細粒度マルチスレッド処理向けプロセッサ Datarol-II の構成とその評価

川野哲生[†] 日下部茂[†]
谷口倫一郎[†] 雨宮真人[†]

超並列計算機の設計において，もっとも大きな問題の1つにプロセッサ間通信やメモリアクセスに伴うレイテンシ問題がある．マルチスレッド処理によるレイテンシ隠蔽は本問題に対する有効な解決手段である．効果的なマルチスレッド処理を行うためにはプロセッサに高速なコンテクストスイッチ能力が必要とされる．しかしながら従来の RISC 型のプロセッサでは，スレッドの切り替えに伴うレジスタの退避と回復のためのメモリアクセスがオーバヘッドとなり，細粒度マルチスレッド処理を効率的に実行することは困難である．本論文では細粒度マルチスレッド処理向きプロセッサ Datarol-II を提案する．本プロセッサはデータ駆動方式を最適化した Datarol にスレッド実行を導入し一般的な RISC プロセッサと同様のパイプライン処理および高速レジスタの利用を可能とした．また，自動レジスタロードストア機構によりコンテクストスイッチに伴うメモリアクセスを明示的なロードストア命令を用いずかつ通常の処理と並行して行うことにより細粒度処理におけるオーバヘッドを隠蔽する．さらに階層的なメモリシステムと負荷制御機構を導入し価格性能比に優れたメモリシステムを実現する．シミュレーションによる評価により，自動レジスタロードストア機構によるメモリアクセスオーバヘッドの隠蔽効果，優れた耐レイテンシ性能，負荷制御による効果的な階層メモリシステムの実現，が確認され，本プロセッサは超並列計算機用要素プロセッサとして有望であることが分かった．

Datarol-II : Fine-grain Multithread Processor Architecture

TETSUO KAWANO,[†] SHIGERU KUSAKABE,[†] RIN-ICHIRO TANIGUCHI [†]
and MAKOTO AMAMIYA [†]

Latency, which is caused by remote memory accesses and remote procedure calls, is one of the most serious problems in massively parallel computers. In order to eliminate the idle time of processors caused by the long latencies, processors must perform fast context switching among fine-grain concurrent processes. However, since conventional RISC processors are designed for long thread executions, they are inefficient in a fine-grain multithread execution. In this paper, we propose a processor architecture, called Datarol-II, which realizes efficient fine-grain multithread execution by performing fast context switching among fine-grain concurrent processes. In the Datarol-II processor, an implicit register load/store mechanism is embedded in the execution pipeline in order to reduce the memory access overhead caused by context switching. A two-level hierarchical memory system and a load control mechanism are also introduced in order to reduce local memory access latency. Simulation results show the followings: the implicit register load/store mechanism reduces context switching costs; the Datarol-II processor is tolerable for the long latencies; the load control mechanism reduces the memory access traffic enabling hierarchical memory system to work efficiently. By these results, it is shown that the Datarol-II processor is suitable for a processor element of massively parallel computers.

1. はじめに

超並列計算機の設計において，もっとも大きな問題の1つにプロセッサ間通信やメモリアクセスに伴うレイテンシ問題がある．マルチスレッド処理によるレイテンシ隠蔽は本問題に対する有効な解決手段である．効果的なマルチスレッド処理を行うためにはプロセッサに高速なコンテクストスイッチ能力が必要とされる．データ駆動型の計算機は，プログラムに内在する並列性を実行時に自動的に抽出でき，またコンテクストスイッチをハードウェアにより高速に実行することができる等の並列処理に適した性質を持つ．データ駆動方式の第一の問題点はデータの待ち合わせ機構に連想記憶を必要とすることである．そこで，連想記憶を必要としないアーキテクチャとして，これまでMIT

† 九州大学大学院総合理工学研究科
Department of Information Systems, Graduate School of Engineering Sciences, Kyushu University

の Monsoon[6]，電子技術総合研究所の EM-4[8]，九州大学の Datarol[1]等が提案され，その有効性が確認されてきた．しかしながら，Monsoon や Datarol では命令レベルでのマルチスレッド処理に起因する逐次処理の非効率性が指摘された．一方，EM-4 では強連結モデルにより逐次処理効率の改善がなされたが，メモリ使用効率や高速メモリを大量に必要とする等の問題が残された．

近年の RISC 技術に代表される単体のマイクロプロセッサの性能向上は著しい．これは主に，パイプライン技術，高速レジスタの利用，キャッシュによるメモリアクセスレイテンシの低減によるものである．このような高性能マイクロプロセッサを用い，マルチスレッド処理を行う並列計算機として MIT の*T[5]等が挙げられる．しかし，従来のマイクロプロセッサではコンテクストスイッチのコスト，すなわちスレッドの切り替えに伴うレジスタの退避と回復のためのメモリアクセスがオーバヘッドとなり，細粒度のマルチスレッド処理においてはプロセッサの性能を十分に発揮できず，細粒度マルチスレッド処理を効率的に実行することは困難であるという問題がある．

我々はこれらの問題点を解決することを目的として Datarol にスレッド実行機構を導入した超並列計算機用要素プロセッサ Datarol-II を提案する．Datarol はデータ駆動方式に by-reference によるデータ参照を導入しデータフローグラフにおける冗長なフロー制御を削除し最適化したものである．Datarol-II ではさらにスレッド実行機構を導入することにより RISC 的な演算パイプラインや高速レジスタを導入し，逐次処理に対する性能改善や，クロック速度の高速化を行う．また，コンテクストスイッチに伴うメモリアクセスを自動的にかつ通常の命令実行と並行して行い，さらに階層メモリ構成により低レイテンシのメモリアクセスを提供し，高速なコンテクストスイッチを実現する．本論文では Datarol-II プロセッサについて，特にスレッドの実行制御機構，コンテクストスイッチに伴うメモリアクセスオーバヘッドの軽減手法，およびメモリ構成について提案し，その有効性に対する評価を示す．まず 2 章で本プロセッサの構成について述べ，3 章で本プロセッサの性能をシミュレーションにより評価する．また 4 章では細粒度並列処理を指向する他の研究とを比較し，本方式の有効性を検討する．最後に 5 章で結論を述べる．

2. Datarol-II プロセッサアーキテクチャ

我々はデータ駆動方式の問題点を解決するものとして，Datarol と呼ぶ実行モデルの提案を行ってきた[1]．Datarol では，各関数インスタンス（プロセスに相当，以後これを単にインスタンスと呼ぶ）ごとにそれぞれ固有の論理レジスタセットを持ち，インスタンス内の命令間のデータの受渡しにはこの論理レジスタを用いる．論理レジスタを導入することによりコンパイラで冗長なフロー制御を削除し最適化した Datarol と呼ぶマルチスレッドコントロールフローグラフを抽出する．我々はこれまでに Datarol グラフを直接実行するプロセッサ Datarol-I を設計・評価してきた[1),4),10),11)]．Datarol-I では各命令ごとにその継続命令が continuation として明示的に指定され，それに従って処理が進められ，複数のスレッドが命令単位でインタリーブ実行される．しかしながら，このような命令レベルの continuation 指定による実行方式は単一スレッドの実行レイテンシが大きく，また並列度の低いプログラムではパイプライン中にバブルを生じ実行効率の低下を招く問題がある．また，各命令ごとに必要とされる同期処理がパイプラインのボトルネックとなり，クロックの高速化を妨げる要因となる．

本論文で提案する Datarol-II プロセッサは Datarol-I プロセッサの問題点を解決すべく開発したものである．Datarol-II ではスレッド（途中でサスペンドすることなく実行されるコードブロック）を単位とした処理を行う．Datarol-II では Datarol-I 同様 continuation ベースによる実行もサポートされる．Datarol-II での continuation は単一の命令ではなくスレッドを指す．Datarol-II 用コンパイラは Datarol グラフからスレッドを自動的に抽出し，スレッド内のコードの最適化を行う．また，スレッド内の処理には高速なレジスタを用い，従来の RISC プロセッサに類似した演算パイプラインにより高速な処理を行う．スレッド実行化と高速レジスタの導入によりコンテクスト切り替えに伴うレジスタ値の退避や回復のためのメモリアクセスオーバヘッドを生じるが，Datarol-II プロセッサではコンテクストスイッチに伴うメモリアクセスを自動的にかつ通常の演算処理と並行して処理することによりこのオーバヘッドの隠蔽を行う．さらに，Datarol-II のメモリシステムには従来のキャッシュメモリに相当する階層構成を導入し，高速なメモリシステムを実現する．

2.1 Datarol-II プロセッサ構成

Datarol-II プロセッサ（PE）の構成を**図 1** に示す．PE は以下の機能ユニットから構成される．

- **Function Unit（FU）**：Ready Queue（RQ）から実行可能なスレッドを受け取り，スレッド内の命令

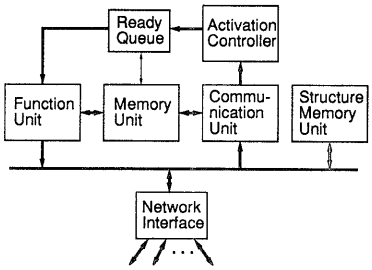


図1 Datarol-IIプロセッサ構成図
Fig. 1 Datarol-II Processor.

を逐次的に実行する．関数起動やリモートメモリアクセス等のパケットの発行も行う．

- **Memory Unit (MU)**：各インスタンスの論理レジスタの内容を保持する．
- **Communication Unit (CU)**：当該PE宛のパケットを受け取り，パケット内のデータをMUへ書き込む．同時にスレッド起動要求をActivation Controller (AC) へ送る．
- **Activation Controller (AC)**：CUからスレッド起動要求を受け取り，内部に用意されたMatching Memory (MM) を用いてスレッドの同期処理を行う．同期に成功した場合は，そのスレッドをRQへ送る．
- **Ready Queue (RQ)**：実行待ちのスレッドを格納する．
- **Structure Memory Unit (SMU)**：SMUではリモートメモリ参照のリクエストパケットを受理し，SMU内に設けられたメモリ (SM) を用い，I-structure機構による同期メモリアクセスを行い，リプライパケットの送出を行う．また，インスタンスの割り付け・回収や負荷制御も行う．
- **Network Interface (Net-IF)**：ルータ，パケットフォーマッタ等から成るネットワークインタフェース部．

Datarol-IIプロセッサでの処理は，FUでのプログラムカウンタを用いたスレッド内命令の逐次実行と，FU-CU-AC-RQから成る循環パイプラインでのcontinuation指定によるsplit-phase実行，の2つで行われる．

2.2 命令セット

Datarol-IIの命令セットを表1に示す．表中で，r_{S1}, r_{S2} はソース論理レジスタ名，r_D はデスティネーション論理レジスタ名を示す．Datarol-II命令は一般的なRISCプロセッサと同様のレジスタ-レジスタ形式の算術・論理演算に加え，インスタンス生成やリモートメモリアクセス等のパケット発行命令，およびスレッド起動命令からなる．各命令はterminateフラグをもち，このフラグがonである命令はスレッドの終端命令であることを示す．リモートメモリ読み出し命令等のレイテンシを伴う命令は結果値が利用可能となった時に起動する継続スレッド (continuation) の指定を行う．continuationは“−> (*dest, mc, join*)”の形式で指定され，*dest* が継続スレッドの開始アドレス，*mc* が同期カウンタ番号☆，*join* が同期数を示す．このcontinuation指定によりリモートメモリアクセス等の結果到着後に *mc* で指定される同期カウンタを用い *join* 数の同期が行われ，同期成功時に *dest* で指定されるスレッドが起動される．また，スレッドを明示的に起動するstart命令も用意されている．

表1 Datarol-II命令セット
Table 1 Datarol-II instructions.

op r_{S1} r_{S2} r_D	算術・論理演算命令
op r_{S1} # r_D	算術・論理演算命令（即値）
start (*dest, mc, join*)	スレッド起動命令
brz r_{S1} *dest*	0分岐命令
brnz r_{S1} *dest*	非0分岐命令
call r_{S1} r_D->(*dest, mc, join*)	インスタンス獲得命令
link r_{S1} r_{S2} *slot*	引数リンク命令
rlink r_{S1} r_D *slot*->(*dest, mc, join*)	結果値返し先リンク命令
receive r_{S1}->(*dest, mc, join*)	引数受取命令
return r_{S1} r_{S2}	結果値リターン命令
rins	インスタンス解放命令
read r_{S1} r_D->(*dest, mc, join*)	I-structure メモリ読み出し命令
iread r_{S1} r_D->(*dest, mc, join*)	
ifread r_{S1} r_D->(*dest, mc, join*)	
write r_{S1} r_{S2}	I-structure メモリ書き込み命令
iwrite r_{S1} r_{S2}	
ifwrite r_{S1} r_{S2}	

Datarol-IIでの関数適用は，まずcall命令により，新たなインスタンスの生成を行った後，link命令により引数データを渡す．また，rlink命令により，関数の結果値の返し先論理レジスタ名（アドレス）を送る．呼ばれ側インスタンスではその初期化スレッドでreceive命令を発行し引数データの到着時に起動するスレッドを設定する．これにより引数データの到着により呼ばれ側インスタンスのスレッドが起動される．

☆ Datarol-IIでは各インスタンスごとに数個の同期用カウンタを持つ．現在の仕様では各インスタンスごとに4 bitの同期用カウンタを8個持つ．

関数の結果値は return 命令により呼出側インスタンスへ返される．

2.3 スレッド起動メカニズム

ここではリモートメモリ参照を例に Datarol-II プロセッサにおけるスレッド起動メカニズムについて説明する．FU でリモートメモリ参照命令が発行されてから，結果データを利用するスレッドが起動されるまでの動作は以下のとおりである．

1. FU で read 命令により，continuation が MU 内のデスティネーションの論理レジスタに書き込まれ，同時にリモートメモリアクセスのリクエストパケットが送出される．
2. リクエストパケットはプロセッサ間ネットワークにより目的のプロセッサの SMU へ送られる．SMU でメモリアクセスを行い，リプライパケットがリクエストパケットの送り元のプロセッサへ返送される．
3. リプライパケットはリクエストパケット送り元プロセッサの CU で受理される．CU はまず，結果データを書き込む先の論理レジスタを MU から読み出し，read 命令発行時にセットした continuation を得る．次に結果データを論理レジスタに書き込む．ここで得られた continuation は AC へ送られる．
4. AC は continuation 指定に従い同期処理を行う．同期が成功した場合はインスタンス番号とスレッド開始アドレスから成るスレッド ID を RQ へ送る．
5. FU はスレッドが terminate するたびに RQ から新たなスレッド ID を読み込み実行する．

2.4 MU の構成

図 2 に MU の構成を示す．MU は各インスタンスの論理レジスタセットの内容を保持する Operand Memory (OM)，OM データの一部のコピーを保持する高速メモリ Register Buffer (RB)，および RB のページテーブルを保持し RB の管理を行うメモリコントローラから成る．RB は 1 インスタンスの論理レジスタセットに相当するページを単位として管理され，OM に対する write back 方式のキャッシュとして機能する．

MU controller は RQ の待ちスレッドを先読みし必要なページをあらかじめ RB へ読み込む（RB プリローディング機構）．またこのとき MU は RQ へ RB のページ番号を渡す．これにより FU は RQ から RB ページ番号を受け取り RB へ直接アクセスすることができる．

CU からのアクセスは MU controller で受理され，ここでまず RB のページテーブルが引かれる．その結果，当該インスタンスが RB 内に存在する場合は，MU controller は RB に，そうでない場合は OM にアクセスする．

2.5 FU の構成

FU での演算では FU 内に設けられたレジスタ（物理レジスタ）が用いられる．一方 Datarol-II の命令はオペランドを論理レジスタ名で指定しているので，MU 内に格納されている論理レジスタの内容を物理レジスタにあらかじめ読み込む必要があり，また演算結果データを論理レジスタへ反映させる必要もある．Datarol-II 方式に限らず一般的にコンテクストスイッチに伴うレジスタ値の退避や回復のためのメモリアクセスが必要となり，これが細粒度処理においては大きなオーバヘッドとなる．Datarol-II プロセッサでは自動レジスタロードストア機構を用いてコンテクストスイッチに伴うメモリアクセスを自動的に行うことによりオーバヘッドの隠蔽を行う．以下では，この自動レジスタロードストア機構を中心に FU の構成を述べる．

Datarol-II プロセッサの FU の構成を図 3 に示す．FU 内には 4 程度の物理レジスタセットが用意され，

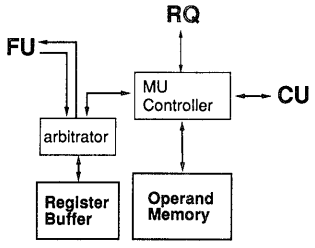


図 2 MU 構成図
Fig. 2 Memory Unit.

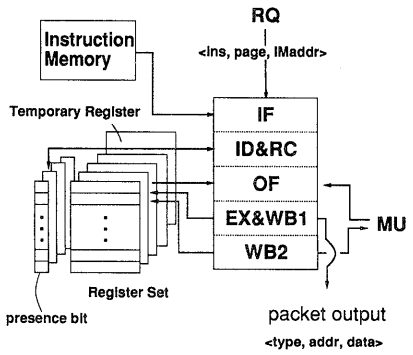


図 3 FU 構成図
Fig. 3 Function Unit.

その1つのレジスタセットがRBの1ページに相当する．FUでのスレッドの実行に際しては1つのスレッドに1つの物理レジスタセットが割り当てられ，そのスレッドで用いる論理レジスタセットに対応づけられる．各物理レジスタはpresence bitを持ち，論理レジスタの値が既に物理レジスタにロードされているか否かを示す．オペランドデータの参照は，まずpresence bitが調べられ，もしこれがonの場合は物理レジスタの内容が用いられ，presence bitがoffの場合は論理レジスタの値を自動的にMUからロードし物理レジスタへセットする．前節で述べたRQによるRBプリローディング機構により，FUがMUへアクセスする場合必要なページはRBへロードされていることが保証されており，FUはRQから受け取ったRBのページ番号を用い必要な論理レジスタの内容をRBから直接読み込むことができる．一方，論理レジスタの更新は対応する物理レジスタの更新時に同時に行われる．すなわち，FU内の物理レジスタはRBに対するwrite throughキャッシュのような振舞をする．これにより明示的なloadやstore命令によって論理レジスタの値を物理レジスタに読み込み，あるいは物理レジスタの値を論理レジスタへ書き込む必要がない．またFU内にはテンポラリ用のレジスタセットもあり，スレッド内での局所的なデータの授受等に用いることができる．

Datarol-IIプロセッサでは上記の自動レジスタロードストア機構がFUの演算パイプライン中に組み込まれ，通常の処理と並行して行われる．FUの演算パイプラインは以下のような5段のステージで構成される．

- **IF**: Instruction Memory (IM) より命令をフェッチする．
- **ID&RC**：命令のデコードとレジスタ内容の存在チェックを行う．
- **OF**：前段の結果に従ってオペランドデータをFUレジスタあるいはMUから読み込む．2つのオペランド双方についてMUからの読み込みが必要な場合は1インタロックが発生する．
- **EX&WB1**：命令実行を行う．また前段でMUからの読み出しを行った場合，そのデータを該当するFUレジスタへ書き込む．
- **WB2**：演算結果をFUレジスタおよびMUへ書き込む．MUへの書き込み部にはライトバッファを用意し，OFステージとのMUアクセス競合を緩和する．

上記パイプライン構成により，Datarol-IIプロセッサではMUへのアクセスと命令実行とを並行して行うことができ，これによりコンテクストスイッチに伴うメモリアクセスのオーバヘッドの隠蔽を行う．

2.6 負荷制御機構

RBの導入によるメモリアクセスの高速化を実現するためには，メモリアクセスに局所性が要求される．そこで，Datarol-IIプロセッサでは負荷制御機構によりアクティブなインスタンス数を制御することによりメモリアクセスの局所性を向上させる．

新たな関数起動のためにFUでcall命令が発行されると，インスタンスの割り付け要求パケットが生成され，これがSMUへ送られる．SMUは通常callパケットを受け取ると，新たなインスタンスの割り付けを行い，送り元のインスタンスへ新たに割り当てられたインスタンス番号をreturnパケットとして送り返す．次に新たに割り付けられたインスタンスの初期化スレッド起動パケットが送出され，初期化スレッドが起動される．

ここで，SMUでは初期化スレッド起動パケットの送出を行う前に，プロセッサの負荷状態☆を調べ，負荷値があらかじめ設定した規定値より高い場合は初期化スレッド起動パケットの送出をペンディングし，それをSM内にキューイングする．負荷値が規定値より下がった時，SM内にキューイングされた初期化スレッド起動パケットが取り出され送出される．このようにDatarol-IIプロセッサでは初期化スレッドの起動をペンディングすることにより負荷制御を行う．

3. 評　　価

3.1 評価の方法

我々はDatarol-IIプロセッサの性能評価のため，クロックレベルでの各部の動作を再現するソフトウェアシミュレータを作成した．以下では本シミュレータを用いた性能評価結果について述べる．

表2に今回の評価に用いたパラメータを示す．RBのアクセスレイテンシは1clockで，各クロックサイクルで読み込みあるいは書き込みのいずれかを発行でき，読み出しの場合次のサイクルでデータが利用可能であるとした．OMについてはD-RAMの使用を想定し，アクセスレイテンシを3clock, RB-OM間での転送にはページモードの使用を想定し1ページ (16 word) の転送時間を18clockとした．PE間ネットワークは2次元のトーラス網とし，Net-IF内の遅延 $ly_{Net\text{-}IF}$ (clock) とネットリンクのバンド幅 th_{max}

☆ プロセッサの負荷値の計算法については現在数種類検討中である[10]．後述の実験ではRQ内のパケット数を負荷値として用いた．

表 2 シミュレーション パラメータ
Table 2 Simulation parameters.

FU レジスタセット数	4 set
FU ライトバッファ数	2 entry
RB ページ数	32 page
RB アクセスレイテンシ	1 clock
OM アクセスレイテンシ	3 clock
OM ページデータ転送時間	18 clock
ネットワーク形態	2 次元トーラス網
NET-IF 内遅延（$ly_{Net\text{-}IF}$）	2 clock
ネットリンク当たりスループット（th_{max}）	0.5 packet/clock
パケットサイズ	70 bit（ヘッダ（6 bit）＋アドレス(32 bit)＋データ(32 bit)）
負荷制御方式	RQ 内の待スレッド数を負荷値とし，RB ページ数の 1/4 を閾値に設定
負荷分散方式	当該 PE および隣接 PE のうちで負荷値の最小 PE でインスタンス割り付け

(packet/clock) によりモデル化している．つまり，1 つのパケットを送るためにはネットリンクでの転送に $1/th_{max}$ クロック必要となり，したがって隣接 PE 間の通信（送り元の PE の Net-IF の入力から送り先 PE の Net-IF の入力まで）の最小遅延 ly_{min} は $ly_{Net\text{-}IF}+1/th_{max}$ クロックとなる．

今回用いた例題プログラムは以下の通りである．

- **Matrix**：64×64 行列の行列積計算プログラム．16×16 プロセッサで実行．配列データは各 PE に分散配置．
- **LUD**：64×64 行列の LU 分解プログラム．4×4 プロセッサで実行．配列データは各 PE に分散配置．
- **Queen**：8 クイーンの配置問題の全解探索プログラム．4×4 プロセッサで実行．

3.2 自動レジスタロードストア機構の効果

FU の自動レジスタロードストア機構の効果について，**図 4** に明示的な命令により論理レジスタ値のロードストアを行った場合（図中 A）と，自動レジスタロードストア機構を用いた場合（図中 B）の実行クロック比とその内訳を示す．また，明示的な命令により論理レジスタ値のロードストアを行った場合の全実行クロックに対するロードストア命令実行クロックの割合を図中の括弧内に示す．これから，特に **Matrix** と **Queen** においては明示的なロードストア命令を用いた場合，ロードストア命令が実行クロックのかなりの部分（約 40%）を占めオーバヘッドとなっていることがわかる．一方，自動レジスタロードストア機構を用いた場合にはメモリアクセスの多くが命令実行とオーバラップして行われ，全体として 33%～16%の実行クロック数が削減されていることが確認でき，自動レジスタロードストア機構が効果的に機能していることがわかる．なお，自動レジスタロードストア機構を用いた場合，**Matrix** と **Queen** の実行において FU のアイドル時間が増加しているが，これは主に FU の OF ステージで 2 つのソース論理レジスタ値の両方が RB から読み込まれる場合に発生したパイプラインバブルによるものである．

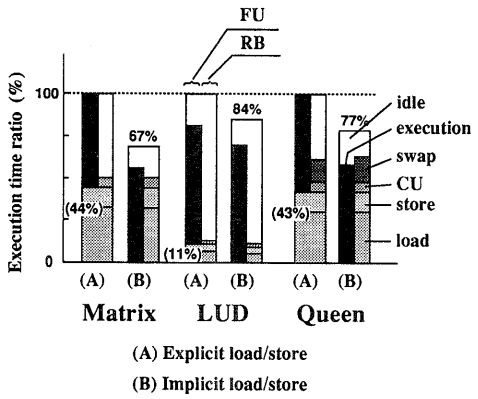


図 4 自動レジスタロードストア機構の効果
Fig. 4 Effect of implicit load store mechanism.

3.3 耐レイテンシ性能

Datarol-II プロセッサでは，細粒度スレッド間のコンテクストスイッチを高速に行うことによりリモートメモリアクセス等のレイテンシ隠蔽を行う．ここでは **Matrix** プログラムを用いネットワークのパラメータ（ly_{min} および th_{max}*）を変え Datarol-II プロセッサの耐レイテンシ性能を調べる．

ここで，まず **Matrix** プログラムの性質について述べる．本プログラムは 64×64 行列の各要素の計算のそれぞれにインスタンスを割り当てる．すなわち 4096 インスタンス（各 PE 当たり 16 インスタンス）の計算が並行して行われる．各インスタンスの処理は 1 イタレーションが 20 命令からなるループで，各イタレーション中に 2 要素の配列データ（32 bit）の読み出しのため 2 つのリモートメモリアクセスを発行する．各リモートメモリアクセスの平均距離（ネットワークのホップ数）は 8 で，トーラスネットワークではネットリンクが PE 数の 2 倍あるので，これらから各ネットリンクの必要スループットは $((2\times8)/20)/2=0.4$ packet/clock となる．また，各 PE で 16 インスタンスが同時に実行されているので，$20\times(16-1)=300$clock 程度まではレイテンシ隠蔽が可能であると推測される．

図 5 に **Matrix** を実行したときの FU の演算ステー

* 実際には $ly_{Net\text{-}IF}$ と th_{max} を変化．

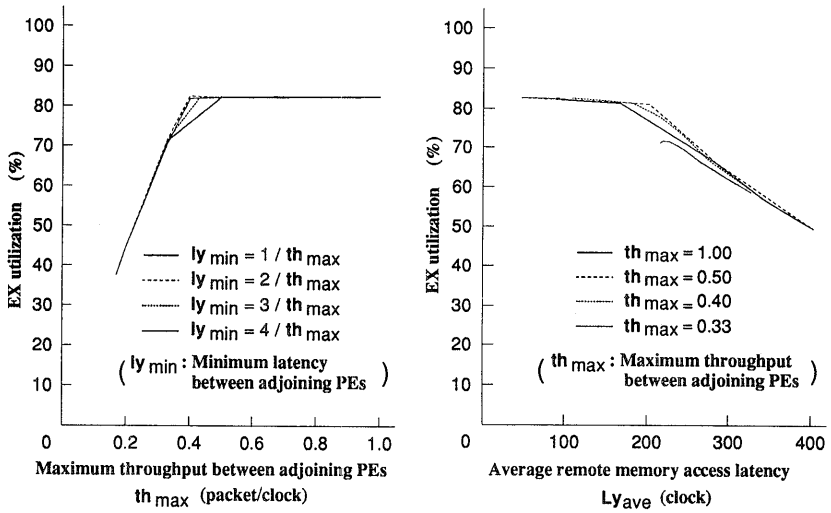


図5 ネットワーク性能による演算部稼働率の変化（Matrix）
Fig. 5 Datarol-II performance to the change of network parameters.

ジ（EX）の稼働率を示す．同図左はネットワークのスループットを変化させた場合のEX稼働率の変化で，スループットが十分ある場合（0.4 packet/clock以上），高稼働率が達成されていることがわかる．また，同右図に，レイテンシをより増加した場合について，シミュレーションにおけるリモートメモリアクセスの実際のレイテンシの平均を横軸としたグラフを示す．同図において，リモートメモリアクセスの平均レイテンシが200 clock以下のとき，EX稼働率へのレイテンシ増加の影響はほとんどなく，レイテンシ隠蔽が効果的に行われていることがわかる．平均レイテンシが約200 clockの場合のリモートメモリアクセスのレイテンシ分布を調べると，300 clock以内が約7割，残り3割が300 clock以上であった．これから，推測値（300 clock）に見合うレイテンシ隠蔽が達成されているとみなすことができる．

以上の結果より，Datarol-IIプロセッサはその高速コンテクスト切替え能力により，リモートメモリアクセスのレイテンシを隠蔽し効率的な処理を行えることが確認できる．

3.4 負荷制御の効果

Queen は比較的大きな並列性を持つプログラムで，実行時に多数のインスタンスを生成する．このようなアプリケーションでは，各PE内のアクティブなインスタンス数がRBのページ数を越え，RB-OM間のページ転送が頻発しこれがネックとなって実行時間の増加を招く傾向にある．Datarol-IIプロセッサでは負荷制御機構によりPE内のアクティブなインスタンス数

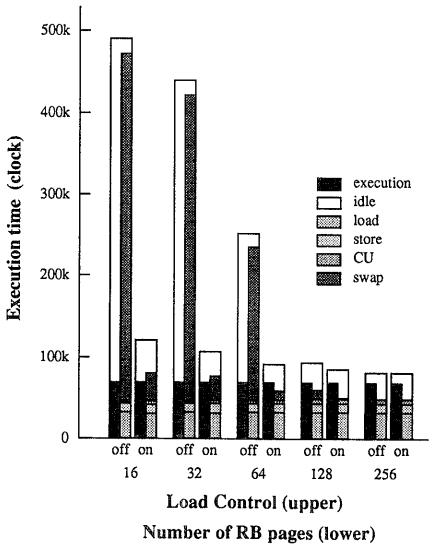


図6 負荷制御の効果（Queen）
Fig. 6 Effect of load control.

を制限することにより，RB-OM間のページ転送量を抑制する．今回のシミュレーションでは負荷値としてRQ内の待ちスレッド数を用い，RBページ数の1/4を閾値とした負荷制御を行いその効果を検証した．

図6にQueenプログラムにおいて，各PEのRBのページ数を変化させ，それぞれ負荷制御を行わない場合（off）と負荷制御を行った場合（on）の実行時間を示す．負荷制御を行わない場合，ページ数が少なくなるにつれページ転送量（図中swap）が増加し，これがネックとなり実行時間が増加する．一方，負荷制御を

行った場合，負荷制御を行わない場合に比べページ転送の頻度が軽減され，実行時間の増加が緩和されている．負荷制御を行うことにより RB のページ数が 64 以上ではページ転送のオーバヘッドは通常の処理に隠れ実行時間にほとんど現われていないことが確認でき，RB が効果的に機能していることがわかる．もし RB が無く FU から直接 OM をアクセスすると仮定すると，メモリアクセス時間が RB を用いたときの数倍*かかり，それによって実行時間も長くなる．ハードウェアコスト的にも 64 ページ程度ならば許容できる範囲であり，RB の導入により処理の高速化が達成されたことがわかる．

4. 関連研究

Datarol-II 同様にデータ駆動方式をベースとして細粒度のマルチスレッド処理の効率的実行を目指した並列計算機として電子技術総合研究所の EM-4[8]や MIT の*T[5]等が挙げられる．

EM-4 では直接待ち合わせ方式により連想記憶を必要としないデータの待ち合わせ（同期）を実現した．また，強連結枝モデルにより定義される強連結ブロック（Datarol-II のスレッドに相当）内の命令を優先的に実行することにより先行制御パイプラインや高速レジスタの利用を可能とした．しかしながら EM-4 の直接待ち合わせ方式では命令のメモリアドレスと待ち合わせに使用されるメモリアドレスの間に 1 対 1 の対応関係があるためメモリの使用効率が悪く，待ち合わせも 2 入力に限られる．Datarol-II ではデータの格納先と同期用カウンタをそれぞれ明示的に指定することによりメモリセルの再利用が可能であり，多入力の同期指定も行うことができる．EM-4 の強連結ブロックは 1 または 2 の入力パケットにより起動されるため，多入力を許す Datarol-II に比べ粒度の低いものになり，高速レジスタの利用範囲も限られたものになる．Datarol-II では by-reference によるデータ参照方式によりデータ駆動方式における冗長なデータのコピーや同期の削減も行われる．また，EM-4 では同期処理部と演算実行部が直接結合されているため同期失敗時にパイプラインバブルを生じる．一方 Datarol-II では同期処理部と演算実行部の間にスレッドキューが設けられているため，同期失敗によるバブルの発生はない．さらに，EM-4 には階層的なメモリシステムがなく，大量な高速メモリを必要とする問題もある．

*T は Monsoon[6]で得られた知見をもとに現在開発が行われている並列計算機である．*T は演算処理用プロセッサ（dP），パケットハンドリングおよび同期処理用プロセッサ（sP），リモートメモリアクセス用プロセッサ（RMem）の 3 つの論理プロセッサから成り，sP と dP はスレッドキューにより接続される．全体的な構成は Datarol-II にかなり類似している．Datarol-II との相違点は Datarol-II がそれぞれ専用の機能ユニットにより構成されるのに対し，*T では市販の高性能マイクロプロセッサを使用する点である．市販の高性能マイクロプロセッサは長いスレッドを効率良く実行するように設計されているため，メモリアクセスや分岐が頻繁に発生する細粒度スレッド実行においてはプロセッサの性能を十分に発揮できない．スーパースカラ実行によりコンテクストスイッチに伴うメモリアクセスを通常の処理と並行して行うことも可能ではあるが，そのためにはメモリアクセス命令とその結果値を使用する命令間を埋めるだけの並列性がスレッド内に要求され，Datarol-II に比べより長いスレッドを抽出できるコンパイラを用意する必要がある．また，実際に数千，数万プロセッサ規模の並列計算機を作る場合を想定すると，1 プロセッサあたりの占有面積や消費電力等が問題となり，Datarol-II のような専用設計が有利になるものと思われる．

5. おわりに

本論文では細粒度マルチスレッド処理向きプロセッサ Datarol-II を提案した．本プロセッサはデータ駆動方式を最適化した Datarol にスレッド実行を導入し一般的な RISC プロセッサと同様のパイプライン処理および高速レジスタの利用を可能とした．また，自動レジスタロードストア機構によりコンテクストスイッチに伴うメモリアクセスを明示的なロードストア命令を用いず，かつ通常の処理と並行して行うことにより細粒度処理におけるオーバヘッドを隠蔽する．さらに階層的なメモリシステムと負荷制御機構を導入しコストパフォーマンスに優れたメモリシステムを実現した．

シミュレーションによる評価により，自動レジスタロードストア機構によるメモリアクセスオーバヘッドの隠蔽効果，優れた耐レイテンシ性能，負荷制御による効果的な階層メモリシステムの実現，が確認された．これらから本プロセッサは超並列計算機用要素プロセッサとして有望であると思われる．

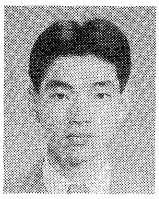
参考文献

1) Amamiya, M. and Taniguchi, R.: Datarol: A

* ここでは OM アクセスレイテンシを 3 クロックとしているので 3 倍．

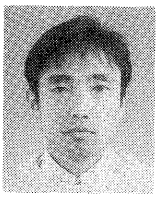
Massively Parallel Architecture for Functional Language, *Proc. SPDP*, pp. 726-735 (1990).
2) Dally, W. J., Chien, A., Fiske, S., Horwat, W., Keen, J., Larivee, M., Lethin, R., Nuth, P. and Wills, S.: The J-Machine: A Fine-grain Concurrent Computer, *Proc. 11th IFIP*, pp. 1147-1153 (1989).
3) Grafe, V. G. and Hoch, J. E.: The Epsilon-2 Multiprocessor System, *J. Parallel and Distributed Computing*, Vol. 10, No. 4, pp. 309-318 (1990).
4) Kusakabe, S., Hoshide, T., Taniguchi, R. and Amamiya, M.: Parallelism Control and Storage Management in Datarol PE, *Proc. IFIP World Congress*, Vol. 1, pp. 535-541 (1992).
5) Nikhil, R. S., Papadopoulos, G. M. and Arvind: *T: A Multithread Massively Parallel Architecture, *Proc. 19th ISCA*, pp. 156-167 (1992).
6) Papadopoulos, G. M. and Culler, D. E.: Monsoon: an Explicit Token-Store Architecture, *Proc. 17th ISCA*, pp. 82-91 (1990).
7) Taniguchi, R., Kawano, T. and Amamiya, M.: A Distributed-Memory Multi-Thread Multiprocessor Architecture for Computer Vision and Image Processing: Optimized Version of AMP, *Proc. 26th ICSS*, Vol. 1, pp. 51-160 (1993).
8) 児玉祐悦，坂井修一，山口喜教：データ駆動型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，Vol. 32, No. 7, pp. 849-858 (1991).
9) 立花 徹，谷口倫一郎，雨宮真人：データフロー解析による関数型言語 Valid のコンパイル法—Datarol プログラムの抽出アルゴリズム—，情報処理学会論文誌，Vol. 30, No. 12, pp. 1628-1638 (1989).
10) 星出高秀，園田浩二，谷口倫一郎，雨宮真人：並列計算機 Datarol マシンにおける資源管理と負荷制御方式，信学技法，Vol. CPSY 91-7, pp.25-32 (1991).
11) 星出高秀，日下部茂，谷口倫一郎，雨宮真人：Datarol マシンにおける並列展開戦略，JSPP' 93 論文集，pp.347-354 (1993).

(平成 6 年 9 月 19 日受付)
(平成 7 年 2 月 10 日採録)



川野 哲生（学生会員）

1968 年生．1991 年熊本大学工学部電気情報工学科卒業．1993 年九州大学大学院総合理工学研究科情報システム学専攻修士課程修了．現在，同専攻博士後期課程在学中．並列計算機アーキテクチャの研究に従事．電子情報通信学会会員．



日下部 茂（正会員）

1966 年生．1989 年九州大学工学部情報工学科卒業．1991 年九州大学大学院総合理工学研究科情報システム学専攻修士課程修了．同年より同専攻助手．並列処理記述言語，関数型言語，データフローモデルに基づく細粒度並列処理の研究に従事．

谷口倫一郎（正会員）

1980 年九州大学大学院工学研究科修士課程修了．同年同大学院総合理工学研究科助手．平成元年同助教授．工学博士．画像理解，並列処理，画像処理システムに関する研究に従事．電子情報通信学会篠原記念学術奨励賞，本会論文賞受賞．人工知能学会，電子情報通信学会各会員．現在，本会コンピュータビジョン研究会幹事．

雨宮 真人（正会員）

1942 年生．1967 年九州大学工学部電子工学科卒業．1969 年同大学院工学研究科修士課程修了．同年日本電信電話公社武蔵野電気通信研究所入所．以来，プログラミング言語・処理系，自然言語理解，データフロー・アーキテクチャ，並列処理，関数型/論理型言語，知能処理アーキテクチャ，等の研究に従事．現在九州大学大学院総合理工学研究科情報システム学専攻教授．工学博士．電子情報通信学会，ソフトウェア科学会，人工知能学会，IEEE, ACM, AAAI 各会員．