推論プロセッサ UNIRED II：プロセッサ・アーキテクチャの評価†

島田 健太郎†† 小池 汎平†† 田中 英彦††

推論プロセッサ UNIRED II は並列マシンの要素プロセッサとして，Committed Choice 型言語 Fleng を高速に実行するために設計された専用プロセッサである．並列推論マシン PIE 64 の推論ユニットにおいて，ネットワーク・インタフェース・プロセッサ，管理プロセッサと協調動作する．UNIRED II は多重コンテクスト処理機構，三つのメモリ・バスなど特徴あるプロセッサ・アーキテクチャを採り，Fleng の処理を効率化する強力な命令セットを備えて汎用プロセッサに比べて高い性能を達成することを狙う．多重コンテクスト処理とは，パイプラインに複数の命令流（コンテクスト）を流し，インタロックやリモートなメモリ参照の待ち合わせを回避して動的に充足率を高めるものである．本論文では，UNIRED II 単体の評価のために作成したシミュレータにより行った，プロセッサ・アーキテクチャの評価結果について述べる．評価結果では，十分な単体性能 (naive reverse 30 で約 900 KRPS, 8 queen で約 600 KRPS) が達成されることを確認した．また，多重コンテクスト処理によるパイプライン・インタロックの抑制やリモート・アクセスのレイテンシに対する効果などを示し，UNIRED II のアーキテクチャが並列マシンの要素プロセッサとして適した特性を持っていることを確かめた．

1. はじめに

論理型言語の効率的な並列実行を目指して作られた Committed Choice 型言語は，論理型言語としての特性のほかに並列実行の機能を言語仕様として含んでいる．このため，従来の逐次的な手続き型言語を実行することを前提に設計されている汎用プロセッサの上では，大きなセマンティクス・ギャップがあるために，特に複数台での並列動作においてオーバヘッドを抑えて高い処理性能を得ることには困難が予想される．そこでわれわれは Committed Choice 型言語 Fleng[7] を実行する並列推論マシン PIE 64[8] に対し，その要素プロセッサの中核として，Fleng の並列実行に特に専用化した推論プロセッサ UNIRED II を設計した．同様の目的で開発されている並列推論マシンとしては，他に ICOT の PIM/p[5]，PIM/m[6] などがある．これらは Committed Choice 型言語 KL 1[4] を実装し，UNIRED II と同じく専用化された要素プロセッサを持つ．

UNIRED II の設計に当たり，われわれは特に次の二つのことを基本的な目標とした．

- Fleng の実行が汎用プロセッサに比べて十分高速化されること
- 並列マシンの要素プロセッサとして特に適した特性を備えること

われわれは前者についてはワークステーション上に Fleng の処理系を作成した経験に基づき，特に Fleng をコンパイルして高速に実行するための命令セットを設計した．また手続き型言語に比して多くなるメモリ・アクセスに対し，命令，データの読み書きを分離した三つのメモリ・バスを設けた．後者については，多重コンテクスト処理を採り入れることで複数台の並列動作時の同期待ちのオーバヘッドを抑えた．さらにプロセッサ間通信／同期／プロセス管理などへの接続点として専用のコプロセッサ・コマンド・バスを設け，実用規模の並列マシンの構成に対し柔軟な対応を可能とした．

UNIRED II は CMOS ゲートアレイ LSI として実現されるが，ここではその回路の詳細設計に基づいてクロック・レベルのシミュレータを作成し，実チップでの評価に先立ってシミュレーションによるアーキテクチャの評価を行った．今回行った評価では主に多重コンテクスト処理の効果などプロセッサ・アーキテクチャを中心としており，命令セットの評価については別の機会に譲る．次章以降，まず並列推論マシン PIE 64 および推論プロセッサ UNIRED II の概要を述べ，今回の評価の中心となる多重コンテクスト処理について説明する．次に UNIRED II のパイプライン構成およびリモート・メモリ・アクセスの機構について説明した後，今回行ったシミュレーションのモデルについて述べ，シミュレーションによる評価結果を

† The Inference Processor UNIRED II : Evaluation of Its Processor Architecture by KENTARO SHIMADA, HANPEI KOIKE and HIDEHIKO TANAKA (Department of Electrical Engineering, Faculty of Engineering, University of Tokyo).

†† 東京大学工学部電気工学科

示し，考察をする．

2. 推論プロセッサ UNIRED II

2.1 並列推論マシン PIE 64

初めに並列推論マシン PIE 64 の構成について簡単に述べる．

PIE 64 は 64 台の推論ユニット（Inference Unit: IU）を回線交換方式の3段オメガ・ネットワークで接続した構成を採る[13]．このネットワークは負荷情報に基づいて最小負荷 IU を選ぶ自動負荷分散接続機能を持ち，同じものが2系統（Data Access/Allocation Network: DAAN および Process Allocation Network: PAN）ある．

各 IU には UNIRED II の他に，ネットワークと接続して Fleng 向きの高度な通信／同期機能を提供するネットワーク・インタフェース・プロセッサ (NIP)[9),12)]，プロセス管理，入出力等を行う管理プロセッサ（MP）がある[11)]．MP にはプロセス管理・負荷分散戦略などに柔軟な対応を可能とするため，汎用プロセッサ SPARC を用いている．**図1**に PIE 64 の IU の構成を示す．このように，並列マシンの要素プロセッサにおける処理は逐次計算のプロセッサにおける処理に比べて機能的に多くのものがあるが，これらは分割して並列に実行することが可能である．PIE 64 ではこれらを「計算」「通信・同期」「管理」の三つに分け，それぞれ UNIRED II，NIP，MP が分担して並列に実行する．これらの三種のプロセッサはローカル・メモリをバス結合で共有するほか，専用のコプロセッサ・コマンド・バスによってコマンド／リプライの送受を行い，協調して動作する．これらのプロセッサ間のコマンドは，Fleng のプロセスやデータを扱うのに適するように決定されている．

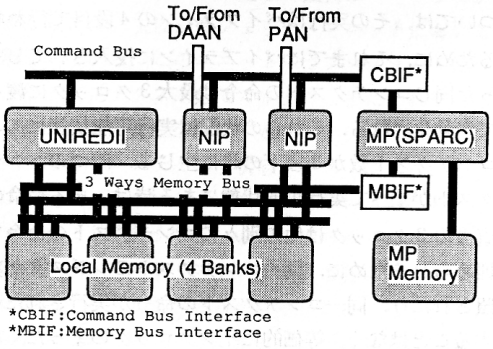


図 1 PIE 64 の IU の構成
Fig. 1 Organization of the inference unit of the parallel inference machine PIE 64.

2.2 UNIRED II の特徴

推論プロセッサ UNIRED II は並列マシンの要素プロセッサとして，Committed Choice 型言語 Fleng を効率良く実行するために設計された専用プロセッサである．その主な特徴としては次のようなことが挙げられる．

- 記号処理向きの機能としてタグ・アーキテクチャを採用，Fleng の処理を効率化している．
- 命令バス，メモリ読み出しバス，メモリ書き込みバスの三つの分離したメモリバスを持ち，バンド幅の広い並列メモリアクセスを行う．
- 多重コンテクスト処理によって動的なパイプライン充足率の向上を行う．
- Fleng の処理を効率化する強力な命令セット[10)]を持つ．

UNIRED II では，すべての命令は1ワード（32ビット）長であり パイプラインの1クロックで実行される．これらには並列環境下では重要な Test and Set を行うような命令も含まれ[10)]，このような命令は2回のメモリ・アクセスが必要であるが，UNIRED II では読み出し／書き込みを分離したメモリ・バスをパイプラインの別々のステージで扱うことにより，シングル・サイクル命令として実現した．またデータのワード長も同じく 32 ビットであり，ガーベジ・コレクションなどのマーク部2ビットのほか，タグ部2ビット，データ部28ビット（ポインタ型），あるいはタグ部6ビット，データ部24ビット（定数型）からなっている．

2.3 多重コンテクスト処理

UNIRED II ではその大きな特色として，各パイプライン・スロットに複数のコンテクストからの命令を動的に投入して実行する多重コンテスト処理を行っていることがある．これは，**図2**に示すように，プロセッサ内にあらかじめ複数のコンテクスト（プロセス）を投入しておき，各クロックごとに実行可能なコンテクストから命令単位にスケジューリングを行って，パイプラインに命令を投入するものである．すなわち，UNIRED II はパイプライン共有型の MIMD プロセッサとして動作する．これにより，パイプライン化された命令の実行においてパイプライン・インタロックの低減を行うことが可能なほか，他 IU へのリモートなデータ参照の待ちが生じた時に速やかにそのコンテクストを待機状態にすることによって，別

のコンテクストの命令でパイプラインを充足することができる．これらは Fleng が並列言語であることから，実行時に多数のプロセスが生成されることに着目し，これを単一プロセッサ内でも積極的に利用したものと言える．

UNIRED Ⅱ の多重コンテスト処理における大きな特徴は，実行可能なコンテクスト数が少ない時には，同じコンテクストの命令も連続してパイプラインに投入できることである（図2の5〜7段目）．この時同一コンテクスト内の命令は，通常のプロセッサと同じようにインタロックを掛けて依存関係を保証しながら動作する．このため，並列度が低下して実行可能なコンテクスト数が少なくなった時にも，極端に大きな性能低下はない．同様な機能を備えたものとしては，他に HEP[1]，MASA[2]，FLATS 2[3] などが挙げられるが，これらは循環パイプライン構成となっており，UNIRED Ⅱ のような同一コンテクストの連続投入は行っていない．これに対し UNIRED Ⅱ は多数台を結合して並列計算機を構成することが前提条件となっており，十分なコンテクスト数を保つためには（台数）×（コンテクスト多重度）だけの並列度が常に必要となる．そのため，並列度が低下して実行可能なコンテクスト数が減った時にも，ある程度性能を保つことを目指したのである．

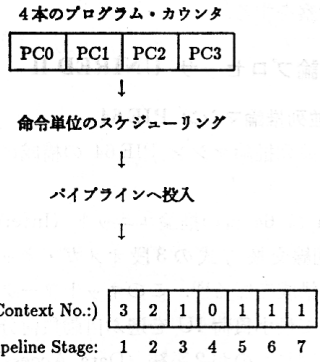


図 2 多重コンテクスト処理の流れ
Fig. 2 The multi-context processing mechanism of the inference processor UNIRED Ⅱ.

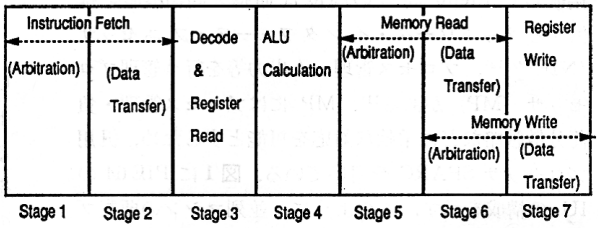


図 3 UNIRED Ⅱ のパイプライン構成
Fig. 3 Pipeline organization of the inference processor UNIRED Ⅱ.

現在最大のコンテクスト数は，回路規模を適当な範囲に抑えるために4としている．

3. UNIRED Ⅱ の内部構成

3.1 UNIRED Ⅱ のパイプライン構成

UNIRED Ⅱ は全体で7段のパイプライン構成を採る（図 3）．段数が多いのは，PIE 64 の IU 上ではローカル・メモリを NIP，MP とバス結合で共有することから，各メモリ・アクセスがそれぞれアービトレーション・フェーズとデータ転送フェーズの2段になっていることと，メモリに対する Test and Set 型の処理を連続して行うために，メモリ読み出しとメモリ書き込みがパイプラインの別々のステージで行われることによる．図3において，1段目および2段目で命令フェッチが行われ，3段目で命令のデコードおよびレジスタ読み出し，4段目で実行が行われる．さらに，5，6段目においてメモリの読み出し，6，7段目でメモリの書き込み，また7段目ではさらにレジスタへの書き込みが行われている．

パイプライン構成で重要なことは，パイプライン・インタロックがどこでどのような時に起こる可能性があるかである．UNIRED Ⅱ では，まずメモリからの読み出しデータの遅延に対して，これは6段目で読み込まれるために，直後の3クロック以内にその宛先レジスタを（3段目において）参照すると，データ待ちとなってインタロックが生じる．またジャンプ命令については，その実行はパイプラインの4段目で行われるために，それまでにパイプラインに投入されてしまった同じコンテクストの命令は最大3クロックに渡って無効化される．これらの状況は実行可能状態にあるコンテクスト数が3以下の時に生じる．四つのコンテクストがすべて実行可能状態にある時は，ある命令の直後の3クロックは他の別々のコンテクストの命令で埋められるために，同一コンテクストのレジスタが参照されたり，同一コンテクストの命令が実行されたりすることはなく，等価的にインタロックのまったくないパイプラインが実現される．

3.2 UNIRED II の内部構成

推論プロセッサ UNIRED II では，図3のパイプライン構成を実現するために図4のような内部構成を採っている．これらは汎用レジスタ・ファイルの他，次の六つのブロックよりなる．

- 命令フェッチ・ユニット（Instruction Fetch Unit）
  命令バスを通じて命令フェッチを行うユニット．多重コンテクスト処理を行うためのコンテクストのスケジューリング機能を含む．
- 命令実行ユニット（Decode & Execution Unit）
  命令のデコード，レジスタ読み出し，および ALU 演算を行うユニット．レジスタ間演算命令はこのユニットで実行される．
- ヒープ管理ユニット（Heap/Scratch Unit）
  ヒープ・レジスタ，スクラッチ領域レジスタを含み，ヒープ領域等の管理を行うユニット．
- ロード/ストア・ユニット（Load/Store Unit）
  ロード命令やストア命令などによるメモリ参照を実行するユニット．
- コマンド送信ユニット（Command Sender）
  コプロセッサ・コマンド・バスを通じてコプロセッサ・コマンドを発行するユニット．
- コマンド受信ユニット（Command/Reply Receiver）
  外部から送られてきたコプロセッサ・コマンドを受信，および先に発行したコプロセッサ・コマンドの結果のリプライを受け取るユニット．

前述した多重コンテクスト処理の機構はハードウェア構成としては，主に命令フェッチ・ユニットにおいて実現されている．このユニットにはプログラム・カウンタが4コンテクスト分で4本あり，またそれぞれのコンテクストの状態を保持するコンテクスト状態レジスタがある．コンテクスト状態レジスタで示される状態は empty（空きコンテクスト），active（実行可能），sleeping（リプライ待ち）の三つである．このユニット内のコンテクスト・スケジューラは各コンテクストの状態レジスタの値から各クロックごとに実行すべきコンテクストを決定する．スケジューリングのアルゴリズムは単純で，コンテクスト番号の順に active の状態にあるコンテクストを探すだけである．このため，

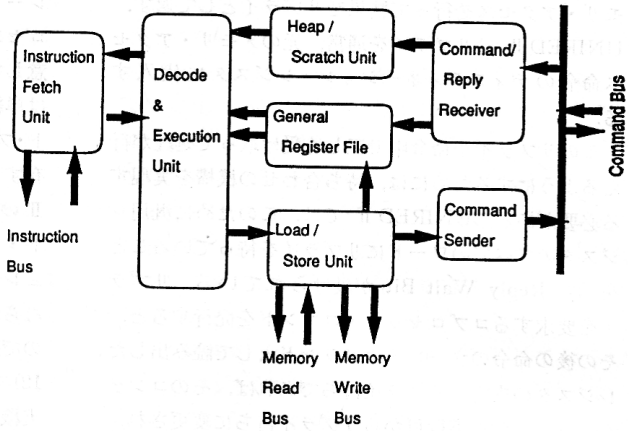


図 4 UNIRED II の内部構成
Fig. 4 The internal block diagram of UNIRED II.

コンテクスト・スケジューラも状態数5（コンテクスト数＋すべてが empty である状態）の順序回路で済んでいる．

命令フェッチ・ユニット以外のユニットにおける多重コンテクスト処理に関する機構としては，汎用レジスタ・ファイルが4コンテクスト分で4倍になっていることのほかは，各パイプライン・ステージにどのコンテクストがそれぞれ存在するのかを示す2ビットのコンテクスト番号レジスタがあるだけである．汎用レジスタ・ファイルについては，そのハードウェアコストとしては約1万ゲートほどかかっており，これはチップ全体のほぼ 1/4 に当たる．なおパイプライン部分の回路はすべて人手で設計されているが，レジスタ・ファイル部分は CAD による自動生成で設計された．

3.3 リモート・アクセスの機構

UNIRED II では，メモリ参照を行う命令では基本的にリモートなメモリ参照を行う機能も併せ持っており，リモート／ローカルの区別をしない統一的なメモリ・アクセスが可能である．UNIRED II は内部にプロセッサ ID レジスタを持ち，メモリ参照命令の実行において対象となるメモリ・アドレスのプロセッサ ID 部（PIE 64 では 64 台でのグローバルなメモリ・アドレス 28 ビットの上位6ビット）と自分のプロセッサ ID レジスタの内容を比較し，これが異なっていると他プロセッサのメモリへのリモートな参照であると判断してリモート・メモリ・アクセスを行うコプロセッサ・コマンドの発行に切り替わる．PIE 64 ではこのリモート・メモリ・アクセス・コマンドは各 IU において NIP が受け取り，ネットワークを通じたメ

モリ・アクセスを行って結果をリプライとして返す．UNIRED Ⅱ はリプライを通常，元のメモリ・アクセス命令のディスティネーション・レジスタに代入する．

このリプライを命令中で正しく受け取って処理が行えるようにするためには，待ち合わせの機構を実現する必要がある．UNIRED Ⅱ では，このために汎用レジスタのすべてのワードにリプライを待っていることを示す Reply Wait Bit が設けられている．リプライを要求するコプロセッサ・コマンドを発行すると，その後の命令のソース・オペランドとして読み出したレジスタの内容がリプライ待ちであれば，そのコンテクストの状態が実行可からリプライ待ちに変更され，その命令は取り消されて実行が停止する．その後コプロセッサ・コマンド・バスでリプライが受け取られると，コンテクストの状態が再び実行可に戻り，停止した命令から実行が再開される．

4. シミュレーションによる評価

4.1 シミュレーション・モデル

今回行ったシミュレーションはレジスタ・トランスファ・レベルのものであり，UNIRED Ⅱ 単体での評価を行うことを目的としている．UNIRED Ⅱ では多重コンテクスト処理を行っているため，単体でも最大四つのプロセスの並列動作が可能である．

図 5 に今回行ったシミュレーションのモデルを掲げる．UNIRED Ⅱ 単体の評価が目的であり，3 章で述べた UNIRED Ⅱ の内部については正確に再現しているが，他の部分については適当なエミュレーションを行っている．すなわち，ローカル・メモリは三つのバスにより常に最大の速度でアクセスできる 3 ポート・メモリとし，また UNIRED Ⅱ から発生するコプロセッサ・コマンドを受けて NIP，MP の動作をエミュレートするコマンド・プロセッサ，およびゴール（プロセス）・キューを置く独立したキュー・メモリを用意している．ゴール・キューによるスケジューリングは単純な LIFO (Last-In/First-Out) である．コマンド・プロセッサはローカル・メモリを UNIRED Ⅱ と共有するが，これは 3 本のバスのうち W バス (UNIRED Ⅱ のデータ書き込みバス) に結合されている．コマンド・プロセッサによるコプロセッサ・コマンドのエミュレーションに要するクロック数を**表 1** に掲げる．これらのクロック数については，特に NIP に対するもので結果をリプライとして返してくるコマンドは文献 12) によるものとほぼ同等である．また，PIE 64 の実機では MP と 4 台の NIP* が並列動作するが，今回用いたシミュレーション・モデルでは図 5 に示すようにコマンド・プロセッサ 1 台しか考えていない．この違いをいくらか吸収するために，MP/NIP 別に長さ 4 のコマンド受信キューを設けて，UNIRED Ⅱ 内の各コンテクストからの先行したコマンド発行を許すようにした．各コマンドの詳細は文献 11)，12) を参照されたい．

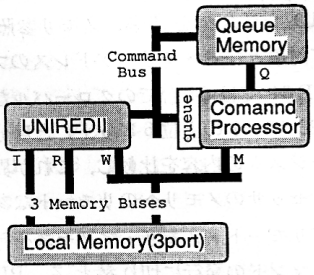


図 5 シミュレーション・モデル
Fig. 5 The simulation model used for the evaluation.

4.2 単体での性能

初めに，UNIRED Ⅱ 単体でリモートなメモリ参照のない状態で性能評価を行った．用いたサンプル・プログラムは append 100 (長さ 100 のリストの append)，nreverse 30 (長さ 30 のリストの naive reverse)，quick sort 50 (長さ 50 のリストの quick sort)，primes 100 (100 までの素数の生成)，8 queen (8 クイーン問

* PIE 64 の実際の IU では，NIP はネットワークの各系統に master/slave が 1 個ずつ計 4 個付き，また MP のコマンド・バス・インタフェースには同様のコマンド・キューが設けられている．

表 1 コマンド・エミュレーションに要するクロック数
Table 1 The number of clock cycles which are necessary for emulating co-processor commands issued by the inference processor UNIRED Ⅱ.

コマンド	説明	送出先	リプライ	クロック数
newgoal	子ゴールのキューへの登録	MP	無	1
endreduce	ゴール実行終了の通知	MP	無	1
suspend	サスペンドの通知	MP	無	7
deref	変数のデレファレンス	NIP	有	16
bind	変数の束縛	NIP	有	17
read 2	リモートなリスト・セルの読み出し	NIP	有	19
activates	ローカルな変数からのアクティベイト	NIP	無	8

題）である．**表2**にこれらのサンプル・プログラムの概要を示す．また，これらのプログラムによる性能測定の結果を**図6**に示す．クロックを10MHz（設計値）とした時の値である．

図6でappend 100は並列度が全くないので（すべてTail Recursion）平均のコンテスト数が1を越えず多重コンテクスト処理が行われないため，パイプライン・インタロックにより性能が低下している．すなわち表2によれば，append 100の1リダクション当たりの命令数はほぼ8であるので，もしインタロックを避けるのに十分なコンテクスト数があれば1.25 MRPS* の性能が見込める．しかし，コンテクスト数が1を越えないため3.1節で述べたようにインタロックが生じ，CPIが約1.8まで増加して性能低下を招いている．**図7**に各プログラムでの平均コンテクスト数を示す．

* Reduction Per Second.

図7では，他にnreverse 30も平均コンテクスト数が3強で4に満たないため多重コンテクスト処理の効果が十分に働かず，若干の性能低下を生じている．表2によれば1リダクション当たり約9.8命令であるので約1MRPSの性能が期待されるが，図6では900 KRPS程度となっている．これに対しqsort 50，8 queenでは，append 100，nreverse 30より1リダクション当たりの命令数が多いものの（表2），十分なコンテクスト数が得られているのでCPIもほぼ1となっている．

また図6でprimes 100の性能が低下しているのは，主としてUNIRED Ⅱが乗算／除算器を持っていないことから，これに含まれる整数除算を命令列で展開して行っているためである．他のプログラムに比べこのprimes 100では，実行される命令の大部分が除算を展開して実行するための算術演算と条件分岐命令である．表2においても1リダクション当たりの命令

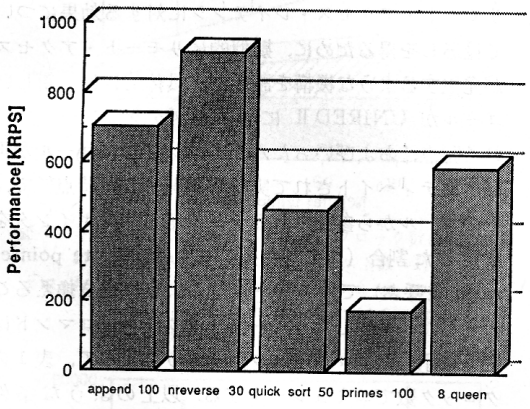


図6 サンプル・プログラムによる性能
Fig. 6 Simulated performance of the inference processor UNIRED Ⅱ measured by the sample programs.

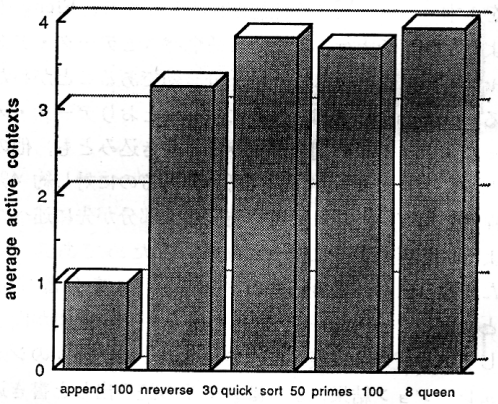


図7 平均コンテクスト数
Fig. 7 Average number of active contexts in the inference processor UNIRED Ⅱ measured by the sample programs.

表2 シミュレーションに用いたサンプル・プログラム
Table 2 Several aspects of sample programs which are revealed by the simulation.

プログラム	append 100	nreverse 30	qsort 50	primes 100	8 queen
総クロック数	1435	5427	8162	41068	656011
リダクション回数	101	496	380	726	38878
サスペンド回数	0	29	122	103	558
総実行命令数	816	4858	7747	39352	647933
1リダクション当たりの命令数	8.08	9.79	20.39	54.20	16.67
CPI (Clock cycles Per Instruction)	1.759	1.117	1.054	1.044	1.012
データ・メモリ・アクセス回数 (read)	306 (21.3%)	1843 (34.0%)	2317 (28.4%)	1764 (4.3%)	165833 (25.3%)
(write)	301 (20.1%)	1663 (30.6%)	1996 (24.5%)	1547 (3.8%)	141134 (21.5%)

（（ ）内は総クロック数に対する割合）

数が他より多い（54.2 で qsort 50，8 queen などの倍以上）のは，この展開された除算の1回の実行に 120 ステップほどかかることによる．このような特性は他にも浮動小数点計算等を含む応用プログラムで問題となるが，PIE 64 では各 IU に MP として汎用プロセッサ SPARC とその FPU が搭載される予定であるので，時間のかかる数値計算などはそれらに任せてしまうことが可能である．これについては，PIE 64 のシステムとして別途評価を行う予定である．

4.3 メモリバスの効果

次に，3本のメモリ・バスの有効性を調べるために，バスの本数をシミュレータ上で疑似的に変えて性能測定を行った．結果を**図8**に示す．メモリ・バスを1本にした場合（1 bus），命令フェッチ・バスとデータ・アクセス・バスの2本にした場合（2 bus），命令フェッチ／データ読み出し／データ書き込みの3本にした場合（3 bus）の，各サンプル・プログラムでの性能である．図では primes 100 以外ではメモリ・バスを1本にすると大きく性能が低下しており，Fleng のような言語を実行する上では命令バスとデータ・アクセス・バスを分けることは十分有効であることがわかる．primes 100 では，表2に示したとおりデータ・メモリ・アクセスの割合が読み出し書き込みとも，他の四つのプログラムが 20〜30% であるのに対し約 4% 程度しかない．これは実行時間の大部分が先に述べたように除算の実行に費やされているためである．また，読み出し／書き込みのメモリ・バスを分離すると，同じく primes 100 以外では 5〜10% 性能が向上している．さらに表2に示されるように，今回のシミュレーション結果によればメモリの読み出しと書き込みの割合がほぼ同等で，一般に通常の手続き型言語の場合に比べ書き込みの割合が高い．読み出し／書き込みのバスを分離することは，そのような特性にも適していると考えられる．

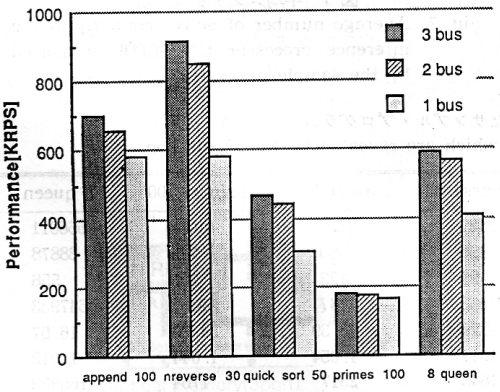


図 8 メモリ・バスの数と性能

Fig. 8 Performance vs. number of the memory buses of the inference processor UNIRED Ⅱ.

ここで述べた評価は，UNIRED Ⅱ 単体からみたメモリ・バスの本数の効果についてであるが，PIE 64 の実際の IU では UNIRED Ⅱ のほかに NIP，MP からのメモリ・アクセスがあり，これらのトラフィックを3本のバスを最大限に利用して分割するので，さらに大きな効果を期待できる．これについては IU の実機稼働後に実機で評価を行う予定である．

4.4 リモート・アクセス・レイテンシに対する効果

UNIRED Ⅱ では，多重コンテクスト処理によって複数の命令流で動的にパイプラインを充足するので，リモート・アクセスのレイテンシを緩和する効果を期待できる．今回行ったシミュレーションは図5に示したように UNIRED Ⅱ 単体での評価が目的であるが，リモート・アクセス・レイテンシに対する効果について見通しを得るために，疑似的にリモート・アクセスが発生するような機構を設けた．具体的には，新しいゴールが UNIRED Ⅱ によりリダクションを開始される時点，およびいったんサスペンドしたゴールが再びアクティベイトされて実行が再開された時点* で，そのゴールから参照されるデータ構造中のポインタを指定した割合（この割合をここでは remote pointer ratio と呼ぶ）でリモートになるように書き換えることとした．発生したリモート・アクセス・コマンドは図5に示したコマンド・プロセッサによって，表1のクロック数でエミュレートする．以上のような条件で，UNIRED Ⅱ に投入される最大のコンテクスト数を1から4まで変えてみた時の総クロック数の変化およびその内訳を測定した．結果を**図 9〜11** に示す．用いたサンプル・プログラムは 8 queen である．図ではそれぞれ下から実行された命令数，ジャンプなどのためフェッチした後無効化された命令数，パイプラインがインタロックにより停止したクロック数，リプライ待ちなどで実行可能なコンテクストがなくパイプラインがスリープしたクロック数である．すべての合計が総実行ロック数となる．

図で最大のコンテクスト数が1の時（図9）は，多重コンテクスト処理が行われないため，リモート・

* このようにしたのは，シミュレーションはあくまで UNIRED Ⅱ 単体で行っているので，アクティベイトの原因となる変数の束縛処理では常にローカル・データに具体化されるためである．

アクセスが増えるに従って，そのリプライ待ち合わせでパイプラインがスリープしている期間（図の白い部分）が大きく増大している．最大コンテクスト数が2以上になると（図 10, 11），多重コンテクスト処理によってパイプラインがスリープしている期間は減少しており，最大コンテクスト数が増えるに従って，その効果も大きくなっている．これにより多重コンテクスト処理がリモート・アクセスのレイテンシに対して有効に働いていると言うことができる．さらにこれらの図では，最大コンテクスト数が増えるに従って命令の無効化（図の斜線部分）およびパイプラインのホールド（図の薄い網掛け部分）のパイプライン・インタロックも抑えられていることがわかる．

また逆に，インタロック機構を入れて同一コンテクストの連続実行も可能としたために，最大コンテクスト数が1の時（図9）でも4の時（図11）に比べ約半分程度の性能が得られている．さらにインタロック機構の有効性を確認するために最大コンテクスト数が1でインタロック機構がない場合のシミュレーション結果を**図 12** に掲げる．コンテクスト数が1の時にイン

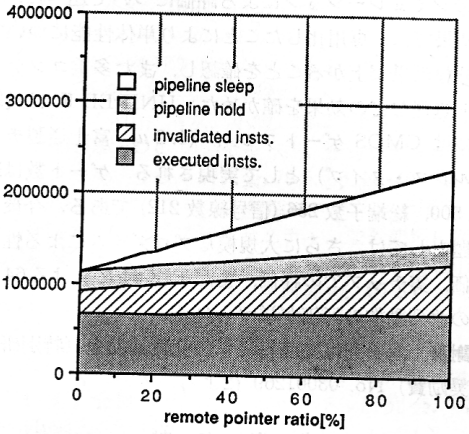


図 9 リモート・アクセスによる実行クロック数の割合の変化（8 queen, 最大コンテクスト数＝1）
Fig. 9 Total clock cycles and items vs. remote access (8-queen, the maximum number of contexts=1).

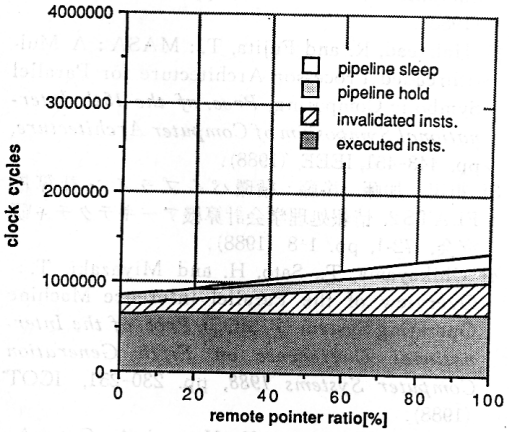


図 10 リモート・アクセスによる実行クロック数の割合の変化（8 queen, 最大コンテクスト数＝2）
Fig. 10 Total clock cycles and items vs. remote access (8-queen, the maximum number of contexts=2).

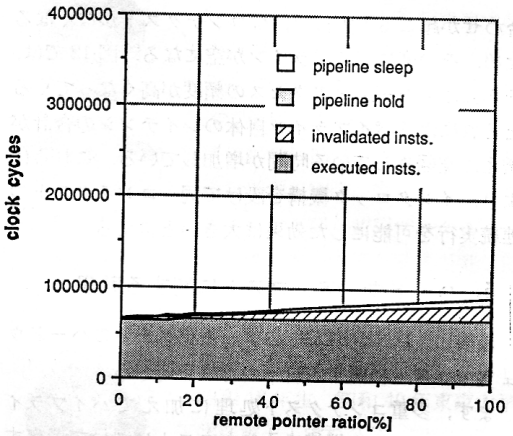


図 11 リモート・アクセスによる実行クロック数の割合の変化（8 queen, 最大コンテクスト数＝4）
Fig. 11 Total clock cycles and items vs. remote access (8-queen, the maximum number of contexts=4).

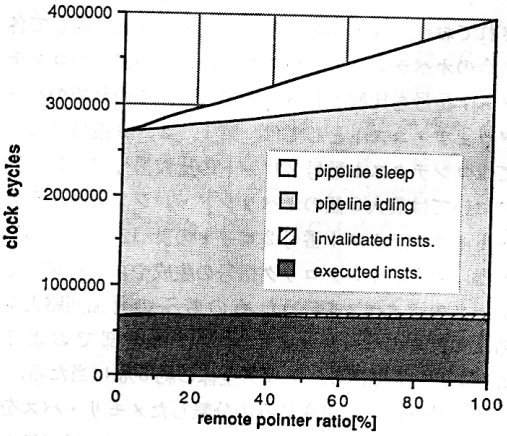


図 12 リモート・アクセスによる実行クロック数の割合の変化（8 queen, 最大コンテクスト数＝1, パイプライン・インタロックなし）
Fig. 12 Total clock cycles and items vs. remote access (8-queen, the maximum number of contexts=1, without the pipeline interlocking mechanism).

タロック機構がないと，3.1 節で述べたように命令間の依存関係を正しく保つために連続した4クロック以内に同一コンテクストの命令が存在できないので，命令の発行は4クロックに1回になる．図 12 では，そのために命令が発行できずにパイプラインが空回りしているクロック数を薄い網掛け部分で示している．これは図9のパイプライン・インタロックにより無効になったクロック数に比較すると，非常に大きいことがわかる．また最大コンテクスト数が1の時は，リモート・メモリ・アクセス・コマンドのリプライ待ち合わせが起こると実行可能なコンテクストがなくなるため，その度にパイプラインが空になる．図12では，そのためリモート・アクセスの頻度が高くなってくるとそれによるパイプライン自体のレイテンシの合計が増え，空回りしている時間が増加している．これらにより，インタロック機構を設けて同一コンテクストの連続実行を可能にした効果は大きいと言える．

5. ハードウェア・コストに関する考察

最後に，UNIRED Ⅱ のアーキテクチャのハードウェア・コストに関する若干の考察を述べる．

まず，多重コンテクスト処理に加えてパイプライン・インタロック機構を入れたコストについて考察する．インタロックの機構そのものは，普通のシングル・コンテクストのプロセッサとほぼ同様である．違いとしては3.2節で述べたようにパイプラインの各ステージに2ビットのコンテクスト番号レジスタが追加されており，ステージ間の依存関係の検出に際して各命令のオペランド・レジスタ番号のほかにこのコンテクスト番号も比較されることである．このためのハードウェア・コストとしては，第1，2，3段目においてはコンテクスト番号2ビットの比較器*，4〜7 段目においては最大2個のオペランド・レジスタ番号5ビット＋コンテクスト番号2ビットの計 12 ビットの比較器，およびインタロック信号の生成やステージ間でデータのバイパスを行うための若干の付加回路となる．これらのハードウェアのコストは全部でおよそ2,000ゲートであり，チップ全体の約5% に当たる．

また読み出し／書き込みの分離したメモリ・バスを用いて，これをパイプラインの別々のステージで扱うことは，回路規模は若干大きくなるものの (UNIRED Ⅱ のパイプライン構成では段数が1段増加するが，これは全体の約 8% のゲート数に当たる)，この部分の制御回路が分離独立されることでその状態数が減って回路が単純化される効果がある (UNIRED Ⅱ では読み出し／書き込みの制御回路の状態数はそれぞれ6および8であるが，これを一つにすると 30 状態程度になると思われる)．さらに Test and Set など2回のメモリ・アクセスを行うような処理もパイプラインを乱さずに実現することができる．

* オペランド・レジスタは3段目で初めてアクセスされる (図3参照)．

6. おわりに

並列推論マシン PIE64 の推論プロセッサ UNIRED Ⅱ のシミュレーションによる評価について述べた．評価結果では，専用化したことにより単体性能について十分な効果が上がることを確認し，また多重コンテクスト処理などの効果を確かめた．UNIRED Ⅱ は最終的には CMOS ゲートアレイ (1.2μ，富士通製チャネルレス・タイプ) として実現される．ゲート数は約 42,500，総端子数 256 (信号線数 212) である．今後の課題としては，さらに大規模なプログラムによる性能評価，実チップでの評価，また PIE64 における64台での並列動作の評価が挙げられる．

謝辞 本研究は文部省科学研究費補助金（特別研究員奨励費）No. 03001269 による．

参考文献

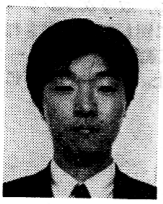
1) Jordan, H. F.: Performance Measurments on HEP—A Pipelined MIMD Computer, *Proc. of the 10th Annual International Symposium on Computer Architecture*, pp. 207-212, ACM (1983).
2) Halstead, R. and Fujita, T.: MASA: A Multithreaded Processor Architecture for Parallel Symbolic Computing, *Proc. of the 15th International Symposium of Computer Architecture*, pp. 443-451, IEEE (1988).
3) 市川，加藤，後藤：循環パイプライン計算機 FLATS2, 情報処理学会計算機アーキテクチャ研究会，72-1, pp. 1-8 (1988).
4) Chikayama, T., Sato, H. and Miyazaki, T.: Overview of the Parallel Inference Machine Operating System (PIMOS), *Proc. of the International Conference on Fifth Generation Computer Systems 1988*, pp. 230-251, ICOT (1988).
5) Shinogi, T., Kumon, K., Hattori, A., Goto, A., Kimura, Y. and Chikayama, T.: MACRO-CALL Instruction for the Efficient KL1 Implementation on PIM, *Proc. of the International Conference on Fifth Generation Computer Systems*

1988, pp. 953-961, ICOT (1988).
6) 中島，武田，中島：PIM/m 要素プロセッサのアーキテクチャ，Technical Report, TR-564, ICOT (1990).
7) Nilsson, M. and Tanaka, H.: Massively Parallel Implementation of Flat GHC on the Connection Machine, *Proc. of Fifth Generation Computer Systems 1988*, pp. 1031-1040, ICOT (1988).
8) Koike, H. and Tanaka, H.: Multi-Context Processing and Data Balancing Mechanism of the Parallel Inference Machine PIE 64, *Proc. of Fifth Generation Computer Systems 1988*, pp. 970-977, ICOT (1988).
9) 清水，小池，島田，田中：並列推論マシン PIE 64 のネットワーク・インターフェース・プロセッサ，並列処理シンポジュウム JSPP '89, pp. 99-106, 情報処理学会 (1989).
10) 島田，下山，清水，小池，田中：推論プロセッサ UNIRED Ⅱ の命令セット，情報処理学会計算機アーキテクチャ研究会, 79-5, pp. 33-40 (1989).
11) 日高，小池，田中：並列推論エンジン PIE 64 の推論ユニットのアーキテクチャ，並列処理に関する「琉球」サマーワークショップ，電子情報通信学会コンピュータシステム研究会，CPSY 90-44, pp. 37-42 (1990).
12) 清水，小池，田中：PIE 64 のネットワーク・インターフェース・プロセッサ LSI の詳細，情報処理学会計算機アーキテクチャ研究会，87-5 (1991).
13) Takahashi, E., Shimizu, T., Koike, H. and Tanaka, H.: A Study of a High Bandwidth and Low Latency Interconnection Network in PIE 64, *Proc. of Pacific Rim Conference on Communications, Computers and Signal Processing*, pp. 5-8, IEEE (1991).

(平成3年 8 月 2 日受付)
(平成3年12月 9 日採録)

島田健太郎（正会員）

昭和 39 年生．昭和 63 年東京大学工学部電気工学科卒業．平成 2 年同大学大学院工学系研究科情報工学専攻修士課程修了．現在同博士課程在学中．平成 3 年度より日本学術振興会特別研究員．計算機アーキテクチャ，並列処理等に興味を持つ．



小池　汎平（正会員）

昭和 36 年生．昭和 59 年東京大学工学部電子工学科卒業．平成元年同大学院工学系研究科情報工学専攻博士課程満期退学．同年東京大学工学部電気工学科助手．工学博士．平成 3 年東京大学工学部電気工学科講師，現在に至る．並列計算機アーキテクチャ，および，並列プログラミング言語に関する研究に従事．本会学術奨励賞受賞．日本ソフトウェア科学会，ACM 各会員．

田中　英彦（正会員）

昭和 18 年生．昭和 40 年東京大学工学部電子工学科卒業．昭和 45 年同大学院博士課程修了．工学博士．同年東京大学工学部講師，昭和 46 年助教授，昭和 62 年教授．昭和 53 年～54 年ニューヨーク市立大学客員教授，現在に至る．計算機アーキテクチャ，並列推論マシン，知識ベース，オブジェクト指向プログラミング，分散処理，CAD，自然言語処理，等の研究を行っている．'計算機アーキテクチャ'，'VLSI コンピュータ Ⅰ，Ⅱ'，'ソフトウェア指向アーキテクチャ'（いずれも共著），'情報通信システム' 著．電子情報通信学会，人工知能学会，日本ソフトウェア科学会，IEEE，ACM 各会員．