計算機アーキテクチャ　96－20
（1992. 10. 23）

超並列向きプロセッサ結合網の提案

楊 愚魯　　天野 英晴

慶應義塾大学 理工学部

横浜市 港北区 日吉 3-14-1 安西・天野 研究室

あらまし　　**概　　要**

2^{16} 以上のプロセッサ数を持つ超並列計算機用の結合網 Mx+x を提案する。Mx+x は 2 次元 mesh を基本とし、リンクの追加により遠隔ノードとの交信時に経由するノード数を減らしている。リンクの付加は、x 軸、y 軸の両方について±2 メッシュ上を離れたノードに対して再帰的に行なう。この結果として 2^{16} のノード数で、ノード当たりリンク数 (degree)8 でネットワークの最大ノード経由数 (diameter) は 15 を実現する。さらに、実装時の基板配置、ルーティングアルゴリズムに関しても検討する。Mx+x では、遠隔リンクの方向が単純で、提案された基板配置法を用いると、ほとんどが、基板と垂直または並行な直線で実現できる。

和文キーワード　　**結合網、超並列、ルーティングアルゴリズム、基板の配置**

An interconnection network for massivelly parallel computers

Yang YuLu　　Hideharu Amano

Faculty of Science and Technology, Keio University

3-14-1 Hiyoshi Kohoku-ku Yokohama

Abstract

Abstract

An interconnection network Mx+x is proposed for massively parallel computers with up to 2^{16} nodes. Adding remote linkes to the torus network recursively, the Mx+x realizes a smaller diameter (15 for 2^{16} nodes) than that of the hypercube with smaller number of links per node (8). In this paper, practical disposition of the boards and routing algorithm of the Mx+x are also discussed. Using the proposed board disposition mothod, a remote link is implemented with single vertical/horizontal linear line with a few exception.

英文 key words　　interconnection network, massivelly parallel computer, routing algorithm, board disposition

1 はじめに

プロセッサとメモリから成るノードを、リンクにより結合した構造を持つ、いわゆるマルチコンピュータ型の大規模並列計算機 [1] は、数千ノードを持つシステムが商用化される時代を迎えつつある。このようなマルチコンピュータにおいては、ノード間の結合トポロジがシステムの構成、性能に大きな影響を与える。結合トポロジは、かつては hypercube 結合が多く用いられたが、最近のシステムでは Mesh 結合を用いる場合が多くなっている [2][3][4]。これは、以下の理由によっている

- hypercube 結合のノード当たりのリンク数 (degree) がノード数 n に対して $log_2 n$ であり、拡張が難しい上、千ノード以上ではリンク数が多く実装が困難である。
- 大規模科学技術計算では Mesh 結合が有効な場合が多い。

一方で、最近では、さらに大規模な、プロセッサ数が 2^{16} に及ぶ超並列計算機の開発が始まろうとしている [5][6]。

単純な 2 次元 Mesh は、diameter(最も遠い 2 つのノード間に迂回なしにメッセージを送る場合のノード間転送回数) がノード数 n に対して $\sqrt{n}$ であるため、n が 2^{16} に及ぶ場合、大き過ぎ、超並列計算機の結合トポロジとしては採用できない。

しかし、大規模科学技術計算は超並列計算機の応用分野のひとつとしても有力であり、Mesh 結合を内蔵した結合網はこれらの問題で有利である。また、現在の Mesh 結合並列計算機上で開発されたアルゴリズムの移植という観点でも Mesh 構造の実現は必要である。De Bruijn, Pradhan, μ-STAR 網等、グラフ理論を基にした超並列結合網 [7] が数多く提案されているが、これらの結合網は、Mesh 構造のエミュレーションが困難である。

このため、Mesh 構造を基にした超並列用結合網がいくつか提案されている [8][9] が、いままでのところルーティングや実装に関する考慮があまり行なわれていない。そこで、われわれは Mesh を基にした結合網 Mx+x を提案し、ルーティング、実装等実際的な面に関して検討を行なう。

2 Mx+x 結合網

2.1 結合法

Mx+x は、二次元 Torus を基本とする。ここでは、まず、対象とするシステムをノード数を N^2(N=n^m) に限定し[1]、仮に下のようなノード番号を定める。

(0,0)	(1,0)	(2,0)	(3,0)	⋯	(N-1,0)
(0,1)	(1,1)	(2,1)	(3,1)	⋯	(N-1,1)
(0,2)	(1,2)	(2,2)	(3,2)	⋯	(N-1,2)
(0,3)	(1,3)	(2,3)	(3,3)	⋯	(N-1,3)
⋮	⋮	⋮	⋮	⋱	⋮
(0,N-1)	(1,N-1)	(2,N-1)	(3,N-1)	⋯	(N-1,N-1)

[1] n,m は Mx+x を構成する場合の基数の選び方によって異なる.

基本的に Torus 構造に対して、バイパスリンクを定める場合、最も効果的なのは対角線方向である。今、各ノード (x, y) が、$(x \pm n, y \pm n)$ と結ぶ 4 本の付加リンクを持つとすると、この付加リンクは新たな Torus 構造を形成する。ここでは、n のことを**基数**と呼ぶ。n は任意の値をとることができるが、ここでは 2 に限定して説明する。

図 1(a) に、n=2 とした場合に形成される新たな Torus 構造を示す。これらの 4 本のリンクを、その形状と接続するノードの位置から x2 と呼び、形成される上位 Torus を x2 Torus と呼ぶ。Mx+x は、この x2 Torus について、さらに、±2 の上位 Torus を作る (図 1(b))。この上位 Torus は、リンクの方向は基本とする Torus と並行で、ノード間の距離が 8 であることから、+8 Torus と呼ぶ。同様に+8 Torus に対して上位 Torus を形成すると、x16 Torus となる。Mx+x は、このように上位 Torus を再帰的に構成していく。(この名前付けの方法に従うと、基本となる Torus は+1 Torus と呼ぶことができる。)

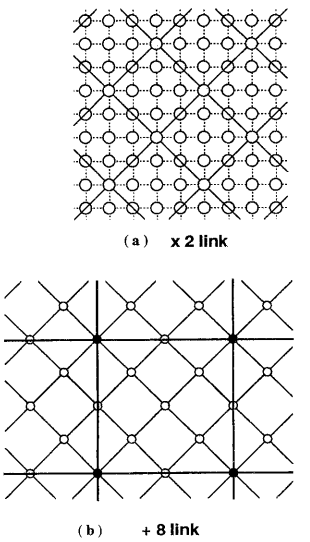

図 1: Mx+x の構成の原理図

図中には形成可能な上位 Torus の一部しか示していないが、全てのノードについて、上位 Torus の形成は可能であり、各ノードで全ての上位 Torus に対応するリンクを持ったら、degree は膨大になってしまう。ここでは、Mx+x の各ノードは基本となる +1 Torus と、上位 Torus のどれか 1 つに対応するリンクを持つと定める。このことにより、各ノードの degree は 8 になる。システムサイズが極端に大きくなれば、ノードに付け加える Torus の数を増やし、degree を 12,16,... とすることは可能であるが、現在のシステムの規模 (2^{16}程度) を考えると、8 で十分対処可能である。

ここで、基数を 2 に取った場合、下位 Torus 上には、独立な上位 Torus が 8 つ形成できる。図 2中にこの様子を示す。形成できる Torus を全て描くと混乱するため、代表を

1つ描き、後はノードの記号を変えてある。同一記号同士のノードを結ぶと8つTorusが形成できることがわかる。すなわち、各ノードはどの上位Torusに属するか（あるいは、どのTorusを持つことができるか）により、図2に示すような8種類に分類されることになる。ここで、図中の白丸で示すノードを基準にし、+1 Torusの番号付けにならって、上位Torusに対して番号を付ける。さらに、上位のTorusにも同様に番号を付けることにより、任意のTorusに下のような識別子を与えることができる。

```
(x2 Torus 番号,+8 Tours 番号,x16 Torus 番号,...)
```

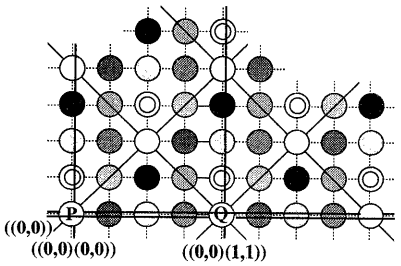

図 2: 形成される上位Torusの番号付け

識別子は、上位のTorusほど長くなる。これは、形成できるTorusの数が上位に成るほど8倍ずつ増えるためである。例えば、図2中のP印のx2 Torusを((0,0))と呼ぶ。そのx2 Torus上に再帰的に作った+8 Torusのうち、相対的にP印の位置に当たるTorusは、((0,0)(0,0))、Q印の位置に当たる+8 Torusは((0,0)(1,1))となる。当然のことながら、あるノードが定まれば、そのノードの持つことの出来る上位Torusは、それぞれの再帰的に構成したレベルについて、一意に定まる。

ここでは、各ノードは基本となる+1 Torusの他に一種類しか上位Torusを持てないため、どのノードがレベルのTorusを持つかが問題になる。この選択については、以下のトレードオフが存在する。まず、上位Torusの種類および数を増やすと、diameterは小さくすることができるが、交信の局所性に対処できず、平均ノード間距離の点で不利になる場合も生じる。ノードによりさまざまなTorusを用いれば、平均ノード間距離の点で有利だが、ルーティングアルゴリズムが複雑になる。

本論文では、これらのトレードオフを考慮し、以下のようなTorusによるMx+xについて検討する。

- x2: ((0,1)), ((0,2)), ((1,0)), ((1,3))
- +8: ((0,0)(*,*)), ((0,3)(*,*))
- x16: ((3,0)(*,*)(*,*)), ((3,3)(*,*)(*,*))

ここで、(*,*)は、そのレベルで取り得るすべてのTorusを意味する。すなわち、この方法では、((0,0))、((0,3))のx2 Torusに対応する全てのノードが+8 Torusのために用いられ、((3,0)), ((3,3))のx2 Torusに対応する全てのノードがx16のために用いられる。すなわち、x2 Torusが決まった時点で、どの上位Torusを持つか定まる。これは、上位でさらに用いるTorusを変えると、ルーティングアルゴリズムが複雑になるためである。また、さらに上位のTorusである+64 Torusは用いず、x2 Torusの数を他の上位Torusに対して倍の数、用意している。これは、ノード間の平均距離と近接交信を考慮したためである。上記の構成のMx+xのリンク構成を図x2結線図,+8結線図,x16結線図に示す。

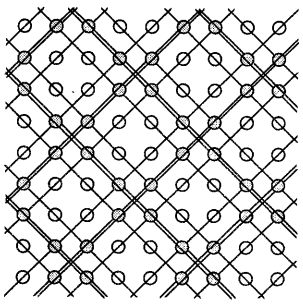

図 3: x2結線図

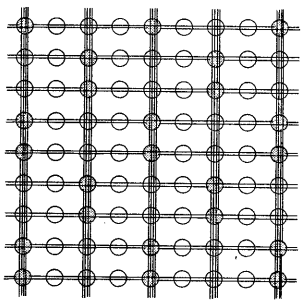

図 4: +8結線図

ここで、基本となる+1 Torusの番号に対する結合アルゴリズムの詳細を示す。条件部はC言語に準拠する。$(x,y) \rightarrow (u,v)$はノード(x,y)と(u,v)間にリンクを設けることを意味する。

```
if ((x%4==0 && ( y%4==0 || y%4==3 )) ||
    (x%4==2 && ( y%4==1 || y%4==2 ))) {
                                /* ノード+8 */
   (x,y) -> (x+8,y);            /* b1 */
   (x,y) -> (x-8,y);            /* b2 */
   (x,y) -> (x,y+8);            /* b3 */
   (x,y) -> (x,y-8);            /* b4 */
}else{
```

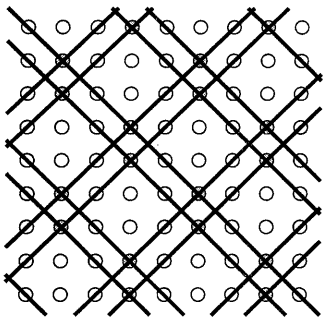

図 5: x16 結線図

```
   if ((x%4==3 && ( y%4==0 || y%4==3 )) ||
       (x%4==1 && ( y%4==1 || y%4==2 ))) {
                              /* ノード x16 */
      (x,y) -> (x+16,y+16); /* a1 */
      (x,y) -> (x-16,y+16); /* a2 */
      (x,y) -> (x+16,y-16); /* a3 */
      (x,y) -> (x-16,y-16); /* a4 */
    }else{                    /* ノード x2 */
      (x,y) -> (x+2,y+2);
      (x,y) -> (x-2,y+2);
      (x,y) -> (x+2,y-2);
      (x,y) -> (x-2,y-2);
    }
}
```

最も遠い遠隔ノードへの経由ノード数 (diameter) と、シミュレーションにより求めた平均距離を表 1に示す。

表 1: Mx+x 網の diameter と平均距離

node 数	256	1024	4096	65536
diameter	7	8	9	15
平均距離	4.076	5.244	6.445	10.768

2.2 ルーティングアルゴリズム

Mx+x 網のノードは、それぞれに対して上位 Torus が決まっており、その種類により、ルーティングアルゴリズムが異なる。ここでは、まず、アウトラインを示す。

dx = 目的ノードの番号の x 座標 - 出発ノードの番号の x 座標;
dy = 目的ノードの番号の y 座標 - 出発ノードの番号の y 座標;

```
if ( dx,dy > 範囲 a ) {
   最も近い x16 ノードへ行き、遠隔リンクを利用する。
}else{
   if ( |dx|+|dy| < 範囲 b ) {
      基本となる Torus 結合を利用する。
   }else{
      if ( |dx|,|dy| < 範囲 c ) {
         最も近い x2 ノードへ行き、遠隔リンクを
         利用する。
      }else{
         最も近い+8 ノードへ行き、遠隔リンクを
         利用する。
      }
}
```

ここで範囲 a, 範囲 b, 範囲 c はそのノードの持つ上位 Torus の種類によって、異なる数字になる。次に、例として、x16 ノードに対してのルーティングアルゴリズムを以下に示す。

```
/* Routing for node type x16a */
if ( ( |dx+1|>(16+4) && |dy|>4 ) ||
     ( |dx+1|>(8+4) && |dy|>(8+4) ) ||
     ( |dx+1|>4 && |dy|>(16+4) )     )   {
   funx16();          /* moving on x16 net */
}else{
   if (|dx+1|<=4 && |dy|<=4 )   {
      if (|dx|+|dy| <= 3) {
          funMesh();  /* moving on Mesh net */
      }else{
         if (dx < -dy) {
            x=x-1; y=y; /* to node type x2b */
         }else{
            x=x; y=y+1; /* to node type x2c */
         }
      }
   }else{
      x=x+1; y=y;       /* to node type +8a */
   }
}
funx16()     /* move on the x16 net */
{  if (dx>=0)  x=x+16; else x=x-16;
   if (dy>=0)  y=y+16; else y=y-16;
}
funMesh()    /* move on Mesh net */
{  if (dx==0) {
      x= x; y= dy>0 ? y+1 : y-1;
   }else{
      x= dx>0 ? x+1 : x-1;  y=y;
   }
}
```

現在のところこのルーティングアルゴリズムは最適ではなく、また、各ノードの持つ Torus の種類を各ノードに持たせるようにすれば、より簡単化される。表 1 に示す diameter はこのアルゴリズムによるものである。

3 基板の配置と結線

Mx+x のノード間の結線は、基板の配置を工夫することにより、基板間の配線が単純化されるように構成されている。ここでは、現在の実装技術を考慮の上、一つの基板上に 16 ノードが置かれ、4×4 のメッシュを構成すると仮定する。まず、全体のノードを図 6に示すように短冊型に切り、基板をびょうぶ状に折り畳み、基本ブロックを構成する。

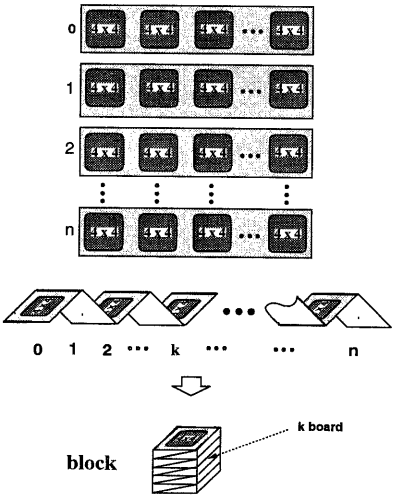

図 6: 分解図

次にこのブロックを図 7のように、立体的に配置し、螺旋状に番号が増えるように、ブロックを割り当てる。以上のような配置により、結線は、例外を除いて基板と垂直または、同一平面上になり、斜めの線をできる限り排除する。

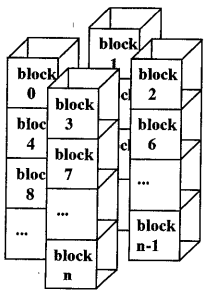

図 7: 配置図

まず、基板上のノードの配置を図 8に示す。ここで、x2 ノード間には基板上での斜めの配線が必要になる。しかし、折り畳みと螺旋状の配置により、基板間をまたがる配線は、基板と同一平面上で基板内と同一の方向に収まる。

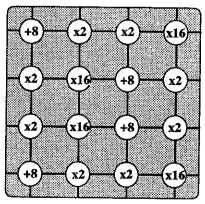

図 8: board

次に、x16 の配線は、対応するノードが必ず立体的に同一の位置に配置されるため、図 9に示すように、基板と垂直の線で実現される。図は、中央の x16 ノード (黒丸) からの遠隔リンクを示し、a1-a4 は結線アルゴリズム上で示したリンク名に対応する。ただし、基板には垂直方向の線を通すフィードスルーが必要になる。

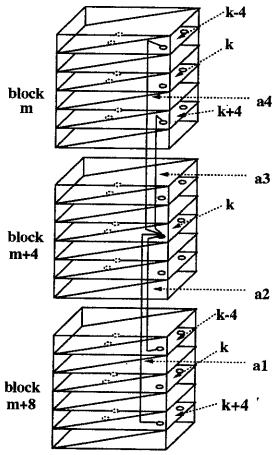

図 9: x16 ノード同士の結線

図 10は+8 ノード同士の結線を示す。ここでも、4 本のリンクのうち、3 本は同一平面上での実装が可能だが、b3 リンクのみは、立体的に斜めの配線が必要になる。

Mx+x において、これらの遠隔リンクは、種類別に全て方向が揃っており、+8 のリンク一つを除いては、立体的に斜めの線はなく、単純な配線パターンで実現できる。また、x2,+8,x16 の三種類のノードで結線方向が異なるため、基板上で、結線の過密部分、過密方向を生じない利点もある。また、Mx+x の遠隔リンクは、以下の性質を持つ。

性質

4 × 4 の格子 (基板) に分割した場合、同一基板上のすべての+8 ノードの同一種類のリンクは、他の同一基板上の

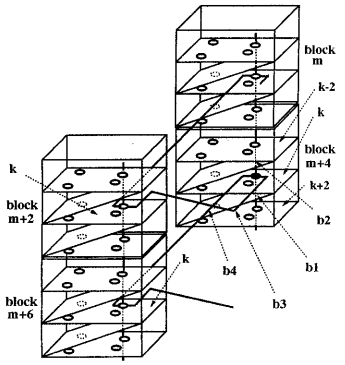

図 10: +8 ノード同士の結線

同一相対位置のノードとの間に接続される。同様の性質はx16 ノードに対しても成立する。

基板上のノードに、全体の格子と同様に、相対位置 (bx,by) を与えると、block n の中の基板 k のノードの番号 (x,y) は、$(4*k+bx, 4*n+by)$ で表すことができる。(n,k,bx,by は正の整数、$bx \leq 3, by \leq 3$)。

さて、今+8 ノード (x,y) は、$(x\pm8, y), (x, y\pm8)$ と接続される。この x,y を n,k,bx,by を用いて表すと、$(4*k+bx\pm8, 4*n+by), (4*k+bx, 4*n+by\pm8)$ となる。このノードは $(4*(k\pm2)+bx), 4*n+by), (4*k+bx, 4*(n\pm2)+by)$ となる。したがって、同一基板 ($k\pm2$)、同一相対位置 (bx,by) であることがわかる。x16 についても同様である。

この性質は実装を容易にするだけでなく、後に述べるように、ルータチップの導入を可能にする。

4 他のネットワークとの比較

Mx+x の diameter、degree を Hypercube および、Mx+x 同様 Mesh を基本とした超並列計算機用結合網 SNT[8] と比較した結果を下に示す。Mx+x, SNT の degree はシステムのサイズによらず一定 (ここでは 8) であるが、Hypercube の degree はシステムのサイズによって増加するため、括弧内に数値を示した。

表 2: Mx+x と Hypercube の diameter 比較

node 数	256	1024	4096	65536
Hypercube	8 (8)	10 (10)	12 (12)	16 (16)
SNT_X	5	7	10	—
SNT_W	5	6	8	—
Mx+x	7	8	9	15

これによると、Mx+x の diameter は Hypercube よりも小さく、SNT とほぼ同等である。正確には、同じ degree の SNT_X よりやや大きく、SNT_W よりやや小さい。ただし、これは、本論文で検討した Mx+x の構成が、交信の局所性と実装の容易さを重視し、+64 Torus を用いず、x2 Torus を二倍の密度に設定しているためであり、diameter のみを考えて構成すれば、さらに小さくすることが可能である。しかし、本論文で検討した構成を用いても、Hypercube の diameter に比べ、サイズが 2^{16}の範囲ならば、小さい値を得ている (サイズが 2^{16}の場合 degree は半分で済む)。

一方でルーティングに関しては Mx+x の方が Hypercube に比べ、複雑である。しかし、故障時の経路変更等を考慮すると、ルーティングは、何らかの形でテーブル引きを行なうことが予想され、多少の複雑さは許容される。また、現在、各ノードに自分の持っている上位 Torus の番号を保持させることにより、簡単にルーティングを行なう方法を検討中である。

5 遠隔リンク用ルータチップの導入

今回検討した Mx+x の構成は、他の多くの結合網に比べ実装が容易である。しかし、超並列計算機の結合法としては、リンク数 8(遠隔リンクが基板当たり 64) でも過大であるとする意見もある。また、x1 Torus は近接リンクで済むのに対し、x2, +8, X16 Torus 用のリンクは単純パターンとはいえ、ある程度遠くの基板に対して接続される必要がある。このため、近接リンクと遠隔リンクを同一 bit 数で同一交換チップで扱うのは、実装上無理がある。

そこで、遠隔リンク専用のルータチップを設け、遠隔リンクを一括して扱うことを考える。ここで、先に示した Mx+x の性質が利用される。同一基板上の同一種類のノードの遠隔リンクはかならず、同一基板に対して接続されるため、これを共有することができる。Mx+x とリンク数のバランスを取るためには、

- x2 のリンクに対して 8 本
- +8 のリンクに対して 4 本
- x16 のリンクに対して 4 本

の 1 基板当たり 16 本のリンクが必要になる。このうち、x2 のリンクのうち半分の 4 本は自分の基板内にフィードバックされる。最も基本的な実装法は、図 11に示すように、32×32 の交換を行なうルータを用いる方法である。交換 bit 数が小さければ、32×32 程度の交換を行なうことのできるルータは 1 チップに収めることができる [10]。さらに、我々は SSS(Simple Serial Synchronized) 型スイッチ [11] で提案した多出力型の交換スイッチ [12] の利用を検討しており、衝突の少ない交換が低コストで可能になる。この方法を用いると、1 ノード当たりのリンク数は 5 となり、基板間の遠隔リンク数も 1/4 になる。同様な方法で、耐故障性も考え、8 ノードに対して 16×16 のルータを用意する構成も考えられる。ただし、この方式では、1 ノード当たりのリンク数には変化はないが、基板当たりの遠隔リンク数は 32 本になり、削減の効果は小さい。

以上提案したルータを用いる方法は、本論文で検討した構造と同じ diameter、平均距離での交信が可能である

が、リンクが共有されるため、各リンクの負荷が増大する。どの程度性能に差が生じるかは、大規模なシミュレーションが必要である。

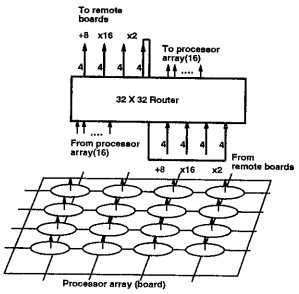

図 11: ルータを用いた実装法

6 おわりに

Mesh を内蔵し、実装が容易で、かつ degree,diameter に優れた結合網のクラス Mx+x を提案し、その構成のひとつに関して実装法、diameter, 平均距離等を検討した。

Mx+x は用いる上位 Torus の組合せにより様々な構成を取ることが可能である。このことは、対象とするシステムの交信パターンにあわせた構成を取ることによる性能の向上が期待でき、魅力的である。本論文で検討した構成は、x2 Torus が他の倍の密度で存在するため、近傍ノードに対する結合が密である。これは、交信の局所性に対する配慮であるが、x2 のリンク数が過大に設定されている可能性もある。x2, +8, x16, +64 の各上位 Torus を均等に持つ構成や、その他考えられる様々な構成の性質についても検討する必要がある。

また、それらの構成に対するルーティングアルゴリズムの簡単化も重要な課題である。さらに Mx+x は、階層的な Torus 構成を基本とするため、階層構造を潜在的に持っている。このため、階層構造やグローバルバスを持つメッシュに対するアルゴリズム ([13] 等) が使える可能性があり、ルーティングアルゴリズムとも関連して検討中である。

謝辞

本研究のスタートにあたって有意義な議論をいただいた Fordham 大学の D.Frank Hsu 教授に感謝いたします。また、論文に対し有意義なコメントをいただいた慶応大学理工学部寺沢卓也氏、木村哲郎氏に感謝いたします。

参考文献

[1] W.C.Athas, C.L.Seitz, Multicomputers: Message-Passing Concurrent Computers, IEEE Computer, Vol. No. 8, Aug. 1988

[2] Intel Corp. Paragon XP/S product overview, 1991.

[3] K.Kurihara, D.Chaiken, A.Agarwal, Latency Tolerance through Multithreading in Large-Scale Multiprocessors, Proc. of Int'l Symp. on Shared Memory Multiprocessing (ISSMM), pp. 91-101, Tokyo, Apr. 1991

[4] H.Ishihata,T.Horie,S.Inano,T.Shimizu, S.Kato,and M.Ikesaka, "Third Generation Message Passing Computer AP1000," International Symposium on Supercomputing, Nov. 1991, pp. 46-55.

[5] 松本 尚、平木 敬. "超並列計算機上の共有メモリアーキテクチャ". CPSY92-26. pp. 47-55.

[6] 田辺 昇、小柳 滋. "三次元実装に基づくマルチパラダイム超並列テラフロップスマシンのアーキテクチャ" CPSY92-24. pp. 33-40.

[7] 奥川峻史、"超並列コンピュータ向き結合網の検討" 信学技報、CPSY91-13 (1991).

[8] 岩崎 一彦、イゼリ クリスチャン、 佐藤 裕二 "超並列計算機用 VLSI に適した結合網の一提案". 電子情報通信学会 論文誌. vol.J75-D-I NO.8 AUGUST. 1992.

[9] 武本 充治、松本 尚、平木 敬. "Mesh+: 放送機能を持った高信頼メッシュ型ネットワーク". CPSY92-28. pp. 65-70.

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

[11] Hideharu Amano, Luo Zhou, Kalidou Gaye, "SSS (Simple Serial Synchronized)-MIN; a novel multi stage interconnection architecture for multiprocessors" Proceedings of the IFIP 12th World Computer Congress Madrid, Spain, 7-11 September 1992. pp. 571-577.

[12] 天野 英晴、藤川 義文, "マルチステージネットワーク: PBSF (Piled Banyan Switching Fabrics)" 情報処理学会研究報告, 92-ARC-94-5. pp.33-40.

[13] D. Carlson, "The Mesh with a Golbal Mesh," Proc. on 1st International Conference on Supercomputing systems, pp.615-627, Dec.1985.