社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2008－SLDM－134／2008－EMB－8 (11)
2008 / 3 / 27

組込みシステム向けハードワイヤードTCP/IPオフロード・エンジンの小型化実装・高性能化手法

橋本 浩二† モシニャガ ワシリー†

†福岡大学工学部電子情報工学科 〒814-0180 福岡県福岡市城南区七隈 8-19-1
E-mail: †{khashi, vasily}@fukuoka-u.ac.jp

あらまし 情報家電・組込みシステムの一部にはIPスループット向上を目的として，ハードワイヤードTCP/IPオフロード・エンジン（TOE）が実装されているが，その「最大ピークTCP/IP通信スループット／消費電力」をより高めることが求められている．そこで我々は通信アプリケーション特性に着目し，小さい回路コストで前述の課題に対応可能な高性能・低レイテンシTOEを開発すべく回路・ソフトウェアの構成について検討を行っている．本発表では我々が提案するTCP/IP受信回路の構成および開発状況を報告し，あわせて受信パケットに基づき簡素的に処理内容を推測し，DMA実行する手法についても概説する．

キーワード 組込みシステム，TCP/IP，ハードワイヤードTOE

Design Technique of Low-Size and High-Performance Hardwired TCP/IP Offload Engine for Embedded Systems

Koji HASHIMOTO† and Vasily G. MOSHNYAGA†

† Department of Electronics Engineering and Computer Science, Fukuoka University
8-19-1 Nanakuma, Jyonan-ku, Fukuoka-shi, Fukuoka, 814-0180 Japan
E-mail: †{khashi, vasily}@fukuoka-u.ac.jp

Abstract A hard-wired TCP/IP offload engine (TOE) is implemented in several kind of information appliances and embedded systems in order to accelerate the IP-based communication throughputs. In recent years, TCP/IP throughput peak value is becoming an important factor for the systems but they are required strongly to keep low power consumption. Therefore we are developing a new TOE structure with high-performance, low-latency and small area that utilizes behaviors of the communication layer software applications. This paper describes structure of the packet reception block of TOE and our experimental implementation report. We also propose new efficient methods by speculating process details of the incoming packet simply and then try to execute DMA transaction beforehand.

Keyword Embedded system, TCP/IP, Hardwired TOE

1. はじめに

近年の計算機システムにおいて，インターネットやLANといったネットワークへの依存度は高まる一方であり，システム負荷全体に対するネットワーク・パケット処理負荷の割合を低減することが求められる．したがって計算機システムのネットワーク・インタフェース・コントローラ（NIC）回路内部において，データリンク層（イーサネットではMAC層と呼ばれる）上で送受信されるMACフレームのCRC-32を自動的に計算・検証することは一般的に行われている．最近はさらに，ネットワーク層（IP, ARP等）以上におけるパケット内部フォーマットを簡易的に解析し，IPヘッダ，TCP，UDPのチェックサムを自動的に計算・検証するハードウェア・ロジック回路[1]や，ヘッダーデータ部間においてデータ部の先頭のワード境界を任意に揃えるようにパッドを挿入する[2]といった機構を備えるものや，TCP/IPのハードウェア化[3,4]といった高機能NICが登場している．これらの取り組みはパケット個々の送受信にかかるソフトウェア処理負荷を低減するのに大きく貢献している．こうした手法は今日の高性能計算機システムやネットワーク機器に組み込まれるハイエンド・ネットワーク・プロセッサで一般的に導入されている．

最近，いわゆるデジタル情報家電や組込みシステム等においてIPベースの通信機能を備えるものが増えている．対象アプリケーションによって大きく異なるが，要求される通信スループットは今後，全般的に増大すると予想される．しかし組込みシステムではシステム全体コスト上の強い制約があるのが通常で，既存ネットワーク・プロセッサのような大規模・高性能回路を搭載することは稀である．またソフトウェア改修作業を回避可能な手法が強く好まれる．必然的に，要求通信スループットを満足するためにはシステム動作周波数を高めるしかない．しかし動作周波数の引き上げは消費電力の増大をもたらすため，機器の小型化・携帯型機器への適用が困難になる．そこで最近は[5]に示すような組込みシステム向けハードワイヤードTCP/IP回

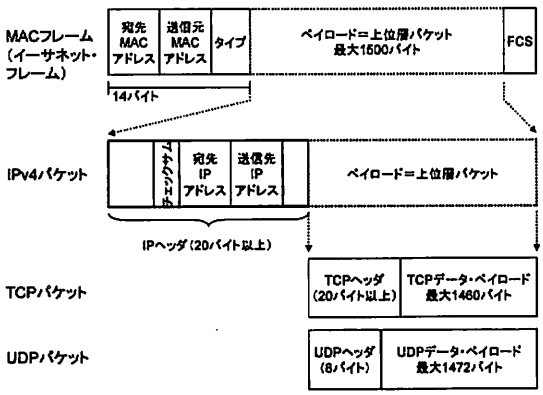

図1 MAC フレーム，IPv4,TCP,UDP パケット・フォーマットの概要

路部品が登場しているものの，用途が限られている．そこで我々は通信スループット／消費電力／回路コスト／アプリケーション特性を総合的に考慮した新しい組込みシステム向けの汎用 IP ベース・ネットワーク・インタフェース技術開発に取り組んでいる．本発表ではこれまでの成果と今後の課題を述べる．

2. 技術的課題

2.1. 既存技術

IP ネットワーク上でパケット送受信機能を有する一般的な計算機システムのアプリケーション（あるいは OS が備える一機能）では，OS が提供するプロトコル・スタック（API 仕様例：ITRON TCP/IP API 仕様 Ver2.0 [6]）との間に通信端点といった語句で定義されるコネクションを構築することにより，通信パケットの送受信を実現している．

NIC の回路構成は送信部と受信部に大きく分けられる．NIC 受信部では，物理（PHY）層から MII インタフェースを介して受け取った MAC フレーム（イーサネット・フレーム）に対して CRC 検証を施した後，いったんバッファに保存する．この処理層は MAC 層と呼ばれる．一方で，TCP コネクションが確立されてデータ転送シーケンス状態にあるプロトコル・スタックは，設定されている通信端点に基づいた受信 IP パケットの条件（上位プロトコルの情報，ヘッダのフィールド値，サイズなど）をあらかじめ決定しておく．この条件エントリは少なくとも同時に存在可能な通信端点の個数以上存在する．プロトコル・スタックでは NIC 受信部のバッファに格納されたパケットが上記条件に一致するか否かを探索・判定し，一致すれば通信端点が必要とするデータ転送（もしくはその一部）を実施する．

2.2. 課題

アプリケーション層がデータ受信のために設定する通信端点では 1 個以上のパケットを受信し，2 個以上のパケット・データから連続したデータ・ストリーム・ブロックを再構成する．またアプリケーション層ではさらに大規模なデータ・ストリームを受信するよう想定している場合も考えられ，その場合，通信端点を 1 個以上設けてデータを分割受信することになりソフトウェア処理負荷はそれだけ増加してしまう．

次に，プロトコル・スタックに目を向けてみる．現在の組込みシステムではさほど用いられていないが，既存の高機能 NIC を搭載する計算機システムでは，IP，TCP，UDP ヘッダ部と TCP，UDP データ部の分割化やチェックサム検証等を行うことで，メインプロセッサ上で動作するプロトコル・スタックのソフトウェア処理負荷を低減している．NIC 処理過程で得られた解析情報は通信端点探索・判定において有用であり，プロトコル・スタック処理負荷の低減に寄与することができる．ただしプロトコル・スタックのソフトウェアとしての汎用性や，高機能 NIC を搭載することによる回路コスト増加が問題となる．

NIC－プロトコル・スタック間でのパケットまたは MAC フレームのやり取りで用いられる，いわゆるディスクリプタ・テーブルの仕様構成に関しても，IP，TCP，UDP ヘッダとデータ領域を別個に取り扱い可能なものは，数ある多機能 NIC の中でも特に限られる．しかしながらデータ部分を格納する主記憶領域情報をアプリケーション層と NIC が直接やり取りできれば，NIC～プロトコルスタック～アプリケーション層間メモリ・コピーが削減できるため，大幅な処理性能向上が見込まれる．

一般に，受信した MAC フレームが MAC 層 FIFO バッファに蓄積され，プロトコル・スタックによる読み出しは開始されるまでには，幾分長いクロック・サイクル時間を要する．したがってバッファ・オーバーフローによって生じるフレーム消失を防止し，通信スループットを確保するためには十分なメモリ容量の FIFO を配置せざるを得ない．2MAC フレームを格納可能な FIFO を設けるならば，1 フレーム＝1518 バイトであるため，3KB 超の SRAM 容量が必要になる．さらに，消費電力の点からもクロック周波数を高めることが難しい．

高速ネットワーク物理層規格（ギガビット・イーサネット/ IEEE-802.3ab, MIMO Wi-Fi/ IEEE-802.11n 等）から得られる高い通信バンド幅を享受できるアプリケーション展開が望まれている．しかし，既存のソフトウェア・ベースのプロトコル・スタックでは対応し難く，既存 NIC をそのまま搭載することも非常に難しいという現状にある．

3. ハードワイヤード TOE 受信部の仕様検討

3.1. 基本方針

前章に示した課題に対処すべく，コスト的制約が強い組込みシステム用 SoC に集積可能な程度に小型かつ十分に高性能な NIC を開発する．当初の目標として，MAC 層から受け取ったフレームを IP，TCP，UDP レベルでヘッダ情報を解析し，適切な主記憶アドレスへパケット・データ部を収納するまで自動処理回路を構築すべく，仕様検討・プロ

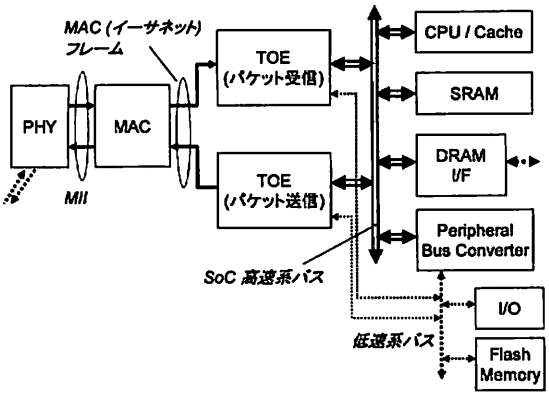

図2　提案 TOE を内蔵する SoC 構成例

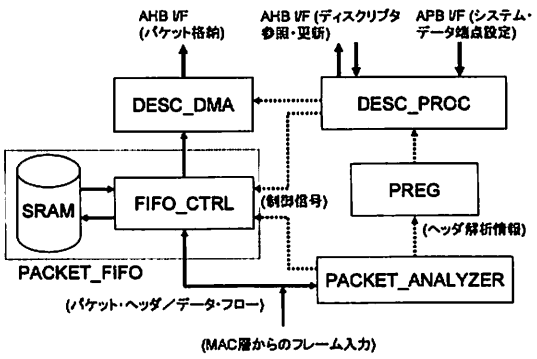

図3　TOE 受信部の内部構成概要

トタイプ設計を行う．このような機能を備えるネットワーク・パケット処理回路は一般に TCP/IP オフロード・エンジン（TOE）と呼ばれる．本節ではハードワイヤード TOE 受信部仕様の基本方針を示し，次節では策定した回路仕様を概説する．

基本方針は以下の通り：

(1) 提案 TOE（受信部）は，指定したメモリ・アドレスに TCP/UDP データ・ペイロードを格納する機能をハードウェアで提供し，アプリケーションが設ける TCP/UDP 通信端点へデータを受け渡す．

(2) 上記アドレス指定には既存 NIC 同様，ディスクリプタ・テーブル方式を採用する．回路コストを抑えるため，TOE 内部のレジスタ・ビット数を必要最小限に抑えるようにする．

(3) TOE を内蔵する SoC のクロック周波数を X[MHz]，上記データ受信の実効スループットを Y[M ビット/秒]とするとき，(Y×2)＞X を性能目標として掲げる．

(4) TOE 受信部，送信部それぞれに含まれるパケット・バッファのサイズを，それぞれ IP パケット 1 個分程度とし，回路内 SRAM 面積を抑える．

(5) 既存 NIC 回路との機能的上位互換性を維持する．

提案 TOE を内蔵する SoC 構成例を図 2 に示す．この構成例ではメモリ・バスを高速系と低速系に分離し，高速系に CPU，キャッシュ，主記憶 SRAM，DRAM I/F などが接続されると想定する．NIC 内部は PHY 層，MAC 層，そして TOE（送信部，受信部）の階層構造を採る．PHY 層で受信された MAC フレームは MAC 部から TOE 受信部に伝達される．TOE 受信部は高速系バスに接続されるものとし，主記憶に対するメモリアクセス・バンド幅を確保する．

3.2. TOE 受信部の回路仕様

前節の基本方針を元に回路仕様を策定する．TOE 受信部は 6 ブロックで構成されるものとし，SoC 内部バス規格として高速系に AMBA AHB を，低速系に APB を用いることとする．各ブロックの名称，機能的内訳は以下の通り：

3.2.1. PACKET_ANALYZER

MAC 層から入力される MAC フレームに含まれる各種ヘッダ部位を解析する役割を担う．

(1) まず MAC ヘッダ部から IP，ARP/RARP，その他，に分別する．IPv4 パケットである場合，上位層プロトコルが TCP，UDP，その他，に分別する．本ブロックで解析されて得られたヘッダ内フィールド情報を PREG に格納する．IP，TCP，UDP ヘッダ・サイズと，それ以後のデータ領域サイズ情報も合わせて算出し，PREG に格納する．

(2) MAC 層からパラレル・ビット系列で入力されるフレームはそのまま PACKET_FIFO に直接伝達されるようにし，PACKET_ANALYZER は FIFO への入力制御を担う．

(3) ヘッダ解析と同時に，IP ヘッダおよび TCP，UDP ヘッダのチェックサム計算を実施する．MAC フレームの CRC 計算については MAC 層で実施されるものとし，PACKET_ANALYZER は MAC フレームと共に計算結果も受け取る．チェックサム，CRC のいずれかでエラーが発生した場合，FIFO に格納したパケットをフラッシュし，エラー発生ログを DESC_PROC に伝達する．

3.2.2. PREG

PACKET_ANALYZER が生成した解析情報を一時的に蓄積するレジスタである．FIFO に保持可能なパケット最大個数は PREG のレジスタ本数と一致している必要がある．

3.2.3. PACKET_FIFO

MAC 層からの入力されるフレームを一時的に保存するための FIFO メモリである．内部は FIFO コントローラと 1R/1W-SRAM で構成される．少なくとも 1 個の MAC フレームを完全に格納できるだけの SRAM 容量が必要であるが，TOE 受信部の実装方針に応じて，最大格納可能パケット個数を増やしてよい．

3.2.4. DESC_PROC

PREG に格納された情報およびディスクリプタ・テーブルで記述されたデータ端点(詳細は後述)情報に基づいて，FIFO に格納されたヘッダおよびデータ部をそれぞれ所望

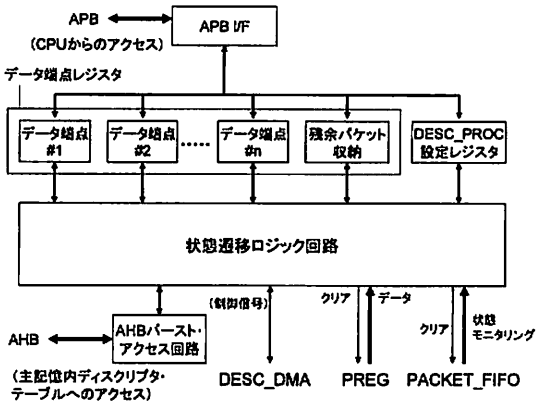

図4 DESC_PROC 内部ブロック構成の概要

の主記憶アドレスに転送させる役割を担う．

図 4 に DESC_PROC の内部ブロック構成を示す．内部の状態遷移はハードワイヤード・ロジックで実現するものとする．機能詳細は以下の通り：

(1) FIFO からのパケット読み出しとメインメモリへ書き出すための DMA 処理は DESC_DMA が担うものとし，DESC_PROC はその制御を担う．

(2) DESC_PROC 内部には 1 個以上のデータ端点管理レジスタを有する．データ端点とはプロトコル・スタックの通信端点におけるデータ受信条件であり，そのエントリ数は通信端点数にあわせて複数個備える．

(3) データ端点条件に合致し受信されたパケットは，プロトコル・スタックがあらかじめ設けておいた，データ端点固有のディスクリプタ・テーブル（図 5）にて登録・管理される．受信したパケット個々に 1 個のディスクリプタ・テーブルが消費され，パケットのヘッダ部，データ部の格納先アドレス指定，サイズ，プロトコル解析レポートを DESC_PROC とプロトコル・スタック間で授受するのに用いる．データ端点それぞれに設けるディスクリプタ・テーブル個数は任意である．

(4) データ端点それぞれに，ポインタ参照によって単方向チェーン化された 1 個以上のディスクリプタ・テーブルで構成されるオブジェクトが割り当てられ，これによりパケット・キュー機能が実現される．

(5) データ端点を設定する際，予定データ受信サイズ初期値をセットし，パケット受信のたびにデータサイズ分減算する機構を備える．また，TCP 順序番号をモニタし，不一致パケットを排除する．

(6) 個々のデータ端点は TCP-RST パケットを確実に受信できるよう，特別なディスクリプタ・テーブルをパケット・キュー用とは別に設ける．

(7) いずれのデータ端点条件にも合致しなかった残余パケット（TCP-SYN あるいは TCP，UDP 以外のプロトコル・パケット等）を格納するため，データ端点と同様に汎用パケット格納レジスタおよびパケット・キュー用のディスクリプタ・テーブル・チェーンを設ける．

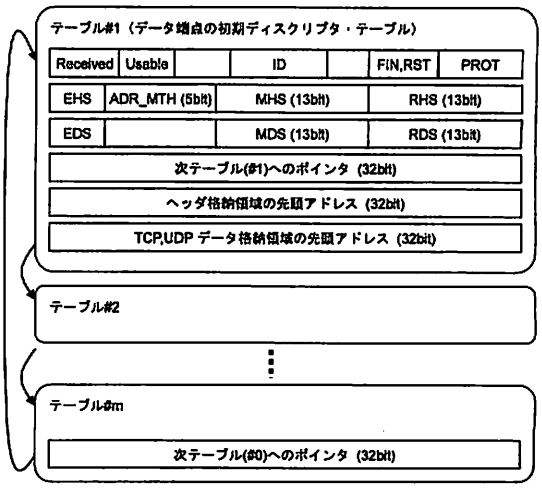

図5 データ端点ディスクリプタ・テーブルの基本構成，単方向チェーン構造

(8) データ端点レジスタ群，残余パケット管理レジスタ，設定レジスタは APB に接続され，CPU から自在にアクセス可能である．それとは別個に，ディスクリプタ・テーブルを高速アクセス可能にするため，内部に AHB バーストアクセス回路を設ける．

(9) 1 パケットの処理が完了したら PREG 格納情報をクリアし，次パケット受信まで待機する．データ端点が複数個存在する場合，今のデータ端点が探索順序の先頭となるよう順序設定を修正する．

3.2.5. DESC_DMA

PACKET_FIFO に格納されているパケットを読み出し，パケットのヘッダ部とデータ部をそれぞれ DESC_PROC から指示される任意アドレス・任意サイズでもって AHB にバースト書き込みする役割を担う．

4. ハードワイヤード TOE 受信部の実装・評価

前章に示した基本仕様を元に，それぞれの回路ブロックの詳細仕様を策定した．また Verilog 言語を用いて論理回路設計を行った．論理合成ソフトウェアには東大 VDEC が提供する Synopsys DesignCompiler を，スタンダード・セル・ライブラリには VDEC 提供 Hitachi 0.18um プロセス・ルールを用いた．論理合成結果（2NAND 換算セル数）は下表の通り：

PACKET_ANALYZER	5998
PREG	3989
FIFO_CTRL	883
DESC_DMA	4628
DESC_PROC	40016
計	55514 セル

表 1 TOE 受信部の回路規模（セル数）

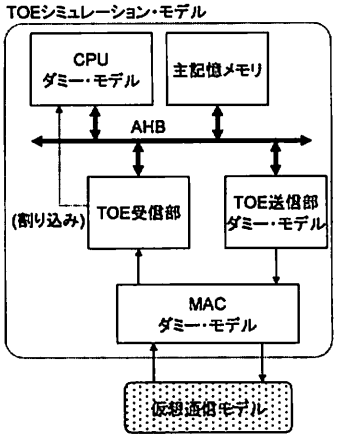

図6 TOE 受信部性能評価シミュレーション・モデル

実システムとして実現するには TOE 送信部および MAC 部を組み込む必要がある．TOE 送信部および MAC 部に関しては現在，回路を設計中であるがそれらを加えても論理回路全体で 100k セルを超えないと予測される．したがって FIFO 内部 SRAM（最低 1518 バイト）を含めても，本 TOE を搭載することによる組込みシステム用 SoC のコスト増加はそう大きくならないと予測される．

次に，HDL シミュレータ上で動作する TOE 受信部性能評価用 TOE シミュレーション・モデル（図 6）を構築し，TOE 受信部データ端点による TCP データ受信の性能評価シミュレーションを実施した．ここで，シミュレーション記述で構成された仮想通信モデルを TCP 通信の要求元，TOE シミュレーション・モデル側を相手先とし，MAC, TOE 送信部，CPU コアを全てシミュレーション記述によるダミー・モデルとした．また MAC と仮想通信モデルを直接接続し，PHY 層同等の MAC フレーム授受を担うものとした．シミュレーション条件の詳細は以下の通り：

- MAC 内部には既存の IP コア仕様を元に算出した伝達遅延（MAC フレームの CRC 検査，CRC 生成に起因する）を有し，TOE シミュレーション側には 32 ビット・パラレル I/O を提供するものとする．
- MAC－仮想通信モデル間を 100Mbit/s，シミュレーション・モデル全体クロック周波数を 25MHz とする．
- 仮想通信モデル側でのパケット送出遅延を 0 とし，連続送出可能なパケットは間隔無く送出される．
- 仮想通信モデルが TCP コネクション確立後に送出する TCP パケットのデータ・ペイロード長を 1460 バイトとし，連続して 4 パケット送出する．TOE シミュレーション・モデル側が 1 パケット分を主記憶に格納し終わると，TOE 受信部は CPU に割り込みを発生する．
- 以上の動作を繰り返す．

仮想通信モデルが 1 パケット目を送出し始めてから TOE 受信部が割り込みを発生するまでに平均 4131 サイクルを要した．次に，1 パケット目の割り込み発生から 3.4ms 遅

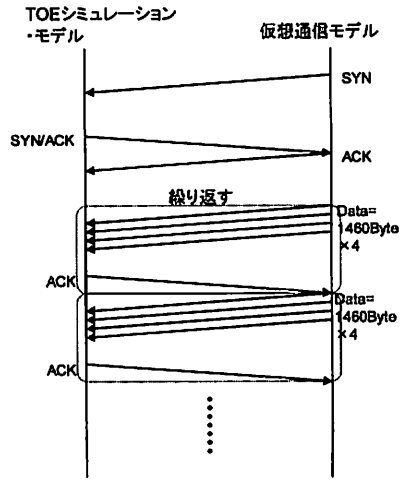

図7 TOE 受信部性能評価シミュレーションにおける TCP 通信でのパケット送受信フロー

延の後，仮想通信モデルが TCP-ACK パケットを受信すると想定した．なおこの遅延時間は既存の ARM7 ベース・システム（TCP/IP プロトコル・スタック・ソフトウェアが標準添付されている μITRON 準拠 OS で動作）のデバッグ結果から算出した．図 7 は上記条件によるパケット送受信フローを表したものである．

以上のデータより，繰り返しフェーズ 1 つ分の開始～終了までのサイクル数は 3.4[ms]／40[ns]＋4131 = 89131，
通信スループットは最大ピーク時で

1460[Byte]×4[packet]×8[bit]÷89131[cycle]÷40[ns]
≒13.1[M ビット/秒]

となる．実効通信スループットは TCP コネクション確立から切断までの時間から算出されるため，上記の見積り値では正確な値ではない．しかし所期の目標値をおおよそ満足すると判断するに足りる，有力な指標である．

5. 性能向上策の検討

5.1. 未解決課題

本稿で提案した TOE 受信部回路により通信スループットの高性能化を実現できる．だが通信安定性・アプリケーション層からみたときのレスポンス性をより改善するためには，抜本的な処理性能向上策が必要である．現状の TOE 受信部のアーキテクチャ方針は，基本的に従来からの NIC 内部およびプロトコル・スタックの階層的構造を引き継いでいる．そのため，下記の未解決 3 課題への対応が急務である：

(1) IP ヘッダ，TCP，UDP のチェックサム．

IPv4 での TCP, UDP は MAC フレーム・レベルの CRC 検証（MAC 層で処理される内容である）に加えて IP ヘッダ，TCP，UDP それぞれのレベルでチェックサム検証を実施する（全て PACKET_ANALYZER 部で処理される）．IPv4 へ

ッダのチェックサム検証は IP および TCP，UDP のヘッダを解析して必要な情報を抽出する際，同時に計算可能なので性能上特に問題にはならない．だが TCP，UDP のチェックサム検証ではパケットの最後尾までデータを参照して計算する．したがって PACKET_ANALYZER 処理が完了しパケットが FIFO に格納され終わった後， DESC_PROC の処理が開始するような回路仕様となっている．このギャップは本 TOE の主な性能ボトルネック要因となっている．

(2) DESC_PROC におけるデータ端点探索時間．

組込みシステムにおいては回路コスト面での制約が強いため，データ端点情報を DESC_PROC 内部に全てレジスタで配置するのではなく，多くを主記憶上に配置せざるを得ない．しかるに複数のデータ端点が設けられる場合，受信パケットがどのデータ端点条件に該当するかを探索するのに要するクロック・サイクル数が増加する．

(3) PHY 層の性能向上に伴う問題．

10Base から 100Base そして 1000Base と，PHY 層規格は継続的に進化している．それに伴い MAC フレーム送信側の送信スループット・ピークは必然的に高まる．しかし TOE 受信部の処理スループットとの差を補填すべく大容量 FIFO を設置しようにも，回路コスト的制約が強いため困難である．よって MAC フレームが TOE 受信部に受け渡されずロストする状況が頻発し，処理スループットの低下を引き起こす可能性がある．

5.2. 解決策の骨子

下記に記すように，推測に基づく投機的処理機構を組み込んだ受信パケット処理系を構築し，本課題に対処することを検討している．

1. MAC フレーム入力から主記憶への格納に至るフローにおいて，CRC 検証の後を並列化（フローA とフローB）する．フローA は基本的に従来同様とする．
2. フローB では，データ端点を探索ではなく推測により決定し，その仮データ端点情報により，フローA よりも格段に短時間でデータ部 DMA 処理を開始する．
3. DMA が終了するまでにフローA におけるデータ端点探索が終了し，その後，真のデータ端点が確定する．真データ端点がフローB で推測した端点と同一ならばすでに DMA が終了しているので，CPU にパケット受信を通知してよい．同一でなければ真データ端点情報に基づき DMA を実行しなおす．

図 8 に上記をフローチャート化したものを示す．フローB におけるフォーマット解析・情報抽出フェーズでは，TOE 受信部が直近で処理したデータ端点受信パケットとの一致性を把握することを主目的とする．一般に組込みシステムにおいては通信相手がさほど多岐に渡らないこと，大量の連続したデータ送受信を目的とするウィンドウ・スライド TCP 通信割合は高いと予測される．したがって，相当に雑な推測であっても性能向上に寄与する可能性が高く，詳細については今後検討していく．

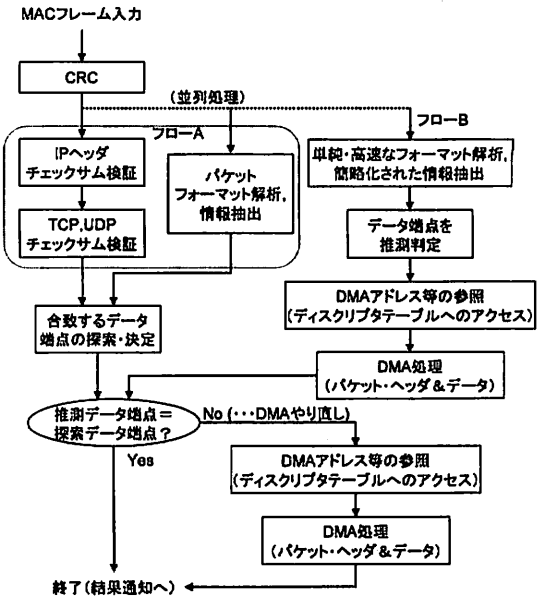

図8　NIC の処理スループット向上を実現可能な処理フローの構成例

6. おわりに

本発表では我々が提案する TOE 受信部の基本構成およびその実装評価を示した．また，その次段階として検討している高性能化手法の骨子を示した．今後も TOE 受信部回路設計を進めるとともに，同時に，TOE 送信部と本 TOE 用プロトコル・スタックの開発にも取り組む．

文　献

[1] Hewlett Packard Company, "ネットワークプロトコルスタックのためのハードウェアによるチェックサム支援機構," 日本国特許公開 H11-168451, 1999 年.

[2] Hewlett Packard Company, "ネットワークアダプタシステム," 日本国特許公開 H06-078001, 1992 年.

[3] 田上敦士 他, "ハードウェア記述言語による TCP/IP プロトコルの実装," 信学研報 CPSY2000-9, Vol.100, No.86, pp.17-24, 2000 年.

[4] 長谷川洋平，下西英之，村瀬 勉，"ハードウエア TCP-NIC(TCP オフロードエンジン)の TCP 動作検証と性能評価・考察," 第 13 回インターネット技術第 163 委員会研究会，日本学術振興会産学協力研究委員会，2003 年 5 月.

[5] IP コア TOE(IP2TCP-101), 株式会社アイピースクエア http://www.ip-square.com/toe.html

[6] "ITRON TCP/IP API 仕様 Ver.2.00.00," 社団法人トロン協会, 2006 年 7 月 http://www.assoc.tron.org/