情報処理学会論文誌　Vol. 52　No. 12　3715–3728 (Dec. 2011)

推薦論文

Gigabit/10 Gigabit Ethernetに対応した高効率TCP/IPオフロードエンジン

田中信吾[†1]　山浦隆博[†1]　山口健作[†1]
菅沢延彦[†1]　谷澤佳道[†1]　渋谷尚久[†1]

ネットワークを流れる映像トラフィックの増加により，特に高速大容量な通信を指向した組み込みの分野ではTCP/IP処理を行うCPUの動作周波数や消費電力の増大が問題となってきた．そこで我々は，専用ハードウェアを用いたTCP/IPオフロードエンジンNPEngine^{TM}を開発し，Gigabit Ethernet向け実装で従来の組み込みCPUに比べて動作周波数あたりで80倍，消費電力あたりで22〜29倍の伝送レートを実現した．また，10 Gigabit Ethernet向け実装では90 MHzで双方向同時9 Gbps以上の伝送レートを実現し，省電力FPGAを用いた際の装置全体の消費電力の予測値は約2 Wと従来のPCによる実現と比べて大幅に低い消費電力となった．

Highly Efficient TCP/IP Offload Engine for Gigabit/10 Gigabit Ethernet

Shingo Tanaka,[†1] Takahiro Yamaura,[†1]
Kensaku Yamaguchi,[†1] Nobuhiko Sugasawa,[†1]
Yoshimichi Tanizawa[†1] and Naohisa Shibuya[†1]

The increase of video streaming traffic over networks has created several problems for host processors that process the TCP/IP protocol. The problems, such as higher required operating frequency and/or increased power consumption, are especially important when the processors are used in embedded systems. In order to solve these problems, we have designed and implemented a hardware-based TCP/IP offload engine, called NPEngine^{TM}. Compared to a conventional embedded CPU, NPEngine^{TM} configured for Gigabit Ethernet can achieve about 80 times higher throughput at the same operating frequency, or 22 to 29 times higher throughput for the same power consumption. NPEngine^{TM} configured for 10 Gigabit Ethernet achieves a bidirectional throughput of over 9 Gbps at 90 MHz operating frequency. The total power consumption of the system using a low power FPGA is estimated to be about 2 W, which is much lower than that of a conventional PC-based system.

1. はじめに

昨今，映像のハイビジョン化による高画質化・高解像度化，ネット映像配信の視聴ユーザ数の爆発的な増加が進んでおり，ネットワーク上を流れる映像データの伝送レートが増大している．それに対し，ネットワークの下位層である物理層/リンク層に関しては，Gigabit/10 Gigabit Ethernet[1]，高速無線LAN（IEEE 802.11n，802.11ad）などの出現で見られるように伝送帯域の向上が進んでいるものの，上位層のTCP/IP[2)–4)]の通信プロトコル処理は依然としてCPU処理に頼っている状況が続いている．

TCP/IPの処理負荷については以前より問題視[5)]されており，TCP/IP通信の伝送レート1 bpsを得るためにはPC向けCPUの動作周波数約1 Hzを要するといわれている[6)]．そのため，下位層の進化に合わせて，たとえば伝送レートを10倍に向上させるとすると，CPUの動作周波数を10倍に向上させる必要があるということになり，ホストCPUの動作周波数の増大，さらには，それにともなう消費電力の増大が課題となる（**図1**）．これは

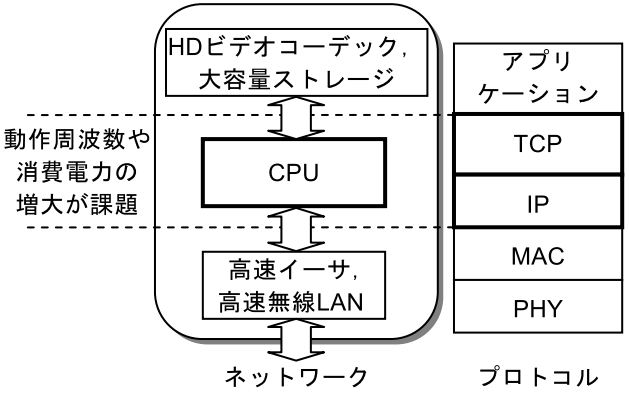


図1　ネットワーク機器の構成と課題
Fig. 1　Structure and issues of network equipment.

†1 株式会社東芝研究開発センターネットワークシステムラボラトリー
Network System Laboratory, Corporate Research and Development Center, Toshiba Corporation
本論文の内容は2010年7月のマルチメディア，分散，協調とモバイル（DICOMO）シンポジウム2010で報告され，IOT研究会主査により情報処理学会論文誌ジャーナルへの掲載が推薦された論文である．

CPU のマルチコア化によっていくらかは軽減されるものの，PC 向け CPU より性能が劣る組み込み CPU においては深刻であり，たとえば高いビットレートの映像コンテンツを扱うスタジオ系の映像関連機器などでは問題となっている．また，ハイエンドサーバ領域においても，たとえば市場が急速に拡大しているインターネットを用いた VOD（Video On Demand）サービス向けの，大多数の配信サーバで構成されるクラウドデータセンタの省電力化の妨げにもなっている．

そこで我々は，以上の問題を解決する TCP/IP オフロードエンジン NPEngine^{TM} を開発したので，本論文でその詳細について述べる．

2. 従来技術の課題

TCP/IP オフロードエンジン（以下，TOE）とは，TCP/IP 処理を行う処理エンジンをホストに追加し，ホスト CPU から TCP/IP 処理の負荷をオフロードするものである[7)]．TOE は今までに多数の方式が提案されているが，これらの多くは当然ながらホスト CPU からのオフロード自体を目的としたものであり，追加した処理エンジンのデータパケットの処理をプロセッサベースで行っているものが多い[8)–10)]．そのため，確かにホスト CPU の負荷は減少するものの，ある意味負荷を TOE に移動しただけともいえ，特に組み込み機器においてはトータルの消費電力の低減には必ずしも貢献せず，前述の消費電力の課題を解決できないという問題がある．TOE をすべてハードウェアで実装したものは，たとえば文献 11)，文献 12)，文献 13) で提案されているが，これらはハードウェアにして伝送レートを向上させることのみを目的としており，低消費電力化に関する検討・評価を行ったものではない．そのほかに文献 14)，文献 15) などがあげられ，これらは本論文と同じく低消費電力化を志向したものであるが，結果としてはあまり良い性能が得られていない．また，ハードウェアとソフトウェアの組合せで実現したものとして文献 16)，文献 17)，文献 18)，文献 19) があるが，これらも主にオフロードを目的としており，いずれも低消費電力化を考慮して設計されたものではない．

そこで本研究では，PC・サーバ分野だけでなく，特に省電力の要求が高い組み込みシステム応用も視野に入れ，高伝送レートと低消費電力の二律背反を改善し，高い処理効率を実現する TOE である NPEngine^{TM} を開発した．次章ではまずそこで用いられている提案方式について述べる．

3. 提 案 方 式

3.1 概　　要

消費電力を抑えるためには，一般に，① 動作周波数を下げる，② 回路規模を抑える，の 2 つが重要である．① については処理を効率良く専用ハードウェアに落とすことが必要である．ただし，そのままでは ② の回路規模の増大が避けられないため，少なくともデータパケット処理の頻度と同等の頻度で発生する繰返し処理など，負荷の高い処理のみを選んで効率良くハードウェアに落とす必要がある．

以上の方針に基づき，NPEngine^{TM} では回路規模を抑えるハイブリッド構成，データ転送処理の処理効率を最大限に高めるダイレクト転送方式とパイプライン処理方式，TCP 処理で不可欠な ACK 処理を完全ハードウェアで実現するハードウェア ACK 処理方式を取り入れた．以降，3.2〜3.5 節で各方式について詳細を説明し，3.6 節でそれら提案方式を実現する専用ハードウェアの構成を説明する．

なお，紙面の都合上，以降の説明では主に受信処理のみを扱っているものもあるが，送信処理についても基本的には同様なので，適宜読み替えていただきたい．

3.2 ハイブリッド構成

NPEngine^{TM} の概略構成を**図 2** に示す．NPEngine^{TM} は，従来すべて CPU で行っていた TCP/IP 通信処理のうち，下位層の処理，および，高速化が期待される TCP データパケットの処理のみを専用ハードウェアで行い，残りの制御パケットの上位層の処理を CPU で行うハイブリッド構成としている．NPEngine^{TM} は，これら専用ハードウェア，制御パ

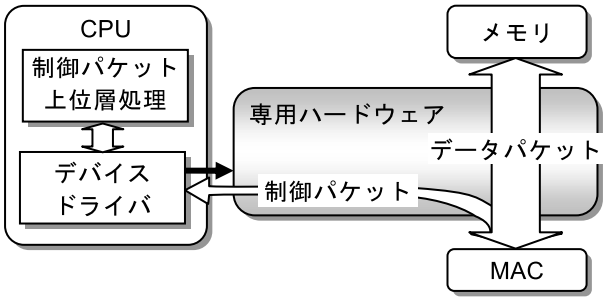


図 2　NPEngine^{TM} の概略構成
Fig. 2　Architecture of NPEngine^{TM}.

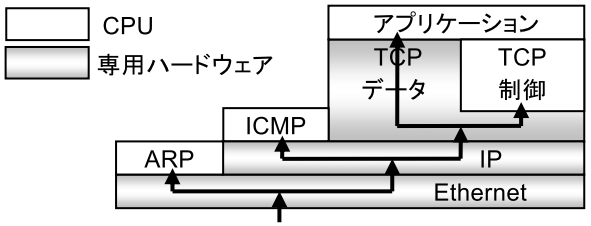


図 3　NPEngine^{TM} のプロトコルスタック構成
Fig. 3　Protocol stack architecture of NPEngine^{TM}.

ケット上位層処理ソフトウェア，専用ハードウェアのデバイスドライバソフトウェアで構成される．

TCP/IP 通信では，データを転送するデータパケットのほかに，ARP パケット，ICMP パケット，TCP SYN パケット，TCP FIN パケットなど様々な制御パケットが送受信されるが，これら制御パケットに対する処理は複雑なものとなり，すべてハードウェアで実装すると回路規模が増大してしまう．一方で，これらの制御パケットはデータを転送するパケットではなく，一般に送受される頻度も少ないため，ハードウェア化による高速化を行ってもほとんどメリットがない．

そこで本方式では，パケット受信時に**図 3** のようにまずハードウェアでパケットの処理を開始し，TCP データパケット（再送データパケットも含む）や ACK パケットに関してはそのまま最後までハードウェアで処理を行い，それ以外の制御パケットに関しては，TCP データパケットではないと判断された時点でデバイスドライバを介してホスト CPU に渡され，それ以降の上位層の処理をホスト CPU に引き継ぐ．

このような構成によって，全体をハードウェアのみで構成する場合と比べてもデータ転送の処理効率をまったく落とすことなく，一方で，それ以外の処理はすべてソフトウェアで構成することでハードウェアの回路規模を極力抑えているため，回路の新規開発コストやデバイスコストの削減，さらには，回路規模の縮小による消費電力の削減に貢献する．また，ソフトウェアとハードウェアで機能が重複することなく，相互に補完的な機能のみを備えるように構成することから，ソフトウェア側にとってもハードウェアが持つ機能を備える必要はなくなり，コードサイズやフットプリントの削減につながるというメリットもあげられる．

なお，前章で述べたように，従来でもソフトウェアとハードウェアを組み合わせた構成はいくつか提案されているが，いずれもハードウェアの開発量を最小限に抑えることを最優先

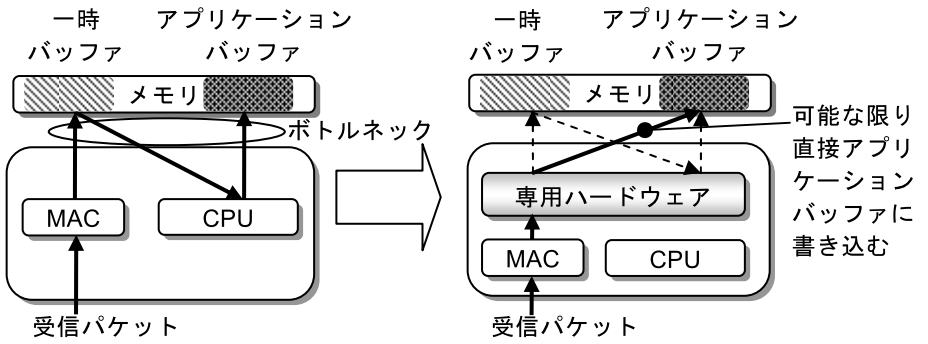


図 4　従来の一般的な受信処理（左）と NPEngine^{TM} のダイレクト転送方式による受信処理（右）
Fig. 4　Conventional generic receive processing (left) and receiving processing of NPEngine^{TM} with direct transfer method (right).

し，たとえば文献 17)，文献 18) にみられるように，パケットの並べ替え処理や再送パケット処理，フロー制御処理，輻輳制御処理などデータパケット処理の一部の処理までもソフトウェア処理で構成しており，データ転送の処理効率が幾分犠牲になっていた．それに対して本方式は，すべてのデータパケットと ACK パケットの処理を完全にハードウェアで作り込むことでデータ転送の処理効率を最大限高めており，特に高速通信時の低消費電力化を最優先しているところに特徴があるといえる．

3.3　ダイレクト転送方式

NPEngine^{TM} の低消費電力に最も貢献しているのは，送受信するデータをアプリケーションバッファに直接読み書きするダイレクト転送方式である．

従来のソフトウエアによる受信処理では，受信パケットはまず MAC（Media Access Controller）によりいったん一時バッファに書き込まれ，その後に TCP/IP 処理を行う CPU によってアプリケーションバッファにコピーされており，このコピー処理にともなう複数回のメモリアクセスがボトルネックとなっていた（**図 4**（左図））．なお，ここでいうアプリケーションバッファとは，Socket API の recv()/send() 関数の引数で指定するバッファそのもののことを意味する．

この問題を解決する方法として，従来技術でもゼロコピーと呼ばれる技術がいくつか提案されている[20]．しかしながら，これらの多くは PC ベースの環境を想定しており，ハードウェア（引用文献では NIC）に大きなメモリが必要であったり，ホスト CPU の MMU を利用していることから，アプリケーションバッファに対するホスト CPU 以外のデバイスに

よる DMA を想定していなかったりするなど，今回のような組み込み機器を想定した場合には必ずしも有効に機能しない．

そこで本方式では，図 4（右図）のように動作することでこの問題を解消する．まず，MAC が受け取ったパケットを直接専用ハードウェアが受け取り，メモリに書き込む前に TCP/IP のプロトコル処理を行う．プロトコル処理によって特定した TCP のコネクション（送信元と送信先の IP アドレスおよび TCP ポート番号の組合せで特定される）からどのアプリケーションバッファにデータを書き込むか特定し，さらに，TCP ヘッダに付加されているシーケンス番号から，特定したアプリケーションバッファ上のどのオフセット位置に書き込むべきかを特定し，特定した位置に対して専用ハードウェアが直接データを書き込む．シーケンス番号から書き込むオフセットを特定していることから，TCP の再送やネットワーク経路上でのパケット順序入れ替えによって発生した Out-Of-Order パケットを受信しても，In-Order パケットと同じ処理時間で，データを元の順序どおりに並べてバッファに書き込むことができる．なお，CPU にキャッシュや MMU がある場合は，一般的な DMA 処理と同様に，キャッシュのダーティラインの書き戻し処理や MMU の論理アドレスと物理アドレスの変換処理などが必要となるが，基本的な原理は以上のとおりである．

しかしながら，実際には受信データがアプリケーションバッファに書き込めない場合も存在する．TCP ではウィンドウベースのフロー制御のメカニズムがあるため，基本的にはアプリケーションバッファが確保されたときに，そのサイズ分のウィンドウを対向側に広告すれば，対向側はそこまでしかデータを送信してこない．しかしながら，以下の 2 つの理由により，広告ウィンドウはアプリケーションバッファより大きなサイズを広告せざるをえない場合がある．

1 つ目は，TCP のウィンドウスケールオプション[21] を使用し，たとえば n（n は 1 以上）ビット分のウィンドウスケールを行った場合，サイズをバイト単位で指定できなくなるためである．この場合，少なくともアプリケーションバッファのサイズを包含する，2 の n 乗の倍数となる若干大きめのサイズで広告する必要がある．2 つ目の理由は，TCP には Silly Window Syndrome 問題があり，送信側のその回避アルゴリズムによって，受信ウィンドウが 1 MSS（Maximum Segment Size）以下になると，送信側がデータ送信を止めてしまうためである[22),23)]．アプリケーションバッファのサイズが MSS の倍数でない限り，最後のパケットは必ず 1 MSS 未満になるので，そのままでは対向側から最後のパケットが送られてこない．よってこの場合は，それを防ぐためにアプリケーションバッファより若干大きめのウィンドウを広告する必要がある．

以上の理由により，アプリケーションバッファからはみ出たデータが送られてくる場合が生じる．さらには，アプリケーションによっては，Socket API の select() 関数でデータの受信を待つ場合など，アプリケーションバッファを指定せずにデータの受信を強いる場合もある．このような複数の理由から，データを受信した際，そのデータを書き込むべきアプリケーションバッファが存在しない場合が存在する．

そこで本方式では，その場合に限り，受信したデータを例外的に一時バッファに書き込む．そして，次に新たにアプリケーションバッファが指定されたとき，そのバッファリングしたデータをそのアプリケーションバッファの先頭にコピーするように動作することで，この問題を回避する．すなわち，原理的にダイレクト転送ができない一部の例外的なデータに関してのみ一時バッファを用いる柔軟性を加えることで，TCP/IP 通信やアプリケーションバッファ指定に制約を設けることなく，あらゆる動作条件やアプリケーションソフトウェアで利用可能とした．

一時バッファは，前述の理由により最低 1 MSS（通常は約 1.5 KB）あればよく，アプリケーションがこのサイズと比べて十分に大きいサイズのアプリケーションバッファを指定して recv() 関数をコールするようにすれば，コピー処理のオーバヘッドをほぼ解消することができる．たとえばアプリケーションバッファのサイズが 1.5 KB 以下だとすべてのデータがコピーされることになるが，15 KB だとコピーされるデータは 1/10，150 KB だと 1/100 と十分小さくなる．この場合，本方式によってメモリアクセス回数を従来比で約 1/3 にすることができるので，約 3 倍の処理効率の向上が見込める．

3.4　パイプライン処理方式

ダイレクト転送方式に加え，さらにハードウェアの処理効率を高めるための TCP/IP プロトコル処理のパイプライン化について述べる．

図 5 に受信処理の様子を示す．パイプライン化を行わない従来のシーケンシャルな処理では，パケットを受信するとまず TCP/IP のヘッダの解析を行い，パケットが属するコネクションを識別する．コネクションを識別すると，次にそのコネクションのコネクション情報（現在どのシーケンス番号まで受信したか，ACK シーケンス番号，データを書き込むバッファのアドレス，など当該コネクションに関連する諸々のコンテキスト情報）を読み出し，これらの情報を用いて受信データのメモリ上の書き込みアドレスを判定する．アドレスを判定すると実際にそのアドレスに対してコピーを行い，また，コピーすると同時に TCP データのチェックサム計算を行う．すべてのデータを書き込み終わり，また，チェックサムが正しい（データが壊れていない）ことを確認すると，パケットを受信したことを記憶する

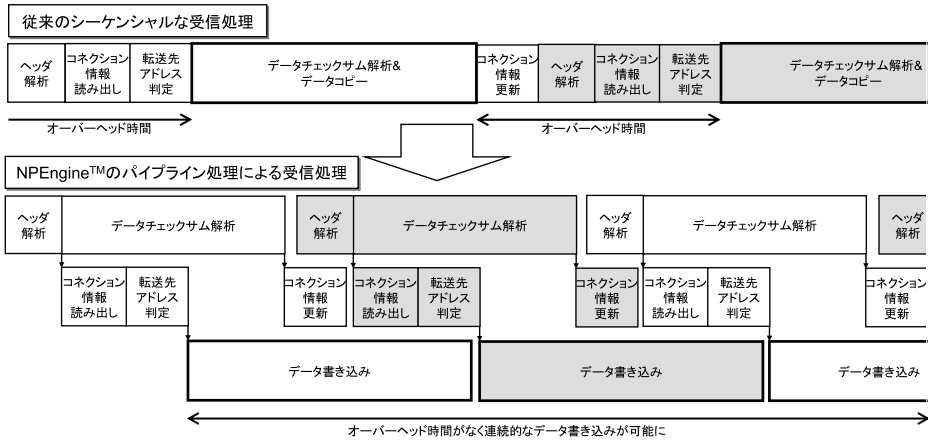


図 5　従来のシーケンシャルな受信処理（上）と NPEngine^{TM} のパイプライン処理による受信処理（下）
Fig. 5　Conventional sequential receiving processing (upper) and pipelined receiving processing of NPEngine^{TM} (lower).

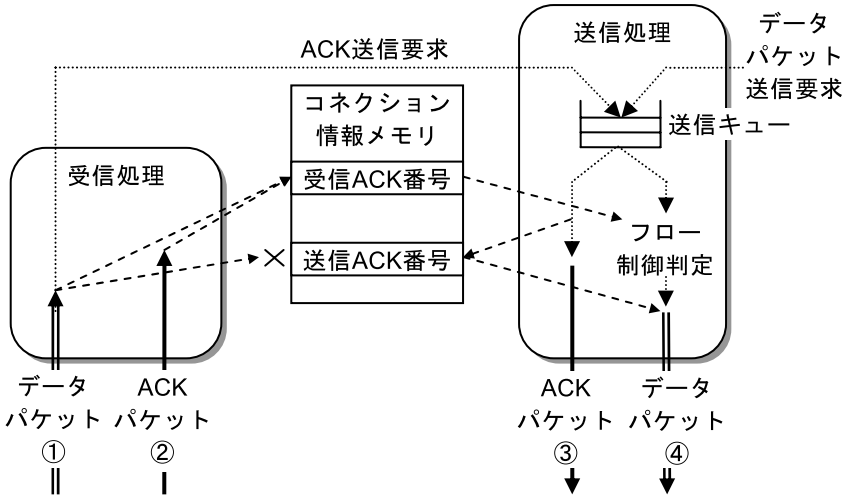


図 6　ACK 処理の概要
Fig. 6　Overview of ACK processing.

ため，受信したヘッダ情報に基づいてコネクション情報を更新する．このようにパケット処理は複数のステップで構成されている．しかしながら，通信処理という観点では，これのうちデータコピー処理以外の処理の時間はすべてオーバヘッド時間といえる．

そこで本方式では，ハードウェアの並列性を生かし，これらの複数のステップをパイプライン処理することで，このオーバヘッド時間を解消する．まず，これらの処理系を，ヘッダ&データチェックサムの解析を行うパケット解析レーン（図中上段），コネクション情報の読み書きと判定処理を行う制御レーン（図中中段），データ書き込みを行うデータ転送レーン（図中下段）の 3 レーンに分け，これらを図のように並列動作させる．あくまで各処理の順序は変えられないため，単一パケットの処理を並列化するのではなく，あるパケットのデータ書き込みを行っている最中に，次のパケットの前処理を並行して行うように動作する．これによって，あるパケットのデータ書き込みとその次のパケットのデータ書き込みを，空き時間なく連続して実行することが可能になるため，前述のオーバヘッド時間を削減することができ，ひいては処理効率を高めることが可能となる．

3.5　ハードウェア ACK 処理方式

TCP の ACK 処理の頻度はデータパケット並みに高いことから，効率化のためにはハードウェアで処理することが必須である．ACK 処理をハードウェアで行うこと自体についてはたとえば文献 18) にも記載があるが，具体的な処理方式が明示されていないため，本節ではそれについて説明する．なお，TCP の ACK 処理は，TCP 仕様に従った種々の複雑なバリエーションがあるが，以下ではそのうち基本的な ACK 処理で代表して動作を説明する．

TCP は双方向通信なので，一般に TCP/IP オフロードエンジンはパケットの受信処理回路と送信処理回路を別々に備える．受信処理側はパケットの受信を契機に動作し，送信処理側はパケットの送信を契機に動作するため，これらは互いに非同期で動作している．よって，パケットの受信を契機に動作する受信処理側で発生した ACK 応答要求に基づいて送信処理側で ACK パケットを送信するためには，ACK 応答要求を受信処理側から送信処理側に乗せ換える必要があり，またその際の ACK 番号やウィンドウサイズなどの ACK 情報の伝達に注意が必要である．

ACK 情報の受け渡しに関して，ACK 番号で代表して**図 6** にその概要を示す．受信処理側と送信処理側は，コネクション情報メモリによって ACK 番号などのコネクション情報を共有している．まず，データパケット（図中 ①）または ACK パケット（図中 ②）を受信した際，そのパケットに含まれる ACK 番号（図中の受信 ACK 番号）を送信処理側に伝えるため，ここは単純にコネクション情報メモリに書き込む．送信処理側ではデータパケッ

ト（図中 ④）を処理する際，この情報を TCP フロー制御の判定に利用する．一方で，受信したパケットがデータパケットの場合は ACK 応答する必要があるので，受信したデータのシーケンス番号から，次に受信を期待するシーケンス番号を応答する ACK 番号（図中の送信 ACK 番号）とし，その番号を添えて ACK 送信要求を送信処理側に渡し，送信キューに詰める形で送信処理側への乗せ換えを行う．ほどなく送信キューから ACK 送信要求が取り出されると，それをもとに ACK パケット（図中 ③）が送信される．

ここで，単純に送信 ACK 番号も受信 ACK 番号と同様に，受信処理側からコネクション情報メモリに直接書き込み，送信処理側がパケット送信の際にそれを読み出すという方法が考えられる．その方がより早く送信 ACK 番号を送信処理側に伝えられ，すでにキューに溜まっている送信パケット処理でこれをただちに読み出せば，結果的に ACK 応答時間が速くなるという利点も考えられる．しかしながら，この方法では次のような問題が生じる．

たとえば，シーケンス番号が連続したデータパケットを 6 つバースト的に受信した際，TCP の遅延 ACK[23] の仕組みにより 2 パケットに 1 パケットの頻度で ACK パケット送信要求が発生するので，3 つの ACK 送信要求が発生する．ACK 送信要求は送信キューを介して要求が実行されるが，いくらか遅延が発生する．特に，他のデータパケットの送信要求がキューに溜まっている場合などは，それらの送信を待つことになる．よって，ACK パケットを送信する際にはデータパケットの受信がすでに止まっている場合があり，この場合は，受信した最後のデータパケットから求められた送信 ACK 番号がコネクション情報メモリに書き込まれている状態で，送信処理側で 3 つの ACK パケットの送信処理が連続して行われる．結果的にこれらは同じ ACK 番号を付加して ACK パケットを応答することになり，いわゆる重複 ACK として相手に応答され，相手に不要なデータ再送を促してしまう．これは明らかに期待する動作とは異なる．

そこで本方式では，ACK パケット送信要求の中に ACK 送信番号を含め，それをもとに ACK パケットを送信することで，送信 ACK 番号を正しく増加させながら連続する ACK パケットを送信可能とし，さらに，これらの ACK パケットの送信処理の際に，送信 ACK 番号を受信処理側ではなく送信処理側からコネクション情報メモリに書き込む．こうすることによって，図のように後続するデータパケット（図中 ④）にこの送信 ACK 番号を含めることができ，送信 ACK 番号の受信側から送信側への受け渡しを適切に行うことができる．

以上の方式により，非同期な送信処理と受信処理の間での適切な ACK 情報の受け渡しを行い，これによってハードウェアによる高速な ACK 処理を実現可能とした．

3.6　専用ハードウェア構成

以上の提案方式を実現する専用ハードウェアの構成を**図 7** に示す．点線が制御の流れ，太線実線がデータの流れを示している．専用ハードウェアは，3.5 節で述べたとおり送信処理部と受信処理部に分けられ，これらは CPU からのレジスタ設定や割込み通知などを受け付けるホスト I/F 部を介して制御される．送信処理部は，ホスト CPU などが準備した送信メモリ上の送信データまたは制御パケットを DMA で読み出して MAC へパケットを出力し，受信処理部は，MAC からパケットを入力してデータまたは制御パケットとして受信メモリに DMA で書き込んでホスト CPU に渡す．これらの処理の際に，コネクション情報メモリを用いて必要なコネクションごとのコンテキスト情報を，ホスト CPU，送信処理部，受信処理部の間で共有しながら処理を行う．なお，送信メモリ，受信メモリ，コネクション情報メモリは説明の便宜上分けているが，ホスト CPU のワークメモリも含め，これらは物理的に同じメモリであってもよい．

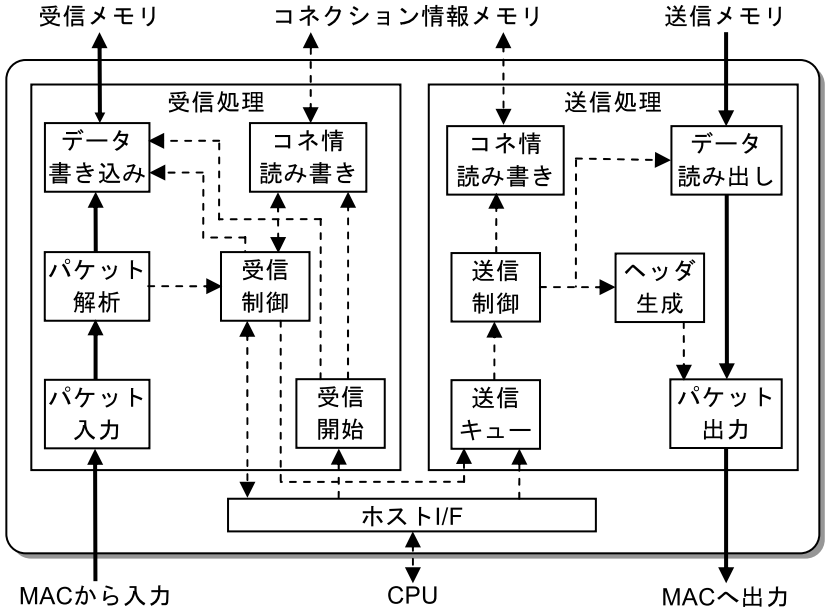


図 7　専用ハードウェア構成
Fig. 7　Block diagram of dedicated hardware.

受信処理部を詳細に説明すると次のようになる．まず，受信開始部がホスト CPU からの受信開始時を受け，アプリケーションバッファの各種初期化処理を行う．その後に MAC からパケットを受けると，パケット入力部で MAC からのパケット入力の I/F 処理を行い，パケット解析部でパケットのヘッダ解析とチェックサム解析を行う．受信制御部はその結果を受け，解析したコネクションを指定して，コネクション情報読み書き部に対してコネクション情報の探索と読み出しを指示する（探索は Linux の TCP/IP スタックなどと同様のハッシュリストベース）．これら解析されたパケット情報と読み出されたコネクション情報をもとに，受信制御部は各種受信処理判定を行い，受信したパケットがデータパケットの場合は，データ書き込み部がデータを抽出して受信メモリ上のアプリケーションバッファまたは一時バッファのアドレスに DMA でデータを書き込む．ここで一時バッファに書き込まれたデータは，後ほど受信開始部が次の受信開始処理を行った際，データ書き込み部によって次のアプリケーションバッファに DMA でコピーされる．以上のデータパケットの処理が終わると，コネクション情報読み書き部によって，コネクション情報の更新書き込みが行われる．一方，受信したパケットが制御パケットの場合は，ホスト CPU によって指定された制御パケットのバッファアドレスをホスト I/F から取得し，データ書き込み部がそこに DMA で書き込んでホスト CPU に通知する．

一方，送信処理部は，まず送信キュー部がホスト I/F からのデータ送信要求，制御パケット送信要求，受信処理部からの ACK パケット送信要求などを受け，これらをキューイングする．キューイングされた要求は順次取り出され，送信制御部によってパケット送信処理が制御される．送信制御部はまずコネクション情報読み書き部にコネクション情報の読み出しを指示し，読み出されたコネクション情報をもとに，送信パケットがデータパケットの場合はヘッダ生成部にヘッダ生成指示，データ読み出し部に送信するデータの DMA による読み出し指示を行い，パケット出力部がこれらをマージしてパケットとして構成し，MAC へ出力する．パケット出力部では TCP のチェックサム計算も行い，チェックサム値をヘッダに付加する．データパケットの場合は，1 つの送信要求でまず送信メモリ上のアプリケーションバッファの先頭から 1 パケット分のデータが送信され，アプリケーションバッファにデータがまだ残っている場合は，再びデータ送信要求がキューイングされ，それによって 2 つ目のパケット送信が行われ，という具合に送信するデータがパケットごとに分割されて送信される．送信パケットが ACK パケットの場合は，データ読み出しが行われず，ヘッダ生成部によってヘッダのみが生成され，MAC に出力される．制御パケットの場合は，ヘッダは生成されず，データ読み出し部によって制御パケット全体が DMA で読み出され，MAC に出力される．また，出力するパケットが制御パケット以外の場合は，コネクション情報読み書き部が，今回の処理によって更新されたコネクション情報の更新書き込みを行う．

なお，TCP のフロー制御，輻輳制御，ACK 制御などの各種の複雑な処理は，受信制御部と送信制御部がパケットごとに従来のソフトウェアと同等の判定・場合分けをハードウェア・ロジックで処理することで実現している．

4. 実装と性能評価

4.1 概　　要

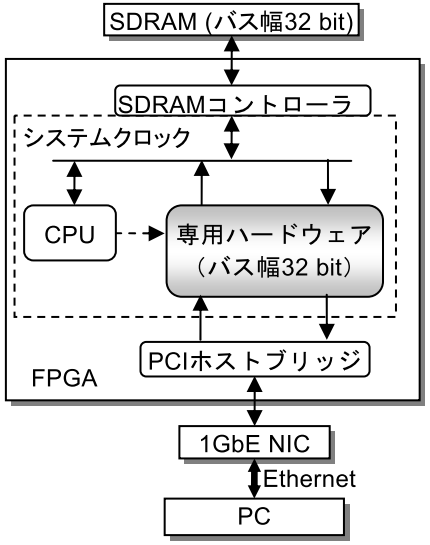
NPEngine^{TM} の性能を評価するため，FPGA 上に実装して TCP のデータ受信性能を評価した．実装は Gigabit Ethernet 向けの 1 Gbps 版 NPEngine^{TM} と，10 Gigabit Ethernet 向けの 10 Gbps 版 NPEngine^{TM} の 2 通り行った．1 Gbps 版に関しては，32 bit 組み込み CPU との親和性を考慮してデータのバス幅を 32 bit とし，最大動作周波数に関しては，前述したダイレクト転送，パイプライン処理，ハードウェア ACK 処理の組合せにより，伝送レート = 動作周波数 × データバス幅 × メモリコントローラ効率（約 0.8），程度が見込まれることから，1 Gbps を実現するために必要な動作周波数 40 MHz を最大動作周波数と設定した．10 Gbps 版に関しては，動作周波数をそのまま 10 倍の 400 MHz とすると FPGA での実装が困難になるため，実現容易な周波数としてその 1/4 の 100 MHz とし，代わりにデータバス幅を 4 倍の 128 bit とした．これらを**表 1** にまとめる．専用ハードウェアは Verilog-HDL で実装し，データバス幅をパラメータ化することによって，1 Gbps 版と 10 Gbps 版のコードを共通化した．機能としてはともに TCP 基本仕様である RFC793 [2] に準拠，輻輳制御アルゴリズムは NewReno [24] である．以降，それぞれについて実装した構成および評価結果を説明する．

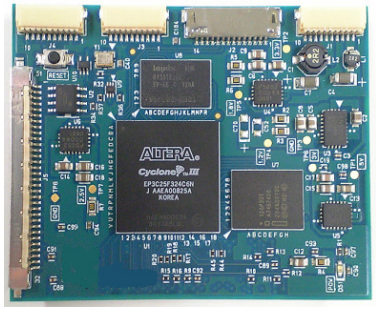
4.2 1 Gbps 版 NPEngine^{TM}

評価環境の構成図と FPGA ボード外観を**図 8** に示す．FPGA（Altera® 社 Cyclone® III [25]）には内部バス幅 32 bit の専用ハードウェア，ホスト CPU としての 32 bit 組み込み

表 1　実装した NPEngine^{TM} のデータバス幅と最大動作周波数
Table 1　Data bus width and maximum operational frequency of implemented NPEngine^{TM}.

	データバス幅[bit]	最大動作周波数[MHz]
1 Gbps 版 NPEngine^{TM}	32	40
10 Gbps 版 NPEngine^{TM}	128	100





（mini-PCIソケットは基板裏側）

図 8　評価環境の構成（左）と FPGA ボード外観（右）
Fig. 8　Block diagram of evaluation environment (left) and top view of FPGA board (right).

CPU（Altera® 社 Nios®II[26]，Fast 構成，4 KB キャッシュ，MMU なし，OS は Nios®II 用パッチを適用した uClinux 2.6.19-uc1），SDRAM コントローラなどを組み込み，これらをシステムバスで接続した．SDRAM および SDRAM コントローラは，専用ハードウェアが書き込んだデータをホスト CPU のアプリケーションなどが読み出したり，また，専用ハードウェアがバスを占有しないようにしたりするための帯域を確保する必要があるため，システムクロックの倍の周波数で動作させた．ネットワークインタフェースのため PCI ホストブリッジも組み込み，一般的な Gigabit Ethernet 対応 mini-PCI ネットワークインタフェースカードを接続した．組み込み CPU を用いた従来のソフトウェア（前述の uClinux）による TCP/IP 処理と比較するため，以上の構成から専用ハードウェアを除き，従来の OS によるプロトコルスタックを使用するように構成したものも比較対象として用意した．これらの構成で，それぞれ十分な性能を持つ PC と Ethernet（MTU 1,500 Byte）で接続し，システムクロックの動作周波数を変化させながら，CPU 上のテストアプリケーションから Socket API の受信関数 recv() をデータ長 4 MB で連続的にコールした場合の，FPGA のコア消費電力およびアプリケーションデータの伝送レートを計測した．

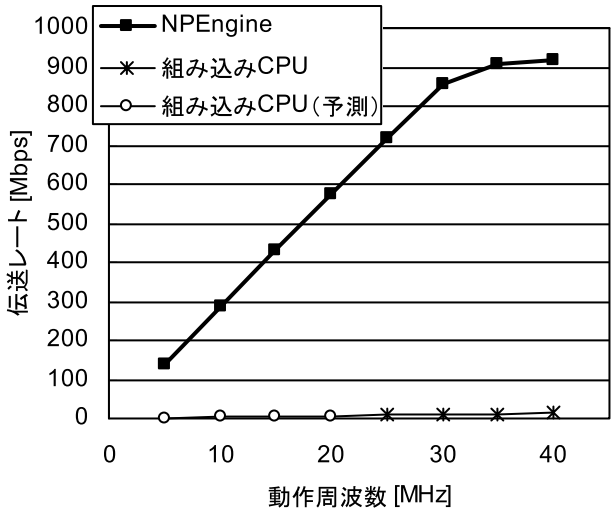


図 9　同じ周波数におけるソフトウェア処理との伝送レート比較
Fig. 9　Comparison of transfer rates of software-based system and NPEngine™ at same frequency.

動作周波数あたりの伝送レートの評価結果を**図 9** に示す．双方とも動作周波数にほぼ比例して伝送レートが上昇しており，NPEngine™ に関しては周波数 30 MHz 程度からリンク帯域 1 Gbps による制約で伝送レートが飽和し始めている様子が分かる．なお，PCI ホストブリッジの都合上，組み込み CPU の方は動作周波数 25 MHz より低い周波数での動作が困難であったため，25 MHz 以上の測定値から線形予測（伝送レート [bps] = 0.33 [bps/Hz] × 動作周波数 [Hz]）した値を記載している．組み込み CPU の動作周波数あたりの伝送レートは実測値 25〜40 MHz の範囲で 0.32〜0.34 bps/Hz，NPEngine™ の動作周波数あたりの伝送レートは飽和前の〜30 MHz までで 28.0〜28.6 bps/Hz となっており，NPEngine™ は組み込み CPU と比較して約 80 倍の伝送レートを実現していることが分かる．また，最大動作周波数 40 MHz で 918 Mbps と 1 Gbps に近い性能が出ていることから，期待どおりの性能が出ていることも分かる．なお，ヘッダなどによるオーバヘッドを除いた伝送レートの理論限界値 949 Mbps からの低下要因は，主に Socket API の recv() 関数のコールに関わるソフトウェア処理のオーバヘッドによるものである．

また，同じく 32 bit 組み込み CPU として代表的な ARM®920T（動作周波数 200 MHz）についても同じ測定を行ったところ，伝送レートが約 65.5 Mbps となり，動作周波数あた

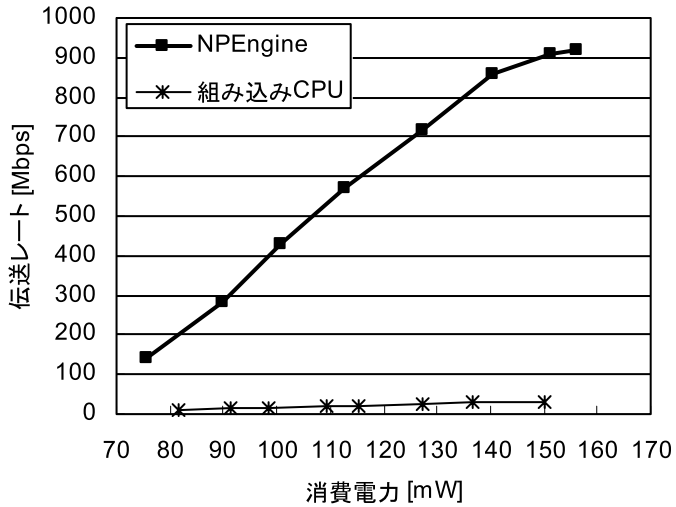


図 10　同じ消費電力におけるソフトウェア処理との伝送レート比較
Fig. 10　Comparison of transfer rates of software-based system and NPEngine™ at same power consumption.

りの伝送レートが Nios®II とほぼ同じ約 0.32 bps/Hz となった．すなわち，以上の組み込み CPU の結果は ARM®920T でも同様になると推定される．

次に消費電力に関して，ソフトウェア処理と比べて NPEngine™ は回路を追加している分，同じ動作周波数では消費電力の増大が避けられないが，伝送レートが前述のように大幅に向上している．そこで，有効性を明確にするために消費電力あたりの伝送レートとして測定データを整理した．結果を**図 10** に示す．動作周波数あたりの伝送レートの結果と同様に，NPEngine™ はソフトウェア処理と比較して消費電力あたりでも大幅に高い伝送レートを実現しており，たとえば消費電力 80 mW 付近では組み込み CPU が約 9 Mbps なのに対して NPEngine™ は約 200 Mbps，消費電力 140 mW 付近では組み込み CPU が約 30 Mbps なのに対して NPEngine™ が約 858 Mbps であり，NPEngine™ はソフトウェア処理の 22〜29 倍の伝送レートを実現していることが分かる．なお，以上は受信について評価を行っているが，送信についても同様なので，結果は省略する．

以上，NPEngine™ はソフトウェア処理と比較すると，回路規模は増えるものの大幅に高い性能を実現することができることが分かる．**表 2** に回路規模を含めた比較結果をまとめる．

表 2　ソフトウェア処理との比較まとめ
Table 2　Summary of comparison of software-based system and NPEngine™.

	32 bit 組み込み CPU	**32 bit NPEngine™**
追加回路規模	なし	**129K ゲート(*1)**
動作周波数 あたりの伝送レート	0.32〜0.34 bps/Hz	**28.0〜28.6** **bps/Hz**
同一消費電力(*2) における伝送レート	80 mW 時 9 Mbps 140 mW 時 30 Mbps(*3)	**80 mW 時 200 Mbps** **140 mW 時 858 Mbps(*3)**

(*1)・・・Logic Element(LE)数から 1 LE = 12.5 ゲートで換算
(*2)・・・Altera®社 Cyclone®III FPGA 全体のコア消費電力
(*3)・・・Altera®社 Nios®II（Fast 構成）使用時

4.3　10 Gbps 版 NPEngine™

評価環境の構成図と FPGA ボード外観を**図 11** に示す．FPGA（Altera 社 StratixII GX[27]）には内部バス幅 128 bit の専用ハードウェア，ホスト CPU としての 32 bit 組み込み CPU（Altera® 社 Nios®II[26]，Fast 構成），SDRAM コントローラなどを組み込み，これらをシステムバスで接続した．SDRAM および SDRAM コントローラは，NPEngine™ が書き込んだデータをホスト CPU のアプリケーションなどが読み出す分の帯域も確保する必要があるため，システムクロックの倍の周波数で動作させた．専用ハードウェアのバス幅は 128 bit となっており，また，ネットワークインタフェースのために FPGA 内部に 10 Gigabit Ethernet 向け MAC も組み込み，CX-4 ケーブル（XAUI[28]）で対向となる PC サーバマシンと Ethernet（MTU 1,500 Byte）で接続した．以上の構成で，システムクロックの動作周波数を変化させながら，CPU 上のテストアプリケーションから Socket API の受信関数 recv() をデータ長 4 MB で連続的にコールした場合の，FPGA のコア消費電力およびアプリケーションデータの伝送レートを計測した．

結果を**図 12** の「HW 補助無し」に示す．見て分かるように，周波数の上昇にあわせて伝送レートが上昇し，90 MHz 付近で飽和している．しかしながら，最大伝送レートが 8.7 Gbps であり，1 Gbps 版に比べて TCP データの理論限界 9.49 Gbps との乖離が大きい．これは，1 Gbps 版同様，recv() コールに関わるソフトウェア処理のオーバヘッドによるものであるが，10 Gbps では単位時間あたりに実行される recv() コールが 1 Gbps 時の 10 倍になり，CPU の動作周波数の向上分以上に頻度が上がっているため，オーバヘッドがそれによって増加しているためと考えられる．

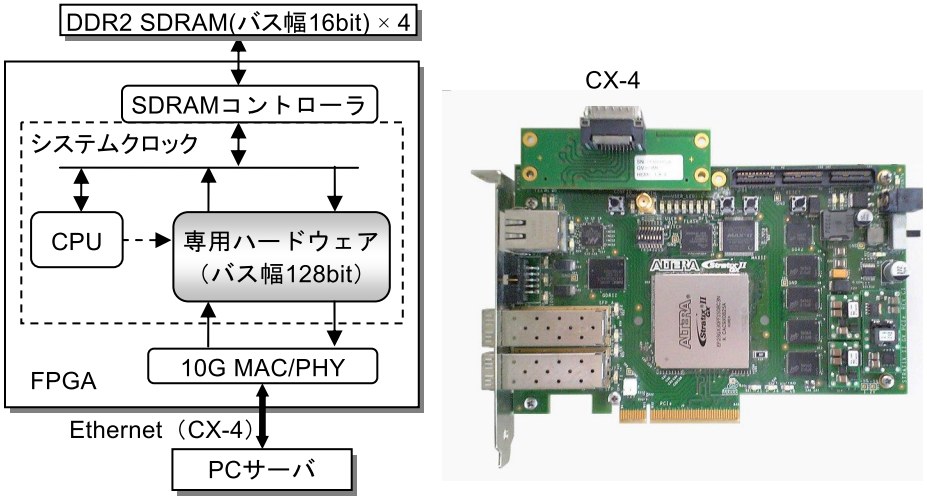


図 11　評価環境の構成（左）と FPGA ボード外観（右）
Fig. 11　Block diagram of evaluation environment (left) and top view of FPGA board (right).

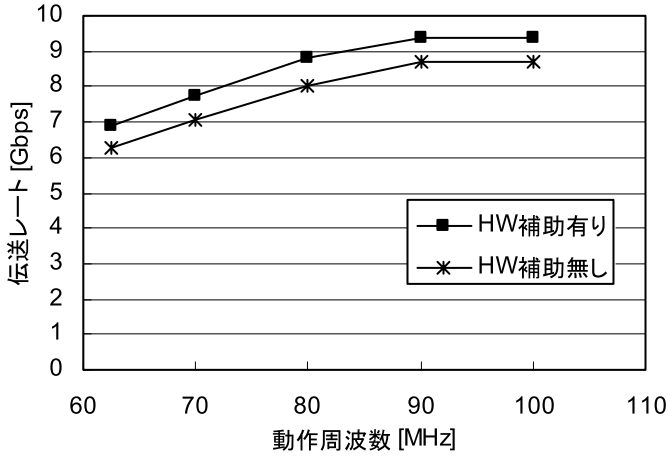


図 12　10 Gbps 版 NPEngine^{TM} の受信伝送レート
Fig. 12　Receiving transfer rates of 10 Gbps NPEngine^{TM}.

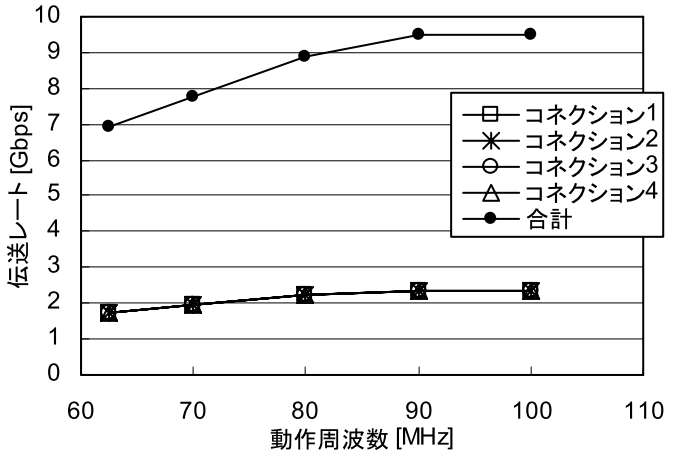


図 13　10 Gbps 版 NPEngine^{TM} のマルチコネクションでの受信伝送レート
Fig. 13　Receiving transfer rates of 10 Gbps NPEngine^{TM} in multi-connection transfer.

そこで，recv() 関数をコールする処理（専用ハードウェアに対してアプリケーションバッファを指定して受信指示を行う処理）についてもハードウェアを追加して連続実行させる（HW 補助）ことで，このオーバヘッドの排除を試みたのが図 12 の「HW 補助有り」である．見て分かるように，90 MHz 以上で 9.4 Gbps とほぼ理想的な伝送レートを実現することができている．このように受信指示を行う処理についても簡単なハードウェア補助を行うことで，10 Gbps 版 NPEngine^{TM} は 90 MHz で理論限界近くの伝送レートを実現できることが分かる．

以上の HW 補助ありの条件で，4 つのコネクションでデータ受信を行った結果を**図 13** に示す．4 つのコネクションそれぞれに均等に伝送レートが振り分けられており，また，それらを合計すると図 12 とほぼ同等の伝送レートになるので，マルチコネクションでの通信も適切に行われていることが分かる．さらに，送受双方向同時通信の結果を**図 14** に示す．双方向通信でも送受ともに片方向通信時とほぼ同一の伝送レートが実現できており，送受の間での不均衡もほとんどもないことが分かる．このように，NPEngine^{TM} はマルチコネクションや双方向同時通信においても安定して高い伝送レートを実現できることが分かる．

消費電力については，前述の 1 Gbps 版の実測値をもとにして，

消費電力 $\propto$ 回路規模 $\times$ 動作周波数

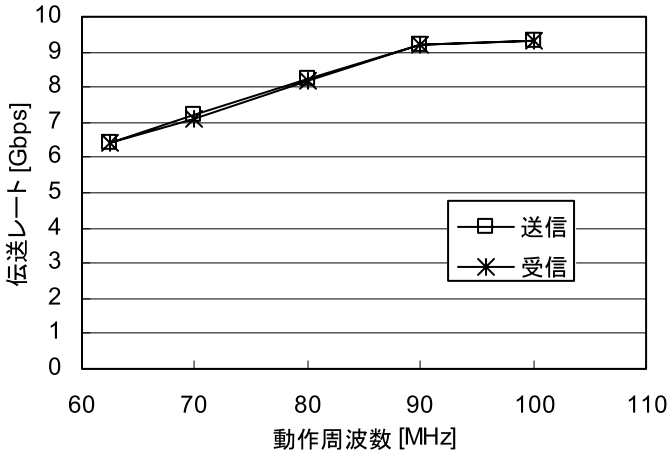


図 14　10 Gbps 版 NPEngine^{TM} の双方向同時通信の伝送レート
Fig. 14　Transfer rates of NPEngine^{TM} in simultaneous and bi-directional transfer.

表 3　1 Gbps 版 NPEngine^{TM} および 10 Gbps 版 NPEngine^{TM} の FPGA コア消費電力
Table 3　FPGA core power consumptions of 1 Gbps NPEngine^{TM} and 10 Gbps NPEngine^{TM}.

	回路規模 [ゲート(*1)]	動作周波数 [MHz]	消費電力 [mW]
1 Gbps 版 NPEngine^{TM} 単体	129K	40	58
1 Gbps 版 FPGA 全体	296K	40, 66, 80	156(*2)
10 Gbps 版 NPEngine^{TM} 単体	**202K**	**90**	**203**
10 Gbps 版 FPGA 全体	**483K**	**90, 156.25, 180**	**564**

(*1)・・・Logic Element(LE)数から 1 LE = 12.5 ゲートで換算
(*2)・・・図 10 に記載の実測値

と簡略化し，理論限界近くの伝送レートを実現する際の FPGA コア消費電力（FPGA は 1 Gbps 版と同じ FPGA [25] を仮定）の予測を行った．結果を**表 3** に示す．参考のため 1 Gbps 版の値も併記している．10 Gbps 版 FPGA 全体のコア消費電力に関しては，回路全体のうち，各動作周波数のクロックドメインに属する部分ごとに回路規模 × 動作周波数を計算してこれらを積算し，その合計値に消費電力が比例するものとして，1 Gbps 版 FPGA の積算値に対する比を，1 Gbps 版の消費電力実測値に掛けて算出した．1 Gbps 版 NPEngine^{TM} 単体と 10 Gbps 版 NPEngine^{TM} 単体のコア消費電力に関しては，それぞれ対応する FPGA 全体のコア消費電力に，NPEngine^{TM} 部分の回路規模 × 動作周波数の値の FPGA 全体の積算値に占める割合を掛けて算出した．

表 4　10 Gbps 版の装置全体の消費電力
Table 4　Total power consumption of equipment with 10 Gbps NPEngine^{TM}.

	消費電力[mW]
FPGA コア	564(*1)
FPGA XAUI I/O	675(*2)
FPGA DDR2 SDRAM I/O	513(*2)
DDR2 SDRAM チップ	154(*3)
合計	1906

(*1)・・・表 3 に記載の値
(*2)・・・Cyclone®IV GX での見積もり[30]
(*3)・・・データシート[31]を元にした 4 チップ構成での 180MHz 動作時のバースト転送時の見積もり

見て分かるように，10 Gbps 版 NPEngine^{TM} の単体消費電力は 203 mW，CPU および MAC を含めた FPGA 全体のコア消費電力は 564 mW である．そのほか，**表 4** に記載のとおり FPGA の I/O 消費電力（Cyclone®III には XAUI がないため Cyclone®IV GX [29] で見積り）や DDR2 SDRAM チップの消費電力などすべて合計すると装置全体で 1,906 mW，電源効率などを考慮しても 2 W 強となる．つまり，10 Gbps 版 NPEngine^{TM} を省電力 FPGA（Cyclone®IV GX）で実装することで，全体で 2 W 程度の消費電力で 10 Gbps クラスの TCP/IP 通信が実現できることを意味している．通常，10 Gbps クラスの TCP/IP 処理を行うには 100 W クラスの PC やサーバ（CPU，メモリ，チップセット，10G NIC）などが必要であることを考えると，2 W という消費電力は圧倒的に低い消費電力であるといえる．

4.4　40 Gbit Ethernet・大規模コネクションへの拡張

40 Gbit Ethernet に向けた伝送レートの向上に関しては，評価に使用した FPGA より高速な FPGA [32] で，10 Gbps 版 NPEngine^{TM} の最大動作周波数 100 MHz の 2 倍の 200 MHz でのコンパイルおよび実機での基本動作を確認しており，同じ専用ハードウェアおよびホスト CPU を倍速で動作させることで，そのまま 20 Gbps まで拡張することが可能である．これに加えて，データのバス幅を 2 倍の 256 bit にすれば，既存の FPGA でも 40 Gbit Ethernet への対応が可能となる見込みである．

なお，周波数に関しては，必要な伝送レートを実現するためには専用ハードウェアとメモリコントローラの周波数が重要であり，データ転送処理をほとんど行わないホスト CPU は必ずしも倍速化する必要はないと考えるが，ホスト CPU に対する具体的な性能要件の整理については今後の課題である．また，ホスト CPU の周波数は，コネクション制御の応答性やその他のアプリケーション処理の処理速度など，データ転送処理以外の処理も考慮に入れて決定されものであることから，専用ハードウェアと非同期の周波数になる場合も想定されるが，そのような場合であっても，専用ハードウェアとの間で適切な非同期受け渡し処理を行えば，問題なく動作すると考える．

コネクション数の大規模化に関しては，基本的にコネクション情報メモリをコネクション数分だけ用意すれば，送信処理に関しては特に問題は生じない．受信処理に関しては，パケット受信時に行うコネクション情報の探索処理があるが，基本的にはハッシュリストベースの探索を行っているので，コネクション情報メモリ内のハッシュテーブルのエントリ数をコネクション数に比例して増加させれば，平均探索時間はほとんど増加しないため，問題は生じない．よって，コネクション数に関しては，コネクション数に比例してコネクション情報メモリを増やすことで大規模化が可能となる見込みである．

4.5　従来 TOE との比較

従来報告されている TOE のうちいくつかのものについては，動作周波数と伝送レートに関する評価結果が記載されている．内部のデータバス幅についてはいずれも記載がないためあくまで目安にはなるが，処理効率の比較が可能である．

1 Gbps 版 NPEngine^{TM} との比較は，文献 14)，文献 15) は 25 MHz 駆動で 13.1 Mbps とのことなので周波数あたりの伝送レートは約 0.5 bps/Hz，文献 17) は 125 MHz で 296 Mbps なので約 2.4 bps/Hz，文献 18) は CPU 300 MHz と専用ハードウェア 66 MHz で約 900 Mbps なので約 3〜14 bps/Hz となり，NPEngine^{TM} の表 2 記載の 28 bps/Hz はこれらのいずれよりも十分に高い値となっている．

10 Gbps 版 NPEngine^{TM} との比較としては文献 13) があげられる．これは 40 Gbps を TOE 4 つで実現しており，1 つあたりは 10 Gbps で動作周波数は 125 MHz である．NPEngine^{TM} は 90 MHz で 9.4 Gbps と 10 Gigabit Ethernet のほぼ限界レートを実現しており，文献 13) の約 2/3 の周波数で実現できていることが分かる．

以上，著者らの知る限りでは NPEngine^{TM} は従来のいずれの TOE と比較しても優れた処理効率を実現している．なお，消費電力については，いずれの文献にも比較可能なデータが見当たらなかった．回路規模については，特に TCP/IP は非常に複雑なプロトコルなため，それを実際どこまで実装しているかによって大きく変わり，さらに，ソフトウェアとの組合せで実現している場合は，その役割分担によっても大きく変わるが，比較をするのに必要なこれらの情報が十分な比較対象が見当たらなかった．

5. おわりに

高い伝送レートと低消費電力を目指した TCP/IP オフロードエンジン NPEngine^{TM} を開発した．回路規模を抑えるハイブリッド構成，データ転送処理の処理効率を最大限に高めるダイレクト転送方式，パイプライン処理方式，ハードウェア ACK 処理方式を採用し，Gigabit Ethernet 向け実装に関しては，従来の組み込み CPU に比べて動作周波数あたりで約 80 倍，消費電力あたりで 22〜28 倍の伝送レートを実現した．また，10 Gigabit Ethernet 向け実装に関しては，90 MHz で 9 Gbps 以上の伝送レートで，マルチコネクションや双方向通信においても安定した性能を実現できることを確認した．また，10 Gigabit Ethernet 向け実装の装置全体の消費電力の見積りは約 2 W となり，従来のような 100 W クラスの PC やサーバによる実現と比べて大幅に低い消費電力となった．著者らの知る限りこれらの性能を上回る TCP/IP オフロードエンジンはなく，従来にない高い処理効率を実現したといえる．

今後は幅広い通信プロトコルへの対応，無線 LAN 対応などを進めていく．

参　考　文　献

1) Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications, IEEE Standard 802.3 (2003).
2) Transmission control protocol, Internet Engineering Task Force RFC793 (Sep. 1981).
3) Internet protocol DARPA Internet program protocol specification, Internet Engineering Task Force RFC791 (Sep. 1981).
4) Stevens, W.R.: *TCP/IP Illustrated, Volume 1: The Protocols*, Addison-Wesley, Reading, MA (1994).
5) Clark, D.D., Jacobson, V., Romkey, J. and Salwen, H.: An analysis of TCP processing overhead, *Communications Magazine*, Vol.27, No.6, pp.23–29, IEEE (1989).
6) Foong, A., Huff, T., Hum, H., Patwardhan, J. and Regnier, G.: TCP performance re-visited, *Proc. IEEE International Symposium on Performance of Systems and Software* (Mar. 2003).
7) Currid, A.: TCP offload to the rescue, *Queue*, Vol.2, No.3, pp.58–65 (2004).
8) Henriksson, T., Nordqvist, U. and Liu, D.: Embedded Protocol Processor for Fast

and Efficient Packet Reception, *Proc. 2002 IEEE Int'l Conf. Computer Design: VLSI in Computers and Processors* (*ICCD 02*), pp.414–419, IEEE CS Press (2002).
9) Hoskote, Y., Bloechel, B.A., Dermer, G.E., Erraguntla, V., Finan, D., Howard, J., Klowden, D., Narendra, S., Ruhl, G., Tschanz, J.W., Vangal, S., Veeramachaneni, V., Wilson, H., Xu, J. and Borkar, N.: A TCP Offload Accelerator for 10 Gb/s Ethernet in 90-nm CMOS, *IEEE Journal of Solid-State Circuits*, Vol.38, No.11, pp.1866–1875 (2003).
10) Wun, B. and Crowley, P.: Network I/O Acceleration in Heterogeneous Multicore Processors, *14th IEEE Symposium on High-Performance Interconnects* (*HOTI'06*), pp.9–14 (2006).
11) Uchida, T.: Hardware-based TCP processor for Gigabit Ethernet, *IEEE Trans. Nuclear Science*, Vol.1, pp.1631–1637 (2008).
12) 田上敦士ほか：ハードウェア記述言語による TCP/IP プロトコルの実装，信学技報，CPSY-2000-9, Vol.100, No.86, pp.17–24 (2000).
13) Shrikumar, H.: 40 Gbps De-Layered Silicon Protocol Engine for TCP Record, *Proc. Design, Automation and Test in Europe, 2006, DATE '06*, Vol.1, pp.1–6 (Mar. 2006).
14) 橋本浩二ほか：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，通信，組込技術とネットワークに関するワークショップ ETNET2008，信学技報 DC，ディペンダブルコンピューティング，Vol.107, No.559 (2008).
15) Hashimoto, K. and Moshnyaga, V.G.: A new approach for TCP/IP offload engine implementation in embedded systems, *2010 Conference Record of the 44th Asilomar Conference on Signals, Systems and Computers* (*ASILOMAR*), pp.1249–1253 (Nov. 2010).
16) Chung, S.M., Li, C.Y., Lee, H.H., Li, L.H., Tsai, Y.C. and Chen, C.C.: Design and implementation of the high speed TCP/IP Offload Engine, *Proc. IEEE International Symposium on Communications and Information Technologies*, pp.574–579 (Oct. 2007).
17) Wu, Z.-Z. and Chen, H.-C.: Design and Implementation of TCP/IP Offload Engine System over Gigabit Ethernet, *ICCCN 2006, Proc. 15th International Conference*, pp.245–250 (Oct. 2006).
18) Jang, H., Chung, S.H., Kim, D.K. and Lee, Y.S.: An Efficient Architecture for a TCP Offload Engine Based on Hardware/Software Co-design, *J. Inf. Sci. Eng.*, Vol.27, No.2, pp.493–509 (2011).
19) Jang, H., Chung, S.H. and Yoo, D.H.: Design and implementation of a protocol offload engine for TCP/IP and remote direct memory access based on hardware/software coprocessing, *Microprocessors & Microsystems*, Vol.33, No.5-6 (Aug. 2009).
20) Chu, H.K.J.: Zero-Copy TCP in Solaris, *Proc. USENIX 1996 Annual Technical Conference*, San Diego, California (Jan. 1996).
21) TCP Extensions for High Performance, Internet Engineering Task Force RFC1323 (May 1992).
22) Window and Acknowledgment Strategy in TCP, Internet Engineering Task Force RFC813 (July 1982).
23) Requirements for Internet Hosts – Communication Layers, Internet Engineering Task Force RFC1122 (Oct. 1989).
24) The NewReno Modification to TCP's Fast Recovery Algorithm, Internet Engineering Task Force RFC2582 (Apr. 1999).
25) Altera® 社 Cyclone®III, available from ⟨http://www.altera.co.jp/products/devices/cyclone3/cy3-index.jsp⟩.
26) Altera® 社 Nios®II, available from ⟨http://www.altera.co.jp/products/ip/processors/nios2/ni2-index.html⟩.
27) Altera® 社 Stratix®II GX, available from ⟨http://www.altera.co.jp/products/devices/stratix-fpgas/stratix-ii/stratix-ii-gx/s2gx-index.jsp⟩.
28) XGMII Extender Sublayer (XGXS) and 10 Gigabit Attachment Unit Interface (XAUI), IEEE 802.3, Section 4, Chapter 47.
29) Altera® 社 Cyclone®IV GX, available from ⟨http://www.altera.co.jp/products/devices/cyclone-iv/cyiv-index.jsp⟩.
30) Altera® 社 Cyclone®III and Cyclone®IV PowerPlay Early Power Estimator, available from ⟨http://www.altera.co.jp/support/devices/estimator/cy3-estimator/cy3-power_estimator.html⟩.
31) Samsung 社，DDR2 SDRAM, available from ⟨http://www.samsung.com/global/system/business/semiconductor/product/2009/1/13/050459ds_k4t1gxx4qe_rev11.pdf⟩.
32) Altera® 社 Stratix®IV, available from ⟨http://www.altera.co.jp/products/devices/stratix-fpgas/stratix-iv/stxiv-index.jsp⟩.

(平成 23 年 6 月 27 日受付)
(平成 23 年 9 月 12 日採録)

推　薦　文

本論文は高い性能を持つ TCP/IP 通信プロトコルをハードウェアで実現しただけでなく，

最近社会から強く求められている低消費電力化も両立させて実現した研究について述べたものである．高速な TCP/IP 通信の実現とその低消費電力化の両立は今後ネットワーク構築において必須の要求項目となる可能性が高い．本論文は理論とアイデアを提出しただけでなく，実装し評価を行ったことについても述べており信頼性が高く，内容もユニークで新規性があり，有用性が高い．以上の理由により，本論文を受賞論文に推薦する．

（インターネットと運用技術研究会主査　山之上卓）

田中　信吾

平成 13 年東京大学大学院工学系研究科機械工学専攻修士課程修了．同年（株）東芝入社．通信プロトコル，組み込みシステム関連の研究開発に従事．現在，（株）東芝研究開発センターネットワークシステムラボラトリー研究主務．

山浦　隆博

平成 18 年北海道大学大学院情報科学研究科メディアネットワーク専攻修士課程修了．同年（株）東芝入社．現在，（株）東芝研究開発センターネットワークシステムラボラトリーで，通信プロトコル，組み込みシステム関連の研究開発に従事．電子情報通信学会会員．

山口　健作

平成 12 年東京大学大学院理学系研究科物理学専攻修士課程修了．同年（株）東芝入社．情報セキュリティ，通信プロトコル関連の研究開発に従事．現在，（株）東芝研究開発センターネットワークシステムラボラトリー研究主務．

菅沢　延彦

平成 15 年慶應義塾大学大学院政策・メディア研究科修士課程修了．同年（株）東芝入社．通信プロトコル，組み込みシステム関連の研究開発に従事．現在，（株）東芝研究開発センターネットワークシステムラボラトリー研究主務．電子情報通信学会，日本ソフトウェア科学会各会員．

谷澤　佳道（正会員）

平成 14 年慶應義塾大学大学院理工学研究科開放環境科学専攻修士課程修了．同年（株）東芝入社．通信プロトコル，ネットワークシステム関連の研究開発に従事．現在，（株）東芝研究開発センターネットワークシステムラボラトリー研究主務．

渋谷　尚久

平成 10 年早稲田大学大学院理工学研究科情報ネットワーク専攻修士課程修了．同年（株）東芝入社．無線システム・通信プロトコル関連の研究開発に従事．平成 19 年（株）東芝 PC&ネットワーク社へ異動，PC 等デジタル機器向けソフト・サービス系の商品企画に従事．現在，（株）東芝デジタルプロダクツ&サービス主務．