

PNスーパースカラプロセッサの性能評価

有田隆也 伊藤広明 曾和将容
名古屋工業大学電気情報工学科

細粒度の並列度を抽出する方式として、我々は既に機能分割型スーパースカラプロセッサによる複数命令流実行方式を提案している。この方式では、プロセッサを複数の異なる処理ユニットに分割し各処理ユニットに独立に命令流を供給する。各処理ユニットは命令間の先行関係に従って高速にトークンを通信し命令実行を制御する。本論文では、3分類した機械語命令に基づくプロトタイプモデルに関して、シミュレーションにより本実行方式の性能評価を行う。特に、(1)命令3分類による負荷分散、(2)抽出並列度、(3)トークン通信時間の影響やトークン送受信命令のオーバーラップ実行の効果について分析する。

Performance Evaluation of the PN Superscalar Processor

Takaya ARITA Hiroaki ITO Masahiro SOWA

Nagoya Institute of Technology

Faculty of Electrical Engineering and Computer Science

Gokiso, Nagoya 466, Japan

E-mail: ari@debris.elcom.nitech.ac.jp

The PN(Parallel Neumann) superscalar processor has an experimental architecture which can exploit fine-grain parallelism. This processor consists of the several functional processing units, each of which has a separate instruction stream and communicates the control tokens with other units through a very simple mechanism according to the dependence relation between two instructions. In this paper the performance of the PN processor is estimated by simulation. Especially, (1) function partitioning among the functional processing units, (2) extracted parallelism, (3) effects of token-communication time, are analyzed on the basis of the simulation results.

1. まえがき

動作クロック周波数が限界に近付きつつある中でプロセッサの性能を向上させるためには、細粒度の並列性抽出は重要であると考えられる⁽¹⁾。我々がPN(ParallelNeumann)プロセッサと呼び提案している、複数命令流をもつ機能分割型スーパースカラプロセッサ⁽²⁾⁽¹⁰⁾⁽¹¹⁾は、レジスタランスフェレレベルから機械語命令レベル程度における並列実行を可能とするアーキテクチャを有する。

PNプロセッサでは、各処理ユニットが命令の種類ごとに非同期に動作し、必要に応じて命令間の先行関係に基づくトークン(制御パルス)通信を行うことにより命令実行を制御する。したがって、PNプロセッサは、VLIW⁽³⁾型プロセッサに比べて、きめ細やかな並列性を抽出できる可能性があり、またキャッシュや遅延時間の不確定なネットワークなどを組み込むのが容易であるなどの長所を持っている⁽¹²⁾。プロトタイプモデルでは、機械語命令をコンパイル時に、(1)データ転送命令、(2)演算命令、(3)分岐関連命令の三つに分類し、それぞれを専用の処理ユニットで実行する。

本稿は、PNプロトタイププロセッサを組み込んだPNコンピュータの特性を、次の3点に関してシミュレーションに基づいて評価する。第一に、処理の3分割による負荷の分散に関して、各命令の出現頻度に基づいた評価を行う。第二に、PNプロセッサで抽出される並列度、及びこれに対する命令実行時間の影響を調べる。第三に、処理ユニット間のトークン通信について、その通信時間の実行時間に対する影響と通信タイミングの最適化による送受信命令のオーバーラップ実行の効果を評価する。これらのシミュレーションを通してPNプロトタイププロセッサの基本性能を評価する。

2. PNコンピュータの基本構造

2.1 PNグラフの実行

PNプロセッサが実行するプログラムであるPNグラフは図1のような形態をとる。ノードはアクタと呼び機械語命令ひとつを表し、アークは命令の実行順序に関する先行関係を表す。縦に並んだアクタ列それぞれを別の処理ユニットで実行するので、同一列のアクタ間のアークは同一処理ユニットによるプログラムカウンタを用いた実行の順序を表し、縦に並んだアクタ列間を結ぶアークは処理ユニット間のトークン通信を表している。

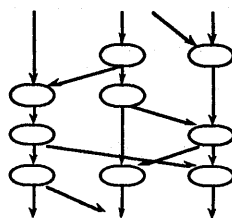


図1 PNグラフの形態

アクタ列間のアークは命令実行の先行関係のみを表しているので、送信側ユニットは受信側ユニットが対応する受信点に到達していなくても、その到達を待つ必要はない。また、PNグラフではトークンを送受する各列の命令は互いに順序付けられており、送受信関係も確定している。したがって、トークン送受信においては、送受先相手のアクタを個々に指定する必要はなく、送受信ユニットを指定するだけでよい。結局、トークン送信元の処理ユニットがわかり、トークン到着の数を記憶するように受信ユニットを構成すると、そこに制御パルスを送るだけで、極めて高速なトークン通信が実現される。各処理ユニット間は完全結合ネットワーク(処理ユニット数が3なら理論的には3本の信号線)で結ばばよい。

2.2 プロトタイプモデルの基本構成

プロトタイプモデルでは、命令は原則として、(1)データ転送命令、(2)演算命令、(3)分岐関連命令の3種類に分けられており、プロセッサ内にはそれらに応じた3つの処理ユニットがある。PNプロトタイププロセッサの基本構成を図2に示す。TU、AU、BCUは、データ転送命令、演算命令、分岐関連命令をそれぞれ実行する処理ユニットであり、それぞれを、転送ユニット、演算ユニット、分岐制御ユニットと呼ぶ。

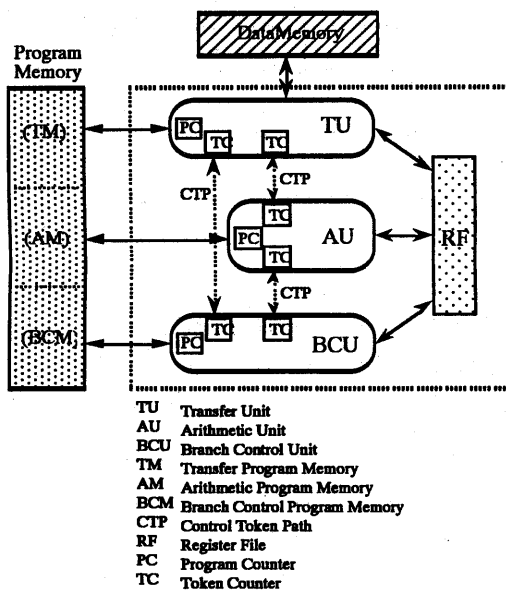


図2 PNプロセッサのプロトタイプ構成

各処理ユニット間には、トークンを伝達する信号線CTPが張られている。各処理ユニット内のTCはトークン通信のためのカウンタであり、トークンカウンタと呼ぶ。TCは送信元処理ユニット1個に対して1個ずつ用意されている。PCはプログラムカウンタである。各処理ユニットにはそれぞれ

プログラムメモリ(TM, AM, BCM)が接続されており、独立に命令をフェッチする。データメモリは転送ユニットにのみ接続されている。各処理ユニットの共有資源としてマルチポートのレジスタファイル(RF)がある。各処理ユニットは異なるレジスタに対して同時にアクセスすることができる。同一レジスタへの同時読み出しも可能である。

2.3 処理ユニット間的高速トークン通信

図3(a)に示すようなPNグラフにおける各命令は同図(b)のように表される。同図で、LOAD.a R0,R1は、LOAD命令の実行後にaで示される処理ユニット(AU)に対してトークンを送ってから、次の命令に進むことを表している。また、t.ADD.b R1,R2,R3は、tで示される処理ユニット(TU)からトークンが送られてきているのを確認した後、ADD命令を実行し、その後、bで示される処理ユニット(BCU)に対してトークンを送ってから、次の命令に進むことを表している。このように、処理ユニット間のトークン通信の有無と相手処理ユニットは、機械語命令の前後に付加するタグ（前置タグ/後置タグ）によって示される。

トークンカウンタはこのようなトークン通信のために用いられる。トークンの送信は受信処理ユニット内の送信処理ユニットに対応するトークンカウンタのインクリメントを意味する。また、送信処理ユニットからトークンが到着しているかどうかは自処理ユニット内の対応するトークンカウンタの値が正であるかどうかということに等しい。したがって、もしトークンカウンタがゼロならば相手処理ユニットによる制御パルスの送信を待ち、そうでないならばそのトークンカウンタをデクリメントしてその命令を実行する。このように、トークンカウンタは命令実行の進み具合の差を処理ユニット間にまたがる先行関係の数によってカウントするものである。このような方式の採用によりPNプロセッサでは極めてシンプルで高速な

命令実行制御が実現された。

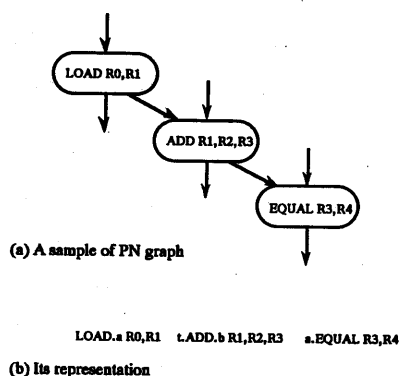


図3 PNグラフの表現

3. シミュレーション評価方式

3.1 評価内容

3.1.1 実行命令の分布

PNプロトタイププロセッサの各処理ユニットがもつ命令の出現頻度を調べる。そして、その結果に基づいて、3つの処理ユニットによる負荷分散に関して評価する。

3.1.2 抽出並列度

PNプロセッサの基本性能を知るために、抽出される平均並列度を調べる。また、命令の実行時間が変動したときの抽出並列度に対する影響も調べる。

3.1.3 トークン通信

トークン通信は前章で示したようにカウンタを用いた極めてシンプルな方式で実現されており、それに要する時間も極めて小さいと考えられる。しかも、PNプロセッサではトークン受信前に命令フェッチなどを行うことや命令実行終了前にト

ークンを送信することも可能である。このようにトークン送受信タイミングの許容範囲が比較的広いので、このタイミング最適化の効果は小さくないと考えられる。ここでは、その効果も考慮に入れて評価を行う。

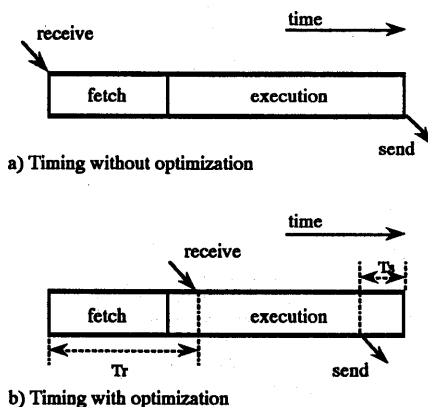


図4 トークン送受信タイミングの最適化

図4に示すように通信関連時間を定義する。 T_c は通信時間であり、トークンを送出してからそれが受けとられトークンカウンタが設定されるまでの時間を表している。 T_s は、その命令の実行終了時刻よりどれだけ早くトークンを送れるかという時間を表しており、逆に T_r はトークンを受け取るまでにどれだけ命令の実行（フェッチも含む）を進めることができるかという時間を表している。ここでは、 $T_s + T_r$ を通信最適化時間と呼ぶ。 TD は実質通信時間で、

$$TD = T_c - (T_s + T_r)$$

のように表される。タイミングをずらして $T_s + T_r$ の値を大きくすることは、実質通信時間を減らすという最適化に相当する。実質通信時間が負であることは、送信側処理ユニットの命令と受信側処理ユニットの命令がオーバーラップして実行され

ることを意味する。

3.2 評価方法

PNプロセッサはそれぞれの処理ユニット内でパイプライン化が可能であるが、本評価ではこれを想定しない。各命令の実行時間は従来の平均的なプロセッサの命令実行時間を参考にして設定した⁽⁴⁾⁽⁵⁾。またトークン通信以外の評価では、トークン通信に関してTD=0と仮定している。

分岐の方式として、分岐制御ユニットが他処理ユニットに対して真偽だけではなく分岐先アドレスも送信するかどうか、また分岐先アドレスを送信するとしたときには、条件分岐だけでなく無条件分岐の場合も送信するかどうかという選択がある⁽⁶⁾。本稿ではハードウェア量がもっとも小さくなる方式を採用した。すなわち、分岐制御ユニットは条件分岐時に真偽だけを他処理ユニットに通信し、無条件分岐に関しては各処理ユニットが独立に分岐する。

評価に使用したプログラムは下記の6個であり、UNIX上に記述されたシミュレータで実行時間を測定する。

Bubble	バブルソート
Quick	クイックソート
Puzzle	パズル「Knight & Rook」
Fibonacci	フィボナッチ数列計算
Matrix	行列積計算
Sieve	素数計算

4. シミュレーション評価結果

4.1 実行命令の分布

各命令の出現分布に関するシミュレーション結果を表1に示す。同表には、各処理ユニットの命令に関して、設定した実行時間、6つのプログラムに関する相対出現頻度(%)、実行時間と相対出現頻度の積である重み付き実行時間が示されている。各ユニットごとの重み付き実行時間の小計

49.9, 54.6, 28.3を加算した値132.8は、PNプロセッサで実行された命令の実行時間の平均を表している。同表の小計欄より、演算ユニットが実行する命令の相対出現頻度44%は、転送ユニットの実行する命令の相対出現頻度31%の約1.4倍あるものの、重み付き実行時間はほぼ同程度であるので両処理ユニットの稼働率はほぼ等しいことがわかる。

表1 各命令の動的出現頻度と平均実行時間

ユニット名	命令	実行時間	出現頻度	平均時間
転送ユニット	LOAD	200	8.02	16.04
	STORE	200	4.89	9.78
	JUMP	100	4.29	4.29
	BRANCH	150	10.10	15.14
	CALL	200	0.37	0.74
	RETURN	200	0.47	0.93
	NOP	100	2.92	2.92
	小計		31 %	49.9
演算ユニット	ADD	100	19.98	19.98
	SUB	100	2.82	2.82
	MUL	400	1.81	7.24
	DIV	400	0.11	0.44
	MOVE	80	4.96	3.97
	JUMP	100	4.00	4.00
	BRANCH	150	9.27	13.90
	CALL	200	0.37	0.74
	RETURN	200	0.47	0.93
	NOP	100	0.55	0.55
	小計		44 %	54.6
分岐制御ユニット	EQUAL	80	0.84	0.67
	NOTEQ	80	0.73	0.58
	GREAT	80	1.30	1.04
	GREQ	80	3.52	2.82
	LESS	80	1.12	0.89
	LSEQ	80	2.59	2.07
	JUMP	100	4.00	4.00
	BRANCH	150	9.27	13.90
	CALL	200	0.37	0.74
	RETURN	200	0.59	1.18
	NOP	100	0.39	0.39
	小計		25 %	28.3

分岐制御ユニットに関しては、その稼働率があまり高くない時には、他処理ユニットに先行して分岐条件を判断し、その結果などを転送することにより、他処理ユニットの分岐に関する処理を軽減化することができる⁽⁷⁾。逆に、分岐制御ユニットの稼働率が1に近づくと、他処理ユニットを

待ち状態にさせる場合が多くなることが考えられる。よって、PNプロセッサの理想的な負荷分散の状態は、転送ユニットと演算ユニットで負荷が等しく、分岐制御ユニットはそれに比較して負荷が小さめであることである。表1によると、分岐制御ユニットの重み付き実行時間の小計は転送ユニットの約57%(28.3/49.9)、演算ユニットの約52%(28.3/54.6)であるが、分岐制御ユニットの分岐条件の先行判断による他処理ユニットの処理軽減化を考慮すると、好ましい方向にあると考えられる。

4.2 抽出並列度

プログラムを実行したときに抽出される平均並列度を図5に示す。各命令の実行時間を個別に変化させて測定した。各命令に対して並列度は同様の变化を示すため、典型例として、LOAD/STORE命令の実行時間を横軸にとっている。平均並列度は1.3から2.3までの間にあることが示されている。

平均並列度は、命令の種類間に存在する非均一性⁽⁸⁾の変化により、単調な傾向は示さず、途中までは増加し、ある点で減少に転ずるという特性が5つのプログラムで見られる。これは、LOAD/STORE命令を実行する転送ユニットの稼働率が他処理ユニットの稼働率に比べて十分小さいうちは、実行時間の増加に従って平均並列度は増加し、ある点で最大値をとるが、それ以降は転送ユニットの稼働率の増加に従って相対的に他処理ユニットの稼働率が小さくなっていくために減少するからである。Bubbleに関しては平均並列度が増大することがない。これは、LOAD/STORE命令の実行時間を20以下と設定したときでも、データ転送ユニットの稼働率が約80%であり、他のプログラムにおける稼働率(17から45%)に比較して十分大きいからである。

Sieve, Fibonacci, QuickではLOAD/STORE命令の

実行時間が200付近で平均並列度は最大値をもつ。したがって、命令実行時間がこの付近にあるとき、負荷分散により並列度が適切に抽出されているといえる。本シミュレーションでは、LOAD/STORE命令の実行時間の標準設定値は200であり(表1)、この設定は妥当であるといえる。bubble, puzzleのように実行時間が0に近いところで並列度が最大になるプログラムに関しては、データキャッシュを搭載することにより、より効率的な負荷分散がなされることが考えられる。

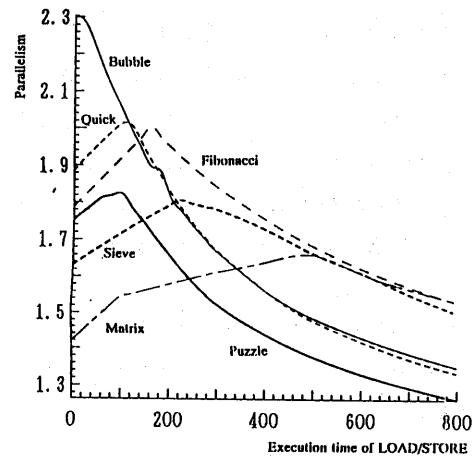


図5 LOAD/STORE命令実行時間と並列度

4.3 トークン通信

各プログラムにおける、トークン通信時間(T_c)の実行時間に対する影響を図6に示す。通信最適化時間は0としている($T_s=T_r=0$)。各プログラムとも、トークン通信時間の増加に従い線形に実行時間は増加する。グラフの傾きは、所要時間の増大がプログラム実行時間に直接影響するようなパス(クリティカルパス)上の通信回数に比例する。この通信回数は6つのプログラムのうちSieveが最大でMatrixが最小であることが知れる。

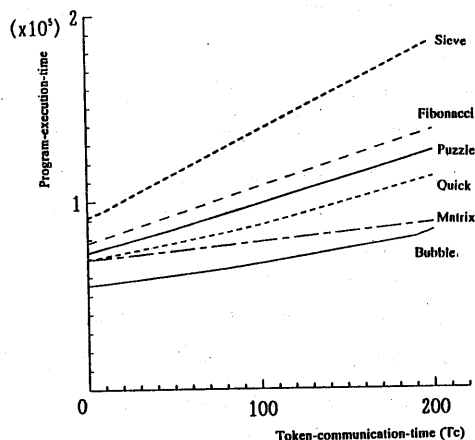


図6 トークン通信時間の影響

プログラム実行時間に対する、トークン送受信タイミングをずらすことによる最適化の影響を図7に示す。横軸は通信最適化時間 (T_r+T_s) を表している。この値が大きいくほど、送信側と受信側の命令がオーバーラップして実行される時間が大きくなる ($T_c=0$ とする)。最適化による効果が傾きとして表されている。この傾きも図6と同じようにクリティカルパス上にある通信の回数に比例している。通信最適化時間は命令や命令間の関係によって異なるが、命令フェッチ、デコード、PC更新、実行段階の一部などの処理を含んでいる。この時間は60から70程度であると考えられる。このことと図7より、PNプロセッサはタイミング最適化により、5から23%程度の速度向上がはかれると考えられる。このような最適化は、VLIWのようにコンパイル時に完全に命令単位のスケジューリングが確定するプロセッサでは困難なものである。

図7のどのプログラムについても傾きが変わる点が存在する。これは、クリティカルパス上にある命令間の通信最適化時間を減らして（実質通

信時間を減らして）いく場合、その時間がある程度まで小さくなるとそのクリティカルパスが移動し、結果として新しいクリティカルパス上の通信回数が減るため、プログラム実行時間の減少率が小さくなるからである。

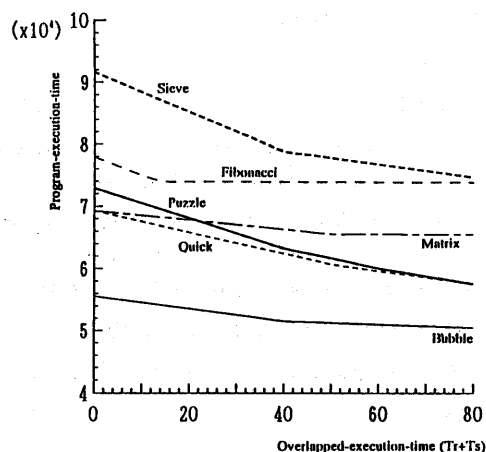


図7 送受信命令のオーバーラップ実行の効果

5. むすび

複数命令流を有する機能分割型スーパースカラプロセッサであるPNプロセッサをシミュレーションにより評価した。命令の種類による分割については、転送ユニットと演算ユニットの負荷はほぼ等しいと考えられる。分岐制御ユニットは負荷は低めであるが、分岐条件の先行判断を行うことを考慮すると好ましい方向にあると考えられる。また、サンプルプログラムの実行で抽出される並列度は1.3から2.3までの範囲にあることが示された。サンプルプログラム以外でもこの値を中心とした値をとることと思われる。

トークン通信時間に関しては、通信時間増加の影響は小さくないが、PNプロセッサに採用した高速通信方式のため通信時間は極めて小さいと考えられる。しかも、トークン通信のタイミングを

最適化することにより実質通信時間TDを負に見積もることが可能である。このとき、トークンを送信する命令と受信する命令がオーバーラップして実行され、このために5から23%程度の速度向上がなされる。

現在、各処理ユニットのパイプライン化などを含む詳細な設計や本プロセッサで採用した同期方式の一般化⁽⁸⁾などを行っている。

参考文献

- (1) M.D. Smith, M.S. Lam and M.A. Horowitz: " Boosting Beyond Static Scheduling in a Superscalar Processor ", Proceedings of the 17th Annual International Symposium on Computer Architecture, pp. 344-354 (1990).
- (2) 曾和将容、有田隆也、河村忠明、高木浩光 : 機能分割型プロセッサによる複数命令流実行方式、信学論文誌 D-I、Vol. J73-D-I、No. 3、pp. 280-285 (1990).
- (3) J.A. Fisher: " Very long instruction word architecture and the ELI-512 ", Proc. of 10th International Symposium on Computer Architecture, pp. 140-150 (1983).
- (4) 高木浩光、曾和将容、有田隆也 : P N コンピュータに向けた命令セットアーキテクチャ、信学技報、CPSY-89-8 (1989).
- (5) "Motorola MC68000 16 bit microprocessor user's manual", MOTOROLA INC.(1980).
- (6) 有田隆也、高木浩光、河村忠明、曾和将容 : P N プロセッサにおけるフロー制御方式について、情処学第 39 回全大、pp.1896-1897 (1989).
- (7) 河村忠明、曾和将容 : P N コンピュータにおける分岐処理：信学技報、CPSY88-36 (1988).
- (8) NP. Jouppi, "The Nonuniform Distribution of Instruction-Level and Machine Parallelism and Its Effect on Performance", IEEE Trans. on Computers, Vol. 38, No. 12, pp. 1645-1658 (1989).
- (9) 高木浩光、有田隆也、河村忠明、曾和将容 : 問題が持つ先行関係のみを保証する高速な静的実行順序制御機構の構成法、並列処理シンポジウム JSPP'90 論文集、pp. 57-64 (1990).
- (10) 田崎明久 : P N コンピュータのアーキテクチャに関する研究、群馬大学情報工学科昭和61年度卒業論文(1987).
- (11) 有田隆也、伊藤広明、高木浩光、曾和将容 : P N プロセッサの機能分割とトークン通信に関する評価、情処学第 41 回全大(6)、pp. 145-146 (1990) .
- (12) 有田隆也、高木浩光、曾和将容 : P N スーパー标ラプロセッサの命令実行制御方式の評価、電子情報通信学会技術研究報告、CPSY90-84, PP. 9-14 (1990).