計算機アーキテクチャ 89−16
（1991. 7. 19）

並列推論エンジン PIE64 のメンテナンス環境

日高 康雄　小池 汎平　田中 英彦
東京大学 工学部 電気工学科

大規模な並列計算機の開発では，メンテナンスを容易にすることが大切である．PIE64 は，専用メンテナンスハードウェア「タコ」を備えており，専用ロジックアナライザ，ネットワークメンテナンス，ホストインタフェース，クロック分配などの機能を提供する．ホスト計算機上に用意されるメンテナンス用のソフトウェアには，きめの細かい 1 回限りのハードウェアデバッグと，何度も繰り返し行なわれるテストのそれぞれに適したツールを用意している．自己診断ができない構成要素を多数持った並列計算機では，タコのような専用ロジックアナライザを備えたメンテナンス環境が有効であることがわかった．

Maintenance Environment of Parallel Inference Engine PIE64

Yasuo HIDAKA, Hanpei KOIKE and Hidehiko TANAKA

Department of Electrical Engineering, Faculty of Engineering,
The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo, 113, JAPAN

Maintenance is crucial for developing a large scale parallel computer. PIE64 has a dedicated maintenance hardware called "TAKO", which offers several functions such as logic analyzers, interconnection network maintenance, host computer interface, clock distribution, etc. There are also several maintenance softwares on the host computer. Those softwares were designed to be suitable for either one time detailed hardware debugging or boring repetitive tests. We find such maintenance environment with dedicated logic analyzers is useful for developing parallel computers with many elements that can not perform self-diagnosis.

1 はじめに

大規模な並列計算機のハードウェアを実現させる場合，デバッグやテストでトラブルが発生することが多い．これには，並列に動作する部分が多いために，同時に測定しなければならない箇所が多いこと，複数のプロセッサが関連するとその動作を予測しにくく，状況の再現が困難であること，また，単体のプロセッサとしては動作しても，複数のプロセッサをつなぐと，クロックのスキューが問題となることなどの理由が考えられる．このような並列計算機を机上の空論ではなく，現実の機械として組み上げ，稼働させるためには，それが研究目的の機械であっても，デバッグやテストの容易さなどの実装技術を重視する必要がある．また，大規模な並列計算機では繰り返し構造が多くあるため，同様の調整を何度も行なわなければならない．これには，1回だけ行なえば済むような調整とは，本質的に異なった調整方法が要求される．

高並列推論エンジン PIE64 には，そのハードウェア・ファームウェア開発を支援するために，「タコ」と呼ばれるメンテナンス専用のハードウェアが備えられている．タコは当初，要素プロセッサ間のネットワークを効率良く調整するための専用ロジックアナライザとして開発されたが，現在は，PIE64 の筐体に常備され，PIE64 全体に及ぶメンテナンス機能を担っている．

本稿では，このタコをはじめとする PIE64 のメンテナンスハードウェアと，ホスト計算機上のソフトウェアから構成されるメンテナンス環境について述べる．

同じようにメンテナンスを重視した並列計算機に，メンテナンスアーキテクチャをとっている電総研の SIGMA-1, EM-4 がある [1, 2]．これはプログラムのデバッグやシステムの初期化などに重点をおいたものであるが，本稿で述べるメンテナンス環境は，ハードウェア障害の診断により重点をおいており，システム開発の効率化をねらったものである．

第2節では，PIE64 について概説する．第3節では，PIE64 のメンテナンス機構である「タコ」について述べる．そして，第4節ではホスト計算機上のメンテナンスソフトウェアについて述べる．

2 並列推論エンジン PIE64

PIE64[4] は，64 台の推論ユニット (Inference Unit – IU) と 2 系統のネットワークとから構成される．図1に，PIE64 の全体構成を示す．

2 系統のネットワーク [5] は，4×4 のクロスバー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．それぞれのネットワークは，Process Allocation Network (PAN), Data Access Network (DAN) と呼ばれるが，これら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．また，このネットワークには自動負荷分散機能があり，負荷最小の推論ユニットを自動的に選択してそこに接続することができる．

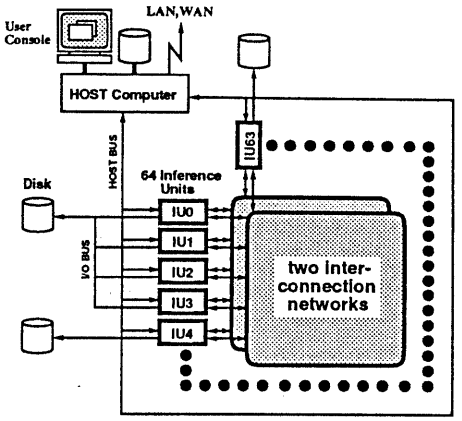

図 1: PIE64 の全体構成

図 2に推論ユニット [6] の構成を示す．我々は，細粒度高並列処理では，通信のレイテンシと同期コストに加えて，負荷分散やスケジューリングなどの並列処理管理が重要になると考え，これらの処理能力を補うために，協調動作する三種類のプロセッサで 1 台の推論ユニットを構成した．

UNIRED (Unifier/Reducer) は，コミティッドチョイス型言語 Fleng で記述された応用プログラムを高速実行する RISC 型のプロセッサで，ユーザの記述したプログラムは，UNIRED によって実行される．NIP (Network Interface Processor) は，「通信と同期」処理を担うプロセッサで，ネットワークの経路制御側を Master NIP，被制御側を Slave NIP が担当する．ネットワークが 2 系統あるため，1 台の推論ユニットは 4 個の NIP を持つ．

SPARC は，「管理」処理を担当するプロセッサで，並列処理管理カーネルと呼ばれるファームウェア (制御プログラム) を実行する．SPARC を用いたのは，さまざまな並列処理管理アルゴリズムの実験を可能とする汎用性と，確実に動作するメンテナンスの容易さを重視したためである．そして，SPARC は，浮動少数点演算コプロセッサ (FPU) と，SPARC バス，ノーウェイトでアクセス可能な SPARC 専用メモリを持つ．

これら 3 種類 6 個のプロセッサは，コマンドバスによって結ばれ，コマンド / リプライをやりとりして協調動作する．SPARC からは，コマンドバスを通じて，UNIRED, NIP のレジスタを読み書きすることもで

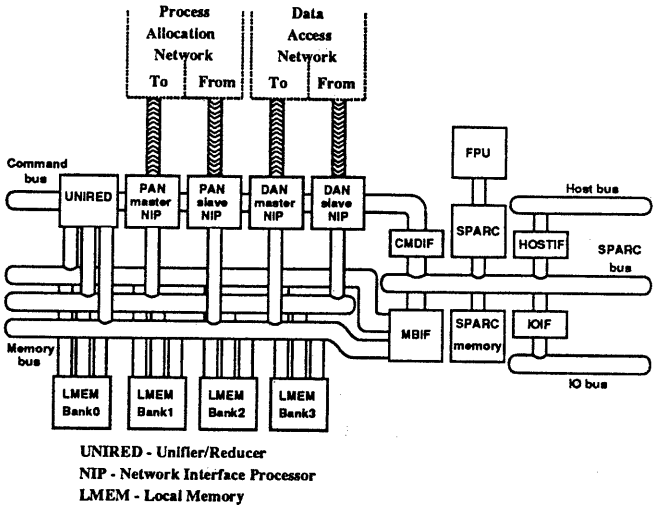

図 2: 推論ユニットの構成

きる. また, これらのプロセッサは, 3本のメモリバスを介して4バンク構成のローカルメモリを共有する.

そして, SPARC からは, IO バスを介してディスクなどの周辺機器をアクセスすることができる. IO バスには, 標準規格である SCSI バスを採用した.

フロントエンドとなるホスト計算機からは, ホストバスを介して, SPARC バスをアクセスすることができ, SPARC のアクセス可能な全ての資源を, 直接アクセスすることができる.

3 PIE64 のメンテナンス機構: タコ

PIE64 のメンテナンス機構「タコ」の機能とその位置付けを図3に示す.

前述したように, タコは当初, ネットワークの調整・テストを, 推論ユニットを使わずに相互結合網単体で行なえるようにするために開発されたが, 現在は, PIE64 の筐体に常備され, 推論ユニットの調整にも使われるほか, PIE64 とホスト計算機とのインタフェース, PIE64 全体へのクロック分配など, PIE64 全体に及ぶメンテナンス機能を担っている.

3.1 ネットワークメンテナンス

タコは, ネットワークメンテナンス機能として, 8個のプローブからなる専用ロジックアナライザと, ネットワークのスキャンインタフェースを持っている. スキャンインタフェースでは, ネットワークの動作を止めた状態で, 各クロスバースイッチの接続状態の読み出しと設定ができる.

専用ロジックアナライザとしての主な特徴には, 以下のものがある.

- ネットワークの基本構成要素である4×4のクロスバースイッチの, 経路制御側4ポートと被制御側4ポート, 合計8ポートを同時にテストできるように, 8個のプローブを備えている.
- プローブの接続には, ネットワークで使用しているコネクタと同じコネクタを使用し, コネクタを1個接続するだけで, データ信号と制御信号あわせて38本の信号を観測できる.
- 信号観測モード以外に, NIP エミュレーションモードを持つ. 制御信号の入出力方向は, Master NIP, Slave NIP のどちらをエミュレートするかによって決まる. データ信号の入出力方向は, Master NIP のエミュレーションでは自分の出力する DIR 信号, Slave NIP のエミュレーションでは, ネットワークから受けとる DIR 信号によって決定する.
- 制御信号を観測して, それに従った信号観測, 信号生成が可能である.
- クロック分配機能と連動して, 単発のクロックを発生させることができ, シングルステップでのデバッグが可能である.

これらの特徴のうち, エミュレーションモードにおける信号生成の機能は, PIE64 のネットワークに特化した機能であるが, NIP のエミュレーションをせずに信号観測だけをするのであれば, 汎用のロジックアナライザとして使うことができる.

そこで, 推論ユニット基板には, 各バス (SPARC バス, コマンドバス, 3本のメモリバス) にあらかじめ接続されたコネクタを用意し, タコのプローブを接続できるようにしている. また, 推論ユニット基板上の

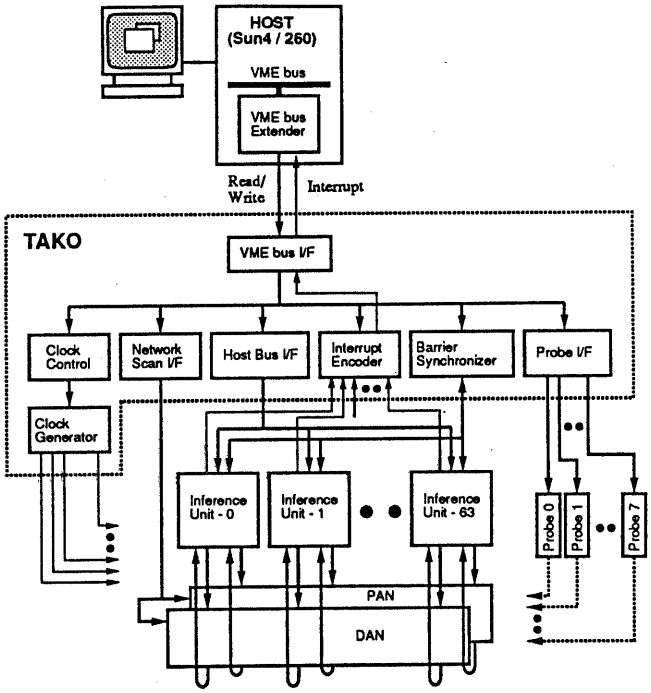

図 3: タコの機能と位置付け

NIP とネットワークの間にも，タコのプローブを接続するためのコネクタを用意している．タコのロジックアナライザ機能を，表 1にまとめる．

3.2 ホストインタフェース

PIE64 のホスト計算機には，SUN4/260 を使う．タコは，VME バスとのインタフェース，4 系統のホストバス (筐体毎に 1 つのホストバスがあり，16 台の推論ユニットが接続される) とのインタフェースを持ち，推論ユニット上のレジスタ，メモリをホストからアクセスする機能，各推論ユニットからホストへ割り込みをかける機能を提供する．

ホストバスは，ホスト計算機からのアクセス専用のバスであり，各推論ユニットがバスマスタになることはできない．

3.2.1 推論ユニットのレジスタ，メモリのアクセス

各推論ユニットの SPARC のアドレス空間 (16M Byte) は，VME バスの物理アドレス空間 (32 ビット拡張アドレス空間) にマッピングされ，Unix 上のプログラムは，/dev/vme32d32 を mmap システムコールで論理アドレス空間にマッピングして，直接アクセスすることができる．

このマッピングには以下の 3 種類があり，それぞれ VME バス上の異なるアドレスにマッピングされている．

- Direct Mapping
  各推論ユニットのアドレス空間を，全て異なるアドレスにマッピングする．この方式は，全ての資源がアドレスのみで区別されるため，プログラミングが容易であるが，マッピング領域が広大 (1G Byte) となるために，ホストから頻繁に全推論ユニットの資源を一様にアクセスすると，SUN4 MMU の PMEG (Page Map Entry Group) の入れ換えが頻発し，ホストのパフォーマンス低下を招く恐れがある．このため，本方式は，アクセスがまばらに行なわれるような場合に適していると考えられる．
- Window Access
  これは，ホストバス毎に一つのウィンドウを用意し，16 ユニットのうちの 1 台の推論ユニットをウィンドウを通してアクセスする方式である．この方式は，アクセスする前に，16 台のうちの 1 台を選択するバンクレジスタを設定する必要があるため，プログラミング上の注意が必要であるが，全推論ユニットを頻繁にアクセスする場合は，Direct Mapping よりもマッピング領域が狭いため，PMEG の入れ換えを抑えることができる．
- Broadcast

表 1: タコのロジックアナライザ機能

項目	説明
プローブ数	8 個
観測信号数 (プローブ当たり)	データ信号 32 本, 制御信号 6 本, 補助入力信号 2 本
アクウィジションステップ数	64K ステップ
サンプリング周波数	10MHz または 5MHz (PIE64 本体のクロックと共通)
NIP エミュレーション機能	Master NIP または Slave NIP を模擬して, 信号記録と信号生成の両方を行なう. データ信号の入出力の向きは, 制御信号の DIR に従う. (DIR は, Master NIP では出力信号, Slave NIP では入力信号.)
シーケンス制御機能	アクウィジション・パターンメモリの各ステップに, シーケンサの制御コマンドを入れて, 入力される制御信号を監視しながら, 複雑な信号記録や信号生成ができる.
イベント検出機能	制御信号, 補助入力信号を監視して, 設定された条件を満たした時にイベントを発生できる. イベントの発生によって, プローブのシーケンサをスタート, ストップさせることができる.
プローブ接続方法	データ信号, 制御信号は, 50 ピンまたは 80 ピンのコネクタ 1 個で 1 つのプローブを接続. 補助入力信号は, 個別にテストフックで接続.

これは, 書き込み専用のウィンドウで, 全推論ユニットに同時に書き込みを行なう方式である. この機能は, 初期プログラムのロードや, 全ての SPARC に一括して割り込みをかける時などに用いる.

3.2.2 ホストへの割込み

各推論ユニットから出されたホストへの割込み要求は, タコによって一つにまとめられて, ホストに割込み要求が出される. タコには, 割込み要求を出している推論ユニット番号を示すレジスタがあり, どの推論ユニットからの割込みかを直ちに特定できる. ホスト計算機の Unix カーネルには専用のデバイスドライバを組み込み, この割込み要求を処理する.

ホスト計算機から各推論ユニットの SPARC への割込みは, ホスト計算機から推論ユニット上のレジスタに書き込みを行なうことによって発生させる. このホスト計算機からの割込みには, 3 本の割込みを用意している.

また, 通常割込み以外にも, ホスト計算機から推論ユニットをリセットしたり, 10MHz クロックの停止や単パルス動作などを行なうことができる.

上記のホストインタフェース機能の他に, 全推論ユニットの SPARC の間のバリア同期をとるため機構が, ホストバス上の信号線とタコ上の簡単な回路で実現されている. これは, 全推論ユニットの出力値の AND をとった値を, 各推論ユニットに戻すというもので, それが 8 系統用意されている. このうち 3 系統は, SPARC に割り込みを起こすこともできる.

3.3 クロック分配

PIE64 は全体で一つの基準クロックに同期して動作する. 基準クロックは, タコの上のクロックオッシレータで作られた後, ツリー状に構成されたクロックバッファで駆動され, 等長配線の同軸ケーブルによって, 推論ユニット, ネットワークに分配される. 以下に, タコのクロック分配機能の特徴を挙げる.

- 基準クロックの周波数は通常 10MHz であるが, デバッグ時には半分の 5MHz に落すこともできる.
- ロジックアナライザ機能と連動して, クロック供給の停止, 単発のクロックの生成などができる.
- ネットワークに供給するクロックは, ネットワークの接続時間を短縮するために, 位相を 90° 単位でずらすことができる.

一方, 推論ユニットに用いる SPARC には, 図 4に示すような周波数 20MHz デューティ比 75% の 2 相クロックを供給しなければならず, そのスキュー条件を満たすためには, 4 倍の周波数の 80MHz のクロッ

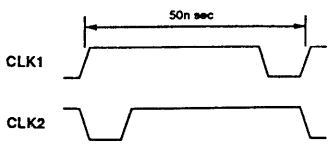

図 4: SPARC 用のクロック波形

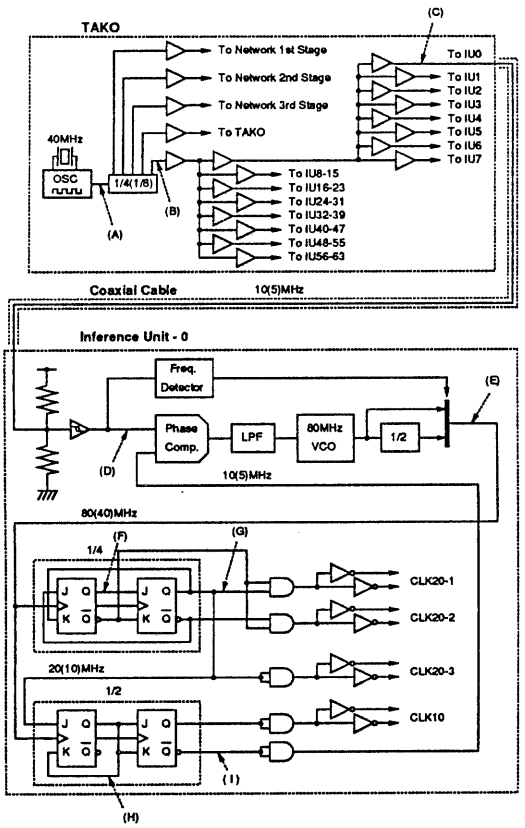

図 5: PIE64 のクロック分配回路の概要

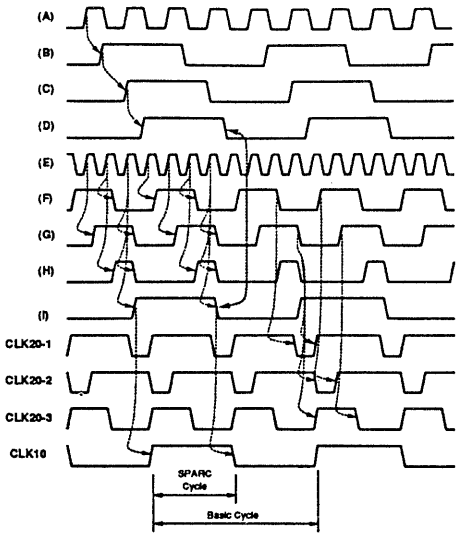

図 6: クロック分配回路のタイムチャート

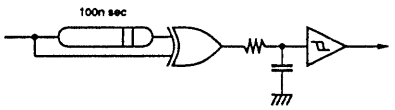

図 7: クロック周波数検出回路

クからそれを作る必要がある.

このSPARC用のクロックも, 10MHzの基準クロックに同期させなければならないが, タコから各IUに供給する方法では, 80MHzのままで供給しても20MHzにしてから供給しても, デューティ比の保証, 2相クロック間のスキュー条件, 基準クロックとの間のスキュー条件の全てを満たすのは困難である.

そこで推論ユニット上にPLL(Phase Locked Loop)回路を載せて, 内部で発生させた80MHzを分周して作った10MHzを, タコからの基準クロックに同期させるようにした. この80MHzからSPARC用のクロックを生成すれば, 基準クロックに同期したクロックが得られる. クロック分配回路の概要を図5に, 回路の動作タイミングを図6にそれぞれ示す.

また, 周波数可変範囲の広いVCOの入手が困難であったため, このPLL回路では, VCOの発振周波数は80MHzに固定して, 5MHzが供給された場合には, 分周比を1/8から1/16に切替えるようにしている. この切替えにはジャンパピンを使わずに, 供給されている基準クロックの周波数を判断する図7のような簡単な回路 (図5中のFreq. Detector) を用いており, タコ上の基準クロックの周波数を切替えるだけで, PIE64全体の動作周波数を切替えられるようにしている.

4 PIE64のメンテナンスソフトウェア

ホスト計算機上のソフトウェアの助けを借りて行なわれるメンテナンス作業には, 以下のものがある.

- ハードウェアのデバッグ, 保守
- ハードウェアのテスト
- ファームウェアのデバッグ

はじめの二つは, ハードウェアに関するメンテナンス作業であるが, このうち, ハードウェアのデバッグ, 保守は, 障害の存在が明らかである場合に, その原因を特定して取り除く作業であるのに対して, ハードウェアのテストは, 多数の量産品や繰り返し部分に対して, 障害の有無を調べていく繰り返し作業である. 前者は, できるだけ詳細に分析していく, きめの細かい作業が必要であり, 後者は, なるべく短時間で確実に, 障害の有無を判定する必要があるため, これらは本質的に異なった作業となる.

ネットワークのメンテナンス作業を行なうために, 以下のツールを作成した.

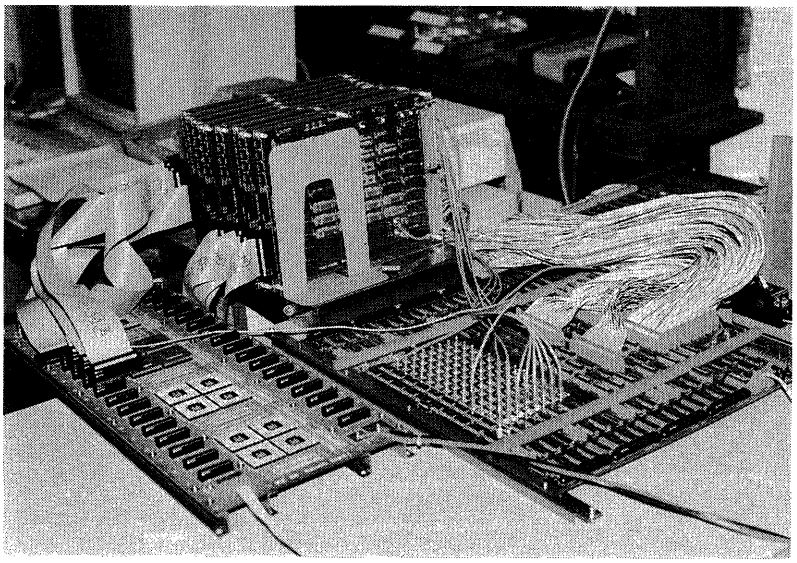

図 8: ネットワークの基板テストの様子

- nethack
  タコのロジックアナライザ機能を活用するツールで，各プローブのモード設定，イベントパタンの設定，アクウィジション・パターンメモリ内容の設定や読みだし，ファイルへのセーブなどの機能を持つ．コマンドはテキスト形式で与え，ユーザが与える対話型の動作と，コマンドをスクリプトファイルから読み込むバッチ型の動作ができる．
- xtimechart
  nethack でセーブしたアクウィジション・パターンメモリ内容のファイルを，X Window 上にタイムチャート形式で表示するツール．動作の結果が正常かどうか，どこがどのように異常かを詳細に調べるために使う．
- nhdiff
  nethack でセーブしたアクウィジション・パターンメモリ内容の二つのファイルを，比較するツール．正常であることを xtimechart で確認済みであるファイルと，正常であるかどうかを調べたいファイルを比較するために使う．
- xnetscan
  タコのスキャンインタフェースを使って，ネットワークの接続状態を読み出して X Window 上に表示したり，X Window 上で指示した接続状態に，ネットワークを設定するツール．

デバッグ，保守作業には，nethack, xtimechart, xnetscan を使い，テストには，nethack, nhdiff と，適宜作成したシェルスクリプトと nethack スクリプトを使用する．

これらのツールのおかげで，ネットワークのデバッグ，テストを順調に行なうことができた．例えば，図8の写真に示すように，ネットワークを組み上げる前のネットワーク基板のテストは，全部で 96 個あるクロスバースイッチ毎に行なったが，決められた手順でテストすることによって，障害がない場合は 1 箇所当たり 10 分程度の作業で済んだ．ネットワークを組み上げた後は，図 9のようにさまざまな組み合わせでのテストを行ない，正常動作を確認している．

このように，並列計算機では同様のテストを何度も繰り返さなければならない．これが要素プロセッサのテストであれば，自己診断プログラムを走らせて，各部の動作が正常であるかどうかを調べることができるのであるが，プログラム可能なプロセッサを持たない相互結合網のような部分は，単体では自己診断を行なえない．要素プロセッサと接続した後ならば，要素プロセッサ上の診断プログラムでテストできるが，多段結合網のように構成要素の数が非常に多い場合は，組み上げる前にもテストを行なう必要がある．タコのような専用ロジックアナライザを用いる方法は，このように自己診断ができない構成要素が多数存在する場合に特に有効であると考えられる．

今後は，推論ユニットハードウェアのメンテナンス，ファームウェアのデバッグを行なうために，以下のような拡張や新たなツールを予定している．

- xnethack
  テキストベースの nethack を X Window 対応にして操作性を向上させ，xtimechart や xnetscan と連係して動作するようにしたツール．
- xtimechart (拡張)
  タイムチャートだけでなく，推論ユニットの

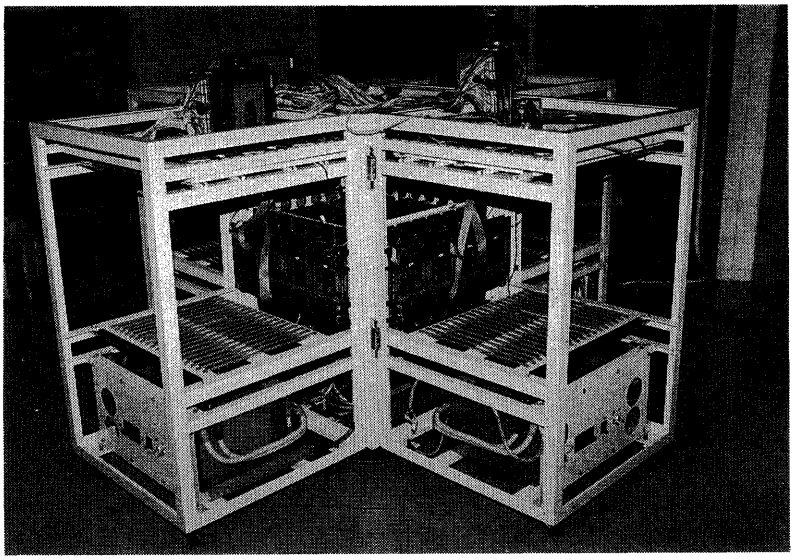

図 9: 組上がったネットワークのテストの様子

SPARC バス，コマンドバスの命令やアドレスなどは，ニーモニックやシンボルで表示できるようにして，タコのプローブをデバッグモニタとして使えるようにする．

- piegdb
  SPARC のファームウェアをデバッグするためのソースコードデバッガ．SPARC 上には，レジスタの読み書きやシングルステップ実行をするためのモニタプログラムをのせ，これを使ってリモートデバッグを行なうように改造した GDB(GNU Debugger) をホスト計算機上で走らせる．

5 おわりに

本稿では，PIE64 のメンテナンス環境を構成するハードウェアとホスト計算機上のソフトウェアについて述べた．ハードウェアに関しては，メンテナンス機構「タコ」の機能を中心に，専用ロジックアナライザ，ネットワークメンテナンス，ホストインタフェース，クロック分配について述べた．ソフトウェアに関しては，ハードウェアのデバッグとテストの違いと，それらを支援するために用意したツールについて述べ，今後，用意する予定のツールについても述べた．また，自己診断ができない構成要素が多い場合には，専用ロジックアナライザが有用であることを指摘した．

1991 年 6 月現在，ネットワークは完全に動作しており，推論ユニットは SPARC がほぼ動作している状態である．今後は，ゲートアレイで作成している NIP, UNIRED を含めて，推論ユニットのデバッグ，テストを行ない，SPARC のファームウェア等の開発を行なっていく予定である．

なお，本研究は文部省特別推進研究 No.62065002 の一環として行なわれた．

参考文献

[1] Hiraki, K., Nishida, K., Sekiguchi, S. and Shimada, T. : Maintenance Architecture and Its LSI Implementation of a Dataflow Computer with a Large Number of Processors, Proc. Int. Conf. Parallel Processing, IEEE, pp.584-591 (1986).

[2] 児玉, 坂井, 山口: データ駆動型シングルチッププロセッサにおけるメンテナンスアーキテクチャ, 第 39 回 情報処理学会 全国大会, pp.1788-1789 (1989).

[3] 日高, 高橋, 小池, 清水, 田中: PIE64 のネットワーク.メンテナンス, ホストインタフェース, クロック分配機構 : タコ, 第 40 回 情報処理学会 全国大会, pp.1171-1172 (1990).

[4] 小池, 田中: 並列推論エンジン PIE64, bit 臨時増刊 並列コンピュータアーキテクチャ, Vol.21, No.4, pp.488-497 (1989).

[5] 高橋, 小池, 田中: 並列推論マシン PIE64 の相互結合網の作成および評価, 情報処理学会論文誌, Vol.32, No.7, (1991).

[6] 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, コンピュータシステム研究会 CPSY90-44, 電子情報通信学会技術研究報告, Vol.90, No.144, pp.37-42 (1990).