計算機アーキテクチャ 110−17
設 計 自 動 化 73−17
（1995. 1. 20）

MCM設計のためのパフォーマンスと物理的制約を考慮した回路分割手法

桂 嘉志記　小出 哲士　若林 真一　吉田 典可

広島大学 工学部

〒724 東広島市鏡山一丁目4番1号

近年，新しい実装技術としてマルチチップモジュール(MCM)が注目を集めている．MCMでは，基盤上にベアチップを直接マウントするため，プリント基板(PCB)に比べ，基板面積，配線長を大幅に改善することができる．しかし，MCM回路分割では，パフォーマンス向上のためにチップ間の配線遅延や，物理的な制約としてチップ面積やチップのI/Oピン数を考慮する必要があるため，従来のICに対する回路分割手法をそのまま適用することができない．そこで本稿ではこれらの制約を同時に考慮したMCM回路分割手法を提案する．提案手法では，分割問題に0-1整数計画法を適用し，さらに制約を考慮した2種類のクラスタリングを用いて，問題のサイズをできるだけ小さくすることにより実用的な計算時間内で解を求めることが可能である．

A Circuit Partitioning Method Considering Performance and Physical Constraints for Multi-Chip Module Layout Design

Yoshinori KATSURA, Tetsushi KOIDE, Shin'ichi WAKABAYASHI and Noriyoshi YOSHIDA

Faculty of Engineering, Hiroshima University

4-1, Kagamiyama 1 chome, Higashi-Hiroshima 724, JAPAN

Recently, Multi-Chip Module (MCM) has been attracted as a new packaging approach. MCM has smaller size and shorter total wire length than Printed Circuit Board (PCB), becasuce MCM can mount bare chips on board directly. But conventional circuit partitioning methods for ICs can not apply the MCM circuit partitioning problem, because we must consider delays between chips for performance constraint, and areas and the number of I/O pins of chips for physical constraint. In this paper, we present a circuit partitioning method for MCM considering such constraints. The proposed method applies 0-1 integer programming to the partitioning problem. The method also uses two clustering methods considering the constraints to make the size of the problem as small as possible so that the proposed method can find a solution in a practical computation time.

1 まえがき

電子回路の最終的な実現形態としては，全回路を1チップに実現することが望ましいと考えられる．しかし，現在のVLSI製造技術では，100万ゲート規模のシステムを1チップに実現することは不可能である．そのため，従来から電子システムの実装技術としてプリント基板（PCB）が一般的に用いられている．PCBによる実装では，個々のICをパッケージし，それらをPCB上にマウントし，配線を行なう．そのため，PCBでは，チップ間の距離や大きさが制限され，高集積化には限界があった．しかし，近年のVLSIの高性能化につれて，PCBによる実装が高性能なシステムを設計することを妨げる障害となっており，より優れた実装技術が求められるようになってきた．そこで，PCBに代わる実装技術として，複数のチップや部品を1つの基板に詰め込んだマルチチップモジュール(MCM)が注目を集めている．MCMは，図1に示すように，セラミックなどの基板上にベアチップを直接マウントしたものであり，従来の実装技術であるPCBに比べ，基板面積，配線長を大幅に改善することができる．このため，既存のLSI製造技術，高密度実装技術，CAD技術を利用すると，1チップに相当する高性能なシステムを設計することが可能となり，RISCチップなどの高速演算処理，IC・LSIの複合演算処理などの必要性からそのニーズが急速に高まっている．

このように高密度・高性能が要求されるMCMに従来から提案されているIC, PCBに対する回路分割，クラスタリング，及び配置手法[1, 2, 3, 5, 14, 15]をそのまま適用することは望ましくない．その理由として，MCMでは高密度な配置となるため，レイアウト設計においては，消費電力，及び放熱を新たに考慮する必要がある．また，チップ間の配線遅延がチップ内の配線遅延に比べ，非常に大きいため，チップ間の配線遅延を陽に考慮する必要がある．さらに，物理的な制約としてチップ面積やチップのI/Oピン数を考慮する必要もある．そのため面積制約しか考慮していない従来の分割手法は，MCMの回路分割問題には向いていないと考えられる．

上述のパフォーマンスと物理的制約を考慮に入れた分割手法としては，遅延制約とチップ面積制約を考慮したもの[7, 8, 9, 11, 13]や，I/Oピン制約とチップ面積制約を考慮したもの[10, 12, 16]がある．これらの制約を考慮した分割問題に対する解の求め方として，例えば文献[12]では2次計画問題に定式化し，2次の目的関数を線形式に変換することにより解を求めている．しかし，遅延，チップ面積，及びチップのI/Oピンの3つの制約を同時に考慮した分割手法はまだ知られていない．そこで，本稿ではこれら3つの制約を同時に考慮したMCMシステム分割手法を提案する．提案手法では，分割問題に0-1整数計画法を適用し，さらに制約を考慮した2種類のクラスタリングを用いて，問題のサイズをできるだけ小さくすることにより実用的な計算時間内で解を求めることが可能である．

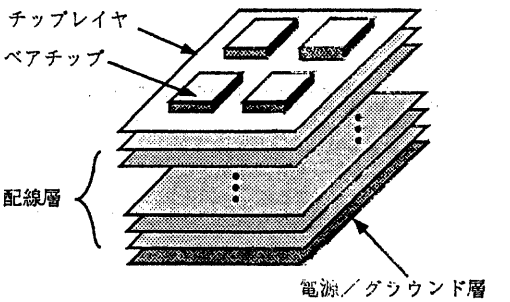


図1 MCMモデル

以降では，2節で提案手法で用いる用語，システムグラフ，及び配線遅延モデルを定義し，MCMシステム分割問題の定式化を行なう．3節では，提案アルゴリズムの詳細について説明する．最後に4節で提案アルゴリズムを計算機上で実現しシミュレーション実験を行なった結果について述べ，考察を行なう．

2 準備

2.1 MCMレイアウトモデル

本稿では手法の記述を簡単にするために，組合せ回路とレジスタのみから構成されるシステムを考え，クロック発生・分配回路は無視する．入力として与えられたシステムを，回路素子を節点 $N=\{n_1, n_2, \cdots, n_I\}$，ネットを枝 E とするグラフ $G=(N,E)$ で表し，G をシステムグラフと呼ぶ．また，チップ集合を $C=\{c_1, c_2, \cdots, c_J\}$ としたとき，$M=(G,C)$ をMCMシステムと呼ぶ．MCM基板上のチップの位置，面積 Ac_j $(1 \leq j \leq J)$，I/Oピン数 IOc_j $(1 \leq j \leq J)$，及びチップ $c_p, c_q \in C$ 間の配線遅延 Dc_{pq} $(1 \leq p, q \leq J)$ はあらかじめ与えられているものとする．また各回路素子の面積 n_i $(1 \leq i \leq I)$，及び回路素子間の最大許容遅延遅延 Dn_{lm} $(1 \leq l, m \leq I)$ も与えられるものとする．多端子ネットは2端子分解されているものと仮定する．ノード $n_l, n_m \in N$ 間の結合度 W_{lm} はノード間の重み（配線数）を表す．本稿では，チップの面積は全て同一であると仮定する．

2.2 配線遅延モデル

遅延制約は，図2で示すように，与えられた回路の任意のフリップ・フロップ (FF) 間，またはI/OパッドとFF間の最大許容遅延 D_{ff} によって与えられる．つまり，その間の全てのパスの遅延時間が D_{ff} 以下であれば遅延制約は満たされたことになる．そして，文献[4]のゼロスラックアルゴリズムを適用し，各ノード $n_l, n_m \in N$ 間の最大許容遅延 Dn_{lm} $(1 \leq l, m \leq I)$ を求める．

回路素子をノード，信号の流れを有向枝で表したグラフを制約グラフと呼ぶ．図2の回路に対し，最大許容遅延 D_{ff} を 38 に設定し，ゼロスラックアルゴリズムを適用して求めた結果を制約グラフで表すと，図3のようになる．ここで，各ノードに附属している数値はノードの内部遅延を示す．ノード間において，信号が出力として出ていくノードのことをソースノード，信号が入力として入ってくるノードのことをシンクノードという．例えば，ノード n_1, n_2 間においてソースノードは n_1，シンクノードは n_2 である．

簡単のため，配線遅延は距離に比例すると仮定すると，ネット net_i の遅延時間 d_i の近似式は，

$$d_i(L) = Ro * (c * L + \sum_j C_{ij}) \qquad (1)$$

で表すことができる [6]．ここで，

Ro : 出力等価抵抗
c : 単位長当りの配線容量
L : ネット net_i の配線長（半周近似）
$\sum_j C_{ij}$: ロード容量の総和

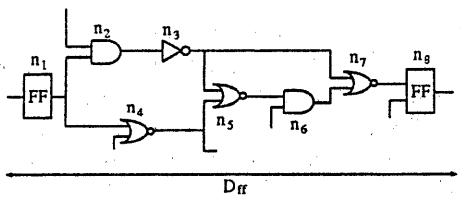


図 2 タイミング制約

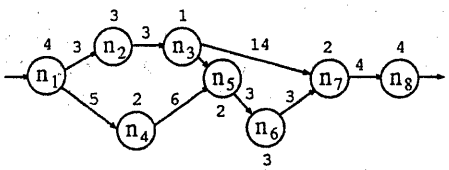


図 3 制約グラフ

2.3 MCM システム分割問題の定式化

本稿で取り扱う MCM 分割問題は，チップ間の配線遅延，チップ面積，及びチップの I/O ピン制約を満たし，かつ，カット数が最小となる分割を求めることである．以下に，本稿で扱う分割問題の定式化を示す．

[制約を考慮した分割問題]

[入力] MCM システム $M = (G, C)$
[出力] 各ノードのチップ割り当て $F : N \rightarrow C$
[制約] 1) 配線遅延制約
2) チップの I/O ピン制約
3) チップ面積制約
[目的関数] カット数

図 4は，システムグラフが与えられたとき，全ての制約を満足しながらカット数が最小になるようにノードをチップに割り当てた例である．ここで，チップ面積は全て 3，チップの最大 I/O ピン数は 4 である．また，ノード面積は n_1, n_4 が 2 で，それ以外のノードは 1 である．ノード間の最大許容遅延は，1 $unit\_delay$ とした．ここで，$unit\_delay = minDc_{pq}$ $(1 \leq p, q \leq J)$ とは，隣あったチップ間の配線遅延のことである．そのため，チップ 1, 3 間とチップ 2, 4 間の配線遅延は 2 $unit\_delay$ となる．それ以外のチップ間は全て 1 $unit\_delay$ である．OE_{ij} をノード n_i がチップ c_j に割り当てられたときチップ c_j から出ていくネット数とすると，カット数は $\frac{1}{2}\sum_{i=1}^{I}\sum_{j=1}^{J} OE_{ij}$ で表すことができる．この場合のカット数は 6 である．

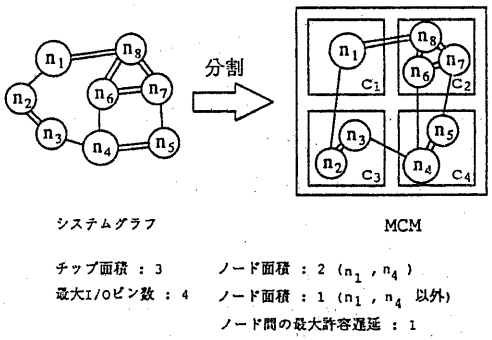


図 4 制約を考慮したシステム分割

3 MCM システム分割手法の概要

提案手法は大きく分けて 4 つのフェーズからなる．本稿では，全回路素子数が 10^6 程度で，複数のチップで実現されるような大規模システムを取り扱うため，実用的な時間内で解を得るために，最初に，制約を考慮した 2 種類のクラスタリング（フェーズ 1,2）を行なうことにより，取り扱うノード数を減少させる．次に，フェーズ 3 で総配線長の最小化を目的とした初期分割を行ない，初期解を得る．このとき，制約違反が生じている場合があるため，フェーズ 4 では部分回路に対する回路分割問題を 0-1 整数計画問題に定式化し，線形計画法を用いて解くことで，チップ間の遅延，面積，I/O ピン制約を満たしながら，カット数が最小となるようにノードの移動を行な

う．この操作を解の改善があるかぎり繰り返す．以下では，各フェーズの詳細について述べる．

3.1 フェーズ1：単位遅延に基づくクラスタリング

MCMの分割問題では大規模なシステムを取り扱うことになる．そのため，後で述べる0-1整数計画法に基づく分割手法を適用した場合，ノード数が増加すると実用的な計算時間内で解が求まらないことが予想される．ノード数を減少させるための有効な手法としてクラスタリングがあるが，これまでに知られているクラスタリング手法[2, 3]をそのままMCMのクラスタリングに適用することは望ましくない．その理由として，それらの手法はチップ間の配線遅延，チップ面積，及びチップのI/Oピン制約を考慮していないからである．そこで，これらの制約を考慮したクラスタリング手法を提案する．まず，フェーズ1では，異なったチップに割り当てられると必ず遅延制約を違反してしまうノード同士をクラスタリングする[7]．すなわち，$Dn_{lm} < unit\_delay$ となるノード同士が異なったチップに割り当てられてしまうと，必ず制約違反を起こす．そこで，$Dn_{lm} < unit\_delay$ となる全てのノード同士を文献[7]の手法を用いてクラスタリングする．この操作を全てのノード間の最大許容遅延が $unit\_delay$ 以上となるまで繰り返す．これにより，接続のあるノード間には必ず $unit\_delay$ 以上の遅延が許されることになる．もし，フェーズ1においてクラスタノードの面積が制約として与えられたチップ面積より大きくなった場合，許容解は存在しないことになる．

3.2 フェーズ2：制約に基づくクラスタリング

フェーズ3で述べる0-1整数計画法に基づく分割を行なう時に，大規模データに対しては，そのままでは実用的な時間内で解が求まらない場合がある．そこで，我々は，チップ間の遅延，チップ面積，及びチップのI/Oピン制約を考慮しつつ問題のサイズを小さくする新しいクラスタリング手法を提案する．

提案手法は遅延が最もクリティカルで，ノード面積が小さく，結合度の大きなノード同士をクラスタリングする．ここで，遅延が最もクリティカルであるとは，$Dn_{lm} - unit\_delay$ の差が最小な場合のことを言う．この段階では，実際にノードをチップに割り当てたときのチップのI/Oピン数は分からないため，直接的にチップのI/Oピン制約を考慮することはできないが，結合度の大きなノード同士をクラスタリングするとI/Oピン制約を満たし易くなるので，ノード間の結合度を考慮することによって，間接的ではあるがI/Oピン制約を考慮することが可能となる．

任意のノード対 $n_l, n_m \in N$ が与えられたとき、コスト関数 $CF1(l,m)$ を以下のように定義する．

$$CF1(l,m) = \alpha \times W_{lm} + \beta \times (Ave\_chip - (An_l + An_m)) + \frac{1}{Dn_{lm} - unit\_delay}$$

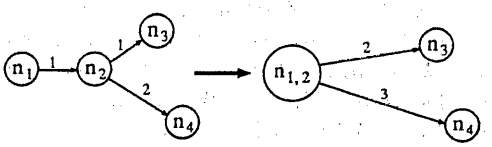


図5 クラスタリングによる最大許容遅延の変更

ここで，α, β はパラメータ．また，フェーズ1より $Dn_{lm} - unit\_delay > 0$ である．このコスト関数により，ノード n_l, n_m 間の3つの制約に対する度合が求まり，$CF1(l,m)$ の大きい順にノード n_l, n_m のクラスタリングを行う．この操作をクラスタノード数がある定数以下となるまで繰り返す．アルゴリズムの概略を以下に示す．

[フェーズ2：制約に基づくクラスタリング]

ステップ2.1：任意のノード対 n_l, n_m のコスト $CF1(l,m)$ を計算

ステップ2.2：制約を満たし，最大のコスト $CF1(l,m)$ を持つノード対 n_l, n_m をクラスタリング

ステップ2.3：ノード数がある定数以上のとき，ステップ2.1に戻る

ステップ2.4：終了

フェーズ1，2において，クラスタリングしたノード n_l, n_m は必ず同じチップに割り当てられるため，n_l, n_m 間の遅延はそれほど大きくない．そこで制約として与えられた最大許容遅延を有効に使うために，その後に続くノード間の最大許容遅延を変更する．このようにノード間の最大許容遅延を変更しても，フリップ・フロップ間の最大許容遅延 D_{ff} は満足する．図5は，ノード n_1 とノード n_2 がクラスタリングされたときの最大許容遅延の変更を示す．クラスタノード $n_{1,2}$ とノード n_3 及びノード n_4 間の最大許容遅延がノード n_1 とノード n_2 間の最大許容遅延分だけ増加し，遅延制約が緩和された．このため，ある程度クラスタリングをした方が，遅延制約が緩和され，解が求まり易くなると考えられる．

3.3 フェーズ3：初期分割

フェーズ3では，フェーズ4の0-1整数計画法に基づく分割改良のための初期解を生成する．ここでは，0-1整数計画法により，初期解を求めるが，一般に，I/Oピン制約，及びカット数は2次式となるため制約は配線遅延とチップ面積のみを考慮し，目的関数はチップ間の配線長の総和とする．そのためフェーズ3において得られた初期解には，制約違反が生じる場合がある．この制約違反は，以降のフェーズ4の改良フェーズで取り除く．

[0-1整数計画法に基づく初期分割問題]

$$\min \text{カット数} \quad \sum_{l=1}^{I'} \sum_{m=1}^{I'} Dc_{pq}(x_{lp} + x_{mq})$$

s.t. 1) 配線遅延

$$Dc_{pq}(x_{lp} + x_{mq} - 1) \leq Dn_{lm}$$

2) チップ面積

$$\sum_{i=1}^{I'} An_i x_{ij} \leq Ac_j \quad (1 \leq j \leq J)$$

3) チップ割り当て

$$\sum_{j=1}^{J} x_{ij} = 1 \quad (1 \leq i \leq I')$$

ここで，$x_{ij} = \begin{cases} 1 & n_i \text{を} c_j \text{に割り当てる} \\ 0 & otherwise \end{cases}$

実際の遅延制約は，$x_{lp} = x_{mq} = 1$ ときのみの制約を満足していればよいが，それ以外の時は必ず制約を満たすので，ここで示した遅延制約は実際の遅延制約をも必ず満足する．また，ノードは必ず1つのチップに割り当てられるので，$\sum_{j=1}^{J} x_{ij} = 1$ となる．

3.4 フェーズ4：0-1 整数計画法に基づく分割改良

フェーズ3で得られた初期分割に基づいて OE_{ij} を計算する．ただし，初期分割では制約違反が存在する．接続のあるノード同士が一度に移動する場合はカット数を線形式で表すことができないため，フェーズ4ではノード集合の極大独立点集合を求め，求めた集合に対して0-1整数計画法を適用する．解の改善があるかぎり0-1整数計画法を繰り返す．

しかし，0-1整数計画法をそのまま適用すると，解の収束性が悪く，実用的な時間内で解が求まらない場合がある．そのため，0-1整数計画問題を線形計画問題に変換する．そして，線形計画法を適用して解を求め，求まった解のうち整数解の変数に対してはその変数を定数として再度線形計画法を適用する．この操作を全ての変数が整数解となるまで繰り返す (図6参照)．ここで，解を収束させるために，線形計画法によって求めた解に対し，必ず1つの変数 x_{ij} は整数解にする．また，収束率を向上させるために1%の誤差以内ならば，その変数は整数解とする．ただし，小数解を整数解に変更するとき，全ての制約を満たすものだけを変更する．しかし，そのときに全ての制約を満たすものが1つもない場合は，制約違反の最も小さいものを変更する．そのため，このとき求めた解においては制約違反が生じる場合がある．

3.4.1 極大独立点集合の抽出

カット数は，1つのノードが移動する場合は線形式で表すことができるが，複数のノードが同時に移動する場合は線形式で表すことができず，必ず2次式となる．2次計画法を適用すれば解を求めることができるが，小規模な問題しか取り扱えないので，MCMのような大規模回路に対しては適用できない．そこで，カット数を線形式で表すため，まず，極大独立点集合を求め，その集合に含まれるノードに対し，0-1整数計画法を適用する．極大独立点集合に属するノードだけが移動する場合，互いに接続がないため複数のノードが同時に移動しても，正確にカット数を表現することができる．

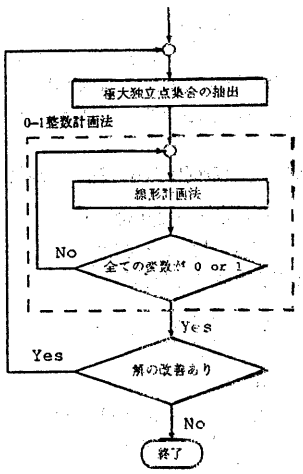


図 6 フェーズ4

フェーズ2までによって求まったクラスタノードの集合を CN とする．以後，クラスタノードのことを単にノードと呼ぶ．1度にカットが改善されやすくするために，なるべく多くのノードを極大独立点集合に含め，また，同じノードが続けて選択されることを避けるために，ノード $n_i \in CN$ に対するコスト関数 $CF2(n_i)$ を以下のように定義する．

$$CF2(n_i) = degree(n_i) + \gamma \times count(n_i)$$

ここで，$degree(n_i)$ はノード n_i $(n_i \in CN)$ の次数，$count(n_i)$ はノード n_i $(n_i \in CN)$ が今までに極大独立点集合に含まれた回数，γ はパラメータである．提案手法では，求めたコスト関数の小さい順にノードをソートし，

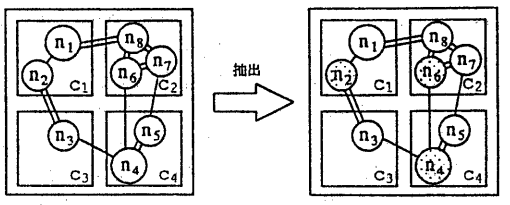


図 7 極大独立点集合の抽出

その順番に従って極大独立点集合に含められるか調べ，順次加えていく．実験では，一度選ばれたノードは連続して選ばれにくくするために，γをノードの平均次数に設定した．表7に極大独立点集合の抽出状況の例を示す．ノード n_2, n_4, n_6 が極大独立点集合に選ばれた．

3.4.2 制約付き分割問題の0-1整数計画問題への定式化

フェーズ3で選ばれた極大独立点集合に含まれるノードに対し，0-1整数計画法を適用する．

[0-1整数計画法に基づく分割問題]

$$\min \text{総配線長} \quad \sum_{i=1}^{I'}\sum_{j=1}^{J} OE_{ij}x_{ij}$$

s.t. 1) 配線遅延

$$Dc_{pq}(x_{lp}+x_{mq}-1) \le Dn_{lm}$$

2) チップのI/Oピン数

$$\sum_{i=1}^{I'} OE_{ij}x_{ij} \le IOc_j \quad (1 \le j \le J)$$

3) チップ面積

$$\sum_{i=1}^{I'} An_i \le Ac_j \quad (1 \le j \le J)$$

4) チップ割り当て

$$\sum_{j=1}^{J} x_{ij} = 1 \quad (1 \le i \le I')$$

ここで，$x_{ij} = \begin{cases} 1 & n_i \text{を} c_j \text{に割り当てる} \\ 0 & otherwise \end{cases}$

3.4.3 遅延制約式の簡略化

0-1整数計画法において遅延制約は，

$$Dc_{pq}(x_{lp}+x_{mq}-1) \le Dn_{lm} \quad (1 \le l,m \le I) \quad (1 \le p,q \le J)$$

で表される．ここで，Dn_{lm}はノード n_l, n_m 間の最大許容遅延，Dc_{pq}はチップ c_p, c_q 間の配線遅延を表す．

しかし，この制約式はノードとチップのすべての組合せを必要とするので，最悪の場合，遅延の制約式は $I'^2 \times J^2$ (ここで，I'はノード数，Jはチップ数) となってしまう．そのため，仮にフェーズ2でクラスタノード数を100に設定しても8分割の場合，遅延の制約式は640000も必要となり，実用的な計算時間内で解を求めることは非常に困難である．しかしながらこの制約式は冗長なので，以下のようにして制約式を減らすことが可能である．

例えば，図8(a)のように，ノード n_1, n_2 間，n_1, n_3 間に遅延制約があった場合 (n_1, n_2 間の最大許容遅延は1.5 *unit_delay*, n_1, n_3 間の最大許容遅延は1.7 *unit_delay*)，4分割においては32通りの制約式が考えられる．ここで，$x_{lp} = x_{mq} = 1$ のとき，$Dc_{pq}(x_{lp}+x_{mq}-1) \le Dn_{lm}$ の制約を満足するような制約式は，x_{lp}, x_{mq} がどんな値 (0 or 1) をとっても必ずその制約式は満足する．そのため，遅延の制約式から除くことができる．そこで，このような制約式を取り除くと，遅延の制約式は8通りとなる (図8(b))．この8つの式は $x_{lp} = x_{mq} = 1$ のとき，遅延制約違反を生じて，これらの式を図8(c)のように変換することができる．ソースノード $n_l \in CN$ がチップ c_p に割当てられたとき，各シンクノード n_m $(1 \le m \le M)$ において，遅延制約違反を生じるチップ割当の組合せの数が P_m $(1 \le m \le M)$ 個あると仮定する．すると，全てのシンクノード n_m $(1 \le m \le M)$ の遅延制約違反を生じるチップ割当の組合せの数 P は $\sum_{m=1}^{M}\sum_{i=1}^{P_m}$ となる．それらの制約式を1つにまとめた式

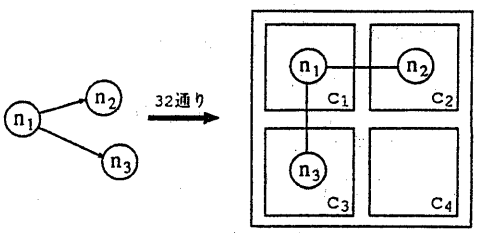
$$P * x_{lp} + \sum_{m=1}^{M}\sum_{i=1}^{P_m} x_{mq_i} \le P$$

は元の制約式を満足するため，図8(d)のように制約式を変えることができる．図8(d)では，$P = 2$ となる．

これより，以下のことが言える．

定理1

ソースノード n_l がチップ c_p に割当てられたとき遅延制約違反を生じるシンクノード n_m のチップ割当の全ての組合せの数を P 個とする．このとき，これらの制約式を1



(a)

$$\begin{aligned}
2(x_{11}+x_{24}-1) &\le 1.5 \\
2(x_{11}+x_{34}-1) &\le 1.7 \\
2(x_{12}+x_{23}-1) &\le 1.5 \\
2(x_{12}+x_{33}-1) &\le 1.7 \\
2(x_{13}+x_{22}-1) &\le 1.5 \\
2(x_{13}+x_{32}-1) &\le 1.7 \\
2(x_{14}+x_{21}-1) &\le 1.5 \\
2(x_{14}+x_{31}-1) &\le 1.7
\end{aligned}$$

(b) 8通り

$$\begin{aligned}
x_{11}+x_{24} &\le 1 \\
x_{11}+x_{34} &\le 1 \\
x_{12}+x_{23} &\le 1 \\
x_{12}+x_{33} &\le 1 \\
x_{13}+x_{22} &\le 1 \\
x_{13}+x_{32} &\le 1 \\
x_{14}+x_{21} &\le 1 \\
x_{14}+x_{31} &\le 1
\end{aligned}$$

(c) 8通り

$$\begin{aligned}
2x_{11}+x_{24}+x_{34} &\le 2 \\
2x_{12}+x_{23}+x_{33} &\le 2 \\
2x_{13}+x_{22}+x_{32} &\le 2 \\
2x_{14}+x_{21}+x_{31} &\le 2
\end{aligned}$$

(d) 4通り

図8 遅延制約式の簡略化

つにまとめた制約式を

$$P * x_{lp} + \sum_{m=1}^{M} \sum_{i=1}^{P_m} x_{mq_i} \leq P \qquad (2)$$

とすると，この制約式は，元の制約式

$$Dc_{pq}(x_{lp} + x_{mq} - 1) \leq Dn_{lm} \qquad (3)$$

が満足するときは，すべて満足し，元の制約式が満足しないときは，必ず違反する．

（証明）

(2) 式において，$Dc_{pq} \leq Dn_{lm}$ $(\forall_{F1(n_l)=c_p, F1(n_m)=c_q})$ ならば，x_{lp}, x_{mq} がどんな値 (x_{lp}, x_{mq} は 0 or 1) を取ろうとも常に (2) 式を満足する．そのため，これらの制約式は取り除いても元の制約式は満足する．そこで，$Dc_{pq} > Dn_{lm}$ のときのみ考える．

このとき，(2) 式を満足しない場合は，$x_{lp} = x_{mq} = 1$ の場合のみである（それ以外のときは (2) 式を満足する）．そのため，$x_{lp} + x_{mq} \leq 1$ となる制約式を考えたとき，(2) 式の条件式を満足する（x_{lp}, x_{mq} は 0 or 1）．

今，ソースノード n_l がチップ c_p に割り当てられたとき各シンクノード n_m $(1 \leq m \leq M)$ において，$Dc_{pq} > Dn_{lm}$ となるチップ割当の組合せの数が P_m $(1 \leq m \leq M)$ 個あると仮定する．すると，全てのシンクノード n_m $(1 \leq m \leq M)$ の $Dc_{pq} > Dn_{lm}$ となるチップ割当の組合せの数 P は $\sum_{m=1}^{M} \sum_{i=1}^{P_m}$ となる．これらの制約式は互いに独立なため，1 つにまとめることができる．つまり，$Dc_{pq} > Dn_{lm}$ となる制約式

$$x_{lp} + x_{m_1 q_1} \leq 1$$
$$x_{lp} + x_{m_2 q_2} \leq 1$$
$$\vdots$$
$$x_{lp} + x_{m_P q_P} \leq 1$$

は

$$P * x_{lp} + \sum_{m=1}^{M} \sum_{i=1}^{P_m} x_{mq_i} \leq P$$

と書き直すことができる．

逆に，もし，(2) 式において制約式を満たさないとすると，$Dc_{pq} > Dn_{lm}$ かつ $x_{lp} = x_{mq} = 1$ である ($Dc_{pq} > 0$ かつ $Dn_{lm} > 0$)．このとき，(1) 式の左辺 $\geq P + 1$，よって，左辺 > 右辺となり，制約式を満たさない． □

定理 2

遅延制約の制約式はたかだか $I' \times J$ (I' はノード数，J はチップ数) で表すことができる．

（証明）

定理 1 より，(1) 式は，ソースノード n_l がチップ c_p に割り当てられたときの遅延制約の制約式を表す．よって，遅延制約の制約式はたかだか $I' \times J$ である． □

4 シミュレーション実験

提案手法を SPARC Server 1000 (135MIPS) 上で C 言語を用いて実現し，シミュレーション実験を行なった．データは MCNC ベンチマークデータを採用した．表 1 に実験に用いたデータの特徴と各制約を示す．比較手法として文献 [12] の手法を用いた．これはチップ面積制約と I/O ピン制約を考慮した分割手法である．I/O ピン制約は文献 [12] で報告されている最大 I/O ピン数を上限とした．また、チップ面積制約はノード面積の総和をチップ数で割った値を 1.1 倍したものを上限とした．比較手法の結果は，文献 [12] で報告されている結果をそのまま引用した．文献 [12] において，チップ面積，タイミング違反数，及び計算時間は記述されていなかった．表 2 に比較手法の結果を示す．提案手法において，フェーズ 2 におけるクラスタノード数は，データ primGA1 のときは 300，データ primGA2 のときは 1000 とした．また、フェーズ 4 では，10 回続けて改善がみられない場合，終了することにした．

表 1 実験データ

No.	データ名	*#nodes*	*#nets*	IOc_j	Ac_j
1	primGA1	833	904	90	3.85
2	primGA2	3014	3029	335	7.19

No. : データ番号 IOc_j : I/O ピン制約
#nodes : ノード数 Ac_j : 面積制約 $[mm^2]$
#nets : ネット数

表 2 文献 [12]

No.	#cuts	#I/O	area	#vio	time(s)
1	334	90	–	–	–
2	992	335	–	–	–

表 3 に制約としてチップ面積と I/O ピン制約のみを考慮して 8 分割を行なったときのフェーズ 3 終了時点の提案手法の結果を示す．文献 [12] との比較するとカット数が平均で約 74.8%増加している．チップ面積は満足しているが，I/O ピン制約は満足していない．表 4 に表 3 と同じ制約におけるフェーズ 4 終了時点の提案手法の結果を示す．文献 [12] との比較するとカット数が平均で約 20.9%改善された．フェーズ 3 との比較するとカット数が平均で約 53.7%改善された．また，チップ面積と I/O ピン制約は満足するようになった．遅延制約違反も減少した．表 5 に制約として配線遅延，チップ面積，I/O ピン制約を考慮して 8 分割を行なったときのフェーズ 3 終了時点の提案手法結果を示す．表 3 と比較すると，カット数が平均で約 107.0%増加する結果となった．しかし，表 3 では遅延制約を考慮していないため，遅延制約違反が生じているが，表 5 は遅延制約は満足している．しかし，面積制約，I/O ピン制約はともに違反している．表 6 に表 5 と同じ制

表 3 フェーズ 3（制約：チップ面積，I/O ピン）

No.	#cuts	#I/O	area	#vio	time(s)
1	482	267	3.84	28	4
2	2036	804	7.19	197	42

表 4 フェーズ 4（制約：チップ面積，I/O ピン）

No.	#cuts	#I/O	area	#vio	time(s)
1	253	85	3.83	18	309
2	818	325	7.19	120	2046

表 5 フェーズ 3（制約：遅延，チップ面積，I/O ピン）

No.	#cuts	#I/O	area	#vio	time(s)
1	632	289	7.94	0	6
2	2230	889	8.85	0	76

表 6 フェーズ 4（制約：遅延，チップ面積，I/O ピン）

No.	#cuts	#I/O	area	#vio	time(s)
1	334	87	3.82	0	296
2	1066	321	7.19	0	2794

No.：データ番号　area：チップ面積 $[mm^2]$
#cuts：カット数　#vio：タイミング違反数
#I/O：I/O ピン数

約におけるフェーズ 4 終了時点の提案手法の結果を示す．フェーズ 3 との比較するとカット数が平均で約 49.7%改善された．表 4 と比較すると，カット数が平均で約 7.5%増加する結果となった．しかし，すべての制約を満足する結果が得られた．全体的に，フェーズ 4 でかなり時間がかかっているのが分かる．フェーズ 4 において，1 回のループはフェーズ 3 より短いが，ループ回数が多いため，結果的に時間がかかる．しかし，計算時間はいずれも実用的な時間内であるといえる．

5 あとがき

チップ間の配線遅延，チップ面積，及びチップの I/O ピン制約を考慮した MCM 分割手法を提案した．0-1 整数計画法を適用することにより，制約付き分割問題を解くことができた．また，実用的な時間内で解を求めるために，制約に基づくクラスタリング手法を提案した．チップ面積，及びチップの I/O ピン制約を考慮した場合，実験結果より，平均で約 20.9%ほど比較手法よりカット数が改善された．今後の課題としては，大規模データに対する実験とその評価が挙げられる．

謝辞

本研究を行うにあたり，プログラムの作成にご協力頂いた本学学部生大平祥広君に感謝致します．本研究の一部は文部省科学研究費補助金試験研究 (B)(2)（課題番号 06558042）による．

参考文献

[1] C. M. Fiduccia and R. M. Mattheyses: "A linear time heuristic for improving network partitions," Proc. of 19th Design Automation Conference, pp. 175–181 (1982).
[2] L. Hagen and A. B. Kahng: "Fast spectral methods for ratio cut partitioning and clustering," Proc. of IEEE International Conference on Comput.-Aided Des., pp. 10–13 (1991).
[3] L. Hagen and A. B. Kahng: "A new approach to effective circuit clustering," Proc. of IEEE International Conference on Comput.-Aided Des., pp. 422–426 (1992).
[4] P. S. Hauge, R. Nair and E. J. Yoffa: "Circuit placement for predictable performance," Proc. of IEEE International Conference on Comput.-Aided Des., pp. 88–91 (1987).
[5] B. W. Kernighan and S. Lin: "An efficient heuristic procedure for partitioning graphs," The Bell System Technical Journal, 49, pp. 291–307 (1970).
[6] E. S. Kuh and M. Shih: "Recent advances in timing-driven physical design," Proc. of Asia-Pacific Conference on Circuits and Systems, pp. 23–28 (1992).
[7] E. S. Kuh, M. Shih and R.-S. Tsay: "Performance-driven system partitioning on multi-chip modules," Proc. of 29th Design Automation Conference, pp. 53–56 (1992).
[8] E. S. Kuh, M. Shih and R.-S. Tsay: "Integer programming techniques for multi-way system partitioning under timing and capacity constraints," Proc. European Conference on Design Automation with the European Event in ASIC Design, pp. 294–298 (1993).
[9] E. S. Kuh, M. Shih and R.-S. Tsay: "Timing-driven system partitioning by constraints decoupling method," Proc. IEEE Multi-Chip Module Conference, pp. 164–169 (1993).
[10] K. Roy and C. Sechen: "A timing driven n-way chip and multi-chip partitioner," Proc. of IEEE International Conference on Comput.-Aided Des., pp. 240–247 (1993).
[11] M. Shih and E. S. Kuh: "Quadratic programming for performance-driven system partitioning," Proc. of 30th Design Automation Conference, pp. 761–765 (1993).
[12] M. Shih and E. S. Kuh: "Circuit partitioning under capacity and I/O constraints," Proc. of IEEE Custom Integrated Circuits Conference, pp. 659–662 (1994).
[13] M. Shih, E. S. Kuh and R.-S. Tsay: "System partitioning for multi-chip modules under timing and capacity constraints," Proc. IEEE Multi-Chip Module Conference, pp. 123–126 (1992).
[14] Y.-C. Wei and C.-K. Cheng: "Towards efficient hierarchical designs by ratio cut partitioning," Proc. of IEEE International Conference on Comput.-Aided Des., pp. 298–301 (1989).
[15] Y.-C. Wei and C.-K. Cheng: "Ratio cut partition for hierarchical designs," Proc. of IEEE International Conference on Comput.-Aided Des., 10, 7, pp. 911–921 (1991).
[16] N.-S. Woo and J. Kim: "An efficient method of partitioning circuits for multiple-FPGA implementation," Proc. of 30th Design Automation Conference, pp. 202–207 (1993).