並列推論マシン PIE64 の推論ユニット間通信

清水 剛，小池 汎平，田中 英彦
東京大学 工学部

概要

我々が開発を進めている並列推論マシン PIE64 のネットワーク・インタフェース・プロセッサ (NIP) は、PIE64 の各推論ユニット (IU) 内部と相互結合網とのインタフェースを行なうプロセッサである。

NIP は、データ転送やプロセス間同期、および、分散メモリ環境におけるガベージ・コレクション支援のコマンドをネットワークを介して実行し、PIE64 での並列論理型言語 FLENG の実行処理における並列処理機能を支援する。

本稿では、NIP の内部ハードウェア構成、及び NIP 各部での処理動作をふまえ、PIE64 の相互結合網を介して行われる 2 個の NIP 間での通信のプロトコル、性能などについて述べる。

INTER-PE COMMUNICATION
OF THE PARALLEL INFEERENCE MACHINE
PIE64

Takeshi Shimizu, Hanpei Koike, Hidehiko Tanaka
Department of Electrical Engineering, Faculty of Engineering,
University of Tokyo
Hongo 7-3-1, Bunkyo-ku, Tokyo 113, Japan

Abstract

The Network Interface Processor (NIP) of PIE64 is a processor used for the interface between each Inference Unit (IU) and the interconnection network of PIE64. NIP executes three types of commands between 2 IUs through the interconnection network. One is to transfer data, another is to exchange process synchronization messages and to manage goal suspension information directly by hardware, and the other is to support global garbage collection under the distributed memory systems of PIE64.

This paper shows the internal hardware organization of NIP, reports how it works, and describes the communication protocol on the interconnection network of PIE64.
1 はじめに


各 IU には、IU 内部と相互結合網とのインタフェースをなすネットワーク・インタフェース・プロセッサ (Network Interface Processor: NIP) が用意される。NIP は PIE64 で行なわれる並列推論処理において、2 つの IU 間でのデータ転送、Fleng の論理変数を含むプロセス間同期、PIE64 の一括ガーベージ・コレクション [7] といった並列処理機能をハードウェア・レベルで直接実行する [8]。

本稿では、NIP の内部ハードウェア構成、及び NIP 各部での処理動作を含め、PIE64 の相互結合網を介して行なわれる 2 個の NIP 間での通信のプロトコル、性能などについて述べる。

2 並列推論マシン PIE64

PIE64 は、64 台の推論ユニットを、2 系統の自動負荷分散機構と多段ネットワークにより組合せた構造を持ち、基本言語として並列論理型言語 Fleng およびその上位言語である Fleng++ を実行する。

PIE64 の 2 つの相互結合網は、2 系統のネットワークは、プロセス分散網 (Process Allocation Network: PAN)、および、データ配置網 (Data Allocation/Accessing Network: DAN) と呼ばれ、どちらも自動負荷分散機能を備えた 64×64 の回線交換式の多段網である。この 2 つのネットワークにはサーバーワークス上の差異はない。

各 IU には Fleng の実行処理を受け持つ推論プロセッサ UNIREDI [9]、IU 間通信・Fleng のプロセス間同期を支援する NIP、IU の管理や出力処理を担当する管理プロセッサとしての SPARC がある。各ネットワークには、ネットワークへの接続を出す、いわば、ネットワークへの出入口にあたるマスタ NIP と、ネットワークからの要求に従って動作する、いわば、ネットワークからの入口にあたるスレーブ NIP が接続されるので、各 IU には計 4 個の NIP が実装されることになる。

これらの計 6 個のプロセッサ (UNIREDI、NIP × 4、管理プロセッサ) が協調動作し、Fleng の実行をすすめるが、この IU 内部のプロセッサ間通信は、高速・高方向性 32 ビット幅の専用コマンド・バスを介して、コマンド / リプライのやり取りによって行なわれる。

また、これらのプロセッサは、3 系統の同期メモリ・バスを介して、4 個のローカル・メモリ・バンクを共有する。

3 ネットワーク・インタフェース・プロセッサ

NIP は以下のような並列処理支援のための機能を IU 内部に提供する。IU 内部のプロセッサは、マスタ NIP に対してコマンドを発行することにより、これらの機能を利用できる。

- データ転送処理
- サスペション・レコード・リストの管理によるプロセス間同期処理
- 一括ガーベージ・コレクション支援

このうち、データ転送と一括ガーベージ・コレクション支援に関しては、マスタ NIP とネットワークを介して接続された相手側のスレーブ NIP とが協調動作して処理を実行する。

また、プロセス間同期処理のうちのリスト処理にあたる部分は、接続先のスレーブ NIP で行なわれる。IU 内部でローカルにサスペション・レコード管理機能を利用する場合には、スレーブ NIP にコマンドを送ることもできる。

データ転送処理においては、以下のような特徴を持つ。

- Fleng のデータタイプを考慮した転送命令形式
- 自動負荷分散機構を利用したデータ転送
- 構造体系領域内の未定義論理変数領域の一意性を保証

第 3 項の機能は、リストやベクトルに UNDEF 型のデータを持つ未定義論理変数の実体が埋め込まれていた場合、データ転送時に
その要素を、もとの未定義変数型の実体を指定変数型のポインタとしてそれ代えて転送するものである。

プロセス間同期処理については、以下のコマンドが用意されている。

* suspend コマンドを起動したコンテクストを、その原因となった未定義変数のパス・レコード・リストに登録するための suspend コマンド

* 未定義変数型に値の束縛を試み、その変数のパス・レコード・リストに登録されていたコンテクストをアクティブタイプするための bind コマンド

* リモートに実体のあるコンテクストがアクティブタイプされたことを、そのリモートの IU に通知するための activate コマンド

* リモートで実体のあるコンテクストをアクティブタイプしたことを、そのリモートの IU に通知するための activate コマンド

また、一括ガス・コレクション支援に関しては、以下の機能が用意されている。

* リモートのマーキングを支援する mark コマンド

* コンパクション・フェーズでのリモート・ポインタの書き戻しをする restore コマンド

4 NIP のハードウェア

NIP の内部ハードウェア構成、および、各部の処理動作について述べる。

4.1 NIP の内部ブロック構成

NIP の内部ブロックは、大きく分けて以下の 4 つのブロックから成る。

* コマンド・バス・インタフェース部

* メモリ・バス・インタフェース部

* データ転送処理部

* プロセス間同期処理部

コマンド・バス・インタフェース部は IU 内の他のプロセッサ（管理プロセッサ SPARC、推論プロセッサ UNIREDI、他の NIP）の間で、コマンド／リプライの送受信を行なう。一部の性

能測定用のカウンタ、および、プロセッサの動作モード設定用のレジスタを持つ。

メモリ・バス・インタフェース部は、IU 内部の 3 本のメモリ・バスと接続され、そのうちの 1 本に対しては IU 内ローカル・メモリに対する読み書きを行い、他の 2 本に関して、ロック・アドレスの監視をしている。また、負荷データやメッセージ受信に使用されるヒープ・メモリ管理用のレジスタを持つ。このヒープ・メモリ管理用のレジスタは 1 組の予備ヒープ・メモリ用のレジスタを含み、2 組が予備としてあり、スレープ NIP で使用される。

データ転送処理部は、ネットワークを介したデータ転送処理、及び、一括ガス・コレクション支援のコマンドを実行する。また、ネットワークの自動負荷分散機能に対応して、ネットワークを通じて流れてくる 8 ビットの負荷分散情報を監視するためのコンパクションボード、ネットワークを介したデータ転送時のデータ列の垂直バーティア生成するパーティアジェネレータを持つ。データ転送時に使用されるアドレス生成用のカウンタと転送長を計量するカウンタはメモリ・バス・インタフェース部にある。

プロセス間同期処理部は、スレープ NIP において、パス・レコードの管理のためのリスト処理を行う。これらタイプはそれぞれ独自にシークエンスを持ち、協調動作をすることとしても、並行処理を可能としている（図 1）。

4.2 データ転送処理ブロック

データ転送処理ブロックを示す。データ転送処理においては、

1. 最大効率時に、1 クロック 1 ワードのデータ転送が可能となること

2. IU 内ルーカル・メモリには、アドドリーション+アドレス出力、データ転送の 2 フェーズからなるバイブレイン的なアクセスがなされることがある

3. ネットワークに約 50 ns の遅延が存在すること

4. 構造データ内で埋め込まれた未定義変数領域の任意性を保証するため、この変数領域
図1: NIP の内部ブロック構成

をポイントに変換して転送ができること

等の要求及び制約を満たすために、データ読みだし側の NIP で 3 段、ネットワークを媒介に順にデータ書き込み側の NIP で 3 段のパイプラインを構成する形になっている。

読みだし側での 3 段は、図 2 中の BUF0 → BUF1 → NOB のパスで、前 2 段は、それぞれ、アドレス出力とデータ・フェッチのステージに相当し、その時点でのリクエスト・アドレスが前 2 段で保持され、前記 4 の変数型ポイントへの変換処理に利用される。このポイントとデータの置き換えは、3 段目のネットワークへの出力段に移行する BUF1 → NOB の段階で行なわれる。

書き込み側での 3 段は、図 2 中の NIB → BUF2 → NOB のパスで、それぞれ、ネットワークからのデータ受信、メモリ・リクエスト、データ書き込みのステージに相当する。このデータの流れの様子を図 3 に示す。

パイプライン的に読み書きされるメモリを、ネットワークを挟んで接続し、両側の制御回路の状態を 1 クロックの遅延 3 ある信号線を用いて伝えあって分散制御しているため、書き込み側でメモリパスのアクセス権を確保できなかった場合にパイプラインの乱れが生じ、読み出し側がオーバーランをしてしまう。

3 前述のとおり、ネットワークの信号遅延自体は半クロック程度であるが、同期回路の設計上、ネットワークを 1 クロックの遅延とみなしている

図2: データ転送処理ブロックのデータパス

図 3 に示した以外のデータ転送処理部のレジスタは全て、パイプラインの乱れを解消するためのバッファとして利用される。このバッファリングは、送信側で 2 ワード、受信側で 3 ワード保持される形になった。この構成にすれば、もっともゲート数が少なく、かつ、他の機能の実装と併せて考慮した場合に自然に実装できるからである。

また、一括ガページ・コレクション支援時のリモート・ポイントのマークと、コンパクショング後のポイント書き込み処理の場合には、一部のレジスタがハッシュング及びキャッシュングのために利用される。

これら他の他に、プロセス間同期処理のうち、ネットワーク側から要求された bind 処理に対応して、未対称導通変数からのサスペジョン・レコード・リストの取り出しと値の乗算のアトミックなオペレーションについては、このブロックで行なわれ、サスペジョン・レコード・リストのリスト処理に必要なパラメータのみがコマンドと同一の形でプロセス間同期処理部に送られる。

4.3 プロセス間同期処理ブロック

プロセス間同期処理ブロックのデータバスを図 4 に示す。
このブロックでは、プロセス間同期処理のうちの、suspend処理におけるサスペシジョン・レコード・リストへのコンテクストの追加登録、および、activates処理におけるサスペシジョン・レコード・リストの回収を主として行なう。

サスペシジョン・レコード・リスト用のメモリは、フリーリストの形でスレーブNIPによって管理されている。

このブロックは、これらのコマンドのための引数レジスタ（VAR, VAL）、サスペシジョン・レコード・リストの処理に関用されるテンポラリ・レジスタ（SUSREC, BTMP, STMP）、フリーリストを保持するレジスタ（FREE）を持つ。また、リスト処理時にボインタに対して必要に応じてタグの書き換えを行う。

PIE64のローカル・メモリ・バスはバイブレイン的にアクセスが行なわれるため、あるメモリ・アクセスの結果を出力する次のメモリ・アクセスを行うようなら、シークエンシャルなメモリ上のデータに対する処理をする場合にはスループットが低下しない。

そこで、スレーブNIPでは、リスト処理時の1つのセルの処理に必要な4回から5回のメモリ・アクセスに対し、アクセス順序の制約を崩さない範囲で動的にアクセス順序を変え、PIE64のメモリ・バス・プロトコルに適合した高速なリスト処理を実現している。

5 通信プロトコル

PIE64において、相互結合網を介してNIPが行なう推論ユニット間通信のプロトコルについて述べる。

PIE64のネットワーク自体はマルチ・キャストをサポートするよう設計・製作がされているが、NIPはマルチ・キャストの機能を使用せず、通信はすべて1対1通信で行なわれる。通信は以下の手順によって進められる。

1. マスタNIPの接続要求
2. スレーブNIPの接続受理
3. マスタNIPからコマンドの引数の送信
4. スレーブNIPからのパラメータ送信
5. データ転送
6. 回線切断
実行されるコマンドの種類によっては、省略されるフェーズもある。また、リモートな IU へのデータ書き込みを行なう場合のパラメータ転送終了後（データ転送前）と全てのコマンドでのデータ転送終了後（回線切断前）に、それ以前までにネットワーク上でやり取りしたデータ列の垂直バーチャリティを、データ書き込み側に送信し、チェックを行う。ネットワーク上での通信プロトコルの例として、readn コマンドによるベクタ読み出し転送時のタイミングを図 5 に示す。

ネットワークを介した 2 つの NIP は、STB と ACK の 2 本の信号線によりお互いの状態を伝えあい、データの転送を行うが、1 クロック 1 ワードの転送をするために、1 ウェートとのハンドシェークをするのではなく、これら 2 本の信号線は、そのサイクルで、データを送信していることと、データの受信が可能か状態であることを表した制御回路の状態信号となっている。

図 5 の例では、マスタ NIP 側からの接続要求がネットワークを通り、スレーブ側で受け取られたのち、ベクタの先頭に格納されているベクタ長が読み出され、それがスレーブ側から転送パラメータとして送信された後に、バイブライン的な転送状態に入ることが示されている。

6 デッドロックの回避

プロセス間同期処理においては、スレーブ NIP からマスタ NIP への activate コマンドあるいは suspend コマンドが発行される状況があるので、ストアアンドフォワード・デッドロックが発生する危険がある。

すなわち、いま、簡単に説明するためにネットワークは 1 系統しか使っていないものとしたうえで、

1. bind コマンドが IU 0 と IU 1 で立て続けに、互い同時にそれぞれの IU で複数回生成されたとする（この場合、両側で計 4 回以上）。

2. 実体が IU 0 にある未定義論理変数でススペンドしていたコンテクストの実体は IU 1 であり、IU 1 の未定義論理変数でススペンドしていたコンテクストの実体は IU 0 にあるものとする。

と言うと、以下のような状態でデッドロックににおいる。

- Master NIP は、bind コマンドのためにネットワークに接続要求をしており、相手側 Slave NIP の接続応答側で Busy。
- Slave NIP のデータ転送処理部がその 1 つ前に受信した bind コマンドから派生した activates コマンドをプロセス間同期処理部へ送ろうとして Busy。
- Slave NIP のプロセス間同期処理部はさらにその 1 つ前に受信した binds で生じた activate コマンド発行のために、Master NIP の空きを持って Busy。

このような状況は bind コマンドと、すでに束縛されている変数に引き続きで発行された suspend が含まれていても起こり得る。これを図 6 に示す（実線部分）。

この回避のための方法としては、スレーブ NIP において、プロセス間同期処理部が BUSY で、データ転送処理部が送ろうとして suspend コマンドか activate コマンドを受け付けることができず、かつ、データ転送処理部にネットワークからの接続要求が到達した場合に、この
図 6: デッドロックの回避 (1)

プロセス間同期処理部に送るべきコマンドを コマンド・バスを通じて管理プロセッサ側に送し、あとで、管理プロセッサから スレーブ NIP にたいして再発行するようにしている。スレーブ NIP 内部の調停は、ネットワークと コマンド・バスとでは、コマンド・バス側の優先順位が高いように実装している。

もう 1 つの方法としては、PIE64 に 2 系統のネットワークがあることをファームウェア上で積極的に利用する方法が挙げられる。

すなわち、

1. リモート変数に対する bind コマンド及び suspend コマンドを発行する マスタ NIP を、2 系統あるネットワークのうちの一方のもの（例えば PAN）に限るよう、UNIREDI と管理プロセッサに指示する。

2. bind / suspend コマンドを実行するスレーブ NIP が activate コマンドを発行する場合は、もう一方のネットワーク（例えば DAN）のマスタ NIP を用いるようにスレーブ NIP をモード設定する。

3. bind コマンドでバインドする値が変数であった場合、コンテクストの登録し直しのために、通常は suspend コマンドを発行するが、そこで代わりに activate コマンドを発行するようにスレーブ NIP をモード設定する。また、suspend コマンドに

おいて、変数が行い違いで別の変数にバインドされてしまった時も suspend コマンドではなく activate コマンドを発行するように設定する。activate コマンドで送られたコンテクストは、UNIREDI で処理されて結局スケジュールするだから、若干のオーバーヘッドを生じるが、論理的に誤っている訳ではない。この条件により、NIP のコマンドから NIP のコマンドが起動される連鎖は、高々 2 段までに抑えられ、デッドロックが回避される。

これを図 7 に示す。この方式では bind / suspend コマンドの実行のスレーブプットが半分に落ちてしまうことで問題がある。

図 7: デッドロックの回避 (2)

7 性能

基本的なコマンドの実行クロック数については、表 1 のようにになる。NIP の動作モード設定による若干変化するが、最大小で動作する場合をしめす。ここでは、ネットワーク、及び、メモリ・バスのアクセス時に衝突がおこらない場合を考える。また、クロックは、NIP の基本クロックである 10MHz を想定する。

この表中の "Master" は、コマンド・パス上にコマンドのリクエストが出されてから、そのリプライが返るまでのターン・アラウンド・タイムをあらわす。この時点で、すでにネットワークの解
表1: NIPの基本コマンド実行時の処理時間

<table>
<thead>
<tr>
<th>コマンド</th>
<th>Master</th>
<th>Network</th>
<th>Slave</th>
</tr>
</thead>
<tbody>
<tr>
<td>read1</td>
<td>14</td>
<td>11</td>
<td>7</td>
</tr>
<tr>
<td>read2</td>
<td>16</td>
<td>12</td>
<td>8</td>
</tr>
<tr>
<td>readn</td>
<td>17+n</td>
<td>13+n</td>
<td>9+n</td>
</tr>
<tr>
<td>write1</td>
<td>16</td>
<td>14</td>
<td>9</td>
</tr>
<tr>
<td>write2</td>
<td>20</td>
<td>18</td>
<td>14</td>
</tr>
<tr>
<td>writen</td>
<td>22+n</td>
<td>20+n</td>
<td>16+n</td>
</tr>
<tr>
<td>suspend</td>
<td>15</td>
<td>13</td>
<td>10</td>
</tr>
<tr>
<td>bind</td>
<td>19</td>
<td>16</td>
<td>6n+12</td>
</tr>
</tbody>
</table>

単位:クロック

放が行われており、マスタNIPは新しいコマンドを受け付けることができる。“Network”は、このオペレーションによってネットワークが占有される期間であり、また、“Slave”は、マスタNIPにコマンドが送られてから、スレーブNIPでの処理がすべて終了するまでのクロック数を表す。プロセス間同期コマンドでは、サスペシショ
ン・リストのリスト処理のため、ネットワーク接続解放後処理が継続する。bindコマンド欄のnは、サスペション・リストのリスト長をあら

8 終りに

PIE64のNIPのハードウエア構成、内部データ・バス及び処理動作、並びに、ネットワークを介した通信プロトコルについて述べた。

現在、シミュレーションによる最終チェックを行っている。

今後の課題としては、サンプルの性能評価、および、PIE64のIUの設計・実装に伴って、実際の相互結合網を使用した接続試験が行われ

なお、本研究は文部省の特別推進研究No. 62065002の一環である。

参考文献