静的スケジュール可能な階層型相互結合網の研究

森村 知弘

平成 16 年度

論文要旨

マルチプロセッサ構成の並列計算機において，ピーク性能はプロセッサ数を増やすことによって比較的容易に向上させることが可能であるが，実際にプログラムを動かした場合の実効性能を向上させることは難しい．実効性能をスケーラブルに向上させるためには，大規模化について配慮すると共に並列化コンパイラなどによる最適化技法を適用しやすい構成を取る必要がある．並列計算機の重要な構成要素の一つである相互結合網も，スケーラブルでコンパイラによるスケジュールがしやすい構成が要求される．

本論文では，スケーラブルかつスケジュールしやすい相互結合網，R-Clos を提案し，その基本特性と転送性能について評価する．R-Clos は 3 段の多段結合網 (MIN: Multi-stage Interconnection Network) である Clos 網を再帰的な階層構造により複数個接続した多段結合網である．階層構造の導入により，256 プロセッサを 4 プロセッサ/クラスタで 3 階層の R-Clos で接続した場合，同じサイズの Clos 網の 34%の交点数で実現可能である．また，従来型の再帰的な Clos 網である recursive 5-stage Clos 網と比較しても 57%の交点数で実現可能であり，大規模システムに容易に対応可能であることがわかる．また，確率モデルに基づく転送シミュレーションより，全アクセスに占める近傍アクセスの割合が 80%になると，基本ネットワークの Clos 網の性質が現れ，従来型の recursive 5-stage Clos 網よりも高いスループットと低い転送遅延が得られることが明らかになった．

また，予めコンパイル時にアクセスする宛先とタイミングが解析可能である場合に，提案した相互結合網 R-Clos において，コンパイラがパケット転送の送信時刻と経路を制御することにより，無衝突にパケットを宛先に配送するパケット転送スケジュール法を提案する．評価のために，ランダムに生成した宛先のアクセスパターンに対して提案したスケジュール手法を施し，全データ転送が完了する時刻をシミュレーションで求めた．その結果，出線競合による衝突の起こらないクロスバスイッチで転送した場合の遅延と比較して，オーバヘッドは 0.5%以内となった．

一方で，プログラム中には静的にアクセス先と時刻が定まるブロックと定まらないブロックが存在するため，提案したスケジュール法を用いても，スケジュール不能なアクセスによってスケジュールされたアクセスが乱される．そこで本論文では，スケジュール可能なアクセスとそうでないアクセスが混在しても，スケジュールを乱すことなく効率良く転

送を行えるスイッチアーキテクチャである MGF スイッチを提案する．これをゲートアレイ上に実装したところ一般的な単一のクロスバスイッチで構成されるスイッチに対して約 1.4 倍の 10 万ゲートで実装可能で，最大動作周波数は約 50MHz である．また，このチップの設計データに基づいてシミュレータを作成し，転送シミュレーションを行った結果，スケジュールされたパケットは衝突なく転送可能であることが確かめられ，スケジュールされていないパケットについても，単一のクロスバスイッチによる転送遅延の最大で約 60%に抑えられた．

最後に，提案した結合方式 (R-Clos) とパケット転送スケジュール方式と MGF スイッチアーキテクチャを組み合わせた総合的な評価を行った．ランダムな宛先に対するアクセスパターンを生成し，これに対して提案したスケジュール手法を施し，MGF スイッチで構成される R-Clos において評価した結果，コンパイラの静的スケジュールを乱さずに，高負荷においても安定したスループットを維持してパケットが転送されることを明らかにした．

Abstract

In order to maximize effective performance of large scale multiprocessors, techniques on static scheduling with sophisticated compilers have been developed. An interconnection network, that is a critical component of multiprocessors, is also a target of static scheduling.

In this paper, a hierarchical multi-stage interconnection network (MIN), R-Clos, which is both highly schedulable and scalable, is proposed. R-Clos consists of hierarchically extended Clos networks connected with extra intermediate stages. Frequent local communication between neighboring processing elements are supported with high bandwidth rearrangeable Clos networks used in a basis of R-Clos. A large-scale system can be built owing to hierarchical structure with less hardware than that of other flat MINs.

The transfer performance of R-Clos is analyzed by probabilistic simulations. The simulation results show that R-Clos achieves reasonable performance under local communication dominant traffic.

Secondary, scheduling schemes of packets transfer on R-Clos are proposed. Using the schemes at compile time, the parallelizing compiler and scheduler can arrange the packet's routes, and schedule the access time to avoid any packet collisions.

Additionally, in order to keep the static scheduling under partially scheduled traffic, which includes certain unscheduled packets, a switch architecture called MGF switch is proposed. MGF switch has two sets of the transfer channel for scheduled packets and non-scheduled packets respectively. We fabricated the MGF switch on 0.35 μm gate array. The amount of hardware of the switch chip is about one hundred thousands gates and the maximum frequency of the switch chip is about 50 MHz.

Through the instruction(clock) level simulation, the scheduling scheme performance on MGF switch is analyzed. The result of the evaluation by the simulation shows the availability of proposed network systems, the combination of R-Clos, the scheduling scheme and MGF switch architecture.

目 次

図目次

表目次

第1章　緒 論

1.1　本研究の背景と目的

近年, マルチメディア処理の普及や, インターネット接続の高速化に伴い, より多くの計算能力が必要となってきた. この要求に応えるため, 単一のマイクロプロセッサの高速化技術は飛躍的に向上する一方, SMP(Symmetric Multi Processor) などのマルチプロセッサ構成の並列計算機も普及してきた.

前者の高速なマイクロプロセッサとしては，VLIW(Very Long Instruction Word) やスーパスカラプロセッサがあげられ，どちらもプログラム中の命令レベルの並列性 (ILP: Instruction Level Parallelism) を利用することにより, チップ内の演算器を複数並列使用して，スループットを向上させることによって高速化を果たしている．このため, 比較的小規模な構成をとる場合に有効と考えられている．これに対し, 後者のマルチプロセッサ構成の並列計算機は, 古くから研究[GRM+68] [SRe77] され, ILP よりも大きい粒度の TLP(Thread Level Parallelism) を利用する．科学技術計算などでは，ループレベルの並列化によって TLP を効率良く利用し，大規模なシステムで大きな性能向上を実現することが可能である．しかし，このためには，プログラマが問題から並列性を抽出し，ターゲットのマシンにあわせた並列化を行う必要があった．これはユーザの大きな負担となり, 大規模マルチプロセッサの普及の問題点の 1 つになっていた.

このような並列化をユーザに頼らず，自動的に行う並列化コンパイラの研究[CMM+89, MJS+96, 岡本 96, HNS+99]は，近年急速に進みつつあり，これをマルチプロセッサ上で実行する技術が実用化され，以前よりも身近に利用されるようになってきた．しかし，このような並列化コンパイラの性能を最大限に引き出すためには，マルチプロセッサ自体の構造が本質的にコンパイラによる静的なスケジューリングに向いている必要がある．すなわち，マルチプロセッサの実効性能とピーク性能の差を埋めるためには，プログラム中からできる限りの並列性を抽出する並列化ソフトウェア技法と，これを効率良く実行できるハードウェアアーキテクチャが必要である.

相互結合網はマルチプロセッサのハードウェアアーキテクチャの重要な構成要素の 1 つである．大規模マルチプロセッサの相互結合網についての研究は 80 年代から活発に行なわれており，その変遷も激しい．初期に用いられたハイパーキューブなどの次元数の高い直接網は，2 次元，3 次元メッシュやトーラスなどの次元数の低い直接網に取って代わられ，最近は，より高いバンド幅を持つ Fat-tree などの間接網が広く用いられるようになっている．しかし，静的スケジュールに適したプロセッサ自体やメモリシステムの構造が比較的よく検討されているのに対して，重要な構成要素の 1 つである相互結合網は，その結合方式などについては検討されてきたが，コンパイラによるスケジューリングの対象として扱われることが少なかった．コンパイラによるスケジューリングを考慮に入れない相互

結合網を利用することにより，プロセッサ間交信に要する時間と，混雑による遅れが，スケジューリングを大幅に乱し，性能が大幅に低下する可能性がある．すなわち，並列化コンパイラの並列化技術の効果を得るためには，結合網をスケジューリング対象として扱い，解析可能な交信をスケジュールすることによりネットワーク上の混雑を低減させる必要がある．交信をスケジュールするアプローチは，大きく分けて 2 つあり，1 つはプログラム実行中のアクセス発行時におけるルータあるいはスイッチなどのハードウェアの動的な設定変更による経路調整やアクセスタイミング制御による動的スケジュールがあり，もう 1 つはコンパイラがプログラムコンパイル時に，データ交信について解析を行い, 解析可能な交信に関して, 予め経路やアクセスタイミングをスケジュールする静的スケジュールがある．

本論文では，並列化コンパイラによる静的スケジュールに適し，かつ大規模化が容易な間接網 R-Clos を提案し，コンパイラがコンパイル時に解析可能な交信に施す静的スケジュール手法の詳細と，実現のためのハードウェアを検討する．

まず, 将来の並列化コンパイラの技法として, マルチグレイン並列化手法[笠原 91, 岡本 96]に着目した．従来の自動並列化コンパイラなどの並列化ソフトウェアが主に行っているループのイタレーション間の並列性を利用してループを分割するループ並列化は, 複雑なデータ依存や制御依存が存在する一部のループには適用することが難しい．マルチグレイン並列化手法は，これに加えて, より大きな並列性，すなわち，サブルーチンブロック (SB) やループのリピテーションブロック (RB) や基本ブロック (BB) 間の並列性を抽出して並列処理する粗粒度並列処理 (マクロデータフロー)，およびこれらブロック内のステートメント間に存在するより小さな並列性を利用した近細粒度並列処理をも行うことにより，高い実効性能を実現することができる．

R-Clos はこのマルチグレイン並列化手法に適合するように設計されており，強力な転送能力を持ったクラスタが比較的疎に接続された階層的な構成を持つ．階層内は複数のパスを持つ一方，階層間をただ 1 つのパスに限ることで，データ転送の静的スケジューリングを容易に行うことが可能である．

R-Clos は，慶應義塾大学が STARC との共同研究により開発しているマルチグレイン並列化コンパイラ向きマルチプロセッサシステム ASCA 用の接続網として検討されたが，コンパイラによるスケジュールを行うマルチプロセッサ一般に利用可能である．MGF スイッチは，コンパイラが静的解析できないデータアクセスによる予測不能なトラヒックと，静的スケジュールされたデータアクセスによるトラヒックを分離して転送し，静的スケジュールが乱さないことを目的に提案し，LSI チップに実装した．

1.2 本論文の構成

本論文の構成は，第 2 章で，本研究の背景となるマルチグレイン並列処理とプロセッサ ASCA の概要について示し，第 3 章でマルチプロセッサ ASCA 用の相互結合網として提案する R-Clos を示す．第 4 章で R-Clos の基本特性とその転送性能を確率モデルシミュレーションによって評価する. 第 5 章では, R-Clos 上で静的解析可能なデータ配送に関してコンパイラがこれらをスケジュールする交信スケジュール方式を提案し，提案したスケジュール手法とノンブロッキングなクロスバスイッチで転送した場合を比較して評価する. 第 6

章では, スケジューリング不能なブロックを含む実トラヒック下においても，提案したスケジューリング方式をサポートするスイッチアーキテクチャMGF を提案・実装し, 提案したスケジューリング方式と MGF スイッチアーキテクチャを組み合わせた総合的な転送評価を行っている. 最後に第 7 章では，結論を述べる.

第2章　研究の背景

本章では本研究の背景として，マルチグレイン並列処理に適したハードウェア構成を特徴とするマルチプロセッサ ASCA を紹介する．

その後, マルチグレイン並列化処理手法について説明するとともに，そのデータ移動の特性について考察する．加えて，従来の自動並列化処理を考慮したマルチプロセッサにおける相互結合網を調査し，考察した結果に基づいて問題点を指摘する．

2.1　マルチプロセッサ ASCA

2.1.1　ASCA の概要

マルチプロセッサ ASCA は,

1. 自動並列化コンパイラの静的最適化技術
2. マルチグレイン並列化処理技法

の 2 つのソフトウェア技法の効果を最大限に得られるハードウェアアーキテクチャを持つ．

1．の自動並列化コンパイラによる静的最適化技術のために，ASCA は各ハードウェア構成要素が動作予測可能な設計としている．これに加えて ASCA では，従来のループ分割並列化では抽出不能であった粗粒度，近細粒度の並列性を合わせて利用する 2．のマルチグレイン並列化処理技法に合わせた構成を持つ．

ASCA は次の特徴的な要素を持つ:

- 動的最適化を排除した，静的動作予測可能なカスタムプロセッサ MAPLE
- 静的スケジューリングを妨げないための，ソフトウェア (コンパイラ) 制御キャッシュ機構
- 計算処理とオーバラップしてデータ転送を制御するデータ転送用コントローラ (DTC)
- データの転送遅延の予測が可能なプロセッサ間相互結合網
- 階層的に処理されるマルチグレイン並列処理の特性を考慮した階層構造のプロセッサクラスタリング

次節では，これらのハードウェアによって構成される ASCA の全体像を示す．

2.1.2 ASCA の全体構成

ASCA の全体構成を図 2.1 に示す. ASCA は, 4 または 8 個のプロセッシングエレメント (PE) から構成されるプロセッシングクラスタ (PC) と共有変数の格納に使用する集中共有メモリ (CSM: Centralized Shared Memory) を階層的な相互結合網で結合したマルチプロセッサシステムである. ASCA の PE は単純な構造で, ハードウェアコストもかからないため, 一番最下層のクラスタは 1 チップに集積化されることを想定している．ASCA に用いられる階層的な相互結合網は本研究の主題であるため，後にまとめて紹介することにし，その他の部分について簡単に述べる．

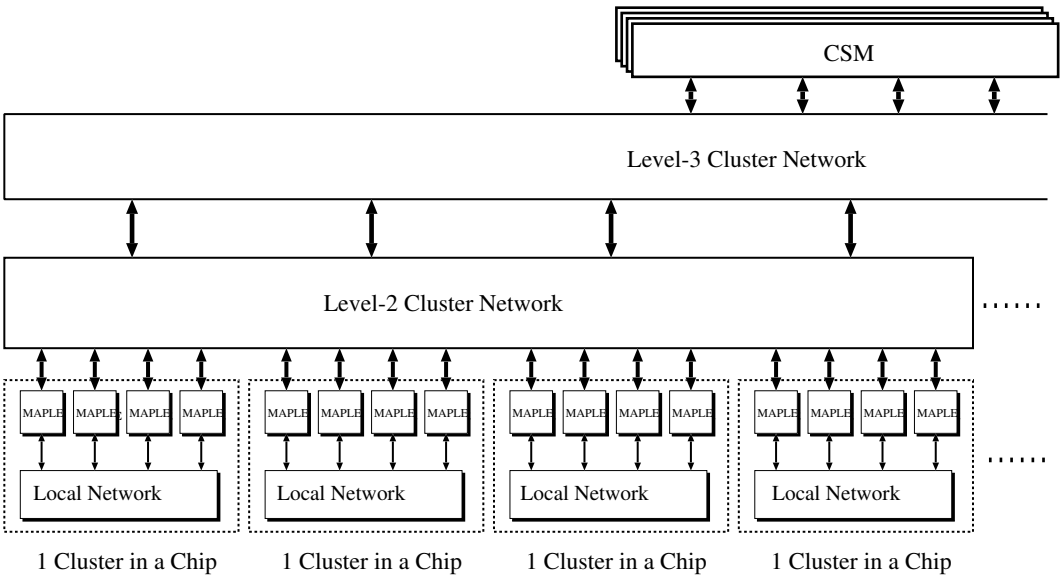


図 2.1: ASCA の構成

(1) プロセッシングエレメント MAPLE ASCA のプロセッシングエレメントである MAPLE は，2.2 で示したとおり，ローカルメモリ (LM: Local Memory)，分散共有メモリ (DSM: Distributed Shared Memory)，ネットワークインタフェース，およびプロセッサ MAPLE で構成される．

ローカルメモリは，通信の必要がないデータを置く．中粒度並列処理では，プロセッサ間で交信の必要がないデータセットは，可能な限りローカルメモリに配置される．

分散共有メモリは，外部 PE からの書き込みによるデータ通信のためのメモリで，他 PE からの読みだしはできない．データ転送がコンパイラによって静的に解析可能である場合，通信にはこの分散共有メモリに書き込むだけで行える．

プロセッサはさらにプロセッサコア (MAPLE Core)，データ転送コントローラ (DTC: Data Transfer Controller)，およびオンチップキャッシュメモリからなる．

1. MAPLE Core
   コンパイラへのハードウェアの visibility を高めるため，実行時不確定性の原因となる動的最適化技術 (dynamic scheduling，out of order issue, speculative execution) などを排除しつつも，パイプライン処理 (5 段) と強力な IEEE 754 準拠の浮動小数点演算

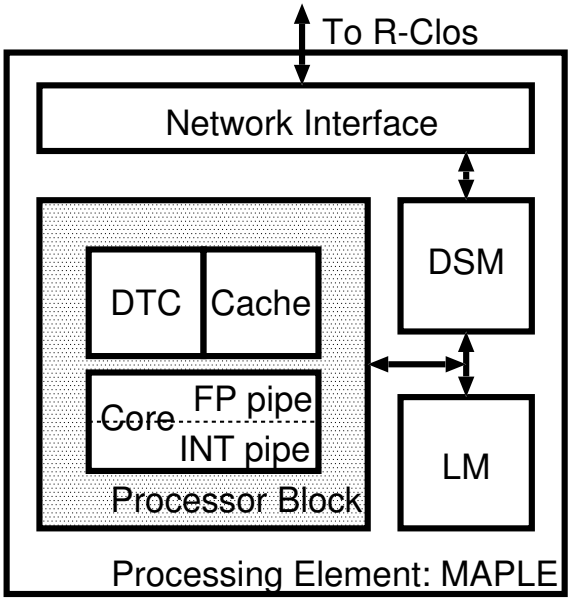


図 2.2: プロセッシングエレメント MAPLE の構成

器を備える. 単純なハードウェア構成とすることで，従来のプロセッサコアよりも少ない面積で実装することが可能で高集積化が見込めるとともに，低消費電力で動作可能である.

2. ソフトウェア制御キャッシュ機構
   キャッシュメモリは性能向上に大きく寄与しているが，キャッシュメモリのミスヒットは実行時不確定性の原因となる. そこで ASCA では，キャッシュメモリへのデータのロードおよびストアなどを可能な限りコンパイラが制御することによって予期せぬミスヒットを回避する.

3. データ転送コントローラ (DTC)
   プロセッサの行う計算処理を妨げることなく，大量のデータを効率良くキャッシュメモリに用意するためのコントローラ. コンパイラが生成するデータ転送指令コード (DTC コード) をコントローラが実行することによって, メインのプロセッサの MAPLE の処理とオーバラップしてデータ転送が実行される.

(2) メモリ階層 ASCA のメモリ階層は, 各 PE から平等にアクセスできる集中共有メモリ (CSM) と PE 内に配置された分散共有メモリ (DSM), ローカルメモリから構成される. CSM は粗粒度並列処理における共有変数の受渡しに使用される. DSM は, 近細粒度並列処理や中粒度並列処理において他の PE とのデータやフラグの受渡しの際に使用される．コンパイラが，データ参照の頻度や，PE 間でのデータ依存関係などを考慮して最適なメモリへデータを配置することにより，データアクセスの遅延を可能な限り低減させる狙いがある.

2.2　マルチグレイン並列処理におけるデータ移動の特性

ここでは，並列化ソフトウェア技法のうち最も効果的な並列化処理技報の１つであるマルチグレイン並列処理[笠原 91]をターゲットとして，マルチグレイン並列処理手法におけるデータ移動の特性について述べる．

2.2.1　マルチグレイン並列化処理技法の概要

マルチグレイン並列化処理技法では, 次の 3 つの粒度の並列性を抽出して, これらを組み合わせて最大限の並列性を抽出する. 3 つの粒度における並列化は，次のとおりである．

- 粗粒度並列化 : 関数ブロック (SB: Subroutine Block)，反復ブロック (RB: Repetition Block)，および基本ブロック (BB: Basic Block) などの固まり (これをマクロタスク (MT) と呼ぶ) 同士の並列性を利用して並列化する．
- 中粒度並列化 : ループ (RB) において繰返し間の並列性を利用して，各イタレーションを各 PE に割り当てて並列化する．
- 近細粒度並列化 : 基本ブロック (BB) 内のステートメント間の依存を調べ，ステートメント間の並列性を利用して，これを各 PE に割り当てて並列化する．

マルチグレイン並列化処理技法は，逐次的に記述されたプログラム全体を基本ブロック，ループ，サブルーチン等のマクロタスクに分割し，そのマクロタスクを実行時にプロセッサクラスタに割り当てる. プロセッサクラスタ内部では，可能ならば Do-all 処理や Do-across 処理，そうでなければ階層的マクロデータフロー処理，近細粒度並列処理を行いプログラムの並列性を抽出する. したがって，マルチグレイン並列化処理技法は，前述の 3 つの粒度の並列化処理を組み合わせて使用する．

2.2.2　粗粒度並列処理 (マクロデータフロー処理)

粗粒度並列化は，プログラムをマクロタスクに分割し，分割したマクロタスク間の依存関係について解析して，個々のマクロタスクがそれぞれ PE に割り当てられることにより，マクロタスク間の並列性を利用して並列処理を行う．ここでは，この粗粒度並列化を行う手順について説明する．

(1) マクロタスク　粗粒度並列処理においてプログラムはマクロタスク (MT) に分割される. MT はブロック途中への飛込みが無いステートメントの集合からなるブロックである. その形状から疑似代入ブロック (BPA)，繰返しブロック (RB)，サブルーチンブロック (SB) に分けられる．BPA は，1 つ又は複数の基本ブロック (BB) から構成される．これらのブロックは，プログラムの制御構造を元に分割される．ダイナミックスケジューリングやマクロタスク間のデータ転送オーバヘッド等を考えて，分割した MT の再融合による最適化や，MT 内にさらにマクロデータフロー処理を適用する階層的マクロデータフロー処理を行うことがある. 逐次プログラムを MT に分割した例を，図 2.3 に示す．

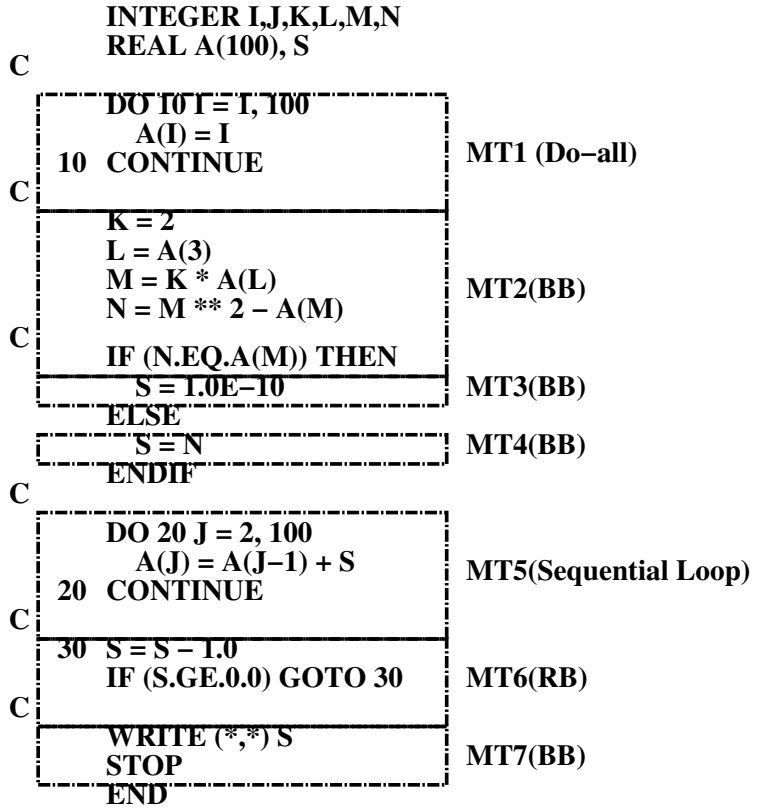


図 2.3: MT 分割の例

(2) マクロフローグラフ　MT に分割した後，MT 間のコントロールフロー，データフローを表すマクロ (データ) フローグラフ (MFG) を生成する. 図 2.3 のマクロフローグラフを図 2.4 に示す. MFG の各ノードは BPA，RB，SB 等の MT を示し，エッジには点線で示したコントロールフローと，実線で示したデータ依存関係がある. ノード内の小円は条件分岐を示す. 後方へのエッジは RB に含まれるため，MFG は閉路のない有効グラフ (DAG: Directed Acyclic Graph) となる.

(3) マクロタスクグラフ　さらに MFG から MT 間の最大並列性を表す MT の最早実行可能条件を求める. MT n の最早実行可能条件とは，MT n が最も早く実行可能になる条件である. MT の最早実行可能条件は，マクロタスクグラフ (MTG) という DAG で表すことができる. 図 2.4 から作ったマクロタスクグラフを図 2.5 に示す. MTG の各ノードは MT を示し，点線のエッジはコントロール依存，実線のエッジはデータ依存を示す. 各ノード内の小円は条件分岐を示し，それから出ている実線のエッジはコントロール依存とデータ依存の両方を表す. 複数のエッジを束ねている実線と点線の弧は，各々AND 条件と OR 条件を意味している. AND 条件で束ねられたエッジは，ある条件が成り立った時に同時に解決される依存関係を示す. 一方，OR 条件で束ねられたエッジは，条件分岐によってどちらかのエッジで示された依存関係が解決されるという意味である. 例えば，図 2.5 の MT2 は，MT3 か MT4 のどちらかに分岐するという意味である. ノードに入ってくるエッジにも同

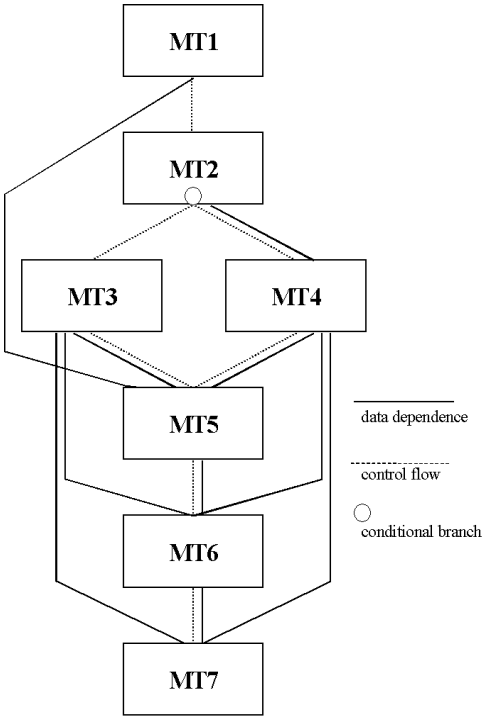


図 2.4: マクロフローグラフ (MFG)

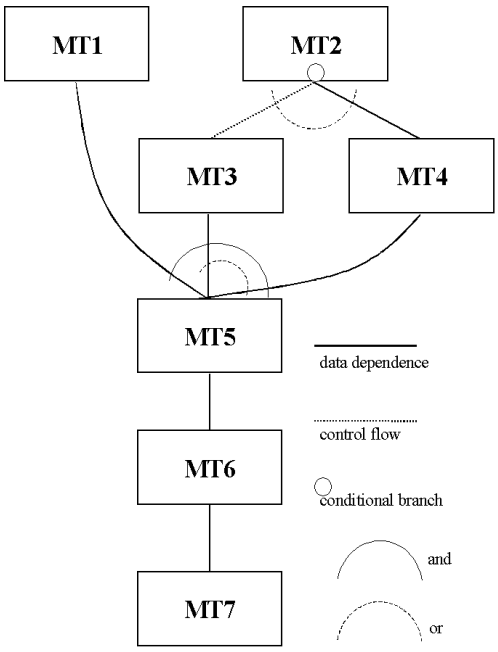


図 2.5: マクロタスクグラフ (MTG)

様な弧があり，AND 条件は束ねられたエッジで示されるすべての依存関係が解決された時にその MT が実行可能になるという意味である. 一方 OR 条件は，束ねられたエッジで示されるどれか 1 つの依存関係が解決された時にその MT が実行可能になるという意味である. 例えば，MT5 は MT1 が終了するという条件と，MT3 か MT4 のどちらかが終了するという条件の 2 つが成り立った時に実行可能となる.

(4) タスクのスケジューリング　マクロデータフロー処理では，条件分岐のような実行時まで確定できない問題に対処するため，MT はダイナミックにプロセッサクラスタ (PC) へスケジューリングされる. ダイナミックスケジューリングは粗粒度の MT に対して行われるためにスケジューリングオーバヘッドは小さく抑えられる. さらに本手法では OS のシステムコールによるダイナミックスケジューリングではなく，コンパイラがスケジューリングルーチンを含めたコードを生成することにより，ダイナミックスケジューリング自身のオーバヘッドも小さく抑える. MT を PC に先行割当することによって，ダイナミックスケジューリングのオーバヘッドを減らすことができる. これは，コンパイル時に計算した MT の予想実行時間により，各 PC に割り当てられた MT を実行し終わる時間を予想し，それが最も早い PC へ MT をあらかじめ割り当てることによって行われる.

(5)MT 間データ転送　MT 間のデータ転送は，基本的には集中共有メモリ (CSM) へのロードストアによって行なわれる (図 2.6). しかし，配列などの大量のデータを受渡すと，PE と CSM 間のネットワークを使用することによるオーバヘッドが無視できない. このオー

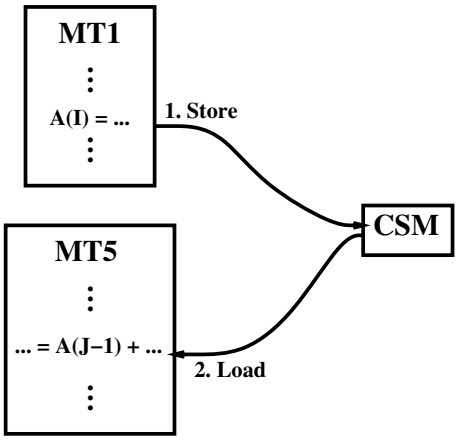


図 2.6: MT 間のデータ転送

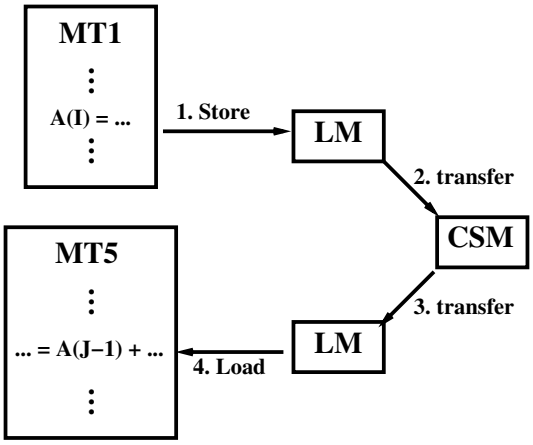


図 2.7: データローカライズ

バヘッドを減らすために，データローカライズ (図 2.7)，プレロード，ポストストア処理を用いる．

データローカライズは，MT がアクセスするデータを比較的アクセスのオーバヘッドが小さいローカルメモリ (LM) や分散共有メモリ (DSM) へ置くことにより，データ転送のオーバヘッドを減らす方法である．プレロードは，次に実行される MT がデータローカライズによって必要となったデータを，プロセッサとは独立にデータを転送可能なモジュールを用いて，前の MT 処理中に CSM から DSM へ転送しておく処理である．また，ポストストアとは MT が終了すると，直ちに次の MT を実行開始し，実行が終了した MT が生成したデータを，次の MT の処理と同時に DSM から CSM へ転送する処理である．

2.2.3 中粒度並列処理

MT は，PC にダイナミックにスケジューリングされる．PC に割り当てられた MT が Do-all ループを含む場合，この Do-all ループは PC 内の PE によってイタレーションレベルの粒度で並列化される．さらに Do-all 以外のループも，様々なリストラクチャリング手法を用いて，イタレーションレベルの粒度で並列化される．図 2.8 に 3PE を用いて並列化した場合の概念図を示す．

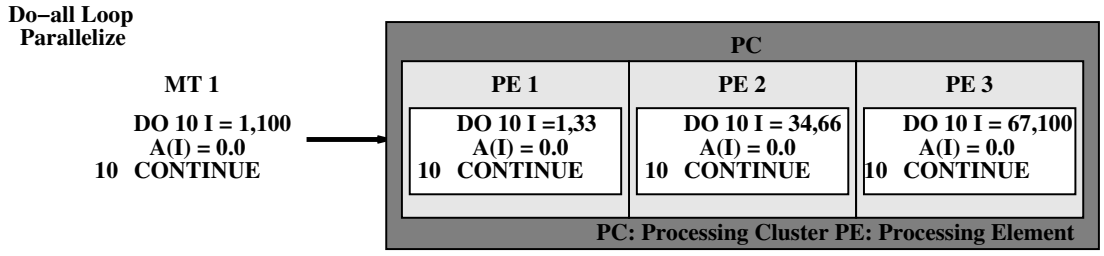


図 2.8: ループ並列化

Do-all の場合，誘導変数，ループ終了判定等を転送するのみで基本的にループ中の PE 間のデータ転送は起こらない. Do-across の場合，ループ中のデータ転送が起こるが，一度のデータ転送量は少なく，転送先は事前に決定される.

2.2.4　近細粒度並列処理

近細粒度並列処理とは，ステートメント単位での並列化を行う処理である. スーパスカラプロセッサが行っている命令レベルで並列処理を行う細粒度並列処理と区別するために, 本論文ではステートメント単位で並列処理を行う技法を特に近細粒度並列処理と呼ぶ.

MT が BPA やループ並列化できないループであった場合，MT 内部は近細粒度に並列化され，PC 内部のプロセッサエレメント (PE) によって並列処理される. 図 2.9 は，MT を 2PE で近細粒度並列処理を行った場合の例である.

近細粒度並列処理では，基本ブロックに対してのみ並列処理を行うことで，プログラムによる実行時不確定性はない．これにより，近細粒度タスクのプロセッサへの割り当てはコンパイル時に行うことができる．すなわち，静的スケジューリングが可能である．静的スケジューリングはコンパイル時にスケジューリングをすませてしまうもので，スケジューリングのオーバヘッドは皆無になる．さらに，マシンクロックレベルでの厳密なコードスケジューリングを行うことができればすべての同期を除去した無同期近細粒度並列処理が可能である. 図 2.10 左図は，図 2.9 で同期を行った場合の処理を表し，図 2.10 右図は，無同期で実行した場合の処理を表している. 無同期処理を行うと PE の無駄な待ち時間と同期フラグ操作の時間を取り除くことができ，その分 MT を高速に実行可能となる. しかし，これを行うためにはコンパイラがシステムの状況をコンパイル時に完全に把握する必要があり，そのためのアーキテクチャサポートは必須となる.

近細粒度並列処理では，ステートメント間の単純変数の受渡しが行われるため, 一度のデータ転送量は少ない. また，転送先はコンパイル時に決定される.

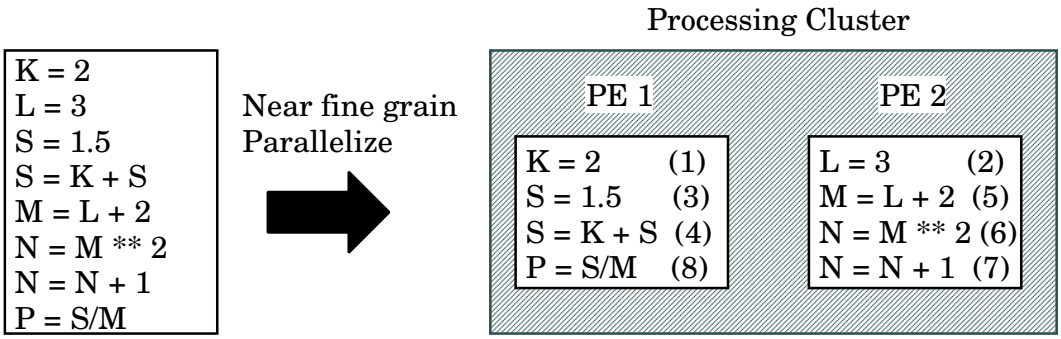


図 2.9: 近細粒度並列化

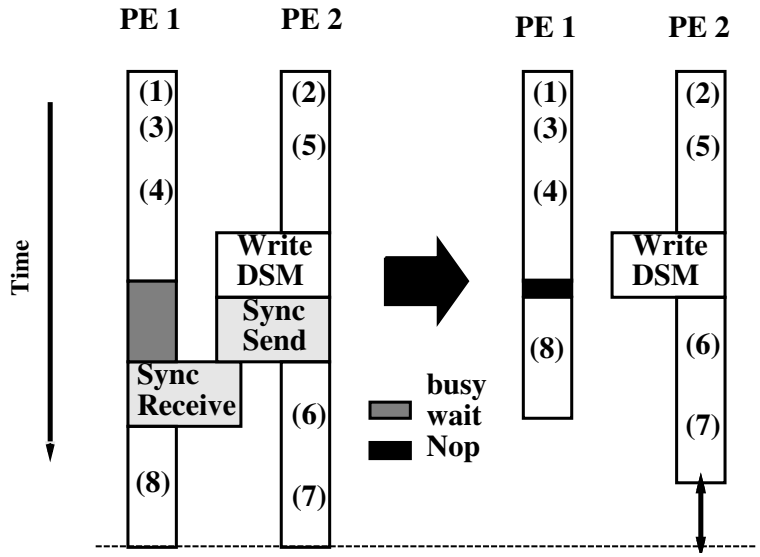


図 2.10: 無同期実行

2.2.5 マルチグレイン並列処理におけるデータ移動のまとめ

これまでの粗粒度・中粒度・近細粒度並列処理のデータ移動から，マルチグレイン並列処理におけるデータ移動をまとめる:

- 粗粒度並列処理において 動的に MT が各 PC(Processing Cluster) に割り当てられ，このときに集中共有メモリから PC へ MT のデータセットの読込みが発生する．
- PC に割り当てられた MT に対して中粒度並列処理 (Do-all) が適用される場合はループ中のデータ移動は起こらない．
- PC に割り当てられた MT に対し中粒度並列処理 (Do-across)，もしくは近細粒度並列処理が適用される場合は，PC 内の PE 間で小容量のデータ転送が発生する．
- MT の割り当てに伴う粗粒度並列処理のデータ移動は，実行時不確定性をもつため，動的に行われるのに対し, MT 内の処理中に行われるデータ移動は，中粒度 (Do-across) でも近細粒度並列処理においてもデータの転送先があらかじめコンパイル時に静的に定まるため，静的スケジュール可能である．
- MT が終了すると，CSM へデータの書戻しが発生する．

ここで，これらの特性について粗粒度並列処理における MT の割当てに伴う実行時に定まる動的なデータ移動と，MT 内の静的スケジューリング可能なデータ移動は混在する場合があることに注意されたい．これは，例えば PC1 の MT1 が終了して CSM へデータを書戻しを行っている場合に，他の PC2 で MT2 の近細粒度並列処理が実行中で，PC2 内の静的スケジューリング可能なクラスタ内データ交信と，PC1 と CSM の実行時に確定する動的なデータ交信が同時に行われるような場合が相当する．

このため，それぞれが互いのデータ交信を妨げない独立したトラフィックをもつ必要がある. すなわち，ネットワークも各 MT が処理される PC 単位で，閉じたパーティショニングが可能である必要がある.

また，近細粒度並列処理の項で述べた，正確な静的スケジューリングに基づく同期操作を一切排除した無同期実行を可能とするためには，コンパイル時にデータ転送遅延が明確に計算可能でなければならず，先に述べた，実行時に定まる動的なデータ移動のトラフィックと分離・独立して転送されるためのアーキテクチャサポートが不可欠となる. これについては，第 6 章にて述べる.

以上のことから，マルチグレイン並列処理において相互結合網に要求される事項は，次のようになる:

（1）集中共有メモリと PE を接続する大転送容量のグローバルネットワーク

（2）PC 内の PE を接続する高速なローカルネットワーク

（3）近細粒度並列処理における静的スケジューリングを妨げないための，PC 内で閉じたパーティショニングが可能なネットワーク

（4）静的スケジューリングのための一定の転送遅延を保証するネットワーク

これらを満たすネットワークの概念図は，図 2.11 に示すとおりである.

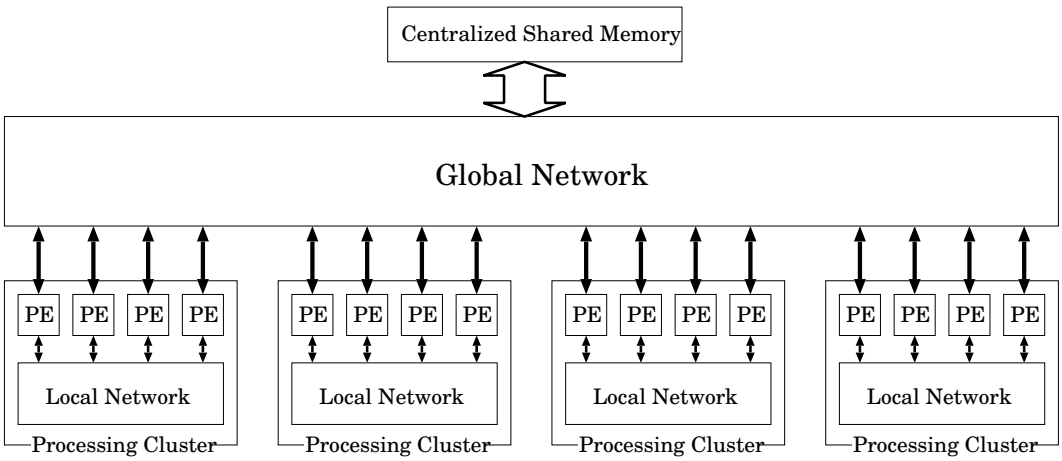


図 2.11: マルチグレイン並列処理に適したネットワークモデル図

2.3　並列化ソフトウェア技法を考慮した相互結合網

この節では，前の節の考察を踏まえた上で，これまでに自動並列化コンパイラなどの並列化ソフトウェア技法をターゲットとして開発されたマルチプロセッサにおける相互結合網について調査し，問題点について指摘する．

2.3.1　OSCAR

OSCAR は，自動並列化コンパイラが行うマルチグレイン並列処理 (並列度の粒度に応じた並列処理) を効率よく行うために開発されたマルチプロセッサで，1988 年に早稲田大学で実装された．OSCAR の構成は，図 2.12 に示すように 3 本の共有バスに 8 個のプロセッシングユニット (PU) が接続され，共有バスを介して集中共有メモリ (CSM) にアクセス可能となっている．OSCAR は，CSM の他に，分散共有メモリとローカルメモリを持つことにより集中共有メモリに対するアクセスを低減させている．

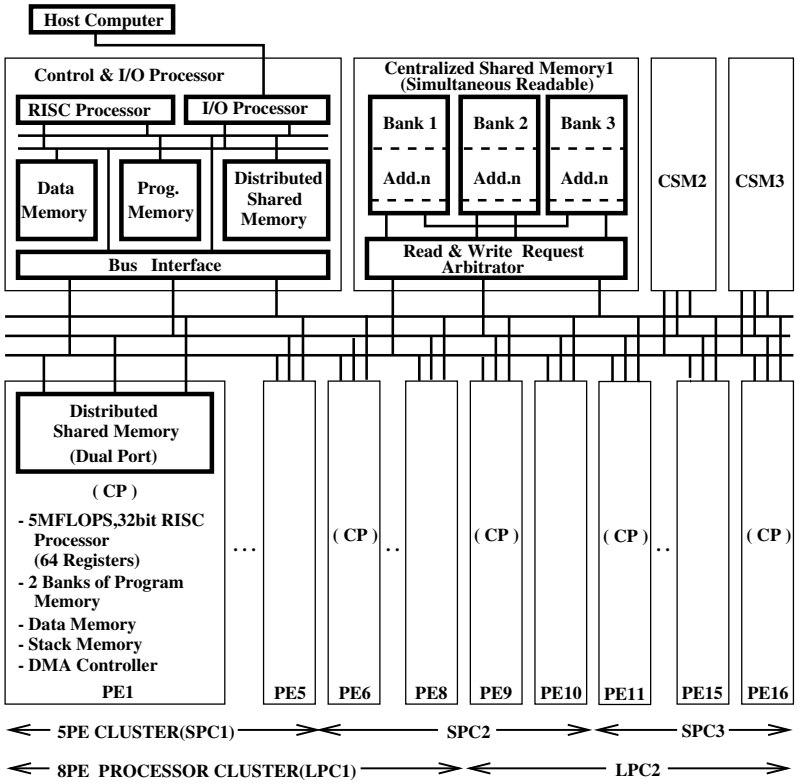


図 2.12: OSCAR の全体構成

OSCAR では静的解析可能なブロックにおいて，共有バスの所有についてスケジュールを行い，バスアクセスの競合が起こらないように制御する (詳細は 5.4 節参照)．したがって，プロセッサ間のデータ転送において必要となる同期操作をまったく行わずに実行すること (無同期実行) が可能となっており，この結果，並列性を最大限に抽出したプログラムをさらに高速化することが可能である[尾形 93]．

OSCAR は，マルチグレイン並列処理用並列計算機アーキテクチャの試作機として実装された．しかし, 実装された時期と最初の試作機であったことから OSCAR の多重バスには，次の問題点がある．

- 多重バス (3 本) であるため，転送容量が不足
- 規模拡張能力が乏しい
- グローバルトラヒック (大域通信: CSM へのアクセス) とローカルトラヒック (近接通信: DSM へのアクセス) が混在

このような問題点から，数十から数百のプロセッサを持つ中規模並列計算機には，多段結合網が用いられることが多い．

2.3.2 Cedar

Cedar は，Illinois 大学で 1993 年に開発された，マクロデータフロー処理を念頭においたマルチプロセッサである．その構成は，Alliant 社の小規模マルチプロセッサ FX-8 を 1 つのクラスタとして，これを 2 段の多段結合網 (Omega 網[D.H75]) を介して図 2.13 のように Global Memory に接続する．クラスタ内は，図 2.14 に示されたように，8 つの Processor，クラスタメモリに接続されている 4 way にインタリーブされたキャッシュとグローバルネットワークインタフェースをクラスタスイッチで結合している．

Cedar では，マクロデータフローによってマクロタスクを各クラスタに動的スケジューリングによって割り当てると，クラスタ内でこのマクロタスクを静的スケジューリングによって効率よく実行する．また，クラスタ内のネットワークは，クロスバで接続されているためコンパイラが静的にスケジューリングするのが容易である．このようにグローバルの階層とクラスタのローカル階層の 2 階層に分けることによって，静的スケジューリングと動的スケジューリングをミックスして効率よく行うことが可能である．

しかしながら Cedar の結合網には，次の問題点が存在する．

- クラスタ構成に柔軟性が持てない．
- グローバルネットワークの Omega 網は，ECL デバイスを用いた大掛かりなものであったにも関わらず，転送遅延が大きい．
- 衝突時の混雑によって，転送時間の予測が困難である．

このため，Cedar は 32PE でデータパラレル的な並列処理を行った評価は発表されているが，マクロデータフロー処理的な使われ方の評価は発表されていない．

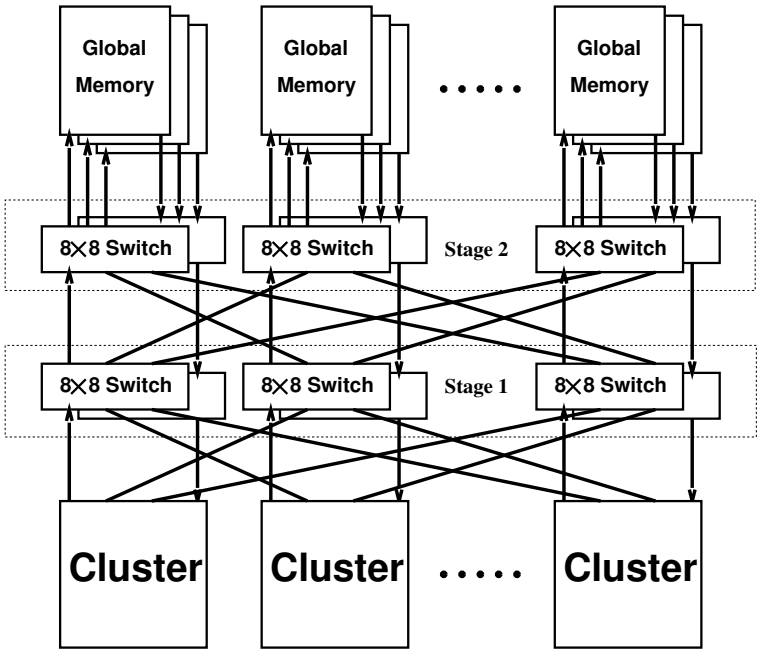


図 2.13: Cedar の全体構成

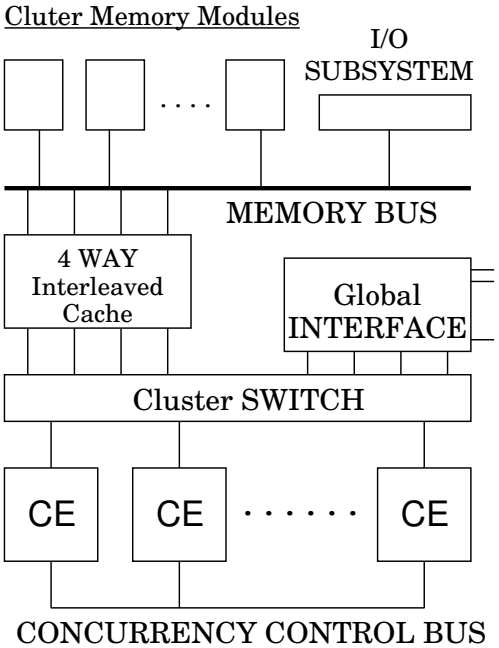


図 2.14: Cedar のクラスタ内構造 (FX-8)

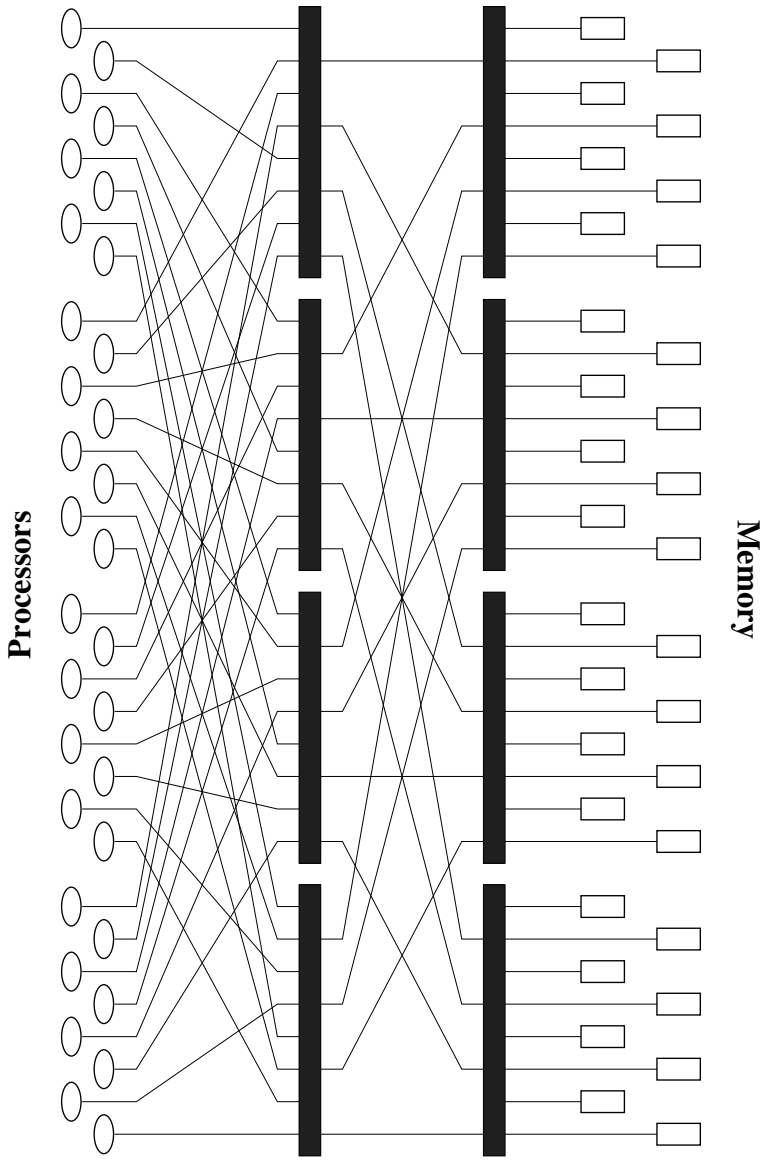


図 2.15: Cedar の相互結合網の構成

2.3.3　Multiscalar Project

1995 年に Winsconsin 大学によって提案された Multiscalar［GST95］［QEJ99］は, コンパイラのプログラム解析によるサポートに基づき，複数のフェッチエンジンと実行エンジンを並列動作させることにより逐次プログラムの高速化を図っている．

Multiscalar のコンパイラは, 逐次プログラムをいくつかの task のシーケンスに分解する．この task は, 単一のエントリポイントを持つ命令の静的グループである．図 2.16 に示すように, task はループやサブルーチンを内部に含むことができ, たくさんの出力を持っていても良いが, 入力エントリは 1 つであり, 開始点のアドレスで指定される．各 task の出力は別の task の入力ポイントを指示する．task 間はレジスタもしくはメモリの間で任意のデータ依存性を持ってよく, プログラムが逐次実行される場合と等価な動作を, ハードウェアが保証する．コンパイラは task 分割時に実行コードに加えて header と呼ばれるハードウェア実行を支援する情報を生成する．これには, 値を出力するレジスタ, 出力ポイントのリスト, 次の task の開始点が含まれる．

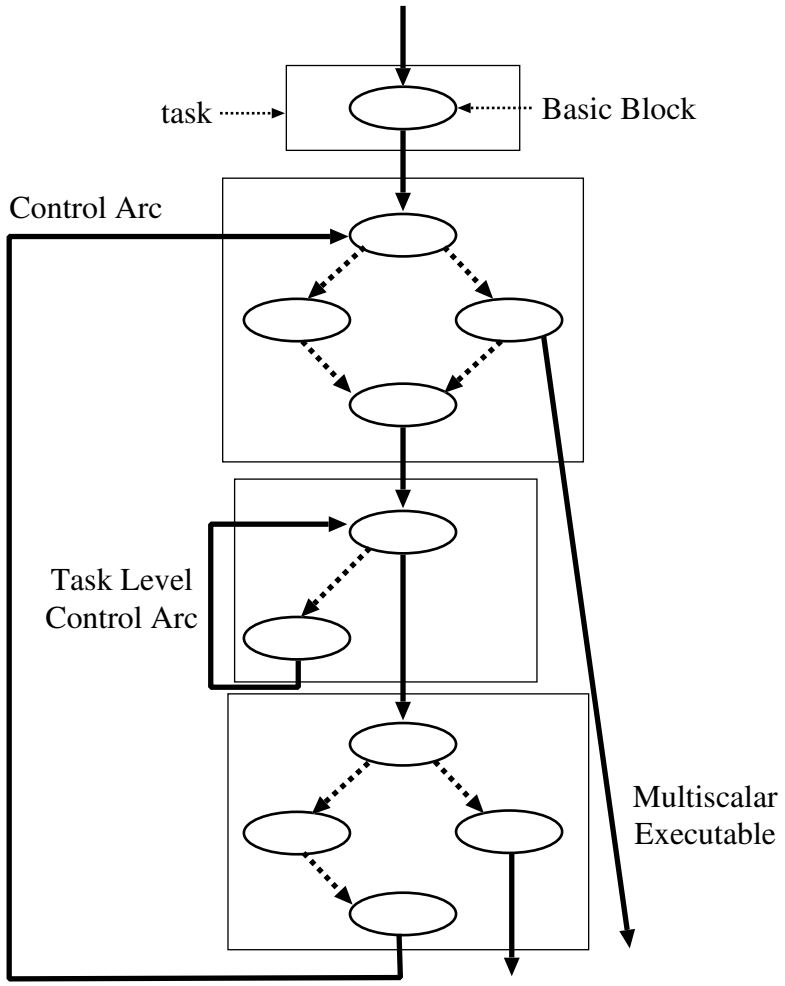


図 2.16: task の例

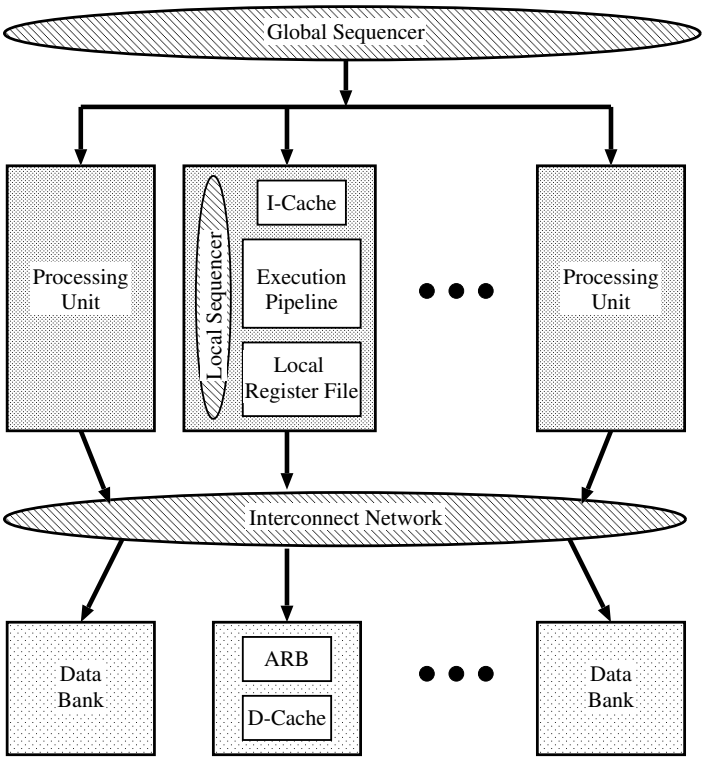


図 2.17: Multiscalar のアーキテクチャ

Multiscalar のハードウェアを図 2.17 に示す．Global Sequencer はどの task が実行されるのかを指示する．投機的な task はリング構造上に相互接続された Processor Unit で発生される．Multiscalar のプロセッサは, 割り当てられた task に対して同時に命令をフェッチし, 独立にデコード, レジスタリネーミングを行う．実行ユニットは簡単なスーパスカラを想定している．データの受渡しは, リング上の隣接プロセッサ間の共有レジスタを用いて行う．レジスタ間の受渡しは, コンパイラによって指定される．同期はメモリ上の特殊なハードウェア ARB(Address Resolusion Buffer) によって行う．このハードウェアはメモリに対する load と store を受とり, それらの依存関係が保たれているかどうかを確認して, これを満足するようにストールさせる．この ARB は集中実装されている．

評価の結果, 4Unit で Out-of order 発行を許す実装を行うと, SPECint では 1 倍から 2 倍, SPECfp では約 3 倍の性能向上を達成した［QEJ99］.

Multiscalar の task は, macro data flow の task に相当し, ほぼ ASCA の基本ブロックに相当する．すなわち, Multiscalar は ASCA における粗粒度並列処理を実行する構成を持つ. しかし問題の持つ粗粒度並列性は限られていることから, これでは並列性が不足する．このため Multiscalar は, プロセッサが投機的に task を発生して，生成した task をリング構造の隣の Processor Unit に割り付けることにより, 並列性を増強している．task のスケジュールも的な分岐予測技術を用いている点が注目される．これに対して ASCA はコンパイラで検出した階層的な並列性を利用するために階層構造を持ち, 投機的実行は行わない．このため, コンパイラによって解析仕切れない問題に対しても Multiscalar がある程度の性能

が発揮できるのに対して ASCA は基本的にコンパイラ頼みになっている点が異なる．一方で, ASCA の方がプロセッサ構成が簡単であり, スケールアップも容易である．

Multiscalar の接続網は，論文中に明記されている部分が少ないが，次の 2 つにより構成されると考えられる．

- Processing Unit 間を直結してリング状に接続する直結リンク
- D-Cache，ARB を内蔵したデータバンクとの間の接続網

前者は，task を投機的実行する際に，接続している Processing Unit 間で生成した task を受渡すのに使われ，後者は実行中のメモリアクセスに用いられる．Multiscalar はさほど大規模な構成を考えていないため，後者の接続網はクロスバあるいは，簡単な多段接続網で充分と考えられる．

これまで紹介した OSCAR, Cedar および Multiscalar の結合網についてまとめる．

- OSCAR は共有バスが大規模化のボトルネックとなる．
- Cedar は多段結合網をクラスタ内とクラスタ間で採用することにより高いスケジュール能力を持つが，フラットな構成であるためスケールアウトが困難である．
- Multiscalar Processor の結合網は，チップ内の結合網としてはリング構造で，タスク間のフローにおけるデータ受渡しには適しているが，Peer to Peer の通信が基本であり，一度に複数のアクセスが発生する場合には向いておらず局所的な通信向けである．

次の章では，クラスタ内で高い転送能力を持ち，スケジューリングに適し, かつ規模拡張性に優れた階層構造の多段結合網について提案する．

第3章　階層型多段結合網 R-Clos

前章で，マルチグレイン並列処理のデータ移動の特性について議論し，望ましい結合網の形態について触れた．本章では，提示されたこれらの条件を満たす相互結合網として Clos 網に階層構造を導入した階層型多段結合網 R-Clos を提案する．

3.1　多段結合網:MIN

多段結合網 (MIN: Multi-stage Interconnection Network) は，クロスバスイッチを 1 つのスイッチングエレメントにして，これを複数段にわたってスイッチ同士を接続して結合している．このような構成をとることによって１つのスイッチの経路切り替え数の段数乗個のデスティネーションノードへのアクセス経路を形成可能であり，共有バスに比べはるかに転送能力が高い．

MIN を用いたマルチプロセッサの一般的な形としては，共有メモリを各プロセッシングエレメントに分けて実現し，これを MIN によって各プロセッサと図 3.1 のように接続する．このとき，MIN によって結合されたネットワークにおいては，一般的にプロセッサが他のプロセッシングエレメント内の分散共有メモリ (DSM: Distributed Shared Memory) に値を書き込むことによってデータ交信が行われる．基本的にプロセッサは自分に割り当てられた分散共有メモリとローカルメモリのみを読み，必要なデータは，そのデータを所有しているプロセッサに書き込んでもらうという受動的な形態をとっている．

このような構成を持つ多段結合網は，メモリ上にデータを上手に配置してあげることによって，ネットワークにおける衝突をなくし，大量のデータを同時にアクセスすることが可能である．このためコンパイラがプログラムのアクセスデータの静的解析を行い，最も効率の良いアクセスパターンが形成可能なメモリ配置をしてあげる静的スケジューリングの研究は古くから考えられ，慶應義塾大学の SNAIL［笹原 95］などもこれを目標として実装された．このように多段結合網のデータアクセスの並べ替え能力の高さは，マルチプロセッサの結合網として用いる利点に挙げられている．

MIN は，その転送能力によりブロッキング網，リアレンジブル網，ノンブロッキング網に分類される．このうちマルチプロセッサの結合網として実装例が多いのは，Omega 網［D.H75］，Generalized Cube［HS78］網などのブロッキング網である．しかし，これらのブロッキング網は，異なる出力に対しても衝突が生じる方式であるため，より転送能力に優れ構成に柔軟性を持つ Clos 網［C. 53］に注目した．

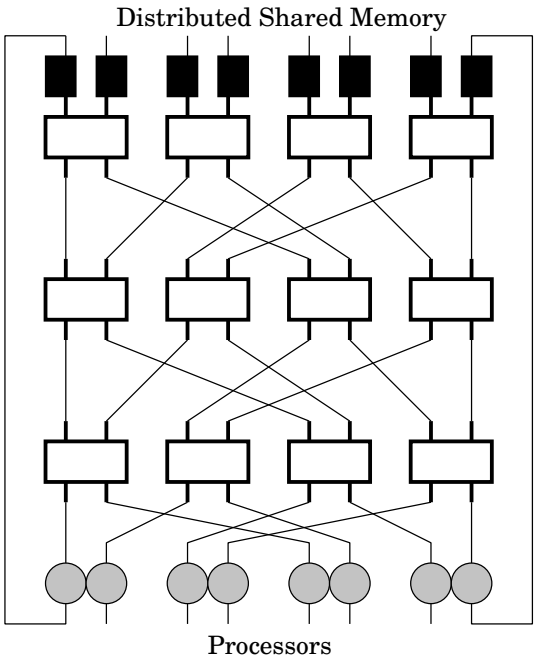


図 3.1: 一般的な MIN(Omega 網) によるマルチプロセッサ

3.2　Clos 網

Clos 網は，図 3.2 に示すように，入力段，中間段，出力段の 3 段からなる MIN で，これら 3 段をその機能からそれぞれ distributor，exchanger，concentrator と呼ぶ．各段の接続は，スイッチの出力ポート数 k を基数とした k 進数シャッフル接続である．

入/出力段のスイッチの入/出力数が n，中間段が $r \times r$ のスイッチ m 個で形成される Clos 網は，$V(m,n,r)$ という形で定義される．Clos 網では m, n の値によって網の能力が次のように定まることが知られている．

- $m < n$:ブロッキング
  すべての送信元と送信先のペアに対して独立した経路を作成することができないため，パケットの衝突が発生する．
- $m \geq n$:リアレンジブル
  すべての送信元と送信先のペアに対して独立した経路を作成可能であるが，そのために一度の経路設定の試行の後，経路を再構成する必要がある．
- $m \geq 2n - 1$:ノンブロッキング
  すべての送信元と送信先のペアに対して独立した経路を，経路の再構成の必要なしに構築可能である．

このように Clos 網は，m，n，r によって異なる性質を持つネットワークを構成することが可能である．このうちノンブロッキング Clos 網は，パケットの経路を動的スケジューリングしなくても，無衝突で送信先に到着させることが可能であるが，ブロッキング網に

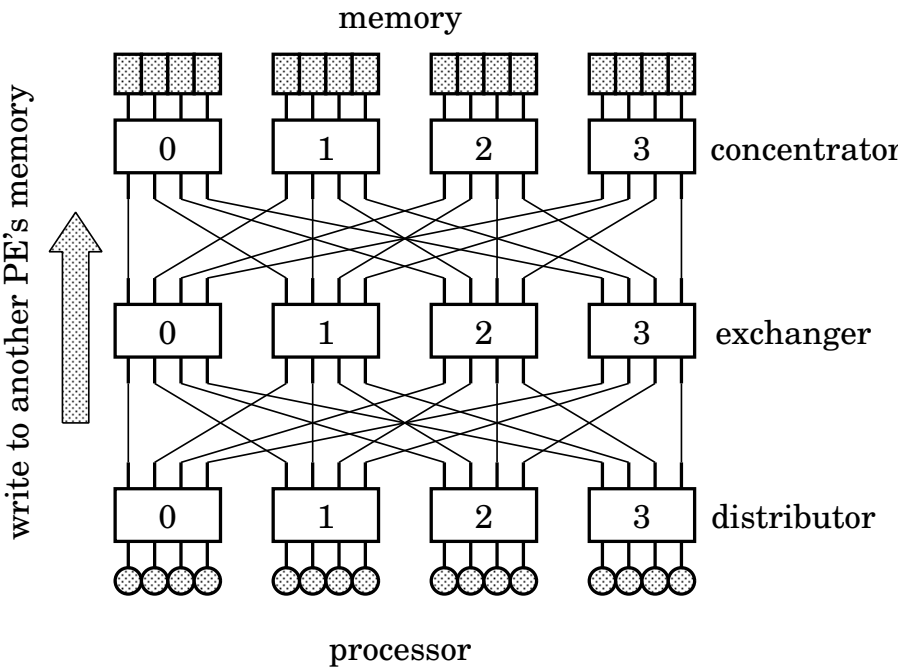


図 3.2: Clos 網

比べれば，はるかに大きなハードウェア量を必要とするため大規模なシステムを結合するのは不向きである．

一方で，リアレンジブル Clos 網は，パケット経路を一度構築したのちに再構成する必要があるが，ノンブロッキング Clos 網よりも少ないハードウェアで済む. このため，再構成を高速に行うための手法[TS96][D.R92]や，パケットの経路制御によってより少ないハードウェア量で衝突を回避する条件が研究されている[YJ99]．

3.3　R-Clos - **階層構造の導入** -

リアレンジブル Clos 網は，単純な構成と転送容量の点で，優れた特徴を持っているが，他の MIN 同様，交信の局所性を利用することができず，スケールを大きくするとスイッチのサイズ，ハードウェア量が膨大になる問題点がある．

そこで我々は，等距離間接網の MIN が持つこれらの利点を活かしつつ，これらを階層化することによって，効率の良い大規模スケーラブル結合網を構築することを考えた．この構想は，

- 近傍 PE との通信は高速に，
- 共有メモリおよび遠方 PE との通信をカバーした
- 規模拡張性に優れたネットワーク

を目指している．

上記の目的を達成するためは，近傍 PE との低レンテイシ転送ネットワークとして Clos 網を採用し，これの中間段 (exchanger) に階層構造を導入することによって複数の Clos 網を接続した，大規模なグローバルネットワークが形成可能な R-Clos を提案する．

Clos 網の階層化の方式については, 様々な形態をとることが考えられるが，Clos 網の最大の特長である中間段の構成変更による転送能力の自由度を利用するために，中間段を拡張させて階層化させることとした．階層化の方式についての考察は, 後の 3.6 節で述べる．

3.3.1　R-Clos の構成

R-Clos は，Clos 網の持つ近傍 PE との低レイテンシ・高バンド幅である性質を活かし，Clos 網を基とした階層木構造のトポロジを形成することによって大規模なグローバルネットワークの形成をサポートする．このため，V(k,k,k) のリアレンジブル Clos 網を基本ネットワークとして，この中間段を階層構造に拡張することによって複数の Clos 網を再帰的に階層化する．

R-Clos の基本ネットワーク $V(k,k,k)$ の Clos 網からの階層拡張化及び構成を説明するため，次に定義を示す:

定義 1 [基本ネットワーク部 -Clos 網-]

R-Clos の基本ネットワーク部は，それぞれの段が，スイッチングエレメント $k \times k$ のクロスバ k 個で構成される Clos 網とする．各スイッチングエレメントの入力ポート，出力ポートに，0 〜 $k-1$ のラベルを振る． 入力段，中間段，出力段を，それぞれ distributor, level-1 exchanger， concentrator と名付ける．各段のスイッチングエレメントに，0 〜 $k-1$ の数を振り，$j(0 \le j \le k-1)$ 番目の distributor，level-1 exchanger，concentrator をそれぞれ D_j, $E_{1,j}$, C_j と表す．このとき，各段の接続を，shuffle exchange によって次のように定義する．

- D_j の出力ポート $m(0 \le m \le k-1)$ は，$E_{1,m}$ の入力ポート j に接続する．
- $E_{1,j}$ の出力ポート $m(0 \le m \le k-1)$ は，C_m の入力ポート j に接続する．

上記によって定義される Clos 網は，［C. 53］によってリアレンジブルであることが証明されている．

定義 2 [$R-Clos\ (i,k)$ の定義 $(0 \le i)$]

$R-Clos\ (i,k)$ は，階層 i の，k を基数とする R-Clos を示すこととする．はじめに，$R-Clos\ (0,k)$ は，$k \times k$ のクロスバスイッチとし，$R-Clos\ (1,k)$ は，定義 1 により定義された Clos 網とする．$R-Clos\ (i,k)\ (i>1)$ は，k 個の $R-Clos\ (i-1,k)$ を次で定義される拡張段によって接続することによって形成される．

1. k 個の $R-Clos\ (i-1,k)$ を相互接続するために，$k \times k$ のスイッチングエレメント k 個 (level-i exchanger と定義，$E_{i,m}$, $0 \le m \le k-1$) によって階層 i の拡張段を形成する．(階層 1 の段は，基本ネットワーク部の level-1 exchanger)

2. $R-Clos\,(i-1,k)$ に，0 〜 $k-1$ までの数をふり，$j(0 \le j \le k-1)$ 番目 $R-Clos\,(i-1,k)$ を，$R-Clos\,(i-1,k)_j$ と表す．

3. $R-Clos\,(i-1,k)_j$ のすべての $E_{i-1,m}$ に拡張リンク (上り) 用の拡張出力ポート 1 つを追加して，このポートを k 番目の出力としてラベルを割り当てる．

4. $R-Clos\,(i-1,k)_j$ の $E_{i-1,m}$ の出力ポート k と，$E_{i,m}$ の入力ポート j を接続する．

5. 拡張リンク (下り) 用の拡張入力ポート 1 つを

   - $i > 2$ である場合は，$R-Clos\,(i-1,k)_j$ の $E_{i-1,m}$ に
   - $i = 2$ である場合は，$R-Clos\,(1,k)_j$ の C_m に

   追加して，このポートに k 番目の入力としてのラベルを割り当てる．

6. すべての $E_{i,m}$ の出力ポート j を，

   - $i > 2$ である場合は，$R-Clos\,(i-1,k)_j$ の $E_{i-1,m}$ の入力ポート k
   - $i = 2$ である場合は，$R-Clos\,(1,k)_j$ の C_m の入力ポート k

   に接続する．

定義 2 を階層 1 から階層 2 の R-Clos への拡張を例に説明する．k 個の Clos 網を接続するために，k 個の $k \times k$ のスイッチングエレメントで構成する 1 段の拡張段を level 1 ステージに追加する．新たに追加した拡張段を level 2 ステージと呼び，この段のスイッチングエレメントを level-2 exchanger と呼ぶ．拡張した level 2 ステージと level 1 ステージを，定義 2 によって接続する．図 3.3 は，Clos 網から階層拡張化した R-Clos の詳細を示している．

上位階層の exchanger の入力/出力ポートと拡張入力/出力ポートを接続するリンクを基本ネットワークの Clos 網内のステージ間接続 (基本リンク) と区別して，拡張リンクと呼ぶことにする．定義 2 より階層 2 から階層 1 へのダウンリンクの拡張リンクは，他の階層間の接続と異なり図 3.4 のように基本ネットワーク部の最終段の concentrator に付加される拡張入力ポートに接続する点に注意されたい．

このように定義 2 によって，$R-Clos(i,k)$ は，$R-Clos(i-1,k)_0$，$R-Clos(i-1,k)_1$，…，$R-Clos(i-1,k)_{k-1}$ から構成され，これら各階層の $R-Clos(i,k)$ を，level-i のクラスタと呼ぶ．

図 3.5 に，4 個の $R-Clos(i-1,4)$ を level-i exchanger によって相互接続して構成される $R-Clos(i,4)$ を示した．例えば k を 4 とした場合の level-$i-1$ から level-i へのアップリンクは，$R-Clos(i-1,4)_1$ に属する $E_{i-1,2}$ の拡張出力を $E_{i,2}$ の入力ポート 1 に接続する．一方，level-i から level-$i-1$ へのダウンリンクでは，例えば $E_{i,3}$ の出力ポート 2 は $i > 2$ では $R-Clos(i-1,4)_2$ の $E_{i-1,3}$ の拡張入力ポートに接続し，$i = 2$ では C_3 の拡張入力ポートに接続する．

また，拡張ネットワークの拡張段 E_n のスイッチングエレメントの個数 $W(E_n)$ は，次の式で表される．

$$W(E_n) = (\frac{N}{k})(\frac{1}{k})^{n-1}, \qquad (1 < n < R) \tag{3.1}$$

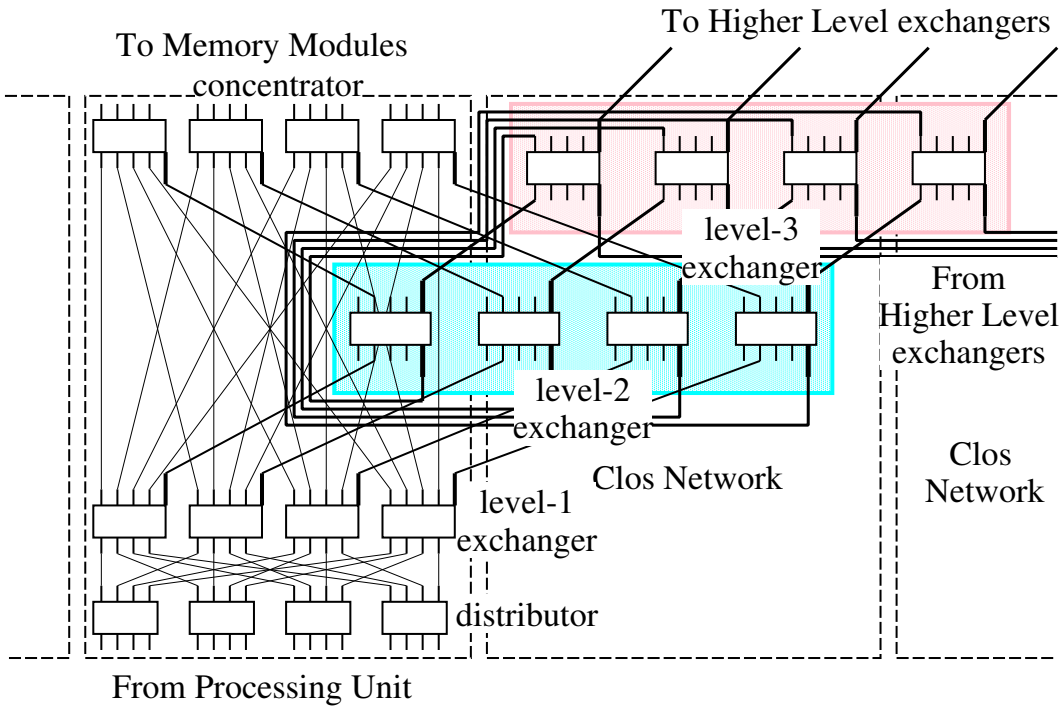


図 3.3: R-Clos の構成 (詳細図)

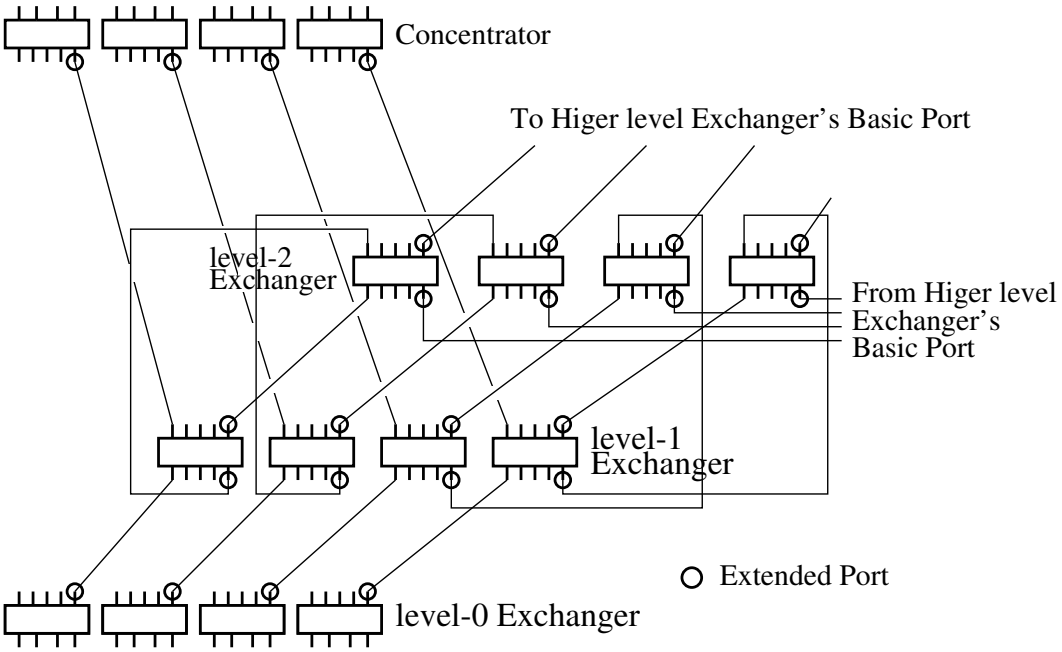


図 3.4: 拡張ネットワークの接続と拡張ポート

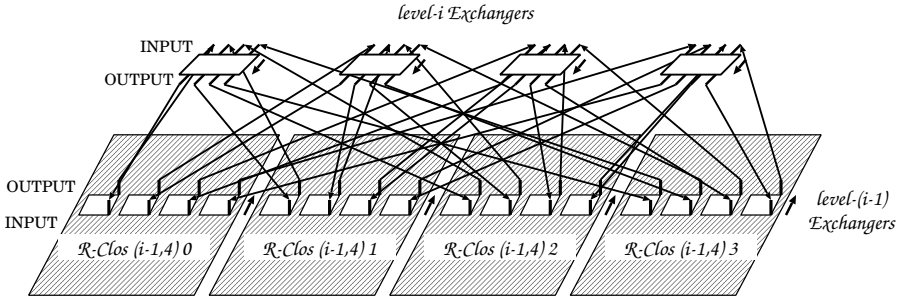


図 3.5: 階層間の接続 level-$i-1$/level-i

R-Clos に接続される全 PE 数を N とすると，形成される階層の最大数 R は，

$$R = \log_k N - 1 \tag{3.2}$$

となり，拡張ネットワークの段数 S は，$R >= 2$ で $R-1$ となる．

例として，$k = 4$，階層数 3，全 PE 数が 256 である場合の R-Clos を図 3.6 に示す．拡張ネットワークの接続は，図の明確化のため一部のみを示している．

3.4 ルーティング

3.4.1 導入

はじめに，ルーティングアルゴリズムを求める準備として，階層クラスタ，スイッチの入力ポート，出力ポートに k 進数のラベルを振ったのと同様に，PE の位置に k 進数のラベルを振る．例えば，PE のラベルが，$x_{i-1}x_{i-2}\ldots x_1x_0y$ と表される時，この PE は，クラスタ $R-Clos(i-1,k)_{x_{i-1}}$，$R-Clos(i-2,k)_{x_{i-2}}$，$\ldots$，$R-Clos(1,k)_{x_1}$，$R-Clos(0,k)_{x_0}$ に所属する distributor D_{x_0} の入力ポート y に繋がれている．ここで，送信元 $s_{R-1}s_{R-2}\ldots s_1s_0$ から，送信先 $d_{R-1}d_{R-2}\ldots d_1d_0$ への転送について経路を，送信元と送信先が同一の Clos 網である場合とそうでない場合について考える．

3.4.2 基本ネットワーク部 (Clos 網) 内の転送

この場合，$s_{R-1}s_{R-2}\ldots s_2 = d_{R-1}d_{R-2}\ldots d_2$ が成り立つ．定義 1 から，Clos 網内の基本リンク のみを使用することによって，送られているパケットの位置は，下位 2 桁が交換される．したがって，ルーティングに使用されるタグ T は，

$$T = \{*, d_1, d_0\}, \qquad (* = \forall z \,|\, z = 0, 1, \cdots, k-1) \tag{3.3}$$

したがって，式 (3.3) から，distributor ではどの出力ポートを使用しても正しくルーティングされる．このため基本ネットワーク部内での転送では経路冗長性が存在する．

例として，k が 4 である場合の，PE0 から PE15 までの任意の PE が PE13 に対してデータを送信する場合のルーティングを考える．13 は，4 進数で表すと 31 となるため，$d_1 = 3$，$d_0 = 1$ である．各段のスイッチの入/出力ポートは，0 から順に k 進数でラベルが振られて

いるため，d_1 は Clos 網内での相対スイッチ番号，d_0 はそのスイッチでのポート番号となる．Clos 網では，定義 1 から各段は shuffle exchange で接続されるため，ルーティングタグは $*, 3, 1$ となる．ただし $*$ は，0 から 3 までの任意の整数でよいため，経路としては 4 通りの経路が考えられることになる．

3.4.3　拡張ネットワーク部を使用する転送

送信元と送信先が異なる基本ネットワーク部の Clos 網に属する場合，$s_{R-1}s_{R-2}\cdots s_2 \neq d_{R-1}d_{R-2}\cdots d_2$ である．したがって，下位 2 桁の交換のみをおこなう基本リンクだけでは正しくルーティングできないので，拡張リンクを使用して拡張ネットワーク部を通過させる．R-Clos は，定義 2 で示すように，階層 2 の $R-Clos(2,k)$ の $E_{2,m}$ の出力ポート j が，階層 0 の $R-Clos(0,k)_m$ に直接接続されているため，distributor で level-0 クラスタ ($R-Clos(0,k)_m$ で構成) が決定される．したがってルーティングタグの先頭には，送信先の level-0 クラスタのラベルが入れられる．そして，level-1 exchanger に到達したパケットは，送信元と送信先が同じクラスタ内に入る階層 (つまり $d_{R-1}d_{R-2}\cdots d_{r+2} = s_{R-1}s_{R-2}\cdots s_{r+2}$) となるまで拡張リンクを使用して階層を上がり続け (r 回 拡張ポートを選択する)，level-$r+1$ exchanger に到達する．パケットの位置ラベルは，拡張リンクを 1 つ上がる度に，右に 1 桁シフトされる (上位の桁には 0 が挿入)．逆に拡張リンクを使用して階層を 1 つ下がる度に，パケットの位置ラベルは，左に 1 桁シフトされる．このためルーティングタグ T は，次のようになる．

$$T = \{d_1, \overbrace{k, \cdots, k}^{r}, \overbrace{d_{r+1}, d_r, \cdots, d_2}^{r}, d_0\} \tag{3.4}$$

ただし，k は，拡張出力ポートを表す．式 (3.4) から，拡張ネットワークを使用するデータ転送は，静的にただ 1 通りに定まる．

説明のために，k が 4 である場合の PE0 から，PE228 に転送する場合を例に示す．PE0 は 4 進数で表すと 0 である．一方，PE228 を 4 進数で表すと 3210 となる．PE0 と PE228 が同一クラスタ内に所属する階層は 3 である (s_4 と d_4 がはじめて同じ 0 となるため，r は 2) ことから，ルーティングタグ T は $1, 4, 4, 3, 2, 0$ と求まる．このとき送信パケットは，2 回 exchanger の拡張リンクを使用して level-3 exchanger に到達したのち，そこから出力 3 を選択して $R-Clos(2,4)_3$ に入り，続いて出力 2 を選択して $R-Clos(1,4)_2 \cdot R-Clos(2,4)_3$ の C_1 に入力され，最後に出力 0 を選ぶことにより PE228 に届けられる．

3.5　R-Clos を用いた共有メモリ型並列計算機の構成例

R-Clos を用いて共有メモリ型並列計算機を構成する場合は，図 3.7 のように，どの PE からも等距離の位置に共有メモリを配置するのが一般的であると思われる．

PE 間のローカルな通信は，他の MIN を用いた並列計算機と同様に $R-Clos(1,k)$(つまり Clos 網部分) を通して相手 PE のメモリに対する単方向の書込みによって行われる．一方，集中共有メモリへの書込みや，読出し要求パケットは階層のアップストリームネットワーク (上位階層へのリンクを使用) によって実現され，逆に集中共有メモリからの読み出しデータパケットはダウンストリームネットワーク (下位階層へのリンクを使用) を使用することによって行われる．

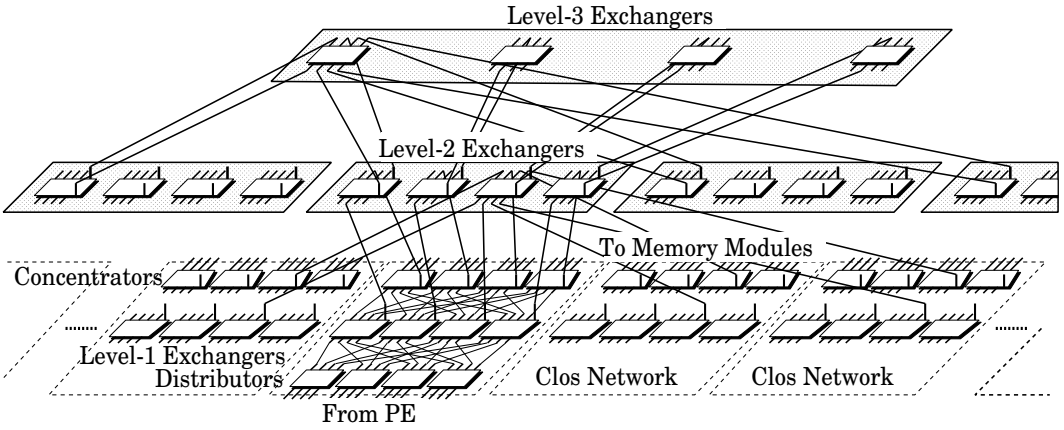


図 3.6: R-Clos の構成 (全体図) - k = 4，256 PE

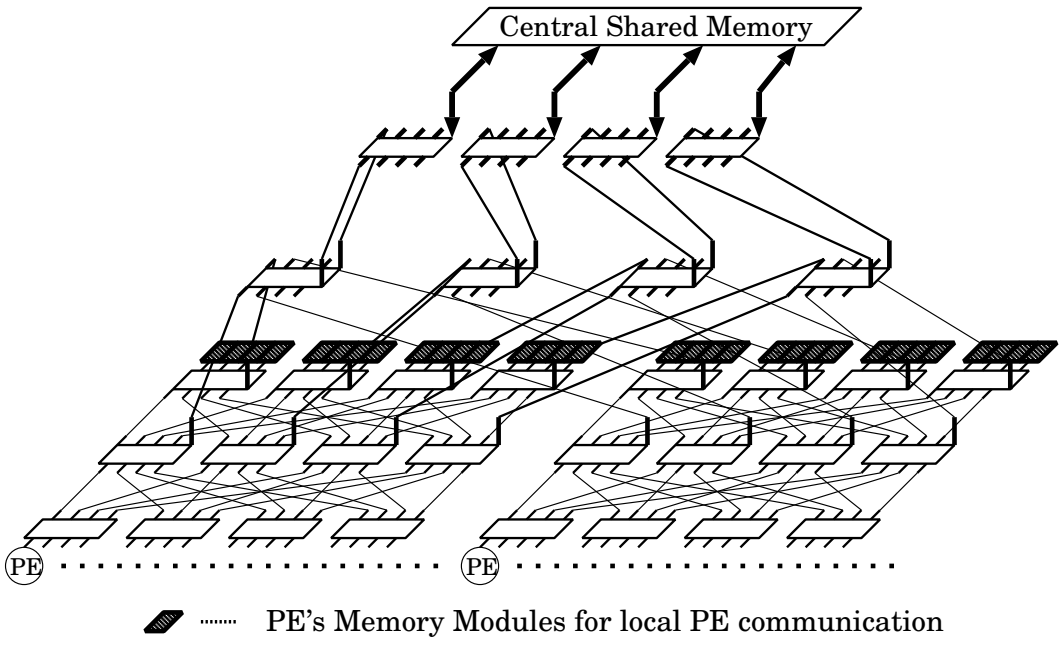


図 3.7: R-Clos を用いた共有メモリ型並列計算機の接続例

このように，R-Clos は共有メモリや送信先 PE のメモリに対する単方向のデータの書込みだけではなく読出しのような双方向の通信もサポートした MIN である．

3.6　階層化方式の検証

ここでは，Clos 網の階層化について, R-Clos で採用した中間段の再帰階層構造の妥当性を検証する．Clos 網の特長である中間段の構成と転送能力との関係を考慮すると，$V(m,n,r)$ で定義される Clos 網において, 中間段のスイッチ数 m を増やすことによって，転送経路が増加する性質がある．中間段に, 階層化の拡張を施すことにより，中間段のスイッチ数 m に応じた階層間転送能力を確保することが可能である．また中間段に階層化の拡張経路を設けることにより，同じ入力段からの分散されたパケットが, それぞれの中間段を用いて最大 m 個の階層間経路を形成することが可能である．これらの性質を説明するため, R-Clos を双方向 MIN で表したものが図 3.8 である．一般に単方向の MIN のトポロジは, 双方向 MIN で等価なトポロジを実現可能である．Clos 網を双方向 MIN で実装すると, 中間段を頂点とする 2 段構成の階層ネットワークになる．仮に頂点である中間段以外の入力段で階層化させても，Clos 網の特徴である中間段における並べ替え能力を利用することができず，同一の入力段で階層経路を利用できるアクセスは同時にだた 1 つに限られてしまい, ベースとする結合網として Clos 網を採用する利点が得られない．一方で，中間段に階層化の拡張を施すことにより，Clos 網の並べ替え能力を階層化にも利用することができ, 同一の入力段から，最大で m の階層経路をとることが可能である．したがって，Clos 網の階層化の拡張の方式としては, 提案手法が Clos 網の並べ替え能力を利用することができる点で適していることがわかる．

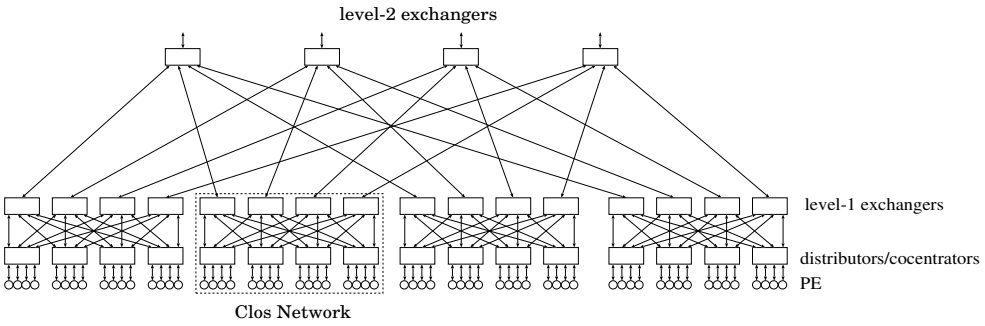


図 3.8: R-Clos の双方向 MIN による表現

3.7　関連研究

3.7.1　MIN の拡張

MIN は等距離間接網であるため，交信の局所性を満足することができない．この特性を補うためにいくつかの手法が提案されている．MIN の段の途中からバイパス路を設ける方式［TK91］は，Generalized Cube［HS78］や Baseline［CT80］などに適用すれば，階層的なアクセスを実現可能である．しかし，この構成を取ると，異なった階層に接続されたメモリとは転送を行なうことができない．

MIN のスイッチングエレメントに U-turn 経路を設ける方法［JJ84］も MIN が本来持っている階層性を生かすことができるが，U-turn 制御によりエレメントの構造が複雑になる．また，この方法は，後に示す Fat Tree と類似したトポロジとなる．一方，図 3.9 に示す 3 次元 MIN［塙 敏 98］は，一方向のみの転送で局所性を満足するための拡張である．この方法では，$PU_{i,j}$ は i 行および j 列のメモリのみ MIN を 1 回介してアクセス可能であり，それ以外のメモリに対してはループ構造を用いることで MIN を 2 回，巡回してアクセスを行なう．

これらの MIN の拡張は，単方向通信路で構成され，クラスタ内にのみ強力な転送能力を有する R-Clos とはかなり性質が異なっていると言える．

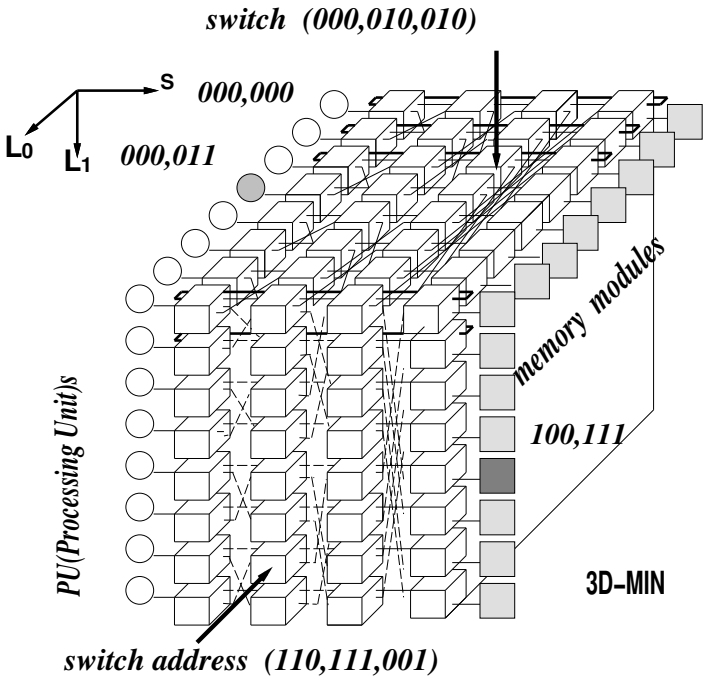


図 3.9: 3 次元 MIN

3.7.2 不等距離間接網

base-m n-cube［TSe91］，ハイパクロス［He91］など不等距離間接網はいずれも交信の局所性を利用することができるが，階層性があり，R-Clos と関連が深いのは Fat-tree［C.E85］である．

Fat Tree は，多重化されたツリーで，上向きリンク数 p と下向きリンク数 q，及び階層数 r の組で Fat Tree(p,q,r) で表される．このとき，接続可能な PE 数は，q^r 個である．Fat Tree は，容易に大規模システムに対応可能である点と，その柔軟な階層多重木構造から，近年の商用ネットワーク［FWA+02］のトポロジとして用いられている．

R-Clos の上位階層の接続は Fat tree のトポロジと類似しているが，次の相違点がある．

1. R-Clos は MIN の拡張であるため，転送路は基本的に単方向である．
2. クラスタ内は Clos 網を内包しており，多数の冗長経路を持つ．

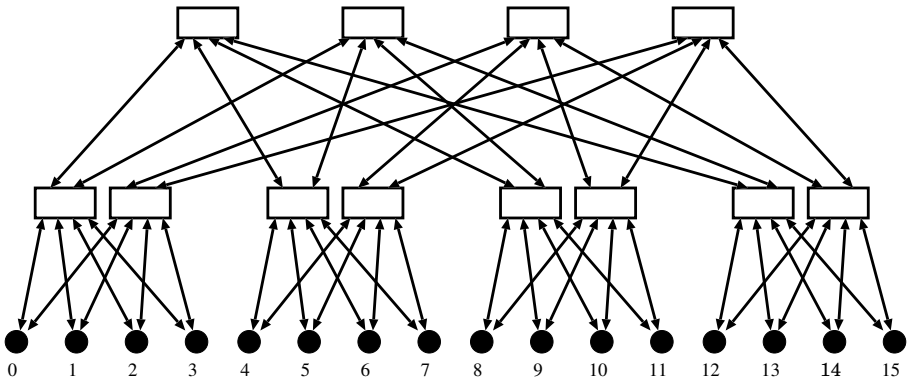


図 3.10: Fat Tree

3.7.3 recursive n-stage Clos

R-Clos と同様に，Clos 網の特徴や高い転送能力をより大規模なシステムに適応させるための結合網が Clos. C によって提案されている．この方法は，3 段の構成の Clos 網を再帰的に構成することにより，より多数のノードを接続可能にする方法［C. 53］で，本論文の方法に近いため，ここで詳しく紹介する．本論文では，これを R-Clos と区別するために recursive $(2s-1)$-stage Clos 網 ($s \geq 3$) と呼ぶこととする．

recursive $(2s-1)$-stage Clos 網を説明するため，recursive 5-stage Clos 網を例にする．recursive 5-stage Clos 網 (s が 3 の場合にあたる) は，3 段の Clos 網の中継段である exchanger を，Clos 網で置き換える．すなわち，中継段を Clos 網で構成し，これに入力段と出力段を接続することにより，より多くのノードを接続する．中間段の役割をする Clos 網の数によって，Clos 網と同様にネットワークの転送能力をノンブロッキング，リアレンジブル，ブロッキングと変化させることが可能である．中間段が $V(n,n,n)$ である Clos 網 (これを正方の Clos 網と呼ぶ) である場合，全ノードの接続数 N は n^3 である．図 3.11 に，$n=4$

の正方の recursive 5-stage Clos 網を示した．この図では理解を容易にするためステージ間の接続は特徴的なものについてのみ示した．

recursive $(2s-1)$-stage Clos 網は，recursive $(2(s-1)-1)$-stage Clos 網を中継ネットワークとしてこれに入出力段を接続する．このとき recursive $(2s-1)$-stage Clos 網が接続可能な全ノード数 N は，正方の Clos 網の再帰ルールで構成した場合，n^s である．

このように recursive $(2s-1)$-stage Clos 網は，中間段を再帰的に構成することによって大規模な結合網を実現するため，R-Clos に比べ i) 転送容量が大きい が，ii) ハードウェアのコストが高く，iii) 転送遅延が一様にかかるため，データ配置の局所性を活かすことができない．

このため，Clos による recursive $(2s-1)$-stage Clos 網は，大局的でデータの局所性などがないグローバルなネットワークとしては適切であるが，並列計算機などのようなデータ配置の局所性が性能向上に大きく寄与するシステムには向いていない．

R-Clos と，この recursive $(2s-1)$-stage Clos の基本特性は 4.1 節 にて, 転送性能の評価の比較は 4.3 節，ハードウェアコストの比較は 4.2 節でそれぞれ示す．

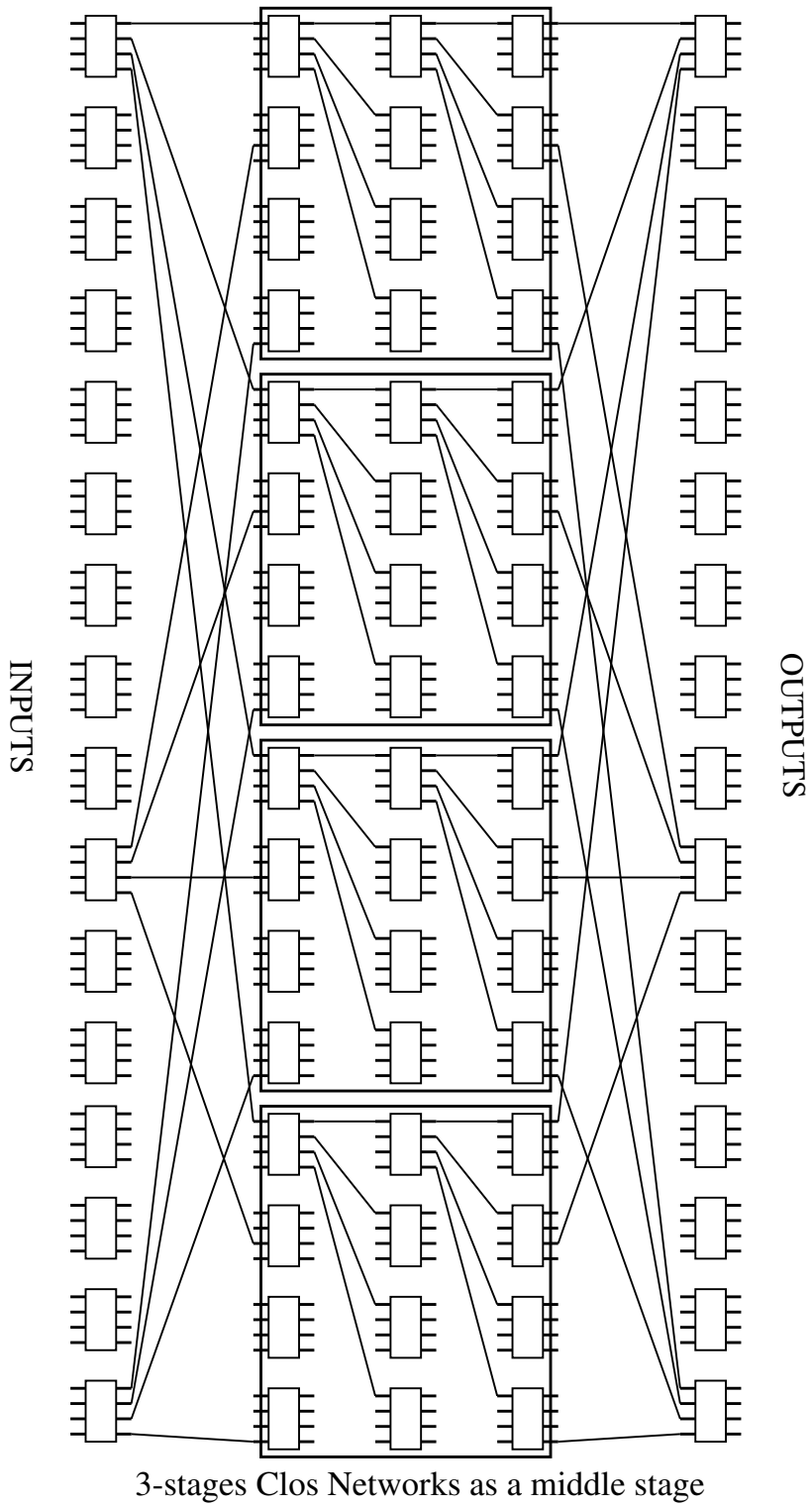


図 3.11: recursive 5-stage Clos 網

第4章　基本転送特性の評価

4.1　ネットワークの基本特性

提案した R-Clos と，比較のために Clos 網，および後の 3.7.3 節にて紹介する recursive $(2s-1)$-stage Clos 網について，表 4.1 に，スイッチの入出力端子数，接続 PE 毎の階層数，最小転送ホップ数，最大転送ホップ数を示す．

ここで，比較対象の Clos 網は正方 (スイッチの端子数 $k = \sqrt{N}$) のリアレンジブル Clos 網とした．recursive $(2s-1)$-stage Clos($s \geq 3$) も，スイッチ端子数 k に基づいて再帰化したもので，r-$(2s-1)$st-Clos と略記している．表中の拡張レベルは，再帰数または階層数を示している．

表 4.1: ネットワークの構成とルーティングホップ数の比較

PE 数	ネットワーク	スイッチサイズ	構造 (拡張レベル)	ホップ数 最小	ホップ数 最大
64	Clos	8×8	フラット	3	3
	r-5st-Clos	4×4	中間段再帰 (2)	5	5
	R-Clos	4×4，4×5，5×5	階層構造 (2)	3	4
256	Clos	16×16	フラット	3	3
	r-7st-Clos	4×4	中間段再帰 (3)	7	7
	R-Clos	4×4	階層構造 (3)	3	6

表 4.1 より，R-Clos は，Clos 網に比べてスイッチサイズが小さくて済み，recursive $(2s-1)$-stage Clos に比べると近傍 PE に対するルーティングホップ数が小さいことがわかる．

4.2　ハードウェア量の見積もり

次に，R-Clos のハードウェア量をスイッチの交点数で評価し，同じサイズの Clos 網および recursive $(2s-1)$-stage Clos と比較する．N を接続されている PE 数とし，これをネットワークの入力数，および出力数と等しいものとする．基本ネットワーク部の Clos 網を $V(k,k,k)$ とすると，式 (3.1) と式 (3.2) から交点数 CP_{rclos} を計算して求めると, 次の式で

表される．

$$
\begin{aligned}
CP_{rclos} &= CP_{distributor} + CP_{exchanger} + CP_{concentrator} \\
&= Nk + \sum_{r=1}^{R-1} (k+1)^2 \frac{N}{k} \left(\frac{1}{k}\right)^{r-1} + k^2 \frac{N}{k} \left(\frac{1}{k}\right)^{R-1} \\
&\qquad - k\frac{N}{k} + k(k+1)\frac{N}{k} \\
&= 2Nk + \frac{N(k+1)^2}{(k-1)} \left(1 - \left(\frac{1}{k}\right)^{R-1}\right) + \frac{N}{k^{R-2}} \\
&= 2Nk + \frac{N(k+1)^2}{(k-1)} \left(1 - \left(\frac{k^2}{N}\right)\right) + k^3 \\
&= \frac{N(3k^2+1) - k^2(3k+1)}{k-1}
\end{aligned}
\tag{4.1}
$$

表 4.2 に，$k = 4$ の場合の R-Clos と Clos 網と recursive $(2n-1)$-stage Clos 網の交点数を示す．右から 1 番目と 2 番目の項目は，それぞれ recursive $(2n-1)$-stage Clos 網，および Clos 網に対する R-Clos の交点数の比率を示している．この表から R-Clos は，Clos 網および recursive $(2n-1)$-stage Clos よりも少ないハードウェア量で大規模なシステムを結合することが可能であることがわかる．4 階層に重ねると Clos 網の約 17%，recursive 9-stage Clos 網の約 45%のハードウェア量となる．

表 4.2: ハードウェア量の比較 [交点数] ($k = 4$)

# of PEs	R-Clos	Clos	r-Clos	$\frac{R-Clos}{Clos}$	$\frac{R-Clos}{r-Clos}$
64	976	1,536	1,280	64%	76%
256	4,112	12,248	7,168	34%	57%
1,024	16,656	98,304	36,864	17%	45%

4.3　シミュレーションによる転送能力の評価

提案した R-Clos の転送性能を調べるため，PE 毎の単位サイクル当たりのパケット送信数によって正規化した PE 毎の単位サイクル当りのパケット受信成功率 (normalized accepted traffic) と平均転送遅延 (average latency) を確率モデルシミュレーションによって評価した．シミュレーションは，C++言語で記述された R-Clos シミュレータに後述するアクセスパターンによるパケットを毎クロックサイクル投入することによって行った．R-Clos シミュレータは，各スイッチの動作，およびスイッチ間の転送を毎クロックレベルでシミュレート可能である．投入されたパケットはクロック毎に前進し，distributor から exchanger を経由して concentrator に転送されて送信先の PE に届けられる．シミュレーションに用いたトラヒックは，R-Clos のデータアクセス局所性能力を調べるため，宛先が完全にランダムな uniform traffic と，それぞれ全パケット中の 50%，80%が Clos 網内で閉じた転送である localized traffic において評価した．uniform traffic および localized traffic の宛先は，C 言語のライブラリ関数 random を用いて宛先を生成した．クラスタ構成基数 k は 4 で，2 階層

のとき 64 PE，3 階層では 256 PE を接続した構成 (それぞれ R-Clos(2,4) と R-Clos(3,4)) となっている．スイッチは，標準的な入力側にパケットキュー (深さは 5) をもつ入力キュー方式のものを想定した．パケットの転送は，SAF(Store And Forward) routing で行われ，パケット同士の衝突が発生した場合はスイッチ内で調停が行われ，調停に負けたパケットはパケットキューで出力が空くまで待機させられる．6.4 節で述べる，R-Clos 用のスイッチとして試作実装したチップの設計データに基づき，スイッチを通過するのに 4 サイクルかかるものとしている．比較対象の recursive $(2s-1)$-stage Clos 網においても同じ通過サイクル数と仮定しているが，これは MIN においては非同期自己ルーティング型の制御に基づいたスイッチを用いるのが一般的であり，ルーティングタグはソースノードで作成され，ルーティングの違いによる通過サイクル数の違いは起こらないと考えられるためである．

図 4.1 に示すグラフの横軸は normalized accepted traffic で，縦軸は average latency をサイクル数で表している．したがってグラフ中の曲線が垂直に立つ normalized accepted traffic がネットワークの飽和スループットを示す．

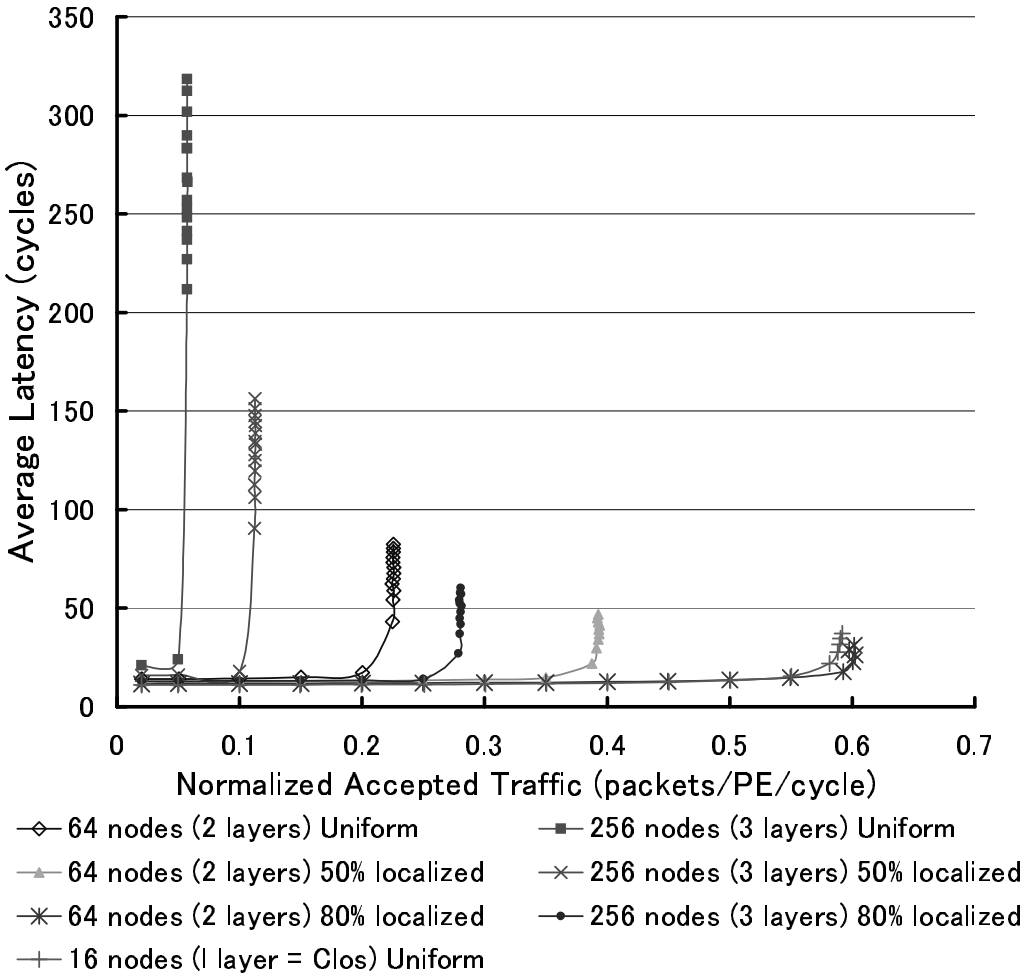


図 4.1: R-Clos 階層化による飽和スループットとレイテンシへの影響

まずはじめに，R-Clos の階層構造による転送能力の低下がどの程度あるかを図 4.1 に示した．プロットは，アクセス発行率で，64 ノードの場合では 0.02 から 0.80 まで 0.02 刻みで，256PE の場合は，0.05 から 0.80 まで 0.05 刻みで打たれている．飽和している場所では，プロットの刻み幅も密になっており，normalized accepted traffic，average latency ともに飽和していることを示している．

16PE の場合は，純粋なリアレンジブル Clos 網であり，階層化をすることによる影響を評価するための目安となっている．Clos 網では，飽和スループットがおよそ 0.6 に近いところとなるが，uniform traffic における，64PE (2 階層) の R-Clos では飽和スループットが約 0.22 程度，3 階層の 256PE ではさらに低下して約 0.06 程度で飽和している．この結果より，R-Clos 網は広域的な交信に対しては，階層間転送容量の不足からかなりの性能低下がみられることが判る．この性能低下は，階層化において，上位階層のリンクが細くなるためで，ほとんどの広域通信が，最上位階層まで上ってから転送されるため，アップリンクにアクセス集中してしまうためである．

次に, 局所性のある通信に対する評価を見ると，64PE で通信の局所性が 50%の場合は，飽和スループットが約 0.39，平均レイテンシは 50 サイクル程であり，80% となると飽和スループットは 0.6 まで向上する．さらに階層を重ねた 256PE では，通信の局所性が 50%で飽和スループットが約 0.12 で，80%となると，約 0.28 まで改善される．これは，R-Clos の基本ネットワーク部の Clos 網の転送能力を高さを示しているもので，レイテンシも，近接アクセスの割合が増加すると激減し，広域通信に比べ，およそ 1/5 のサイクル数で飽和する．

次に，同じ Clos 網を用いて拡張した MIN である recursive $(2s-1)$-stage Clos と転送性能を比較した結果を図 4.2 と図 4.3 に示した．トラヒックパターンは，uniform traffic と Clos 網内に閉じたデータ転送の局所性が 50%，65%，80%の localized traffic で行った．

64PE の場合は，通信の局所性が 50%以下ではリアレンジブル網である recursive 5-stage Clos の飽和スループットが大幅に良い結果を示すが，全体的に転送遅延は R-Clos よりも高い．通信の局所性が 65%までででは recursive 5-stage Clos に及ばないが，通信の局所性が 80%になると，局所性を考慮された R-Clos の飽和スループット及びレイテンシが recursive 5-stage Clos をいずれも上回る結果となる．また，通信の局所性に対する飽和スループットも，recursive 5-stage Clos が 50%以下では，ほとんど同様の傾向を示すのに対し，R-Clos ではいずれも大きく向上している．

256PE の場合も 64PE の場合と同様の傾向を示し，通信の局所性が 80%になると R-Clos の飽和スループットは約 0.28 近くまで向上し，レイテンシも 50 ステップ超で飽和する．一方で，recursive 7-stage Clos は，局所性 80%のときにやや飽和スループットの改善が見られる (0.02 程) 程度である．また，64 PE の場合よりも通信の局所性がより顕著に現われ，65%の局所性で R-Clos は，80%の局所性を持つトラヒックにおける recursive 7-stage Clos を 上回る飽和スループットと低いレイテンシを示す.

この結果から，大規模化がより進んだ場合には，R-Clos と recursive $(2s-1)$-stage Clos の差は縮まり，より小さい局所性で同等以上の性能を示すものと予想できる．したがって，大規模なシステムで，並列計算機のような近傍 PE との通信の局所性が存在するトラヒックで用いる場合は転送性能とハードウェア量の観点からも R-Clos は有利である．

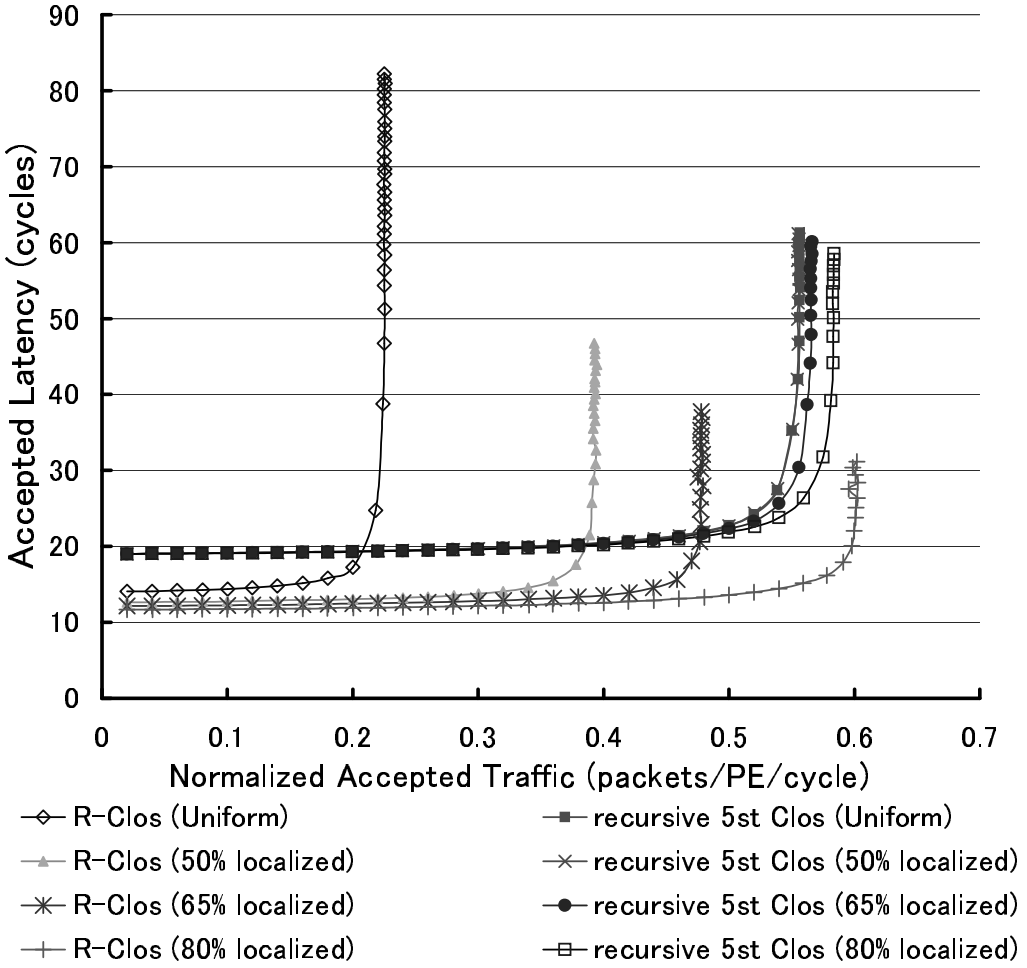


図 4.2: recursive 5-stage Clos と R-Clos の比較 (64PE)

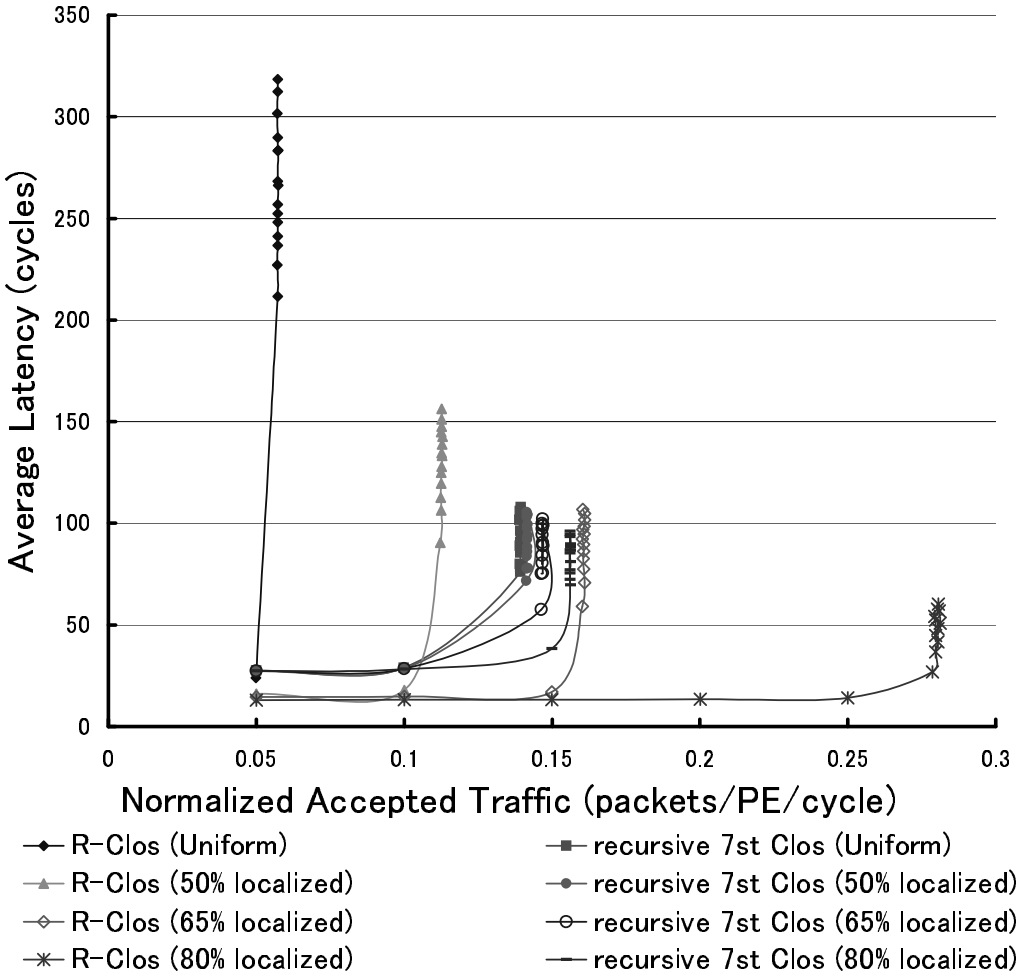


図 4.3: recursive 7-stage Clos と R-Clos の比較 (256 PE)

第5章　R-Clos におけるデータ転送スケジューリング手法

5.1　転送スケジューリングの概要

提案するスケジューリング手法は，コンパイル時に転送開始時刻と転送先があらかじめわかっているデータ転送について，コンパイラによって経路選択と転送開始時刻を調節することにより，R-Clos 網内で衝突を防ぎ，転送性能を改善することを目的とする．

R-Clos で用いる基本ネットワークの Clos 網はリアレンジブル網に分類されるため，宛先が異なるパケットについては経路選択を適切に行なえば無衝突な経路を形成可能である．そこで基本ネットワークの Clos 網に閉じたデータ転送に関しては可能な限り，衝突が起こらないように転送経路を調整して経路を形成する．

一方，拡張ネットワーク部を含めた R-Clos はブロッキング網であるため，基本ネットワーク部から外へ出るパケット同士の競合が存在する．また基本ネットワークの Clos 網から外に出るかどうかに関わらず宛先が同一のパケットについては衝突が発生するため，これらの場合にはパケットの送信時刻を調整する．

5.2　パケット転送スケジューラの構成

R-Clos 上のパケット転送をスケジュールするパケット転送スケジューラの構成について述べる．パケット転送スケジューラは図 5.1 に示すとおり，次の 3 つから構成される．

i. route arranger ： 基本ネットワーク部の転送経路を調整する．

ii. R-Clos packet tracer ： R-Clos 上のパケット転送をトレースする．

iii. access time scheduler ： パケットの送信時刻をスケジュールする．

ある PE から，あるパケットが時刻 t_{sch} に送信されるとする．送信時刻と交信相手は，コンパイラがあらかじめ予測可能と仮定する．実際は，コンパイラの予測可能性を高める設計が行なわれているマルチプロセッサであっても，すべての交信が予測可能ではないが，この点についての配慮は次章で検討することにする．

5.2.1　スケジュールの流れ

パケット転送スケジューラの全体の大まかな流れは次のとおりである．

1. スケジューラは，時刻 t_{sch} に送信される全パケットの集合 P_{sch} について，経由する最高階層数 l $(1 \leq l \leq R)$ で分類し，それぞれのパケットの集合を $P_{sch}(l)$ とする．例

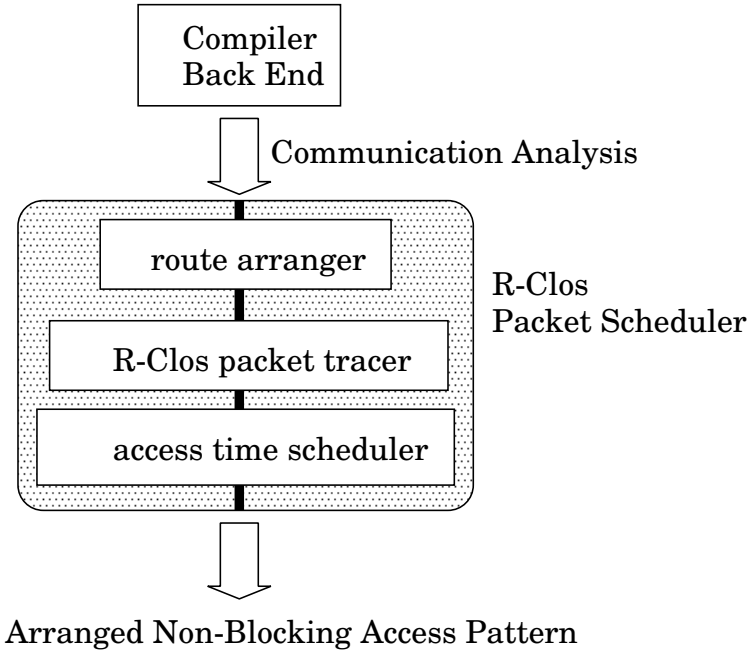


図 5.1: R-Clos パケットスケジューラの構成

えば，宛先が送信元 Clos 網内である場合は，経由する最高階層 l は 1，経由する最高階層が level-2 exchanger である場合は l は 2 である．

2. P_{sch} の全パケットについて，route arranger で経路調整を行い，経路を決定する．

3. l を 1 に定め，経由最高階層 l であるパケットに注目する．

4. $P_{sch}(l)$ の各パケットに対して，時刻 t_{sch} から時刻を 1 つずつ進めて R-Clos packet tracer 上でパケットを移動する．この際，各段の出力が複数パケットで競合する場合は調停を行い，調停の結果敗れたパケットを P_{sch} から取り去り，次の時刻に発行時刻を遅らせる．敗れたパケットの送信時刻は access time scheduler によって後にずらされる．

5. $P_{sch}(l)$ が目的地に到着したら，l を増やす．

6. l が，システムの最高階層 R 以下であれば 4 へ戻り，そうでなければ t_{sch} におけるスケジューリングを終了する．

上記のようにパケットの発行時刻について $P_{sch}(l)$ の各パケットをスケジューリングし，t_{sch} における R-Clos 上の全階層にわたるパケットの転送をスケジュールする．各階層において目的地に到着したパケットは，スケジューリング成功パケットとして，時刻 t_{sch} に発行するためのコードを生成する．

各段における出線競合の際の調停は，パケットの属性 (スケジューラによって静的にスケジュールされているかどうか，宛先が基本ネットワーク内かどうか，など) および送信

されてから経過したクロックサイクル数 (age) などで決定される．詳細については，5.2.2 節において述べる．

各時刻におけるスイッチの状態 (出力の予約状況，入力パケットの有無とパケットのヘッダ情報) は，各段における調停の際にスケジュール表を用いて管理する．スケジューラは，各時刻のスケジュール表を保持し，t_{sch} 以降のスケジュールにこれらの情報を反映させている．

5.2.2　経路調整

3.4.2 節で述べたように，R-Clos の基本ネットワーク部が $V(n,n,n)$ の Clos 網である場合，目的の concentrator に対して図 5.2 のように n 個 (図 5.2 では 4 個) の有効な経路が存在し，経路選択は 1 段目の distributor の出力選択によって定まる．したがって, スケジューラは，1 段目の distributor においてどの出力ポートを選択するかを調整することにより，経由する level-1 exchanger を選択し，目的の出力段の concentrator への経路を選択する．

まずはじめに，経路冗長性のない拡張ネットワーク部を使用する Clos 網間転送パケットの転送経路として level-1 exchanger を割当て，続いて経路冗長性のある基本ネットワーク部に閉じたローカル Clos 網内転送パケットの転送経路として空いている level-1 exchanger を割当てる．

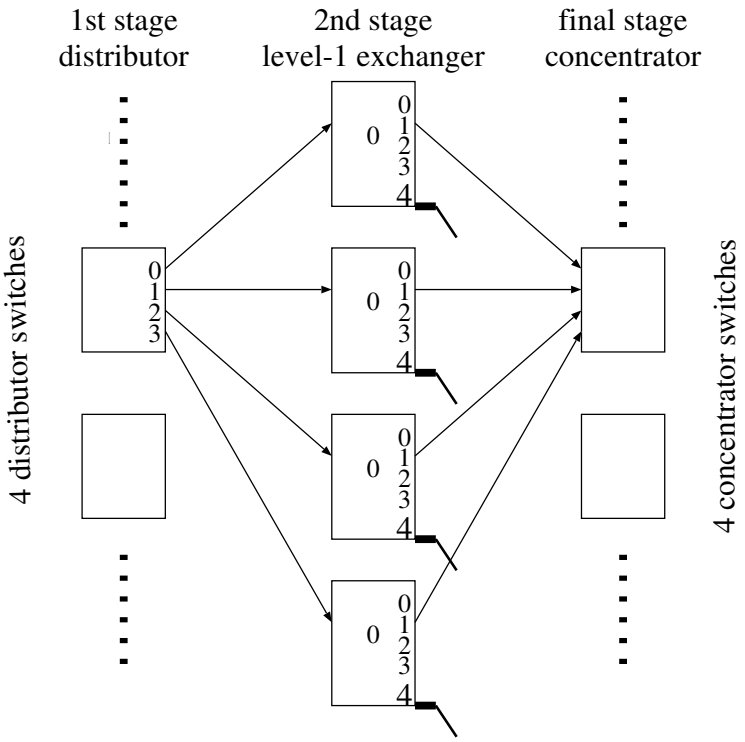


図 5.2: concentrator に対する有効な経路

(1) Clos 網外転送の経路

R-Clos を構成する Clos 網が $V(n,n,n)$ で構成されるとき，Clos 網から外へのリンクは n 本である．この n 本は，それぞれ level-1 exchanger 拡張出力ポートから，level-2 exchanger の入力に 1 対 1 接続される．拡張ネットワーク部を使用するパケットの転送は，アクセスの優先度に基づいて順番に level-1 exchanger の拡張出力の割当てによって経路が決定される．こうして形成された経路が，この後に行うローカル Clos 網内の転送経路と重複しないように，i 番目の level-1 拡張ポートを割当てられたパケットの送信元 distributor の番号を OCA(Outer Connection Array) と呼ばれる配列の添字 i の場所に格納し，ローカル Clos 網内転送の経路調整時にこれを参照する．

ルーティングの式 (3.4) から，経由する level-1 exchanger の拡張出力が競合した場合は，アクセスの優先度などから調停を行い，調停に負けたパケットの送信時刻を 1 つ後ろのクロックサイクルにずらす．また，経路の冗長性がないので，level-2 以上の階層におけるスイッチの出線競合があった場合は，access time scheduler によって送信時刻を後ろにずらして調整して衝突を回避する．

(2) Clos 網内転送に対する経路調整

Clos 網を対象として，packing strategy と呼ばれる経路再形成の方法が提案されている［YJ99］．この方法は，Clos 網においてパケットの経路形成時の中間ステージの exchanger を選ぶ手法の 1 つで，これによってノンブロッキング転送を実現する中間ステージの数 m をより小さい数で実現する．

以下に提案する方法は，packing strategy の考え方を応用したスケジュール法である．Clos 網内における衝突のすべては，2 段目の level-1 exchanger の出線競合である．これは目的とする出力段 (level-0 concentrator) への接続を要求するパケットが level-1 exchanger で複数存在する場合である．そこで route arranger は，level-1 exchanger の出力ポートを，そのスイッチに接続している distributor によって予約し，競合しないように distributor の出力 (経路選択) を調整する．

(2-a) 諸定義

t_{sch} に発行されるパケットの経路調整のために必要となるいくつかの定義を行う．

定義 1　AL_i(Access List)

Clos 網内の i 番目の distributor からのパケットの目的の出力段 (level-0 concentrator) の番号を保持したリスト

例えば，0 番目の distributor から，それぞれ $0,1,2,3$ への level-0 concentrator にデータが転送される場合は，$AL_0 = \{0,1,2,3\}$ となる．

定義 2 VPC_i(Valid Port Counter)

concentrator i に対する現在の有効な経路数を示す．level-1 exchanger の出力 i が割当てられるたびに値が 1 つ減る．すなわち，$VPC_i = 0$ である場合，i 番目の出力段 (level-0 concentrator) に接続可能な出力端子は 1 つも存在しないことを意味する．

定義 3 CT_i(Check Tag)

AL_i に対して level-1 exchanger の出力端子の予約を行ったかどうかを示すフラグ (0 は未, 1 は済を示す)

定義 4 予約行列 : Reservation Matrix(RM)

$V(m,n,r)$ で表現される Clos 網における level-1 exchanger の出力端子の予約状況を m 行 r 列で表した行列を予約行列 RM と定義する．このとき RM は，次のように表される:

$$RM = \left[RM_{i,j}\right], \quad (0 \leq i \leq m, 0 \leq j \leq r)$$

ただし，$RM_{i,j}$ は， Clos 網内の i 番目の level-1 exchanger の出力ポート j を予約した distributor の番号を示す.

以上から，AL は distributor から concentrator への接続要求の状態を表し，VPC は各 concentrator への有効経路数を示し，RM は level-1 exchanger の出力の予約状況を示すことになる．これらの情報を元に route arranger は，AL から接続要求を取得すると，VPC によって有効経路数を確認して，経路形成可能であるかを判断する．VPC が正であると実際に経路の割当てを試行し，RM によって経路割当て，すなわち level-1 exchanger の出力の予約を行う．詳細については，次節の経路形成手順にて述べる．

定義 5 出線競合検出行列 : OM

$V(m,n,r)$ で表現される Clos 網における最終段 concentrator の出力端子の予約状況を n 行 r 列で表した行列を出線競合検出行列 OM と定義する．$OM[i][j]$ は，i 番目の concentrator の j 番目の出力端子を予約しているパケットの情報を保持する．

出線競合検出行列 OM は，最終段での出線競合が起こるのを予め検出し，Clos 網内パケット転送スケジュールに出線競合を引き起こすパケットのうち最も優先度の高いものを選び，調停に負けたパケットが RM において level-1 exchanger を無駄に占有する可能性を排除する．ただし，調停の際における優先度は 5.2.2 節の AL の選択手法と同じ順番に評価が行われ，パケットが選択されるものとする．

(2-b) 経路形成手順

スケジュールは，各 distributor が順番に空いている出力ポートをスイッチ番号の小さい方から予約し，一通りの予約が行われると，全 AL が空集合になるまで同じことを繰り返す．次に詳細を示す．

アルゴリズム ： Clos 網内パケット転送スケジュール

1. ある時刻 t_{sch} に発行されるパケットのルーティングタグから AL_i を生成し，次の初期化を行う:
   - VPC_i の値を r に設定する．
   - RM の要素すべてを負の値に設定する．
   - OM の要素すべてを無効な状態に設定する．
   - 割当てを行ったかを示すチェックタグ CT_i を 0 にする．
2. 全 AL_i を走査することによって OM を作成し，衝突を検出したら調停に基づき，負けた要素を AL_i から取り除く．
3. CT_i が 1 でない，かつ空集合でない AL_i からを AL を 1 つ選択する (これを $x, (0 \leq x \leq m)$ とする).
4. AL_x から，要素を 1 つとり出す (これを y(ただし $y \neq x$) とする).
5. $VPC_y > 0$ である場合，出力ポート y が空いている level-1 exchanger を 1 つ選択 (これを $as\,(0 \leq as \leq m$ かつ $OCA_{as} \neq x)$ とする) し，$RM_{as,y}$ に x を書き込んで入力ポート x から出力ポート y への接続を予約し，CT_x を 1 にセットし，VPC_y を 1 つ減らして 7 に進む．
6. $VPC_y = 0$ であれば，既に経路はすべて埋まっているため t_{sch} における転送は不可能であるため，対応するパケットの発行時刻を遅らし 8 に進む．
7. CT_i が 1 でない $AL_{i,cs}$ が存在すれば 3 へ，そうでなければ次へ．
8. CT_i をすべて 0 にリセット $(^{\forall}i \mid 0 \leq i \leq m)$ する．
9. $AL_{i,cs}\{^{\forall}i \mid 0 \leq i \leq m\}$ が空集合であれば経路形成終了, そうでなければ 3 へ．

上記の手順でスケジュールが行われると，level-1 exchanger の出力の予約状況が VPC と RM と OCA によって管理されるため，衝突が起こることは無い．

手順 3. の対象となる AL を選択する方法と 4. のリストから要素を 1 つ取り出す順番の決定については，様々な方式が考えられる．基本ネットワークがリアレンジブルな Clos 網であるため，理論的には宛先の異なるすべての転送経路を形成可能であるため，どの順番で割当てても必ずすべての宛先の異なる経路が形成可能なはずである．しかしながら，すべての経路を形成可能とする最適解を求めるために，形成可能な経路パターンを全探索した場合，$({}_nP_n)^n$ (例えば n が 4 である場合，$({}_4P_4)^4 = 331776$) の中から無衝突に転送可能で

経路の配置を毎時刻選びだせねばならず効率が良くない．そこで AL やリストの要素を選ぶ順番について発見的手法に基づいて行うこととし，次のような項目について，値の大きいものから優先的に選択することとした．

- age ： 経路形成の対象パケットの age(最初に発行された時刻からの経過時刻) の最大値
- nodeage ： 経路形成の対象パケットの送信ノードの累積遅延
- 要求接続数 ： AL_i の要素数

上記に通常用いられる Round Robin(RR) による方法を組み合わせると，AL を選択する評価順序として有効なものは，次の方法が考えられる．

1. Nums-Age-RR ： 1) 要求接続数 2) age 3) Round Robin
2. Age-Nums-RR ： 1) age 2) 要求接続数 3) Round Robin
3. RR ： 1) Round Robin
4. Nums-NodeAge-RR : 1) 要求接続数 2) nodeage 3) Round Robin
5. NodeAge-Nums-RR : 1) nodeage 2) 要求接続数 3) Round Robin

例えば，Nums-Age-RR は，接続要求数の大きいものから選び，これが同数である場合は，次に age の大きいもの，さらに age も等しければ最後には Round Robin で選択するということを意味している．各スイッチにおいて，複数のパケットが出線競合を生じた場合の調停も，上記の評価順序で行う．すなわち，AL の選択に用いた順序と同じ順序で競合パケットを調停する．

また，対象の VPC が正であるにも関わらず，上記のアルゴリズムの割当てでは経路形成が不能である場合についても，既に形成された接続を修正することによって割当て可能となる場合がある．このために，上記の経路割当てを行った後に割当て不能なものに関して経路形成を修正するパスを追加すること (2 パス化) によって性能を改善することが可能である．

発見的手法に基づくスケジュール手法が最適解に対してどの程度近い結果となるか，また，割当ての 2 パス化の性能に対する寄与については 5.3.1 節にて詳しく述べる．

5.2.3　スケジュールアルゴリズムの計算量

対象のネットワークの全接続ノード (PE) 数を N とし，R-Clos の基本ネットワーク部の Clos 網を $V(n,n,n)$ として，提案したスケジューリング手法の最悪の場合の計算量を求める．

Clos 網 ($V(n,n,n)$) 内の転送に関するスケジューリングにおいては，出線競合の検出・排除に $O(n^2)$ 要する．さらに各 distributor の AL を作成するのに $O(n\log_2 n)$ 要し，AL を選択してポートを割当てる動作を 1 巡すると最悪 $O(n^2)$ の計算量を要する．n 巡するとすべ

てのアクセス割当てが完了するので，Clos 網内転送スケジュールに要する計算量は $O(n^3)$ となる．

一方，Clos 網外の転送についても，アルゴリズムの動作は同じであるためこの計算量以下で求めることができる．よって，R-Clos 全体でのスケジュールアルゴリズムの計算量は $O(\frac{N}{n}n^3)$ となる．

5.3　スケジューリング手法の評価

ここでは，提案したスケジューリング手法を評価する．評価した項目は次の通り:

- 衝突を回避するために生じた NOP コードによるオーバヘッド
- scheduled packet と common packet が混在するトラヒック下における R-Clos シミュレータによる転送性能

いずれの場合も，C 言語の random 関数を用いてランダムに生成されたアクセスパターンを用意し，これに対してスケジューリングを施した．評価環境の詳細については，それぞれの評価の節で述べる．

5.3.1　スケジューリングによるオーバヘッドの評価

提案したスケジューリングを施した場合，経路制御によって衝突を回避できない最終段の出線競合や拡張ネットワークにおける出線競合が発生した場合，スケジューラは，パケットの転送時刻を後ろの時刻にずらすことによって衝突を防ぐ．したがって調整された場合には，発行予定の時刻には何もパケットを送らないこと (NOP) になり，オーバヘッドとなる．ここでは経路調整が不可能で，access time scheduler によって衝突回避のために挿入される NOP(待機コード) によるオーバヘッドを評価した．

(1) 基本ネットワーク部の Clos 網内転送スケジューリングの評価

最初に Clos 網内のスケジューリング手法を次の条件で評価した．

i. ネットワークサイズは 4PE/クラスタ，16 PE(1 階層).

ii. アクセスコードの生成ステップ数は 10,000 ステップ.

用意したアクセスコードに対し，5.2.2 節であげた 5 つのヒューリスティックな手法 (Nums-Age-RR，Age-Nums-RR，RR，Nums-NodeAge-RR，NodeAge-Nums-RR) と，それぞれの手法について経路形成を 2 パス化したもの (それぞれに with Opt と表記) を適用した．また，出線競合以外にパケットが衝突しないクロスバにおけるコードステップ数を経路形成成功率 100%の場合と考え，比較対象に用いた．ここで Xbar age，Xbar nodeage はクロスバスイッチの出線競合におけるパケットの勝敗を定める優先順位を age で行うか nodeage で行うかを示している．また，まったく経路調整を行わない場合を NR として，これも比較対象とした．

評価の結果を図 5.3 に示す．グラフの横軸は，生成されるアクセスの発行率で，縦軸はスケジュール後のコードステップ数で元の 10,000 ステップからの増分が，スケジューリングによるオーバヘッドを示している．アクセスの発行率は，ネットワークの同期クロックのサイクル (これを 1 ステップ) に対するアクセスの占める率で，1.0 であればすべてのクロックにおいてアクセスが発行される．

アクセス発行率が最大の 1.0 の場合でコードステップ数は，スケジューリング前のステップ数のおよそ 1.67 倍から 1.72 倍になる．発行率が低い場合は，age か nodeage を採用し

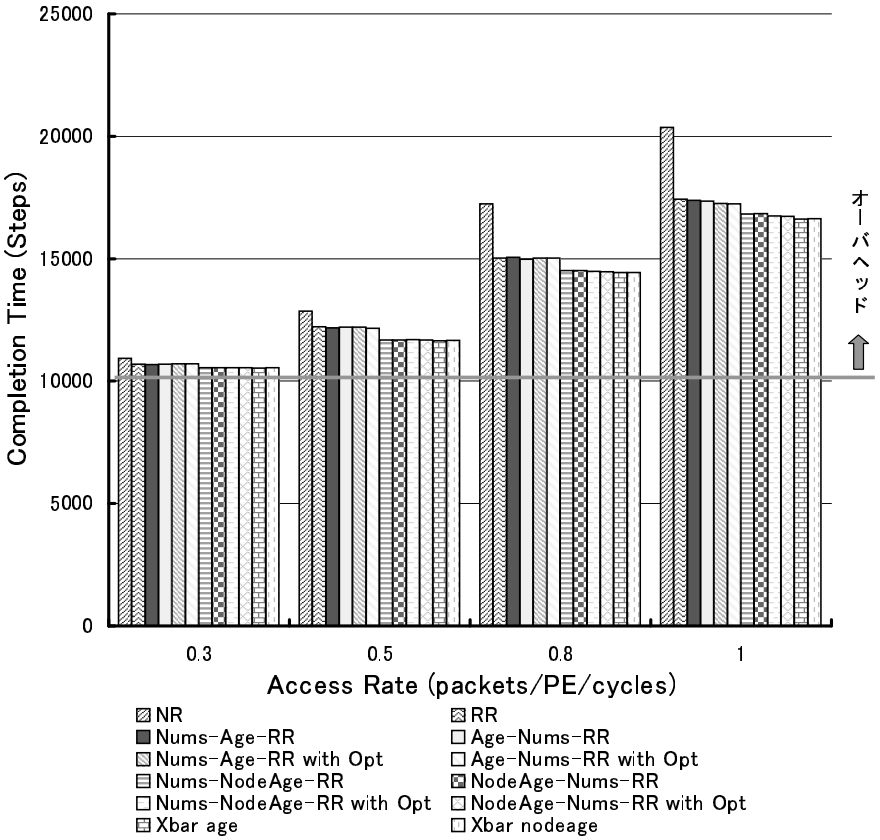


図 5.3: 1 階層 16PE における転送完了時間

ているかによって差が出ている他はほとんど差がないが，発行率が高くなると 2 パス化したものがそうでないものに対して若干性能が良くなり，RR は他に比べてオーバヘッドが大きい．また，age と nodeage のアルゴリズムの差も大きくなっている．

これをさらに詳しく解析するため，Clos 網内転送における，NOP を挿入せずに経路形成が可能であった率 (経路形成成功率) と，処理ステップ数の関係を図 5.4 に示した．アクセス発行率は 1.0 とした．図 5.4 より，経路形成成功率は，多くの方法で 99%以上とる．特に NodeAge-Nums-RR with Opt では，クロスバと近い値となり，最適解にかなり近づいている．

一方で，経路形成成功率が高くてもコードステップ数が逆に多くなってしまう現象が見られ，経路形成成功率がコードステップ数に必ずしも対応していないことが判る．すなわち，2 パス化した Nums-Age-RR with Opt と Age-Nums-RR with Opt の 2 つは，1 パスの nodeage の 2 つのアルゴリズムよりも経路形成成功率は高いが，コードステップ数は多い．本来ならば，経路形成成功率が高ければ，コードステップ数は少なくなるはずである．これについては，表 5.1 よりそれぞれの出線競合発生数を比較すると，2 パス化した age を利用した 2 つのアルゴリズムの方が圧倒的多くなっていることが判る．このことから，経路形成に失敗することによる競合よりも，もともと宛先が重複することによる出線競合によるものが多いことがうかがえる．

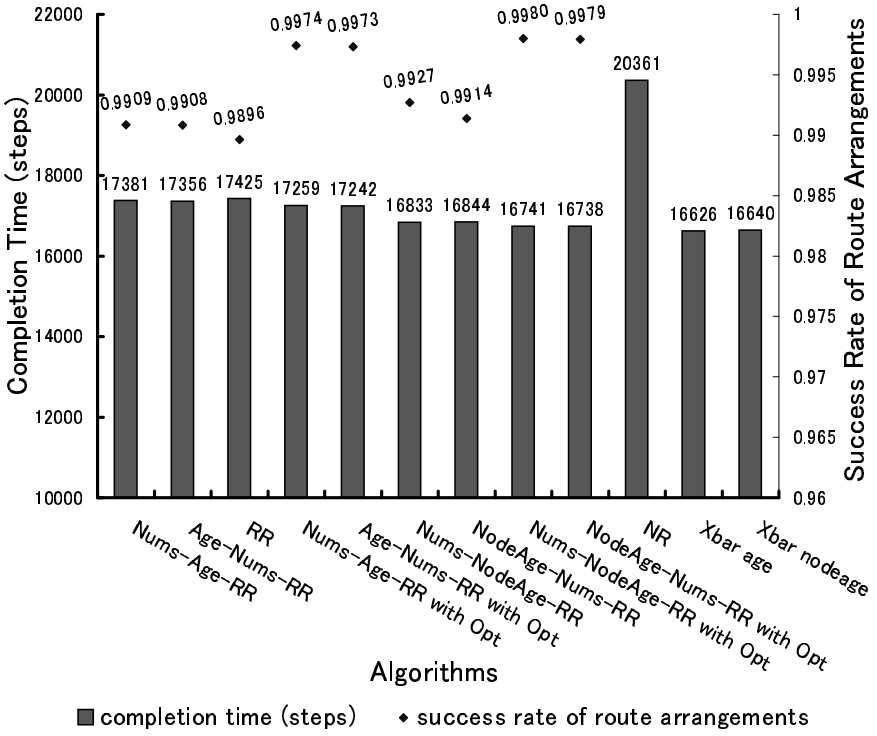


図 5.4: step あたりの経路形成成功率とコードステップ数の関係

表 5.1: 発生した出線競合アクセス数

アルゴリズム	出線競合数
Nums-Age-RR with Opt	59,045
Age-Nums-RR with Opt	59,025
Nums-NodeAge-RR with Opt	56,839
NodeAge-Nums-RR with Opt	56,980

これらの結果から，経路形成成功率よりも宛先が重複した場合の出線競合におけるアクセスの調停が，コードステップ数に影響を与えると推測した．これを裏付けるために，各アルゴリズムの各 PE の処理完了時間の分散を調べると，Nums-Age-RR with Opt で 313,556.2, Nums-NodeAge-RR で 13.9 となり高い経路形成成功率を示しても，分散が大きいほうが性能が悪いことを示す結果が出た．これより，各 PE のアクセスを均等に処理可能な nodeage の方が，age を用いるよりも効果が大きいことが裏付けられた．以上をまとめると，Clos 網内転送における経路調整に適したアルゴリズムは NodeAge を用いた NodeAge-Nums-RR with Opt か Nums-NodeAge-RR with Opt といえ，出線競合による衝突しか起こらないクロスバによるオーバヘッドに対して 0.5% 以内となり，最適解を求める計算量を考慮すると，十分な結果であると思われる．この場合，2 パス化したことによる効果は最大 0.5%程度である．

(2) 拡張ネットワーク部を含めた R-Clos 全体のデータ転送スケジューリングの評価

次に Clos 網外を含めた R-Clos でのスケジューリング手法の評価を Clos 網内転送と同じ 10 個のアルゴリズム (うち 2 パス化 5 つ) と NR について行った．評価環境を次に示す．

i. ネットワークサイズ は 4PE/クラスタ，64PE(2 階層)

ii. 用意したアクセスコードのステップ数は 10,000

iii. 有効アクセス発行率 は 0.6

iv. アクセス中の発行元 Clos 網内転送発行率は 0，0.3，0.5，0.8 の 4 種類

評価の結果を図 5.5 に示す．グラフの横軸は，アクセス中の自 Clos 網内転送の割合を示し，縦軸はアクセス完了時間を示す．すべてが Clos 網外への転送である場合はアクセス完了時間は，アクセス発行率 60%で元の約 2.81 倍になるが，Clos 網内転送の割合が増加するにつれ減少し，80%が Clos 網内転送である場合では 1.34 倍にまで減少する．アルゴリズムによるオーバヘッドの差であるが，RR が比較的多い以外はほとんど差がなく 1 階層の場合よりも差が出にくい結果となった．Clos 網内転送率が増加すると，各アルゴリズムと NR との差が大きくなるとともに，nodeage と age の差が出てくる．これは Clos 網外への転送スケジューリングがネットワークの上位階層のリンク数に依存し，パケットの発行時間をずらすことによってのみ衝突が回避できるのに対し，Clos 網内の転送では最終段での出線競合以外の衝突は，経路制御で回避できるためである．この結果から，近接アクセスの割合を高めるようにデータの配置を適切に行うことによってスケジューリングのオーバヘッドを低減できることがわかる．

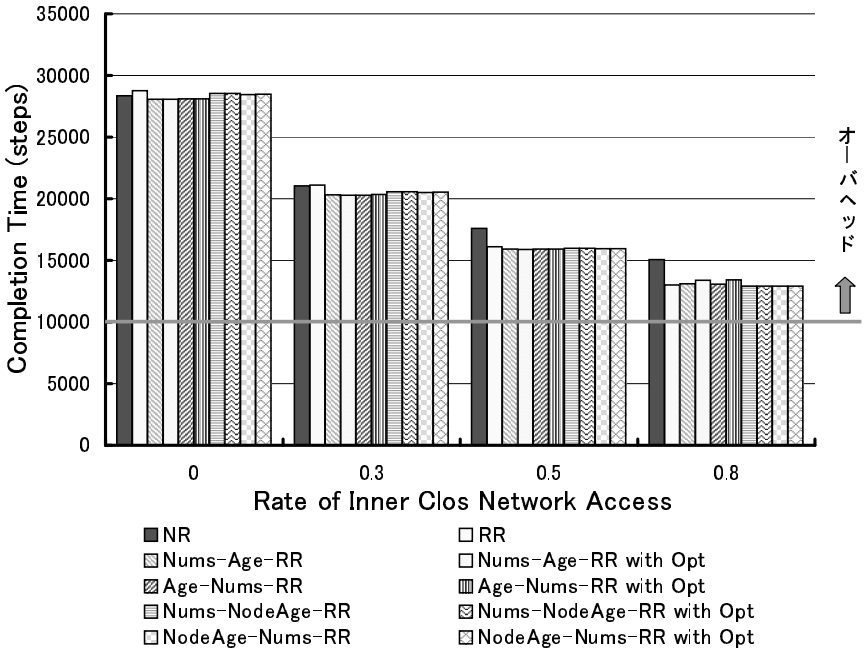


図 5.5: 2 階層 64PE における転送完了時間

5.4　スケジュールについての関連研究

MIN のスケジュールに関する研究は数多いが，これらは与えられた permutation に対してスイッチングエレメントの設定を制御することで無衝突を実現する手法である．今回提案した手法では Clos 網内のスケジュールに packing strategy［YJ99］を利用しているが，経路制御に加えてパケットの発行時刻を制御することにより衝突を回避する点で従来の研究とは異なっている．同様のアプローチで自動並列化コンパイラ・スケジューラに特化したマルチプロセッサは，Illiois 大学の Cedar［JTA+91］と早稲田大学の OSCAR［笠原 88］がある．1991 年の Illinois 大学で開発された Cedar は，クロスバをクラスタ内，Omega 網をクラスタ間の接続網として用いていたが，商用機を利用したため，接続網の正確な静的スケジューリングを行うことはできなかった．

一方、1987 年に早稲田大学で開発された OSCAR は，各々が分散共有メモリとローカルメモリを持つ 16 台のプロセッサエレメントが 3 セットの集中型共有メモリモジュールへ 3 本のバスにより結合された分散／集中共有メモリ型マルチプロセッサシステムである．OSCAR は，マルチグレイン並列処理用並列計算機アーキテクチャのプロトタイプとして，十分な機能と性能を持っており，マルチグレイン自動並列化コンパイラを備え，様々なアプリケーションが実装された［小幡 98］． OSCAR では，コード生成ルーチンが静的スケジューリングの結果を用いデータ転送タイミング，バスの競合による遅延などを推定し，各命令の実行タイミングをクロック単位でスケジュールする．

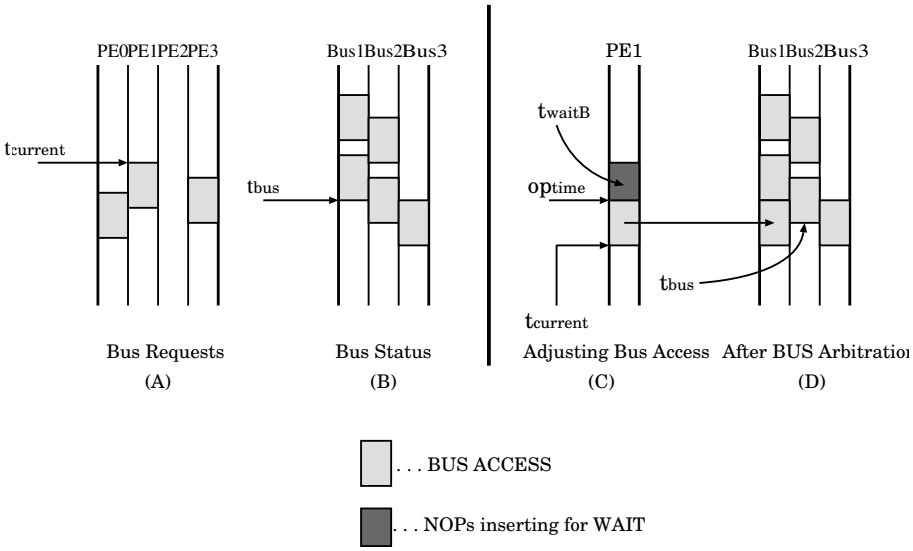


図 5.6: バスアクセスの調整

ここでは，説明のために，$t_{current}$ における PE のバスアクセス要求が図 5.6 の (A) で，その時のバスの利用状況が (B) で示されるような場合を考える．コードスケジューラは最も早くバス獲得要求を出している PE1 に着目し，これにバスの割当てを試みる．$t_{current}$ からバスが最も早く開放される時刻 t_{bus} まで，PE1 には WAIT コード，すなわち NOP を挿入して (C) に示したように t_{wait} だけ待機したのち，BUS1 に割当てられる．BUS1 への割り当てが終了すると注目する時刻 $t_{current}$ を進め，コードの実行が終るまでバス割り当てを

スケジュールする．このようにバスアクセスを調整し，データ通信による同期コードを除去することによって無同期実行を行うことに成功[尾形 93]した．しかし，結合方式にバスを採用したため大規模なシステムに拡張することが困難であった．

本論文で提案する R-Clos におけるデータ転送スケジューリング手法は，MIN の並び替え能力の高さに着目し経路調整を行い，経路調整で衝突回避できなかったデータ転送について，OSCAR におけるバスアクセス調停手法をもとにアクセスタイミングを調整することにより，効率の良いスケジューリングをスケーラブルな MIN に適用することを可能とした．

第6章　MGFスイッチアーキテクチャ

一般に自動並列化コンパイラ・スケジューラが静的スケジューリングを適用可能な部分は限られており, すべての通信を完全に静的解析することは困難である．このため, ネットワーク上にはスケジューリングされた通信とスケジューリングされていない通信が混在することになる．特にマルチグレイン並列処理においては, MT 割り当て時に発生する動的なデータ転送と, 近細粒度並列処理時のコンパイラの管理の下に行われるデータ転送が, 同一のトラヒック中に混在してしまう．このため従来のスイッチの構成では, コンパイラによって静的スケジューリングされたパケットが, 動的なパケットのデータ転送によって, ブロックされてしまいスケジューリングが乱される恐れがある．そこで, 我々は, 静的スケジューリングされたデータ転送とそうでないデータ転送が混在した場合に有効な MGF(Multi channels with Greedy Forwarding) スイッチアーキテクチャを提案する．

6.1　概要

MGF スイッチアーキテクチャは, 次の 2 点を目標としている:

- 静的にスケジューリングされたデータ転送の転送遅延の保証
- 動的なデータ転送のバンド幅の確保

このために MGF スイッチは, 次のような特徴を持つ:

- スケジューリングパケット専用の通信チャネル (これを s-channel と呼ぶ) を付加し, スケジューリングされていないパケットの通信チャネル (これを c-channel と呼ぶ) と合わせて, それぞれ独立した通信路を用意する．
- 独立した通信路のためのクロスバ (計, 2 つ) を内包する．
- パケットの種別 (スケジューリングされている (これを scheduled packet と呼ぶ)/ されていない (これを common packet と呼ぶ)) をスイッチの入力側のデコーダで判別し, それぞれの通信チャネルに振り分けて転送する．
- 出力部では, 2 つの通信チャネルを調停してどちらかを出力させるが, s-channel にパケットが存在する場合は, そちらが必ず優先される．

図 6.1 に, これらの特徴を持つ MGF スイッチの概念図を示す.

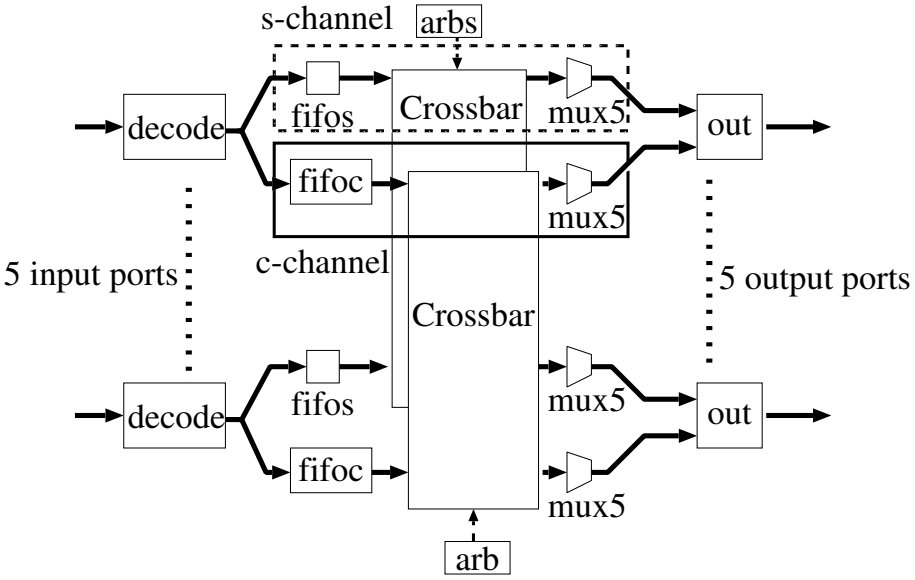


図 6.1: MGF スイッチ の概念図

次に, MGF スイッチアーキテクチャ におけるパケットの流れを示す.

i. 入力ポートに届いたパケットは, その種類 (scheduled/common) によってデマルチプレクサによって, 適切な入力チャネルバッファに送信される.

ii. 入力チャネルに入ったパケットは queue の先頭にくると, ルーティングタグに示された出力に送信される.

iii. 出力側で, そのポートを要求しているパケットの調停が行なわれ, 勝ったパケットがマルチプレクサから出力バッファに送出される.

iv. もし, s-channel の出力バッファにパケットが存在すれば, それが出力ポートに送出される.

v. c-channel の出力バッファにパケットが存在し, かつ s-channel の出力バッファが空であれば, それが出力ポートに送出される.

このように MGF では, scheduled packet の転送を優先させつつも, 出力ポートが空いていれば, 積極的に common packet も転送させていく効率の良い, 高スループットな転送を実現する. 加えて, スケジュールされたパケットが無衝突であることを利用してバッファおよび調停機構を排除する一方, スケジュールされていないパケットを任意のクロックでカットスルーできるように設計されている点が他の仮想チャネルを持つスイッチとは異なる点である.

6.2 MGF スイッチのモジュール構成

MGF スイッチは, decode, fifo, arb, mux, out の 5 つから構成される (図 6.2). 各モジュールの機能について簡単に説明する. ここでは, スイッチのサイズは 5 入力 5 出力を想定している.

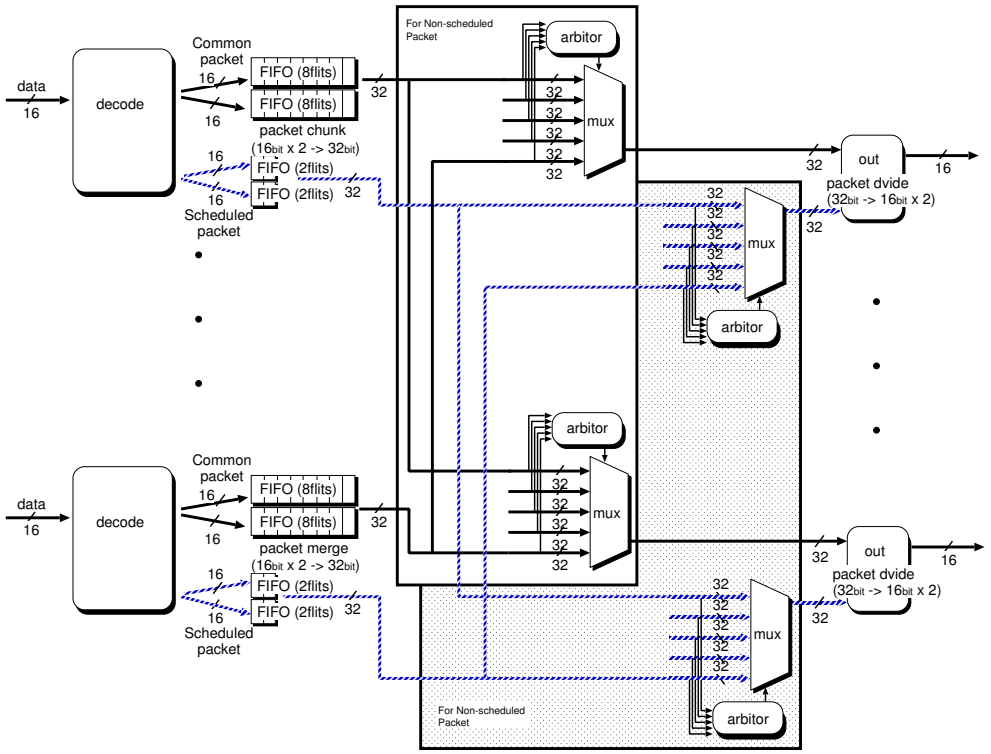


図 6.2: MGF スイッチ の構成図

6.2.1 decode

入力されたパケットのヘッダを解析する. ヘッダには scheduled packet か common packet かが示されており, これに基づいて通過させる通信チャネルを, それぞれ s-channel あるいは c-channel に振り分ける. 振り分けられたパケットは, それぞれパケットバッファに送られる.

6.2.2 fifo

decode によって s-channel と c-channel に振り分けられたパケットが一時的に格納される先入れ先だしキュー (FIFO: First In First Out queue) のパケットバッファである. common packet の場合, クロスバの調停によって, 要求した出力先が獲得できないパケットはこのバッファに格納され, 出力先が獲得できるまでこちらの fifo に格納されることになる. 一方, コンパイラなどで予めスケジューリングされ, 出線競合することのない scheduled packet

はブロックされることがないため, fifo のサイズは 1 packet 分のみで, 調停時に一時的に保持するためのレジスタで十分である.

6.2.3 arb

クロスバの調停器のモジュールで, fifo の先頭のパケットの要求出力先について調停し, 調停によって出力先を獲得したパケットは, 要求した出力先の mux(multiplexor) モジュールに送られる.

6.2.4 mux

5 入力 1 出力の multiplexor で, 5 つの入力ポートの fifo モジュールからのパケットを取り出し, arb の信号にしたがって out モジュールに出力する.

6.2.5 out

s-channel および c-channel の 2 つのクロスバから転送されたパケットの出力バッファで, s-channel のパケットが存在すれば優先的に出力される. c-channel のパケットは s-channel の出力バッファが空にならないと出力されない. このように出力バッファ側でブロックすることにより, c-channel のパケットも前進できる可能性が大きくなり, スループットの向上に役立つ.

6.3 パケットフォーマット

ここでは, MGF スイッチで取り扱うパケットについて述べる. MGF スイッチのパケットは, 次の

- scheduled packet
- common packet
  - CSM (read, forward)
  - CSM (read, backward)
  - CSM (write)
  - Peer to Peer (write)
  - broadcast/multicast

の 6 種類がある. 大きな分類では, コンパイラによって静的にアクセスが解析されたパケットで, スケジューリングされたものである scheduled packet と実行時不確定性によって解析できなかった common packet に分けられる。common packet はさらにその行き先, 用途によって上記のように 4 つに分類される. ここでは, 各パケットに共通である事項について述べる. スイッチ間のデータ幅は 16bit であると仮定し, 1flit を 16bit で構成する. パケッ

ト長は scheduled packet や broadcast/ multicast packet では 4 フリット (16×4 ビット) で固定で，それ以外については可変長となっている．先頭 2flit がヘッダである．MGF スイッチでは，2flit 単位で処理されるために, 2flit 毎にデータかヘッダかを区別するための field がある．

共通部分のフォーマットに関して説明する。

表 6.1: パケットフォーマット詳細 (共通部)

記号	bit 数	内容および定義
PT	1bit	パケット種類を示す scheduled packet は 1 common packet は 0
FT	1bit	このフリットと次のフリットの 32bit 分のタイプを示す ヘッダの場合 (先頭の flit) には 1 データの場合にはには 0 となる

次の節からは, 各パケットについての詳細について説明する．

6.3.1 scheduled packet

scheduled packet(図 6.3) は, マルチグレイン自動並列化コンパイラによって近細粒度並列処理が適用される MT 内の PE 間の交信を, 5.2.2 節で紹介したスケジューリング手法で無衝突に転送するように調整されたパケットである．したがって，近細粒度並列処理で交信範囲は MT を割り当てられたクラスタ内に限られるため，今回の MGF スイッチでは基本ネットワーク部である Clos 網内に限定した．このパケットは, 決められた転送遅延で転送先に届けられることが保証されなければならないため, すべての種類のパケットの中で最も優先度が高いパケットである．

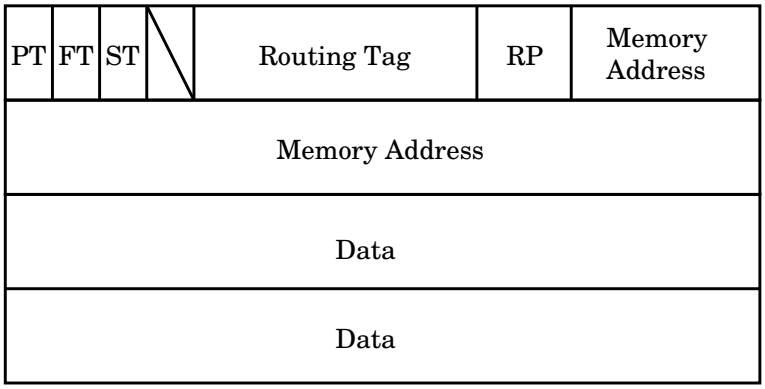
この種類に分類されるパケットは, 書き込まれる宛先によって 2 つ存在する．

1 つは, 送信先 PE の交信用メモリに対する書込みパケットでもう 1 つは, ASCA の PE である MAPLE の特徴の一つであるレシーブレジスタ転送機構による送信先 PE 内のレシーブレジスタに書き込むためのパケットである．レシーブレジスタ転送機構は，近細粒度並列化処理の場合にネックとなる交信のオーバヘッドを減らすため，メモリを介さず直接送信先のプロセッサチップ内のレジスタ (受信用) に書き込むことでダイレクトにパイプラインに必要なデータを供給するメカニズムである．

各 bit 毎の説明は, 表 6.2, 6.3 に示すとおりである．

a) scheduled packet (For inner Clos Network)

-1. To DSM



-2. To Receive Register

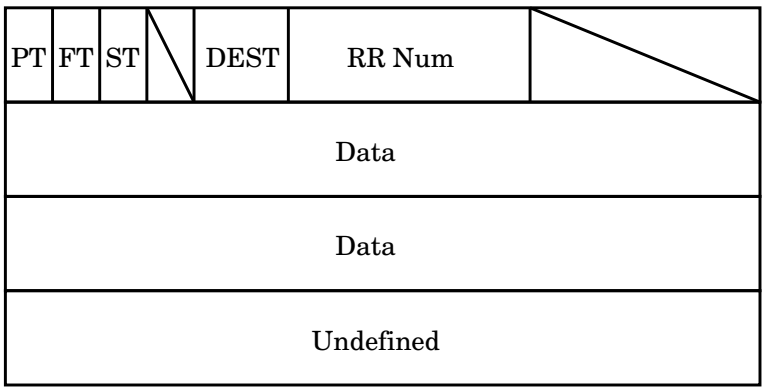


図 6.3: scheduled packet フォーマット

表 6.2: パケットフォーマット詳細 (for 交信メモリ)

記号	bit 数	内容および定義
ST	1bit	交信用メモリ通信なのかレシーブレジスタ間通信なのかを区別 交信用メモリ通信で 1 となる
RP	2bit	ルーティングポインタ 通過したスイッチの数が 00 から 11 で記録されている
Routing Tag	6bit	ルーティングタグ パケットのルーティング経路が記録 00 から 11 の 2bit で 1 階層分, 全部で 3 階層分の記録

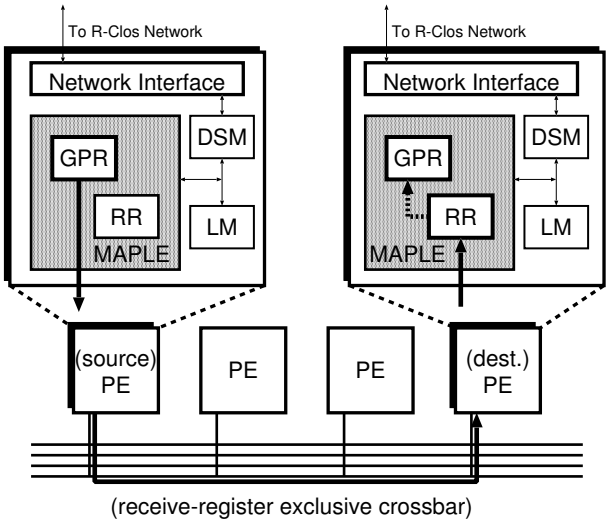


図 6.4: レシーブレジスタ転送機構概念図

表 6.3: パケットフォーマット詳細 (Revive Register)

記号	bit 数	内容および定義
ST	1bit	交信用メモリ通信なのかレシーブレジスタ間通信なのかを区別 レシーブレジスタ間通信で 0 となる
DEST	2bit	目的のレシーブレジスタの接続されている port 00 から 11 まで

パケットは Clos 網内を転送されるため, distributor から level-1 exchanger を経由して, 出力段の concentrator に送られる.

6.3.2 CSM に対する読み書き

CSM に対する通信は, 次の 3 つがある:

1. CSM への読出し要求
2. CSM からの PE に向けた 読出しデータ転送
3. CSM への書込みデータ転送

1. の Read 要求は, CSM のどのアドレスからどれだけデータを取ってくるかという要求パケットであるためパケットサイズは小さい. しかし CSM から PE に対する読出しデータの転送および, CSM へのデータの書込みはデータ自体が流れるために flit 数も大変多いものになる. また CSM との通信は, PE から CSM へ向かっていくデータの流れと, CSM から PE へと向かうデータの流れの 2 つの流れが存在する. PE から CSM へと向かうデー

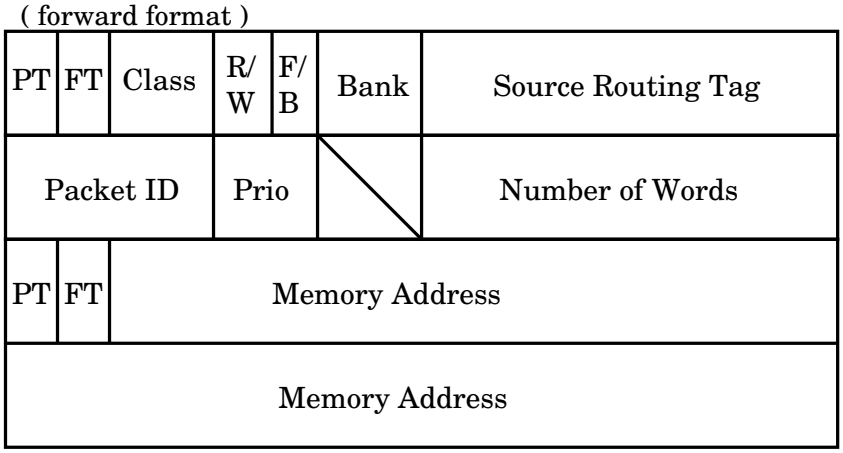
タの流れを up stream, CSM から PE へと向かうデータの流れを down stream と呼ぶことにすると, CSM へのデータ要求はデータ量の少ない up stream, CSM から PE へ向けたデータはデータ量の多い down stream, CSM へのデータの書込みはデータ量の多い up stream としてそれぞれ違う特徴を持っていることになる.

そのためこれらを 3 つに分類してパケットのフォーマットを図 6.5, 6.6 のように定義した.

b). CSM packet

-1. read

(forward format)



(backward format)

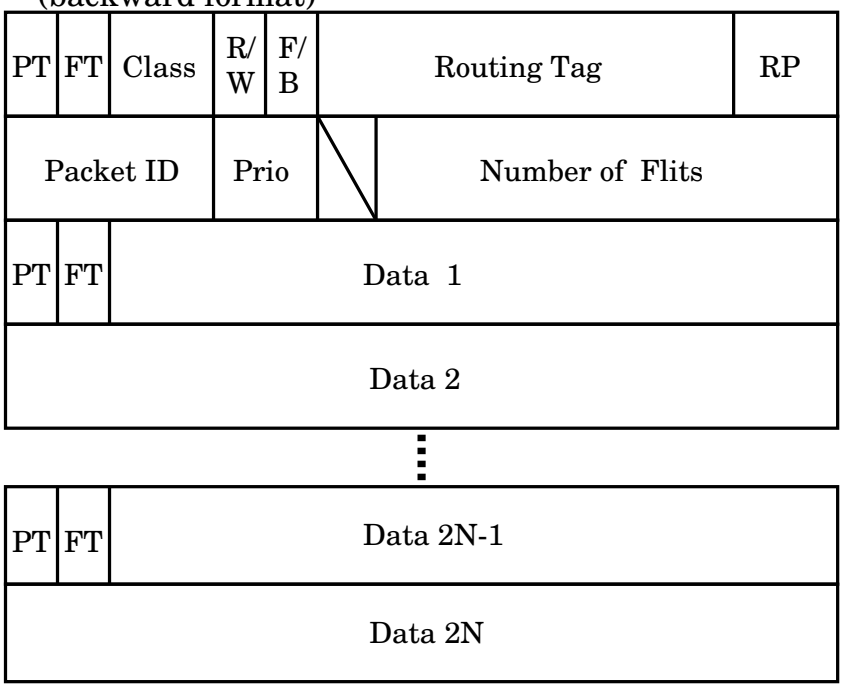


図 6.5: CSM リードパケットフォーマット

表 6.4: パケットフォーマット詳細 (CSM packet - Read backward)

記号	bit 数	内容および定義
Class	2bit	common packet (3 種類) の区別 CSM との通信で 01 となる
R/W	1bit	読出しと書込みの区別 読出しなので 0 となる
F/B	1bit	up stream(forward) と down stream(backward) の区別 down stream(backward) なので 1 となる
Routing Tag	8bit	ルーティングタグ パケットのルーティング経路が記録 00 から 11 の 2bit で 1 階層分、全部で 4 階層分の記録
RP	2bit	ルーティングポインタ 通過したスイッチの数が 00 から 11 で記録されている
Prio	2bit	パケットの優先度 00 から 11 までの 4 つのいずれか 00 が優先度が高い
Number of Flits	9bit	このフリットのデータの長さが記録されている Data 長だけが記録されている 2 から 512 までの偶数

表 6.5: パケットフォーマット (CSM packet - Write)

記号	bit 数	内容および定義
Class	2bit	common packet (3 種類) の区別 CSM との通信で 01 となる
R/W	1bit	読出しと書込みの区別 読出しなので 1 となる
Bank	2bit	目的の CSM のバンク 00 から 11 までの 4 つのいずれか
Prio	2bit	パケットの優先度 00 から 11 までの 4 つのいずれか 00 が優先度が高い
Number of Flits	9bit	このフリットのデータの長さが記録されている Data 長だけが記録されている 2 から 512 までの偶数

-2. write

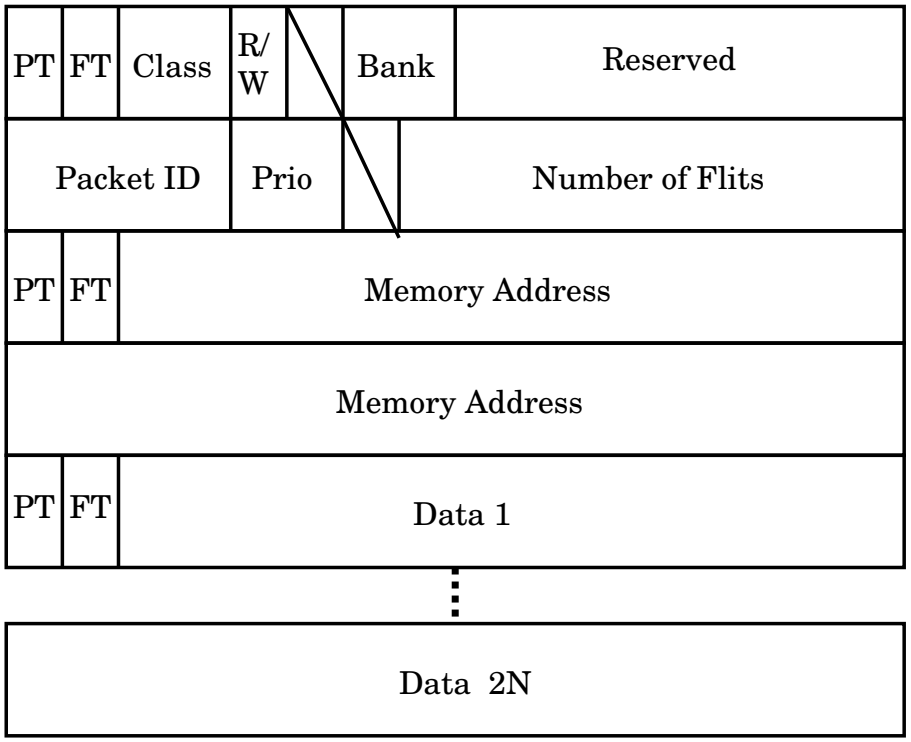


図 6.6: CSM ライトパケットフォーマット

6.3.3 P2P(Peer to Peer) 通信 (書込み)

このパケットは, 直接リモートの PE の交信用メモリにデータを書き込む場合に用いられる．このパケットは, 相手側 PE に到達するために, CSM パケットの read と同じような構成になっている．異なる点は, CSM パケットが R-Clos の最上位階層までルーティングされるのに対し, P2P パケットは, 相手側の PE と同じクラスタに所属される階層までルーティングされる点である．

c). P2P packet (write to another PE's DSM)

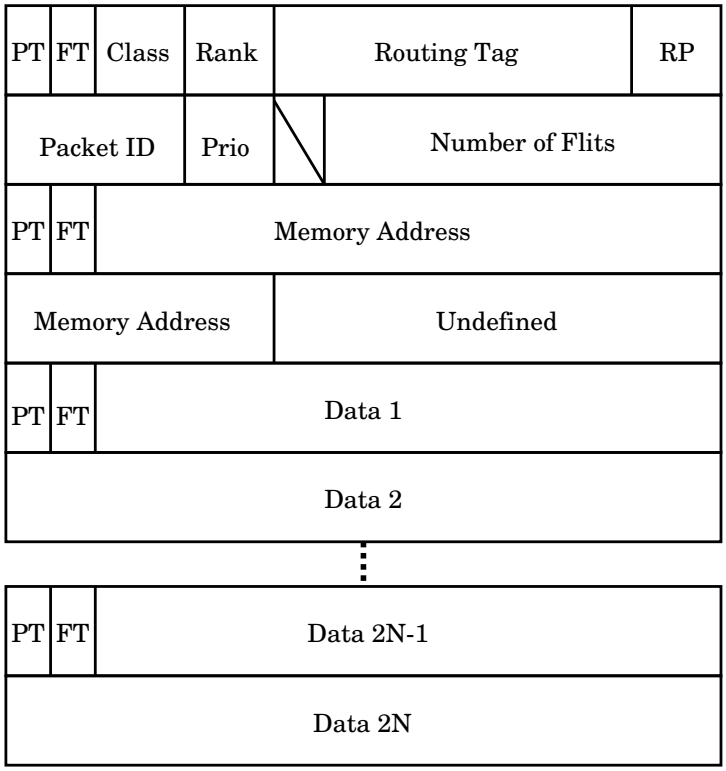


図 6.7: P2P 通信パケットフォーマット

表 6.6: パケットフォーマット (P2P パケット)

記号	bit 数	内容および定義
Class	2bit	common packet (3 種類) の区別 Peer To Peer の通信で 10 となる
Rank	2bit	目的の exchanger の階層 00 から 11 までの 3 つのいずれか
Routing Tag	8bit	ルーティングタグ パケットのルーティング経路を記録 00 から 11 の 2bit で 1 階層分、合計 4 階層分を記録
RP	2bit	ルーティングポインタ 通過したスイッチの数が 00 から 11 で記録されている
Prio	2bit	パケットの優先度 00 から 11 までの 4 つのいずれか 00 が優先度が高い
Number of Flits	9bit	このフリットのデータの長さを記録 2 から 512 までの偶数

6.3.4 ブロードキャスト/マルチキャスト通信

このパケットは, ブロードキャストもしくはマルチキャスト用のパケットである．クラスタ内で同期を取る場合などに用いられる．flit 長は 2 flit で固定である．R-Clos は階層 Tree 構造であるため, どの階層までのマルチキャストするかを指定することによって, マルチキャストする範囲を柔軟に設定することが可能である．

d). broadcast/multicast packet

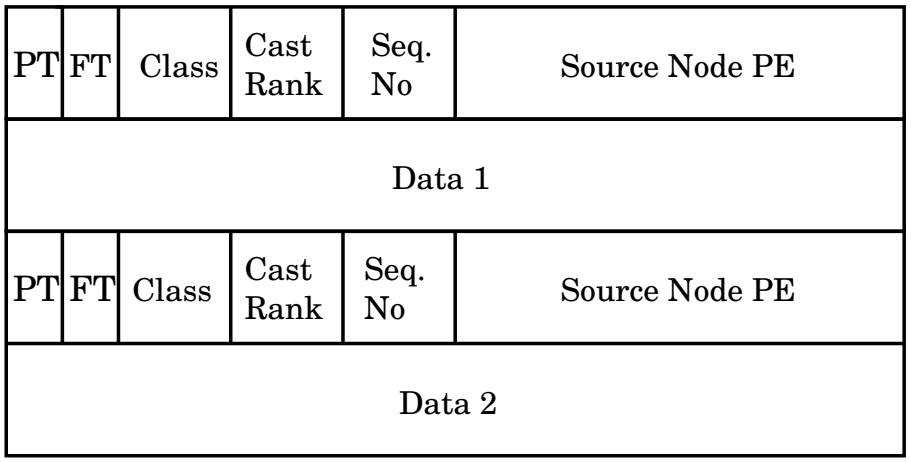


図 6.8: ブロードキャスト/マルチキャスト通信パケットフォーマット

表 6.7: パケットフォーマット (broadcast/multicast packet)

記号	bit 数	内容および定義
Class	2bit	common packet (3 種類) の区別 ブロードキャスト/マルチキャスト通信で 11 となる
Seq.No	2bit	パケット毎につけられている番号 00 から 11 までの 2bit
Source Node PE	8bit	ルーティングタグ パケットのルーティング経路を記録 00 から 11 の 2bit で 1 階層分を記録し, 合計 4 階層分を記録
Cast Rank	2bit	ブロードキャストを行う exchanger の階層を示す 01 から 11 までの 3 つ階層のいずれかを指定する

6.4 MGF スイッチの実装

MGF スイッチのハードウェア量と動作速度の評価を行なうために、図 6.2 の内部構成を持つ 5 入力 5 出力のスイッチを試作した。実装は表 6.8 に示す日立製作所のゲートアレイ (VDEC のサポートによる) を用いて行なった。

論理合成を行った結果を表 6.9 に示す．表中の BC は, 論理合成を行ったツールである Synopsys 社の Design Compiler における単位で, セル数を表す．日立製作所のゲートアレイの場合, 1 Cell = 2 NAND と換算できるので, ゲート数はこの値を倍にした 96334 で である．

実装した MGF スイッチの仕様を表 6.10 に示す．今回は利用できるライブラリの制限から, s-channel の FIFO の深さは 2 で, c-channel の FIFO の深さは 8 である．回路規模は約 10 万ゲートに収まる程度で, 動作周波数は約 50MHz で動作可能である．

配置配線後のチップのレイアウトは, 図 6.9 に示した．

表 6.8: 利用したチップのテクノロジ

プロセス	0.35μm, PolySi : 1 層 メタル配線 : 5 層
実効ゲート長	0.25μm
電源電圧	3.3 V
チップサイズ	5.9 mm 角
信号ピン数	190
下地ゲート数	143 kG(2 NAND 換算)
パッケージ	BGA256

表 6.9: 論理合成後の回路規模

RAM モジュールなし	43,487 BC
RAM モジュール	4,680 BC
合計	48,167 BC

表 6.10: 実装した MGF スイッチの仕様

ゲート数	96334 (48167 BC)
動作周波数	51.46 Mhz
入出力 bit 幅	16 bits
サイズ	5 入力 5 出力
転送サイクル (クロック) 数	4

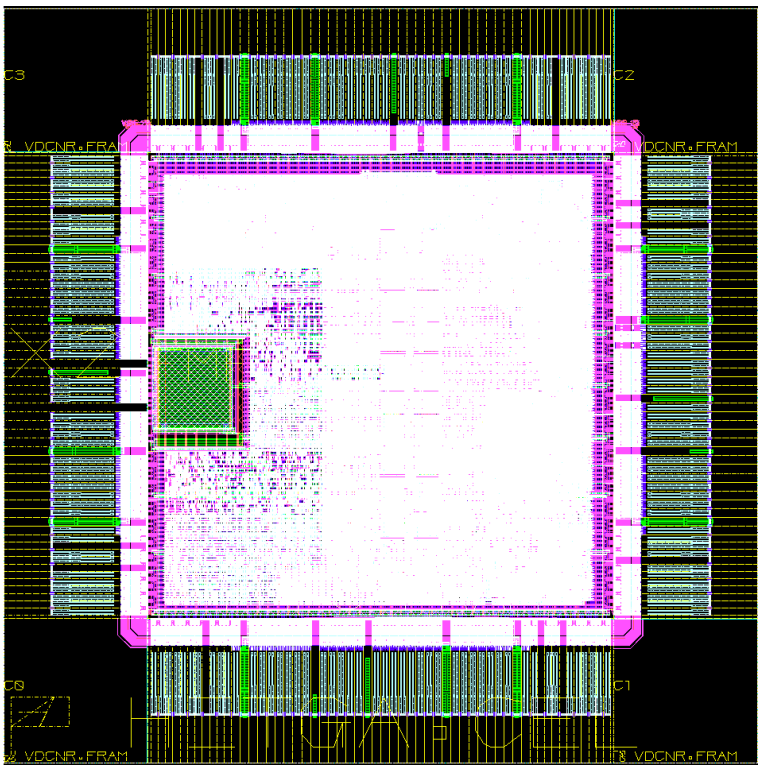


図 6.9: MGF スイッチチップ レイアウト図

6.5 MGF スイッチの評価

実装した MGF スイッチ の転送性能評価を Verilog-HDL に基づくシミュレーションによって行った．比較のために，通常のリンクあたり単一のパケットキューをもつ single channel のスイッチ (以下，SINGLE と表記) についても同様の評価条件で評価した．

6.5.1 評価条件

- 対象は，スイッチ 1 個 (つまりクロスバスイッチ単体)
- 生成したパケットの行き先のポートはランダムに決定
- パケットフォーマットもランダムに決定
- パケット長は 4flit で固定
- 設定したアクセス発行率にしたがってパケットを発行

6.5.2 結果: 転送遅延

図 6.10 に，アクセス発行率と全パケットのの平均転送遅延時間をグラフ化したものを示す．比較のために，SINGLE のスイッチの評価も載せている．MGF スイッチアーキテクチャを使用した場合には，SINGLE のスイッチを使用した場合と比較して，遅延時間は最大で 60%，平均約 70%に抑えられる．

また，スケジューリングパケットの遅延時間は，アクセス発行率が変化し，全体の遅延時間が増加しても，ほぼ横ばいに推移していることがわかる．スケジューリングパケットの微少な増加は，出線競合のために生じているものである．実際に ASCA システムで使用される場合には，このパケットはスケジューリングが施されるために，衝突することはなく遅延時間は一定で，最大で SINGLE の遅延の約 30%程度に抑えられる．他方で，スケジューリングされていないパケットの遅延時間も，MGF では転送経路が 2 つ独立して存在しているために SINGLE よりも最大で 60%程の遅延に抑えられている．

よって，このスイッチアーキテクチャは，優先する必要のあるスケジューリングがなされている近細粒度のパケットは遅延時間がほぼ一定であり，優先されないパケットも効率よく転送されることが示された．

次に，SINGLE のスイッチと MGF スイッチのハードウェア量の比較を合成後のゲート数によって評価した．この結果を表 6.11 に示した．MGF switch は，一般的な構成の最も単純なスイッチである SINGLE の約 1.4 倍程度のハードウェア量で実装することが可能であることがわかった．

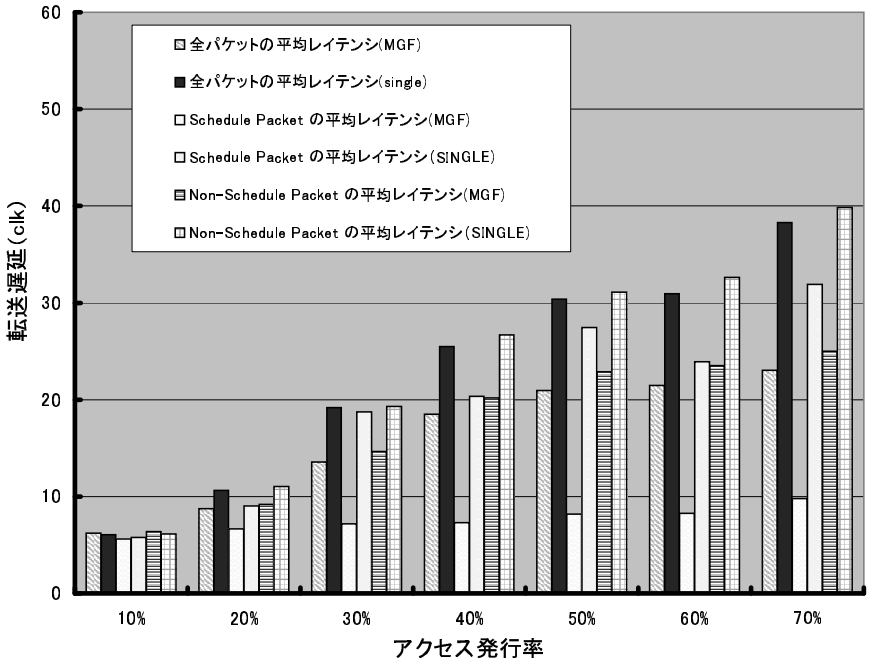


図 6.10: スイッチチップの転送遅延時間

表 6.11: ハードウェア量の比較 (合成後ゲート数)

SINGLE	MGF	MGF/SINGLE
34,594	48,167	1.39

6.6　スケジューリング手法と MGF スイッチを組み合わせた評価

6.6.1　スケジューリング手法の効果

ここでは実際に R-Clos シミュレータに scheduled packet と common packet を流し込み，アクセス完了時刻によって提案したスケジューリング手法の効果を検証する．利用した R-Clos シミュレータは C++言語によって記述され，クロックサイクル毎のスイッチ内のパケットの動作をシミュレート可能である．R-Clos シミュレータを構成するスイッチのモデルは，6 章で示した MGF スイッチと，ポートあたり単一の通信チャネルしか持たないスイッチ (ここでは IQSW と呼ぶ) の 2 つを用意し，MGF スイッチアーキテクチャの効果を評価した．ただし，MGF スイッチは実装したチップと等価の動作をするもので，R-Clos シミュレータと同様に C++ 言語で記述されている．想定したスイッチの詳細は次の通りである．

- スイッチサイズは 4 × 4 を基本とし，exchanger や concentrator は R-Clos の構成に従って拡張ポートを付加

- 実装したスイッチの仕様に基づき，通過に要するステップ数は 4
- 1 ラインのビット幅はパケットと同じと想定
- 入力側のパケットキューの深さは 5 で，MGF スイッチの s-channel では深さは 1，c-channel では 5

ネットワークサイズは，2 階層 64PE の R-Clos である．まずスケジュール対象のアクセスパターンとして (1) 60%のアクセス発行率 (2) そのうちの 50%が発行元 Clos 網内転送であるものを用意し，そのままのアクセスコードと，これにスケジュールを施したアクセスコードを用意した．さらにスケジュールの対象にならないアクセスパターンとスケジュールされたアクセスパターンが混在した場合の転送性能を評価するために，スケジュール後のアクセスコードにある確率でランダムに動的生成するアクセス (スケジューリングなし，すなわち common packet) を加えた．NOP を静的解析不能でスケジュールできないアクセスとして発行する確率を 0，0.3，0.5，0.8 のそれぞれ 4 つの場合について評価した．ここでのスケジュール手法は Nums-Age-RR とした．

図 6.11 は，スケジュール後のコードステップ数についての比較である．グラフの横軸は，ランダムに生成するアクセスの発行率である．動的生成アクセスが 0 である場合は，スケジューリングをしない場合が最もよく，この次にスケジューリングあり (MGF)，スケジューリングあり (IQSW) と続く．これは，スケジューリングによるオーバヘッドの方がスケジューリングをしないで転送した場合の衝突による遅延よりも大きかったことを示している．これはスケジューリングなしで衝突した場合，衝突パケットは衝突したステージに留まることができるのに対し，スケジューリング過程で衝突した場合は，次の時刻に最初の段から再送されてしまうためである．動的アクセスが増加するにつれ，スケジューリングなしとスケジューリングあり (IQSW) のアクセス完了時刻は増加していくが，スケジューリングあり (MGF) では，スケジューリングは乱されることなく規定時間内に転送が完了する．これによって，スケジューリングを行なっても，IQSW のようなスケジューリングしたトラヒックとそうでないトラヒックを区別できないスイッチでは効果が発揮できず，MGF スイッチのような構成が必要である．

次に，スケジューリングなし (IQSW)，スケジューリングあり (IQSW) 及びスケジューリングあり (MGF) の転送性能を調べるため，ステップ当りに受理されるパケット数を求め，図 6.12 に示した．この評価でもやはりスケジューリングあり (MGF) が，他の 2 つが動的生成アクセスの増加にともない処理率が低下するのに対して，安定して高い処理率を維持していることが見てとれる．

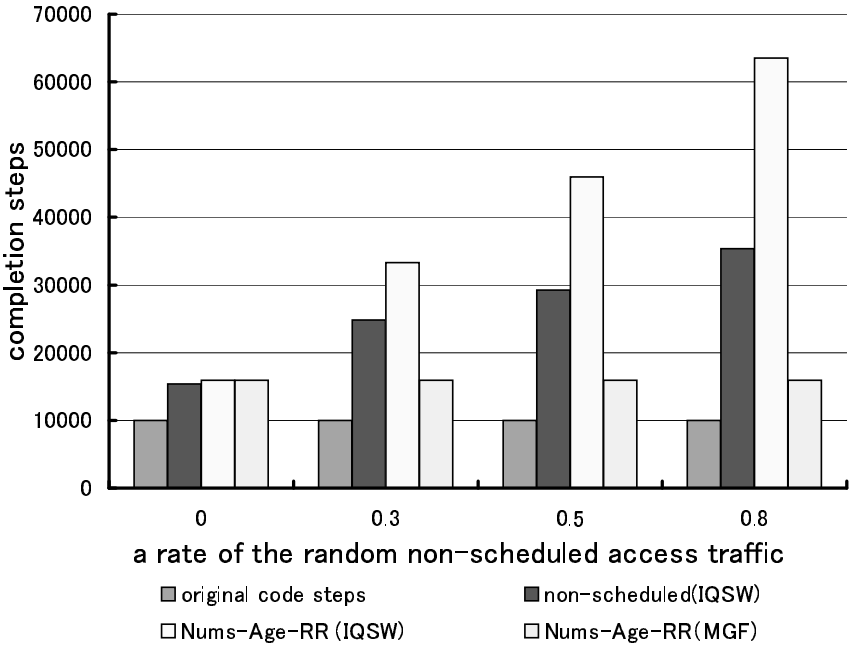


図 6.11: スケジュール後のコード量 (ステップ数) の比較

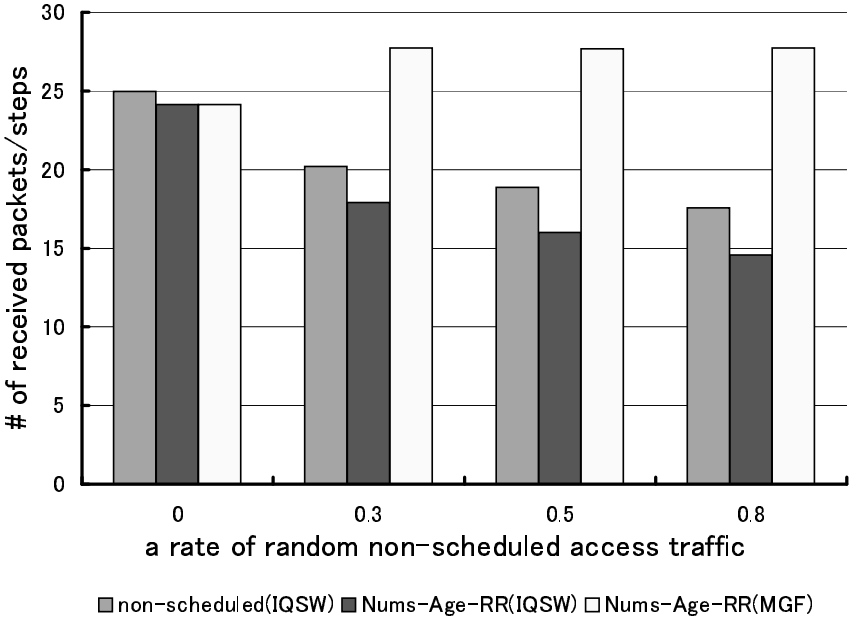


図 6.12: ステップあたりのパケット受理数

6.7 関連研究

ここでは，これまでに提案・実装された，MGF スイッチと類似のスイッチアーキテクチャをとるスイッチについて調査し，MGF スイッチとの相違について考察する．

まず，一般的なスイッチの構成について述べ，仮想チャネル (Virtual Channel) と呼ばれる，物理的な入/出力リンクに対して複数のバッファをスイッチ内に持つことによって，スイッチのスループットを向上させる仮想チャネル型のスイッチアーキテクチャについて採り上げる．

6.7.1 一般的な多段結合網のスイッチの構成

一般的な多段結合網に用いられるスイッチの構成は，図 6.13 のように，クロスバスイッチ，出線競合調停用のアービタ，そしてパケットバッファから成る．パケットバッファは，出線競合などでパケットが衝突した場合に，パケットを格納するために用いられ，FIFO(First In First Out) の形をとる．

パケットバッファの位置は，入力側につける入力キュー方式と出力側につける出力キュー方式があり，出力キュー方式が優れている [MMS87] ことが知られている．しかしながら最近になって効率良く処理をパイプライン化することにより入力キューでも変わらない性能を示すことが可能であること [RCD02] が知られはじめた．図 6.13 は，入力キュー方式のものを示している．まずはじめに，入力キュー方式について述べ，次に output queue 方式を述べる．

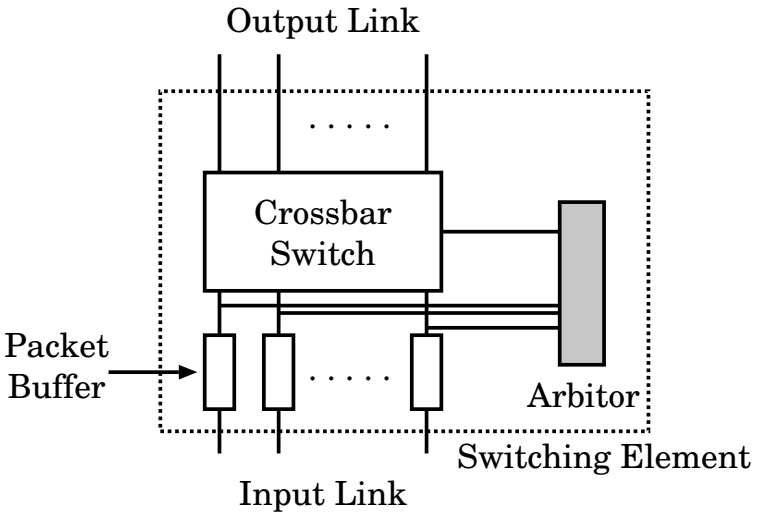


図 6.13: 一般的なスイッチの構成 - 入力キュー -

6.7.2 入力キュー方式

入力キュー方式は，キューの構成の仕方で, 次の 2 つに分けることができる．

- 各入力ポートに対して single buffer で構成するもの
- 各入力ポートに対して multiple buffer で構成するもの

入力キューによるスイッチは，ブロックされたパケットを格納するための single queue, もしくは multiple queue を持つ．

single queue による方式では，衝突によってブロックされたパケットは，行き先に関わらず，この queue に格納される．そして queue の先頭にあるパケットのみが，転送される．仮に先頭パケットが要求した出力ポートが，busy である場合, これに続くパケットもブロックされてしまう (これを HOL(head of line blocking) と呼ぶ) という欠点がある．このため，この方法によるスループットはおよそ 60% 当りで飽和してしまう［MMS87］．

1 つの input buffer に multiple queue を実現する方式は，図 6.14 に示したように，1 つの入力ポートに各出力ポートに対応した queue を用意する．このため single queue のような HOL のオーバヘッドはない．これらの multiple queue は, 普通は静的に配置されるが，不均一のトラヒックに対しては，バッファの浪費につながり効率が悪い．そこで，トラフィックに応じて queue のスペースを動的に割り当てる，DAMQ［YG96］(dynamically allocated multi-queue) が提案された．DAMQ では，各 queue の先頭のパケットのみが送信される権利を有するが，同じポートのこれらの queue のうちの 1 つしか同時に送信することができない．それゆえ 1 つの input buffer につき 1 つのデータ出力となる．仮に，ある 1 つの入力ポートのみが複数の出力ポートに対するパケットを持っている場合は，ある出力ポートがこの input buffer に対するアクセスが完了して，パケットの送信が終わるまで，他の出力ポートはアイドル状態で待ち続けなければならない．さらに，同時にある input buffer から 1 つの出力ポートのみがパケットを読み出せるようにする複雑なハードウェア機構と，input buffer が要求している出力ポートを衝突しないように各 input buffer に割り当てるための集中型アービタの機構が必要となる．

この他に，図 6.15 のように，各入力ポートに各出力に対応した複数の input buffer を持つ方式があり，理論的にこの方式 (crosspoint queue という［MPA95］) は 100%のスループットを示すが，入力ポートに関連づけられたバッファスペースは，動的に共有されないために，ある出力ポートが頻繁に使用されるようなトラヒックに対しては, 逆にその出力ポートに割り当てられたスペースに制限されたパフォーマンスになってしまうという問題点がある．これを改善させた方法として，出力ポートとは関連づけられてない k 個 (スイッチの入出力数と同じ数) の input buffer を各入力ポートに持たせる方法がある．この方法では，パケットは最も収納パケットが少ないバッファに格納される．この場合，クロスバが crosspoint queue や DAMQ よりも複雑な機構となるが，その変わりに multiple buffer によって同時に複数のアクセスが可能な上に，バッファスペースの割当てが crosspoint queue よりも自由度が高いため，頻繁に使用される出力ポートに多くのバッファスペースを割り当てられるという利点がある．

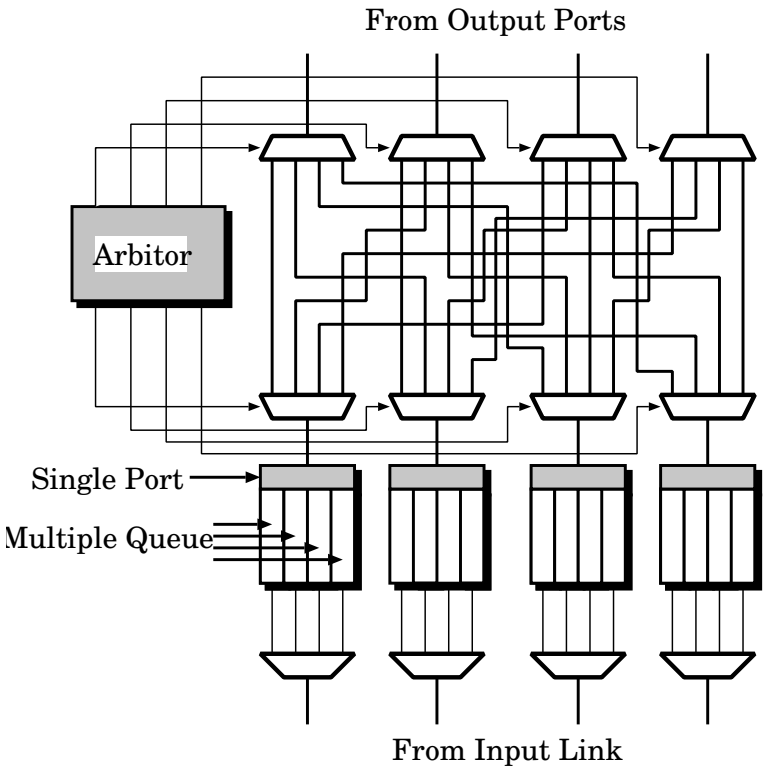


図 6.14: multiple queue を持つ入力キュー方式

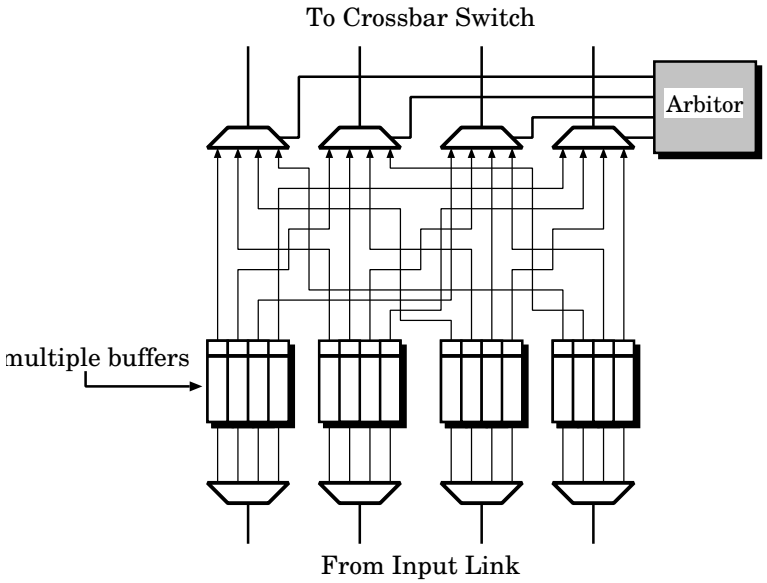


図 6.15: multiple buffer を持つ入力キュー方式

6.7.3 出力キュー方式

出力キュー方式 は，各出力ポートにそれぞれ output buffer を持たせる方式と，全出力ポートで 1 つの output buffer を共有する方式に分類できる．後者の方式は，前者よりも良いパフォーマンスを示し，またハードウェアも簡単に実装可能であり，最も一般的である［C.P94］．以降は，後者の方式について述べる．この方式は，単一のバッファ内に複数のキューを持つ方式であるので，同時に読出しもしくは，書込みによるアクセスが 1 回のみ可能であるためオーバヘッドが大きい．バッファが単一のモジュールで実装されている場合は，このオーバヘッドをなくし，毎サイクルごとに各ポートがアクセスできるようにする方法は，次の 2 つがある．

まず 1 つめは，メモリを k(スイッチの入力 (出力) 数)flit の幅のブロックとして扱い，すべての読出し，書込みは k flit 幅の chunk と呼ばれる単位で行なわれる．各ポートは，少なくとも k cycle に 1 回，chunk 単位に読出しと書込みをすることができる．ただし，バッファに書く前に各入力ポートからフリットを k flit にまとめる作業と，読出し前に k-flit の chunk をシリアル化する作業が必要となる．この結果，毎サイクルごとに 1 flit ずつ処理するのと同じスループットを得ることが可能となる．このような手法は，IBM の SP2［CDBe95］に用いられている．

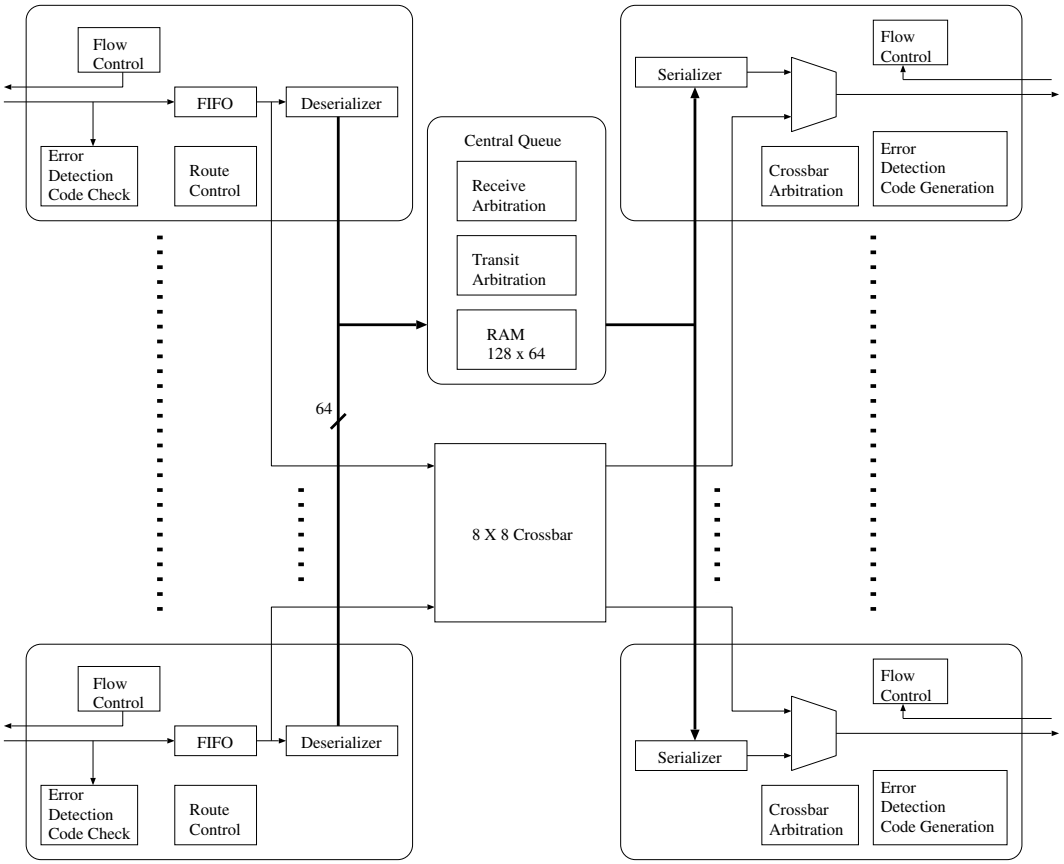


図 6.16: SP2 の Vulcan Switch Chip

もう 1 つの手法は，メモリモジュールを複数のバンクに分け，1 つのバンクを k flit 幅で構成する代わりに，モジュールを k-flit 幅のバンクとして実装する．そして，1 度に k flit のデータを読み書きするのではなく，bank 0 から bank $k-1$ まで 1 cycle ずつ順番に，パイプライン化して読み書きを行なう．これにより，ある入力ポートが bank 1 に書き込んでいる間に，他の入力ポートが bank 0 に書き始めることが可能である．この手法は，先の k flit 幅の大きいメモリを使用する方法とほとんど同じパフォーマンスを示すが，その実装は，より簡単に済む．

このように仮想チャネルを利用することによりスイッチのスループットを高める技術は多いが，パケットの優先度に応じて任意のクロックでカットスルー可能である点が MGF の利点である．また，MGF スイッチはコンパイラによってスケジュールされた優先度の高い scheduled packet は衝突が起こらないため，scheduled packet に対するパケットバッファおよび調停のためのアービタを持つ必要がない点で有利である．一方で，コンパイラによってスケジュールすることができない common packet に対する c-channel については，入力キュー方式や出力キュー方式の従来技術をそのまま適用することが可能であり，これを適用することによって，c-channel のスループットが向上する可能性があり，今後の検討課題である．

第7章　結 論

本論文では，並列化コンパイラの行う静的解析に基づくスケジューリングに焦点を置き，これを実現することによって並列化処理の効果を最大限に引き出すことを狙った相互結合網 R-Clos，転送遅延を保証するパケット転送スケジュール方法，および結合網を構成するスイッチアーキテクチャの３つによる統合的なネットワークアーキテクチャを提案し，それぞれ評価，考察をおこない，最後にこれらを組み合わせて統合的な性能について評価を示した．

R-Clos は，転送容量が大きい MIN の中でも特に高い並べ替え能力と大転送容量を備えた Clos 網に着目し，これに階層構造を導入した．これにより，R-Clos は近傍 PE との通信の局所性を活かしたローカルネットワークを提供するとともに，共有メモリ型並列計算機における共有メモリとの大域的な通信のためのグローバルネットワークの機能も備えている．これまで多段結合網では，局所性を持つデータ配置の利点を利用することができない点が指摘されてきたが，R-Clos はこれを克服し，他の局所性を考慮した MIN よりもさらに転送能力の高い Clos 網で高バンド幅・低レイテンシなローカル通信を実現している．これを裏付けるために簡単な確率モデルシミュレーションによって転送能力を評価したところ，システムサイズが 256PE である場合，全通信の 65%以上が近傍の Clos 網内で行われる場合では，recursive $(2s-1)$-stage Clos と同等以上の転送性能を示すことがわかった．この結果より，ソフトウェアの最適化技術の１つであるデータの局所配置を行うことにより，R-Clos は Clos 網の性質により，大幅に性能向上が可能であることを示している．このデータの局所配置テクニックと R-Clos を組み合わせた性能向上についての評価は今後の課題の 1 つである．

加えて，必要とするハードウェア量についても，他のフラットな構成の MIN はもとより，再帰拡張させた recursive $(2s-1)$-stage Clos よりも大幅に少なくてすむことが示され，規模拡張に優れていることを示した．規模拡張の容易さについては、ハードウェアコストの観点とは別に，構造的にも容易に拡張することが可能となっている．これは R-Clos が再帰的に階層化するためで，同じ構成のものを複数用意し，これを追加されたスイッチの段で相互に接続するだけでより大規模なシステムに拡張することが可能である．このように R-Clos は，柔軟にシステムサイズを変更できる．

相互結合網における転送遅延の保証のための，パケット転送スケジュール法の提案については，リアレンジブル Clos 網における中継段の接続法を交互に交換する接続を１つのスイッチに集める packing strategy［YJ99］などを参考に，経路を中継段に割り当てる 5 通りのヒューリスティックな手法を提案した．このスケジュールの背景としては，予めアクセスする時間と宛先のわかるプログラムブロックにおけるパケット転送を無衝突にスケジュールすることによって，転送遅延を低く抑え，性能に対する影響を少なくする目的があった．提案したヒューリスティックなスケジュール法を評価したところ，理想的なノンブロッキ

ング条件を満たすクロスバによる転送と比較して，0.5%以内のオーバヘッドに抑えることが可能であることを示した．

一方で，クラスタ外の Clos 網より外のアクセス経路についてのスケジュールについては改善の余地があることが分かった．ローカルの Clos 網よりも外側のアクセスについては，出線が競合した場合の調停の方式が影響を与えることが本論文の評価により明らかになったが，これを利用して，個々のアクセスについてより正確なクリティカルパス情報を付加し，これに基づいた調停を行った場合，およびタスクグラフの形状から調停における優先度を決定したりする方法などが考えられる．これらによってどの程度の改善が図れるかは今後の課題である．

相互結合網を構成するスイッチアーキテクチャである MGF スイッチは，静的解析不能なプログラム区間における，スケジュールできないアクセストラヒックが，静的解析可能なプログラム区間における，スケジュールされたアクセストラヒックを妨害しないようなスイッチ構成と転送方式を実現した．MGF スイッチは，スケジューリングされたパケット用の転送チャネルとスケジューリングされていないパケット用の転送チャネルの 2 つを独立して用意し，入口で入力されたパケットをこれらのチャネルに振り分け，出力モジュールでこれらを選択的に出力する．この MGF スイッチによって，相互結合網のスイッチ間のリンクをスケジュール用とスケジュールされていない用のトラヒックで共用させつつ，スイッチ内では，個々のスイッチングを妨害することなく効率良く転送できる．この MGF スイッチを 0.35μm のゲートアレイ上に実装したところ，単一の通信チャネルのクロスバスイッチに比べておよそ 1.4 倍のゲート数 (約 10 万ゲート) で，約 50MHz の周波数で動作する．これらの回路規模や動作周波数は, 最先端のチップなどに比べて優れているわけではないが, 0.35μ のゲートアレイで試作した点と, 実際に設計して動作させることを目的とした点を考慮すると, アーキテクチャの実現性について示すことができ，十分な成果が得られたといえる．また，回路規模や動作周波数については, 使用しているチップのプロセスや配線技術に左右されるため, 最新のチップ技術を用いれば, チップ面積をより小さく, より高速な動作周波数で動作させることは可能である．

この MGF スイッチを命令レベルのシミュレーションによって評価したところ，スケジュールされたパケットが，スケジュールされていないパケットに妨害されることなく定められた通信遅延で宛先に転送されることを確認した．

最後に，MGF スイッチで構成された結合網 R-Clos における，提案したスケジュール法の有効性を確認するために、確率モデルでランダムに宛先，発行タイミングを設定したアクセスを生成し，この一部分が静的解析可能であるとしてスケジュールを施して評価を行った．評価の結果，静的解析不能でスケジュールされていないアクセスが混在するなかでも，スケジュールが乱されることがないことが確認され，高負荷においても安定して高いスループットと低い転送遅延を保てることが確認された．今後の改良点としては，静的解析不能である場合でも，効率の良い転送を行うためのスイッチアーキテクチャについて検討し，特に MGF スイッチの c-channel については，提案されている高スループット化のテクニックを適用することで性能向上が見込まれる．

以上のことをまとめると，ソフトウェアが静的解析可能なプログラム区間に対して，提案した結合網，スケジュール手法，スイッチアーキテクチャを用いることで, コンパイラによって, プログラム中のデータ移動を静的にスケジュールすることが可能となった．これよ

り, 今回提案した結合網 R-Clos，パケット転送スケジュール方式，およびスイッチアーキテクチャの組合わせは，静的解析を利用したソフトウェア技法について有効であり，ソフトウェアのアクセスパターン解析能力が高くなるほど効果の向上が望める．今後はアプリケーションの特性解析を進め, 本論文で提案したソフトウェアの静的解析によるアプローチが効果的である場合について研究していくことが重要であると考える．

謝辞

本研究の機会を与えて下さり，7 年間に渡って絶えず御指導頂いた慶應義塾大学理工学部情報工学科 天野 英晴 教授に深く感謝いたします.

また，本研究をまとめるにあたり，本論文の草稿を丁寧に査読して頂き，有益な議論に多くの時間を割いて下さった慶應義塾大学理工学部情報工学科 山本 喜一 助教授，山崎 信行 助教授, 笹瀬 巌 教授に深謝いたします.

本研究の一部は，半導体理工学研究センター (STARC) によるものです．産業界からの立場として多くの助言を頂いた，STARC 研究推進本部長 小澤 時典 氏，上級研究員 増田 英司 氏，上級研究員 平田 雅規 氏，上級研究員 宮田 操 氏，主査 桧垣 信生 氏，客員研究員 瀧塚 博志 氏，客員研究員 近藤 佳久 氏，客員研究員 枝廣 正人 氏，客員研究員 加藤 桂史 氏，客員研究員 宮森 高 氏に感謝いたします.

本プロジェクトに共に取組んだ岩井 啓輔氏 (健在 防衛大学校), 藤原 崇 氏（現在 株式会社東芝），坂本 勝人 氏（現在 株式会社日立製作所），川口 貴弘 氏（現在 特許庁），仲村 柄 真人 氏（現在 株式会社日立ソフトウエア），小野 徹行 氏（現在 株式会社日立製作所），酒井 敦 氏（現在 株式会社東芝），鈴木 貴幸 氏（現在 リーマンブラザーズ証券），阿部 剛（現在 日本電気）氏，小川 陸（現在 株式会社東芝） 氏，鯉渕 道紘 氏，樫山 彰史 氏（現在 野村総研），安福 健太 (現在 株式会社東芝) 氏，桜澤 秀樹（現在 株式会社日立製作所）氏に心より感謝いたします.

岩井 啓輔 氏と阿部 剛 氏には，特に長い間，本研究プロジェクトでお世話になるばかりではなく, 私生活においてもサッカーの話題やワールドカップの観戦, 数々の研究会や国際会議のための旅行など，大変楽しい研究生活を送ることができました．ありがとうございました．深く感謝いたします.

鯉渕 道紘 氏には，相互結合網の専門家として数々の助言を頂きました．深く感謝いたします.

CS カップで共に闘い抜いた天野研究室および安西・山崎研究室のサッカー部のメンバーに感謝いたします.

博士課程時代に在勤していた慶應義塾大学理工学 ITC のスタッフの皆様には，コンピュータの管理全般にわたる知見を頂くとともに，充実した勤務環境の下で，研究と IT スキルの習得を両立することができました．深く感謝いたします.

本研究をさまざまな面から支えてくれた天野研究室，安西・山崎研究室，会社の同期, すべての友人に感謝いたします.

最後になりましたが，博士課程に進学すること，および好きな研究に没頭することに理解を示し, つらい時に励まし支えてくれた家族に心より感謝いたします.

2004 年 4 月 会社の寮 にて

参考文献

[C. 53] C. Clos. "A study of non-blocking switching networks". *Bell system Tech.J*, Vol. 32, pp. 406–424, Mar. 1953.

[CDBe95] C.B.Stunkel, D.G.Shea, B.Abali, and et al. "The SP2 High-Performance Switch". *IBM System Journal*, Vol. 34, No. 2, pp. 185–204, 1995.

[C.E85] C.E.Leiserson. "Fat-trees: Universal networks for hardware-efficient supercomputing". *IEEE Trans. on Computers*, Vol. C-34, No. 10, pp. 892–901, Oct. 1985.

[CMM+89] C.D.Polychronopoulus, Milind B. Girkar, Mohammad R. Haghighat, Chia L. Lee, Bruce P. Leung, and Dale A. Schouten. "Parafrase-2: An Environment for Parallelizing, Partitioning, Synchronizing, and Scheduling Programs on Multiprocessor". In *Proceedings of the International Conference on Parallel Processing*, pp. II39–48, Aug. 1989.

[C.P94] C.Partridge. "Gigabit Networking". *Addison-Wesley*, Vol. , No. , p. , 1994.

[CT80] C.Wu and T.Feng. "On a Class of Multistage Interconnection Networks". *IEEE Trans. on Computers*, Vol. C-29, No. 8, pp. 694–702, Aug. 1980.

[D.H75] D.H.Lawrie. "Access and Alignment of Data in an Array Processor". *IEEE Trans. on Computers*, Vol. C-29, No. 8, pp. 1145–1155, 1975.

[D.R92] D.Rana. "A Control algorithm for three-stage non-blocking networks". In *IEEE GLOBECOM'92, Communicate for Global users*, pp. 1477–1481, 1992.

[FWA+02] Fabrizio Petrini, Wu-chun Feng, Adolfy Hoisie, Salvador Coll, and Eitan Frachtenberg. "The Quadrics Network: High-Performance Clustering Technology". *IEEE Micro*, Vol. 22(1), pp. 46–57, January-February 2002.

[GRM+68] Gearge H.Barnes, Richard M.Brown, Masao Kato, David J.Kuck, and Daniel L.Slotnick. "The Illiac IV Computer". *IEEE Trans. on Computers*, Vol. C-17, No. 8, pp. 746–757, Aug. 1968.

[GST95] G.S.Sohi, S.E.Breach, and T.N.Vijaykumar. Multiscalar Processors. In *Proc. of the 28th ISCA*, Vol. , pp. 415–425, 1995.

[He91] H.Kadota and et al. "Parallel Computer ADENART -Its Architecture and Application-". In *Proc. of ICS91*, pp. 1–8, June 1991.

［HNS+99］ H.Saito, N.Stavrakos, S.Carroll, C.D.Polychronopoulos, and A.Nicolau. ”The Design of the PROMIS Compiler”. In *Computational Complexity*, pp. 214–228, 1999.

［HS78］ H.J.Siegel and S.D.Smith. ”Study of multistage SIMD interconnection networks”. In *ISCA'78*, pp. 223–229, 1978.

［JJ84］ J.P.Shen and J.P.Hayes. ”Fault-Tolerance of Dynamic Full Access Interconnection Networks”. *IEEE Trans. on Computers*, Vol. C-33, No. 3, pp. 241–248, Mar. 1984.

［JTA+91］ J.Konicek, T. Tilton, A. Veidenbaum, C.Q. Zhu, E.S. Davidson, R. Downing, M. Haney, M. Sharma, P.C. Yew, P.M. Farmwald, D. Kuck, D. Lavery, R. Lindsey, D. Pointer,, J. Andrews, T. Beck, T. Murphy, S. Turner, and N. Warter. ”The Organization of the Cedar System”. *Proc. of International Conference on Parallel Processing*, pp. I49–I56, Aug. 1991.

［MJS+96］ M.W.Hall, J.M.Anderson, S.P.Amarasinghe, B.R.Murphy, S.W.Liao, E.Bugnion, and M.S.Lam. ”Maximizing Multiprocessor Performance with the SUIF Compiler”. *IEEE Computer*, pp. 84–89, June 1996.

［MMS87］ M.Karol, M.Hluchkj, and S.Morgan. ”Input versus Output Queueing on a Space-Division Packet Switch”. *IEEE Trans. on Comm.*, Vol. 35, No. 12, pp. 1345–1356, Dec. 1987.

［MPA95］ M.Katevenis, P.Vatsolaki, and A.Efthymiou. ”Pipelined memory shared buffer for VLSI switches”. In *ACM SIGCOMM Conference*, pp. 39–48, Aug. 1995.

［QEJ99］ Q.Jacobson, E.Rotenberg, and J.E.Smith. Future Generation Processors:Using Hierarchy and Replication. In *Proc. of IWIA*, Vol. , pp. 59–66, 1999.

［RCD02］ R.Slvaram, C.B.Stunkel, and D.L.Panda. ”HIPIQS: A High-Performance Switch Architecture Using Input Queuing”. *IEEE Trans. on Parallel and Distributed Systems*, Vol. 13, No. 3, pp. 275–289, March 2002.

［SRe77］ Swan, R.J., and et al. ”The implementation of the CM multiprocessor”. In *NCC*, pp. 645–655, 1977.

［TK91］ Tobagi, F.A. and Kwok, T. ”The Tandem Banyan Switching Fabric: a Simple High-Performance Fast Packet Switch”. In *Proc. of IEEE INFOCOM91*, 1991.

［TS96］ T.T.Lee and S.Y.Liew. ”Parallel Routing Algorithm in benes-clos networks”. *IEEE Trans. on Computers*, Vol. , No. , pp. 279–286, 1996.

［TSe91］ Tanabe, N., Suzuoka, T., and et al. ”High Performance Interconnection Networks for Highly Parallel Computer PRODIGY”. In *Proc. of Int. Conf. on Parallel Processing*, pp. 509–516, Aug. 1991.

［YG96］ Y.Tamir and G.L.Frazier. ”Dynamically-Allocated Multi-Queue Buffers for VLSI Communicatin Switches”. *IEEE Trans. on Computers*, Vol. 41, No. 6, pp. 725–737, June 1996.

［YJ99］ Y.Yang and J.Wang. ”Wide-Sense Nonblocking Clos Networks Under Packing Strategy”. *IEEE Trans. on Computers*, Vol. 48, No. 3, pp. 265–284, Mar. 1999.

［岡本 96］ 岡本 雅巳, 合田 憲人, 吉田 明正, 笠原 博徳, 成田 誠之助. ” 実用レベルのマルチグレイン FORTRAN コンパイラの開発”. 情報処理学会研究報告 96-ARC-106, pp. 43–48, Oct. 1996.

［笠原 88］ 笠原 博徳, 成田 誠之助, 橋本 親. ”OSCAR(Optimally SCheduled Advanced multiprocessoR) のアーキテクチャ”. 信学論, Vol. J71-D, No. 8, pp. 1440–1445, August 1988.

［笠原 91］ 笠原 博徳. ” 並列処理技術”. コロナ社, 1991.

［笹原 95］ 笹原 正司, 寺田 純, 大和 純一, 塙 敏博, 天野 英晴. SSS 型 MIN に基づくマルチプロセッサ SNAIL. 情報処理学会論文誌, Vol. 36, No. 7, pp. 1640 – 1651, Jul. 1995.

［小幡 98］ 小幡 元樹, 松井 巌徹, 松崎 秀則, 木村 啓二, 稲石 大祐, 宇治川 泰史, 山本 晃正, 岡本 雅巳, 笠原 博徳. ”OSCAR FORTRAN Compiler を用いたマルチグレイン並列性の評価”. 情報処理学会研究報告 98-ARC-130-3, pp. 13–18, Aug. 1998.

［塙 敏 98］ 塙 敏博, 朱 笑岩, 亀井 貴之, 天野 英晴. ” 多次元構造を持つ MIN”. 情処学論, Vol. 39, No. 6, pp. 1672–1680, Jun. 1998.

［尾形 93］ 尾形 航, 吉田 明正, 合田 憲人, 岡本 雅巳, 笠原 博徳. ” スタティックスケジューリングを用いたマルチプロセッサシステム上の無同期近細粒度並列処理”. *JSPP ‘93*, 1993.

論文目録

本研究に関する論文

【公刊論文】

- 森村知弘, 岩井啓輔, 天野英晴：階層型多段結合網 R-Clos, 電子情報通信学会論文誌 Vol.J86-DI No.5 pp.283–pp.292 May., 2003.
- 森村 知弘, 岩井 啓輔, 天野 英晴：階層型多段結合網 R-Clos における通信スケジューリング電子情報通信学会論文誌 DI(採録決定).

【国際会議】

- T. Morimura, K. Iwai and H. Amanno: Multistage Interconnection Network Recursive Clos (R-Clos):Emulating the hierarchical multi-bus model for Multi-grain ParallelProcessing, 11'th International Conference on Parallel andDistributed Computing Systems., pp.99–pp.104 Sep., 1998.
- T. Morimura, K. Iwai and H. Amano:"Multi-Stage Interconnection Network Recursive-Clos(R-Clos)II: a scalable network for a compiler directed multiprocessor ASCA", Proc. of THE INTERNATIONAL CONFERENCE ON PARALLEL AND DISTRIBUTED PROCESSING TECHNIQUES AND APPLICATIONS (PDPTA'2001). pp.1165–pp.1171 Vol. III, 2001.
- Keisuke Iwai, T. Morimura, T. Fujiwara, K. Sakamoto, T. Kawaguchi, K. Kimura, H. Amano and H. Kasahara: Communication optimization of the interconnection network of ASCA:a multiprocessor for multi-grain parallel processing, Sixteenth IASTED International Conference APPLIED INFORMATICS - AI'98, pp.262–pp.264, Mar, 1998.
- T. Abe, K. Iwai, T. Morimura, R. Ogawa, K. Yasufuku and H. Amano：Near fine grain parallel processing using a multiprocessor with MAPLE, Proc. of International Conference APPLIED INFORMATICS(PDCN2002), pp.267–pp.272, Feb, 2002.
- T. Abe, T. Morimura, T. Suzuki, K. Tanaka, M. Koibuchi, K. Iwai, H. Amano: ASCA chip set: Key components of multiprocessor architecture for multi-grain parallel processing, Proceedings of An International Symposium on Low-Power and High-Speed chips COOL CHIPS IV pp.223–pp.247, Apr., 2001.

【国内の会議】

- 岩井 啓輔, 川口 貴裕, 森村 知弘, 酒井 敦, 阿部 剛, 天野 英晴: コンパイラ主導型マルチプロセッサ ASCA のアーキテクチャ, 2000 年記念並列処理シンポジウム JSPP2000 論文集, pp.3–pp.10, May, 2000.

【研究会ほか】

- 森村 知弘, 岩井 啓輔, 天野 英晴: 多重バスをエミュレートする多段結合網 R-Clos, 電子情報通信学会技術報告, Vol. 97, No. 225,pp. 85–92, Aug., 1997.
- 森村 知弘, 岩井 啓輔, 天野 英晴: 階層構造を持つ多段結合網 R-Clos の改良, 電子情報通信学会技術報告, Vol. 99, No. 252, pp. 79–86, Aug., 1999.
- 森村 知弘, 田中 健介, 天野 英晴, 岩井 啓輔: スケジューリングを考慮した多段結合網スイッチチップの実装, 電子情報通信学会技術報告 VLD2001-8-17 pp.43–pp.50 Vol. 101 No. 46 May, 2001.
- 森村 知弘, 岩井 啓輔, 天野 英晴：多段結合網 R-Clos における通信スケジューリング, 情報処理学会研究報告, 2003-ARC-151-4 pp.19–pp.24 Mar., 2003.
- 岩井 啓輔, 藤原 崇, 森村 知弘, 天野 英晴, 木村 啓二, 尾形 航, 笠原 博徳: マルチグレイン並列処理用マルチプロセッサシステム, 電子情報通信学会技術報告, Vol. 97, No. 225,pp. 77–84, Aug., 1997.
- 岩井 啓輔, 森村 知弘, 阿部 剛, 天野 英晴: MAPLE cluster におけるレシーブレジスタ割り当て手法, 情報処理学会研究報告, 2002-ARC-146-7, pp.19–pp.24 Feb., 2002.
- 岩井 啓輔, 阿部 剛, 森村 知弘, 天野 英晴：MAPLE 用近細粒度並列化コンパイラ, 情報処理学会研究報告, 2003-ARC-151-4 pp.19–pp.24 Mar., 2003.

本研究に関する特許出願

【国内特許】

- 森村 知弘, 天野 英晴: マルチプロセッサシステム装置, 特開 2002-259352.

その他の論文

【公刊論文】

- 岩井 啓輔, 川口 貴弘, 鈴木 貴幸, 森村知弘, 阿部剛, 天野英晴：並列計算機 ASCA の要素プロセッサによる近細粒度並列処理, 電子情報通信学会論文誌 Vol.J85-DI No. 6 pp.491–pp.502 Jun., 2002.

- 岩井 啓輔, 阿部 剛, 森村 知弘, 天野 英晴：近細粒度処理におけるレシーブレジスタ割当て手法, 電子情報通信学会論文誌 DI（採録決定）.