細粒度スレッド処理のためのコンテクストスイッチ機構ー並列計算機Datarol-IIの階層メモリシステムー

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

日下部，茂
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

出版情報：並列処理シンポジウム，1994年，pp.81-88，1994-05
バージョン：
権利関係：

細粒度スレッド処理のためのコンテクストスイッチ機構
— 並列計算機 Datarol-II の階層メモリシステム —

川野哲生　日下部茂　谷口倫一郎　雨宮真人
九州大学総合理工学研究科

概要

超並列計算機において効率的な処理を行うためには，リモートメモリアクセス等により生じるレイテンシを隠蔽する必要があり，高速コンテクスト切替え能力が必要とされる．本稿で提案する Datarol-II プロセッサは，自動レジスタロード/ストア機構，階層メモリシステム，負荷制御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オーバーヘッドを削減し効率的な細粒度スレッド処理を実現する．本稿ではシミュレーション評価により本方式の有効性についても示す．

Abstract

In a massively parallel computer, one of the most important problems is latencies caused by remote memory accesses. Therefore, the processor must perform fast context switching among fine-grain concurrent processes. In this paper, we propose the Datarol-II processor, that can efficiently execute a fine-grain multi-thread program. In the Datarol-II processor, we introduce an implicit register load/store mechanism in the execution pipeline. A two-level hierarchical memory system is also introduced in order to reduce local memory access latency.

1　はじめに

超並列計算機用要素プロセッサには耐レイテンシ性能が要求され，細粒度プロセス間のコンテクストスイッチを高速に行えるアーキテクチャの開発が必要である．一般に細粒度プロセスの多重処理環境においては，頻繁に発生するコンテクストスイッチ毎に実行環境の保存・再現のためのメモリアクセスが発生しこれがオーバーヘッドとなる．また，メモリアクセスに対する局所性が乏しく，従来のキャッシュのようなメモリ高速化手法を用いることも困難である．

現在，我々は Datarol-II と呼ぶ並列計算機の開発を行っている．Datarol-II はデータ駆動方式をより最適化した Datarol アーキテクチャ[1] をもとに設計され，効率的な細粒度マルチスレッド処理を目指したマシンである．Datarol-II では独自のパイプライン構成により，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い，さらにメモリアクセスと命令実行とをオーバーラップして処理することにより，コンテクストスイッチに伴うオーバーヘッドを隠蔽する．また，演算部とメモリの間に Register Buffer と呼ぶ高速なメモリを設け，さらに負荷制御機構を導入しアクティブなプロセス数を制限することによりメモリアクセスの局所性を向上させ，効果的な階層メモリを実現する．本稿では Datarol-II マシンのプロセッサエレメントのメモリ構成およびコンテクストスイッチに伴うメモリアクセス方法について述べ，ソフトウェアシミュレータを用いた評価により本方式の有効性を示す．さらに，Datarol-II プロセッサのメモリ構成と従来のキャッシュとの相違についても述べる．

Fast context switching mechanism for fine-grain multi-thread processor
Tetsuo Kawano, Shigeru Kusakabe, Rin-ichiro Taniguchi, Makoto Amamiya
Department of Information Systems, Graduate School of Engineering Sciences, Kyushu University

2 Datarol-IIプロセッサアーキテクチャ

2.1 Datarolアーキテクチャ

我々はこれまでデータ駆動方式の利点である，(a)プログラムに内在する並列性の自動抽出，(b)ハードウェア機構による高速コンテクストスイッチとそれによるレイテンシ隠蔽，等に着目し，データ駆動方式をより最適化したアーキテクチャの研究を行ってきた．

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)毎にそれぞれ固有の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レジスタ(以後，これを論理レジスタと呼ぶ)を用いる．Datarolは，このレジスタの導入によりデータ駆動方式における冗長なフロー制御を削除し最適化したマルチスレッドコントロールフローグラフである．

Datarol-IIプログラムは，Datarolからスレッド(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)を抽出し命令間のフロー制御をスレッド間の制御に置き換えたものである．Datarol-IIプログラムにおける関数は，複数のスレッドから成り，プロセッサ内ではスレッドを単位とした処理を行う．

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}はソースレジスタ名，r_D はディスティネーションレジスタ名を示す．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の指定(continuation)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[1]，*join*が同期数を示す．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，スレッドは`start`命令によって明示的に起動することもできる．

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．親インスタンスより渡された引数データは初期化スレッド内の`receive`命令により受け取られ，`receive`命令はそれぞれの引数に対応するスレッドを起動する．関数の結果値は`return`命令により親インスタンスに返される．

2.3 プロセッサ構成

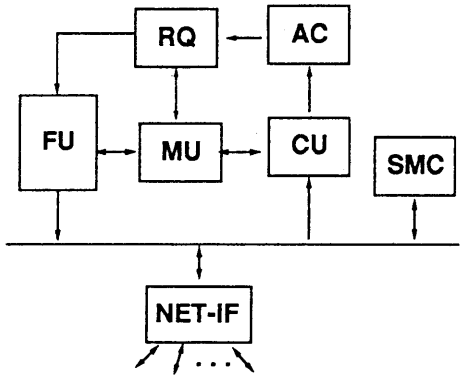

図1: Datarol-IIプロセッサ構成図

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

- **FU(Function Unit)**: RQから実行可能なスレッドを受け取り，スレッド内の命令を逐次的に実行する．また，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．
- **CU(Communication Unit)**: 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ACへ送る．
- **AC(Activation Controller)**: 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．
- **RQ(Ready Queue)**: 実行待ちのスレッド

[1] Datarol-IIでは各インスタンス毎に数個の同期用カウンタを持つ．カウンタ数はインプリメントによるが，現在の仕様では8となっている．

表 1: Datarol-II 命令セット

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}	

を格納する.

- **SMU(Structure Memory Unit)**: SMC 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提供する. また, インスタンスの割り付け・回収や負荷制御も行う.
- **MU(Memory Unit)**: 各インスタンスの論理レジスタの内容を保持する.
- **NET-IF(Network Interface)**: 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ーフェース部.

Datarol-II プロセッサでの処理は, (1) 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, (2)continuation 指定による split-phase 実行, の 2 つで行われる. (1) は FU で, (2) は FU-CU-AC-RQ による循環パイプラインにより実現される.

2.4 Function Unit

FU は RQ から実行可能なスレッドを受け取り, そのスレッド内の命令を逐次的に実行する. RQ からは実行可能なスレッドのインスタンス番号, スレッド開始アドレス, および RB ページ番号 (後述) が送られる. FU での演算には, FU 内にあるレジスタ (以後, FU レジスタと呼ぶ) が用いられる. そこで, 新たなスレッドの実行を開始する場合, 当該スレッドの属するインスタンスの論理レジスタの値を一旦 MU から FU レジスタへと読み込む必要がある. また, 他のインスタンスへの切替え時には更新された FU レジスタの値を MU 内の論理レジスタへ反映させる必要がある. このコンテクストスイッチに伴う MU へのアクセスがオーバーヘッドとなる. 細粒度のスレッド処理においてはコンテクストスイッチが頻繁に発生し, このオーバーヘッドが顕著に現れる. これは Datarol-II 方式に限ったものではなく, 従来のプロセッサにおいても同様に発生する問題である.

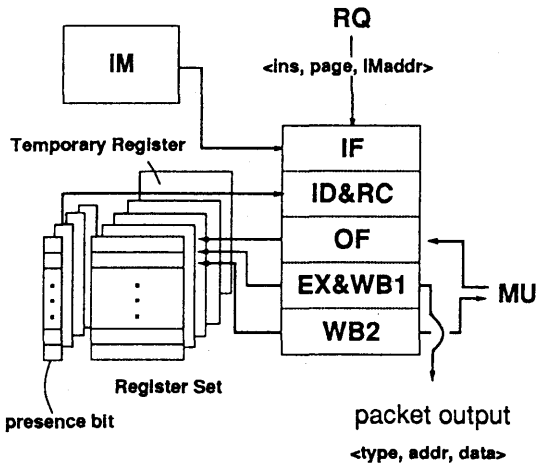

図 2: Function Unit 構成図

[自動レジスタロード/ストア機構]

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のオーバーヘッドの削減を行うため，自動レジスタロード/ストア機構を導入した．FUには数セット(4程度)のレジスタが用意され，各スレッドの実行にはこの内の1セットが割り当てられる[2](図2参照)．この各FUレジスタセットが1インスタンスの論理レジスタセットへと対応する．また，各FUレジスタにはその内容の存在を示すpresenceビットがあり，FUでの論理レジスタの参照の際にそのpresenceビットが検査され，この内容がoffである(レジスタ内容が存在しない)場合，読み出しアクセスはMUに対して行われる．MUへのアクセスはRBのページ番号と論理レジスタ番号を指定することにより，当該インスタンスの論理レジスタの内容を得ることができる．このようなpresenceビットを用いたレジスタ内容の自動的なロード機構により，スレッドの開始時に陽にload命令を用いてMUへのアクセスを行う必要がない．また，レジスタへの演算結果の書き込みはFUレジスタと該当するMUアドレスの双方に行われ，これによりスレッドの実行終了時に陽にstore命令を用いてレジスタの内容をMUへ退避する必要もない．以上の機構によりコンテクスト切替えに伴う実行環境の保存・再現のためのload/store命令発行のオーバーヘッドを解消する．

さらに，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インターロックが発生する．
- **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こともできる．

このような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ーヘッドの隠蔽を行う．

2.5 Memory Unit

図3にMUの構成を示す．MUは各インスタンス毎に割り当てられた論理レジスタセットの内容をOperand Memory(OM)に保持する．さらに，MUにはOMへのアクセスの高速化を実現するためRegister Buffer(RB)と呼ぶ高速メモリが導入されている．RBはページを単位として管理され，この1つのページが1インスタンスの論理レジスタセットに一致する．MU controllerはRBのページ管理，RB-OM間のページ置き換え制御を行う．

MU controllerはRQから待ちスレッドのインスタンス番号を受け，RBのページ番号を返す．このとき該当するインスタンスがRBに存在しない場合は，MU controllerはRBに該当インスタンスを読み込んだ後にRBページ番号を返す．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のページテーブルをが引かれる．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．

2.6 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FUでcall命令が発行さ

[2]さらに，全てのスレッドにおいてテンポラリレジスタセットが利用可能で，同一スレッド内でのデータの授受に用いられる．

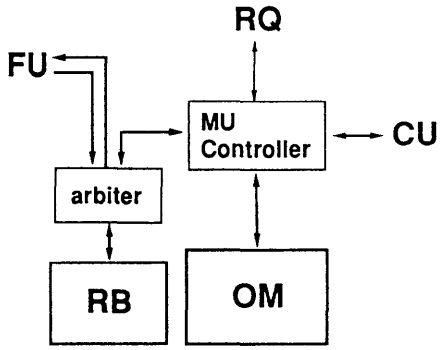

図 3: Memory Unit 構成図

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[3]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，初期化スレッドの起動をペンディングすることによりDatarol-IIプロセッサでの負荷制御が行われる．

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について述べる．

表2に今回の評価に用いた各部のパラメータを

[3]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[11]．今回は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

FU レジスタセット数	4
FU ライトバッファ数	2 entry
RB ページ数	32 page
RB アクセスサイクル	1 access/clock
OM アクセスレイテンシ	3 clock
OM 行アクセスサイクル	1 access/clock
ネットワーク形態	2 次元トーラス網
ネットリンク当たり最小通信遅延	2 clock
NET-IF 内遅延	2 clock
ネットリンク当たりスループット	0.5 packet/clock
負荷制御方式	閾値：待ちスレッド数 (RB ページ数の 1/4)
負荷分散方式	隣接 PE 間での負荷最小 PE へインスタンス割り付け

示す．また，今回用いた例題プログラムは以下の通りである．

- **Matrix**：64×64 の行列積．16×16 プロセッサで計算．
- **LUD**：64×64 の行列の LU 分解．4×4 プロセッサで計算．
- **Queen**：8 クイーンの配置問題 (全解探索)．4×4 プロセッサで計算．

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/ストア機構の効果を確かめるため，(A) 明示的な命令によりレジスタ値の load/store を行った場合，(B) 自動レジスタロード/ ストア機構を用いた場合，の比較を行う．図 4に各プログラムについて (A) の実行時間を 1 とした実行時間比と，その内訳を示す．同図から，(A) の場合は load/store 命令が実行クロック数のかなりの部分を占めこれがオーバーヘッドとなっていることがわかる．一方，(B) の場合はメモリアクセスの多くが命令実行とオーバーラップして実行され，かなりの実行クロック数が削減されたことが確認できる．

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

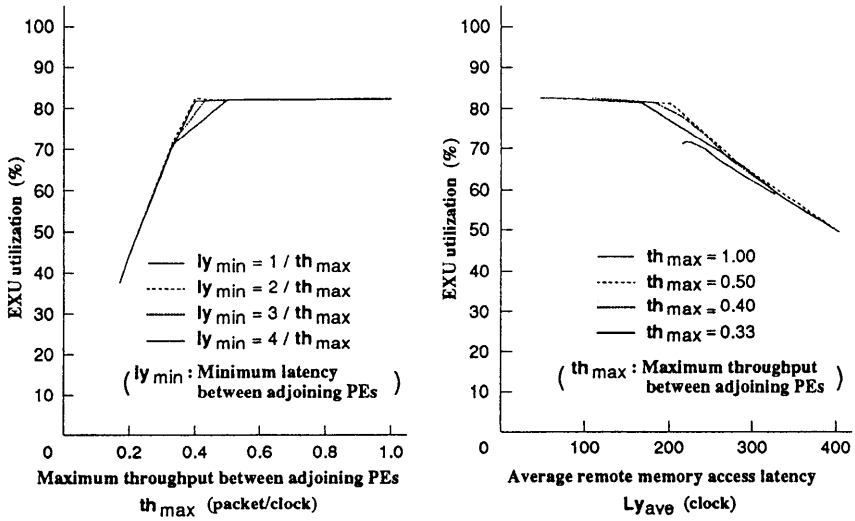

図 5: ネットワーク性能による演算部稼働率の変化 (Matrix)

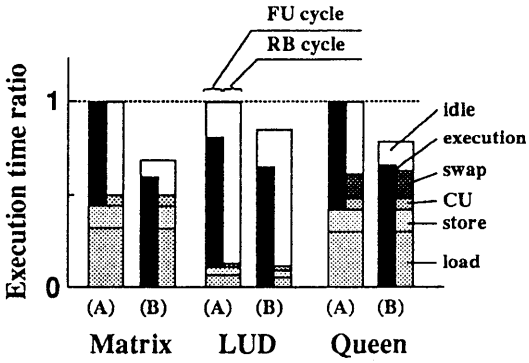

図 4: 自動レジスタロード/ストア機構の効果
(A):自動ロード/ストア機構を用いない場合
(B):自動ロード/ストア機構を用いた場合

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 (スループット，レイテンシ) 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 の行列の各要素毎にインスタンスを割り当て計算する．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 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 $((2 \times 8)/20)/2 = 0.4$ paket/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 の演算ステージ (EXU) 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 EXU 稼働率の変化で，スループットが十分ある場合 (0.4packet/clock 以上)，高稼働率が達成されていることが分かる．また，レイテンシをより増加した場合 (同右図)，リモートメモリアクセスの平均レイテンシが 200clock 以下のとき[4]，EXU 稼働率へのレイテンシ増加の影響はほとんどなく，レイテンシ隠蔽が行われていることが分かる．

以上の結果より，Datarol-II プロセッサはその

[4] 平均レイテンシが 200clock 以内のとき，ほとんどのリモートメモリアクセスは 300clock 以内に終了している．

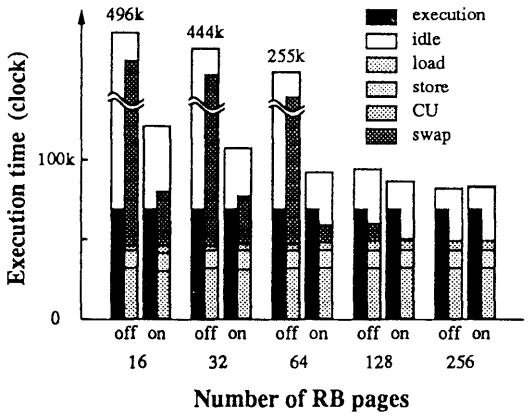

図 6: 負荷制御の効果 (Queen)
off: 負荷制御なし
on: 負荷制御あり

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た．

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場合がある．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．

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と負荷制御を行った場合の実行時間を示す．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，負荷制御を行なった場合，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ー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分かる．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[5]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分かる．

4 従来のキャッシュとの相違

Datarol-II プロセッサでは図 7に示すような階層メモリ構成となっており，RB は OM に対するキャッシュ，そして FU レジスタは RB に対するキャッシュとみなすことができる．そこで，本節では従来のキャッシュと Datarol-II のメモリ構成の違いについて述べる．

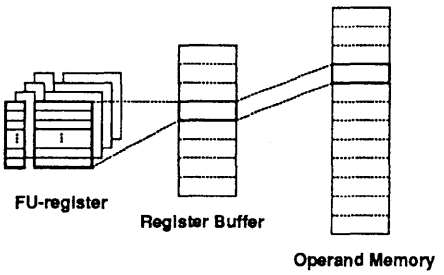

図 7: Datarol-II プロセッサのメモリ階層

4.1 RB-OM 間の制御

Datarol-II プロセッサでは OM から RB へは RQ からのリクエストにより，FU からアクセスされる以前に必要なデータ (論理レジスタセットの内容) があらかじめ読み込まれる (先行制御)．このため，FU からのアクセスは全て高速な RB に対して行われる．

従来のキャッシュではこの先行制御を実現するためプリフェッチという手法が用いられる．プリフェッチには大きく分けて，(1) プリフェッチ命令を用いるもの，(2) 現在アクセスされているアドレス近辺のデータをあらかじめキャッシュに読み込むもの，の 2 種類ある．しかしながら細粒度スレッド処理を前提にした場合，(1) はプリフェッチ命令と実際の参照命令との間を埋める命令を挿入

[5] ここでは OM アクセスレイテンシを 3 としているので 3 倍．

するのが困難，(2) はスレッド１つあたりのデータ数は少なく参照の空間的局所性が低く有効に機能しない，等の問題があり，細粒度スレッド処理には不向きである．

また，Datarol-II プロセッサでは RQ から FU へ実行可能スレッドを送る際，スレッドの情報 (インスタンス番号とスレッド開始アドレス) に加え，RB のページ番号も送られる．したがって，FU からのアクセスは RB のページ番号を用いて行われ，RB のアクセスのたびにページ番号の検索 (アドレス変換) を行う必要がなく，この点も有利である．

4.2 FU レジスタ-RB 間の制御

Datarol-II プロセッサでの FU レジスタを従来のプロセッサにおけるレジスタに対応すると考えると，Datarol-II プロセッサでは自動レジスタロード/ストア機構により，メモリアクセスを load/store 命令を用いずに，さらに通常の処理と並行して行えるため有利である．

また，FU レジスタセットを RB に対する極少量のキャッシュとしてみると，制御は極単純であり，またアドレス変換についても各スレッドで１度のみ行えば良く，この段のキャッシュ追加によるハードウェアコストへの影響は少ないと思われる．

5 おわりに

本稿では Datarol-II プロセッサのメモリシステムおよび高速コンテクストスイッチのための機構について述べ，またソフトウェアシミュレータを用いその効果についての評価を行った．この結果，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のオーバーヘッドの多くを隠蔽でき，細粒度のスレッド処理を効率的に行うことができることを確認した．また，階層的なメモリ構成と負荷制御機構を導入することによりコストパフォーマンスに優れたメモリシステムの実現を可能とした．

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

[3] V.G.Grafe and J.E.Hoch, "The Epsilon-2 Multiprocessor System", Journal of Parallel and Distributed Computing, Vol.10, No.4, pp.309-318, (1990)

[4] S.Kusakabe, T.Hoshide, R.Taniguchi and M.Amamiya, "Parallelism Control and Storage Management in Datarol PE", Proc. IFIP world congress, Vol.1, pp.535-541, (1992)

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

[6] G.M.Papadopulos and D.E.Culler, "Monsoon: an Explicit Token-Store Architecture", Proc. 17th ISCA, pp.82-91, (1990)

[7] R.Taniguchi, T.Kawano and M.Amamiya, "A Distributed-Memory Multi-Thread Multiprocessor Architecture for Computer Vision and Image Processing: Optimized Version of AMP", Proc. 26th ICSS, Vol.1, pp.151-160, (1993)

[8] 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)

[9] 児玉，坂井，山口，"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，32, 7, pp.849-858, (1991)

[10] 立花，谷口，雨宮，"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，30, 12, pp.1628-1638, (1989)

[11] 星出，園田，谷口，雨宮，"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，CPSY91-7, pp.25-32, (1991)

[12] 星出，日下部，谷口，雨宮，"Datarol マシンにおける並列展開戦略"，JSPP'93 論文集，pp.347-354, (1993)