3D−7

並列チャネル配線アルゴリズムの実装

鈴来 響太郎* 船曳 信生† 天野 英晴* 武藤佳泰‡
*慶應義塾大学 †住友金属株式会社 ‡Case Western Reserve 大学

はじめに

平行な二本の端子列間の配線を行うチャネル配線は、LSIのレイアウト時に頻繁に用いられ、特に多層のチャネルの配線には多大な計算時間を要する。この多層のチャネル配線を高速化するために、ニューラルネットワークに基づく並列アルゴリズム [1] を並列計算機上に実装し、評価を取った。

並列アルゴリズム

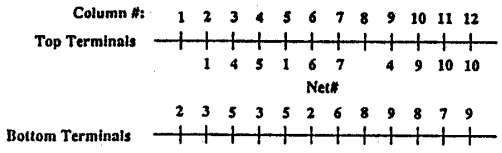
端子列に平行な方向の座標をカラム、垂直な方向の座標をトラックとする。また、平行方向の配線のみを行う配線層と垂直方向の配線のみを行う配線層を合わせて1層のレイヤとして扱う。

図1(a)は、10ネットのチャネル配線問題である。端子列の中には、1から10の間のネット番号が与えられている端子がある。同じネット番号を持つの端子同士の接続経路を決定することが解くべき問題である。

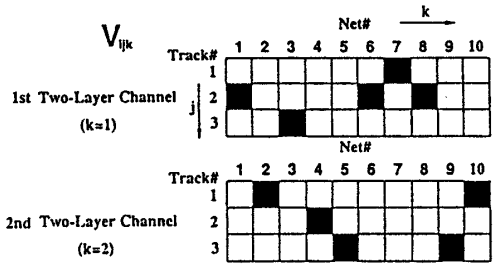
図1(b)は、図1(a)の問題を、各層3トラックの2層のレイヤに配線する場合の、各ニューロンの出力の様子である。各ネットは1つしか水平成分を持てない。そのために、水平方向の成分を割り付けるだけで、自動的に垂直成分のレイヤ番号とトラック番号も決まる。各ネットは、互いに水平方向にも垂直方向にも重なってはならない。

この例では、各ネットを割り付けるレイヤ番号とトラック番号を示すために6(=3×2)個のニューロンが用いられている。このため、10ネットの問題のためには、合計60(=10×3×2)個のニューロンが必要である。出力V_{ijk}は、i番目のネットの、j番トラック、k番レイヤに対応するニューロンの出力に対応しており、出力が1の時は、そのネットが対応するレイヤ、トラックに割り付けられていることを示し、0の時はそうでないことを示している。

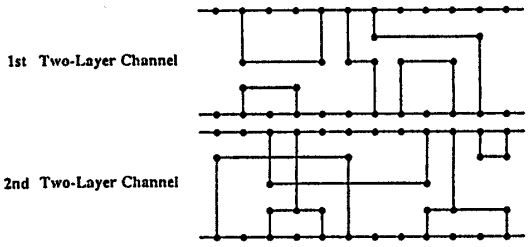
図1(c)は、図1(b)の出力状態に対応する解である。



(a) 10ネット問題



(b) 各ニューロンの出力状態



(c) 対応する解

図1 問題の表現方法

nネットmトラック1レイヤの並列アルゴリズムは、次のようになる。

0. t=0, A=B=1, C=10, U_max=20, U_min=−20, T_max=500にセットする。

1. 各$U_{ijk}(t)$を0とU_minの間のランダムな値に初期化し、各$V_{ijk}(t)$を0に初期化する。

2. 各U_{ijk}の変化分$\Delta U_{ijk}(t)$を計算する。

$$\Delta U_{ijk}(t) = -A\Big(\sum_{q=0}^{m-1}\sum_{r=0}^{l-1} V_{iqr}(t) - 1\Big)$$

$$-B\Big[\sum_{\substack{p=0\\ p\neq i\\ head_i \le head_p \le tail_i}}^{n-1} V_{pjk}(t)\Big]g(t)$$

$$-B\Big[\sum_{\substack{p=0\\ p\neq i\\ head_p \le head_i \le tail_p}}^{n-1} V_{pjk}(t)\Big]g(t)$$

$$-B\Big[\sum_{\substack{p=0\\ p\neq i}}^{n-1} T_{ip}\sum_{q=0}^{j} V_{pqk}(t) + \sum_{\substack{p=0\\ p\neq i}}^{n-1} B_{ip}\sum_{q=j}^{m-1} V_{pqk}(t)\Big]g(t)$$

An implementation of a parallel algorithm for channel routing problems

Kyotaro SUZUKI* , Nobuo FUNABIKI† ,
Hideharu AMANO* , Yoshiyasu TAKEFUJI‡
*Keio University
†Sumitomo Metal Industries, Ltd.
‡Case Western Reserve University

$$+C\,h(\sum_{q=0}^{m-1}\sum_{r=0}^{l-1}V_{iqr}(t))$$

$$g(t)=\begin{cases}V_{ijk}(t), & \text{if (t mod 10) < 5.}\\ 1, & \text{その他.}\end{cases}$$

$$T_{ip}=\begin{cases}1, & \text{上側の端子がi番ネットで下側の端子}\\ & \text{がp番ネットであるようなカラムが存}\\ & \text{在する場合.}\\ 0, & \text{その他.}\end{cases}$$

$$B_{ip}=\begin{cases}1, & \text{下側の端子がi番ネットで上側の端子}\\ & \text{がp番ネットであるようなカラムが存}\\ & \text{在する場合.}\\ 0, & \text{その他.}\end{cases}$$

$$h(x)=\begin{cases}1, & \text{if x = 0.}\\ 0, & \text{その他.}\end{cases}$$

3. 各 $U_{ijk}(t+1)$ の更新

$$U_{ijk}(t+1)=U_{ijk}(t)+\Delta U_{ijk}(t)$$

4. 各 $U_{ijk}(t+1)$ をチェックする。

$$U_{ijk}(t+1)=\begin{cases}\text{U\_min}, & \text{if } U_{ijk}(t+1)<\text{U\_min.}\\ \text{U\_max}, & \text{if } U_{ijk}(t+1)>\text{U\_max.}\end{cases}$$

5. 各 $V_{ijk}(t+1)$ の更新

$$V_{ijk}(t+1)=\begin{cases}1, & \text{if } U_{ijk}(t+1)>0 \text{ and}\\ & U_{ijk}(t+1)=max\{U_{iqr}(t+1)\}\\ & \text{for q=0,..,m-1, r=0,..,l-1.}\\ 0, & \text{その他.}\end{cases}$$

6. 全てのネットが重ならずに割り付けられるか、あるいは、t=T_max になればアルゴリズムは終了。そうでなければ、t に 1 を加えて、ステップ 2 へ行く。

実装

アルゴリズムは、マルチプロセッサ ATTEMPT-0[2] 上に実装した。ATTEMPT-0 は、16 プロセッサ (MC68030+68882) と共有メモリモジュールが IEEE-Futurebus によって、ライトスルー・スヌープキャッシュを介して接続されている (図 2)。

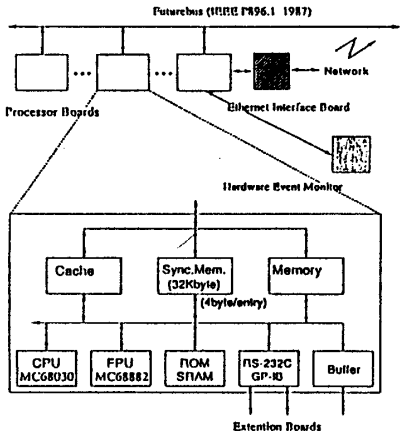


図 2 ATTEMPT-0

出力 V は、どのプロセッサも必要とするので、共有メモリに配置し、入力 U は、ローカルメモリに配置した。同期は、V の更新の開始時と終了時、収束判定の部分で取っている。

性能評価

図 3 は、並列プログラムのプロセッサ数を変えた時の、シーケンシャル版との速度の比較である。レイヤ数に関係なく、8 プロセッサを使用した場合で、シーケンシャル版の約 6 倍速くなっている。

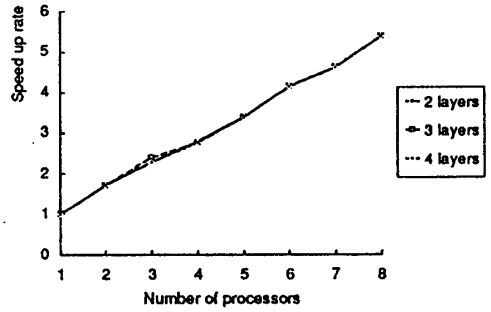


図 3 プロセッサ数による速度比

共有メモリのキャッシュのヒット率は大変高く、8 プロセッサの場合でも、98%以上を保っている。これは、キャッシュメモリ上に、唯一の共有データである出力 V が収まってしまうためである。

図 4 は、プロセッサ数別の交信時間とその内訳を示している。この図によると、キャッシュブロックの replace が交信時間のほとんどを占めており、バス待ちによるネックは問題になっていない。また、交信時間の実行時間全体に占める割合は、8 プロセッサ時でも 0.3%以下であった。これは、少ないバス利用と、Attempt-0 の大きなキャッシュブロックサイズ (64byte) のためである。

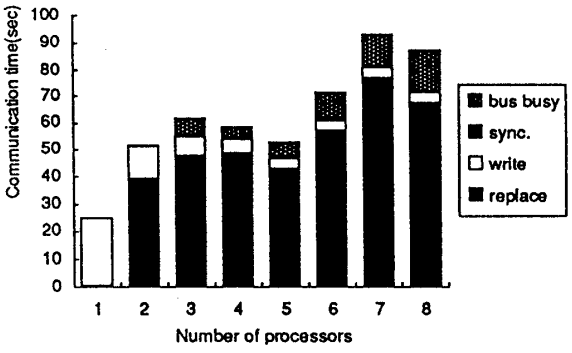


図 4 プロセッサ数と交信時間

参考文献

[1] N.Funabiki and Y.Takefuji, "A Parallel Algorithm for Channel Routing Problem," to be appeared in IEEE Trans. on CAD.

[2] H.Amano,T.Terasawa and T.Kudoh,"Cache with synchroniza-
tion mechanism,"Proc. of IFIP Congress89, Aug. 1989, pp1001-1006.