アルゴリズム 16−12
（1990．7．17）

バスへの同時送信に対する排他制御

中野浩嗣† 増澤利光‡ 都倉信樹†

†大阪大学基礎工学部

‡大阪大学情報処理教育センター

あらまし

バスには，同時送信の際の処理に関して種々のモデルが提案されている．その中でもArbitraryモデルが最も強力であるが，実現は困難であると見られている．本稿では，Arbitraryモデルのバスで行われる任意の通信が，定数倍のハードウェア量のCommonモデルのバスに置き換えて，定数時間で実現できることを示す．よって，通信を実現するのに要する時間とハードウェア量の定数倍の差を無視すれば，Arbitraryモデルと Commonモデルの能力差はないといえる．

Mutual Exclusion for Simultaneous Sending to Buses

Koji NAKANO† Toshimits MASUZAWA‡ Nobuki TOKURA†

†Faculty of Engineering Science, Osaka University

‡Education Center for Information Processing, Osaka University

††‡Toyonaka-shi, Osaka 560, Japan

ABSTRACT

Several bus models are proposed with different capabilities intended for management of simultaneous sending. An arbitrary model, the most powerful bus model, has been considered to be difficult to realize. We show methods to simulate the arbitrary model in constant time by using the common model that is more feasible. Therefore, if we can ignore the constant factor, there is no difference in their capabilities between the arbitrary model and the common model.

1．まえがき

1次元格子状や2次元格子状に並んだプロセッサ（それぞれ，単に1次元格子，2次元格子と呼ぶ）を，大域的な通信の行えるバスにより接続したモデルの研究は最近よく行われている．例えば，ソーティング問題やグラフに関する問題を効率よく解くアルゴリズムが知られている．1つのバスに複数のプロセッサがデータの送信を行うと，送信データの衝突がおこる．文献(5)や(9)では，このような同時送信が行われた場合の処理に関して，Exclusive（同時送信を許さない），Common（同じ値ならば同時送信を許す），Arbitrary（同時送信の場合，任意の1つの値が実際に通信される），Priority（同時送信の場合，最も優先順位の高いプロセッサの送信した値が実際に通信される）などのモデルを提案している．他のモデル上でのアルゴリズムが修正なくそのまま動くという意味でPriority，Arbitrary，Common，Exclusiveの順に能力が高い．逆にハードウェアでの実現性に関しては，Exclusive，Common，Arbitrary，Priorityの順に実現が容易であると考えられている．バスを用いたプロセッサネットワーク上で種々の問題を高速に解くアルゴリズムを開発されている[(1)(2)(4-11)]．

本稿では，Arbitraryモデルのバスを，より実現性の高いバスを用いて実現する方法を，同時送信の回避に注目して述べる．そのためには，どのようなハードウェアが利用できるかを厳密に定める必要がある．バスとして利用できるのは，1単位時間に1ビットだけ同時通信可能なCommonモデルのバス（1ビットCommonバスと呼ぶ）のみとする．1ビットCommonバスは，現在の技術でも，実現化はそれほど困難ではない．

wビットのデータが同時通信可能なCommonモデルのバスを実現するには，1ビットCommonバスをw本並べればよい．ところが，Arbitraryモデルの場合，1ビットCommonバスをw本並べただけでは，実現できない．Arbitraryモデルを実現するためには，そのバスに対して送信命令を実行するプロセッサの中から1つを任意に選ぶ必要がある．例えば，1ビットCommonバスを用いて，2分探索の手法を用いれば，nをそのバスに接続するプロセッサの数とすると，O（logn）時間で送信命令を実行するプロセッサの中から1つを選ぶことができる．この方法では，1ビットCommonバスをw本並べれば，wビット同時転送可能なArbitraryモデルのバスで行われる1回の通信をO（logn）時間で実現できる．しかし，バスによるプロセッサネットワーク上のアルゴリズムは，ほとんどO（logn）やO（$\log^2 n$）といったような計算時間であり，O（logn）倍のオーバヘッドは無視できない．

本稿では，wビット同時転送可能なArbitraryモデルのバスで行われる通信を定数時間で実現する2つの方法を述べる．これらの方法は，送信命令を実行するプロセッサの中から1つを任意に選ぶために，1ビットCommonバスを層状に巧妙に配置したものである．特に3節と4節で，1ビットCommonバスのみ用い，送信命令を実行するプロセッサの中から1つを定数時間で任意に選ぶ方法を述べる．このとき，ハードウェア量として，バスを配置するのに要する総面積を考えるならば，$w=\Omega(\log n)$のとき，wビットのデータを同時転送するためのw本の1ビットCommonバスと，その高々定数倍のハードウェア量の1ビットCommonバスを付加すればよい．よって，通信を実現するのに要する時間とハードウェア量の定数倍の差を無視すれば，ArbitraryモデルとCommonモデルの能力差は無いといえる．

2．モデル

図1のように，プロセッサを直線上に並べ，バスをその上に配置したものをバス付き1次元格子と呼ぶ．プロセッサは，通信命令を付加して拡張されたRAMであり，各プロセッサは同期して動作する．配置されるバスは，すべて1ビットCommonバスであるものとする．プロセッサをPE(0),PE(1),...,PE(n-1)と表す．バスは層状に配置されており，下から順に第1層，第2層,...と呼ぶ．図1の第3層はPE(0),...,PE(7)の上を通過するバスとPE(8),...,PE(15)の上を通過する2つのバスから構成されている．これらのバスをそれぞれ，[0,7]，[8,15]と区間で表す．そして，第3層に配置されるバスを集合{[0,7]，[8,15]}と表す．つまり，PE(s),PE(s+1),...,PE(t)の上を通過するバスを[s,t]と表し，1つの層のバス配置は，そこに配置されるバスの集合で表す．バス付き1次元格子やバスモデルに関する厳密な定義は他の文献（例えば(8)(9)）を参照されたい．

---

† f,gを$\mathbb{N}$（自然数全体からなる集合）から$\mathbb{R}$（実数全体からなる集合）の関数とする．ある正定数c, n_0が存在し，任意の$n \geq n_0$に対して$f(n) \leq c \cdot g(n)$が成り立つとき，$f(n)=O(g(n))$と記す．逆に，$f(n) \geq c \cdot g(n)$が成り立つとき，$f(n)=\Omega(g(n))$と記す．また，対数関数logの底は，2とする．

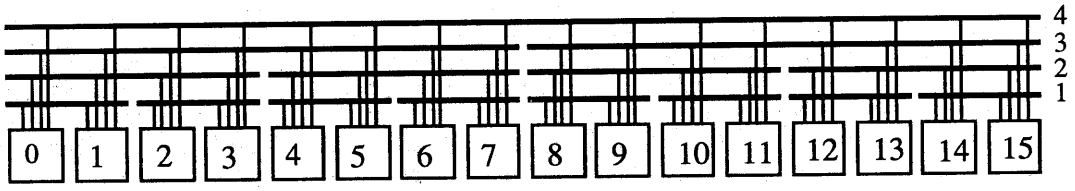

図1　2分木配置

次に，Arbitraryモデルのバスで行われている通信をシミュレートするために，バス付き１次元格子上で送信命令を実行しようとするプロセッサの中から１つを任意に選ぶ方法を示す．ここで注意しなければならないのは，任意に１つ選ぶというのは，送信命令を実行しようとするプロセッサから等確率に１つ選ぶということではない．シミュレートするアルゴリズムにとって，都合がいいように１つ選べば，Arbitraryの機能が実現できることになる．ここでは，送信命令を実行しようとするプロセッサのうち，最も右端のものを選ぶことにする．厳密には次のように形式化される．

［定義１］（最右探索）

（入力）各PE(i)（$0 \leqq i \leqq n-1$）に，0または1（これをinp(i)と表す）が任意に与えられる．これは，inp(i)＝1のとき，PE(i)は送信命令を実行することを意味する．

（出力）1を与えられたプロセッサの中で最も番号の大きいプロセッサを求める．つまり，各PE(i)は次のout(i)を得る．

$$out(i)=1 \quad (i=\max\{k \mid inp(k)=1\})$$
$$=0 \quad (i\neq\max\{k \mid inp(k)=1\})$$

このとき，PE(i)を**最右プロセッサ**と呼ぶ．但し，inp(k)＝1であるようなkが存在しないとき，全てのiについて，out(i)＝0とする．　□

３．最右探索の実現法１

図１のようなバス配置を２分木配置という．次に２分木配置において，最右探索を定数時間で行う方法を述べる．２分木配置は一般には次のようになる．

（２分木配置）

●第j層（$1 \leqq j \leqq \log n$）のバス配置は，

$\{[0, 2^j-1], [2^j, 2\cdot 2^j-1], \ldots, [n-2^j, n-1]\}$

である．

●各PE(i)（$0 \leqq i \leqq n-1$）は，第1層～第$\log n$層においてその上を通過する全てのバスと接続する．

但し，nは２のべきとする．nが２のべきでない一般の場合は，$\log n$を$\lfloor \log n \rfloor$にするといったような適切な修正を加えることにより，同様に最右探索が行える．

次に２分木配置で最右探索を行うアルゴリズムを示す．このアルゴリズムは，並列に２分探索を行うもので，この手法によるＰＲＡＭ上のアルゴリズムが，文献(3)に示されている．

２分木配置の各バス（[s,t]と表す）にはPE(s), PE(s+1),...,PE(t)が接続するが，PE(s),PE(s+1),...,PE((s+t-1)/2)は[s,t]の左側に，PE((s+t+1)/2),...,PE(t-1),PE(t)は右側に接続するという．

第$\log n$層のバスは，[0,n-1]であるが，inp(i)＝1かつ[0,n-1]の右側に接続する各PE(i)（$n/2 \leqq i \leqq n-1$）は，バス[0,n-1]に1を送信し，左側に接続する各PE(j)（$0 \leqq j \leqq n/2-1$）はバス[0,n-1]に受信命令を実行する．このとき，inp(i)＝1かつ右側に接続するPE(i)が存在すれば，受信命令により1を受信するので，左側に接続するプロセッサは自分が最右プロセッサでないことを知ることができる．いま，inp(i)＝1かつ右側に接続するPE(i)が存在する場合を考える．このとき，第$\log n$-1層のバス[n/2,n-1]を用いて，同様のことを行えばよい．以下同様に下の層に対して繰り返し，１回の繰り返しごとに，最右プロセッサの候補を半分に絞っていけば，最右プロセッサを求めることができる．繰り返しは$\log n$回行われるので，Ｏ（$\log n$）時間要とする．各層に対して，順に行われる送受信を同時に行うことにより，定数時間で終了するようにしたのが，次のアルゴリズムである．

［最右探索アルゴリズム１］

inp(i)＝0であるPE(i)はout(i):=0として，なにも行わない．inp(i)＝1である各PE(i)（$0 \leqq i \leqq n-1$）は，次のステップ１と２を順に実行する．

（ステップ１）第1層～第$\log n$層の各バスに同時に次の

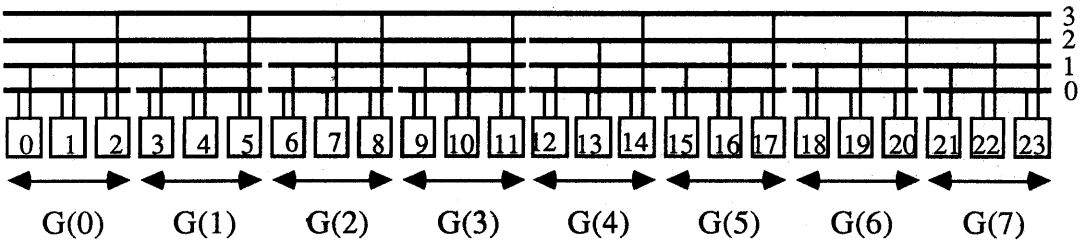

図2　変形2分木配置

動作を行う．
①第j層（1≦j≦logn）において，バスの右側に接続するとき，そのバスに1を送信する．
②第j層（1≦j≦logn）において，バスの左側に接続するとき，そのバスからデータを受信する．
（ステップ2）ステップ1の②において，1を受信することがなかったとき，つまり，受信命令を実行した全てのバスからデータを受信できなかったとき，out(i):=1とする．それ以外の場合out(i):=0とする．　□

最右探索アルゴリズム1では，ステップ1で，各プロセッサがlogn本のバスに対して送受信を同時に行う．また，ステップ2において，最大logn個の1ビットのデータに対して論理和を求める必要がある．一般に，RAMであるプロセッサにはこのような並列動作は認めないことが多い．この点に関する議論は5節で行う．

次の定理が成り立つ．

［定理1］最右探索アルゴリズム1は，2分木配置付き1次元格子上で，最右探索を定数時間で行う．

（証明）d=lognとし，dの帰納法で正当性を証明する．
基底段階　d=1のとき，明らかに成り立つ．
帰納段階　d-1以下のとき成り立つとして，dのときにも成り立つことを証明する．

プロセッサ数がnの2分木配置は，プロセッサ数がn/2，層数logn-1の2分木配置を2つならべ，第logn層に全てのプロセッサと接続するバス[0,n-1]を配置したものとみなせる．そこで，PE(0),PE(1),...,PE(n/2-1)上の層数logn-1の2分木配置を左2分木配置と呼び，PE(n/2),PE(n/2+1),...,PE(n-1)上の2分木配置を右2分木配置と呼ぶことにする．

いま，最右プロセッサをPE(r)とする．次の2つの場合に分けて考える．
①0≦r≦n/2-1のとき．

帰納段階の仮定より，左2分木配置において，PE(0),...,PE(n/2-1)上の最右探索が正しく行える．

②n/2≦r≦n-1のとき，

inp(i)=1であるPE(i)（0≦i≦n/2-1）は，第logn層から1を受信するので，out(i)=0である．PE(r)が属する右2分木配置において，帰納段階の仮定より，正しく最右探索が行える．よって，PE(0),...,PE(n-1)において正しく最右探索が行える．　□

4．最右探索の実現法2

ここで述べるアルゴリズムは，最右探索を定数時間で行う．しかし，3節で述べたアルゴリズムと異なり，1単位時間に高々1回ずつの送受信命令を行うだけである．

図2で表される配置を変形2分木配置と呼ぶ．一般には次のようになる．

（変形2分木配置）

●PE(0),...,PE(n-1)と表されるが，アルゴリズムの表記を簡単にするため，$n=d\cdot 2^d$を満たすdを用い，プロセッサを$PE\langle i,j\rangle$（$0\leqq i\leqq 2^d-1, 1\leqq j\leqq d$）と表す．このとき，$PE\langle i,j\rangle$は$PE(i\cdot d+j-1)$に相当する．

●第k層（$0\leqq k\leqq d$）層（注意：ここまでは，下から第1層，第2層と層番号を数えたが，ここでは，記法を簡便にするため，第0層，第1層と数える）のバス配置は，

$\{[0,d\cdot 2^k-1],[d\cdot 2^k,2\cdot d\cdot 2^k-1],\ldots,[n-d\cdot 2^k,n-1]\}$

である．

●各$PE\langle i,j\rangle$は，第0層と第j層のバスとのみ接続する．

$n=d\cdot 2^d$を満たす整数dが存在しないときにも，前に述べたような適切な修正を加えることにより，以下に示すアルゴリズムは同様に実行できる．変形2分木配置とは，$PE\langle i,1\rangle,\ldots,PE\langle i,d\rangle$を一つのプロセッサとみなすと，2分木配置と同じになる．各i（$0\leqq i\leqq 2^d-1$）について，$PE\langle i,1\rangle,\ldots,PE\langle i,d\rangle$を第iグループと呼び，G(i)と表すことにする．2分木配置では，各プロセッサがlognのバスと接続したが，変形2分木配置では，

PE<i,1>,...,PE<i,d>がそれぞれ第1層～第d層に接続することにより役割を分担する．また，各プロセッサは第0層のバスと接続するが，これは，各グループ内で情報を交換するのに用いる．

次に，変形２分木配置上で最右探索を行うアルゴリズムを述べる．まず，その概略を述べる．

［最右探索アルゴリズム２の概略］

（ステップ１）各G(i)ごとに，1を与えられたプロセッサが存在するか否か調べる．つまり，

inp<i,1> or inp<i,2> or･･･or inp<i,d>

を求める．これをu(i)と表す．ここで，各inp<i,j>は，PE<i,j>に与えられる入力，つまりinp(i･d+j-1)を意味する．

（ステップ２）u(0),...,u(2^d-1)のうち最も右端の1を求める．つまりmax {s | u(s)＝1} （＝Rと表す）を求める．このとき，最右プロセッサは，G(R)に属する．

（ステップ３）PE<R,1>,...,PE<R,d>のうち，最も右端の1を求める．つまり，max {s | inp<R,s>＝1} （＝rと表す）を求める．このとき，PE<R,r>が最右プロセッサである．　（概略終わり）

ステップ２には，最右探索アルゴリズム１のアイデアを用いる．ステップ３に関して，若干の補足を行う．いま，inp<R,j_1>＝inp<R,j_2>＝･･･＝inp<R,j_k>＝1（j_1＜j_2＜･･･＜j_k）であるとし，他のinp<R,j>はすべて0であるとする．このとき，PE<R,j_k>が最右プロセッサである．PE<R,j_1>,PE<R,j_2>,...,PE<R,j_k>がそれぞれ第j_1層～第j_k層において接続するバスをB(j_1),B(j_2),...,B(j_k)とすると，B(j_k)が最も長く，このバスは，B(j_1),B(j_2),...,B(j_{k-1})を含むような形になる．この性質を利用し，最右プロセッサであるPE<R,j_k>が接続するバスB(j_k)を求める．

次にアルゴリズムの詳細を述べる．第j層のバスがPE<s,j>,PE<s+1,j>,...,PE<t,j>に接続するとする．このとき，PE<s,j>,PE<s+1,j>,...,PE<(s+t-1)/2,j>はこのバスの左側に接続し，PE<(s+t+1)/2,j>,...,PE(t-1,j>,PE<t,j>は右側に接続するという．

［最右探索アルゴリズム２］

各PE<i,j>は作業用変数として，ローカルメモリに，u(i),v<i,j>,w<i,j>およびw(i)をもつ．また，これらの作業用変数は0で初期化されているものとする．

（ステップ１）

(1-1)各PE<i,j>は，inp<i,j>＝1のとき，第0層のバスに1を送信する．各PE<i,j>は第0層のバスに受信命令を実行し，1を受信したとき，u(i):＝1，なにも受信しなかったとき，u(i):＝0とする．

（ステップ２）u(i)＝1であるPE<i,1>,...,PE<i,d>のみ次の(2-1)と(2-2)を実行する．

(2-1)各PE<i,j>は第j層のバスの右側に接続するとき，このバスに1を送信する．左側に接続するとき，このバスに対して受信命令を実行する．このとき，1を受信したならば，v<i,j>:＝1とし，なにも受信しないならば，v<i,j>:＝0とする．

(2-2)各PE<i,j>は，v<i,j>＝1のとき，第0層のバスに1を送信する．各PE<i,j>は，第0層のバスに対して受信命令を実行する．このとき，1を受信しなければ，u(0),...,u(2^d-1)のうち，u(i)が最も右端の1である．以下では，このようなiをRと表す．

（ステップ３）

(3-1)G(R)に属する各PE<R,k>は，inp<R,k>＝1のとき，第k層のバスに1を送信する．全てのPE<i,j>（0≦i≦2^d-1,1≦j≦d）は第j層のバスに受信命令を実行する．このとき，1を受信したならw<i,j>＝1とし，なにも受信しなければw<i,j>＝0とする．

(3-2)各PE<i,j>は，w<i,j>＝1のとき，第0層のバスに1を送信する．各PE<i,j>は第0層のバスに対して，受信命令を実行する．このとき，1を受信したならw(i):＝1とする．なにも受信しなければw(i):＝0とする．つまり，w(i):＝w<i,1> or w<i,2> or･･･or w<i,d>とする．

(3-3)w(0),...,w(2^d-1)の中でw(i)＝1である最も右端のものを見つける．その結果，w(RG)が右端の1であるとする．つまり，RG＝max {i | w(i)＝1} とする．これはステップ２をw(0),...,w(2^d-1)に対して実行してやればよい．

(3-4)w(0),...,w(2^d-1)の中でw(i)＝1である最も左端のものを見つける．その結果，w(LG)が左端の1であるとする．つまり，LG＝min {i | w(i)＝1} である．これは(3-3)と同様に行うことができる．

(3-5)G(RG)の各PE<RG,j>は第j層のバスの右側に接続するとき，このバスに1を送信する．G(R)の各プロセッサはこれを受信する．

(3-6)G(LG)の各PE<LG,j>は第j層のバスの左側に接続するとき，このバスに1を送信する．G(R)の各プロセッサはこれを受信する．

(3-7)G(R)のプロセッサで(3-5)と(3-6)の両方において，1を受信したものが最右プロセッサである．これをPE<R,r>とすると，このプロセッサは，out<R,r>:＝1とする．他の全てのPE<i,j>は，out<i,j>:＝0とする．

（アルゴリズム終わり）

次の定理が成り立つ．

［定理２］最右探索アルゴリズム２は，変形２分木配置付き１次元格子上で，最右探索を定数時間で行う．

（証明）各ステップは定数時間で行えるので，最右探索アルゴリズム２は定数時間で終了する．次に，正当性について述べる．ステップ２で求めたG(R)に最右プロセッサが含まれるのは，明らかである．ステップ３終了時に，正しく最右プロセッサが求められていることを示す．

先に述べたように，$inp<R,j_1>=inp<R,j_2>=\cdots=inp<R,j_k>=1$ $(j_1<j_2<\cdots<j_k)$ であるとし，他の$inp<R,j>$はすべて0とする．同じく，$B(j_1),B(j_2),\ldots,B(j_k)$をそれぞれ，$PE<R,j_1>,PE<R,j_2>,\ldots,PE<R,j_k>$が接続するバスとする．$B(j_k)$と接続するプロセッサを$PE<s,j_k>,PE<s+1,j_k>,\ldots,PE<t,j_k>$とすると，$w(s)=w(s+1)=\cdots=w(t)=1$であり，他のw(j)は0である．よって，s＝LGとt＝RGが成り立つ．$PE<RG,j_k>$は，バス$B(j_k)$の右側に接続し，$PE<LG,j_k>$は，$B(j_k)$の左側に接続するので，$PE<R,j_k>$は(3-5)と(3-6)の両方において1を受信する．次に$PE<R,j>$ $(j\neq j_k)$ が両方において1を受信することはないことを示す．

①$j>j_k$のとき，$PE<LG,j>$と$PE<RG,j>$は第j層において，同じ側に接続するので，いずれか一方は，1を送信しない．

②$j<j_k$のとき，$PE<LG,j>$が第j層において接続するバスと，$PE<RG,j>$が第j層において接続するバスは異なるので，$PE<R,j>$は両方において1を受信することはない．

よって，最右探索アルゴリズム２はステップ３終了時に正しく最右プロセッサを求める．　□

５．　実現法の比較

まず，最右探索アルゴリズム１と２を用いてwビット同時通信可能なArbitraryモデルのバスを実現することを考えてみる．w本の１ビットCommonバスに２分木配置あるいは変形２分木配置を付加すると，層数は，いずれの場合も$w+O(\log n)$となる．よって，$w=\Omega(\log n)$のとき，本質的に必要な層数であるwに高々定数倍の層数を付加することで，Arbitraryモデルのバスが定数時間で実現できることになる．また，既知のアルゴリズムでは，１単位時間に$\log n$ビット同時転送可能なバスを仮定することが多い．

最右選択アルゴリズム１と２を比較する．２分木配置では，各プロセッサは$\log n$本のバスに接続し，これらのバスに対して同時に送受信命令を実行する．一方，変形２分木配置では，各プロセッサは２つのバスに接続し，これらのバスのうち，１単位時間に高々１つのバスにしか送受信命令を実行しない．ところが，最右選択アルゴリズム１は１単位時間で終了するのに対して，最右探索アルゴリズム２は十数ステップの動作が必要である．

さらに，複数のプロセッサが複数のAribtraryモデルのバスで接続されているプロセッサネットワークで，これらのアルゴリズムを適用した場合について考えてみる．このとき，１つのプロセッサは複数のバスに接続する．最右探索アルゴリズム１では，受信命令を実行しないプロセッサは，アルゴリズムに参加しない．よって，受信命令を実行しないバスに対しては，なんの動作も行わなくてよい．ところが，最右探索アルゴリズム２では，受信命令を実行しないプロセッサも，アルゴリズムに参加しなくてはならない．よって，あるバスに対して送受信命令を実行しなくても，そのバス上の通信をシミュレートするために，最右探索アルゴリズムで定められた送受信命令を実行する必要がある．

以上より，最右探索アルゴリズム１では，$\log n$本の１ビットCommonバスに対して同時に送受信を行う必要がある．また，最右探索アルゴリズム２では，プロセッサがk本のArbitraryモデルのバスと接続するとき，それらのバスを実現するために，k本の1ビットCommonバスに対して同時に送受信を行う必要がある．いずれの場合も，プロセッサはＲＡＭであり，逐次動作のみ許すとした前提に反し，並列動作が必要である．

ところが，最右探索アルゴリズム１と２は非常に単純であり，各プロセッサの動作は，有限オートマトンで表現できる．よって，この有限オートマトンを実現するような，簡単な順序機械を各プロセッサに付加すればよい．したがって，現実への応用を考えると，実現のためにプロセッサの入出力ポートに付加する回路は，加算などを定数時間で行うために必要な回路より単純であり，ハードウェア量も小さい．

また，本稿で示した２つの方法は，最右探索によりArbitraryモデルのバスをシミュレートしているので，バスに接続するプロセッサの優先順位が昇順または降順に定められているようなPriorityモデルのバスをシミュレートすることができる．

6．むすび

本稿では，Arbitraryモデルや優先順位の制限されたPriorityモデルの現実的な実現法を示した．

既知の結果として，２次元格子の各行各列に１本づつO（logn）ビット同時通信可能なバスを配置したモデル上で，グラフの連結成分を求める次のアルゴリズムが知られている[5],[6]．

①バスモデルがCommonのとき，O（$\log^2 n$）時間
②バスモデルがCommonのとき，O（logn）確率時間
③バスモデルがArbitraryのとき，O（logn）時間

本稿で示したArbitraryバスを実現する方法を用いると，１ビットCommonバスのみ用いても，ハードウェア量を本質的に増加させることなく，O（logn）時間でグラフの連結成分を求めることができる．

ArbitraryやPriorityモデルのバスによるプロセッサネットワークは，非現実的であり，実現は容易でないとみなされていたが，本稿では，これを覆したことになる．よって，バス結合のプロセッサネットワーク上でアルゴリズムを考える際に，少なからぬ影響を与えると思われる．

文　献

(1) A.Aggarwal: "Optimal bounds for finding maximum on array of processors with k global buses", IEEE Trans. on Comput., C-35, 1, pp.62-64(1986).

(2) S.H.Bokhari:"Finding maximum on an array processor with a global bus", IEEE Trans. on Comput., C-33, 2, pp.133-139(1984).

(3) B.S.Chlebus, K.Diks, T.Hagerup and T.Radzik: "Efficient simulations between concurrent-read concurrent-write PRAM models", 13th MFCS(LNCS 324), pp.231-239(1988)

(4) 藤田聡,山下雅史,阿江忠:"多重バス結合並列プロセッサ上の最適時間ソーティングアルゴリズム", JSPP'90, pp.33-40(1990).

(5) 岩間一雄:"バス結合並列モデルと最適化アルゴリズム", 信学技報, COMP 86-53(1986).

(6) K.Iwama and Y.Kambayashi:"An O(log n) Parallel connectivity algorithm on the Mesh of Buses", Information Prosessing 89, pp.305-321 (1989).

(7) V.K.P.Kumar and C.S.Raghavendra: " Array processor with multiple broadcasting", J. of Parallel and Distributed Comput., 4, pp.173-190(1987).

(8) 中野浩嗣,増澤利光,萩原兼一,都倉信樹:"バス付き１次元格子上の並列ソーティングアルゴリズム", 信学論(DI), J72-D-I, pp.631-641(1989).

(9) 中野浩嗣,増澤利光,萩原兼一,都倉信樹:"バス付き２次元格子上の任意の通信をシミュレートする並列アルゴリズム", 信学論(DI), J73-D-I, 3, pp.269-279(1990).

(10) 中野浩嗣,増澤利光,萩原兼一,都倉信樹:"画像連結成分を効率よく求めるバス付き２次元格子上の並列アルゴリズム", 信学論(DI), J73-D-I, 4, pp.403-414(1990).

(11) Q.F.Stout:"Meshes with multiple buses", Proc. of 27th FOCS, pp.264-273 (1986).