計算機アーキテクチャ　127－1
設　計　自　動　化　86－1
（1997．12．11）

並列計算機ネットワーク用ルータ・チップの自動設計システム

村上祥基†　朴泰祐†

並列計算機を対象としたネットワーク・トポロジやルーティング・アルゴリズムの，設計段階における有効性は，一般に何らかの簡略化されたモデルを基にした計算機シミュレーションによって示される．しかし，実装段階を想定した，より精密な性能評価を行なうためには，実際に各要素部品を設計した上で再度評価を行なう必要があり，この作業は設計のための環境や専門知識等を必要とし，容易ではない．

本稿では，汎用相互結合網シミュレータ生成システム INSPIRE をベースに，対象とするネットワークの専用ルータ・チップを自動設計するシステムを提案する．これは，ネットワーク仕様記述ファイルから，ルータ・チップの VHDL 記述を自動生成するトランスレータをベースとしている．この出力を市販の自動合成ツールに与えることにより，設計レベルでの評価を容易に行なうことが可能となる．

Design Automation System of Router Chip for Network of Parallel Computer

YOSHIKI MURAKAMI† and TAISUKE BOKU†

Generally, the effectiveness of network topologies or routing algorithms of interconnection network for parallel computers is confirmed by simple modeled computer simulation on their designing phase. For more detailed performance evaluation, however, we need detailed design of all components for the network construction. It is difficult because it requires logic circuit designing environment and skill for design description.

In this paper, we propose an automatic design system based on INSPIRE, a general purpose network simulator generating system. It is based on a translator which converts the network specification file for INSPIRE into VHDL description of the router chip for specified network. The output VHDL description can be synthesized by generally used CAD tools, and this system makes the detailed performance evaluation of the network very easy.

1. はじめに

並列計算機における処理は，各 PU (Processing Unit) がネットワークを介して互いに通信することで行なわれる．そのため並列計算機では，ネットワークの転送性能が重要な問題となってくる．このネットワークの転送性能に関わってくるネットワーク・トポロジとルーティング方式は，現在まで様々なものが提案され，主として計算機シミュレーションによってその有効性が確かめられてきた．計算機シミュレーションは仮想モデルを対象としてシミュレーションを行うため，ネットワーク内で実際にメッセージの通信を行なうルータは理想的なものを仮定する場合が多い．しかし，実際に提案するネットワーク・トポロジとルーティング方式を実現するルータを設計・評価した際に，期待した転送性能が得られない，またはハードウェア・コストを考慮すると有効とはいい難いといった場合が生じる可能性がある．そのため，提案するものの有効性を示すには，計算機シミュレーションによる評価とルータの設計レベルでの評価が必要となる．しかし，設計レベルでの評価を行なうには人的コストや設計のための専門知識，そして設計に必要な環境などが必要となり，多くの場合，そこまで踏み込んだ設計が行なわれない場合が多い．そこで本研究では，提案するネットワーク・トポロジとルーティング・アルゴリズムの計算機シミュレーションによる評価と，ルータの設計レベルでの評価を容易にとれる環境のための，ルータ・チップの自動設計システムを提案する．

我々の研究室では，並列計算機用相互結合網の性能評価ツールとして，汎用相互結合網シミュレータ生成システム INSPIRE[1)]を開発している．INSPIRE は，ネットワークの諸特性を記述したファイル (Network Description File, 以下 NDF[1)]) を与えることにより，そのファイルで定義したネットワーク専用のシミュレー

† 筑波大学 電子・情報工学系
Institute of Information Sciences and Electronics, University of Tsukuba

タを生成する．そして，このシミュレータを利用することにより，ネットワークの転送性能の評価をクロック単位で行なうことができる．本研究では，このINSPIREにおいてネットワークの諸特性を記述しているNDFを利用し，ルータ・チップを自動設計することを考える．NDFには，ネットワーク中のルータの入出力ポート数とバッファのサイズ，そしてルータにおけるパケットのルーティング・アルゴリズムの情報がNDLと呼ばれる専用言語で記述されている．そこで，これらの情報に，実際にルータ・チップを作成する上で必要なルータ間とルータ内の信号処理の情報や，バッファの制御アルゴリズムの情報などを付加することで，ルータ・チップのハードウェア記述言語による記述を生成できるようにする．本研究では，ルータ・チップの自動設計システムである「NDHL (Network Description to Hardware description Language) トランスレータ」を開発し，ターゲットとなるNDFをこのトランスレータに入力することにより，ハードウェア記述言語の一つであるVHDL[2]記述を得られるようにする．これにより，トランスレータで生成されたVHDL記述を，市販のVHDL対応の自動合成ツールに入力することで，容易に設計レベルでの評価をとることができるようになる．本研究では，VHDL対応の自動設計ツールにはSynopsys社のものを使用する．INSPIREとNDHLトランスレータを使用することにより，計算機シミュレーションによる評価と設計レベルでの評価を容易に行なうことができる．

本稿では，まず研究の背景となる並列計算機ネットワークとルーティング方式，そしてINSPIREについて述べる．次にNDHLトランスレータについて述べ，その後トランスレータによって生成されるルータの仕様と回路構成について述べる．そして最後に，トランスレータの評価と考察を行なう．

2. 並列計算機用ネットワーク

並列計算機用ネットワークは大きく直接網・間接網の2つに分類することができる．直接網はPU間が直接的に接続されているのに対し，間接網はクロスバ・スイッチ (以下XBと略) を介し間接的にPU間が接続されている．ただし，一般には直接網・間接網共に直接PUがPUまたはXBに接続されているのではなく，何らかのルータ・スイッチを介して接続されている．直接網の代表的な例としては，ハイパキューブ・ネットワークやMesh/Torusネットワーク等がある．また，間接網の代表的な例としては，Multistage Interconnection NetworkやMDX[3]等がある．

並列計算機用ネットワークの例として，間接網の1つであるハイパクロスバ・ネットワークについて説明する．図1は，$4\times4\times4$構成の3次元ハイパクロスバ・ネットワークである．n次元のハイパクロスバ・ネットワークは，PUをn次元空間に正方格子上に配置し，1次元方向に並んだ各PUをXBにより完全結合する．これにより，送受信PUのアドレスが1次元だけ異なる場合は，XBを1回通過するだけで相手PUにメッセージを送信することができる．XB間のメッセージ転送をEX (エクスチェンジャ; ルータ・スイッチに対応) において自動的に行うことにより，中継PUを介さないルーティングが実現される．

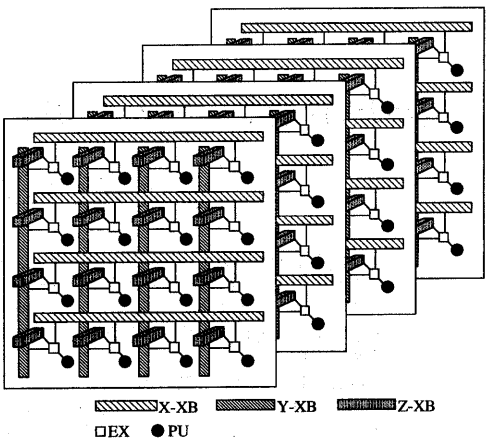

図1　3次元HXB (4×4×4)

このように間接網では，XBもネットワークの転送性能に大きく関わってくる．従って，本研究では，PUとPUまたはルータを接続しているルータをEX，そしてルータ間を接続しているルータをXBとし，EXとXBの両方のVHDL記述をNDHLトランスレータによって生成できるようにする．これにより，直接網・間接網の両者に対応できる．

3. ルーティング方式

この章では，ルーティング方式の2つの要素である転送経路の決定方式とデータ転送方式について説明する．

3.1 経路の決定方式

経路の決定方式には大別すると固定ルーティングと適応ルーティングの2つがある．

3.1.1 固定ルーティング

固定ルーティングでは，パケットの転送元PUと転送先PUの位置関係のみによって転送経路が一意に決定される．この方式の長所は，ルーティング方式が単純なためルータの論理回路が簡単なものとなり，動作周波数の高速化が容易に行なえることである．しかし，パケットの転送先に偏りがあった場合，パケット同士の衝突が頻繁に発生し転送性能が低くなってしまう．また，パケットの転送経路上に故障が生じた場合も転送経路が一意であるため故障箇所を回避できず，パケッ

ト転送が行なえなくなる.

3.1.2 適応ルーティング

適応ルーティングでは, 混雑や故障といったネットワークの状況により, 動的にパケット転送の経路が決定される. 従って, 混雑状況によって転送経路の決定を行なうことでパケット同士の衝突を回避し, 転送性能の低下を軽減することができる. また, 故障箇所を回避してパケットを転送させることで耐故障性が実現できる. この方式の欠点は, このような複雑な転送を行なうためルータの回路構成が複雑となり, 動作周波数が固定ルーティングに比べて遅くなり, 回路も大規模化するということである.

適応ルーティングを行なって転送性能の向上を行なう場合に,「先読み」と「予約」という特殊な処理が必要となる場合がある[4]. ハイパクロスバ・ネットワークなどで, EX でパケットの出力先が決定すると, XB での出力先が一意に決定されるような場合がこれにあたる. このようなルーティングの場合, 適応ルーティングの利点を活かすには, EX において次の経路を決定する際, 接続している各 XB を通して, その先に接続している EX のバッファの情報を知る必要がある. この処理を「先読み」という. また, この「先読み」の処理を行なってから, 実際にその EX のバッファにパケット・ヘッダが到達するのには何クロックか必要となるので, 到達するまでの間バッファをそのパケットが「予約」する必要がある. このように「先読み」と「予約」の処理は, パケットのルーティングを決定する際に, 次のルータを介し, もう 1 つ先のルータの情報を知る必要がある場合に行なわれる. そこで本研究では,「先読み」と「予約」のために専用の信号線と論理回路を用意して, この処理を実現している. また,「先読み」と「予約」ができるのは EX とし, XB を介してそのための情報が転送されるものとする.

3.2 データ転送方式

パケットのデータ転送方式は, 主に store&forward 方式, wormhole 方式, そして virtual cut-through 方式の 3 つの方式に分類される. この 3 つの方式の中では一般に virtual cut-through 方式が最も転送性能が高いが, この方式はバッファの構造が複雑となり, 最大動作周波数の高速化が難しい. そのため, 本研究では, バッファの構造が簡単な wormhole 方式を用いている. wormhole 方式について以下に説明する.

wormhole 方式のパケット転送を**図 2** に示す. 図 2 において, 送信 PU と受信 PU への転送経路上の中継点に置かれるいるのがルータである. この方式ではパケットのヘッダが各中継点のルータにより経路を決定し, 各ルータは到着したパケットのボディをパイプライン的に次のルータに転送する.

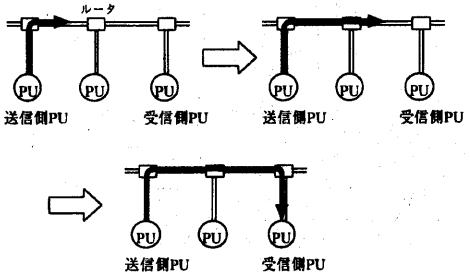

図 2 wormhole ルーティング

4. INSPIRE

汎用ネットワーク・シミュレータ生成系 INSPIRE (Interconnection Network Simulator with Programmable Interaction and Routing for performance Evaluation) の概要について説明する. INSPIRE の構成は**図 3** の様になっている. INSPIRE の本体は, シミュレータ・カーネル及びトランスレータから成る. このカーネルは C 言語により記述されている. ユーザはネットワークの諸特性, プロセッサの動作をそれぞれ NDF (Network Description File) , PBF (PU Behavior File)[1] という 2 つのファイルにより定義する. PBF は C 言語により記述され, NDF はネットワーク記述言語 NDL (Network Description Language)[1] により記述される. INSPIRE にこの NDF 及び PBF の 2 つのファイルを入力すると, トランスレータにより NDL の記述が C 言語の記述に変換され, INSPIRE のカーネルと結合されることにより, NDF と PBF における定義に従う相互結合網シミュレータが生成される. INSPIRE はタイム・スライス方式の時刻管理を採用しているため, 毎クロックごとに全 PU と全メッセージのシミュレーションが行なわれる. これによりユーザは, ネットワークの転送性能をクロック単位で評価することができる.

4.1 NDF

NDF では, ネットワーク規模, ネットワーク資源, ネットワーク・トポロジ及びルーティング・アルゴリズム等のネットワークの諸特性が定義される. NDF は, ネットワークの諸特性を記述するため INSPIRE 専用に開発されたネットワーク記述言語 NDL により記述される. この NDL の記述は, 資源記述部, 結合記述部及びルーティング記述部に分かれる.

- 資源記述部:
  PU 及びルータ等のネットワーク資源について宣言を行なう. 具体的には, PU 数とネットワーク中のルータ数, 各ルータのサイズと virtual channel の本数を宣言する.
- 結合記述部:

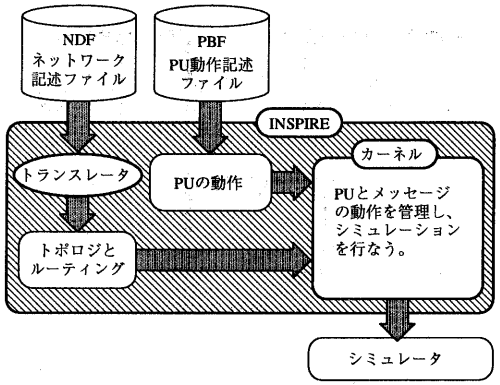

図3　INSPIREの構成

資源記述部で宣言した各資源 (PU 及びルータ) の入出力の結合を定義する. この記述を全てのネットワーク資源に対して行なうことにより, ネットワーク・トポロジが定義される.

- ルーティング記述部:
  資源記述部で宣言した各資源におけるルーティング方式を定義する. ルーティング方式は, NDL で用意されている命令と関数を使用することで定義される. NDL 上では virtual cut-through 方式や store&forward 方式も記述可能であるが, 本システムでは wormhole 方式に対応する記述のみを対象とする.

5. NDHL トランスレータ

この章では, NDHL トランスレータの概要と NDF の制約について説明する.

5.1 トランスレータの概要

NDHL トランスレータは, INSPIRE においてネットワークの諸特性を定義している NDL 記述を, そこで定義されている各ルータ (EX と XB) の VHDL 記述に変換するトランスレータである. トランスレータによって生成される VHDL 記述は, NDF で定義される各ルータに必要となる要素回路 (詳細は 6.3 節参照) の VHDL 記述と, それらを接続する各ルータの上位階層の回路の VHDL 記述で構成される. NDF から得られる情報は, 各ルータの入出力ポート数, virtual channel の本数及びルーティング・アルゴリズムの情報である. まず, トランスレータは, ルーティング・アルゴリズムの情報から, 各ルータにおいてパケットのルーティングを行なう回路の VHDL 記述を生成する. この回路にはバッファも含まれる. 次に, トランスレータは, 各ルータの入出力ポート数と virtual channel の本数の情報から, 各ルータで必要となるデータの入出力を行なう回路や, メッセージの衝突が起こった時の調停回路, そしてこれらの回路を接続する上位階層の

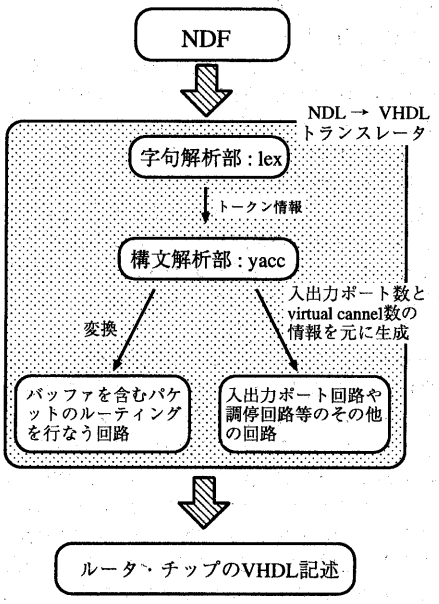

図4　NDHL トランスレータの概念図

回路の生成を行なう. また, INSPIRE で用意されている NDL は非常に自由度が高い言語のため, 記述できる全てのルータ・チップを生成するのは難しい. そこで, 本研究では並列計算機に用いられるネットワークのルータ・チップを目標とし, それに従い NDL 記述に制限を与える. この制限については, 次節で説明する.

NDHL トランスレータによって生成されるルータ・チップは転送性能の比較を第一目的としているので, 特定のアーキテクチャを想定していない. よって, アーキテクチャに依存しない, ネットワークにおいてメッセージの転送を行なうのに最低限必要な機能のみを備えている.

NDHL トランスレータの概念図を図 4 に示す. トランスレータは yacc と lex を用いて記述され, NDF を入力とする. 入力された NDF は lex で字句解析され, その字句解析した結果の情報がトークン情報として yacc に渡される. そして yacc は, 構文解析を行ない必要となるルータの情報を抽出し, NDF で定義されている各ルータ・チップの VHDL 記述が生成される.

5.2 NDL 記述の制約

変換の対象となる NDF は, 次の制限を満たすものとする.

- ネットワーク:
  - 並列計算機に用いられる直接網及び間接網
- ネットワーク・サイズ:
  - 各次元の PU 台数は 16, 次元数は 3 次元まで
- ルーティング方式:

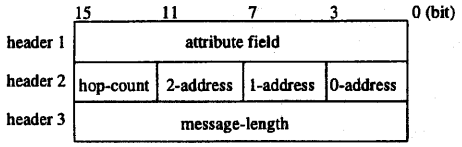

図5　パケット・ヘッダ形式

- 転送経路の決定方式: 固定ルーティング及び適応ルーティング
- データ転送方式: wormhole ルーティング
- デスティネーション・ルーティングが可能なもの

また, ルータ・チップの入出力ポート数は同じものとする.

NDL は, 計算機シミュレーションを対象として作られた言語である. そのため, 実回路上で実現するのが難しい関数などが用意されている. そこで本研究では, トランスレータで生成するルータ・チップの仕様により, NDL で使用できる関数, 命令, 演算子, そして構文を制限する. ただし, 上で述べたネットワークとルーティング方式を記述するのに必要なものは, すべて使用できる. ルータ・チップの仕様については次の章で説明する.

6. ルータ・チップの仕様

この章では, パケットの形式とルータ・チップの構成とルータのパイプライン構成, そして INSPIRE で想定しているルータとの違いについて説明する.

6.1 パケット形式

想定するパケット・ヘッダの形式を図5に示す. パケット・ヘッダは図5に示す通りヘッダ1, ヘッダ2, ヘッダ3の3 flit (1 flit = 16 bit) で構成されている. ヘッダ1は, ユーザがパケットに属性値を与えたい時に使用するための領域である. 次にヘッダ2は, デスティネーション・アドレスとパケットのホップ・カウント数 (パケットが通過したルータ数) の情報が格納されている. 図のデスティネーション・アドレスの情報は, 右から1次元目, 2次元目, 3次元目のアドレスである. 最後にヘッダ3には, パケット長が格納されている.

6.2 データ・パスとパイプライン構成

INSPIRE では, ネットワークのデータ・パスのバンド幅は自由に設定できる. しかし, 本研究ではトランスレータで出力されるルータを, 1チップで実現することを考えている. 従って, データ線のバンド幅は, パケット転送の高速化のためパケット・ヘッダのデスティネーション・アドレス情報を1クロックで処理することと, 1チップで実現した際のピン数を考慮して16 bit とする. また, クロックは2相クロックを用いる.

サイズが 4×4 で virtual channel を 2 本使用した

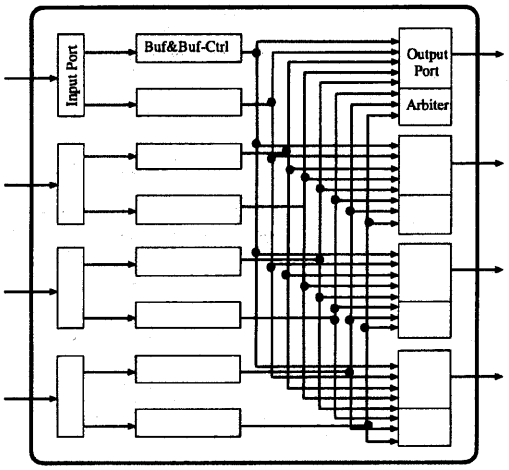

図6　ルータのデータ・パス

時のルータ・チップのデータ・パスを図6に示す. 基本的に NDHL トランスレータで生成されるルータ・チップのデータパスはこのようになるが, 「先読み」と「予約」の機能を使用する場合は, そのための専用の入出力ポートと回路が追加される. 図中の Input Port, Output Port はそれぞれ入出力回路を示し, Buf&Buf-Ctrl はバッファ&バッファ制御回路, そして Arbiter はアービタ回路を示す. ルータ・チップを構成する回路については次節で説明する.

入力ポートに送られてきたパケットは, バッファに格納される. 格納されたパケットは, バッファ制御回路においてヘッダの行き先アドレスが参照され, 対応する出力ポート回路に送られる. この時パケット間で衝突が生じた時には, 出力ポートに用意されているアービタ回路において, 各バッファに用意された優先度に対して疑似ランダム方式の優先度処理が行なわれ, 優先度の高いバッファにあるパケットが選択される. 出力ポート回路は次段のルータのバッファの状態を見て, 空いているならば選択したパケットを出力する.

このルータ・チップのパイプライン構成を図7に示す. パイプラインは3段で構成され, 1段目がバッファへのパケットの格納と出力ポートの選択のために必要となる各種レジスタの設定, 2段目が出力ポートへのポート獲得の要求, 3段目で調停と出力が行なわれる. パイプラインの各段は基本的に1クロックで処理されるが, 3段目はアービタ回路からポートの獲得要求に対する ack が返ってくるまで wait 状態になる.

n 入力 n 出力のルータ・チップの信号線の内訳は, 表1に示す通りとなっている. 表1の n_p は, 入力ポートか出力ポートにおいて PU と接続しているポートの数を示している.

6.3 ルータを構成する回路

ルータを構成する回路について説明する. まず, 基

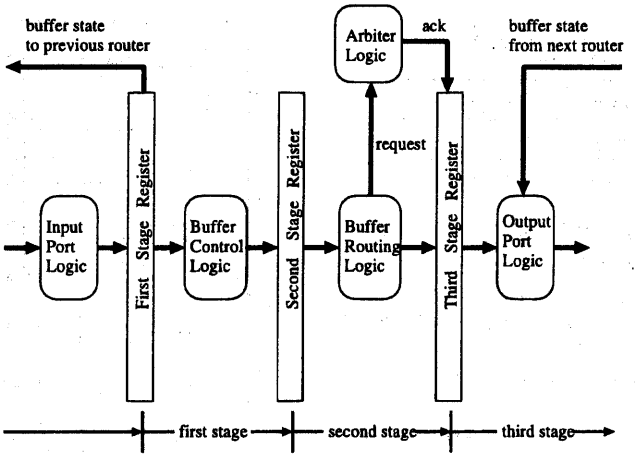

図7　ルータのパイプライン構成

表1　ルータ・チップの信号線の内訳

信号線名	信号線数
基本となる信号線	
data path	16
valid/invalid	1
clock	1
reset	1
basic_total	$34n + 2$
virtual channel 使用時の追加信号線	
vc_control	2
vc_total	$4n$
「先読み」と「予約」使用時の追加信号線	
reserve path	6
rsv_valid/invalid	1
rsv_ack	1
rsv_total (EX)	$8(n - n_p)$
rsv_total (XB)	$8n$

本となるのは次の4つの回路である.

- バッファ&バッファ制御回路
- アービタ回路
- 入力ポート回路
- 出力ポート回路

「先読み」と「予約」の機能を使用する場合に, XB では上記の回路に次の回路が加わる.

- 予約要求の処理回路

また, 基本となる回路に「先読み」と「予約」専用の機能と入出力ポートが追加される.

6.3.1　バッファ&バッファ制御回路

ルータ・チップのパイプライン構成が3ステージで構成されているので, バッファの容量は3 flit である. バッファ制御回路は, パケット・ヘッダが到着すると, 属性値, デスティネーション・アドレス及びパケット長をそれぞれレジスタに格納する. パケット長を格納するのはカウンタ・レジスタで, このレジスタの値はパケット・ヘッダに続くボディの各 flit が到着する度にデクリメントされる. そして次のクロックに, デスティネーション・アドレスの情報を基に, パケットの行く先を決める処理が行なわれる. この処理は NDF に記述されているルーティング・アルゴリズムに従って行なわれるが, EX では「先読み」と「予約」を行なうか行なわないかで処理の流れが異なる. 「先読み」と「予約」を行なわない場合は, 目的の出力ポートにポートの獲得要求を出す. この要求がアービタ回路で処理され ack が返ってきたら, バッファ制御回路はパケット転送の処理を始める. この処理はカウンタ・レジスタの値が0になるまで行なわれる. カウンタの値が0になると, この回路は次のパケット・ヘッダの到着待ち状態になる. ack が返って来ない場合は, 再度ポートの獲得要求を出すという処理を ack が返ってくるまで繰り返す. 「先読み」と「予約」を行なう場合は, ポートの獲得要求と同時に予約要求をアービタ回路に出す. そして, この2つの要求に対して ack が返ってくるまでこの処理を繰り返す.

6.3.2　アービタ回路

各出力ポートに1個, アービタ回路が置かれる. この回路は, 各バッファからの付属する出力ポートへの獲得要求を処理する. 各バッファからの獲得要求が複数来た場合は, 現在優先度が最も高いバッファからの要求について処理する. この処理はクロック周波数を上げるため, 2段のパイプラインに分けて行なわれる. まず1段目でバッファ番号によって2組に分けられた要求に対して優先度処理が行なわれる. 次の2段目で, その各組で選択された要求が処理され, その1つが選択される. この処理の結果, このバッファには ack 信号が送られ, 出力ポート回路にはこのバッファからのパスを選択するように指示が与えられる. 優先度は各アービタ回路が各々バッファに持っていて, バッファ

からの要求がない時に毎クロック, または出力ポートを獲得していたパケットの転送が終了した時にシフトし, パケット転送の偏りを防いでいる.

EX にあるアービタ回路では, 獲得要求と共に予約要求が来た場合は, ack を返したバッファの予約要求を次の XB に転送する. また, XB では EX から予約要求が送られてきた場合は, 優先度の最も高い EX に対して ack を返す.

6.3.3 入力ポート回路

前段のルータからの制御信号を基にして, 各パケットを指定のバッファに送る. バッファが 1 個の場合は, パケットをスルーするだけの回路となる.

6.3.4 出力ポート回路

各バッファに格納されているパケットをアービタ回路から送られてきた信号を基に選択し, その選択したパケットを出力する.

6.3.5 予約要求処理回路

EX から転送された予約要求を, 予約したい EX に接続されている出力ポートのアービタ回路に送る.

6.4 INSPIRE のルータとの相違

INSPIRE は計算機シミュレーションのため, 理想的なルータを想定している. そのため, NDHL トランスレータが生成するルータとは次の点で仕様が異なる.

- ルータ内でパケットのルーティングに必要とする時間:
  INSPIRE のルータでは, パケットのルーティングの処理が 1 クロックで全て終了するが, トランスレータのルータでは 3 クロックかかる. また, 「先読み」と「予約」をした時のルーティングの処理にかかる時間は, INSPIRE のルータは 0 なのに対し, トランスレータのルータでは 5 クロックかかる.
- パケットが衝突した際の調停アルゴリズム:
  INSPIRE のルータではパケットが衝突した時, FGFS (First Generate First Service) で処理されが, トランスレータのルータでは, 疑似ランダムで処理される.
- バッファの容量:
  INSPIRE のルータでは 1 flit, トランスレータのルータでは 3 flit である.

7. 評価と考察

ここでは NDHL トランスレータの評価のために, 実際に 3 次元 Mesh ネットワークを NDL で記述し, トランスレータで変換した例を示す. ただし, 現在完成しているトランスレータは, バッファ&バッファ制御回路と出力ポート回路の VHDL 記述しか生成できない. そのため, 必要となる他の回路の VHDL 記述は, トランスレータの出力形式に従って実際に記述した. 3 次元 Mesh ネットワークの NDF のルーティング記述部を図 8 に示す. Mesh ネットワークは直接網なので, PU と EX のみが記述されている. 図 8 の記述を NDHL トランスレータで変換した結果を図 9 に示す. トランスレータで変換された図 9 の VHDL 記述にルータ・チップ間とチップ内での信号処理を行なう記述を追加することで, バッファ& バッファ制御回路の VHDL 記述が生成される. また, 3 次元 Mesh ネットワークの NDF の資源記述部の情報から, 出力ポート回路の VHDL 記述が生成される.

```
routing{
    pu(i, j, k){
        route(0);
    }

    ex(i, j, k){
        char x, y, z;

        x = msg_dest_dim(0);
        y = msg_dest_dim(1);
        z = msg_dest_dim(2);
        if(x == i && y == j && z == l
            route(6);
        if(x > i ) route(0);
        if(x < i ) route(1);
        if(y > j ) route(2);
        if(y < j ) route(3);
        if(z > k) route(4);
        if(z < k) route(5);
    }
}
```

図 8　3 次元 Mesh の NDL 記述 (ルーティング部)

表 2　3 次元 Mesh ネットワークの EX の合成結果

トポロジ名	ゲート数	最大動作周波数 (MHz)
3 次元 Mesh	22569	37.20

NDHL トランスレータで生成した 3 次元 Mesh ネットワークの EX (7 入力 7 出力) の VHDL 記述を, Synopsys 社の VHDL Compiler と Design Compiler を使用して合成した結果を表 2 に示す. 最大動作周波数は, すでに実現されている他のマシンのルータ・チップと比較すると低い値となっている. この原因となるクリティカル・パスは, NDF に記述されているルーティング・アルゴリズムに従って目的の出力ポートにポート獲得の要求を出す処理の部分である. この理由は次の通りである. ルータ・チップのルーティング・アルゴリズム以外の部分は, NDL 記述の入出力ポート数と virtual channel 数の情報をもとに生成され, ネットワーク・トポロジとルーティング方式にかかわらず同じ回路構成となる. そのためこれらの部分は, クリティカル・パスの最適化がなされている回路を NDHL トランスレータが生成することができる. しかし, ルーティング・アルゴリズム部は, ネットワーク・トポロジとルーティング方式によって異なるため, トランスレータはクリティカル・パスを最適化した回路を生成できない. そのため, このような結果となった.

また, この時に NDHL トランスレータの正当性に

```
function ROUTING(data_usr,
                 data_header: in std_logic_vector(15 downto 0);
                 x_address, yaddress,
                 z_address: in std_logic_vector(3 downto 0)
                 )return std_logic_vector is
variable temp_request: std_logic_vector(14 downto 0);
variable msg_hop_cnt: std_logic_vector(3 downto 0);
variable x, y, z: std_logic_vector(3 downto 0);
begin
temp_request := "0000000000000000";
msg_hop_cnt := data_header(15 downto 12);
z := "0000";
y := "0000";
x := "0000";
x(3 downto 0) := data_header(3 downto 0);
y(3 downto 0) := data_header(7 downto 4);
z(3 downto 0) := data_header(11 downto 8);
if(x = x_address and y = a_address and z = z_address) then
   temp_request(6) := '1';
   return temp_request;
end if;
if(x > x_address) then
   temp_request(0) := '1';
   return temp_request;
end if;
if(x < x_address) then
   temp_request(1) := '1';
   return temp_request;
end if;
if( y > y_address) then
   temp_request(2) := '1';
   return temp_request;
end if;
if(y < y_address) then
   temp_request(3) := '1';
   return temp_request;
end if;
if(z > z_address) then
   temp_request(4) := '1';
   return temp_request;
end if;
if(z < z_address) then
   temp_request(5) := '1';
   return temp_request;
end if;
return temp_request;
end ROUTING;
```

図9　3次元 Mesh の VHDL 記述 (ルーティング部)

ついての検証を行なった．生成した EX に適当な入力信号のパターンを与え，トランスレータで生成した VHDL 記述が NDF で定義したルーティング・アルゴリズムに正しく従うことを確認した．

現在のトランスレータの NDF を VHDL 記述に変換するアルゴリズムでは，図9 に示すように，INSPIRE の NDF の構文をそのまま VHDL において同様の意味を持つ構文に変換する．そのため，NDF を記述する時にクリティカル・パスを考慮しないと，最大動作周波数が低くなる可能性がある．従って，ユーザが同じネットワーク・トポロジとルーティング方式を NDF で記述した時に，クリティカル・パスを考慮するかどうかでトランスレータで生成されるルータ・チップの性能は大きく異なる．これは自動設計システムとしてはあまり望ましくない．よって，クリティカル・パスの最適化を考慮した変換が今後の課題と言える．

8. おわりに

本研究では，NDHL トランスレータを提案した．NDHL トランスレータを使用することにより，INSPIRE でネットワークの諸特性を記述している NDF から，ルータ・チップの VHDL 記述を生成できる．トランスレータの対象は，ネットワークとして並列計算機に用いられる間接網及び直接網，ルーティング方式として固定ルーティングと適応ルーティングで，wormhole 方式のものである．これにより，このトランスレータと市販の VHDL 対応の自動合成ツールを組み合わせることにより，計算機シミュレーションによる評価と，設計レベルの評価を容易にとることができるようになった．

現在のトランスレータが生成するルータの VHDL 記述では，性能評価を主目的としているため，転送性能に最低限必要な回路のみ備え，電源や信号線の遅延は考慮していない．しかし，これらを考慮に入れることで，ユーザはハードウェア言語より抽象度の高い NDL を記述することで，実際にルータ・チップを作成することができるようになる．

現在，NDHL トランスレータの変換アルゴリズムは，NDF に記述されているルーティング・アルゴリズムをそのまま VHDL 記述に変換している．そのため，NDF におけるネットワークの記述に，クリティカル・パスが大きく影響を受ける．よって，今後の課題として，クリティカル・パスの最適化を考慮した変換と，まだ生成できていない回路の変換等が挙げられる．

謝　辞

本研究に関し貴重な御意見を頂いた，筑波大学坂井修一助教授ならびにアーキテクチャ研究室グループ諸氏に深く感謝します．なお，本研究の一部は科学研究費補助 (奨励 A09780234) によるものである．

参考文献

1) 原田智紀 他，"並列処理ネットワークのための性能評価用シュミレータ生成系 INSPIRE" 情報処理研報 Vol.95, No.80, pp.65-72,1995
2) R. Lipsett, C. Schaefer and C. Ussery, "VHDL:Hardware Description and Design" Kluwer Academic Publishers,1989
3) 村田淳 他，"大規模並列計算機用結合網クラス MDX の提案と評価" JSPP'96 論文集,pp.137-114,1996.
4) 曽根 猛 他，"ハイパクロスバ・ネットワークにおける virtual channel の動的選択による適応ルーティング"，情報処理学会論文誌，Vol.37，pp.1409-1418，1996