2003－ARC－154　(7)
2003／8／5

データ駆動型ネットワークプロセッサにおける高速パケット分類処理

森川大智[†]　岩田　誠[†]

基幹ネットワークの全光化に伴って，境界ルータや家庭内ゲートウェイを高機能化・高速化する要請が高まっている．本稿では，特に処理負荷の重いパケット分類処理を対象に，アルゴリズムの高速パイプライン化を図り，パイプライン処理能力に優れたデータ駆動型プロセッサ上で効果的にソフトウェア実現するプロセッサ構成を提案する．そして，試作ハードウェアによる回路規模の評価から，数%のハードウェアコスト増で提案回路が構成可能であり，最大 12M pps(IPv4 packets per second) の性能を達成できること示す．

Fast Packet Classification on a Data-Driven Network Processor

Daichi Morikawa[†] and Makoto Iwata[†]

With developing all-optical communication networks, highly-functional and high-speed boundary routers and home gateways are becoming to be required. In this paper, high-speed pipelined algorithm for packet classification which is one of heavy load functions within boundary routers are proposed. The paper then presents its software implementation scheme on a data-driven processor having flexible capability of pipelined parallel processing. Finally, it is shown by simulation and experimental hardware evaluation that the scheme achieves maximum performance 12M IPv4 packets per second by only 6 % additional hardware cost.

1. はじめに

超高速パケットネットワーク環境の構築には，多様化する通信サービスやプロトコルを，高コストな専用 ASIC ではなく，ソフトウェアによりプログラム可能なネットワークプロセッサ NP(Network Processor) 上で実現する方式が必須になりつつある[1)]．しかしながら，昨今，多様なサービスに対応した処理が NP 単独では困難になる程，伝送リンク速度が向上してきている．このため，特に処理負荷の高いパケット分類処理や各種テーブル検索処理に専用化した高速 LSI を補助的に用いた実装方法が採られている[2)]．この方法では，NP の最大の特徴であるプログラマビリティが損なわれてしまう．

これに対して，我々は，しなやかな高速 NP の実現を目指して，パイプライン処理能力に優れたデータ駆動型処理方式に基づくネットワークプロセッサ DDNP(Data-driven network processor) の構成法を検討している．

本稿では，NP が提供する機能の中でも，ファイアウォールや QoS 制御などに欠かせないが処理負荷の高いパケット分類を，データ駆動型チップマルチプロセッサ[3)] 上でパイプライン並列に実現する方式を提案する．また，本方式により，既存の 32bit データ駆動型プロセッサに 30k ゲート弱の回路を追加するだけで，12Mpps(packets per second) の性能で IPv4 パケットを分類できることを示す．

2. データ駆動型ネットワークプロセッサ:DDNP

DDNP は，動的データ駆動処理方式を採用しているため，原理的に文脈切替のオーバーヘッドがない．このため，非決定的に到着する複数の IP パケットストリームを受理しても，各々のパケットの要求に応じた異なるプログラムをオーバヘッドなく同時並列に実行可能である．さらに，DDNP の回路実現には，大域的なクロック同期が不要な自己タイミング型同期パイプライン機構を用いている．そのため，パイプライン内の一時的な負荷変動を緩衝できるロバスト性を付与できると同時に，パケット処理に特化した命令の追加/拡張が同期型パイプラインに比べて容易である．さらに，プロセッサ間通信時の同期命令発行や厳密なスケジューリングが不要なため，既存の NP に比べてスケーラブルにチップマルチプロセッサ化が可能である．

このように原理的に優れた特徴を有する DDNP をさらに高性能化するには，徹底的なパイプライン処理を実現することが重要である．すなわち，プロセッサ内，チップマルチプロセッサ，およびマルチチップの全ての水準のパイプラインを有効に活用できる方式の検討が必要である．特に，各種のテーブル検索時やパケットキューイング時に発生するメモリアクセス遅延がパイプライン流量の低下を起こさない構成の検討が重要である．当然のことながら，従来の NP でも導入されている，ワードやバイト境界をまたぐようなデータ処理命令等，パケット処理に特化した命令セットの洗練化も同時に必要となる．

† 高知工科大学
Kochi University of Technology

3. パイプライン並列パケット分類

パケットの分類は，ネットワークセキュリティ機能やQoS機能の中心的役割を担っており，NPによる高速処理が期待されている．具体的には，入力パケットのヘッダやペイロードの中のいくつかの情報 (フィールド) を，分類のためのルールDB中の条件 (入力パケット中の複数のフィールドに対する照合規則) と照合し，照合結果に基づいて必要なパケット操作を決定する処理である．レイヤ4パケットの分類では通常，IPアドレス対 (送信側と受信側)，レイヤ4のポート番号 (送信側と受信側) とプロトコル種別の5つのフィールドが照合対象になる．

nフィールドを照合対象としたパケット分類において，分類に要する時間を最短にするには，フィールド毎に並列に分類規則の照合を行い，最後に各照合結果を統合すればよい．しかし，単純に同時並列処理を行うとルールDBを冗長に構成する必要があり，フィールド数が増えれば，必要なメモリ量が指数関数的に増大する．そこで，各フィールドの部分照合を逐次実行し，徐々に探索空間を縮退させながら照合する再帰的分類方式が多数提案されている[4]．

本稿で実現するパケット分類処理は図1に示すように，次の2種類のモジュールから構成される．

(a) Index Jump 部： 入力パケットのフィールドを基に本テーブルを参照し，該当する検索木へのポインタを出力する．

(b) Field Matching 部： Index Jump部において選択された検索木を検索する部分であり，最長一致検索アルゴリズムを用いて，各フィールド照合をパイプライン並列に実行する．

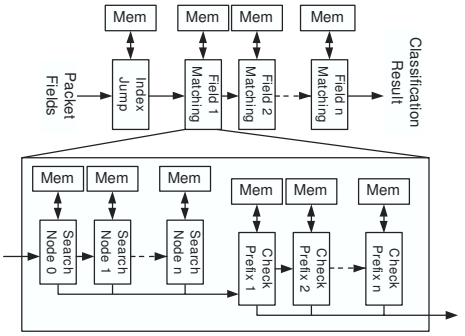

図 1 パケット分類処理のパイプライン並列化.
Fig. 1 SuperPipelined Packet Classification.

3.1 LC-Trie 木によるメモリ参照削減

パケット分類において，フィールドと規則の照合法には，

- 完全一致照合
- プレフィックス照合
- 範囲一致照合

の3通りがある．完全一致照合および範囲一致照合は，いずれもプレフィックス照合の特殊な場合として取り扱える．照合法をプレフィックス照合に限定すれば，アルゴリズムの一元化が可能となり，計算コストの削減につながる．プレフィックス照合には最長一致検索手法が用いられる．

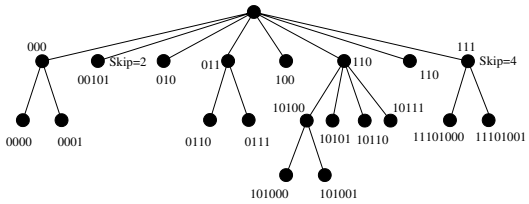

図 2 LC-Trie 木構造.
Fig. 2 LC-Trie structure.

本提案では，データ構造上パイプライン化が容易であり，かつ簡単な計算で最長一致検索が実現できる，木構造を用いたアルゴリズムの一種であるLC(Level Compress)-Trie[5]を用いる．一般的に，木構造を用いるアルゴリズムの場合，ノード数が大きくなり検索にかかる平均深度 (ルートからリーフまでのパスの平均長) が長くなると，ノード探索毎のメモリ参照が増大し検索遅延が増加する．さらに，ノード数の増加はメモリ使用量の増加につながるという問題点がある．Level Compress法は各ノードが2^kの子ノードを保持する一種のMultiWay木構造をとることでツリーを深さ方向にさらに圧縮する方式であり，検索木の深さは検索エントリ数に対し$O(\log\log n)$で増加するという特徴を持つ．これより得られたLC-Trieは平均深度を削減でき，木検索に必要なメモリ参照回数の削減およびメモリ使用量の削減が望める．

LC-Trie木を用いた最長一致検索手法は，

(1) LC-Trie木検索
(2) プレフィックス検査

の処理からなり，それぞれにメモリ参照が発生する．

たとえば，IPv4のルーティングテーブルを検索する場合，LC-Trie木の検索深度が最大5レベルあれば検索可能である．それに伴い，LC-Trie木検索の結果を受けて行われるプレフィックス検査も最大5レベルで表現できる．ただし，プレフィックス検査のためのメモリ参照頻度は低く，2回以上のメモリ参照が起こることは希である．大部分のメモリ参照はLC-Trie木の検索で起こる．IPv4の検索においては，LC-Trie木の検索に要する平均深度は2以下であり，全体で平均3回のメモリ参照により検索処理を行える．

そこで，図1に示すように，各処理で発生する1メモリ参照を1ステージとし，LC-Trie木検索，およびプレフィックス検査を各5ステージに展開し，各ステージの処理が完了し次第，次ステージに移行するようパイプラインを構成する．

3.2 LC-Trie 木の分散

パケット分類のためのルールセットの大規模化に伴って，ルールに偏りがあると，検索木が不均質になる．そこで，各フィルタの先頭数ビットを基に検索木を分割する手法[6]を導入する．これにより，サブツリーおよび各次元のバラ

ンスを改善できる．

LC-Trie 木を図 3 に示すように Index Jump (Index Jump) テーブルに格納し，各フィールドに対する照合を流れ作業的にパイプライン並列に探索する．

検索木を分割する際，および Index Jump テーブルを参照する際には，図 4 に示す計算式を用いて，各フィールドの先頭 x ビットを取り出し，これらのビットを連結した値を Index として用いる．

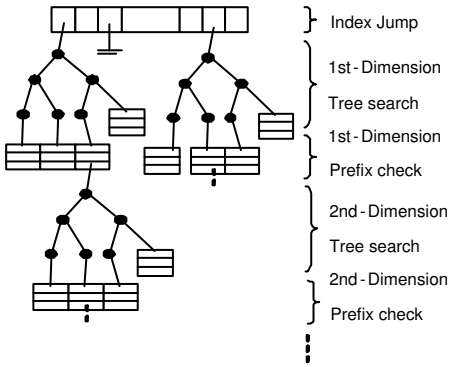

図 3 パケット分類処理の分類木構造.
Fig. 3 Structure of a packet classification tree.

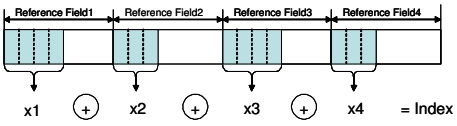

図 4 Index Jump に用いる Index 生成法.
Fig. 4 Index Construction for Index Jump.

このとき，Index Jump テーブルのエントリ数は，パケット分類に用いるフィールドのビット長 $x_i(1 \leq i \leq n)$ により決まり，$O(2x_i)$ で線形に増加するため，適切なビット選択が必要である．平均的に分散したルールセットの場合，検索木は選択ビット数にかかわらず，Index Jump テーブルにほぼ均等に割り当てられる．しかし，実際のルールセットではかなりの偏りが予想される．実際に，Mae-West のルーティングテーブル[7] から 4 次元のルールセットを作成し調べたところ，Index Jump テーブルのエントリ数を単純に増加しても，フィルタ群が特定エントリに集中する傾向が見られた．このことから，ルールセットの分布に応じて，Index Jump テーブルに必要最小限のリソース消費で実現可能なビット数をヒューリスティックに設定する必要があると言える．

例えば，生成したルールセットの先頭 2 つのフィールドを宛先/送り元 IP アドレスとし，残りを宛先/送り元ポート番号と仮定したとき，各アドレスフィールドに 4 ビット，各ポートフィールドに 2 ビットを適用し Index Jump テーブルを生成する．すると，ルール群に対して，最大サイズとなる検索木でも，分割前の約 10 分の 1 となり，実用的な検索木が生成できる．また，ローカルサイトやホームゲートウェイなどの比較的小規模な網においては大多数のフィルタの先頭ビットが同じ値を示すことが多い．これでは Index 生成用の選択ビットとして機能せず，Index Jump テーブル内の有効データ数が少なくなり，Index Jump の効果が薄れる．この場合には，偏りのあるフィールド条件を 2 つのサブフィールド条件に分割する．つまり，次元を $(N+1)$ に拡張すれば，Index Jump テーブルによる分散効果を大きくできる．

4. 実 現 法

前章で，パケット分類アルゴリズムをメモリ参照を考慮してパイプライン並列に展開した．本アルゴリズムを DDNP 上に実現すれば，2 章で述べたデータ駆動型アーキテクチャの特徴より，自然なパイプライン処理を構成できる．本章では，分類アルゴリズムをより効果的に実現するためのデータパケット構成，メモリ参照を中心とした演算命令の追加，プロセッサ構成およびメモリ I/F 構成，ソフトウェアについて述べる．

4.1 データパケット構成

DDNP は，データパケットに 32bit×2 ワードのデータを保持でき，SIMD 命令により複数のデータを同時に演算可能である．しかし，SIMD 命令は同一の演算を一括処理するときには有効だが，最長一致検索ではデータごとに検索木の探索回数が異なるため効果を得にくい．

そこで，パケット分類処理がフィールドおよび中間データを用いて行われ，この対が分類処理が終了するまで保たれることに着目し，フィールドと中間データを単一データパケットで表現する．DDNP の各プロセッサは図 5 のとおり，待ち合わせ (MM)，演算 (FP)，命令フェッチ (PS) を行うが，単一データパケットで実現することで，MM での待ち合わせ回数を削減できる．さらに，分類処理を単一データパケットのデータフローで表現でき，後述する分類処理のソフトウェア記述を容易にできる．

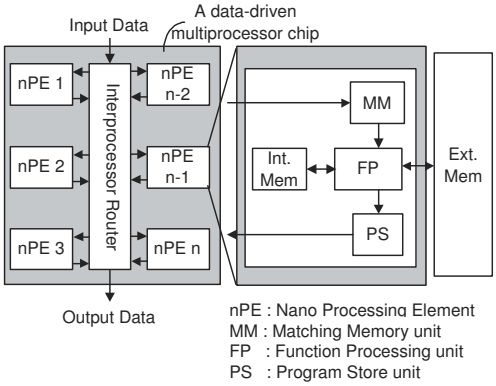

図 5 DDNP プロセッサ構成.
Fig. 5 DDNP Processor Structure.

4.2 FieldMatching 命令セット

FieldMatching(FM) 命令セットは，FieldMatching 部

において前節のデータパケットを解釈し，処理するための複合命令群である．FieldMatching 部における LC-Trie 木検索およびプレフィックス検査は，ノード情報取得用アドレス計算処理，テーブル参照，検索終了判定処理から成るが，これらを個別の命令として定義すると，命令実行毎にデータパケットがプロセッサ内のパイプラインを周回するため遅延が大きくなる．そこで，各処理を複合し 1 命令で演算を行う．

4.2.1 LC-Trie 木検索命令

本命令は，LC-Trie 木検索の 1 ノード分の処理を行う (図 6)．まず，入力された検索用情報 (Intermidiate Search Result) とフィールドデータの部分ビット列 (Prefix of IP Address) からノード情報参照用のアドレスを計算する．次に，計算されたアドレスを基に LC-Trie Table を参照する．テーブルから得られるデータには，2 つのメモリアドレス計算用情報 (Skip, Branch) とポインタが含まれており，計算用情報中の Branch が 0 であればリーフと判定し，検索を終了する．1 であれば子ノードが存在するため，次の検索用情報を生成し出力する．

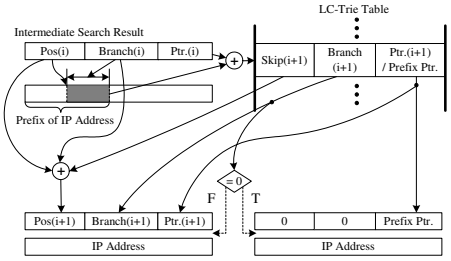

図 6 LC-Trie 木検索命令.
Fig. 6 Search Trie instruction.

4.2.2 プレフィックス検査命令

本命令は，まず LC-Trie 検索で得られたポインタを基に，該当するプレフィックスおよびプレフィックス情報を Check Prefix Table より参照する．そして，検索対象データとプレフィックスが一致するかを判定する．一致すれば検索結果を出力して終了する．一致しなければ，次のプレフィックス検査を行うための情報取得用ポインタを出力する．このとき，図 7 に示すように，Check Prefix Table から参照するデータは 2 ワードあり，メモリ参照を 2 回行う必要がある．そこで，メモリ参照を同時に行いメモリ参照遅延を抑制するため，メモリ参照ユニット (MAU) を 2 つ並列に接続し，独立したメモリにアクセスできるようにプロセッサ構成を拡張する (図 8).

4.2.3 マルチプロセッサ構成とメモリ I/F 構成

DDNP は，整数演算用プロセッサ (INT)，およびテーブル参照用プロセッサ (TBL/ExtTBL) のマルチプロセッサで構成される．

INT は整数演算，およびビット演算を行うプロセッサである．また，小容量の内蔵オンチップメモリ (内部テーブル) を保持している．TBL はメモリ参照を主体とし，内部テーブルと外付けオフチップメモリ (外部テーブル) へのアクセスが可能であり，TBL に LC-Trie 木検索命令を，ExtTBL にプレフィックス検査命令をそれぞれ追加する．

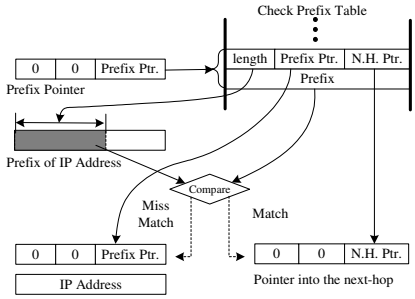

図 7 プレフィックス検査命令.
Fig. 7 Check Prefix instruction.

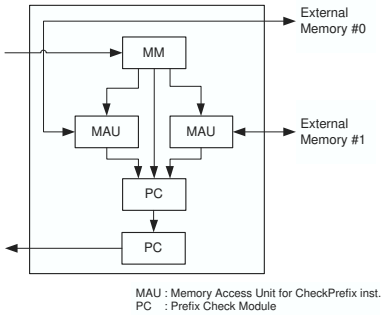

図 8 メモリ I/F 拡張 TBL ナノプロセッサ.
Fig. 8 A TBL nano processor that extended memory I/F.

分類処理データは，LC-Trie 木検索の多くが 2 レベル未満で終了することから，もっとも頻繁に参照される第 1 レベルのノード情報を TBL の内部テーブルに，第 2 レベル以降のノード情報およびプレフィックス検査情報は外部テーブルにそれぞれ分散配置する．図 9 に示す構成では，2 つの TBL を用いて LC-Trie 木検索処の負荷分散を行い，分類処理データを 2 つの外部メモリに分散配置している．また，TBL と ExtTBL が同じメモリを共用するため，プロセッサとメモリの間に調停機構を用意する．これらのマルチプロセッサ構成は，処理対象のテーブルサイズやターゲットクラスのトラヒックの負荷によって選択的に構成できる．

4.3 パケット分類，最長一致検索ソフトウェア

n 次元のパケット分類を行うためのソフトウェアを図 10 に示す．n 個の入力データは，まず Index Jump テーブルの参照ポインタを得るための Index 計算を IndexJump モジュールにて行う．その後，フィールド数 n に応じて，連結された FieldMatching モジュールで最長一致検索を行う．

FieldMatiching モジュール内は図 11-(A) に示す構成をとる．はじめに，入力データは LC-Trie 検索用のデータを含むパケットに結合される．その後，LC-Trie 木検索命令 (SearchTrie) およびプレフィックス検査命令 (CheckPrefix) を用いた最長一致検索処理を n 回行う．

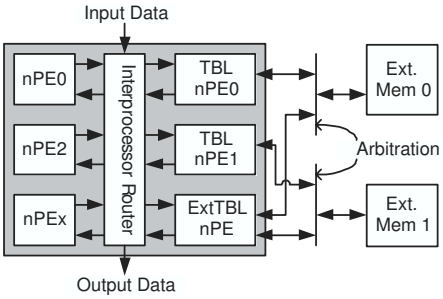

図 9 マルチプロセッサ構成とメモリ I/F.
Fig. 9 Multiprocessor and memory I/F.

また，IPv6 の送り元／宛先アドレスのような長いデータに対する最長一致検索は図 11-(B) に示すプログラムで実現される．IPv6 のアドレス長は 128 ビットあるが，一方で，データ駆動パケットのデータフィールド長は 32 ビットである．そのため，宛先アドレスを 4 分割して，先頭の 2 データパケット (64 ビット) でまず最長一致検索を行い，さらにプレフィックスが長い場合には後半 2 データパケットを適宜呼び出し，継続して検索する．

図 11 中で，SearchTrie はループで行っているが，ループ回数が静的で，かつある部分の検索データが十分少ない場合は，木構造を用いず単純なメモリ参照命令に置き換えて，検索レートの向上を図ることができる．

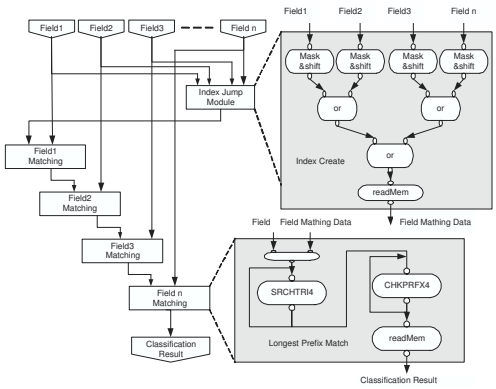

図 10 パケット分類処理のデータフローグラフ.
Fig. 10 A data-driven program for the superpipelined packet classification.

5. 評　　価

提案した FM 命令セットを含むプロセッサを追加した場合の回路規模，および性能の評価のため，評価用 DDNP ボード (図 12) を試作した．本ボードは，シャープ (株) が試作した信号処理用 32bit データ駆動プロセッサ，および FPGA チップ (APEX DSP DB EP20K 1500E) から構成され，FM 命令セットを含むナノプロセッサ (Ext nPE#1/#2) を FPGA チップ上に実装している．各チップの接続関係を図 13 に示す．FPGA に実装したナノプロセッサは，動作性能を 32bit データ駆動プロセッサの動作性能に対して

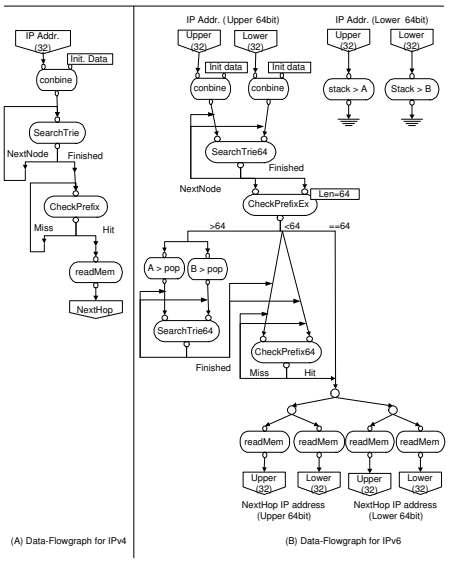

図 11 最長一致検索のデータフローグラフ
Fig. 11 A data-driven program for the superpipelined LPM.

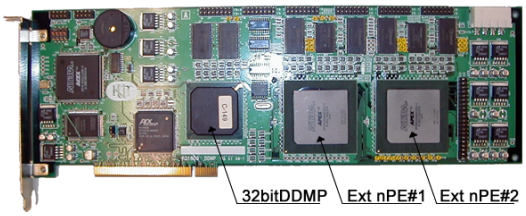

図 12 評価用 DDNP ボード.
Fig. 12 The evaluation board of DDNP.

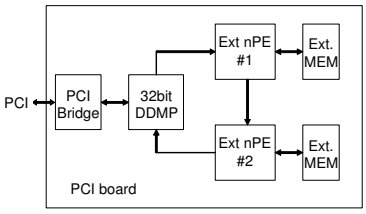

図 13 評価用 DDNP ボード構成.
Fig. 13 Configuration of the DDNP evaluation board.

等価的にスケーリングして回路実装しているため，測定結果がそのまま比例換算可能である．これを用いて，必要最低限のハードウェア追加で提案方式を実現できることを示す．そして，パケット分類を実現したとき，十分な性能向上が得られるかを評価する．

5.1 回 路 規 模

論理合成に Menter Graphics 社の Leonard Spectrum を用い，FPGA をターゲットとしてゲートレベルの回路規模を見積もった．なお，論理合成ツールが出力するデータは FPGA の論理セル数となるため，これを論理セルあたりの使用ゲート数を基に換算した（表 1）．これより，約 6%程度のゲート数増加で IPv4 用のプロセッサ構成可能であることが判った．さらに，IPv6 用回路は IPv4 用回路に高々1.4%のゲート数増加で拡張可能である．

表 1 機能拡張に伴うゲート数、および増加数.
Table 1 The number of logic gates and increases.

	Total gates	Increases
Existing DDP	481K	—
Add IPv4 Inst.	536K	27.5K
Add IPv4 & Pv6 Inst.	549K	34.0K

表 2 最長一致検索性能.
Table 2 Lookup performance
(measurements on a Mae East routing table) .

	IPv4			IPv6		
	55K	110K	550K	55K	110K	550K
Existing DDP	2.5	2.1	1.8	1.3	1.2	1.0
Propose scheme	51.0	43.3	36.7	27.0	23.0	19.5

Lookup Rate : M lookups/sec

5.2 パケット分類処理性能

まず，最長一致検索処理において，FM 命令セットの追加による性能を評価するため，既存の DDNP と性能比較を行った.

評価対象として，現在最も大規模なルーティングテーブルを保持するノードのひとつである Mae West の IPv4 および IPv6 のルーティングテーブル 50K を用いた．さらに，ルーティングテーブルの増加に伴う検索レートの増減を確認するために，110K エントリおよび 550K エントリのルーティングテーブルを合成して評価した．各エントリを写像した LC-Trie 検索木のうち，もっとも頻繁に参照される第 1 レベルを内部テーブルに，第 2 レベル以降を外部テーブルに分散配置した．また，プレフィックス検査用データはいずれも 1 回で完了するものとし，外部テーブルに配置した．評価の結果，表 2 に示すとおり，FM 命令セットの追加によって劇的な性能向上を達成できることが判った.

次に，パケット分類の性能を，最長一致検索処理性能を基に見積もった．また，Index Jump を用いた検索木のバランス改善を検証するために，Index Jump テーブルのサイズが 2，256，4096 の場合について試行した．パケット分類処理では，1 入力パケットにつきメモリ参照回数は，Index Jump 部で 1 回，ツリー検索部で L 回必要である．ただし，LC-Trie 木の深さは $O(\log\log n)$ で増加するため，$L = \log\log n$ である．フィールドの数を M，フィルタの各条件を木構造に写像するために必要なノード数を $n_i (1 < i < L)$ とすると，パケット分類処理に必要なメモリ参照回数 N は最悪の場合，$\sum_{i=1}^{M} \log\log n_i$ であり，検索時間は $O(M \log\log n)$ となる.

表 3 より，Index Jump を用いれば，分割後のルールセットのエントリ数を大幅に削減できることが判った．また，分類レートは約 12MLookup/sec であり，平均的には約 24Gbps 程度の IPv4 パケットストリームの分類が可能なことを確認した.

表 3 提案パケット分類手法の性能見積もり.
Table 3 Peformance of the proposed packet classification scheme.

Index Jump Table Size	2	256	4096
Divided Rule Set	78846	14861	6882
Classicifation Rate	11M	12M	12M

Divided Filter Set：FilterSet/IndexJumpEntrie
Classicifation Rate: Packets/sec

6. むすび

本稿では，高速パケット分類処理を行うアルゴリズムをパイプライン並列に実現する方法を述べ，DDNP による実現法を提案した.

パケット分類の核となる最長一致検索手法には，メモリアクセス数の最少化，および検索データサイズの最小化の観点から，LC-Trie 検索法を用いた．そして，木検索における複数メモリ参照をパイプライン並列に処理することで，検索レートを向上させた．さらに，ルールセットの大規模化に伴う検索木の偏りを平滑化するために，ルールセットをヒューリスティックに分割し，1 検索木あたりのデータ量を極小化し，少ない記憶量で高速なパケット分類処理を実現した.

そして，これらのアルゴリズムを少ない命令実行数で効率的にパイプライン実現できるパケットデータ構成および FM 命令セットを提案し，これらを実装したプロセッサ構成，およびメモリ I/F 構成を示した．FM 命令セットの追加によるプロセッサの回路規模の増加は，IPv4 用で高々6%であった．さらに，IPv6 用回路は IPv4 用回路に約 1.4%程度のゲート数増加で拡張できた.

参考文献

1) L. Gwannap and B. Wheeler, “A Guide to Network Processors Third Edition,” *MicroDesign Resources,* July,2002.
2) J. Bolaria and B. Wheeler, “A Guide to Classification and Traffic Management Coprocessors Second Edition,” *MicroDesign Resources,* May,2002.
3) H. Terada, S. Miyata and M. Iwata, “DDMP's: self-timed super-pipelined data-driven multimedia processors,” *Proc. of the IEEE,* 87(2), pp.282-296, 1999.
4) P. Gupta and N. McKeown,“Algorithms for Packet Classification,” IEEE Network,15(2),pp.24-32,2001. Proc. of IEEE Infocom, vol.3, pp.1193-202, 2000.
5) S. Nilsson and G. Larlsson,“IP-address lookup using LC-Tries,” IEEE J. on Sel. AIC,17(6),pp.1083-1092,1999.
6) T.Y.C. Woo,“A modular approach to packet classification: algorithms and results,” *Proc. of IEEE INFOCOM,* vol.3, pp.1203-22, March,2000.
7) IPMA Project - http://www.merit.edu/ipma/