平成 14 年度

修士学位論文

Data Driven Processing
非同期パイプライン処理を用いた
パケットアセンブリ方式の研究

A Study on Packet Assembly System Implemented by Data Driven Processor Based on Asynchronous Pipeline

1055123　　小林　寛征

指導教員　　島村　和典

2003/02/24

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

Data Driven Processing
非同期パイプライン処理を用いた
パケットアセンブリ方式の研究

小林　寛征

ネットワークの高速化に加え,peer-to-peer 技術を用いたアプリケーションの普及により最近 2 年の基幹網内を流れるトラヒック量は, 年約 3 倍の割合で増加している. それに対し, ネットワークノードのプロセッサの性能は,18 ヶ月で 2 倍の割合で向上する Moore の法則の予測が一般的である. 現状の伸び率が今後も続くと, ネットワークノードのプロセッサの処理がネットワーク高速化のボトルネックとなる.

基幹網内を流れる IP パケットのサイズは, アクセス網の MTU によって制限された 1500Bytes 以下が大半である. そこで, プロセッサに掛かる負荷を軽減するため, 宛先·送信元の IP アドレスが同じ IP パケット同士を, アクセス網と基幹網の間にあるエッジルータにおいて結合, 分解するパケットアセンブリ方式を研究している. この方式は, 従来の転送方式に IP パケットの待合せ, 結合, 分解の処理を追加するものであり, 遅延と遅延の揺らぎが増加する. 待合せ以外の遅延を最小限に抑えるため, 結合, 分解の処理を高速化する必要がある. そこで, 一般に利用されているノイマン型プロセッサと異なり, 非同期パイプラインによる自律制御に加え並列処理の記述が容易と, 高速な処理が可能なデータ駆動型プロセッサ (Data Driven Processor, 以下 DDP) を用いてこの処理を実装する研究をおこなった.

本論文では, パケットアセンブリ方式の DDP を用いた実装について検討を行う. その検討により得られた情報を元に, アセンブリ処理の高速化の観点から, テーブル参照, 比較で生じる冗長な処理に対する改善案として, 投機実行を提案する. さらに,DDP を

ネットワークプロセッサとして用いるための実装上の観点から, キャッシュの先読みが読み込みに対し先行する問題と,IP パケット全体に対する処理が困難な問題に対し, 改善案として Long Data 緩衝機能と,DDP パケット対 IP パケット命令の提案につき報告する.

キーワード　パケットアセンブリ, データ駆動プロセッサ, 非同期パイプライン, 投機実行, Long Data 緩衝機能, DDP パケット対 IP パケット命令

Abstract

A Study on Packet Assembly System Implemented by Data Driven Processor Based on Asynchronous Pipeline

Hiroyuki KOBAYASHI

The amount of backbone network traffic has grown at three times a year during the past few years, because of broadbandization of network and spreading of applications using peer-to-peer communication. On the other hand, the capability of network processors has been growing at double every 18 months, according to the Moore's Law. If these growth rate remains at the current trend, network processors would be traffic bottleneck.

Less or equal 1500 Bytes packets, restricted by Maximum Transfer Unit established every datalink, constitute the majority of backbone network traffic. So, a study on Packet Assembly System that assembles and disassembles the same destination and resource packets to one packet, is made in routers that contact point between backbone network and access network. This system decreases the processor load of backbone network node but increases delay and delay jitter due to process of packet waiting, assembly, disassembly. This system was tried to be implemented by Data Driven Processor(DDP), differ from von Neumann processor and allow easy definition of parallel processing, based on asynchronous pipeline.

This paper shows that decreases the processing time of packet assembly process that use the DDP for minimizing delay and delay jitter by packet assembly system,

and problems and remedies on incorporating DDPs into network processors. With the object of greatly accelerates the execution of assembly processing, speculative execution is proposed for reference and compare to IP address table. From the viewpoint of implementation by DDP, Long Data buffering function and DDP packet-to-IP packet operation is suggested for chache access error and not operation to IP packet.

key words Packet Assembly System, Data Driven Processor, Asynchronous pipeline processing, Speculative execution

目次

図目次

表目次

第 1 章

研究の目的

1.1 パケットアセンブリ方式にかかる処理速度の向上

パケットアセンブリ方式とは, 従来の Internet Proctocol(以下 IP) パケットの転送方式に,

- 宛先・送信元の同じ IP パケットの待合せ
- 待合せた IP パケット同士の結合·分解

の処理を追加するものである [1]. 追加される処理により遅延と遅延の揺らぎが生じる. これらは, ネットワークの品質に影響を与える [2]. データ駆動型プロセッサ (Data Driven Processor, 以下 DDP)[3] を用い,IP パケットの結合·分解における処理の高速化をはかり, 遅延と遅延の揺らぎを抑えることを目的とする.

1.2 データ駆動型プロセッサのネットワークプロセッサ化

DDP は, ノイマン型プロセッサのクロックにより処理を進める同期パイプラインと異なり, データによって自律的に処理を進める非同期パイプラインという特徴を持つ. このため, データ到達検知から処理開始までの時間が短く,IP パケットのような到達間隔に周期性の無いデータ処理に適していると考える. ただし, 現在の DDP は, マルチメディア処理用に作成されており, 周期性のある固定長データの処理に特化している. そこで, 周期性の無い可変長データである IP パケットを対象とした処理の実装で得られる情報から, ネットワークプロセッサとして用いるために必要な, 且つ, 汎用的な命令・機能を提案する.

第 2 章

研究の背景

本章では, 本研究の要となるパケットアセンブリ方式とデータ駆動型プロセッサについて説明する.

2.1 パケットアセンブリ方式

パケットアセンブリ方式とは, 現在の IP パケット転送方式では行われていない経路上での IP パケットの結合·分解を行うものである. これにより, 基幹網内のネットワークプロセッサに掛かる負荷を軽減する ([4, 5]).

2.1.1 ネットワーク利用帯域の増加率とコンピュータ性能向上率

ネットワークトラヒックの利用帯域は,Odlyzko らの 1996-2000 年におけるアメリカ-スイス間のトラヒック調査 (図 2.1) の分析によれば,12 ヶ月で 2 倍の増加率である [6]. また,JPIX の Daily Traffic の推移 (図 2.2)[7] を見ると, 特に 2000-2002 年の最近 2 年は,peer-to-peer を用いたアプリケーションの普及などの原因で,12 ヶ月で約 3 倍と急速な増加を示している. 今後も, 光ケーブルを用いたネットワーク速度の向上や, 映像配信技術の発展·普及のほか,IP 電話などの様々なサービスが IP ネットワークを用いて実現されることによって, 利用帯域はさらに増加すると考えられる. これに対し, コンピュータの処理性能は,Moore の法則から 18 ヶ月で 2 倍の速度で進歩すると予測される. しかし, 半導体製造プロセスの微細化の技術的な問題から, いつまで処理性能が Moore の法則にしたがって向上するかわからない. このまま性能向上率が推移したとしても, 増加率の違いから, 将来, ネットワークノードのプロ

セッサがボトルネックとなることは避けられない.

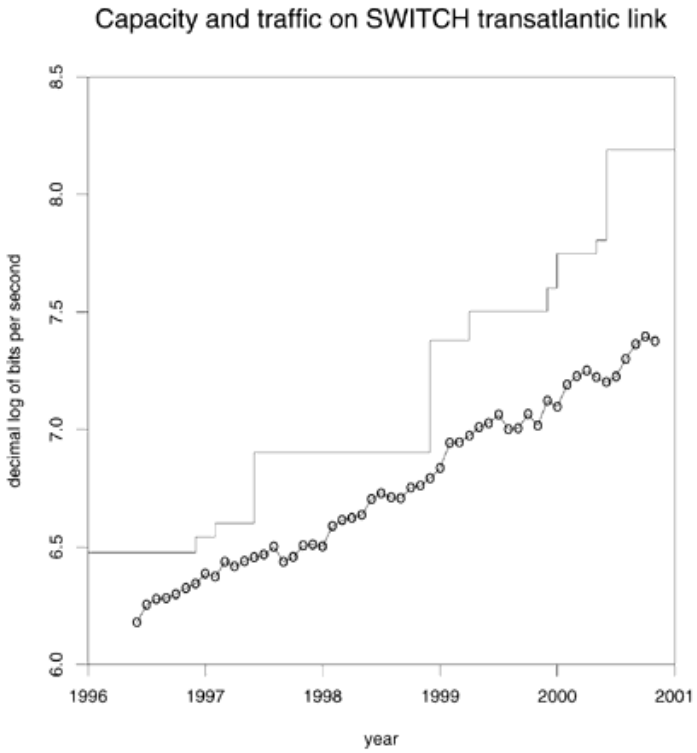

図 2.1　アメリカ-スイス間のトラヒック推移

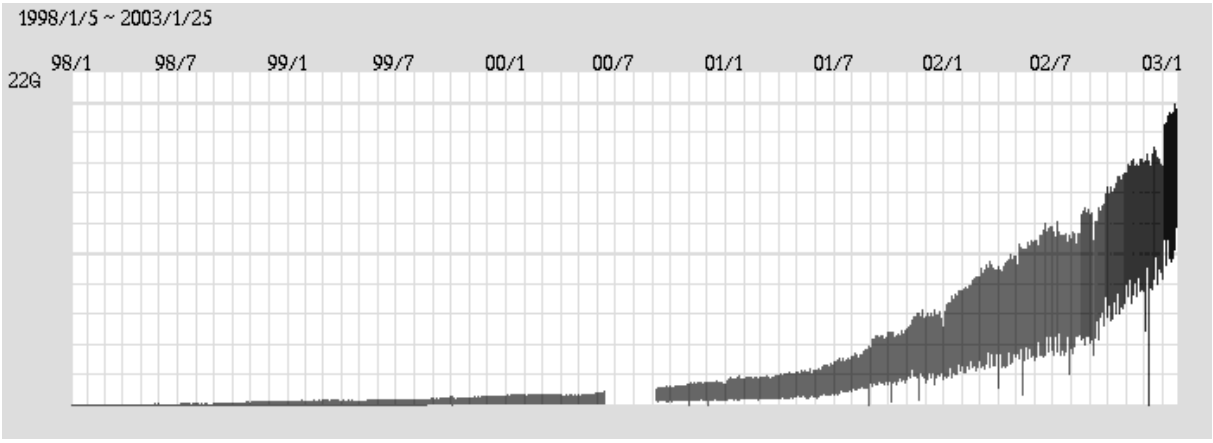

図 2.2　JPIX のトラヒック推移

2.1.2　トラヒック中の小サイズパケット

基幹網内を流れる IP パケットのサイズ分布を, 図 2.3 に示す [8]. これらは,TCP の Ack パケットや経路情報の伝達の他に,Maximam Transfer Unit(最大転送単位, 以下 MTU) と経路 MTU 探索によるものが大きい.

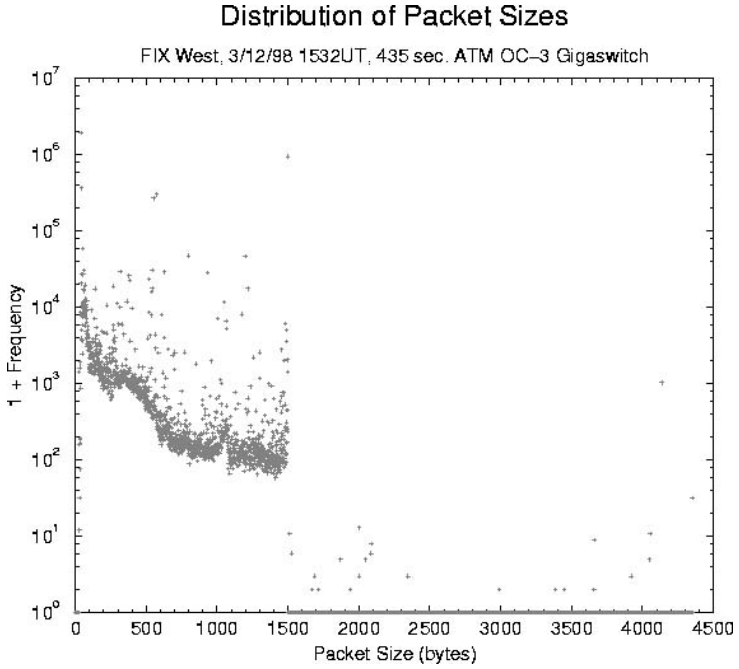

図 2.3　基幹網内の IP パケットサイズ分布

Maximum Transfer Unit(最大転送単位)

アプリケーションが, あるサイズのデータを相手先のコンピュータに送信する場合, そのデータは,IP パケットに格納されて伝送される. この IP パケットのサイズの最大値は, 送信端末が接続されているデータリンクに定められた MTU によって決まる (表 A.1). これは, パケットロスによってデータ転送効率が低下するのを抑えるためで, 転送速度とパケットロスの割合からデータリンクの種類毎に決まっている. データのサイズ>MTU の時,IP は, データを IP パケットに分割 (フラグメント)·格納して伝送する. 一度送信した IP パケットは,MTU の異なるデータリンクを通る場合でも, 他の分割された IP パケットが同一経路上

を転送される保証が無いため, 再構築 (デフラグメント) は相手先のコンピュータのみで行う. 経路上の MTU に合わせた再分割は行なわれる. このため, 基幹網内に流れる IP パケットは, アクセス網内の MTU 以下のパケットが多い.

経路 MTU 探索

IP 層より上位のトランスポート層では, パケットロスによってデータを全て再送するという効率の悪さを考慮し, 送信時に伝送経路上の最小 MTU を探索し, データ自体を探索結果より得られた値に分割 (セグメント) して,IP 層でフラグメントしない方法をとっている [9].

小サイズパケットが与える影響

上記の理由によって, 基幹網内を流れる IP パケットのサイズは,1500 サイズ以下が大半を占める. これらの小サイズパケットが基幹網に与える影響として, 次の点が考えられる.

- ネットワークプロセッサの負荷増加

  ノードのプロセッサ負荷は, ノードが転送する IP パケットのパケット数の多さ (Packets Per Second(以下,pps) 値) に比例する. 小サイズパケットが多い場合, 当然 pps 値も高くなる. ネットワーク利用帯域の増加で, 小サイズパケットも増加し, ネットワークプロセッサの負荷も増加する.
- 基幹網内のスループット低下

  小サイズパケットが多いと, パケット間のプリアンブルや管理情報のため, 全転送量に対するデータの割合は小さくなる. このため, 基幹網内のスループットが低下する.

2.1.3　パケットアセンブリ

3 点の問題, すなわち

- ネットワークプロセッサの性能向上率,

- ネットワークプロセッサへの負荷増加,
- 基幹網内のスループット低下

の問題を解決するため, アクセス網と基幹網の MTU の差が大きいことに注目し, パケットアセンブリ方式を研究している [1].MTU の異なるアクセス網と基幹網の間にあるエッジルータで, 宛先と送信元の同じ IP パケット同士を待合せ, 結合 (アセンブリ (assembly))·分解 (ディスアセンブリ (disassembly)) する方式を図 2.4 に示す.

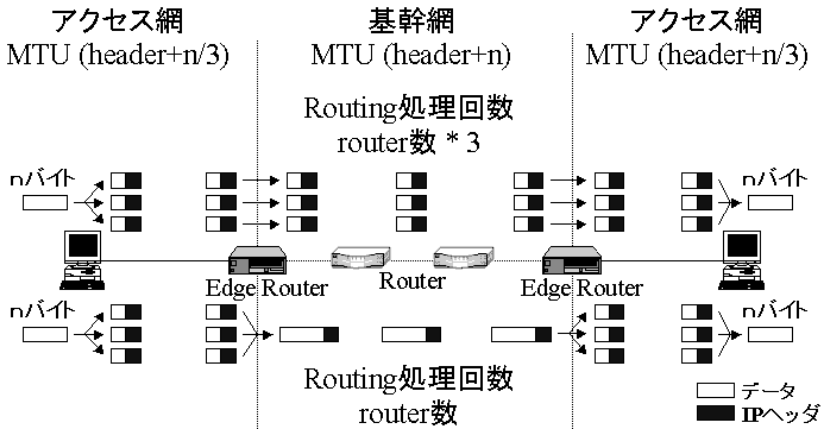

図 2.4　通常の転送方式 (上) とパケットアセンブリ方式 (下)

パケットアセンブリ方式のメリット

- 基幹網内のプロセッサ負荷の軽減

  複数のパケットを一つに結合することにより, 基幹網内のパケット数は減少する. これにより, ルーティングに掛かるネットワークプロセッサへの負荷は軽減できる.

- 基幹網内のスループット向上

  一度のルーティング処理で複数の小サイズパケットを送信できるため, プリアンブルをはさむ必要が無く, スループットの向上が期待できる. また,IP パケットの結合方式の中には, 共通するヘッダ情報を削除して結合する方式がある. 削除できるヘッダ情報は,8Bytes と少ない. しかし, 小サイズパケットの結合では, ヘッダ長とデータ長の割合

から 8Bytes の占める比率は大きい.

パケットアセンブリ方式のデメリット

- 遅延の増加

  通常の転送方式に比べ, 宛先·送信元の同じ IP アドレスを待合せる時間と, 結合·分解する処理時間分の遅延が増加する. また, 待合せをやめるまでのタイムアウト時間分の遅延が増加する.

- 遅延のばらつきの増加

  IP パケットを結合する個数やタイムアウトなど, 個々の IP パケットが, 結合を行うエッジルータから出力されるタイミングが異なる. このため, 遅延のばらつきが生じ, そのばらつきの差も大きくなる.

- ネットワークフローのスループットの低下

  基幹網内のスループットは増加する. しかし, ネットワークフローのスループットは, 遅延の増加で低下する.

パケットアセンブリ方式の種類

パケットアセンブリ方式として, いくつかの種類を検討している.IP パケットを結合する条件は,2 種類ある.

- シングルフローアセンブリ

  宛先·送信元の IP アドレスが一致するパケット同士を結合する.

- マルチフローアセンブリ

  宛先·送信元の IP prefix が一致するパケット同士を結合する.

結合された IP パケットの結合方式は,4 種類ある.

- IP ヘッダカットアセンブリ (図 2.5(a))

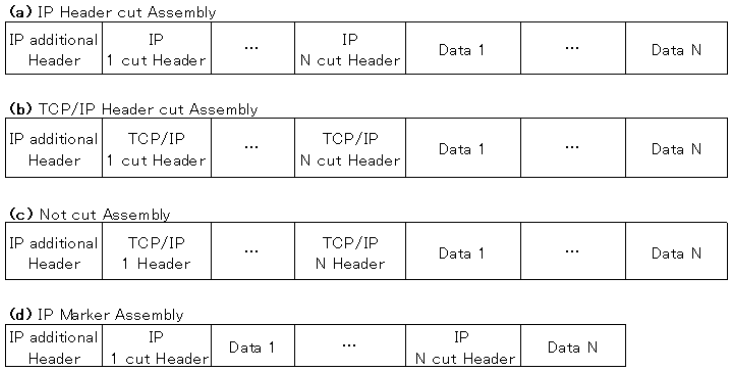

図 2.5　アセンブリパケットフォーマットの種類

  - IP ヘッダ内の共通するフィールドを削除し,
  - IP ヘッダと IP データを分けて結合する.
- TCP/IP ヘッダカットアセンブリ (図 2.5(b))
  - IP ヘッダと TCP ヘッダ内の共通するフィールドを削除し,
  - TCP/IP ヘッダと TCP/IP データを分けて結合する.
- Not カットアセンブリ (図 2.5(d))
  - IP ヘッダ内の共通するフィールドを削除せず,
  - IP ヘッダと IP データを分けて結合する.
- IP マーカーアセンブリ (図 2.5(e))
  - IP ヘッダ内の共通するフィールドを削除し,
  - IP ヘッダと IP データを一緒に結合する.

  IP ヘッダが, 結合されたデータの切れ目 (Marker) となる.

2.2　データ駆動型プロセッサ

DDP は, ノイマン型プロセッサと異なる特徴を持つ. その特徴と仕組みについて本研究と関係のある部分について説明する.

2.2.1　ノイマン型プロセッサとの比較

DDP とノイマン型プロセッサとの動作の違いを説明する.

パイプライン

両プロセッサのパイプライン構成と処理開始のタイミングの違いを図 2.6 に示す. ノイマン型プロセッサは, クロックという同期信号によってデータの有無に係わらず, 後続段 (successive stage) の Data Latch にデータを送る同期パイプラインである. このため, 図 2.6 上の演算回路　,　,　の処理開始は, クロックに制御される. これに対し,DDP は,C 素子と呼ばれる回路で後続段の Data Latch にデータがあるか否かを確認し, データが無ければ Data Latch へデータを送り, 有れば空くまで待機する. データによって駆動する非同期パイプラインである. これにより, 演算回路の処理時間に関係なく図 2.6 下のように処理開始のタイミングを早めることができ, 処理を高速化することができる. また, 同期パイプラインは, 全ての回路に絶えず電気信号を流しており, 消費電力や発熱問題が生じる. 非同期パイプラインは, データが存在しない場合や, 先の Data Latch にデータがある場合はデータを送らないため, 不要な電力を消費しないで済む.

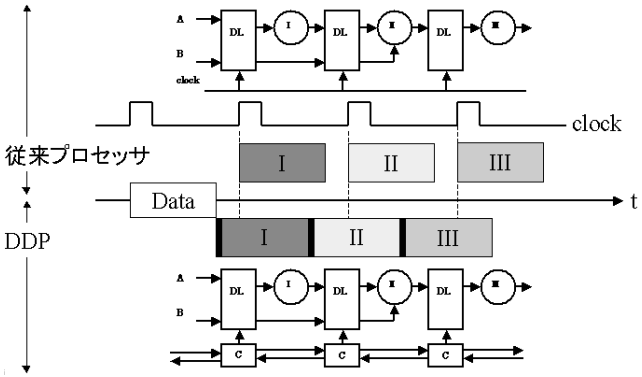

図 2.6　クロック同期パイプライン (上) と非同期パイプライン (下)

2.2.2　データ駆動プロセッサの仕組み

DDP 動作原理の仕組みについて, 本研究と関係のある部分を説明する.

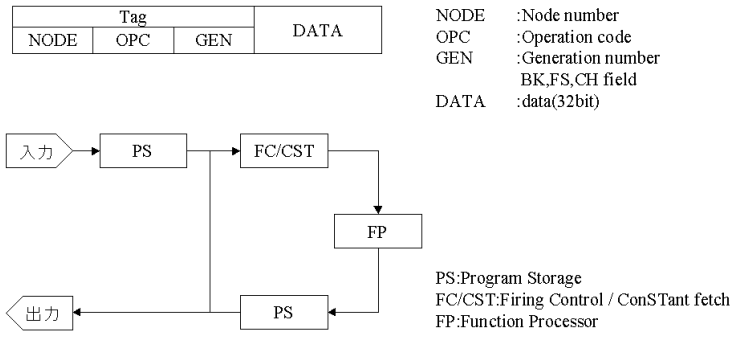

図 2.7　DDP の DDP パケットフォーマット (上) とナノプロセッサブロック図 (下)

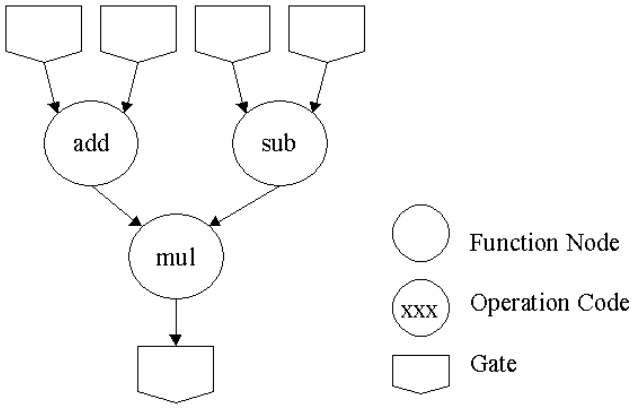

図 2.8　DDP で動作するプログラム (フローグラフ)

DDP パケット

DDP パケットフォーマットについて, 以後の説明に必要なフィールドだけを図 2.7(上) に示す. また,DDP のプログラムは, データフローグラフと呼ばれ, その例を図 2.8 に示す. DDP は, データを 32bits に分割し,DDP パケットという単位で処理する.DDP パケットには,DDP パケットを管理するためのタグ情報が存在する. 本研究で用いるタグ情報は,

- NODE ノード番号

OPC を実行するノードを識別するために割当てられる番号である. ノードとは, 図 2.8 の Function Node を表す.

- OPC 命令コード

  32bit のデータに対する演算命令を表し, 図 2.8 の Operation Code を意味する.

- 世代番号, BK(2), FS(8), CH(12)

  () 内は, 評価システムで用いた bit 長を示している.

  DDP 内部での DDP パケットの整合性を保証する. ノードにおいて, ノード番号と世代番号と一致する DDP パケット同士が処理される.

  本研究では,FS は IP パケットの分類に,CH は 32bit に分割された IP パケットのどのデータであるかを表すオフセットとして用いる. さらに,FS は,IP アドレスマッチングテーブルのテーブル番号としても利用する.

待合せ処理

DDP では, ノードで処理対象となる 2 つの DDP パケットを待合せ, 命令を実行する. この処理を実現する構造について説明する.

DDP の最小構成単位は, ナノプロセッサである.DDP は, 複数のナノプロセッサで構成される. 評価システムの DDP には, ナノプロセッサが 10 個搭載されている. そのナノプロセッサの構成は, 既に図 2.7(下) に示した. 各ブロックの役割は次の通りである.

- PS (Program Strage)

  入力された DDP パケットのノード番号から, 次の処理を行うノード番号と命令コードを DDP パケットに格納する.

- FC/CST (Firing Control/ConSTant fetch)

  DDP パケットの待合せをノード番号と世代番号を用いて行う. ノードで命令を実行する対象となる右項と左項の DDP パケットのうち, 先に届いたものが FC/CST のメモリに格納される. 次に届いた DDP パケットのノード番号と世代番号が先に格納されたも

のと一致する場合, 二つのデータを 1 つの DDP パケットに格納に Function Processor ブロックへ渡す. 一致しない場合は, 一致する DDP パケットが届くまでメモリに格納される.

どちらかが定数の場合は, 一致した場合と同様に定数とデータを 1 つの DDP に格納し FP へ渡す.

- FP (Function Processor)

  命令コードに従って, 二つのデータに対し命令を実行し, 実行結果を DDP パケットに格納して PS へ送る.

2.2.3　並列処理

DDP は, ノード番号と世代番号によって処理の整合性を確保している. このため, 世代番号を強制的に変更しない限り,DDP パケットを混同した処理は行われない. この特徴から,DDP は並列処理が可能であり, フローグラフの記述も世代番号の重複を起こさないよう配慮するだけで良く, 処理記述が容易である.

2.2.4　ネットワークプロセッサとしての有用性

DDP は, パイプラインと演算に関して, 全てデータによって処理が駆動するデータ駆動型プロセッサである. このため, データ処理が決定している特性を持つ処理に適している. また, 非同期パイプラインと並列処理によって高速に処理できることが期待される. これは, ネットワークプロセッサとして適していると考えられる.

第 3 章

DDP への実装

3.1　実装するパケットアセンブリ方式

パケットアセンブリ方式について,2.1.3 項に示した 4 種類の検討を行っている (p.7). 本研究では, シングルフローアセンブリ,Marker Assembly フォーマットを採用した. 採用理由は, 他の 3 つのフォーマットと比べ,

- IP パケットをヘッダ部とデータ部を分けてバッファに格納する必要が無く, バッファへのアクセス頻度を抑えることができ, また, リソースの節約が可能,
- 出力時にヘッダ部の後ろにデータ部を付ける処理が不要で, さらに, 分解時に IP ヘッダから IP データの位置を特定する計算が不要で処理が簡易,

という点である. 各フォーマットの処理特性の比較を表 3.1 に示す. パケットアセンブリ方式の検討に際し, 分解・結合処理の高速化を目的としており, 簡易な処理である Marker Assembly が最適と考えた. そのフォーマットを図 3.1 に示す.

表 3.1　図 2.5 の各フォーマットの特性

フォーマット	処理内容	バッファアクセス頻度	リソース占有度
(a)	複雑	多	大
(b)	複雑	多	大
(c)	複雑	多	大
(d)	簡易	少	小

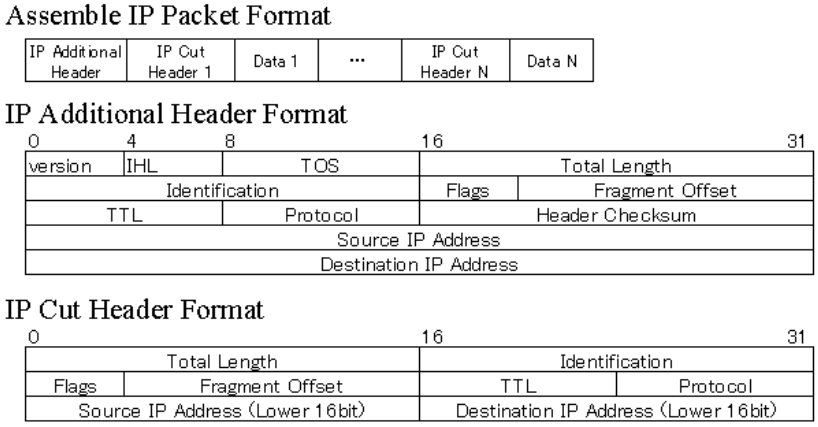

図 3.1　Marker Assembly のフォーマット

3.2　本実装装置の役割と位置付け

本研究の実装に用いた評価システムのハードウェア構成を, 図 3.2 に示す. 評価システムに搭載された DDP は,DDMP(Data Driven Media Processor) と呼ばれ, マルチメディア処理に特化された DDP である.

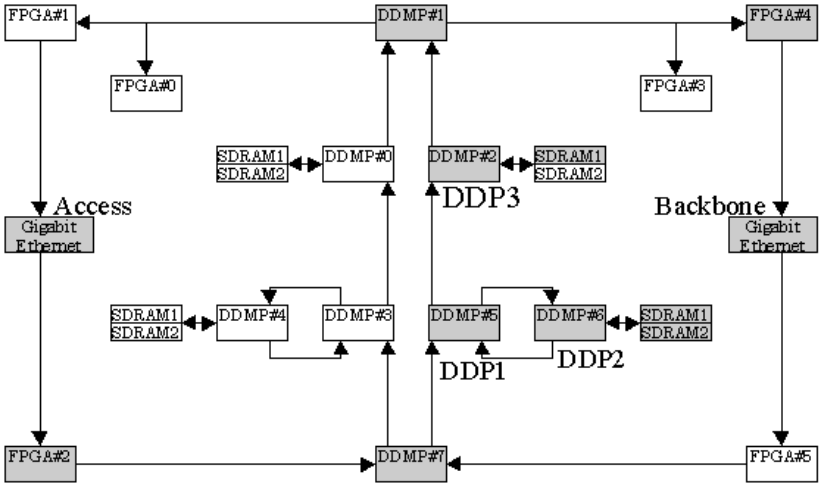

図 3.2　評価システムのハードウェア構成

結合処理には,

- 投入された IP パケットを一時的に格納するためのバッファと,
- アセンブリした IP パケットを格納するバッファと,

- 出力する IP パケットを格納するバッファが,

必要となる. さらに, 出力の際シリアル出力する必要があり, その順序を決定する情報を

- アセンブリ対象から外れた IP パケットと,
- アセンブリした IP パケットで,

共有する必要がある. また, 評価システムの構成上,

- 偶数の DDMP にのみ,SDRAM が 2 つ用意されており,
- 図 3.2 中の矢印の方向にのみ DDP パケットを送信可能,

という制約がある. 以上の理由から, 各 DDMP に対し処理を図 3.3 の灰色の部分に, 表 3.2 に示すように割当て, 実装を行った. また, 本文では, 説明する上で各 DDP の役割と処理内容を明確にするため, ハードウェア構成と各 DDP に割当てた処理を図 3.3 に示した表現を用いる.DDP の各要素の呼び方が異なるのみで, 各プロセッサの処理や機能に変更は無い.

表 3.2　評価システムの各 DDMP に割当てた処理

図 3.2 の名称	**対応する図 3.3 の名称と役割**
Gigabit Ethernet, DDMP#7	Gigabit Ethernet(Access Network)
DDMP#5	DDP 1 (IP Address Matching)
DDMP#6	DDP 2 (Assembly)
SDRAM1	SDRAM (Input Buffer)
SDRAM2	SDRAM (Assembly Buffer)
DDMP#2	DDP 3 (Output)
SDRAM1	SDRAM (Output Buffer)
DDMP#1, Gigabit Ethernet	Gigabit Ethernet(Backbone Network)

アクセス網側と基幹網側のデータリンクが Gigabit Ethernet であり,MTU の異なるデータリンクではない. 本研究は, 小サイズパケットの結合·分解の処理速度短縮を目的とし,DDP

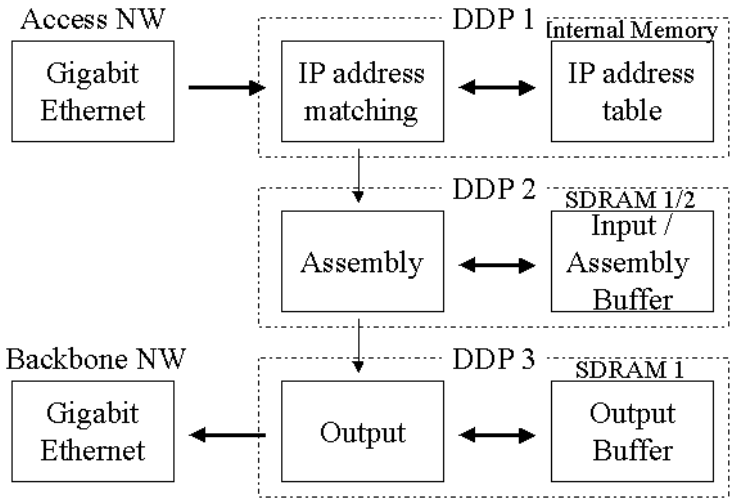

図 3.3　本文で説明に用いるハードウェア名と処理構成

での実装と効果の検証を行う. また,DDP のネットワークプロセッサとして利用するにあたり, 現段階の問題の抽出も目的としている.DDP に実装したパケットアセンブリ方式が, 実際に利用できるよう改善を進めたときに,ATM といった基幹網のデータリンクとして一般的なものを適用する. ただし, 現装置で,Gigabit Ethernet と同等の処理能力が得られた場合には,IP のフラグメント機能を用いて, 装置と基幹網側のエッジルータ間に,IP デフラグメント装置を入れて基幹網側の MTU に対応させ, 研究を行っていく.

3.3　アセンブリ処理

各 DDP で行っているアセンブリ処理について説明する.

3.3.1　実装に用いた評価システムの処理能力

評価システムの DDP は,140MHz 相当で動作している. ノード 1 つの処理 (一命令のパイプライン) にかかる時間は, 命令の種類に依存する. 命令ごとの所要時間を表 3.3 に示す. ただし, 本論文では, ノードの所要時間を 294nsec で統一して計算している.

Gigabit Ethernet から DDP へ IP パケットを構成する DDP パケットの投入間隔

表 3.3　Function Node にかかる所要時間

命令コードの種類	所要時間 (nsec)
算術演算	294
論理演算	294
シフト	294
条件分岐	294
DDP パケット複製	294
内部メモリアクセス	301
内部メモリアクセス+演算	301
キャッシュアクセス	280
キャッシュアクセス+演算	280
SDRAM の処理を含めたキャッシュアクセス	287

は,1Gbit を 1 秒間に 32bit 単位で投入するため,33MHz となっている. 約 30nsec 間隔である. これより,Ethernet 最大フレーム長 1518Bytes を投入し終えるまでの時間は,((1518/4)*30)/294 = 38.72... 約 39 段となる.

3.3.2　DDP 1 - IP アドレスマッチング

IP パケットの識別に,DDP パケットの世代番号の FS フィールドを用いる. この値は,Gigabit Ethernet から DDP へ投入される際, 自動的に付与される. これを FS-A とする. 対して,IP アドレステーブルとバッファを構成する SDRAM のメモリアドレスの算出にも,FS フィールドを用いる. 今回の実装では,SDRAM を 32 個のテーブルに分割し,FS 番号 (以下,FS-B) で識別する仕様とした. アセンブリバッファに格納する際, 自動的に付与された FS-A を, 処理に都合の良い FS-B に変換する. この FS-A と FS-B の関係を図 3.4 に示す.

本実装における IP アドレスマッチングは, 宛先 IP アドレス下位 5bit を FS-B の偶数連番に合わせるため 1bit 左シフトし,FS-B と対応させる.FS-B に対応した SDRAM にアセン

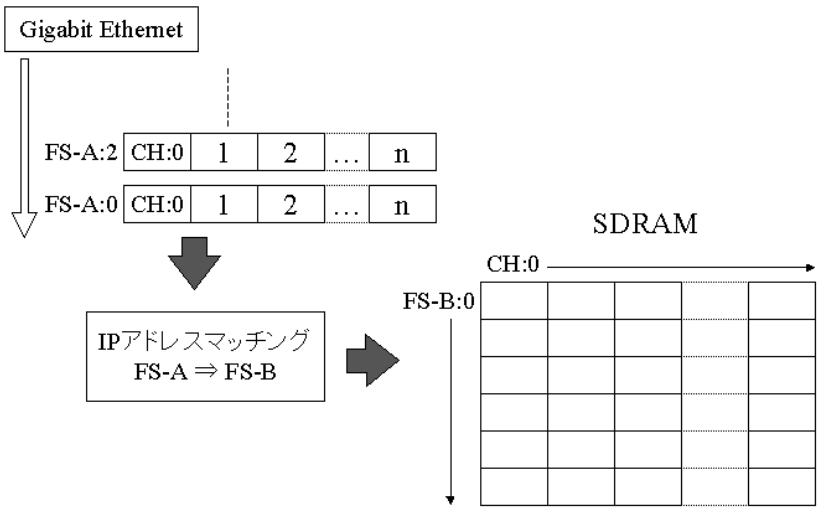

図 3.4　世代番号による入力 DDP パケットと SDRAM のメモリアドレスの関係

ブリ対象として格納されている IP パケットの宛先·送信元 IP アドレスと比較し, 一致していればアセンブリ, していなければ出力する. また,IP パケットが格納されていなければ, アセンブリ対象として投入された IP パケットを SDRAM へ格納し,IP アドレスの組をテーブルに保存する.

この処理で, 投入された IP パケットの処理が確定し, その結果を IP アドレステーブルに反映するまで, 後続の投入された IP パケットを並列して処理できない. なぜなら,IP アドレス比較から反映までの間に, 宛先 IP アドレス下位 5bit が同じ IP パケットが同様の処理を行うと, まだ IP パケットが格納されてない場合, テーブルの重複利用が起きる.

本実装では, これを避けるため,IP アドレステーブル参照から, 処理確定,IP アドレステーブルへ結果を反映するまでの間, 後続の処理をロックし, 逐次処理を行う.

DDP 1 のフローグラフと最大処理経路

DDP 1 の全処理を図 3.5 のフローグラフに示す. また, 図 3.5 中の IP アドレスマッチングモジュールの内容を図 3.6 に示す.DDP 1 における処理の最大命令段数は,DDMP#7 に存在する Gigabit Ethernet から IP ヘッダ部分を構成する DDP パケットを分岐する 1 段と, 図

3.5 中の灰色のノードを経由する 3 段, 図 3.6 中の灰色のノードを経由する 16 段の合計 20 段となる. また, このうち, 逐次処理すべきクリティカルパス部分は 15 段である.

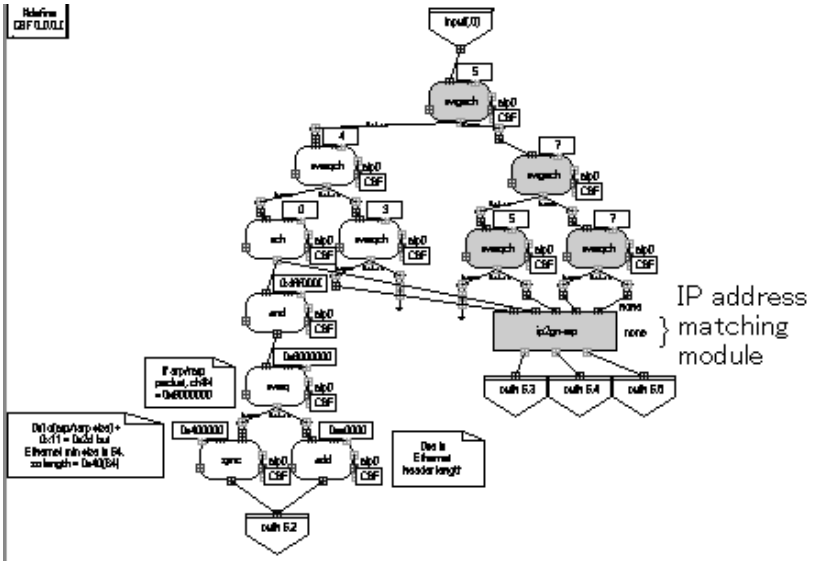

図 3.5　DDP 1 全体のフローグラフ

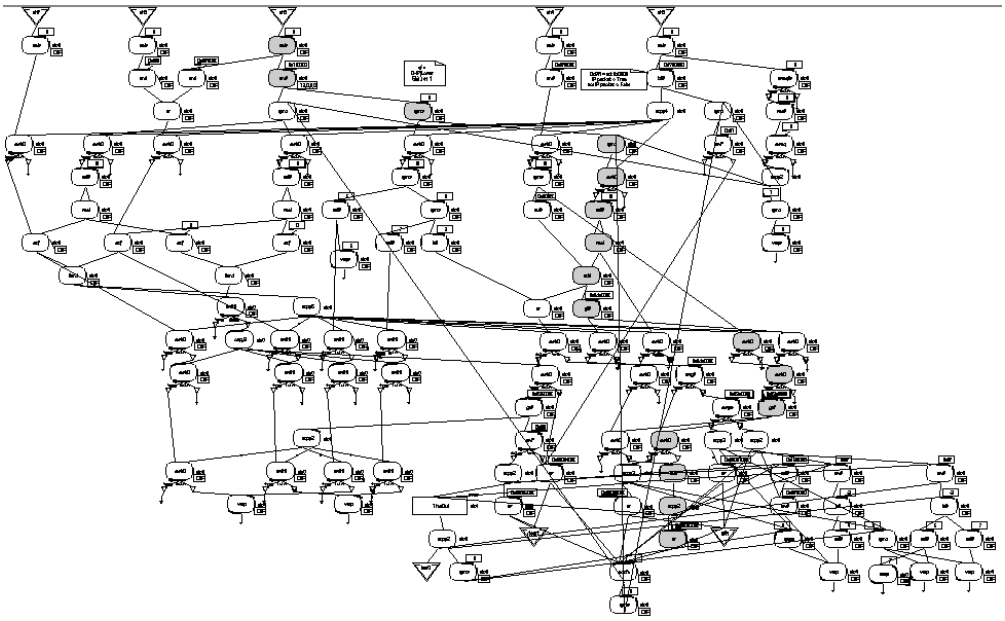

図 3.6　IP アドレスマッチングモジュールのフローグラフ

3.3.3　DDP 2-アセンブリ

DDP 2 は,DDP 1 の結果を受けた処理を行う. 処理の内容は, 下記となる.

- IP アドレスマッチングで一致した場合

1-(a) アセンブリ処理を行い,IP パケットの出力は行わない.

1-(b) アセンブリ処理の結果, それ以上 IP パケットをアセンブリできないサイズの場合, アセンブリ後の IP パケットを出力のため DDP 3 へ渡す.

1-(c) アセンブリ処理を行うと MTU を越えてしまう場合, アセンブリバッファに格納されている IP パケットを出力のため DDP 3 へ渡す (前出力とする). その後, 投入された IP パケットを先頭パケットとしてアセンブリバッファに格納する.

1-(d) アセンブリ処理を行うと MTU を越えてしまう場合, 前出力を行う. さらに, 投入された IP パケット単体のサイズが MTU に近く, アセンブリできない場合, 出力のため DDP3 へ渡す.

- IP アドレステーブルが空きだった場合

2-(a) IP パケットが他の IP パケットをアセンブリできるサイズである場合, 先頭パケットとしてアセンブリバッファに格納する.

2-(b) 単一パケットで既にサイズが MTU に近く, それ以上 IP パケットをアセンブリできない場合, 出力のため DDP 3 へ渡す.

- 一致しなかった場合,

3-(a) アセンブリを行わず出力するため DDP 3 へ渡す.

アセンブリ処理は,IP パケットを構成する全 DDP パケットの世代番号 FS-A を,IP アドレスマッチングより得られた世代番号 FS-B に変更しなければならない. 本実装では, 世代番号 FS-B に変更を行うノードの命令コードを直接変更することによって実現している. このため, 全 DDP パケットがこの命令を通過するまで, 他の IP パケットを構成する DDP パケットの処理を並列して行うことができない.

アセンブリ処理を並列に処理した場合, 現在の処理している IP パケットを構成する DDP パケットにアセンブリ処理行っている最中に, 他の IP パケットが, この IP パケットを前出力する処理を発行すると, キャッシュを介したアセンブリバッファへのアクセスに失敗し, エラーが生じてしまう. このため, アセンブリ処理自体も逐次処理にしなければならない. 逐次処理にかかる時間は, アセンブリ処理の種類によって異なり, その値は, 図 3.7 となる.DDP 2 における最大命令段数は 122 段であり, このときの逐次処理に必要な命令段数は,122 段+2 段 (次の処理を開始するトリガーを生成) となる.

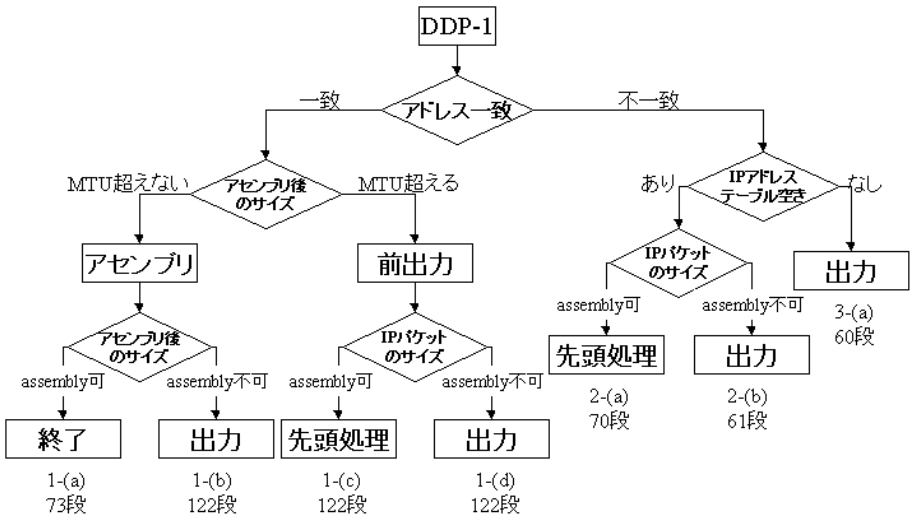

図 3.7　DDP 2 の処理プロセスと最大命令段数

DDP2 のフローグラフと最大処理経路

DDP2 の全体の処理を図 3.8 のフローグラフに示す. また, 最大命令段数が 122 段となる処理のうち, 図 3.7 の 1-(b) の処理経路を図 3.8, 図 3.9, 図 3.10 の灰色のノードで示す. 命令コードを変更されたノードは, 逐次処理の原因となるノードである. 逐次処理の最大値を求めるため,IP パケット分の DDP パケットを生成するノードは,Ethernet フレーム最大長を生成するために必要な 11385nsec(約 39 段) と, その命令自身の 1 段の合計 40 段として計算する.

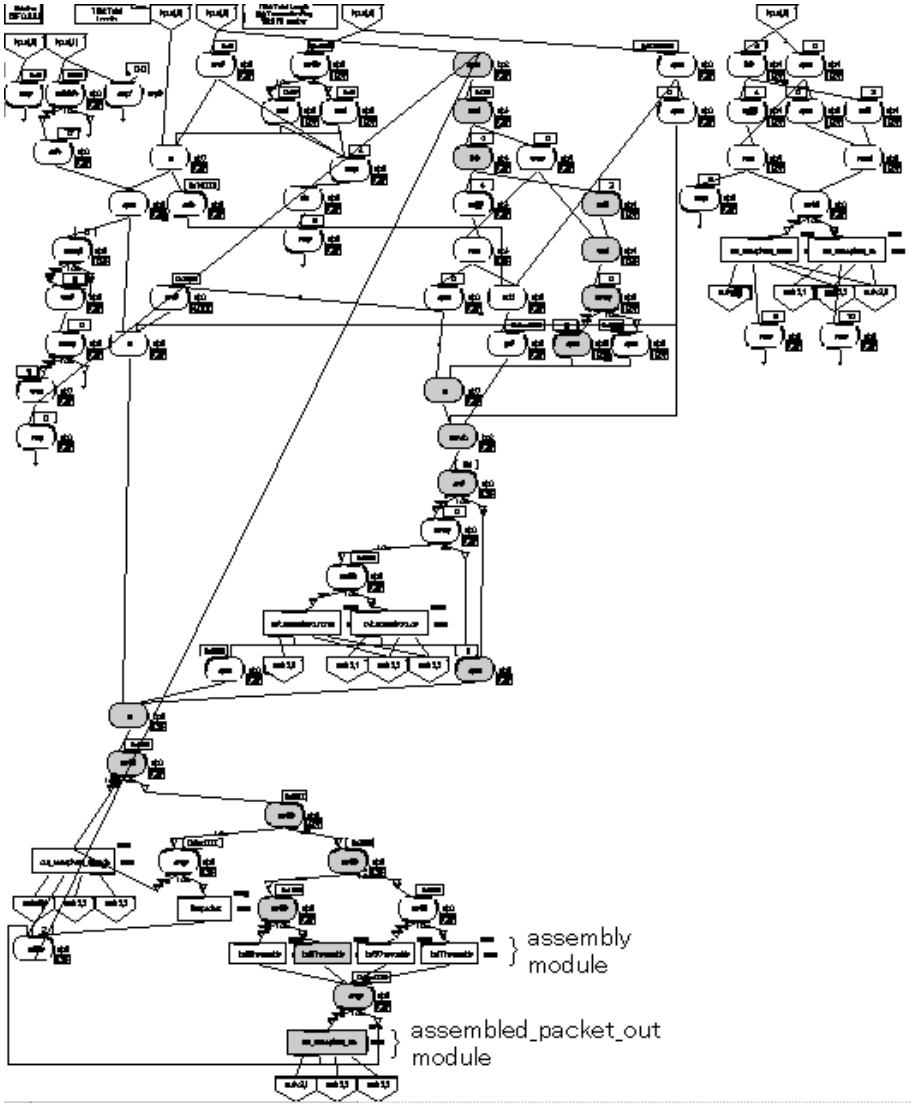

図 3.8　DDP 2 の全体のフローグラフと最大処理経路

3.3.4　DDP 3-出力

DDP 3 は,DDP 2 より送られてくる IP パケットの IP ヘッダチェックサムを再計算し,IP パケット毎にシリアライズして Gigabit Ethernet に送信する処理を行う. これは,Ethernet フレームを正しく送信するために必要となる.

DDP 3 の最大命令段数は 70 段で, このうち, シリアライズして送信する逐次処理となるのは,45 段である.

3.3 アセンブリ処理

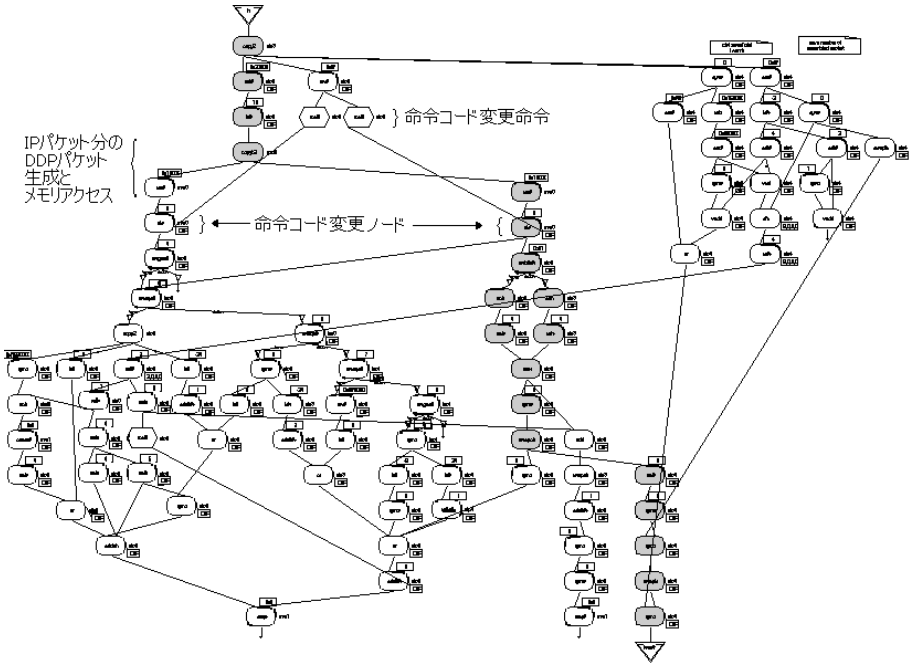

図 3.9 アセンブリ処理のフローグラフ

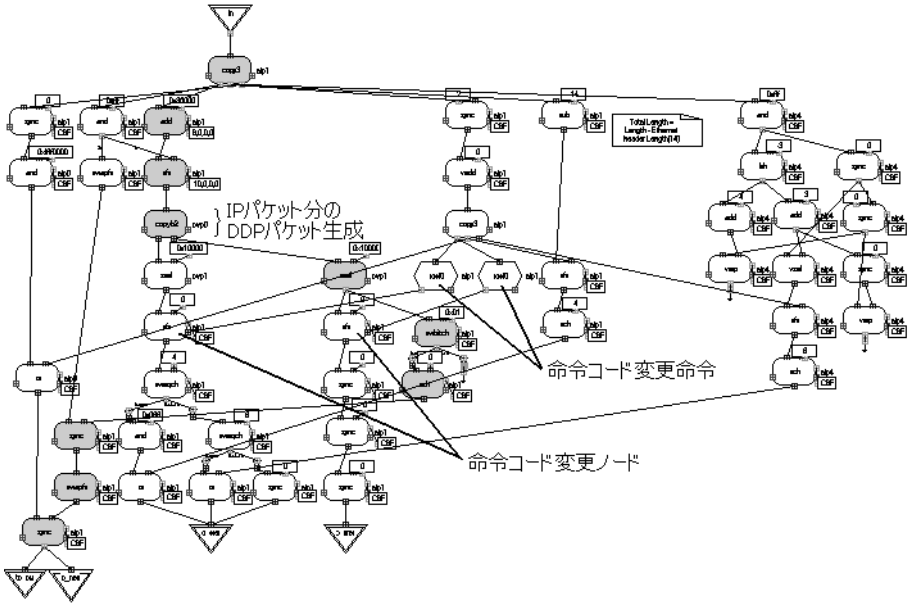

図 3.10 アセンブリされた IP パケットの出力用処理のフローグラフ

DDP 3 のフローグラフと最大処理経路

DDP 3 のフローグラフと, 最大命令段数となるアセンブリされた IP パケットの出力の経路を図 3.11 に示す.

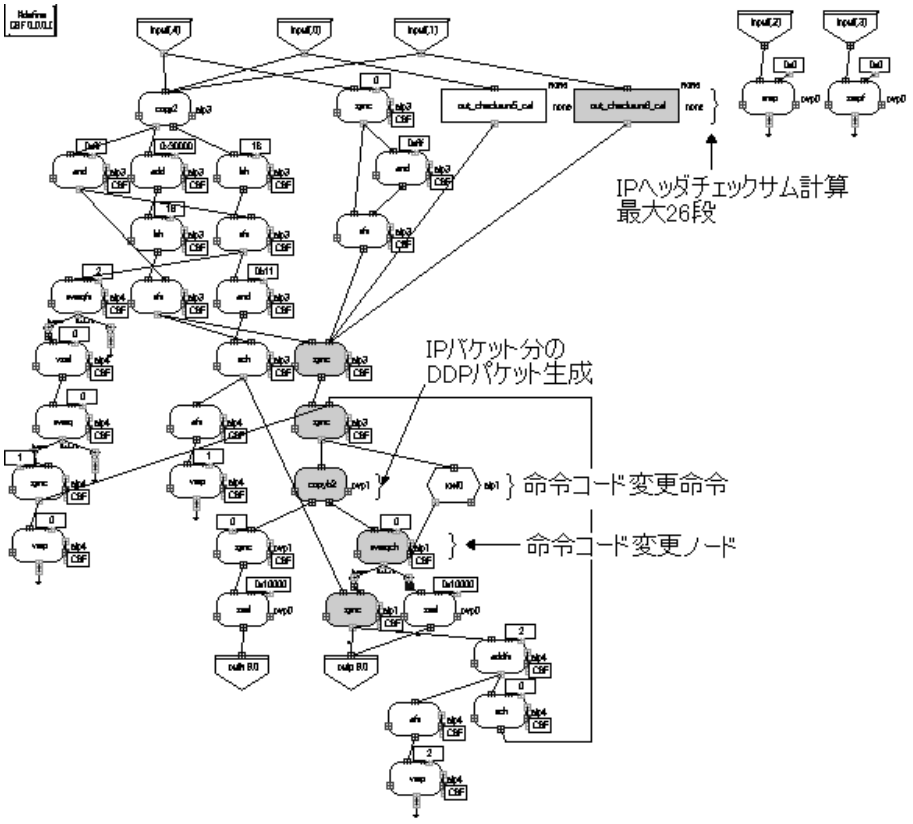

図 3.11　DDP 3 の全体のフローグラフと最大処理経路

3.3.5　本実装の処理速度 (理論値)

各 DDP における所要時間は, 表 3.4 となる. 本実装は, 各 DDP の処理間に依存関係無く, また,DDP 内では並列処理が可能である. このため,DDP の処理速度は, 逐次処理の最大時間となる.DDP2 の 124 段の処理が最大の逐次処理となり, 本実装の処理速度は,124*294 = 36456nsec で 1 パケットを処理できることになる. この 124 段は,Ethernet の MTU の IP パケットを処理した場合を想定している. よって,1 秒間に処理できる IP パケット数は,27430pps(Packets Per Second) となる.

表 3.4　各 DDP における所要時間 (294nsec/段)

DDP i	**最大命令段数**	**逐次処理の最大命令段数**
DDP 1	20	15
DDP 2	122	124
DDP 3	70	45

2002 年 1 月 24 日 09:00 から 2002 年 2 月 31 日 09:00 までの高知工科大学からインターネットへのゲートウェイのトラヒック集計結果は,IP パケット総数-170,971,037 パケット, 総 Bytes 数-39GBytes であった [2].IP パケットの平均サイズは,228Bytes となる. これを元に, 本実装装置を介したネットワークの転送処理能力は,50.032Mbps となる.

アクセス網と基幹網へは Gigabit Ethernet 1000Mbps で接続されており, 本実装装置では実用に耐えない. そこで, 逐次処理にかかる命令段数を減少できる方法について検討しなければならない.

第 4 章

問題点の抽出と改善案

実装したパケットアセンブリ方式の処理速度向上のため, 次の改善案を提案する.

4.1 投機実行

IP アドレスマッチング処理は, アドレステーブル参照, 比較の繰り返しという逐次処理である. 本実装では, アドレステーブルと IP アドレス下位 5bit が 1 対 1 で対応しているため, 参照, 比較の処理は 1 度で済んでいる. しかし, 実用化する場合, テーブル数を増やし, アセンブリ対象 IP アドレス数を増やす必要がある. このとき, テーブル参照·比較にかかる処理時間が問題となる.

4.1.1 投機実行の必要性

DDP に投入された 1 つの宛先 IP アドレスに対し, 利用できるテーブル数を U と仮定する. 投入された IP アドレスとテーブルの持つ IP アドレス情報を比較するため,DDP で逐次処理を行うとすると, 図 4.1(a) となる. 命令段数は, 最大 2U 段となり, 非効率である.

DDP の並列処理の特徴を生かし,n 個のテーブルを同時に処理した場合は, 図 4.1(b) となる. しかし,n 個の比較で一致するものがあるか, または, 一致せず次の n 個の比較を行う必要があるかを判断するために, 比較結果を統合する比較結果同士の論理和演算 (図 4.1 の OR) が $log_2 n$ 段必要になる. この命令段数は, 最大 $(4 + log_2 n)$U/n 段となる. 処理速度を向上させるには,n の値を増やすことである. しかし,n を増やした分,$log_2 n$ 段の非効率な命令が必要となる.

IP アドレスマッチング処理は, マッチング結果をテーブルに反映するまで, 次の IP パケットに対する処理を停止する. 処理速度を向上させるために,$log_2 n$ 段の命令を抑える必要がある.

図 4.1 の R,C,AS は次の処理を表す.

R　　: IP アドレステーブル内の IP アドレスを参照,

C　　: DDP に投入された IP パケットの IP アドレスと比較,

AS　: 比較結果をアセンブリ処理へ出力.

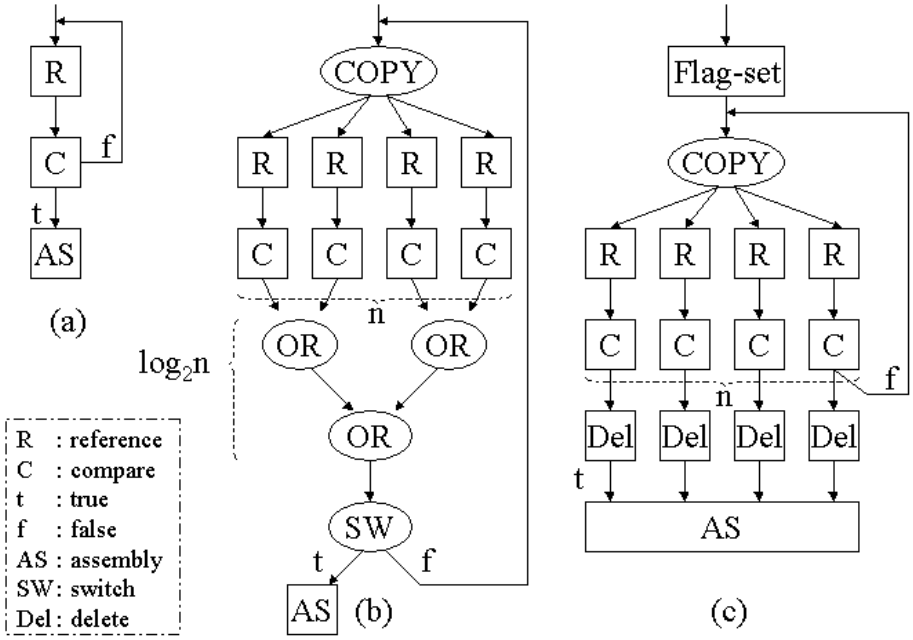

図 4.1　IP アドレステーブル参照·比較処理

4.1.2　投機実行の仕組み

投機実行を, 図 4.1(c) に示す.n 個のテーブル参照·比較のうち, 一致するものが見つかった時点で, 他の並行して処理されている n-1+1(次の n 個を比較する命令, 図 4.1(c) の f にあたる) 個のパイプラインを破棄するものである [10].

1. DDP パケットのフラグセットを行う命令 (Flag-set) で投機実行 flag をセットする.

2. セットされた投機実行 flag は,n-1+1 個に分岐した DDP パケットに継承される.
3. 分岐した処理うち,1 つのテーブル比較で一致した場合,delete 命令が実行される.
   一致しなかった場合は, 自身のパイプラインを破棄する.
4. delete 命令が実行されると, 他の n 個のパイプラインを破棄する.
5. 全て一致しない場合は,False を経由する DDP パケットのみが残り, 次の n 個の比較命令を開始する.

投機実機を実装した場合の参照, 比較の結果判断には,Flag-set と delete の 2 命令のみとなる. よって, 最大命令段数は 2+3U/n 段となる.

delete 命令の仕組み

図 4.2 を元に,delete 命令の仕組みについて説明する. 現 DDP に追加することは,

- DDP パケットに delete フラグ (reset,flag, 各 1bit) を追加し,
- ナノプロセッサに,Delete Packet ブロック (DP) を規定すること

である.delete 命令の動作原理は,

1. DP が delete 命令を持つ DDP パケットを受信すると
   （a）delete 命令パケットの reset を 1 にセットして送信する.
   （b）以降,reset=1 の DDP パケットを DP が再び受信するまで,flag=1 の DDP パケットを破棄する.
   （c）FC メモリにある待機状態の DDP パケットのうち,flag=1 のものを全て削除する.
2. delete 命令パケットが, 再度 DP に到着するまでの間に,DP - FC/CST - FP - PS - DP 間にある flag=1 の DDP パケットは,DP で全て破棄される.
3. delete 命令は PS で次の OPC を得るが,reset=1/flag=1 は保持している. このため,DP は,reset=1/flag=1 で,delete パケットが 1 周したことを検出して,reset,flag をクリアした後,DP は DDP パケットを通過させる通常状態に戻る.

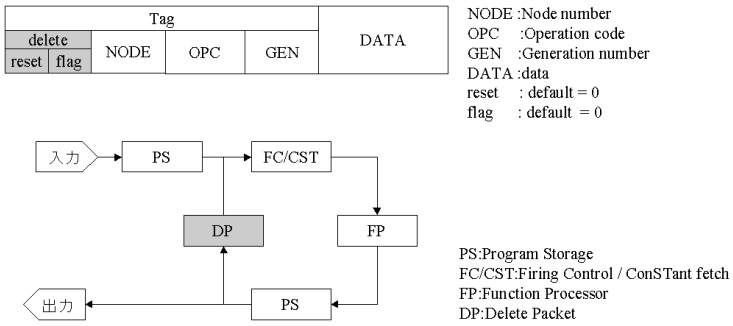

図 4.2　投機実行時の DDP パケットフォーマット (上) とナノプロセッサブロック図 (下)

投機実行の DP 状態遷移図

図 4.3 は, 投機実行における DP の状態遷移図である. 状態数は 2 つで規定する. 通常状態は,DP に入る DDP パケットを FC/CST へ通過させる状態を示し, 削除状態は,delete 命令によって DP で flag=1 の DDP パケットを破棄する状態を表す.

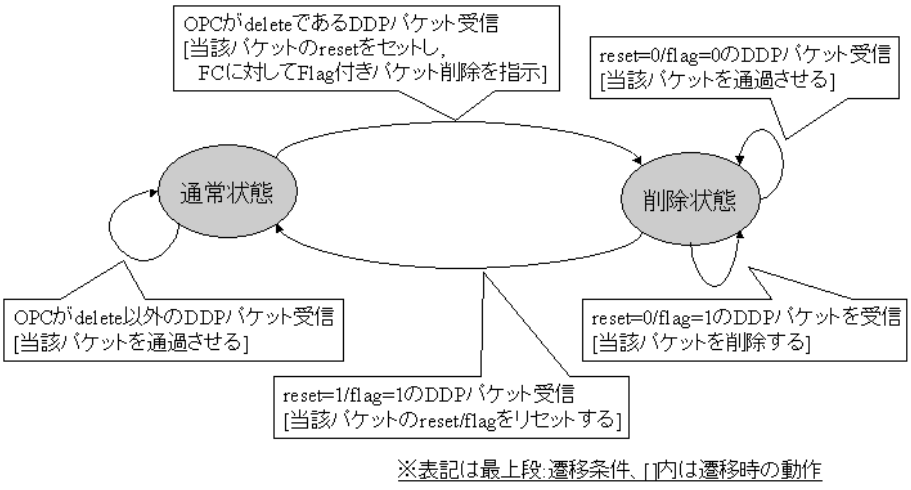

図 4.3　投機実行の DP 状態遷移図

4.1.3　処理速度比較

図 4.1 に示した 3 種類のテーブル参照･比較に掛かる命令段数を, 並列処理する n の値で比較した関係を図 4.4 に示す. 投機実行を導入した場合は, そうでない場合と比べ, 並列比較数

に拘らず約 4 割の段数で処理できる.

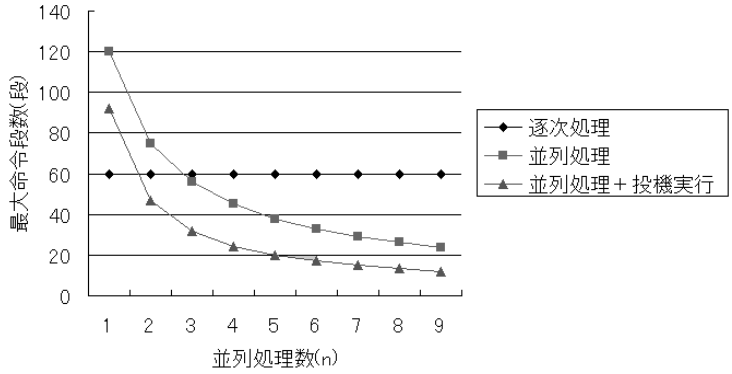

図 4.4　総テーブル数 U=30 の投機実行処理オーダー

4.1.4　利用例

本実装装置の IP アドレスマッチングの予備検討として,IP アドレス下位 5bit に IP アドレステーブルを 2 つ用意した実装の検討を行った. このときのフローグラフは, 図 4.5 である. この 2 つのテーブルに対する参照, 比較と, このテーブルに空きがあるかどうかを参照, 比較 (IP アドレスが格納されていないかどうか) する処理に対して, 投機実行を実装したフローグラフを, 図 4.6 に示す. 実装時の命令段数を, 表 4.1 に示す.

表 4.1　投機実行未実装時と実装時の処理段数比較

	一致時の命令段数	**不一致時の命令段数**
投機実行未実装時	18-19	27
投機実行実装時	19	22

一致時に関しては,Flag-set 命令が入る分, 同等もしくは 1 命令遅くなる. 不一致時は,5 命令分早く結果が出力できる. また, 投機実行を実装した場合,IP アドレスマッチング処理に必要となるノード数を 33 個削減することができ, リソースを開放することができる.

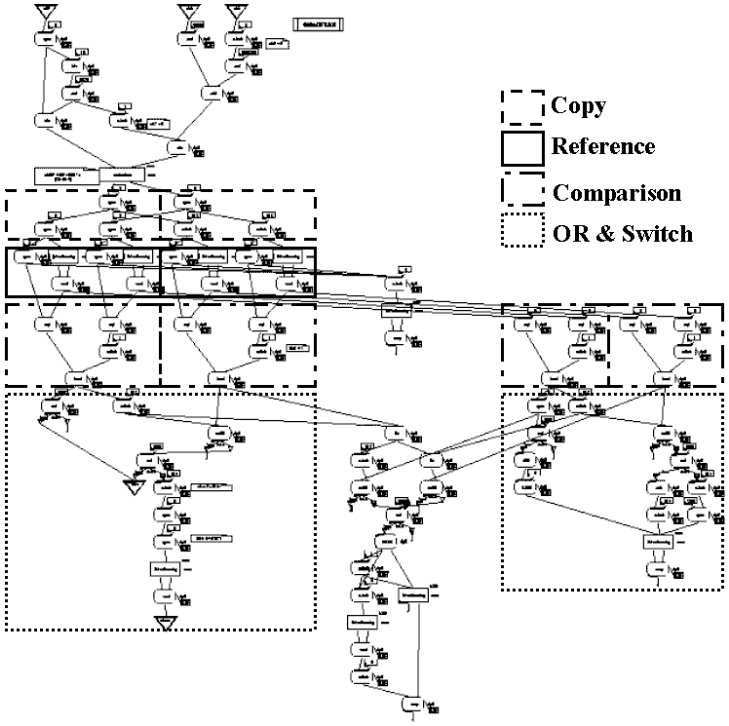

図 4.5　2 テーブルを割り当てた IP アドレスマッチングフローグラフ

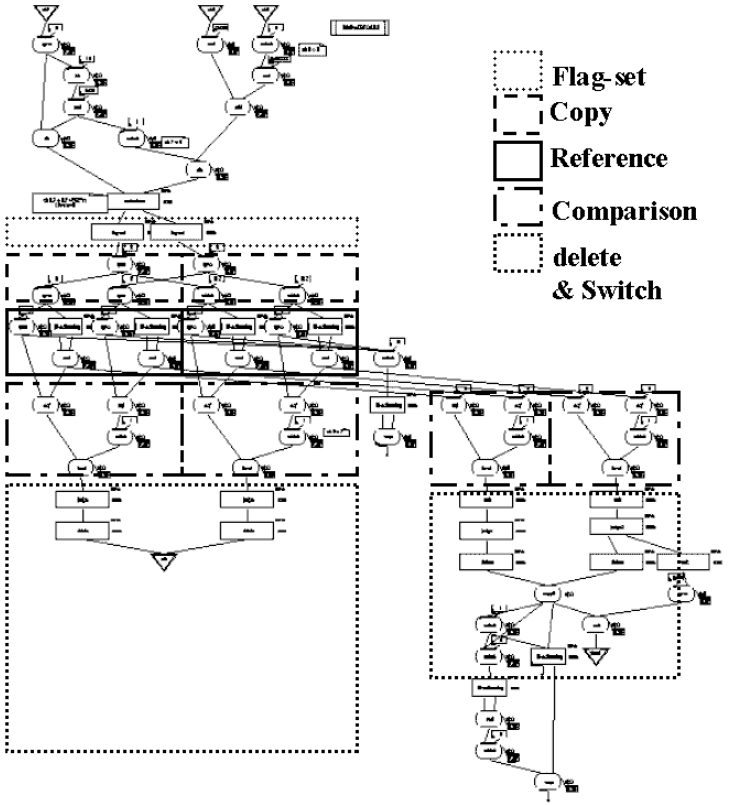

図 4.6　2 テーブルを割り当て, 投機実行を実装した IP アドレスマッチングフローグラフ

4.2 Long Data 緩衝機能

Gigabit Ethernet から入力される IP パケットは,32bits ごとに DDP パケットに格納され,33MHz(約 30nsec) 間隔で投入される. これに対し, 現在の DDP は,140MHz 相当の速度で動作している. 一命令は, 約 294nsec で動作している. この速度差から, 処理の実行できない IP パケット分の DDP パケットでリソースを消費することを避けるため, 処理が確定するまで,IP パケットを SDRAM に格納している. 処理が決定次第, 同じ 33MHz 間隔で参照される.

評価システムは,SDRAM の特性を生かすため, キャッシュを介してアクセスを行う. キャッシュと SDRAM の関係を図 4.7 に示す. キャッシュは,64Bytes を 1Page とする 256Bytes,4Pages で構成される. キャッシュから SDRAM へのデータ書き込みは, キャッシュ 1Page 分のデータが更新されると発生する (図 4.7(左)).SDRAM からキャッシュへの読込みは, キャッシュ 1Page の参照が終わると,4Page 先の 1Page のデータを SDRAM より先読みする (図 4.7(右)). よって,IP パケット長 (Ethernet フレーム長) が, 先読みの発生する 4Page(256Bytes) を超える場合,IP パケットは, 参照データから 4Page 以上先のデータを投入し終えていなければならない.4Page(256Bytes) を投入する時間として 1920nsec(命令数にして 6.5 命令分) である. もし,IP パケット参照開始までの時間が 1920nsec 未満の場合,SDRAM への先読みが書き込みを追い越し, メモリアクセスに失敗し, 誤動作してしまう.

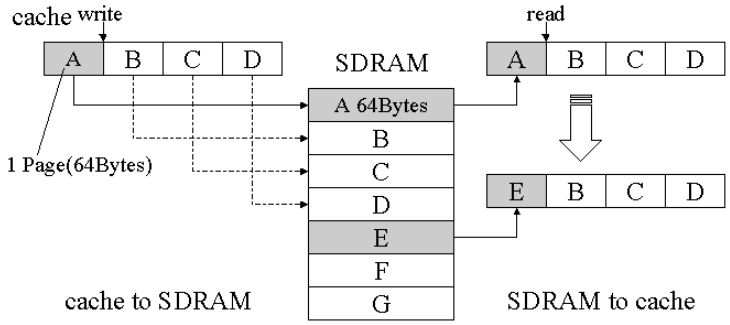

図 4.7 cache と SDRAM の関係

4.2.1 問題

現在の実装では,IP パケットの処理が決定するまでに,IP アドレスマッチング (最大 20 段) の処理が入るために問題は起きない. しかし, 今後 DDP の処理性能が向上した場合に, この問題が生じる.DDP の処理性能が 4 倍の 560MHz 相当で動作した場合に,

(a) IP パケットが全て投入される 1518/4*30=11385nsec 後に行う.
(b) キャッシュへの先読みが書込みより先行してしまう問題を回避するため,4Page 分の書込みが終了する 4*16*30=1920nsec 後に行う.
(c) IP アドレスマッチング終了後に行い, 先読みが先行する問題を回避するために, 処理を適宜待機させる.

これら 3 種類のタイミングによって DDP 2 の処理開始した命令段数の違いを, 図 4.8 に示す. DDP の処理性能が 4 倍になると,IP アドレスマッチングの処理に掛かる時間は 1/4 の 1480nsec に短縮される. しかし,IP パケット投入間隔とバッファからの IP データ読出し間隔は,Gigabit Ethernet の動作に適した 30nsec 間隔に固定されている. よって,IP パケットを構成する DDP パケット全てを投入し終える 11385nsec(図 4.8(a)) と,4Page 投入し終わる 1920nsec(図 4.8(b)) は変わらない. このため, 処理高速化のために IP アドレスマッチング後に DDP-2 の処理を開始すると, 段数を最小限に抑える図 4.8(b) では,4Page 分の書込みが終了しておらず, キャッシュへの先読みが先行しアクセスエラーとなる.

本実装は,(a) のタイミングを用いて実装しているため,DDP の性能が向上した場合でも対処可能である. しかし,DDP パケットが投入される毎に順次処理を進めるデータ駆動型の利点が失われてしまう.

そこで, 図 4.8(c) に示す IP アドレスマッチングが終了次第, 投入される毎に順次処理を進められるように, 処理対象の DDP パケットが投入されたかどうかを確認し, されている場合は処理を続け, されていない場合は, されるまで処理を待機する機能を提案する.

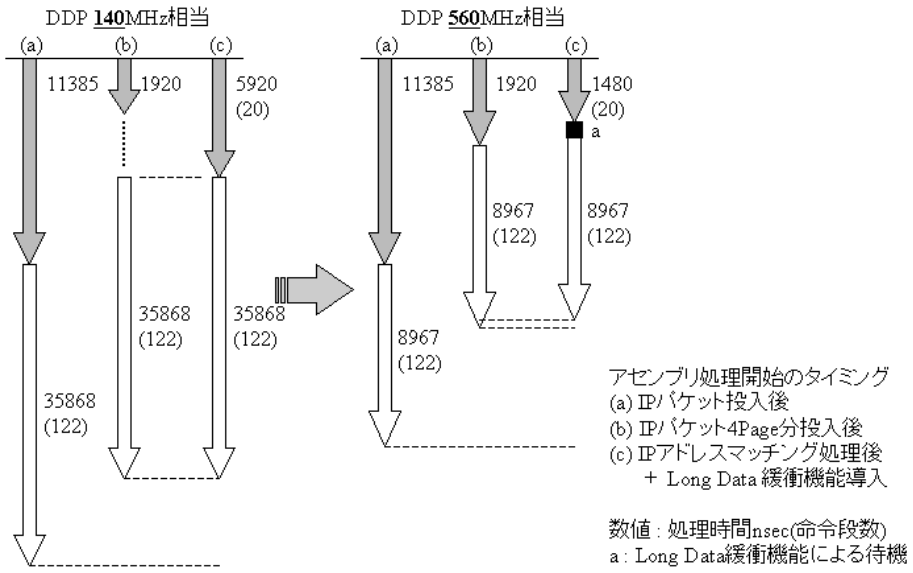

図 4.8 アセンブリ処理開始のタイミングによる処理段数の違い

4.2.2 緩衝機能の仕組み

Gigabit Ethernet から DDP へ投入される DDP パケットの世代番号 FS,CH は, 入力される IP パケット毎に偶数連番が,DDP パケット毎に連番が割り振られる. このため,DDP にどの IP パケット (FS) の, どのデータ (CH) まで投入されたかを示す世代番号 FS,CH の情報を保存し, それを確認することは容易である. ただし,DDP の命令を用いて確認するには, 一命令に 294nsec かかるため, それ以上の間隔でしか情報の更新を確認できず, また, 後続の処理を待機させるため, リソースを無駄に消費し, 非効率である. そこで,DDP 内で IP パケット分の DDP パケットを生成する copyb2 命令において, この情報を確認し, まだ IP パケットのデータが投入されていなければ, 生成処理を待機する機能を提案する.

現在の DDP に実装するものは,copyb2 命令が実装されている FP で,CH を 1 加算しながら DDP パケットを生成する部分に MH(Memory access & Hold) ブロックと Memory ブロックを追加する.IP パケットを構成する DDP パケットをインプットバッファ (SDRAM) へ格納するとき, 同じナノプロセッサ内の FIFO-copyb2 と異なる下側を通過し処理される. そこで,Memory ブロックの保持する投入された IP パケットの数を示す in_FS と,IP パケットのオフセット数を示す in_CH の更新方法は, インプットバッファへ格納する DDP パケッ

トが通過する際に,DDP に投入された IP パケットを構成する DDP パケットであると判断できるなんらかの条件を用いて, 更新を行う. その条件については今後の検討課題となる. 処理の流れは, 下記となる.

1. copyb2 命令が発行され, 世代番号 FS=x, CH=0 から DDP パケットの生成を開始する.
2. copyb2 のループで, CH=CH+1 を行う.
3. Gigabit Ethernet から投入した DDP パケットの世代番号 in_FS, in_CH が格納されたメモリにアクセスする.
4. FS ≦ in_FS and CH ≦ in_CH であれば,DDP パケットを生成する.
   それ以外は,DDP パケット生成を待機する.
5. CH = max_CH+1(max_CH, 生成する DDP パケットの数) の時,copyb2 命令を終了する.

copyb2 の状態遷移図

Long Data 緩衝機能を追加した copyb2 の状態遷移図を, 図 4.10 に示す.

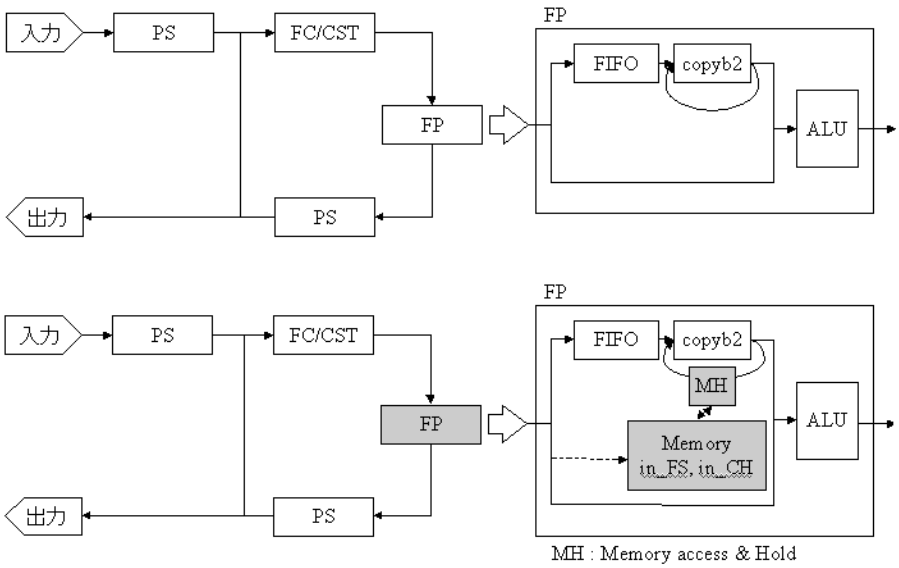

図 4.9　現在のナノプロセッサのブロック図 (上) と Long Data 緩衝機能を実装したナノプロセッサのブロック図 (下)

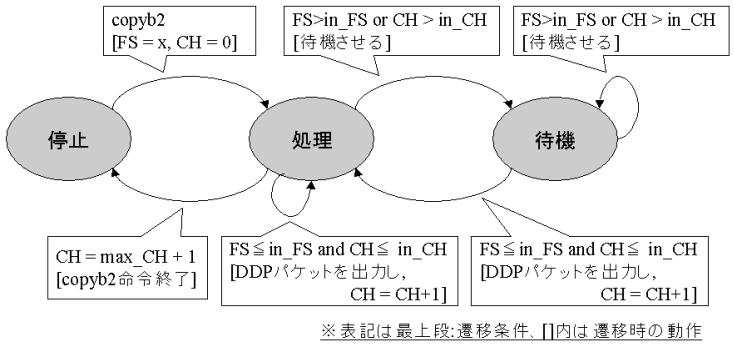

図 4.10　Long Data 緩衝機能における copyb2 命令の状態遷移図

4.3　DDP パケット対 IP パケット命令

現在の DDP は,1 つの DDP パケットに対し 1 つの DDP パケットによって命令を実行する. しかし,IP パケットを構成する n 個の DDP パケット (iDDP パケットとする) に対して, 1 つの DDP パケットで命令を実行することはできない. 代替方法として,IP パケットに行う処理のパラメータを持った DDP パケット (pDDP パケットとする) を,iDDP パケットを複製し, ノードへ渡す方法と, 命令コードを直接変更し,iDDP パケット全てをその命令を通過させる方法がある. 本実装では, 後者の方法を用いた. 二つの方法を用いたフローグラフを図 4.11 に示す. 後者は, 命令コードを直接変更しているので, この処理中に他の iDDP パケットへの処理を行うと命令コードが変更され, 途中から値が変わってしまう. よって, 処理が終了するまでは他の処理をブロックするしかない.

4.3.1　問題

DDP パケット複製方法と命令コード直接変更方法との二つの方法には, 重大なデメリットがあり, 最善の方法とは言い難い. それぞれのメリット, デメリットを併せて表 4.2 に示す.

そこで, 並列処理可能な, 処理時間の短い,DDP パケット対 IP パケット命令を実現する構造を提案する.

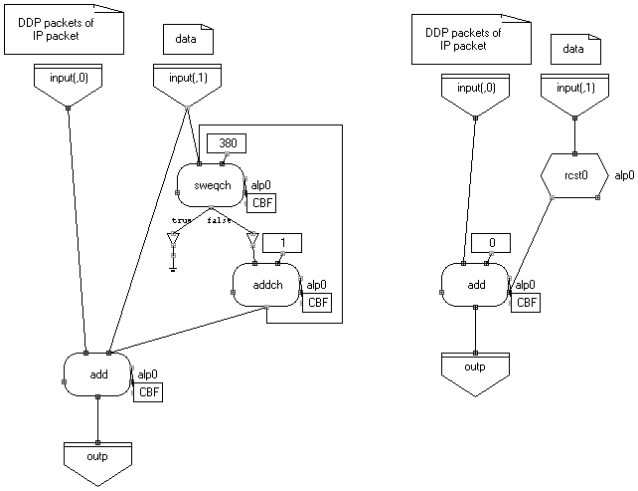

図 4.11 DDP パケット複製方法 (左) と命令コード直接変更方法 (右) のフローグラフ

表 4.2 DDP パケット複製方法 (上) と命令コード直接変更方法 (下) のメリットとデメリット

複製する方法	
メリット	他の IP パケットの iDDP パケットと並列処理が可能
デメリット	世代番号 CH の変更, 最後を判断する分岐条件の 2 命令が必要 2 命令ずつ待合せが生じ, 待合せにリソースを消費する 処理時間が 2(n-1)*294nsec と長くなる

命令コードを変更する方法	
メリット	処理時間 $n * \frac{1}{33M}$ と短い
デメリット	他の IP パケットの iDDP パケットとの並列処理が不可能

4.3.2 DDP パケット対 IP パケット命令の仕組み

並列処理が可能で, 処理時間の短い DDP パケット対 IP パケット命令を実現する機構について説明する.

iDDP パケットを 1 つの DDP パケットとして擬似的に認識させ, 命令を処理する. 現 DDP に追加することは,

- DDP パケットのタグに 1bit のフラグ (more フラグ) と,

4.3 DDP パケット対 IP パケット命令

- DDP のナノプロセッサの FC/CST で擬似フラグによる処理と,
- FC/CST に命令が DDP パケット対 IP パケット命令か否かを識別する 1bit の情報

である. ただし, 前提条件として

- pDDP パケットが iDDP パケットよりも先に待合せとして FC/CST に入力される,
- iDDP パケットの末尾以外に more フラグ=1 がセットされている,
- iDDP パケットの末尾の DDP パケットが最後にノードに入る,

ことが必須である.

このときのブロック図を図 4.12 に示す. more フラグの 2 つの役割は,

- DDP パケット全体に対する役割

  DDP パケット (more flag=0) と iDDP パケット (more flag=1) の識別

および,

- iDDP パケットに対する役割

  末尾の iDDP パケット (more flag=0) とそれ以外 (more flag=1) の識別

である. 処理の流れは, 次のようになる.

1. 処理する値を持つ pDDP パケット入力

   FC/CST に待合せ情報を格納する.
2. iDDP 先頭パケット入力 (more フラグ:1, 有効)

   FC/CST でノード番号と CH を除く世代番号で待合せを行う.

   待合せ後も, 処理のパラメータを持つ pDDP パケットの情報は削除しない.
3. 後続の iDDP パケット入力 (more フラグ:1, 有効)

   iDDP 先頭パケットと同様の処理を行う.

4. iDDP 末尾パケット入力 (more フラグ:0, 無効)

通常の DDP パケット対 DDP パケットの命令として処理. 待合せ情報は削除する.

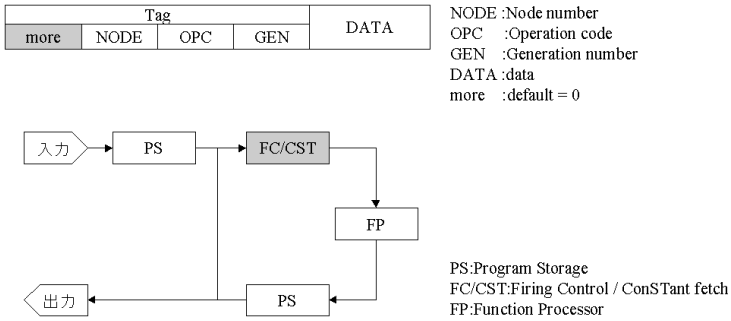

図 4.12　DDP パケット対 IP パケット命令搭載時の DDP パケットフォーマット (上) とナノプロセッサのブロック図 (下)

DDP パケット対 IP パケット命令の状態遷移図

DDP パケット対 IP パケットの命令を実行する際の FC/CST の状態遷移図を図 4.13 に示す.

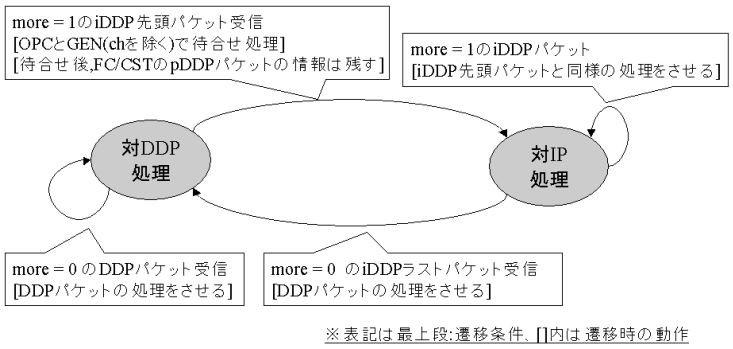

図 4.13　DDP パケット対 IP パケット命令の状態遷移図

4.3.3 DDP パケット対 IP パケット命令導入による効果

DDP 2 への効果

本命令導入により,IP パケット対 DDP パケット命令は, 他の処理をロックする必要が無い. このため,IP パケット対 DDP パケット命令を 40 段として計算していたものが, その命令自体の 1 段として計算でき, 次の処理に移ることができる. ただし,IP パケット同士の依存関係が存在など, 並列処理を行うために次に挙げる問題が存在する.

- 前出力中に先頭処理が生じた場合
  - 問題：キャッシュ

    前出力のために先読みされている読込み用キャッシュに, 先頭処理によって書き込み用キャッシュに書き込まれるデータのうち, 先頭 4Page が, 読込み用キャッシュに書き込まれ (読込み開始時に, 前もって先頭 4Page を読み込み様キャッシュに書き込む手間を省くための機能), 途中のデータに誤りが生じる.
  - 改善方法：

    先頭 4Page 書き込み機能を削除すれば, 逆に必要な命令段数が増加する恐れがある. また, 一連の処理で同じキャッシュアドレスに書き込みを行うため, 改善は不可能である.
- 出力処理中に先頭処理が生じた場合
  - 問題：キャッシュ

    前出力中に先頭処理が生じた場合と同様
  - 改善方法：

    出力処理を行っているキャッシュを利用する (前の IP パケットに対して依存関係が有る)IP パケットへの処理を待機させ, 後続の依存関係の無い IP パケットの処理を優先させる.
- 先頭処理中にアセンブリ処理 or アセンブリ処理中にアセンブリ処理が発生した場合
  - 問題：キャッシュ

書き込み用キャッシュの同じ Page に同時にアクセスが発生した場合, 誤りが発生する.

– 改善方法：

∗ 同じキャッシュにアクセスを発生させない機構の導入する.

∗ 先頭 or アセンブリ処理中の IP パケットにアセンブリできる IP パケットの処理を待機させ, 他の IP パケットを優先させる (出力処理中に先頭処理発生の回避方法と同様).

- 先頭処理 or アセンブリ処理中に出力処理が発生した場合

– 問題：キャッシュ

キャッシュの先読みが書き込みより先行する,Long Data 緩衝機能で述べたものと同じ問題が生じる.

– 改善方法：

Long Data 緩衝機能の応用で, 回避可能と考える.

以上に挙げた問題点を改善方法によって解決できた場合の DDP 2 に実装した処理プロセスと最大命令段数を, 図 4.14 に示す. 図中の処理の右下の数字は,(IP パケット対 DDP パケット命令段数) → (本命令導入後の命令段数) を示す.

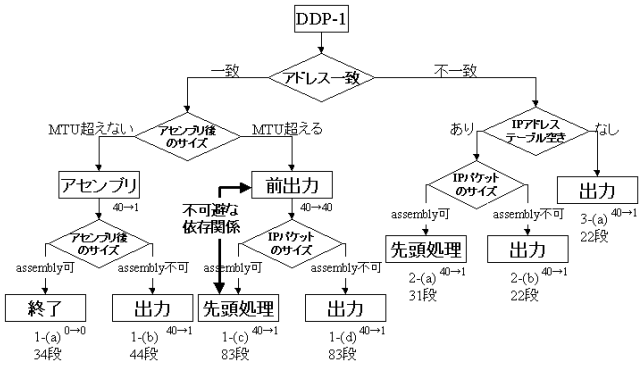

図 4.14　DDP パケット対 IP パケット命令導入時の DDP 2 の各処理プロセスと最大命令段数

DDP 3 への効果

DDP-3 で本命令を用いることはできる. しかし, シリアル出力のため処理速度向上は得られない.DDP 2 で IP パケット対 DDP パケット命令を用いた場合, 出力対象のデータが DDP 3 に到達していない可能性が有る.Long Data 緩衝機能を用いて回避することも可能ではあるが,Gigabit Ethernet へ出力する際,30nsec 間隔 (前後に余裕はある) で DDP パケットを Gigabit Ethernet へ出力しなければならないというハードウェア制限 (しない場合出力処理が破棄される) により, 処理を待機させることは不可能である.

DDP パケット対 IP パケット命令導入による並列数

1 つのノードで DDP パケット対 IP パケットを処理する場合,30nsec 間隔で iDDP パケットが nPE に到達する. 一命令が 294nsec のため,1 つの IP パケットの処理が終了するまで,nPE に 10.5 個の DDP パケットが存在する.nPE が一度に保持できる DDP パケット数を 32 個 (正確な数値は不明, ただし, この値前後で並列処理能力を越える) とすると,3 つの IP パケットまでしか並列処理が行えない.

処理速度

DDP パケット対 IP パケット命令を導入し, そのときの問題点を解決できた場合の各 DDP の処理段数を表 4.3 に示す. 逐次処理の最大命令段数は, 前出力中に先頭処理が生じる図 fig:ddp2-process2 の 1-(c) となる場合の 83 段に制約される. よって,83(段)*294(nsec)=24402(nsec) 間隔で処理可能で, 約 40980pps で, 約 74.747Mbps となる. 実装時の 50.032Mbps に比べ, 約 49%の処理能力の向上が期待できる.

表 4.3　DDP パケット対 IP パケット命令導入時の各 DDP における所要時間 (294nsec/段)

DDP i	**最大命令段数**	**逐次処理の最大命令段数**
DDP 1	20	15
DDP 2	87	83
DDP 3	70	45

4.3.4　DDP パケット対 IP パケット命令のまとめ

IP パケット対 DDP パケット命令によって処理を並列化できる. さらに, 処理の高速化を図ることができ,40 段相当の処理時間を 1 段に抑えることができる. ただし, 投入される IP パケットに前後の依存関係がある場合, 即ち, 宛先·送信元 IP アドレスが同じ IP パケットが連続して投入された場合などに, 並列処理を行うと, キャッシュの同じ Page に異なるデータを同時に更新する恐れがある. このため, 本章の提案方式を用いただけでは, 実装した処理の速度向上は難しい. 依存関係によって生じる逐次処理を最小限に抑えるためには, さらにキャッシュのアクセスで生じる競合問題について検討を行う必要がある.

また, 並列処理可能な IP パケットの増加によって,DDP の処理能力を越える可能性もある. これには,DDP の処理能力を越えない様にするスケジューリング機構か, または,DDP に掛かる負荷を分散させる機構を検討する必要がある.

第 5 章

まとめ

5.1 まとめ

ネットワークの高速化が進み,peer-to-peer を用いたアプリケーションや,IP 電話, 映像配信などの普及によって, 今後もネットワークトラヒックは増加する. ネットワーク利用帯域の伸び率が年約 3 倍に対し, ネットワークノードのプロセッサの性能は,18 ヶ月で 2 倍の向上率である. 現状の推移を続けると, プロセッサの処理能力がボトルネックとなる. 特に, トラヒックの集中する基幹網内のプロセッサの負荷が高くなる.

その負荷を軽減するために, 基幹網とアクセス網との間にあるエッジルータにおいて, 宛先と送信元の IP アドレスが同じ IP パケット同士を基幹網の MTU を越えないサイズまで結合·分解を行うパケットアセンブリ方式を研究している. この方式は, プロセッサ負荷の軽減と基幹網内のスループット向上というメリットがある. しかし,IP パケットの待合せと結合·分解の処理によって, 遅延の増加と遅延の揺らぎの幅の増加というデメリットが考えられる. そこで, この処理を非同期パイプラインの特徴を持つデータ駆動型プロセッサに実装し, 処理速度の向上を検討した. また, ノイマン型プロセッサの性能の異なる DDP をネットワークプロセッサとして用いるため, マルチメディア処理用に作成された DDP でネットワークトラヒックを扱う際に生じる問題点と, その改善案について検討した.

本研究で実装したパケットアセンブリ装置の処理能力は,124(命令段数)*294(nsec) = 36456nsec 間隔で投入される IP パケットを処理できる. これより,27430pps となる. 高知工科大学のトラヒック集計結果から得られた IP パケットの平均サイズ 228Bytes を考慮すると,50.032Mbps となる. 基幹網の速度は,1Gbps 以上であるため, 現段階では, 実用に耐え

ない.

アセンブリ処理の高速化のために,IP アドレステーブル参照, 比較を高速化する投機実行と,DDP 内部で IP パケットを構成する DDP パケットを扱う処理の効率化を実現する,Long Data 緩衝機能と DDP パケット対 IP パケット命令を提案した. 投機実行と Long Data 緩衝機能は, 将来の実用化に向けて必要な技術である. ただし, 実装を目的とて作成した本装置に適用できないため, 速度向上につながらない.DDP パケット対 IP パケット命令は, その命令の導入によって生じる,IP パケット間の依存関係 (IP アドレスの同じ IP パケットが連続した場合) に起因する新しい問題を解決できたと仮定すると,84(命令段数)*294(nsec)=24402nsec 間隔で IP パケットを処理できる. 言い換えると,40980pps,74.747Mbps となる.

今後の研究展開は, 上記に挙げた改善案が実際に DDP に搭載されるよう詳細の設計や, 汎用性を考慮した工夫, それに対する費用対効果の算出を行う. また, パケットアセンブリ方式の処理の相互関係を洗い出し, 改善案の導入によって生じる新たな問題に対する改善案を提案していく.

5.2　今後の研究展開

アセンブリ処理の解決すべき問題として,

- IP アドレスマッチングの処理時間
- タイムアウト

と,DDP パケット対 IP パケット命令を導入する際に生じる新しい問題点である

- キャッシュアクセス
- IP パケット処理順序
- 並列化による DDP の処理能力超過

に対する改善の見通しについて説明する.

5.2.1　IP アドレスマッチングの処理時間

IP アドレスマッチングを全て並列化して処理した場合,IP アドレステーブルを格納している内部メモリの参照時に, 問題が生じる. これは,IP アドレステーブル参照からマッチング結果を再び IP テーブルに反映するまでに 15 段の命令が存在することが原因で, テーブルが空きであった場合, 宛先 IP アドレス下位 5bit が一致する IP パケットが 15 段 (4410nsec) の間に IP アドレステーブルを参照すると, 同じ結果を得てしまう. そのまま処理を続行すると,IP アドレステーブルが重複利用される. 投入された IP パケットを処理する前に, 前の IP パケットの処理が終了しているか, していなければ, 宛先 IP アドレスが一致しているかどうかを検出し, 同じ IP アドレステーブルを参照する場合のみ, ブロックする方法の導入を検討するその概念を図 5.1 に示す.

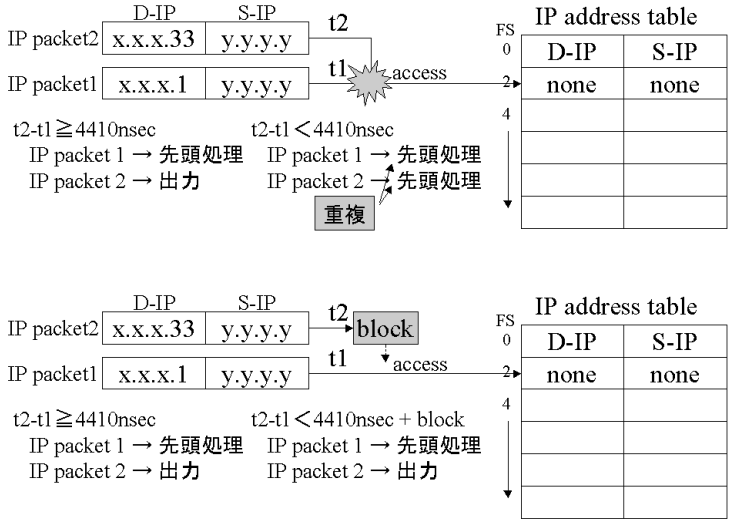

図 5.1　並列処理時の IP アドレスマッチング問題 (上) と改善案 (下)

5.2.2　タイムアウト

現在, パケットアセンブリ対象の IP パケット待合せのタイムアウト時間は, アセンブリ対象となる先頭の IP パケット到達から 5msec である.DDP には, クロック信号が無く, 時間の概念を持たない. そこで,5msec の検出に, タイムアウト用 DDP パケットを 5 段の命令

(1470nsec) を 3401 回ループさせることによって実現している. しかし, この方法では,DDP の特徴に反しており, また, 先頭 IP パケット毎に DDP パケットを生成, ループさせねばならず, リソースが無駄に消費される. さらに,IP アドレステーブル数が多い場合,DDP の処理能力を落とす結果につながってしまう.

そこで,IP パケットの投入を検知し, 投入間隔の平均時間を元に計算する方法を検討する. しかし, この場合, ループさせる場合に比べ, 時間を正確に計ることができない. この方式を実装する場合,IP パケットの待合せの時間の精度が, ネットワークにどれだけ影響するか検証を行う.

5.2.3　キャッシュアクセス

アセンブリ処理を並列化した場合, 宛先·送信元 IP アドレスの同じ IP パケットが連続して投入された場合で, 後の IP パケットが, アセンブリ処理後のサイズが MTU を越えるために,SDRAM へ格納された IP パケットを出力する処理を行ったとする. このとき, 前の IP パケットのアセンブリ処理, または, 先頭パケット処理を行っている最中の場合,Long Data 緩衝機能を提案する原因となった状態が発生してしまう.

逆に, 後の IP パケットが, 前の IP パケットの後ろにアセンブリを行ったとする. このとき, 前の IP パケットの処理が終了していなかった場合, 前の IP パケットの情報と後の IP パケットの情報が, キャッシュの同じ Page に対して書き込みが発生する可能性がある (図 5.2(上)). 書き込みが発生した場合は,SDRAM に格納された IP パケットのデータに誤りが生じる.

そこで, 前の IP パケットと後の IP パケットとの間に依存関係がある場合のみ, 処理をブロックする方法の導入を検討する (図 5.2(下)).

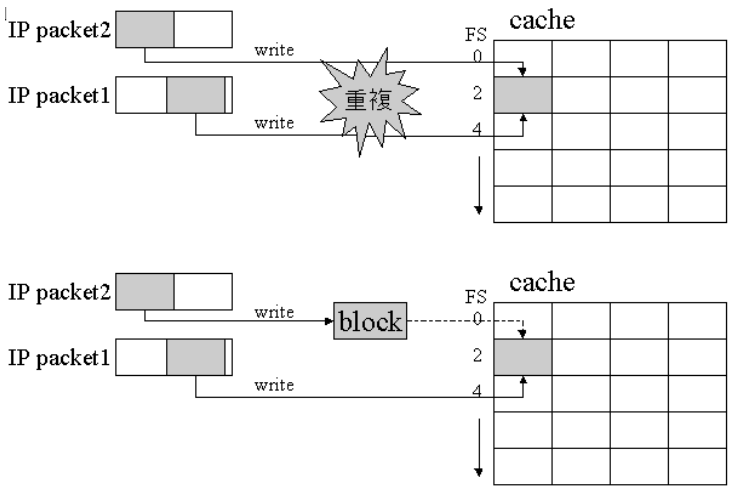

図 5.2　並列処理時のキャッシュアクセス問題 (上) と改善案 (下)

5.2.4　IP パケット処理順序

SDRAM へのアクセスがノーウェイトで行われた場合,SDRAM アクセス問題は発生しない. しかし,SDRAM のアクセス速度が, プロセッサの速度と同等になるのは現実的でない. そこで, 現在, 投入パケット毎に逐次処理している部分を改め, 前に処理中の IP パケットと依存関係がある場合, その IP パケットに対する処理をブロックし, 後続の IP パケットに処理を優先させる機構の導入を検討する. その概念を図 5.3 に示す. これは, 先に挙げた IP アドレスマッチングの処理時間と SDRAM アクセスの両問題を解決できるものとなる.

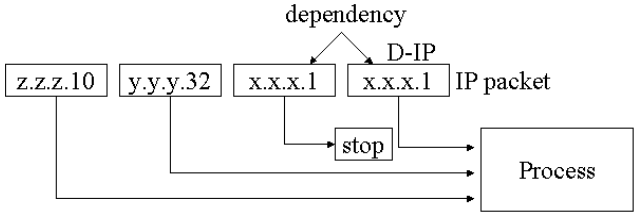

図 5.3　依存関係にある IP パケットの処理を停止する概念図

5.2.5　並列化による DDP の処理能力超過

DDP は,DDP 同士の相互接続が用意で, 評価システム (図 3.2) を例にとると, 図 5.4 に示すように, 左右 (左:DDMP#7 - DDMP#3 - DDMP#4 - DDMP#0 - DDMP#1 系, 右:DDMP#7 - DDMP#5 - DDMP#6 - DDMP#2 - DDMP#1 系) で別々の処理をさせることも, 同じ処理をさせることも可能である. そこで, アセンブリ処理を高速化 (並列処理数の増加) するために, これらを用いる方式を検討する. ただし,DDMP#7 と DDMP#1 とで, どちらに処理を分配するかを決定する負荷分散機構を設ける必要がある. この負荷分散機構の方式について

- IP アドレスによって分配する
- プロセッサ負荷を予測して分配する
- メモリアクセス頻度を予測して分配する

などが考えられる.

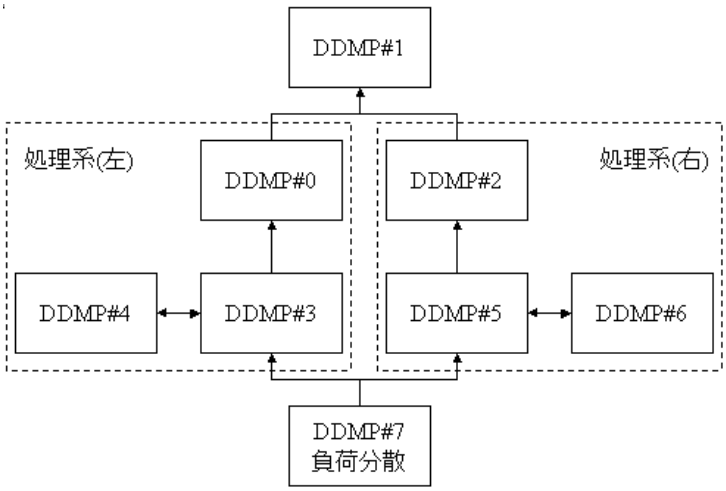

図 5.4　負荷分散を構成したときの評価システム

謝辞

本研究を行うに際し, 多大なるご指導, ご鞭撻を頂いた, 本情報システム工学コースの島村 和典教授, ならびに, 通信·放送機構高知トラヒックリサーチセンターの高松 希匠研究員, ならびに, 神田 敏克フェロー (元研究員) に深く感謝致します. 本研究に必要な機器を提供して下さった通信·放送機構高知トラヒックリサーチセンターの皆様に感謝致します. また, 本講座院生 中平 拓司氏, 浦西 慶規君, 橋本 江里子さん, 山岡 徹也氏, 三谷 乙弘君, 中平 和友君, 谷岡 亮介君, 山田 敦君に感謝致します. さらに, 在学中, 親切なる御助言を頂いた岡田 守教授, 岩田 誠教授をはじめとする諸先生方に心より感謝致します.

参考文献

[1] 神田 敏克, 島村 和典; A throughput improvement on Routers by the control of packet size, 信学技報, SSE2000-114, IN2000-45(2000-09).

[2] 浦西 慶規, 神田 敏克, 島村 和典; インターネットトラヒック特性を考慮したパケットアセンブリ転送方式の効率化に関する検討, 信学技報, NS2001-237, IN2001-1931(2002-03).

[3] Hiroaki Terada, Souich Miyata, Makoto Iwata; DDMP's: Self-Timed Super-Pipelined Data-Driven Multimedia Processors, Proceedings of THE IEEE, vol 87, no.2, February,1999.

[4] 小林 寛征, 神田 敏克, 島村 和典; パケットアセンブリによる中継ルータ CPU の負荷軽減に関する一考察, 電気関係学会 四国支部連合大会 講演論文集 12-20,2000.

[5] 小林 寛征, 神田 敏克, 島村 和典; パケットアセンブリによる基幹網の処理軽減に関する一考察, 電子情報通信学会 2001 年総合大会講演論文集, B-7-202,2001.

[6] Andrew Odlyzko; Internet Growth:Myth and Reality, Use and Abuse, http://www.cisp.imp/november_2000/odlyzko/11_00odlyzko.htm, November 2000 (Jan, 2003).

[7] http://www.jpix.ad.jp/jp/techncal/traffic.html, Japan Internet Exchange (01/26/2003).

[8] Packet Sizes and Sequencing, http://www.caida.org/outreach/resources/learn/packetsizes/index.xml, Cooperative association for internet data analysis, April 1998(02/23/2003).

[9] http://www.ietf.org/rfc/rfc1191.txt, Path MTU Discovery, RFC1191, IETF (Jan, 2003).

[10] 小林 寛征, 高松 希匠, 島村 和典; データ駆動型プロセッサによるパケットアセンブリの

実装と評価, 信学技報, CS2002-138, IE2002-126(2002-12).

付録 A

各データリンクの MTU

表 A.1　データリンクごとの MTU(Bytes)

データリンク	MTU	Total Length
IP の最大 MTU	65535	-
Hyperchannel	65535	-
IP over HIPPI	65280	65320
16Mb IBM Token Ring	17914	17958
IP over ATM	9180	-
IEEE802.4 Token Ring	8166	8191
IEEE802.5 Token Ring	4464	4508
FDDI	4352	4500
Ethernet	1500	1514
PPP	1500	-
IEEE802.3 Ethernet	1492	1514
IP の最小 MTU	68	-