SSS 型 MIN に基づくマルチプロセッサ SNAIL の実装と評価*

笹原正司, 寺田純, 大和純一, 塙敏博, 天野英晴†
慶應義塾大学理工学部
E-mail: {sasahara, terada, yamato, hanawa, hunga}@aa.cs.keio.ac.jp

概要

我々は高速なプロセッサ・メモリ間ネットワークとして SSS ネットワークアーキテクチャを提案している。この SSS 型 MIN は, 基本的にシリアル転送を行なう事で高速性とピンネックを解消したもので, 既にチップの実装が行なわれている。
現在, このチップを用いて,16 プロセッサからなるマルチプロセッサ SNAIL の作成を行なっており,12 プロセッサ・4 メモリモジュールで稼働中である。
本論文では,SNAIL 上に並列プログラムを実装し, データプリフェッチやメッセージコンバインの効果を含め, ネットワークおよび SNAIL の評価を行なう.

Abstract

Simple Serial Synchronized (SSS) Multistage Interconnection Network (MIN) is a novel MIN architecture for connecting processors and memory modules in multiprocessors. Synchronized bit-serial communication simplifies the structure/control, and also solves the pin-limitation problem. The simple structure allows the use of a high frequency clock rate and a high throughput network topology. An LSI implementation of the SSS-MIN has been completed.
Using these chips, a multiprocessor prototype called SNAIL with 16 processors/16 memory modules is under development, and a system with 12 processors/4 memory modules is operational. The small overhead of the SSS-MIN is demonstrated with some parallel benchmark programs. The performance enhancement mechanisms such as data prefetch and message combining are also evaluated.

1 はじめに

多段結合網 (Multistage Interconnection Network: MIN) は中規模 (数十から数百プロセッサ) の並列計算機におけるプロセッサとメモリ間の接続に多く用いられる結合方式であり, 現在までに様々な研究がなされてきた. これらの MIN では, メモリに対するアクセスはパケットの形で行なわれ, パケットはネットワーク内のスイッチングエレメント間をビットパラレル (8~64bit) で転送される. 結合網のトポロジは Omega 網を代表とするブロッキング網で, スイッチングエレメント内で衝突が起きた場合, 片方のパケットはエレメント内のバッファに格納される事になる.

この様な従来型の MIN は, 高速, 高密度実装に困難な点が多く, これを用いたマルチプロセッサシステムの実装の遅れ, 規模縮小 [1][2] の原因の 1 つになってきた. まず, Store&forward 方式でパケットが転送される為, 各スイッチにパケット格納の為のバッファやその制御機構を設けなくてはならず, この為エレメントは複雑なものとなってしまう. 更に, ビットパラレル転送を行なう為にはプロセッサ当たり 8~64bit の入出力線が必要となり, LSI チップ中に実装できるエレメント数がピン数によって制限されてしまう. これによって, 全体として用いるチップ数, 基板面積が増大することとなり, 高速なクロックの利用が困難になる.

これらの問題点の解決の為, 我々は, SSS(Simple Serial Synchronized) 型 MIN を提案している [3][4]. SSS 型 MIN では, パケットは基本的にはビットシリアルに, フレーム信号に同期して転送される. ビットシリアルの同期転送にする事により, エレメントの制御機構が単純化され, 高速なクロックが利用可能になると共に, LSI のピン数制約の問題を解決し, 高密度実装が可能になる.

又, 動作上の特徴を生かす事により, 少ないハードウェアでメッセージコンバインの実装が可能である. 更に, SSS 型 MIN は極めて規則的な動作を行なう為, 全プロセッサを同期して動作させる事でマルチプロセッサシステム全体を規則的に動作させる事ができる. この動作の規則性は通常のプログラミングに際して高いデバッガビリティを提供するだけではなく, インストラクションレベルでのスケジューリングを容易にする.

我々は, SSS 型 MIN のプロトタイプチップを実装し, これを用いたマルチプロセッサプロトタイプ SNAIL(SSS Network Architecture ImpLementation) を開発している. 現在, 16 プロセッサのシステムを調整中で, 12 プロセッサの構成で評価を行なう事ができたので, この結果を報告し, 検討する.

2 SSS 型 MIN

2.1 SSS 型 MIN の基本構造

SSS 型 MIN の基本動作を図 1に示す. プロセッサからのアクセスは MIN との間にあるバッファでシリアル化され, ISDN 等で用いられている ATM 用交換システム同様, フレームクロックと呼ばれるクロックに同期してネットワークに入力される. スイッチングエレメント内にはバッファは設けられておらず, 各エレメントは 1bit 分の記憶しか行なわない為, ネットワークは全体としてスイッチング機能のあるシフトレジスタの様な動作をする. そして全ステージを通過する分の遅延の後, パケットの先頭が出口から出力される.

SSS 型 MIN ではパケットバッファを持たない為, スイッチに入力された二つのパケットが同一出力に向かった場合, 一方のパケットは正しい方向に進むが, 他方は正しい方向に転送されない. この様な場合, パケットに用意されたコンフリクトビットがセットされ, 以後このパケットはデッドパケットとみなされ, 他のパケットを妨害しない.

パケットのアドレスの先頭が出口に到達すると, ネットワークの全てのエレメントの状態が設定され, 入力から出力への論理的なパス (これをトレースと呼ぶ) が設定される. トレースが設定されると, パケットのコンフリクトビットを MIN の出口で参照する事により, パケットが正しく転送されたかを調べ, ACK(ACKnowledgement) もしくは NAK(NotAcKnowledgement) がプロセッサ側に返される. このアクノリッジ信号は, MIN の各段でレジスタにストアされる事はなく, マルチプレクサと IC 内の配線による遅延のみでプロセッサ側に到達する. プロセッサ側が NAK を検出した場合, プロセッサは次のフレームでパケットを再送しなくてはならない.

---

*SNAIL:A Multiprocessor Based on SSS Network Architecture

†Masashi SASAHARA, Jun TERADA, Jun-ichi YAMATO, Toshihiro HANAWA, Hideharu AMANO Dept. of Electrical Engineering, Keio Univ.

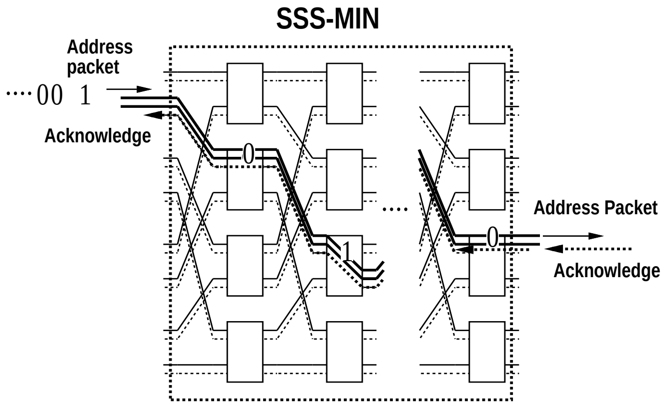

図 1: Structure of the SSS-MIN

2.2 パイプライン化サーキットスイッチング

アドレス転送によってトレースが設定されると，アクノリッジおよびデータパケット転送のトレースも設定される．SSS 型 MIN ではアドレス，データ，アクノリッジは独立しており，図 2 に示す通り，これらの転送はオーバーラップして行われる．図中，フレーム i でアドレスが転送され，トレースが形成されると，同時に各入力に対してアクノリッジ信号 (ACK/NAK) が返される．正しく転送が行なわれ，ACK が返送された場合は，このトレースを利用し，フレーム $i+1$ でデータの転送が行なわれる．(これをパイプライン化サーキットスイッチングと呼ぶ.)

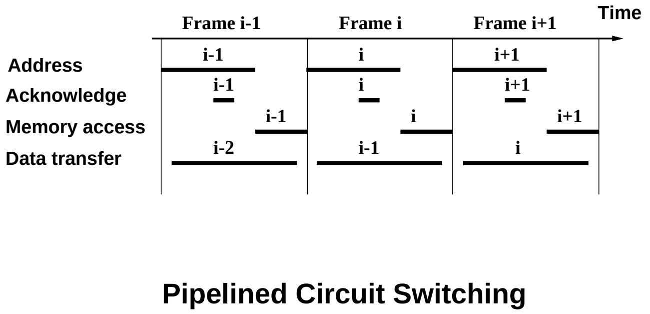

図 2: SSS-MIN でのパケット ·信号のタイミング

2.3 高速化手法

2.3.1 メッセージ結合機能

メッセージ結合機能は同一宛先のパケットをエレメント内で結合し，1 つのパケットとして扱うことにより，Hot stop へのアクセス集中による Tree saturation を回避する方法である [5]．この機能の効果はシミュレーション等で確認されているが，従来型の MIN でこの機能を実現する為には 6 倍から 32 倍のハードウェアを要する [6] 為，実現が困難であった．

しかし，SSS 型 MIN ではシリアル転送の特性を生かして簡単に実現できる．図 4(a) に読みだしアクセス同士の結合の実現例を示す．各エレメントはアドレスパケット通過時に入力された 2 つのパケットの全アドレスを比較する．そして，両方のアドレスが完全に等しければ，データ転送路のマルチプレクサを図中に示す様に Backward Broadcasting モード (B) にセットし，全体として Tree 状のトレースを形成する．読み出されたデータはこのトレースに従ってマルチキャストされ，同時に各プロセッサに到着する．この方法では，比較とマルチキャストがビットシリアルに行われる為，付加回路が簡単で，動作速度に与える影響も少ない．

この方法では Test&Set の結合も実現されている (図 4(b))．アドレス通過時には，図 4(a) 同様にエレメント間に Tree 状のトレースが形成されるが，データの転送時に，片方のデータ転送路に強制的に H レベル (all-1) が挿入される．この為，ただ一つのプロセッサだけが，メモリからのデータを受け取り，残りのプロセッサは all-1 のデータを受け取る．読み出し直後，メモリコントローラによってメモリの値は all-1 とされるので，全体として Test&Set 操作の結合が実現される．

2.3.2 ステージの飛び越し

アドレスパケットのルーティングタグにより一度トレースが設定されてしまえば，アドレスパケットの残りと，データパケットは，既に設定された経路をたどっていけば良い．この為，各ステージで記憶を行う必要は無く，応答確認信号 (ACK/NAK) 同様に，動作速度に応じて，幾つかのステージを飛び越して転送する事ができる．

2.4 結合網のトポロジ

SSS 型 MIN では，パケット同士の衝突が起きた場合，入力バッファから再送する必要があり，性能低下の大きな原因となる．この為，SSS 型 MIN で用いる結合網は従来型 MIN で用いられるブロッキング網よりも高い通過率が要求される．一般に通過率の高い網はスイッチングエレメントを数多く必要とするが，SSS 型 MIN ではスイッチングエレメントの構造が簡単である為，ある程度の数は許容できる．

そこで，我々は，B-ISDN(Broadband Integrated System Digital Network) で用いられるパケット交換用に提案した TBSF(Tandem Banyan Switching Fabrics)[7][8] を検討し [3]，更にこれを改良した PBSF(Piled Banyan Switching Fabrics)[9] を提案した．これらの結合網は同一宛先のパケットを複数通過させる事ができ，ノンブロッキング網より高い通過率を持つ．ここでは，SNAIL で用いられる TBSF について簡単に紹介する．

TBSF は，図 3に示す様に banyan 網 (Omega 網) を次々と接続し，各網の出口にバイパス路を設けた構造を持つ．宛先に到着したパケットはバイパス路によりそのままメモリモジュールに送られ，衝突により到着しなかったパケットのみが次の段の banyan 網に入力される．後に述べる様に，SSS 型 MIN では，パケット内に他のパケットとの衝突を示す conflict bit を設け，衝突により希望しない方向に送られたパケットが正しくルーティングされたパケットを妨害しない機構を用いている．ある banyan 網でセットされた conflict bit は，次の網の入力でリセットされ，次の banyan 網では最初の網と同様なルーティングが行なわれる．

この網は厳密にはノンブロッキングではないが，同一宛先のパケットを複数通過させる事ができる為，ノンブロッキング網より高い通過率を持ち，交信の偏りに対しても強靭である．TBSF は B-ISDN 用の交換機に用いる場合，各 banyan 網から時間的にずれて到着するパケットを揃える部分が問題になる．しかし，SSS 型 MIN として用いる場合，到着順にメモリのアクセスを行なえば良いので，メモリ利用率を上げる事ができ，利点に転じる．更に，モジュラリティに優れ，耐故障性にも優れた点を持つ [10]．性能面では banyan(omega) 網を三次元的に接続した構造を持つ PBSF[9] に及ばないが，ハードウェア量が小さく，実装が容易である．評価の結果，ビットシリアルメッセージコンバイン，ステージの飛び越し等の高速化技法を用いる事により 256 入力について，従来型の MIN の 1/8 程度のチップ数で同等もしくはより優れた性能を実現できる事が明らかになっている [9]．

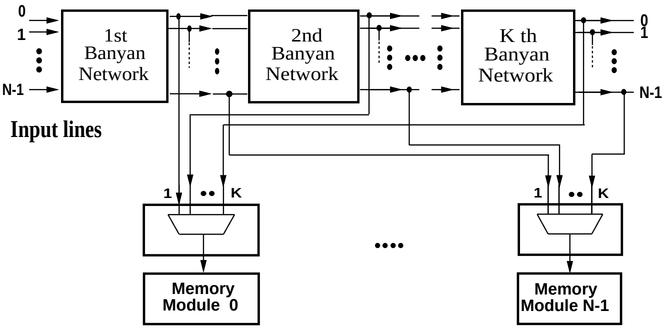

図 3: Tandem Banyan Switching Fabrics

3 マルチプロセッサプロトタイプ SNAIL

3.1 SNAIL の概要

実際のアプリケーションプログラムを用いて SSS 型 MIN の性能評価をする為, マルチプロセッサプロトタイプ SNAIL(SSS Network Architecture ImpLementation) を実装中である. 図 4 の様に, プロセッサとローカルメモリから成る 16 個の PU は SSS 型 MIN を介して共有メモリモジュール 16 個と接続されている. 1 つのプロセッサは Ethernet インタフェースを持ち, ホストのワークステーションと結合されている. 又, パフォーマンスモニタとデバッガのシステムは SNAIL と独立に動作し, MIN の両側を監視している. 全てのシステムは単一システムクロック (50MHz) と, フレームクロック (50/163.125MHz) を共有し, 同期動作する.

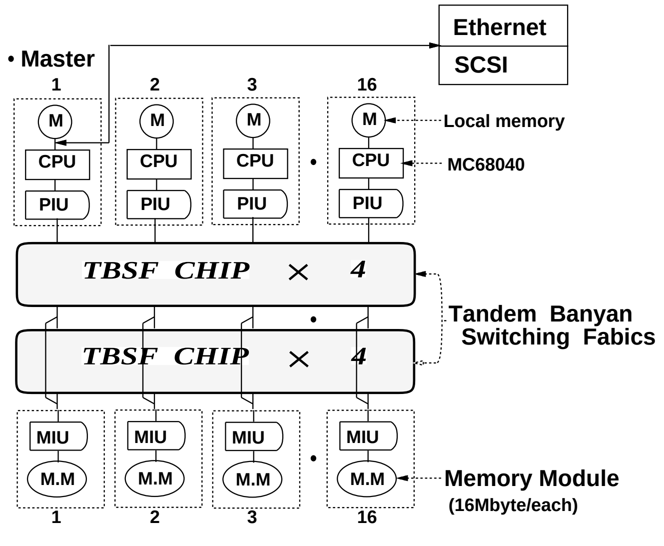

図 4: Diagram of a multiprocessor SNAIL

SNAIL の主な目的は SSS 型 MIN の性能評価にある. 特に, SNAIL は 10 以上のプロセッサを持つマルチプロセッサの中で, メッセージコンバイン機能を実装した初めてのマシンである. 又, SNAIL は SSS 型 MIN の動作の再現性を利用して, 静的スケジューリングのテストベッドとしても設計している. SSS 型 MIN では, 動作が単純で同期して行なわれる為, 動作の再現性が極めて高い. 並列化コンパイラや最適化コンパイラが進歩すれば, ネットワークのレイテンシを隠蔽し, ほとんどの同期操作を省くようなインストラクションレベルでスケジュールしたコードを生成する事ができる [11].

このため, SNAIL は以下のような機能を提供している.

- 全プロセッサは SSS 型 MIN で用いているクロックに同期して動作する. 全ての共有メモリモジュールのリフレッシュと I/O の動作も SSS 型 MIN のフレームクロックに同期している. SSS 型 MIN ではパケットの転送はフレームに同期しており優先度は原則として固定である為, プロセッサの動作のタイミングが狂うことはない. この様なシステムでは, インストラクションレベルのスケジューリングによって, ほとんどの同期のコードを省略する事ができる. しかも, 並列プログラムはいつも全く同じ動作をする.

- 共有メモリに対しては, プリフェッチ, ライトバッファ, 共有メモリとオンチップキャッシュ間のブロック転送をサポートしている. これらの機能を用いれば, 最適化コンパイラによって, 共有メモリアクセスのレイテンシを隠蔽する事ができる.

- 従来型 MIN を用いたマルチプロセッサでは, ハードウェアモニタを用いて共有メモリへアクセスの解析をしようとしても, 非常に多くの線をモニタしなくてはならず, 不可能に近い. しかし SSS 型 MIN では, パケットが通過する線が少ないため, それらをモニタしてトラフィックを解析する事ができる. このモニタ機構を用いて, 共有メモリブレークポイントも実装している. プロセッサがある共有メモリアドレスにアクセスすると, ネットワークの動作は停止し, 全てのプロセッサに割り込みがかかる. これらの機能は, スケジューリングされたコードの性能解析とデバッグに有効である.

以上の目的も満足するため, SNAIL は絶対性能よりも柔軟性を重視した設計になっている. PU-MIN インタフェースとメモリコントローラは Xilinx 社の FPGA XC3090 を用いて実装している. FPGA の動作速度は速くはないが, コンフィグレーションデータを書き換える事により, ハードウェアを変更する事ができる.

SNAIL で用いた SSS 型 MIN の詳細を述べる前に, PU とメモリモジュールを簡単に述べると:

- PU

  Motorola 社の MC68040 を使用. SSS 型 MIN で用いるクロックは全プロセッサで使用. 今回の実装では, 各プロセッサはローカルメモリとして 0.5Mbyte の SRAM を持つ.

  プロセッサのバスと SSS-TBSF チップ間のパラレル/シリアル変換をする周辺回路と SSS 型 MIN に対するインタフェースは FPGA(Xilinx 社 XC3090) を 2 チップ使用している. このインタフェース内には 2 ワードのアドレスパケットバッファと 2 ワードのプリフェッチバッファを実装している. NAK 信号が MIN から返ってきた場合には, アドレスパケットは自動的にアドレスパケットバッファに再び挿入される.

  プロセッサは Ethernt を介してホストのワークステーションと接続され, ブートストラップ用の ROM 中の簡単なモニタ機能によりワークステーションからプログラムをロードする.

- メモリモジュール

  共有メモリは各モジュールが 4Mbyte の DRAM で, 全体で 64Mbyte であり, オンチップキャッシュのブロックサイズ (4 ワード) でインタリーブしている.

  メモリコントローラは, パラレル/シリアル変換と同期操作, ブロック転送, メモリリフレッシュ等の制御をしている. 又, 全てのメモリに対する Test&Set と Fetch&Dec の 2 つの同期機構を行なっている.

  動作の再現性を確保するため, 全メモリモジュールのリフレッシュはフレーム信号に同期して行なわれる.

メモリインタフェースも Xilinx 社の FPGA XC3090 1 個で構成される.

図 5: SNAIL の概観

3.2 プロトタイプ SSS-TBSF チップ

SNAIL では, ビットシリアルメッセージコンバイン機能等全ての機能を持つ SSS-TBSF ゲートアレイチップを用いている[1]. 16 入出力のオメガ [12] 網を 1 チップに実装し, TBSF は, 単に直列に接続すれば構成できるようになっている. 256 入出力のオメガ網は 16×2 個のチップを接続する事で構成できるため, 64 チップで Banyan 網 2 段の TBSF を構成する事ができる.

このチップの仕様を以下の表に示す.

表 1: SSS-TBSF チップの仕様

Clock: 50MHz	Bit width: 3bit(address), 2bit(data)
Size: 16-in/out	Max bandwidth: $250 Mbits/sec \times 16$
セル使用数: 9838	セル使用率: 49%
信号ピン数: 180	テクノロジ: $1.0\mu m$ CMOS sea-of-gates

図 6の様に, チップは入力インタフェース, ネットワーク, 出力インタフェースで構成される. 入力インタフェースは, トレースをセットする間, アドレスパケットの先頭を保持する為のバッファを持っている. 有効パケットのコンフリクトビットも入力インタフェースでリセットされる. ネットワークは 2 入出力のスイッチングエレメント 32 個を用いて 4 段のオメガ網になっている. 出力インタフェースはパケットが届いているかをチェックし, ACK/NAK 信号を生成する. 出力インタフェースは 1 ビットだけを保持する.

使用したセル数は 9838 で, メッセージコンバイン機能には全体の 20%しか使用していない. この事から, ビットシリアルメッセージコンバイン機能を低コストで実装している事が分かる.

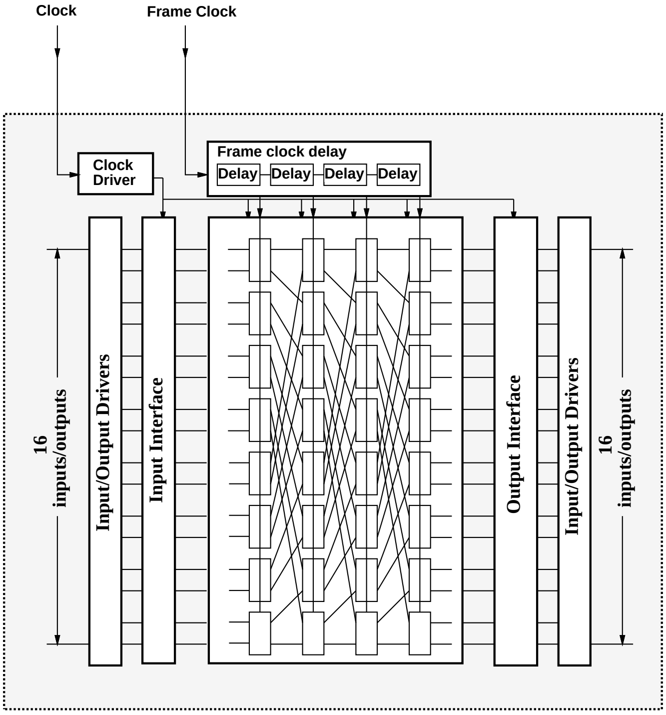

図 6: SSS-TBSF チップの構成

1 つのスイッチングエレメントは 192 セルで構成され, 70%以上のセルはアドレス線のコントローラに使用されている.

3.3 SSS-TBSF チップのスイッチング機構

前に述べた様に, SSS-TBSF チップ 1 つで, 16 プロセッサ, 16 メモリモジュールを接続する事ができる. しかし, SNAIL ではビットスライスにチップを 4 つ用い, アドレス幅 12 ビット, データ幅 8 ビットにして, レイテンシを改善している. Banyan 網 2 段にするには, 8 チップ (4×2) を使用する事になる. 最大バンド幅は, 各プロセッサ/メモリモジュールが 125Mbyte/sec であるので, 全体で 2Gbyte/sec になる.

共有メモリへのアクセスは以下の様にして行なう.

- リード : プロセッサはメモリからデータが戻ってくるまで待つ.
- ライト : プロセッサはアドレスパケットバッファが空いていれば, すぐに次の命令を実行できる.
- プリフェッチ : ライトと同様, プロセッサはすぐに次の命令を実行できる. プリフェッチバッファにデータが届いていれば, 遅延無しでデータを読む事ができる. そうでなければ, プロセッサはリードと同様に待つ事になる.
- ブロック転送 : 共有メモリのオンチップデータキャッシュを使用している時には, キャッシュライン (4 ワード) が DRAM のページアクセスモードで即座に転送を行なう. 行列演算や, 共有メモリとローカルメモリ間のデータ転送に有効である.
- 同期操作 : SNAIL では Test&Set と Fetch&Dec をサポートしている. Test&Set 動作は MIN でコンバインされ, Fetch&Dec はメモリインタフェース内で行なわれる. SNAIL では, SSS 型 MIN を介して共有変数へのアクセスを必要としない簡単なバリア同期機構を実現している.

ビットシリアルメッセージコンバイン機能はリード, プリフェッチ, Test&Set で行なわれる.

---

[1] このチップは川崎製鉄株式会社の協力のもとに作成された.

4　性能評価

現在, SNAIL は 12 プロセッサ 4 メモリモジュールで稼働中である. また, ネットワークは, TBSF1 段 (つまり, 単純な SSS オメガ網, ただしサイズは 16 プロセッサ用) である. 図 5 に示すように, SSS-TBSF チップとメモリモジュールを搭載している PCB ボードに 3 枚の PCB ボードが繋がっており, その 3 枚のボードにはそれぞれ 4 プロセッサ搭載している.

ここでは, 次に示すアプリケーションプログラムを用いて SNAIL および SSS 型 MIN の性能評価, メッセージコンバインの効果などの評価を行なう.

- **Four coloring** : Hopfield モデルのニューラルネットワークを用いてアメリカ合衆国 48 州の 4 色問題を解くプログラム.
- **N-Queen** : 同様のニューラルネットワークを用いて N-Queen 問題を解くプログラム. 問題のサイズは 32×32.
- **Jacobi** : ヤコビ法により 200 元の三重対角行列を係数に持つ線形一次方程式を解くプログラム.

これらの並列アプリケーションはローカルメモリに関して最適化がなされている. つまり, ローカルで使用されるインストラクションやデータはローカルメモリ上に置かれる. プログラムは C 言語で各プロセッサの動作を直接記述し, OS のオーバヘッドはない. しかし, 後に述べるようにぎりぎりの最適化を行なったコードではない. 又, MIN を最大限に混雑させる為にこれらのアプリケーションでの全ての同期には Test&Set を用いたバリア同期を使用している.

4.1　SSS 型 MIN のレイテンシ

MIN 内での衝突がない場合の共有メモリへのアクセス時間を表 2に示す. ローカルメモリは 80nsec でアクセス可能なので, 共有メモリへのアクセスはローカルの約 4 倍かかる事になる[2].

表 2: メモリアクセス時間

Frame period	320nsec
32bit data read/synchronization	360nsec
1line block transfer	800nsec

図 7は 1 プロセッサにおける実行時間を基準とした時の並列プログラム処理のスピードアップ率である. 並列実行でのオーバヘッドを示す為に, 逐次実行での処理結果を合わせて示してある. 逐次実行の場合, 全ての変数やインストラクションをローカルメモリに置き, 同期操作を行なわない.

この図によると, 1 プロセッサでは逐次処理は並列処理に比べ 40%から 50%程度高速であることが分かる. 1 プロセッサでは同期によるオーバヘッドはないため, 実行時間が増えた分は SSS 型 MIN のレイテンシによるものである. 共有メモリへのアクセス時間がローカルメモリの約 4 倍であることを考えると, この程度のオーバヘッドは妥当であるといえる.

この図によると, 処理速度がプロセッサ数に比例して上がっている事が分かる. ただし **N-Queen** では並列性の限界から, 他の二つに比べその効果は薄い. **Four coloring** では, ”Super Linear”な台数効果が得られている. これは並列化により, 逐次処理の際の無駄な検索を省くことができるためである. **Jacobi** では, 共有メモリへのアクセスが頻繁であるため, 比較的並列化のオーバーヘッドが大きい. しかし, 12 プロセッサでの並列実行における処理時間は 1 プロセッサ逐次実行の約 8 倍である. この結果からは, 共有メモリアクセスによるレイテンシは, SNAIL のパフォーマンスを著しく低下させてはいないことがわかる.

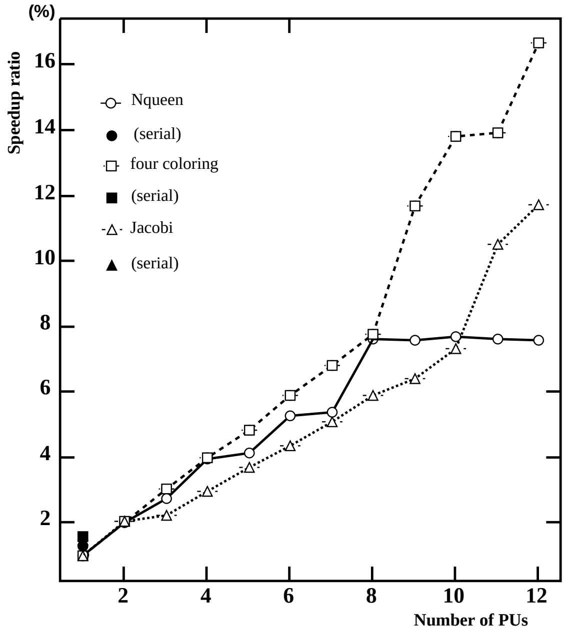

図 7: スピードアップ率

さらに, レイテンシはプリフェッチ操作を用いて隠蔽させることができる. **Jacobi** では, 共有メモリ上に置かれているベクタを読み, これにローカルメモリ上の行列要素を繰り返し掛け合わせている. この場合, プリフェッチを用いて”i”番目のリクエスト実行の前に”$i+1$”番目の読み出しをリクエストする事が可能となる. 図 8ではこのプリフェッチによる効果を示している. これによると, 5%から 10%の性能向上が見られる.

4.2　SSS 型 MIN の混雑

上記のアプリケーションにおける SSS 型 MIN の混雑の程度をパフォーマンスモニタを用いて測定した. 図 9では, プロセッサが送出した全パケットの衝突の割合を示している. 各スイッチにおいて, 上段から入ってきたパケットは下段の入力からのパケットよりも高い優先度を持っている. このため, この割合はプロセッサが接続されている位置に左右される. 図 9に示されている結果は, ほぼ中間の優先度を持っているプロセッサのものである.

N-Queen では, この値はそれほど大きいものではない (5%–25%) が, **Jacobi** では共有メモリ上の同じベクタを頻繁にアクセスするため, どのプロセッサでも 30%以上となっている. この割合はプロセッサ数が増えるにつれ **Jacobi** では約 1%, **N-Queen** では 4%づつ上昇している. SSS 型 MIN では衝突パケットは自動的に次のフレームで再送されるため, プロセッサから見るとアクセス時間が長くなる. この結果から, パケットの衝突により, 共有メモリアクセスのレイテンシは平均して 10%–30%引き延ばされることとなる. しかし, 図 7に示した 3 つアプリケーションでは, このレイテンシの延びによる SNAIL の性能低下は深刻ではない. 今回は 4 メモリモジュールでのシステムであるので, 16 メモリモジュール全てを用いてアクセスを分散させれば, この値は下がる事が予測される.

図 10は, プロセッサ台数対フレーム利用率を示している. フレーム利用率とは, あるフレームでプロセッサがパケット転送を行なう割合をさす.

この図によると, フレーム利用率は全アプリケーションで

---

[2] SNAIL のローカルメモリは FPGA の内部遅延の問題で十分な速度が得られていない. これについては現在改良中である.

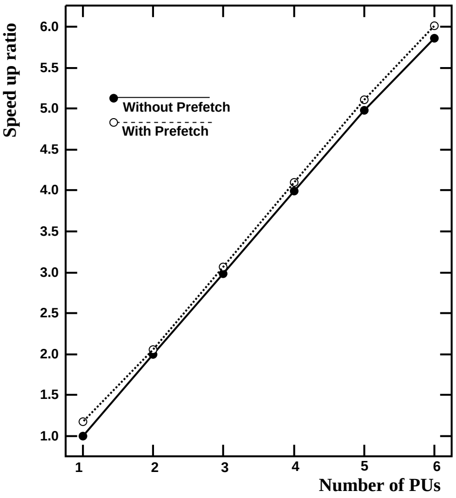

図 8: プリフェッチによるスピードアップ率 (**Jacobi**)

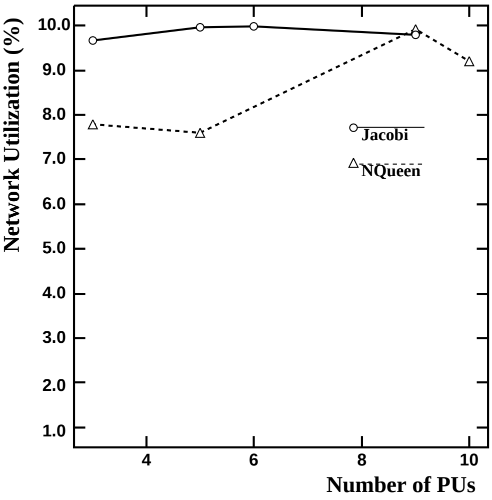

図 10: フレーム利用率

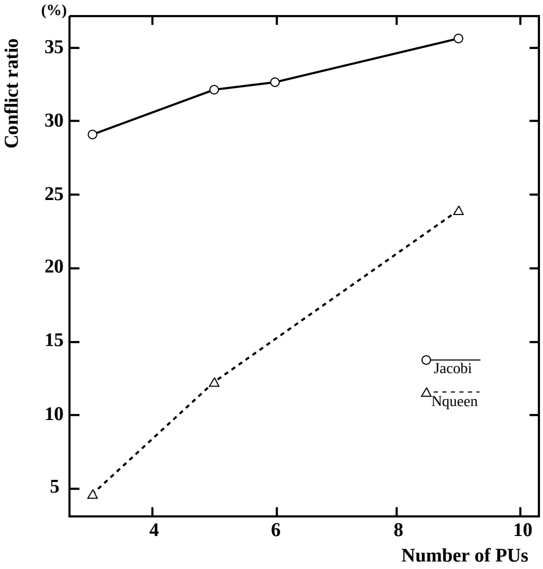

図 9: パケット衝突率

10%程度であることが分かる．この割合は共有メモリアクセスの頻度によるが, 衝突パケットの再転送によっても上昇する. 図 10では, プロセッサ数が増加するにつれて利用率が約 0.5%づつ上昇していることを示している．これは, 図 9に示されている衝突パケットの増加が原因であるが, ネットワーク利用率への影響は, これらのアプリケーションではそれほど大きくない. このことから, これらのアプリケーションでは, TBSF1 段 4 メモリモジュールの構成でも十分なバンド幅が得られることがわかる. ただし, 後に述べるように, それぞれのアプリケーションでぎりぎりの最適化を行なった場合, フレーム利用率, 混雑はかなり増大する可能性がある.

4.3　メッセージコンバインの効果

従来型の MIN における Hot Spot による性能の低下やメッセージコンバインの効果は様々な論文で、解析、評価が行なわれている [5] [6][13]. SSS 型 MIN は衝突によってパケットの再送が行なわれる点で、[15] らによる Circuit Switching 転送の Dropped のケースに似ている。このモデルに基づき、確率モデルとシミュレーションによって SSS 型 MIN における Hot Spot とコンバインの解析を行なった結果、パケットの再送が Tree Saturation によるネットワークの麻痺状態を防ぐため、SSS 型 MIN は、従来型 MIN に比べ、Hot Spot に対しても破滅的な性能低下は起きないことがわかった [16]。さらに、コンバインの効果を評価したところ、16 プロセッサでフレーム使用率が 35%であれば、アクセスのうち 5%が同一の Hot Spot に対して発生された場合は、コンバインによって約 5%の性能向上が予想されている [16]。

さて、実際のアプリケーションプログラムで、メッセージコンバインの効果を調べた所, 全アプリケーションで 12 プロセッサシステムにおいてでさえ実行時間改善には至らなかった. SNAIL のメッセージコンバインが効果を発揮することが期待されるのは, データの読み出し時と同期操作である. それぞれについて, メッセージコンバインの効果がどの程度になるか, さらに評価し, 検討した.

4.3.1　読み出しのメッセージコンバイン

先に用いた 3 つのアプリケーションを含め, 多くの科学技術計算では, 共有メモリ上のデータを多数のプロセッサに対しマルチキャストあるいはブロードキャストする必要があり, 少なくともこの場合は, メッセージコンバインは有効に働くはずである. それにもかかわらず, ここでの評価に効果が現れていないのは, 現状で, 図 7, 図 9に示すように, パケット衝突自体が性能の向上をさほど阻害していないためである. また、これらのアプリケーションにおいて同一番地に対して全体のアクセスの 5%におよぶアクセスが集中するような事態は起きておらず、ブロードキャストされるデータが存在したとしても、その番地は分散されており、理論解析の結果とは単純に比較することはできない。

ここで評価に用いたアプリケーションプログラムはある程度複雑なデータ構造を持ち, プロセッサ数に依存せずどんなサイズの問題も解けるような工夫が施されているため, ある意味では無駄な部分が多い. そこで, 以下のようなより単純な 3 つの問題について, サイズを固定とみなして, 不要な操作を削る, できる限りレジスタを利用する, ポストインクリメント操作など 68040 の命令セットを有効に利用する, 等のアセンブラを考慮した最適化を施してメッセージコンバインの効果を調べた.

- 行列, ベクトル積 (MATMUL): 共有メモリ上の単精度浮動小数点のベクトルに対し, それぞれのプロセッサがローカルメモリに持つ行列の行の積和を取る. 行列のサイズはプロセッサ数と等しくした.

- ブーリアンマトリックスの積和 (BMATMUL): 同上の方法で, ブール行列の積和を取る. 演算時間が短いため, さ

らに共有メモリのアクセスが頻繁である.

- データのブロードキャスト (BC): 共有メモリ上の 4Kbyte のデータをすべてのプロセッサがローカルメモリ上に転送する. データやプログラムの初期値ロードに頻繁に用いる操作である.

これらの操作におけるメッセージコンバインによる性能向上を図 11に示す. この図をみると, BMATMUL が最も効果が高く 20-40%程度にのぼり, これに MATMUL, BC が続く. これは, BMATMUL が演算およびローカルメモリのアクセスが最も小さいためで, 予想された通りの結果である. しかし, このプログラムでレジスタ宣言をはずして実行したところ, 効果は 1%程度に下がってしまったことから, ローカルメモリのアクセスがかなり性能に効いていることがわかる. これは, 現在の SNAIL のローカルメモリがさほど高速でなく, アクセス速度比が共有メモリの 1/4 に過ぎないためである. もっとも効果がありそうな BC の性能向上がさほどではないのも同様な理由によっている.BC の場合, さらにアクセスがモジュールに対してすみやかに分散される点も性能向上が小さい原因となっている.

以上の評価結果から, 現在の SNAIL の CPU 性能と SSS 型 MIN の性能では, プログラムをぎりぎりに最適化した段階で, はじめて読み出しコンバインの効果が現れることがわかった. より性能の高い CPU を用いた場合, あるいはプロセッサ数の多い場合は, さほど最適化しない状態でも効果が現れることが予想される.

4.3.2 同期操作におけるメッセージコンバイン

同期の頻度を上げるため, **Jacobi** での行列サイズをプロセッサ数と等しく取り (例えば, 12 プロセッサでは 12 × 12), 実行時間を測定した場合でも, 性能向上は僅かに 1%程度であった. これは以下のように考えられる.

SNAIL における同期操作のメッセージコンバインは Test&Set の場合のみ有効で, Fetch&Dec のメッセージコンバインは行なわない.Test&Set のコンバインは, プロセッサが Busy wait を行なう Barrier 同期では, その効果はほとんど期待できない. 詳細に説明すると以下の通りである. Test&Set で作る典型的な Barrier 同期は以下のようなステップで実行される. (1) マスタまたは早く処理を終了したプロセッサが計数セマフォア領域 C にプロセッサ数 (プロセス数)n を書き込む.(2) 他のプロセッサは同期領域 X に対し Test&Set を行なう.0 が取れたら (3) へ, 取れなければ Test&Set を繰り返す (Busy wait1).(3) セマフォア領域 C をデクリメントし, 同期領域 X に 0 を書き込み, C が 0 になるまで待つ (Busy wait2).

ここで, SNAIL においては Test&Set のメッセージコンバインが Busy wait1 で, 読み出しのメッセージコンバインが Busy wait2 で有効になる. しかし, Busy wait1 で 0 がとれなかった場合, いずれにせよ再び Test&Set を行なわなくてはならず, 衝突を起こしたことによってパケットが再送されるのと全く変わらない. そればかりか, 場合によってはコンバインせずにハードウェアによって直後のフレームで再送する方が早めに 0 が取れたことを知ることができる可能性すらある. このように Test&Set のコンバインは, 0 がとれなかったことで別の操作を行なうようにプログラムされていなければ, 効果が期待できない. 現在の SNAIL の場合, Busy wait2 の読み出し時のコンバインによって, 同期領域が 0 になったことを早く知ることができるため, これが多少の性能改善に役立っていると考えられる. しかし, 今までのアプリケーションでは, 同期操作自体さほど頻繁でないため, 効果は限定されている.

では, Fetch&Dec のコンバインを実装したらどうか, という考え方がある. しかし, Fetch&Dec を用いると Busy wait1 自体が不要となり, 基本的にプロセッサ数分のアクセスで計数セマフォアのカウントダウンが終了する (プロセス単位の計数セマフォアをそのまま共有メモリに置く場合この限りでない.). このため, 同期に達する実行時間が完全に揃っていない限りは Fetch&Dec のコンバイン自体が起きる可能性が低い. この場合でも Busy wait2 は必要なので, 結局 Hot Spot はこちらのアクセスが中心となる.SSS 型といえども Fetch&Dec のコンバインには相当な付加ハードウェアが必要であり, コストに見合う性能向上は困難であると考えられる. すなわち, プロセッサに対する Barrier 同期には, メッセージコンバインはさほど有効に働くとは考えられず, むしろ専用の大域的な同期機構 ([17] など) を設けた方が有利である可能性が高い.

結論として現在の SNAIL では, コンバインにより同期操作の速度を改善するよりも, 専用のフレーム単位の同期機構を設けて, これにより基本的な場所でのみ同期を行ない, あとはスケジューリングにより同期をはずしてしまう方向が有効と考えられる. この場合, 読み出しコンバインのブロードキャスト能力をより有効に利用することができる. 現在この方法による性能向上をテスト中である.

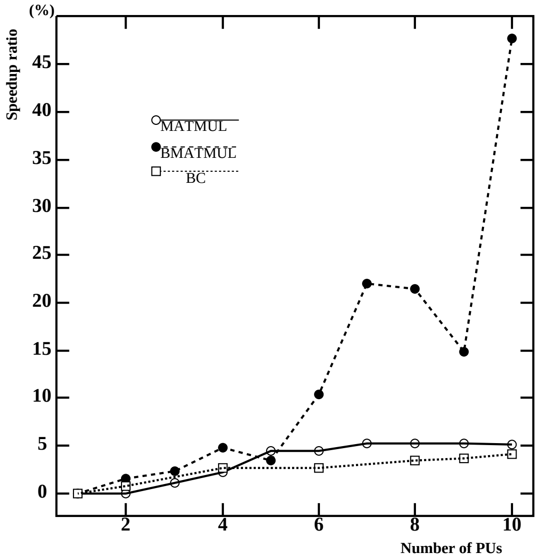

図 11: メッセージコンバインによる性能向上

この最も極端な例として, 全てのインストラクション及びデータを共有メモリ上に置き, 実行時間を測定した例を図 12 に示す. メッセージコンバインなしでは, インストラクションフェッチの際の衝突のため, プロセッサ数が増えるに従って実行時間が増加している. 一方, メッセージコンバインありでは実行時間はほぼ一定である. この場合, 共有メモリ上の同じインストラクションをフェッチするためにメッセージコンバインが有効に機能し, ほとんど SIMD 動作をしている事になる.

5 結論

SSS-MIN を用いたマルチプロセッサプロトタイプ SNAIL は, 現在 12 プロセッサ 4 メモリモジュールで稼働中である. SNAIL では, 16 プロセッサからのパケットが 1μCMOS SSS-オメガゲートアレイによって 50MHz のクロックでスイッチングされる. 又, わずかに 20%のハードウェアを付加するだけでメッセージコンバインの実装を可能にしている.

数種のアプリケーションを用いて行ったこれまでの評価の結果, SSS-MIN におけるレイテンシと同期のオーバヘッドはそれほど大きくなく, そのバンド幅は十分である事が分かった. ビットシリアルなメッセージコンバインによる性能向上については, 最適化をぎりぎりに行なった場合, 読み出しにおけるコンバインが有効になることがわかった.

現在, 最大サイズ (16 プロセッサ 16 メモリモジュール) での稼働, ローカルメモリの高速化, フレーム単位のバリア同期機

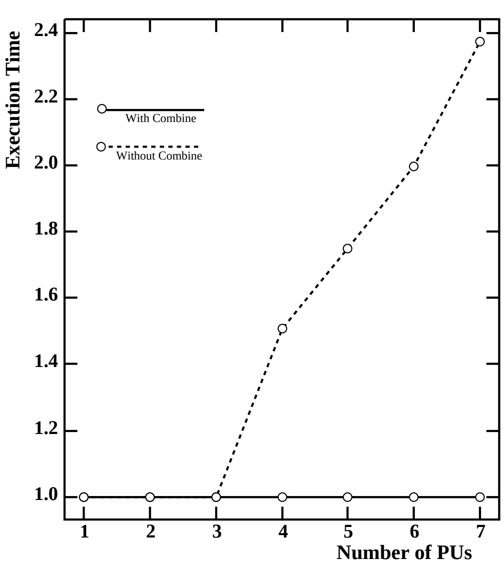

図 12: 全コードを共有メモリ上に置いた際の実行時間

構に関してデバッグ中である. また現在のアプリケーションの最適化と, SPLASH ベンチマークを実装中である. さらに今後, スケジューリングによる同期操作の除去, この場合のコンバインの有効な利用法についても検討を加える予定である.

謝辞:

SSS 型 MIN チップは川崎製鉄株式会社の御協力により開発されました。深く感謝致します。又、実装に当たりお世話になりました周洛氏、安川英樹氏に深く感謝致します。更に貴重な御助言を頂きました東京工科大学の工藤知宏専任講師に深く感謝致します。

参考文献

[1] Phister, G.F. et al.: *The IBM Research Parallel Processor Prototype (RP3): Introduction and architecture*, Proc. of 1985 Int. Conf. Parallel Processing, (1985).

[2] J.Konicek, et al.: *The Organization of the Cedar System*, in Proc. 1991 Int. Conf. Parallel Processing, Aug. 1991.”

[3] H.Amano, L.Zhou, K.Gaye: *SSS(Simple Serial Synchronized)-MIN: a novel multi stage interconnection architecture for multiprocessors*, Proc. of the IFIP 12th World Computer Congress, Vol.I, pp.571-577, Sept. 1992.

[4] 天野, 周, 藤川: *SSS (Simple Serial Synchronized)* 型マルチステージネットワーク, 情報処理学会論文誌 Vol.34, No.5, pp. 1134-1143, (1993).

[5] A.Gottlieb, R. Grishman, C. P. Kruskal, K.P.Mcauliffe, L.Rudolf, and M.Snir: *The NYU Ultracomputer - Designing an MIMD Shared Memory Parallel Computer*, IEEE Trans.on Comput. vol. c-32, No.2, Feb.1983.

[6] Phister, G.F. and Norton, V.A.: *Hot Spot Contention and Combining in Multistage Interconnection Networks*, IEEE Trans. on Comput. vol. c-34, No.10, (1985).

[7] 坂元, 荒川, 正木, 井上, 天野: 自己ルーチングスイッチの構成とその評価, 信学技報 ISSE88-30 8, (1988).

[8] F.A.Tobagi and T.Kwok: *The Tandem Banyan Switching Fabric: a Simple High-Performance Fast Packet Switch*, Proc. INFOCOM91, Apr. 1991.

[9] 天野 英晴, 藤川 義文: マルチステージネットワーク *PBSF*, 情報処理学会計算機アーキテクチャ研究報告 No.94-5, (1992)

[10] F.Chiussi, H.Amano, F.A.Tobagi: *A 0.8μm BiCMOS Sea-Of-Gatesimplementation of the Tandem Banyan Fast Packet Switch*, Proc. of IEEE Custom Integrated Circuits Conference May, 1991.

[11] H.Kasahara, H.Honda, S.Narita: *Parallel Processing of near fine grain tasks using static scheduling on OSCAR*, Proc. IEEE SuperComputing'90, pp.856-864 1990.

[12] D.H.Lawrie: *Access and Alignment of Data in an Array Processor*, IEEE Trans. on Comput. vol. c-24, No.12, Dec. 1975.

[13] Lee, G.H. Kruskal, C.P. and Kuck, D.J.: “The Effectiveness of Combining in Shared Memory Parallel Computers in the Presence of 'Hot Spots' ”, Proc. International Conference on Parallel Processing, pp.35-41, (1986).

[14] Dickey, S.R. and Percus, O.E.: “Performance differences among combining switch architectures”, Proc. International Conference on Parallel Processing, pp.I110-117, (1992).

[15] Wu, C.L. Lee, M.: “Performance Analysis of Multistage Interconnection Network Configurations and Operations, ” IEEE. Trans. Comput. **41**, No.1 pp.18-27, (1992).

[16] K.Gaye, 塙 敏博, 天野 英晴: “SSS 型 MIN における hot spot の影響とメッセージ結合の効果,” 電子情報通信学会論文誌 D-I, (1994 年 5 月号掲載予定).

[17] 松本 尚: “Elastic Barrier:一般化されたバリア型同期機構,” 情報処理学会情報誌, Vol.32, No.7, pp.886-896 (1991).