計算機アーキテクチャ 119－42
（1996. 8. 29）

RWC-1におけるスレッド実行と基本性能

岡本一晃† 松岡浩司† 廣野英雄†
横田隆史† 坂井修一††

超並列計算機においては、遠隔メモリ操作や遠隔手続き呼び出しに伴うレイテンシが大きな問題となる。これを解決する手段として、マルチスレッド処理によるレイテンシの隠蔽が挙げられる。さまざまな粒度におけるマルチスレッド処理を効率良く実行するためには、ハードウェアによる効果的な支援が期待できるスレッド処理機構が必須である。超並列計算機RWC-1は、マルチスレッド処理を最適化するスレッド処理機構を有しており、通信や同期のオーバヘッドの削減を図っている。本稿では、RWC-1上のマルチスレッド実行について述べ、その基本性能を示す。またレジスタ転送レベルのシミュレータでの評価で、遠隔メモリ操作や遠隔手続き呼び出しなどが、マルチスレッド処理によるレイテンシ隠蔽によって、効率よく実現できることを示す。

Multithread Execution Mechanisms on RWC-1 and their basic performance

KAZUAKI OKAMOTO,† HIROSHI MATSUOKA,† HIDEO HIRONO,†
TAKASHI YOKOTA† and SHUICHI SAKAI††

Latencies of remote memory access and remote procedure call are serious problems on a massively parallel computer. In order to improve these declines of efficiency, it is quite effective to hide latencies by multithreading. Thread execution mechanism which is effectively supported by the hardware is indispensable to realizes efficiently multithread execution. In this paper, we describe the multithread execution mechanisims of a massively parallel computer RWC-1, and present their basic performance. Both remote memory access and remote procedure call are realized efficiently by multithreading.

1. はじめに

我々は新情報処理開発機構(RWCP)において、中長期的な展望に立った汎用超並列計算機の研究・開発を行っており、その第一段階として、要素プロセッサ数1000規模の超並列計算機RWC-1の開発を進めている[1]。

一般に超並列処理システムにおいては、問題に内在する並列性を最大限に抽出し、それぞれをシステムにおける演算処理ノードの物理的な並列性に写像することが重要である。ここで処理の効率化を図るためには、それぞれの実行単位（アクティビティ）の各演算処理ノードへのマッピングやスケジューリングを最適化することが必要になる。この時、多数の演算処理ノードの間でのアクティビティの並列度をあげると、演算処理ノード間での通信が頻繁に発生する。こうした超並列処理システムにおいて、大きな問題の一つとなるのが、演算処理ノード間の通信や同期、さらにメモリアクセスなどの際に生じるレイテンシである。これを解決する有効な手段としては、マルチスレッド処理によるレイテンシの隠蔽が考えられるが、効果的にレイテンシを隠蔽するマルチスレッド処理を実現するためには、演算処理ノードが高速低オーバヘッドのスレッド切り替え機能を持つ必要がある。特に汎用の超並列計算機においては、命令レベルの細粒度並列処理から逐次性の高い処理まで、さまざまな粒度におけるマルチスレッド処理を効果的にサポートするスレッド処理機構を持つことが必須である。

我々は、超並列計算機のマルチスレッド処理を最適化するプロセッサアーキテクチャとして、既にRICA(Reduced Interprocessor-Communication Architecture)を提案している[3]。現在開発を進めている超並列計算機RWC-1には、このアーキテクチャに基づいてメッセージの授受や同期のオーバヘッドの軽減を図るスレッド処理機構を実装し、その有効性を検証することを目的としている。

本稿ではRWC-1におけるマルチスレッド処理を考え、RWC-1の要素プロセッサRICA-1におけるス

† 新情報処理開発機構
Real World Computing Partnership
†† 電子技術総合研究所
Electrotechnical Laboratory

レッド処理機構とその基本性能について述べる。

2. RWC-1 のスレッド処理機構

2.1 実行モデル

マルチスレッド処理を最適実行するような超並列アーキテクチャを構築するためには、まずスレッドの制御を自然な形で実現できる実行モデルを考えることが重要である。これに対し筆者らは、メッセージによって運ばれるコンティニュエーションが自動的にスレッドを起動するモデル「コンティニュエーション駆動実行モデル」をすでに提案している[2)]。ここでコンティニュエーションとは、スレッドが起動される位置を示す指標である。具体的にはコンティニュエーションは、そのスレッドが起動されるプロセッサ位置(番号)、命令領域およびデータ領域のそれぞれの先頭位置へのポインタの組み合わせで表される。

コンティニュエーション駆動実行モデルにおいては、コンティニュエーションと引数データとの組み合わせが、メッセージの形でプロセッサ内もしくは複数のプロセッサ間を輸送され、このメッセージの到着がスレッド起動のトリガとなる。したがって、本実行モデルに基づくマルチスレッド計算機において処理を効率化するためには、メッセージ受信、スレッド起動、同期処理、メッセージ生成、メッセージ送信といった一連の処理がパイプライン化され、ハードウェアによって効率よく支援される必要がある。

我々が開発を進めている超並列計算機 RWC-1 は、コンティニュエーション駆動実行モデルに基づいたマルチスレッド計算機であり、マルチスレッド処理の最適化を実現する並列アーキテクチャとして提案した RICA の有効性を検証するものである。

2.2 処理機構

前述のコンティニュエーション駆動実行モデルに基づいて、効率良いマルチスレッド処理を実現するために、RWC-1 では以下のようなスレッド処理機構を実装している。

2.2.1 メッセージ処理機構

他プロセッサからのメッセージを受信し、それにより新規スレッドの起動を行う処理機構である。メッセージの処理オーバヘッドを軽減するために、メッセージの到着と同時にデータをレジスタに注入し、かつスレッドを起動する。具体的には、受信したメッセージのコンティニュエーションが制御レジスタに、引数データは汎用レジスタに注入される。また、コンティニュエーションの中の命令番地がプロセッサのプログラムカウンタに書き込まれることで、スレッドの起動あるいは切り替えが起こる。

効果的なマルチスレッド処理の実現のために、現在実行中のスレッドと独立に、次のスレッドを起動するメッセージの処理を実行する。したがって、到着したメッセージのレジスタへの注入は、ハードウェアにより自動的に行われる。さらには、メッセージの注入と新規スレッドの実行はパイプライン化されており、引数データの汎用レジスタへの注入とスレッドの実行が、特にデータ依存関係のない限り並行して行われる。

また、スレッド切り替えのオーバヘッドを軽減するため、レジスタセットを複数用意し、新しいスレッドへの移行をレジスタセットの切り替えだけで行うようにしている。さらにレジスタセットが溢れた場合でも、レジスタの退避および復帰の作業とスレッド実行とを重畳化して、そのオーバヘッドの軽減を図る。

2.2.2 同期処理機構

マルチスレッド処理においては、スレッド間の同期を高速にとることが重要になり、ハードウェアによる支援がその効率化に大きく貢献すると考えられる。しかし並列処理において求められる同期の形態は、一般に処理する問題に依存し、多様である。例えば、新たに fork したスレッドからの戻り値を待つ場合や生産者 - 消費者間のデータ依存関係を保持するための同期、ベクトル演算のような定型的な繰り返し演算におけるベクタの要素同士の対応付けなど、2 つのスレッド間でとる同期もあれば、多数のスレッド間で処理の歩調を合わせるためにとる、バリア同期のようなものもある。こうしたさまざまな形態の同期すべてをハードウェアで支援するのは、実装上困難である。

そこで RWC-1 では、さまざまな形態の同期の効率化を考えるために、全ての同期における基本の形態として 2 つのメッセージ間の同期をとるマイクロ同期を考え、高速なマイクロ同期機構のみをハードウェアで実現している。そして、問題に依存するさまざまな形態の同期に関しては、すべてマイクロ同期をソフトウェア的に組み合わせることで柔軟に対応していく。

同期もメッセージ処理同様、全てをハードウェアにより自動的に処理されるのが効率面からは望ましいが、同期処理にはメモリアクセスが伴うためページフォルトなどの例外が発生する場合が考えられ、すべてをハードウェアで対応するのは実装上困難である。そこで RWC-1 では、マイクロ同期処理機構については専用の命令 (bsync) によって陽に起動することとし、例外発生時の対応をソフトウェアで取れるよう実装している。

2.2.3 メッセージ生成機構

効率良いメッセージの生成および送出のためには、ハードウェアの支援によるメッセージ生成・送出パイプラインが必須である。さらに独立したメッセージ生成・送出パイプラインを設置することにより、他の処理と並列に実行されることが重要である。また、メッセージの生成および送出はソフトウェアによって明示的に実行されるべきである。RWC-1 では専用命令 (mkpkt) によりメッセージ生成・送出パイプラインを起動する。パイプラインの実行には複数クロックを要するが、パイプラインの起動のみを命令から 1 クロックで行うことによ

り、他命令の実行と重畳化される。

RWC-1のメッセージはコンティニュエーションと引数データにより構成されるので、メッセージの生成は、1) コンティニュエーションの生成、2) メッセージ形成、の2ステップから成る。しかし、一つのコンティニュエーションから複数のメッセージを形成する場合が少くないので、コンティニュエーションの生成は独立した命令 (mkcnt) によって実行され、生成されたコンティニュエーションは汎用レジスタ上に保持される。ここで生成されるコンティニュエーションは、メッセージの送信先として利用される他に、新たに起動するスレッドが戻り値を持つ場合の戻り位置としても使われる。この場合、コンティニュエーションはメッセージの引数データとしてプロセッサ間を輸送される。

また、処理のパイプラインが独立して動作するので、メッセージ送出とスレッド終了との非同期化も可能である。すなわち、メッセージ生成のパイプラインを起動すれば、メッセージが出終わっていなくても、スレッドの終了・切り替えができる。

3. 実装と基本性能

前節で、効果的なマルチスレッド処理を実現するRWC-1のスレッド処理機構について述べた。ここではその実装形態として、RWC-1の要素プロセッサRICA-1について紹介する。

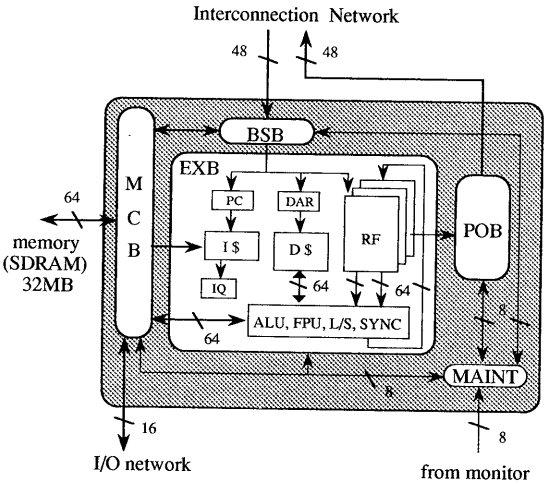
3.1 RICA-1：マルチスレッド型プロセッサ

RICA-1は、RISCアーキテクチャに基づく演算処理パイプラインと、マルチスレッド処理のための通信パイプラインとが高度に融合された、マルチスレッド型プロセッサである。RICA-1の基本構成図を図1に示す。

RICA-1は、メッセージの受信処理部 (BSB)、演算処理実行部 (EXB)、メッセージ生成・送出部 (POB)、外部メモリとのインタフェース部 (MCB)、プロセッサの保守部 (MAINT) の5つのブロックから成る。

演算処理実行部はRISCアーキテクチャに基づき、1) 命令フェッチ、2) 命令デコードおよびオペランド読み出し、3) 演算実行、4) レジスタ書き込み、の4ステージからなる。スレッドの実行中は、このパイプラインが一般的なスーパスカラ型RISCプロセッサとして動作し、演算処理を行う。また、ロード・ストア、メッセージ生成・送出、I/O入出力、浮動小数点演算など実行に複数クロックを要するものについては処理パイプラインを別置し、パイプラインの起動を1クロックで行って、特にデータに依存関係がない限り並列に動作することを可能にしている。さらにI/Oポートを別置し、大量の入出力によりプロセッサ間通信が妨げられることを防いでいる。

一方、マルチスレッド処理のための通信機構は、1) メッセージ処理 (スレッド起動)、2) 同期処理、3) スレッド実行、4) メッセージ生成および出力、からなる循環パイプラインを基本としている。このうち、メッセージ処理とスレッドの実行は、メッセージの到着によってハードウェアが自動的に起動する。これに対し、同期処理とメッセージ生成および送出は、命令実行により起動される。特に同期処理は、データキャッシュへのアクセスが伴うので、効率化のために演算処理実行部の中に実装している。



BSB: Buffering & Scheduling Block
MCB: Memory Control Block
POB: Packet Output Block
MAINT: Maintenance Block
EXB: EXecution Block
PC: Program Counter
DAR: Data Address Register (26bits)
IQ: Instruction Queue
I$: Instruction Cache (4KB)
D$: Data Cache (8KB)
RF: Register File (64bits, 32words, 3sets)

図1 RICA-1の内部構成図

RICA-1のスレッド処理機構の仕様を表1に示す。また、RICA-1のハードウェア諸元を表2に示す。

3.2 評価

RWC-1上のスレッド実行について、レジスタ転送レベルシミュレータによる評価を行った。なお、RICA-1は2命令同時発行型のスーパスカラアーキテクチャを採用している。

3.2.1 スレッドの起動

スレッドの起動はメッセージの到着によって行われる。具体的には、受信したメッセージのコンティニュ

表1 RICA-1のスレッド処理機構の仕様

処理機構	起動方法	パイプライン長
メッセージ受信	自動 (メッセージ到着)	3段
マイクロ同期	命令 (bsync)	4段
メッセージ送信	命令 (mkpkt)	4～14段 (可変長)

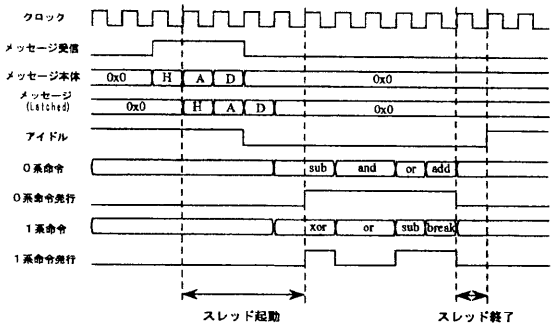


図2 スレッド実行の評価

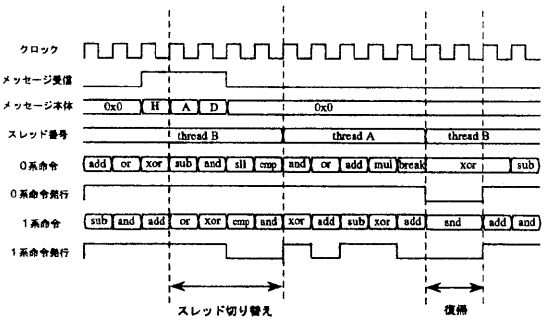


図3 スレッド切り替えの評価

エーションに含まれる論理命令アドレスをプログラムカウンタにセットすることによりスレッドを起動する。図2に示す通り、RICA-1では受信したメッセージを一度内部でラッチし、次のクロックでスレッドを起動して最初の命令フェッチを行う。したがってメッセージを受信してから最初の命令が実行されるまで、4クロックである。

メッセージは最小3ワードで構成され、引数データの個数により最大13ワードまでの可変長であるが、スレッドの起動時間は1クロックである。

3.2.2 スレッドの終了

スレッドの終了は専用命令(break)の実行により実現される。図2に示すとおり、スレッドの終了は1クロックで行われる。

3.2.3 スレッドの切り替え

RICA-1では4レベルの優先度を実装しており、より高い優先度のメッセージを受信することでスレッドの切り替え(プリエンプション)を行う。この時、RICA-1は複数のレジスタセットを備えているので、レジスタ退避のオーバヘッドは生じない。プリエンプションの処理は基本的にはスレッドの起動と同じであるが、図3に示すとおり新しいスレッドの最初の命令が実行されるまでは前のスレッドの命令を実行する。これにより実行効率を落とさずにスレッド切り替えを行う。一方、高い優先度のスレッドがbreak命令を実行することで、元のスレッドに復帰する。この時は命令フェッチ、デコードの2ステージ分だけ、命令実行に隙間が生じる。

表2 RICA-1のハードウェア諸元

LSI実装：	
テクノロジ	0.5 μm CMOS Standard Cell 3 metal layers
パッケージ	527 pin Ceramic PGA
チップサイズ	20.0mm × 20.0mm
ゲート数	Random logic 200k gates Internal RAM 13k bytes
通信ポート：	
入力ポート 出力ポート 転送レート	48 bits × 1 port 48 bits × 1 port 300MB/sec/port
I/O ポート：	
入力ポート 出力ポート 転送レート	16 bits × 1 port 16 bits × 1 port 50MB/sec/port
動作周波数	50MHz
各パイプラインの段数：	
整数乗除算器	8段
浮動小数点演算器	6段
Load/Store	3段
メッセージ生成	4～14段(可変長)

3.2.4 同期処理

ここでは、すべての同期形態の基本としてハードウェアが支援するマイクロ同期機構を評価する。RICA-1のマイクロ同期機構は、専用のbsync命令によって起動される。bsync命令は、先に受信したメッセージをメモリ上に退避し、同期の対になるメッセージを受信した時に、退避した先着メッセージを汎用レジスタ上に復帰して、後続の命令実行を続行する。この際、先着メッセージを受信・処理してから後続メッセージを受信するまでの間は、他のスレッドに実行を切り替えることにより、実行効率を落さない。マイクロ同期の処理時間をレジスタ転送レベルシミュレータ上で評価すると(図4)、先着メッセージを受信してからbsync命令が起動されるまでに4clock、bsync命令が実行され先着メッセージがキャッシュ上に退避される時間が5clock、同じく後続メッセージの受信処理が4clock、bsync命令が実行され先着メッセージが復帰されるまでの時間が8clockである。したがってこのマイクロ同期機構により、一回の同期に要する処理時間は20clockになり、これは、プロセッサが50MHzで動作している時、一回の同期が400nsecで行えることを示している。また、メッセージ受信処理と、先着メッセージの復帰処理の一部とは、他スレッドの実行と重畳化できる。図4の例では、同期処理20clockのうち10clockは、他のスレッドの命令実行とオーバラップしているのがわかる。

3.2.5 リモートメモリ参照

RWC-1において、リモートメモリを参照する場合、リモートプロセッサにメッセージを送信し、リモートプ

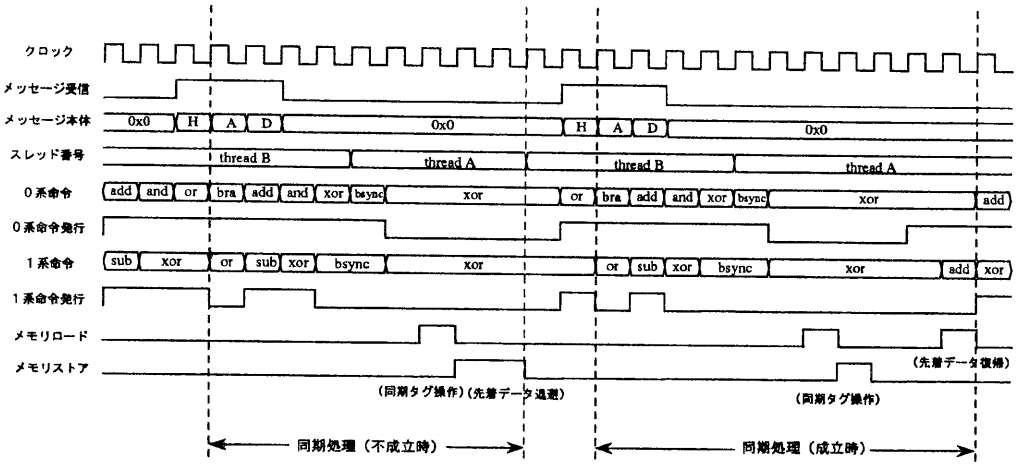


図4　マイクロ同期処理の評価

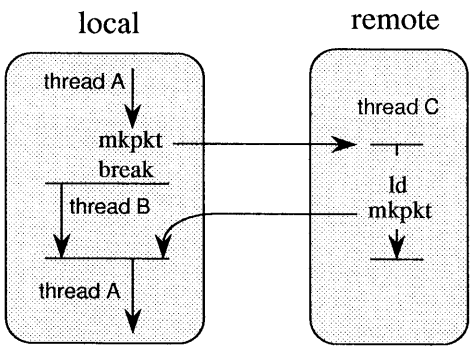


図5　リモートメモリ参照

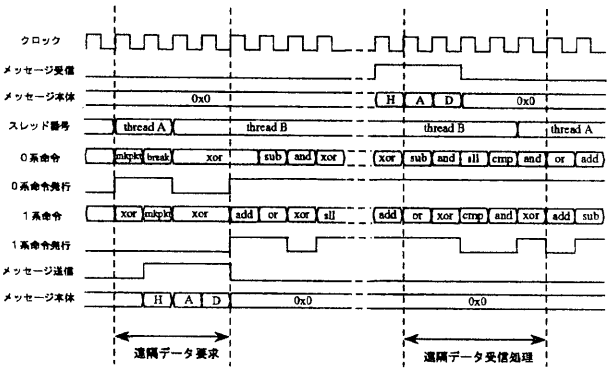


図6　リモートメモリ参照の評価 (caller 側)

ロセッサ上にメモリ参照のスレッドを起動する方法と、I/O 転送路を介して DMA 転送を行う方法とがある。後者はページ単位の大量データのコピーに用いられるが、ここでは細粒度マルチスレッド処理により関係の深い前者について評価する (図 5)。

リモートメモリを参照したい場合、まず caller 側のプロセッサではメッセージを生成・送信して、現スレッドを中断する。そして、別のスレッドに実行を切り替える。リモート側からメッセージが返ってきたら、元のスレッドが復帰され、処理が続行される。ここで、リモートメモリ参照に要する時間は、メッセージの生成・送出に 4clock、callee 側での処理 (メッセージ受信、メモリ参照、メッセージ生成・送出) に 13clock、さらにメッセージの受信・汎用レジスタへの注入に 5clock、の計 22clock にネットワークの転送時間を考慮したものとなり、これらはそのままレイテンシとなって、避けることができない。しかし図 5 に示すマルチスレッド処理を採用すると、メッセージ生成パイプラインの起動 (mkpkt 命令の実行) に 1clock、thread A から thread B への切り替え (break 命令の実行) に 1clock の、計 2clock の処理時間後は thread B が実行される。その後 callee 側から戻ってくるメッセージは、thread B の実行と並行して受信され、レジスタに注入される。この時 thread B から thread A への復帰は、ハードによって自動的に行われるので、ここでオーバヘッドは生じない。したがって、このマルチスレッド処理によるリモートメモリ参照では、最初の 2clock 以外のレイテンシを thread B の実行により隠蔽することができ、実行効率を落さない。このようなマルチスレッド処理は、例えば遠隔データのプリフェッチの際に有効である。遠隔データのプリフェッチを行うプログラムを、レジスタ転送レベルのシミュレータ上で実行した結果を図 6 に示す。ここで、thread A は遠隔データのフェッチを行うスレッド、thread B はその遠隔データとは直接は関係ないローカルな処理である。

3.2.6　遠隔手続き呼びだし

遠隔プロセッサへの手続き呼び出しについて考える。新たなスレッドを fork する際に必要な処理は、戻り値がない場合、caller 側では新規スレッドのコンティニュエーションを生成し、引数とともにメッセージに載

表3 遠隔手続き呼び出し処理

	caller 側		callee 側	
	処理時間 (clock)	オーバヘッド (clock)	処理時間 (clock)	オーバヘッド (clock)
remote fork				
(1 arg)	8	5	11	7
(7 args)	18	5	15	7
remote fork with return				
(1 arg)	23	19	13	8
(7 args)	30	19	16	9

せて送出することである。一方、callee 側では、メッセージを受けとった後、スレッドを起動するための新たなフレームを獲得しなければならない。さらにスレッドが戻り値を持つ場合、caller 側では戻り位置 (return continuation) の生成を行い、callee 側ではそれを用いて戻り値を載せたメッセージを生成し、送出する処理が加わる。遠隔手続き呼び出しの処理をレジスタ転送レベルシミュレータ上で実行した時の処理時間を表 3 に示す。また、これらの処理時間のうち、マルチスレッド処理により隠蔽できないものを、オーバヘッドとして示す。

以上のように、RWC-1 では複数のプロセッサにまたがる処理においても、レイテンシを隠蔽し数クロック程度のオーバヘッドでこれが実現できることが示された。これは、RWC-1 の要素プロセッサ RICA-1 が、細粒度のマルチスレッド処理を効率よく実行できることを示している。

4. おわりに

本稿では、超並列計算機 RWC-1 のスレッド処理機構を示し、レジスタ転送レベルシミュレータによって基本性能の評価を行った。

RWC-1 では遠隔メモリ操作などのレイテンシを、マルチスレッド処理により隠蔽することを目的に、効果的なマルチスレッド処理を実現するスレッド処理機構を実装している。スレッドの起動や切り替えを高速に行うために通信の手段であるメッセージによって直接スレッドを起動し、そのためメッセージの受信処理や生成・送出などの通信パイプラインが、ハードウェアの支援により効率よく動作することを示した。そして、これを実現するスレッド処理機構を RWC-1 の要素プロセッサ RICA-1 に実装し、効率良くマルチスレッド処理が行われることをレジスタ転送レベルシミュレータ上の評価により示した。

RICA-1 はランダムゲート数 20 万ゲート、内蔵メモリ 13KB のゲートアレイによって実現される、超並列計算機向けマルチスレッド型プロセッサである。RISC アーキテクチャに基づいた演算処理パイプラインと、大域並列処理を容易化する通信パイプラインとが高度に融合し、また複数のレジスタセットを用意することで、高効率のマルチスレッド処理を実現している。

今後は、レジスタ転送レベルのシミュレータを用い、複数プロセッサの動的な挙動についてより詳細な評価を行うとともに、実機の開発を進めて実機上での評価を行う予定である。さらにこのプロセッサを 1024 台実装する、超並列計算機 RWC-1 の開発を進めていく予定である。

謝辞

本研究を遂行するにあたり、有益な御指導、御討論をいただいた島田 RWC 研究所長、石川超並列ソフトウェア研究室長、超並列ソフトウェア研究室員の諸氏、ならびに RWC 超並列アーキテクチャワーキンググループの諸氏に感謝いたします。

参考文献

1) S. Sakai, et al.: RWC-1 Massively Parallel Architecture, Proc. of HPCC'94, pp.33-38 (1994).
2) S. Sakai, K. Okamoto, Y. Kodama and M. Sato: Reduced Interprocessor-Communication Architecture for Supporting Programming Models, Proc. of Programming Models for Massively Parallel Computers 1993, pp.134-143 (1993).
3) S. Sakai, Y. Kodama, et al.: Reduced interprocessor-communication architecture and its implementation on EM-4, Parallel Computing Vol.21, No.5, pp.753-769 (1995).
4) 松岡他、超並列計算機 RWC-1 用プロセッサチップの設計、信学技報 CPSY95-18 (1995).
5) 岡本他、超並列計算機 RWC-1 の命令セットアーキテクチャ、信学技報 CPSY95-36 (1995).
6) 岡本他、RWC-1 のマルチスレッド処理機構、情処研報 95-ARC-113 (1995).
7) 横田、松岡、岡本、廣野、坂井：超並列向け相互結合網 MDCE の提案と評価、情報処理学会論文誌、Vol.36, No.7, pp.1600-11609 (1995).