修 士 論 文

# VLIW アーキテクチャを適用した高速、低消費電力ネットワークプロセッサの研究

北陸先端科学技術大学院大学
情報科学研究科情報システム学専攻

五由出 将嗣

2006 年 3 月

修 士 論 文

# VLIW アーキテクチャを適用した高速、低消費電力ネットワークプロセッサの研究


指導教官　日比野靖 教授

審査委員主査　日比野靖 教授
審査委員　田中清史 助教授
審査委員　井口寧 助教授

北陸先端科学技術大学院大学
情報科学研究科情報システム学専攻

410048 五由出 将嗣



提出年月: 2006 年 2 月

## 概 要

近年、ネットワーク利用者の急激な増加や大容量データ転送に伴い、ネットワークのさらなる高速化が求められている。。こういった背景からソフトウェアによる実装が可能で、並列化による性能向上が容易なネットワークプロセッサを用いることによって、柔軟性にとんだ高機能なルータを短期間でかつ安価に開発することを期待されている。ネットワークプロセッサではパケットフォワーディングの処理が大部分であり、データ移送(load/store)、条件分岐、論理演算の並列実行が可能な VLIW アーキテクチャに適している。我々が提案する手法は、このネットワークプロセッサに対して VLIW アーキテクチャを用い、高速化と消費電力の低減を検討する。

# 目 次

# 第1章　はじめに

## 1.1　背景と目的

既存の高速ルータには汎用プロセッサと ASIC を利用した構成が大半であるが、高性能 ASIC チップの開発には莫大な費用と時間がかかってしまう。こういった背景からソフトウェアによる実装が可能で、並列化のよる性能向上が容易なネットワークプロセッサを用いることによって、柔軟性にとんだ高機能なルータを短期間でかつ安価に開発することが期待されている。我々が提案する手法はこのネットワークプロセッサに対して VLIW アーキテクチャを用い、高速化と消費電力の低減を可能とするものである。ネットワークプロセッサではパケットフォワーディングが処理の大部分であり、データ移送 (load/store)、条件分岐、論理演算の並列実行が可能な VLIW アーキテクチャに適している。VLIW アーキテクチャは、スーパスカラマシンと比較して単純であるが、事前に並列に実行できる処理を１つの命令としてまとめ、プロセッサへと与えるためにコンパイラを使用するので、トータルとしての処理速度向上は見込めなかった。しかしながら、ネットワーク処理の典型的な作業であるパケットフォワーディングは数十種類であるため、コンパイラに頼らなくてもあらかじめソフトウェア的に条件を与えておくことで、最適な並列動作を行うことができる。つまり、VLIW の基本理念であるハードェア自体は単純なものとし、複雑な部分はソフトウェアで行うことができる。我々はこういった特徴に着目し、ネットワークプロセッサの高速化に取り組む。
[1]

## 1.2　本論文の構成

**第２章** ネットワークの仕組み、ネットワークプロセッサについて述べる

**第３章** VLIW アーキテクチャについて述べる

**第４章** 本研究で提案するパケットフォワーディングを高速に行う手法について述べる

**第５章** 評価対象の、ツリー探索において、VLIW 適用前と後で比較し、その結果の考察について述べる。

[1] VLIW(Very Long Instruction Word)

**第６章** まとめと今後の課題。

# 第2章　ネットワークプロセッサ

本章では現在のネットワークの概要について述べる。

## 2.1　ネットワークプロセッサとは

ネットワークプロセッサとは、Intel 社のネットワーク機器向け半導体アーキテクチャである、IXA(Internet eXchange Processor) アーキテクチャに基づいたネットワーク機器に特化したプロセッサのことである。従来、ネットワーク機器の心臓部には、各社が製品ごとに個別に ASIC（特定用途向け IC）を開発し、ハードウェアでの処理を行ってきた。こうした方式は、ネットワーク技術がより高度化、複雑化していく中では開発効率が悪く、柔軟性に欠ける。こうした状況に対し、パソコンにおけるマイクロプロセッサのように比較的、汎用的な半導体チップをネットワーク機器の心臓部に据え、ソフトウェアで各種ネットワーク技術を実装していくモデルを提唱した。その核となる半導体チップの基本設計が IXA である。ネットワークプロセッサの動作は主にパケットフォワーディング（パケットの受信、テーブルルックアップ、送信）の処理に費やされる。特にテーブルルックアップとは、ルータ自身が受信したパケットをどこの出口から送信するかを記憶した動的に変化するルーティングテーブルと呼ばれる表を保持しており、この表を参照する作業は毎パケットごとに実行されるために一秒間に数千回から数万回に及ぶ。またこのテーブルルックアップの作業に時間を費やしてしまうために、ネットワーク全体のボトルネックとなってしまっている。

### 2.1.1　IP ネットワーク通信の仕組み

ネットワークの世界では、マシンやネットワーク機器などの通信を行う拠点の最小単位をノードと呼び、通信したい相手ノードを特定するには何らかの番号や ID を個別に付けておくのが望ましい。これが IP アドレスである。IP アドレスは 4 バイトからなる数値で、実際には 2 進数のビット列として使用されているが、人間が理解しやすいように 1 バイトごとに 10 進数に変換してピリオドで区切って表記される。IP アドレス全世界において唯一でなければならなく、それによって通信先および通信元ノードがそれぞれ特定される。IP ネットワークでは、データはパケットと呼ばれる小さなデータの固まりに格納される。パケットは、送信元 IP アドレス、送信先 IP アドレスなど、IP ネットワークに必

| Internet | | | | | |
|---|---|---|---|---|---|
| Destination | Gateway | Flags | Refs | Use | Interface |
| default | 140.252.13.33 | UG S | 0 | 3 | le0 |
| 127.0.0.0 | 127.0.0.1 | UG S R | 0 | 2 | lo0 |
| 127.0.0.1 | 127.0.0.1 | U H | 1 | 55 | lo0 |
| 128.32.33.5 | 140.252.13.33 | UGHS | 2 | 16 | le0 |
| 140.252.13.32 | link#1 | U C | 0 | 0 | le0 |
| 140.252.13.33 | 8:0:20:3:f6:42 | U H L | 11 | 55146 | le0 |
| 140.252.13.34 | 0:0:c0:c2:9b:26 | U H L | 0 | 3 | le0 |
| 140.252.13.35 | 0:0:c0:6f:2d:40 | U H L | 1 | 12 | lo0 |
| 140.252.13.65 | 140.252.13.66 | U H | 0 | 41 | sl0 |
| 224 | link#1 | U C | 0 | 0 | le0 |
| 224.0.0.1 | link#1 | U H L | 0 | 5 | le0 |

図 2.1: ルーティングテーブル

要なヘッダ情報とともに、実際に送りたいデータを内部に含んでいる。また、パケットの最大サイズはあらかじめ決められており、送信すべきデータがこれよりも大きい場合はいくつかのパケットに分割し、それぞれ個別に相手に届けられ、再び結合される。

## 2.2 経路探索手法

### 2.2.1 ルーティングテーブル

ネットワークにおけるルーティングにおいて最も重要なのがルータに入ってきたパケットの中から宛先 IP アドレスを読み出し、ルーティングテーブルと呼ばれる表から、最適な出口へとフォワーディングしてやることである。ルーティングテーブルは図 ?? のようなものであり、重要な項目は Destination、Gateway、Interface であり、入力されてきたパケットの中の宛先 IP アドレスとルーティングテーブル内の Destination の中にそのアドレスに該当するものがあるか探す。該当するものがあればそのアドレスの Interface 番号の出口へとフォワーディングされ、また、該当するあどれすが存在しなければ Destination 内の default に該当する Interface の出口へとフォワーディングされる。

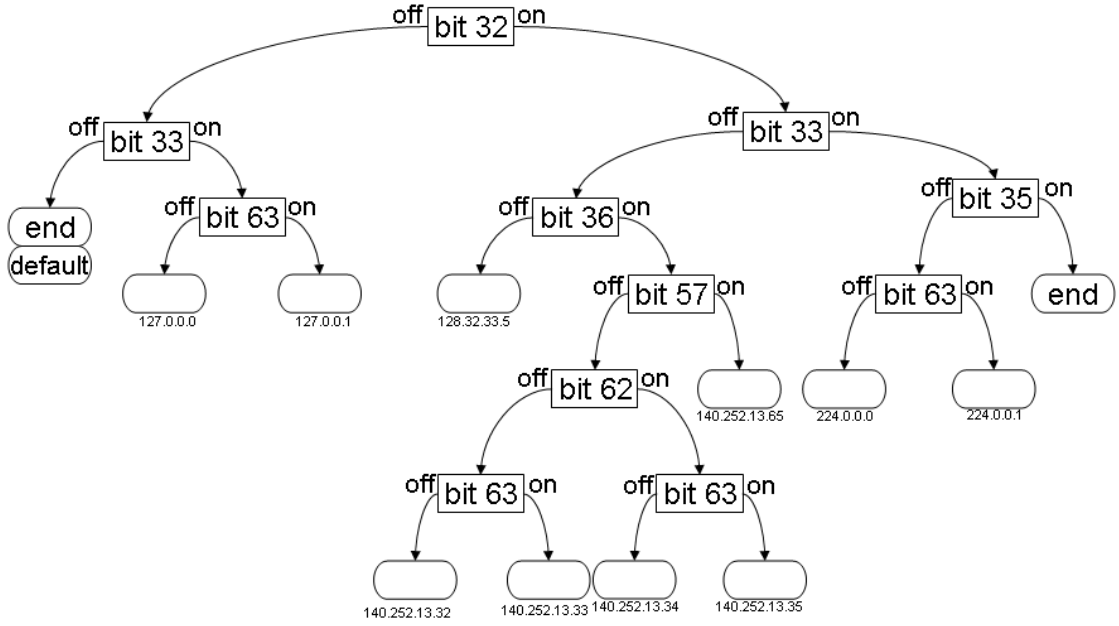


図 2.2: パトリシアツリー

### 2.2.2 パトリシアツリー

この探索の動作は、視覚的に我々が見易いように図???のように表記されているが、ルータ内の記憶には以下の図??? に示すツリー構造となって記憶されている。このツリー構造はパトリシアツリーと呼ばれているもので、この考えは [Sklower 1991] の VanJacbson のものであり、現在のネットワークではこの方式のツリー構造によって探索されている。パトリシアツリー構造は一方向分岐を取り除いたバイナリラディックスツリーであり、ビット単位で枝が一本しかない無駄なノードをまとめて圧縮し、効率化をはかったツリー構造である。特徴として、経路の追加や削除といった作業には手間取るが、それ以上に頻繁に行われるテーブルルックアップには非常に高速に行える構造となっている。このパトリシアツリー構造においてチェックするビット位置の決定は、各ノード以下のリーフのとあるビット位置において、0 と 1 が混在しているびっとをチェックビット t して掲げてやる。またこのとき、ビットは先頭ビットから順にみていってやる。

### 2.2.3 探索手法

この判定は若い番号から順番に行っていく。ノードのビット番号が３２から６３ビットになっているのは、インターネットソケットアドレス構造体のビットオフセットが図????

のようになっており、IP アドレスの部分は３２から６３ビット目までとなっているからである。図 ?? を利用し、見ていくと、宛先 IP アドレス 128.32.33.5 を持ったパケットが入ってきたと考えてみる。そうすると、このアドレスをバイナリ表記したのは図???? であり、まずこの中の３２ビット目をチェックするとビットは１つまり ON であるため右へと分岐する。この作業を繰り返していくと 32-¿Right-¿33-¿Lefr-¿36-¿Left-¿Exit となり、無事に出口を見つけることができる。では宛先 IP アドレスが 128.32.33.6 を持ったパケットが入ってきたときはどうであろうか？このアドレスをバイナリ表記すると先ほどのアドレス 128.32.33.5 とは、62 と 63 ビット目のビットしか異ならないため、このままツリー構造では、同じ出口に出力されてしまう。そこで、実際には、出力先が見つかった後に、その出口が保持している IP アドレスとパケットの保持している IP アドレスが一致しているかチェックした後に出力し、また、一致しなかったパケットは Default へとフォワーディングされる仕組みとなっている。

## 2.3 VLIW

VLIW(Very Long Instruction Word) アーキテクチャとは、複数種類の命令操作（演算、メモリアクセス、分岐、etc）を１つの命令としてまとめて同時に実行する。同時実行される操作 (Operation) の数は一定であり、実行できる操作が無い場合は「何もしない (Nop)」命令で埋められる。複数の操作を一つの 命令に埋め込むので、命令の長さが従来のプロセッサに比べて極めて長い。１２８ビットあるいは２５６ビットといった長いフォーマットの命令で、複数の演算ユニットを使った処理を並列に記述するアーキテクチャ。例えば１命令で 32 ビット命令を 4 個分（あるいは 8 個分）の処理が可能になる。また並列動作を行うためにコンパイル時にソフトウェア的にスケジューリングされる。よってソフトウェアの負担は大きくなるがハードウェアが単純になり回路規模が小さくなるので消費電力の抑制にも効果がある。VLIW とスーパスカラとの、並列実行における比較を図 ??? に示す。VLIW では複数の命令はコンパイラによって並列実行できる VLIW 命令へと変更、スケジューリングされるソフトウェア主導型である。それに対し、スーパスカラ型は、ハードウェアスケジューラによってスケジューリングされ並列実行されるハードウェア主導型である。よって VLIW アーキテクチャの方が回路が単純となり、消費電力も減り、かつ耐クロック性も向上する。

# 第3章　ルーティングモデル

本章では、現状のツリー探索においてツリーの深さの違いによる差がどれほどのものかを検証し、次章での提案手法に生かすものである。

## 3.1　現状の探索アルゴリズム

現状でのアルゴリズムは、各ノードが所持している checkbit の情報を参考に、その指定されたビット列を入力パケットを見て、OFF ON で判定し左右へと分岐し、リーフへと達すると、リーフは固有のアドレス番号を所有しており、そのアドレス番号と合致判定を行い、出力するものである。

### 3.1.1　実験条件

IP パケットサイズは最大長が 65535(Octet) であるが、イーサネットや一般的なネットワークの MTU(Maximum Transmission Unit) が 1500(Octet) であるため今回はパケットサイズを 1500(Octet/packet) とした。探索に用いるパトリシアツリーは以下に示す図??? とし、探索に用いる Key を 140.252.13.32(Level 4)、 140.252.13.39(Level 4)、140.252.13.40(Level 1) の三種類とする。

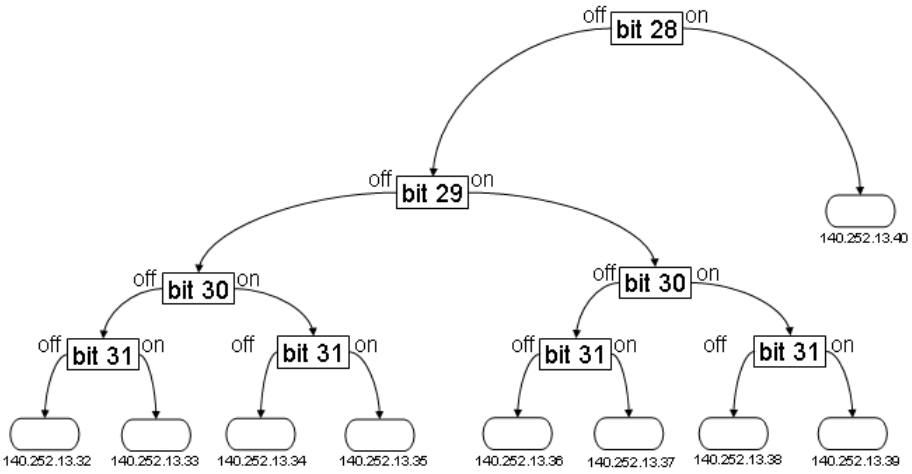
off
bit 28
on
off
bit 29
on
140.252.13.40
off
bit 30
on
off
bit 30
on
off
bit 31
on
off
bit 31
on
off
bit 31
on
off
bit 31
on
140.252.13.32
140.252.13.33
140.252.13.34
140.252.13.35
140.252.13.36
140.252.13.37
140.252.13.38
140.252.13.39

図 3.1: aaa

# 第4章　まとめ

本稿では、
　筆者は、シソーラスに全ての語を収録できると考えるのは幻想であると思うし、

# 参考文献

[1] 佐藤理史, 実例に基づく翻訳, 情報処理, Vol. 33, No. 6, pp673–681, 1992.