並列マシン Cenju 上の電気系 CAD

山内　宗

日本電気 (株) C&C 研究所

概要 我々は、主に電気系 CAD をターゲットとして並列計算機アーキテクチャの研究を行なって来た。本稿では、MIMD 型分散共有メモリ型並列計算機 Cenju シリーズ上で稼働する実アプリケーションのうち、回路シミュレーション、LSI 自動配線、グラフ分割処理等、莫大な計算量を必要とする電気系 CAD の基本アルゴリズム、並列化 / 実装手法、通信プリミティブ (RPC、ブロック転送、リモート・メモリ・アクセス等) と同期手法 (barrier、flag 等) の選択、及び評価結果について報告する。

We have researched parallel computer architectures focusing on CAD applications. In this paper, we describe basic algorithms, parallelizing/implementation methods, inter-processor commnication primitives (remote procedure call, block transfer, remote memory access), synchronizing methods and evaluation results of circuit simulation, LSI router and graph partitioner.

1　はじめに

近年、MIMD (Multiple Instruction streams Multiple Data streams) 型並列計算機の研究開発が進み、商用化も開始されている。我々は、主に電気系 CAD をターゲットとして MIMD 型並列計算機アーキテクチャの研究を行なって来た。これは、電気系 CAD はシミュレーション、レイアウト関連等、莫大な計算量を必要とするものが多く、また、高い並列性を有する応用が多いと考えられたからである。

並列計算機における効率良い処理を阻害する問題点として、以下の点が挙げられる。

- 通信のオーバーヘッド
- 負荷分散の不均衡による同期のオーバーヘッド
- 逐次部分のオーバーヘッド

本稿では、我々が今まで開発して来た MIMD 型並列計算機の上に電気系 CAD の応用を実装するにあたり、上記の問題を如何に克服して高い並列性を得て来たかについて述べる。

2　MIMD 型並列計算機 Cenju

回路シミュレーションを高速化することを主目的として、マイクロ・プロセッサを用いた MIMD 型分散共有メモリ型並列計算機 Cenju[1] を完成させたのが 1987 年であり、その後、マイクロプロセッサの技術革新に伴い Cenju2[2]、Cenju-3[3] を開発してきた。

以下に、Cenju, Cenju2, Cenju-3 のマシンアーキテクチャの特徴をまとめる (以後 Cenju, Cenju2, Cenju-3 を総称して Cenju シリーズと呼ぶことにする)。

- Cenju

Cenju は、8 個のクラスタで構成され、各クラスタ間はパケット交換の多段網で接続されている (図 1参照)。各クラスタは、バスで接続された 8 台の要素プロセッサ (PE) で構成されている。各 PE はモトローラの MC68020　(20MHz) と 4MByte の主記憶で構成され、メモリ空間は各 PE に分割された分散共有メモリの構成をとっている。

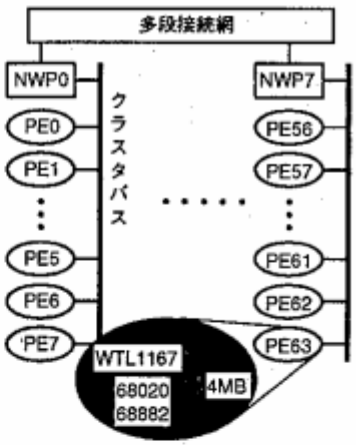

図 1: 並列計算機 Cenju のブロック図

- Cenju2
  Cenju2 は、最大 256 台の PE を接続可能であり、各 PE はパケット交換の多段網に直接接続されている (図 2参照)。各 PE は MIPS R3000　(25MHz) が 2 個と 64MByte の主記憶で構成され、メモリ空間は各 PE に分割された分散共有メモリの構成をとっている。
- Cenju-3
  Cenju-3 は、最大 256 台の PE を接続可能であり、各 PE はパケット交換の多段網で接続されている (図 3参照)。各 PE は MIPS

CAD Applications on Parallel Machine Cenju.

Tsukasa YAMAUCHI

C&C Research Laboratories, NEC Corporation

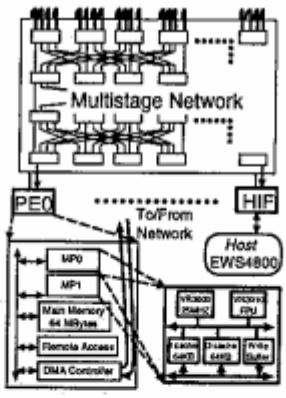

図 2: 並列計算機 Cenju2 のブロック図

R4400 (内部 150MHz) と 64MByte の主記憶及びハードワイヤード化されたネットワーク・インタフェースで構成され、メモリ空間は各 PE に分割された分散共有メモリの構成をとっている。

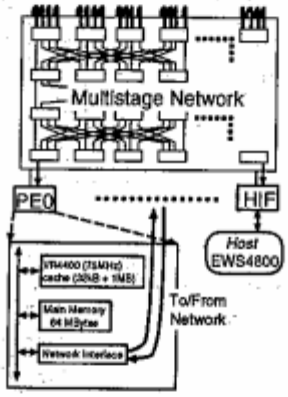

図 3: 並列計算機 Cenju-3 のブロック図

3 プログラミング環境

Cenju シリーズのプログラムは、C、Fortran 等の手続き型言語＋プロセッサ間通信や同期のためのプリミティブという形で記述されるが、Cenju シリーズはどれも分散共有メモリの構成であり、ソース・レベルでのソフトウェア互換性をなるべく保つことを基本としている。Cenju シリーズで使用可能な代表的なプロセッサ間通信、同期のためのプリミティブを以下に示す。

- 遠隔メモリ・アクセス
  Cenju シリーズは分散共有メモリ構成なので、ローカル・メモリ以外のアドレスへのアクセス (read/write) は、システムにより自動的に遠隔メモリへのアクセスと判断され、ネットワーク経由で該当するメモリを所有する PE へ要求が送られる。ネットワークは回線交換ではなくパケット交換なので、遠隔メモリ・アクセスにおいては、write と比較して read が遅い。
- 遠隔手続き呼び出し
  遠隔 PE のプログラム中の指定した関数を呼び出すのが「遠隔手続き呼び出し」である。「遠隔手続き呼び出し」は排他的な処理であり、キュー操作等に用いることが可能である。
- バリア同期
  各 PE は、プログラム中のバリアまで到達すると同期待ち状態に入り、特定のグループ内の PE、あるいは全 PE が全てバリアに到達した後に先に進むことが可能になるというのがバリア同期である。Cenju シリーズにおいては、予め登録したグループ内のバリア同期と全 PE のバリア同期の 2 種類がある。
- ブロック転送
  連続領域に存在する大量のデータを PE 間で転送する場合には、ブロック転送のプリミティブを用いると効率が良い。これは遠隔メモリ・アクセスと異なり、システムが自動的に転送を認識するのではなく、プログラマが明示的に転送を記述する。

4 回路シミュレーション

Cenju は当初回路シミュレーションを高速化することを主目的として開発された [1]。回路シミュレーションは、回路の素子特性を元に、回路の各節点に関する非線形常微分方程式を解き、回路の振舞いをアナログレベルで解析するものである。

回路シミュレーションの基本的な処理手順 (直接法) を図 4に示す。処理の重さの比率としてはモデル評価が 83%、線形求解が 9% 程度を占めるので、これらを並列化することが重要である。図 5に示す様に、モデル評価は各素子毎に並列に処理することが可能であるが、回路を表現しているマトリックスが大規模ランダムスパースマトリックスなので逐次処理である線形求解の部分が並列処理の効率を下げるネックとなる。

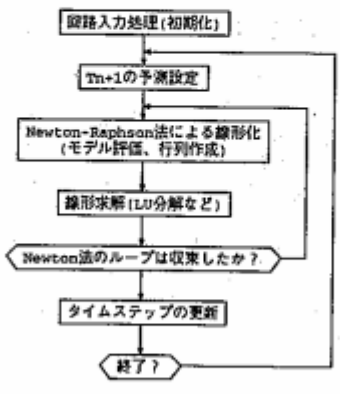

図 4: 回路シミュレーションの処理手順

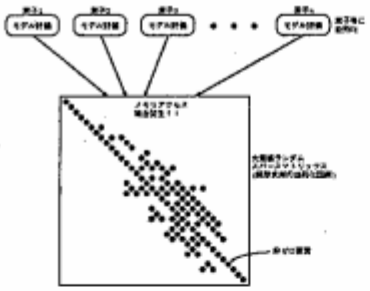

図 5: 直接法での並列化の問題点

4.1 回路シミュレーションの並列化手法

直接法をそのまま並列化すると前述の様な問題が生じ、効率が悪いので、我々は「モジュール分割法に基づく回路シミュレーション」のアルゴリズムを開発した。これは、(1) 回路を部分回路に分割し、各々の部分回路を各プロセッサに割り当て、各プロセッサはその部分回路についてのモデル評価及び部分回路の行列作成を行なう、(2) 一台のプロセッサで部分回路間の接続に関する線形方程式を解いて境界変数の解を求める (逐次部分)、(3) 境界変数の解を元に、各プロセッサが部分回路の線形方程式を解く、という処理を繰り返すものである (図 6、7参照)。この様にすることにより、一台のプロセッサで線形求解する部分回路間の境界変数の方程式の大きさをより小さく且つ密なものにすることが可能となり、逐次部分のボトルネックを減らすことが可能となる。そして、部分回路間の境界変数の方程式の非零要素率は 20 ～ 30% と元の行列よりかなり高いので、その線形求解の主な処理である LU 分解を並列化したところ、より高い台数効果が得られた (図 8参照)。

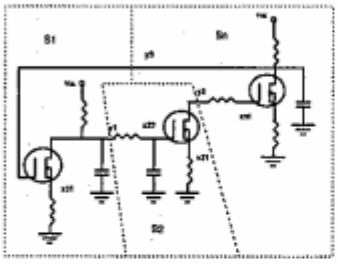

図 6: モジュール法の概念

4.2 負荷分散と通信パターン

提案した手法においては、回路をどのように分割するかが性能向上の上で重要である。負荷分散を考慮すると、モデル評価にかかる処理時間を均等にするために各プロセッサが受け持つ部分回路が含む素子数をなるべく均一にする必要がある。また、部分回路間にまたがる配線の本数は、線形求解する行列の大きさに関係するので、逐次部分を減らすためになるべく少ない本数になる様に回路を分割することが望ましい。

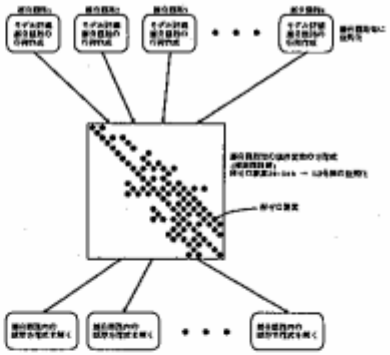

図 7: 高い並列性を得る工夫

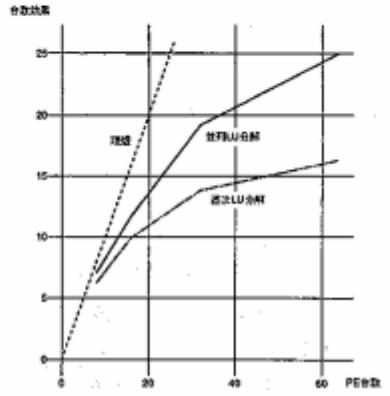

図 8: 回路シミュレーションの Cenju 上での評価

プロセッサ間の通信は、部分回路間の境界変数に関わる分だけであり、通信パターンは線形求解をするマスタプロセッサ (LU 分解を並列化した場合もとりあえずマスタプロセッサに集める) とスレーブプロセッサ間のブロードキャストとギャザーだけである。この様に通信パターンが規則的なので、共有メモリアクセスよりもシステムのオーバーヘッドが少ないブロック転送 (ライト) を用いて回路シミュレーションを実装した。

5 LSI ルーター

配線処理は, VLSI を開発する際の重要な課題の一つとして挙げられるが, 莫大な処理時間とメモリ空間を必要とする. 従って, 今後大規模な VLSI の配線を行なうためには, 配線アルゴリズムの並列化による高速化が重要である. また, 大規模な VLSI の配線処理が必要とする莫大なメモリ空間については, データを並列計算機の各プロセッサに分散配置するという方法をとり, プロセッサ数を増すことによって, より大きな VLSI の配線処理にも対応が可能となる. 従って, 今後 VLSI が大規模化するにつれて, 配線処理の並列化は重要となると考えられる.

5.1 Cenju 上での評価

改良線分探索法をベースとした配線アルゴリズムを用い、チャネルレス・ゲートアレイを配線対象とした、並列 LSI ルーター PROTON[4][5] を Cenju 上に実装、評価した。配線アルゴリズムは大附らの「改良線分探索法」を基本としたものである。このアルゴリズムは、障害物等の周辺にその領域を回避する様な配線経路の候補となる線 (エスケープ・ライン、EL) を予め生成し、それらの EL の中から最小折れ曲がりの経路を求める。基本的な処理手順を図 9に示す。

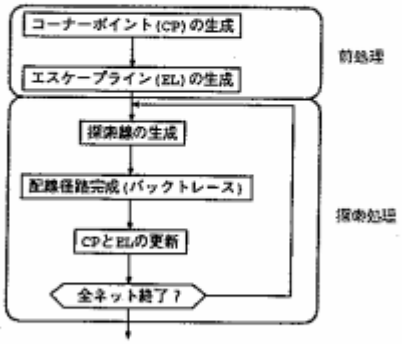

図 9: LSI ルーター PROTON の処理手順

5.1.1 配線処理に内在する並列性

配線処理には、一つのネットを配線する処理に内在する並列性 (ネット内並列性) と複数のネットを同時に配線することによって得られる並列性 (ネット間並列性) の 2 つのレベルの並列性が内在すると考えられる。

PROTON では、図 10の様に配線領域を配線方向に従って帯状に分割して各 PE にそれぞれの配線領域内での線分の探索を担当させることにより、ネット内並列性を用いている。

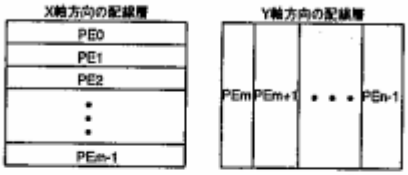

図 10: 配線領域の各 PE への割り当て

ネット間並列性を用いた種々の並列配線アルゴリズムが提案されているが、高い並列性を得ようとすると配線率、配線の品質が低下するというものが多い。しかし、PROTON では、以下に述べる様に概略配線の結果を用いることにより、逐次アルゴリズムと比較して全く配線率や配線の品質を損なわずにネット間の並列性を引き出している。

一般的に、LSI の自動配線は概略配線と詳細配線の二つのフェーズで構成されていることが多い。これは、概略配線を用いることによって、探索空間を絞り込んで詳細配線の処理時間を削減し、配線の順番が配線の品質、配線率に与える影響を減らすためである。しかし、詳細配線は概略配線の領域内で行なわれるということを考えると、概略配線領域が重ならないネットは同時に詳細配線処理を行なうことが可能であることがわかる。そこで PROTON では、概略配線の領域が重ならないそれらのネットを並列に配線することにより、ネット間の並列性を引き出している。

PROTON でどの様にネット間の並列性を用いているかを図 11に示す。

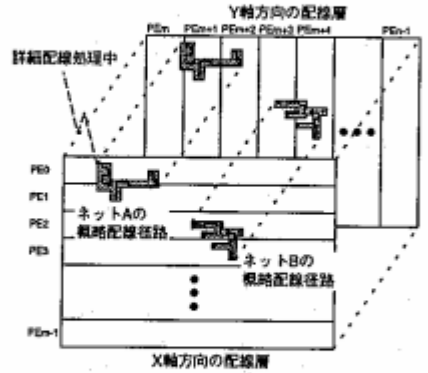

図 11: ネット間並列性

同時に配線可能なネットのグループを一つの RU(Routing Unit) として扱い、一つの RU の配線が終ったら全ての PE が同期をとり、次の RU の配線へと進む。その様子を図 12に示す。この図は PE 台数が 8 台の場合であり、9 つのネット (#1 ～ #9) は、各々の概略配線に従って 3 つの RU (*RU*0 ～ *RU*2) に分けられている。

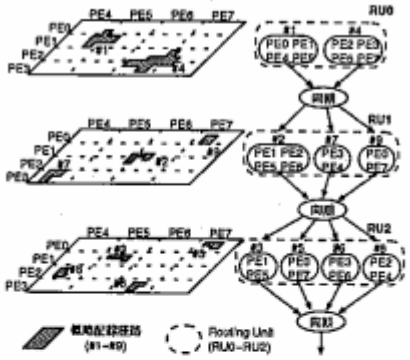

図 12: ネット間並列性と RU

各ネットの間では概略配線径路の重なり以外に依存関係が無いので、RU の選択はスケジューリングというよりはマッピングに近い処理となる。この様なマッピングには、bin-packing アルゴリズムが適している。但し、単純に概略配線径路の重なりだけでマッピングをすると、負荷分散のばらつきが生じるおそれがあるので PROTON では、それ以外にネットを構成するピンペア数、概略配線径路の面積等も考慮にいれる様に bin-packing のアルゴリズムに改良を加えている。

5.1.2 PROTONの配線結果及び評価

図13にPROTONの評価結果(速度向上比)を示す。これは、「ゲートアレイA(2,832×2,742格子、2,495ネット(4,454ピンペア)、障害物57,487個)」、「ゲートアレイB(1,537×1,790格子、5,842ネット(12,591ピンペア)、障害物151,982個)」の二種類の配線結果である。これを見ると、ゲートアレイAでは、台数効果が17倍程度で飽和してしまっているが、ゲートアレイBでは、PE数が64台の時に43倍の台数効果が得られている。ゲートアレイAでは、セルが配線領域に中心部により多く存在しているために、PROTONの様な領域分割に基いた並列処理では負荷分散に偏りが生じてネット内並列性が損なわれたことが原因と考えられる。また、ゲートアレイAでは、各ネットの大きさが大きく、ネット間並列性があまり高くないということも原因として考えられる。一方、ゲートアレイBは、配線領域に一様に各セルが分布しており、各ネットの大きさも小さいのでネット内、ネット間共に高い並列性があり、43倍という高い台数効果が得られたと考えられる。

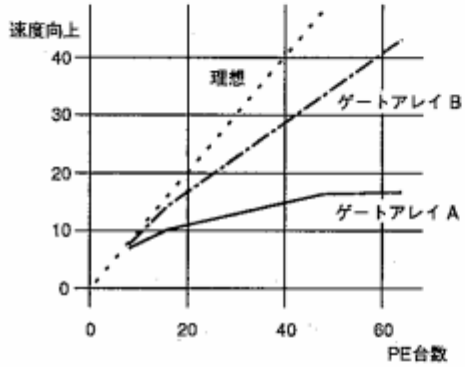

図 13: 台数効果

5.2 Cenju2上での評価

PROTONは並列マシンCenju用に開発したLSIルーターであったが、(1)線分探索をベースとしたアルゴリズムでは、径路長を短くすることが難しい、(2)Cenju2はRISCプロセッサを採用したので、プロセッサ間通信と処理の比率がかなり変化した、等の理由により、大幅なアルゴリズムの変更を必要とした。以下にCenju2用に開発したPROTON2[6]の基本配線アルゴリズムとその並列化手法を示す。

5.2.1 基本配線アルゴリズム

PROTON2では、図14に示す様に、障害物の周囲に生成したエスケープ・ラインの交点を頂点とし、頂点間の物理的な距離を枝の重みとする無向グラフを生成し、その上でDijkstraのアルゴリズムを用いて端子間の最短配線径路を求める(この際、端子からも探索線を生成して、端子をグラフに含んでおく)新たな配線アルゴリズムを考案した。

5.2.2 並列化手法

PROTON2では、(1)ネット内並列性の粒度は小さ過ぎる、(2)ネット内並列性を用いると

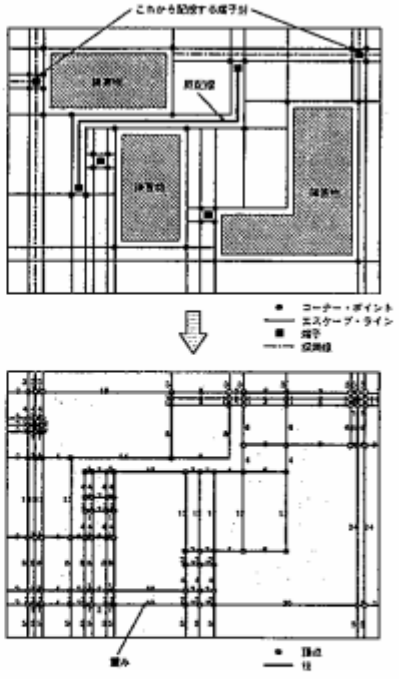

図 14: エスケープ・ライン、重み付き無向グラフ生成

配線径路の質(配線長等)を高く保つことが困難、の二つの理由から、領域分割に基づく並列性とネット間並列性のみを用いることとした。エスケープ・ラインの生成と更新は領域分割に基づく並列性を用い、グラフの生成、配線径路の探索はネット間並列性を用いている。

- 領域分割に基づく並列性
  エスケープ・ラインの生成や更新は配線領域に存在する障害物の情報のみで可能なので、配線領域の分割に基づく並列性を用いるのが適している。各配線層の配線方向(主軸方向)にそって配線領域を帯状に分割し、各プロセッサ(PE)は、自分が担当する領域内のエスケープ・ラインを生成、更新する。
- ネット間並列性
  LSIルーターPROTONの節で記述した様に、概略配線領域が重ならないネットは同時に詳細配線処理を行なうことが可能である。そこでPROTON2でも同様に概略配線の領域が重ならないそれらのネットを並列に配線することにより、ネット間の並列性を引き出している。
  具体的には、グラフの生成、配線径路の探索にネット間の並列性を用いる。しかし、エスケープ・ラインの生成、更新は領域分割に基づく並列性を用いており、グラフの生成、配線径路の探索に必要なエスケープ・ラインは各PEが分散して保持している。従って、あるネットに関してグラフの生成、配線径路の探索を行なう場合には、(1)必要なエスケープ・ラインの転送、(2)配線結果の分配(エスケープ・ラインの更新のため)、をする必要がある。

実装上に関しては、通常この様な探索問題ではリスト構造をとることが多いが、(1) キャッシュのヒット率を上げる、(2) プロセッサ間通信を高速なブロック転送にする、等の理由から配列表現を用いる等の工夫をした。

5.2.3 PROTON2 の処理フロー

以上の 2 種類の並列性を考慮した PROTON2 の処理フローを図 15に示す。

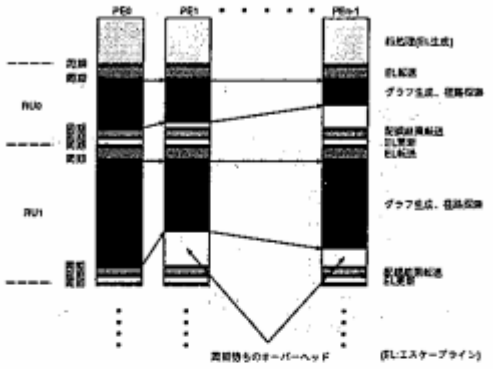

図 15: PROTON2 の処理フロー

各 PE は自分が配線処理をするネットの概略配線径路内に含まれるエスケープ・ライン (EL) を受けとり (EL 転送)、その EL からグラフを生成し径路を探索する。そして、一つの RU に含まれるネットの配線が終ったら全ての PE が同期 (バリア) をとり、配線結果を互いに転送し、領域分割に従ってエスケープ・ラインの更新を行ない、次の RU の配線へと進む。

5.2.4 ネット間並列性の抽出

高いネット間並列性を得るためには、同時配線可能なネットのグループ (RU) をどの様に選ぶかが重要である。各ネット間では概略配線径路の重なり以外には依存関係は無いが、同じ RU 内のネットの配線時間にバラツキがあると、図 15に示す通り同期待ちオーバーヘッドが大きくなり、処理効率の低下招く。従って、配線時間を見積り、それに従って RU のスケジューリングをするのが望ましい。なお、スケジューリング方式は負荷 (配線時間) の順にネットをソートし、その順番に同時に配線可能なネットを選ぶという greedy アルゴリズムとした。また、負荷の見積り指標としては、ピンペア数×概略配線径路の面積を用いた。

5.2.5 PROTON2 の配線結果及び評価

本論文では、ゲートアレイ A (30k ゲート、2 層、5,842 ネット (12,591 ピンペア))、ゲートアレイ B (42k ゲート、3 層、8,368 ネット (17,948 ピンペア))、ゲートアレイ C (250k ゲート、3 層、27,148 ネット (59,835 ピンペア)) の三種類のゲートアレイ (ゲート敷き詰め型) を用いて PROTON2 の評価を行なった。

図 16に台数効果のグラフを示す。これを見ると、中規模のゲートアレイ (A、B) でも、台数効果が 7 倍程度で飽和しているが、これはネットの配線処理時間に 2 桁から 3 桁ものばらつきがあるためである。特に、クロック系等のピンペア数、概略配線径路の面積共に大きなネットが原因となっていると考えられるが、この様な特殊なネットの数は限られており、対象となるゲートアレイの規模が大きくなる (数百 k ～数 M ゲート) につれて逐次部分としての影響は減少すると考えられ、確かにゲートアレイ C ではプロセッサ台数 63 台の Cenju2 で 21.6 倍という高い台数効果が得られている。

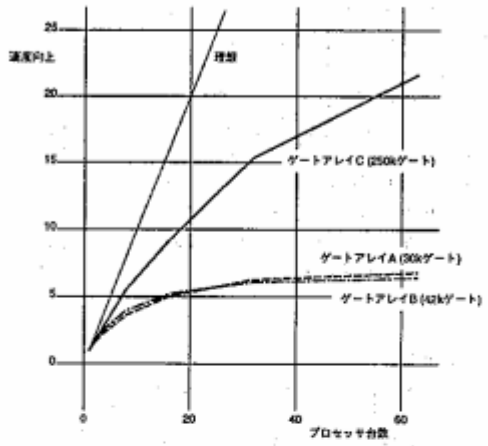

図 16: 台数効果

5.3 Cenju-3 上での評価

Cenju-3 は Cenju2 と近いアーキテクチャであり、ソフトウェアの互換性も高いので、PROTON2 をそのまま移植し評価をとった。図 17は ゲートアレイ C を配線対象とした場合の台数効果を Cenju2 と比較したものである。Cenju2 との比較を容易にするために、縦軸は Cenju2 の 1 プロセッサの性能で正規化している。このグラフを見ると、PROTON2 を実行した場合に、Cenju-3 は Cenju2 の 3 ～ 4 倍の性能が得られることがわかる。

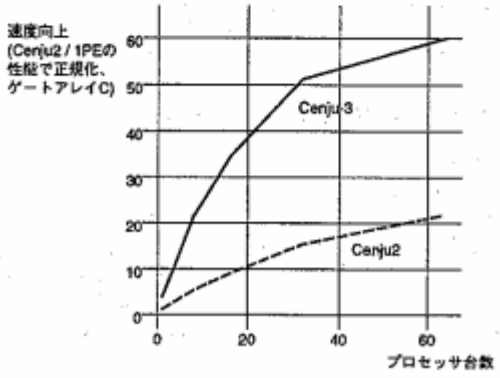

図 17: Cenju3 上での評価 (Cenju2 との比較)

6 並列グラフ分割アルゴリズム

グラフ分割問題は、電気系 CAD 分野における重要な問題のひとつであり、例えば、回路シミュレーションにおける回路分割や、LSI レイアウトにおける配置処理に利用されている。グラフ分割問題は、カット数が最少となるようにグラフのノードを複数のグループに分割する問題であり、ノード数を均等に 2 分割する場合でも NP 完全問題であることが知られている。我々は、グラフを k 分割する並列処理手法を提案し、並列計算機 Cenju2 上で実装、評価した [7]。

6.1 並列グラフ k 分割手法

我々は、k 分割されたグループから、2 つのグループを選んで得られる全てのペアに対して、非同期に 2 分割処理を適用する並列グラフ分割手法を提案した。

本手法の手順を以下に述べる。

(1) 初期解としてグラフを k 個のグループに任意に分割する。
(2) それらのグループから並列に実行可能な任意のグループのペアを選択し、それぞれのペアに対して異なるプロセッサを割り当て、2 分割処理を行なう。
(3) 全てのペアの最適化が終了するまで (2) を繰り返す。
(4) カット数が収束するまで (2) と (3) を繰り返す。

図 18に示すように、2 分割処理は対象としているグループに含まれるノードと、そのグループに含まれるノード間のネットのみを扱うので、同じグループを含まないペアは並列処理が可能である。よって、k 分割の場合、並列処理できる最大の 2 分割処理数は $k/2$ である。

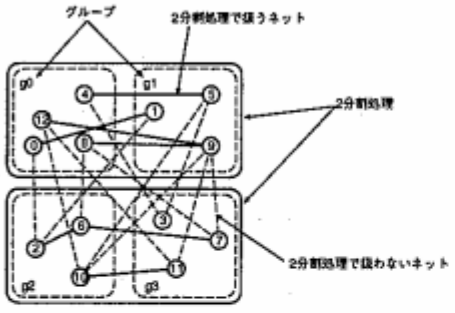

図 18: 2 分割処理の並列処理

6.2 2 分割処理のアルゴリズム

本手法で用いる 2 分割最適化アルゴリズムは、Fiduccia と Mattheyses のアルゴリズム (FM 法) を元に、クラスタリング手法を採り入れたものである。

FM 法は、Kernighan と Lin の方法 (KL 法) を改良したアルゴリズムであり、(1) 多端子ネットグラフを扱うことができる、(2) ノードの重みの総和を任意の比率に分割できる、(3)N ノードの分割を $O(N)$ の計算時間で処理できる、という特徴を持つ。

クラスタリング手法は、複数のノードをクラスタ化し、クラスタ単位での交換を行うことで、単純な FM 法に比べて、より良質の解を得ることを目的とする (局所解に陥ることを避ける)。

6.3 プロセッサへの割り当て順序

本手法で並列処理を行うとき、より高い並列性を得るためには、できる限り多くの 2 分割タスクを同時に実行できるように、プロセッサへの割り当て順序を考慮する必要がある。そこで、グラフの彩色問題を利用して 2 分割タスクの割当て順序を決定することによって、並列性を改善する。実際には、実行時にプロセッサへの 2 分割タスク割り当て状況も考慮し、あらかじめ定められた 2 分割タスクが実行出来ない場合は、実行可能な 2 分割タスクを優先して実行する。

6.4 実装方法

本手法を *Cenju2* 上に実装する方法を示す。1 台のプロセッサをマスタとして、分割結果の維持と各 2 分割タスクの起動を行ない、残りのプロセッサをスレーブとして、2 分割タスクを実行する。また、分割対象のグラフは全てのプロセッサのメモリ上に配置する。

まず、マスタ上で乱数を用いて初期分割を行なう。次に、グラフ N 彩色問題を利用して静的に定められた順序と、実行時のプロセッサへの 2 分割タスクの割り当て状況から、次に実行する 2 分割タスクの対象グループと、割り当てるスレーブを決定し、そのスレーブに対象グループとその時点での分割状態を転送して分割タスクを開始する。スレーブは 2 分割タスクが終了後、分割結果をマスタに転送して処理を終了する。

これらの通信は、リモートライト (バーストメモリ転送) によって行なう。

6.5 実データを用いた評価

対象としたゲートアレイの規模は約 11K ゲートであり、グラフのノード数は約 1800 個である。最適化ループ 1 回に要した時間を処理時間とし、1 台のスレーブを用いたときの処理時間を 1 として、複数台のスレーブを使用した時の速度向上率を評価した。分割数は 10 分割、20 分割および 40 分割とした。実験結果を図 19に示す。

10 分割時には 4 台の PE で最大 2.76 倍、20 分割時には 10 台で最大 8.54 倍、40 分割時には 14 台で最大 10.7 倍の速度向上が得られた。

6.6 速度向上を妨げる要因

速度向上を妨げる要因には以下のものがある。

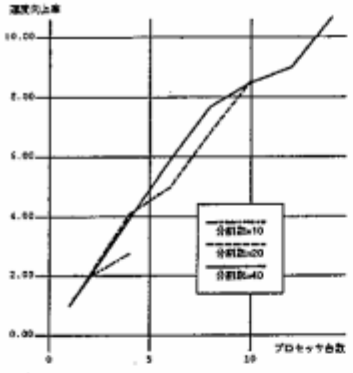

図 19: 並列 k 分割手法の速度向上率

(1) 2 分割タスクの並列度の低下
未処理のペアが少なくなった時にプロセッサの稼働率を落とすことになる。また、タスク割り当ての順序が最適でないために、並列処理可能なペア数が減少する場合もある。

(2) マスタのタスク割り当て処理
並列処理することによって生じたオーバーヘッドである。各 2 分割処理の対象ノード数が少ない場合やプロセッサ台数が多くなった場合に、相対的にオーバーヘッドの割合が大きくなる。マスタがボトルネックになってスレーブの稼働率が低下するのを避けるために、集中管理ではなく、分散管理手法を検討する必要がある。

(3) 通信オーバーヘッド
1 台のプロセッサによる逐次処理と比較すると、マスタとスレーブ間の通信オーバーヘッドが存在する。本手法では、(1)2 分割タスク開始時および終了時ののマスタ、スレーブ間の整数配列のリモートライト、(2)2 分割タスク起動および終了時のタスク制御のためのリモートライト、の通信が存在する。
また、これらの通信によるオーバーヘッドは、(a) データがスイッチを経由してプロセッサ間を移動するのに要する時間、(b) 通信がマスタに集中、衝突することによって生じる待ち時間、の 2 つの要素に分けられる。
(a) のデータ通信時間は、並列処理をすることで固定的に生じるものである。(2) のタスク制御のための通信データ量はわずかなので、(2) の整数配列のリモートライトのためのデータ通信のオーバーヘッドを、スレーブ 1 台時の、マスタとスレーブの稼働率から求めてみる。図 20に示すように、スレーブが 1 台の場合、スレーブの待ち時間は通信時間とマスタのタスク割り当て時間の和になっている。表 1から、データの通信時間は、スレーブ 1 台で処理する時間の数 % である。このオーバーヘッドは、プロセッサ台数が増え、全体の処理時間が短縮されると、相対的に大きくなる。
一方、(b) のマスタに通信が集中することによるオーバーヘッドに関しては、最適化タスクの起動、終了が非同期的であるため、プロセッサ台数が少ない場合は通信の衝突の確率は低く、影響は少ない。しかし、プロセッサ台数が多く通信頻度が多い場合、あるいは分割数が多く、2 分割タスクの処理時間が短い場合には、スレーブの稼働率に与える影響は大きくなる。これらのオーバーヘッドの定量的な評価と対策は今後の課題である。

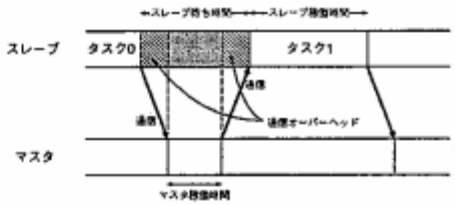

図 20: スレーブ 1 台時の処理の流れ

表 1: スレーブ 1 台時のプロセッサの稼働率

分割数	マスタの稼働率	スレーブの稼働率
10	0.35 %	99.1 %
20	0.98 %	97.7 %
40	2.62 %	94.5 %

参考文献

[1] 中田 他, 「並列回路シミュレーションマシン Cenju」, 情報処理 (30 周年記念特集号), Vol. 31, No.5, pp.593–601, (May, 1990).

[2] 松下、山内、中田, 「並列マシン Cenju2 のアーキテクチャ」, 情報処理学会計算機アーキテクチャ研究会資料 Vol.92, No.64, pp. 17-23, Aug(1992).

[3] 広瀬, 加納, 丸山, 中田, 浅野, 稲村, 「並列コンピュータ Cenju-3 のアーキテクチャ」, 情処研報 Vol.94, No.66, pp.121-128, Jul.(1994).

[4] T. Yamauchi, A. Ishizuka, T. Nakata, N. Nishiguchi and N. Koike, *'PROTON: A Parallel Detailed Router on an MIMD Parallel Machine'*, ICCAD-91, pp340-343 (1991).

[5] 山内, 中田, 石塚, 西口, 小池, 「MIMD 型並列計算機上の LSI ルーター PROTON −」, 情処論文誌, Vol.34 No.4, pp.699-707 (1993).

[6] 山内, 中田, 石塚, 高見沢, 小池, 「並列 LSI ルーター PROTON2 −並列マシン Cenju2 上での評価−」, 並列処理シンポジウム JSPP'94, pp271-278 (May, 1994).

[7] 黒岩 他, 「並列計算機 Cenju2 上での並列グラフ分割アルゴリズムの実装と評価」, 第 6 回 回路とシステム軽井沢ワークショップ (1993).