推論プロセッサ UNIREDII: シミュレーションによる評価

島田 健太郎,　小池 汎平,　田中 英彦
{shimada, koike, tanaka}@mtl.t.u-tokyo.ac.jp
東京大学 工学部 電気工学科

概要

推論プロセッサ UNIREDII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。ここでは、UNIREDII単体の評価のために作成したシミュレータにより行なった主な評価結果について述べる。評価結果では、多重コンテクスト処理などの有効性を確認し、充分な単体性能が達成されることを確かめた。

1　初めに

現在、我々は並列推論マシン PIE64[1] の開発を行なっている。PIE64 は、対象言語として Committed Choice 型言語 FLENG、及びそのユーザ用上位言語である並列オブジェクト指向言語 FLENG++ を実行する。PIE64 は 64 台の推論ユニット (IU) を 2 系統の自動負荷分散機能を持つ回線交換ネットワークで結合した構成を採る [2]。各 IU にはネットワークと接続し FLENG 向きの高度な通信 / 同期機能を提供するネットワーク・インターフェース・プロセッサ (NIP)、IU 内での FLENG の実行処理を行なう推論プロセッサ UNIREDII、プロセス管理等を行なう管理プロセッサ (MP) として汎用プロセッサ SPARC がある [5]。我々はこの推論プロセッサ UNIREDIIについてこれまでアーキテクチャ、命令セットの設計を行ない [6] 、更に CMOS ゲートアレイとして実現するために回路の詳細設計を行なってきた。今回この回路の詳細な仕様をもとに、性能評価及びゲートアレイ化時の機能テスト・データ生成を目的として、レジスタ・トランスファ・レベルのシミュレータを作成した。ここでは、UNIREDIIのハードウェア構成の概要、及びシミュレータによる多重コンテクスト処理などのアーキテクチャの評価結果について述べる。

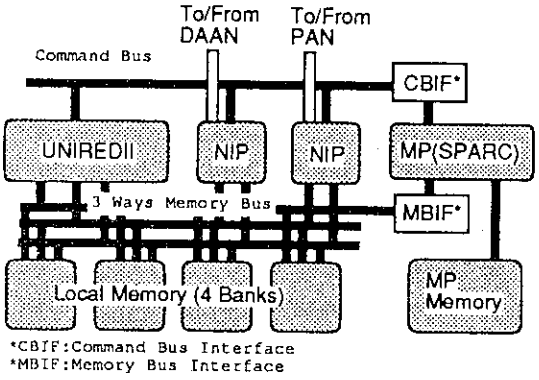

図 1: PIE64 の IU の構成

2　推論プロセッサ UNIREDII

2.1　並列推論マシン PIE64

初めに並列推論マシン PIE64 と、PIE64 における UNIREDIIの役割について簡単に述べる。

PIE64 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 [2]。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 及び Process Allocation Network: PAN) ある。

各 IU には UNIREDIIの他に、ネットワークと接続して FLENG 向きの高度な通信 / 同期機構を提供するネットワーク・インターフェース・プロセッサ (NIP) [3][4]、プロセス管理、入出力等を行なう管理プロセッサ (MP) がある [5]。図 1 に

The Inference Processor UNIREDII: Evaluation by Simulation
by Kentaro SHIMADA, Hanpei KOIKE, and Hidehiko TANAKA
Department of Electorical Engineering, Faculty of Engineering, University of Tokyo

PIE64のIUの構成を示す。IU内に於いて、これらの3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他、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ない、協調して動作する。例えば、他IUへのリモートなメモリ参照では、NIPへ向けてリモート・アクセス・コマンドが送出され、NIPによってネットワークを通して通信が行なわれて結果がリプライとして返される。また、新しいプロセスが生成されると、これは管理プロセッサ(MP)に受け渡され、MPによって最適な負荷分散/スケジューリングが決定される。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。

このように、PIE64においてUNIREDIIはNIP、MPと協調動作をしながら、FLENGの実行を行なう。即ち、FLENGのプログラムはUNIREDIIの命令列にコンパイルされて実行される。

2.2 UNIREDIIの特徴

前節で述べたように、推論プロセッサUNIREDII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。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なう。
- 多重コンテクスト処理[7]によって動的なパイプライン充足率の向上を行なう。
- コプロセッサ・コマンドバスを持ち、NIP、MPとの間でのコマンド/リプライの通信プロトコルをサポートしている。
- FLENGの処理を効率化する強力な命令セット[6]を持つ。

UNIREDII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。またデータのワード長も同じく32ビットであり、GCなどのマーク部2ビットの他、タグ部2ビット、データ部28ビット(ポインタ型)、

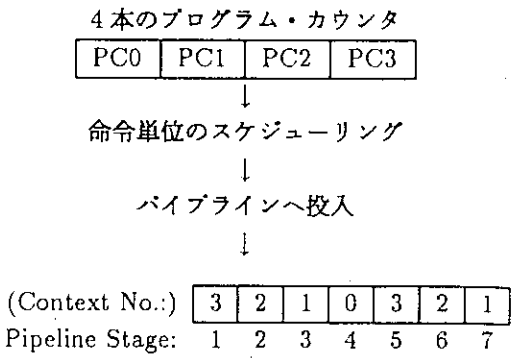

図2: 多重コンテクスト処理の流れ

或はタグ部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

UNIREDII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クスト処理[7]を行なっていることがある。これは、図2に示すように、プロセッサ内に予め複数のコンテクスト(プロセス)を投入しておき、各クロック毎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なって、パイプラインに命令を投入するものである。これにより、パイプライン化された命令の実行に於いてパイプライン・インターロックの低減を行な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。特に後者のリモート・データ参照のレイテンシに対する効果は、並列マシンの要素プロセッサとして適した特性である。

UNIREDIIの多重コンテクスト処理に於ける大きな特徴は、実行可能なコンテクスト数が少ない時には、同じの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。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ー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、

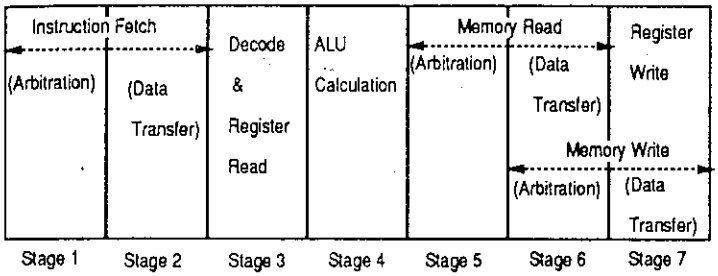

図 3: UNIREDIIのパイプライン構成

極端に大きな性能低下はない。

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 UNIREDIIのハードウェア構成

3.1 UNIREDIIのパイプライン構成

UNIREDII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、PIE64 の各 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 Read-Modify-Write 型の処理を連続して行なうために、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なわれることによる。メモリ・アクセスのスループットは 1 クロック 1 ワードとなる。図 3において、1 段目及び 2 段目で命令フェッチが行なわれ、3 段目で命令のデコード及びレジスタ読み出し、4 段目で実行が行なわれる。更に、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 段目では更にレジスタへの書き込みが行なわれている。UNIREDIIではこのように内部パイプラインにおいてもメモリ・アクセスを重視した構成になっており、汎用手続き型言語に比べメモリ・アクセスが多いと予想される FLENG のような記号処理言語の処理を効率化することを目指している。

パイプライン構成で重要なことは、パイプライン・インター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。UNIREDII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ーロックが生じる。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 段目で行なわれるために、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い。

コンテクスト間でも起こるパイプライン・インターロックとしては、変数に対するバインド命令のような、ロックを掛けてメモリを読み書きする命令に関するものがある。これは、デッドロックを避けるために UNIREDII内部では一時に一つのコンテクストしかメモリ・ロックを掛けられないことによるものである。即ち連続したクロックでメモリにロックを掛ける命令が実行されようとすると、後続の命令は別のコンテクストであっても先の命令の処理が終了するまで待たされる。

3.2 UNIREDIIの内部構成

推論プロセッサ UNIREDII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なうユニット。多重コンテクスト処理を行なうための

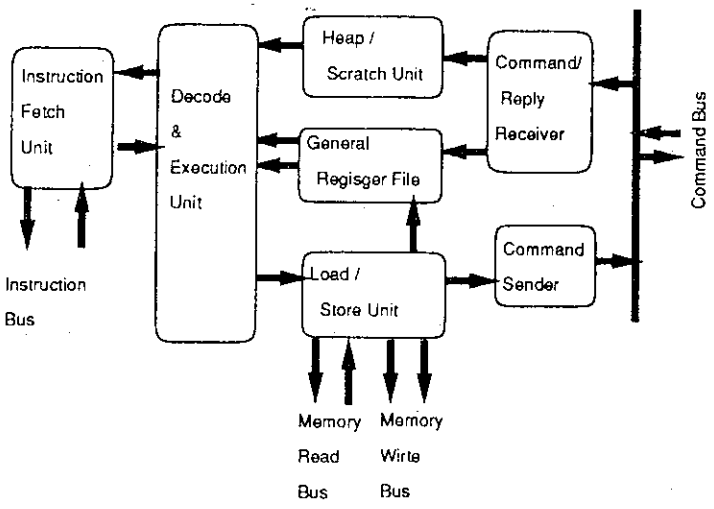

図 4: UNIREDIIの内部構成

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、及び 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。デレファレンス命令等のために未束縛変数のタグ (UNDEF) を検出したり、バインド命令等による不可分なメモリ読み書きを行う機能を持つ。
- 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。コマンド長 2 ワード、リプライ指定 1 ワードまでの汎用的なコマンドを発行する機能を持つ。
- 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、及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。受信したリプライを待って停止しているコンテクストがある時は、そのコンテクストの実行再開を命令フェッチ・ユニットへ指

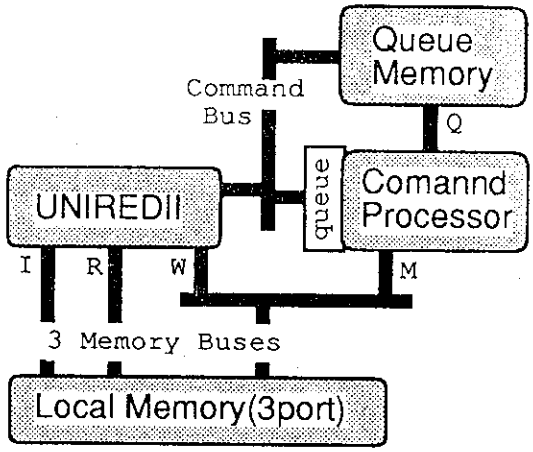

図 5: シミュレーション・モデル

示することも行う。

UNIREDIIのコプロセッサ・コマンド・バスによるコマンド・プロトコルでは、他のプロセッサに送ったコマンドの結果をリプライとして受け取ることが出来るようになっている。コプロセッサ・コマンド命令では、このリプライを任意の汎用レジスタで受け取るように指定できる。

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

今回行なったシミュレーションはレジスタ・ト

表 1: コマンド・エミュレーションに要するクロック数(最低値)

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	変数のバインド	NIP	有	17
read2	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

表 2: シミュレーションに用いたサンプル・プログラム

プログラム	append 100	nreverse 30	qsort 50	primes 100	8 queen
総クロック数	1435	5427	7954	41068	656011
リダクション回数	101	496	381	726	38878
サスペンド回数	0	29	118	103	558
総実行命令数	816	4858	7654	39352	647933
1 リダクション当たりの命令数	8.08	9.79	20.09	54.20	16.67
CPI	1.759	1.117	1.039	1.044	1.012
データ・メモリ・アクセス回数 (read)	306	1843	2252	1764	165833
(() 内は総クロック数に対する割合)	(21.3%)	(34.0%)	(28.3%)	(4.3%)	(25.3%)
(write)	301	1663	1975	1547	141134
	(20.1%)	(30.6%)	(24.8%)	(3.8%)	(21.5%)

ランスファ・レベルのものであり、UNIREDII単体での評価を行なうことを目的としている。U-NIREDIIでは多重コンテクスト処理を行なっているため、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に今回行なったシミュレーションのモデルを掲げる。UNIREDII単体の評価が目的であり、3節で述べたUNIREDII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なっている。即ち、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UNIREDII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、及びゴール(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UNIREDIIと共有するが、これは3本のバスの内Wバス(UNIREDII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[4]によるものとほぼ同等である。また、PIE64の実機ではMPと4台のNIP[1]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、UNIREDII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[4][5]を参照されたい。

4.2 単体での性能

初めに、UNIREDII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なった。用いたサンプル・プログラムはappend 100(長さ100のリストのappend)、naive reverse 30(長さ30のリ

[1]PIE64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・インターフェースには同様のコマンド・キューが設けられている。

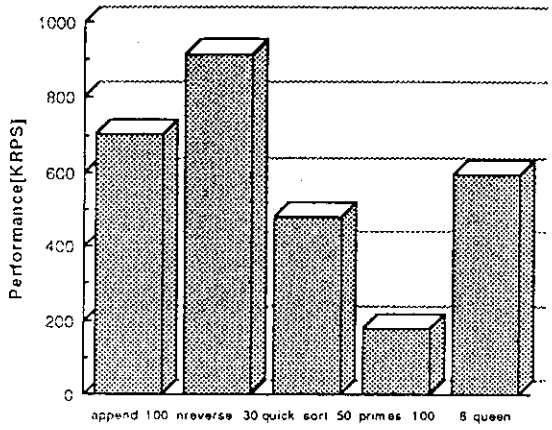

図 6: サンプル・プログラムによる性能

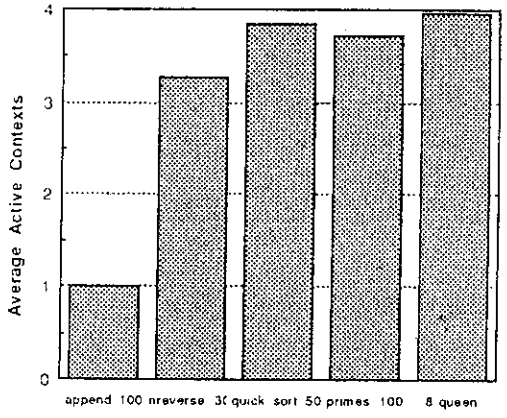

図 7: 平均コンテクスト数

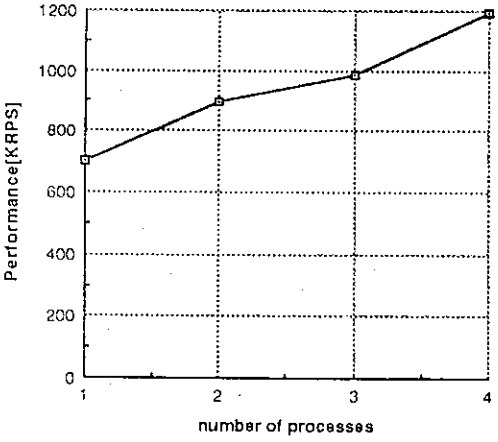

図 8: append 100 のプロセス数と性能

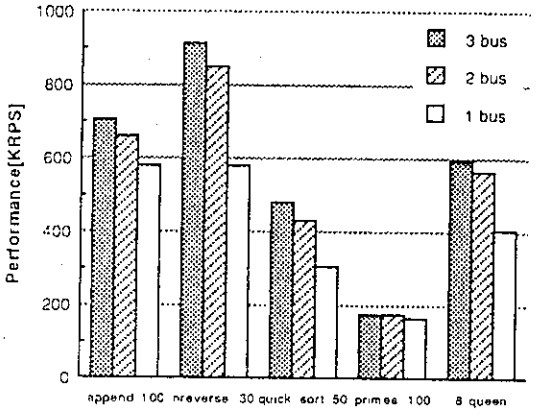

図 9: メモリ・バスの数と性能

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 クィーン問題) である (表 2)。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 (すべて Tail Recursion) であるため平均のコンテクスト数が 1 を越えず、パイプライン・インターロックにより性能が低下している。図 7に各プログラムでの平均コンテクスト数を示す。更にパイプライン・インターロックの影響を確認するために append 100 を 1 個から 4 個まで同時に動かしてみた時の性能の変化の様子を図 8に掲げる。図のようにコンテクストを埋めるプロセス数が増えるに従ってインターロックが減り、性能が向上している。

他に図 6で primes 100 の性能が低下しているのは、主として UNIREDII 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なっているためである。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、PIE64 では各 IU に M-P 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。

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で疑似的に変

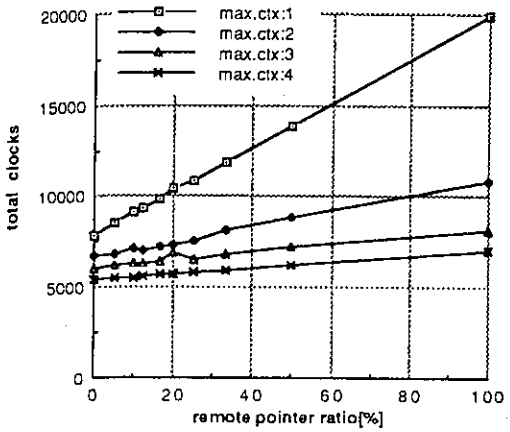

図 10: リモート・アクセスによる総クロック数の変化 (naive reverse 30)

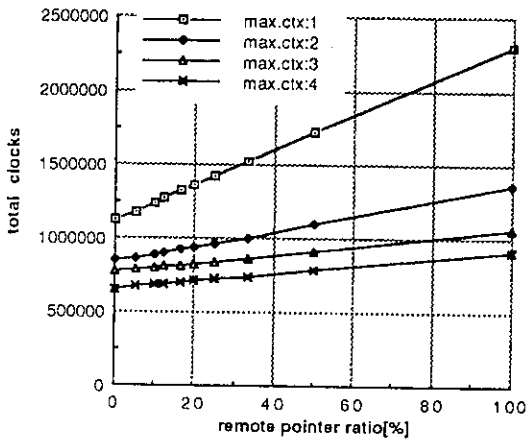

図 11: リモート・アクセスによる総クロック数の変化 (8 queen)

モリ・バスを 1 本にした場合 (1bus)、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 (2bus)、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 (3bus) の、各サンプル・プログラムでの性能である。図のように、データ・メモリ・アクセスの少ない primes 100 以外ではバス 1 本の時には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充分有効であることがわかる。また、読み出し / 書き込みのメモリ・バスを分離すると primes 100 以外では 5 ～ 10% 性能が向上しており、ある程度効果があると言える。

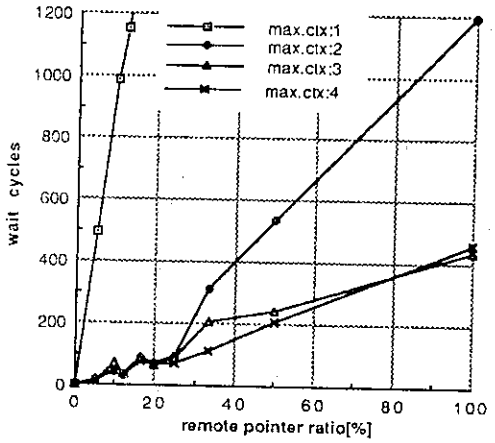

図 12: リモート・アクセスによるウエイト・サイクルの変化 (naive reverse 30)

4.4 リモート・アクセス・レイテンシに対する効果

UNIREDⅡでは、多重コンテクスト処理によって小さなコストでコンテクスト・スイッチングが可能となっており、リモート・アクセスのレイテンシに対してある程度の効果を期待できる。今回行なったシミュレーションは図5に示したようにUNIREDⅡ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 UNIREDⅡによりリダクションを開始される時点、及び一旦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

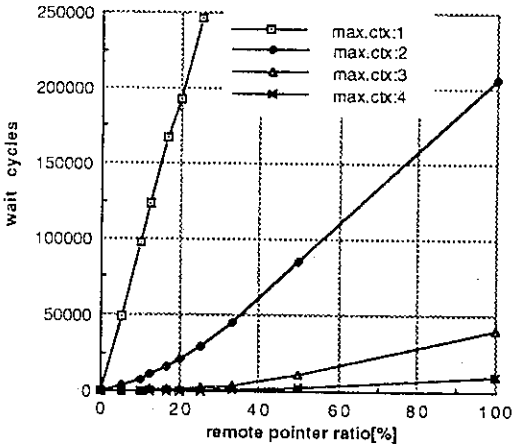

図 13: リモート・アクセスによるウエイト・サイクルの変化 (8 queen)

点[2]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[3]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、UNIREDII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を測定した。結果を図10及び図11に示す。用いたサンプル・プログラムは図10がnaive reverse 30、図11が8 queenである。

8 queenでは図11において、最大コンテクスト数が1の時、即ち多重コンテクスト処理が行なわれない時は、リモート・アクセスの増加に従って総クロック数が大きく増大している。これに対し、最大コンテクスト数が2以上の時は多重コンテクスト処理の効果により、ある程度リモート・アクセスが発生しても総クロック数の大きな増加はない。また、最大コンテクスト数が2から4に増えるに従って、リモート・アクセスによるクロック数の増大を抑える効果が大きくなっている。図13にコマンド発行待ち及びリプライ待ちでUNIREDIIのパイプラインが停止しているサイクル数の変化を示す。図のように最大コンテクスト数が2～4の間で、リモート・アクセスによるウエイト・サイクル数の増大が抑えられていることがわかる。図10及び図12も同様な傾向にあるが、naive reverse 30では平均のコンテクスト数が3強であるので(図7)、最大のコンテクスト数が3の時と4の時の違いが図12にほとんど表れていない。しかしある程度の大きさの応用プログラムでコンテクスト数が充分にある時は、図13に見るように最大コンテクスト数が大きいほど効果が期待でき、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、有効に働いていると言うことができる。

5 終わりに

並列推論マシンPIE64の推論プロセッサUNIREDIIのシミュレーションによる評価について述べた。UNIREDII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42300、総端子数256(信号線数212)である。今後の課題としては、更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。

なお、本研究は文部省特別推進研究No.62065002の一環である。

[2] このようにしたのは、シミュレーションはあくまでUNIREDII単体で行なっているので、アクティベイトの原因となるバインド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。

参考文献

[1] Koike,H.and Tanaka,H.: "*Multi-Context Processing and Data Balancing Mechanism of the Parallel Inference Machine PIE64*" Proc. of Fifth Generation Computer Systems, ICOT, November 1988.

[2] Takahashi,E., Shimizu,T., Koike,H., and Tanaka,H.: "*A Study of a High Bandwidth and Low latency Interconnection Network in PIE64*" Proc. of Pacific Rim Conference on Communications, Computers and Signal Processing, IEEE, May 1991

[3] 清水, 小池, 島田, 田中: "並列推論マシンPIE64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, 情報処理学会, 1989年2月

[4] 清水, 小池, 田中: "PIE64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年3月

[5] 日高, 小池, 田中: "並列推論エンジンPIE64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 CPSY90-44, 1990年7月

[6] 島田, 下山, 清水, 小池, 田中: "推論プロセッサUNIREDII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, 1989年11月

[7] 島田, 小池, 田中: "推論プロセッサUNIREDIIの多重コンテクスト処理" 情報処理学会 第40回全国大会 1L-9, 1990年3月