平成 14 年度

修士学位論文

データ駆動型ネットワークプロセッサ DDNP における高速パケット分類方式

Fast Packet Classification on a Data-Driven Network Processor: DDNP

1055118　　森川 大智

指導教員　　岩田 誠

2003 年 2 月 5 日

高知工科大学大学院 工学研究科 基盤工学専攻

情報システム工学コース

要 旨

データ駆動型ネットワークプロセッサ DDNP における高速パケット分類方式

森川 大智

将来の基幹ネットワークの全光化に伴い，光リンク速度でディジタル信号を授受し，これらに対して多様な通信サービスを柔軟に提供できるネットワークプロセッサ (NPU) の実現が望まれる．本研究の最終目標は，高速かつしなやかな NPU 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) の構成法を確立することにある．

NPU が提供する機能の中でも，パケット分類や各種テーブル検索処理は非常に処理負荷が高く，高速化の鍵とされている．本論文では，各種サービスに応じたパケット分類をパイプライン並列化に処理可能な方式を提案する．各ルールの照合に LC-Trie による最長一致検索を用いて，検索処理のボトルネックとなるメモリ参照を極小化する．さらに，パイプライン並列化により，メモリ参照遅延を隠蔽する．また，ルールセットの大規模化による，検索木の偏りを緩和するため，Index Jump テーブルを導入した．そして，本提案方式を含む，DDNP アーキテクチャのハードウェアレベルのエミュレーションを可能とし，性能評価，回路規模の見積もりを行うために評価ボードを試作した．評価ボードを用いた評価の結果，本方式は 6 %程度の回路拡張で約 12MLookup/sec(IPv4) の性能を実現できることを示す．

キーワード　ネットワークプロセッサ，パケット分類，最長一致検索，データ駆動，自己同期パイプライン

Abstract

Fast Packet Classification on a Data-Driven Network Processor: DDNP

Daichi MORIKAWA

With developing all-optical communication networks, highly-functional and high-speed boundary routers and home gateways are becoming to be required. To satisfy these requirements, novel NPU with programmability and high speed performance close to performance by hardware implementations is expected to be developed. The final objective of this research is to establish the construction of a network processor (DDNP) based on the data-driven scheme having highly pipelined processing capability.

In this paper, a high-speed pipelined algorithm for packet classification which is one of heavy load functions within boundary routers is proposed. The paper then presents its software implementation scheme on a data-driven processor that having flexible capability of pipelined parallel processing. Then, an evaluation board for DDNP which evaluates performance and measures logic scale of hardware including the proposed scheme is described in this paper. 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.

key words Network processor, Packet classification, Longest prefix matching, Data-driven, Self-timed super-pipeline

目次

図目次

図目次

図目次

表目次

第 1 章

序論

インターネットの利用数の急速な増加は，通信ネットワークに革命をもたらしている．背景には，通信ネットワークのインフラストラクチャが既存電話網からデータ主体のパケット通信ネットワークへの推移がある．IP データトラヒックは急激に増加している．実際に，大勢のユーザがエンターテイメント，リアルタイム系の音声や映像ストリーミング配信，魅力的なコンテンツ，ニュース，安全な商取引，データストレージ，共有ファイルアクセス，情報相互交換，巨大なファイル，データの伝送などさまざまなインターネットサービスへアクセスしている．さらに，IP ネットワーク上で用いられる様々なアプリケーションの普及は，新たな特性を持つ IP トラヒックの増加を促進している．例えば，従来のベストエフォート型のトラヒックに加えて，ミッションクリティカルなトラヒックを優先的に扱うなど，異なったサービス品質を提供するためにトラヒックの要求に応じた扱いを行うことが必要となっている．加えて，エンタープライズコンピューティング/ネットワーキングリソースは，より広範囲に多様なサービスを提供するため，悪質な攻撃から保護されなければならない．そのため，ルーティング，ファイアウォール，NAT(Network Address Translation)，QoS(Quality of Service)，帯域監視，帯域制御といったネットワーク機構が，今日のネットワーク装置にとって不可欠なものとなりつつある．さらに，より複雑な転送ポリシーや転送方式は，情報を効率的に分散するために重要となる．

また，異なるネットワークシステムを要求するトラヒックの増加によって，高速なネットワークインフラストラクチャの必要性が激増している．DWDM(Dense Wavelength Division Multiplexing) や光 AMP のような光伝送技術の発展によって，広帯域幅の伝送路が実現されており，2.5Gbps から 10Gbps の伝送レートを達成している．さらに，40Gbps

の実現も視野に入っている．近い将来に光ファイバは一般加入者のテラビット級のデータ伝送を可能とするだろう．

しかし，いくら伝送路が高速・大容量になっても，高速・大容量化された伝送路を流れる大量の IP パケットを，その速度で処理できなければ意味が無くなる．つまり，ネットワーク装置のパケット交換処理が高速にならない限り，インターネット全体のトラヒック増加に対処するバックボーンが構築できない．現状のネットワーク装置は，電気的に IP パケットを処理し転送するため，光通信によって回線が速くなれば，処理が追いつかなくなる．そこで，IP バックボーンにおいては，波長単位の経路制御や IP パケットをも光スイッチによりスイッチングすることによる，ネットワークの全光化の研究が盛んに行われている．

一方で，単純に IP パケットのフォワーディングを行うだけでなく，より複雑なサービスを提供できる処理が要求されている．今日では，レイヤ 3 およびレイヤ 4 フィールドを参照するマルチルックアップ処理がネットワーク装置の一般的な特徴となっている．次世代のネットワーク装置は，レイヤ 5 からレイヤ 7 のフィールド，あるいはペイロードによって，パケットの解析や分類を行うようになる．ゆえに，ヘッダ情報を通してパケットを識別するだけでは不十分であり，負荷調整，ウェブスイッチング，不正進入検知とセキュリティ等のネットワークアプリケーションでは，より深いパケットの解析を要求する．このような状況下では IP パケットを光信号のまま処理するのは困難であり，電気信号に変換した上で，各種処理を行う必要がある．

これまでのネットワーク装置は，専用ハードウェアで構成された ASIC(Application Specific Integrated Circuit) か，ソフトウェアを汎用 CPU 上で動作させて実現してきた．ASIC はこれまでの速度重視の性能向上においては劇的な能力を発揮してきた．しかし，今日の複雑多様化するサービスに対応するためには，その開発期間はあまりにもかかりすぎる．ソフトウェアによる実現法は，柔軟性を持ち合わせており新しいプロトコルにも即対応可能である．しかし，ネットワークの光伝送技術は Guilder の法則によると 6ヵ月ごとに 2 倍に増加するとされてるが，一方で半導体技術は Moore の法則に従えば 18ヶ月に 2 倍しかならず，汎用 CPU 上で動作するソフトウェア実現法では満足な速度が得られない．こうし

た問題点において，従来の速度性能と柔軟性のトレードオフの考え方を抜本的に変える新しいタイプの半導体デバイスとしてネットワークプロセッサ (NPU) が注目されている [1].

NPU は，パケット処理向けに最適化されたプログラミング可能な半導体デバイスである．これにより，ハードウェアの速度性能を維持しつつ，ソフトウェアによりプログラム可能であることであることから，新しいプロトコルにも柔軟に対応可能となる．よって，今後の製品ライフサイクルの短縮と，大幅な開発コストの削減をもたらす．一般にレイヤ 2 からレイヤ 7 ルーティングにおけるタスク (ヘッダ解析，パターンマッチ，ビットフィールド操作，テーブルルックアップ，パケット変形，データ移動) を処理するように設計されている．

現在，各 NPU ベンダーはこれらのタスクをパイプライン構成することにより，リンク速度への追従を図っている．例えば，Intel の NPU である IXP2800 は図 1.1 に示すように，パケット処理用にパケットエンジン (PE) を 15 個パイプライン接続し，マルチスレッドにより動作する．各 PE は RISC ライクの命令セットと大量のレジスタを保持するカスタムプロセッサコアである．IXP2800 は，1.4GHz という動作周波数を必要とし，消費電力は 14W に上る．

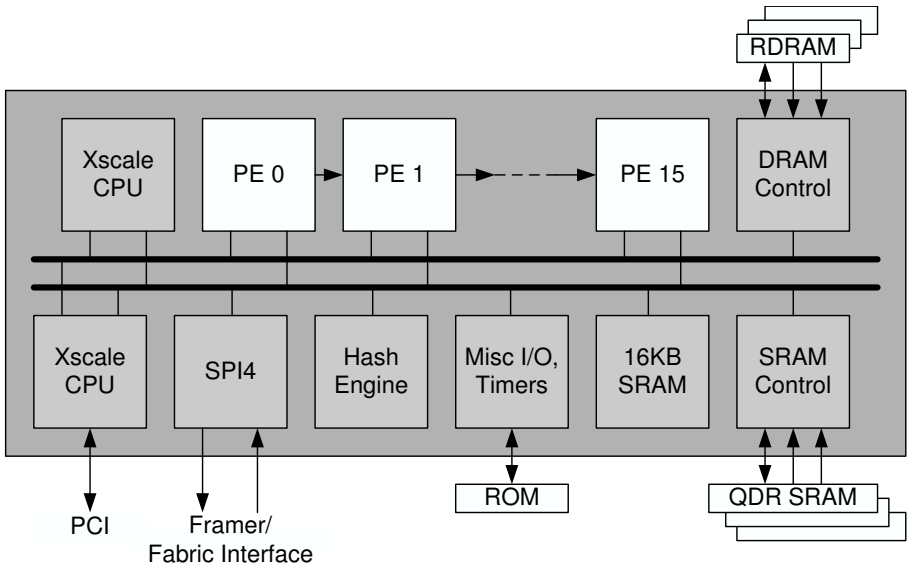

図 1.1　Intel IXP2800 ブロックダイアグラム

Motorola の C-5 も同様に，Motorola が開発したカスタム 32-bit RISC コアである

Channel Processor(CP) と呼ばれる複数のプロセッサコアを並列に配置することによりトータルで 8Gbps の性能を得ている (図 1.2).

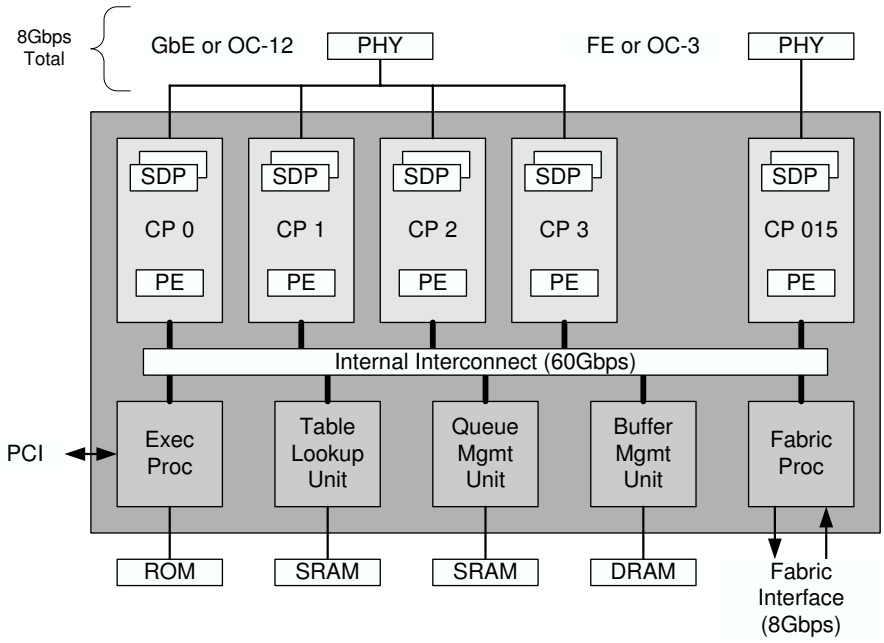

図 1.2　Motorola C-5 ブロックダイアグラム

上記の例のように，既存の NPU の多くは，バス接続あるいは相互接続ルータによりパケットエンジンや，各種専用プロセッサを接続している．今後さらに高速化，多様化するであろう，ネットワークサービスに柔軟に対応するには，よりパイプライン並列度を高める必要がある．しかし，いずれもクロックベースで動いているため，すべてのプロセッサ間で何らかの協調制御が必要になり，パイプライン並列化の阻害要因となる．さらに，集積度が上がると消費電力は格段に上がってしまう．

本研究の最終目的である，データ駆動型ネットワークプロセッサ DDNP は，データ駆動原理に基づいて構成されているため，プロセッサ間の協調制御の必要がない．さらに，自己同期型パイプラインにより自然なパイプライン並列化が可能となるうえ，中央支配的なクロック制御の必要が無く，パイプライン段間の局所的な同期のみ行えばよいので，極少電力化も図れる．よって，今後さらに肥大するトラヒックに対して，高速パケット処理性能を柔軟に提供する NPU アーキテクチャとして期待できる．

NPU は雑多なネットワークを潤滑に処理するよう設計されているが，昨今のリンク速度の向上において，NPU 単独で多様なサービスに対応することを困難にしている．特に，処理負荷の高いパケット分類や各種テーブル検索処理が高速化の鍵となっている．そのため，現在はパケット分類や各種テーブル検索処理を専用化した高速 LSI を補助的に用いた実現方法を取っているしかし，この方法では，NPU の最大の特徴であるプログラマビリティが失われてしまう．

本論文では，各種サービスに応じたパケット分類をパイプライン並列に処理可能な高速アルゴリズムを提案し，このアルゴリズムをデータ駆動型プロセッサの特徴であるパイプライン並列処理の徹底的な活用によって高速にソフトウェア実行する方式を提案する．

パケット分類は，アプリケーションの要求によって，パケットのヘッダあるいはデータペイロードのさまざまなフィールドとセグメントを調べ，静的な，場合によっては動的なルックアップデータベースと比較することにより識別を行う．ネットワーク装置の基本的機能は，あるポートから受信した入力パケットをパケットの示す宛先まで，あるいは最も近いノードまでフォワードする機能であり，フォワーディングはパケット分類の基本的なものである．

- MAC アドレスフォワーディング

  パケットは，レイヤ 2 の宛先 MAC アドレスと，場合によっては仮想 LAN タグによって分類される．MAC アドレスは，妥当な MAC アドレステーブルに対してルックアップ処理をすることによって，LAN セグメントにおいて，パケットの最終的な，あるいは中間の宛先を識別する．データベースのアップデートのために送り元 MAC アドレスの学習処理も分類タスクに含まれる．

- IP ルーティング

  IP ルーティングパケット分類タスクは，レイヤ 3 にあたる IP ヘッダの宛先 IP アドレスに対し，ルックアップ処理をする．今日，一般的に使われている IP ルーティングの基本フォームの多くは CIDR(Classless Inter-Domain Routing) である．CIDR は最長一致フォワーディングアルゴリズムを使用している．CIDR において，経路はプレフィッ

クスとプレフィックス長で表現される．プレフィックス長は意味のあるプレフィックスのビット数で決定される．CIDR はより長いプレフィックスであることで，経路がより特定できると考える．最長一致フォワーディングアルゴリズムはもっとも長いプレフィックスを持つ一致するエントリを見つけることで，より特別な経路を選択する．

表 1.1 に，10 進表現のアドレスと長さを使って，重複する経路の例を示す．この例では，192.168.201.112 という宛先アドレスを持つパケットがすべてのエントリと一致する．よって最長プレフィックスを持つエントリはエントリ 2 番である．192.168.201.104 の宛先アドレスはエントリ 1，3，4 と一致するので，エントリ 3 が最長プレフィックス一致となる．

表 1.1　CIDR ルーティングテーブルの例.

Entry	Prefix	Length	Mask
1	192.168.192.0	18	255.255.192.0
2	192.168.201.112	32	255.255.255.255
3	192.168.201.0	24	255.255.192.0
4	192.168.200.0	21	255.255.248.0

- アクセス制御リスト (ACL:Access Control List) フォワーディング

  高次ネットワークは，コンピュータ，ワークステーション，ネットワーク装置が相互接続するメッシュ構成になる．基本的な場合，あるネットワークノードから他のネットワークノードへのアクセスは抑制されない．しかし，ある程度の制御を取り入れることにより，総てのネットワークは，帯域幅，アクセス，その他ネットワークのアクセス制御ポリシーに基づいたサービスにより提供される，システムに流れるパケットを分割する．アクセス制御ポリシーはネットワーク内の規則によってパケットをフィルタリングするために実装される．IP ネットワークにおいて，ACL ベースのフィルタリングは，パケットの 5 つの情報を元に分類する．5 つの情報は 5 タプルと呼び，図 1.3 に示す，レイヤ 3 の送り元/宛先 IP アドレス (32bit)，レイヤ 4 の送り元/宛先ポート (16bit)，

レイヤ 4 プロトコル (8bit) を指す．抽出されたフィールドに対応するルールは一般的に範囲一致を使う．このマルチフィールドパケット分類は，仮想カプセル化されたファイアウォール機能の要素であり，ネットワークの入出力側のトラヒックをフィルタリングする．マルチフィールド分類の形態をとるほかのアプリケーションには，QoS 機能がある．QoS は帯域幅，遅延，ジッタなどの情報に基づいてアクセスサービスのレベルを決定する．ネットワークサービスはアプリケーション，一般ユーザによって分散されることが可能となる．

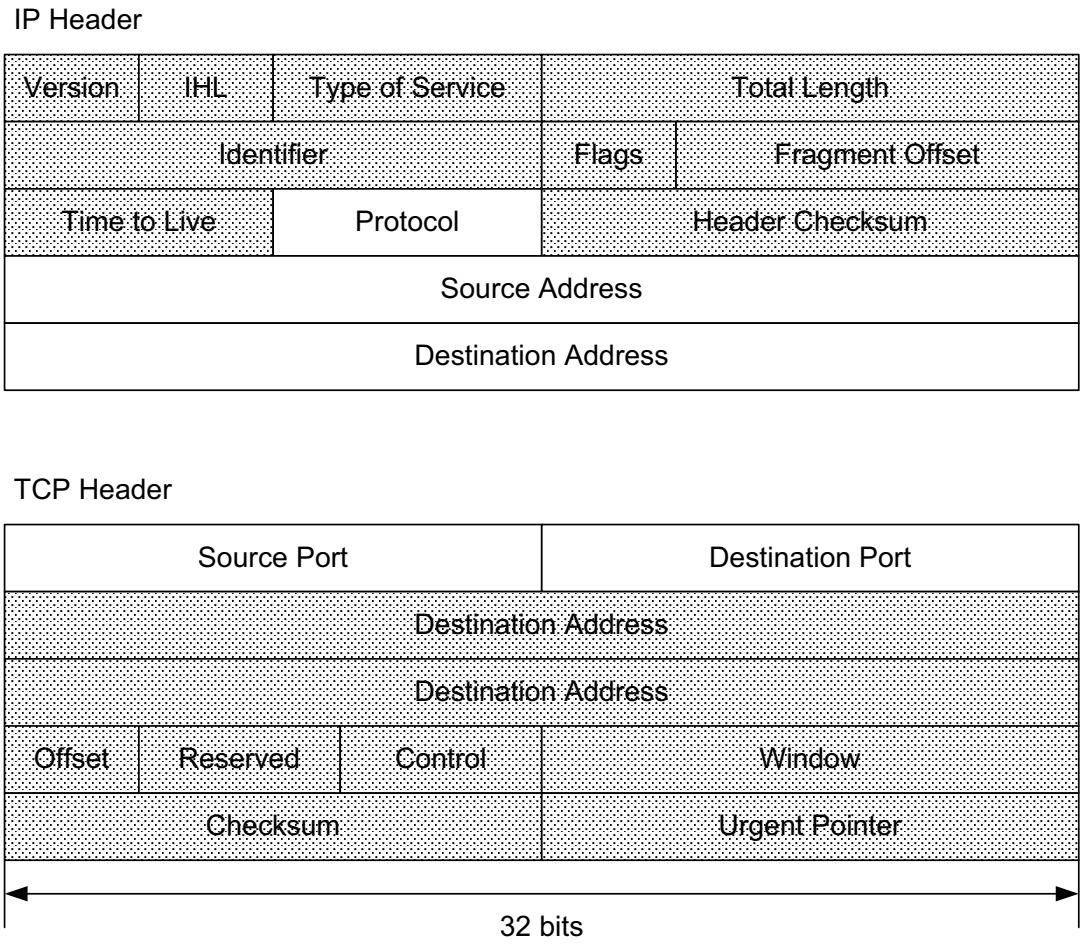

図 1.3　レイヤ 4 で用いるヘッダフィールド.

- コンテンツ処理

  インターネットは数え切れないほどのウェブベースサービスによって進展している．ウェブベースアプリケーションは HTTP を主に用いる．ホスト名とウェブページの

URL は HTTP パケットのデータペイロード内に記述されている．HTTP パケットは通常ヘッダに格納されている宛先ポート番頭によって識別されるが，アクセスパスを識別するための，ウェブホストの仕様や URI においては，パケットデータペイロードから解析する必要がある．これらのフィールドはパケット内の特定の位置には存在せず，それらの位置を見つけ分類する処理するためには複雑な検索処理が必要となる．これらの処理は，ウェブキャッシング，ウェブスイッチング，その他のウェブベースのアプリケーションによって要求される．より進んだコンテンツ処理は解析のためと，位置確認のために，利用者の識別にクッキーフィールドを用いる．これらのテキスト文字列はワイルドカードや正規表現により検索する．

昨今，多様なサービスに対応した処理が NPU 単独では困難になる程，リンク速度が向上してきている．そのため，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は，パイプライン並列に実現するしかない．これまでに専用化した高速 LSI を補助的に用いた実装方法が採られている [2]．しかし，この方法では，NPU 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．

本論文では，各種サービスに応じたパケット分類をパイプライン並列に処理可能な高速アルゴリズムを提案し，このアルゴリズムをデータ駆動型プロセッサの特徴であるパイプライン並列処理の徹底的な活用によって高速にソフトウェア実行する方式を提案した．本論文の，第 2 章において，パケット分類の概要と，既存のパケット分類方式および実現法の問題点について述べ，DDNP における高速ソフトウェア実現の要件を明確にする．第 3 章では，パケット分類の高速ソフトウェア実現をするために，分類規則の照合にパイプライン並列化した最長一致アルゴリズムを用いることで高速化を図るとともに，大規模なルールセットに対しては，ヒューリスティックに分割し，検索空間を限定することで，記憶空間の局所化を行う手法について述べる．第 4 章では，提案方式を DDNP に実現するにあたり，更なる高速化を行うため，データ駆動型ソフト (複合命令) の提案を行い，それらを実装したメモリ参照に特化したナノプロセッサの構成を提案する．第 5 章では，提案方式を含む，DDNP アー

キテクチャの回路規模および性能評価をハードウェアレベルで行うため試作した，DDNP用評価ボードについて述べる．第 6 章では，評価ボードを用いた提案方式の性能評価，および提案方式の実装における回路規模の評価を行う．そして，第 7 章では本研究の成果，今後の課題を考察する．

第 2 章

パケット分類と既存方式，実現法の問題点

2.1 緒言

パケット分類は，多種多様なサービスを適正に使用するために重要な役割を担う．

本章では，まずパケット分類の定義を行い，パケット分類を明確にする．そして，既存の方式および実現法における問題点を述べ，パケット分類の高速ソフトウェア実現の用件を明らかにする．

2.2 パケット分類の定義

パケット分類は，ファイアウォールや QoS など，ユーザの要求に応じて異なるサービスを提供するために欠かせない処理である．具体的には，用途に応じた分類規則を示す大規模なルールセットに対して，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) を各規則に応じて検査し，総ての規則に合えば，あらかじめ要求されたサービスを行う．

サービス毎の要求は，ルールセットで示される．ルールセット内のルールは，フィールドに応じた規則を示しており，それら総てを Π で表すと，

$$\Pi = (\pi_1, \pi_2, ..., \pi_n)$$

であり，各ルールは，分類に必要なフィールド群と，ポリシーで表される．フィールド群を

F，ルールを γ とすると，

$$\gamma = (f, \pi) \ \ \{f \subseteq F \ \ \& \ \ \pi \in \Pi\}$$

となる．このとき，ルールセット C は，

$$C = ((f_1, \pi_1), (f_2, \pi_2), ..., (f_n, \pi_n))$$

$$\bigcup_{i \leq n} f_i = F$$
$$f_i \cap f_j = \phi, \ \ j \ with \ i \neq j$$

である．総てのパケット分類方式は，この定義に従い分類処理を行う．

2.3　既存のパケット分類方式

n 個のフィールドを対象とした (n 次元) パケット分類では，ルールセットの大規模化に伴い計算量が激増する．

既存のパケット分類方式は，表 2.1 示すとおり 4 つに大別できる [3]．これらの分類方式を以下の観点から述べ，問題点を明らかにする．

- 分類速度
- 記憶容量
- 扱えるフィールド数
- 実装の容易さ

表 2.1　パケット分類方式の種類.

Category	Algorithm
Basic data structure	Linear search, caching, hierarchical tries, set-prunning tries
Geometry based	Grid-of-tries, AQT, FIS
Heuristic	RFC, hierarchical cuttings, tuple-space search
Hardware only	Ternary CAM, bitmap-intersection

- ソフトウェア方式

  最も単純な分類アルゴリズムは，各ルールを線形に検索する Linear Search である．本アルゴリズムが必要とする記憶空間は，ルールセットをそのままメモリ構造に写像するため非常に効果的に使用できる．さらに，ルールセットをそのまま用いるため，検索に用いるデータ構造は単純そのものであり，容易に変更可能である．しかし，ルールセットが大規模化すると，分類レートも比例して長くなるという問題点がある．

  そこで，分類レートを削減する方式として，一般に Trie 構造が用いられる．図 2.1 は，Hierarchical Tries アルゴリズムにおける検索木の例を示している．本方式は，各ルールを BinaryTree で表現し，各ルールの検索木を逐次検索していく．このとき，分類レートは，ルールセットのサイズに依存せず，分類に用いるフィールド数とフィールドのデータ幅に依存する．また，記憶容量は主にルール数，フィールド数，各フィールドの幅に支配される．

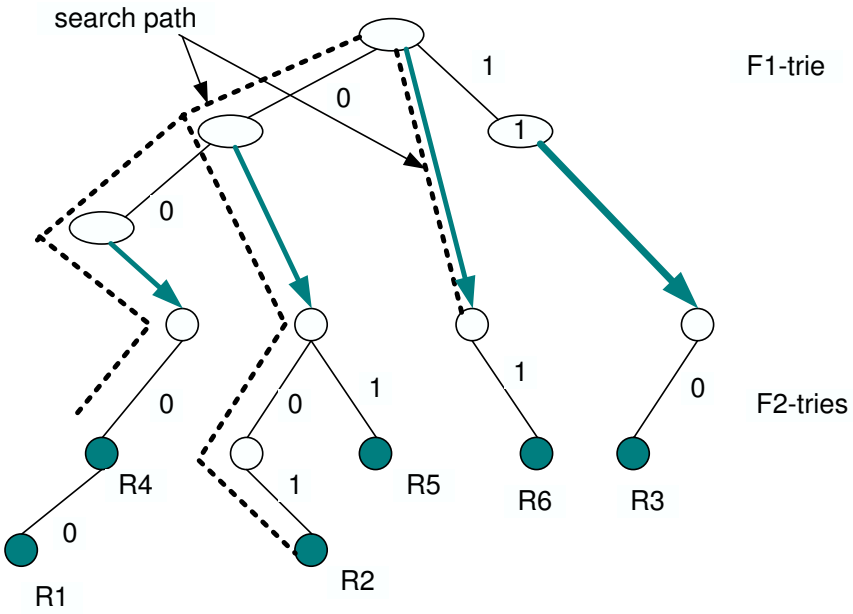

図 2.1　Hierarchical Trie

- ジオメトリベース方式 [4] [5]

  ルールセットを n 次元平面に写像することで，n 次元点位置決定問題として扱う．図 2.2 に示す 2 次元点位置決定問題では，図中の点 P の座標が与えられたとき，垢締め定められた領域のどの領域に P が含まれるかを求める問題である．P の座標をパケットのフィールド値，領域をルールセットの定める値域と対応付けられる．本方式において，分類レートと記憶空間は完全なトレードオフの関係にある．ルール数 n，フィールド数 F のとき，$O(\log n)$ 分類レートを達成する場合，記憶空間は $O(n^F)$ 必要であり，記憶空間を $O(n)$ とした場合は，$O(log^{F-1}\ n)$ の分類レートが必要となる．そのため，ルール数が 1000 でフィールド数が 3 の場合，1GB の記憶容量が必要になる，あいは，100 回のメモリアクセスが必要となり，現実的ではない．

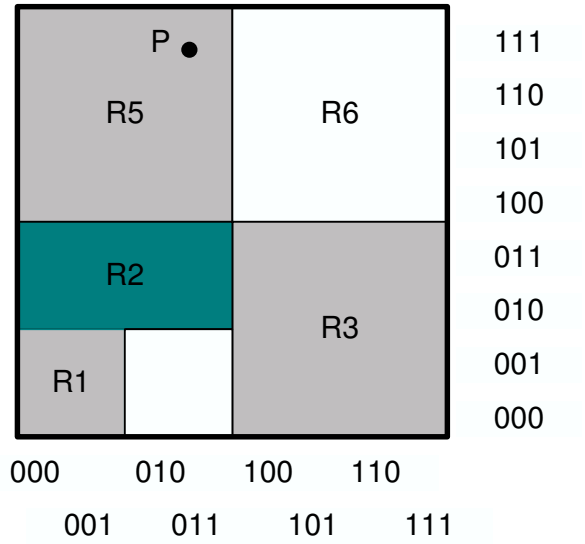

図 2.2　ジオメトリベース

- ヒューリスティック方式 [6] [7]

  図 2.3 に示す，Recursive Flow Classification アルゴリズムは，各フィールドを細かく分割し，分割したデータを基に並列に段階的に検索を進めていく方式である．各フィールドは現実のルールセットを基にヒューリスティックに分割する．分類レートは次元数にのみ依存し，n 次元のルールセットにも実現的な性能を有する．ただし，多大な記憶空間を必要とする．

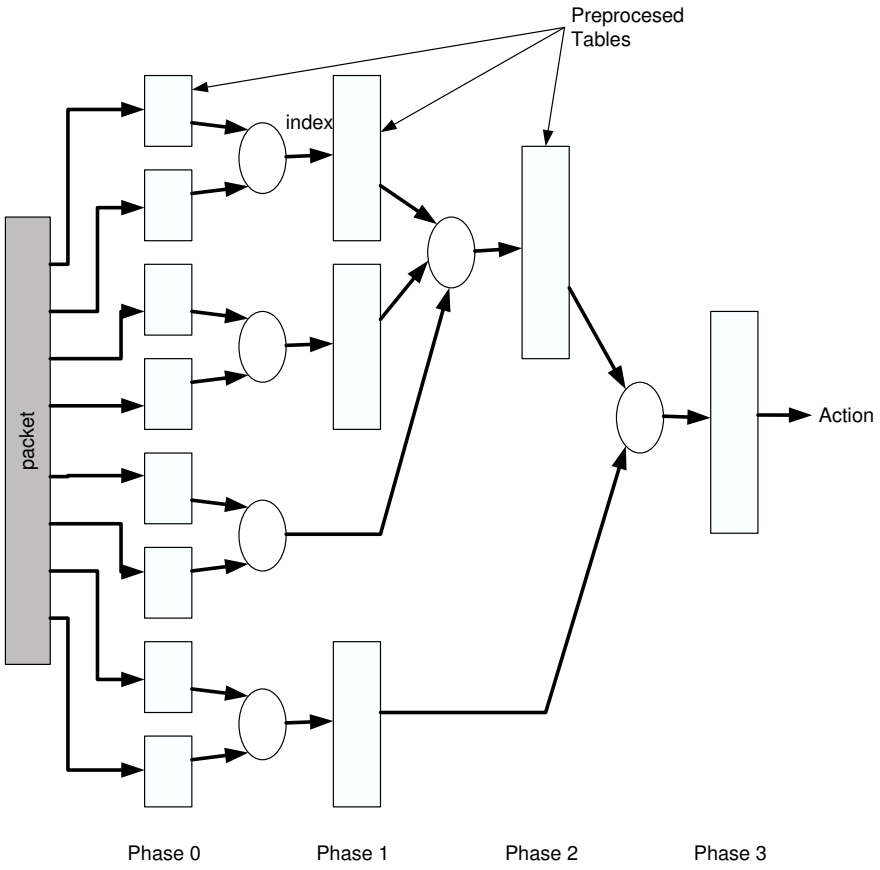

図 2.3 Recursive Flow Classification

- ハードウェア方式 [2] [8] [9]

  Bitmap-intersection は，ハードウェア実装に最適化された方式であり，小規模なルールセットにおいては，高速な分類処理と少ない記憶空間で実現可能な方式である．しかし，ルールセットが大規模化すると，ルールセットのサイズに応じて必要な記憶空間が急増し，メモリバンド幅も線形に増加してしまう．

  もうひとつの代表的な方式として，連想メモリを用いたハードウェアデバイスである TCAM がある．TCAM のセルは 0，1，および*を保持することが可能である．*は don't care を表し，ワイルドカードを含むルールの一致処理を TCAM が実現するためにマスクとして用いられる．TCAM は連想メモリを用いることにより，非常に優れた検索性能を実現しており，パケット分類に理想的であるように見える．しかし，CAM は回路コストが高い上，ルールセットの大規模化に柔軟に対応できない．そのため，今

後半導体技術の向上により，高性能な TCAM が実現するかもしれないが，依然として特異なデータ構造を操作する効率的なアルゴリズムによる解決法を用いる必要性がある．

以上より，いずれの方式も小規模のルールセットにおいては絶大な分類レートを誇るが，ルールセットが増大したときには対応できないという問題点がある．

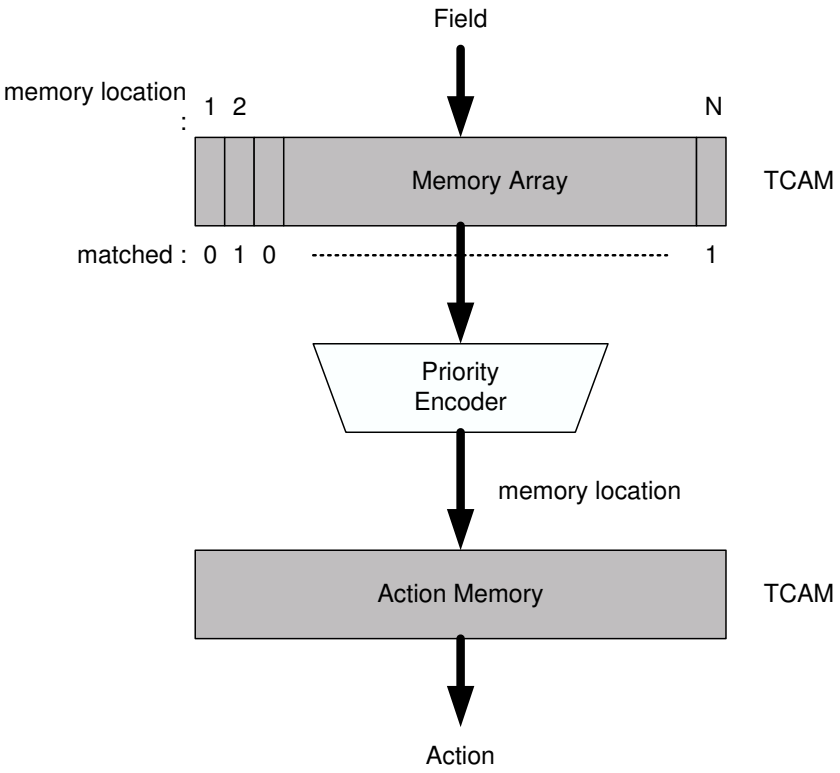

図 2.4　TCAM

表 2.3 より，いずれの方式も，主に分類レートはルール数に依存し，記憶容量はフィールド数に依存する．扱える次元数が少ないものは，静的な低次レイヤのパケット分類なら性能を達成できるが，今後の複雑化する高次レイヤの分類には対応できない．また，次元数の制約のない分類方式においては，各フィールドの部分照合を並列に実行し, 徐々に探索空間を縮小しながら検索する再帰的分類を行う方式が多く見られる．このとき，分類レートを向上させる反面，検索データの記憶量が爆発的に増大する．仮に記憶空間を節約すれば，限定的な次元数の分類しかできなくなる問題点がある．

表 2.2　各方式の分類レートと必要記憶容量

Algorithm	Time	Storage
Linear search	N	N
Hierarchical trie	W^d	NdW
Set-pruning trie	dW	N^1
Grid-of-tries	W	NW
AQT	aW	NW
FIS-tree	$(l+1)W$	$1 \times N^{(l+1/l)}$
RFC	d	N^d
HI-Cuttings	d	N^d
Tuple-space search	M	N
Ternary CAM	1	N
Bitmap-intersection	$dW + N/memwidth$	dN^2

N-Number of Rules W-Width of dimension

d-Number of dimensions a-Tunable integer parameter

2.4　既存のパケット分類実現法

現在の NPU は，レイヤ 2 およびレイヤ 3 レベルのパケット分類を NPU 内に統合された検索処理機構により実現する．しかし，エッジルータ等では，転送速度は低いものの，複雑な分類処理を必要とする．パケット分類は非常に負荷の大きな処理であるため，現在のエッジルータでは一般的に NPU で実行せずコプロセッサの形態をとり，パケット分類プロセッサとして NPU と連動しつつ独立して実現している．

接続形態は大きく分けて，

- Lookaside 方式
- Flow-through 方式

がある．

Lookaside 方式は，図 2.5 に示すように，パケットデータパスには直接接続されず，NPU との通信により動作する．しかし，現在，本方式で NPU とパケット分類コプロセッサを接続する標準のインタフェースが存在しない．そのため，多くのパケット分類コプロセッサは，ほとんどの NPU が実装している ZBT SSRAM インタフェースを用いて情報のやり取りを行う．メモリインタフェースで接続しているため，NPU 側から見ると，パケット分類コプロセッサも一種のメモリとして認識できるため，容易にデータの授受が行えるという利点がある．一方で，メモリインタフェースを用いて接続しているため，データ転送時にオーバヘッドが生じる．レイヤ 4 アプリケーションレベルのパケット分類に用いるデータ量ではオーバヘッドによる影響は少ないものの，レイヤ 7 アプリケーションにおいてパケットデータペイロードすべてを送る場合には深刻な影響を及ぼす．

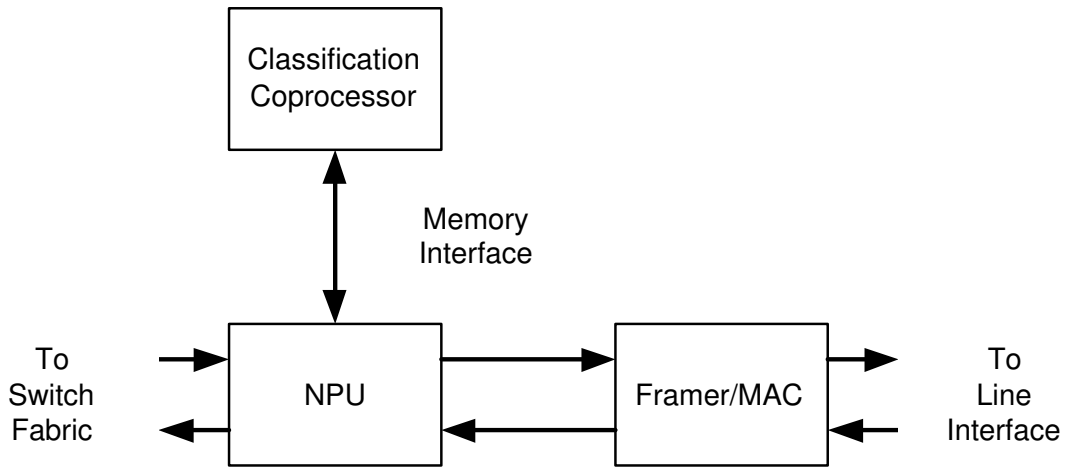

図 2.5　Lookaside 方式

Flow-through 方式は，図 2.6 に示すように，パケットデータパスに直接接続される．そのため，高速分類可能なパケット分類コプロセッサを用いればラインレートを達成できる．ただし，ingress 側に接続する場合は，パケットデータの入出力インタフェースに標準のフレーマインタフェースを用いるが，一方で，egress 側に接続する場合は，スイッチファブリックのインタフェースを用いる必要があり，インタフェースの変更がボトルネックとなる．

また，WAN/MAN コアルータでは，OC-192 に達するポート速度でパケット分類を実行

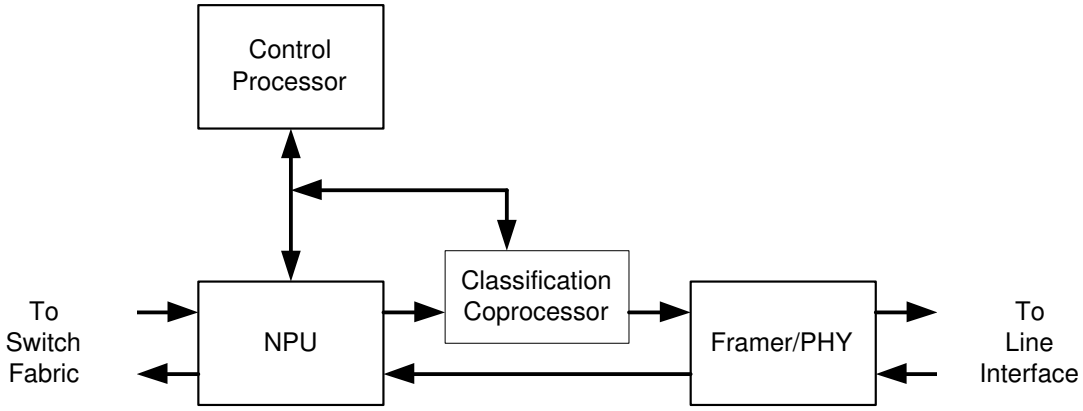

図 2.6　FlowThrough 方式

するために，検索処理のみを行う特殊な検索エンジンと，それらの制御機構を備えた ASIC が用いられる．検索エンジンは，パケット分類コプロセッサからパケット解析機構を除いたものであり，一般に CAM を用いて実現されている．検索エンジンの主要な機能は提供されるキーに基づいた高速分類を実行することである．ただし，前節で述べたように CAM ベースはルールセットが膨大なときに，コスト高となる．表 2.3 に各種パケット分類におけるエントリサイズ，エントリ数と必要な CAM の容量について示す．

表 2.3　MAN コアルータにおける必要エントリサイズ，エントリ数，および CAM 容量

Lookup	Entry size	Entries	CAMs needed
MAC Address	48 bits	64K	4.5Mb
MPLS Label	32 bits	256K	9.0Mb
IPv4 Route	32 bits	512K	2 x 9Mb
IPv6 Route	128 bits	128K	2 x 9Mb
IPv4 Five-tuple	104 bits	256K	4 x 9Mb
IPv6 Five-tuple	296 bits	64K	2 x 9Mb

2.5　結言

本章では，パケット分類の定義を明らかにした．そして，既存の分類方式，実現法の問題点について述べた．その結果，ルールセットの規模が小さい場合には単純でメモリ効率の良い線形検索か，あるいはハードウェアによる実現が有効であることが判った．また，規模の大きいルールセットに対しては，逐次的に検索すると必要な分類レートがルールセットの増加に比例して低下するため，フィールド毎の部分検索をパイプライン並列に実行する方式が必須となる．ただし，既存の方式では，検索に必要な記憶空間が増大する，検索アルゴリズムが複雑になるといった弊害がでる．

以上のことより，パケット分類の高速ソフトウェア実現においては，

- パイプライン並列化可能
- 記憶空間の極小化
- 上記の要求を簡便な処理の複合的な利用により実現可能
- 簡単なハードウェア機構により実現可能

が重要になるといえる．

第 3 章

パイプライン並列パケット分類処理方式

3.1 緒言

前章より，検索空間の増大による分類レートの急激な低下を防ぐためには，パイプライン並列処理による実現法が極めて有効であることが判った．ただし，並列度の向上に伴い，処理に必要な記憶空間が増大することは明らかであり，記憶空間の縮退を考慮する必要がある．本研究では各フィールドの照合処理を流れ作業的にパイプライン並列に順次実行する方式を提案する．このとき，分類レートと記憶空間のトレードオフ，異なるフィールド長や種別に対する柔軟性を得るため，以下の手法を採用する．

- ルールセットの大規模化に伴うルールの偏りに対する検索木のバランスを確保するため，Index Jump テーブル [11] を導入し，検索木を分割する．
- 最長一致検索アルゴリズムとして，メモリ参照数の極小化による検索遅延の削減，かつパイプライン並列化が容易な LC(Level Compress)-Trie 木構造を用いる．

3.2 ルールの照合法

パケット分類において, ユーザが定義可能な照合法は以下の 3 通りである．

- 完全一致
- プレフィックス一致

- 範囲一致

これらの照合法において, 完全一致照合は照合するルールと同一の長さのプレフィックスとの照合とみなせるため, プレフィックス照合の特殊な場合として取り扱える．また, 範囲照合については, たとえば, 8 ≦ x ≦ 11 は [8, 11] = [1000, 1011] = prefix(10*), x ≦ 1023 は [, 1023] = [, 0000001111111111] = prefix(000000*) として表現できる．さらに 1023 < x は [1024,] = [0000010000000000,] = prefix(000001*), prefix(00001*), (0001*), (001*), (01*), (1*) として表現可能であり, これもプレフィックス照合として取り扱える．以上より, 完全一致照合，範囲照合はプレフィックス照合で代替可能であり，最長一致アルゴリズムを適用すれば, パケット分類に必要な照合処理が実現できる．

照合法をプレフィックス一致に限定することにより，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．また，照合法の置換により，場合によっては照合データの表現が冗長になるが，照合データを統一することにより，マクロに照合データの圧縮が可能となり，結果として記憶コストの抑制につながる [13] ．

3.3　最長一致検索

パケット分類における分類条件の照合法としてプレフィックス一致照合をおこなうにあたり，最長一致検索アルゴリズムを適用する．

最長一致検索には，ツリー型のデータ構造によりルックアップテーブルを表現し，検査 bit に基づいて親ノードから小ノードへ分岐をし，検索を行う方式がある [14][15][16]．しかし，この検索方法では，ツリーをたどるときの枝分かれ毎に，比較判定と，時間のかかるメモリアクセスが必要なため，検索時間がかかる．また，この方式は汎用プロセッサ上でソフトウェアとして実行されるため，専用ハードウェアによる実現法ほど高速化が望めない．

ルーティングテーブルを Trie 構造に写像すると図 3.1 のようになるが，これでは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．

この問題を解決するために現在用いられている手法の一つが Path Compress 法であるで

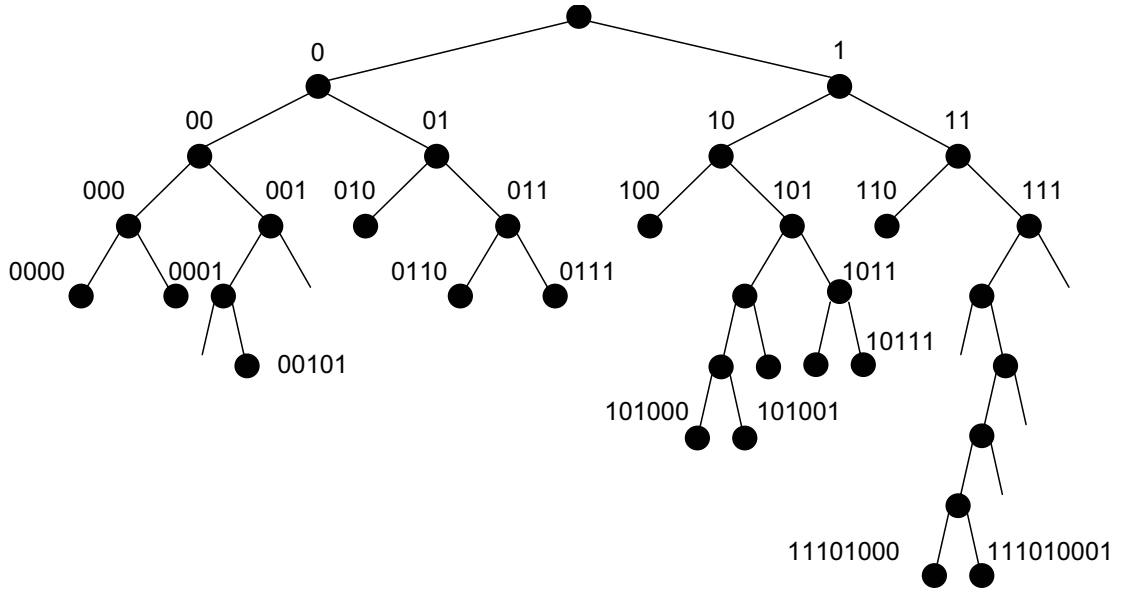

図 3.1 Binary Tree.

ある．この手法は，Binary Tree における子ノードが一つしかない中間ノードを削除し，パス検索に必要ないビットの長さをノードに保持することでパス長を縮小する方法であり，一般に Patricia Tree と呼ばれる．図 3.1 の Trie を Pass Compress することによって生成された Patricia Tree を図 3.2 に示す．Patricia Tree では，木構造におけるリーフの数を n としたとき，総ノード数は必ず $(2n-1)$ になる．この統計的な特徴はよく知られている．しかし，ツリー構造に写像する対象データの長さが長くなった場合に，Path Compress 法ではデータ数に比例して深さ方向にノード数が増え，平均深度の増加を抑制することができず，メモリアクセス回数，およびメモリ使用量の削減には至らない．

この問題を解決する方法が Level Compress 法である．Level Compress 法は各ノードが 2^k の子ノードを保持する MultiWay ツリー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，

- 検索木の深さはフォワーディングテーブルのエントリ数に対し $O(\log \log n)$ で増加する．
- 検索木の深さはアドレス長と独立しているため，大きなプレフィックスにも対応可能．

という特徴を持つ．これより得られた LC(Level Compressed)-Trie は平均深度の劇的な削

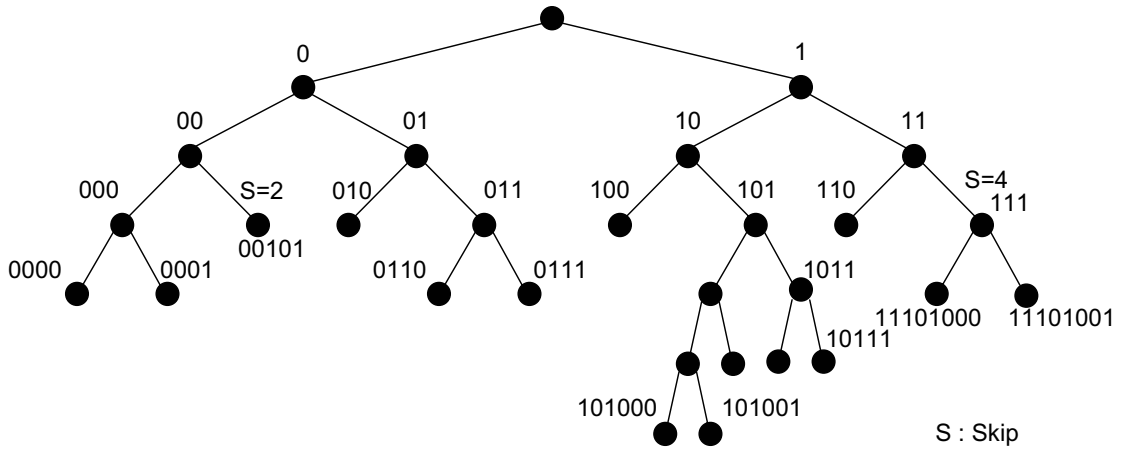

図 3.2 Patricia Tree.

減を可能とし，ツリー検索に必要なメモリアクセス回数の削減およびメモリ使用量の削減が望める．

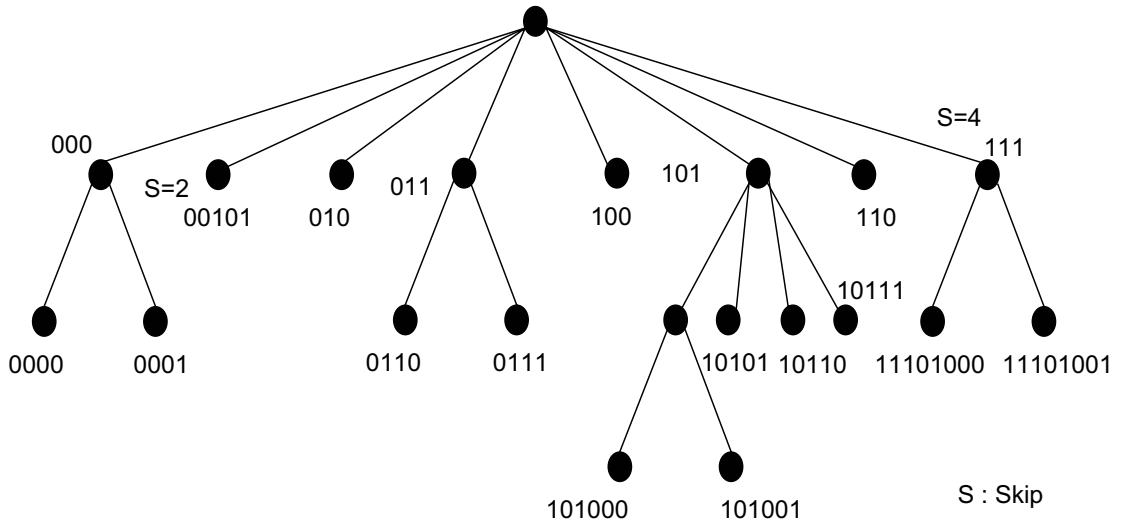

図 3.3 Level Compress Trie.

LC-Trie の各ノードは検索処理を可能とするため，以下のデータを持つ．

- Branch： 値が k のとき，ノードは 2^k 個の子ノードを持つことを示す．また，$k = 0$ のときは，ノードが終端 (リーフ) であることを示す．
- Skip： Path Compress を行った際に省かれた深さ方向のノード数を示す．検索時に検索 bit を Skip 分だけ飛ばす．

- Pointer： ノードが持つ一つの子ノードへのポインタである．図**??**の LC-Trie 木を想定した場合，ポインタはノードが持つ子ノードの左端を指す．リーフの場合は，プレフィックスを格納しているテーブルへのポインタを返す．

図 3.3 のデータ構造を配列表現を表 3.1 に示す．

表 3.1 LC-Trie 木の配列表現

	Branch	Skip	Pointer
0	3	0	1
1	1	0	9
2	0	2	2
3	0	0	3
4	1	0	11
5	0	0	6
6	2	0	13
7	0	0	12
8	1	4	17
9	0	0	0
10	0	0	1
11	0	0	4
12	0	0	5
13	1	0	19
14	0	0	9
15	0	0	10
16	0	0	11
17	0	0	13
18	0	0	14
19	0	0	7
20	0	0	8

表 3.2 プレフィックスの配列表現

nbr	string
0	0000
1	0001
2	00101
3	010
4	0110
5	0111
6	100
7	101000
8	101001
9	10101
10	10110
11	10111
12	110
13	11101000
14	11101001

3.4　検索木の平衡化

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，各ルールの先頭数ビットを基に検索木を分割する手法 [11] 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．

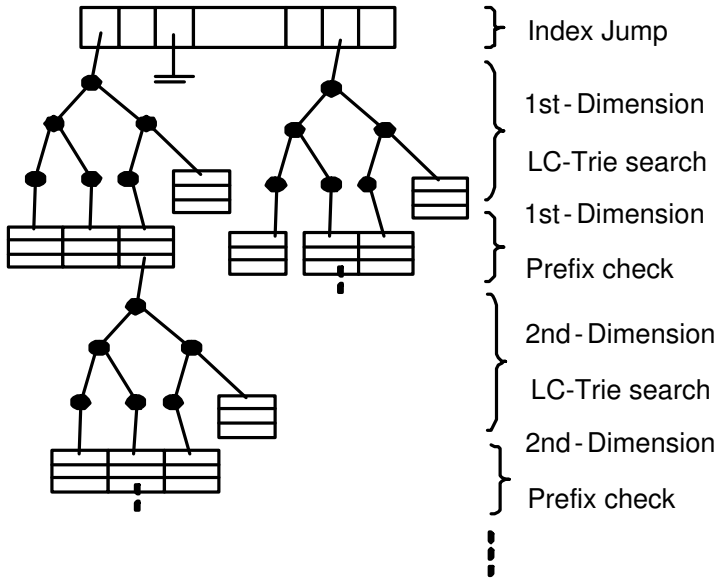

図 3.4　パケット分類処理の分類木構造.

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 3.5 に示す計算式を用いて，各キーフィールドの先頭 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．

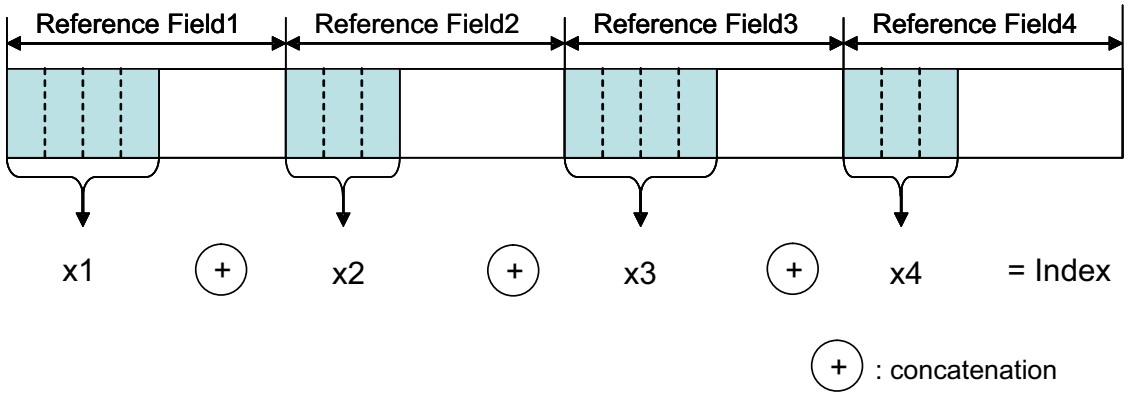

図 3.5　Index Jump に用いる Index の生成法.

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 $x_i(1 \leqq i \leqq n)$ 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 [17] から 4 次元のルールセットを作成して調べたところ，Index Jump テーブルのエントリ数が増加しても，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 つのキーフィールドを宛先/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ことを確認した．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．このような場合には，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 $(N+1)$ に拡張することで，Index Jump テーブルによる分散効果を大きくできる．

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選択ビットがすべて同じ値を示すことが多い．これでは 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．このような場合には，図 3.6 のように，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 テーブルの分散を大きくできると考えられる．

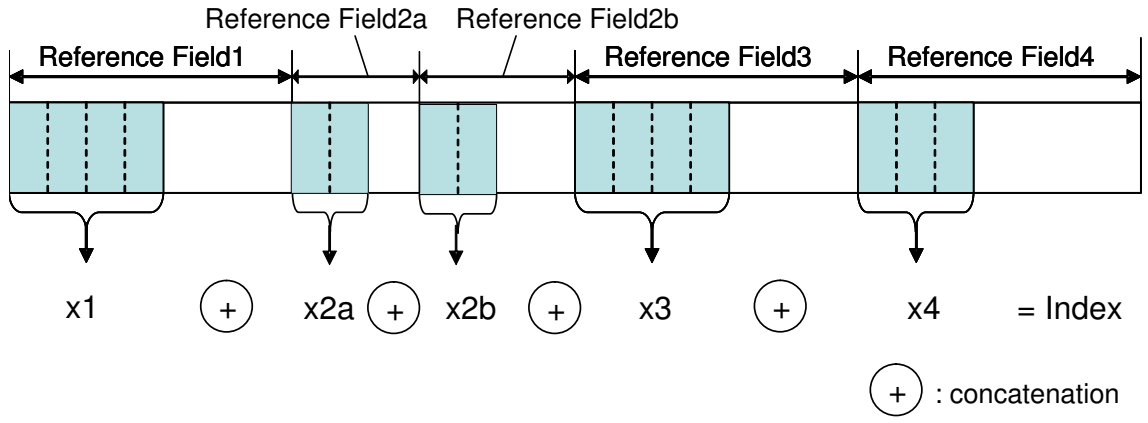

図 3.6　次元数の拡張.

3.5　分類方式のパイプライン並列化

分割時に使用した先頭ビットは, 各検索木を示す Index Jump テーブルのアドレスとして使用し, パケット分類時には, まず入力パケットの各フィールドから同テーブルのアドレスを計算し, グループを選択, そこから得られる検索木に対して最長一致検索処理を行う. すなわち, Index Jump 部とツリー検索部の 2 つのモジュールで構成し, ツリー検索部は, LC-Trie 検索部と Prefix 検査部から構成する. このとき，各処理におけるメモリ参照が全体のボトルネックとなる．そのため，各モジュールはメモリアクセスを伴う処理単位で構成され, これらのパイプライン並列処理によってメモリアクセス遅延を隠蔽し, メモリアクセスレートと等価な最大検索レートを実現している.

1. 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．Index Jump に使用されるアドレスは前節で述べたように, 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である.
2. フィールド検索部 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, LC-Trie 検索アルゴリズムを用いている．これによって，検索木を深さ方向に圧縮して，平均メモリアクセス回数を極小化できる．図 3.7 に示すように，本モジュールは LC-Trie 木を探索する LC-Trie 検索部と，木探索の結果を検査する Prefix 検査部からなる．これらをパイプライン並列に次元数だけ繰り返して，検索処理を行う.

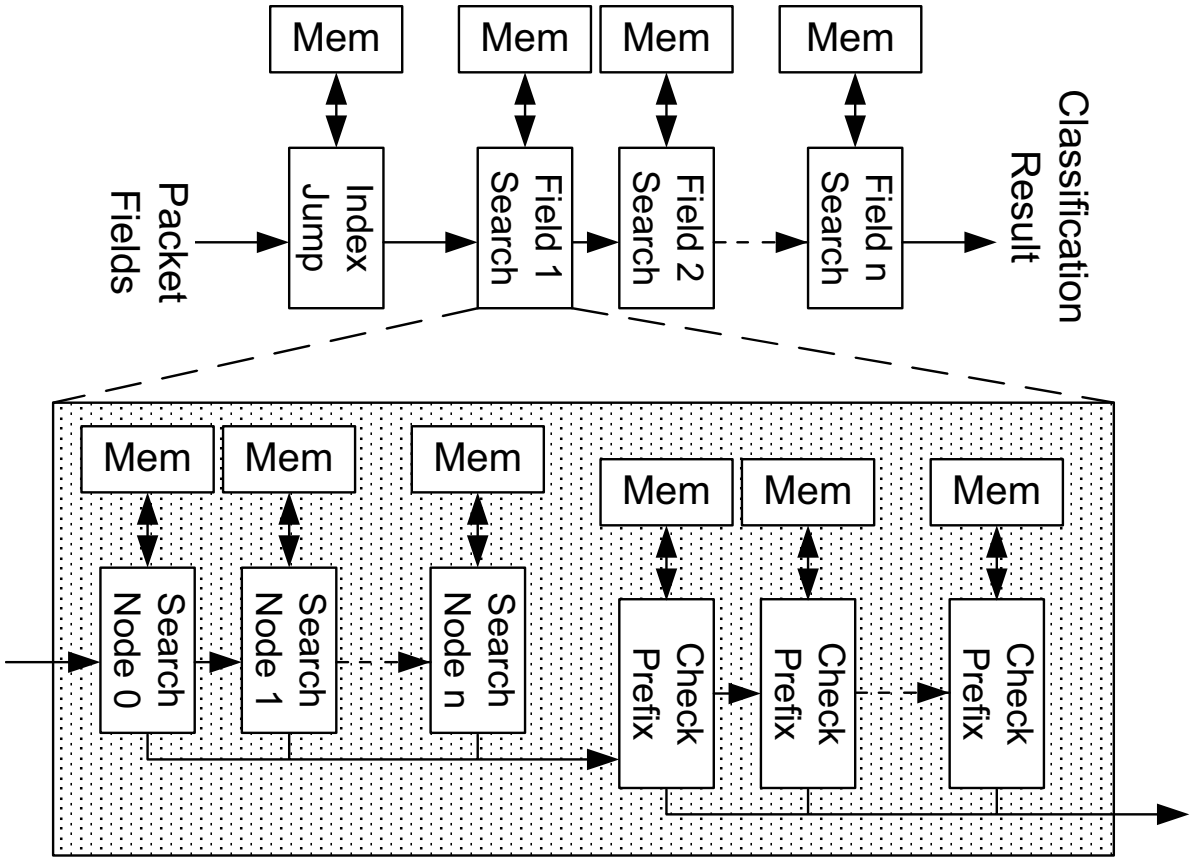

図 3.7　パケット分類処理のパイプライン並列化.

これらを，既存の専用 LSI と同程度の高速性を持ち，かつソフトウェア処理可能なデータ駆動型プロセッサ上に構成する．Index Jump テーブル参照のためのアドレスは，現行データ駆動型プロセッサの単純な論理演算により高速に計算する．また，最長一致検索処理は，LC-Trie 検索の処理遅延を極小化する特徴命令の実装と，調停機能付きメモリ I/F と分散メモリ構成により実現する [19].

3.6　結言

本章では，検索空間の増大による分類レートの急激な低下を防ぐため，各フィールドの照合処理を流れ作業的にパイプライン並列に順次実行する方式を提案した．まず，分類規則の照合法をプレフィックス一致に統一し，最長一致検索アルゴリズムを用いることにより，計算コストの削減を行うと共に，記憶空間の効率的な利用を可能とした．最長一致検索アルゴ

リズムに，検索遅延の少ない，パイプライン並列化の容易な LC-Trie を用いて，検索遅延の極小化を図った．さらに，大規模化するルールセットに対して，Index Jump テーブルを導入することにより，検索木のバランスをとり，検索木の偏りを緩和した．そしてこれらを，メモリ参照遅延を隠蔽するため，メモリ参照を伴う処理単位でパイプライン並列化した．

第 4 章

DDNP における提案パケット分類方式の実現法

4.1 緒言

現在，単一の VLSI チップ上で統合された，自己同期型のスーパーパイプラインで構成され，ビデオ信号処理アプリケーションに特化したデータ駆動マルチプロセッサ DDMP が開発されている [20]．本チップは，低消費電力 1.2 ワット未満で，8600MOPS(Million Operations per Second) を達成することができる．

本章では，ネットワーク処理に特化したデータ駆動型ネットワークプロセッサにおけるパケット分類処理機構の実現法を示す．

4.2 データ駆動型ネットワークプロセッサ：DDNP

データ駆動手法は，処理に必要なデータがすべて揃ったときに，プログラムノードによって示された命令を独立して行うことが可能である．特に，動的データ手法は，データインスタンスの異なるプログラムの並列実行を，コンテキスト切り換え，スケジューリングなしで行うことが出来る．すなわち，データパケットにより動作が制御されるため，多数のプロセッサは効率的に協力することができる．そのため，それらの処理能力はプロセッサ間の協調制御の必要がない．さらに，DDMP では回路実装そのものが自己同期型のパイプラインハードウェアによって実現されており，自然にパイプライン並列化が行われる．さらに，図

4.1 で示される相互プロセッサルータのようなマルチ入出力パケットルータに接続されるため，各ナノプロセッサ (nPE) が異なる命令セットを操作することが可能であり，パケット処理に特化した nPE として，システムの拡張が容易に行える．

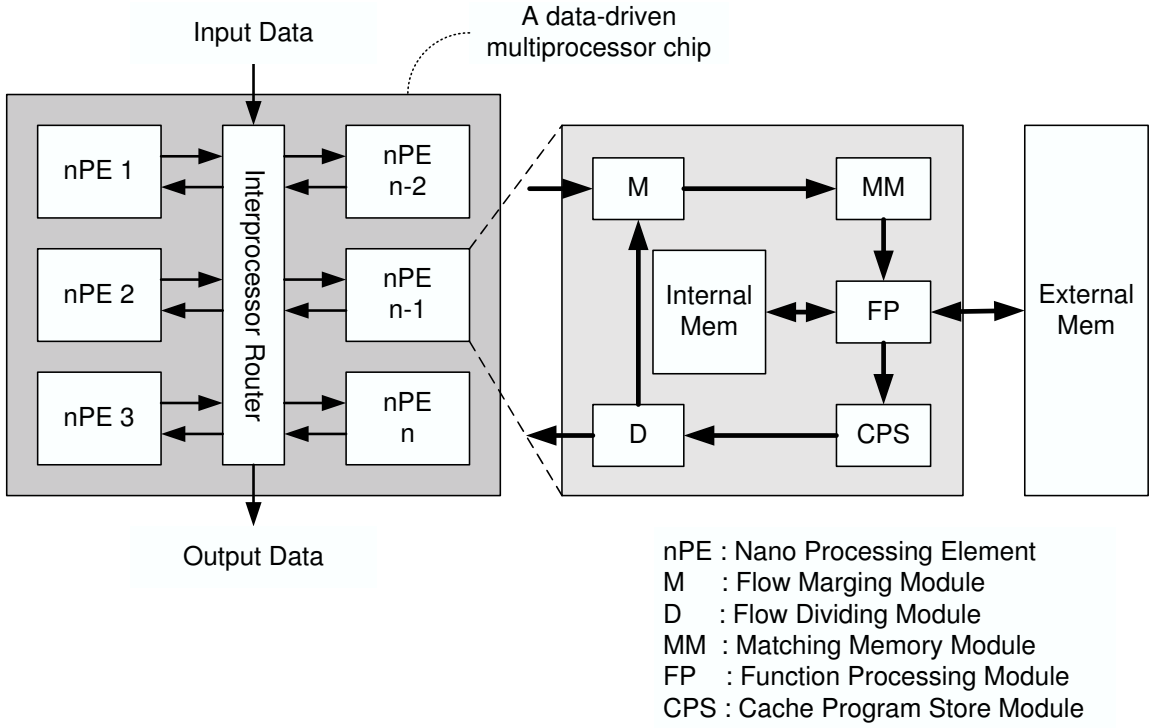

図 4.1　データ駆動型ネットワークプロセッサ構成.

これらの性質によって，図 4.2 に示すように，ヘッダ処理や，キューイング機構，その他の演算機構を備えた DDNP にパケット分類に特化した機能回路を持つ nPE を容易に追加実装することが可能となる．

その上，本プロセッサはデータ駆動原理に基づいて構成されているため，プロセッサ内で処理負荷が大きくなり，他の処理が実行できない場合でも，様々な種類のプロセッサを容易にマルチプロセッサシステムとして相互に接続することで負荷分散を可能とする．

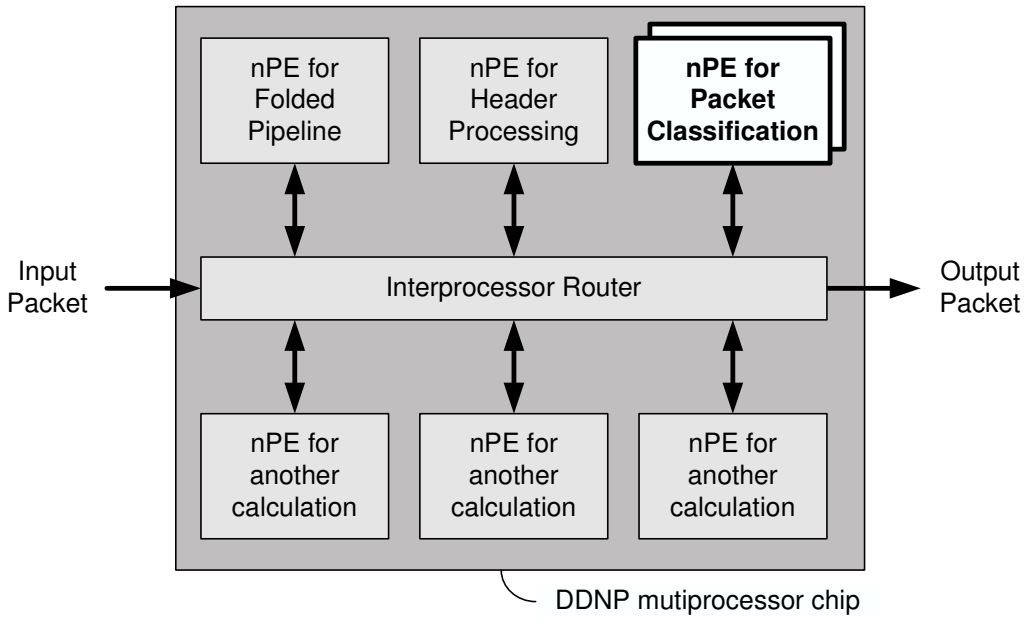

図 4.2　パケット分類用 nPE を持つ DDNP.

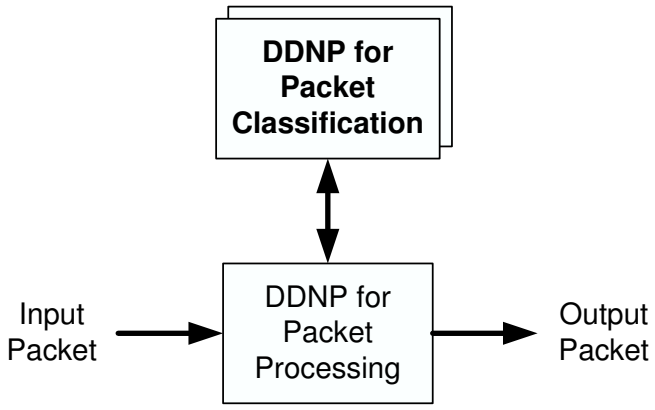

図 4.3　パケット分類機構を持つプロセッサと DDNP の接続.

4.3　データ駆動パケットフォーマット

データ駆動型プロセッサで扱うデータは，データ駆動パケットと呼ばれる形式で表される．パケットフォーマットを図 4.4 に示す．データ駆動パケットは，データ駆動原理を実現するため，演算データにデータの行き先情報，演算コード情報，プログラムのアドレス情報などの制御タグ (図中 Control Data) を付加したデータ形式である．データ部 (図中 DL) は，DDNP では IP パケット処理を対象とするため，4oct(32bit) の精度を採用した．さらに，処理に必要な中間データを保持する領域 (図中 DX) を付加し，1 パケットで処理対象データ

と中間データを完結することにより，プロセッサ内を滞留するパケット数の削減を行う．

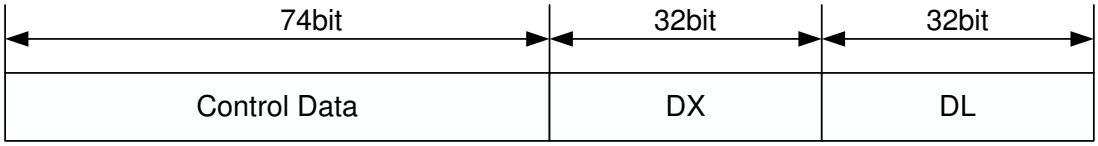

図 4.4　データ駆動型ネットワークプロセッサの基本パケットフォーマット．

4.4　パケット分類特徴命令

本研究では，メモリアクセス回数の削減，およびパイプライン並列による性能向上を達成するために，平均メモリアクセス回数の少ない LC-Trie 法を採用した．しかし，既存の命令セットは単純な処理機能しか有しておらず，検索アルゴリズムをソフトウェア的に実現すると，プログラム全体の命令数およびクリティカルパス長が大きくなり，効率良く実現できない．そこで，LC-Trie 法の特性に基づいて，命令の複合化を行い，プログラム中の命令数を極小化することで，検索レートの向上と検索時間の短縮を図る．

LC-Trie は以下の 2 つの処理ステージで実現される．そこで，各ステージの処理内容に沿って複合化した特徴命令を作成した．

- LC-Trie 木検索
- プレフィックス検査

4.4.1　LC-Trie 木検索命令

LC-Trie 木検索命令 (SearchTrie) は，LC-Trie 木の 1 レベル分の検索を行うための特徴命令である．本命令は，以下の処理をパイプライン並列に行う．

1. テーブル (LC-Trie Table) アドレスの計算
2. ノード情報の取得
3. ノード情報から検索終了したかどうかの判定

4. 検索終了：プレフィックス検査データへのポインタを出力
5. 検索未終了：次ノード取得用ポインタを出力

検索過程のフローを図 4.5 に示す.

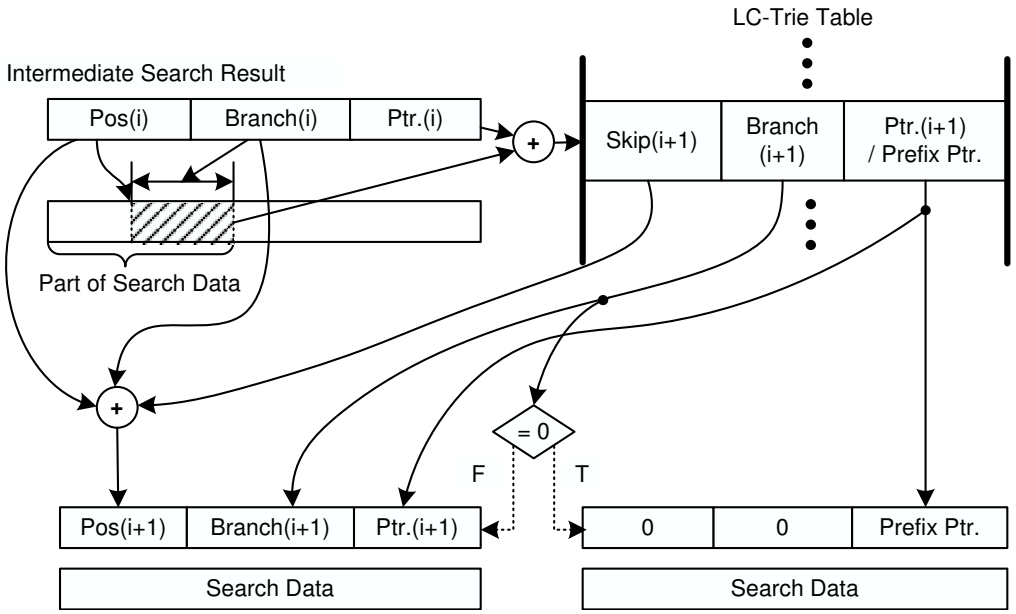

図 4.5　LC-Trie 木検索命令.

本命令は 32 ビットデータ (IPv4 アドレス等) を用いる命令と，32 ビット以上のデータ (IPv6 アドレス等) を扱う命令があり，ソフトウェアにより選択的に使用可能である．以下に各命令の動作を示す.

- SRCHTRI4

  32bit データに対して，内蔵メモリまたは外付けメモリ内の検索テーブルを参照し，検索処理を行う命令である．図 4.6 に示すとおり，1 つの入力パケットを受け，処理結果によって選択出力する.

  入力パケットの DX データ領域は，検索する 32bit のデータが格納される．入力パケットの DL データ領域は，図 4.7 に示すフィールドに分割されて解釈される．LC-Trie は，前章で述べたとおり，ノード情報として Branch，Skip，Pointer を保持しており，DL データ領域の Branch，Pos，Ptr はそれぞれに対応する．ただし，Skip フィールドに

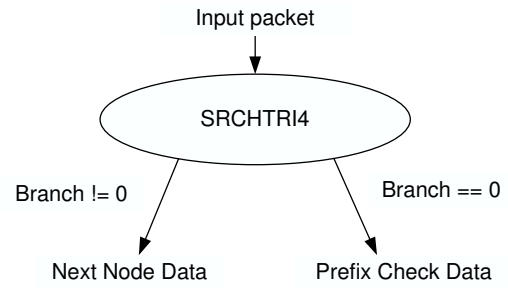

図 4.6 SRCHTRI4 命令.

関しては，処理の簡略化のため本実現においては現ノードからの省略分を表すのではなく，プレフィックスの先頭から，Skip 後の位置までのビット数を表す.

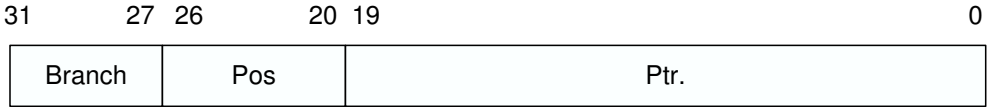

図 4.7 SRCHTRI4 命令における DX データ領域のフィールド解釈.

DL データ領域の Branch，および Pos の値を使用して，DX データ領域から図 4.8 に示すビットフィールドを抜き出し，そのフィールドを整数とみなして値 Val を得る.

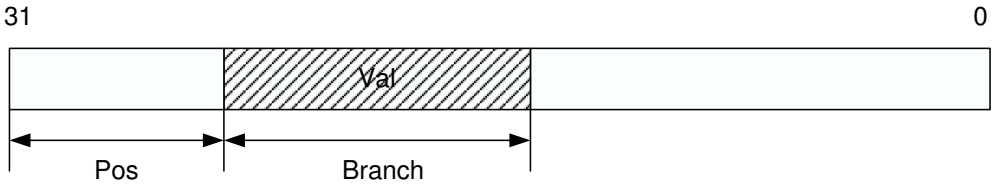

図 4.8 SRCHTRI4 命令における DX データ領域のフィールド解釈.

DX データ領域の Ptr にオフセットに相当する値 Val を加えた値が，次ノードを得るための検索テーブルのアドレスとして使用される．得られたノード情報は，DL データ領域に格納される．その後，DL データ領域の Branch が 0 であれば，LC-Trie 木の検索が終了したとみなす．このとき，DL データはプレフィックス検査を行うためのデータを保持していることになり，プレフィックス検査処理へパケットが出力される．DL データ領域の Branch が 0 で無ければ，さらに子ノードを保持しているということを意味するため，Branch が 0 になるまで上記の処理を繰り返す.

- SRCHTRI6

  64bit データに対して，内蔵メモリまたは外付けメモリ内の検索テーブルを参照し，検索処理を行う．図 4.9 に示すとおり，2 つの入力パケットを受け，処理結果によって 2 パケットを選択出力する．

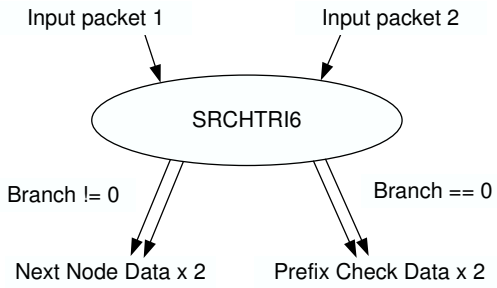

図 4.9 SRCHTRI6 命令.

各入力パケットの DX データ領域には，それぞれ検索する 32bit のデータが格納される．各データは，図 4.10 に示すように結合され，64bit データとして解釈される．

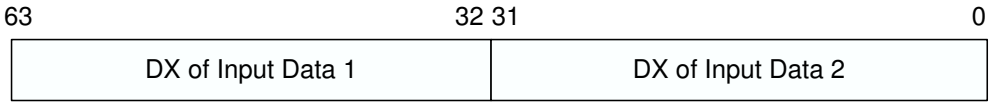

図 4.10 SRCHTRI6 命令における DX データ領域のフィールド解釈.

入力パケット 1 の DL データ領域は，図 4.11 に示すフィールドに分割されて解釈される．LC-Trie は，前章で述べたとおり，ノード情報として Branch，Skip，Pointer を保持しており，DL データ領域の Branch，Pos，Ptr はそれぞれに対応する．ただし，Skip フィールドに関しては，処理の簡略化のため本実現においては現ノードからの省略分を表すのではなく，プレフィックスの先頭から，Skip 後の位置までのビット数を表す．

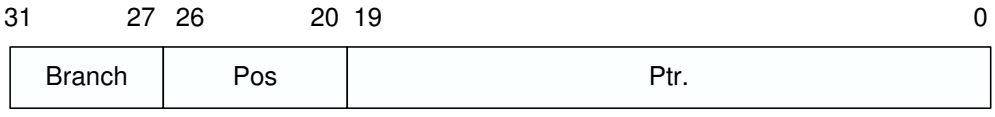

図 4.11 SRCHTRI6 命令における DX データ領域のフィールド解釈.

DL データ領域の Branch，および Pos の値を使用して，DX データ領域から図 4.8 に

示すビットフィールドを抜き出し，そのフィールドを整数とみなして値 Val を得る．

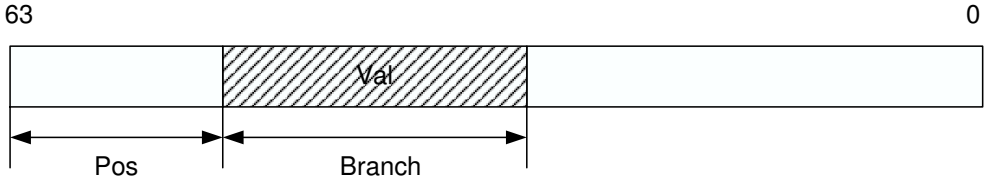

図 4.12　SRCHTRI4 命令における DX データ領域のフィールド解釈．

DX データ領域の Ptr にオフセットに相当する値 Val を加えた値が，次ノードを得るための検索テーブルのアドレスとして使用される．得られたノード情報は，双方のパケットの DL データ領域に格納される．その後，DL データ領域の Branch が 0 であれば，LC-Trie 木の検索が終了したとみなす．このとき，DL データはプレフィックス検査を行うためのデータを保持していることになり，プレフィックス検査処理へ 2 つパケットが出力される．DL データ領域の Branch が 0 で無ければ，Branch が 0 になるまで上記の処理を繰り返す．

4.4.2　プレフィックス検査命令

プレフィックス検査命令 (CheckPrefix) は，SearchTrie 命令によって得られたプレフィックスの正当性を検査し，NextHop 情報を得るためのポインタを出力する命令である (図 4.13)．SearchTrie 命令と同様に，以下の処理をパイプライン並列に行う．

1. プレフィックスと検査データの取得

2. IP アドレスとプレフィックスの比較

3-1. 一致：検索結果の出力

3-2. 不一致：プレフィックス検査用ポインタの出力

検査過程のフローを図 4.13 に示す．

本命令は，SearchTrie 命令同様，32 ビットデータ (IPv4 アドレス等) を用いる命令と，32 ビット以上のデータ (IPv6 アドレス等) を扱う命令があり，ソフトウェアにより選択的に使

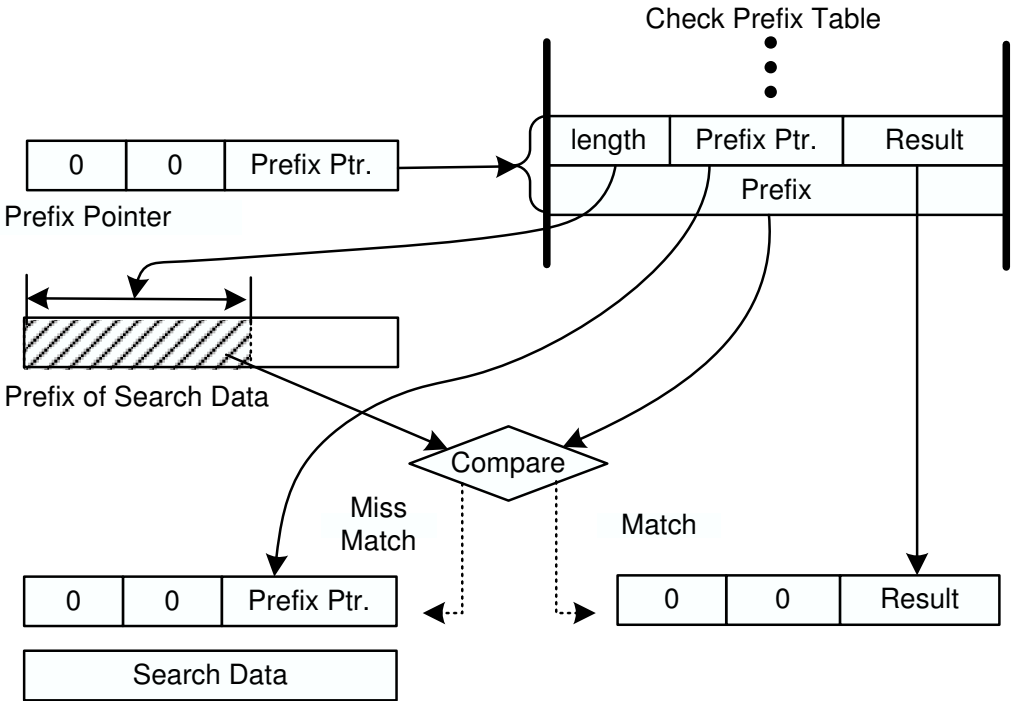

図 4.13 プレフィックス検査命令.

用可能である.

- CHKPRFX4

  32 ビットデータに対して，内蔵メモリまたは外付けメモリ内のプレフィックス検査用データ 2 ワードを参照し，プレフィックス検査処理を行う．図 4.14 に示すとおり，1 つの入力パケットを受け，検査結果によってパケットを選択出力する.

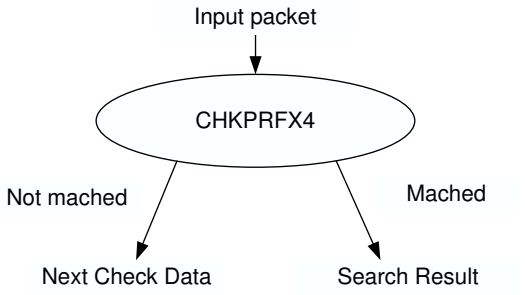

図 4.14 CHKPRFX4 命令.

入力パケットの DX データ領域は，検査する 32bit のデータが格納される．入力パケットの DL データ領域は，メモリ参照用のアドレスとして使用する．まず，DL データ領

域の値を基にメモリ参照を行い，プレフィックス検査用データ (図 4.15-(A)) を取得する．同時に，DL データ領域の値に 1 を加算したものでメモリ参照を行い，検査対象とするプレフィックス (図 4.15-(B)) を取得する．

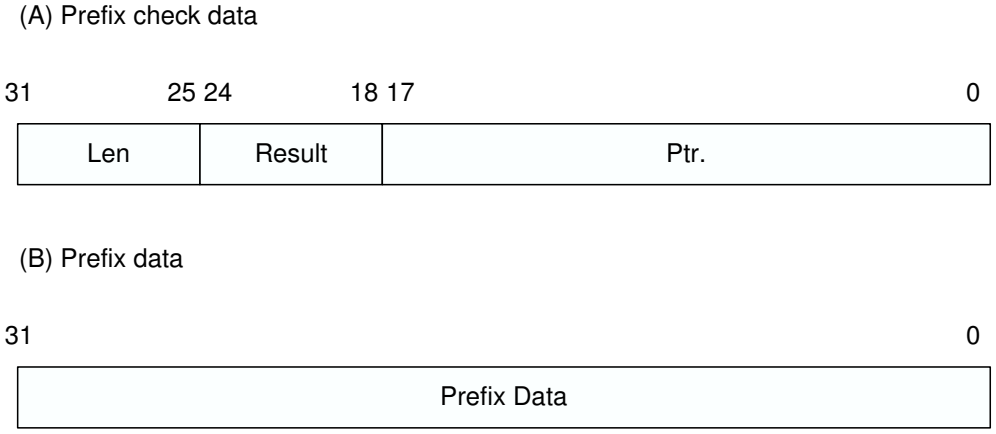

図 4.15　CHKPRFX4 命令で使用するデータフィールドの解釈.

検査データである DX データ領域の上位側 Len ビットと，Prefix Data の同じく上位側 Len ビットとの間で，一致・不一致の比較を行う．

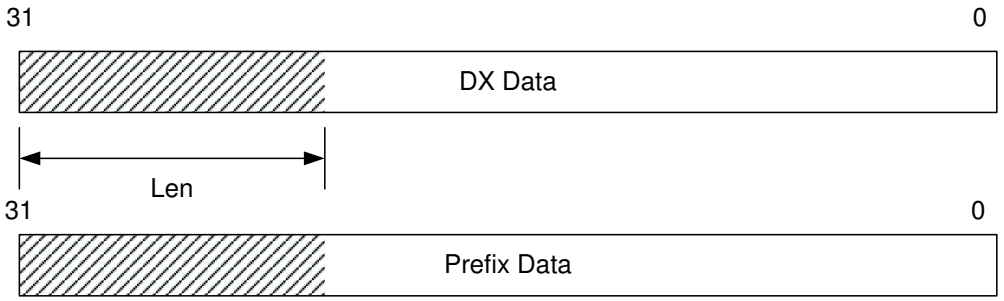

図 4.16　CHKPRFX4 命令におけるプレフィックスの一致・不一致検査.

一致した場合は，プレフィックス検査用データ中の Result を DL データ領域に格納し，検索結果として出力される．不一致の場合は，プレフィックス検査用データ中の Ptr を DL データ領域に格納し，プレフィックス再検査用ポインタとして出力される．

- CHKPRFX6

  64 ビットデータに対して，内蔵メモリまたは外付けメモリ内のプレフィックス検査用データ 3 ワードを参照し，プレフィックス検査処理を行う．図 4.17 に示すとおり，1 つ

の入力パケットを受け，検査結果によってパケットを選択出力する．

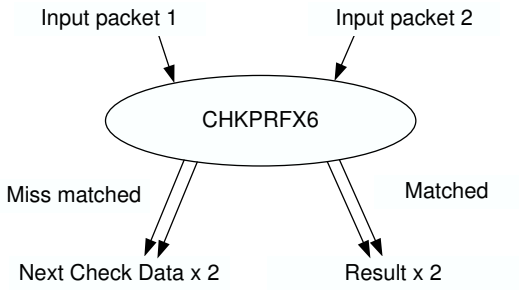

図 4.17 CHKPRFX6 命令.

各入力パケットの DX データ領域には，それぞれ検索する 32bit のデータが格納される．各データは，図 4.18 に示すように結合され，64bit データとして解釈される．

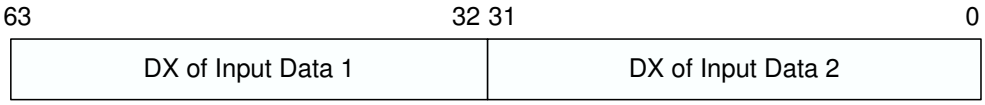

図 4.18 SRCHTRI6 命令における DX データ領域のフィールド解釈.

入力パケット 1 の DL データ領域は，メモリ参照用のアドレスとして使用する．まず，DL データ領域の値を基にメモリ参照を行い，プレフィックス検査用データ (図 4.19-(A)) を取得する．同時に，DL データ領域の値に 1 を加算したものと，2 を加算したものをアドレスとしてメモリ参照を行い，検査対象とするプレフィックスの下位 32 ビット (図 4.19-(B)) と，プレフィックスの上位 32 ビット (図 4.19-(C)) をそれぞれ取得する．

検査データである DX データ領域の上位側 Len ビットと，Prefix Data の同じく上位側 Len ビットとの間で，一致・不一致の比較を行う．

一致した場合は，プレフィックス検査用データ中の Result を DL データ領域に格納し，検索結果として 2 パケットを出力される．不一致の場合は，プレフィックス検査用データ中の Ptr を DL データ領域に格納し，プレフィックス再検査用ポインタとして 2 パケットを出力される．

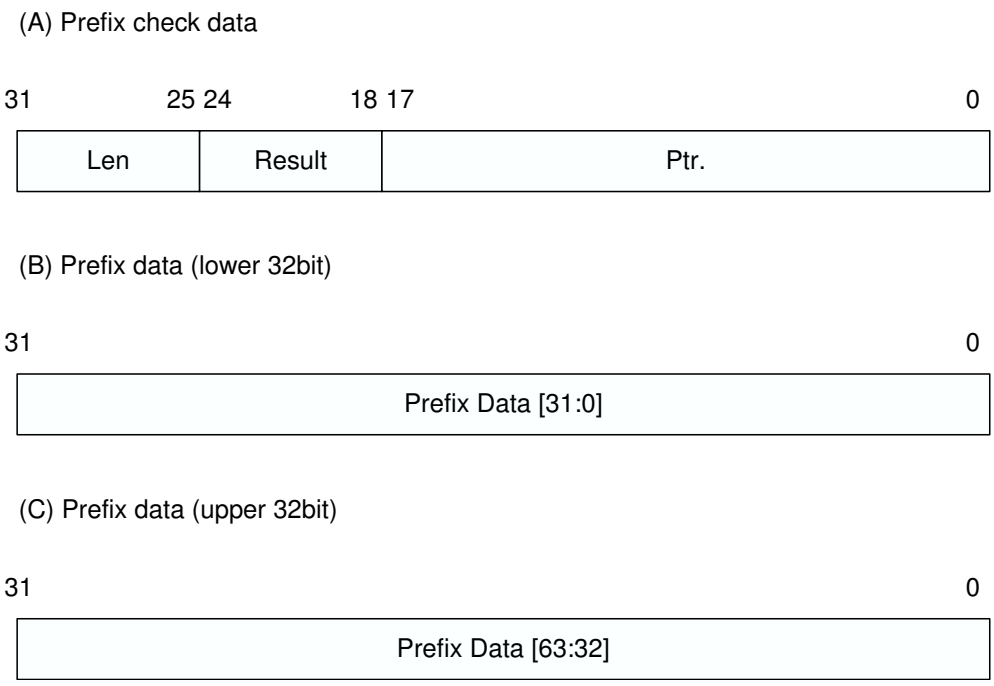

図 4.19 CHKPRFX6 命令で使用するデータフィールドの解釈.

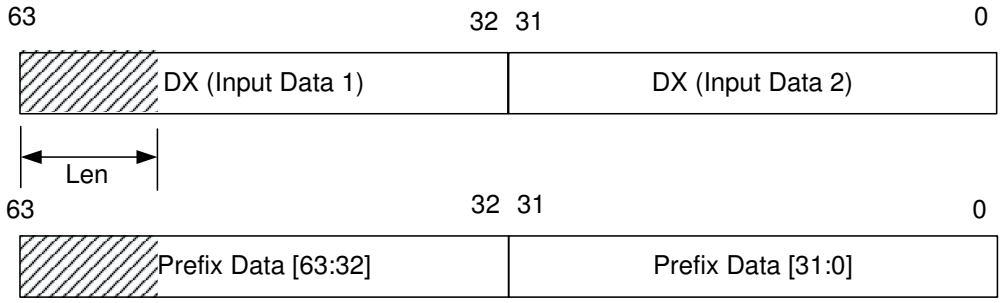

図 4.20 CHKPRFX6 命令におけるプレフィックスの一致・不一致検査.

- CHKPRFXD

64 ビットデータに対して，内蔵メモリまたは外付けメモリ内のプレフィックス検査用データ 3 ワードを参照し，プレフィックス検査処理を行う．CHKPRFX6 の機能に加え，128 ビットプレフィックス検査に向けた拡張機能を有している．図 4.21 に示すとおり，1 つの入力パケットを受け，検査結果によってパケットを選択出力する．

各入力パケットの DX データ領域には，それぞれ検索する 32bit のデータが格納される．各データは，図 4.22 に示すように結合され，64bit データとして解釈される．

入力パケット 1 の DL データ領域は，メモリ参照用のアドレスとして使用する．ま

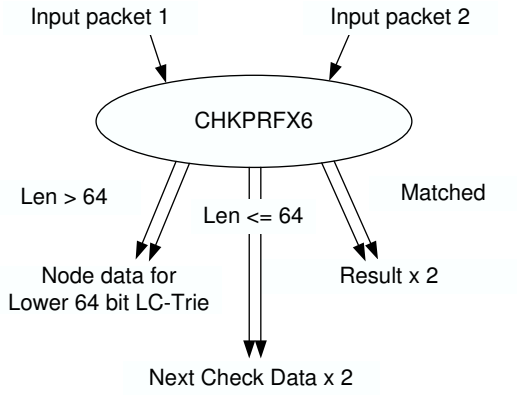

図 4.21 CHKPRFXD 命令.

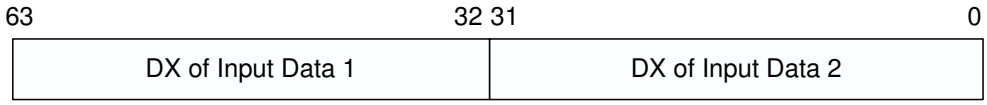

図 4.22 SRCHTRID 命令における DX データ領域のフィールド解釈.

ず，DL データ領域の値を基にメモリ参照を行い，プレフィックス検査用データ (図 4.23-(A)) を取得する．同時に，DL データ領域の値に 1 を加算したものと，2 を加算したものをアドレスとしてメモリ参照を行い，検査対象とするプレフィックスの下位 32 ビット (図 4.23-(B)) と，プレフィックスの上位 32 ビット (図 4.23-(C)) をそれぞれ取得する．

検査データである DX データ領域の上位側 Len ビットと，Prefix Data の同じく上位側 Len ビットとの間で，一致・不一致の比較を行う．

Len の値は 1 から 128 まで設定可能で，Len が 0 の場合は，128 とみなされる．64 以上が設定されると，64 ビット全ビット比較となる．

一致して，かつ Len $>$ 64 の場合は， プレフィックス検査用データ中の Ptr が DL データ領域に格納され，新たな LC-Trie 検査のためのポインタとして出力される．一致して，且つ Len $\leq$ 64 の場合は，Result の値が DL データ領域に格納され，検索結果として出力される．不一致の場合は，プレフィックス検査用データ中の Ptr の値が DL データ領域に格納され，プレフィックス再検査用ポインタとして出力される．

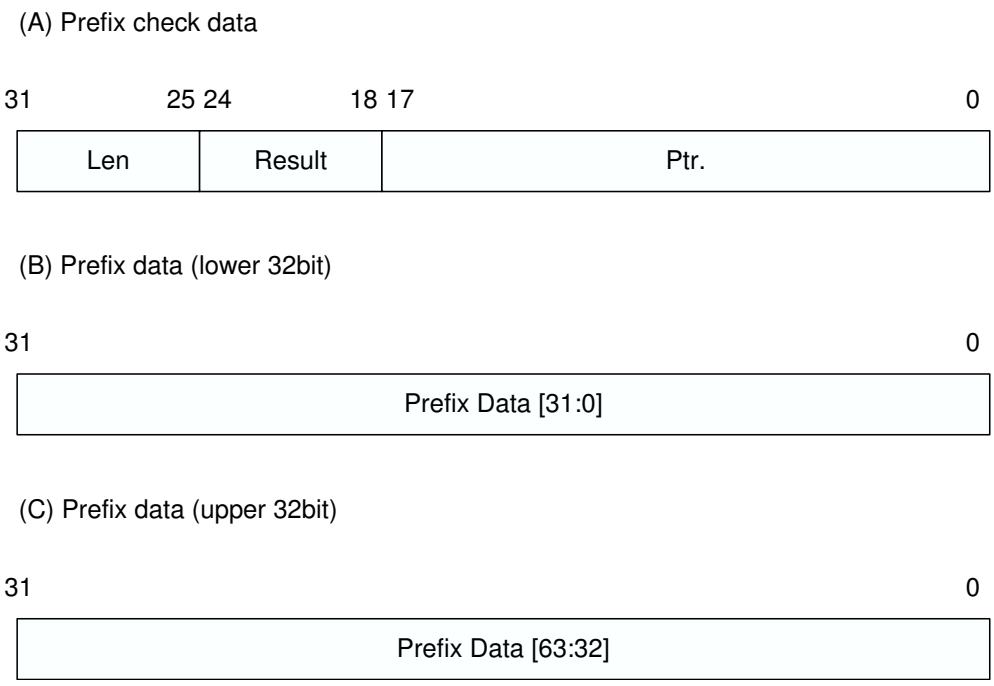

図 4.23　CHKPRFXD 命令で使用するデータフィールドの解釈.

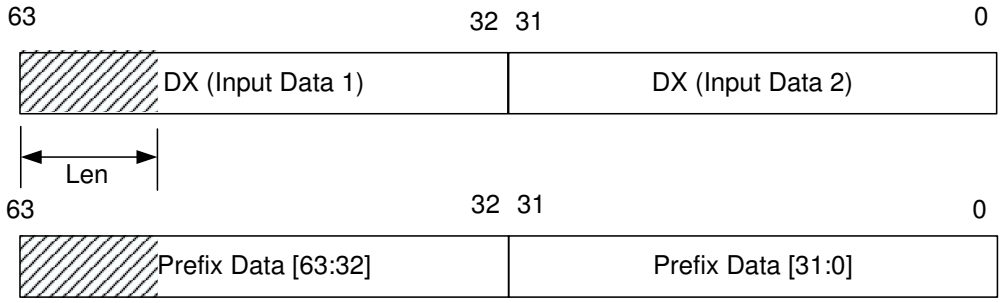

図 4.24　CHKPRFXD 命令におけるプレフィックスの一致・不一致検査.

4.5　パケット分類ソフトウェア

パケット分類処理用のデータ駆動プログラムは，前章で述べたように，Index Jump 部および Field Classification 部からなる．入力フィールドより，まず Index Jump テーブルの参照ポインタを得るための Index 計算を行う．Index 計算は単純なビット演算命令の組み合わせにより表現される．その後，フィールド数 n に応じて，SearchTrie 命令および CheckPrefix 命令を用いた最長一致検索処理を n 回行う.

図 4.25 はパケット分類処理のためのデータ駆動プログラムを示している.

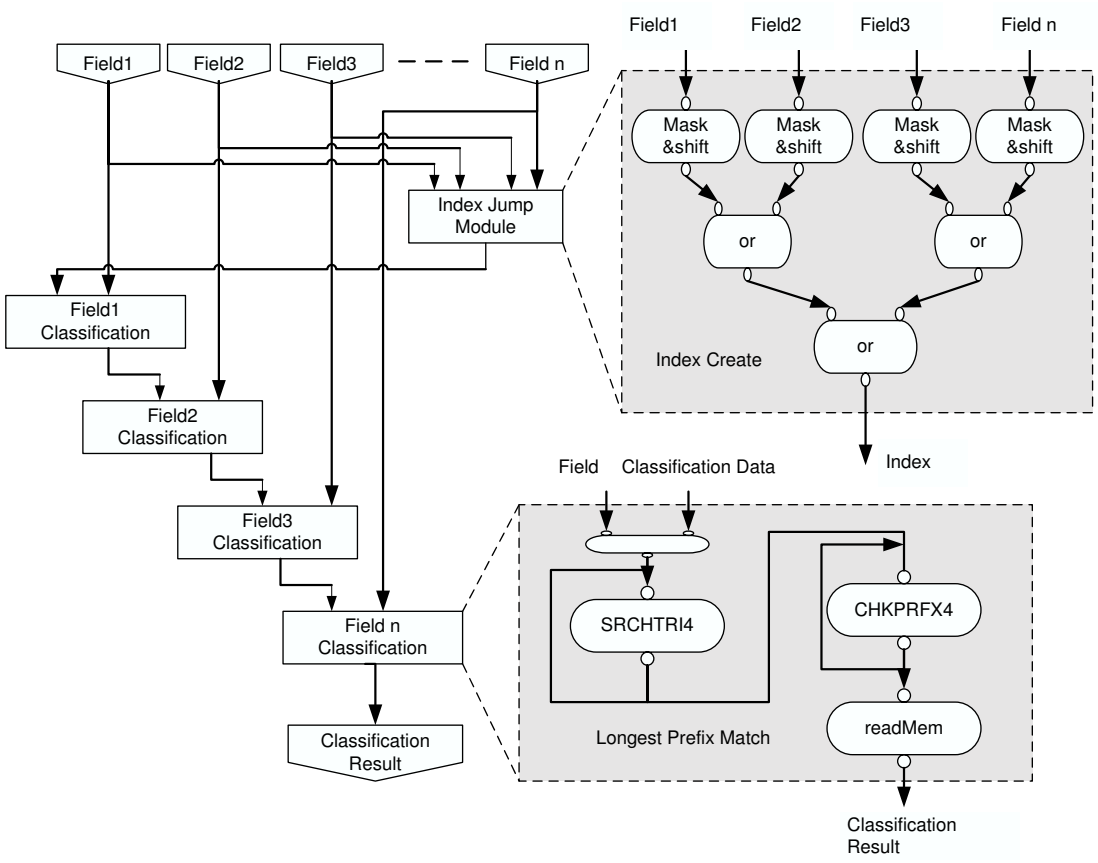

図 4.25　パケット分類処理のデータフローグラフ.

図 4.26 は，Field Classification 部における (A)32 ビットデータに対する最長一致検索と，(B)64 ビットデータに対する最長一致検索を示している．

(A) は IPv4 の宛先アドレス (32 ビット) から NextHop を求めるプログラムである．まず，入力された IP アドレスは，LC-Trie 検索用初期データを付加され，1 つのデータ駆動パケットで表現される．次に，LC-Trie 検索用命令 (SRCHTRI4) にて内部テーブル (或いは外部テーブル) に格納されている LC-Trie 木のノード情報を基に，木の探索を行う．そして，探索が終了すると，プレフィックス検査用命令 (CHKPRFX4) を用いて検索の正当性を検査する．プレフィックスが一致すると，テーブル参照命令 (readMem) を用いて NextHop を出力する．

(B) では IPv6 の最長一致検索を行う．ただし，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．

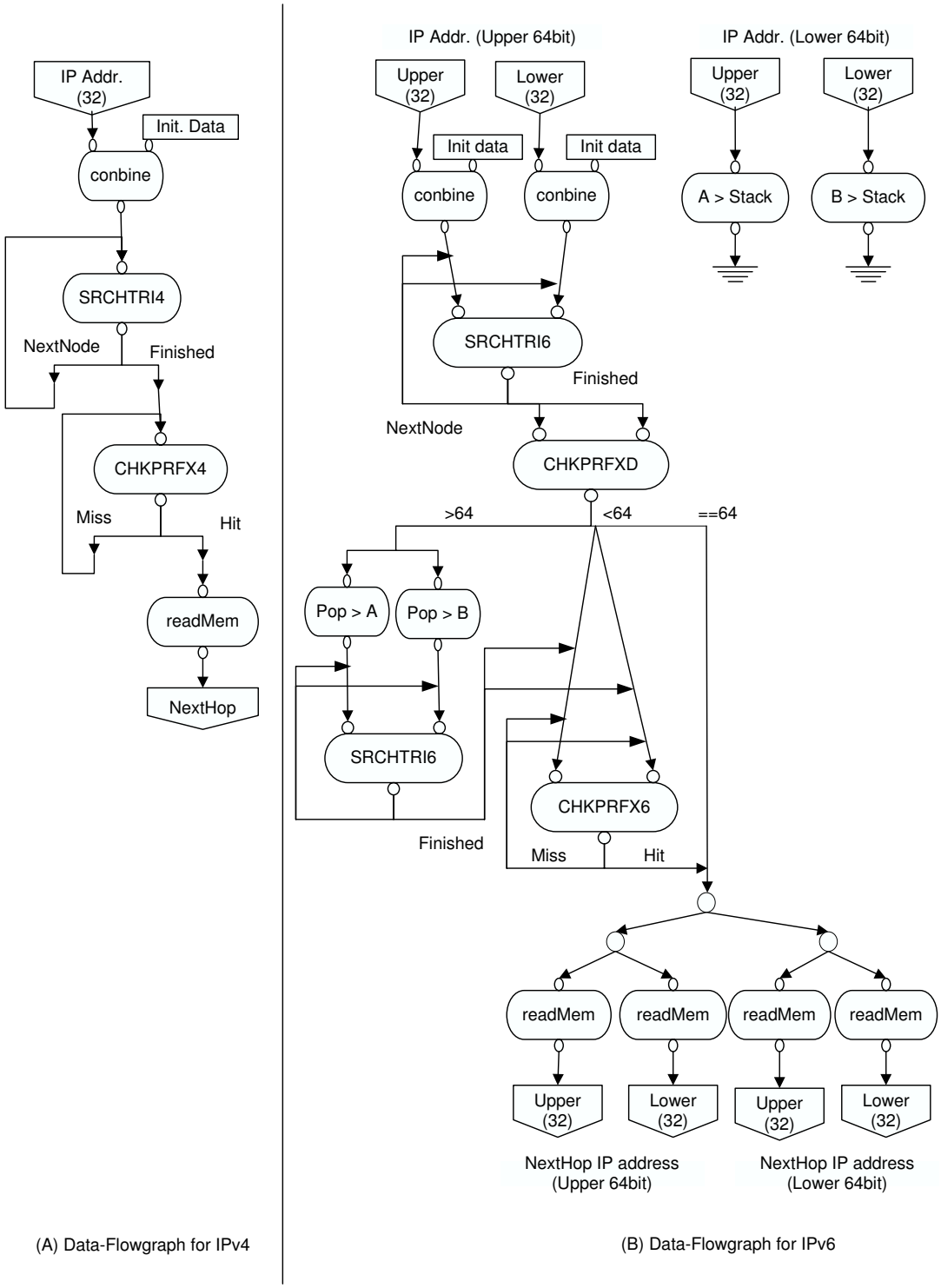

図 4.26　最長一致検索のデータフローグラフ.

4.6　ナノプロセッサ構成

ナノプロセッサを構成するに当たり，専用ハードウェア実現法の欠点である柔軟性の欠如，汎用プロセッサによる実現法の欠点である低性能の両者を克服するために，次の要件を満足する nPE を構成する必要がある．

- 必要最低限のハードウェア追加により追加命令を実現でき，既存の命令セットも実行可能なこと．
- メモリアクセスの並列パイプライン化によって，メモリ参照遅延を隠蔽できること．
- ルールセットの分布に応じて，最適なプログラム，およびメモリ割当てを柔軟に設定可能なこと．

本研究で使用するデータ駆動型プロセッサは，整数演算用 nPE，およびテーブル参照用 nPE(TBLnPE) から成るマルチプロセッサ構成である．基本構成は現行のデータ駆動型マルチプロセッサ DDMP と同等であり，整数演算用 nPE は，既存の整数演算用 nPE を使用する．テーブル参照 nPE は，メモリアクセス命令を主体とし，内蔵オンチップメモリ (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る．さらに，各テーブル参照用 nPE は独立してアクセス可能なメモリをそれぞれ持ち，計 2 個のメモリを外部テーブルとして持つ．これらのテーブル参照用 nPE はソフトウェアで選択的に使用できる．これより，それぞれのテーブル参照用 nPE に異なったテーブル情報を持つメモリ参照プログラムを実装できる．あるいは，2 つの nPE に同一のテーブル情報を分割配置することも柔軟に行える．本研究においてテーブル参照用 nPE は，最もメモリ参照の頻度が高い最長一致検索処理において使用する．このとき，既存のテーブル参照命令セットを継承しつつ，高速化を図るため後述する最長一致検索用特徴命令を実装する．

また，最長一致検索処理において，プレフィックス検査命令が 1 回の処理においてメモリ参照を 2 回行う必要があるため，処理遅延が大きくなる．そのため，プレフィックス検査を行う ExtTBLnPE を用意し，LC-Trie 検索命令を行う TBLnPE と協調動作させることで更なる高速化を図る．ExtTBLnPE は，図 4.27 に示すとおり，2 回のメモリ参照を 1 命令

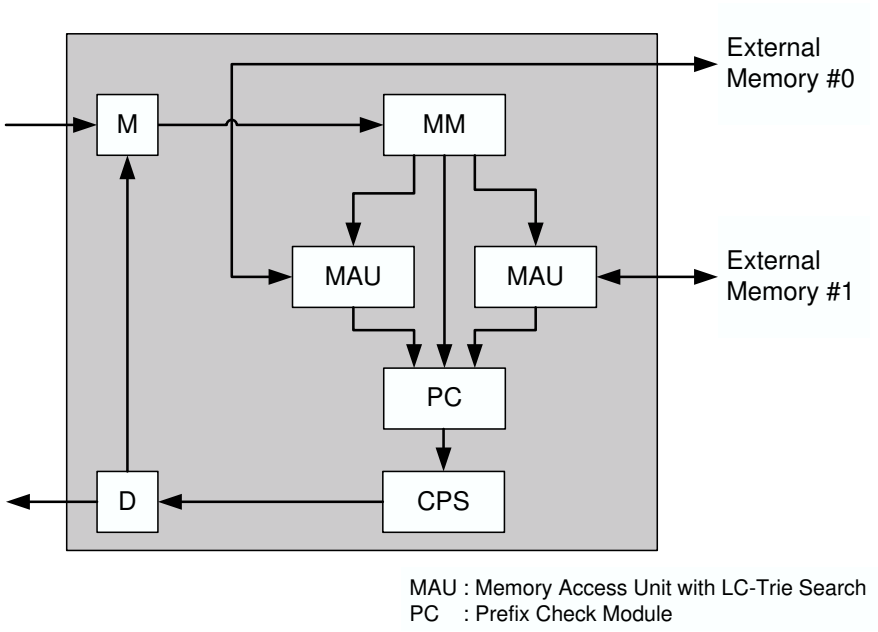

図 4.27　2 ポートメモリ参照機能搭載拡張 TBLnPE.

サイクルで実行するため，2 つの独立したメモリに並列にアクセスするよう設計している．

なお，TBLnPE と ExtTBLnPE が，同じメモリを共用するため，図 4.28 に示すように，nPE とメモリの間に調停機構を実装する．

本構成において，LC-Trie 検索用データとプレフィックス検査用データを各メモリに適宜分散格納することにより，処理負荷を低減する．

4.7　結言

本章では，DDNP のアーキテクチャを明確にし．既存の命令セットでは効率的な検索アルゴリズムのソフトウェア実現が困難であることから，最長一致検索処理に関する特徴命令を提案した．そして，それらを用いたソフトウェアの記述例を示した．さらに，メモリ参照を含む処理を効率的に実行可能な TBLnPE に提案特徴命令を追加実装した．その上で，外付けメモリ参照時にボトルネックとなるプレフィックス検査命令を効率的に実行するため，2 ワード分のメモリ参照を同時並列に行う 2 ポートメモリ参照機能搭載拡張 TBLnPE を構成した．これにより，プレフィックス検査時に発生するメモリ参照回数を半分に削減可能と

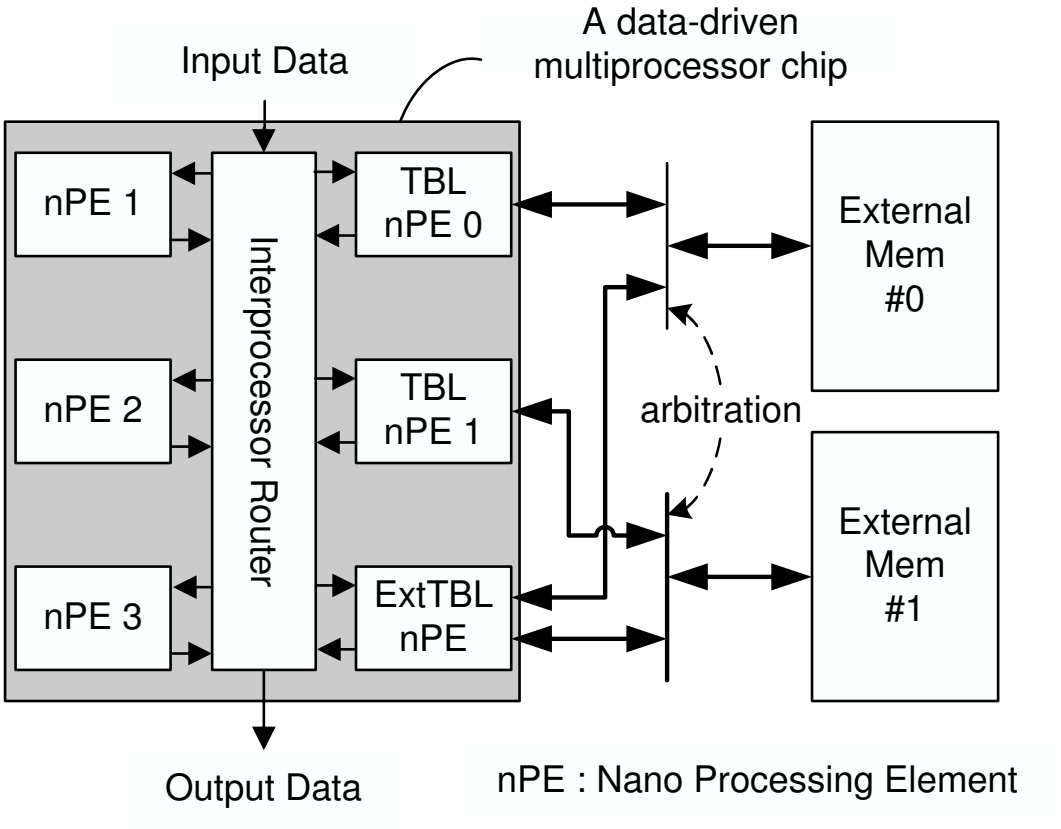

図 4.28　メモリの調停 I/F と分散メモリ構成.

なる．また，外付けメモリは TBLnPE と拡張 TBLnPE で共用するため，調停機構を付加する．

第 5 章

DDNP 用評価ボードの試作

5.1 緒言

本研究において，自己同期型パイプラインにより構成された IP パケット向きアーキテクチャに関する各種性能評価を実施するために必要なエミュレーション機能と性能を備えた，DDNP 用評価ボード (図 5.1) を試作した．

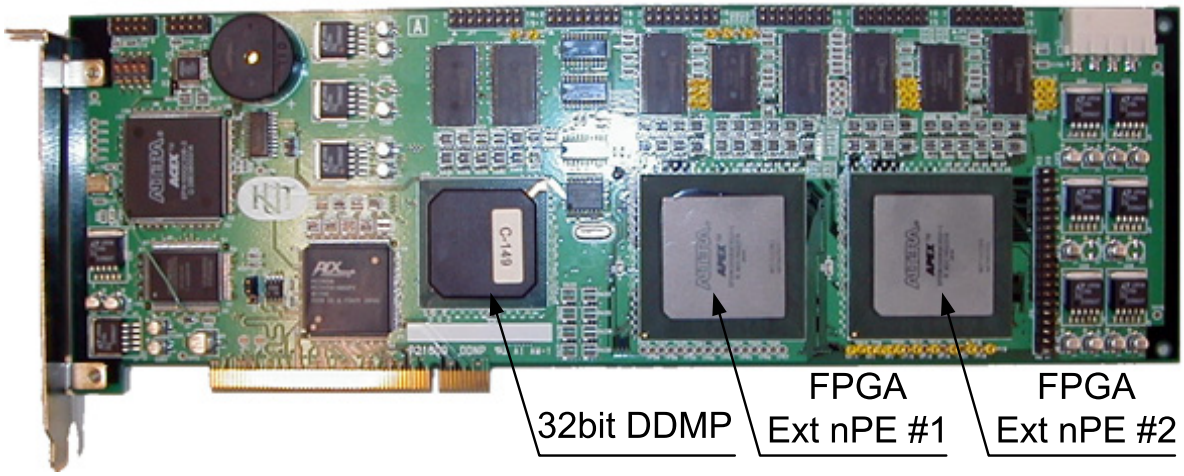

図 5.1　評価用 DDNP ボード．

本ボードに搭載されるデータ駆動型 VLSI チップは，自己タイミング型パイプラインハードウェア機構によりデータ駆動方式を実現したアーキテクチャ，特に，IP パケット処理向きアーキテクチャの各種性能評価を効果的に実施するために，次の要件を満たすこととする．

- エミュレーション評価の誤差を極力少なくするために，本研究テーマの基本原理である自己同期型パイプライン機構により実現されていること

- 多次元パイプライン処理構造を模擬するために，トークンの追い越しを許す動的データ駆動方式に基づき動作すること
- 多数のデータ駆動型プロセッサを有機的にパイプライン接続したシステムアーキテクチャの評価を実施するために，多様な接続構造をチップ上に実現する必要がある．したがって，複数のプロセッサが 1 チップ上に集積化され，その処理性能が通常の汎用マイクロプロセッサの性能以上のピーク性能を有していること

さらに，IP パケット処理向きアーキテクチャについて多用な評価を実施するために，本ボードには，回路構成を自由に再構成可能な FPGA チップが 2 個搭載されていることに加えて，次の要件を満たす，自己同期パイプラインによるデータ駆動型プロセッサがそれぞれ 1 つずつ実装されていること．

- 上記のデータ駆動型 VLSI チップと互換性のあるパケットフォーマットにより，入出力データの授受が可能なこと

以上の要件を満たすデータ駆動型 VLSI チップならびに FPGA チップを搭載したボードは，次の用件を満足していること．

- データ駆動型 VLSI チップと FPGA チップ間の接続には，エミュレーション評価の誤差を極力少なくするために，本研究テーマの基本原理である自己同期型パイプライン機構により実現されていること
- 本ボード上でのソフトウェアの実行状況を観測するための機能を有し，これらの機能がホストコンピュータ経由で制御可能なこと．

本研究においては，前章で述べた分類機構を含む，DDNP アーキテクチャの回路規模および性能評価をハードウェアレベルで行うため，FPGA に最長一致検索用の複合命令 SearchTrie 命令と CheckPrefix 命令，ならびに各種テーブルメモリ参照命令を含む命令セットを備えた nPE を実装した．本章では，本ボードの詳細仕様について述べる．

5.2　評価ボードの詳細仕様

本評価ボードは，シャープ (株) が試作した信号処理用 32bit データ駆動型プロセッサ (DDMP)，および FPGA チップ (APEX DSP DB EP20K 1500E) から構成され，今回新たに提案した特徴命令を含む nPE(Ext nPE#1/#2) を FPGA チップ上に実装した．

各チップの接続関係を図 5.2 に示す．図のように DDMP と各 FPGA は環状に接続されており，完全な自己同期パイプラインを形成する．FPGA の内部動作速度は約 10MHz，チップ間のデータ転送レートは 1.8M packet/sec である．外付けメモリとして 7M access/sec の SRAM，および 1M access/sec の SDRAM を搭載し，排他利用が可能である．FPGA に実装した nPE は，動作性能を 32bit データ駆動型プロセッサの動作性能に対して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．

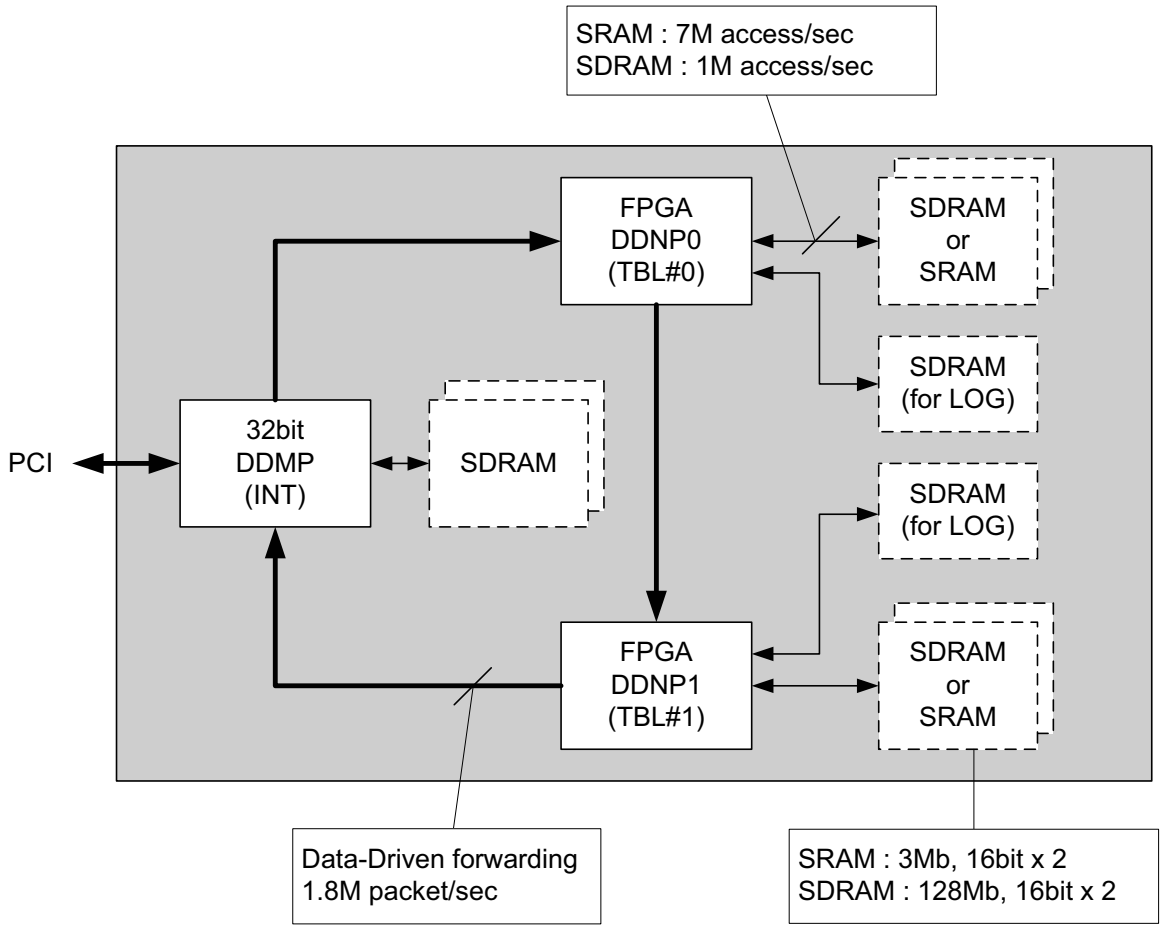

図 5.2　評価用 DDNP ボード構成.

5.2.1 32bitDDMP チップ

32bitDDMP は整数演算用 nPE およびその命令セットを保持する．本チップ上では，既存のデータ駆動プログラムを実行する．さらに，本評価ボードのリアルタイムデータ入力用の制御プログラムを実行する．これは，本評価ボードにおいてホストマシンとのデータの授受は PCI 経由で行われるが，PCI インタフェースはデータ転送速度が遅く，本評価ボード内の動作速度に対するリアルタイムデータ入力が困難なため，一時本チップに接続された，SDRAM 内に入力データを保持し，その後バースト的に読み出すことによって，チップ内動作速度等価な入力レートを実現させるために必要である．

本チップは，データ駆動パケットの入出力ポート，ブリッジとの HostI/F ポート，および SDRAM インターフェースポートを持つ．

- DDMP パケットの入出力ポートは，出力側を FPGA0 と，入力側を FPGA1 と接続し，DDMP パケットデータを 15Mpacket/sec の速度で転送する．
- ブリッジとは上り下りあわせて 30Mpacket/sec の転送速度で DDMP パケットデータを転送する．SDRAM とは 32bit 幅，133MHz の転送速度で通信を行う．
- フローグラフパケット，データパケットないし制御パケットはホスト PC から PCI，ブリッジ経由で投入する．フローグラフパケットは 32bit-DDMP 内の PS メモリに保持される．パケットデータは入力パケットとなる．
- SDRAM は 32bit 幅，133MHz の転送速度を持つインターフェースにより接続する．

5.2.2 FPGA チップ

FPGA チップは，既存データ駆動プロセッサにはない機能回路を実現するために用いられる．拡張 TBL と呼ぶ最長一致検索処理を超高速に行うための特長命令セットをサポートしたナノ PE を実装する．本評価ボードでは，FPGA 上に最長一致検索専用ナノ PE を搭載する．ただし，本 FPGA はユーザにより書き換え可能であり，上記の動作を変更することを可能とする．FPGA の書き換えは PCI を通して JTAG により行う．プログラムは FPGA

内の EEPROM に保持される．

- 内部動作速度は約 10M Hz
- チップ間のデータ転送レートは 2.5Mpacket/sec
- 外付けメモリとして SRAM(7M access/sec)，あるいは SDRAM(1M access/sec) を排他使用．

本研究において，FPGA 上に最長一致検索処理を行うための，特徴命令，およびメモリインタフェースを備えた nPE を構成した．本 nPE は，既存のメモリ参照用 nPE である TBLnPE の回路構成を流用して，新規回路を追加実装した．内部構成を図 5.3 に示す．

入力パケットは，まず既存回路内の Entry PS において，本 nPE 内の処理回路を使用するかどうかを決定する．入力パケットが，本 nPE での演算を要する場合は，追加回路に転送される．追加回路は，定数制御 (CST) を行った上で，待ち合わせ (FC) に入る．発火条件がそろうと，HSP0，および HSP1 は，最長一致検索処理を含んだメモリ参照処理を行う．このとき，チップ内メモリを参照する処理 (HSP0) か，チップ外メモリを参照する処理 (HSP1) が選択実行される．HSP1 が参照する外部メモリは，SRAM，および SDRAM があり，事前にパラメタ設定によって使用メモリが指定されてされている．演算された，パケットは次の処理を決定し (PS)，それによって，本 nPE 内を周回するか，他の nPE へ移動する．

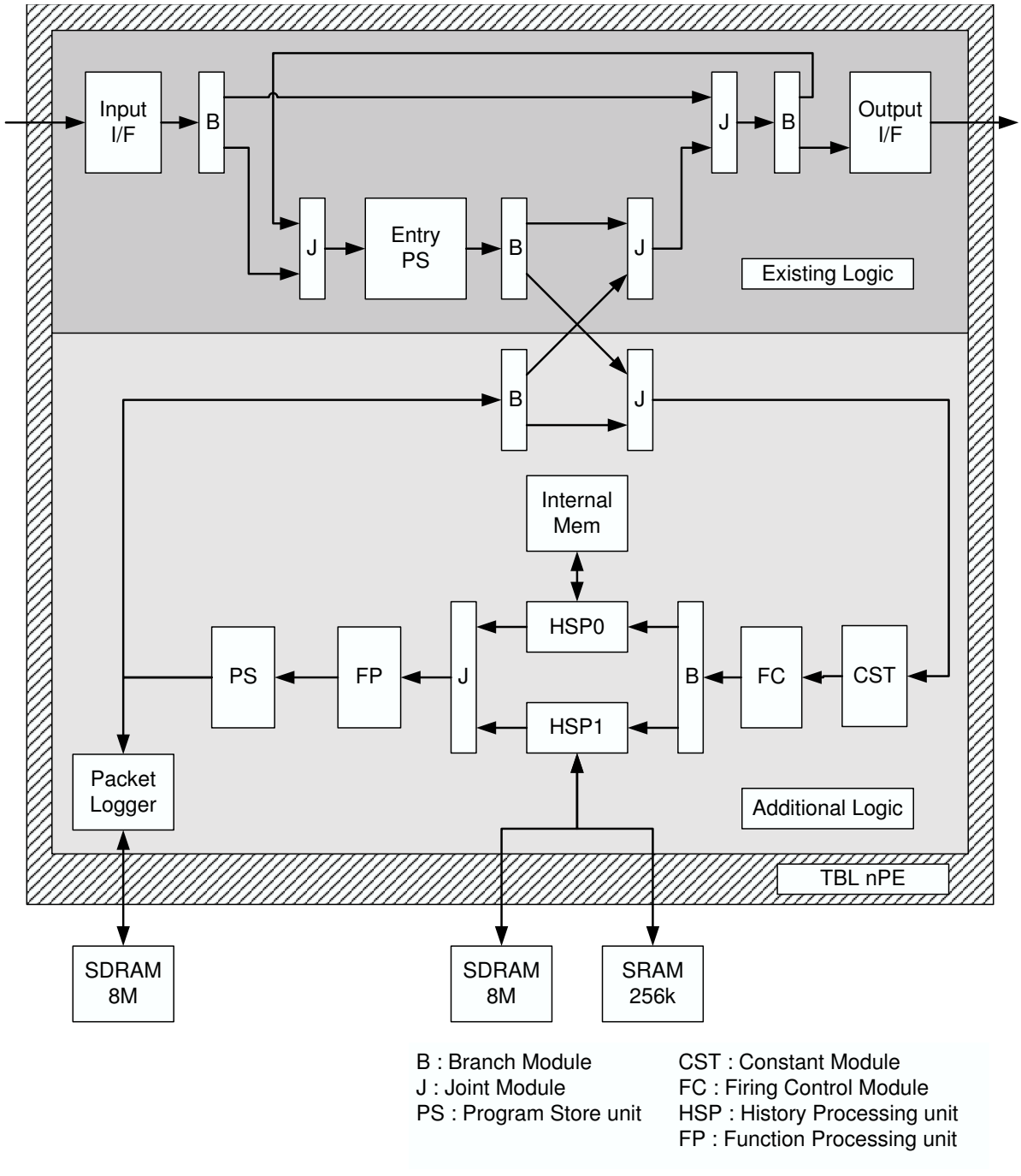

図 5.3　拡張 TBLnPE の内部構成.

5.3 結言

本研究において，自己同期型パイプラインにより構成された IP パケット向きアーキテクチャに関する各種性能評価を実施するために必要なエミュレーション機能と性能を備えた，DDNP 用評価ボード (図 5.1) を試作した．試作にあたり，本ボードにおいて自己タイミング型パイプラインハードウェア機構によりデータ駆動方式を実現したアーキテクチャ，特に，IP パケット処理向きアーキテクチャの各種性能評価を効果的に実施するために，

- データ駆動型 VLSI チップと FPGA チップ，およびチップ間の接続には，エミュレーション評価の誤差を極力少なくするために，本研究テーマの基本原理である自己同期型パイプライン機構により実現されていること
- 搭載データ駆動型 VLSI チップと FPGA 間は互換性のあるパケットフォーマットにより，入出力データの授受が可能なこと

等を必要要件とした．そして，前章で述べたパイプライン並列化パケット分類方式に必要な回路機能を FPGA 上に付加した．

本ボードを用いた性能評価，回路規模については第 6 章で述べる．

第 6 章

性能評価

6.1 緒言

本章では,

1. 最長一致検索の性能評価
2. パケット分類の性能評価
3. パケット分類用回路の回路規模

を行い，提案方式の有効性を示す.

6.2 最長一致検索の性能評価

パケット分類の性能を評価する上で，まず分類処理の中核的処理をになる最長一致検索機構について，現行 DDMP[21] と提案方式での性能比較を行う．DDMP は，現行の集積化技術を用いると仮定してシミュレーションにより求めた．このとき，プロセッサ内の処理対象データのビット長 32bit，自己タイミング型パイプラインの動作速度 140M データ/秒とした．各拡張テーブル参照用 nPE からアクセスするテーブルメモリの仕様は，内部テーブル (アクセス時間:7.1 nsec.)，外部テーブル (アクセス時間：10.1 nsec.) とした．また，現行 DDMP，提案方式ともに TBLnPE を 2 つ用いたときの性能を示す.

ルーティングテーブルが，すべて内部テーブルに格納可能な場合，メモリアクセス遅延は限りなく無視できる．しかし，MaeWest 等の大規模サイトのルーティングテーブルは巨大なため，内部テーブルにすべて格納しきれない．そこで，最も頻繁にアクセスされる

LC-Trie 検索木のレベル 1 のみを 2 つの拡張テーブル参照 nPE の内部テーブルに分割格納し，レベル 2 からレベル 5 までは外部テーブルに格納する．

このとき，IP アドレス検索時間は，内部テーブルへのアクセス遅延が外部テーブルへのアクセス遅延を越えないようにルーティングテーブルを分割配置すると，外部テーブルへのメモリアクセス時間に制約される．外部テーブルへのアクセスは LC-Trie 検索時とプレフィックス検査処理の両方で発生するため，検索レートはメモリアクセスレートに支配される．

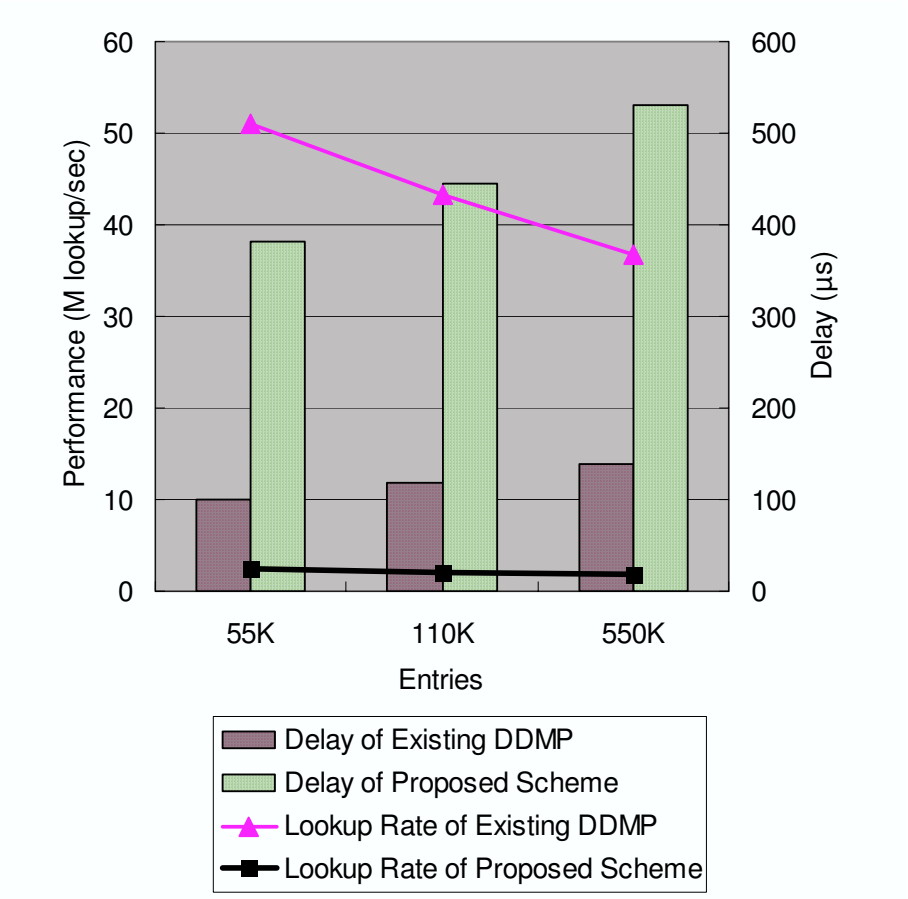

図 6.1　既存 DDMP と提案手法の検索遅延および検索レート（IPv4 アドレス検索)

まず，IPv4 アドレス検索に関して評価を行ったところ，図 6.1 のような結果が得られた．表より，検索遅延は約 5 分の 1 程度に削減されており，検索レートは 6 倍程度になっている．これより，提案した実現法は，現行の DDMP に特徴命令を追加するだけで，劇的な性能向上を達成できることが分かった．また，検索レート 51M Lookup/sec は，最短の TCP

ACK パケット (384 ビット) の場合 18Gbps，平均的な IPv4 パケット (2000 ビット) の場合 96Gbps を受理可能なことを示している．

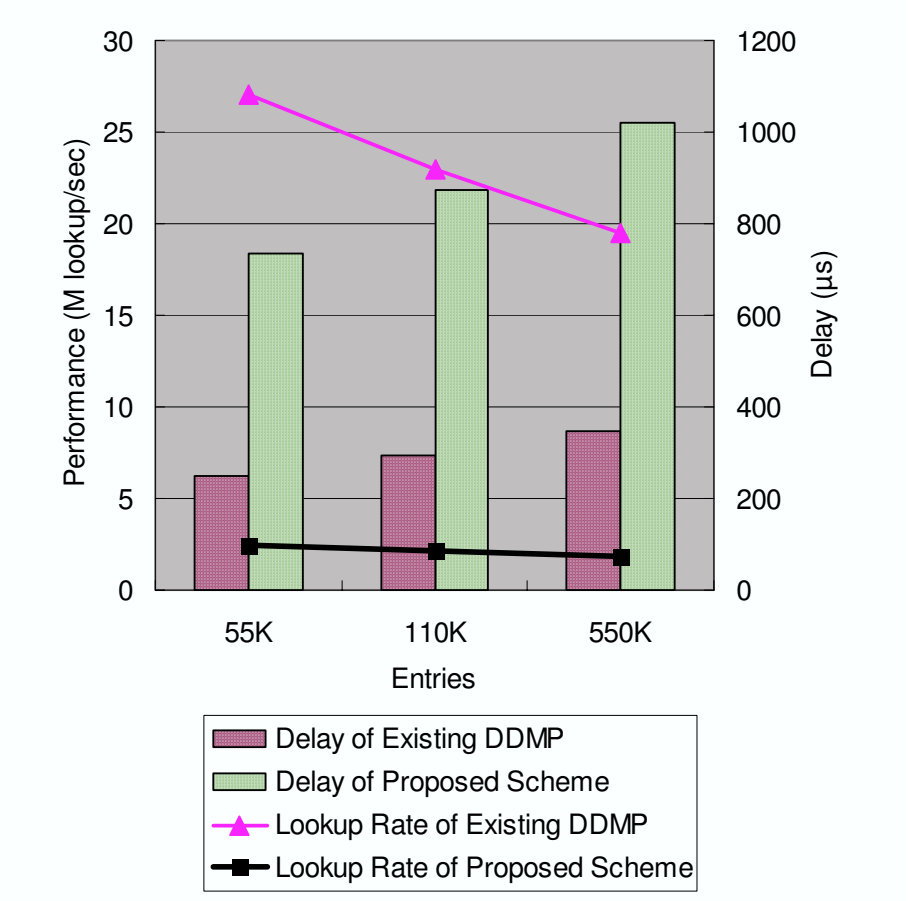

図 6.2　既存 DDMP と提案手法の検索遅延および検索レート (IPv6 アドレス検索)

同様に，IPv6 アドレス検索に関して評価を行ったところ，図 6.2 のような結果が得られた．表より，やはり検索遅延は約 5 分の 1 程度に削減されており，検索レートは 6 倍程度になっている．このとき，検索レートは 27M Lookup/sec であり，最短パケットで 10Gbps，平均長の IPv6 パケットで 54Gbps を受理可能なことを示している．

6.3　パケット分類の性能評価

次に，4 次元のルールセットを用いたパケット分類を評価した．ルールセットは MaeWest のルーティングテーブルより，擬似的に作成した．また，Index Jump テーブルの Index 生

成には，生成したルールセットの先頭 2 つのキーフィールドを宛先/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 ビットを適用する．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．

結果，図 6.3 に示すように，Index Jupm を用いることにより，分割後のルールセットのエントリ数を大幅に削減できていることが判る．これにより，検索すべき部分検索木が縮退され，検索性能の向上につながっている．分類レートは，12M Lookup/sec であり，平均パケット長 2000bit のとき約 24Gbps 程度の IPv4 パケットストリームを分類可能なことを示唆している．

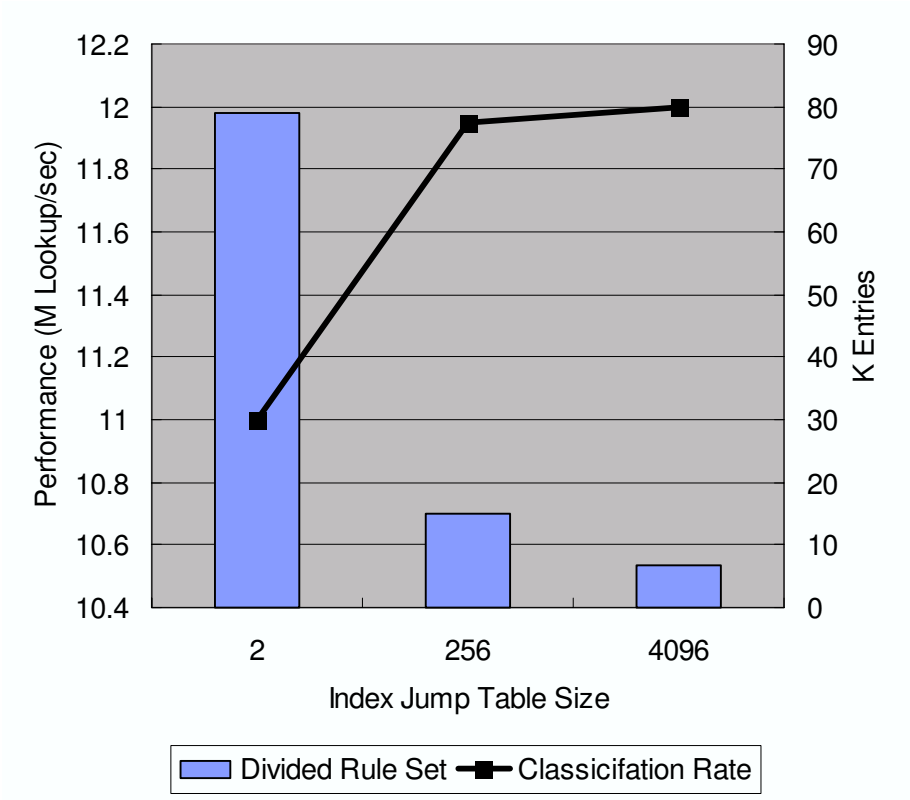

図 6.3　Index Jump テーブルの導入における分割ルールセットのサイズと分類レート

6.4　パケット分類用回路の回路規模

新たに追加した特徴命令を含む nPE に必要な回路規模を試算した．論理合成に Mentor Graphics[18] 社の Leonard Spectrum を用い，前述の 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，これを論理セル辺りの使用ゲート数を基に換算した（表 6.1）．これより，約 6%程度のゲート数増加で IPv4 用の回路拡張で構成可能であることが判る．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ことが明らかとなった．これは，1 チップ LSI として問題なく実現可能な規模である．

表 6.1　機能拡張に伴うゲート数，および増加数.

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

6.5　結言

本章では,

1. 最長一致検索の性能評価
2. パケット分類の性能評価
3. 新規回路追加による回路規模の評価

を行った.

1. では，IPv4 アドレス検索に関して，約 51M Lookup/sec の検索レートを達成し，最短の TCP ACK パケットの場合 18Gbps，平均的な IPv4 パケットの場合 96Gbps を受理可能なことを示した．そして，IPv6 アドレス検索においては，検索レートは 27M Lookup/sec

であり，最短パケットで 10Gbps，平均長の IPv6 パケットで 54Gbps を受理可能なことを示した．

2. では，Index Jump の導入により，検索空間を大幅に限定することが可能となった．また，分類性能として，平均パケットのとき約 24Gbps 程度の IPv4 パケットストリームを分類可能なことが判った．

3. では，提案回路の回路規模を試算した．IPv4 用回路は 6%程度の回路増加で構成できることが判り，IPv6 用回路はさらに約 1.4%の回路追加により拡張可能であることが明らかとなり，1 チップ LSI として問題なく実現可能な規模であることが判明した．

第 7 章

結論

通信ネットワークのパラダイムシフトにより，ネットワーク装置に求められる性能は，複雑化の一途をたどっていおり，ルーティング，ファイアウォール，NAT，QoS，帯域監視，帯域制御といったネットワーク機構が，今日のネットワーク装置にとって不可欠なものとなりつつある．こうした要求に柔軟に応えるべく，パケット処理に特化した半導体デバイス NPU が登場した．NPU に対する要求要件は，ハードウェア実現に迫る高速性を持ち，かつ，プログラマブルであることである．現在，各ネットワークベンダーはこの要件を満たすべく，しのぎを削っている．

一方で，リンク速度の向上に伴い，NPU 単独で多様なサービスに対応することが困難になってきている．このため，特に処理負荷の高いパケット分類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現方法が取られている．現在，最長一致検索エンジンとして，制御回路と CAM で構成された LSI が低価格化しつつある．これらは，一般的に検索専用エンジンとしてコプロセッサの形で提供され，簡単な機構により高速検索が可能である．一方で，フィルタ数の増加やキー長に依存した構成になるなど，今だ課題もある．すなわち，この方法では，NPU の最大の特徴であるプログラマビリティが失われてしまう．

本論文では，各種サービスに応じたパケット分類をパイプライン並列に処理可能な高速アルゴリズムを提案し，このアルゴリズムをデータ駆動型プロセッサの特徴であるパイプライン並列処理の徹底的な活用によって高速にソフトウェア実行する方式を提案した．

パケット分類におけるルールの照合法をプレフィックス照合に統一し，最長一致検索アルゴリズムを適用する．最長一致検索アルゴリズムには，メモリ参照回数を極小化し検索遅延を削減可能な LC-Trie 検索法を用いた．そして，メモリ参照をパイプライン並列に処理する

ことで検索遅延に依存せず検索レートの向上が可能であることに着目し，LC-Trie 検索法をパイプライン並列化したアルゴリズムを提案し，データ駆動型プロセッサ上に実装した．さらに，大規模なルールセットに対しては，Index Jump テーブルを導入し，検索木をヒューリスティックに分割することで, 1 検索木あたりのデータ量を縮退し，検索木の偏りを緩和した．

そして，本提案パケット分類方式を含む，DDNP アーキテクチャのハードウェアレベルでの回路規模，および性能評価を行うため試作した DDNP 用評価ボードについて述べた．本評価ボードを用いて，性能評価したところ，IPv4 パケットストリームにおいて約 24Gbps のパケット分類を実現できることが明らかとなった．また，分類機構を実現するための回路コストとして，IPv4 用最長一致検索機構は，既存回路に約 6 %程度のゲート追加で実現でき，IPv6 用最長一致検索機構はさらに高々1.4 %程度ゲートを追加するだけで拡張可能なことが判った．これより，高速なパケット分類処理機構や各種テーブル検索処理機構を達成し，1 チップ化するのに十分な回路規模で実現できていることを示した．加えて，大規模なデータを取り扱う場合は，マルチプロセッサ構成をとることで容易に対応可能である．

また，関連する研究として，パケット分類処理後の複数プロトコルが混在するパケットフロー処理，MPLS，Diffserv 等のクラスベース QoS 制御をポリシーに応じてソフトウェアにより定義し，各種サービスやフローに応じた柔軟なトラヒック管理を折り返し型自己同期パイプライン機構の導入により可能とする超高速 QoS 制御方式 [22][23] 等についても検討を進めている．

すなわち，バックボーンノードからアクセスノードまでプログラムの変更のみで，同一プロセッサで対応可能であることを示唆している．さらに，アクセスノード等の多くの負荷のかからないネットワーク環境においては，1 チップでルーティング機能を提供する SoC(System on Chip) を実現可能となり，製造コストの劇的な削減が見込まれる．

以上のことより，今後ますます多様化するであろうサービスが要求する品質に対応できる柔軟なネットワークプロセッサとして DDNP が実現できると思われる．

今後は，アプリケーションレイヤを対象としたより高次のパケット分類処理に関する研究

を進める予定である．ただし，アプリケーションレイヤにおけるパケット分類では，分類のための条件となる情報が，パケットデータペイロードに格納されているため，場合によってはペイロード全体の解析しなければならない．そのため，ペイロードを解析する処理方式の考案が必要となる．いったん情報が取り出せると，後の分類処理は，本論文で提案した方式を適応すれば実現できる．また，パケット分類機構をはじめ，関連研究である折り返し型優先キューによるスケジューリング機構，クラスベース QoS 制御機構などを含めた DDNP チップの試作を行う．

謝辞

本研究において，懇切丁寧に御指導，御鞭撻を賜った岩田 誠教授に心より感謝の意を表します.

指導教員の寺田 浩詔高知工科大学副学長には大変お忙しいにもかかわらず折りをみては暖かい声をかけていただき，また的確なご助言もいただき大変感謝いたします.

情報システム工学科長 島村 和典教授，および菊池 豊助教授には副査として有益なご助言をしていただき深く感謝します.

情報システム工学科 大森 洋一助手には DDMP プログラムについての重要な示唆をいただきました．ここに感謝の意を表します.

情報システム工学科の諸先生方には，授業および研究で熱心なご指導，ご助言をいただきました．ここに感謝の意を表します.

日本テレコム (株) 林 秀樹氏にはディスカッションなどを通じて最新の価値ある情報を提供していただくとともに研究についてご助言頂きました．ご協力に感謝いたします.

本研究で使用したデータ駆動型プロセッサシミュレータの提供，および評価ボードの試作協力をしていただいたシャープ株式会社の方々，ならびに株式会社ジーニックの関係者各位に深く感謝の意を表します.

大学院修士課程岩田研究室の橋本 正和氏，別役 宣奉氏には DDMP および Java プログラミングについて貴重なご意見を頂きました．感謝いたします.

ネットワークチームの同士として共に研究に励んだ，志摩 浩氏，岩井 秀樹氏，宮崎 康徳氏に感謝の意を表します.

日頃からご支援いただいた岩田研究室の皆さん，関係者ご一同に心からお礼を申し上げます.

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] P. Gupta and N. McKeown,“Algorithms for Packet Classification,” IEEE Network,15(2),pp.24-32,2001.

[4] V. Srinivasan, S. Suri, G. Varghese and M. Waldvogel, “Fast and Scalable Layer Four Switching,” *Proc. of ACM Sigcomm,* pp.203-14, September,1998.

[5] M.M.Buddhikot, S.Suri and M.Waldvogle, “Space Decomposition Techniques for Fast Layer-4 Switching,” Protocols for High-Speed Networks 1999, pp.25-42, 1999

[6] P. Gupta and N. McKeown, “Packet Classification on Multiple Fields,” *Proc. ACM SIGCOMM,* pp.147-60, September, 1999.

[7] V.Srinivasan, S.Suri, and G.Varghese, “Packet Classification Using Tuple Space Search,” *In SIGCOMM,* pp.135-146, 1999.

[8] F.shafai, k.J.Schultz G.F.R.Gibson, A.G.Bluschke and D.E.Somppi, “Fully Parallel 30-MHz, 2.5 Mb CAM,” *IEEE Journal of Solid-State Circuts,*vol.33, no.11, 1998

[9] T.V.Lakshman and D.Stiliadis “High-Speed Policy-based Packet Forwarding Using Efficient Multi-dimensional Range Matching,” Proc. ACM SIGCOMM ’98, pp.203-214, 1998.

[10] 高橋直久, “フィルタ弁別関数の部分計算に基づく実時間パケット分類器,” Proc. WIT’99, pp.190-197, 1999

[11] T.Y.C. Woo,“A modular Approach to Packet ’classification: Algorithms and Results,” *Proc. of IEEE INFOCOM,* vol.3, pp.1203-22, March,2000.

[12] P.Gupta and N.McKeown, "Packet Classification using Hierarchical Intelligent Cutting," IEEE Micro, vol.20 ,pp.34-41, 2000

[13] A. Feldmann and S. Muthukrishnan, "Tradeoffs for packet classification," Proc. of IEEE Infocom, vol.3, pp.1193-202, 2000.

[14] S. Nilsson and G. Larlsson,"IP-address Lookup Using LC-Tries," IEEE J. on Sel. AIC,17(6),pp.1083-1092,1999.

[15] M.Degrmark, A.Brodnik, S.Carlsson, and S.Pink, "Small Forwarding Tables for Fast Routing Lookups," *ACM SIGCOMM '97,*27, pp.3-14, 1997.

[16] H.H.Tzeng and T.Przygienda, "On Fast Address-Lookup Algorithms," *IEEE Journal On Selected Areas In Communications,* 17(6), pp.1067-1081, 1999.

[17] IPMA Project - http://www.merit.edu/ipma/

[18] Mentor Graphics - http://www.mentorg.co.jp/

[19] D. Morikawa, M. Iwata and H. Terada, "Superpipelined IP-Address Lookups on a Data-Driven Network Processor," *PDCS 2001,* pp.431-436, 2001.

[20] 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.

[21] 岡本俊弥, "データ駆動型メディアプロセッサ," 情報処理学会誌第 39 巻 3 号, pp.208-214, 1999.

[22] H. Hayashi, M. Iwata and H. Terada, "An Autonomous Class-Based QoS Control Utilizing a Self-Timed Folded Pipeline," *in Proc. of the 4th Int. Conf. on Advanced Comm. Tech.,* pp.469-474, 2002.

[23] H.Hayashi, M.Iwata, T.Hosomi and H.Terada, "A Self-Timed Pipeline Implementation of Class-Based QoS Control," *8th International Conf. on HiPC 2001 Workshop on Embedded Systems,*pp.6-10,December 2001.

[24] P. Tsuchiya, "A Search Algorithm for Table Entries with Non-contiguous Wildcarding," unpublished report, Bellcore.

[25] A.Prakash and A.Aziz, “Packet Classification Using BDDs and Pipelined SRAMs,” *Interconnection Networks Spring 2002,* , 2002.

[26] F.Baboescu and G.Varghese, “Scalable Packet Classification,” Proc. SIGCOMM ’01, pp.199-210, 2001

[27] 森川大智, 岩田誠, “データ駆動型最長一致検索機構の超高速パケット分類処理への応用,” FIT 2002, 2002.