(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第5094482号
(P5094482)
(45)発行日　平成24年12月12日(2012.12.12)　(24)登録日　平成24年9月28日(2012.9.28)

(51)Int.Cl.　F I
H04L 12/56 (2006.01)　H04L 12/56　300D

請求項の数 6　(全 21 頁)

(21)出願番号　特願2008-58703(P2008-58703)
(22)出願日　平成20年3月7日(2008.3.7)
(65)公開番号　特開2009-218743(P2009-218743A)
(43)公開日　平成21年9月24日(2009.9.24)
審査請求日　平成23年2月25日(2011.2.25)

(73)特許権者　000001007
キヤノン株式会社
東京都大田区下丸子３丁目３０番２号
(74)代理人　100076428
弁理士　大塚　康徳
(74)代理人　100112508
弁理士　高柳　司郎
(74)代理人　100115071
弁理士　大塚　康弘
(74)代理人　100116894
弁理士　木村　秀二
(74)代理人　100130409
弁理士　下山　治
(74)代理人　100134175
弁理士　永川　行光

最終頁に続く

(54)【発明の名称】処理装置及びその処理方法

(57)【特許請求の範囲】
【請求項１】
受信されたＩＰパケットを処理する処理装置であって、
第１の記憶手段を制御する制御手段と、
フラグメントされたＩＰパケットの受信状況を示すデータが第２の記憶手段から前記第１の記憶手段へ転送されている間に、前記ＩＰパケットのヘッダが正常に受信されたか否かを判断し、その判断の結果に応じて前記第１の記憶手段に記憶されたデータの更新を前記制御手段に指示する処理手段と、を有し、
前記処理手段は、前記受信されたＩＰパケットの識別子を前記制御手段に通知し、前記制御手段が前記第１の記憶手段に記憶されていないと判断したデータを前記第２の記憶手段から前記第１の記憶手段へ転送している間に、前記ＩＰパケットのヘッダが正常に受信されたか否かを判断することを特徴とする処理装置。
【請求項２】
前記処理手段は、前記受信されたＩＰパケットの識別子を前記制御手段に通知し、前記制御手段が前記フラグメントされたＩＰパケットの受信状況を示すデータを前記第２の記憶手段から前記第１の記憶手段へ転送している間に、前記ＩＰパケットのヘッダが正常に受信されたか否かを判断することを特徴とする請求項１に記載の処理装置。
【請求項３】
前記制御手段は、前記フラグメントされたＩＰパケットの、フラグメントされる前のパケットにおける位置に応じた順序で、前記第２の記憶手段から前記第１の記憶手段にデー

タを転送することを特徴とする請求項１又は２に記載の処理装置。
【請求項４】
前記処理手段は、前記フラグメントされたＩＰパケットのうち、正常に受信されたＩＰパケットを示すデータが記憶されるように、前記第１の記憶手段に記憶されたデータの更新を前記制御手段に指示することを特徴とする請求項１乃至３の何れか１項に記載の処理装置。
【請求項５】
受信されたＩＰパケットを処理する処理装置の処理方法であって、
制御手段が、第１の記憶手段を制御する制御工程と、
処理手段が、フラグメントされたＩＰパケットの受信状況を示すデータが第２の記憶手段から前記第１の記憶手段へ転送されている間に、前記ＩＰパケットのヘッダが正常に受信されたか否かを判断し、その判断の結果に応じて前記第１の記憶手段に記憶されたデータの更新を前記制御手段に指示する処理工程と、を有し、
前記処理工程では、前記受信されたＩＰパケットの識別子を前記制御工程に通知し、前記制御工程が前記第１の記憶手段に記憶されていないと判断したデータを前記第２の記憶手段から前記第１の記憶手段へ転送している間に、前記ＩＰパケットのヘッダが正常に受信されたか否かを判断することを特徴とする処理装置の処理方法。
【請求項６】
コンピュータを請求項１乃至４の何れか１項に記載の処理装置として機能させるためのプログラム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、ＩＰ（Internet protocol）通信を利用するアプリケーション機器や、ＩＰ通信制御機器におけるプロトコル処理に関し、特に、複数のフラグメントパケットに分割して送信されたＩＰパケットの受信技術に関する。また、ＩＰプロトコル処理の高速化をＴＯＥ（TCP/IP Offload Engine）技術によって実現する組み込み機器に好適である。
【背景技術】
【０００２】
プロトコル処理の中でプロセッサ処理量が大きい処理の１つにＩＰリアセンブルがある。ＩＰ通信では、送信元或いは伝送経路上で１つのＩＰパケットを複数のＩＰパケットに分割して転送し、受信側で分割元の１つのパケットを復元して受信するメカニズムがある。１つのＩＰパケットを複数のＩＰパケットに分割して転送する処理はＩＰフラグメント（IP Fragmentation）と呼ばれている。一方、分割された複数のＩＰパケットを受信し、元のＩＰパケットに復元する処理はＩＰリアセンブル（IP Reassembly）と呼ばれている。ＩＰリアセンブルは、ＲＦＣ７９１に仕様が公開されている。
【０００３】
また、ＩＰリアセンブルには、ＲＦＣ７９１に記載の処理方法と、ＲＦＣ８１５に記載の処理方法の２つのアルゴリズムが知られている。本発明は、前者の方法に関与するものであるので、前者の方法について紹介しておく。尚、ここでは、前者のＩＰリアセンブル処理方法を、ビットマップテーブル方式と呼称する。
【０００４】
ビットマップテーブル方式によるＩＰリアセンブルでは、受信側が送信元ＩＰパケットのペイロードデータの８オクテット分を１ビットに割り当てたビットテーブルを用意する。ＩＰデータグラムは最大長が６５５３５オクテットであるため、用意するビットテーブルは、約８キロビットの長さが必要である。そして、到着する各フラグメントパケットのペイロードデータを保存しながら、フラグメントデータのオフセット位置と長さに相当する、該ビットテーブルのビットをセットしていくことで、フラグメントデータの受信状況を管理する。その後、送信元のＩＰパケットのペイロード長さに相当するビットテーブル上のビットが全てセットされると、分割元のＩＰパケットの再組み立てに必要なデータが

揃ったことを意味し、ＩＰリアセンブル処理の完了となる。

【０００５】

近年、ネットワーク通信が急速に高速化しており、既に民生用にギガビットネットワーク対応のイーサネット（登録商標）製品が普及している。更に、現在ＩＥＥＥ８０２．３ワーキンググループにおいて４０メガビット／秒、或いは１００メガビット／秒の伝送速度を実現可能な規格が標準化されようとしている。こうしたネットワーク通信の高速化に伴い、ＩＰ通信を利用するアプリケーションも送受信するデータ量も増大しており、通信端末機器にとって通信プロトコル処理にかかる負荷が高くなってきている。

【０００６】

従来、ＩＰ通信のプロトコル処理は、ＴＣＰ／ＩＰ（Transmission Control Protocol/Internet Protocol）プロトコルスタックというソフトウェアで実現されることが多い。ＩＰ通信機能を有する組み込み機器においても、ＩＰプロトコル処理をソフトウェアで実装する場合が多い。しかし、小規模な組み込み機器では、製造コストや電力消費量の面で有利である、動作周波数が低いＣＰＵを搭載することが多いため、必要なソフトウェアによるプロトコル処理のプロセッサ負荷が高くなってしまう。

【０００７】

そのため、従来の処理では高い通信性能を実現することが困難になってきている。また、通信処理にかかる処理負荷が原因で、アプリケーション処理にＣＰＵリソースを十分に割り当てられないことも問題となる。

【０００８】

このような問題への対策として、プロセッサの通信処理負荷を軽減し、高速なＩＰ通信を実現するＴＯＥ（TCP/IP Offload Engine）と総称される技術がある。このＴＯＥとは、アプリケーションを実行するプロセッサとは別の処理手段により、ＴＣＰ／ＩＰプロトコル処理の高速化を図る技術である。

【０００９】

ＴＯＥでは、プロトコル処理全体又は一部をハードウェア処理（集積回路）で実行して高速化するケースが多い。また、組み込み機器においては、ＴＯＥをＬＳＩ化してチップに内蔵する実装も行われている。

【特許文献１】特開２００７－２７４０５６号公報

【発明の開示】

【発明が解決しようとする課題】

【００１０】

上述したビットマップテーブル方式のＩＰリアセンブルでは、分割前のＩＰパケットの最大パケットサイズに対応するために必要なテーブルサイズは１Ｋバイト（８１９２バイト）である。また、ＩＰリアセンブルの同時並行処理数と同数のビットマップテーブルが必要となる。通信の高速化により単位時間あたりの受信ＩＰパケット数が増加すると、並行処理するＩＰリアセンブルも増加し、ビットマップテーブルに必要なメモリサイズが増加してしまう。

【００１１】

一方、上述のＴＯＥ技術によってハードウェア処理でＩＰリアセンブルの高速化を実現するためには、アクセス遅延が小さいオンチップメモリ上にビットテーブルを形成することが望ましい。しかしながら、並行処理可能な数に見合う大きなメモリ容量が必要となり、メモリコストが高価であるオンチップメモリでは、メモリコストが高くなってしまう。一方、小容量のメモリでは並行処理可能な数に制限が生じることが問題となる。

【００１２】

このビットデータでフラグメントデータの到着状況を管理する先行技術として、例えば特許文献１がある。特許文献１では、ＩＰフラグメントの分割サイズを狭い範囲に限定することで、フラグメントパケットに暗黙的な断片番号を付与し、フラグメントパケットの受信状況の管理を単純化して高速化している。

【００１３】

この方法では、予めＩＰパケットを分割するサイズを限定しているので、フラグメントパケットの受信状況を管理するビット幅を８Ｋビットよりも小さくすることが可能であるため、ＩＰリアセンブルに必要なメモリサイズを縮小することができる。しかしながら、分割サイズを狭い範囲に制限するため、任意サイズのフラグメントパケットのＩＰリアセンブルを実行するには不適である。

【００１４】

本発明は、フラグメントされたＩＰパケットのリアセンブル処理を高速化し、並行処理にかかるメモリコストを減少させることを目的とする。

【課題を解決するための手段】

【００１５】

本発明は、受信されたＩＰパケットを処理する処理装置であって、第１の記憶手段を制御する制御手段と、フラグメントされたＩＰパケットの受信状況を示すデータが第２の記憶手段から前記第１の記憶手段へ転送されている間に、前記ＩＰパケットのヘッダが正常に受信されたか否かを判断し、その判断の結果に応じて前記第１の記憶手段に記憶されたデータの更新を前記制御手段に指示する処理手段と、を有することを特徴とする。

【発明の効果】

【００１７】

本発明によれば、フラグメントされたＩＰパケットのリアセンブル処理を高速化し、並行処理にかかるメモリコストを減少させることができる。

【発明を実施するための最良の形態】

【００１８】

以下、図面を参照しながら発明を実施するための最良の形態について詳細に説明する。

【００１９】

［第１の実施形態］

図１は、第１の実施形態におけるネットワーク通信装置の構成の一例を表すブロック図である。図１に示すように、ＣＰＵ１０２、ＲＯＭ１０３、ＲＡＭ１０４がシステムバス１０１に接続されている。ここで、ＣＰＵ１０２は、ＲＯＭ１０３に格納されたシステムプログラムをＲＡＭ１０４に読み出して実行する。ＲＡＭ１０４は、システムプログラムを実行時に使用される主記憶装置である。

【００２０】

１０５に示すブロックは、本発明に係るＩＰプロトコル処理装置であり、システムバス１０１に接続されている。ＩＰプロトコル処理装置１０５は、ＣＰＵ１０２によって実行されるアプリケーションの通信に必要なＴＣＰ／ＩＰプロトコル処理、及びネットワーク１１７へのデータ送受信を実行する。

【００２１】

ＩＰプロトコル処理装置１０５において、バスブリッジ回路１０７は、内部のローカルバス１０６をシステムバス１０１に接続するもので、ローカルバス１０６とシステムバス１０１間のデータ転送が可能である。そして、ＩＰプロトコル処理装置１０５と、ＣＰＵ１０２、ＲＯＭ１０４、ＲＡＭ１０４で構成される外部のシステムとは、それぞれのバス回路が相互に接続されて通信データや制御データの入出力をバス転送によって行える仕組みになっている。

【００２２】

また、ＴＣＰ／ＩＰプロトコル処理を実行するための２つのローカルプロセッサ１０８、１０９、ローカルＲＡＭ１１０、データ転送を行うためのＤＭＡコントローラ（ＤＭＡＣ）１１１がローカルバス１０６に接続されている。更に、ＩＰリアセンブルのビットテーブル処理を実行するリアセンブルビットマップコントローラ（ＲＢＭＣ）１１２、プロトコル処理のタイマー処理に利用する通信タイマー１１３がローカルバス１０６に接続されている。更に、プロトコル処理の様々な管理情報の格納と検索を行うための連想メモリであるＣＡＭ（Content Addressable Memory）１１４がローカルバス１０６に接続されている。更に、暗号化通信処理に必要な暗号化／復号化の計算処理を実行する暗号処理部１

１５、ネットワーク１１７に対してデータ送受信を行う通信制御部１１６がローカルバス１０６に接続されている。

【００２３】

ＩＰプロトコル処理装置１０５は、ＩＰプロトコル処理を複数のプロセッサ処理による並列処理で実行する。更に、１１１から１１５の各ハードウェア処理部を利用し、ＴＣＰ／ＩＰプロトコルのパケット送受信と、送信フロー制御や輻輳制御、通信エラー制御、更にＩＰｓｅｃやＳＳＬ／ＴＬＳ等の暗号通信プロトコル処理を実行する。

【００２４】

また、ローカルプロセッサ１０８、１０９が実行するプログラムはＲＯＭ１０３に保存されており、ＩＰプロトコル処理装置１０５の起動時にローカルＲＡＭ１０９上にコピーされる。ローカルプロセッサ１０８、１０９は、ローカルＲＡＭ１０９からプログラムを読み出して実行する。

【００２５】

尚、図１では、２個のローカルプロセッサ１０８、１０９が図示されているが、ソフトウェア処理を実行するプロセッサの個数は２つに限定されるものではない。

【００２６】

ＤＭＡＣ１１１は、ＩＰプロトコル処理装置１０５の内部又は外部のメモリデバイスやハードウェアモジュール間における送受信データや制御データの転送処理を行う。ＲＢＭＣ１１２は、ビットマップ方式（ＲＦＣ７９１）のＩＰリアセンブルアルゴリズムで使用するビットテーブルのビット操作処理や、ＲＡＭ１０４との間でビットテーブルのデータ転送を行う。

【００２７】

通信タイマー１１３は、ＩＰリアセンブルにおけるタイムアウト時間の計測や、ＴＣＰプロトコル通信における再送処理、確認応答送信などの各種タイマーで必要な時間計測のように、ＩＰプロトコル処理における様々なタイムアウト処理に利用される。暗号処理部１１５は、ＩＰＳｅｃやＳＳＬ（Secure Socket Layer）／ＴＬＳ（Transport Layer Security）などの暗号通信プロトコル処理で処理負荷が大きい暗号化／復号化の演算処理を実行する。

【００２８】

通信制御部１１６は、ネットワーク１１７のＭＡＣ処理（伝送メディア制御処理）や、フレームデータの送受信を行う機能が実装される。ネットワーク１１７は、例えばイーサネット（登録商標）のような有線ネットワークであるが、無線ネットワークや、光ファイバーネットワーク等であっても良い。

【００２９】

ここで、ＲＢＭＣ１１２の実施構成の一例を、図２を参照して説明する。図２は、第１の実施形態におけるＲＢＭＣの構成の一例を表すブロック図である。

【００３０】

ＲＢＭＣ１１２は、レジスタ２０１、リソース管理部２０２、内部メモリ（ＳＲＡＭ）２０３、ビット処理部２０４、ＤＭＡ機能部２０５で構成される。まず、レジスタ２０１はＲＢＭＣ１１２の設定や起動などの制御インタフェースであり、複数のレジスタセットである。そして、このレジスタ２０１に対してローカルプロセッサ１０８、１０９が読み書きを行い、ＲＢＭＣ１１２を設定、制御する。

【００３１】

リソース管理部２０２は、複数のビットテーブルのデータ管理を行う。管理情報データは、図１に示すローカルＲＡＭ１０９に記憶される。ビット処理部２０４は、内部メモリ２０３に格納されたビットテーブルデータの任意の範囲に対してビット操作処理を行う。このビット操作処理には、任意長のビット列に対して全ビットを１にセット、全ビットを０にクリア、値が１であるビット数のカウント、全ビットを１にセットしたときに０から１に変化したビット数のカウントが含まれる。

【００３２】

内部メモリ２０３は、この例ではアクセス遅延が非常に小さいメモリとして実装される。ＲＢＭＣ１１２は、内部メモリ２０３上にビット処理部２０４がビット操作処理を行うビットテーブルデータを一時格納する。ビット処理部２０４のビット操作処理において、内部メモリ２０３に対してデータの読み書きが実行される。

【００３３】

ＤＭＡ機能部２０５は、リソース管理部２０２によって制御され、内部メモリ２０３とＲＢＭＣ１１２の外部にあるメモリとの間でビットテーブルのデータ転送を実行する。

【００３４】

以上の構成において、ＩＰリアセンブルのビットテーブルをＲＡＭ１０４に確保する。そして、ＲＢＭＣ１１２がビットテーブルのビット操作処理を行うときには対象のビットテーブルデータを内部メモリ２０３に一時格納し、一時格納したビットテーブルデータに対してビット操作処理を実行する。

【００３５】

ここで、ＩＰリアセンブルのビットテーブルのビット操作処理を、図３を参照して説明する。ＩＰリアセンブルで使用するビットテーブルは、先頭ビットから順番に各ビットが分割元ＩＰパケットのペイロードの先頭から８オクテット毎にマッピングされる。即ち、各ビットは、ＩＰリアセンブルによって対応する８オクテットが受信済みであるか否かを記憶しておくフラグである。例えば、図３に示すビットテーブル３０１で、値が１であるビットが対応する８オクテットは、既に受信済みであることを表している。

【００３６】

ＩＰリアセンブルでは、受信したフラグメントパケットのＩＰヘッダの内容から、そのフラグメントパケットが運ぶフラグメントデータについて、分割前のＩＰパケットのペイロードにおけるフラグメントオフセットとフラグメントサイズを得ることができる。この情報を元に、受信したフラグメントデータが対応するビット範囲の各ビットを、受信済みを表す１に書き込んでいく。また、最後尾のフラグメントパケットを受信すると、分割元のＩＰパケットの全長を得ることができ、ビットテーブル上で必要なビット幅が確定する。従って、分割元ＩＰパケットのペイロード長分のビット列が全て値１で埋まったとき、分割元のＩＰパケットを構成する全てのフラグメントデータを受信したことになる。

【００３７】

図３に示す例では、３０１のビットテーブルで、３０３は受信したフラグメントデータの分割元ＩＰパケットペイロードにおけるオフセット位置を表し、３０４はフラグメントデータに対応するビット幅を表している。ここで、ビットテーブルはフラグメントされたＩＰパケットの受信状況を管理するためのテーブルである。

【００３８】

ビット処理部２０４は、フラグメントデータを受信すると、３０３の位置から３０４の幅だけ各ビットに１を書き込む。その結果、３０１のビットデータは３０２に示すように更新される。３０５に表す部分は、新規に０から１に変化したビットであることを表している。ビット処理部２０４はこのようなビット操作処理を実行する。

【００３９】

また、ビット処理部２０４は値が０から１に変化したビット数をカウントし、ＲＢＭＣ１１２のレジスタ２０１に新規に受信したフラグメントデータサイズを設定する。

【００４０】

上述したように、ビット処理部２０４は、アクセス遅延が小さい内部メモリ２０３上のデータに対してビット操作処理を行う。これにより、ＩＰリアセンブルを高速化することができるため、内部メモリ２０３は、例えばＳＲＡＭを実装することが考えられる。

【００４１】

しかし、アクセス遅延の小さい内部メモリ２０３に、並行処理する全てのＩＰリアセンブルのビットテーブルを置くことは、並行処理可能なＩＰリアセンブル数に応じたメモリ容量が必要となり、メモリコストが非常に高くなってしまう。

【００４２】

従って、全てのビットテーブルはＲＡＭ１０４に確保し、ＩＰリアセンブルで処理対象となるビットテーブルデータだけを内部メモリ２０３にコピーし、一時格納することで、内部メモリ２０３のメモリコストを抑えることができる。

【００４３】

第１の実施形態では、並行処理する各ＩＰリアセンブルに対してＩＰリアセンブルＩＤという識別子を付与し、ビットテーブルに対応付けて管理する。そして、ＲＡＭ１０４上のビットテーブルのうち、ビット操作処理の実行頻度が高いビットテーブルを内部メモリ２０３に一時格納しておくようにする。

【００４４】

図４は、ＩＰリアセンブルのビットテーブルをＩＰリアセンブル識別子で管理する状態を示す図である。この例では、ある時点において、ＲＡＭ１０４上に処理途中であるＮ個のビットテーブルが確保されており、内部メモリ２０３には４個のビットテーブルが一時格納されている状態を示している。

【００４５】

図４に示すように、ＲＡＭ１０４には、４０５～４１１に示すＮ個のビットテーブルが任意のメモリアドレスに配置されている。このうち、ＩＰリアセンブルＩＤが１、３、Ｍ、Ｎ－１であるビットテーブル４０６、４０８、４０９、４１１が、それぞれ内部メモリ２０３の４０１、４０２、４０３、４０４に示す場所に一時格納されている。

【００４６】

即ち、この内部メモリ２０３へのデータの格納方法は、ビットテーブルサイズを単位とするフルアソシエイティブ方式である。図４に示す内部メモリ２０３の使用例では、ある時点において、ＩＰリアセンブルＩＤが１、３、Ｍ、Ｎ－１であるビットテーブルが内部メモリ２０３に置かれている。従って、その時点から次に受信するフラグメントパケットが、これらのＩＰリアセンブルの何れかであれば、ＲＡＭ１０４から内部メモリ２０３にビットテーブルを転送する必要がない。

【００４７】

しかしながら、内部メモリ２０３にビットテーブルを格納していないＩＰリアセンブルのフラグメントパケットを受信した場合、処理対象であるビットテーブルをＲＡＭ１０４から内部メモリ２０３にコピーする必要がある。また、例えば内部メモリ２０３にビットテーブルを格納する空きが無い場合、一時格納しているビットテーブルの何れかをＲＡＭ１０４に書き戻し、空いた場所に次の処理対象となるビットテーブルを一時格納する必要が生じる。

【００４８】

そのため、ＲＢＭＣ１１２内のリソース管理部２０２において、リアセンブルＩＤ毎にビットテーブルの管理を行う。リソース管理部２０２がＩＰリセンブルＩＤ毎に保持するビットテーブル管理情報７０は、次の７０１～７１０を含む。７０１はリアセンブルＩＤである。７０２はこのビットテーブルを使用するＩＰリアセンブルにおいて最後に処理したＩＰリアセンブルの処理時刻の記録である。７０３は最後に処理したＩＰリアセンブルの時間間隔（到着間隔）の記録である。７０４はＲＡＭ１０４にあるビットテーブルのメモリアドレスである。７０５は内部メモリ２０３にビットテーブルが一時格納されているか否かを示すフラグである。７０６～７１０は内部メモリ２０３にビットテーブルを一時格納しているときに使用するパラメータである。

【００４９】

７０６は内部メモリ２０３上のビットテーブルデータのアドレスであり、７０７はその内部メモリ２０３上でのビットテーブルデータのサイズを示す。７０８はビットテーブルデータが内部メモリ２０３で変更されたかを示すフラグである。そして、７０９はビットテーブルの変更されたビット範囲の先頭位置を意味するオフセットアドレスであり、７１０は変更範囲を示すデータサイズである。各ビットテーブルの管理情報をローカルＲＡＭ１１０に保存し、ＲＢＭＣ１１２のリソース管理部２０２によって更新される。

【００５０】

リソース管理部２０２は、ＲＡＭ１０４からビットテーブルデータを転送して内部メモリ２０３に格納するとき、ビットテーブルデータを一時格納する空きがあるか否かを、ビットテーブル管理情報をチェックして判定する。また、空きがない場合には、ビットテーブル管理情報にある最後に処理したＩＰリアセンブルの時間間隔７０３を比較し、最も長い到着間隔であるＩＰリアセンブルのビットテーブルを選択する。選択されたビットテーブルは内部メモリ２０３からＲＡＭ１０４にデータを書き戻し、内部メモリ２０３に空き領域を作成する。

【００５１】

このようにして、ＩＰリアセンブルでビットテーブルを内部メモリ２０３に一時格納していない場合にも、ＲＡＭ１０４から内部メモリ２０３にビットテーブルデータを転送して一時格納することを実行する。

【００５２】

尚、内部メモリ２０３及びＲＡＭ１０４間のデータ転送は、ＲＢＭＣ１１２内のＤＭＡ機能部２０５が実行する。

【００５３】

次に、ＩＰプロトコル処理装置１０５で実行されるＩＰリアセンブルの全体的な処理の流れを、図５を参照して説明する。図５は、ＩＰリアセンブルにおけるＤＭＡＣ１１１、ＲＢＭＣ１１２、ローカルプロセッサ１０８の時間的な処理シーケンスを示す図である。尚、図５に示す例は、受信ＩＰパケットがフラグメントパケットであることを前提としたシーケンスである。

【００５４】

通信制御部１１６がＩＰパケットを受信すると、はじめにＤＭＡＣ１１１がＳ５０１で受信ＩＰパケットの転送を開始し、通信制御部１１６からＲＡＭ１０４へ転送が行われる。また、受信ＩＰパケットの先頭からＩＰヘッダまでの部分に相当するサイズのデータはＲＡＭ１０４に転送されると共に、ローカルＲＡＭ１１０へも同時転送される。そして、Ｓ５０２でＩＰパケットの転送が、先頭のＭＡＣヘッダ（リンク層フレームのヘッダ）とＩＰヘッダ部分に相当するデータサイズまで終わると、ローカルプロセッサ１０８へ受信通知（Ｓ５０３）を行う。更に、ＤＭＡＣ１１１は通知を行った後も、Ｓ５０４で引き続きペイロード部分の転送を続行する。

【００５５】

一方、ローカルプロセッサ１０８は、Ｓ５０５でＤＭＡＣ１１１からの通知を受けて、ＩＰパケットの受信処理を開始する。即ち、ローカルプロセッサ１０８は受信ＩＰパケット全体がＲＡＭ１０４に転送されるのを待たずに、ＩＰパケットの受信処理を開始する。次に、ローカルプロセッサ１０８はＳ５０６でローカルＲＡＭ１１０に転送されたＭＡＣヘッダとＩＰヘッダの内容を解析する。ここで、宛先ＭＡＣアドレス、宛先ＩＰアドレスが受信可能であるかチェックする。更に、Ｓ５０６でＩＰパケットヘッダの内容に基づきフラグメントパケットであるかを判定する。

【００５６】

次に、Ｓ５０６でフラグメントパケットであることが判定されると、Ｓ５０７で、その受信フラグメントパケットを対象とするＩＰリアセンブルのリアセンブルＩＤの検索処理を行う。続いて、Ｓ５０８でＲＢＭＣ１１２に対してＳ５０７で検索されたリアセンブルＩＤに対応するビットテーブルのプリロードを起動（Ｓ５０９）する。このプリロードは、処理対象のビットテーブルデータを内部メモリ２０３に一時格納する処理である。

【００５７】

起動指示により、ＲＢＭＣ１１２は、Ｓ５１０で内部メモリ２０３にそのリアセンブルＩＤのビットテーブルが格納されているかを調べ、格納されていない場合はＲＡＭ１０４に確保されているビットテーブルを一時格納する。即ち、Ｓ５１０でＲＡＭ１０４と内部メモリ２０３の間でビットテーブルのデータ転送を行う。

【００５８】

一方、ローカルプロセッサ１０８は、ＲＢＭＣ１１２のプリロードを起動した後、その

処理完了を待たずに、Ｓ５１１で受信ＩＰパケット（即ちフラグメントパケット）のＩＰヘッダチェックサムの検証を実行する。ＩＰヘッダのチェックサムが正しいならば、ＩＰヘッダの内容が正しいので、そのＩＰパケットは受信可能であると判定する。

【００５９】

次に、Ｓ５１２でＲＢＭＣ１１２に対して、ビットテーブルへのビット操作処理を起動（Ｓ５１３）する。そして、ローカルプロセッサ１０８は、Ｓ５１４のビット操作処理の完了を待機する。

【００６０】

起動指示により、ＲＢＭＣ１１２は、Ｓ５１４で内部メモリ２０３に一時格納しておいたビットテーブルデータに対するビット操作処理を実施する。この処理でビット値を０から１に変更したビット数分のデータサイズ、即ち新規に受信したフラグメントサイズを、レジスタ２０１に設定し、処理完了を通知（Ｓ５１５）する。

【００６１】

ここで、ローカルプロセッサ１０８が通知を受信すると、ＩＰリアセンブル処理で新規に受信したフラグメントデータサイズを得ることができる。

【００６２】

一方、ＤＭＡＣ１１１がＳ５１６で、受信ＩＰパケットの全体をＲＡＭ１０４に転送し終えると、ローカルプロセッサ１０８に転送完了を通知（Ｓ５１７）する。これにより、ローカルプロセッサ１０８がＳ５１８でＤＭＡ転送完了通知を受信すると、ローカルプロセッサ１０８はこのＩＰパケットの受信処理を終了する。

【００６３】

以上がＩＰプロトコル処理装置１０５で実行されるＩＰリアセンブルの全体的な処理の流れである。

【００６４】

第１の実施形態では、Ｓ５１１でＩＰヘッダチェックサム検証後に受信ＩＰパケットが受信可能であるかを判定している。しかし、その判定後のＳ５１４でＩＰリアセンブルのビットテーブルのビット操作処理に至った段階で、ＲＢＭＣ１１２が内部メモリ２０３にビットテーブルデータを格納するのではない。前段のＳ５０８でＲＢＭＣ１１２にビットテーブルのプリロードを起動している。つまり、ローカルプロセッサ１０８がＳ５１１でＩＰヘッダチェックサムの検証を行っている間にＲＢＭＣ１１２はＳ５１０でプリロード処理を実行する。

【００６５】

このような処理シーケンスにより、ローカルプロセッサ１０８のＩＰパケット受信処理とＲＢＭＣ１１２のプリロード処理を並列的に実行する。これにより、ＲＢＭＣ１１２が内部メモリ２０３にビットテーブルデータを一時格納する際に、内部メモリ２０３とＲＡＭ１０４の間でデータ転送が発生したとしても、その時間的なオーバーヘッドは無視することができる。

【００６６】

Ｓ５１１の結果、ＩＰパケットが受信可能であるならば、Ｓ５１２でＩＰリアセンブルを実行する。ここで、既に内部メモリ２０３にはビットテーブルデータが格納されているため、Ｓ５１４のビット操作処理は高速に実行される。もし、Ｓ５１１でＩＰパケットが受信不可と判定した場合でも、Ｓ５１０では内部メモリ２０３にビットテーブルデータを確保しただけであるため、ビットテーブルの状態を変更していないことは明白である。

【００６７】

次に、上述したＩＰリアセンブルの処理シーケンスにおけるＳ５０６～Ｓ８０８で実行されるローカルプロセッサ１０８の処理を、図６及び図７を参照して詳しく説明する。

【００６８】

図６は、ＩＰｖ４（ＩＰバージョン４）パケットのＩＰヘッダのフォーマットを示す図である。ＩＰパケットがフラグメントパケットであるか否かは、フラグメント継続フラグ６０６とフラグメントオフセット６０９で判定する。フラグメント継続フラグ６０６が１

であれば、このＩＰパケットがフラグメントパケットであり、更に後続するフラグメントデータを運ぶ別のＩＰパケットが存在していることを示す。

【００６９】

また、フラグメント継続フラグ６０６が０であるが、フラグメントオフセット６０９が０でなければ、このＩＰパケットはフラグメントデータの最後尾を運ぶフラグメントパケットである。このような判定方法により、受信したＩＰパケットがフラグメントパケットであるかを判定する。

【００７０】

また、同じ分割元パケットを構成するフラグメントパケットは、ＩＰヘッダ内のＩＰ識別子６０５、プロトコル６１１、送信元ＩＰアドレス６１３、宛先ＩＰアドレス６１４の４つのフィールド値が同一でなければならない。

【００７１】

第１の実施形態では、これらの４つのパラメータとＩＰリアセンブルを識別するためのリアセンブルＩＤを関連付けてＣＡＭ１１４に検索エントリデータを保存する。この検索エントリデータの保存は、新規のＩＰリアセンブルを開始し、新規のリアセンブルＩＤを決定したときに実行する。また、フラグメントパケット受信時のＩＰリアセンブルの検索処理は、ＣＡＭ１１４に対して、ＩＰヘッダから得られる４つのフィールドを検索キーとしてリアセンブルＩＤを検索することである。この検索の結果、リアセンブルＩＤが見つからなかった場合は、新規のＩＰリアセンブルを開始し、新規のＩＰリアセンブルを決定する。

【００７２】

図７は、ローカルプロセッサ１０８の処理（図５のＳ５０５～Ｓ５０８）を示すフローチャートである。まずＳ７０１で、受信ＩＰパケットのＭＡＣヘッダとＩＰヘッダの各種パラメータを読み出して取得する。次に、Ｓ７０２で宛先ＭＡＣアドレスと宛先ＩＰアドレスが受信可能であるかを判定する。判定した結果、受信できないアドレスである場合は、その受信パケットを破棄し、この処理を終了する。

【００７３】

また、宛先ＭＡＣアドレスと宛先ＩＰアドレスが共に受信可能なアドレスである場合はＳ７０３へ処理を進める。このＳ７０３では、Ｓ７０１で取得したＩＰヘッダ内のフラグメント継続フラグ６０６とフラグメントオフセット６０９の値により、このＩＰパケットがフラグメントパケットであるか否かを調べる。ここで、フラグメントパケットでは無い場合、ＩＰリアセンブルを実行しないで、通常のＩＰパケット受信の処理を実行する。

【００７４】

一方、Ｓ７０３でフラグメントパケットであると判定した場合はＳ７０４へ処理を進め、フラグメントパケットのリアセンブルＩＤを取得するために検索処理を実行する。尚、この検索処理については上述した通りであるので、その説明は省略する。

【００７５】

続いて、Ｓ７０５でリアセンブルＩＤが検索できたか否かを判定し、検索できなかった場合はＳ７０６へ処理を進める。このＳ７０６では、フラグメントパケットは新規にＩＰリアセンブルを開始する必要があるので、新しいリアセンブルＩＤを決定する。そして、パケットのＩＰ識別子、プロトコル、送信元ＩＰアドレス、宛先ＩＰアドレスの４つの値に新しいリアセンブルＩＤを関連付けて、ＣＡＭ１１４に検索エントリを登録しておく。更に、新規のリアセンブルＩＤに対応するビットマップテーブルをＲＡＭ１０４上に確保する。続くＳ７０７では、Ｓ７０６で新規に決定したリアセンブルＩＤと、ＲＡＭ１０４上に確保したビットテーブルのメモリアドレスをＲＢＭＣ１１２のレジスタ２０１に設定する。このとき、ＲＢＭＣ１１２のリソース管理部２０２が新しいＩＰリアセンブルのビットテーブル管理情報をローカルＲＡＭ１１０に作成する。

【００７６】

また、Ｓ７０５でリアセンブルＩＤが検索できた場合及びＳ７０７での処理を終了した場合はＳ７０８へ処理を進める。このＳ７０８では、リアセンブルＩＤ、ＩＰヘッダから

取得したフラグメントオフセット６０９、ＩＰパケットのペイロードの長さであるフラグメントデータサイズ、ＲＢＭＣ１１２のレジスタ２０１に設定してプリロード処理を起動する。

【００７７】

次に、上述したＩＰリアセンブルの処理シーケンスにおけるＳ５１０で実行されるＲＢＭＣ１１２のビットテーブルのプリロード処理を、図８を用いて説明する。

【００７８】

図８は、ビットテーブルのプリロード処理を示すフローチャートである。まずＳ８０１では、レジスタ２０１に指定にされたリアセンブルＩＤに対応するビットテーブルが内部メモリ２０３に一時格納されているかを調べる。ここで、内部メモリ２０３に格納されていれば、この処理を終了する。

【００７９】

また、指定されたリアセンブルＩＤのビットテーブルが内部メモリ２０３に格納されていなければＳ８０２へ処理を進め、内部メモリ２０３にビットテーブルを格納する空きがあるか否かを調べる。もし空きがあればＳ８０７へ処理を進め、空きが無ければＳ８０３へ処理を進める。

【００８０】

Ｓ８０３では、内部メモリ２０３にビットテーブルを格納しているＩＰリアセンブルについて、各々が最後に処理されたときの前回処理からの時間間隔を比較し、最も時間間隔が長いＩＰリアセンブルを選択する。ここでは、リソース管理部２０２がＩＰリセンブルＩＤ毎に保持するビットテーブル管理情報７０を参照し、内部メモリ格納フラグ７０５がオンであるビットテーブル管理情報の中で、最後に処理したＩＰリアセンブルの時間間隔７０３の値で比較する。即ち、ビットテーブル管理情報の７０３の値が最も大きいリアセンブルＩＤを選択する。

【００８１】

次に、Ｓ８０４で、選択したリアセンブルＩＤの内部メモリ２０３上のビットテーブルが変更されたかを調べる。ここでは、選択されたリアセンブルＩＤのビットテーブル管理情報で７０８の内部メモリデータ変更フラグがオンであるかをチェックする。もし、変更されていればＳ８０５へ処理を進め、そうでなければＳ８０７へ処理を進める。

【００８２】

Ｓ８０５では、選択されたリアセンブルＩＤの内部メモリ２０３上のビットテーブルが変更された範囲だけをＲＡＭ１０４上にあるビットテーブルに書き戻す処理を行う。ＲＢＭＣ１１２は、ビット操作処理の際、ビットテーブル管理情報の７０９のデータ変更オフセットと、７１０のデータ変更サイズを更新する。そのため、書き戻す必要のあるビットテーブルの変更部分は７０９から７１０の範囲である。ここでは、内部メモリ２０３上のビットテーブルの変更部分だけをＲＡＭ１０４上のビットテーブルに書き込んで反映する。

【００８３】

次に、Ｓ８０６では、ＲＡＭ１０４に書き戻したビットテーブルについて、管理情報の７０５から７１０の全てのフィールドをクリアし、Ｓ８０７へ処理を進める。

【００８４】

このＳ８０７では、指定されたリアセンブルＩＤのビットテーブルデータを内部メモリ２０３に読み込む。内部メモリ２０３上の読み込む場所は、Ｓ８０２で見つかった空きであるか、Ｓ８０３で選択したリアセンブルＩＤのビットテーブルがあった場所である。

【００８５】

次に、Ｓ８０８で、レジスタ２０１に設定されたフラグメントオフセットとフラグメントデータサイズの範囲のビットテーブルデータを優先して先に、内部メモリ２０３に転送する。この部分以外のビットテーブルデータは後から転送する。これは、ビットテーブルの全体を内部メモリ２０３に転送する前に、ビット操作処理が起動された場合に、直ちにビット操作処理を行うことを可能にするためである。以上のようにして、ビットテーブル

のプリロード処理が実行される。

【００８６】

次に、上述したＩＰリアセンブルの処理シーケンスにおけるＳ５１４で実行されるＲＢＭＣ１１２のビットテーブルのビット操作処理を、図９を用いて説明する。

【００８７】

図９は、ビットテーブルのビット操作処理を示すフローチャートである。まずＳ９０１で、通信タイマー１１３から現在のタイマー値を取得して現在時刻を求める。また、指定されたリアセンブルＩＤのビットテーブル管理情報中の最後に処理したＩＰリアセンブルの処理時刻７０２との差分値を求め、１個前に処理実行したＩＰフラグメントパケットとの時間間隔を取得する。そして、管理情報の７０２と７０３のフィールドを更新する。

【００８８】

次に、Ｓ９０２で、指定されたリアセンブルＩＤのビットテーブルが内部メモリ２０３上に一時格納されているかを調べる。ここでは、ＲＢＭＣ１１２のリソース管理部２０２がそのリアセンブルＩＤのビットテーブル管理情報の内部メモリ格納フラグ７０５がオンであるかを調べ、オンであるときはＳ９０６へ処理を進め、オフであるならばＳ９０３へ処理を進める。

【００８９】

このＳ９０３では、ＲＡＭ１０４にあるビットテーブルに対して直接ビット操作を施す処理モードとなる。次に、Ｓ９０４では、ＲＡＭ１０４上のそのビットテーブルに対してレジスタ２０１に指定されたフラグメントオフセットと、フラグメントデータサイズに対応するビット範囲を１にセットする。そして、Ｓ９０５では、ＲＡＭ１０４上のビットテーブルの中で０から１に変更されたビットの数をレジスタ２０１に設定して、Ｓ９１０へ処理を進める。

【００９０】

一方、Ｓ９０６では、内部メモリ２０３上のビットテーブルデータに対してビット操作を施すモードとなる。次に、Ｓ９０７では、内部メモリ２０３上のそのビットテーブルに対してレジスタ２０１に指定されたフラグメントオフセットと、フラグメントデータサイズに対応するビット範囲を１にセットする。そして、Ｓ９０８では、ＲＡＭ１０４上のビットテーブル中で０から１に変更されたビットの数をレジスタ２０１に設定する。次に、Ｓ９０９で、内部メモリ２０３上のビットテーブルで変更した部分のデータオフセットとサイズをビットテーブル管理情報の７０９と７１０にそれぞれ記録しておく。そして、Ｓ９１０へ処理を進める。

【００９１】

このＳ９１０では、ビット操作処理の終了をローカルプロセッサ１０８に通知し、この処理を終了する。

【００９２】

尚、Ｓ９０２からＳ９０３へ処理を進める場合、アクセス遅延の大きいＲＡＭ１０４上のビットテーブルに対してビット操作処理を行う。そのため、Ｓ９０２からＳ９０６へ処理を進める、内部メモリ２０３にあるビットテーブルへのビット操作処理に比較して処理にかかる時間が大きくなってしまう。しかし、上述した図５に示すシーケンスのように、ＲＢＭＣ１１２のプリロード処理を実行するので、Ｓ９０２からＳ９０６へ処理を進めることになり、ビット操作処理を高速に行うことが可能である。

【００９３】

本実施形態によれば、ＲＦＣ７９１に記載のＩＰリアセンブル方法において、ＲＢＭＣ１１２がアクセス遅延の小さい内部メモリ２０３にビットテーブルを一時格納してビット操作処理を行えるため、ビット操作処理の高速化を実現できる。

【００９４】

また、受信したフラグメントパケットのＩＰパケット受信処理の実行と、ＲＢＭＣ１１２のプリロード処理が並列処理化されるため、ＲＢＭＣ１１２が内部メモリ２０３にビットテーブルを格納するための時間的なオーバーヘッドを無視可能である。

【００９５】
更に、プリロード処理において、内部メモリ２０３に空きがない場合に、フラグメントパケットの時間的な到着間隔の大きいＩＰリアセンブルのビットマップテーブルをＲＡＭ１０４に書き戻すような制御を行う。それゆえ、フラグメントパケットの到着間隔の短い、即ちＩＰリアセンブルの処理頻度が高いビットテーブルを優先的に内部メモリ２０３に格納しておくことができ、ＲＡＭ１０４と内部メモリ２０３間のビットテーブルのデータ転送を減らすことが可能となる。
【００９６】
［第２の実施形態］
次に、図面を参照しながら本発明に係る第２の実施形態を詳細に説明する。尚、第２の実施形態におけるＩＰプロトコル装置の構成は、第１の実施形態で説明した図１及び図２と同様である。即ち、図１に示すＩＰプロトコル処理装置１０５、図２に示す内部構成は同じであり、リアセンブルビットマップコントローラ（ＲＢＭＣ）１１２も、第１の実施形態の構成と同じとする。
【００９７】
第１の実施形態では、ＲＢＭＣ１１２の内部メモリ２０３に並行して処理を行っている全ビットテーブルの中から、直近に処理されたＩＰリアセンブルの幾つかについてビットテーブルの全体データを内部メモリ２０３に一時格納していた。しかし、第２の実施形態では、内部メモリ２０３に一時格納するデータはＲＡＭ１０４上に確保している全ビットテーブルについて各々の一部データのみを一時格納する。
【００９８】
図１０は、全ビットテーブルの一部分データのみを内部メモリ２０３に格納する状態を示す図である。１００８～１０１４はＲＡＭ１０４にＮ個のＩＰリアセンブルの全ビットテーブルが確保されていることを表している。また、内部メモリ２０３の１００１～１００７には、ビットテーブル１００８～１０１４における一部データのみが格納されていることを表している。
【００９９】
この例では、リアセンブルＩＤが０のビットテーブル１００８の一部データが１００１に格納されており、リアセンブルＩＤがＭのビットテーブル１０１２の一部データが１００５に格納されている。
【０１００】
尚、内部メモリ２０３への格納方式はダイレクトマップ方式とする。即ち、内部メモリ２０３では、各ビットテーブルの一部データの格納場所とサイズを固定に定義し、個々の格納場所に格納する対象データは、ＲＡＭ１０４上に確保された各ビットテーブルの１Ｋオクテットの範囲に限定化する。
【０１０１】
このような内部メモリ２０３の格納方法では、格納するデータサイズを小さくすることで、内部メモリ２０３に必要なメモリサイズを小さくすることが可能である。また、内部メモリ２０３に格納するデータサイズを１回のＩＰリアセンブルに必要となる最大サイズ分を格納するようにする。これにより、ＲＢＭＣ１１２のプリロード処理でＲＡＭ１０４と内部メモリ２０３間で転送するデータサイズを、ビットテーブル全体を転送するよりも小さくすることが可能である。
【０１０２】
ここで、内部メモリ２０３に格納するサイズは、ＩＰプロトコル処理装置１０５が接続するネットワークのＭＴＵ値を元にして決定する。例えば、一般的なイーサネット（登録商標）の場合、ＭＴＵは１５００であるので、１つのＩＰパケットが運ぶペイロードの最大長は、ＩＰヘッダの最小サイズを引いた１４８０である。
【０１０３】
よって、１つのフラグメントパケットが運ぶフラグメントデータは最大長が１４８０となり、必要なビットテーブルデータは１８５ビットである。つまり、各ビットテーブルに

ついて２４バイト（１９２ビット）分のビットデータを格納すれば、１回のフラグメントパケットのＩＰリアセンブルでは十分なデータサイズになる。

【０１０４】

また、第１の実施形態では、ビットテーブル全体を内部メモリ２０３に一時格納しているが、１つのビットテーブルに必要なサイズは１Ｋバイト（８１９２ビット）であるため、約４０倍の個数のビットテーブルを一時格納することが可能とある。

【０１０５】

しかしながら、第１の実施形態では、ＲＢＭＣ１１２のプリロード処理で、既に内部メモリ２０３にビットテーブルが確保されている場合は、ＲＡＭ１０４からビットテーブルデータを転送する必要が無かった。つまり、同じＩＰリアセンブルが、連続的に処理する場合や、単位時間当たりの実行頻度が大きい場合には、ビットテーブルデータの転送回数を減らす効果がある。

【０１０６】

逆に、第２の実施形態の方法は、フラグメントパケットが重複するフラグメントデータを持つことが頻発する状況でない限り、毎回のＩＰリアセンブルにおいてＲＡＭ１０４と内部メモリ２０３へのデータ転送が発生してしまう。

【０１０７】

しかし、ビットテーブルデータの転送サイズが小さく、また、前述したようにＩＰリアセンブルが、ローカルプロセッサ１０８のＩＰパケット受信処理と、ＲＢＭＣ１１２のプリロード処理が並列的に実行される。

【０１０８】

このことから、ＲＡＭ１０４と内部メモリ２０３間のビットテーブルデータ転送にかかる時間的オーバーヘッドは無視でき、ＩＰリアセンブル処理速度の低下にはならない。

【０１０９】

また、同時に並行処理可能なＩＰリアセンブルの数により、アクセス遅延の小さい内部メモリ２０３にかかるメモリコストを低減することが可能になる。

【０１１０】

［第３の実施形態］

次に、図面を参照しながら本発明に係る第３の実施形態を詳細に説明する。尚、第３の実施形態におけるＩＰプロトコル装置の構成は、第１の実施形態で説明した図１と同様である。即ち、図１に示すＩＰプロトコル処理装置１０５の内部構成は同じである。しかし、第１の実施形態で説明したリアセンブルビットマップコントローラ（ＲＢＭＣ）の内部構成が異なる。

【０１１１】

図１１は、第３の実施形態におけるＲＢＭＣの構成の一例を示す図である。１１００は図２に示すＲＢＭＣ１１２に相当する第３の実施形態におけるＲＢＭＣである。ＲＢＭＣ１１００は、レジスタ２０１、リソース管理部２０２、１次内部メモリ１１０１、ビット処理部２０４、ＤＭＡ機能部２０５、２次内部メモリ１１０２で構成される。尚、図２に示す構成と同様なものには同一の符号を付している。

【０１１２】

ＲＢＭＣ１１００の構成において、第１の実施形態との違いは、内部メモリ２０３を、１次内部メモリ１１０１と２次内部メモリ１１０２の２つとしたことである。

【０１１３】

図１２は、第３の実施形態におけるビットテーブルデータの内部メモリへの格納方法を示す図である。ＲＡＭ１０４には、並行して処理を行っている全てのＩＰリアセンブルのビットテーブル４０５～４１１を確保している。そして、２次内部メモリ１１０２には、ＲＡＭ１０４の任意のアドレスに確保されたビットテーブルの内、４個のビットテーブルが一時格納されている。図１２に示す例では、リアセンブルＩＤが０、３、Ｍ、Ｎ－１の４個のビットテーブル４０５、４０８、４０９、４１１が２次内部メモリ１１０２の１２０９～１２１２の場所に格納されている。更に、２次内部メモリ１１０２に一時格納され

ている各ビットテーブルの一部分データ１２０５～１２０８が１次内部メモリ１１０１の１２０１～１２０４に格納されている。

【０１１４】

この格納方法は、１次内部メモリ１１０１への格納は２次内部メモリ１１０２に対するダイレクトマップ方式である。

【０１１５】

また、２次内部メモリ１１０２への格納では、ＲＡＭ１０４上の任意のアドレスに配置されたビットテーブルに対してビットテーブルサイズ単位でのフルアソシエイティブ方式である。

【０１１６】

このようなＲＢＭＣ１１００の構成及びビットテーブルデータの格納方法では、フラグメントパケットの到着間隔が短く、単位時間当たりのＩＰリアセンブル処理の回数が多いビットテーブルを優先して２次内部メモリ１１０２に格納しておく。このようにすることで、ＲＡＭ１０４へのビットテーブルデータの転送を少なくすることが可能となる。

【０１１７】

更に、１次内部メモリ１１０１と２次内部メモリ１１０２の間では、ビットテーブルの中の一部データのみを転送するため、データ転送のオーバーヘッドを極小化できる。

【０１１８】

また、１次内部メモリ１１０１と２次内部メモリ１１０２は、ＲＢＭＣ１１００内部で高速にデータ転送を行うことが可能である。よって、１次内部メモリ１１０１へのビット操作処理により書き込みが行われると同時に、２次内部メモリ１１０２に書き込むようにするライトスルー（write through）制御を行い、１次メモリ及び２次メモリ間のデータ同期制御を簡便化する。

【０１１９】

更に、ＩＰリアセンブルで処理対象となるビットテーブルが２次内部メモリ１１０２に存在しない場合は、ＲＡＭ１０４から処理対象の範囲のデータのみ直接、１次内部メモリ１１０１に直接転送することでプリロード処理を高速化する。このプリロード処理では、上述のライトスルー制御を行う場合、１次内部メモリ１１０１のデータは書き戻す必要がない。

【０１２０】

このように、ビットマップテーブル方式のＩＰリアセンブルで複数のＩＰリアセンブルを並列的に実行する場合にも、ビットテーブルをアクセス遅延の小さいメモリ上に置いてビット操作処理の高速化とメモリに必要な容量を抑えることが可能となる。本発明はＩＰプロトコル通信を高速に実行する通信端末機器や、ＴＯＥにおいて好適である。

【０１２１】

上述した実施形態によれば、ビットテーブルのビット操作処理をアクセス遅延の小さい第２の記憶手段に保持するデータに対して実行することで、ＩＰリアセンブルの高速化を図ることができる。

【０１２２】

また、処理途中である全てのＩＰリアセンブルのビットテーブルを第１の記憶手段に配置し、ビット操作を行うビットテーブルのデータだけをアクセス遅延の小さい第２の記憶手段に置くことで、第２の記憶手段に必要な記憶容量を少なくすることができる。例えば、第１の記憶手段にＳＤＲＡＭ、第２の記憶手段にＳＲＡＭを用い、一般にメモリコストが高価なＳＲＡＭを小容量で実装し、安価なＳＤＲＡＭには少なくとも全ビットテーブルを保存可能な大容量で実装することでメモリコストを抑えることができる。つまり、ＩＰリアセンブルの並行処理可能な数を低メモリコストで拡大することができる。

【０１２３】

更に、ビット操作処理を行うビットテーブルのデータを処理開始の事前に第２の記憶手段に用意し、フラグメントパケットの到着間隔が短いＩＰリアセンブルのビットテーブルデータを優先的に第２の記憶手段に保持する。これにより、第１の記憶手段と第２の記憶

手段の間で発生するデータ転送の時間的なオーバーヘッドを平均的に小さくできるため、ＩＰリアセンブルの処理高速化が可能となる。

【０１２４】

尚、本発明は複数の機器（例えば、ホストコンピュータ、インターフェース機器、リーダ、プリンタなど）から構成されるシステムに適用しても、１つの機器からなる装置（例えば、複写機、ファクシミリ装置など）に適用しても良い。

【０１２５】

また、前述した実施形態の機能を実現するソフトウェアのプログラムコードを記録した記録媒体を、システム或いは装置に供給し、そのシステム或いは装置のコンピュータ（ＣＰＵ若しくはＭＰＵ）が記録媒体に格納されたプログラムコードを読出し実行する。これによっても、本発明の目的が達成されることは言うまでもない。

【０１２６】

この場合、コンピュータ読み取り可能な記録媒体から読出されたプログラムコード自体が前述した実施形態の機能を実現することになり、そのプログラムコードを記憶した記録媒体は本発明を構成することになる。

【０１２７】

このプログラムコードを供給するための記録媒体として、例えばフレキシブルディスク、ハードディスク、光ディスク、光磁気ディスク、ＣＤ－ＲＯＭ、ＣＤ－Ｒ、磁気テープ、不揮発性のメモリカード、ＲＯＭなどを用いることができる。

【０１２８】

また、コンピュータが読出したプログラムコードを実行することにより、前述した実施形態の機能が実現されるだけでなく、次の場合も含まれることは言うまでもない。即ち、プログラムコードの指示に基づき、コンピュータ上で稼働しているＯＳ（オペレーティングシステム）などが実際の処理の一部又は全部を行い、その処理により前述した実施形態の機能が実現される場合である。

【０１２９】

更に、記録媒体から読出されたプログラムコードがコンピュータに挿入された機能拡張ボードやコンピュータに接続された機能拡張ユニットに備わるメモリに書込む。その後、そのプログラムコードの指示に基づき、その機能拡張ボードや機能拡張ユニットに備わるＣＰＵなどが実際の処理の一部又は全部を行い、その処理により前述した実施形態の機能が実現される場合も含まれることは言うまでもない。

【図面の簡単な説明】

【０１３０】

【図１】第１の実施形態におけるネットワーク通信装置の構成の一例を表すブロック図である。

【図２】第１の実施形態におけるＲＢＭＣの構成の一例を表すブロック図である。

【図３】ＩＰリアセンブルにおけるビットテーブルのビット操作処理を説明するための図である。

【図４】ＩＰリアセンブルのビットテーブルをＩＰリアセンブル識別子で管理する状態を示す図である。

【図５】ＩＰリアセンブルにおけるＤＭＡＣ１１１、ＲＢＭＣ１１２、ローカルプロセッサ１０８の時間的な処理シーケンスを示す図である。

【図６】ＩＰｖ４（ＩＰバージョン４）パケットのＩＰヘッダのフォーマットを示す図である。

【図７】ローカルプロセッサ１０８の処理（図５のＳ５０５～Ｓ５０８）を示すフローチャートである。

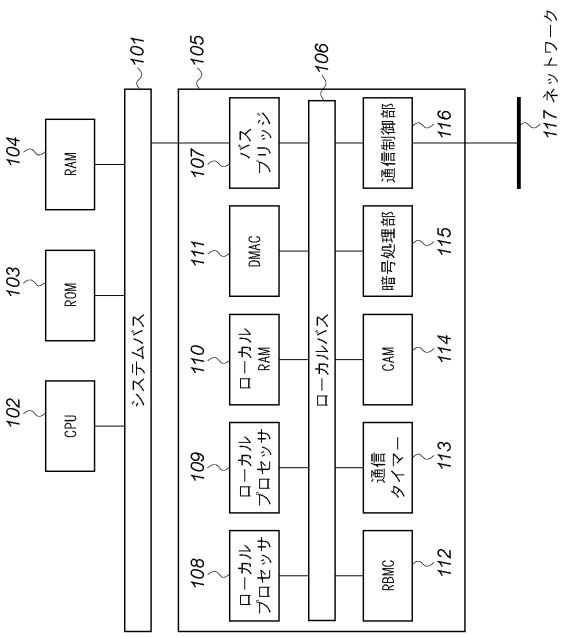
【図８】ビットテーブルのプリロード処理を示すフローチャートである。

【図９】ビットテーブルのビット操作処理を示すフローチャートである。

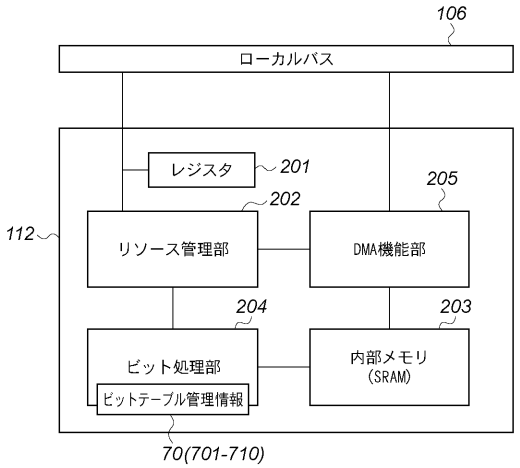
【図１０】全ビットテーブルの一部分データのみを内部メモリ２０３に格納する状態を示す図である。

【図１１】第３の実施形態におけるＲＢＭＣの構成の一例を示す図である。
【図１２】第３の実施形態におけるビットテーブルデータの内部メモリへの格納方法を示す図である。
【符号の説明】
【０１３１】
１０１　システムバス
１０２　ＣＰＵ
１０３　ＲＯＭ
１０４　ＲＡＭ
１０５　ＩＰプロトコル処理装置
１０６　ローカルバス
１０７　バスブリッジ
１０８　ローカルプロセッサ
１０９　ローカルプロセッサ
１１０　ローカルＲＡＭ
１１１　ＤＭＡコントローラ（ＤＭＡＣ）
１１２　リアセンブルビットマップコントローラ（ＲＢＭＣ）
１１３　通信タイマー
１１４　ＣＡＭ
１１５　暗号処理部
１１６　通信制御部
１１７　ネットワーク
２０１　レジスタ
２０２　リソース管理部
２０３　内部メモリ
２０４　ビット処理部
２０５　ＤＭＡ機能部

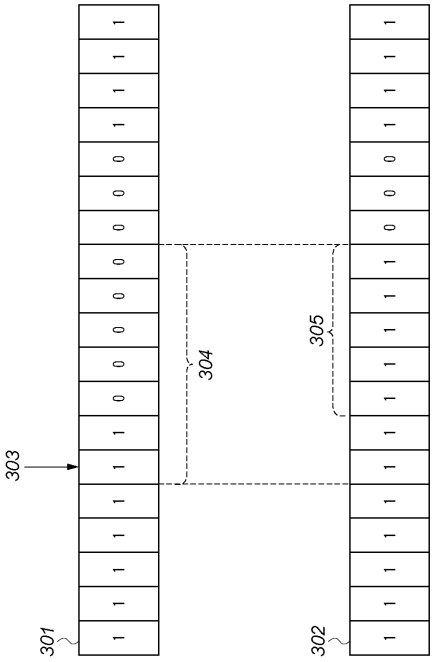
【図１】


101
105
106
117 ネットワーク
104 RAM
103 ROM
102 CPU
システムバス
107 バスブリッジ
111 DMAC
110 ローカルRAM
109 ローカルプロセッサ
108 ローカルプロセッサ
ローカルバス
通信制御部 116
暗号処理部 115
CAM 114
通信タイマー 113
RBMC 112

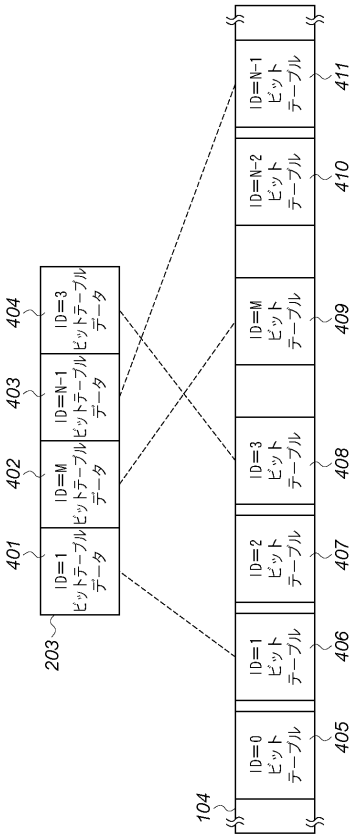
【図２】


106
ローカルバス
112
レジスタ 201
202
リソース管理部
205
DMA機能部
204
ビット処理部
ビットテーブル管理情報
203
内部メモリ
(SRAM)
70(701-710)

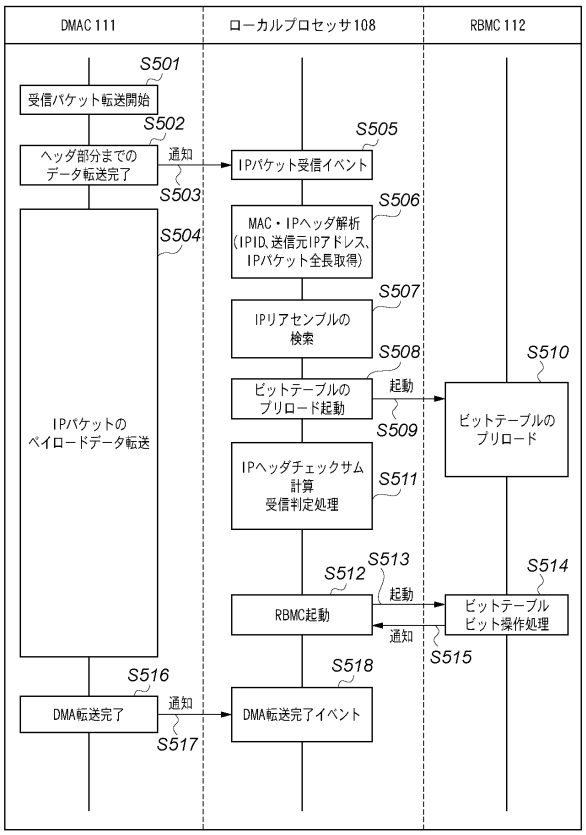
【図３】


303
304
305
301
302

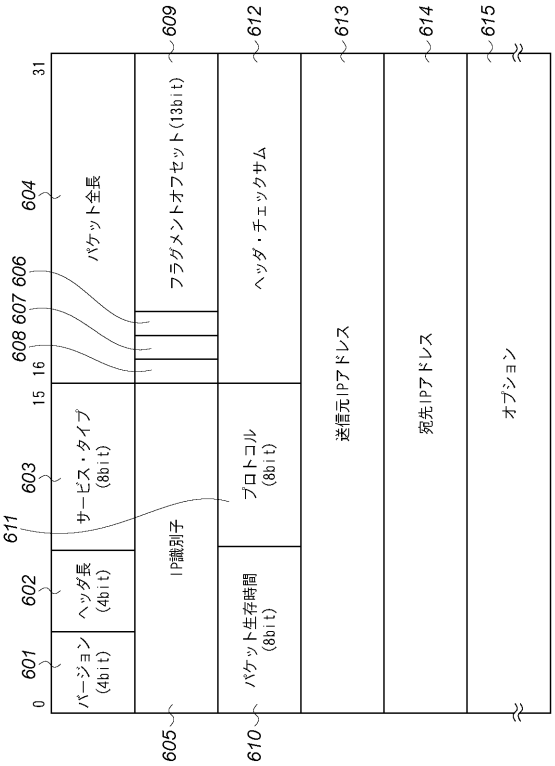
【図４】


404
403
402
401
203
ID=3 ビットテーブルデータ
ID=N-1 ビットテーブルデータ
ID=M ビットテーブルデータ
ID=1 ビットテーブルデータ
ID=N-1 ビットテーブル 411
ID=N-2 ビットテーブル 410
ID=M ビットテーブル 409
ID=3 ビットテーブル 408
ID=2 ビットテーブル 407
ID=1 ビットテーブル 406
ID=0 ビットテーブル 405
104

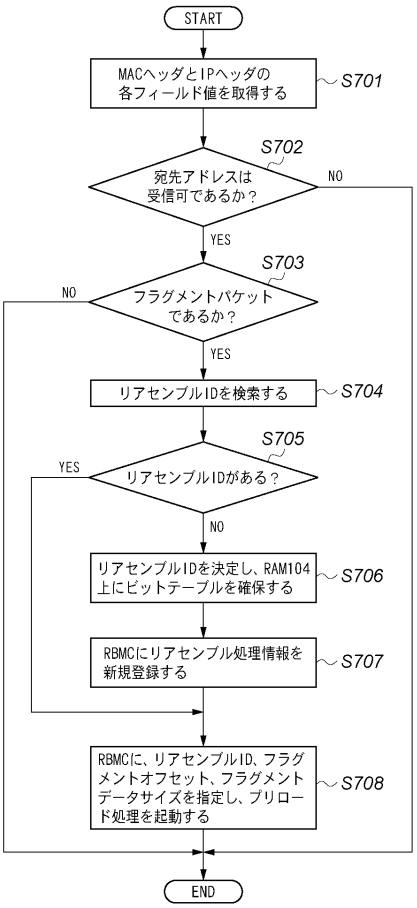
【図５】


DMAC 111
ローカルプロセッサ108
RBMC 112
S501
受信パケット転送開始
S502
ヘッダ部分までのデータ転送完了
通知
S503
S504
IPパケットのペイロードデータ転送
S505
IPパケット受信イベント
S506
MAC・IPヘッダ解析（IPID、送信元IPアドレス、IPパケット全長取得）
S507
IPリアセンブルの検索
S508
ビットテーブルのプリロード起動
起動
S509
S510
ビットテーブルのプリロード
IPヘッダチェックサム計算受信判定処理
S511
S512
S513
RBMC起動
起動
通知
S514
ビットテーブルビット操作処理
S515
S516
DMA転送完了
通知
S517
S518
DMA転送完了イベント

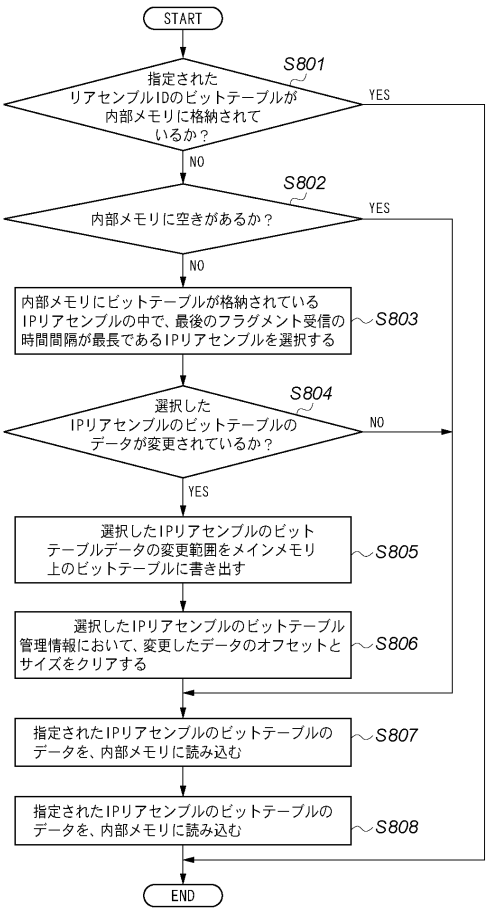
【図６】


609
612
613
614
615
31
16
15
0
604
606
607
608
603
611
602
601
605
610
パケット全長
フラグメントオフセット(13bit)
ヘッダ・チェックサム
送信元IPアドレス
宛先IPアドレス
オプション
サービス・タイプ(8bit)
IP識別子
プロトコル(8bit)
ヘッダ長(4bit)
パケット生存時間(8bit)
バージョン(4bit)

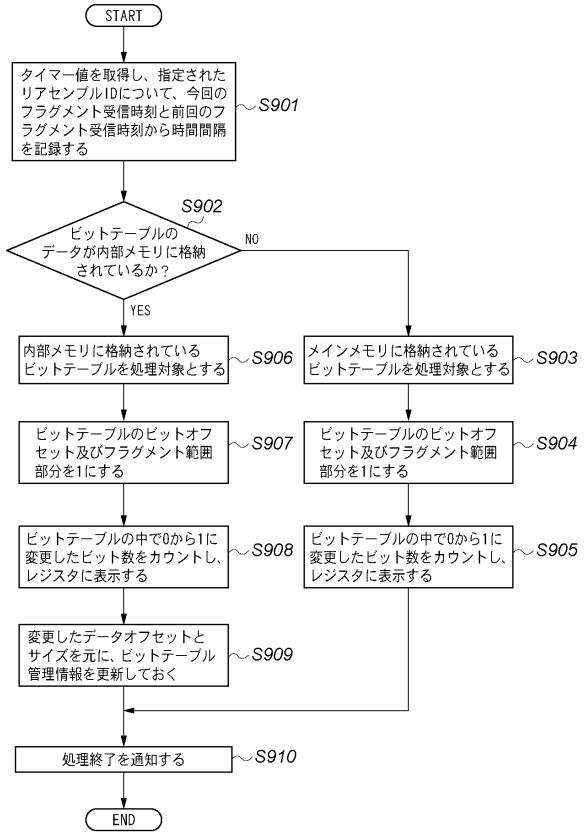
【図７】


START
MACヘッダとIPヘッダの各フィールド値を取得する
S701
S702
宛先アドレスは受信可であるか？
NO
YES
S703
フラグメントパケットであるか？
NO
YES
リアセンブルIDを検索する
S704
S705
リアセンブルIDがある？
YES
NO
リアセンブルIDを決定し、RAM104上にビットテーブルを確保する
S706
RBMCにリアセンブル処理情報を新規登録する
S707
RBMCに、リアセンブルID、フラグメントオフセット、フラグメントデータサイズを指定し、プリロード処理を起動する
S708
END

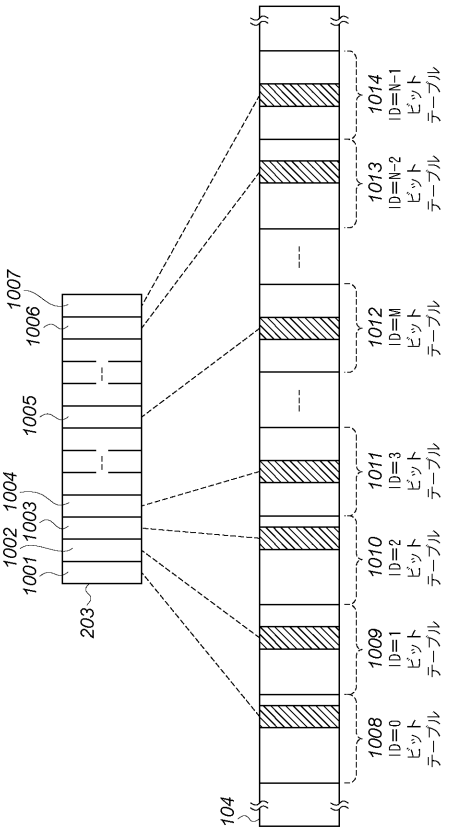
【図８】


START
S801
指定されたリアセンブルIDのビットテーブルが内部メモリに格納されているか？
YES
NO
S802
内部メモリに空きがあるか？
YES
NO
内部メモリにビットテーブルが格納されているIPリアセンブルの中で、最後のフラグメント受信の時間間隔が最長であるIPリアセンブルを選択する
S803
S804
選択したIPリアセンブルのビットテーブルのデータが変更されているか？
NO
YES
選択したIPリアセンブルのビットテーブルデータの変更範囲をメインメモリ上のビットテーブルに書き出す
S805
選択したIPリアセンブルのビットテーブル管理情報において、変更したデータのオフセットとサイズをクリアする
S806
指定されたIPリアセンブルのビットテーブルのデータを、内部メモリに読み込む
S807
指定されたIPリアセンブルのビットテーブルのデータを、内部メモリに読み込む
S808
END

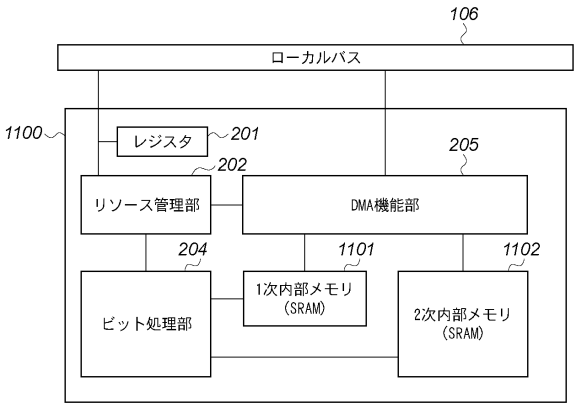
【図９】



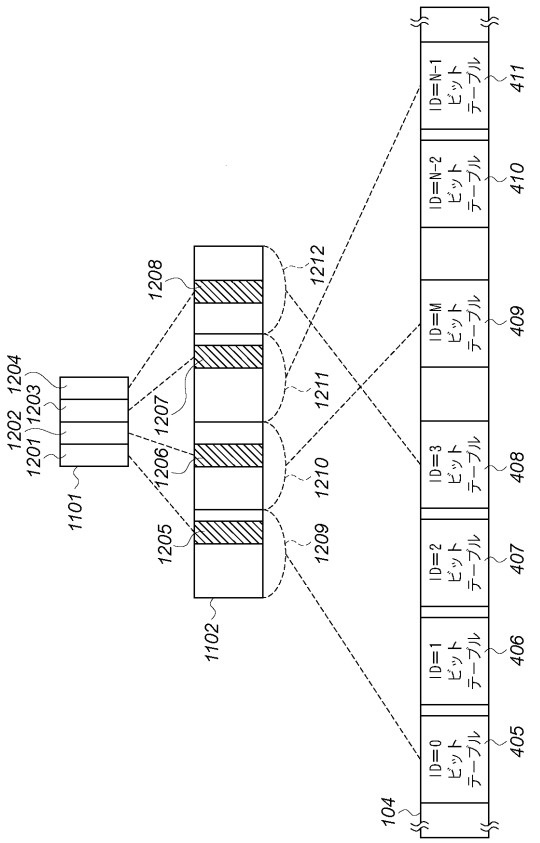
【図１０】



【図１１】



【図１２】



フロントページの続き

(72)発明者　今尾　英司
東京都大田区下丸子３丁目３０番２号　キヤノン株式会社内

審査官　安藤　一道

(56)参考文献　特開２００６－００５８７８（ＪＰ，Ａ）
特開平１１－２６１６４９（ＪＰ，Ａ）
特開平０５－３２７７７１（ＪＰ，Ａ）
特開２００６－０１４１４３（ＪＰ，Ａ）
特開２００７－２７４０５６（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｈ０４Ｌ　１２／５６