPIコントロール・レジスタ846のためのTPI CONTROL MEMORY BASE ADDRESSが入っている。このように、TPIコントロール・レジスタ846は、PCIバス222の入出力とメモリ・スペースの両方でアクセスが可能である。送信リスト・メモリ・ベース・アドレス・レジスタ842には、送信リストデコード・ロジック814によってデコードされるTX CNTL LIST808bのためのTRANSMIT LIST MEMORY BASE ADDRESSが入っている。受信リスト・メモリ・ベース・アドレス・レジスタ843には、受信リストデコード・ロジック812によってデコードされるRX CNTL LIST808aのためのRECEIVE LIST MEMORY BASE ADDRESSが入っている。パケット・データ・メモリ・ベース・アドレス・レジスタ844には、TPI220のデータ・バッファ807に対応するPACKET DATA MEMORY BASE ADDRESSが入っている。PACKET DATA MEMORY BASE ADDRESSは、送信リスト・デコード・ロジック814と受信リスト・デコード・ロジック812の両方によってデコードされる。

20

【0164】

次に図35を参照する。該図は、TPI220に使用されるTPIコントロール・レジスタ846の定義の図解である。TPIコントロール・レジスタ846は、RCV__DATA__COUNTレジスタ847b、XBSIZE3レジスタ848a、XBSIZE2レジスタ848b、XBSIZE1レジスタ848c、XBSIZE0レジスタ848d、RBSIZE3レジスタ849a、RBSIZE2レジスタ849b、RBSIZE1レジスタ849c、RBSIZE0レジスタ849d、NET__PRI3レジスタ850a、NET__PRI2レジスタ850b、NET__PRI1レジスタ850c、NET__PRI0レジスタ850d、TLAN0メモリ・ベース・アドレス(MEMORY BASE ADDRESS)レジスタ851、TLAN1メモリ・ベース・アドレス・レジスタ852、TLAN2メモリ・ベース・アドレス・レジスタ853、およびTLAN3メモリ・ベース・アドレス・レジスタ854を含む。

30

40

【0165】

RCV__DATA__COUNTレジスタ847bは、TPI220が処理した受信データ・パケットの最大サイズを格納する。TPI220は、この値を取り出してRX CNTL LIST 08aのCOUNTフィールド833に入れる。XBSIZEレジスタ848a～dの各々は、それぞれのポートについてDWORD単位の送信バースト・サイズを保持している。すなわち、PORT24にはXBSIZE0、PORT25にはXBSIZE1、PORT26にはXBSIZE2、そしてPORT27にはXBSIZE3である。XBSIZEの送信バースト・サイズの値は、それぞれのポートに対してEPSM210からデータを要求できるだけの十分なパケット・バッファ・スペースがそれぞれの

50

T P I T X F I F Oにあるかどうかを判定するとき、T P I 2 2 0のH S B T X F I F Oコントロール・ロジック8 2 2およびポート状態ロジック8 2 0によって用いられる。R B S I Z Eレジスタ8 4 9 a～dの各々は、それぞれのポートについてD W O R D単位のH S B受信バースト・サイズを保持する。すなわち、P O R T 2 4にはR B S I Z E 0、P O R T 2 5にはR B S I Z E 1、P O R T 2 6にはR B S I Z E 2、そしてP O R T 2 7にはR B S I Z E 3である。R B S I Z Eの受信バースト・サイズの値は、それぞれのポートからE P S M 2 1 0に対する受信データ転送を要求できるだけの十分なパケット・データがそれぞれのT P I R X F I F Oにあるかどうかを判定するとき、H S B R X F I F Oコントロール・ロジック8 2 1およびポート状態ロジック8 2 0によって用いられる。図解した実施例において、X B S I Z EおよびR B S I Z Eレジスタ8 4 8、8 4 9の値はそれぞれが等しく、またT B U Sの値とも等しい。しかし、X B S I Z Eレジスタ8 4 8およびR B S I Z Eレジスタ8 4 9は、必要に応じて任意のバースト転送値でプログラミングされる。

10

【0166】

N E T _ P R I レジスタ8 5 0は、それぞれのポートに関するそれぞれのネットワーク優先権の値を保持する。すなわち、P O R T 2 4にはN E T _ P R I 0、P O R T 2 5にはN E T _ P R I 1、P O R T 2 6にはN E T _ P R I 2、そしてP O R T 2 7にはN E T _ P R I 3である。これらの値は、送信リスト・デコード・ロジック8 1 4がそれぞれのポートの送信優先権を設定するために使用される。T L A N 0メモリ・ベース・アドレス・レジスタ8 5 1は、P O R T 2 4についてT L A N 0 M E M O R Y B A S E A D D R E S SというP C Iメモリ・アドレスを保持する。T L A N 1メモリ・ベース・アドレス・レジスタ8 5 2は、P O R T 2 5についてT L A N 1 M E M O R Y B A S E A D D R E S SというP C Iメモリ・アドレスを保持する。T L A N 2メモリ・ベース・アドレス・レジスタ8 5 3は、P O R T 2 6についてT L A N 2 M E M O R Y B A S E A D D R E S SというP C Iメモリ・アドレスを保持する。T L A N 3メモリ・ベース・アドレス・レジスタ8 5 4は、P O R T 2 7についてT L A N 3 M E M O R Y B A S E A D D R E S SというP C Iメモリ・アドレスを保持する。これらのレジスタのそれぞれを、起動時にC P U 2 3 0が各T L A N 2 2 6のアドレスを認識してから初期化する。これらの値はP C I T X F I F Oコントロール・ロジック8 1 6に供給され、このロジックがP C Iバス2 2 2上にそれぞれの送信コマンドを出してパケット送信動作を開始するために該値を使用する。

20

30

【0167】

次に図36を参照する。該図は、ネットワーク・スイッチ102の初期化、起動あるいはリセット時におけるC P U 2 3 0のP C I初期化動作を図解したフローチャートである。最初のステップ855において、C P U 2 3 0はP C Iバス2 2 2のコンフィギュレーションを行い、それぞれのT L A N 2 2 6をP C Iメモリ・スペースにマッピングし、そして、このコンフィギュレーションをP C Iバス2 2 2を介してT P I P C Iコンフィギュレーション・レジスタ8 3 5に書き込む。P C Iバス2 2 2のコンフィギュレーションを行う手順は既知であり、ここではさらに説明しない。

【0168】

40

特に、D E V I C E _ I Dレジスタ8 3 6 aは、標準のP C IデバイスIDレジスタであり、その値は0 x 5 0 0 0 hに設定される。V E N D O R _ I Dレジスタ8 3 6 bは標準のP C IベンダIDレジスタであり、その値は0 x 0 E 1 1 hに設定される。S T A T U Sレジスタ8 3 7 aは標準のP C Iデバイス状態レジスタである。C O M M A N Dレジスタ8 3 7 bは標準のP C Iデバイス・コマンド・レジスタである。C L A S S _ C O D Eレジスタ8 3 8 aは標準のP C Iデバイス・クラス・コード・レジスタであり、その値は0 x 0 6 0 2 0 0 hに設定される。R E V _ I Dレジスタ8 3 8 bは標準のP C Iデバイス改定IDレジスタであり、その値は0 x 0 0 hに設定される。B I S Tレジスタ8 3 9 aは標準のP C I B I S T状態レジスタであり、その値は0 x 0 0 hに設定される。H D R _ T Y P Eレジスタ8 3 9 bは標準のP C Iヘッダ・タイプ・レジスタであり、その

50

値は0x80hに設定される。LATENCY（待ち時間）レジスタ839cは標準のPCI待ち時間レジスタであり、CPU230によって初期化される。CACHELSZレジスタ839dは標準のPCIキャッシュ・ライン・サイズ・レジスタであり、CPU230によって初期化される。MAXLATレジスタ845aは標準のPCI最長待ち時間レジスタであり、その値は0x00hに設定される。MINGNTレジスタ845bは標準のPCIデバイス・ミニマム・grant・レジスタであり、その値は0x00hに設定される。INTPINレジスタ845cは標準のPCIデバイス割り込みピン・レジスタであり、その値は0x00hに設定される。INTLINEレジスタ845dは標準のPCIデバイス割り込みライン・レジスタであり、CPU 230によって設定される。

【0169】

ステップ855では、さらにCPU230が0xFFFFFFFFhの値を次のそれぞれのレジスタに書き込む。すなわち、TPI CONTROL IO BASE ADDRESSレジスタ840； TPI CONTROL MEMORY BASE ADDRESSレジスタ841； TRANSMIT LIST MEMORY BASE ADDRESSレジスタ842； RECEIVE LIST MEMORY BASE ADDRESSレジスタ843； およびPACKET DATA MEMORY BASE ADDRESSレジスタ844に書き込む。それぞれへの書き込み完了後、TPI220が各レジスタ内の値を、指示された特定のレジスタに求められる量の入出力（I/O）またはメモリ・スペースを示す値に置き換える。CPU230は、それに応答して各レジスタ内のそれぞれの新しい値を読み取り、各レジスタにベース（基準）・アドレスを書き返し、そのエンティティをPCI I/Oまたはメモリ・スペースにマッピングする。

【0170】

特に、必要なメモリ・スペースの量を決定してから、CPU230はCONTROL IO BASE ADDRESSをTPI CONTROL IO BASE ADDRESSレジスタ840に書き込んで、TPIコントロール・レジスタ846の入出力スペースへのアクセスを可能とし、CPU230はTPI CONTROL MEMORY BASE ADDRESSをTPI CONTROL MEMORY BASE ADDRESSレジスタ841に書き込んでTPIコントロール・レジスタ846のメモリ・スペースへのアクセスを可能とし、CPU230はTRANSMIT LIST MEMORY BASE ADDRESSをTX CNTL LIST808bメモリ・ブロックのアドレスに対応するTRANSMIT LIST MEMORY BASE ADDRESSレジスタ842に書き込み、CPU230はRECEIVE LIST MEMORY BASE ADDRESSをRX CNTL LIST808aのアドレスに対応するRECEIVE LIST MEMORY BASE ADDRESSレジスタ843に書き込み、そしてCPU230はPACKET DATA MEMORY BASE ADDRESSをデータ・バッファ807のPCIアドレスに対応するPACKET DATA MEMORY BASE ADDRESSレジスタ844に書き込む。

【0171】

次のステップ856aにおいて、CPU230はPCIバス222上のそれぞれのTLAN226に対して1つずつ問い合わせを行い、存在するTLANの数、およびそれらのTLANの対応するPCIアドレスを認識する。続くステップ856bで、CPU230は問い合わせたTLAN226を既知で休止の状態に初期化する。そしてCPU230は、次のステップ857でTLAN226がそれ以上存在するかどうかを調べ、もし存在すればステップ856aに戻って、次のTLAN226に対して問い合わせを行い、PCIバス222上のTLAN226がすべて初期化されるまでこれを繰り返す。この時点では、TLAN0 MEMORY BASE ADDRESS、TLAN1 MEMORY BASE ADDRESS、TLAN2 MEMORY BASE ADDRESS、およびTLAN3 MEMORY BASE ADDRESSの値は既知である。

【0172】

次のステップ858において、CPU230は、図35に関して前述したように、TPI

10

20

30

40

50

コントロール・レジスタ 846 を適切な値に初期化する。これは、T L A N 0 M E M O R Y B A S E A D D R E S S、T L A N 1 M E M O R Y B A S E A D D R E S S、T L A N 2 M E M O R Y B A S E A D D R E S S、および T L A N 3 M E M O R Y B A S E A D D R E S S の値を含む。続くステップ 859 で、C P U 230 は、R E C E I V E L I S T M E M O R Y B A S E A D D R E S S をチャンネル・パラメータ・レジスタ 828b に書き込み、各 T L A N 226 の受信動作の始動を開始する。受信動作の開始はステップ 960 で完了し、C P U 230 が各 T L A N 226 のコマンド・レジスタ 828a に対して書き込みを行う。このように初期化されて、それぞれの T L A N 226 は受信コントロール・リストを要求するために、P C I バス 222 を要求して、直ちに受信動作を始める。

10

【0173】

次に図 37 を参照する。該図は、各 T L A N 226 についてネットワーク・スイッチ 102 が行う受信動作を図解するフローチャートである。動作は第 1 ステップ 861a で始まり、T L A N 226 は、P C I アービタ 811 に P C I バス 222 を要求してこれを受け取る。T L A N 226 は第 2 ステップ 861b で R E C E I V E L I S T M E M O R Y B A S E A D D R E S S を P C I バス 222 にアサートして受信コントロール・リストを要求し、T P I 220 が第 3 のステップ 861c で受信コントロール・リストをその T L A N 226 に供給する。受信コントロール・リストは、受信したデータ・パケットをどこで、もしくはどのように送信するかを T L A N 226 に知らせるための P A C K E T D A T A M E M O R Y B A S E A D D R E S S を含む。次の第 4 のステップ 861d で、T L A N 226 は P C I バス 222 の制御権を放棄する。

20

【0174】

T L A N 226 は、次のステップ 862a において、最終的にネットワーク 112 からデータ・パケットを受信し、ステップ 862b において、P C I バス 222 の制御権を要求してこれを受け取る。T L A N 226 はステップ 862c で、P A C K E T D A T A M E M O R Y B A S E A D D R E S S を P C I バス 222 上のアドレスとして用い、1 バーストのデータの書き込みを行い、一方 T P I 220 は、ステップ 862d において、そのデータを選択された T P I R X F I F O に書き込む。書き込みバーストの完了と同時に、T L A N 226 は次のステップ 862e に移って、P C I バス 222 の制御権を放棄する。さらに次のステップ 865 において、T L A N 226 は、最終 D W O R D の書き込み動作で示されるべき、パケット・データの全体の T P I R X F I F O に対する送出が完了したかどうかをチェックし、まだであれば、動作はステップ 862b へ戻り、T L A N 226 はもう一度 P C I バス 222 を要求するために別のバースト・データを送る。

30

【0175】

T L A N 226 は、データ・パケットの最終部分を送り終わった後、最後の反復動作を行って、T P I R X F I F O に対しパケットの終わりを知らせる。特に T L A N 226 は、T P I 220 の R X C N T L L I S T 808a 内の P A C K E T _ S I Z E フィールド 832a および C S T A T フィールド 832b に対して、最後の 1 D W O R D の転送を実行する。この D W O R D の転送によって、R X C N T L L I S T 808a が完了したばかりのデータ・パケットのサイズで更新され、C S T A T フィールド 832b 内のフレーム完了ビットが更新される。T P I 220 はこの書き込み動作をステップ 865 で検知して動作完了を表す内部フラグをセットし、ステップ 866 において、その適宜の状態をポート状態ロジック 820 に渡す。動作はステップ 861a へ戻って、別の受信コントロール・リストを要求する。

40

【0176】

次に図 38 を参照する。該図は、T P I 220 から E P S M 210 への H S B 206 を介した受信データ転送動作を図解するフローチャートである。動作は最初のステップ 876 で開始し、T P I 220 のポート状態ロジック 820 が、T P I R X F I F O のいずれか 1 つに存在する T P I コントロール・レジスタ 846 で用意されたそれぞれの R B S

50

I Z Eと比べて等しいか大きい一定量のデータを検知するか、もしくは、T L A N 2 2 6によって表示されているそのポートに関するパケットの終わりE O Pを検出する。

【0177】

次のステップ877では、T P I 2 2 0がE P S M 2 1 0のポーリングに応答し、各T P I R X F I F O内に十分なデータが存在するか否かを表すP K T _ A V A I L [6] *信号を多重化の方式で適切にアサートする。このポーリングは、独立して発生し、クラリフィケーションの目的で含まれている。T P I 2 2 0のいずれかのT P I R X F I F O内に十分なデータが存在することをP K T _ A V A I L [6] *信号が表示した場合、E P S M 2 1 0の使用可能な受信バッファ内に十分なバッファ記憶スペースがあれば、E P S M 2 1 0はH S B

10

206上でリード・サイクルを開始する。

【0178】

T P I 2 2 0のポート状態ロジック820は、H S B 2 0 6上のリード・サイクルを検出し、当該のT P I R X F I F Oを選択して次のステップ879でデータを供給する。それからT P I 2 2 0は、ステップ880において、E P S M 2 1 0に対しH S B 2 0 6を介してデータ・バーストを転送する。ステップ880でのデータ転送中、次のステップ881aで、ポート状態ロジック820がH S B 2 0 6を介した現在のデータ転送がパケットの始めであると判断すれば、データ転送中にT P I 2 2 0がステップ881bにおいてH S B 2 0 6上でS O P *信号をアサートする。同様に、ステップ880でのデータ転送中、ステップ882aにおいて、ポート状態ロジック820がH S B 2 0 6を介した現在のデータ転送がパケットの終わりであると判断すれば、データ転送中にT P I 2 2 0が、ステップ882bにおいて、H S B 2 0 6上でE O P *信号をアサートする。ステップ882aまたは882bから、動作はステップ876へ戻る。

20

【0179】

次に図39を参照する。該図は、E P S M 2 1 0からT P I 2 2 0へパケット・データを送るためのH S B 2 0 6を介した送信データ転送動作を図解するフローチャートである。動作はステップ890で開始し、T P I 2 2 0のポート状態ロジック820がT P I T X F I F Oのいずれか1つに、対応するX B S I Z Eと比較して等しいか大きいバッファ・スペースがあることを検知する。動作は次のステップ891へ進み、ポート状態ロジック820は、E P S M 2 1 0のポーリングに응答してB U F _ A V A I L [6] *信号を多重化の方式で適切にアサートし、対応するT P I T X F I F O内に使用可能なバッファ・スペースがあることを表示する。前述したように、このポーリングは独立して発生し、クラリフィケーションの目的で含まれている。次のステップ892において、十分なスペースのあるT P I T X F I F Oに対してE P S M 2 1 0が転送するだけの十分なデータがあるとき、E P S M 2 1 0は、H S B 2 0 6上でそのT P I T X F I F Oに対応するポートへのライト・サイクルを開始する。続くステップ893では、T P I 2 2 0のポート状態ロジック820がH S B 2 0 6上のライト・サイクルを検出し、指示されたポートに適切なT P I T X F I F Oを選択する。次のステップ894において、E P S M 2 1 0はT P I 2 2 0に対してH S B 2 0 6を介し1バーストのデータを転送し、T P I 2 2 0はそのデータをT P I 2 2 0内の対応するT P I T X F I F Oに書き込む。

30

40

【0180】

ステップ895aにおいて、T P I 2 2 0がステップ894のデータ・バースト中にS O P *信号がアサートされたことを検知した場合、そのパケット・サイズを持っているデータの先頭のD W O R Dは、ステップ895bにおいてP A C K E T S I Z Eタグ・レジスタ819cに入れられる。ステップ896aにおいて、T P I 2 2 0が、ステップ894でのデータ・バースト中にE O P *信号がアサートされたことを検知すれば、T P I 2 2 0はステップ896でパケットの終わりを表すT P I 2 2 0内のフラグをセットする。ステップ896aまたは896bから、動作はステップ890へ戻る。

【0181】

50

次に図40を参照する。該図は、各T L A N 2 2 6に関するネットワーク・スイッチ102の送信動作を図解するフローチャートである。最初のステップ867において、T P I 2 2 0はT P I T X F I F Oのいずれか1つの中にデータを検知し、それに応答してP C I バス2 2 2を要求し、P C I アービタ8 1 1からこれの制御権を受け取る。次のステップ868で、T P I 2 2 0は、対応するT L A N 2 2 6のコマンド・レジスタ8 2 8 aに送信コマンドを書き込む。T P I 2 2 0は、その後ステップ869で、P C I バス2 2 2の制御権を放棄する。

【0182】

続くステップ870aにおいて、送信コマンドを受け取ったT L A N 2 2 6は、P C I バス2 2 2の制御権を要求し、P C I アービタ8 1 1からこれの制御権を受け取り、T P I 2 2 0に対して送信コントロール・リストを要求する。次のステップ870bで、T P I 2 2 0はP C I バス2 2 2の制御権を持っているT L A N 2 2 6に送信コントロール・リストを供給し、T L A N 2 2 6はその送信コントロール・リストをその送信コントロール・リスト・バッファ8 2 7 bに入れる。続くステップ870cにおいて、T L A N 2 2 6はP C I バス2 2 2の制御権を放棄するが、直ちにステップ870dでP C I バス2 2 2の制御権を再要求する。再びP C I バス2 2 2の制御権を得ると、T L A N 2 2 6はステップ871aでT P I 2 2 0に対して1バーストのデータを要求し、送信コントロール・リストの実行を開始する。特に、T L A N 2 2 6は、ステップ871aにおいて、P C I バス2 2 2上でP A C K E T D A T A M E M O R Y B A S E A D D R E S Sをアサートする。続くステップ871bでは、T P I 2 2 0がそれに応答して、対応するT P I T X F I F Oを選択してイネーブルし、P C I バス2 2 2を介してT L A N 2 2 6にデータを供給する。それぞれのデータ・バースト後、T L A N 2 2 6は、ステップ871cにおいてP C I バス2 2 2の制御権を放棄する。ステップ872aにおいて、データの packets 全体の転送が完了していないと判定すると、動作はステップ870cに戻り、T L A N 2 2 6は再びP C I バス2 2 2の制御権を要求し、最終的に該制御権を取り戻す。

【0183】

ステップ872aにおいて、パケットの送信が完了していると判定すると、動作はステップ873aに移り、T L A N 2 2 6はT P I 2 2 0に対してデータ転送完了の旨を書き込み、T P I 2 2 0はそれに応答して動作の完了を表示する。特に、T L A N 2 2 6はT X C N T L L I S T 8 0 8 bのC S T A Tフィールド8 3 2 bに最後の1 D W O R Dの書き込みを行い、C S T A Tフィールド8 3 2 b内のフレーム完了ビットをセットする。さらに、T X C N T L L I S T 8 0 8 bのP A C K E T _ S I Z Eフィールド8 3 2 aが、T P I 2 2 0によってT L A N 2 2 6に送信されるべきデータ・パケットのサイズでロードされる。T L A N 2 2 6は、書き込み動作を完了すると、ステップ873bでP C I バス2 2 2を放棄する。ステップ873bから、動作は次の送信動作に備えてステップ867に戻る。

【0184】

C P U 2 3 0による初期化後、T P I 2 2 0はT L A N 2 2 6と協調して動作するように構成されることによって、C P U 2 3 0がネットワーク・スイッチ102の他の重要なタスクや機能を遂行することができる点は、高い評価に値する。C P U 2 3 0は、P C I バス2 2 2上のデバイスのタイプや数を確認し、対応するアドレス値を割り当てて、P C I メモリおよび入出力スペースを初期化する。C P U 2 3 0は、T L A N 2 2 6のアドレス値をT P I 2 2 0に供給する。さらに、C P U 2 3 0はT P I 2 2 0のアドレスの初期値をそれぞれのT L A N 2 2 6に供給し、コマンドを挿入して動作を起動する。T L A N 2 2 6は、コントロール・リストを要求して該コントロール・リストを実行し、そのコントロール・リスト内のアドレスにあるメモリとの間で、データの読み書きを行うように構成される。T P I 2 2 0はまた、各コントロール・リストを更新し、それぞれを、要求している各T L A N 2 2 6に供給するように構成される。さらに、T P I 2 2 0は、適宜のT L A N 2 2 6にコマンドを書き込んで送信動作を開始するよう構成され、また対応する送

10

20

30

40

50

信コントロール・リストを後続の要求に応じて供給するように構成される。このようにして、CPU 230は初期化の実行後は、ネットワーク・スイッチ102の他の機能を自由に遂行することができる。

【0185】

図41は、メモリ212の編成を図解するブロック図である。示した実施例では、メモリ212のサイズは4～16メガバイト(Mbyte)であるが、このメモリ・サイズは可変であって、必要に応じた増減が可能である。図41～図47に示すメモリ・セクション・ブロックの幅、従って各メモリ・ラインの幅は、1DWORDすなわち32ビットである。メモリ212は、ハッシュ・メモリ・セクション902およびパケット・メモリ・セクション904という、2つの主なセクションに分けられる。ハッシュ・メモリ・セクション902はネットワーク・デバイス識別セクションとして機能し、ネットワーク・スイッチ102に結合しているネットワーク106、112内の1つまたは複数のネットワーク・デバイスを識別する。ハッシュ・メモリ・セクション902のサイズは、必要なデバイス、関連のアドレスおよびエントリの数に基づいて、プログラミングすることができる。示した実施例において、ハッシュ・メモリ・セクション902は256キロバイト(Kbyte)のメモリで、最小8K($K=2^{10}=1,024$)から最大16Kまでのアドレスをサポートする。ハッシュ・メモリ・セクション902は、メモリ212内のどこに置かれてもよく、示した実施例ではメモリ212の先頭に位置している。パケット・メモリ・セクション904のサイズは、メモリ212の残りの領域、すなわちハッシュ・メモリ・セクション902が使用していない部分である。

10

20

【0186】

図42は、メモリ212のハッシュ・メモリ・セクション902の編成を示すブロック図である。ハッシュ・メモリ・セクション902は長さが256キロバイトとして示されているが、ハッシュ・メモリ・セクションのサイズは固定、あるいは必要に応じてプログラミング可能であることは理解されよう。ハッシュ・メモリ・セクション902は、一次的なハッシュ・エントリのための1番目の128キロバイトの一次ハッシュ・エントリ・セクション906、およびチェーン(連鎖)・ハッシュ・エントリ用の2番目の128キロバイトのチェーン・ハッシュ・エントリ・セクション908という、2つの128キロバイトのセクションに分かれている。セクション906、908の各々は、それぞれの長さが16バイトの8Kのエントリを含む。

30

【0187】

図43は、一次ハッシュ・エントリ・セクション906とチェーン・ハッシュ・エントリ・セクション908の両方を含む、ハッシュ・メモリ・セクション902内の各エントリを表すハッシュ・テーブル・エントリ910の編成の図解である。各エントリ910は、ネットワーク・スイッチ102に結合しているネットワーク106、112のネットワーク・デバイスの1つに対応する。各一次エントリは1つのハッシュ・アドレスに存在し、ハッシュ・アドレスはそのデバイスのMACアドレスを「ハッシュ」して決定される。特に、ネットワーク・デバイスには、物理アドレスあるいはMACアドレスとも呼ばれる48ビットのハードウェア・アドレスが割り当てられ、このアドレスは製造過程において、あるいはネットワークの設置中に、ジャンパまたはスイッチを設定して各ネットワーク・デバイスに割り当てられる一意の数値である。このMACアドレスの一部は、米国電気電子技術者協会IEEE(Institute of Electrical and Electronics Engineers)によって製造業者に割り当てられたもので、その製造業者のすべての製品に共通しており、ハードウェア・アドレスの他の一部は、ハードウェアの製造業者が割り当てた一意の値である。ハッシュ・テーブル・エントリ910最初の6バイト、すなわちバイト5～0には、その項目に関連するデバイスのMACアドレスが入っている。従ってネットワーク・スイッチ102は、そのMACソース・アドレスを含むデータ・パケットを送信する各ネットワーク・デバイスに、1つのハッシュ・テーブル・エントリを付加する。

40

【0188】

50

ネットワーク106、112内の各ネットワーク・デバイスから送信されるそれぞれのデータ・パケットは、一般に送信元と受信先のMACアドレスを含み、これらは両方とも、いくつかのアルゴリズムの1つに従ってハッシュされるものである。示した実施例においては、各MACアドレスの2つの部分を論理的に結合あるいは比較して対応するハッシュ・アドレスを算出する。各部分は13ビットから16ビットであり、排他的論理和(XOR)のロジックを使用してビット単位的方式で結合され、13から16ビットのハッシュ・アドレスを形成する。例えば、最初の16ビットのMACアドレスMA[15:0]と、次の16ビットのMACアドレスMA[31:16]とのビット単位的方式による論理和が、ハッシュ・アドレスHA[15:0]となる。ある実施例では、ハッシュされた結果の最初の13、14、15、または16ビットが、ハッシュ・アドレスHAとして使用される。あるいは、MACアドレスの最初の13ビットのMA[12:0]を次の13ビットのMA[25:13]とハッシュして、13ビットのハッシュ・アドレスHA[12:0]を得る。もしくは、MACアドレスの最初の14ビットのMA[13:0]を次の14ビットのMA[27:14]とハッシュして、14ビットのハッシュ・アドレスHA[13:0]とするなど、以下同様に行われる。ハッシュ処理には多種多様なアルゴリズムが知られており、当業者には既知のように、アドレス・ビットの特定の組み合わせの結合に用いられること、および本発明は何ら特定のハッシュ法に限定されるものではないことは、理解されであろう。

【0189】

ハッシュ・アドレスは、一次ハッシュ・エントリ・セクション906内のそれぞれのハッシュ・エントリの位置を特定するための実アドレス、またはオフセット・アドレスとして使用される。MACアドレスは一意であるが、ハッシュ・アドレスの場合は、異なった2つのMACアドレスが同じハッシュ・アドレスにハッシュする限りにおいて、一意である必要はない。チェーン・ハッシュ・エントリ・セクション908は、本明細書で後に詳述するように、異なったデバイスの重複したハッシュ・アドレスを格納すべく用意されている。ハッシュ・アドレスでアクセスできる一次ハッシュ・エントリ・セクション906と、一次ハッシュ・エントリ・セクション906の先頭エントリ内にあるリンク・アドレスでアクセス可能なチェーン・ハッシュ・エントリ・セクション908による編成により、少なくとも1つのブランチ動作が節約できる。ポインタのリストを用いてテーブル・エントリにアクセスするのではなく、メモリ212内の最初のエントリは1回のブランチ動作で検索され、次のエントリは2回目のブランチ動作で、というように以下同様である。このように、メモリ212の以上のような編成によってアクセス1回につき少なくとも1つのブランチ動作が節約できるため、ハッシュ・エントリに対するアクセスの効率が向上する。

【0190】

ハッシュテーブル・エントリ910の次のバイト(6)には、デバイスが接続されている関連のポート番号を識別する2進ポート番号(PortNum)が入っており、ここでPORT0のポート番号はゼロ、PORT1のポート番号は1、(CPU230の)PORT28のポート番号は28、というようになっている。次のバイト(7)は、制御およびエイジ情報バイト(CONTROL/AGE)であり、エントリが有効であるかどうかを識別するバリッド・ビット(VVALIDENTRY)を含み、これがロジック1であればそのエントリは有効、ロジック0であればその項目は有効でない、つまり空ビットであることを示す。CONTROL/AGEバイトは、このデバイスに関する最後のソース・アクセスからの経過時間を表す2進のエイジ数(AGE)を含む。最後のソース・アクセスから予め決められた不使用の時間量が経過すれば、デバイスは老化してCPU230によってハッシュ・エントリから削除される。経過時間はいくつかの方法の1つを用いて測られ、その単位は秒かそれ以下、分、時、その他である。デバイスを老化とみなす不使用時間は、プログラミングが可能である。他の実施例において、AGE数は特定のデバイスが「旧」であるかどうかを表すために用いられる1つのビットであり、一定の経過時間あるいはそのような要因で設定される。

10

20

30

40

50

【0191】

次の4バイト (B:8) は、もし適用されていれば、ポートのグループを表す29ビットの仮想LAN (VLAN) のビットマップ値を定義する。VLAN値の各ビットはポートのそれぞれ1つに対応し、デバイスかポートがそのポートとグループ化されれば、セットされる。従ってVLAN値は、その他のポートのうち、どのポートがデバイスとグループ化されたかを識別する。これにより、ネットワーク106、112を任意の組み合わせでグループ化して、ネットワーク・スイッチ102に結合された複数の異なったTLANを形成することができる。例えば、最初の5ポートPORT0～PORT4が一緒になってグルーピングされれば、それぞれのVLAN値は0000001Fhとなる。ここで、hは16進数を示す。PORT2に結合されている1つのデバイスから送られたBCパケットは、PORT0、PORT1、およびPORT3にリピートされ、ネットワーク・スイッチ102のその他のすべてのポートにはリピートされない。VLAN値が全部1か1FFFFFFFhであれば、そのデバイスにグループ化が適用されていないことを表している。1つのデバイスを複数のグループに関連させられることに留意する必要がある。他の実施例においては、各デバイスが属するいくつかのVLANグループがあれば、それらの2つ以上を識別するために1つのVLANフィールドを含むことができる。

10

【0192】

各ハッシュ・テーブル・エントリ910の最後の4バイト (F:C) は、チェーン・ハッシュ・エントリ・セクション908内で、もしあれば、同じハッシュ・アドレスを持った次のエントリを指示するリンク・アドレス (LINK A[31:0] すなわちLINK ADDRESS) である。次のエントリは、チェーン・ハッシュ・エントリ・セクション908内で次の使用可能なロケーションに格納されている。このように、2つの異なったデバイスの2つのMACアドレスが同一のハッシュ・アドレスにハッシュすれば、最初の、すなわち「一次」エントリが一次ハッシュ・エントリ・セクション906に格納され、2番目のエントリがチェーン・ハッシュ・エントリ・セクション908内に格納され、一次エントリのLINK ADDRESSが2番目のエントリを指示する。別のMACアドレスが最初の2つと同じハッシュ・アドレスをハッシュすれば、各追加エントリはチェーン・ハッシュ・エントリ・セクション908に格納され、LINK ADDRESSによって連続した順序で、一緒に連鎖される。従って、最初が2番目を指示し、2番目が3番目を指示し、以下同様となる。それぞれのエントリは、ハッシュ・テーブル・エントリ910のフォーマットに従う。LINK ADDRESSの形式は、適宜自由に定義することができる。LINK ADDRESSは一般に、メモリ212内のハッシュ・メモリ・セクション902を指示するベース・アドレス・ポーション、およびハッシュ・メモリ・セクション902内の実際のエントリへのオフセット・ポーションを含む。下位のアドレス・ビットは、バイト整合のために必要に応じてゼロに設定する。各チェーン内の最後のエントリは、LINK ADDRESSの一部をゼロにセットして識別する。例えば、LINK ADDRESSのビット[31:28]をゼロにセットして、最後のエントリを表す。

20

30

【0193】

図44は、メモリ212のパケット・メモリ・セクション904の編成を示すブロック図である。示した実施例において、メモリ212のパケット・メモリ・セクション904は複数の隣接した等しいサイズのセクタ912として編成され、各セクタ912は、セクタ・プレフィクス914と呼ばれるセクタ情報セクション、および1つまたは複数のパケット・データ・ブロックを含むパケット・セクション916を含む。各セクタ912は、設計を簡略化しオーバーヘッドを下げるため、メモリ212の機能を遂行するメモリ・デバイスのページ・サイズに対応して、そのサイズを2Kバイトにすることが望ましい。示した実施例において、FPM DRAM SIMMは4Kバイトのページ・バウンダリ (境界) で編成され、同期DRAM SIMMは2Kbyteのページ・バウンダリで編成されている。従って、2Kバイトのセクタ・サイズで、サポートされるタイプのメモリ・デバイスに十分である。セクタ912は初期において空であるが、LINK ADDRESS

40

50

Sと一緒にチェーン化されて、空メモリ・セクタのFREE POOL CHAIN（フリープール・チェーン）を形成する。

【0194】

ポート104、110のそれぞれから新しい情報のパケットが受け取られると、1つまたは複数のセクタ912がFREE POOL CHAINから切り離され、1ポートにつき1つのRECEIVE SECTOR CHAIN内で一緒にリンクされる。また、各パケットは、同じまたは別のRECEIVE SECTOR CHAIN内で他のパケットとリンクされ、1ポートにつき1つのTRANSMIT SECTOR CHAIN形成する。このようにして、1つのポートのRECEIVE SECTOR CHAIN内のパケットは、さらに別のポートのTRANSMIT SECTOR CHAINにも入れられる。セクタ912のパケット・セクション916内のデータがすべて受信先のポートへ送信されると、そのセクタは、そのRECEIVE SECTOR CHAINから解放され、再びFREE POOL CHAINに戻ってリンクされる。RECEIVE SECTORおよびFREE POOLチェーンの機能は、本明細書で後に詳述する方式で、1つのセクタから次のセクタへのリンク・アドレスあるいはポインタを用いて遂行される。

10

【0195】

図45は、パケット・メモリ・セクション904の各セクタ912の各セクタ・プレフィクス914の編成の図解である。セクタ・プレフィクス914は、対応するセクタ912の情報を含み、さらに次のセクタ912があれば、それへのリンクとして機能する。プレフィクスの情報部分は、セクタ912内のどこに入っているてもよい点に留意されたい。最初のバイト（0）は、そのときのセクタ912内のパケットまたはパケット片の数を表す2進のセクタ・パケット・カウント（SecPktCnt）を定義する。セクタ・パケット・カウントは、そのセクタにパケット・データが格納されると増分され、受信先のポートによる送信のためにデータが読み出されると減分される。セクタ・パケット・カウントSecPktCntがゼロに減分されたとき、そのセクタがRECEIVE SECTOR CHAINの最後にあるものでなければ、該セクタは解放されてFREE POOL CHAINに戻る。次のバイト（1）は、受信したパケットの送信元ポートを示すセクタ・ソース値（SecSource）である。この値は、そのセクタが解放されてFREE POOL CHAINに戻るとき、当該受信ポート・セクタ・カウント（RxSecCnt）を識別して減分するために必要である。次の2つのバイト（3：2）は、リザーブすなわち未使用となっている。

20

30

【0196】

それぞれのセクタ・プレフィクス914内の次の4バイトは、対応するRECEIVE SECTOR CHAINまたはFREE POOL CHAIN内の次のセクタへのネクスト・リンク・アドレス（NextSecLink）である。同一のリンク・アドレスが両方の目的に使用されているが、異なったリンク・アドレスを用いてもよい。示した実施例において、NextSecLinkアドレスは32ビットで、ベース（基準）およびオフセットの部分から成る。下位の“n”個のビットは、NextSecLinkのセクタ・サイズに応じた整合のために、ゼロにセットしてもよい。整数“n”は、4Kバイトのセクタでは12、2Kバイトのセクタでは11、11Kバイトのセクタでは10、そして512Kバイトのセクタでは9である。示した実施例においては、nは2Kバイトのセクタに11、などとなっている。このようにして、ポート104、110から1つまたは複数のパケットが受け取られると、そのポートによって受信されたその1つまたは複数のパケットを格納すべく、セクタ912のRECEIVE SECTOR CHAINが1つ割り当てられる。複数のセクタ912は、そのチェーン内の各セクタ912のセクタ・プレフィクス914内のNextSecLinkアドレスを用いて、チェーン化の方式で一緒にリンクされる。パケット・データは、各RECEIVE SECTOR CHAIN内のそれぞれのセクタ912のパケット・セクション916内に順に格納される。1つのパケットのパケット・データは、RECEIVE SECTOR CHAIN内のセクタ・バウンダリを越えてもよいという点に留意する必要がある。セクタ・プレフィクス91

40

50

4の最後の8バイト(15:8)は、リザーブすなわち未使用となっている。

【0197】

図46は、パケット・セクション916内の各パケット・データ・ブロックを表す例示的なパケット・データ・ブロック917の図である。パケット・データ・ブロック917は、パケット・ブロック・ヘッダ918およびパケット・データ・セクション920という2つの部分に分かれている。パケット・ブロック・ヘッダ918は、MCB404によって各パケットの前に付加されてパケット・データ・ブロック917を形成するのが望ましい。パケット・ブロック・ヘッダ918の最初の2バイト(1:0)は、パケットの長さをバイト数で定義する15ビットの2進パケット長(PktLength)値、およびCTモードのパケットがポートの停動(stall)のためメモリ212へ転送されたときにセットされる1ビットの中間(ミッド)パケットCT値(MidPktCT)を形成する。MCB404は、TLAN226のポートPORT24とPORT27、およびCPU230のポートPORT28へ送信するとき、PktLengthを含むこの最初のDWORDをパケットに付加する。パケット・ブロック・ヘッダ918の次のバイト(2)は、パケットのソース・ポートすなわち送信元ポート(SourcePort)番号を識別する。これは、ソース・アドレスに関するポート番号を識別するための8ビットで2進のポートID番号である。送信元ポートは、そのパケットが格納されている特定のRECEIVE SECTORCHAINによっても識別される。次のバイト(4)は、宛先ポートすなわち受信先ポート(DestPort)番号を識別する。これは、SourcePort値の場合と同様に、受信先のポート番号を識別するための8ビットで2進のポートID番号である。受信先ポートは、そのパケットが属する特定のTRANSMIT PACKETCHAINによっても識別される。

【0198】

パケット・ブロック・ヘッダ918の4バイト(11:8)は、TRANSMIT PACKETCHAIN内の次のデータ、またはパケット・データ・ブロック917への32ビットのネクスト・リンク・アドレス(NextTxLink)を定義する。送信パケット・カウンタ(TxPktCnt)がゼロまで減分されたとき、TRANSMIT PACKETCHAINの終わりが表示される。NextTxLinkアドレスの下位ビットA0は、次のパケットがブロードキャストであるか否かを示すBCパケット・ビット(NextPktBC)として使用される。NextPktBC=1であれば、次のパケットは後述するブロードキャストの形式であり、もしNextPktBC=0であれば、次のパケットは非ブロードキャストである。NextTxLinkアドレスの次の下位ビットA1は、次のパケットがSnFであるか否かを同様に表示するSnFパケット・ビット(NextPktSnF)として使用される。NextTxLinkアドレスの下位半バイト(4ビット)は、その半バイトの実際の値にかかわらず、バイト整合の目的にゼロと想定してもよいことに留意されたい。従って、例えばNextTxLinkアドレスが読み取られるとき、ビットA[3:0]が実際はNextPktBC=1のような値であっても、これを無視してゼロと想定することができる。これにより、これらのビットは代替用途に使用することができる。示した実施例においては、下位ビットA[3:0]がゼロと想定されるように、データ構造が16バイト整合となっている。

【0199】

示した実施例においては、パケット・データ・セクション920がパケット・ブロック・ヘッダ918の直後に置かれ、パケット・ヘッダ内でデータフィールドの長さが定義される。しかし、示した実施例における各セクタの特定の序列や各値の特定の位置などは多少恣意的であって例示の域を出ないものであり、従って本発明の範囲を越えない限りにおいて、編成は必要に応じて任意である。

【0200】

先に述べたように、パケットは、ポートPORT0~PORT28の各々から検索され、セクタ912の対応する受信セクタ・チェーン(RECEIVE SECTORCHAIN)に格納される。受信セクタ・チェーンは、ポート当たり1つ対応して設けられてい

10

20

30

40

50

る。図48示されるように、第1の受信セクタ・チェーン930がPORT0に対して示され、ここで第1のセクタ931のセクタ・プレフィックス914におけるNext Sec Linkを用いて、セクタ931が別のセクタ932にリンクされる。必要に応じて、セクタ・プレフィックス914におけるリンク・アドレスを用いて、更に他のセクタがリンクされる。また、第2の受信セクタ・チェーン940がPORT1に対して示され、このポートで、第1のセクタ941のセクタ・プレフィックス914におけるNext Sec Linkを用いて、セクタ941が別のセクタ942にリンクされる。あるポートで受取られた各パケットごとに、パケット・ブロック・ヘッダ918が、対応する受信セクタ・チェーンのその時のセクタ（現在セクタ）912のパケット・セクション916において前に受取られたパケット・データ・ブロック917の直後に置かれ、パケット・ブロッ 10
ク・ヘッダ918に、そのパケット・データ・セクション920が後続する。現在セクタ912のパケット・セクション916がパケット・データ・ブロック917を格納中に一杯になると、別のセクタ912がフリープール・チェーン（FREE POOL CHAIN）から割付けられ、当該ポートに対する受信セクタ・チェーンリンクされる。このように、ポートから受取ったパケット・データ・ブロック917は、当該ポートに関して対応する受信セクタ・チェーン内に連続的に配置される。また、セクタ912のパケット・セクションは、パケット全体および（または）パケットの部分を含むことができる。

【0201】

したがって、図48に示されるように、ポートPORT0で受取られたパケット・データ・ブロック934、935および936が、セクタ931、932内に配置される。パケ 20
ット・データ・ブロック935がセクタ931、932に跨がることに注目されたい。同様に、ポートPORT1で受取られたパケット・データ・ブロック944および945が、図示のように、セクタ941、942内に置かれ、パケット・データ・ブロック945がセクタ941、942に跨がっている。

【0202】

各パケットはまた、各宛先ポートに対するパケットの送信パケット・チェーン（TRANSMIT PACKET CHAIN）と関連させられ、該ポートでは、これらのパケットが、Next Sec Linkアドレスを用いて、一緒にリンクされる。各送信パケット・チェーンにおけるパケットは一般に、ネットワーク・スイッチ102により受取られる 30
時間に基いて順序付けられ、その結果、関連する宛先ポートへ送られる時、この順序が維持される。例えば、図48に示されるように、パケット・データ・ブロック934、944がポートPORT10から送られべきであり、そしてパケット・データ・ブロック934がパケット・データ・ブロック944の直前に送られるべきならば、パケット・データ・ブロック934のパケット・ブロック・ヘッダ918のNext Tx Linkアドレスが、パケット・データ・ブロック944を指示する。パケット・データ・ブロック944のパケット・ブロック・ヘッダ918のNext Tx Linkアドレスは、次に送られるべきパケット・データ・ブロックを指示する、の如くである。伝送の実際の順序は、1つのパケットが送信パケット・チェーンへリンクされる時に決定される。CTモード・パケットは、このパケットが受取られる時の初めにリンクされ、SnFモード・パケット 40
は、パケット全体が格納された後にリンクされる。中間パケット暫定CTモード・パケットは、適切な順序付けを保証するため、対応する送信パケット・チェーンの前にリンクされる。

【0203】

図47は、正規（通常）のパケット・ブロック・ヘッダ918を置換する、BC（ブロードキャスト）パケットに対して用いられる128バイトのパケット・ヘッダ922を示すブロック図である。BCパケットにおいては、Next Pkt BC値が前のパケットにセ 50
ットされて現在パケットがBCパケットであることを示す。各送信パケット・チェーンが、伝送されるBCパケットを含む全てのポートに対して維持されるべきである。従って、BCパケット・ヘッダ922は、0～28が番号が付された各ポート（ポート104、110及びCPU230を含む）ごとに、4バイトのリンク・アドレス（Port # Ne

x t T x L i n k) を含み、各 N e x t T x L i n k アドレスが、リストにおける場所 (ポート番号 P o r t #) により識別される対応ポートと関連する送信パケット・チェーンにおける次のパケットを指示する。このように、N e x t T x L i n k アドレスは、バイト (1 1 : 8) で始まり、バイト (1 2 3 : 1 2 0) で終る。第 1 の N e x t T x L i n k アドレス・エントリ (1 1 : 8) は、第 1 のポート P O R T 0 に対するメモリ 2 1 2 における次のパケットと対応し、第 2 のエントリ (バイト 1 5 : 1 2) は、第 2 のポート P O R T 1 に対するメモリ 2 1 2 における次のパケットに対する N e x t T x L i n k アドレスである。このように、C P U 2 3 0 に関する次のパケットに対する N e x t T x L i n k である最後のエントリ (バイト 1 2 3 : 1 2 0) まで続いている。各 B C リンク・アドレスもまた、各送信パケット・チェーンにおける次のパケットが B C パケットか否かを示す次の B C パケット (N e x t P k t B C) ビットと、各送信パケット・チェーンにおける次のパケットが S n F パケットか否かを示す次の S n F パケット (N e x t P k t S n F) ビットとを含んでいる。

10

【0204】

B C パケット・ヘッダ 9 2 2 の最初の 4 バイト (3 : 0) は、正規のパケット・ブロック・ヘッダ 9 1 8 の最後の 4 バイトに類似し、M i d P k t c t 値が B C パケットに対してゼロであることを除いて、P k t L e n g t h、M i d P k t C T、S o u r c e P o r t (ソース・ポート) および D e s t P o r t (宛先ポート) の値を含んでいる。B C パケット・ヘッダ 9 2 2 の次の 4 バイト (7 : 4) は、バイト 2 8 : 0 の各々が B C パケット・データを受取るポートに対応するブロードキャスト・ポート・ビットマップ (B C P o r t s) である。各ビットは、パケットが対応するポートへ送られる時にクリアされる。全ての B C ーポート・ビットがクリアされた時、先に述べた S e c P k t C n t カウンタもまた減分される。

20

【0205】

図 4 9 には、各々が同じ B C パケット 1 0 1 0 を包含する幾つかの送信パケット・リンクを示すブロック図が例示される。この例では、ポート 1、5、1 1 および 1 2 が、V L A N 関数などを用いてグループ化され、その結果、ポート 1 2 の如き 1 つのソース・ポート (例えば、ポート 1 2) で受取られる B C パケット 1 0 1 0 のデータが当該グループにおける残りのポート (ポート 1、5 および 1 1) に複写される。4 つの送信パケット・チェーン 1 0 0 2、1 0 0 4、1 0 0 6 および 1 0 0 8 が、それぞれポート 1、5、1 1 および 1 2 に対して示される。送信パケット・チェーン 1 0 0 2、1 0 0 4 および 1 0 0 6 は、幾つかの一般的な非ブロードキャスト・パケット 1 0 0 0 を B C パケット 1 0 1 0 とリンクする。ポート 1 2 がソース・ポートであるから、B C パケット 1 0 1 0 はポート 1 2 に送られず、従ってこのポートは送信パケット・チェーン 1 0 0 8 には含まれない。B C パケット 1 0 1 0 は B C パケット・ヘッダ 1 0 1 2 を含み、このヘッダは、ポート 1 の送信パケット・チェーン 1 0 0 2 における次のパケット 1 0 0 0 を指示するリンク・アドレス 1 0 1 6 と、ポート 5 の送信パケット・チェーン 1 0 0 4 における次のパケット 1 0 0 0 を指示するリンク・アドレス 1 0 1 8 と、ポート 1 1 の送信パケット・チェーン 1 0 0 6 における次のパケット 1 0 0 0 を指示するリンク・アドレス 1 0 2 0 とを含む、各ポートに 1 つずつリンク・アドレスのリストを含んでいる。このように、送信パケット・チェーン 1 0 0 2、1 0 0 4 および 1 0 0 6 の各々が保持される。各送信パケット・チェーンが 1 つ以上の B C パケットを含み、これが必要に応じて、非連続的あるいは連続的に現れることも判る。

30

40

【0206】

図 5 0 は、1 組の M C B パケット制御レジスタ 1 1 0 2 を示すブロック図であり、これらのレジスタは S R A M 6 5 0 内に備えられて、ネットワーク・スイッチ 1 0 2 の C P U 2 3 0 を含む 2 9 個のポート 1 0 4、1 1 0 の各々に対して同様に備えられている。C P U 2 3 0 は、スパニング・ツリー処理のためのブリッジ・プロトコル・データ・ユニット (B P D U) の送受などのある目的のため、「ポート (P O R T 2 8)」として扱われる。各 M C B パケット制御レジスタ 1 1 0 2 は、受信セクション 1 1 0 4 と送信セクション 1

50

106とを含んでいる。受信セクション1104では、28ビットの受信パケット・ヘッダのベース・ポインタ(RxBaSePtr)が、当該ポートに対する受信セクタ・チェーンの初めである対応ポートに対応するその時の受信パケット・ヘッダのベース(基底)に対するポインタである。メモリ212について先に述べたように、SRAM650に対するデータ構造は、16バイトが割り当てられ、全てのポインタの最下位ビットA[3:0]がゼロと仮定される。28ビットのそのときの受信ポインタ(RxCurPtr)は、当該ポートの受信セクタ・チェーンに関するその時のデータ記憶場所に対するポインタである。RxCurPtr値の下位4ビットは、受信BCパケット表示ビット(RxBC)と、「パケット開始(SOP)」フラグとして用いられる受信伝送進行中(RxIP)ビットと、その時のパケットがセクター境界と交差するかどうかを示す多重セクタ・パケット(MultiSecPkt)ビット1と、送信リンクがパケットの終りで更新されることを示すSnFビット0とを含む、制御ビットである。受信セクション1104は更に、MidパケットCTビット(MidCT)と、RxCurPtrまでのバイトで受取られるその時のパケットの長さに等しい16ビットの受信パケット長(RxPktLn)値と、対応するポートによりその時使用中であるセクタの数を示す16ビットの受信ポート・セクタ・カウンタ(RxSecCnt)と、各ポートまたは受信セクタ・チェーンに対して許容されるCPUプログラムされたセクタ最大数を識別する16ビットの受取りセクタ閾値(RxSecThreshold)値とを含んでいる。RxSecThreshold値は、該RxSecThresholdをRxSecCntと比較することにより、バックプレシャが当該ポートに対して加えられるべきかどうかを決定するために用いられる。バックプレシャがディスエーブル(不動作)状態にされると、RxSecThreshold値を用いて、対応するポートで受取られる更なるパケットをドロップする(捨てる)。

【0207】

受信セクション1104は更に、対応するポートに対する送信パケット・チェーンにおける最後のパケットのベースを示す28ビットのポインタである送信キュー・ポインタ(EndOfTxQPtr)の終りを含んでいる。最後に、送信キューBC(EoQ__BC)の終りが、対応するポートに対する送信パケット・チェーンにおける最後のパケットに対するブロードキャスト・フォーマットを示すようにセットされる。

【0208】

送信セクション1106は、対応するポートに送信パケット・チェーンに関する情報を提供する。送信ベース・ポインタ(TxBaSePtr)は、その時の伝送パケット・ヘッダのベースに対する28ビットのポインタであり、別の28ビットの送信の現在ポインタ(TxCurPtr)が、対応するポートに対するその時のデータ検索場所を指示する。送信ブロードキャスト(TxBC)ビットが、パケット・ヘッダがブロードキャスト・フォーマットであることを示すようにセットされる。送信進行中(TxIP)ビットが論理値1にセットされると、それにより、送信がその時ポートに対して進行中であり、SOPを示す。8ビットの送信ソース・ポート(TxSrcPort)番号は、SOPにおけるパケット・ヘッダから読出されるその時の送信パケットのソース・ポート番号ある。16ビット送信パケット長(TxPktLn)値は、その時の送信パケットに対して送られるべき残りのバイトと等しい。あるパケットが伝送されるとき、パケットのパケット・ブロック・ヘッダ918におけるPktLength値が送信セクション1106におけるTxPktLn値へ複写され、次いでTxPktLn値は、パケットが伝送される時、TXコントローラ606によって減分される。TxPktLn減分されてゼロになると、EPM210が、パケットの終りを示す対応するEOP*信号を生成する。16ビットの最大パケット数(TxPktThreshold)値は、各ポートに対してキューさせられるCPUプログラムされたパケットの最大数に等しい。CPU230を宛て先とするパケットがTxPktThresholdまたはRxPktThresholdの制限を受けないことが判る。最後に、16ビットの送信パケット・カウンタ(TxPktCnt)は、対応するポートに対してその時にキューされるパケットの数に等しい。

10

20

30

40

50

【0209】

図51は、SRAM650に置かれたフリープール・パケット制御レジスタ1108を示すブロック図であり、これらのレジスタは、レジスタのフリープール・チェーン(FREE POOL CHAIN)と関連されている。各フリープール・レジスタ1108は、フリープール・チェーンにおける次の自由なフリー・セクタに対するポインタ(Next Free SecPtr)と、フリープール・チェーンにおける最後のセクタに対するポインタ>Last SecCnt)と、その時利用可能なフリー・セクタの数に等しいフリー・セクタ・カウント(Free SecCnt)と、メモリ・オーバーフロー・フラグ(MOF)がバックプレシャまたはフィルタリング(パケットの抜き取り)の目的のためにセットされる前に許容される、CPUプログラムされたセクタの最小数に等しいフリー・セクタ閾値(Free SecThreshold)数と、その時にメモリ212にあるBCパケット数に等しいBCパケット・カウント(BC PktCnt)と、メモリ212に許容されるBCパケットのCPUプログラムされた最大数に等しいBCパケット閾値(BC PktThreshold)カウントとを含んでいる。

10

【0210】

図52は、メモリ212へのデータ・パケットの受取りのため、およびCT動作モードにおけるデータ・パケットの送信のための、ネットワーク・スイッチ102の動作をフロー図で示している。データは通常、リアルタイムであるいは全体的にパケットの形態におけるネットワーク・スイッチ102のポートPORT0~PORT27により送受信され、セグメント108、14に跨がって送られている間は、細分割されることはない。しかし、ネットワーク・スイッチ102内のFIFOは一般に、全パケットを格納するのに充分なほどには大きくない。このため、パケット・データは、ネットワーク・スイッチ102内で、パケットの一部あるいはパケットの細分割の形態で、1つのFIFOから別のFIFOへ送られる。

20

【0211】

第1のステップ1200において、EPSM210は、信号PKT__AVAILm*の表示により、ポート104、110の一方により受取られる新たなパケットを検出する。次のステップ1202において、パケットの初めの部分即ちヘッダがソース・ポートから検索されて、ハッシュ・リクエスト・ロジック532へ読込まれる。ヘッダは、宛先MACアドレスおよびソースMACアドレスを含んでいる。ハッシュ・リクエスト・ロジック532は、宛先アドレスおよびソース・アドレスとソース・ポート番号を、HASH__DA__SA[15:0]信号中に与え、MCB404へHASH__REQ*信号をアサートする。MCB404は、それに応答して、パケットに対する適切な動作を決定するためのハッシング手順を呼出し、ソース・アドレスおよび宛先アドレスがハッシュされて、このアドレスのいずれかがそれ以前にメモリ212に格納されたかどうかを判定する。MCB404は、HCB402に対して十分な情報が得られる場合に、信号HASH__DONE*をアサートして、パケットに対してとるべき適切な動作を判定する。図52に示されるフロー図は、宛先アドレスおよびソース・アドレスに関する2つの主要部分を含んでおり、これについては後述する。図示した実施例では、宛先アドレスが最初にハッシュされ、ソース・アドレスがその後続くが、これらの手順は同時に実行しても良いし、所望の順番で実行してもよい。

30

40

【0212】

宛先アドレスの場合は、処理はステップ1204へ進み、ハッシング手順が呼出されて宛先アドレスをハッシュする。信号HASH__DONE*に応答して動作がステップ1204からステップ1208へ進んで、ユニキャストとBCパケットの双方に対するスレッシュールド・コンディション(閾値条件)を調べる。ステップ1208において、関連するスレッシュールド・コンディションを新たなパケットが違反するかどうかを判定される。特に、Free SecCnt数がFree SecThreshold数と等しいかあるいはこれより小さければ、パケットをメモリ212に格納するのに十分な余地がないことである。また、Rx SecCntがソース・ポートに対するRx SecThresholdより

50

大きいかあるいはこれに等しければ、ネットワーク・スイッチ102が、パケットをドロップする（捨てる）ことを決定する。BCパケットの場合は、BC__PktThreshold数が、BCパケットの実際数であるBC__PktCnt数に比較されて、BCパケットの最大数が既に受取られたかどうか判定する。ユニキャスト・パケットの場合は、TxSecThreshold数が、宛先ポートに対するTxSecCntに比較される。

【0213】

ステップ1208からステップ1205へ進み、ここでHCB402が、HASH STATUS[1:0]信号から、またスレッシュールド・コンディションのどれかの比較から、パケットをドロップすべきかどうか判定する。このパケットは、先に述べたような様々な他の理由、例えば、ソース・ポートと宛先ポートが等しいなどの理由からドロップされる。パケットがドロップされるべきであれば、動作はステップ1205からステップ1207へ進み、ここでパケットがドロップされるかあるいはバックプレシャ（BC）が加えられる。条件FreeSecThresholdまたはRxSecThresholdが違反され、かつバックプレシャがイネーブル状態にされソース・ポートがハーフ2重モードで動作しているならば、バックプレシャが提供される。さもなければ、パケットがドロップされる。バックプレシャにおいては、EPSM210がHSB206においてバックプレシャ・サイクルを実行して、ソース・ポートに送出側装置に対するジャミング・シーケンスをアサートさせる。ABORT__OUT*信号により示されるように、バックプレシャ指令がソース・ポートにより受入れられなければ、この指令がジャミング・シーケンスのアサートに遅すぎて提供されたことであり、パケットがドロップされる。また、BC__PktThreshold条件が抵触される唯一つのスレッシュールド・コンディションであっても、パケットがドロップされる。ネットワーク・スイッチ102がドロップされるパケットの残部を受取り続けるが、パケットは格納されず、あるいは別のポートへ送られない。動作は、ステップ1207からステップ1214へ進み、ここでMCBコンフィギュレーション・レジスタ448における適切な統計レジスタが、ステップ1207で行われる動作に基いて更新される。当該統計レジスタは、オーバーフロー条件によりパケットがドロップされたかバックプレシャされたかを示す。例えば、ポート当たりの「ドロップされたパケットーバッファなし」カウントがソース・ポートに対して増分されて、パケットがオーバーフロー条件により捨てられるか、あるいは、パケットがバックプレシャされるならば、「バックプレシャされたパケット」カウントが増分されることを示す。

【0214】

パケットがドロップされなければ、動作はステップ1205からステップ1206へ進み、ここで宛先アドレス（DA）がハッシュ・メモリ・セクション902で見出されたかどうか、またパケットがブロードキャストされるべきかどうか判定される。宛先アドレスが認識されず従って宛先ポートが未知であるか、あるいはパケット内のグループ・ビットがセットされるならば、パケットがブロードキャストされる。宛先アドレスが見出されなければ、あるいは、パケットがステップ1206で判定されるようなBCパケットであるならば、パケットがブロードキャストされて動作がステップ1210へ進み、EPSM210のMCB404が、必要に応じて、新たなパケットに対するメモリ212内に別のセクタを割付ける。現在のすなわちその時のセクタがパケットに対して十分な余地を有するならば、新たなセクタは不要である。次いで、動作はステップ1216へ進み、パケットの残りがバースト単位でEPSM210を介してバッファ記憶され、そしてメモリ212へ送られる。ポート設定の如何を問わず、BCパケットがSnFモードで処理され、ここで全パケットが伝送される前にメモリ212に格納される。ステップ1216から、動作がステップ1217へ進んで、パケット・エラーによるパケットの受取り中に信号ABORT__OUT*がアサートされたことを判定する。ポートPORT1～PORT27により、FIFOオーバーラン、ラン・パケット、オーバーサイズ・パケット、パケットが不正FCSを持つこと（フレーム検査シーケンス）の検出、あるいはPLLエラーが検知されたかのような、幾つかのエラー条件が調べられる。パケット・エラーがステップ1217において検出されるならば、動作はステップ1219へ進み、ここでパケットがメモリ2

10

20

30

40

50

12から除去される。

【0215】

パケット・エラーがステップ1217で検出されなければ、動作はステップ1218へ進み、ここでBCパケットのパケット・ヘッダ922におけるブロードキャスト・ポート・ビット・マップBC__Portsが、BCパケットが送られるべきアクティブなポートで更新される。次のポート、即ち、ソース・ポートか、ソース・ポートがCPU230であるならば前送（FORWARDING）状態ではない任意のポートか、あるいは、ソース・ポートがCPU230であるならばディスエーブル状態の任意のポート、および対応するTxPktThresholdより大きいかこれと等しいTxPktCnt数を持つポートを除いて、BCパケットがポート104、110の全てへ送られる。VLANがイネーブル状態にあるならば、ハッシュ・テーブル・エントリ910におけるVLANビット・マップ値もまた調べられ、これが更に、ポートをVLANグループにおけるアクティブ状態の関連ポートに限定する。また、パケットが未知の宛先アドレスによりブロードキャストされるミスBCパケットが、MissBCBitMapレジスタに従って前送される。パケットがいずれのポートにも送られないように、得られたBC__Portのビットマップが全てゼロであるならば、この判定がステップ1205で行われて当該パケットがステップ1207でドロップされ、あるいはパケットはステップ1219でメモリ212から除去されることが判る。

10

【0216】

動作はステップ1218からステップ1220へ進み、得られたBC__Portのビット・マップにおける各ポートに対する送信パケット・チェーンへ、パケットが付加される。特に、パケット・ヘッダ922におけるBC__Portビット・マップで示される各ポートに対するNextTxLinkリンク・アドレスの各々が更新されて、BCパケットを適切なポートの送信パケット・チェーンに挿入する。他の全てのレジスタ、あるいはネットワーク・スイッチ102におけるカウント値および統計数値も、例えばBC__PktCnt数のように同様に然るべく更新される。

20

【0217】

再びステップ1206に戻り、宛先アドレスが見出されたがパケットがBCパケットでなければ、動作はステップ1222へ進み、ここでハッシュ・キャッシュ・テーブル603が更新される。次いで、動作は次のステップ1224へ進み、ここでソース・ポートあるいは宛先ポートのいずれかがSnFモードに対してセットされるかどうか質問される。両方のポートがCTモードにセットされ、等しいポート速度および宛先ポートに対するTBUS設定がソース・ポートに対するTBUS設定に等しいなどの、他のCT条件が満たされるならば、動作はステップ1225へ進み、ここで宛先ポートへの経路が使用中であるかどうか質問される。ステップ1224で決定されるようにSnFモードが指示されるか、あるいはCTモードに指示されるが暫定CTモードが開始されるように宛先ポートがステップ1225で決定されるように使用中であるならば、動作はステップ1226へ進み、ここでEPSM210のMCB404が、必要に応じて、新たなパケットに対してメモリ212内のスペースを割付ける。ステップ1226から、動作はステップ1228へ進み、ここでパケットの残りの部分がEPSM210へ検索され、メモリ212へ送られる。ステップ1217に類似するステップ1229で示されるように、パケットの受取りにパケット・エラーが生じるならば、動作はステップ1219へ進んでメモリ212からこのパケットを除去する。さもなければ、動作は次のステップ1230へ進み、ここでパケットは宛先ポートの送信パケット・チェーンに付加され、適切なリンク・アドレス、カウントおよびチェーンが更新される。

30

40

【0218】

再びステップ1225において、宛先ポートの経路が使用中でなければ、動作はステップ1231へ進み、ここでソース・ポートおよび宛先ポートが、その時のパケットに対する正規のCT動作に関して指定される。正規のCTモードでは、各々の残りのパケット部分がメモリ212へは送られず、その代わり、CTBUF（CTバッファ）528を介し

50

て宛先ポートへバッファ記憶される。パケットのヘッダは、E P S M 2 1 0 の R X F I F O から直接宛先ポートへ送られる。次のステップ 1 2 3 2 は、C T バッファ 5 2 8 へのデータ・パケット部分の受信及び宛先ポートへのパケット部分の転送とを示している。C T 動作の間、次のステップ 1 2 3 3 は、宛先ポートまたは経路が使用中かあるいは利用できないかどうかを質問する。ステップ 1 2 3 3 でのこの質問は、データが主アービタ 5 1 2 により C T バッファ 5 2 8 に受取られる前に行われる。宛先ポートが更なるデータに対してまだ利用可能である間、動作はステップ 1 2 3 4 へループして、全パケットが宛先ポートへ送られたかどうかを判定し、送られなかったならば、再びステップ 1 2 3 2 へ戻り更なるデータを伝送する。全パケットがステップ 1 2 3 4 で判定されたように C T モードで転送された時、このパケットに対する動作が完了する。

10

【0 2 1 9】

ステップ 1 2 3 3 で宛先ポートが使用中または利用できないと判定すると、動作はステップ 1 2 3 5 へ進んで、メモリ 2 1 2 にパケットの残りの部分を受取って、中間パケットの暫定 C T モードを開始する。中間パケット暫定 C T モードでは、パケットの残りの部分がメモリ 2 1 2 にバッファ記憶される。パケットが伝送の途中にあったため、メモリ 2 1 2 へ送られる残りのパケット・データはこのポートに対する送信パケット・チェーンの初めに置かれて、次のステップ 1 2 3 6 で示される適切なパケット順序付けを保証する。正規の C T 動作モードにおけるように、中間パケット暫定 C T モードの間にメモリ 2 1 2 へ供給された各データ部分は、受取り後すぐに宛先ポートへの転送のために利用可能である。

【0 2 2 0】

再びステップ 1 2 0 2 において、動作はソース・アドレスをハッシュするためステップ 1 2 4 0 へ進む。次いで、ステップ 1 2 4 2 へ進み、ソース・アドレスがハッシュ・メモリ・セクション 9 0 2 で見出されたかどうか、かつパケット内のグループ (G R O U P) ビットがセットされたかどうか判定される。ソース・アドレスが見出され、G R O U P ビットがセットされなかったならば、動作はステップ 1 2 4 4 へ進み、ハッシュ・メモリ・セクション 9 0 2 の A G E フィールドが A G E 情報で更新される。例えば、A G E 値はゼロにセットされる。ソース M A C アドレスおよびソース・ポート番号が前のエントリとは対応しないことが判る。このことは、例えば、ネットワークまたはデータ装置が 1 つのポートから他のポートへ移される場合に生じる。この情報は、ステップ 1 2 4 4 において比較され、更新される。

20

30

【0 2 2 1】

ステップ 1 2 4 2 において、ソース・アドレスが見出されず、あるいは G R O U P ビットがセットされたならば、ステップ 1 2 4 6 へ進み、C P U 2 3 0 に対して割込みが生成され、C P U が以降のステップを実行する。次のステップ 1 2 4 8 において、C P U 2 3 0 は、メモリ 2 1 2 のハッシュ・メモリ・セクション 9 0 2 にハッシュ・テーブル・エントリを割り当て、あるいは新たなソース・ポート・アドレスに関するハッシュ・キャッシュ・テーブル 6 0 3 のハッシュ・テーブル・エントリ、あるいはリスト・レーセントリ・ユーズド (最低使用頻度: L R U) セクションを割付ける。次いで、ステップ 1 2 5 0 へ進み、割付けられたハッシュ・エントリにおけるソース M A C アドレス、ソース・ポート番号および A G E 情報等が更新される。

40

【0 2 2 2】

図 5 3 は、メモリ 2 1 2 から 1 つ以上の宛先ポートへデータを送信するためのネットワーク・スイッチ 1 0 2 の一般的動作を示すフロー図である。この送信手順は一般に、以下に述べるように、S n F モードおよび中間パケット暫定 C T 動作モードに適用し、B C パケットに適用する。第 1 のステップ 1 2 6 0 は一般に、先に述べた手順に従って、パケット・データがメモリ 2 1 2 において待ち行列に入れられる (キューされる) ことを表わす。次のステップ 1 2 6 2 へ進み、M C B 4 0 4 が H C B 4 0 2 に対してパケット・データが得られることを示す。中間パケット・データ暫定 C T モードにおいては、この表示は、宛先ポートへ送るためにデータが直ちに得られるので、データの最初のデワード (D W O R D) がメモリ 2 1 2 に格納するため M C B 4 0 4 へ送られると、直ちに与えられる。しか

50

し、S n Fモードの場合は、この表示は、全パケットが送信に先立ち格納されるので、データ・パケットに対するデータの最後のデワードがM C B 4 0 4へ送られた後にのみ与えられる。パケット・データが送信のために利用可能であると、動作はステップ1 2 6 4へ進み、伝送されるパケット・データを受取るために利用可能なバッファ・スペースを宛先ポートが有するかどうか判定される。ステップ1 2 6 4は一般に、先に述べたように、対応する信号B U F _ A V A I L m *に応答するポート1 0 4、1 1 0のそれぞれをポーリングするため、E P S M 2 1 0により行われるポーリング手順を表わす。宛先ポートがパケット・データの受取りに利用可能なバッファ・スペースを持つことを示すまで、動作はステップ1 2 6 4に止まる。

【0 2 2 3】

ステップ1 2 6 4において、送信先ポートすなわち宛先ポートがバッファ・スペースを持つと判定すると、動作はステップ1 2 6 6へ進み、H C B 4 0 2が宛先ポートに対するデータの転送を要求する。そして、ステップ1 2 6 8において、データのバーストがメモリ2 1 2から宛先ポートへ送られる。次のステップ1 2 7 0へ進み、メモリ2 1 2におけるデータの全てが宛先ポートへ送られたかどうか判定される。送られなかったならば、ステップ1 2 6 4へ戻って、宛先ポートがデータの別の転送のため利用可能なより多くのバッファ・スペースを有することになるまで、待機する。最終的には、S n Fモードおよび暫定C Tモードの場合における全データ・パケット、あるいは中間パケット・データ暫定C Tモードの場合における残りのパケット・データが、転送され、これはステップ1 2 7 0で判定される。

【0 2 2 4】

動作はステップ1 2 7 2へ進み、パケットがB Cパケットであるかないか判定される。パケットがB Cパケットであるならば、動作はステップ1 2 7 4へ進んで、全パケットが全てのアクティブ・ポートへ転送されたかどうか判定する。転送されなかったならば、その時のパケットに対しては、動作が完了する。この手順は、パケットが全てのアクティブ・ポートへ転送されるまで、各ポートに対して再び実行される。各B Cパケットに対する各宛先ポートに関して、ステップ1 2 6 4～ステップ1 2 7 0が行われることを、ステップ1 2 7 2と1 2 7 4が表わしていることが判る。このように、全B Cデータ・パケットは、送信のため全てのアクティブな宛先ポートへ送られるまで、メモリ2 1 2に保持される。当該パケットがB Cパケットでなければ、あるいはステップ1 2 7 4で示されるようにB Cパケットに対する全てのアクティブ・ポートへ全パケットが送られた後は、動作はステップ1 2 7 6へ進み、B Cパケットを保持するメモリ2 1 2におけるバッファ・スペースが解放される。特に、パケット・データを保持するセクターは、メモリ2 1 2内のフリー・メモリ・セクタのフリープール・チェーン(F R E E P O O L C H A I N)へ戻される。

【0 2 2 5】

図5 4には、E P S M 2 1 0のハッシュ・ルックアップ動作を示すフロー図が示される。図5 4のフロー図におけるステップは、M C B 4 0 4によって行われる。初期ステップ1 3 0 2は、信号H A S H _ R E Q *のアサートにより示されるハッシュ・リクエストを検出する。H C B 4 0 2は、新たなパケットとしてのパケットのヘッダを識別し、ソース・アドレスおよび宛先アドレス、およびソース・ポート番号を決定し、M C B 4 0 4のハッシュ・コントローラ6 0 2に対する信号H A S H _ D A _ S A [1 5 : 0]を表明する。次に、M C B 4 0 4は、ソース・アドレスおよび宛先M A Cアドレス、およびソース・ポート番号を検索して、ハッシング手順を実施し、これにより、パケットに対する適切な動作を決定する。

【0 2 2 6】

M C B 4 0 4は一般に、各パケットに関して、ソース・ポート番号、ソース・アドレスおよび宛先M A Cアドレスに基づく、4つの動作の1つを行う。特に、ハッシュ・コントローラ6 0 2は、信号H A S H _ S T A T U S [1 : 0]を決定し、これにより、パケットを宛先ポートへ送るようにF O R W A R D _ P K Tをセットし、パケットをドロップして

10

20

30

40

50

無視するようにD R O P _ P K Tをセットし、宛先M A Cアドレスが新規でありかつ未知であってパケットが他の全てのポートにブロードキャストすなわち送信される場合にM I S S _ B Cをセットし、又は、パケットがサブセットの関連ポートに複写されて送信されるべき場合にG R O U P _ B Cをセットする。ステップ1 3 0 2からステップ1 3 0 4へ進んで、以下の式(1)により、パケットを捨てるかどうかを決定する。

【0 2 2 7】

【数1】

$\text{DropPkt} := (\text{SrcState} = \text{DIS}) \text{or} (!\text{FilterHit} \& \text{SrcState} != \text{FWD})$ (1)

但し、S r c S t a t eは、ソース・ポートのスパニング・ツリー状態を識別するものであり、H i l t e r H i tは、ソースM A Cアドレスが予め定めた範囲内にある場合にアサートされるビットであり、間投詞「!」記号は論理的否定を示し、記号「!=」は関数「に不等」を示し、記号「:=」は関数「に等」を示す。各ポートは、H S Bコンフィギュレーション・レジスタ4 4 8に提供され、学習(L R N)、前送(F W D)、ブロック(B L K)、リスニング(L S T)およびディスエーブル状態(D I S)を含む、I E E E仕様8 0 2. 1のスパニング・ツリー関数により決定されるような、5つの状態の1つを持つ。図示した実施例においては、B L K状態とL S T状態は同じものとして処理される。このため、ソース・ポートがディスエーブル状態にされるか、あるいはソースM A Cアドレスが予め定めたフィルタ範囲内になく、かつソース・ポートの状態が前送されなければ、パケットはドロップすなわち捨てられる。

【0 2 2 8】

D r o p P k tがステップ1 3 0 4で死んでであると判定されると、ステップ1 3 0 5へ進み、パケットを無視するかさもなければ捨てるようH C B 4 0 2に命令するように、信号H A S H _ S T A T U S [1 : 0] が0 0 b = D R O P _ P K Tに等しくセットされる。D r o p P k tが偽であるならば、ステップ1 3 0 6へ進み、F i l t e r H i tビットが調べられて、ソースM A Cアドレスが予め定めた範囲内に含まれるかどうかを判定する。この予め定めた範囲は、C P U 2 3 0をソースとし、あるいは宛先とするパケットを識別し、C P U 2 3 0へ送られるブリッジ・プロトコル・ユニット(B P D U)を含む。F i l t e r H i tビットがステップ1 3 0 6で死んでであると判定されると、ステップ1 3 0 8へ進んで、宛先ポート(D s t P r t)を識別する。パケットがC P U 2 3 0からであれば(S r c P r t = C P U)、宛先ポートは、前の動作においてC P U 2 3 0によりセットされる値F l t r P r tに等しくセットされる(D s t P r t := F l t r P r t)。さもなければ、パケットはC P U 2 3 0へ送られる(D s t P r t := P O R T 2 8)。次いで、ステップ1 3 0 8からステップ1 3 1 0へ進んで、以下の式(2)に基づいて、パケットを前送する(F w d P k t)かどうかを判定する。

【0 2 2 9】

【数2】

$\text{FwdPkt} := (\text{DstPrt} != \text{SrcPrt}) \& ((\text{DstState} = \text{FWD}) \text{or} (\text{SrcPrt} = \text{CPU} \& \text{DstState} != \text{DIS}))$ (2)

但し、式(2)において、D s t S t a t eは、宛先ポート(D s t P r t)のスパニング・ツリー状態であり、「&」は論理的A N D演算を示す。このように、宛先ポートおよびソース・ポートが同じでなく、かつ宛先ポートの状態が前送であるならば、あるいはソース・ポートがC P U 2 3 0であり宛先ポートの状態がディスエーブル状態でなければ、パケットは宛先ポートへ送られる。ハッシュ・ルックアップがなくとも、宛先ポートは、C P U 2 3 0であるか、あるいはC P U 2 3 0によりF l t r P r tにより判定されるので、既知である。F l t r P r tが偽であれば、ステップ1 3 0 5へ進んでパケットを捨てる。さもなければ、F l t r P r tが真の場合、ステップ1 3 1 2へ進み、H A S H _ S T A T U S [1 : 0] 信号が1 1 b = F O R W A R D _ P K Tに等しくセットされ、パケットが宛先ポートへ送られるべきことを示す。また、H A S H _ D S T P R T [4 : 0] 信号はD s t P r t宛先ポート番号と関連させられる。

【0 2 3 0】

10

20

30

40

50

ステップ1306において、ソース・アドレスが予め定めた範囲内になく、従ってフィルタされたMACアドレス外であるならば、動作はステップ1314へ進んで、パケットがBCパケットであるか否かを示す、受取ったパケット内のGROUPビットを調べる。GROUPが偽（GROUPビット＝論理値0）であれば、ステップ1316へ進んで、宛先MACアドレス（DA）のハッシュ・ルックアップを行う。MACアドレスは、2つの異なるセット（組）のビットをアドレスから取り、そしてこの2つのセットを一緒にビット単位で論理的に組合わせて比較することにより、ハッシュされる。これにより、先に述べたように、対応する13～16ビットのハッシュ・アドレスを形成する。MACアドレスの任意のビットをハッシング手順の目的のために選定することができる。図55のフロー図に関連して以下に述べる別個のルーチンまたは関数により、実際のルックアップ手順が行われる。

10

【0231】

ステップ1316におけるルックアップ手順が、HITと呼ばれるビットを含む1つ以上の値を必要に応じて返送し、これが宛先アドレスに対するDA_Hitとして、あるいはソース・アドレスに対するSA_Hitとして返送される。HITビットは、ハッシュされたアドレスがハッシュ・メモリ・セクション902に見出されたかどうかを判定する。ステップ1316からステップ1318へ進み、ここでDA_Hit値が調べられてアドレスが見出されたか否かを判定する。このアドレスは、宛先MACアドレスと対応する装置がパケットを前に送信した場合に、メモリ212中に見出される。DA_Hitが真ならば、動作はステップ1310へ進んで、先に述べたようにパケットを前送するかどうかを判定する。ハッシュ・アドレスが見出されずDA_Hitが偽であるならば、ステップ1320へ進み、ここでHASH_STATUS[1:0]信号が10b=MISSED_BCにセットされて、新たなMACアドレスを示す。宛先装置と関連するポート番号が未知であるので、パケットは他の全てのアクティブ・ポート（VLANにより定性化されるポート、および他の論理的ポート）へ、ブロードキャストされて、パケットが適切な宛先装置へ送られることを保証する。最終的には、宛先装置は、ソース・アドレスと同じMACアドレスを含む新たなパケットに答える。この時、ネットワーク・スイッチ102は、MACアドレスをポートとポート番号とに関連付けて、これに対応してハッシュ・メモリ・セクション902を更新する。ステップ1314において、GROUPビットが真（即ち、論理値1）であるならば、動作はステップ1322へ進み、ここでHASH_STATUS[1:0]信号が01b=GROUP_BCにセットされ、パケットが他の全てのポート、あるいはVLAN関数により指定されるポートのグループへブロードキャストされることを示す。

20

30

【0232】

ステップ1305、1312、1320あるいは1322のいずれかから、ステップ1324へ進んで、SrcLookUp値を調べることに、ソースMACアドレスについてハッシュ・メモリ・セクション902を検索するかどうかを判定する。SrcLookUp値は、以下の式（3）に従って決定される。

【0233】

【数3】

40

$$\text{SrcLookUp} := (\text{SrcState} = (\text{LRNorFWD})) \& \text{SrcPrt} \neq \text{CPU} \quad (3)$$

式（3）は、ソース・ポートが学習モードあるいは前送モードにあり、かつ該ソース・ポートがCPU230でない場合は、MACソース・アドレスが探索されることを示している。SrcLookUpがアサートされて真であるとステップ1324で判定されると、動作はステップ1326へ進み、2つの値VLANおよびSecurePortが調べられる。VLANモードのどれかがイネーブル状態にされるならばVLANビットは真であるが、それ以外は偽である。ソース・ポートが確実であればSecurePortが真である、すなわちアサートされ、ここでは新たなアドレスはハッシュ・メモリ・セクション902へは付加されず、未知のソース・アドレスからのパケットが捨てられる。VLANが真でなくポートが確実でなければ、動作はステップ1328へ進み、HASH_DON

50

E*信号がアサートされて、一時的にアサート状態を保つ。この時、信号HASH__STATUSおよびHASH__DSTPRTが、HCB402により捕捉される。

【0234】

ステップ1326において、VLANが真であるか、あるいはSecurePortが真であると判定された場合、あるいはステップ1328が行われた後は、ソース・アドレス・ルックアップの後まで、HASH__DONE*信号のアサートが遅延される。次いでステップ1330へ進み、宛先MACアドレスに関して先に述べたと類似の方法で、ハッシュ・ルックアップがソースMACアドレス(SA)に関して行われる。ステップ1330において、対応する装置に関するハッシュ・アドレスが見出されるならば、値SA__Hitが真に戻される。ステップ1330からステップ1332へ進み、ここで値Src__Hitが調べられる。Src__Hitは、以下の式(4)によりSA__Hitに関連付けられる。

【0235】

【数4】

$$\text{Src_Hit} := \text{SA_Hit} \& (\text{HshPrt} = \text{SrcPort}) \quad (4)$$

式(4)において、ソース・ヒットが生じ(SA__Hitが真)、かつハッシュ・メモリ・セクション902におけるエントリで見出されたポート番号がパケットが受取られた実際のソース・ポート番号と等しければ、Src__Hitは真である。格納されたソース・ポート番号が実際のソース・ポート番号と等しくなければ、以下に述べるように、装置は別のポートへ移されたことであり、ハッシュ・メモリ・セクション902はCPU230により更新される。Src__Hitが真であれば、動作はステップ1334へ進み、VLANが偽ならば、HASH__DONE*信号がアサートされる。次いで、動作はステップ1336へ進み、装置のAGE番号がゼロであるか判定される。AGEがゼロに等しくなければ、AGE番号はステップ1338においてゼロに等しくセットされる。ステップ1336でAGE番号がゼロであると判定された場合、あるいはステップ1338においてゼロにセットされた後、ステップ1340へ進み、VLANビットが再び調べられる。VLANが真であれば、ステップ1342へ進み、ここでハッシュVLANルーチンすなわち手順が調べられて、関連するポートをハッシュ・テーブル・エントリ910における対応するVLANビット・マップ値から決定されたものとして、識別する。ステップ1340でVLANが真でないと判定すると、動作はステップ1344へ進み、既にアサートされていない場合は、HASH__DONE*信号がある期間だけアサートすなわちパルスが発生され、次に否定される。ステップ1344の終了により、この手順の動作が完了する。HASH__DONE*信号の否定信号により、HCB402のハッシュ・ルックアップを終了する。

【0236】

ステップ1332において、Src__Hitが偽ならば、ステップ1350へ進み、LearnDisPrt値を調べることで、ソース・ポートがディスエーブル状態にされたことを学習しているかどうか判定される。もし学習していなければ、ステップ1352へ進み、パケットの新たな情報が適切なレジスタへロードされ、CPU230が割込みされる。CPU230は、これに応答して、ハッシュ・メモリ・セクション902を新たなハッシュ・テーブル・エントリ910で更新する。ソース・ポートがステップ1350でディスエーブル状態にされたことを学習していると判定した場合、あるいはハッシュ・メモリ・セクション902がステップ1352で更新された後は、ステップ1354へ進んで、SecurePortビットを調べる。SecurePortが真ならば、動作はステップ1356へ進み、ここでHASH__STATUS[1:0]信号が00b=DROP__PKTへ変更される。この場合、アドレスが新しく、かつ新アドレスが保全ポートでは許容されないため、新たなパケットがドロップされる。また、必要に応じて、セキュリティ(保全)違反割込みがCPU230に対してアサートされて、セキュリティ違反に回答して適切な処置を行う。ステップ1356からステップ1344へ進む。再びステップ1354において、SecurePortビットが非保全ポートを示す偽であるならば、

10

20

30

40

50

ステップ1340へ進む。ステップ1324において、SrcLookUpが偽であれば、直接ステップ1344へ進む。

【0237】

図55には、ハッシュ・メモリ・セクション902におけるハッシュ・テーブル・エントリ910の全てを探索するためのハッシュ・ルックアップ手順を示すフロー図が示されている。最初のステップ1402において、アドレス値Aがステップ1316または1330から送られる受取られたハッシュ・アドレスに等しくセットされる。動作はステップ1404へ進み、ここで受取られたハッシュ・アドレスと関連する主ハッシュ・エントリ・セクション906内のハッシュ・テーブル・エントリ910が読出される。動作はステップ1406へ進み、VALIDENTRY（エントリ有効）ビットが読出され、新たなパケットのMACアドレスが格納されたMACアドレスと比較される。エントリが有効であり正確な整合がMACアドレス間に生じるならば、動作はステップ1408へ進み、HITビットが真にセットされてハッシュ・ビットを示し、動作は呼出し手順即ちルーチンへ戻る。エントリは有効でないか、あるいはアドレスの整合が起きなかったならば、ステップ1410へ進み、ここでVALIDENTRYビットと、エントリのEOC（チェーン終り）値が調べられる。エントリが有効でないか、あるいはEOCに到達しなければ、動作はHITビットを偽として戻る。さもなければ、ハッシュ・アドレスが、ステップ1412においてハッシュ・エントリ内のリンク・アドレス（バイトF:C）に等しくセットされ、ステップ1404へ戻って、チェーン化されたハッシュ・エントリ・セクション908内の次のチェーン化エントリを試みる。MACアドレス整合による有効なエントリが見出されるまでか、あるいはEOC値に遭遇するまで、動作はステップ1404、1406、1410および1412間をループする。

【0238】

以下のテーブル（1）は、本発明により実現された特定の実施形態におけるCPU230の入出力（I/O）スペース・レジスタを示している。テーブル（1）は、単に例示的に示したものであり、また、該例においては、レジスタが特殊な実施例中か又はそれ以外で実現されるか、若しくは同様なレジスタが異なる呼称で呼ばれている。

【0239】

【表1】

テーブル1: CPU230 I/Oスペース・レジスタ

オフセット(h)	マスク	シャドウェア	アクセス (R/W)	Req_name/Bit_name	説明
0	PCB 406		CPU: R PCB: W MCB: ---- HCB: ----	割り込みソース1 ビット0: MCB_INT 1: MEM_RDY 2: ABORT_PKT 3: STAT_RDY 4-31: リザーブ	(1)
4	PCB 406		CPU: R/W PCB: R MCB: ---- HCB: ----	割り込みマスク1 ビット0: MCB_INT 1: MEM_RDY 2: ABORT_PKT 3: STAT_RDY 4: HASH_MISS 5-31: リザーブ	(2)
8	PCB 406		CPU: R/W PCB: R/W MCB: ---- HCB: ----	パケット情報 RdPkt ビット0: SOP 1: EOP 2-15: リザーブ 16-23: 長さ (EOPに対して) 24-31: リザーブ	(3)
C	PCB 406		CPU: R/W PCB: R/W MCB: ---- HCB: ----	パケット情報 WrPkt ビット0: SOP 1: EOP 2-5: BE (SOPに対して) 6-15: リザーブされる 16-23: 長さ 24-31: リザーブ	(4)

【0240】

【表2】

10

20

30

40

10	PCB 406		CPU: PCB: MCB: HCB: R R/W .-.-. .-.-.	SIMM 存在検出 ビット 0-3: simm1_pd[0..3] 4-7: simm2_pd[0..3] 8-11: simm3_pd[0..3] 12-15: simm4_pd[0..3] 16-31: リザーブ	(5)
14	PCB 406		CPU: PCB: MCB: HCB: R/W W .-.-. .-.-.	ポーリング・ソース (1 & 2) ビット 0: MCB_INT 1: MEM_RDY 2: PKT_AVAIL 3: BUF_AVAIL 4: ABORT_PKT 5: STAT_RDY 6: HASH_MISS 7-31: リザーブ	(6)
18	PCB 406		CPU: PCB: MCB: HCB: R W .-.-. .-.-.	割り込みソース 2 ビット 0: PKT_AVAIL 1: BUF_AVAIL 2-31: リザーブ	(7)
1c	PCB 406		CPU: PCB: MCB: HCB: R/W R .-.-. .-.-.	割り込みマスク 2 ビット 0: PKT_AVAIL 1: BUF_AVAIL 2-31: リザーブ	(8)

【 0 2 4 1 】

【表 3】

10

20

30

40

20	PCB 406		CPU: PCB: MCB: HCB:	R/W R/W --- ---	QCスタティステック情報 ビット 0-1: ポート番号 2-4: QC 番号 5-9: レジスタ番号 10-14: レジスタの数 15-19: レジスタの最大数 20-31: リザーブ	(9)
24	PCB 406		CPU: PCB: MCB: HCB:	R R/W --- ---	総パケット情報 ビット 0-15: パケット長 16-23: ソース・ポート 24-31: 宛先ポート	(10)
28	PCB 406		CPU: PCB: MCB: HCB:	WO R/W --- ---	フラッシュ IFO	(11)
30	PCB 406	MCB 404 HCB 402	CPU: PCB: MCB: HCB:	R/W R R R	EFPM ステップアップ ビット 0: TPI インストール 1: EXP インストール 2: マスタ・スライツ・イネーブル 3-4: QcXferSize[1:0] 5-6: TPIXferSize[1:0] 7: AI_FCS 8: DramWrDis 9: SramWrDis 10-12: Epsm Addr Dcd 13: Clk1Sel 14-21: CPU ポート番号 22-31: リザーブ	(12)

【0242】

【表4】

10

20

30

40

34	PCB 406	HCB 402	CPU: PCB: MCB: HCB: R/W ---- R	ポート・スピード ビット 0: ポート0スピード 1: ポート1スピード : : 27: ポート27スピード 28-31: リザーブ	(13)
38	PCB 406	MCB 404 HCB 402	CPU: PCB: MCB: HCB: R ---- R R	ポート・タイプ ビット 0: ポート0タイプ 1: ポート1タイプ : : 27: ポート27タイプ 28-31: リザーブ	(14)
3C	PCB 406	MCB 404	CPU: PCB: MCB: HCB: R/W R R ----	MEM リクエスト ビット 0-23: Mem アドレス 24: メモリ選択 25: 転送サイズ 26-29: バイト・イネーブル 30: RW 31: ロック済ページ・ヒット	(15)

【0243】

【表5】

10

20

30

40

40	PCB 406	HCB 402	CPU: PCB: MCB: HCB:	R ---- R R	EPISM 訂正 ビット 0-7: 訂正番号 8-31: リザーブ	(16)
54	HCB 402		CPU: PCB: MCB: HCB:	R/W ---- ---- R	HCB 利用セット・アップ ビット 0-7: ポート番号又は総数 8-9: モード 10-31: リザーブ	(17)
58	HCB 402		CPU: PCB: MCB: HCB:	R/W ---- ---- R/W	HCB 利用 ビット 0-31: 平均時間	(18)
5c	HCB 402		CPU: PCB: MCB: HCB:	R/W ---- ---- R	ソース CT_SNF / ポート ビット 0: ポート 0 1: ポート 1 : : 27: ポート 27 28-31: リザーブ	(19)
60	HCB 402		CPU: PCB: MCB: HCB:	R/W ---- ---- R	宛先 CT_SNF / ポート ビット 0: ポート 0 1: ポート 1 : : 27: ポート 27 28-31: リザーブ	(20)

【 0 2 4 4 】

【表 6】

10

20

30

40

64	HCB 402 (High 2 bits of each xfersz)		CPU: PCB: MCB: HCB:	R/W ---- ---- R	XferSize / ポート ビット 0-3: ポート 0 xfersize 4-7: ポート 1 xfersize 8-11: ポート 2 xfersize 12-15: ポート 3 xfersize 16-19: ポート 4 xfersize 20-23: ポート 5 xfersize 24-27: ポート 6 xfersize 28-31: ポート 7 xfersize	(21)
68	HCB 402 (High 2 bits of each xfersz)		CPU: PCB: MCB: HCB:	R/W ---- ---- R	XferSize / ポート ビット 0-3: ポート 8 xfersize 4-7: ポート 9 xfersize 8-11: ポート 10 xfersize 12-15: ポート 11 xfersize 16-19: ポート 12 xfersize 20-23: ポート 13 xfersize 24-27: ポート 14 xfersize 28-31: ポート 15 xfersize	(22)

【 0 2 4 5 】

【表 7】

10

20

30

40

リザーブ

6c	HCB 402 (High 2 bits of each xfersz)		CPU: PCB: MCB: HCB: R/W ---- ---- R	Xfersize / ポート ビット 0-3: ポート 16 xfersize 4-7: ポート 17 xfersize 8-11: ポート 18 xfersize 12-15: ポート 19 xfersize 16-19: ポート 20 xfersize 20-23: ポート 21 xfersize 24-27: ポート 22 xfersize 28-31: ポート 23 xfersize	(23)
70	HCB 402 (High 2 bits of each xfersz)		CPU: PCB: MCB: HCB: R/W ---- ---- R	Xfersize / ポート ビット 0-3: ポート 24 xfersize 4-7: ポート 25 xfersize 8-11: ポート 26 xfersize 12-15: ポート 27 xfersize 16-19: ポート 28 xfersize 20-31: リザーブ	(24)

【 0 2 4 6 】

【表 8】

10

20

30

40

74	HCB 402		CPU: PCB: MCB: HCB:	R/W ---- ---- R	Arb_Mode ビット 0-1: モード値 2-31: リザーブ	(25)
78	HCB 402		CPU: PCB: MCB: HCB:	R/W ---- ---- R	HCB Misc コントロール ビット 0: イネーブル CT FIFO 1: イネーブル Rd Extra WS 2: イネーブル CC Rd/Wr Qc 3: イネーブル CC Rd/Wr Qe 4: 初期的イネーブル AD 5-31: リザーブ	(26)
7c	HCB 402		CPU: PCB: MCB: HCB:	R/W ---- ---- R	ポート・シヤットダウン ビット 0-27: ピット マップ	(27)

【0247】

【表9】

10

20

30

40

80	MCB 404		CPU: PCB: MCB: HCB:	R/W ---- R ----	プログラム・ポート状態 ビット 0-1: 状態値 2-31: リザーブ	(28)
90	MCB 404		CPU: PCB: MCB: HCB:	R/W ---- R ----	ポート状態ビットマップ ビット 0: ポート 0 1: ポート 1 : : 27: ポート 27 28-31: リザーブ	(29)

【0248】

【表10】

10

20

30

40

94	MCB 404		CPU: PCB: MCB: HCB:	R --- R/W ---	ポート状態 # 1 ビット 0-1: Port _0_st[1:0] 2-3: Port _1_st[1:0] 4-5: Port _2_st[1:0] 6-7: Port _3_st[1:0] 8-9: Port _4_st[1:0] 10-11: Port _5_st[1:0] 12-13: Port _6_st[1:0] 14-15: Port _7_st[1:0] 16-17: Port _8_st[1:0] 18-19: Port _9_st[1:0] 20-21: Port _10_st[1:0] 22-23: Port _11_st[1:0] 24-25: Port _12_st[1:0] 26-27: Port _13_st[1:0] 28-29: Port _14_st[1:0] 30-31: Port _15_st[1:0]	(30)
----	------------	--	------------------------------	------------------------	---	------

10

20

30

40

【 0 2 4 9 】

【 表 1 1 】

98	MCB 404		CPU: R PCB: --- MCB: R/W HCB: ----	ポート状態# 2 ビット 0-1: Port _16_st[1:0] 2-3: Port _17_st[1:0] 4-5: Port _18_st[1:0] 6-7: Port _19_st[1:0] 8-9: Port _20_st[1:0] 10-11: Port _21_st[1:0] 12-13: Port _22_st[1:0] 14-15: Port _23_st[1:0] 16-17: Port _24_st[1:0] 18-19: Port _25_st[1:0] 20-21: Port _26_st[1:0] 22-23: Port _27_st[1:0] 24-31: リザーブ	(31)
9C	MCB 404		CPU: R/W PCB: --- MCB: R HCB: ----	宛先ミス ブロードキャスト ビット 0-28: DestMissBCビットマップ 29-31: リザーブ	(32)

【0250】

【表12】

10

20

30

40

a8	MCB 404		CPU: PCB: MCB: HCB:	R/W ---- R/W ----	メモリ・バス・モニタ・コントロール ビット 0-14: モニタ・モード 15: モニタ選択 16-23: モニタ・ポート選択 24-27: フィルタ時間スケール 28: モニタ・クリア 29: カウンタ/フィルタ・ モード 30: Backpress イネーブル 31: アラーム	(33)
ac	MCB 404		CPU: PCB: MCB: HCB:	R/W ---- R ----	メモリ・バス・モニタ・スレッシュホールド ビット 0-7: アラーム・セット・ スレッシュホールド 8-15: アラーム・クリア・ スレッシュホールド 16-19: リザーブ 20-31: ピーク BW	(34)
b0	MCB 404		CPU: PCB: MCB: HCB:	R ---- R/W ----	メモリ・バス利用 ビット 0-31: パーセント利用	(35)
b8	MCB 404		CPU: PCB: MCB: HCB:	R ---- R/W ----	メモリによってドロップしたパケット ビット 0-31: パケットの数	(36)

【0251】

【表13】

10

20

30

40

bc	MCB 404		CPU: PCB: MCB: HCB:	R ---- R/W ----	BCによってドロップしたパケット ビット 0-31: パケットの数	(37)
c0	MCB 404		CPU: PCB: MCB: HCB:	R/W ---- R -----	ハッシュ・テーブルの定義 ビット 0-14: アドレス [16:2] 15-23: アドレス [25:17] 24-25: テーブル・サイズ 26: ロック・ハッシュ・サイクル 27: Vlan グループ BC 28: Vlan ミス BC 29: Vlan ユニキャスト 30-31: リザーブ	(38)
c4	MCB 404		CPU: PCB: MCB: HCB:	R ---- R/W ----	Rx セクタ・カウンタ ビット 0-28: ビットマップ 29-31: リザーブ	(39)
c8	MCB 404		CPU: PCB: MCB: HCB:	R ---- R/W ----	Tx パケット・カウンタ ビット 0-28: ビットマップ 29-31: リザーブ	(40)
cc	MCB 404		CPU: PCB: MCB: HCB:	R ---- R/W ----	ハッシュ・アドレス・ロー ビット 0-31: バイト 0-3	(41)

【 0 2 5 2 】

【表 1 4】

10

20

30

40

d0	MCB 404		CPU: PCB: MCB: HCB:	R ---- R/W ----	ハッシュ・アドレス・ハイ ビット 0-15: バイト 4-5 16-23: ソース・ポート 24: ポート・ミス 25-31: リザーブ	(42)
d4	MCB 404		CPU: PCB: MCB: HCB:	R ---- R/W ----	受信メモリ・セクタによりドロップした パケット ビット 0-31: パケットの数	(43)
d8	MCB 404		CPU: PCB: MCB: HCB:	R ---- R/W ----	送信メモリ・セクタによりドロップした パケット ビット 0-31: パケットの数	(44)
dC	MCB 404		CPU: PCB: MCB: HCB:	R/W ---- R ----	受信オーバーフローによりドロップしたパケット ビット 0-28: ポート・ビットマップ 29-31: リザーブ	(45)
e0	MCB 404		CPU: PCB: MCB: HCB:	R/W ---- R ----	送信オーバーフローによりドロップしたパケット ビット 0-28: ポート・ビットマップ 29-31: リザーブ	(46)
e4	MCB 404		CPU: PCB: MCB: HCB:	R/W ---- R ----	学習ディスプレイエーブル・ポート ビット 0-27: 学習ディスプレイエーブル・ ポート・ビットマップ 28-31: リザーブ	(47)
e8	MCB 404		CPU: PCB: MCB: HCB:	R/W ---- R ----	確実ポート ビット 0-27: 確実ポート・ビットマップ 28-31: リザーブ	(48)

【0253】

【表15】

10

20

30

40

ec	MCB 404		CPU: PCB: MCB: R HCB: ----	R/W ---- R ----	セキユリティ・バイオレーション状態 ビット 0-31: カウント	(49)
f0	MCB 404		CPU: PCB: MCB: R HCB: ----	R/W ---- R ----	セキユリティ・バイオレーション ビット 0-27: ポート・ビットマップ 28-31: リザーブ	(50)
f4	MCB 404		CPU: PCB: MCB: HCB:	R/W ---- R/W ----	メモリ・コントロール ビット 0-1: メモリ・タイプ 2: メモリ・スピード 3: EDO テスト・モード 4: Db1 リンク・モード 5: DisRCPgHits 6: DistxPGHits 7-31: リザーブ	(51)
f8	MCB 404		CPU: PCB: MCB: R HCB: ----	R/W ---- R ----	RAS 選択 ビット 0-31: Rasenx [1:0]	(52)
fc	MCB 404		CPU: PCB: MCB: HCB:	R/W R ---- ----	リフレッシュ・カウンタ ビット 0-9: カウント 10-31: リザーブ	(53)

【0254】

【表16】

10

20

30

40

100	MCB 404 (bit 4-7)		CPU: PCB: MCB: HCB: R/W ---- R	フィルタ・コントロール ビット 0-3: アドレス・イネーブル [3:0] 4-7: マスク・イネーブル [3:0] 8-31: リザーブ	(54)
104	MCB 404		CPU: PCB: MCB: HCB: R/W ---- R	マスク・アドレス・フィルタ・ロー ビット 0-31: バイト 0-3	(55)
108	MCB 404		CPU: PCB: MCB: HCB: R/W ---- R	マスク・アドレス・フィルタ・ハイ ビット 0-15: バイト 4-5 16-31: リザーブ	(56)
10c	MCB 404		CPU: PCB: MCB: HCB: R/W ---- R	アドレス・フィルタ 0・ロー ビット 0-31: バイト 0-3	(57)
110	MCB 404		CPU: PCB: MCB: HCB: R/W ---- R	アドレス・フィルタ 1・ハイ ビット 0-15: バイト 4-5 16-23: 宛先ポート 24-31: フィルタ・マスク 0	(58)
114	MCB 404		CPU: PCB: MCB: HCB: R/W ---- R	アドレス・フィルタ 1・ロー ビット 0-31: バイト 0-3	(59)
118	MCB 404		CPU: PCB: MCB: HCB: R/W ---- R	アドレス・フィルタ 1・ハイ ビット 0-15: バイト 4-5 16-23: 宛先ポート 24-31: フィルタ・マスク 1	(60)

【0255】

【表17】

10

20

30

40

11c	MCB 404		CPU: R/W PCB: ---- MCB: R HCB: ----	アドレス・ファイルタ2・ロー ビット 0-31: バイト 0-3	(61)
120	MCB 404		CPU: R/W PCB: ---- MCB: R HCB: ----	アドレス・ファイルタ2・ハイ ビット 0-15: バイト 4-5 16-23: 宛先ポート 24-31: ファイルタ・マスク2	(62)
124	MCB 404		CPU: R/W PCB: ---- MCB: R HCB: ----	アドレス・ファイルタ3・ロー ビット 0-31: バイト 0-3	(63)
128	MCB 404		CPU: R/W PCB: ---- MCB: R HCB: ----	アドレス・ファイルタ・ハイ ビット 0-15: バイト 4-5 16-23: 宛先ポート 24-31: ファイルタ・マスク3	(64)

【0256】

【表18】

10

20

30

40

12c	MCB 404		CPU: PCB: MCB: HCB:	R ---- R/W ----	MCB 割り込みソース ビット 0: セキュリティ割り込み 1: メモリ・オーバーフローを セット 2: メモリ・オーバーフローを クリア 3: プログードキャスト割り込み をセット 4: プログードキャスト割り込み をクリア 5: 受信割り込み 6: 送信割り込み 7: 失敗した R x パケット 8: BW アラームのセット 0 9: BW アラームのクリア 0 10: BW アラームのセット 1 11: BW アラームのクリア 1 12-31: リザーブ	(65)
-----	------------	--	------------------------------	--------------------------	---	------

【0257】
【表19】

10

20

30

40

130	MCB 404		CPU: PCB: MCB: HCB:	R/W ---- R ----	MCB 割り込みマスク ビット 0: セキュリティ割り込み 1: メモリ・オーバーフローの セット 2: メモリ・オーバーフローの クリア 3: プロードキヤスト割り込み マスクのセット 4: プロードキヤスト割り込み マスクのクリア 5: 受信割り込みマスク 6: 送信割り込みマスク 7: 失敗した Rx パケット 8: BW アラームのセット 0 9: BW アラームのクリア 0 10: BW アラームのセット 1 11: BW アラームのクリア 1 12-31: リザーブ	(66)
-----	------------	--	------------------------------	--------------------------	--	------

10

20

30

40

【0258】

【表20】

134	MCB 404		CPU: PCB: MCB: HCB:	R/W ---- R/W ----	MCB ポーリング・ソース ビット 0: セキュリティ割り込み 1: メモリ・オーバーフローの セット 2: メモリ・オーバーフローの クリア 3: プログラム・キャスト・ポーリ ング・ソースのセット 4: プログラム・キャスト・ポーリ ング・ソースのクリア 5: 受信ポーリング・ソース 6: 送信ポーリング・ソース 7: 失敗した R x パケット 8: BW アラームのセット 0 9: BW アラームのクリア 0 10: BW アラームのセット 1 11: BW アラームのクリア 1 12-31: リザーブ	(67)
138	MCB 404		CPU: PCB: MCB: HCB:	R/W ---- R ----	バックプレッシャ・イネーブル ビット 0-23: リザーブ 24-27: ポート・ビットマップ 28-31: リザーブ	
13c	MCB 404		CPU: PCB: MCB: HCB:	R/W ---- R ----	結合ポートのセット 0 ビット 0-27: ポート・ビットマップ 28-31: リザーブ	
140	MCB 404		CPU: PCB: MCB: HCB:	R/W ---- R ----	結合ポートのセット 1 ビット 0-27: ポート・ビットマップ 28-31: リザーブ	

【0259】

【表21】

10

20

30

40

144	MCB 404		CPU: PCB: MCB: R HCB:	R/W --- R ----	デフォルトVlanビットマップ ビット 0-28: ビットマップ	
148	MCB 404		CPU: PCB: MCB: HCB:	R/W --- --- R	契約ポート ビット 0-7: ポート 8-15: Rx モニタ・ポート番号 16-23: Tx モニタ・ポート番号 24-31: リザーブ	(68)
200-2ff			CPU: PCB: MCB: HCB:	R/W R/W --- ---	クワッド・カスケード0レジスタ	(69)
300-3ff			CPU: PCB: MCB: HCB:	R/W R/W --- ---	クワッド・カスケード1レジスタ	(70)
400-4ff			CPU: PCB: MCB: HCB:	R/W R/W --- ---	クワッド・カスケード2レジスタ	(71)
500-5ff			CPU: PCB: MCB: HCB:	R/W R/W --- ---	クワッド・カスケード3レジスタ	(72)
600-6ff			CPU: PCB: MCB: HCB:	R/W R/W --- ---	クワッド・カスケード4レジスタ	(73)

【0260】

【表22】

10

20

30

40

700-7ff				CPU: PCB: MCB: HCB:	R/W R/W ---- ----	クワッド・カスケード 5 レジスタ	(74)
800-8ff				CPU: PCB: MCB: HCB:	R R/W ---- ----	QC スタティスティック・レジスタ	(75)
900				CPU: PCB: MCB: HCB:	R/W R/W ---- ----	HCB FIFO - BPDU	(76)
a00				CPU: PCB: MCB: HCB:	R/W ---- R/W ----	MCB データ FIFO	(77)
b00-fff						拡張用としてリザーブ	

【0261】

テーブル1の(1)～(77)の説明

(1) CPU 230に対する任意の割り込み(1又は複数)のソース。これらの割り込みはCPU 230が割り込みをアクノレッジ(確認)するときに該CPU 230によってクリアされる

(2) CPU 230に対するマスクされるべき割り込み

(3) このレジスタはCPU 230によって書き込まれる

10

20

30

40

50

- (4) このレジスタはE P S M 2 1 0によって書き込まれる
- (5) このレジスタはシフト・レジスタ・インターフェースを通じてS I M M上の情報を含む
- (6) C P U 2 3 0に対するマスクされている任意の割り込み（1又は複数）のソース
- (7) C P U 2 3 0に対する任意の割り込み（1又は複数）のソース。これらの割り込みはC P U 2 3 0が割り込みをアクノレッジ（確認）するときに該C P U 2 3 0によってクリアされる
- (8) C P U 2 3 0に対するマスクされるべき割り込み
- (9) このレジスタに書き込むC P U 2 3 0は、適当なポートの統計リード（読み出し）を発行するようにQ Cインターフェースに知らせる 10
- (10) このレジスタはE P S M 2 1 0によって書き込まれる
- (11) このレジスタは、書き込まれたときに、F I F O内容をフラッシュ（消去）し、E O Pを受信するまでフラッシュを続ける
- (12) このレジスタは一般的セットアップ・パラメータを保持する
- (13) これはポート速度ビットマップ・レジスタである。ポートに対するビットがリセットされたとき、これは1 0 m h zポートであり、ビットがセットされたとき、これは1 0 0 m h zポートである。即ち、0 = 1 0 m h z、1 = 1 0 0 m h zである。パワーアップ・デフォルトは正しい値を含むべきである
- (14) これはポート・タイプ・ビットマップ・レジスタである。ポートに対するビットがリセットされたとき、これはQ Cポートであり、ビットがセットされたとき、これはT L A Nポートである。即ち、0 = Q C、1 = T L A Nである。パワーアップ・デフォルトは正しい値を含むべきである 20
- (15) これは、C P U 2 3 0からのメモリ転送に対するアドレス及びコントロールを含むレジスタである
- (16) このリード・オンリ・レジスタはE P S M 2 1 0に対する改訂番号（revision number）を供給する
- (17) このレジスタは、H C B 4 0 2使用率（ユーティリゼーション）について観察されるべきポート及びモード・ビットを選択する。その可能なモードは、T X、R X及びその両方である
- (18) H C B 4 0 2ユーティリゼーションは、選択されたポートがバス上にある平均時間である 30
- (19) このレジスタは、どのソース・ポートがC Tを行えるか及びどのものがS n Fを行うことのみできるかを示すための、ポートに対するビットマップである
- (20) このレジスタは、どの宛て先ポートがC Tを行えるか及びどのものがS n Fを行うことのみできるかを示すための、ポートに対するビットマップである
- (21) このレジスタは指定されたポートに対するエクスファースイズ（x f e r s i z e）を含む
- (22) このレジスタは指定されたポートに対するエクスファースイズ（x f e r s i z e）を含む
- (23) このレジスタは指定されたポートに対するエクスファースイズ（x f e r s i z e）を含む 40
- (24) このレジスタは指定されたポートに対するエクスファースイズ（x f e r s i z e）を含む
- (25) このレジスタはアービトレーション（仲裁）・モード値を含む。使用可能なアービトレーション・モードはF C F S、ウエイト（重み付け）、又はラウンド・ロビン（round_robin）である
- (26) H C B 4 0 2のサブセクションに対する種々のコントロール
- (27) ディスエーブルされるべきポートのビットマップ
- (28) このレジスタは、ポート状態ビットマップ・レジスタにおいて示されたポートがどの状態に変更すべきかを教える 50

状態値

条件

0 0 b

ディスエーブルされる

0 1 b

ブロックされる／

聞く

1 0 b

学習する

1 1 b

送る

(29) このレジスタは、どのポートがその状態を変化させるかを示す。このレジスタはプログラム・ポート状態レジスタと組になって、ポート状態レジスタを満たす

(30) 各ポートに対しての2ビットが、以下のように、アービタにポートがどの状態にあるかを教える

状態値

条件

0 0 b

ディスエーブルされる

0 1 b

ブロックされる／

聞く

1 0 b

学習する

1 1 b

送る

(31) 各ポートに対しての2ビットが、以下のように、アービタにポートがどの状態にあるかを教える

状態値

条件

0 0 b

ディスエーブルされる

0 1 b

ブロックされる／

聞く

1 0 b

学習する

1 1 b

送る

(32) 宛て先ミス・ブロードキャスト（放送）・ビットマップ

(33) メモリ・バス214モニタ・コントロールは、メモリ・バス214上で行われる監視（モニタリング）を（それが行われる場合に）セットアップするために用いられる

(34) メモリ・バス214モニタ・スレッシュホールドは、アラームをセットするため及びアラームをクリアするために用いられる

(35) メモリ・バス214ユーティリゼーション・レジスタ

(36) メモリ・スレッシュホールド・カウンタによるメモリ・スペースの欠如によってドロップされる（落とされる）パケットの数。このレジスタはリード（読み出し）されるときにクリアされる

(37) ブロードキャスト・メモリ・スペースの欠如によってドロップされるブロードキャスト・パケットの数

10

20

30

40

50

(38) ハッシュ・テーブルのベースに対するアドレス。ハッシュ・テーブルのサイズはレジスタの定義で説明したものである

(39) 受信セクタ・スレッシュールド・オーバーフローのセット又はクリアの何れかによってCPU230に割り込みを行ったポートのビットマップ

(40) 送信パケット・スレッシュールド・オーバーフローのセット又はクリアの何れかによってCPU230に割り込みを行ったポートのビットマップ

(41) ハッシュ・テーブル内を見たときにミスしたアドレス

(42) 残りのハッシュ・アドレス及びソース・ポート

(43) 受信メモリ・セクタ・オーバーフローによってドロップされたパケットの数。このレジスタは読み出されたときにクリアされる

10

(44) 送信メモリ・セクタ・オーバーフローによってドロップされたパケットの数。このレジスタは読み出されたときにクリアされる

(45) このレジスタは、受信オーバーフローによってパケットをドロップしたポートのビットマップである

(46) このレジスタは、送信オーバーフローによってパケットをドロップしたポートのビットマップである

(47) 学習（ラーニング、learning）ディスエーブル・ポート・ビットマップ

(48) セキュア（機密保護、secure）・ポート・ビットマップ

(49) このレジスタはポート・セキュリティによってドロップされたパケットの合計を含む

20

(50) このレジスタは、セキュリティによってパケットをドロップしたポートのビットマップである

(51) このレジスタはメモリ・タイプ、速度その他を含む

(52) RASがメモリの4Mブロックをイネーブルにする

(53) リフレッシュ・カウンタがメモリ・コントローラに対してリフレッシュ信号を生成する

(54) このレジスタはアドレス・フィルタリング及びアドレスのマスキングをイネーブルにする

(55) このレジスタはアドレス・フィルタリングに対するマスク・ビットを含む

(56) このレジスタはアドレス・フィルタリングに対するマスク・ビットを含む

30

(57) このレジスタはアドレス・フィルタ0のバイト0－3を含む

(58) このレジスタはアドレス・フィルタ0のバイト4－5を含む

(59) このレジスタはアドレス・フィルタ1のバイト0－3を含む

(60) このレジスタはアドレス・フィルタ1のバイト4－5を含む

(61) このレジスタはアドレス・フィルタ2のバイト0－3を含む

(62) このレジスタはアドレス・フィルタ2のバイト4－5を含む

(63) このレジスタはアドレス・フィルタ3のバイト0－3を含む

(64) このレジスタはアドレス・フィルタ3のバイト4－5を含む

(65) このレジスタはMCB404において開始された任意の割り込みのソースを含む

(66) このレジスタはMCB404において開始された任意の割り込みに対するマスキングを含む

40

(67) このレジスタは、マスクされたMCB404において開始された任意の割り込みのソースを含む

(68) このレジスタはプロミスキュアス（無差別、promiscuous）・モードで観察されているポートの値を保持する。また、RXトラフィック及びTXトラフィックが現れるポートを含む

(69) これはカッド・カスケード・レジスタに対するオフセットである。これはQC0に対するものである

(70) これはカッド・カスケード・レジスタに対するオフセットである。これはQC1に対するものである

50

(71) これはカッド・カスケード・レジスタに対するオフセットである。これはQC2に対するものである

(72) これはカッド・カスケード・レジスタに対するオフセットである。これはQC3に対するものである

(73) これはカッド・カスケード・レジスタに対するオフセットである。これはQC4に対するものである

(74) これはカッド・カスケード・レジスタに対するオフセットである。これはQC5に対するものである

(75) これは、カッド・カスケードから読み出されたばかりの統計バッファに対するアドレス・スペースである

(76) これは、パケット・データをHCB402へ送信するため／パケット・データをHCB402から受信するための、FIFOのアドレスである

(77) これは、データをMCB404へ送信するため／データをMCB404から受信するための、FIFOのアドレスである

テーブル(1)のレジスタを明瞭にするため、下記のレジスタ定義を提供する。

【0262】

【表23】

割込み情報

EP SM210からCPU230に対する3つの割込みピン；CPUINTHASHL、CPUINTPKTLおよびCPUINTLがある。CPUINTHASHLは、ハッシュ・ミスが生じた時にのみ代入され、(オフセット'hocにおける)ハッシュ・アドレス・ロー・レジスタを読出すことによりクリアされる。CPUINTPKTLは、パケット・インターフェースFIFOで利用可能なパケットがある時か、あるいはパケット・インターフェースFIFOが更に多くのパケット・データを送るためクリアされるバッファ・スペースを有するならば、代入される。CPUINTLは、4つのあり得るソースに対して代入され；これらソースの1つがMCB404における8つのあり得るソースを指す。割込みソースは、これらソースがマスクされなければ、CPU230を割込みさせる。CPU230が割込みされることなく割込みソースの情報を利用可能にさせるため、ポーリング機構が利用可能である。割込みソースのマスクングが割込みをCPU230からブロックさせるが、情報はいぜんとしてポーリング・ソース・レジスタで利用可能である。例えば、要求される統計数字が利用可能である時にSTAT_RDYマスク・ビットがセットされるならば、割込みは生じないが、CPU230はいぜんとして統計数字がポーリング・レジスタの読出しにより読出す用意があると判定することができる。注：割込みソース・レジスタはこれを読出すことによりクリアされるが、ポーリング・ソース・レジスタはこれをクリアするため書込まれなければならない。

割込みソース1レジスタ：(オフセット='h00)CPU230へ送られるCPUINTL割込みのソース。このレジスタは、EP SM210により更新され、その時割込みがCPU230へ送出される。CPU230がこのレジスタに達すると内容がクリアされる。1ビットにおける1の値は、割込みが生じたことを表示する。デフォルト=32'h0000_0000。

ビット0(W/R) MCB_INTは、割込みがMCB404に生じたことおよび割込みを更に理解するためMCB割込みソース・レジスタが読出される必要があることをCPU230に通知する割込みである。デフォルトは0。

ビット1(W/R) MEM_RDYは、要求されたメモリ・データがバッファ・スペースで利用可能であることをCPU230に通知する割込みである。デフォルトは0。

ビット2(W/R) ABORT_PKTは、ABORT_IN*信号がPCB406へ表明されたことをCPU230に通知する割込みである。デフォルトは0。

ビット3(W/R) STAT_RDYは、要求された統計数字情報がPCB406のバッファ・スペースにおいて用意があることをCPU230に通知する割込みである。デフォルトは0。

10

20

30

40

50

ビット4～31 (RO) 予約 (RESERVED) 常に0として読出す。

割込みソース・レジスタに対する `pcbregs` インターフェース

`McBInt (in)` MCBからの入力、ビット0を判別

`MemRdy (in)` メモリFIFOからの入力、ビット1を判別

`AbortPktInt (in)` HCB402インターフェースからの入力、ビット4を判別

`StatRdyInt (in)` QCインターフェースからの入力、ビット5を判別

`CpuInt__ (out)` 割込みが生じたことを示すCPU230への信号。

割込みマスク1レジスタ (オフセット='h04) CPU230によりマスクされる割込み。任意のビットにおける1の値が、割込みがマスクされることを示す。デフォルト=32'h0000_001f 10

ビット0 (W/R) CPU230に対する`McBInt`割込みをマスク。デフォルトは1

ビット1 (W/R) CPU230に対する`MemRdy`割込みをマスク。デフォルトは1

ビット2 (W/R) CPU230に対する`AbortPktInt`割込みをマスク。デフォルトは1

ビット3 (W/R) CPU230に対する`StatRdyInt`割込みをマスク。デフォルトは1 20

ビット4 (W/R) CPU230に対する`HashMiss`割込みをマスクデフォルトは1

ビット5～31 (RO) 予約。常に0として読出し。

割込みソース2レジスタ (オフセット='h18) CPU230へ送られるCPUINTPKTL割込みのソース。このレジスタはEPSM210により更新され、次に割込みがCPU230へ送られる。CPU230がこのレジスタを読出す時、内容がクリアされる。1ビットにおける1の値は、割込みが生じたことを示す。デフォルト=32'h0000_0000

ビット0 (W/R) `PKT__AVAIL`は、パケット・データがCPU230に対して利用可能であることをCPU230へ通知する割込み。デフォルトは0 30

ビット1 (W/R) `BUF__AVAIL`は、パケット・データを送出するためバッファ・スペースがCPU230に対して利用可能であることをCPU230に通知する割込み。デフォルトは0

ビット2～31 (RO) 予約。常に0として読出し

割込みソース・レジスタに対する `pcbregs` インターフェース

`PktAvailInt (in)` TX FIFOからの入力、ビット2を判別

`BufAvailInt (in)` RX FIFOからの入力、ビット3を判別

`CpuInt__Pkt__ (out)` パケット割込みが生じたことを示すCPU230に対する信号

インターフェース・マスク2レジスタ (オフセット='h1c) CPU230によりマスクされる割込み。任意のビットにおける1の値は、割込みがマスクされることを示す。デフォルト=32'h0000_0003。 40

ビット0 (W/R) CPU230に対する`PktAvailInt`割込みをマスク。デフォルトは1

ビット1 (W/R) CPU230に対する`BufAvailInt`割込みをマスク。デフォルトは1

ビット2～31 (RO) 予約。常に0として読出し

ポーリング・ソース1&2レジスタ (オフセット='h14) このレジスタはマスクされた割込み情報を含み、所望のビットをクリアするため1を書込むCPU230によりクリアされる。このため、CPU230が割込みされる代わりにポーリングすることを可能 50

にする。CPUは代わりにポーリングを欲する任意の割込みソースをマスクしなければならない

ビット0 (W/R) MCB_INTは、割込みがMCB 404に生じたことおよび割込みを更に理解するためMCB割込みソース・レジスタが読出される必要がある。デフォルトは0

ビット1 (W/R) MEM_RDYは要求されたメモリ・データがバッファ・スペースで利用可能であることをCPU 230に通知する割込み。デフォルトは0

ビット2 (W/R) PKT_AVAILは、パケット・データがCPU 230に対して利用可能であることをCPU 230に通知する割込み。デフォルトは0

ビット3 (W/R) BUF_AVAILは、バッファ・スペースがCPU 230がパケット・データを送るために利用可能であることをCPU 230に通知する割込み。デフォルトは0 10

ビット4 (W/R) ABORT_PKTは、ABORT_IN信号がPCB 406へアサートされたことをCPU 230に通知する割込み。デフォルトは0

ビット5 (W/R) STAT_RDYは、要求された統計情報がPCB 406のバッファ・スペースにおいて用意があることCPU 230に通知する割込み。デフォルトは0

ビット6 (W/R) HASH_MISSは、ハッシュ・ミスが生じたことをCPU 230に通知する割込み。

ビット7～31 (RO) 予約。常に0として読出し

ポーリング・ソース・レジスタに対するpcbregsインターフェース 20

McbInt(in) MCBからの入力。ビット0を判別

MemRdy(in) メモリFIFOからの入力。ビット2を判別

PktAvailInt(in) TX FIFOからの入力

ビット2を判別

BufAvailInt(in) RX FIFOからの入力

ビット3を判別

AbortPktInt(in) HCB 402インターフェースからの入力

ビット4を判別

StatRdyInt(in) QCインターフェースからの入力

ビット6を判別

m_HashInt(in) MCB 404からの入力 ビット6を判別。 30

【0263】

【表24】

パケット・データのコンフィギュレーション

パケット転送のため使用される3つのレジスタがあり、1つは受取られたパケットに対し、2つは伝送パケットに対する。受信パケットは、HSB 206からのReadOutPkt信号と関連させられる。送信パケットは、HSB 206からのWriteInPkt信号と関連させられる。注：受信と送信の用語は、HSB 206から参照される。CPU 230は、パケット・データ・バッファをアクセスする前に適切なレジスタをアクセスしなければならない。 40

パケット情報RdPktレジスタ (オフセット='h08) CPU 230により送られるデータのパケットに対する必要な情報。HSB 206から参照される受信パケットデフォルト=32'h0000_0000

ビット0 (W/R) SOP CPU 230からのパケットの初め

1=SOP

ビット1 (W/R) EOP CPU 230からのパケットの終り

1=EOP

ビット2～15 (RO) 予約。常、に0として読出し

ビット16～23 (W/R) EOPがアサートされる時、FIFOにおけるデータ長さ (バイト数)

ビット24～31 (RO) 予約。常に0として読出し
 パケット情報RdPktレジスタに対するpcbrregsインターフェース
 r__Sop(out) HSB206インターフェースに与えられたパケット・インジケータの開始
 r__Eop(out) HSB206インターフェースに与えられたパケット・インジケータの終り
 r__length(out) EOPが表示される時バッファにおけるデータのバイト長
 パケット情報WrPktレジスタ (オフセット='h0c) HSB206により送られるデータのパケットに対する必要情報。HSB206から照会される伝送パケットデフォルト=32'h0000_0000 10

ビット0 (W/R) SOP。HSB206からのパケットの開始
 l=SOP
 ビット1 (W/R) EOP。HSB206からのパケットの終り
 l=EOP
 ビット2～5 (W/R) SOPまたはEOPと関連するDWORDに対するバイト使用可能。通常は、全てのバイトが使用可能化される。l=使用可能化
 ビット6～15 (RO) 予約。常に0として読出し
 ビット16～23 (W/R) FIFOにおけるデータ長さ (バイト数)
 ビット24～31 (RO) 予約。常に0として読出し
 パケット情報WrPktレジスタに対するpcbrregsインターフェース 20

h__SopIn(in) HSB206インターフェースからのSOPインジケータ
 h__EopIn(in) HSB206インターフェースからのEOPインジケータ
 h__ByteValIn(in) HSB206インターフェースからのバイト使用可能
 合計パケットInfo (オフセット='h24) これは、MCB404がパケットをCPU230へ送る前にそのパケットに付加する情報。この値は、CPU向けパケットに対するSOPがある時にセットされる
 デフォルト=32'h0000_0000
 ビット0～15 パケット長
 ビット16～23 (RO) ソース・ポート 30
 ビット24～31 (RO) 宛先ポート
 【0264】
 【表25】
メモリ存在の検出
 SIMM/DIMM存在検出レジスタ (オフセット='h10) システムにおけるSIMMについての情報を保持。この情報は、オンボードのシフト・レジスタからリセットされた僅かに後でロードされる
 ビット0～3 (RO) simm1_pd[0..3]
 ビット4～7 (RO) simm2_pd[0..3]
 ビット8～11 (RO) simm3_pd[0..3] 40
 ビット12～15 (RO) simm4_pd[0..3]
 ビット16～31 (RO) 予約。常に0として読出し
 存在検出レジスタに対するpcbrregsインターフェース
 i__PDSeRIn(in) 存在検出シフト・レジスタからのシリアル入力
 【0265】
 【表26】
クワッドカスケード統計セットアップ
 QC統計Infoレジスタ (オフセット='h20) クワッドカスケード統計レジスタの読出し動作のためのセットアップ情報。CPUは、このレジスタに統計読出しを開始することを書込む。デフォルト=32'h0000_8000 50

ビット0～1 (W/R) ポート番号。これは、その統計数字が読出されるポート番号。
読出すべきポートは、この番号と指定クワッドカスケードにより決定される

ビット2～4 (W/R) Q C 番号。アクセスするクワッドカスケードを指示予約された
組合わせ：3' b 1 1 0 および 3' b 1 1 1

ビット5～9 (W/R) レジスタ番号。これは、指定されたポートに対して読出される
べき第1のレジスタの番号

ビット10～14 (W/R) レジスタ数。これは、読出すべきレジスタ数注：ソフトウ
ェアは、この数を、読出するため利用可能なレジスタ範囲内のレジスタ番号と共に保持する
ため要求される

ビット15～19 (W/R) 最大レジスタ数。これは、クワッドカスケードで利用可能 10
な統計的レジスタの最大数。デフォルト=6'h 1 7

ビット20～31 (W/R) 予約。常に0として読出し
クワッドカスケード統計セットアップ・レジスタに対する p c b r e g s インターフェー
ス

r_Q c S t a t P o r t N o (o u t) 読出された統計に対するポート番号。これは
、0と3間の値。この値は、Q C 数と共に用いられて、スイッチにおけるどのポートが観
察されつつあるかを決定する

r_Q c S t a t Q c N o (o u t) Q c 数。ポート番号と共に用いられる。

r_S t a t R e g N o (o u t) 始動レジスタ番号。これは、読出されるべき最初の
統計レジスタの番号 20

r_N o S t a t R e g s (o u t) 読出すべき統計レジスタ数

r_M a x r e g s (o u t) 存在する統計レジスタの最大数。これは、保持さ
れる統計数字が変更されるならば将来の使用のために特に利用可能。

【0 2 6 6】

【表 2 7】

E P S M 2 1 0 のセットアップ

E P S M セットアップ・レジスタ (オフセット='h 3 0) E P S M 2 1 0 に対する汎
用セットアップ・パラメータ。デフォルト=3 2'h 0 0 0 7 _ 1 0 0 0 または 3 2'h 0
0 0 7 _ 3 0 0 0、c k 1 l s e l 入力に依存

ビット0 (W/R) T P I インストール。1=T P I 2 2 0 インストールデフォルト= 30
0。このビットは、マスタ・スイッチ使用可能 (ビット2) が否定される時にのみ、書込
まれる

ビット1 (W/R) E X P インストール。1=拡張インストール。デフォルト=0。こ
のビットは、マスタ・スイッチ使用可能 (ビット2) が否定される時にのみ、書込まれる

ビット2 (W/R) マスタ・スイッチ使用可能。1=パケット・トラフィック使用可能
。デフォルト=0

ビット3～4 (W/R) Q c X f e r S i z e [1:0] これらビットは、マスタ・スイ
ッチ使用可能 (ビット2) が否定される時にのみ、書込まれる

0 0=H S B 2 0 6 における 1 6 バイト転送サイズ

0 1=H S B 2 0 6 における 3 2 バイト転送サイズ 40

1 0=H S B 2 0 6 における 6 4 バイト転送サイズ

1 1=無効組合わせ

ビット5～6 (W/R) T P I X f e r S i z e [1:0] これらのビットはマスタ・ス
イッチ使用可能 (ビット2) が否定される時にのみ、書込まれる

0 0=H S B 2 0 6 における 1 6 バイト転送サイズ

0 1=H S B 2 0 6 における 6 4 バイト転送サイズ

1 0=H S B 2 0 6 における 1 2 8 バイト転送サイズ

1 1=H S B 2 0 6 における 2 5 6 バイト転送サイズ

ビット7 (W/R) A I F C S。このビットは、クワッドカスケードが F C S ビット
を自動挿入することを可能にするため使用される。これは、C P U 2 3 0 からのパケット 50

に対してのみ用いられる

ビット8 (W/R) `DramWrDis`。これは、セットされた時、CPU 230からDRAMへの書込みを使用不能状態にする。デフォルト=0

ビット9 (W/R) `SramWrDis`。これは、セットされた時、CPU 230から内部SRAMへの書込みを使用不能状態にする

ビット10~12 (W/R) `EP SM210` アドレス・デコード。これらビットは、EP SM210のレジスタ・スペースとメモリ・インターフェースをデコードするため用いられる

ビット13 (RO) `clk1sel`

1 = CLK 2周波数は1X CLK 1周波数である

10

0 = CLK 2周波数は2X CLK 1周波数である

ビット14~21 (RO) CPUポート番号。CPU 230のポート番号を指示。デフォルト=8'h1c

ビット22~31 (RO) 予約。常に0として読出し

EP SMセットアップ・レジスタに対する`pc b r e g s`インターフェース

`clk1sel(in)` `clk11`および`clk2`が同じレートであるかどうかを判別するためピンからの入力

`r_DramWrDis(out)` CPU 230インターフェースに、DRAMへの書込みが使用不能化されることを知らせる

`r_SramWrDis(out)` CPU 230インターフェースに、内部SRAMへの書込みが使用不能化されることを知らせる

20

`r_EP SMAdrDcd(out)` この3ビット数は、CPU 230バスにおけるアドレス・ビット31:29と比較される。EP SMセットアップ・レジスタに対する`h c b r e g s`インターフェース

`r_Mst rSwEn(out)` スイッチがパケット通信量に対して使用可能化されることをアービタなどに対して通知

`r_Tp i I n s t (out)`

`r_Exp I n s t (out)`

`r_NonULBCMode[1:0](out)`

`r_ULBCMode[1:0](out)`

30

`r_A I F C S (out)`

EP SMセットアップ・レジスタに対する`m c b r e g s`インターフェース

`r_DramWrDis(out)` DRAM書込みのCPU要求を使用不能

`r_SramWrDis(out)` 内部SRAM書込みのCPU要求を使用不能

EP SM改訂レジスタ (オフセット='h40) EP SM210の改訂番号

ビット0~7 (RO) EP SM210の改訂番号

ビット8~31 (RO) リザーブ(予約)。常に0として読出し

EP SM改訂レジスタに対する`pc b r e g s`インターフェースなし。

【0267】

40

【表28】

ポート・セットアップ

ポート速度レジスタ (オフセット='h34) 各ポートの速度を含むビット・マップ。

1 = 100MHz; 0 = 10MHz。デフォルト=32'h0f00_0000

ビット0 (W/R) ポート0の速度

ビット1 (W/R) ポート1の速度

:

:

ビット27 (W/R) ポート27の速度

ビット28~31 (RO) 留保済み。常に0として読出し

50

ポート速度レジスタに対する `h c b r e g s` インターフェース

`r_P o r t S p d [ 2 7 : 0 ] ( o u t )` H C B 4 0 2 ブロックに対するポート速度
ビット・マップ

ポート・タイプ・レジスタ (オフセット = ' h 3 8) 各ポートのタイプを含むビット・
マップ。1 = T L A N ; 0 = カッドカスケード。デフォルト = 3 2 ' h 0 f 0 0 _ 0 0 0
0

ビット 0 (W / R) ポート 0 タイプ

ビット 1 (W / R) ポート 1 タイプ

:

:

ビット 2 7 (W / R) ポート 2 7 タイプ

ビット 2 8 ~ 3 1 (W / R) 予約。常に 0 として読出し

ポート・タイプ・レジスタに対する `m c b r e g s & h c b r e g s` インターフェース
`r_P o r t T y p e [ 2 7 : 0 ] ( o u t )` H C B 4 0 2 および M C B 4 0 4 に対す
るポート・タイプ・ビット・マップ

C P U メモリ要求

C P U 2 3 0 によるメモリ要求は、2 つの方法で行うことができる。下記のレジスタが両
方法で用いられる ; C P U 2 3 0 は、初期レジスタ / F I F O メモリ要求法を用いる時に
、レジスタを直接アクセスするのみ。

メモリ要求レジスタ (オフセット = ' h 3 c) C P U は、このレジスタにメモリ読出し
または書込みを要求するよう書込む。この要求された機構は、外部 D R A M または内部 S
R A M のいずれかのアクセスのため用いられる。

ビット 0 ~ 2 3 (W / R) 転送の開始アドレス [2 5 : 2] 。 S R A M アクセスのため
には、ビット 2 3 - 8 が留保されるビット 7 : 0 が 2 5 6 の 2 4 ビット・ワードをアドレ
ス指定する

ビット 2 4 (W / R) 0 = 外部 D R A M アクセス (即ち、パケット & ハッシュ・メ
モリ)

1 = 内部 S R A M アクセス (即ち、パケット制御レジスタ)

ビット 2 5 (W / R) 転送長

0 = 1 転送 (4 バイト)

1 = 4 転送 (1 6 バイト)

注 : 開始アドレス & 転送長さは、転送が 2 K ページ境界に跨がるようにセットされるべき
ではない。これを保証する 1 つの方法は、全てのデータ構造 (ハッシュ・エントリの如き
) が 1 6 バイト整合されることを確認することである

ビット 2 6 ~ 2 9 (W / R) バイト使用可能 [3 : 0] 。 (1 = 表明) 部分ワード書
込みに有効。また、C A S を含まない読出しを行うようセットされた E D O テスト・モー
ドと共に使用される。1 より大きい転送長の書込みのため、B y t e E n a b l e s は 1
1 1 1 でなければならない。これらは、E D O テスト・モードがセットされなければ、読
出しは問題外 (d o n ' t c a r e) である。

ビット 3 0 (W / R) 書込み / 読出し。0 = 読出し、1 = 書込み

ビット 3 1 (W / R) ロックされたページ・ヒット。別の C P U 要求が同じメモリ・ペ
ージ内に続くことを示す。D R A M メモリ・アービタはメモリ・システムが別の要求をす
ることを許容せず、R A S はその時のサイクル後に表明されたままとなる。E D O テスト
・モードのみににおいて用いられる。リフレッシュを含む他の要求側は、セットされる間は
メモリ・アクセスを行わない。S R A M がロックされる間パケット・メモリ通信量入力が
停止するので、S R A M アクセスにおいては決して使用すべきでない (ハードウェアのデ
バギングを除く)。

メモリ要求レジスタに対する `m c b r e g s` インターフェース

`C p u A d r [ 2 5 : 2 ] ( o u t )` 開始アドレス `m e m c t l` および `m c b s r a m`
モジュールをパス

10

20

30

40

50

CpuBE [3:0] (out) memctlおよびmcbsramモジュールへByte Enablesをパス

CpuLn [1:0] (out) memctlおよびmcbsramへ転送長さをパス (00 1のIn=1ならば、00; In=4ならば、11)

CpuMemSel (out) 外部DRAM (0) および内部SRAM (1) データ間のmux制御

CpuWr (out) 書込み/読出しビット=1ならば、memctlおよびmcbsramモジュールへアサート

CpuPgHit (out) ロック・ページ・ヒット・ビット=1ならば、memctlおよびmcbsramモジュールへアサート

CpuReq (out) メモリ要求レジスタが書込まれメモリ選択=0である時、memctlモジュールへアサートCpuAckがアサートされるまで、アサートされたままでなければならない

CpuAck (in) CpuReqが受入れられる時、memctlモジュールからmcbrgsへアサートされる

CpuInternalReq (out) メモリ要求レジスタが書込まれ、メモリ選択=1である時、mcbsramモジュールへアサート。CpuInternalAckがアサートされるまで、アサートされたままでなければならない

CpuInternalAck (in) CpuInternalReqが受信される時、mcbsramモジュールからmcbrgsへアサートされる

注：以下のシーケンスをEDOメモリに対するテストに用いるべきである：

1. EDOテスト・モード・ビットをメモリ制御レジスタにセット
2. DWORDを0000hでテスト中のバンクに書込む
3. ロックド・ページ・ヒット・セットおよびバイト使用可能=1111bを持つ同じDWORDを読み出す。その後、FPM DRAMがMDをフロートさせる間EDO DRAMがMDをローに保持し、約100ns後にMD[0]におけるプルアップ・レジスタがこの線をハイに引上げる
4. ロックド・ページ・ヒット・ビットがクリアされ、バイト使用可能=0000bによりDWORDを再び読み出す。これは、CASがアサートされない読出しである。MD[0]は、EDO DRAMに対してロー、FPMに対してハイとなる
5. インストールされたメモリの各バンクごとにステップ1～4を繰り返す。全てのバンクがEDO DRAMを含むだけで、メモリ・タイプがEDO DRAMへセットされる
6. EDOテスト・モード・ビットをクリアして、メモリ・タイプをセットする。EDOテスト・モードをセットしたままにしてはならない。

【0268】

【表29】

混雑ポート

混雑ポート・レジスタ (オフセット='h148) ポートが混雑モードで観察される制御がレジスタに含まれる。デフォルト=32'h0000_0000。このレジスタは、マスタ・スイッチ使用可能 (EPSMセットアップ・レジスタ) が否定される時にのみ書込まれる

ビット0～7 (W/R) 混雑モードで観察されるポート番号

ビット8～15 (W/R) 受取られつつあるデータが現れるポート

ビット16～23 (W/R) 観察されるポートへ送られるデータが現れるポート

ビット24～31 (W/R) 予約。常に0として読出し。

【0269】

【表30】

高速バス・モニタ

HSB使用セットアップ・レジスタ (オフセット='h54) どのポートがHSB206の使用のためのモニターとなるか制御

10

20

30

40

50

デフォルト = 32'h 0000_0000

ビット0～7 (W/R) ポート番号または合計

ビット8～9 (W/R) モード

ビット10～31 (RO) 予約。常に0として読出し

HSB使用レジスタ (オフセット = 'h 58) HSB206の使用は、選択されたポートがHSB206にある平均時間である。デフォルト = 32'h 0000_0000

ビット0～31 (RO) 選択された平均時間ポートはHSB206にある。

【0270】

【表31】

クワッドスルー／ストアN前送情報

10

ソースCT__SNFレジスタ (オフセット = 'h 5c) ソース・ポートのCT/SnF状態を含むビット・マップ。0 = CT; 1 = SNF

デフォルト = 32'h 0000_0000

ビット0 (W/R) ポート0ソースCT__SNF

ビット1 (W/R) ポート1ソースCT__SNF

:

:

ビット27 (W/R) ポート27ソースCT__SNF

ビット28～31 (W/R) 予約。常に0として読出し

ソースCT__SNFレジスタに対するhcbregsインターフェース

20

TblSrcPrt(in) その時のパケット・ソース・ポート。8ビット入力

r_RxPortCtSnf(out) TblSrcPrtに対するCT__SNF状態1ビット出力

宛先CT__SNFレジスタ (オフセット = 'h 60) 宛先ポートのCT/SnF状態を含むビット・マップ。0 = CT; 1 = SNF。デフォルト = 32'h 0000_0000

ビット0 (W/R) ポート0宛先CT__SNF

ビット1 (W/R) ポート1宛先CT__SNF

:

:

ビット27 (W/R) ポート27宛先CT__SNF

30

ビット28～31 (RO) 予約。常に0として読出し

ソースCT__SNFレジスタに対するhcbregsインターフェース

TblDstPrt(in) その時のパケット宛先ポート。8ビット入力

r_TxPortCtSnf(out) TblDstPrtに対するCT__SNF状態。1ビット出力。

【0271】

【表32】

調停情報

調停モード・レジスタ (オフセット = 'h 74) 調停モード値を含む。デフォルト = 32'h 0000_0000。このレジスタは、マスタ・スイッチ使用可能 (EP SMセッ
トアップ・レジスタ) が否定される時にのみ書込まれる

40

ビット0～1 (W/R) 調停 (アービタレーション) モード

2'b 00: 先入れ先サージ調停モード

2'b 01: 重み付け優先調停モード

2'b 10: ラウンド・ロビン調停モード

2'b 11: これも先入れ先サージ・モード

ビット2～31 (RO) 予約。常に0として読出し

調停モード・レジスタに対するhcbregsインターフェース

r_ArbMode(out) HCB402における調停モジュールで必要である先に示した2ビット値

50

調停重み付けレジスタ#1 (オフセット='h64) 重み付け優先調停モードに対する
ポート0～7の重み

ビット0～3 (W/R) 重み付け優先モードに対するポート0調停重み

ビット4～7 (W/R) 重み付け優先モードに対するポート1調停重み

ビット8～11 (W/R) 重み付け優先モードに対するポート2調停重み

ビット12～15 (W/R) 重み付け優先モードに対するポート3調停重み

ビット16～19 (W/R) 重み付け優先モードに対するポート4調停重み

ビット20～23 (W/R) 重み付け優先モードに対するポート5調停重み

ビット24～27 (W/R) 重み付け優先モードに対するポート6調停重み

ビット28～31 (W/R) 重み付け優先モードに対するポート7調停重み

10

調停重みレジスタ#1に対するhcbregsインターフェース

r__ArbWt0(out) これら4ビットが、重み付け調停モードにおけるポート0
に対する重み付けのためHCB402により使用される

r__ArbWt1(out) これら4ビットが、重み付け調停モードにおけるポート1
に対する重み付けのためHCB402により使用される

r__ArbWt2(out) これら4ビットが、重み付け調停モードにおけるポート2
に対する重み付けのためHCB402により使用される

r__ArbWt3(out) これら4ビットが、重み付け調停モードにおけるポート3
に対する重み付けのためHCB402により使用される

r__ArbWt4(out) これら4ビットが、重み付け調停モードにおけるポート4
に対する重み付けのためHCB402により使用される

20

r__ArbWt5(out) これら4ビットが、重み付け調停モードにおけるポート5
に対する重み付けのためHCB402により使用される

r__ArbWt6(out) これら4ビットが、重み付け調停モードにおけるポート6
に対する重み付けのためHCB402により使用される

r__ArbWt7(out) これら4ビットが、重み付け調停モードにおけるポート7
に対する重み付けのためHCB402により使用される

調停重みレジスタ#2 (オフセット='h68) 重み付けされた優先調停モード
のためのポート8～15に対する重み

ビット0～3 (W/R) 重み付け優先モードのためのポート8調停重み

30

ビット4～7 (W/R) 重み付け優先モードのためのポート9調停重み

ビット8～11 (W/R) 重み付け優先モードのためのポート10調停重み

ビット12～15 (W/R) 重み付け優先モードのためのポート11調停重み

ビット16～19 (W/R) 重み付け優先モードのためのポート12調停重み

ビット20～23 (W/R) 重み付け優先モードのためのポート13調停重み

ビット24～27 (W/R) 重み付け優先モードのためのポート14調停重み

ビット28～31 (W/R) 重み付け優先モードのためのポート15調停重み

調停重みレジスタ#2に対するhcbregsインターフェース

r__ArbWt8(out) これら4ビットが、重み付け調停モードにおけるポート8
に対する重み付けのためHCB402により使用される

40

r__ArbWt9(out) これら4ビットが、重み付け調停モードにおけるポート9
に対する重み付けのためHCB402により使用される

r__ArbWt10(out) これら4ビットが、重み付け調停モードにおけるポート10
に対する重み付けのためHCB402により使用される

r__ArbWt11(out) これら4ビットが、重み付け調停モードにおけるポート11
に対する重み付けのためHCB402により使用される

r__ArbWt12(out) これら4ビットが、重み付け調停モードにおけるポート12
に対する重み付けのためHCB402により使用される

r__ArbWt13(out) これら4ビットが、重み付け調停モードにおけるポート13
に対する重み付けのためHCB402により使用される

50

r__ArbWt14(out) これら4ビットが、重み付け調停モードにおけるポート14に対する重み付けのためHCB402により使用される

r__ArbWt15(out) これら4ビットが、重み付け調停モードにおけるポート15に対する重み付けのためHCB402により使用される

調停重みレジスタ#3 (オフセット='h6c) 重み付け優先モードに対するポート16~23の重み

ビット0~3(W/R) 重み付け優先モードに対するポート16調停重み

ビット4~7(W/R) 重み付け優先モードに対するポート17調停重み

ビット8~11(W/R) 重み付け優先モードに対するポート18調停重み

ビット12~15(W/R) 重み付け優先モードに対するポート19調停重み

10

ビット16~19(W/R) 重み付け優先モードに対するポート20調停重み

ビット20~23(W/R) 重み付け優先モードに対するポート21調停重み

ビット24~27(W/R) 重み付け優先モードに対するポート22調停重み

ビット28~31(W/R) 重み付け優先モードに対するポート23調停重み

調停重みレジスタ#3に対するhcbregsインターフェース

r__ArbWt16(out) これら4ビットが、重み付け調停モードにおけるポート16に対する重み付けのためHCB402により使用される

r__ArbWt17(out) これら4ビットが、重み付け調停モードにおけるポート17に対する重み付けのためHCB402により使用される

r__ArbWt18(out) これら4ビットが、重み付け調停モードにおけるポート18に対する重み付けのためHCB402により使用される

20

r__ArbWt19(out) これら4ビットが、重み付け調停モードにおけるポート19に対する重み付けのためHCB402により使用される

r__ArbWt20(out) これら4ビットが、重み付け調停モードにおけるポート20に対する重み付けのためHCB402により使用される

r__ArbWt21(out) これら4ビットが、重み付け調停モードにおけるポート21に対する重み付けのためHCB402により使用される

r__ArbWt22(out) これら4ビットが、重み付け調停モードにおけるポート22に対する重み付けのためHCB402により使用される

r__ArbWt23(out) これら4ビットが、重み付け調停モードにおけるポート23に対する重み付けのためHCB402により使用される

30

調停重みレジスタ#4 (オフセット='h70) 重み付け優先モードに対するポート16~23の重み

ビット0~3(W/R) 重み付け優先モードに対するポート24調停重み

ビット4~7(W/R) 重み付け優先モードに対するポート25調停重み

ビット8~11(W/R) 重み付け優先モードに対するポート26調停重み

ビット12~15(W/R) 重み付け優先モードに対するポート27調停重み

ビット16~19(W/R) 重み付け優先モードに対するポート28調停重み

ビット20~31(RO) 予約。常に0として読出し

調停重みレジスタ#4に対するhcbregsインターフェース

40

r__ArbWt24(out) これら4ビットが、重み付け調停モードにおけるポート24に対する重み付けのためHCB402により使用される

r__ArbWt25(out) これら4ビットが、重み付け調停モードにおけるポート25に対する重み付けのためHCB402により使用される

r__ArbWt26(out) これら4ビットが、重み付け調停モードにおけるポート26に対する重み付けのためHCB402により使用される

r__ArbWt27(out) これら4ビットが、重み付け調停モードにおけるポート27に対する重み付けのためHCB402により使用される

r__ArbWt28(out) これら4ビットが、重み付け調停モードにおけるポート28に対する重み付けのためHCB402により使用される。

50

【0272】

【表33】

HCB402混雑制御

HCB混雑制御 (オフセット='h78) HCB402に対する混雑制御

デフォルト=32'h0000_0000

ビット0 (W/R) CT FIFO使用可能。1=CT FIFO使用可能

ビット1 (W/R) 読出し使用可能特別待機状態。1=待機状態使用可能

ビット2 (W/R) クワッドカスケードに対する同時読出しおよび書込みの使用可能

ビット3 (W/R) QE110に対する同時読出しおよび書込みの使用可能

ビット4 (W/R) 早期アドレス使用可能

ビット5～31 (RO) 予約。常に0として読出し。

10

【0273】

【表34】

ポート遮断

ポート遮断 (オフセット='h7c) ポートが遮断されるビット・マップ

デフォルト=32'h0000_0000

ビット0～27 (W/R) ポート0ないし27に対するビット・マップ

1=ポートが遮断。#

ビット28～31 (RO) 予約。常に0として読出し。

20

【0274】

【表35】

ポート状態セットアップ

1つのポートの状態をセットアップまたは変更するために、2つのレジスタが書込まれなければならない。書込む第1のレジスタは、変更されるポートのビット・マップを含むポート状態ビット・マップ・レジスタである。書込む第2のレジスタは、状態値を含み、2つのポート状態レジスタのプログラミングを開始するプログラム・ポート状態レジスタである。CPUのポート状態は、常に前送中であり、決して変更できない。

ポート状態ビット・マップ・レジスタ (オフセット='h90) その状態が変化するポートのビット・マップ。1=当該ポート状態をプログラム・ポート状態レジスタにおける値へ変更

30

デフォルト=32'h0000_0000

ビット0 (W/R) ポート0。このビットの設定がポート0の状態の変更を使用可能にする

ビット1 (W/R) ポート1。このビットの設定がポート1の状態の変更を使用可能にする

:

:

ビット27 (W/R) ポート27。このビットの設定がポート27の状態の変更を使用可能にする

ビット29～31 (RO) 予約。常に0として読出し

40

プログラム・ポート状態レジスタ (オフセット='h80) ポート状態。CPUが、ポート状態レジスタのプログラミングを開始する当該レジスタに書込む。当該ポート状態ビット・マップ・レジスタは、「最初に書込まねばならない」

デフォルト=32'h0000_0000

ビット0～1 (W/R) 状態値。この値は、オフセット30におけるビット・マップに示されるポートに置かれる

状態値 条件

00b 使用不能

01b ブロック／リスニング

10b 学習

50

1 1 b 前送

ビット 2 ～ 3 1 (R O) 予約。常に 0 として読出し

ポート状態 # 1 レジスタ (オフセット = 'h 9 4) ポート 0 ないし 1 5 の状態プログラム・ポート状態レジスタおよびポート状態ビット・マップ・レジスタによりプログラムされる

デフォルト = 3 2 'h 0 0 0 0 _ 0 0 0 0

状態値 条件

0 0 b 使用不能

0 1 b ブロック / リスニング

1 0 b 学習

1 1 b 前送

ビット 0 ～ 1 (R O) P o r t _ 0 _ s t [1 : 0]

ビット 2 ～ 3 (R O) P o r t _ 1 _ s t [1 : 0]

ビット 4 ～ 5 (R O) P o r t _ 2 _ s t [1 : 0]

ビット 6 ～ 7 (R O) P o r t _ 3 _ s t [1 : 0]

ビット 8 ～ 9 (R O) P o r t _ 4 _ s t [1 : 0]

ビット 1 0 ～ 1 1 (R O) P o r t _ 5 _ s t [1 : 0]

ビット 1 2 ～ 1 3 (R O) P o r t _ 6 _ s t [1 : 0]

ビット 1 4 ～ 1 5 (R O) P o r t _ 7 _ s t [1 : 0]

ビット 1 6 ～ 1 7 (R O) P o r t _ 8 _ s t [1 : 0]

ビット 1 8 ～ 1 9 (R O) P o r t _ 9 _ s t [1 : 0]

ビット 2 0 ～ 2 1 (R O) P o r t _ 1 0 _ s t [1 : 0]

ビット 2 2 ～ 2 3 (R O) P o r t _ 1 1 _ s t [1 : 0]

ビット 2 4 ～ 2 5 (R O) P o r t _ 1 2 _ s t [1 : 0]

ビット 2 6 ～ 2 7 (R O) P o r t _ 1 3 _ s t [1 : 0]

ビット 2 8 ～ 2 9 (R O) P o r t _ 1 4 _ s t [1 : 0]

ビット 3 0 ～ 3 1 (R O) P o r t _ 1 5 _ s t [1 : 0]

ポート状態 # 2 レジスタ (オフセット = 'h 9 8) ポート 1 6 ないし 2 8 の状態。プログラム・ポート状態レジスタおよびポート状態ビット・マップ・レジスタによりプログラムされる。デフォルト = 3 2 'h 0 3 0 0 _ 0 0 0 0

状態値 条件

0 0 b 使用不能

0 1 b ブロック / リスニング

1 0 b 学習

1 1 b 前送

ビット 0 ～ 1 (R O) P o r t _ 1 6 _ s t [1 : 0]

ビット 2 ～ 3 (R O) P o r t _ 1 7 _ s t [1 : 0]

ビット 4 ～ 5 (R O) P o r t _ 1 8 _ s t [1 : 0]

ビット 6 ～ 7 (R O) P o r t _ 1 9 _ s t [1 : 0]

ビット 8 ～ 9 (R O) P o r t _ 2 0 _ s t [1 : 0]

ビット 1 0 ～ 1 1 (R O) P o r t _ 2 1 _ s t [1 : 0]

ビット 1 2 ～ 1 3 (R O) P o r t _ 2 2 _ s t [1 : 0]

ビット 1 4 ～ 1 5 (R O) P o r t _ 2 3 _ s t [1 : 0]

ビット 1 6 ～ 1 7 (R O) P o r t _ 2 4 _ s t [1 : 0]

ビット 1 8 ～ 1 9 (R O) P o r t _ 2 5 _ s t [1 : 0]

ビット 2 0 ～ 2 1 (R O) P o r t _ 2 6 _ s t [1 : 0]

ビット 2 2 ～ 2 3 (R O) P o r t _ 2 7 _ s t [1 : 0]

ビット 2 4 ～ 2 5 (R O) P o r t _ 2 8 _ s t [1 : 0] C P U ポートは常に前送 (1 1)

ビット 2 6 ～ 3 1 (R O) 予約。常に 0 として読出し

10

20

30

40

50

ポート状態セットアップ・レジスタに対する `mcbregs` インターフェース
`SourcePort [7:0] (in)` `mcbhash` モジュールからのソース・ポート番号
`m_HashDstprt [7:0] (in)` `mcbhash` モジュールからの宛先ポート
`SrcPrtState [1:0] (out)` ソース・ポート・レジスタおよびポート状態レジスタに基づく `mcbhash` に対する組合わせ出力
`DstPrtState [1:0] (out)` `m_HashDstPrt` およびポート状態レジスタに基づく `mcbhash` に対する組合わせ出力。

【0275】

10

【表36】

パケット・メモリ定義

メモリ・セクター情報レジスタ (オフセット = 'ha0) パケット・メモリは、固定数のセクターからなる。このレジスタは、セクター・サイズを定義する。2 K バイトの最小セクター・サイズは、最大パケット (1518 バイト + オーバーヘッド) が1つ以上のセクター境界の抵触を行い得ないことを保証する。2 K バイトの現在唯一つのセクター・サイズがサポートされる。このレジスタは、マスタ・スイッチ使用可能 (E P S M セットアップ・レジスタ) が否定される時にのみ書込まれる

ビット0～1 (W/R) セクター・サイズ。2 K バイトの現在唯一つのセクターがサポートされる

20

00 = 2 K バイト (デフォルト)

01 = 4 K バイト

10 = 8 K バイト

11 = 16 K バイト

ビット2～31 (R/O) 予約。常に0として読出し。

【0276】

【表37】

メモリ・バス帯域幅モニタ

メモリ・バス・モニタ制御レジスタ (オフセット = 'ha8) 当該レジスタにより制御される2つの独立バス・モニタがある。モニタ選択ビット (24) は、どのモニタがアクセス中であるかを選択するため用いられる。このビットはまた、メモリ・バス・モニタ閾値レジスタとメモリ使用レジスタとに対するアクセスを制御する。モニタ・ビットは、当該レジスタの高いバイトのみを書込むことにより独立的にセットすることができる

30

ビット0～9 (W/R) モニタ・モード [9:0]。監視されるべきバスの活動状態のタイプを定義

デフォルトは10'h3FF (全てを監視)

CycleType (1つ以上のビットをセット)

ビット0 パケット (パケット関連トラフィックを監視するようセット)

ビット1 ハッシュ (ハッシュ索引トラフィックを監視するようセット)

ビット2 CPU (メモリに対するCPUアクセスを監視するようセット)

40

ビット3 リフレッシュ (リフレッシュ・サイクルを監視するようセット)

パケット・タイプ (パケット・ビット (0) がセットされるならば、一方または両方のビットをセットしなければならない)

ビット4 ユニキャスト (既知の個々のアドレス・ード・パケットを監視するようセット)

ビット5 ブロードキャスト (グループ・ビット・セットまたはハッシュ・ミスを持つパケットを監視するようセット)

パケットTx/Rc (パケット・ビット (0) がセットされるならば、一方または両方のビットをセットしなければならない)

ビット6 送信 (送信関連トラフィックを監視するようセット)

50

ビット7 受信（受信関連トラフィックを監視するようセット）

パケット・データ／オーバーヘッド（パケット・ビット（0）がセットされるならば、一方または両方のビットをセットしなければならない）

ビット8 データ（パケット転送のデータ部分をモニタするようセット）

ビット9 オーバーヘッド（パケット転送の非データ関連部分、即ち、バス調停、パケット・メモリ保守、未使用サイクルをモニタするようセット）

ビット10～15（RO） 予約。常に0を讀出す

ビット16～19（RO） フィルタ・タイム・スケール。LPフィルタリングのための略々時定数をセット

0h = 75ミリ秒 4h = 300ミリ秒 8h = 予約 Ch = 予約

10

1h = 600ミリ秒 5h = 2.5秒 9h = 予約 Dh = 予約

2h = 5ミリ秒 6h = 20秒 Ah = 予約 Eh = 予約

3h = 40ミリ秒 7h = 2.5分 Bh = 予約 Fh = 予約

デフォルト = 0h。フィルタ・モードにおいてのみ適用

ビット20（W/R） カウント／フィルタ・モード。（デフォルト = 0、フィルタ・モード）

0 = モニターが、フィルタ・タイムスケールにより定義されるように低域通過フィルタとして動作

バス使用レジスタの讀出しは、フィルタ・モニタにおける値に影響を及ぼさない

1 = モニタ・カウント・バス・サイクル、しかし、フィルタ動作は行わない。カウント・モードにある時、讀出し時はバス使用レジスタはクリアされる

20

ビット21（W/R） タイマ・モード。カウント・モードにある時のみ適用（デフォルト = 0）

0 = モニター・モード・ビットにより定義されるサイクルのみをカウント

1 = クロック・サイクルごとにカウンタを増分

ビット22（W/R） バックプレシャ可能化。1 = 全てのポートにおけるバックプレシャを使用可能するため当該モニタからのアラーム使用。デフォルト = 0、不使用可能

ビット23（W/R） ブロードキャスト制御使用可能。1 = 任意のポートから受信されたブロードキャスト・パケットを捨てるためモニタからのアラームを使用

デフォルト = 0、不使用可能

30

ビット24（W/R） モニタ選択。0 = モニタ0（デフォルト）

1 = モニタ1

ビット25（RO） 予約。常に0を讀出し

メモリ・バス・モニター閾値／BWレジスタ（オフセット = 'hac）モニタ選択ビットは、当該レジスタのアクセスに先立ちセットされねばならない

ビット0～7（W/R） アラーム設定閾値。バス使用がこの値に達するかあるいはこれを越えるならば、アラーム・フラグがセットされ、CPU割込みが生成される。使用可能にされるならば、バックプレシャまたはブロードキャスト制御が適用される（デフォルト = 8'h00）

ビット8～15（W/R） アラーム・クリア閾値。バス使用が、この値まで低減するかあるいはこれより低減する時、アラーム・フラグがクリアされ、CPU割込みが生成されるバックプレシャおよびブロードキャストの制御が解放される

40

（デフォルト = 8'h00）

ビット16～23（RO） ピークBW。最後の讀出し以後、最大帯域幅が検出される。讀出し時に、クリアされる

ビット24～31（RO） その時のBW。バス帯域幅使用フィルタのその時の値。00h値は、0%の使用を表わし、FFhの値は100%の使用を表わす

メモリ・バス使用レジスタ（オフセット = 'hb0）当該レジスタのアクセスに先立ち、モニタ選択ビットがセットされねばならない

ビット0～31（RO） バス使用 [31:0]。メモリ・バス使用カウンタカウント・

50

モードでは、カウンタが最後に始動された後は、この値は使用中のバス・サイクル数のカウントである。読出し時にクリアされる。両方の「バス利用レジスタ」が読出された時、両方のフィルタのカウンタが同時に始動する

フィルタ・モードでは、上位8ビットがカウントBWとして閾値／BWに複写されるので、このレジスタを読出す必要がない。BWに対して8ビットより多くを使用することが望ましければ、最大域幅値が常に32'hFFF00__0000となり、かつ選択されるタイム・スロットに応じて最小値が32'hFFF00__0000と32'h00FF__FFFFの間になることに注意すべきである

フィルタ・モードで読出された時は、クリアされない。

【0277】

10

【表38】

メモリ帯域幅モニターに対するmcbrregsインターフェース

SelectedBandWidth[31:0](in) 選択されたモニターに対するメモリ・バス利用レジスタ [31:0]。また、ビット24～31が閾値／BWレジスタにおけるその時のBWである
SelectedMaxBW[7:0](in) 閾値／BWレジスタ・ビット16～23におけるピークBW

Alarm0(in) モニター0に対するアラーム・フラグ。このフラグがセットされクリアされる時、mcbrregsが割込みBWA L A R M S E T 0とBWA L A R M C L R 0を生成する

Alarm1(in) モニター1に対するアラーム・フラグ。このフラグがセットされクリアされる時、mcbrregsが割込みBWA L A R M S E T 1とBWA L A R M C L R 1を生成する

20

r_MonMode0[9:0](out) モニター0に対するモニター・モード

r_MonMode1[9:0](out) モニター1に対するモニター・モード

r_BwScale0[2:0](out) モニター0に対するフィルタ・タイムスケール

r_BwScale1[2:0](out) モニター1に対するフィルタ・タイムスケール

r_CountOnly0(out) モニター0に対するカウント／フィルタ・モード

r_CountOnly1(out) モニター1に対するカウント／フィルタ・モード

r_TimerMode0(out) モニター0に対するタイマ・モード

r_TimerMode1(out) モニター1に対するタイマ・モード

30

r_BackPresOnAlarm0(o) モニター0に対するバックプレシャ使用可能

r_BackPresOnAlarm1(o) モニター1に対するバックプレシャ使用可能

r_BackBcPktsOnAlarm0(o) モニター0に対するブロードキャスト制御使用可能ビット

r_DropBcPktsOnAlarm1(o) モニター1に対するブロードキャスト制御使用可能ビット

r_FilterSelect(out) モニター選択ビット

r_AlarmSet0[7:0](out) モニター0に対するアラーム・セット閾値

r_AlarmSet1[7:0](out) モニター1に対するアラーム・セット閾値

r_AlarmClr[7:0](out) モニター1に対するアラーム・クリア閾値

ClrBwCtr0(out) モニター0に対する利用レジスタが読出される時1クロック

に対してアサート

40

ClrBwCtrl(out) モニター1に対する利用レジスタが読出される時1クロック

に対してアサート

ClrMaxBW0(out) モニター0に対する閾値／BWレジスタが読出される時1

クロックに対してアサート

ClrMaxBW0(out) モニター0に対する閾値／BWレジスタが読出される時1

クロックに対してアサート。

【0278】

【表39】

ドロップ・パケット統計

メモリのオーバーフロー、同報オーバーフロー、受信セクター・オーバーフローおよび送

50

信セクター・オーバーフローによりドロップされたパケットがカウントされる。受信セクター・オーバーフローおよび送信セクター・オーバーフローに対するこれらのカウントおよびビット・マップが保持される。これら条件もまた、CPU 230に対する割込みを生じる。割込み情報が、MCB割込みソース・レジスタに保持される

ドロップ・パケット・メモリ・オーバーフロー・レジスタ (オフセット = 'h b 8) このレジスタは、2つの条件により生じるメモリ・オーバーフローによりドロップされたパケット数を含む。これら条件は、パケットが記憶されている時のハッシュ索引および実際のメモリ・オーバーフロー時に閾値を越えさせられ、これが打切りパケットを生じる
ビット 0 ~ 31 (W/R) メモリ・オーバーフローによりドロップされたパケット数
ドロップ・パケット・ブロードキャスト・オーバーフロー・レジスタ (オフセット = h b c) 10

このレジスタは、ブロードキャスト閾値オーバーフローによりドロップされたパケット数を含む

ビット 0 ~ 31 (W/R) ブロードキャスト閾値オーバーフローによりドロップされたパケット数

ドロップ・パケット受信セクタ・オーバーフロー・レジスタ (オフセット = 'h d 4)

このレジスタは、受信セクタ・オーバーフローにより外されたパケット数を保持する

ビット 0 ~ 31 (W/R) 受信セクタ・オーバーフローにより外されたパケット数

ドロップ・パケット送信セクタ・オーバーフロー・レジスタ (オフセット = 'h d 8)

このレジスタは、送信セクタ・オーバーフローにより外されたパケット数を保持する 20

ビット 0 ~ 31 (W/R) 送信セクタ・オーバーフローにより外されたパケット数

ドロップ・パケット受信セクタ・ビット・マップ・レジスタ (オフセット = 'h d c)

このレジスタは、受信セクタ・オーバーフローによりパケットをドロップしたポートのビット・マップを保持する

ビット 0 ~ 28 (W/R) 受信セクタ使用のオーバーフローを通知するポートのビット・マップ

ドロップ・パケット送信セクタ・ビット・マップ・レジスタ (オフセット = 'h e 0)

このレジスタは、送信セクタ・オーバーフローによりパケットをドロップしたポートのビット・マップを保持する

ビット 0 ~ 28 (W/R) 送信セクタ使用のオーバーフローを通知するポートのビット・マップ 30

ドロップ・パケット統計に対するm c b r e g sインターフェース

x__R x P k t A b o r t e d__ メモリ・オーバーフローによりパケットが打切られた時を通知するXCBからストローブ

D r o p P k t S t b __M e m O F メモリをオーバーフローするのでパケットが外された時を通知するストローブ

D r o p P k t S t b __B C O F ブロードキャスト閾値がオーバーフローするのでパケットが打切られた時を通知するストローブ

D r o p P k t S t b __R x O F 受信セクタ閾値がオーバーフローするのでパケットがドロップされた時を通知するストローブ 40

D r o p P k t S t b __T x O F 送信セクタ閾値がオーバーフローするのでパケットがドロップされた時を通知するストローブ。

【0279】

【表40】

ハッシュ・テーブル定義

ハッシュ・テーブル定義レジスタ (オフセット = 'h c 0) 主要ハッシュ・エントリ・テーブルの基底アドレスおよびサイズを定義。ハッシュ・テーブルの多重コピーがメモリに保持されるならば、E P S M 2 1 0 スイッチを間に持つようにこのレジスタが使用される

ビット 0 ~ 14 (R/O) 主要ハッシュならば、テーブル基底アドレス [16 : 2]。常 50

に 0

ビット 15～23 (RO) 主要ハッシュ・テーブル基底アドレス [25:17] 常に 0
ビット 24～25 (W/R) 主要ハッシュ・テーブル・サイズ [1:0]。(デフォルト
は 00)

00 = キー・サイズ 13 ビット。テーブル・サイズ 1

28 K バイト (8 K 16 バイト・エントリ)

01 = キー・サイズ 14 ビット。テーブル・サイズ 2

56 K バイト (16 K 16 バイト・エントリ)

(基底アドレス・ビット 17 が無視され、内部で 0 に強制される)

10 = キー・サイズ 15 ビット。テーブル・サイズ 5

12 K バイト (32 K 16 バイト・エントリ)

(基底アドレス・ビット 18:17 が無視され、内部で 0 に強制される)

11 = キー・サイズ 16 ビット。テーブル・サイズ 1

メガバイト (64 K 16 バイト・エントリ)

(基底アドレス・ビット 19:17 が無視され、内部で 0 に強制される)

ビット 26 (W/R) ハッシュ・サイクル・ロック。このビットのセッティングが、ハ
ッシュ索引中のメモリ・サイクルをロックさせる。これは、メモリに対するパケット読出
しおよび書込み転送を遅らせることを代償に、ハッシュ索引時間を最短化する。デフォル
トは 0

ビット 31:27 (RO) 予約。常に 0 として読出し

ハッシュ・テーブル定義レジスタに対する m c b r e g s インターフェース

r_HashBaseAdr[25:17](out) 基底アドレスを m e m h a s h モジュールへ通過 r_HashKey
Size[1:0](out) キー・サイズを m e m h a s h モジュールへ通過

r_LockHashCycs(out) ロック・ハッシュ・サイクル・ビットがセットされるなら
ば、m c b h a s h モジュールへアサート

HashLookUpIP(in) ハッシュ索引が進行中でありハッシュ・テ

ブル定義レジスタに対する書込みが無視されるまで延期されるべきことを表示するため m
c b h a s h モジュールによりアサート。m c b h a s h が、H a s h L o o k U p I P
が否定されると任意の立上がりクロック・エッジでレジスタを更新。

【0280】

【表 41】

ソース・ポート学習

ハッシュ・ソース・ミス・レジスタ・ロー (オフセット = ' h c c) ハッシュ・テー
ブルに付加される新しいソース・アドレスのバイト 3:0。これらレジスタがロードされ、
ハッシュ S A が未知であるか、あるいはポートが変化し、かつソース・ポートが学習不使
用にされる時に、割込みが発される。レジスタは、ハッシュ・ソース・ミス・レジスタ・
ハイのレジスタが読出されるまでロックされる (ローのレジスタが最初に読出されねばな
らない)。レジスタがロックされる間遭遇された未知の S A またはポート変化は否定され
る。

ビット 0～7 (RO) ハッシュ・テーブルに追加されるべき M A C アドレスのバイト 0
(高次のアドレス・バイト。グループ・ビット = ビット 0)

ビット 8～15 (RO) ハッシュ・テーブルに追加されるべき M A C アドレスの
バイト 1

ビット 16～23 (RO) ハッシュ・テーブルに追加されるべき M A C アドレス
のバイト 2

ビット 24～31 (RO) ハッシュ・テーブルに追加されるべき M A C アドレス
のバイト 3

ハッシュ・ソース・ミス・レジスタ・ハイ (オフセット = ' h d 0) 新たなソース
・アドレスとソース・ポート I D のバイト 5:4

ビット 0～7 (RO) ハッシュ・テーブルに追加されるべき M A C アドレスの

10

20

30

40

50

バイト 4

ビット 8 ～ 15 (RO) ハッシュ・テーブルに追加されるべきMACアドレスの

バイト 5

ビット 16 ～ 23 (RO) ハッシュ・テーブルへ追加されるべきソース・ポート

ID [7:0]

ビット 24 ～ 31 (RO) 予約。常に0として読出し

学習不能ポート・レジスタ (オフセット='he4) ビット・マップされた学習不能ポート・レジスタ。CPUには適用しない

ビット 0 (W/R) ポート 0 学習不能。1 = 使用不能。デフォルト = 0

ビット 1 (W/R) ポート 1 学習不能。1 = 使用不能。デフォルト = 0

10

...

ビット 28 (W/R) ポート 28 学習不能。1 = 使用不能。デフォルト = 0

ビット 29 ～ 31 (W/R) 予約。常に0として読出し

ソース・ポート学習に対するmcbreregインターフェース

SelectedAdr[47:0](in) memhashモジュールからのソース・アドレス

SourcePort[47:0](in) memhashモジュールからのソース・ポート

SrcMissStb(in) ハッシュSAミスが生じた時memhashモジュールによりアサートされ、ソース・ミス・レジスタおよびソース・ポートが妥当する。ハッシュ・ソース・ミス・レジスタに対してゲートとして使用されるならば、memhashは保持時間を保証する

20

SrcMissLock(out) 学習不能がポートに対してセットされたかどうかアサートする。これは、ソース・ポート入力と学習不能レジスタに基くmemhashに対する組合せ出力であり、連続的に評価される。memhashはサンプルする時を知る。

【0281】

【表42】

ポート・セキュリティ

ソース・ポート・レジスタ (オフセット='he8) ビット・マップされたソース・ポート・レジスタ。(セキュリティが使用可能にされたポートに対して学習不能ビットをセットすることも望ましい)

ビット 0 (W/R) ポート 0 セキュリティ可能。1 = 使用可能

30

デフォルト = 0

ビット 1 (W/R) ポート 1 セキュリティ可能。1 = 使用可能

デフォルト = 0

...

ビット 28 (W/R) ポート 28 セキュリティ使用可能。1 = 使用可能

デフォルト = 0

ビット 29 ～ 31 (RO) 予約。常に0として読出し

セキュリティ違反レジスタ (オフセット='hf0) ポートによりビット・マップされたセキュリティ違反。読出し時にクリア。0に初期設定。最初のビットがセットされる時割込みが発せられ、読出される時クリアされる

40

ビット 0 (RO) セキュリティ違反ポート 0。1 = 違反の発生

ビット 1 (RO) セキュリティ違反ポート 1。1 = 違反の発生

...

ビット 28 (RO) セキュリティ違反ポート 28。1 = 違反の発生

ビット 29 ～ 31 (RO) 予約。常に0として読出し

セキュリティ違反統計レジスタ (オフセット='hec) 全てのポートにおける全セキュリティ違反のカウント。読出し時にクリア。0に初期設定

ビット 0 ～ 31 (RO) セキュリティ違反カウント [31:0]

ポートセキュリティに対するmcbreregインターフェース

SourcePort[7:0](in) memhashモジュールからのソース・ポート番号

50

SecurePort(out) セキュリティ・モードがポートに対してセットされるかどうかアサート。これは、SecurePort入力およびソース・ポート・レジスタに基づくmemhashに対する組合わせ出力であり、連続的に評価される。memhashはサンプルする時を知る

SecViolationStb(in) セキュリティ違反が表示されたソース・ポートに生じたことを示すストローブ。ソース・ポートにより示されるセキュリティ違反レジスタに対してゲートとして用いられるならば、memhashが保持時間を保証する。

【0282】

【表43】

メモリ・コンフィギュレーション

10

メモリ制御レジスタ (オフセット='hf4) 種々のメモリ制御機能。マスタ・スイッチ使用可能 (EP SMセットアップ・レジスタ) が否定される時に、このレジスタのみが書込まれる

ビット0～1 (W/R) メモリ・タイプ

00 = 高速ページ・モードDRAM (デフォルト)

01 = ED0 DRAM

10 = シンクロナスDRAM

11 = 留保

ビット2 (W/R) メモリ速度 (0 = 60 ns、1 = 50 ns)

デフォルトは0ビット3 (W/R)

20

ED0テスト・モード (1 = 可能化)。デフォルトは0

ビット4 (W/R) ダブル・リンク・モード。デフォルトは0

ビット5 (W/R) 使用不能受信ページ・ヒット。デフォルトは0

ビット6 (W/R) 使用不能送信ページ・ヒット。デフォルトは0

ビット7～31 (RO) 予約。常に0として読出し

メモリ制御レジスタに対するmcbrregsインターフェース

r_MemED0(out) メモリ・タイプが01ならば、memctlモジュールに対してmcbrregsによりアサート

r_MemSync(out) メモリ・タイプが10ならば、memctlモジュールに対してmcbrregsによりアサート

30

r_Mem50ns(out) メモリ速度ビットが1ならば、memctlモジュールに対してmcbrregsによりアサート

r_TestForED0(out) ED0テスト・モードが1ならば、memctlモジュールに対してmcbrregsによりアサート

ビット0～1 (W/R) 0000000h～03FFFFFFhに対するRAS選択 (4M)

ビット2～3 (W/R) 0400000h～07FFFFFFhに対するRAS選択 (8M)

ビット4～5 (W/R) 0800000h～0BFFFFFFhに対するRAS選択 (12M)

40

ビット6～8 (W/R) 0C00000h～0FFFFFFFhに対するRAS選択 (16M)

...

ビット30～31 (W/R) 3C00000h～3FFFFFFFhに対するRAS選択 (64M)

RAS選択は次のようにコード化される。即ち、00 = RAS0、01 = RAS1、0 = RAS2、11 = RAS3。デフォルトは常に0

メモリRAS選択レジスタに対するmcbrregsインターフェース

r_RasSelReg[31:0](out) データをmcbrregsからmemctlモジュールへ送る

50

メモリ・リフレッシュ・カウンタ・レジスタ (オフセット='h f c) リフレッシュ要求間のCLKサイクル数を定義

ビット0～9 (W/R) リフレッシュ・カウンタ [9:0]。リフレッシュ・カウンタ×CLK周期は15.625ミリ秒より小さいかこれと等しくなければならない。デフォルトは20.8h。(30ns CLKに対しては、15.60ミリ秒)

ビット10～31 (RO) 予約。常に0として読出し

メモリ・リフレッシュ・カウンタ・レジスタに対するmcbrregsインターフェース

RefReq(out) memctl モジュールに対するリフレッシュ要求ストローブ。memctl が正のエッジにおける要求を検出するので、ストローブは任意の長さでよい。確認は返さない。

【0283】

【表44】

MACアドレス・フィルタリング

CPU230に出入りさせるようパケットを指向するため、宛先アドレスに基づくフィルタリングが行われる。その時2つのみに対する要求が存在するが、4つのフィルタが設けられる。その時要求がなくとも、アドレス比較に「ドント・ケア」を含むマスキングが得られる。1つがCPU230の個々のアドレスを持ち他が(スパニング・ツリーに対する)BPDU多重キャスト・アドレスを持つ2つのフィルタをセットアップすべきである。CPU230でないポートからの受信パケットがフィルタ・アドレスにヒットするならば、(BCまたはMCであっても)パケットはCPU230のみへ送られる。CPU230から起生したパケットがフィルタ・アドレス(BPDUアドレス)にヒットするならば、このパケットはフィルタ・アドレス・レジスタに指定された宛先ポートへ送られる。パケットがフィルタ・アドレスにヒットするならば、ハッシュ・テーブルの宛先索引が迂回される

フィルタ制御レジスタ (オフセット='h 1 0 0) MAC宛先アドレス・フィルタリングを制御

ビット0～3 (W/R) フィルタ使用可能アドレス [3:0]。1=対応するアドレス・フィルタ・レジスタ [3:0] に対する個々の宛先アドレス・フィルタリング使用可能。デフォルトは0

ビット4～7 (W/R) アドレス・マスク使用可能 [3:0]。1=アドレス・フィルタ・レジスタ [3:0] がアドレス・フィルタ・マスク・レジスタを持つならば、マスキング使用可能。デフォルトは0

フィルタ・マスク・レジスタ・ロー (オフセット='h 1 0 4) デフォルト=0

ビット0～7 (W/R) MACアドレス・マスクのバイト0 (1=マスク・アドレス・ビット)

ビット8～15 (W/R) MACアドレス・マスクのバイト1 (1=マスク・アドレス・ビット)

ビット16～23 (W/R) MACアドレス・マスクのバイト2 (1=マスク・アドレス・ビット)

ビット24～31 (W/R) MACアドレス・マスクのバイト3 (1=マスク・アドレス・ビット)

フィルタ・マスク・レジスタ・ハイ (オフセット='h 1 0 8) デフォルト=0

ビット0～7 (W/R) MACアドレス・マスクのバイト4 (1=マスク・アドレス・ビット)

ビット8～15 (W/R) MACアドレス・マスクのバイト5 (1=マスク・アドレス・ビット)

ビット16～31 (RO) 予約。常に0として読出し

フィルタ・アドレス・レジスタ0ロー (オフセット='h 1 0 c)

ビット0～7 (W/R) 前送されるMACアドレスのバイト0

ビット8～15 (W/R) 前送されるMACアドレスのバイト1

10

20

30

40

50

ビット16～23 (W/R) 前送されるMACアドレスのバイト2
 ビット24～31 (W/R) 前送されるMACアドレスのバイト3
 フィルタ・アドレス・レジスタ0ハイ (オフセット='h110)
 ビット0～7 (W/R) 送られるMACアドレスのバイト4
 ビット8～15 (W/R) 送られるMACアドレスのバイト5
 ビット16～23 (W/R) 宛先ポート。ソース・ポートがCPU230ならば、MAC
 アドレスがフィールド・アドレスに一致するならば、このフィールドがパケットをどのポ
 ートへ送るべきかを指定する。ソース・ポートがCPU230でなければ、このフィール
 ドが無視されて、フィールドMACアドレスに対するヒットがCPU230へ送られる
 ビット24～31 (W/R) 予約。常に0として読出し 10
 フィルタ・アドレス・レジスタ1ロー (オフセット='h114) 前参照
 フィルタ・アドレス・レジスタ1ハイ (オフセット='h118) 前参照
 フィルタ・アドレス・レジスタ2ロー (オフセット='h11c) 前参照
 フィルタ・アドレス・レジスタ2ハイ (オフセット='h120) 前参照
 フィルタ・アドレス・レジスタ3ロー (オフセット='h124) 前参照
 フィルタ・アドレス・レジスタ3ハイ (オフセット='h128) 前参照
アドレス・フィルタリングに対するmcbrsgsインターフェース
 SelectedAdr[47:0](in) memhashモジュールからの宛先アドレス
 filterHit(out) フィルタ・アドレス・ヒットが生じるならば、アサートこれは
 、SelectedAdrおよびフィルタ・レジスタに基くmemhashに対する組合せ出力であり、連続的に評価される。memhashはサンプルする時を知る 20
 FilterPort[7:0](out) ソース・ポートがCPU230であれば、FilterPort
 は、フィルタ・ヒットを生成するフィルタ・レジスタからの宛て先ポート・フィールドに
 等しい。ソース・ポートがCPU230でなければ、FilterPortは(EPSM
 セットアップ・レジスタからの)CpuPortに等しい
 SourcePort[7:0](in) memhashモジュールからのソース・ポート番号
 SrcPrtIsCpu SourcePort入力がEPSMセットアップ・レジスタに
 おけるCpuPort番号と整合するならば、アサートされる。

【0284】

【表45】

MCB割込み情報

MCB404には8つの割込みソースがある。割込みソースは、ソースがマスクされなけ
 れば、CPU230を割込みさせる。CPU230が割込みされずに割込みソースの情報
 を得ることを可能にするため、ポーリング機構が使用可能である。割込みソースのマス
 キングは、割込みをCPU230からブロックさせるが、情報はポーリング・ソース・レジ
 スタから依然として得られる。

MCB割込みソース・レジスタ (オフセット='h12c) CPU230へ送られる割
 込みのソース。このレジスタは、EPSM210により更新され、割込みがCPU230
 へ送られる。CPU230がこのレジスタを読出す時、内容はクリアされる。ビットにお
 ける1の値は、割込みが生じたことを示す。デフォルト=32'h0000_0000 40
 ビット0 (W/R) セキュリティ割込み。セキュリティ違反が生じると、この割込みが
 生じる

ビット1 (W/R) メモリ・オーバーフロー・セット。メモリがパケットで一杯になり
 オーバーフロー閾値が送られると、この割込みが生じる

ビット2 (W/R) メモリ・オーバーフロー・クリア。メモリが空になりオーバーフロ
 ー閾値が送られると、この割込みが生じる

ビット3 (W/R) セットのブロードキャスト。ブロードキャスト・パケットがメモリ
 を一杯にし、ブロードキャスト閾値が送られると、この割込みが生じる

ビット4 (W/R) クリアのブロードキャスト。ブロードキャスト・パケットがメモリ
 から空になり、ブロードキャスト閾値が送られると、この割込みが生じる 50

ビット5 (W/R) OFの受取り。ポートがパケットを受取るためのその割付けスペースを越えると、この割込みが生じる

ビット6 (W/R) OFの送出。パケットを送信しているポートがその割付けスペースを越えると、この割込みが生じる

ビット7 (W/R) Rxパケット打切り。パケットが記憶され始め、メモリが超過すると判定されると、パケットが打切られ、この割込みが生じる

ビット8～31 (RO) 予約。常に0として読出し

割込みソース・レジスタに対するmcbrregsインターフェース

割込みマスク・レジスタ (オフセット='h130) CPU230によりマスクされる割込み。任意のビットにおける1の値は、割込みがマスクされることを示す。デフォルト = 32'h0000_0000

10

ビット0 (W/R) セキュリティ割込みに対するマスク

ビット1 (W/R) メモリ・オーバーフロー・セット割込みに対するマスク

ビット2 (W/R) メモリ・オーバーフロー・クリア割込みに対するマスク

ビット3 (W/R) 同報OFセット割込みに対するマスク

ビット4 (W/R) 同報OFクリア割込みに対するマスク

ビット5 (W/R) 受信OF割込みに対するマスク

ビット6 (W/R) 送信OF割込みに対するマスク

ビット7 (W/R) Rxパケット打切り割込みに対するマスク

ビット8～31 (W/R) 予約。常に0として読出し

20

ポーリング・ソース・レジスタ (オフセット='h134) このレジスタは、マスクされた割込み情報を含み、所望のビットをクリアするため、CPU230が1を書込むことによりクリアされる。これにより、CPU230が割込みの代わりにポーリングすることを許容する。CPUは、代わりにポーリングを欲する任意の割込みソースをマスクしなければならない。

ビット0 (W/R) セキュリティ割込み。セキュリティ違反が生じるならば、この割込みが生じる

ビット1 (W/R) メモリ・オーバーフロー・セット。メモリがパケットで一杯となりオーバーフロー閾値が送られると、この割込みが生じる

ビット2 (W/R) メモリ・オーバーフロー・クリア。メモリが空になりオーバーフロー閾値が送られると、この割込みが生じる。

30

ビット3 (W/R) ブロードキャストOFセット。ブロードキャスト・パケットがメモリを一杯にしてブロードキャスト閾値が送られ
と、この割込みが生じる

ビット4 (W/R) ブロードキャストOFクリア。ブロードキャスト・パケットがメモリから空になりブロードキャスト閾値が送られると、この割込みが生じる

ビット5 (W/R) 受取りOF。ポートがパケットを受取るその割付けスペースを越えると、この割込みが生じる

ビット6 (W/R) 送信OF。パケットを送出しているポートがその割付けスペースを越えると、この割込みが生じる

40

ビット7 (W/R) Rxパケット打切り。パケットが記憶され始め、メモリが越えられ
ると判定されると、パケットが打切られてこの割込みが生じる

ビット8～31 (W/R) 予約。常に0として読出し

ポーリング・ソース・レジスタに対するmcbrregsインターフェース。

【0285】

【表46】

バックプレシャ

バックプレシャ使用可能 (オフセット='h138) バックプレシャを使用可能にする
ビット・マップ

ビット0～23 (RO) 予約。常に0として読出し

50

ビット 24～27 (W/R) ビット・マップ

ビット 28～31 (RO) 予約。常に 0 として読出し。

【0286】

【表 47】

ポート・ボンディング

2 組の結合されたポートがある。従って、どのポートと一緒に結合されるかを通知する 2 つのレジスタがある。

(註) 各レジスタにおける僅かに 2 ビットがセットされるべきであり、即ち、2 つのポートと一緒に結合されるべきである。

結合ポート・セット 0 (オフセット='h13c) このビット・マップはこのポートが 10
当該セットにおいて一緒に結合されるかを通知する

ビット 0～27 (W/R) セット 0 に対するビット・マップ

ビット 28～31 (RO) 予約。常に 0 として読出し

結合ポート・セット 1 (オフセット='h140) このビット・マップが、どのポート
が当該セットにおいて一緒に固定されるかを通知する

ビット 0～27 (W/R) セット 1 に対するビット・マップ

ビット 28～31 (RO) 予約。常に 0 として読出し

VLAN

デフォルト VLAN レジスタ (オフセット='h144)

【0287】

ネットワーク・スイッチに対する多重ポート・ポーリング・システムが複数のネットワーク・ポートに対する受送信状態を決定するための有効なシステムを提供することが判る。ポーリング・ロジックが、1 つ照会信号をアサートして複数の送受信状態信号を受取り、これにより多重ポートの状態を一時に受取る。ポーリング・ロジックが、全てのポートの状態を連続的に追跡するように送受信リストを然るべく更新する。このことが、ソース・ポートからのデータを検索する時と伝送のためポートヘデータを提供する時とを決定するためリストを検査する調停および制御ロジックを容易にする。

【0288】

本発明に係る多重バス・アーキテクチャを有するネットワーク・スイッチが以前に公知のネットワーク・スイッチと比較して実質的な改良及び利点を提供することが理解されるであろう。スイッチ・マネージャが一次データ・バスとメモリ・バスとの間に結合されており、これは、それぞれのポート間の且つ或いは作動のモードによってポートとメモリとの間のデータ流を監視して制御する。ポートの各々はネットワークと通信するためのネットワーク・インターフェース、スイッチ・マネージャに結合されたデータ・バス・インターフェース、並びにプロセッサ・バスに結合するためのポート・プロセッサ・インターフェースを含んでいる。このようにして、プロセッサは、オーバーヘッド機能、例えば、スイッチの管理を実行し且つ通常の作動の期間中、データ流に干渉することなくスイッチのポートの各々を構成し、監視し且つ制御するための独立したアクセスを有する。斯くして、プロセッサに関連する実質的なオーバーヘッドは、別のバス・システムによって取り扱われ、その結果、データ・バスのデータ転送のための帯域が増大する。

【0289】

本出願の好適な実施例について説明してきたが、本発明の変形及び変更が本発明の技術的思想を変更することなく可能であることは、当業者に明らかであろう。

【図面の簡単な説明】

【図 1】本発明によるネットワーク・スイッチを含むネットワーク・システムを示す簡略図である。

【図 2】図 1 のネットワーク・スイッチの更に詳細なブロック図である。

【図 3】ネットワーク・スイッチのポートを構成する図 2 のクワッド・カスケード装置形態を示すブロック図である。

【図 4】図 3 に示した特定のクワッド・カスケード装置の信号を示す図である。

10

20

30

40

50

【図 5】図 3 のクワッド・カスケード装置のプロセッサ読出しタイミングを示すタイミング図である。

【図 6】図 3 のクワッド・カスケード装置のプロセッサ書込みタイミングを示すタイミング図である。

【図 7】図 3 のクワッド・カスケード装置のプロセッサ・バースト読出しアクセスを示すタイミング図である。

【図 8】図 3 の各ポートのバッファ状態照会を示す模範的タイミング図である。

【図 9】図 2 の H S B における同時読出し書込みサイクルを示すタイミング図である。

【図 10】図 2 の H S B における同時読出し書込みサイクルを実行する手順を示すフロー図である。

10

【図 11】図 2 のスイッチ・マネージャを示すブロック図である。

【図 12】図 4 のバス・コントローラ・ブロックの更に詳細なブロック図である。

【図 13】図 5 A のバス・コントローラ・ブロックのメモリ内のバッファを示す図である。

【図 14】図 12 のバス・コントローラ・ブロック内の受信ポーリング状態マシンの動作を示す状態図である。

【図 15】図 12 のバス・コントローラ・ブロック内の受信ポーリング状態マシンの動作を示す状態図である。

【図 16】図 12 のバス・コントローラ・ブロック内の受信ポーリング状態マシンの動作を示す状態図である。

20

【図 17】図 12 のバス・コントローラ・ブロック内の受信ポーリング状態マシンの動作を示す状態図である。

【図 18】図 12 のバス・コントローラ・ブロック内の受信ポーリング状態マシンの動作を示す状態図である。

【図 19】図 12 のバス・コントローラ・ブロック内の送信ポーリング状態マシンの動作を示す状態図である。

【図 20】図 12 のバス・コントローラ・ブロック内の送信ポーリング状態マシンの動作を示す状態図である。

【図 21】図 12 のバス・コントローラ・ブロック内の送信ポーリング状態マシンの動作を示す状態図である。

30

【図 22】図 12 のバス・コントローラ・ブロック内の送信ポーリング状態マシンの動作を示す状態図である。

【図 23】図 12 のバス・コントローラ・ブロック内の送信ポーリング状態マシンの動作を示す状態図である。

【図 24】図 11 のメモリ・コントローラ・ブロックの更に詳細なブロック図である。

【図 25】図 11 のプロセッサ・コントローラ・ブロックの更に詳細なブロック図である。

【図 26】図 11 のプロセッサ・コントローラ・ブロックの更に詳細なブロック図である。

【図 27】図 11 のプロセッサ・コントローラ・ブロックの更に詳細なブロック図である。

40

【図 28】図 11 のプロセッサ・コントローラ・ブロックの更に詳細なブロック図である。

【図 29】図 11 のプロセッサ・コントローラ・ブロックの更に詳細なブロック図である。

【図 30】図 2 の Thunder LAN ポート・インターフェース (T P I) を示す簡略ブロック図である。

【図 31】T P I の更に詳細なブロック図である。

【図 32】図 2 の Thunder LAN (T L A N) の各々の構成と機能とを示すブロック図である。

50

【図 3 3】 任意の T L A N により実行される制御リストの全体フォーマットを示す図である。

【図 3 4】 図 2 の P C I バスと関連する T P I により使用される T P I 周辺要素相互接続 (P C I) 構成レジスタの定義を示す図である。

【図 3 5】 T P I により使用される T P I 制御レジスタの定義を示す図である。

【図 3 6】 図 2 の C P U の P C I 初期設定動作を示すフロー図である。

【図 3 7】 T L A N の各々に対する受取り動作を示すフロー図である。

【図 3 8】 図 2 の高速バス (H S B) に跨がる受取りデータ転送動作を示すフロー図である。

【図 3 9】 H S B に跨がる伝送データ転送動作を示すフロー図である。

10

【図 4 0】 T L A N の各々に対する伝送動作を示すフロー図である。

【図 4 1】 図 2 のメモリの構成を示すブロック図である。

【図 4 2】 図 2 のメモリの構成を示すブロック図である。

【図 4 3】 図 2 のメモリの構成を示すブロック図である。

【図 4 4】 図 2 のメモリの構成を示すブロック図である。

【図 4 5】 図 2 のメモリの構成を示すブロック図である。

【図 4 6】 図 2 のメモリの構成を示すブロック図である。

【図 4 7】 図 2 のメモリの構成を示すブロック図である。

【図 4 8】 図 2 のメモリの構成を示すブロック図である。

【図 4 9】 同報パケットを組込んだ幾つかの伝送パケット・リンクを示すブロック図である。

20

【図 5 0】 図 6 のスタティック・メモリの構成を示すブロック図である。

【図 5 1】 図 6 のスタティック・メモリの構成を示すブロック図である。

【図 5 2】 メモリに対するデータ・パケットの受取りと動作のカットスルーモードにおけるデータ・パケットの送出とのための図 2 のネットワーク・スイッチの全体動作を示すフロー図である。

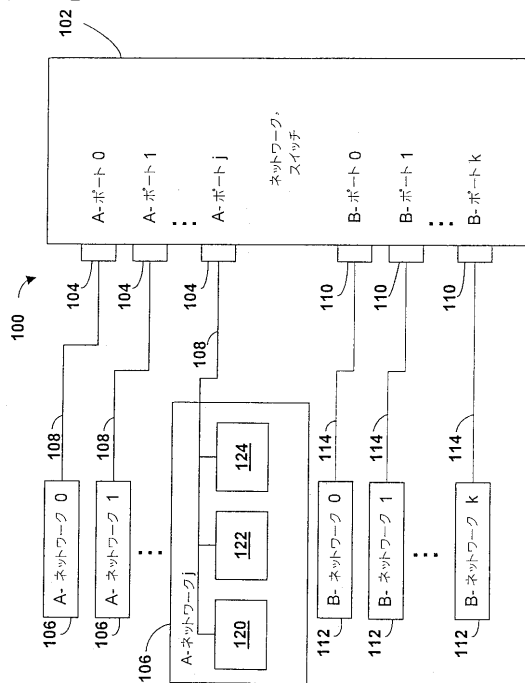
【図 5 3】 メモリからデータ・パケットを伝送するための図 2 のネットワーク・スイッチの全体動作を示すフロー図である。

【図 5 4】 図 2 のスイッチ・マネージャのハッシュ索引動作を示すフロー図である。

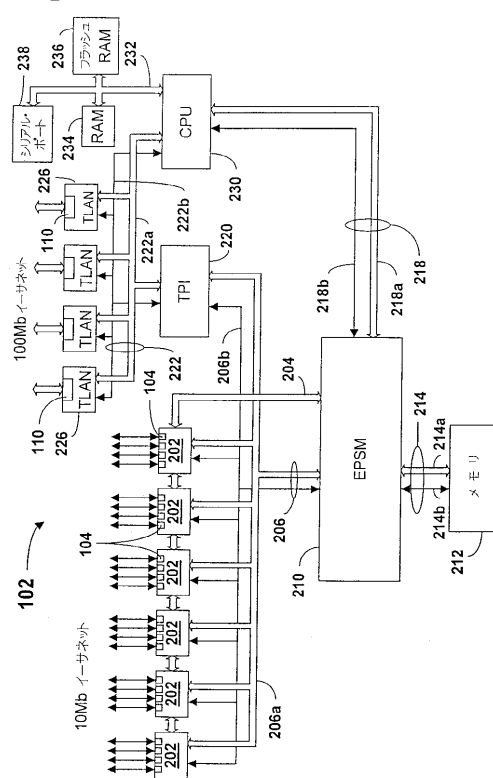
【図 5 5】 図 2 のメモリにおけるハッシュ・テーブル・エントリを探索するためのハッシュ索引手順を示すフロー図である。

30

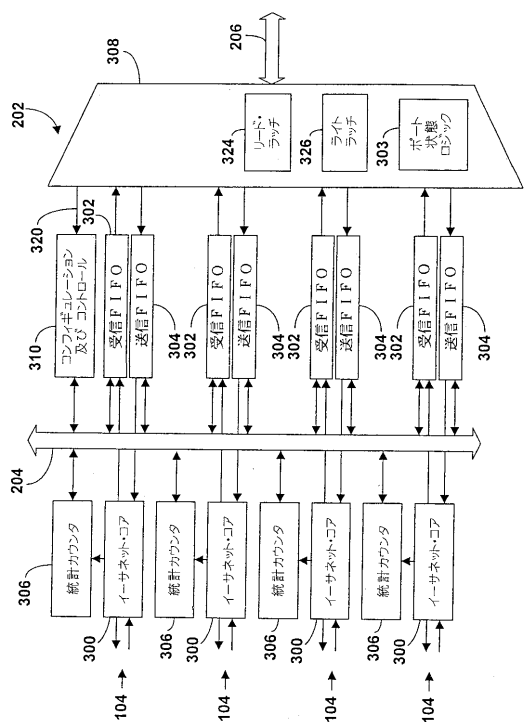
【図 1】



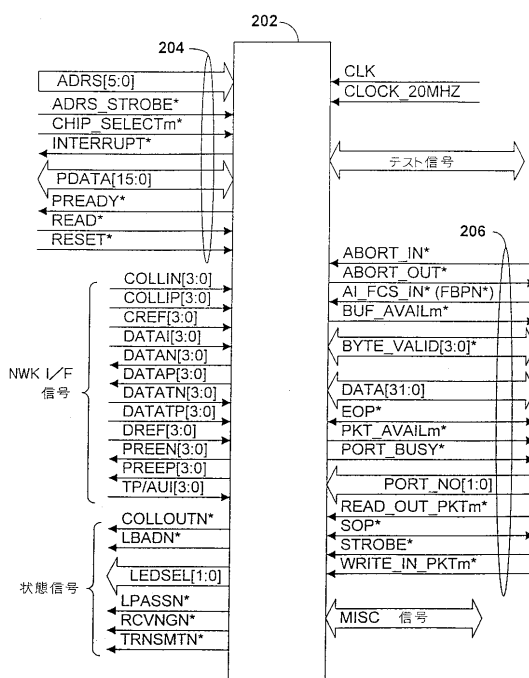
【図 2】



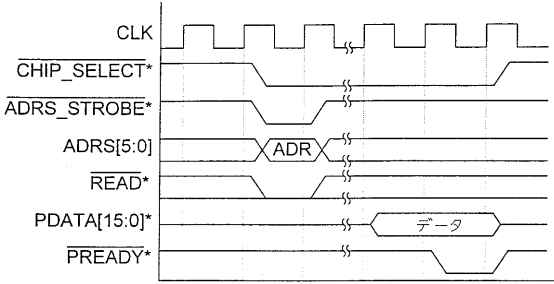
【図 3】



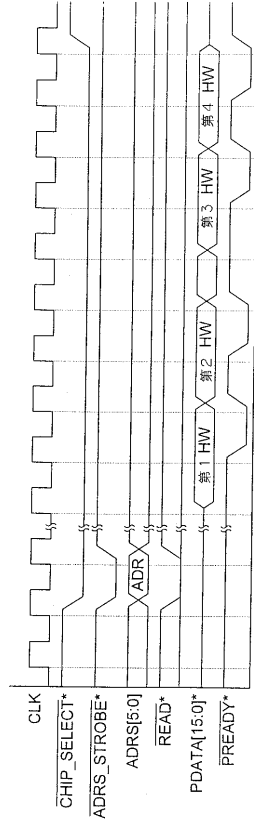
【図 4】



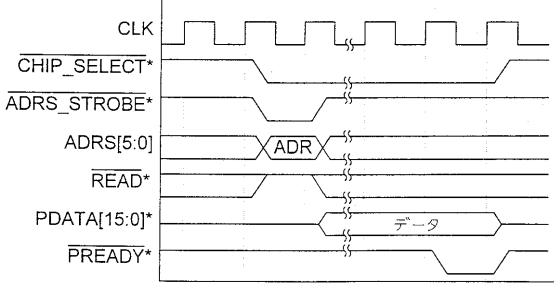
【図 5】



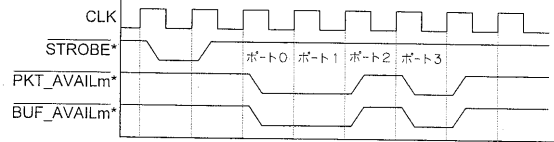
【図 7】



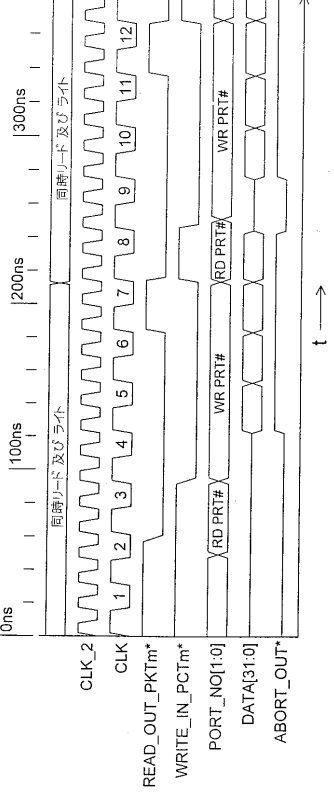
【図 6】



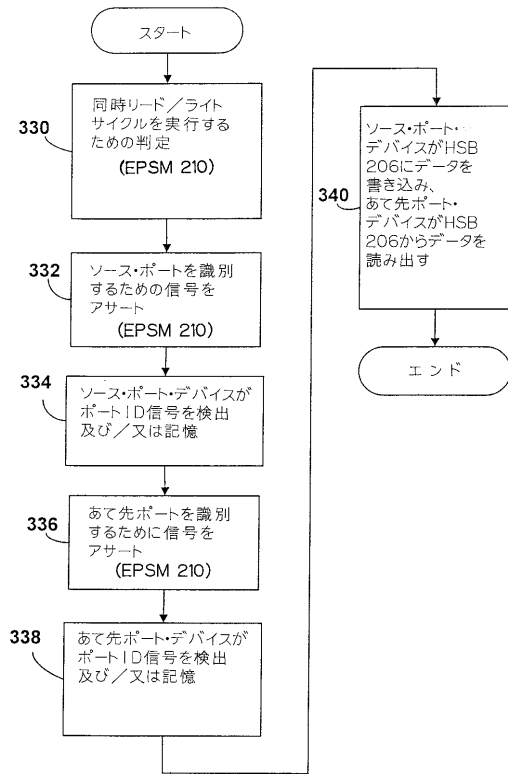
【図 8】



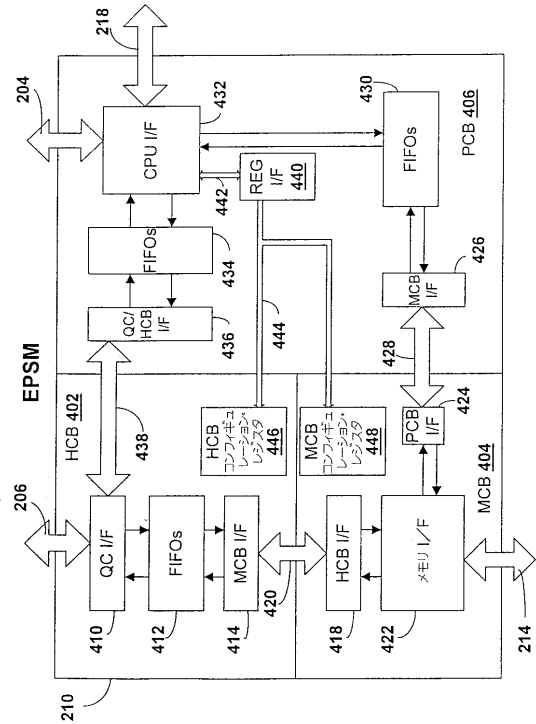
【図 9】



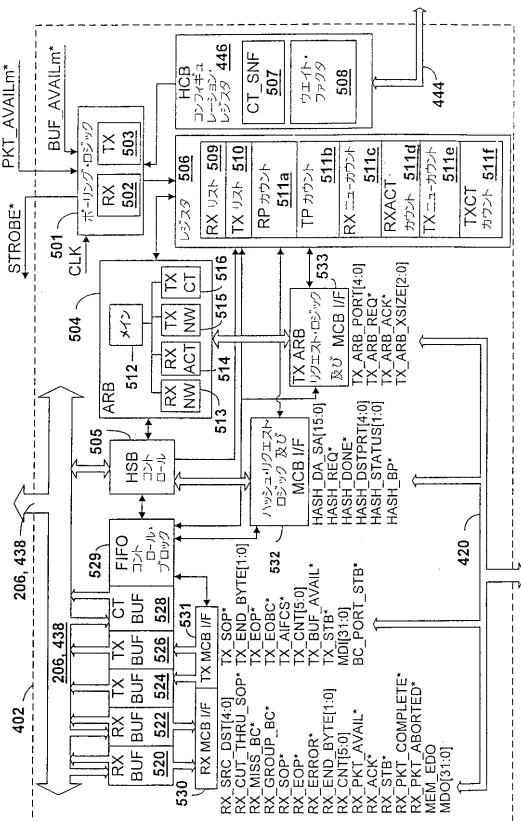
【図 10】



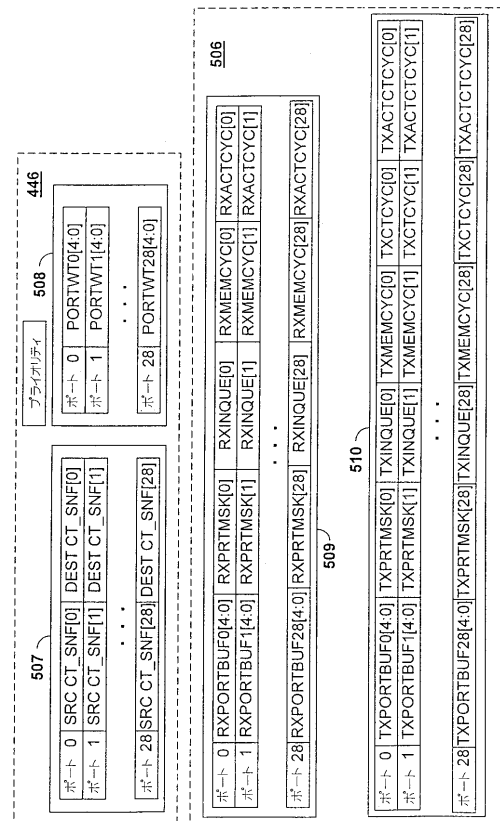
【図 11】



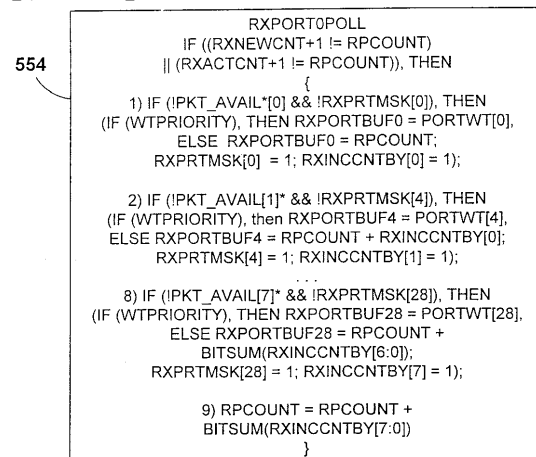
【図 12】



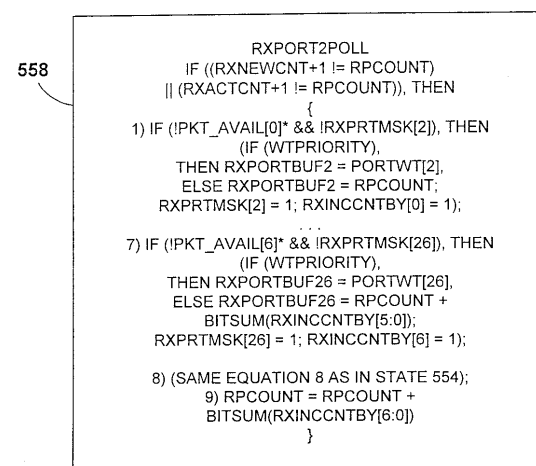
【図 13】



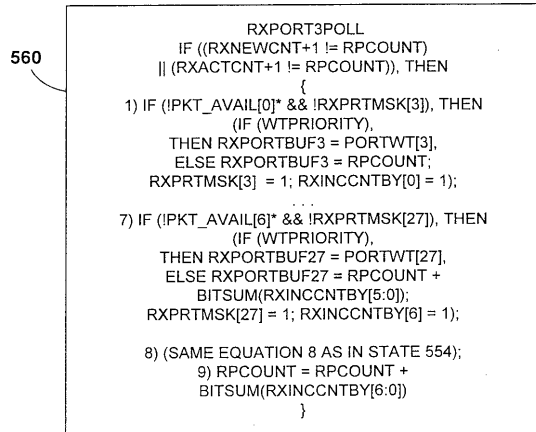
【 図 1 5 】



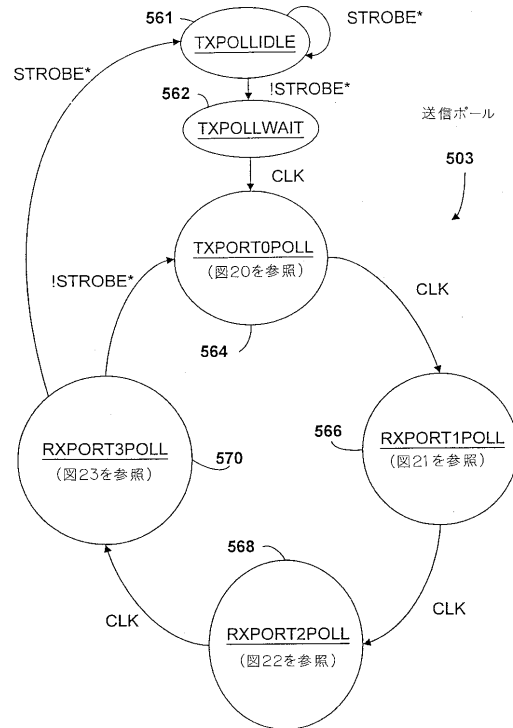
【图 17】



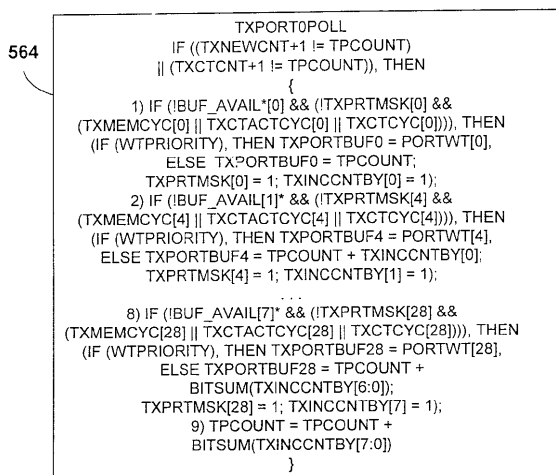
【図 18】



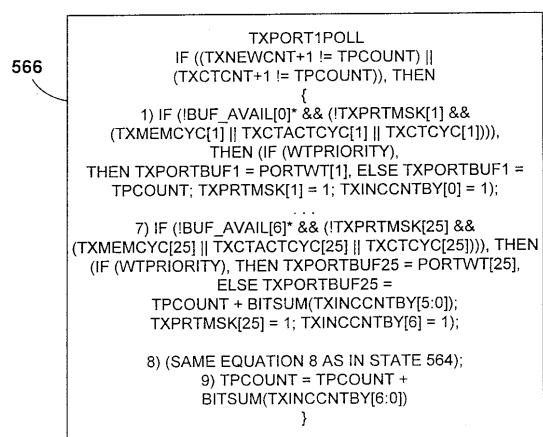
【図 19】



【図 20】



【図 21】



【図 2 2】

568

```

TXPORT2POLL
IF ((TXNEWCNT+1 != TPCOUNT)
|| (TXCTCNT+1 != TPCOUNT)), THEN
{
1) IF (!BUF_AVAIL[0]* && (!TXPRMSK[2] &&
(TXMEMCYC[2] || TXCTACTCYC[2] || TXCTCYC[2]])),
THEN(IF (WTPRIORITY),
THEN TXPORTBUF2 = PORTWT[2],
ELSE TXPORTBUF2 = TPCOUNT;
TXPRMSK[2] = 1; TXINCCNTBY[0] = 1);
...
7) IF (!BUF_AVAIL[6]* && (!TXPRMSK[26] &&
(TXMEMCYC[26] || TXCTACTCYC[26] || TXCTCYC[26]])), THEN
(IF (WTPRIORITY),
THEN TXPORTBUF26 = PORTWT[26],
ELSE TXPORTBUF26 = TPCOUNT +
BITSUM(TXINCCNTBY[5:0]);
TXPRMSK[26] = 1; TXINCCNTBY[6] = 1);
8) (SAME EQUATION 8 AS IN STATE 564);
9) TPCOUNT = TPCOUNT +
BITSUM(TXINCCNTBY[6:0])
}

```

【図 2 3】

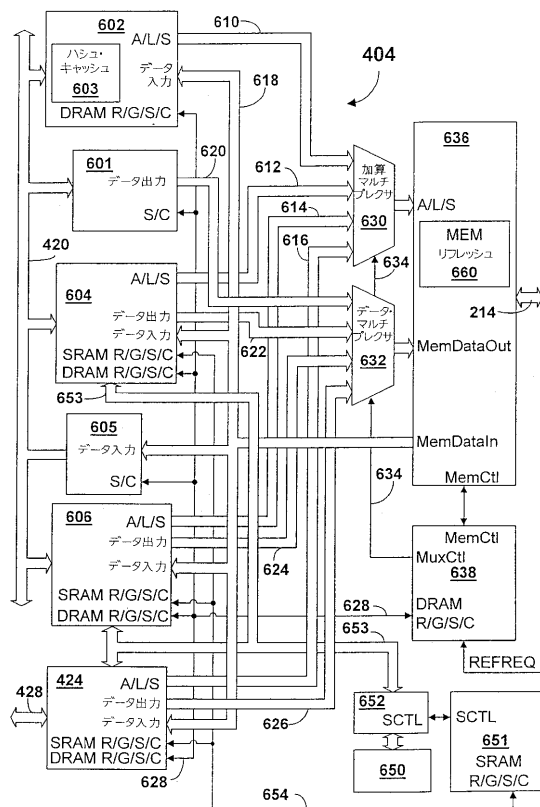
570

```

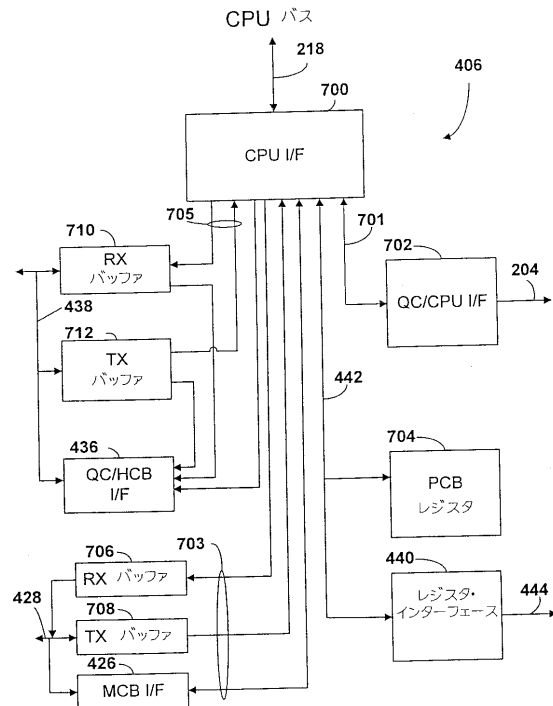
TXPORT3POLL
IF ((TXNEWCNT+1 != TPCOUNT) ||
(TXCTCNT+1 != TPCOUNT)), THEN
{
1) IF (!BUF_AVAIL[0]* && (!TXPRMSK[3] &&
(TXMEMCYC[3] || TXCTACTCYC[3] || TXCTCYC[3]])), THEN
(IF (WTPRIORITY),
THEN TXPORTBUF3 = PORTWT[3],
ELSE TXPORTBUF3 = TPCOUNT;
TXPRMSK[3] = 1; TXINCCNTBY[0] = 1);
...
7) IF (!BUF_AVAIL[6]* && (!TXPRMSK[27] &&
(TXMEMCYC[27] || TXCTACTCYC[27] || TXCTCYC[27]])), THEN
(IF (WTPRIORITY),
THEN TXPORTBUF27 = PORTWT[27],
ELSE TXPORTBUF27 = TPCOUNT +
BITSUM(TXINCCNTBY[5:0]);
TXPRMSK[27] = 1; TXINCCNTBY[6] = 1);
8) (SAME EQUATION 8 AS IN STATE 564);
9) TPCOUNT = TPCOUNT +
BITSUM(TXINCCNTBY[6:0])
}

```

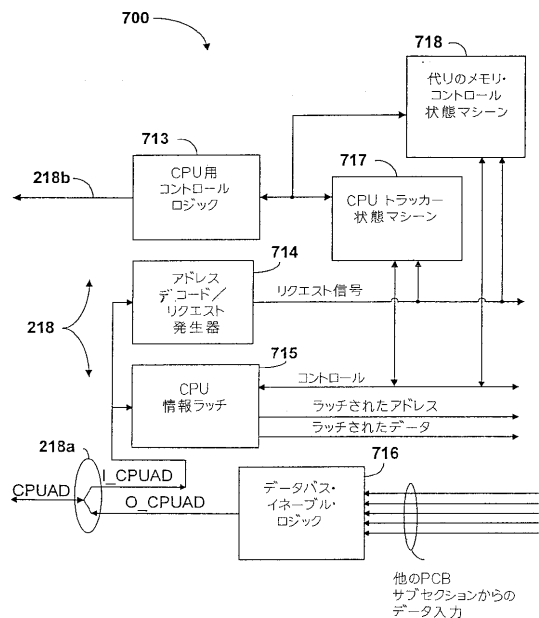
【図 2 4】



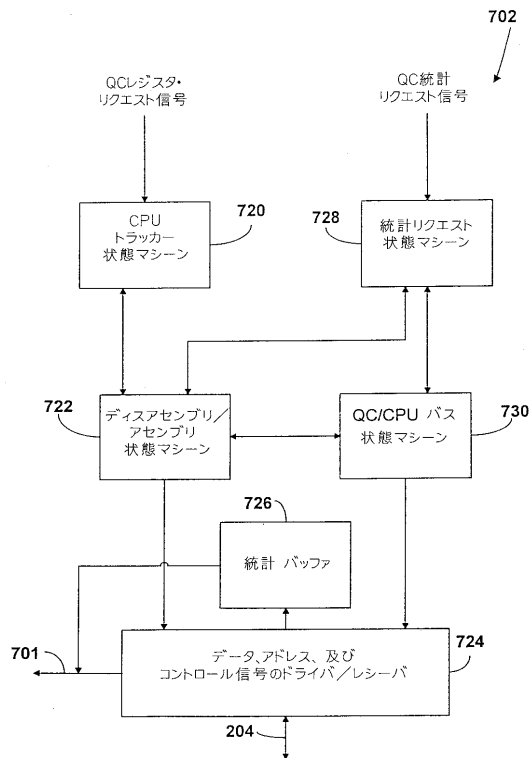
【図 2 5】



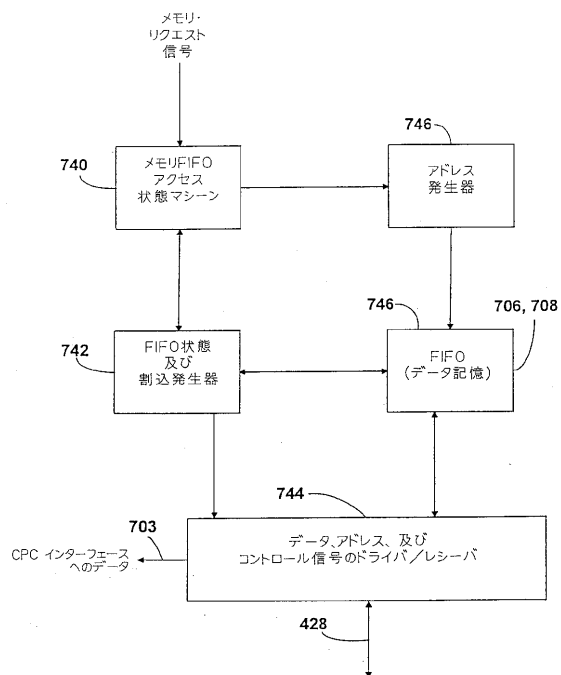
【図 26】



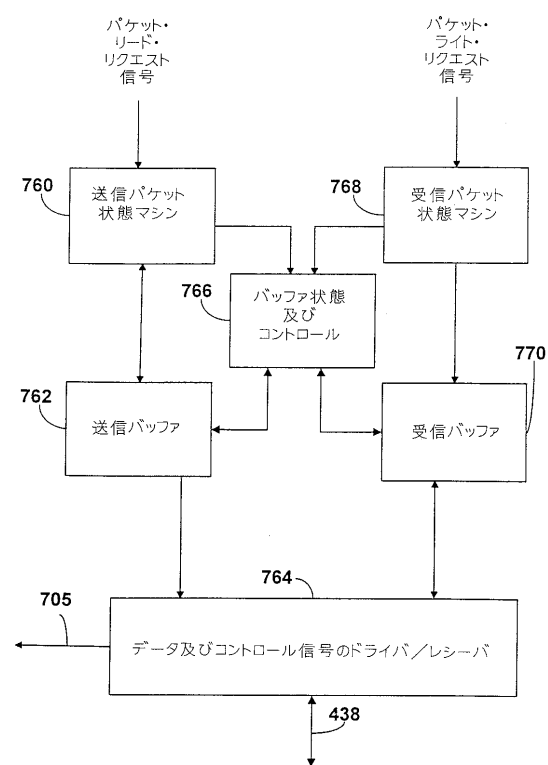
【図 27】



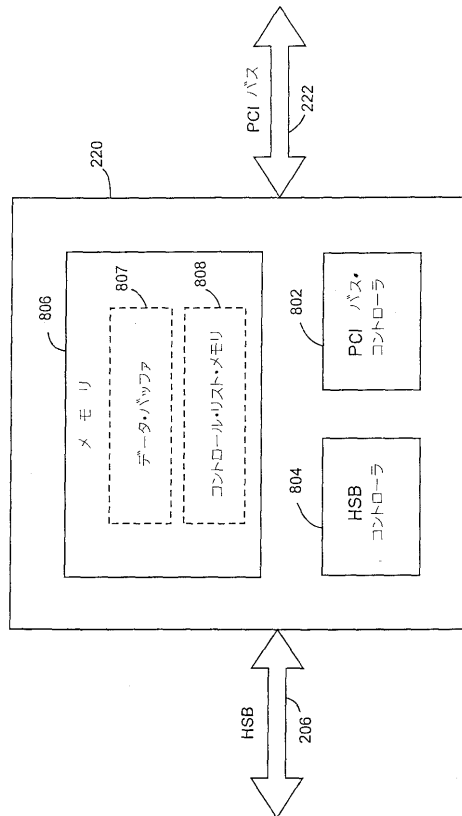
【図 28】



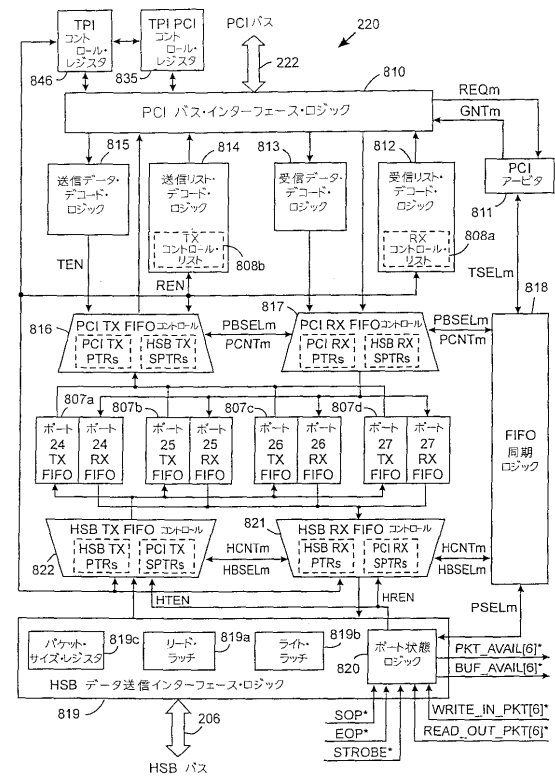
【図 29】



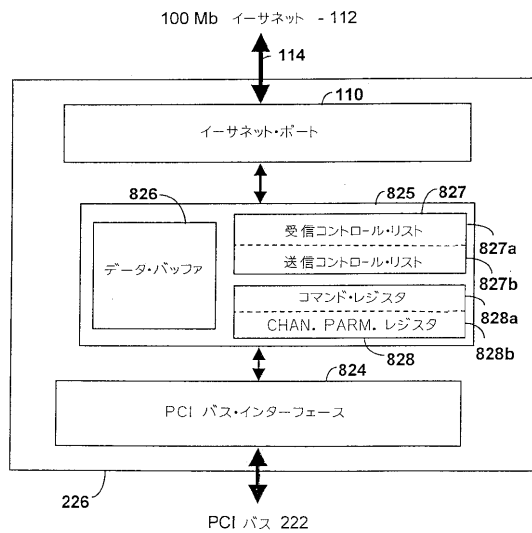
【図 30】



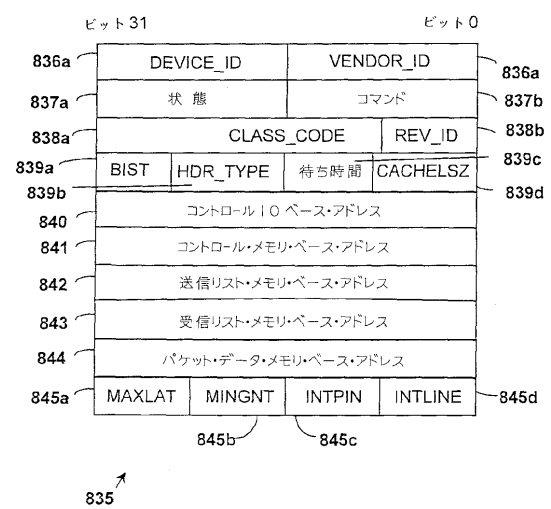
【図 31】



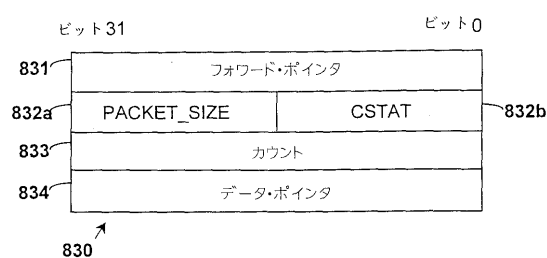
【図 32】



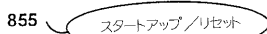
【図 34】



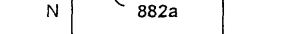
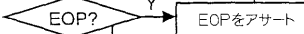
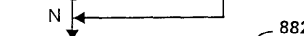
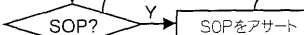
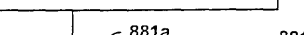
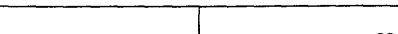
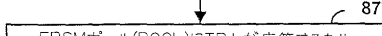
【図 33】



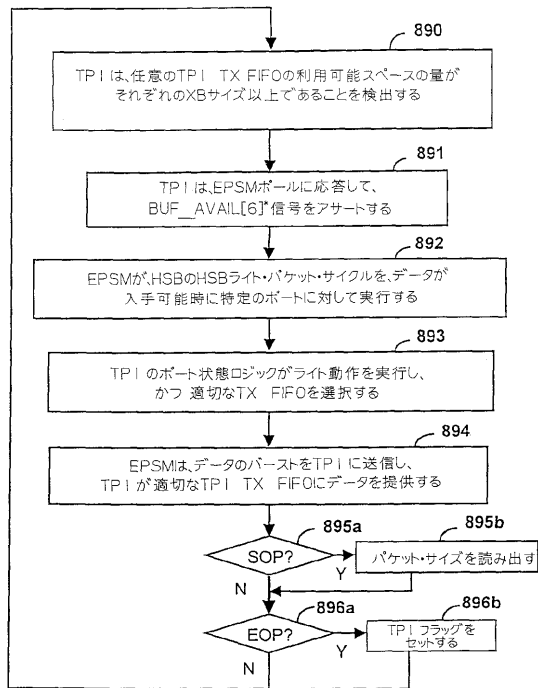
【图 3 6】



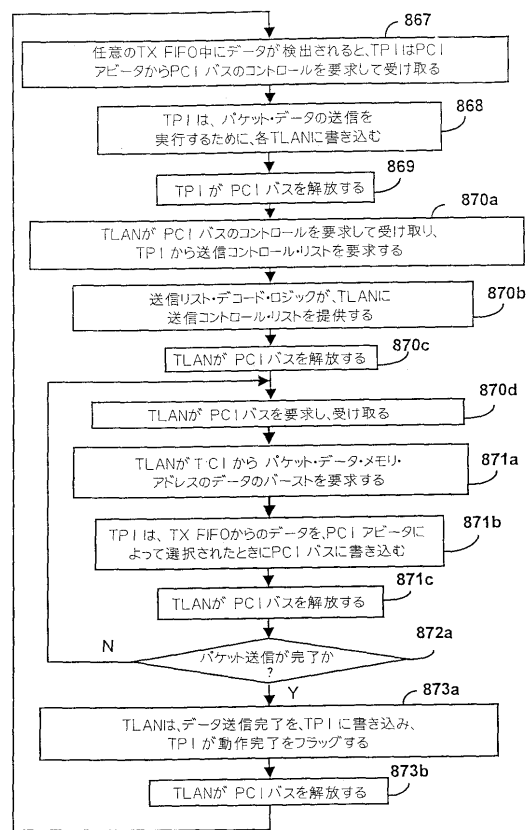
【图 3 8】



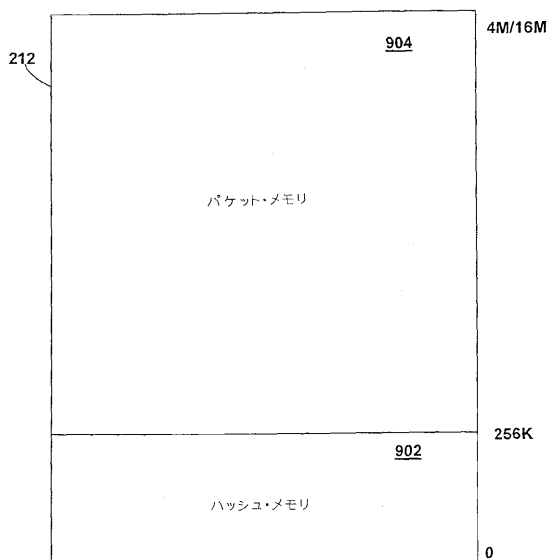
【図 39】



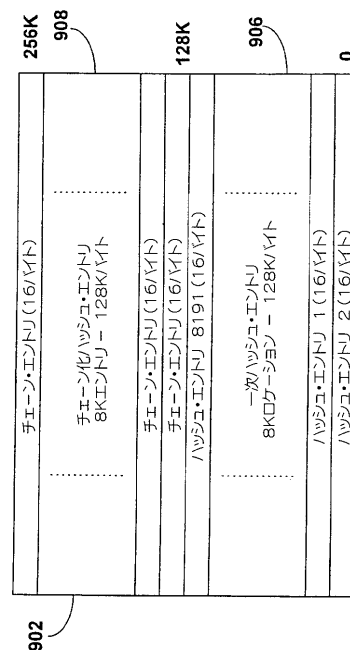
【図 40】



【図 41】



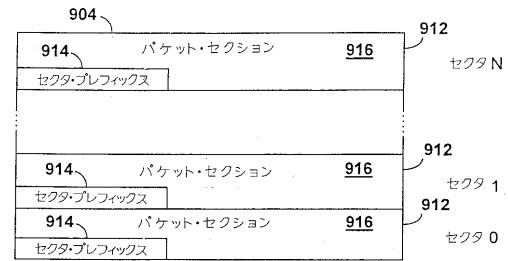
【図 42】



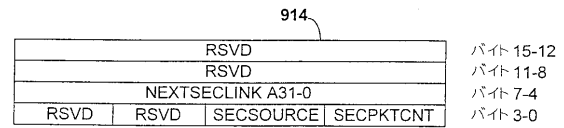
【 図 4 3 】

リンク A31-24	リンク A23-16	リンク A15-8	リンク A7-4,0000	F-C
VLAN バイト 3	VLAN バイト 2	VLAN バイト 1	VLAN バイト 0	B-8
コントロール / AGE	ポート番号			7-4
アドレス/バイト 3	アドレス/バイト 2	アドレス/バイト 1	アドレス/バイト 0	3-0

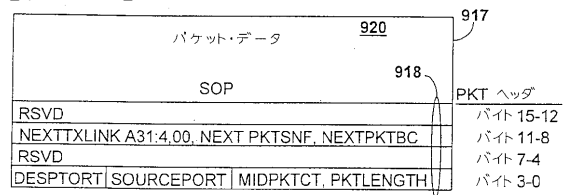
【 図 4 4 】



【图 4 5】



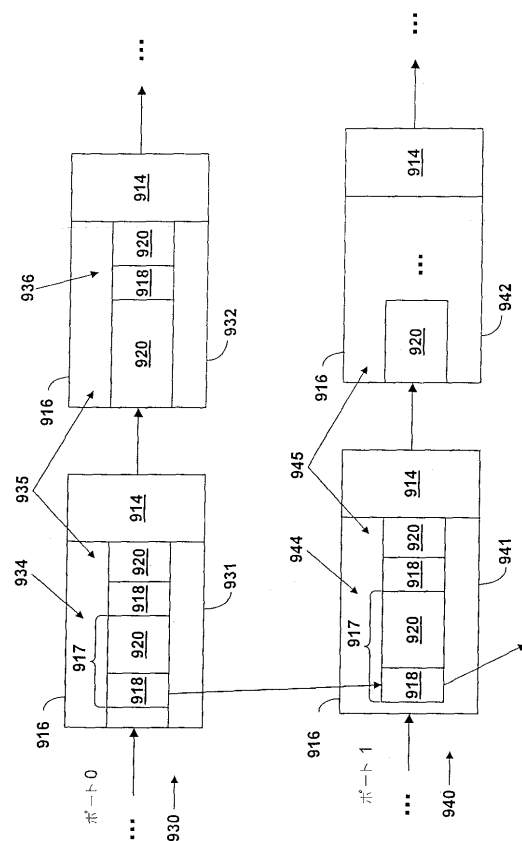
【图 4 6】



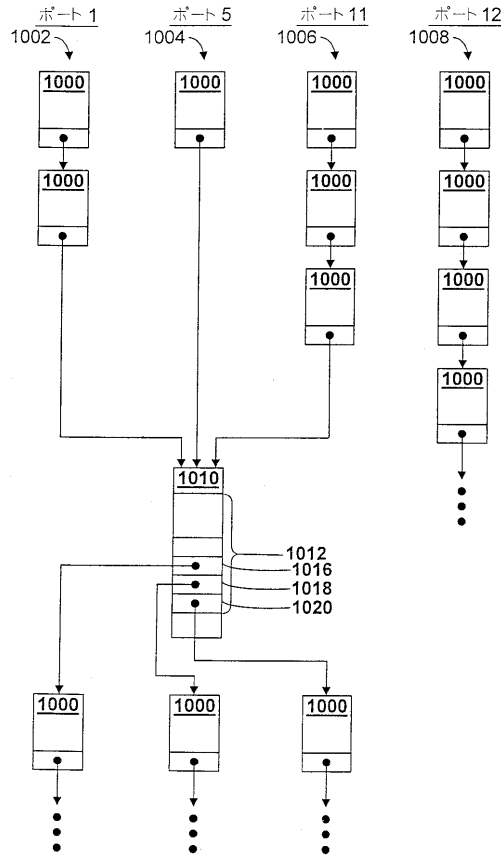
【图 4 7】

922	RSVD		バイト 127-124
	バイト 28 NEXTLINK A31:4.00, NEXTPKTSNF, NEXT PKTBC		バイト 123-120
			
	バイト 1 NEXTLINK A31:4.00, NEXTPKTSNF, NEXT PKTBC		バイト 15-12
	バイト 0 NEXTLINK A31:4.00, NEXTPKTSNF, NEXT PKTBC		バイト 11-8
	BC_PORTS BIT MAP		バイト 7-4
	SOURCEPORT	0, PKTLENGTH	バイト 3-0

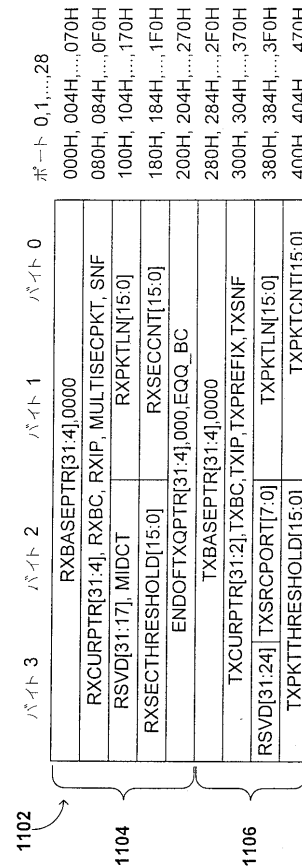
【图 48】



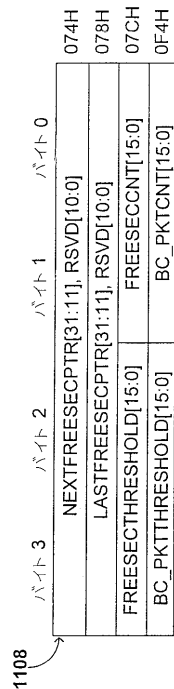
【図 49】



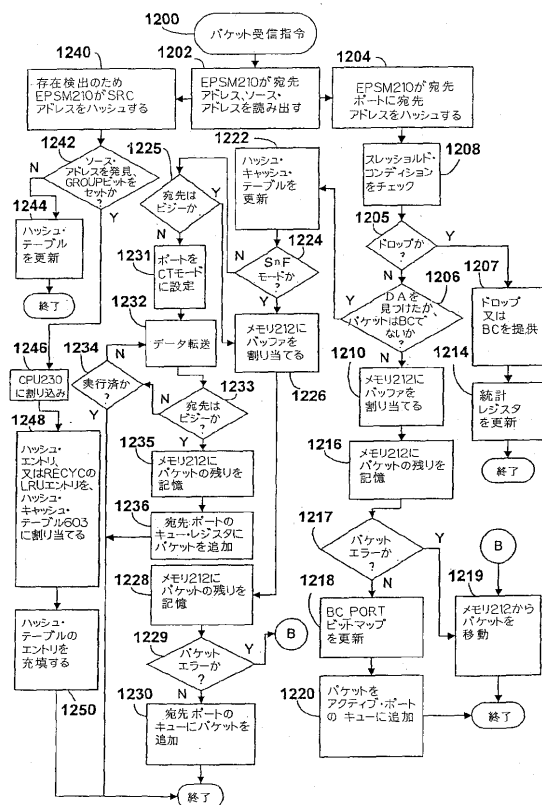
【図 50】



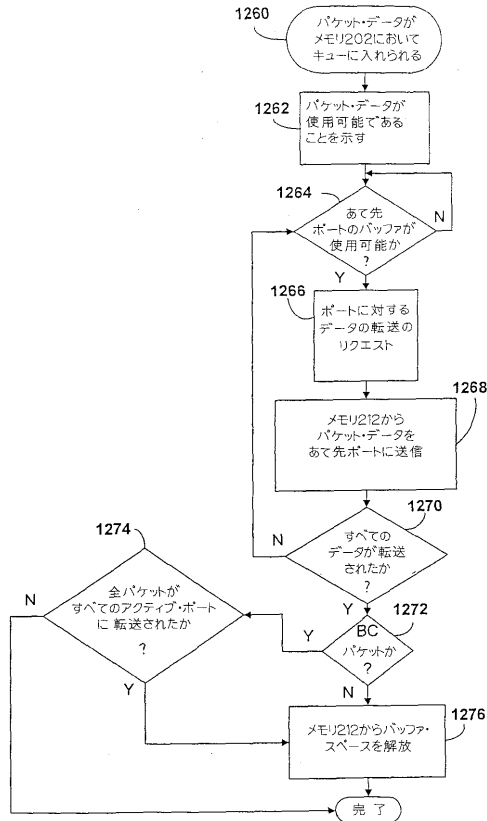
【図 51】



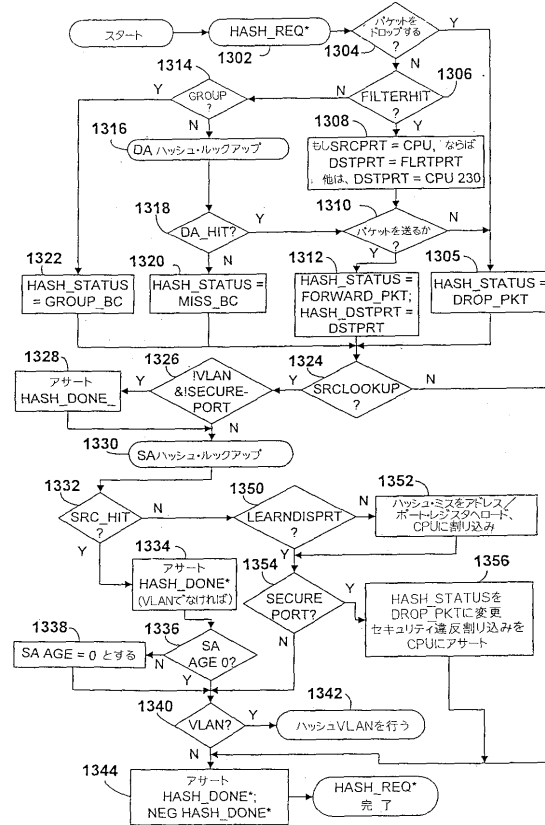
【図 52】



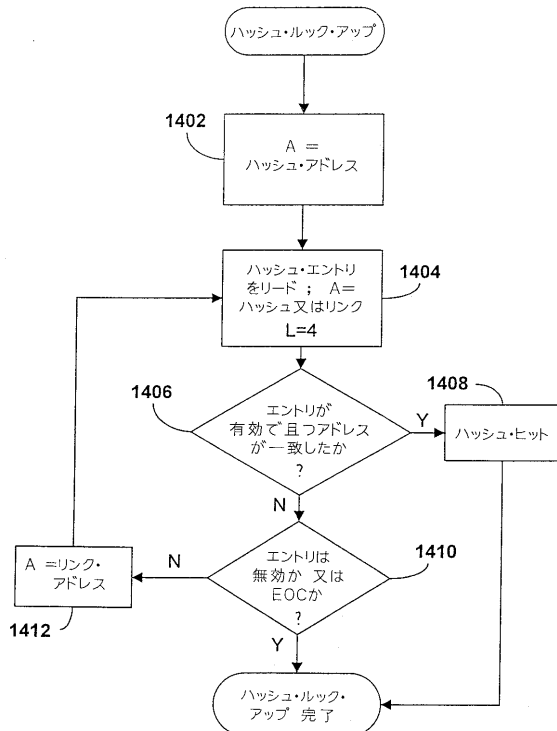
【図 5 3】



【図 5 4】



【図 5 5】



フロントページの続き

- (74)代理人 100071124
弁理士 今井 庄亮
- (74)代理人 100076691
弁理士 増井 忠武
- (74)代理人 100075236
弁理士 栗田 忠彦
- (74)代理人 100075270
弁理士 小林 泰
- (74)代理人 100096068
弁理士 大塚 住江
- (74)代理人 100107696
弁理士 西山 文俊
- (72)発明者 ウィリアム・ジェイ・ウォーカー
アメリカ合衆国テキサス州77070, ヒューストン, ミルズ・リバー 13154
- (72)発明者 デイリー・ビー・コズアー
アメリカ合衆国テキサス州77388, スプリング, フォーレスト・エルムズ・ドライブ 18406
- (72)発明者 パトリシア・イー・ハレスキー
アメリカ合衆国テキサス州77070, ヒューストン, ケイン・クリーク・コート 16106
- (72)発明者 デール・ジェイ・メイヤー
アメリカ合衆国テキサス州77070, ヒューストン, ムーアクリーク 11819
- (72)発明者 マイケル・エル・ウィットコウスキー
アメリカ合衆国テキサス州77375, トムボール, エイヴンプレイス・ロード 16223

審査官 矢頭 尚之

(58)調査した分野(Int.Cl., DB名)

H04L 12/44

H04L 12/66