マルチチップモジュールに対するシステム分割の一手法†

5L－2

桂 嘉志記　小出 哲士　若林 真一　吉田 典可
広島大学 工学部

1 まえがき

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．これは，基盤上にベアチップを直接マウントするため，従来の実装技術であるプリント基板(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，高密度な配置となるため，MCM設計においては，消費電力，及び熱を新たに考慮する必要があり，また，チップ間の配線遅延がチップ内の配線遅延に比べ，非常に大きいので，チップ間の配線遅延を考慮する必要がある．

MCM設計においては，与えられた回路をチップに分割する必要がある．上述の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[1, 2]や，I/Oピン制約と面積制約のみを考慮したもの[3]があるが，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手法は知られていない．そこで，本稿ではこれらの制約を同時に考慮したMCM分割手法を提案する．

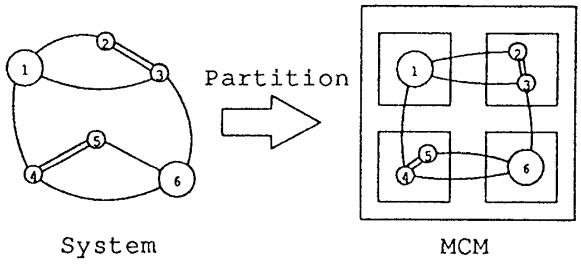
2 準備

簡単のため，組合せ回路とレジスタのみからなるシステムを考え，クロック周辺回路は無視する．また，ネットはあらかじめ2端子分解されているものと仮定する．MCM基板上のチップの面積は全て同一で，その位置，面積，I/Oピン数，及びチップ間の遅延は与えられているものとする，また各回路素子の面積，及び回路間の最大許容遅延も与えられる．本稿では，入力として与えられたシステムを，回路素子を節点V，ネットを枝Eとするグラフ$G=(V,E)$で表し，このグラフをシステムグラフと呼ぶ．本稿で取り扱うMCM分割問題は，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(図1)．

3 MCMシステム分割手法の概要

提案手法では，複数のチップにまたがるような大規模なシステムを取り扱うため，実用的な時間内で解を求めるために，まずクラスタリングを適用して，ノード数を小さくする．そして，0-1整数計画法を適用して，チップ間の遅延，チップ面積，チップの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な分割を求める．提案手法の概要を以下に示す．

†"A System Partitioning Method for Multi-Chip Modules," by Yoshinori KATSURA, Tetsushi KOIDE, Shin'ichi WAKABAYASHI, and Noriyoshi YOSHIDA, Faculty of Engineering, Hiroshima University.



#Chip Area:2　#Node Area:2(node 1,6)
#I/O Pin:4　#Node Area:1(node 2-5)
#Max Delay:1

図1 制約を考慮したシステム分割

ステップ1　単位遅延に基づくクラスタリング
ステップ2　制約を考慮したクラスタリング
ステップ3　初期分割を行い，その解を保持
ステップ4　0-1整数計画法による分割
ステップ5　解が改良されれば解を更新し
　　ステップ4へ，そうでなければ終了．

4 制約を考慮したクラスタリング

これまでに知られているクラスタリング手法をそのまま本稿の問題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．提案手法は2つのステップからなる．ステップ1では，異なったチップに割り当てられると必ず遅延の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．ステップ2では，チップ間の遅延，チップ面積，及びチップのI/Oピン制約に対するノード間のクリティカル度を求め，それに基づくクラスタリングを行う．

4.1 ステップ1 (単位遅延に基づくクラスタリング)

$Dn_{lm} \leq unit\_delay$ となるノード同士をクラスタリングする[1]．ここで，$unit_delay$とは隣あったチップ間の遅延を表し，Dn_{lm}はノードn_l, n_m間の最大許容遅延を表す．この操作より，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．

4.2 ステップ2 (制約に基づくクラスタリング)

チップ間の遅延，チップ面積，及びチップのI/Oピン制約を考慮した新しいクラスタリング手法を提案する．本手法は遅延が最もクリティカルで，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，{ノード間の最大許容遅延 (Dn_{lm}) } - { 隣あったチップ間の遅延 $(unit\_delay)$} の差が最小な遅延のことである．また，ノードの平均面積を Ave_area，ノード n_l, n_m 間の結合度を a_{lm}，ノード n_l, n_m の面積を An_l, An_m で示すものとし，ノード集合 $N1 = \{n_j | 1 \leq j \leq J1\}$ が与えられたとき、コスト関数 $CF(l,m)$ を以下のように定義する．

$$CF(l,m) = \frac{a_{lm}(2Ave\_area - (An_l + An_m))}{Dn_{lm} - unit\_delay} \quad (1 \leq l, m \leq J1)$$

ここで，ステップ 1 より $Dn_{lm} - unit\_delay > 0$ である．このコスト関数により，ノード間の 3 つの制約に対するクリティカル度が求まり，クリティカル度の大きい順にクラスタリングを行う．一度クラスタリングを行うと，Ave_area が変化するので，コスト関数を再度計算する．この操作をノード数がある定数以下になるまで繰り返す．アルゴリズムの概略を以下に示す．

ステップ 2.1： ノード間のコスト関数を計算
ステップ 2.2： 最大のコスト関数を選び，そのノード同士をクラスタリング
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

ここで，ステップ 2.2 で選んだノード同士をクラスタリングしたとき，制約違反を起こした場合は，そのクラスタリングを解除し，次にコストの大きなノード同士を選択する．そして，以下同じことを繰り返す．

5 0-1 整数計画法を用いた制約付き分割手法

5.1 制約付き分割問題の 0-1 整数計画問題への定式化

$OuterEdge$ を，別のチップと接続している枝とすると，OE_{ij} は，ノード n_j がチップ i に割り当てられたときのノード n_j の $OuterEdge$ 数を表す．まずステップ 3 において初期分割を与え，それに基づいて 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と，カット数が正確に求まらないため，ステップ 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，整数計画法を適用する．

[入力]

1) チップの集合 $C = \{c_i | 1 \leq i \leq I\}$
2) チップの面積 $Ac_i \quad (c_i \in C)$
3) チップの最大 I/O ピン数 $IOc_i \quad (c_i \in C)$
4) チップ間の配線遅延 $Dc = \{Dc_{pq} | 1 \leq p, q \leq I\}$
5) クラスタノードの集合 $N2 = \{n_j | 1 \leq j \leq J2\}$
6) クラスタノードの面積 $An_j \quad (n_j \in N2)$
7) クラスタノード間の最大許容遅延
$Dn = \{Dn_{lm} | 1 \leq l, m \leq J2\}$
8) 初期分割 $F1 : N2 \rightarrow C$

[出力] 各ノードのチップへの割り当て $F2 : N2 \rightarrow C$

[目的関数] カット数 $\frac{1}{2}\sum_{i=1}^{I}\sum_{j=1}^{J2} OE_{ij}x_{ij} \rightarrow \min$

[制約] 1) ネットの遅延

$$\sum_{p=1}^{I}\sum_{q=1}^{I} Dc_{pq}(x_{pl} + x_{qm} - 1) \leq Dn_{lm} \quad \forall_{l,m}(1 \leq l, m \leq J2)$$

2) チップ面積 $\sum_{j=1}^{J2} An_j x_{ij} \leq Ac_i \quad (1 \leq i \leq I)$

3) チップの I/O ピン数 $\sum_{j=1}^{J2} OE_{ij}x_{ij} \leq IOc_i \quad (1 \leq i \leq I)$

4) チップ割り当て $\sum_{i=1}^{I} x_{ij} = 1 \quad (1 \leq j \leq J2)$

ここで，$x_{ij} = \begin{cases} 1 & F(n_j) = c_i \\ 0 & otherwise \end{cases}$

実際の遅延制約は，$x_{pl} = x_{qm} = 1$ 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，$\sum_{i=1}^{I} x_{ij} = 1$ となる．

5.2 0-1 整数計画法の概要

5.1 節で定式化した問題に 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．そこで，効率的に解を求めるために，0-1 整数計画問題を線形計画問題に変換して解を求め，求まった解のうち整数解に対しては，定数として再度線形計画法を適用する．この操作を全ての解が整数となるまで繰り返す．アルゴリズムの概要を以下に示す．

ステップ 4.1 線形計画問題を解く
ステップ 4.2 ステップ 4.1 での解 (x_{ij}) のうち，整数解 (0, 1) ならば，$x_{ij} = 0, 1$ にする
ステップ 4.3 すべての 解 (x_{ij}) が 0, 1 ならば終了
ステップ 4.4 ステップ 4.1 に戻る

ここで，収束させるために，ステップ 4.2 において必ず 1 つの変数 x_{ij} は整数解にする．

6 あとがき

今後の課題としては，提案手法に対する計算機によるシミュレーション実験と評価が挙げられる．

文献

[1] E. S. Kuh et al.: "Performance-driven system partitioning on multi-chip modules," Proc. 29th DAC, pp. 53–56 (1992).

[2] M. Shih and E. S. Kuh: "Quadratic programming for performance-driven system partitioning," Proc. 30th DAC, pp. 761–765 (1993).

[3] M. Shih and E. S. Kuh: "Circuit partitioning under capacity and I/O constraints," Proc. CICC, pp. 659–662 (1994).