チップ内ネットワークにおける Fat H-Tree トポロジの性能評価

松谷宏紀[†]　鯉渕道紘[††]　天野英晴[†]

Networks-on-Chips（NoC）におけるトポロジとして，これまでメッシュや単純なツリー構造が使われてきた．一般的に，H-Tree ではメッシュより平均ホップ数が小さくなるが，ツリーのルート付近にトラフィックが集中して性能のボトルネックを生む．Fat Tree ではツリーを多重化しているが，多重度 2 程度では依然としてルート付近の混雑を緩和できない．一方，我々は Fat H-Tree と呼ばれるトポロジを提案しており，高々2 本の H-Tree を組み合わせるだけでメッシュを上回る性能を実現している．本論文では Fat H-Tree の特性を解析した上で，実アプリケーションにもとづいたシミュレーションで性能を測定，さらに，Fat H-Tree ベースの NoC を RTL 設計して結合網のハードウェア量を見積もった．その結果，1) Fat H-Tree ベースの NoC の性能は 2-D Mesh より高く 2-D Torus に匹敵する，2) Fat H-Tree の平均ホップ数は 2-D Torus や Fat Tree より小さい，3) Fat H-Tree は 2-D Mesh よりも結合網に要するハードウェア量が小さい，4) Fat H-Tree は 2-D Torus より配線資源を多く必要とするが，現状のテクノロジにおいて十分に実装可能である，という点を確かめた．

Performance Evaluation of Fat H-Tree Topology in Networks-on-Chips

Hiroki Matsutani,[†] Michihiro Koibuchi[††]
and Hideharu Amano[†]

As on-chip network topologies, two-dimensional mesh, torus, and simple tree based topologies have been widely used. Although H-Tree topology has advantages in average hop counts compared with mesh, its performance is limited due to the congestion around the root of the tree. Fat Tree topology has been proposed to enhance the number of connections toward the root. However, mitigating the congestion is still hard for Fat Tree with a small number of redundant connections, such as two. On the other hand, we have proposed Fat H-Tree topology, which is formed by combining only two H-Trees, and it achieves the higher performance than that in same-sized mesh. In this paper, the performance of Fat H-Tree is evaluated by using real application traces, and the hardware amount of routers for implementing Fat H-Tree is estimated based on the RTL model of NoCs. Consequently, the following conclusions are derived: 1) the performance of Fat H-Tree is higher than 2-D mesh and is close to 2-D torus. 2) the average hop counts in Fat H-Tree are shorter than Fat Tree and 2-D torus. 3) the router hardware amount of Fat H-Tree topology is smaller than that for 2-D mesh. 4) Fat H-Tree requires longer total wiring than that in 2-D torus, but current process technologies can provide the sufficient wiring resources for implementing Fat H-Tree topology.

1. はじめに

半導体技術の進歩により，単一チップ上にプロセッサやメモリ，I/O など複数の設計モジュールをタイル状に実装できるようになった．このようなタイルアーキテクチャでは，タイル同士を結合するチップ内結合網としてチップ内ネットワーク（Networks-on-Chip，NoC）[1] が用いられ，そのネットワークトポロジはアプリケーションの性能とハードウェア量を決定付ける一要素である．

NoC は様々な用途に用いられているが，本論文では安価な組込み向けチップを想定する．このような用途においては，オンチップルータの面積が増えると，アプリケーションを実行する計算コアの面積が直接圧迫されるので，ルータの面積は極力小さくする必要がある．その一方，ピン数によってデータ幅が制限されるチップ間結合網とは異なり，NoC では配線資源が豊富である．例えば，0.1μm プロセスにおいて 3mm 角のタイルは 1 配線層当たり最大 6,000 本の配線を利用できる[1]．

NoC のトポロジとしては，チップ上への 2 次元レイアウトに適したメッシュや単純なツリー構造が代表的である．これら以外にも，大規模並列計算機向けに 2 次元レイアウト可能な階層トポロジがいくつか提案されてきたが，これらのトポロジでは次数が高くルータハードウェア量の点で不利となる．以上を踏まえ，**図 1** に，次数が小さく対称な 2 次元レイアウトが可能なトポロジの代表例を示す．図 1 の各トポロジにおいて四角はルータ，丸は計算コアをそれぞれ表す．

2-D Mesh は，Raw プロセッサ，Trips プロセッサ，ストリーム処理向け NoC である aSOC，picoChip 社の超並行処理プロセッサアレイなど幅広い用途で用いられている．

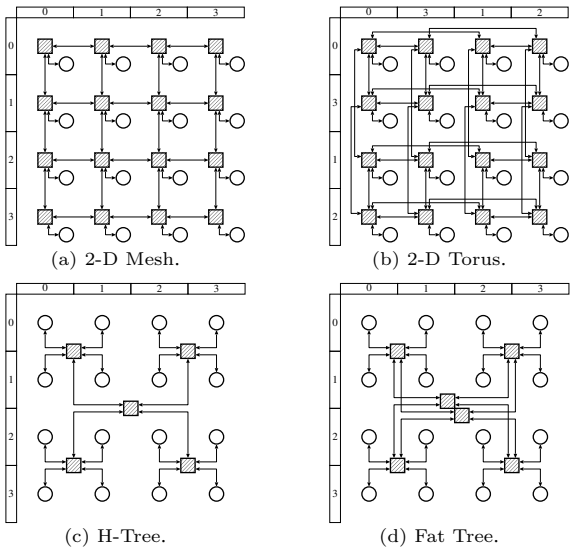
2-D Torus は，**図 1(b)** に示されるように，wrap-around チャネルによってネットワークの端と端がつながる．2-D Torus は文献 1) に加え，IMEC 社の NoC[2] などで用いられている．

H-Tree は，**図 1(c)** に示されるように，4 分木の形をとるため 2 次元平面実装に適しているが，トラフィックがツリーのルート付近に集中しやすい．H-Tree は SCORE や QuickSilver 社の ACM などのリコンフィギャラブルシステムで利用されている．

Fat Tree の例を **図 1(d)** に示す．Fat Tree は（上向きリンク数 p，下向きリンク数 q，ランク数 r）の組みで表記される[3]．図の Fat Tree は $(2, 4, 2)$ である．H-Tree とは異なり，多重化した上位ツリーにより経路分散が可能である．既存の NoC への適応例としては SPIN などがある．

一般的に，ツリー型トポロジではメッシュより平均ホップ数が

† 慶應義塾大学大学院 理工学研究科
Graduate School of Science and Technology, Keio University
†† 国立情報学研究所
National Institute of Informatics



(a) 2-D Mesh. (b) 2-D Torus. (c) H-Tree. (d) Fat Tree.

図 1 代表的なネットワークトポロジ（16 ノード）.

小さくなるが，ツリーのルート付近にトラフィックが集中して性能のボトルネックを生む．Fat Tree ではツリーを多重化しているが，$p = 2$ 程度では依然としてルート付近の混雑を緩和できない場合がある．一方，我々が文献 4) で提案した Fat H-Tree では，2 本のツリーの配置を工夫するだけで，リーフ付近のルータのみを用いたルーティングを可能にした．文献 4) では Fat H-Tree の提案とその定義に重きが置かれており，性能と結合網のコストに関する網羅的な評価は成されていない．本論文では Fat H-Tree の特性を解析した上で，実アプリケーションにもとづいたシミュレーションによって性能を測定，Fat H-Tree ベースの NoC を RTL 設計して結合網のハードウェア量を見積もった．

本論文の構成は次のとおりである．まず 2 章で Fat H-Tree トポロジとそのルーティングを説明する．3 章において Fat H-Tree と既存のトポロジについて理論的に解析し，4 章でハードウェア量と性能について既存のトポロジと比較する．最後に 5 章で本論文をまとめる．

2. Fat H-Tree トポロジ

2.1 節で Fat H-Tree トポロジを定義し，2.3 節で Fat H-Tree 用のデッドロックフリー・ルーティングを示す.

2.1 定　　義

Fat H-Tree は red tree と black tree と呼ばれる 2 つの H-Tree が組み合わさったトポロジである．各計算コアは 2 つのポートを持ち，1 つを red tree との接続に，もう片方を black tree との接続に用いる．Fat H-Tree のフォーマルな定義は文献 4) で示されている．

a) Red Tree の定義

図 2 に red tree の例を示す．丸で描かれた各計算コアには 2 次元座標 (x, y) がそれぞれ割り当てられる．以降，計算コアを rank 0 ルータ呼ぶ．まず，rank 0 ルータに対し次の式で求まる **red tree 座標** $R(R_0, R_1, ..., R_{n-1})$ を割り当てる．

$$R_i = ((x/2^i) \bmod 2) + 2 \times ((y/2^i) \bmod 2) \qquad (1)$$

red tree 座標 $R(R_0, R_1, ..., R_{n-1})$ となる rank 0 ルータの上位ルータを rank 1 ルータと呼び，$R(R_1, ..., R_{n-1})$ の red tree 座標を割り当てる．同様の方法で rank 2 ルータから rank n ルータにも red tree 座標を割り振る．ただし，最上位ランク（rank n）ルータの red tree 座標は R とする．例として 図 2 に R, $R(0)$，$R(0,2)$，$R(0,2,2)$ の座標を示す.

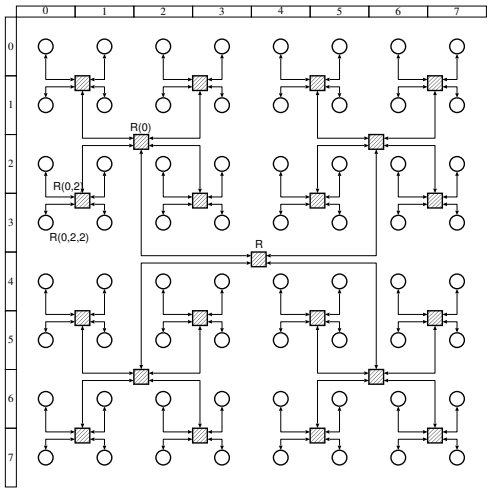


図 2 Red Tree.

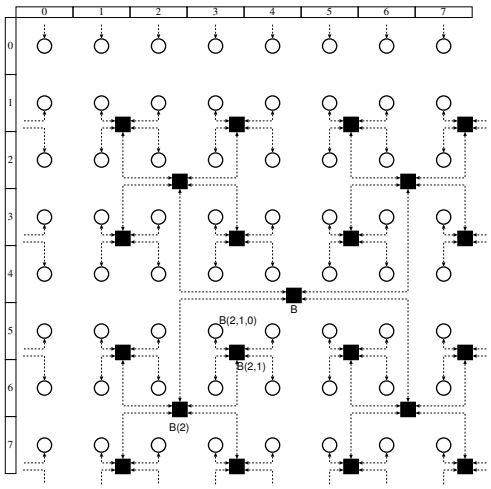


図 3 Black Tree.

b) Black Tree の定義

図 3 に black tree の例を示す．black tree は red tree の座標を左斜め下方向に 1 つずらしたものである．このため rank 0 ルータの **black tree 座標** $B(B_0, B_1, ..., B_{n-1})$ は次の式で求まる．

$$B_i = ((((x-1) \bmod 2^n)/2^i) \bmod 2) + 2 \times ((((y-1) \bmod 2^n)/2^i) \bmod 2) \qquad (2)$$

black tree においても rank 1 ルータから rank n ルータに black tree 座標を割り当てていく．最上位ランク（rank n）ルータの black tree 座標は B とする．例として 図 3 に B，$B(2)$，$B(2,1)$，$B(2,1,0)$ の座標を示す.

c) Fat H-Tree の定義

各計算コア（rank 0 ルータ）に対し，red tree および black tree を接続したものを **Fat H-Tree** と呼ぶ.

Fat H-Tree においては rank 0 ルータと red tree および black tree の rank 1 ルータによってトーラス構造が形成される．Fat H-Tree が内包するトーラス構造を **図 4** に示す.

2.2 2 次元レイアウト

Fat H-Tree は，トーラス同様，畳み込むことで 2 次元チップ上に容易にレイアウトできる．**図 5** に Fat H-Tree の 2 次元レイアウトを示す．図 4 と 図 5 は等価なトポロジである.

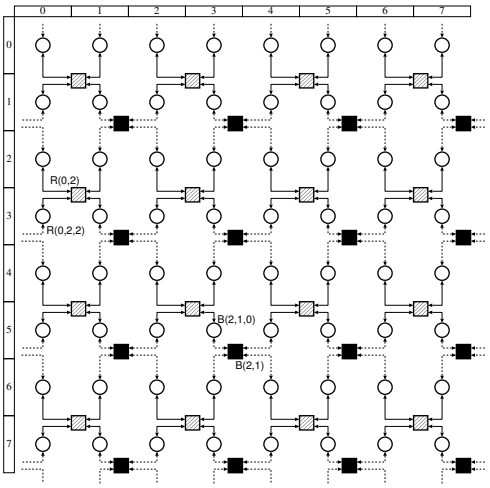


図 4 Fat H-Tree（rank 2 以上のルータは省略）.

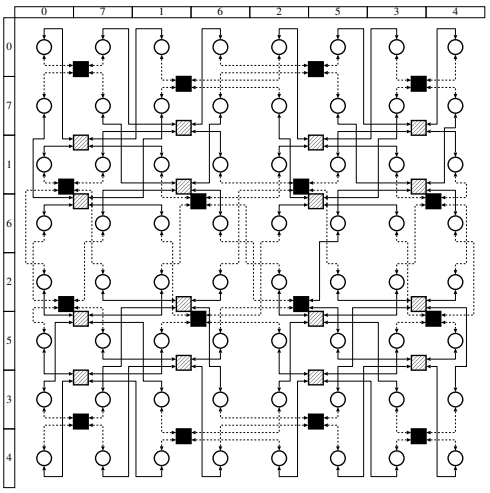


図 5 Folded Fat H-Tree（rank 2 以上のルータは省略）.

2.3 ルーティングアルゴリズム

Fat H-Tree におけるデッドロックフリー・ルーティングとして以下の 3 種類を挙げる.

(1) **Single Tree Routing（STR）：**
送信時に red tree または black tree のどちらか一方を選び，H-Tree と同じ方法でルーティングする.

(2) **Dual Tree Routing（DTR）：**
経路の途中でツリーの切り替えが可能なルーティングであり，STR とは異なり，送信元と宛先の間の任意の最短経路を取ることができる．DTR では，ツリー間の循環依存を取り除くために，red tree から black tree に乗り換えるときに仮想チャネルの番号を切り替える．DTR がデッドロックフリーであることは文献 4) で証明されている．なお，文献 4) のルーティングでは，ツリーの切り替え回数を制限することで必要な仮想チャネルを 2 本に抑えている.

(3) **Torus Routing（TOR）：**
rank 0 ルータおよび rank 1 ルータによって形成されるトーラス構造（図 4）を用いてルーティングする．DTR 同様，ツリーの乗り換え時に仮想チャネル番号を切り替える.

DTR のみ最短経路ルーティングである．DTR と TOR においてデッドロック回避のために必要な仮想チャネル数は $\lfloor H_{max}/4 \rfloor + 1$ 本である．ただし，H_{max} を各ルーティングの最大ホップ数とする.

3. Fat H-Tree の解析

Fat H-Tree のスループット，平均ホップ数，総ワイヤ長，ルータのハードウェア量について理論的に解析する.

3.1 スループットの理想値

スループットの理想値 Θ_{ideal} は次の関係式で表現される[5].

$$\Theta_{ideal} \leq \frac{2bB_c}{N} \tag{3}$$

ただし，N をコア数，b をチャネル当たりの帯域，B_c を channel bisection とする．アレイサイズ $n = \sqrt{N}$，ツリーのランク数 $r = \log_4(N)$ としたとき，**表 1** に各トポロジの B_c を示す．N ノードの Fat H-Tree の B_c は $4n + 8$ となる．この第 2 項は H-Tree 2 つ分（black tree と red tree）の channel bisection であり，第 1 項はルータ-コア間チャネルによって得られる channel bisection である．Fat H-Tree ではルータ-コア間チャネルによって形成されるトーラス構造により，単に H-Tree を二重化する場合に比べて飛躍的に Θ_{ideal} が向上する．Fat H-Tree の実スループットは，4 章でフリットレベルシミュレータを用いて確認する.

表 1 Channel bisection B_c.

	N-node	16-node	64-node	256-node
2-D Mesh	$2n$	8	16	32
2-D Torus	$4n$	16	32	64
H-Tree	4	4	4	4
Fat Tree	$N/2^{r-1}$	8	16	32
Fat H-Tree	$4n + 8$	24	40	72

3.2 平均ホップ数

N ノードのネットワークにおいて有効なソース-ディスティネーション・ペアは $(N^2 - N)$ 個存在するため，平均ホップ数 H_{ave} は次の式で計算できる.

$$H_{ave} = \frac{1}{N^2 - N} \sum_{x,y \in N} H_{(x,y)} \tag{4}$$

ただし，$H_{(x,y)}$ はコア x からコア y へのホップ数とする．**表 2** にユニフォームトラフィックを用いた際の H_{ave} を示す．H_{ave} は使用するルーティングアルゴリズムによっても異なる．Fat H-Tree に関しては STR，DTR，TOR の結果を示す．DTR を使った場合 Fat H-Tree の H_{ave} は 2-D Mesh より 30% 以上，Torus より 20% 以上小さくなった.

表 2 平均ホップ数 H_{ave}（ルータ-コア間も 1-hop にカウント）.

	16-node	64-node	256-node
2-D Mesh (DOR)	4.67	7.33	12.67
2-D Torus (DOR)	4.14	6.06	10.03
H-Tree	3.61	5.43	7.36
Fat Tree	3.61	5.43	7.36
Fat H-Tree (STR)	3.20	5.02	6.90
Fat H-Tree (DTR)	3.20	4.84	6.78
Fat H-Tree (TOR)	3.20	5.65	10.83

3.3 ネットワークの総ワイヤ長

ネットワークの総ワイヤ長 L_w は次式で計算できる.

$$L_w = 2wL_l \cdot l \tag{5}$$

ただし，L_l をネットワークの総リンク長，l を隣接コア間距離の実測値，w をリンクのビット幅とする.

この総リンク長 L_l は隣接コア間の距離を 1-unit としたときのリンクの総延長を表し，ツリーとメッシュ／トーラスを公平に比較するため，コア-ルータ間のリンク長も一律 1-unit と見なす.

本節では，まず，この総リンク長 L_l を求め，ネットワークの総ワイヤ長 L_w については 4.2 節にて検討する．

2-D Mesh には $2(N-n)$ 本のリンクが存在し，それぞれの長さは 1-unit である．また，ノード数分のコア-ルータ間リンクも存在するため，2-D Mesh における総リンク長 L_l は $2(N-n)+N$ となる．一方，2-D Torus ではルータ同士をつなぐリンク長は Mesh の 2 倍となる．

H-Tree において，ある rank i ルータから，それに接続される rank $(i-1)$ ルータへのリンク長は 2^{i-1} である．したがって，rank i ルータから 4 個の下位ルータへの総リンク長は 2^{i+1} となる．ノード数 N の H-Tree において rank i ルータは $N/4^i$ 個存在する．ランク数 $r=\log_4(N)$ のとき，H-Tree の総リンク長 $L_{l,ht}$ は次の式で計算できる．

$$L_{l,ht}=\sum_{i=1}^{r}2^{i+1}\cdot\frac{N}{4^i}=2N\left(\frac{2^r-1}{2^r}\right) \tag{6}$$

同様に Fat Tree の総リンク長 $L_{l,ft}$ は次式で計算できる．

$$L_{l,ft}=\sum_{i=1}^{r}2^{i+1}\cdot\frac{N}{2^{i+1}}=r\cdot N \tag{7}$$

Fat H-Tree は 2 つの畳み込まれた H-Tree から構成される．H-Tree の畳み込みによって各リンクの長さは 2 倍になるが，最上位ランクのリンクのみ長さが 0 に等しくなる．これは，H-Tree を畳み込む場合，4 個の rank $(r-1)$ ルータと 1 個の rank r ルータの位置が重なるためである．したがって，Fat H-Tree の総リンク長 $L_{l,fht}$ は次の式で計算できる．

$$L_{l,fht}=4\cdot\sum_{i=1}^{r-1}2^{i+1}\cdot\frac{N}{4^i}=8N\left(\frac{2^{r-1}-1}{2^{r-1}}\right) \tag{8}$$

表 3 ネットワークの総リンク長 L_l（隣接コア間距離を 1-unit とする）．

	N-node	16-node	64-node	256-node
2-D Mesh	$2(N-n)+N$	40	176	736
2-D Torus	$4(N-n)+N$	64	288	1,216
H-Tree	式 6	24	112	480
Fat Tree	式 7	32	192	1,024
Fat H-Tree	式 8	64	384	1,792

以上をまとめたものが **表 3** である．Fat H-Tree の総リンク長は 2-D Torus より長いが，ピン数に制限されるチップ間結合網とは異なり NoC では配線リソースは豊富であるため，Fat H-Tree による総ワイヤ長の増大は最近のプロセスにおいて問題にはならないと考えられる．これに関しては 4.2 節にて検証する．

3.4 ルータのハードウェア量

ネットワークを構成するルータのハードウェア量は，ルータ単体のハードウェア量とルータの個数の積で見積もることができる．

ルータ単体のハードウェア量は node degree によって左右される．本論文では，ツリーとメッシュを公平に比較するため，ルータ-コア間チャネルも node degree に含めるものとする．Chien が提案したルータコストモデル[6] に当てはめると node degree を d としたとき，ルータのゲート数 $G_{rt(d)}$ は次式で計算できる☆．

$$G_{rt(d)}=29d^2+17d^2+(320+100)d \tag{9}$$

右辺の第 1 項がクロスバ，第 2 項がクロスバのアービタ，第 3 項が物理チャネル d 本分のゲート数に相当する．

☆ ただし，Chien のモデルではデータ幅は 16-bit とし，チャネルバッファのゲート数は考慮していない．

H-Tree におけるルータの個数 $N_{rt,ht}$ は次式で計算できる．

$$N_{rt,ht}=\frac{q^r-1}{q-1} \tag{10}$$

ただし，q は下向きのリンク数とし，$q=4$ である．また，Fat H-Tree は H-Tree 2 つ分なので $N_{rt,fht}=2N_{rt,ht}$ となる．

Fat Tree におけるルータの個数 $N_{rt,ft}$ は次式で計算できる．

$$N_{rt,ft}=\frac{q^r-2^r}{q-2} \tag{11}$$

以上をもとに各トポロジにおけるルータのゲート数を見積もった結果が**表 4** である．Fat H-Tree のハードウェア量は 2-D Mesh や Torus より小さいことがわかる．しかしながら，このゲート数見積もりにはネットワークインターフェイス（NI）部を含めていない．Fat H-Tree のみ 2-port の NI が要るので，Fat H-Tree の実際のゲート数はさらに増えることになる．4 章では，NI を含め NoC 全体を論理合成して実際のハードウェア量を測定する．

表 4 ルータのゲート数 G_{net}.

	N-node	16-node	64-node	256-node
2-D Mesh	$NG_{rt(5)}$	52,000	208,000	832,000
2-D Torus	$NG_{rt(5)}$	52,000	208,000	832,000
H-Tree	$N_{rt,ht}G_{rt(5)}$	16,250	68,250	276,250
Fat Tree	$N_{rt,ft}G_{rt(6)}$	25,056	116,928	501,120
Fat H-Tree	$N_{rt,fht}G_{rt(5)}$	32,500	136,500	552,500

3.5 解析結果のまとめ

本章の解析によって，Fat H-Tree には次の特徴があることがわかった．

- スループットの理論値は 2-D Torus と同程度
- 平均ホップ数が 2-D Torus，Mesh より小さい
- ルータのハードウェア量は 2-D Torus，Mesh より小さい
- ネットワークの総ワイヤ長は 2-D Torus より長い

4 章では，これらの解析結果をシミュレーションによって確かめる．

4. Fat H-Tree トポロジの評価

実アプリケーションにもとづいたシミュレーションによって FatH-Tree の性能を測定，Fat H-Tree ベースの NoC を RTL 設計してネットワークのハードウェア量を見積もる．

4.1 ネットワークのハードウェア量

計算コアとルータ回路を組み合わせることで Fat H-Tree をはじめ各種トポロジが実現される．本評価では，0.18μm スタンダードセルライブラリを用いて，各トポロジごとに 16 コアと 64 コアの NoC を設計した．設計した NoC は Synopsys 社の Design Compiler で合成，各トポロジのハードウェア量を見積もった．

NoC の RTL 設計のために文献 7) のルータ回路を用いた．このルータ回路は 4 段のパイプラインステージから構成され，パケット転送処理は完全にパイプライン化されている．ルーティングはソースルーティングで実装され，パケット転送方式は wormhole 方式を用いる．ルータのデータ幅（フリット幅）は 32-bit とし，各パイプラインステージごとに 1 チャネルにつき 1-flit 分のバッファを持つ．今回の NoC ではコア自体は空のダミー回路とし，ルータ-コア間を接続する NI として 2-flit 分の FIFO を入力と出力にそれぞれ持たせた．Fat H-Tree で必要となる 2-port NI は通常の NI を 2 つ分持たせることで実現した．

図 6 と**図 7** に各ネットワークトポロジの合成結果を示す．図 6 が 16 コア，図 7 が 64 コアの結果である．ルータのハードウェア量だけに着目すれば，表 4 の結果と概ね等しいが，Fat H-Tree に関しては 2-port の NI が必要となるため，NI のハードウェア量は他のトポロジの約 2 倍に増加する．それでも Fat H-Tree

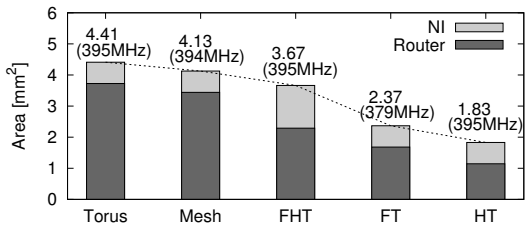


図 6 ルータと NI のハードウェア量（16 コア）.

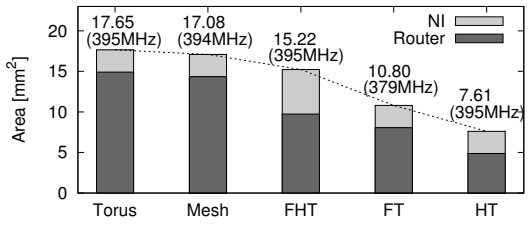


図 7 ルータと NI のハードウェア量（64 コア）.

の面積は 2-D Torus や Mesh より小さく，コスト的に有利であると言える．実際，64 コア の Fat H-Tree のハードウェア量は 15.22mm^2 であり，12mm 角チップ全体の 10.56% で実現できる．なお，動作周波数に関してはトポロジ間に大差は見られなかった．

4.2 ネットワークの配線資源

3.3 節で解析した L_l の値を用いて，ネットワークの総ワイヤ長 L_w を求める．ここでは，12mm 角のチップに 16 コアと 64 コアの NoC を 32-bit の双方向チャネルを用いて実装する場合を考える．このとき 16 コア構成では隣接コア間距離 l は 3mm，64 コアでは 1.5mm となる．以上のパラメータを式 5 に代入することで得られるネットワークの総ワイヤ長 L_w を **表 5** に示す．また，表中のカッコ内にチップの配線資源に占める L_w の割合を示す．ただし，ここでは 0.18μm プロセスを仮定し，ネットワークの配線には，全配線層のうち 2 層分のみを使うものとする☆.

表 5 総ワイヤ長 L_w（単位は mm）と 2 層分の配線資源に占める割合.

	16-node		64-node	
2-D Mesh	7,680	(2.13%)	16,896	(4.69%)
2-D Torus	12,288	(3.41%)	27,648	(7.68%)
H-Tree	4,608	(1.28%)	10,752	(2.99%)
Fat Tree	6,144	(1.70%)	18,432	(5.12%)
Fat H-Tree	12,288	(3.41%)	36,864	(10.24%)

3.3 節の解析により，Fat H-Tree は 2-D Torus より総ワイヤ長が長くなる．しかしながら，6 層以上の配線層が利用できる最近のプロセスにおいては配線資源は十分に豊富であり，図 5 で示したとおり，2 層分の配線資源のうち高々10%程度の出費で 64 コアの Fat H-Tree を実現できた．この結果より，Fat H-Tree は現状のプロセスで十分に実装可能であると言える．

4.3 スループット

4.3.1 シミュレーション環境

各トポロジにおけるスループットを測定するためにフリットレベル・シミュレータを用いた．ここでは，各ルータのスイッチング機構として，I/O バッファ，クロスバ，クロスバコントローラを単純化したモデルを採用し，ヘッダフリットが隣接ルータまたは計算コアに転送されるのに 3 サイクルかかるものとする．パケット転送には wormhole 方式を用い，各チャネルは 1-flit 分のバッファを持つ．パケット長はヘッダ 1-flit 分を含め 16-flit とした．

☆ 最小配線ピッチが 0.8μm の 0.18μm プロセスを用いて評価を行った．12mm 角のチップを想定しているので，1 配線層当たりのトラック数は 15,000 本となり，この配線層 2 層分の総ワイヤ長は 360m である．

各トポロジの性能は使用するルーティングアルゴリズムに左右される．本シミュレーションでは，2-D Torus および Mesh には次元順ルーティング，Fat H-Tree には DTR と TOR を用いる．Fat Tree および Fat H-Tree では代替経路を複数取れる場合がある．そこで，path selection algorithm として Sancho's algorithm[8] を用いてトラフィックの集中をできる限り排除する．なお，構造上デッドロックを回避するために，2-D Torus および Fat H-Tree の DTR では仮想チャネルを 2 本，Fat H-Tree の TOR では仮想チャネルを 3 本用いる．

シミュレーションに用いる通信パターンとして，ユニフォームトラフィックと NAS Parallel Benchmark（NPB）プログラムから得られた通信パターンを用いる．NPB の各プログラムは数値計算を扱う並列アプリケーションであるが，これらの通信パターンにはストリーム処理で頻出する fork/join に似た通信パターンが含まれる．今回，NPB から次のプログラムを用いる：Block Tridiagonal solver（BT），Scalar Pentadiagonal solver（SP），Conjugate Gradient（CG），Multi-Grid solver（MG），large Integer Sort（IS）．プログラムのクラスは “W” とし，計算コア数は 16 と 64 とした．各トポロジの性能はアプリケーションタスクのマッピングによっても変化する．本評価では，式 4 にもとづき平均ホップ数ができるだけ小さくなるようタスクを割り当てた．

4.3.2 シミュレーション結果

図 8 に 16 コアでユニフォームトラフィックを用いた際のスループット（accepted traffic）とレイテンシのグラフを示す．グラフ中の HT は H-Tree，FT は Fat Tree，FHTD は Fat H-Tree の DTR，FHTT は Fat H-Tree の TOR に対応する．それぞれの平均ホップ数は括弧内に示してある．これらの平均ホップ数は解析によって得られた 表 2 の H_{ave} と同じであり，FHTD が最も小さい．スループットに関しては Torus，FHTT，FHTD，Mesh，FT，HT の順となった．Fat H-Tree は 2-D Torus より高い channel bisection B_c を有すが， FHTD のスループットは Torus に劣っている．これは Fat H-Tree で最短経路のみを用いるとトラフィックがツリーのルート付近に集中してしまうためである．FHTT のように rank 2 以上のリンクの使用を諦めれば，Fat H-Tree は Torus と等価となり，Torus に匹敵する性能が得られる．ただし，非最短経路の導入によって FHTT の平均ホップ数は増加する．平均ホップ数の増加は消費電力の増加を引き起こすため，FHTD と FHTT の間には性能と消費電力のトレードオフが存在する．

図 9-**図 13** に，16 コアにおける BT，SP，CG，MG，IS トラフィックのスループットとレイテンシを示す．CG を除き，FHTD は Torus に匹敵する性能を実現できている．なお，all-to-all 通信を含む IS ではユニフォームトラフィック同様，FHTT の性能が高いが，それ以外の比較的単純な通信では FHTT より FHTD のほうが有利なケースが目立つ．これは，単純な通信においては rank 2 以上のリンクを使用しても性能のボトルネックにはならず，むしろこれらの高位リンクが性能向上に寄与するためである．

続いて 64 コアでの評価に移る．**図 14** に 64 コアでのユニフォームトラフィックの結果，**図 15**-**図 19** に NPB プログラムでの結果を示す．64 コアにおいても 16 コアと同様の傾向がみられた．

5. ま と め

文献 4) では Fat H-Tree の提案とその定義に重きが置かれており，性能と結合網のコストに関する網羅的な評価は成されていない．本論文では Fat H-Tree の特性を解析した上で，実アプリケーションにもとづいたシミュレーションで性能を測定，Fat H-Tree ベースの NoC を RTL 設計して結合網のハードウェア量を見積

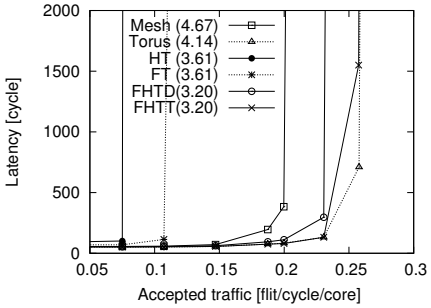


図 8 uniform トラフィック（16 ノード）.

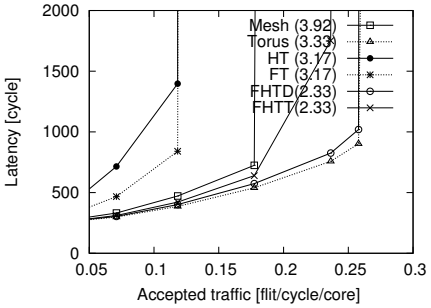


図 9 BT トラフィック（16 ノード）.

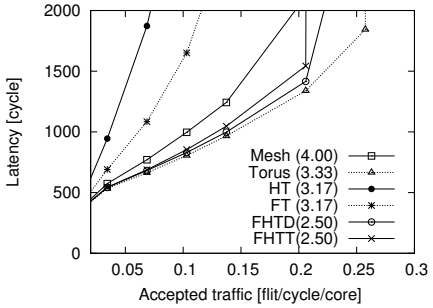


図 10 SP トラフィック（16 ノード）.

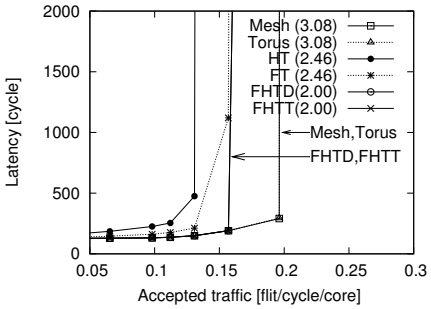


図 11 CG トラフィック（16 ノード）.

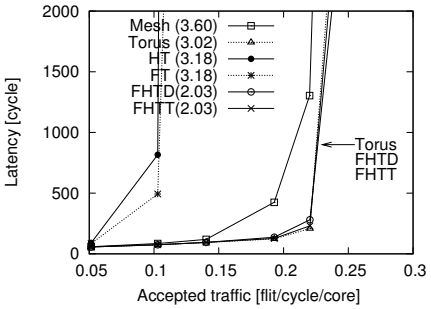


図 12 MG トラフィック（16 ノード）.

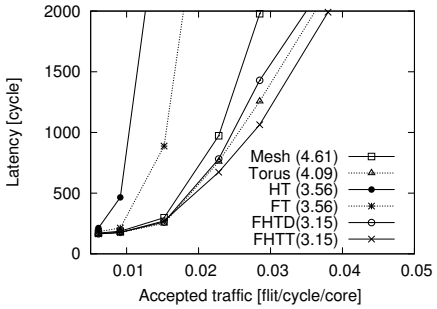


図 13 IS トラフィック（16 ノード）.

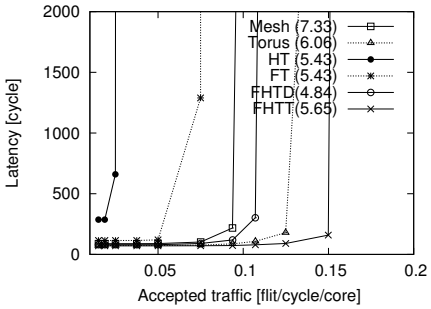


図 14 uniform トラフィック（64 ノード）.

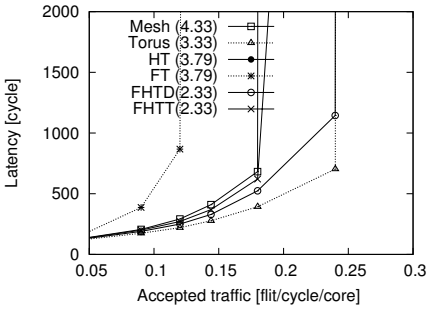


図 15 BT トラフィック（64 ノード）.

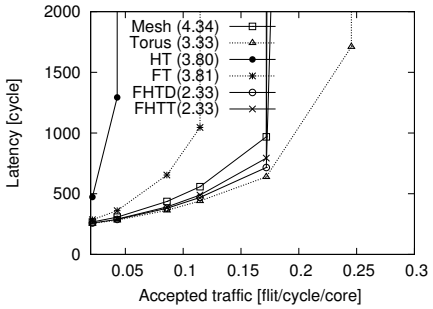


図 16 SP トラフィック（64 ノード）.

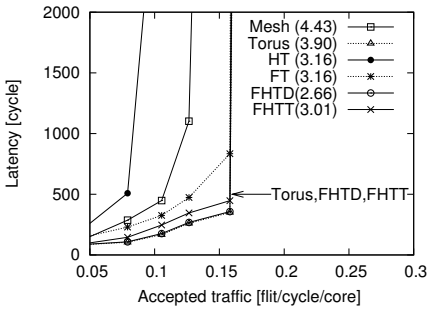


図 17 CG トラフィック（64 ノード）.

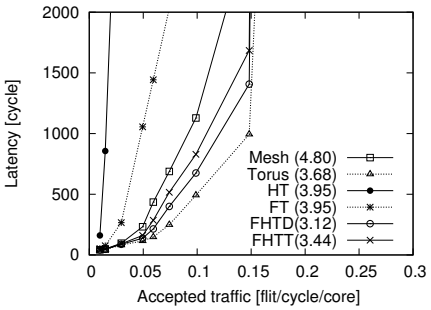


図 18 MG トラフィック（64 ノード）.

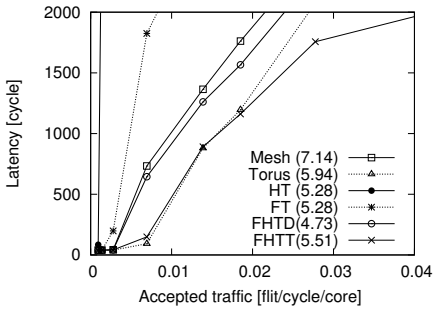


図 19 IS トラフィック（64 ノード）.

もった．その結果，1) Fat H-Tree ベースの NoC の性能は 2-D Mesh より高く 2-D Torus に匹敵する，2) Fat H-Tree の平均ホップ数は 2-D Torus より小さい，3) Fat H-Tree は 2-D Mesh よりも結合網に要するハードウェア量が小さい，4) Fat H-Tree は 2-D Torus より配線資源を多く必要とするが，現状のテクノロジにおいては十分に実装可能である，という点を確かめた．今後は，消費電力に関する解析を行い，Fat H-Tree における DTR と TOR のトレードオフを明らかにする．また，積層チップ向けに Fat H-Tree を 3-D 化することも検討している．

参 考 文 献

1) Dally, W. J. and Towles, B.: Route Packets, Not Wires: On-Chip Interconnection Networks, *Proceedings of the 38th Design Automation Conference*, pp. 684–689 (2001).
2) Marescaux, T. and et. al.: Interconnection Networks Enable Fine-Grain Dynamic Multi-Tasking on FPGAs, *Proceedings of the Field-Programmable Logic and Applications (FPL)*, pp. 795–805 (2002).
3) 天野英晴: 並列コンピュータ, 昭晃堂 (1996).
4) 山田裕, 天野英晴, 鯉渕道紘, 上樂明也, 安生健一朗: リコンフィギャラブルプロセッサアレイ向けチップ内接続網:Fat H-Tree, 電子情報通信学会論文誌. (to be appeared).
5) Dally, W. J. and Towles, B.: *Principles and Practices of Interconnection Networks*, Morgan Kaufmann (2004).
6) Chien, A. A.: A Cost and Speed Model for k-ary n-Cube Wormhole Routers, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 9, No. 2, pp. 150–162 (1998).
7) 松谷宏紀, 鯉渕道紘, 天野英晴: オンチップトーラス網における仮想チャネルフリールーティング, 先進的計算基盤システムシンポジウム (SACSIS) 論文集, pp. 377–384 (2006).
8) Sancho, J. C. and Robles, A.: Improving the Up*/Down* Routing Scheme for Networks of Workstations, *Proceedings of the European Conference on Parallel Computing*, pp. 882–889 (2000).