# 汎用マイクロプロセッサを用いたDatarol-IIプロセッサエレメントにおける細粒度スレッド処理機構

富安, 洋史
九州大学システム情報科学研究院知能システム学部門

川野, 哲生
九州大学システム情報科学研究院知能システム学部門

谷口, 倫一郎
九州大学システム情報科学研究院知能システム学部門

雨宮, 真人
九州大学システム情報科学研究院知能システム学部門

https://doi.org/10.15017/5781

出版情報：九州大学大学院総合理工学報告. 16 (3), pp.331-338, 1994-12. 九州大学大学院総合理工学研究科
バージョン：
権利関係：

九州大学大学院総合理工学研究科報告
第16巻 第3号 331-338頁 平成6年12月

Engineering Sciences Reports, Kyushu University
(KYUSHU DAIGAKU SOGORIKOGAKU KENKYUKA HOKOKU)
Vol. 16, No. 3 pp. 331 - 338 DEC. 1994

# 汎用マイクロプロセッサを用いたDatarol-Ⅱプロセッサエレメントにおける細粒度スレッド処理機構

富安洋史・川野哲生
谷口倫一郎・雨宮真人

(平成6年8月31日 受理)

# Fine Grain Multi-Thread Processing in Datarol-Ⅱ Processor Element Using a Off-The-Shelf Micro Processor

Hiroshi TOMIYASU, Tetsuo KAWANO,
Rin-ichirou TANIGUCHI and Makoto AMAMIYA

In massively parallel processing, one of the most critical issues is the latency problem caused by remote memory accesses and remote procedure calls. To solve the problem, we have been developing a Datarol-Ⅱ processor, which hide the latency efectively by sophisticated fine-grain multi-thread processing.

Although the Datarol-Ⅱ processor requires a great cost of its development, we can significantly reduce the development cost by replacing the thread execution unit of a Datarol-Ⅱ processor with a off-the-shelf micro processor and some additional hardware for context switching. In this paper, we present the design and the performance evaluation of a Datarol-Ⅱ processor based on a off-the-shelf micro processor.

## 1. はじめに

超並列計算における問題のひとつは，プロセッサ間通信によるレイテンシである．このレイテンシを隠蔽するためには細粒度のプロセスを処理の単位とし，レイテンシが生じる場合には他の実行可能な細粒度プロセスに切替えてプロセッサのスループットを保つ手法が有効である[1)2)3)5)]．

我々は細粒度スレッド処理に適したDatarol-Ⅱと呼ぶ並列計算機の設計開発を行ってきた[1)4)]．Datarol-Ⅱはデータフロー方式をより最適化したDatarolアーキテクチャーをもとに設計され，効率的なマルチスレッド処理の実現を目視したマシンである．

Datarol-Ⅱではスレッドと呼ぶ細粒度の処理単位の概念を導入してDatarolアーキテクチャーのハードウェアコストを軽減している．ここで言うスレッドは同期なしに実行できる命令列であり，排他的に実行されるためプログラムカウンタによって制御される．

Datarol-Ⅱはこの細粒度スレッド実行モデルを効率よく実行することを目的としており，特にコンテクストの切替によるオーバーヘッドの削減に主眼が置かれている．コンテクストの切替に伴うメモリアクセスを隠蔽するため，レジスタの自動ロードストア機構を持ち，メモリアクセスとALU演算をオーバーラップして行うことにより，オーバーヘッドを削減する．

ところで，Datarol-Ⅱプロセッサエレメントは専用プロセッサを開発する必要があるため，設計コストの増大が問題となる．しかし，Datarol-Ⅱはスレッド内部の実行にプログラムカウンタを用いるため，このスレッド内部の処理をノイマン型プロセッサを用いて行うことができ，汎用マイクロプロセッサを用いることによって設計コストの増大を押えることができる．

汎用マイクロプロセッサは細粒度スレッド処理のための機構を持たない，コンテクスト切替時のメモリアクセスがオーバーヘッドとなる，といった問題があるが，Datarol-Ⅱで行われるスレッド同期処理や実行可能スレッドの待ち行列の管理を外付け回路で行うことによって実現し，コンテクスト切替時のメモリアクセスはスーパースカラでALU演算とメモリアクセスをオーバーラップすることによってオーバーヘッドを削

*情報システム学専攻博士課程
**情報システム学専攻

減することができる．汎用マイクロプロセッサにこれらの回路を付加することによって完全に専用設計としたものと，同等の能力をもつ Datarol-Ⅱを構成できれば，大幅なコストダウンが可能となる．

また，汎用マイクロプロセッサの速度向上は常に専用プロセッサを持った並列計算機を脅かしているが，汎用マイクロプロセッサを用いることによって Datarol-Ⅱもその速度向上の恩恵をうけることができる．

本稿では，Datarol-Ⅱアーキテクチャーを低コストで実現するために，汎用マイクロプロセッサを用いた Datarol-Ⅱプロセッサエレメントの基本設計を行い，すべてを専用設計とした Datarol-Ⅱと，コンテクスト切替のオーバーヘッドについて比較を行った．

## 2. プロセッサエレメント（PE）設計の方針

### 2.1 Datarol プロセッサエレメント

データフローモデルは，プログラムに内在する並列性の抽出が自動的に行なえるため，並列計算モデルに適している．しかし，ペアオペランドの同期のコストが大きい，メモリの概念がないため無駄なデータのコピーが発生してしまうといった問題点を抱えている．我々はデータフロー方式の問題点を解決するため Datarol と呼ぶアーキテクチャーの提案を行ってきた．Datarol はデータフロー方式にレジスタを導入し，data-flow graph から冗長なフロー制御を削除した Datarol graph を実行のモデルとしている．各関数インスタンス（以下単にインスタンスという）はそれぞれ固有のレジスタセット（論理レジスタとよぶ）を持ち，インスタンス内のデータ受渡しにはこのレジスタを用いる．

Datarol はデータフローから冗長なデータのコピーとフロー制御を削除したが，循環パイプライン方式であるため，並列度が低い場合にパイプラインに空きが生じやすい．専用メモリを多く持っているため，ハードウェアコストが大きい．などの問題があった．

### 2.2 スレッド概念の導入

Datarol-Ⅱではこの問題を解消するために thread 実行概念を導入している．**Fig. 1** に Datarol-Ⅱにおける thread 実行概念を示す．

Datarol-Ⅱで言う thread は thread 内部で必要とされるデータ（引数）が揃った時点で発火され，発火後

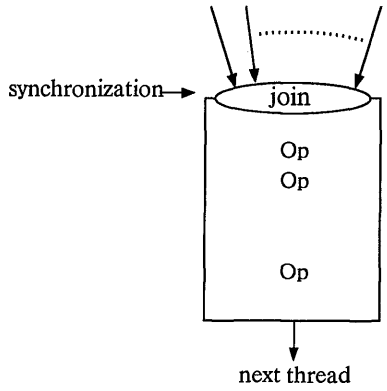


**Fig. 1**　Thread model

は thread 内部の命令列はノイマン型プロセッサと同様にプログラムカウンタを用いて逐次的に実行される．thread 内の命令を全て実行すると自動的に thread は終了し，他の実行可能な thread が起動される．

### 2.3 Native Datarol-Ⅱ

Datarol-Ⅱ（以下 Native Datarol-Ⅱとする）は thread 実行概念を効率良く実装することを目的として設計されている．**Fig. 2** に Native Datarol-Ⅱの構成を示す．Datarol-Ⅱでは thread を最小単位とした実行を行うため，FU（Function Unit）は thread 内の処理を逐次的に行う．thread の同期には同期カウンタを用い，AC（Activation Controller）が自動的にカウンタの更新を行う．

また，Datarol-Ⅱではコンテクスト切替を高速化す

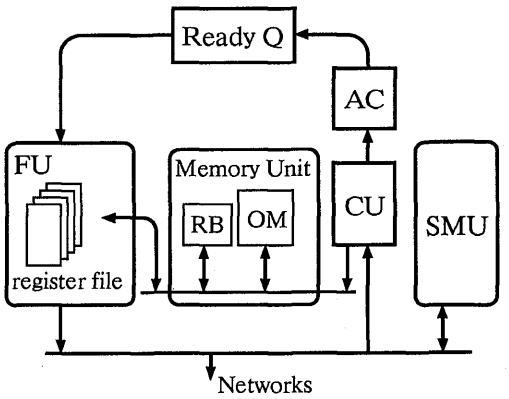


**Fig. 2**　Native Datarol-Ⅱ

るため，自動的にレジスタのロードストアを行う機構を組み込んでいる．Datarol ではインスタンスごとに論理レジスタを持つが，実際には高速なレジスタは FU 内部に少数しか用意できないため，コンテクストの切替にともないレジスタの入れ替え必要となる．レジスタの自動ロードストア機構によって算術論理演算とレジスタの入れ替えがオーバーラップして行われるため，コンテクスト切替時間が削減される．

また，論理レジスタを標準的な DRAM で構成することを想定し，RB (Register Buffer) と呼ぶ少容量の高速メモリを用いてメモリの階層化を行っている．

### 2.4 汎用マイクロプロセッサを使用した Datarol-Ⅱ

Native Datarol-Ⅱは細粒度 thread 実行モデルのために最適化されているが，専用プロセッサを前提としているため，PE 全てをはじめから設計する必要がある．また，クロックを上げて高速化を行う場合も専用プロセッサ開発のためのテクノロジー条件を含めて検討する必要があり，設計コストが上昇する．汎用マイクロプロセッサが使用可能であれば，外付けの細粒度スレッド処理を高速化する回路のみを設計し直せば良いため，大幅に負担が威厳される．

従来の汎用マイクロプロセッサはコンテクスト切替のオーバーヘッドが大きく，細粒度 thread 実行には不向きてあった．しかし，近年汎用のマイクロプロセッサの性能が大きく向上し，特にスーパースカラによりコンテクスト切替で生じるロードストアを他の命令とオーバーラップできるようになった．

thread 同期のためのカウンタのチェックや更新，あるいはインスタンスへの論理レジスタの割り当てなどは依然としてオーバーヘッドとなる．しかし，これらを外付け回路で処理することができれば，Native Datarol-Ⅱとの性能差を縮小できる．

もし，汎用マイクロプロセッサに簡単な外付けの回路を付加することにより十分な性能のプロセッサエレメントを実現することができれば，コストの点で大きく有利になる．

また，汎用マイクロプロセッサは年々高速化しているため，マイクロプロセッサを使用すればマイクロプロセッサの世代交替に応じて高速化をはかることができる．

### 2.5 解決すべき問題

汎用マイクロプロセッサを用いて Datarol-Ⅱを構成する場合に問題となるのは以下の3点である．

- thread 切替時に生じるコンテクスト切替のオーバーヘッド．

  Native Datarol-Ⅱはレジスタの自動ロード機構を備えている．

  汎用マイクロプロセッサを用いた場合は明示的なロードストア命令を入れる必要がある．このロードストアをスーパースカラでどの程度隠蔽できるかが焦点である．
- thread の発火制御機構の実現法．

  Native Datarol-Ⅱと同等の発火制御機構を外付け回路で実現する必要がある．

  Datarol-Ⅱではパケットの形で発火制御機構へ情報が渡されるが，汎用マイクロプロセッサでは自由にパケットを作りにくいためメモリ書き込み等をつかって同等の機構を実現する．
- 外付け回路のハードウェアコスト．

  汎用プロセッサを用いても外付け回路が複雑になりすぎては意義がうすれるため，単純な構成とする．また，将来のマイクロプロセッサの速度向上に対応するためにも，単純な構成であることが必要である．

以上の点に注意して汎用マイクロプロセッサを用いた Datarol-Ⅱの基本設計を行い，コンテクストの切替のオーバヘッドを Native Datarol-Ⅱと比較した．

## 3. PE の構成

### 3.1 構成の概要

**Fig. 3** に汎用マイクロプロセッサを用いた Datarol-Ⅱ PE の構成を示す．ここで，TAC (Thread Activation Controller) は thread 発火制御機構，RQ (Ready Queue) は実行可能な thread を記憶する FIFO, ICU (Instance Control Unit) はインスタンス割り当て時の論理レジスタの管理を行う．TAC, RQ および ICU はメモリ上にマッピングされており，CPU はメモリアクセス命令を用いてこれらを操作する．

CPU，メモリ，TAC 等の各構成要素がバスを介して接続された単純な構成となっている．thread 発火制御に使用されるカウンタのみは専用メモリで構成さ

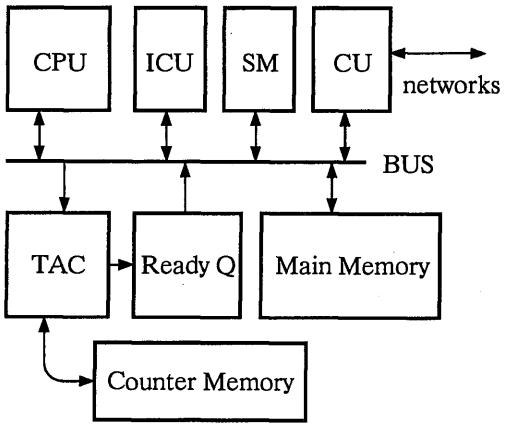


**Fig. 3**　Datarol-Ⅱ PE using a general purpose processor

れ，TAC によって管理される．同期カウンタ以外には専用メモリを使用せず，配線や，I/O pin のコストを大きくしないようにした．

必要とする専用メモリは同期カウンタのみであり，バス以外に必要とする配線も少ないため，コストを低く押えることができるとともに，高速化にも有利である．

汎用マイクロプロセッサを用いた実装方式と Native Datarol-Ⅱと大きく違う部分は，細粒度 thread 処理の高速化を行うために汎用マイクロプロセッサに付加した TAC である．

その他は CPU とのインターフェースに合わせるだけで Native Datarol-Ⅱ と同じ設計とすることができる．また，PE 間ネットワークとのインターフェースからネットワーク側はまったく同一の仕様とすることができる．

### 3.2 CPU

プロセッサエレメントに使用する CPU には，以下のような条件が求められる．

- 特殊目的でなく，一般的に使用されているもの．
  安価に入手可能であること．継続して使用可能であること．次世代のプロセッサに無理なくつながること．などが重要である．
- 高性能であること，特にロードストアの性能が良いこと．
  コンテクスト切替にともなうロードストアを高速に行うものが望ましい．

これらの条件を満たす CPU の一つとして今回は Intel 社の Pentium[6] を想定して評価した．Pentium は PC 用として広く使われており一般性に関しては全く問題が無い．また，性能的にも最新のマイクロプロセッサとして遜色ない．特に 2 つの完全な整数演算ユニットを持ち，ロードストアパイプも 2 つあることから，コンテクスト切り替えが頻繁に起こる細粒度 thread 処理においては有利であると思われる．

他に一般に使用されているマイクロプロセッサとして Super Sparc や R4000 等についても検討したが，これらはロードストアパイプが 1 つであるため，Datarol-Ⅱのようにコンテクスト切替が頻繁に起こる場合は Pentium よりも不利になる．

ロードストアパイプを 2 つ以上持つマイクロプロセッサはまだ一般的ではない．しかし，スーパースカラ度が増え，同時実行可能な ALU 演算命令が増えるに従い相対的にロードストアパイプが不足する．したがって今後ロードストアパイプを複数持つプロセッサが一般的になっていくと思われる．

### 3.3 メモリマップ

汎用マイクロプロセッサを用いた Datarol-Ⅱでは TAC への情報として CPU がバス上に出力するアドレスを利用しているため，メモリマップが重要である．

**Fig. 4** にメモリマップを示す．Data_Area は各インスタンスが変数の格納に使用する領域である．各インスタンスには固定長の frame（Datarol での論理レジスタ）が割り当てられる．この frame のアドレスがそのままインスタンス名として空き領域の管理などに使用される．Code_Area は命令を格納する．

TAC_interface_Area は Data_Area と同じサイズを割り当てられ，この領域に CPU が書き込んだ場合は TAC への命令発行として取り扱われる．従って実際

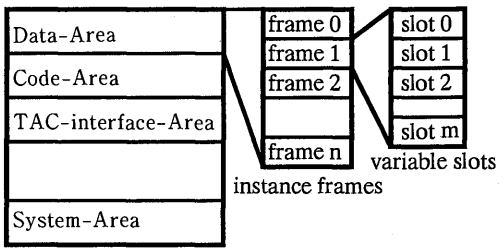


**Fig. 4**　Memory map

にはメモリは実装されない．TAC はバス上のアドレスからアクセス先のインスタンス名（単純にオフセット分を引くことによって得られる）を知り，CPU からのデータで対応する同期カウンタ及び，発火すべき thread の開始アドレスを得る．

その他 ICU, RQ, CU 等は memory mapped I/O として System-Area にマッピングされている．

### 3.4 thread 発火制御機構 (TAC)

TAC による thread の発火制御は以下のように行われる．

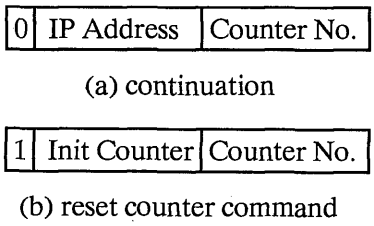


**Fig. 5**　Continuation

step 1　continuation 情報の書き込み

CPU は引数をメモリに書き込むと同時に TAC に thread 起動情報（以下 continuation 情報と呼ぶ）を書き込むことによって TAC を起動する．

step 2　同期カウンタのロード

**Fig. 5 (a)** に continuation 情報のフォーマット，**Fig. 6** にカウンタ memory の構成を示す．

continuation 情報にはカウンタ No. が含まれており，同期カウンタのアドレスは CPU がバス上に出力したアドレスの上位とカウンタ No. から得られる．TAC は continuation 情報の書き込みによって起動し，同期カウンタをロードする．

もし，continuation 情報によって書き込み先が1つの引数しか必要としない場合はカウンタのロードは行わない．

step 3　thread 発火，同期カウンタの書き戻し

カウンタをチェックして同期が成立すれば thread の発火を行う．TAC は RQ に thread の開始アドレスとインスタンス名を出力する．thread の開始アドレスは continuation 情報から得られ，インスタンス名（frame No.）は CPU の出力するアドレスから得られる．

同期が成立しなければ同期カウンタを減算して書き戻しを行う．

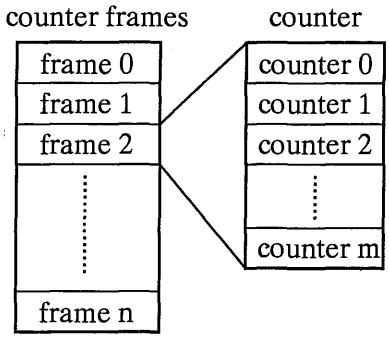


**Fig. 6**　Counter memory

同期カウンタの初期化は CPU が TAC に対して初期化コマンドを書き込むことによって行う．**Fig. 5 (b)** に初期化コマンドを示す．TAC は CPU に代わって同期カウンタを管理することによって thread の同期のオーバーヘッドを軽減する．CPU はカウンタの減算，書き戻し，thread の発火（RQ への出力）などを行う必要が無い．オーバーヘッドは同期カウンタの初期化と continuation 情報の書き込みのみである．

### 3.5 RQ (Ready Queue)

RQ は実行可能な thread を蓄える FIFO である．CPU の負担を減らすため **Fig. 7** に示すように Ready Q の先頭は常に特定番地にマッピングされ，各thread の終了時に CPU は RQ の先頭のアドレスを計算せず常に特定番地をアクセスして次の thread に jump する．

RQ に実行可能な thread がない場合は RQ は dummy の frame No. とシステムで用意された dummy の thread の開始番地を先頭にセットする．この thread

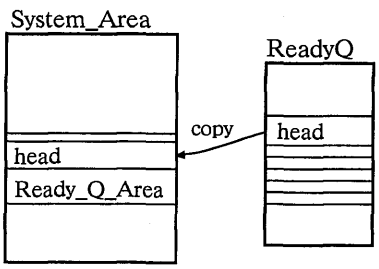


**Fig. 7**　Ready Queue

はすぐに実行を終了し RQ の先頭を読む処理を行う．したがって，RQ が空であるかどうかチェックするコードは必要としない．機械的に RQ の先頭（特定番地にマッピングされているので即値で埋め込むことができる）を読むコードを各 thread の終りに付加するだけでよい．Pentium の場合は 2 命令で thread の切替を終了することができる．

## 4. 性 能 予 測

我々は汎用プロセッサを用いて構成した Datarol-Ⅱの性能評価のため，ソフトウェアシミュレータを作成し，Native Datarol-Ⅱと汎用マイクロプロセッサを用いた Datarol-Ⅱ及び全く付加回路を設けずソフトウェアのみで thread の発火制御を行うものについて，特にスレッド切り替えのオーバーヘッドに関して比較を行った．

これらのシミュレータはクロックレベルで動作し，CPU 部分は Pentium を想定して Pentium のサブセットの動作ができるものを用意した．メモリアクセスのレイテンシはいずれも 3 clock である．また，Pentium 版のデータキャッシュサイズ及び Native Datarol-Ⅱの RB サイズはいずれも 16 KByte である．実際の Native Datarol-Ⅱは専用プロセッサを開発する必要があり，クロックは専用プロセッサによって決まる．Native Datarol-Ⅱのような，専用プロセッサは汎用マイクロプロセッサのように大きなコストをかけて設計できないため，クロックを上げることが難しい．したがって，Native Datarol-Ⅱの動作クロックは Pentium より低くなると思われるが，クロック速度の正確な見積りが非常に難しいため以下の評価では同一とした，使用した例題は以下の 2 つである．

（i） fib：フィボナッチ数を再帰的に求める
　thread 長が非常に短い．汎用プロセッサ版ではコンテクスト切替のオーバーヘッドが大きく現れる．関数展開の幅が大きいためキャッシュミスヒットが起こりやすい．汎用プロセッサ版にとっての最悪値として使用できる．

（ii） Queen：N-queen を全解検索で求める
　thread 長は一般的な長さに近付く．検索問題の典型的な例であり，実際の性能はこちらに近いと思われる．

以下に，2 つの例題の性能予測結果について示す．

**fib**

fib の thread 長はロード命令を含めない場合は 4 命令程度，ロード命令を含めて 6 命令程度であり，非常に短いため fib は汎用プロセッサを用いた実装方式にとって最悪値に近い結果となる．一般的なプログラムにおいては，少なくとも fib より性能が落ちることはないと思われる．**Fig. 8** に fib の実行時間の比較を示す．

thread 長が短いためコンテクスト切替のオーバーヘッドが大きく出ている．特に問題サイズの小さいところで顕著で，Pentium を用いた実装方式の性能は Native Datarol-Ⅱの0.3倍程度しかない．問題サイズが大きくなるにつれ，性能差は小さくなり，0.8倍程度にまで改善される．また，ソフトウェアのみで発火制御を行った場合の実行時間は一貫して付加回路を設けて高速化した場合の 2 倍程度となっている．

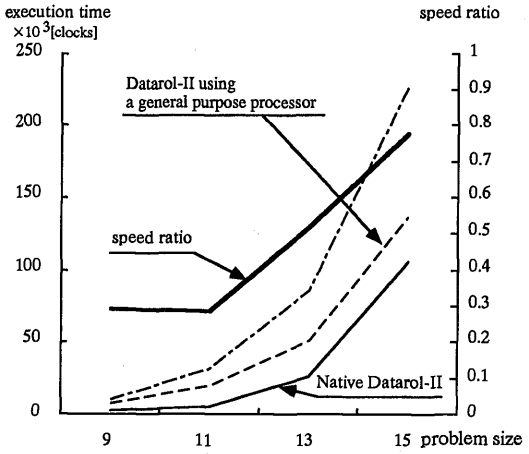


**Fig. 8**　Execution times of fib(n)

問題サイズが大きな部分で Native Datarol-Ⅱとの性能差が小さくなるのは Native Datarol-Ⅱでキャッシュに相当する RB へのロード時のオーバーヘッドが原因である．

**Table 1** に Pentium を用いた実装方式におけるキャッシュヒット率と Native Datarol-Ⅱにおける RB のスワップ時間を示す．Native Datarol-Ⅱの RB は一般的なキャッシュと違いレジスタセットを自動的にロードする構成になっており，単純にキャッシュヒット率を算出することができないため，表中に RB スワップ時間を全実行時間中に占める RB の入れ替えに要した時

**Table 1**　Data cache hit ratio in fib(n)

| problem size | fib(9) | fib(11) | fib(13) | fib(15) |
|---|---|---|---|---|
| Data cache hit ratio [%] | 79.78 | 74.62 | 69.93 | 68.41 |
| RB swap time [%] | 0.0 | 0.0 | 34.7 | 47.5 |

間の割合で示している．

Native Datarol-Ⅱでは，fib (11) までは全くスワップが生じておらず，全てのデータはRB上に存在しているが，fib (13) 以降ではRBの容量が不足するためスワップが行われ始めており，同時に，Native Datarol-Ⅱ とPentium を用いた実装方式との性能差が大きく縮まっている．

これはNative Datarol-ⅡにおけるRBのミスヒットのペナルティが大きいためである．Native Datarol-Ⅱでは自動ロードの対象は論理レジスタセットとなっており，RBへのミスヒットが生じた場合は毎回16 wordのデータすべてをストア，ロードしている．

しかし，fibのthreadで必要なデータは2，3個程度しかなく，16 wordのデータのうち，ほとんどが無駄になっている．このため，先行読み込みを行ってもスレッド長が短いため自動ロードストア機構のスループット自身が不足し，RBのミスヒットが大きなペナルティとなっている．

対して，Pentium版では明示的にロードストア命令を挿入しているため，必要以上にロードストアを行うことがない．加えてPentiumにはデータキャッシュとメモリ間にラインフィルバッファがあり，キャッシュミスヒット時に読み込まれたデータは，ラインフィルバッファに読み込まれると同時に使用可能になるためラインサイズ分のデータを待つ必要が無く，キャッシュミスヒット時のペナルティが少なくなっている．Native Datarol-ⅡではRB上に必要な論理レジスタセットがない場合は，RB上に16 wordのデータ全てをロードするまで遅延が生じる．

### Queen

Queenのthread長さはロード命令を含めない場合は7～8程度，ロード命令を含めて11～12程度とやや長く，一般的な問題に近付く．したがって，一般的な問題における性能はfibよりもQueenの結果に近いと思われる．**Fig. 9** にQueenの実行時間を示す．

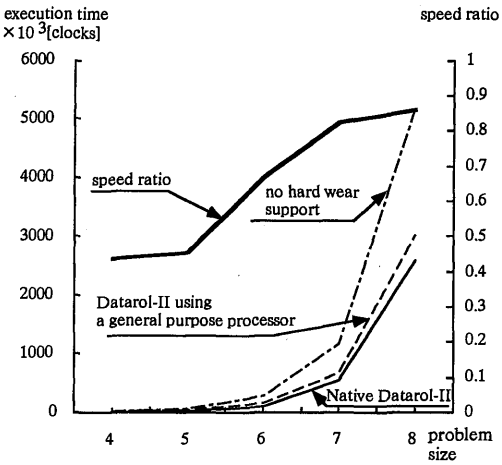


**Fig. 9**　Execution times of queen(n)

fibよりもthreadが長いためコンテクスト切替のオーバーヘッドは相対的に小さくなっており，性能比は0.42～0.85程度となっている．ソフトウェアでは発火制御を行った場合の実行時間はfibと同様に2倍程度である．

また，**Table 2** にQueen実行時のキャッシュヒット率を示すが，fibの場合と同様に問題サイズが大きくなってRBのスワップが現れはじめると性能差が小さくなり，問題サイズが大きいqueen (8) では0.85倍の性能となる．

**Table 2**　Data cache hit ratio in queen(n)

| problem size | queen (4) | queen (5) | queen (6) | queen (7) | queen (8) |
|---|---|---|---|---|---|
| data cache hit ratio [%] | 81.42 | 78.50 | 75.06 | 74.10 | 71.89 |
| RB swap time [%] | 0.0 | 0.0 | 29.6 | 41.6 | 43.9 |

全体としてQueen程度のthread長があれば実行時間は2倍以内におさまると考えられる．実際には汎用マイクロプロセッサは高いクロックで動作させることができるため，十分実用になるといえる．

## 5. 結　　論

我々の研究室では細粒度thread処理に適した

Datarol-Ⅱアーキテクチャを提唱してきた．しかし，Datarol-Ⅱは専用プロセッサを前提としているため設計コストが大きいことが問題であった．

本稿では，Datarol-Ⅱは汎用マイクロプロセッサに細粒度 thread 処理機構を簡単な外付け回路によって付加した構成で実現でき，大幅なコストダウンが可能であることを示し，その性能予測を行った．

この結果，thread 長が N-queen のプログラム程度であれば同一クロック速度においても実行時間は Native Datarol-Ⅱの2倍以下になることが予測できた．また，外づけ回路で発火制御を行うことによってソフトウェアのみによる発火制御に対して2倍の高速化ができることを示した．

実際には汎用マイクロプロセッサは高いクロックで動作するため，汎用マイクロプロセッサを用いた Datarol-Ⅱは十分に実用的な速度で動作する．したがって，専用プロセッサ設計によるコスト増大をまねくことなく，細粒度 thread 処理に適したプロセッサエレメントを構成可能であることが確認できた．

## 参 考 文 献


1) M. Amamiya and R. Taniguchi, "Datarol: A Massively Parallel Architecture for Functional Language", Proc. SPDP, pp. 726-735, (1990).

2) R. S. Nikhil, G. M. Papadopoulos and Arvind, "＊T: A Multithred Massively Parallel Architecture", Proc. 19 th ISCA, pp. 156-167, (1992).

3) W. J. Dally, A. Chien, S. Fiske, W. Horwat, J. Keen, M. Larivee, R. Lethin, P. Nuth and S. Wills, "The J-Machie: A Fine-grain Concurrent Computer", Proc. 11 TH IFIP, pp. 1147-1153, (1989).

4) 川野，日下部，谷口，雨宮，"並列計算機 Datarol-Ⅱのプロセッサエレメントの構成"，情報処理学会研究会報告，93-ARC-101，pp. 137-144，(1993)．

5) 児玉，坂井，山口，"データ駆動シングルチッププロセッサ EMC-R の動作原理と実装"，情報処理学会論文誌，32，7，pp. 849-858，(1991)．

6) "Pentium プロセッサデータブック"，インテルジャパン株式会社．