Amaterous：経路選択法による高性能並列ルータ

田中 孝太郎[†,☆] 大野 和彦[†] 中島 浩[†]

本論文では，経路選択法による新たな並列ルータ *Amaterous* を提案する．これまでに提案された多くの並列配線手法と同様に，Amaterous には大域および詳細の 2 つの配線フェーズがある．しかし，従来手法での並列化ボトルネックである詳細配線結果の大域配線へのフィードバックが Amaterous では除去されており，2 つの配線フェーズが独立にかつ効率的に並列化されている．このフィードバック除去のために，Amaterous では c-path と呼ぶ配線容量を最大化する候補経路を，大域配線に先だってあらかじめ配線する．大域配線では，この c-path の中からいくつかを選択する経路選択法により各ネットの大域経路を決定する．詳細配線では，選択された c-path を変形することによって最終的な詳細経路を得るため，大域経路に対応する詳細経路が必ず得られることが保証される．したがって詳細配線の結果を大域配線が知る必要はなく，詳細から大域へのフィードバックを除去することができる．我々は Amaterous を 16 ノードの PC クラスタに実装し，3 種の MCM ベンチマークによる性能評価を行った．その結果，最大 12 倍の台数効果が得られ，Amaterous の効率性が実証された．また高性能並列ルータの 1 つである Amon2 に比べて，Amaterous は 1.2〜3.6 倍高速であることも明らかになった．

Amaterous: A Parallel Wire Router with Path Selection Method

Kotaro Tanaka,[†,☆] Kazuhiko Ohno[†] and Hiroshi Nakahsima[†]

This paper proposes a new parallel wire routing algorithm named *Amaterous*. As many successful parallel wire routers, Amaterous has global and detailed routers, but the feedback from the detailed to the global is removed so that both routers are parallelized efficiently and independently. For this feedback removal, Amaterous has an additional routing phase named *capacity path (c-path) router* that draws all the possible paths in each rectangular region of each routing layer prior to the global routing. Since the global router picks c-paths to form the global path of a net, it is assured that the global path is always converted successfully to its detailed counterpart which the detailed router generates from the c-paths. Therefore the global router does not need to know the result of the detailed routing and thus the feedback from detailed to global is removed. We implemented Amaterous on a 16-node PC cluster and measured its performance with three MCM benchmarks. The result shows up to 12 fold speedup proving the efficiency of our algorithm. It is also shown that Amaterous is 1.2 to 3.6 times as fast as a state-of-art parallel router Amon2.

1. はじめに

回路の大規模化と複雑化にともない回路設計 CAD の高速化が重要な課題となっており，並列処理の応用が強く期待されている．しかし CAD プログラムの並列化は，問題や洗練されたアルゴリズムに内在する逐次性のために，必ずしも容易ではない．したがって科学技術計算応用でしばしば用いられる do-all 型ループ並列化などは適用困難であり，個々の応用に特化した並列アルゴリズムが求められる．

† 豊橋技術科学大学
Toyohashi University of Technology
☆ 現在，日立マイクロソフトウェアシステムズ
Presently with Hitachi Microsoftware Systems, Inc.

たとえば我々が対象とする配線問題では，ネット間並列性を用いて複数のネット配線を並列に行う方法が考えられる．しかし配線された複数の経路の重なり合いを単純に排除しようとすると，ある配線が完了しなければ別の配線を行うことができず，アルゴリズムは逐次化されてしまう．そこで多くの並列ルータでは，ネットの大まかな経路（配線領域）を決定する大域配線と，その大域経路内で最終的な詳細経路を決定する詳細配線の 2 フェーズで処理することにより，逐次性を緩和して並列化効率を向上させている．すなわち大域経路は相互に重なり合うことができるため，複数ネットの並列配線を比較的容易に行うことができる．また決定された大域経路が重ならないような複数のネットについては，これも容易に詳細配線を並列に行うこと

ができる．

しかしこの大域/詳細配線を用いた並列化には，大域配線が詳細配線に依存するという並列化ボトルネックが存在する．大域配線では一般にネットの配線可能性を何らかの方法で推定するが，この推定の精度が実際の配線可能性に強く影響を及ぼす．推定精度はすでに決定された詳細経路の情報を利用すれば高くすることができるが，そのためには詳細配線から大域配線へのフィードバックが生じる．したがって，高い配線率を得るために推定精度を上げようとすると緊密なフィードバックが必要となり，結果的にネット間に強い逐次性が生じてしまう．

我々は，大域配線結果の信憑性の欠如が，この逐次性の要因であると考えた．すなわち大域経路内に必ず詳細経路が見つかることが保証されるならば，詳細配線から大域配線へのフィードバックは不要となり，並列化ボトルネックが解消する．本論文では，**経路選択法**と呼ぶ新たな手法に基づきこのフィードバックを除去した並列ルータ *Amaterous* を提案する[13]．Amaterous には大域/詳細配線のほかに，c-path 配線と呼ぶ独特の配線フェーズがある．大域配線に先立つこのフェーズでは，配線空間内の各矩形領域の配線容量を最大化するような候補経路である c-path が配線される．大域配線はこの c-path を選択することにより行うため，c-path の変形処理である詳細配線は必ず成功することが保証される．したがってフィードバックは除去され，大域/詳細の 2 つの配線フェーズを独立にかつ効率的に並列化することができる．

以下本論文では，Amaterous とその性能について次のような順序で述べる．まず 2 章では，関連研究を概観しながらフィードバック除去の重要性を示す．次に 3 章で Amaterous の概要を述べた後，4 章でその並列化実装について詳しく述べる．5 章では MCM ベンチマークを用いた性能評価結果を示し，6 章で結論と今後の課題を述べる．

2. 関 連 研 究

2.1 配線問題の並列性

配線問題には，**ネット内並列性**と**ネット間並列性**という 2 つの並列性が内在している．前者を用いた並列ルータでは，あるネットまたはその一部であるサブネットの配線を，複数のプロセッサが協同して行う．たとえば並列化された迷路探索法[9),14),15)]や線分探索法[4)]は，巨大な探索空間（配線空間）から最適解（経路）を求める並列探索の一種と考えることができる．また 2 端子（サブ）ネットをいくつかの部分に分割し，各々の配線をプロセッサごとに行う方法[7)]もネット内並列処理の一種である．

一方，複数の（サブ）ネットを同時に配線するようなルータは，ネット間並列性を利用したものである．たとえば複数の（サブ）ネットに重なり合いが生じないことがあらかじめ保証されているようなルータ[5),7),11),16)]では，並列配線はきわめて容易である．また（サブ）ネット間の重なり合いが生じる可能性があるルータ[4),7),10),12)]でも，その確率が小さければ並列化の効果を得ることができる．

なお文献 4)，10)，16) に示されたアルゴリズムのように，複数ネットの同時配線と個々のネットの並列配線を行って，2 つの並列性をどちらも利用するものも少なくない．また配線フェーズごとに利用する並列性を切り替えるアルゴリズムも，いくつか提案されている[5),7)]．

2.2 大域/詳細配線

大域配線は，巨大な配線空間から比較的小さな領域を抽出することにより，最終的な経路を求める**詳細配線**を短時間で行う方法として多くのルータで用いられている．大域配線が定める領域は大域経路と呼ばれ，詳細配線での経路探索はこの領域内でのみ行われる．したがって詳細経路の探索空間は配線空間全体に比べて格段に小さくなり，詳細配線の速度が大きく向上する．

大域配線には，重み付き迷路探索法[7)]，階層的分割法[1)]，反復改善法[10)]などが知られているが，これらは共通して詳細配線の容易さ，すなわち配線可能性を表すために何らかのコスト指標を用いている．代表的な指標は境界容量と呼ばれ，ある矩形領域の境界を通過できる配線の数を表している．また区画容量[7)]も類似した指標であり，矩形領域（区画）を左右または上下に通過できる配線の本数を表している．

ネット間並列性を利用する並列ルータにとっても，大域/詳細の 2 フェーズに分割することは大きなメリットがある．すなわち，1 つの境界あるいは区画を通過する複数の経路が同時に存在可能であるので，大域経路間の重なりを許容するような大域配線の並列化が可能である．また決定された複数の大域経路間に重なり合いがなければ，それらに含まれる詳細経路間の重なり合いもないことが当然保証されるので，詳細配線の並列化も容易である．したがって並列ルータの多くの成功例では，大域/詳細の 2 フェーズ配線が用いられている[7),9),16)]．

2.3 詳細配線から大域配線へのフィードバック

大域配線で用いるコスト指標である配線容量は，実

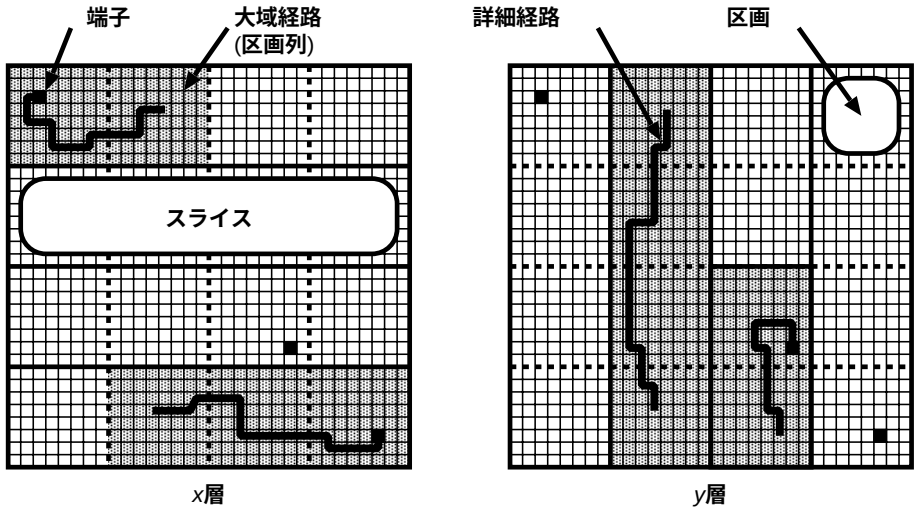


図 1 配線空間
Fig. 1 Routing space.

際の配線可能性の推定値であり，その精度は配線率に大きく影響する．すなわち，推定が楽観的すぎる場合には詳細配線での失敗が頻繁に生じ，逆に悲観的であると大域配線での失敗が頻発する．

この推定精度は文献 7) に報告されているように，詳細配線結果を配線容量に反映するか否かによって大きく左右される．たとえばこの文献では，詳細配線結果をまったく反映しない場合，MCM ベンチマーク[2)]の 1 つである MCC1-75 の未配線率が 2.5%にもなることが示されている．また，詳細配線から大域配線へのフィードバックを行って配線容量を随時更新すればMCC1-75 が完全に配線可能となることから，フィードバックの重要性が強調されている．

しかしこのフィードバックは，並列配線の効率を強く阻害するボトルネックとなる．たとえば，あるネットの大域配線の際にすべての既配線ネットの詳細経路を知らなければならないとすると，大域配線ではネット間並列処理をまったく行うことができなくなる．この逐次性は，前記の文献に示された並列ルータ Amon2 のように，一定数のネットをグループ化してフィードバック頻度を下げることで緩和されるが，推定精度が低下するという悪影響が生じる．またボトルネックの解消も必ずしも十分ではなく，Amon2 の評価結果によればフィードバックにより約 50%のオーバヘッドが生じている．

このフィードバックの存在は，配線容量による配線可能性の推定という方法自体に起因しており，その除去には配線アルゴリズムの本質的な見直しが必要となる．逆にいえば，序章でも述べたように，何らかの方法で大域経路内で確実に詳細配線ができることの保証が得られれば，フィードバックを除去することができる．またフィードバックがなくなれば，大域/詳細配線を完全に分離したフェーズで行えるようになり，両者の間での情報の受け渡し回数を最小化することもできる．

3. Amaterous の概要

3.1 用語の定義

本節では，Amaterous の説明に必要な用語をいくつか定義する（**図 1**）．**配線空間**は，非負整数座標値からなる有限の 3 次元座標空間 $[0, X] \times [0, Y] \times [0, Z]$ である．**ネット**は，配線空間の座標点（格子点）に位置する**端子**の集合であり，それらをすべて接続することによりネットは配線される．あるネットの**サブネット**は，ネットに含まれる 2 つの端子または 1 つの端子とサブネットの対である．したがって n 端子のネット $\{t_1, \ldots, t_n\}$ の配線を，サブネット $\langle t_1, t_2 \rangle$, $\langle \{t_1, t_2\}, t_3 \rangle$, $\ldots$, $\langle \{t_1, \ldots, t_{n-1}\}, t_n \rangle$ を接続する経路を見出すことと定義することができる．

サブネット $\langle S, t \rangle$ の**詳細経路**は，隣接する格子点の順序集合 $\{(x_1, y_1, z_1), \ldots, (x_m, y_m, z_m)\}$ であり，(x_1, y_1, z_1) は S がサブネットであればその詳細経路の要素であり， S が端子であればその座標値である．また (x_m, y_m, z_m) は端子 t の座標値である．さらに詳細経路の任意の要素について，以下のいずれかが成り立つ．

$$\begin{cases} x_{i+1} = x_i \pm 1 \ \wedge \ y_{i+1} = y_i \ \wedge \ z_{i+1} = z_i \\ x_{i+1} = x_i \ \wedge \ y_{i+1} = y_i \pm 1 \ \wedge \ z_{i+1} = z_i \\ x_{i+1} = x_i \ \wedge \ y_{i+1} = y_i \ \wedge \ z_{i+1} = z_i \pm 1 \end{cases}$$

あるネット N の詳細経路の和集合 $\pi_d(N)$ は，他の任意のネット N' に対する $\pi_d(N')$ と共通部分を持ってはならない．すなわち

$$\forall N \forall N' (N \neq N' \rightarrow \pi_d(N) \cap \pi_d(N') = \emptyset)$$

が成り立つ必要がある．

配線空間中の xy 空間を**配線層**といい，配線層を線分集合 $\{x = X^1, \ldots, x = X^k\}$ および $\{y = Y^1, \ldots, y = Y^l\}$ により分割して得られる矩形領域を**区画**という．z 座標が偶数（奇数）の配線層を x 層（y 層）といい，これらの層での経路は大局的に x 軸（y 軸）に平行となる．x 層（y 層）の区画を x 方向（y 方向）に連ねたものを**区画列**といい，その両端が座標空間の左右端（上下端）であるものを**スライス**という．また端子 t が含まれる区画を **t の（端子）区画**という．

サブネット $\langle S, t \rangle$ の大域経路は，区画境界線の交点を座標とする粗い座標空間 $\{0, X^1, \ldots, X^k\} \times \{0, Y^1, \ldots, Y^l\} \times [0, Z]$ を用いて，隣接する区画の順序集合として詳細経路と同様に定義される．ただし経路内で隣接する区画 (X_i, Y_i, z_i) と $(X_{i+1}, Y_{i+1}, z_{i+1})$ が同じ x 層（y 層）にある場合，$Y_i = Y_{i+1}$ $(X_i = X_{i+1})$ でなければならない．すなわち x 層（y 層）に含まれる大域経路の断片は x 軸（y 軸）に平行な区画列であり，これを**大域区画列**という．

あるネット N の大域経路の和集合 $\pi_g(N)$ は，他のネット N' に対する $\pi_g(N')$ と共通部分を持つことができる．また詳細経路の和集合 $\pi_d(N)$ は，$\pi_g(N)$ に含まれなければならない．すなわち；

$$\begin{aligned} &\forall p_d = (x, y, z)(p_d \in \pi_d(N) \rightarrow \\ &\quad \exists p_g = (X^i, Y^j, z)(p_g \in \pi_g(N) \ \wedge \\ &\qquad X^i \leq x < X^{i+1} \wedge Y^j \leq y < Y^{j+1})) \end{aligned}$$

が成り立つ必要がある．x 層（y 層）の区画の**区画容量**は，その区画の左右端（上下端）を同時に接続するような詳細経路の最大数である．

3.2 容量最大化経路 c-path

Amaterous は独立した 3 つの配線フェーズである，c-path 配線，大域配線，詳細配線により，与えられたネット集合の配線を行う．

最初のフェーズである c-path 配線は Amaterous 特有のものである．x 層中の各スライスについて，その**容量最大化経路（c-path）**は理想的に以下の性質を満たすものとして定義される（**図 2**）．

(1) c-path は端子を含む初期障害物や他の c-path と交わらない．

(2) c-path の左端はスライス中の区画の左辺上にあり，右端は同じあるいは別の区画の右辺上にある．

(3) 各区画を通過する c-path の本数は，その区画の区画容量に等しい．すなわち区画を通過する c-path の数は最大化される．

(4) 各区画列を通過する c-path の本数は，その区画列に含まれる区画および区画列を通過する c-path 本数を最大化するという条件下で最大化される．

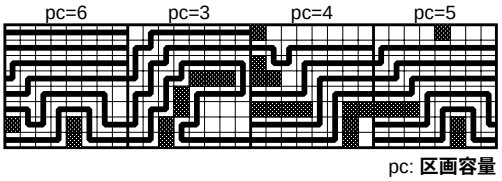


図 2 容量最大化経路（c-path）
Fig. 2 Capacity path (c-path).

また y 層の c-path についても同様に定義される．

3.3 大域/詳細配線

すべてのスライスに対して c-path が配線された後（**図 3** (a)），各端子はその端子区画を含む 2 つ以上の区画を通過する c-path に接続される（図 3 (b)）．この c-path を端子の**初期 c-path** といい，端子区画に隣接する区画のうち初期 c-path が含まれるものを端子の**初期区画**という．

2 端子（サブ）ネットの大域配線では，この初期区画を結ぶような最小コスト経路を求める．大域経路のコストは，経路に含まれる区画のコストの総和であり，各区画での配線混雑を避けるために区画容量が小さいほど区画コストが大きくなるように定められる．ただし大域経路中の区画列に，それを通過する c-path が存在しないものがあれば，大域経路のコストは無限大となる．したがってコストが有限の大域経路の各区画列について，それを通過するような詳細経路が存在すること，すなわち最悪でも c-path を使えば詳細配線が可能であることが保証される．このように Amaterous の大域配線は，c-path が存在するような経路を選択する**経路選択法**によるものであり，大域経路に対応する詳細経路の存在が必ず保証される．

大域経路が定まると，経路上の区画の容量が 1 ずつ減らされ，各区画列について特定の c-path がその経路に占有的に割り当てられる（図 3 (c)）．なお端子とサブネットを結ぶ大域配線は，端子の初期区画とサブネットの大域経路を同様の手順で接続する処理である．

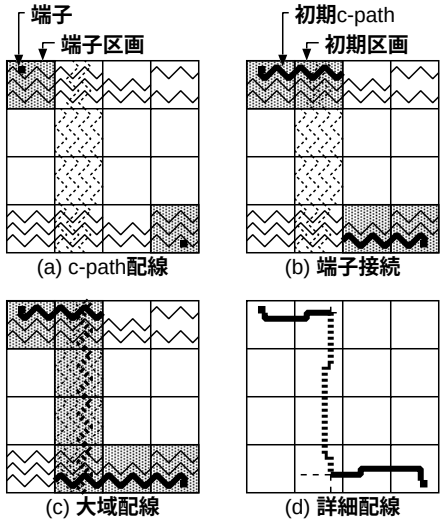


図 3 配線フェーズ
Fig. 3 Routing phases.

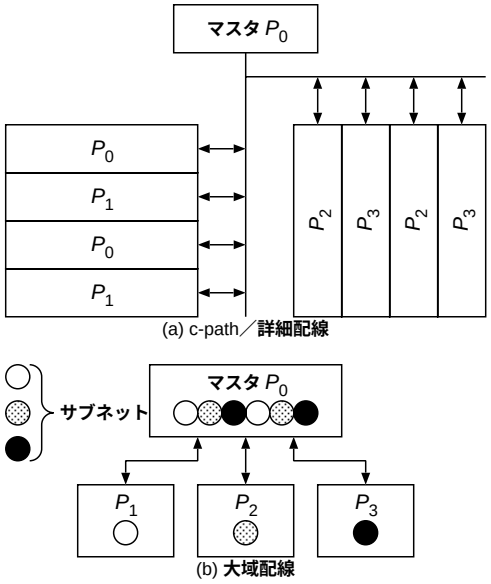


図 4 各配線フェーズの並列化
Fig. 4 Parallelization of routing phases.

すべてのネットに対する大域配線が完了した時点で，各大域区画列にはそれを通過する固有の c-path が割り当てられている．したがってこれらをビアで結合すれば，最低限の配線ができたことになる．しかし図 2 に示したように，c-path には多数の屈曲点があり配線長も非常に長い．そこで詳細配線では各 c-path について，大域区画の両端の結合性を維持しつつ，屈曲点をできるだけ少なくする直線化を行う．最後に直線化された経路をビアで結合し，不要な断片を消去して配線を完了する（図 3 (d)）．

4. Amaterous の並列実装

4.1 並列化の概要

Amaterous の 3 つの配線フェーズである c-path 配線，大域配線，詳細配線は，いずれもマスタ/ワーカのモデルによって並列化される．c-path 配線と詳細配線では**図 4** (a) に示すように，ワーカ・プロセッサの半数に x 層の各スライスがサイクリックに割り当てられ，残りの半数には y 層のスライスが割り当てられる．また 1 つのプロセッサ，たとえば P_0 はマスタとしても機能する．

大域配線では図 4 (b) に示すように，P_0 以外のプロセッサがワーカとして働き，各々に割り当てられたサブネットの大域経路を求める．一方 P_0 はマスタとしてのみ動作し，ワーカが求めた大域経路の整合性チェックを行う．

4.2 c-path 配線

c-path 配線ではまず，マスタが端子と初期障害物の位置情報を各ワーカに伝達する．続いて x 層のスライスを割り当てられたワーカは，各スライスごとに以下のアルゴリズムに従って c-path を配線する．なおこのアルゴリズムは，文献 6) に示された区画容量算出法をベースとしている．

for p = 左端の区画 **to** 右端の区画 **begin**
  $e = p$ の左辺；
  for $g = e$ の下端 **to** e の上端 **begin**
    以下のいずれかに到達するまで，右手をスライスの下端，障害物，端子，あるいは既配線の c-path に触れながら迷路探索を行い（右手法），その軌跡を c-path とする．
      1. スライスの右端
      2. スライスの上端
      3. 一度通過した区画境界
    2 または 3 の場合には，最後に通過した区画境界以後の軌跡は消去する．
  end
end

また y 層の c-path も，同様のアルゴリズムで配線する．

図 2 に示した c-path は，このアルゴリズムによって配線したものであり，3.2 節で述べた 4 条件をすべて満たしている．しかし一般には条件 (4) が満たされるとは限らないため，上記のアルゴリズムは準最適である．なおすべての条件を満たすようなアルゴリズムは計算コストがきわめて大きくなり，また上記の準最

適アルゴリズムでも十分に実用的な解が得られるが，区画列についてさらに多くの c-path が通過するようなアルゴリズムの検討も今後行っていく予定である．

4.3 端子と c-path の接続

c-path がすべて配線されると，次に各端子を c-path に接続して端子の初期 c-path を定める．この接続処理ではまず，どの配線層の c-path に端子を接続するか，すなわち端子の配線層への分配を以下のように区画ごとに決定する．区画に含まれる端子の x および y 座標の最小/最大値をそれぞれ $x_{\min}$，$x_{\max}$，$y_{\min}$，$y_{\max}$ としたとき，$x_{\max} - x_{\min} \leq y_{\max} - y_{\min}$ であれば x 層に，そうでなければ y 層に，それぞれ均等に分配する．

続いて以下のアルゴリズムにより x 層に割り当てられた各端子を c-path に接続する．

(1) 最上端の c-path を c とする．

(2) c と接続可能な端子のうち，c よりも上部にあるものの集合を T_u，下部にあるものの集合 T_l とする．

(3) $T_u = T_l = \emptyset$ であるとき，c の直下に c-path が存在すればそれを c として (2) に戻る．そうでなければ終了する．

(4) c が区画の左端を通過していなければ (7) に進む．通過していれば，T_u の各端子を起点とし，c の任意の点を終点とする最短経路探索を行い，c とこれらの経路の交点（最短接続点）が c の上で最も左に位置する端子を t とする．なおこのような端子が複数ある場合，最短接続点から各最短経路を左優先探索して到達する端子を t とする．また $T_u = \emptyset$ であれば同様の探索を T_l に対して行い，タイブレークの必要があれば交点から右優先探索を行って t を求める．
直感的に述べるとこの選択処理では，c の左端近くに接続しやすい端子が求められる．また接続処理は区画の上部から行うため，接続可能な c-path が上方に存在しない T_u を優先することにより，接続処理が失敗する確率を小さくしている．

(5) t の x 座標を x_t としたとき，$x = x_t + 1$ なる直線を仮想的障壁としたうえで，t を起点とし c 上の任意の点を終点とする右手法迷路探索を行い，その軌跡を t から c への接続経路とする．ただしこの探索で t と c が接続できない場合には，仮想的障壁を取り除いて再度右手法迷路探索を行う．
直感的に述べるとこの処理は，区画のできるだけ上部を迂回して c の左端近くに接続する経路を求めるものであり，仮想的障壁は区画の右側への回り込みを防止する．c-path は区画の下部から順に配線されるため，上部には c-path が存在しない空間が残っていることが多く，それを有効活用することによって他の端子の c-path 接続をできるだけ妨げないようにしている．

(6) t を T_u または T_l から除去し，その結果 $T_u = T_l = \emptyset$ となれば (9) に進む．

(7) c が区画の右端を通過していなければ (9) に進む．通過していれば (4) と同様の処理を左右反転した形で行って t を求める．

(8) (5) と同様の処理を左右反転した形で行って，t を c に接続する．

(9) c の不要部分を消去する．このことにより c の直上の c-path に接続可能な端子が生じることがある．そこで c の上部に c-path が存在すれば c の直上の c-path を，そうでなければ c の直下の c-path を，それぞれ新たな c として (2) に戻る．

y 層に割り当てられた端子についても，同様のアルゴリズムにより c-path との接続処理が行われる．

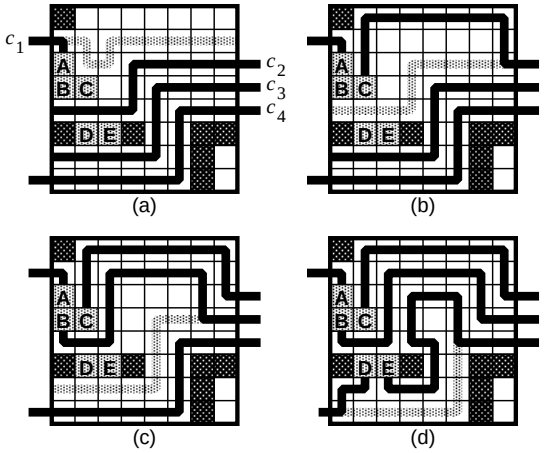


図 5 端子と c-path の接続
Fig. 5 Connecting terminals to c-paths.

図 5 は図 2 の右から 2 番目の区画に A ～ E の 5 端子が配置されている場合に，上記のアルゴリズムによって各端子と c-path が接続される様子を示したものである．この区画中には 4 つの c-path ($c_1 \sim c_4$) が配線されており，c_1 は区画の左端のみを，c_2 と c_3 は右端のみを，c_4 は両端を，それぞれ通過している．以下，図の (a) ～ (d) について説明する．

(a) c_1 について $T_u = \emptyset$，$T_l = \{\mathrm{A}, \mathrm{C}\}$ であり，c_1 と

の最短接続点が最も左側にある A が (4) により選択され，(5) によって c_1 と接続される．

(b) c_2 について $T_u = \{\mathrm{B}, \mathrm{C}\}$，$T_l = \{\mathrm{D}, \mathrm{E}\}$ であり，c_2 との最短接続点が最も右側にある $\mathrm{C} \in T_u$ が (7) により選択され，(8) によって c_2 と接続される．ここで T_u ではなく T_l を優先してたとえば E を選択すると，B や C は c-path に接続できなくなることに注意されたい．また C と c_2 の接続に左手法による迂回経路を用いず，たとえば最短経路を用いると，B は c-path に接続できなくなる．

(c) c_3 について $T_u = \{\mathrm{B}, \mathrm{D}, \mathrm{E}\}$，$T_l = \emptyset$ であり，c_3 との最短接続点は B，D，E に共通である．そこで最短接続点から接続経路を右優先探索して得られる B が (7) により選択され，(8) によって c_3 と接続される．

(d) c_4 について $T_u = \{\mathrm{D}, \mathrm{E}\}$，$T_l = \emptyset$ であり，c_4 との最短接続点が最も左側にある D が (4) により選択され，(5) によって c_4 と接続される．また引き続き E が (7) により選択され，(8) によって c_4 と接続される．

なお接続できない端子が残った場合には，x 層と y 層を担当するプロセッサの間で端子が交換され，再び接続が試みられる．このための通信は，c-path 配線中に生じる唯一のワーカ間通信である．

端子と c-path の接続が完了すると，すべての c-path には固有の識別子を与えられ，それぞれの両端区画の識別子，および端子が接続されていればその識別子が付加される．マスタはこの c-path 情報を収集し，**c-path マップ**と呼ぶデータ構造を生成する．なお c-path の詳細な経路情報はマスタには送られず，詳細配線で使用するために各ワーカ内に保存される．

4.4 大域配線

大域配線ではまず，マスタが各ネットをサブネットに分割する．ネット N の端子数が n であるとき，N は下式に基づきサブネット $S_1, \ldots, S_{n-1}$ に分割される．

$$
\begin{aligned}
&d(t, t') \equiv t \text{ と } t' \text{ のマンハッタン距離}\\
&d(t, S) \equiv \min_{t' \in S} d(t, t')\\
&\quad S_1 = \langle t_1, t_2 \rangle \text{ s.t. } d(t_1, t_2) = \min_{t, t' \in N} d(t, t')\\
&\quad S_i = \langle S_{i-1}, t_{i+1} \rangle \text{ s.t. } t_{i+1} \notin S_{i-1} \ \wedge\\
&\qquad\qquad d(t_{i+1}, S_{i-1}) = \min_{t \in N - S_{i-1}} d(t, S_{i-1})
\end{aligned}
$$

生成されたサブネットは，未配線サブネットを d が小さい順に管理するサブネット・プールに格納される．なおサブネット $S = \langle S', t \rangle$ について，$d(S', t)$ が全体で何番目に小さいかを示す値を S の順位 $rank(S)$ と呼ぶ．

続いてサブネット・プールから，d が小さい順に $2 \times k \times l$ 個の 2 端子サブネットを抽出する．ここで k はワーカの総数であり，l は各ワーカが一度に処理するサブネットのグループの大きさである．すなわち l 個の 2 端子サブネットからなるグループがワーカ数の 2 倍だけ生成され，ワーカ w_i には d が小さいグループ $g_i(1)$ と大きいグループ $g_i(2)$ の 2 個が割り当てられる．また前述の c-path マップ M を複製し，すべてのワーカに与える．

ワーカ w_i は，まず $g_i(1)$ に含まれるサブネットを d が小さい順に処理し，それぞれの大域経路を定める．また経路中の大域区画列の各々について c-path を仮割当てし，w_i がローカルに管理する c-path マップ m_i を更新する．ここで m_i の初期値を $m_i(0) = M$，$g_i(1)$ の処理後の値を $m_i(1)$ としたとき，w_i は両者の差分 $\Delta m_i(1)$（概念的に $m_i(1) - m_i(0)$ と表記する）を記憶しておく．また大域区画列への c-path 仮割当ては，区画列を通過する複数の c-path の中から通過区画数が最小のものの 1 つを選ぶ最良選択により行う．

$g_i(1)$ の大域配線が完了するとその経路情報はマスタに送られ，他のワーカからすでに報告された経路情報との整合性がチェックされる．すなわち各経路に含まれるそれぞれの大域区画について，両端を結ぶ c-path が本当に残っているかどうかをチェックする．もしすでに受理した大域経路への割当てにより c-path が完全に消費されていれば，その経路は却下され，対応するサブネットはサブネット・プールへ戻される．一方 c-path が残っていれば，最良選択により大域区画への c-path 割当てを最終決定し，その際に生じる断片に識別子を付与して利用可能にしたうえで，マスタの c-path マップ M を更新する．ここで $g_i(1)$ の処理後の M の値を $M_i(1)$ とすると，マスタは $\Delta M_i(1) = M_i(1) - M_i(0)$（ただし $\forall i(M_i(0) = M)$）を求める．またサブネット・プールから，2 端子または端子と既配線サブネットからなる l 個のサブネットを d の小さい順に抽出し，これを $g_i(3)$ として $\Delta M_i(1)$ とともにワーカ w_i に与える．

上記の処理をマスタが行っている間，ワーカ w_i は第 2 グループ $g_i(2)$ の配線処理を行い，c-path マップを $m_i(2) = m_i(1) + \Delta m_i(2)$ に更新したうえで経路情報をマスタに送信する．その後マスタから $g_i(3)$ と $\Delta M_i(1)$ を受信すると，まず $m_i(1) \leftarrow m_i(0) + \Delta M_i(1) = M_i(1)$ として $g_i(2)$ の配線前の

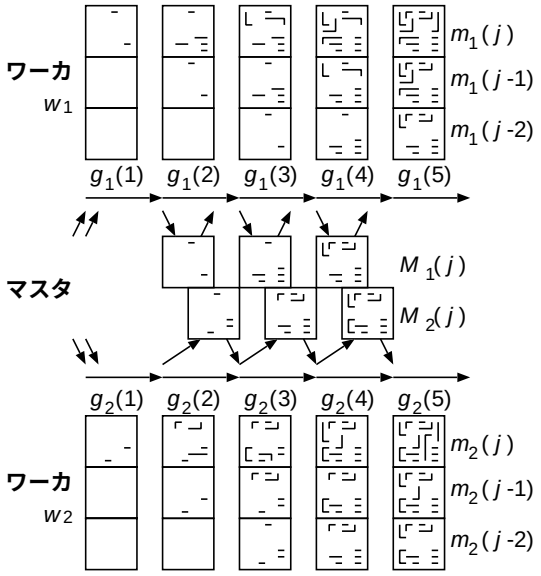


図 6 大域配線でのマスタ/ワーカ間通信
Fig. 6 Master/worker communication in global routing.

時点での大域的に正しい c-path マップを構築し，さらに $m_i(2) \leftarrow m_i(1) + \Delta m_i(2)$ として $m_i(2)$ をほぼ最新のものとする．一般にワーカ w_i がサブネットのグループ $g(j)$ の配線処理を行っているとき，w_i は大域的に正しい $m_i(j-2)$，ほぼ最新の $m_i(j-1)$，および作業用の $m_i(j)$ の 3 バージョンの c-path マップを管理する．

このように 3 バージョンのマップを管理することにより，**図 6** に示すようにマスタ/ワーカ間での通信遅延を隠蔽することができる．この遅延隠蔽の効果は，サブネット・グループの大きさ l が大きいほど増加する．しかし l を大きくするとワーカが管理するマップの鮮度が低下し，ワーカ間での配線の整合性も低下してしまう．したがってこの両者のトレードオフで l の値を定める必要があり，5.5 節で述べるように実験結果に基づき 10 としている．

各ワーカにおいてサブネット S の大域経路を求める処理は，経路を構成する区画に与えられたコストの和が最小のものを求める，最小コスト経路探索により行われる．ここでサブネットの総数を K，S のある大域経路候補中の大域区画列を $\sigma = \{p_1, \ldots, p_n\}$ とすると，σ のコスト $cost(\sigma)$ は区画 p_i の区画容量 C_i，S の順位 $rank(S)$，後述する引き剥し再配線のために S の大域経路が何回引き剥されたかを示す $ripped(S)$，および σ を通過する未割当て c-path の本数 $paths(\sigma)$ により，以下のように定めている．

$$cost(p_k) = C_k^{-1.6+\lambda+\rho}$$

$$\lambda = \begin{cases} 0.0 & \text{if } rank(S) \le K/3 \\ 0.1 & \text{if } K/3 < rank(S) \le 2K/3 \\ 0.2 & \text{if } 2K/3 < rank(S) \end{cases}$$

$$\rho = \begin{cases} 0.0 & \text{if } ripped(S) = 0 \\ 0.1 & \text{if } ripped(S) = 1 \\ 0.2 & \text{if } ripped(S) \ge 2 \end{cases}$$

$$cost(\sigma) = \begin{cases} \sum_{k=1}^{n} cost(p_k) & \text{if } paths(\sigma) > 0 \\ \infty & \text{if } paths(\sigma) = 0 \end{cases}$$

上記のパラメータの具体値は実験により定めたものであり，λ は d が小さいため配線順序が早いサブネットほど区画容量に敏感にするためのものである．すなわち λ が小さければ区画容量が小さい区画を避ける傾向が強まり，後から配線されるサブネットのための余裕が確保されやすくなる．

一方 ρ は，以下に述べる引き剥し再配線に関係する．ワーカが大域経路を発見できなかったサブネットはマスタによって処理され，すでに大域配線が完了している経路を引き剥しながら配線される．この引き剥しの対象となったサブネットは，サブネット・プールに戻されて再配線されるが，一般に配線が困難になる傾向がある．そこで引き剥し対象となったサブネットについては ρ を大きくし，区画容量に対する感度を落として，容量確保よりも配線長の短縮を優先するようにしている．なおマスタによって配線された経路は，アルゴリズムの停止性を確保するために，引き剥し対象とはしない．

4.5 詳細配線

大域配線が完了すると，マスタは c-path マップ M をスライスごとに分割し，各スライスを担当するワーカへ分配する．ワーカはまず，c-path 配線の際に作成しワーカ内に保存してある c-path の詳細な経路情報を，マスタから受けとった c-path マップと統合して，各大域経路に割り当てられた c-path の完全な情報を得る．

続いて x 層を担当するワーカは，以下のアルゴリズムに従って c-path の直線化を行う（**図 7**）．

(a) 大域経路に割り当てられなかった c-path をすべて消去する．

(b) 以下により定義される c-path γ と γ' の関係 $\gamma \prec \gamma'$ により，すべての c-path に全順序を与える．

- γ と γ' が同じ区画を通っている場合，γ が γ' よりも上側（y 座標が大きい側）にあれ

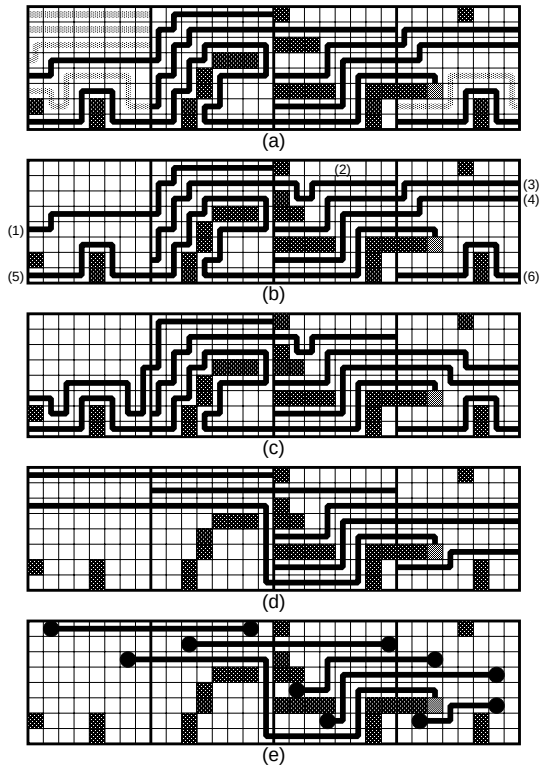


図 7 詳細配線
Fig. 7 Detailed routing.

ば，またそのときに限り $\gamma \prec \gamma'$ である．

- γ と γ' が同じ区画を通っていない場合，γ が γ' よりも左側（x 座標が小さい側）にあれば，またそのときに限り $\gamma \prec \gamma'$ である．

この関係により，任意の $i < j$ について $\gamma_i \prec \gamma_j$ なる c-path の順序集合 $\{\gamma_1, \ldots, \gamma_n\}$ が得られる．

(c) 4.2 節に示したアルゴリズムに従って，$\gamma_n, \ldots, \gamma_1$ の順番で c-path を再配線する．

(d) $\gamma_1, \ldots, \gamma_n$ の順番で，以下を繰返す．

- γ_i をいったん消去する．
- γ_i の両端が区画境界であれば左側の境界から右側の境界へ，そうでなければ端子から区画境界へ，迷路法によって最短経路配線を行う．その際，他の c-path や直線化された経路は障害物と見なし，また複数の経路が得られたならば最も上側の経路を優先する．

なお y 層の c-path も同様に直線化される．またこのアルゴリズムは文献 6) で提案された max-cap アルゴリズムを改良したものである．

次に x 層を担当するワーカは，直線化された経路の詳細な経路情報を y 層を担当するワーカに伝える．すなわち，ある x 層の区画中の詳細経路情報は，その区画と交差する y 層のスライスを担当するワーカに伝えられる．y 層のワーカは各々の隣接する大域区画列について，x および y 層の詳細経路情報から交差位置を求めてビアを配置し，不要となった断片を消去する（図 7 (e)）．

表 1 MCM ベンチマーク
Table 1 MCM benchmarks.

ベンチマーク	c	n	t	s	l	w
MCC1-75	6	802	2495	599×599	4	30
MCC2-75	37	7118	14695	2032×2032	6	80
MCC2-45	37	7118	14695	3386×3386	4	120

c：チップ数，n：ネット数，t：端子数，
s：配線空間サイズ，l：層数，w：スライス幅

5. 評　　価

5.1 Amaterous の実装環境

我々は Amaterous を，450 MHz の Pentium II を CPU とし 128 MB のメモリを持つ PC を，100Base-TX のイーサネットにより 16 台結合した PC クラスタに実装した．プログラムは C++で記述し，並列化には PVM 3.4.β7[3]を，コンパイラは gcc 2.7.2.3 をそれぞれ用いた．

5.2 ワークロード

評価に用いたワークロードは，**表 1** に示す 3 種の MCM ベンチマーク[2]である．表に示した配線層数は Amon2[7]の評価，および高性能逐次ルータ V4R[8]の評価に用いられたものと同じである．また Amon2 との性能比較を公平に行うため，スライスの幅も Amon2 の評価パラメータに合わせてある．

5.3 Amaterous と Amon2 の比較

表 2 は，Amaterous の逐次性能 (S) と 16 プロセッサでの並列性能 (P)，および Amon2 の性能を示したものである．また配線品質の目安として，V4R[8]のデータも示している☆．

なお Amon2 については，文献 7) に示されたデータ (a) が一世代前の並列計算機 AP-1000 を用いて得られたものであるため，公正な比較のために我々の PC クラスタに移植して再評価して性能データ (b) を求めた．しかし表に示すように，この移植版ではどのベンチマークに対しても配線不能ネットが生じている．この理由はおそらく，Amon2 が計算性能に対する通

☆ カッコ書きした実行時間は，Amon2 が 64 プロセッサの AP-1000，V4R が SPARCStationII での測定値であり，Amaterous との意味ある比較は困難である．

表 2 Amaterous と Amon2 の性能比較
Table 2 Amaterous vs. Amon2.

benchmark	program		t	u	l	v
MCC1-75	Amaterous	(S)	12.61	0	477	4990
		(P)	2.41	0	500	4824
	Amon2	(a)	(13)	0	382	2825
		(b)	8.25	1	366	2519
		(c)	8.61	—	—	—
	V4R		(180)	0	394	6993
MCC2-75	Amaterous	(S)	459.99	0	7232	39141
		(P)	61.65	0	7439	40089
	Amon2	(a)	(173)	9	5498	22823
		(b)	50.31	17	4486	14885
		(c)	61.66	—	—	—
	V4R		(3960)	0	5559	36438
MCC2-45	Amaterous	(S)	638.61	0	10105	31688
		(P)	52.19	0	10385	33482
	Amon2	(a)	(316)	0	9052	19554
		(b)	83.67	10	7310	13275
		(c)	103.61	—	—	—
	V4R		(5820)	0	9131	36473

t：実行時間（秒），u：配線不能ネット数，
l：配線長総計（$\times 10^3$ 格子点），v：ビア数

信性能にきわめて敏感であり，AP-1000 に比べて PC クラスタでは相対通信性能が一桁以上劣ることによるものと思われる．また Amon2 には配線不能を救済する引き剥し再配線の機能がなく，大域配線に失敗したネットは単に捨てられる．

いずれにせよ配線不能ネットに対しては，配線処理の大部分が行われていないため，(b) の実行時間は過小評価されたものであることは明らかである．そこでこれを補正するために，実行時間が配線長に比例すると仮定して，(a) の配線長を用いて実行時間を推計した値 (c) を求めた．なお引き剥し再配線は並列化が困難であることから，この補正による実行時間の推計が過大になることはないと考えられる．

この推計値 (c) を用いて Amaterous と Amon2 の性能を比較すると，MCC1-75 では 3.6 倍，MCC2-45 では 2.0 倍，それぞれ Amaterous が Amon2 よりも高速であることが明らかになる．一方 MCC2-75 では両者の性能はほぼ同じであるが，Amon2 ではオリジナル版 (a) においても 9 本の配線不能ネットが残ることに注意が必要である．すなわち，これらのネットを配線するためにさらに 100 万格子点程度の経路が付加されるとすれば，Amon2 の実行時間は約 1.2 倍となる．したがって，Amaterous は Amon2 よりも一般に高速であり，多くの場合には大きく凌駕する性能を持つと結論できる．

一方，両者の配線長を比較すると，Amaterous では 10 ～ 35% の増加が見られ，V4R と比較してもほぼ同程度増加している．この主な原因は，Amaterous では 2 端子（サブ）ネットの大域配線を，各々の端子区画に隣接する初期区画間を結合することによって行うことにある．すなわち初期区画の選択はランダムに近いため，一方の端子の初期区画は 1/2 の確率で他方の端子区画から遠ざかることになる．遠ざかった場合には，大域配線の開始時点ですでにスライス幅の倍程度の損失（配線長増大）が生じていることになるので，この損失を端子数とスライス幅の積でで見積もることができる☆．この値を MCC1-75，MCC2-75，MCC2-45 についてそれぞれ求めると，75×10^3，1176×10^3，1763×10^3 となり，Amaterous の逐次性能と V4R の性能の差分の 90%，70%，181%となる．逆にいえばこの損失を回避するように初期区画を選択すればAmon や V4R と同等の配線品質が得られることになり，現状のアルゴリズムの欠点を明確化すると同時に，Amaterous が本質的に他のシステムに劣るものではないことが明らかになる．

また，この初期区画選択の問題が顕著に現れるのは，2 つの端子が同じ区画にある場合であり，最善の場合でも隣接する初期区画への配線，すなわちスライス幅の 2 倍程度の損失が生じてしまう．両端子の初期区画が異なると損失はさらに大きくなり，平均的にはスライス幅の 3.5 倍程度の損失が生じる．現在，このような近接した端子対の取扱いや，前述の初期区画の選択方法の見直しを行っており，配線長を大幅に短縮できるものと考えている．

またビア数についても Amon2 よりもかなり増加しているが，V4R と比較すると，MCC1-75 が 40%減，MCC2-75 が 10%増，MCC2-45 が 10%減となり，同等かそれ以上の配線品質が得られている．したがって上記の近接端子対の改善のほかに，Amon2 のようにビアに大きなコストを与えることが有効ではないかと考えている．

5.4 台数効果

図 8 は，Amaterous の逐次版の性能を 1 としたときの並列性能を示したものである．図に示されているとおり 16 プロセッサでは，MCC1-75 が 5.0 倍，MCC2-75 が 7.5 倍，MCC2-45 が 12.2 倍の台数効果が得られている．MCC2-45 の結果より，ネット数や配線空間が大きな問題に対しては，Amaterous が良好なスケーラビリティを持つことが明らかになった．

MCC1-75 の結果に見られる速度向上の飽和は，こ

☆ 端子対が同一スライス上にある場合には損失の期待値がより大きくなるので，過大評価ではない．

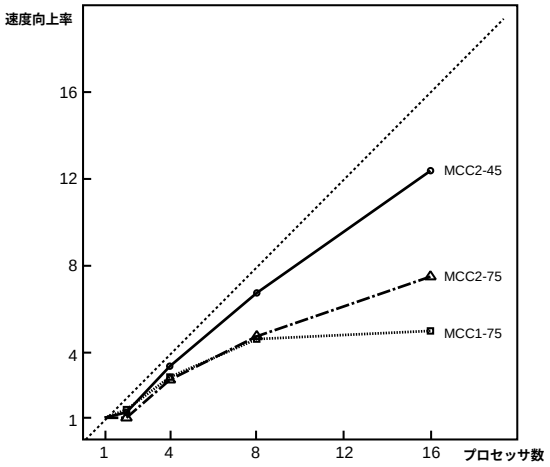


図 8 台数効果
Fig. 8 Parallel speedup.

の問題のネット数や配線空間が小さく，十分な並列性が得られないことに起因している．実際，全体の処理時間の約 60%を占める大域配線の性能は，16 プロセッサにおいても 3.5 倍しか向上していない．改善策としては，この問題のように比較的配線が容易なものに対しては，10 に固定しているサブネット・グループの大きさを増やし，マスタ/ワーカ間の通信遅延隠蔽をより効果的に行うことが考えられる．

また MCC2-75 の速度向上は MCC2-45 よりもかなり劣っており，特に逐次版と 16 プロセッサの並列版では両者の性能が逆転していることが注目される（表 2 参照）．この理由は，MCC2-75 は配線空間の大きさが MCC2-45 の 60%程度であるため，大域配線での経路選択の自由度が小さく，大域配線でのマスタによる引き剥し再配線が頻発することにある．すなわちマスタが引き剥し再配線を始めると，ワーカはサブネット・グループの供給が停止するためアイドルとなり，並列性能が低下してしまう．改善策としては引き剥し再配線をマスタからワーカの 1 つへ委譲し，他のワーカでは投機的に大域配線を続行することが考えられる．

5.5 パラメータ設定

本節では，Amaterous の実行速度や配線品質を左右するパラメータである，スライス幅 w と，サブネット・グループの大きさ l について議論する．

スライス幅

5.2 節で述べたように，今回の評価では w の値を Amon2 の評価パラメータと同一としたが，この値は必ずしも最適であるとは限らない．実際 Amon2 では，64 プロセッサ構成の場合に 1 プロセッサが x あるいは y 層のスライスを 1 つ以上担当できることを最優先して w を定めている☆．したがって今回の評価のように 16 プロセッサ構成であれば，負荷の均衡化の観点からは w を大きくすることも可能であり，以下のメリットが生じると予想される．

- c-path 配線や端子と c-path の接続処理において，スライス境界は通過できない障害物として作用するので， w を大きくすれば障害物が除去されることになる．一方区画の両端の距離が大きくなるため区画を貫通することは困難になるが，障害物除去効果と相殺して c-path の配線本数が同程度となるとすれば，結果的に長い c-path が得られることになり，大域配線での経路選択の自由度が増加する．
- 大域区画の数が少なくなるため，大域配線での探索空間が小さくなる．上記のように c-path 本数が減少せずに経路選択の自由度が増加すれば，探索空間の縮小との相乗効果で大域配線に要する時間が短縮される．5.4 節で述べたように台数効果の飽和要因は大域配線にあり，大域配線時間の短縮は全体的な性能向上に大きく寄与するものと予想される．

一方デメリットとしては，スライス数が減少することによって大域配線以外のフェーズでの負荷の不均衡が生じることがあげられる．しかし，これらのフェーズの 16 プロセッサでの台数効果は 10 倍以上であることが確認されており，またスライスあたりの処理時間はほぼスライス面積に比例するので，極端な負荷の不均衡がなければ大域配線時間の短縮効果が上回るものと考えられる．したがって，たとえば c-path 配線本数やプロセッサ数とスライス数の比により一定の制約を課したうえで，w を大きくすることは有望であり，今後実験による確認や w の適応的設定の方式検討を行う予定である．

サブネット・グループ

サブネット・グループの大きさ l は 4.4 節で述べたように，大域配線のマスタ/ワーカ間の通信遅延隠蔽効果と，ワーカ間での配線整合性のトレードオフで定める必要がある． l を 5，10，20 と変化させて，MCC2-75 を 16 プロセッサで配線する予備実験を行った結果，$l = 5$ と $l = 20$ はどちらも $l = 10$ に対して 10%程度の性能低下が観測された．

$l = 5$ の性能低下要因としては通信遅延の顕在化だけでなく，マスタでのメッセージ処理の定数時間オー

☆ Amon2 では詳細配線を担当するプロセッサにスライスが割り当てられ，大域配線を担当するプロセッサには割り当てられないため，スライス数は 64 よりも小さくてよい．

バヘッドの顕在化，すなわちマスタがボトルネックとなってしまう現象が観測された．一方 $l = 20$ の主な性能低下要因はワーカ間の配線不整合であり，マスタが却下した配線が 40%程度増加した．また配線順序が乱れるために引き剥し再配線の回数も 4 から 8 に倍増しており，5.4 節で述べた再配線中にワーカがアイドルになる問題も要因の 1 つである．

このような実験結果に基づいて今回の評価では l を 10 と定めたが，通信遅延隠蔽の要素にはシステムの計算/通信性能比が強く影響するため，簡単な予備評価などによってシステムに適した値を求める方法を検討している．また 5.4 節で述べたように，MCC1-75 のような問題に対しては l を大きくすることも考えられる．これについては，ネットや端子の数で並列性を，また c-path 本数やスライス数で大域配線の自由度を見積もり，l の値に反映させる方法を検討している．

6. おわりに

本論文では，経路選択法に基づく新たな並列ルータ Amaterous を提案した．Amaterous の特徴は，大域/詳細配線に先だって c-path と呼ぶ容量最大化経路をあらかじめ配線しておくこととにある．また，この c-path を選択する経路選択法により大域経路を定め，選択された c-path の直線化により詳細経路を求めるため，大域経路に対応する詳細経路の存在がつねに保証される．

この c-path 配線と経路選択法によって，詳細配線から大域配線へのフィードバックが不要となり，各配線フェーズを独立かつ効率的に並列化することができる．実際 MCM ベンチマークを用いた評価の結果，16 ノードの PC クラスタで最大 12 倍の台数効果が得られた．また高性能並列ルータの 1 つである Amon2 に比べ，Amaterous は 1.2 ～ 3.6 倍高速であることも明らかになった．

今後の課題としては，大域区画列を貫通する c-path の本数を増やす効率的アルゴリズムの考案，初期区画選択や近接端子対の配線改良よる配線長の短縮，引き剥し再配線のマスタからワーカへの委譲，スライス幅やサブネット・グループの大きさなどの配線パラメータのチューニングがあげられる．

謝辞 Amon2 のソースコードを提供していただいた，Hesham Keshk 氏と京都大学大学院情報学研究科の富田研究室に感謝の意を表する．

参考文献

1) Brouwer, R.J. and Banerjee, P.: PHIGURE: A Parallel Hierarchical Global Router, *Proc. 27th Design Automation Conf.*, pp.360–364 (1990).
2) Collaborative Benchmarking Laboratory, Dept. of Computer Science, North Carolina State Univ.: PDWorkshop'93 Benchmark Set. `http://www.cbl.ncsu.edu/CBL_Docs/pdw93.html` (1995).
3) Computer Science and Mathematics Division, Oak Ridge National Laboratory: PVM: Parallel Virtual Machine. `http://www.csm.ornl.gov/pvm/` (2000).
4) Date, H. and Taki, K.: A Parallel Lookahead Line Search Router with Automatic Ripup and Reroute, *Proc. European Conf. Design Automation*, pp.117–121 (1993).
5) Keshk, H., Mori, S., Nakashima, H. and Tomita, S.: A New Technique to Improve Parallel Automated Single Layer Wire Routing, *Proc. Performance Evaluation of Parallel Systems*, pp.134–141 (1993).
6) Keshk, H., Mori, S., Nakashima, H. and Tomita, S.: Amon: A Parallel Slice Algorithm for Wire Routing, *Proc. Intl. Conf. Supercomputing*, pp.200–209 (1995).
7) Keshk, H., Mori, S., Nakashima, H. and Tomita, S.: A Two Phases, Cooperative Detailed/Global Parallel Wire Routing Algorithm, *Trans. Information Processing Society of Japan*, Vol.37, No.12, pp.2376–2389 (1996).
8) Khoo, K.Y. and Cong, J.: An Efficient Multilayer MCM Router Based on Four Via Routing, *Proc. 30th Design Automation Conf.*, pp.590–595 (1993).
9) Olukotun, O.A. and Mudge, T.N.: A Preliminary Investigation into Parallel Routing on a Hypercube Computer, *Proc. 24th Design Automation Conf.*, pp.814–820 (1987).
10) Rose, J.: LocusRoute: A Parallel Global Router for Standard Cells, *Proc. 25th Design Automation Conf.*, pp.189–195 (1988).
11) Takahashi, Y.: Paralellel Maze Running and Line Search Algorithms for LSI CAD on Binary Tree Multiprocessors, *World Conf. Information and Communication*, pp.128–136 (1989).
12) Takahashi, Y.: Parallel Automated Wire Routing with a Number of Competing Processors, *Proc. Intl. Conf. Supercomputing*, pp.310–317 (1990).
13) Tanaka, K., Ohno, K. and Nakashima, H.: Amaterous: A Parallel Wire Router without Detailed/Global Feedback, *Proc. Intl. Conf. Algorithms and Architecuters for Parallel Processing*, pp.334–351, World Scientific (2000).
14) Watanabe, T., Kitazawa, H. and Sugiyama,

Y.: A Parallel Adaptable Routing Algorithm and Its Implementation, *IEEE Trans. Computer-Aided Design*, Vol.CAD-6, No.2, pp.241–250 (1987).
15) Won, Y. and Sahni, S.: Maze Routing on a Hypercube Multiprocessors, *Proc. Intl. Conf. Parallel Processing*, pp.630–637 (1987).
16) Yamauchi, T., Nakata, T., Koike, N., Ishizuka, A. and Nishiguchi, N.: PROTON: A Parallel Detailed Router on an MIMD Parallel Machine, *Proc. IEEE Intl. Conf. Computer-Aided Design*, pp.340–343 (1990).

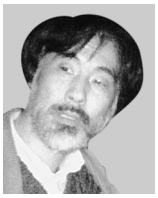
(平成 12 年 8 月 18 日受付)
(平成 13 年 2 月 1 日採録)

田中孝太郎

1975 年生．2000 年豊橋技術大学大学院工学研究科情報工学専攻修士課程修了．同年（株）日立マイクロソフトウェアシステムズ入社．在学中は並列自動配線に関する研究に従事．

大野　和彦（正会員）

1970 年生．1998 年京都大学大学院工学研究科情報工学専攻博士後期課程修了．同年豊橋技術科学大学助手．並列プログラミング言語の設計と最適化に関する研究に従事．博士（工学）．



中島　　浩（正会員）

1956 年生．1981 年京都大学大学院工学研究科情報工学専攻修士課程修了．同年三菱電機（株）入社．推論マシンの研究開発に従事．1992 年京都大学工学部助教授．1997 年豊橋技術科学大学教授．並列計算機のアーキテクチャ等並列処理に関する研究に従事．工学博士．1988 年元岡賞，1993 年坂井記念特別賞受賞．IEEE-CS，ACM，ALP，TUG 各会員．