社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.
CPSY93-28(1993-08)

並列計算機 PIE64 の通信機能の評価

佐藤充, 小池汎平, 田中英彦
{msato, koike, tanaka}@mtl.t.u-tokyo.ac.jp
東京大学工学部

概要

現在、我々の研究室で開発を行なっている並列推論エンジン PIE64 は、実装が進み並列動作が可能な段階に入っている。本稿では、PIE64 の実機で通信性能の測定を行ない、メッセージ生起確率と輻輳との関係、NIP による通信レイテンシの低減効果について調査した結果を報告する。

1　はじめに

現在、我々の研究室では、並列推論マシン PIE64[1] の開発を行なっている。PIE64 は Committed Choice 型言語 Fleng[2] を対象言語としており、64 台の推論ユニット (Inference Unit: IU) が相互結合網によって接続された構成をとっている。

PIE64 の特徴として、その強力な通信機構が挙げられる。PIE64 は相互結合網として、自動負荷分散機能を備えた回線交換方式の多段網を採用し、通信をサポートする構成要素としてネットワーク・インタフェース・プロセッサを搭載している。

一般に並列処理においては、通信によるオーバーヘッドが問題となる。特に Fleng のような細粒度の言語においては、頻繁に通信や同期が発生するため、各通信のオーバーヘッドが、全体の処理時間に対して顕著に効いてくる。

ネットワーク・インタフェース・プロセッサは、この通信の問題に対処するため各推論ユニットに設けられた通信・同期を扱うプロセッサである。ネットワーク・インタフェース・プロセッサは特徴として

- Fleng のゴール間同期処理
- ガベージ・コレクション支援処理
- 相互結合網の自動負荷分散機構を利用したデータ転送処理
- 負荷データ、メッセージ・データを受けとるヒープの管理機構

などを持っており、頻繁に起こる通信や同期に対してコンパクトに処理を行なえるようになっている。特にヒープの管理と推論ユニット間の処理を組み合わせて、Fleng で頻繁に起こるメッセージ送信や負荷転送の高速実行を可能にしている。

本稿では、PIE64 の実機での測定結果をもとに、回線交換方式のネットワークで特徴的な輻輳の問題と、ネットワーク・インタフェース・プロセッサの機能によるメッセージ送信におけるレイテンシ低減効果について調査した結果を報告する。

2 PIE64 の構成

2.1 全体構成

当研究室で開発している PIE64 は、記号処理の高速実行を目的とした並列推論エンジンである。PIE64 の全体構成は図1のようになっている。

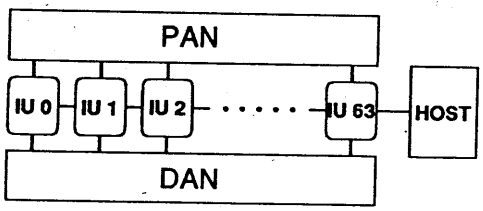

図 1: PIE64 の構成

PIE64 では、推論ユニット (Inference Unit: IU) と呼ばれる基本ユニットが、2 系統の相互結合網によって接続されている。これら 2 系統の相互結合網は同等のものであり、それぞれ PAN(Process Allocation Network)、DAN(Data Allocation/Accessing Network) と呼ばれている。PIE64 上の処理系では、これら2系統のネットワークを使い分けながら IU 間での通信を行なう。

また、PIE64 はホストであるワークステーションと接続されており、外部とのやりとりはこのホストを通じて行なう。

2.2 推論ユニットの構成

推論ユニットは、以下の部分より構成されている (図2)。

- 推論プロセッサ – UNIRED[3]
- ネットワーク・インタフェース・プロセッサ – NIP[4]
- 管理プロセッサ – SPARC
- ローカルメモリ
- SPARC メモリ
- ホスト・インタフェース
- I/O インタフェース

推論ユニットには、推論を行なうメインプロセッサである UNIRED(UNIfire/REDucer)、同期・通信処理を主に担当する NIP(Network Interfece Processor)、並列処理管理を行なう管理プロセッサ (Management Processor: MP) という 3 つのプロセッサが存在し、各プロセッサはコマンド・バスと呼ばれるバスで接続されている。各プロセッサは、コマンド・バスでコマンド/リプライをやりとりすることによって、協調して処理を行なう。

これらのプロセッサは、ローカル・メモリを共有している。UNIRED は 3 本のバスを使い、NIP は PAN、DAN それぞれに対してマスタとスレーブがあり、SPARC からローカル・メモリをアクセスする手段は 2 種類用意されているので、ひとつの IU にバスマスタが 9 つ存在することになる。それらバスマスタとローカル・メモリとの間に十分なスループットを確保するため、PIE64 では 3 本のデータ・バスを用いている。

これら 3 本のデータ・バスには

- Bus 0
  - UNIRED (Instruction Fetch)
  - SPARC (Direct Interface)
- Bus 1
  - UNIRED (Data Read)
  - PAN Master NIP
  - DAN Slave NIP
- Bus 2
  - UNIRED (Data Write)
  - PAN Slave NIP
  - DAN Master NIP
  - SPARC (Staging Buffer Interface)

が接続しており、バスのトラフィックが均等に分散されるように構成されている。

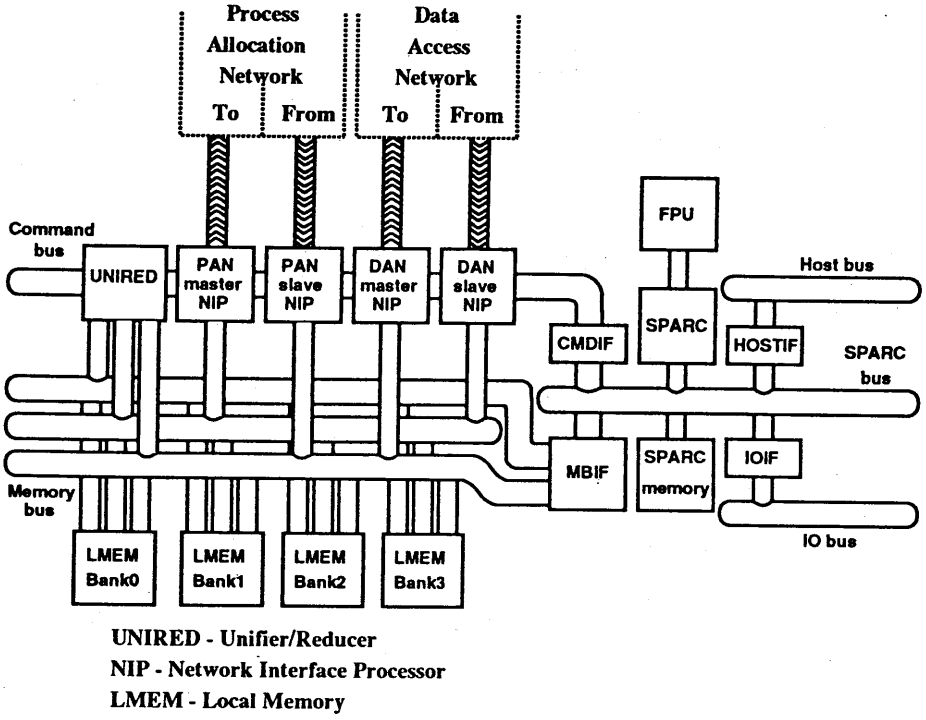

図 2: IU の構成

2.3 相互結合網

2 系統の相互結合網 [5] は、その使用方法で区別されるだけで、独立して動作する全く同じハードウェアである。それぞれの相互結合網は、4×4 のクロスバスイッチを基本要素とした、回線交換方式の 3 段の多段網である。

この結合網の大きな特徴に自動負荷分散機構がある。これは、回線が接続されていない時に各推論ユニットの負荷情報を非制御側から逆方向に相互結合網に流しておいて、相互結合網が自動的に負荷最小の推論ユニットを選択して、回線接続を行なうというものである。

相互結合網の接続は、図3のようになっている。

3 性能測定

PIE64 における通信は、図 4のような手順で行なわれる。ここで重要なことは、受信側では Slave NIP が動作するのみで、UNIRED/SPARC は通信に関与する必要がないということである。特に、メッセージ通信において、送信側が送り先のアドレスを知らない場合には、一般には図 5のような send/recieve を用いるのが普通であるが、PIE64 の場合は受信用バッファの確保も Slave NIP がおこなうので、メインの計算を中断することなく通信処理を実行することができる。

3.1 単純転送

まず、PIE64 の基本的なメッセージ転送速度を測定するため、表 1のような条件のもとで転送速度の測定を行なった。

ここで、WRITEX とは相手先の書き込みアドレスを送信元で指定するメッセージ転送で、受信バッファのアロケートは必要ない。WRITEMN とは、受信用バッファのアロケートを受信側の Slave NIP で行なう転送モードである。MP-Write とは、管理プロセッサである SPARC を用いて、図

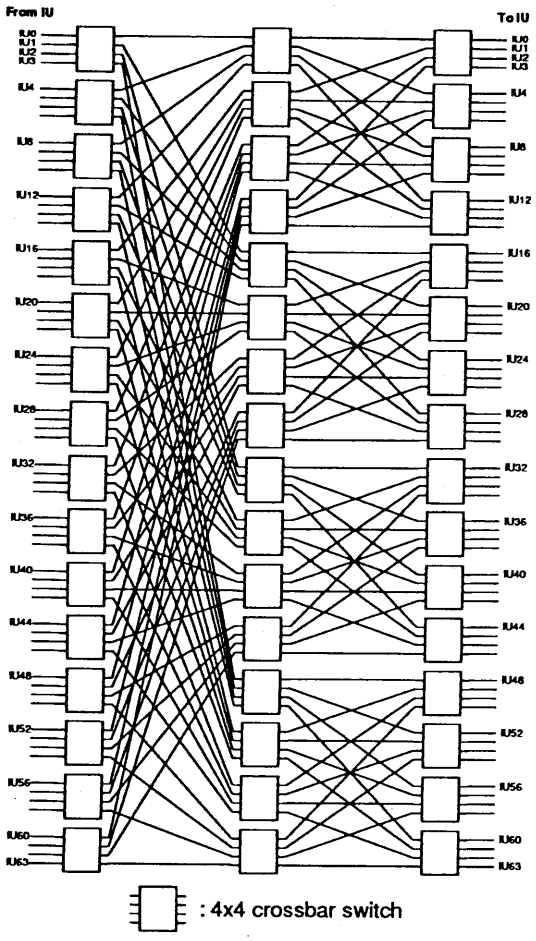

図 3: PIE64 の相互結合網

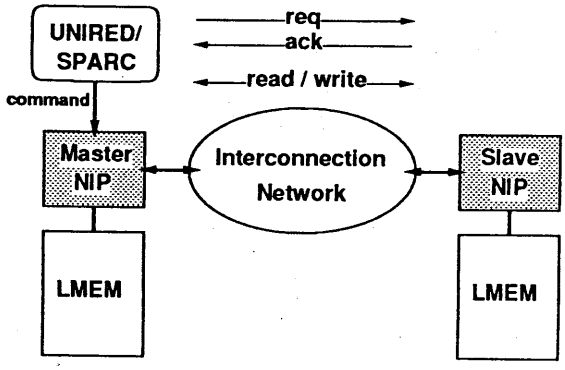

図 4: PIE64 での通信

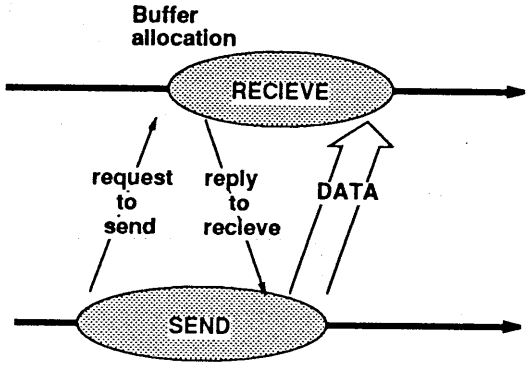

図 5: send/recieve

表 1: 測定条件

# of IU	command	len. of msg.
2	WRITEX	1 ～ 4096
2	WRITEMN	1 ～ 4096
2	MP-Write	1 ～ 4096

5を実現したものである。

PIE64 の IU にはハードウェア・カウンタが備え付けられており、任意の間隔をクロック単位で測定することができる。今回はこのカウンタを用いて測定を行なった。

ここでは、

A: テストプログラムが終了するまでの時間
B: ループの回数は同じで、リモートアクセスを全く行なわなかった時の実行時間

を測定し、

$$T_l = (A - B)/N$$

N : ループ回数

によって得られる値をメッセージ転送にかかった時間 (レイテンシ) とした。

図 6に転送量 [word] に対するレイテンシのグラフを示す。

WRITEMN はWRITEN に比べて、受信側で受信バッファを確保するための時間が余分にかかっている。しかし、この時間は平均 5 クロック程度と非常に小さいものであり、一般のメッセージ

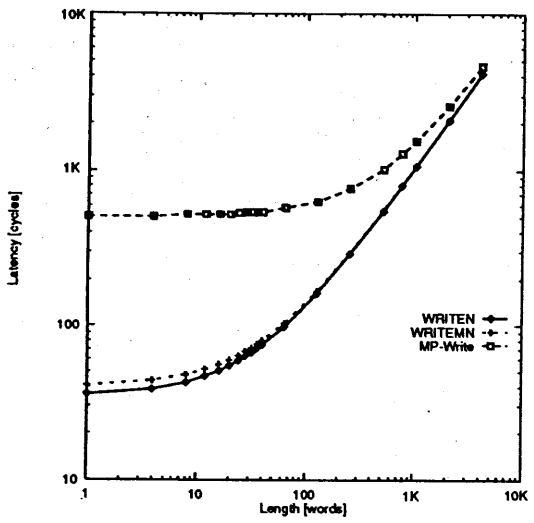

図 6: 単純転送

通信 (送り先のアドレスがわかっていない場合) でも十分高速にメッセージを転送できることがわかる。

一方、MP-Write では、次のような手順でメッセージ転送を行なう。

1. UNIRED は MP に対してメッセージ転送要求を出す
2. MP は相手先の MP に対して、これから送信するメッセージの長さを送り (request:Req)、相手先で受信準備ができるのを待つ
3. MP は相手先から受信準備完了の合図と共に、バッファの先頭アドレスを受けとり (acknowledgement:Ack)、データの転送を行なう

従って、MP-Write では UNIRED ↔ MP と MP↔MP の間で余分な転送を行なっており、その分大きく時間がかかる。その差は、図6より、およそ470クロックであることがわかる。これは WRITEMN でのおよそ 500word 転送分に相当し、特に小さなメッセージの場合にはこの差が顕著に効いてくる。

3.2 混雑した状況での転送

PIE64 では、相互結合網として回線交換型の閉塞網を用いているため、相互結合網が混雑してくると、急激に閉塞率が上昇し、全体のスループットが減少する。

実際のアプリケーションでは、各 IU から様々な通信要求が生じ、相互結合網で通信要求が衝突することが考えられる。そのため、より現実的な評価として相互結合網が混雑した状況での通信性能を測定する必要がある。

相互結合網を混雑させる要因としては、次のようなものが考えられる。

1. IU の台数
2. メッセージ長 (N [word])
3. 通信要求出現頻度 (p)
4. 接続パターン

1. の IU の台数については、すべての IU を用いた[1]。

4. の接続パターンとは、どの IU からどの IU に対して通信するかというものである。これはアプリケーションによって様々に変化することが予想されるが、今回はランダムに接続先を選択した。

3.2.1 ネットワーク待ち時間

メッセージ転送にかかる時間は、次の要素から構成される。

- 回線接続、メッセージ長の送信などの通信のセットアップに必要な時間 (T_s)
- パリティ・チェックなどの接続解放に必要な時間 (T_e)
- メッセージの転送時間 ($T_b \times N$)
- 回線接続待ち時間 (T_w)
- その他(コマンドのやりとり、バスの衝突などによるもの)(T_{etc})

このうち、T_s, T_e, T_bは輻輳によって変化しない値であり、基本転送の結果から、

[1]実際には、現在利用可能な IU 54 台

$$T_s + T_e = 38.9 \text{ [clock]}$$
$$T_b = 1.0 \text{ [clock/word]}$$

という値を得ることができる。

したがって、輻輳を生じた時に問題となるのは回線の接続待ち時間 (T_w) であり、この値の変化がメッセージ転送時間の変化に結び付くということがわかる。

ただし、他のプロセッサ (MP, 他の NIP) が同時に動作している状況では、その他の時間 (T_{etc}) も無視できないものとなってくる。しかし今回の測定は他のプロセッサは動作しない状況で行なったので、T_{etc}の影響はほとんどないものと仮定する。

3.2.2 メッセージ生起確率による変化

図7に、WRITEMN を用いた場合の、メッセージ生起確率と待ち時間との関係を示す。これは、プログラム中のメッセージ転送命令の割合を変化させながら、実際には何クロックに1回メッセージ転送命令が処理されているのかと、回線接続待ち時間を測定し、プロットしたものである。

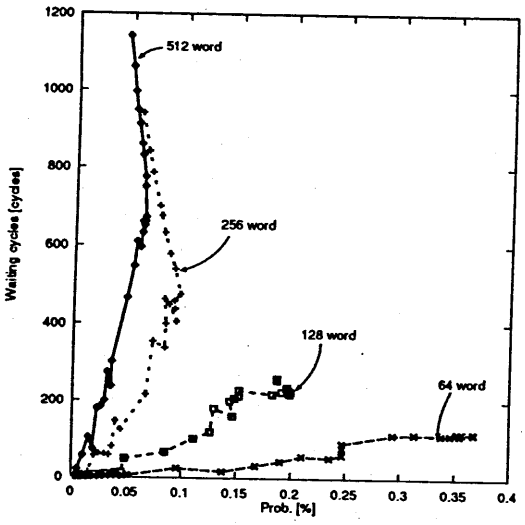

図 7: 待ち時間の変化 (WRITEMN)

ここでの回線接続待ち時間としては、NIP の性能測定用ハードウェア・カウンタで測定した値を用いた。

この図より

- ある領域に達するまで、メッセージ発生頻度と待ち時間はほぼ正比例の関係にある
- その比例係数は、転送長に依存する
- ある領域 (限界領域) に達すると、待ち時間とメッセージ発生頻度は $y = a/x$ の線 上を動く

ということがわかる。

メッセージ発生頻度 (p) と待ち時間 (T_w) が比例している領域にあるとき、つまりプログラム中のメッセージ転送命令の割合が比較的少ない状態にある場合は、メッセージの転送は正常に行なわれている。つまり、一度メッセージの転送命令を発行すると、次の転送命令が発行された時には以前のメッセージはすでに転送されてしまっているという状態である。

この領域では、発生頻度 p と待ち時間 T_w との比例係数 ($\alpha = T_w/p$) は、メッセージ長に依存する。しかし、その関係は比例関係ではない。このことは、同じ量のメッセージを転送するのにも、メッセージ長が長い方がネットワーク待ち時間の観点から見ると有利であるということを表している。これは、短いメッセージでは、転送の前処理 (経路設定や NIP のセットアップ時間等) の影響が大きいためであると考えられる。

さらに、プログラム中のメッセージ転送命令の割合を増加させていくと、ある時点より、発生頻度と待ち時間は比例せず、逆に反比例の関係になる。さらに、メッセージ転送命令の頻度を上げると、その処理時間は遅くなってゆく。ここでは、以前発行されたメッセージ転送が終了しないうちに、次の転送命令が来てしまうという状態になっている。したがって、メッセージを転送する間隔を決定しているのは発生頻度ではなく、待ち時間である。この状態では、発生頻度を上げるほど時間あたりの転送量 (スループット) は少なくなる。

実際の処理系では、このような状態に至らないよう、負荷やデータの配置に注意しなくてはならない。

また、上で述べたように、メッセージ長とαは

比例していないので、限界領域に入らない状態でのスループットの最大値は、メッセージ長が長いほど大きい。

3.2.3 MP-Write

MP-Write では、WRITEMN とは違い、NIP のハードウェア・カウンタで計測した値を待ち時間とするわけにはいかない。そのため、ここでの待ち時間 $T_{w'}$ は次のようにして算出した。

$$T_{w'} = (T_{to} - T_{inst} - T_{trans})/N_c$$

T_{to} : 全体の実行時間
T_{inst} : メッセージ転送を行なわない時の実行時間
T_{trans}: データの転送時間
N_c : 転送回数
このようにして得られた $T_{w'}$ を図8に示す。

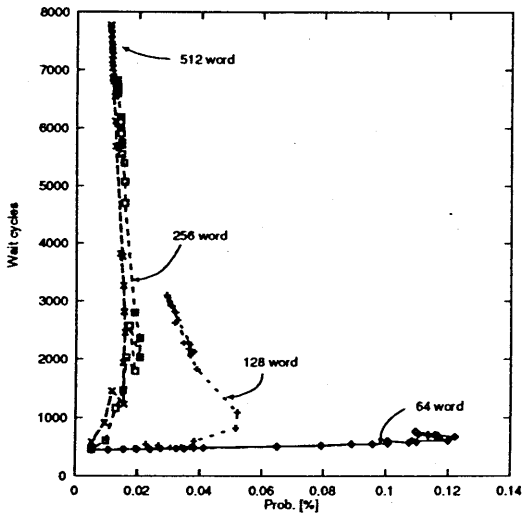

図 8: 待ち時間の変化 (MP-Write)

これと図7とを比較すると、

- MP-Write では基本的にかかる待ち時間が大きい
- そのため、限界となる領域に到達するのが早い (WRITEMN に比べて、生起確立で 1/4 程度)

などということがわかる。

MP-Write では、次のように待ち時間 $T_{w'}$ をさらに細かく分類することができる。

- MP で処理を行なう時間 (T_{mp})
- 相手先に Req を送って、Ack が戻って来るまでの時間 (T_{pa})
- その他 (コマンドのやりとり、バスの衝突など)(T_{etc})

これらの要素の待ち時間に対する影響が、メッセージの生起確率によってどのように変化するかを表したのが図9である (図9は限界領域に到達しない範囲のみ示してある。また全体として $T_{w'}$ ではなく、レイテンシ T_l をとっている)。

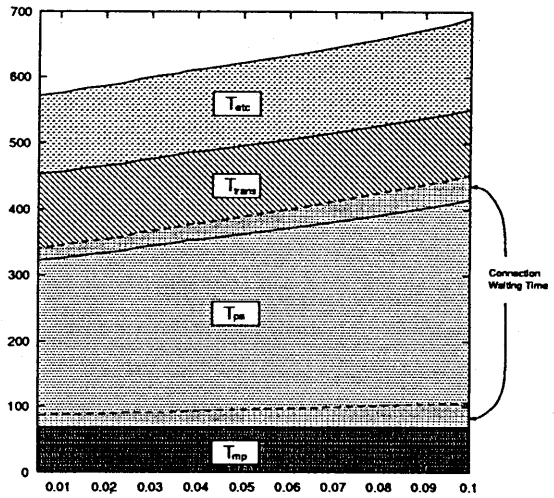

図 9: T_l の内訳 (MP-Write)

図9より、以下のことがわかる。

- メッセージ生起確率が大きくなるほど、回線接続の待ち時間が大きくなる。MP-Write の場合は Req と Ack、データ転送と、3 回相互結合網に接続しなくてはならないので、その影響も WRITEMN に比べて大きくなる。
- レイテンシに対して、特に影響を与えているのは、Req/Ack をやりとりする時間 (T_{pa}) である。この理由としては
  - Req/Ack の 2 回接続を行なうので、待ち時間の影響が大きい

- 相手側のMPの状況によっては、Ackが戻ってくるのが大幅に遅れる可能性がある

ということが考えられる。

Flengは細粒度高並列言語であり、小さな通信が頻繁に行なわれることが予想される。そのため、PIE64でMP-Writeのようなメッセージ転送を行なっていたのでは、すぐに限界領域に達してしまい、転送の効率は非常に悪い。また、回線交換では一度接続してからの双方向の通信は非常に低コストに行なうことができるので、Req/Ackのような細かな通信はまとめて行なった方がよい。これらのことから、NIPに比較的高レベルの通信機能を持たせることによって、通信の待ち時間、ひいては通信のレイテンシを低く押えることができる。

4 まとめと今後の課題

本稿では、並列計算機PIE64でのメッセージ転送速度を測定し、メッセージ送信においてNIPの機能によって得られる効果について確認した。また、相互結合網が混雑した時にメッセージ通信のレイテンシがどのように変化するかを、回線接続待ち時間を基準にして測定した。またその結果として、回線交換方式の特徴を利用したNIPの機能の有用性について確認した。

PIE64のNIPには、ここで利用したリモート・ヒープを扱う機能の他に、Flengをサポートするためのさまざまな通信・同期機構を備えている。今後は、Flengでの通信パターンをもとにした解析とともに、これらFleng支援機能がどのくらい有効に働くかについて、実際のPIE64上でのFleng処理系をもとに測定していく必要があるだろう。

参考文献

[1] 日高康雄, 小池汎平, 高橋栄一, 島田健太郎, 清水剛, 田中英彦, "高並列推論エンジンPI64 研究経過報告 – ハードウェア –", 情報処理学会第46回全国大会, 1993.

[2] M.Nilsson and H.Tanaka, "Massively Parallel Implementation for Flat GHC on the Connection Machine, *Proc. of the Int. Conf. on Fifth Generation Computer Systems*, pp.1031-1040, 1988.

[3] Kentaro Shimada, Hanpei Koike and Hidehiko Tanaka, "UNIRED II: The High Performance Inference Processor for the Parallel Inference Machine PIE64", *Proc. of FGCS 1992*, pp.715-722, 1992.

[4] 清水剛, 小池汎平, 田中英彦, "並列推論マシンPIE64のネットワーク・インタフェース・プロセッサ", 並列処理シンポジウムJSPP'89, pp.99-106, 1989.

[5] 高橋栄一, 小池汎平, 田中英彦, "並列推論マシンPIE64の相互結合網の作成および評価", 並列処理シンポジウムJSPP'90, pp.89-96, 1990.