大規模システムの効率的な階層木分割手法

徳本 守彦 （ローム (株)）

渡邊 孝博 （知能情報システム工学科）

An Efficient Hierarchical Tree Partitioning Method for VLSI Design

Morihiko Tokumoto （ROHM Co.）

Takahiro Watanabe （Department of Computer Science and Systems Engineering）

Recently, a circuit complexity of VLSIs, especially SOCs(Systm-on-a-chip), has been increased more and more due to the requirements of high performance and various functions, and their layout design has become a great difficult task. So that, circuit partitioning is indispensable to an efficient and superior system design, where the whole circuit is partitioned into sub-circuits of a reasonable size.
Circuit partitioning is reduced to a graph partitioning problem. But the problem is known as an NP-complete problem, even if two-way partitioning of a graph with unity node-size and edge-weight. So, we propose a hierarchical tree partitioning method, where two greedy algorithms are executed in some probability. Experimental results show that the proposed method can efficiently make a good circuit partition, and it is very useful for a VLSI design.

Key Words: *Circuit Partitioning, Graph partitioning, Greedy Algorithm, Hierarchical Tree Partitioning*

1 はじめに

携帯情報機器に見られるように，近年の情報機器は急速に発展，普及しており，更なる高機能化や高信頼化が望まれている．これらを実現するためにシステムオンチップと呼ばれるように集積回路の規模の増大と高密度化が進んでいる．その結果，システム設計における制約条件がより複雑なものとなり，開発期間の長期化が問題となっている．この問題を解決するために，システムを合理的なサイズのサブシステムに分割して設計を行う手法が提案されている．しかしながら，分割効率が悪い，最適なシステム分割には膨大な処理時間が必要，分割アルゴリズムのパラメータ設定が困難等，問題点が数多く残されている．

そこで本研究では，従来手法の内で最良の手法と同等の解品質を達成しつつ，それよりもはるかに高速な手法を提案する．さらに，従来手法でのパラメータ設定の困難さの解消を目指す．

提案する手法では，処理対象である回路システムをグラフでモデル化することでグラフ分割問題に帰着して解決する．しかしながら，グラフ分割問題は，グラフ中の全ノード重みと全枝重みが同一の値を持つグラフを単純に２分割するだけでも，ＮＰ困難問題[5]であることが知られている．そこで，近似解を求める手法が数多く提案されている．例えば，任意の初期分割の後，ある評価関数を設定して反復的に改善を加えていくトップダウン方式のＲＦＭ法[3]やボトムアップ方式のＧＦＭ法[3]がある．これらは貪欲戦略であるために最適解が得られる保証が無い．また，解が改善されなくなるまで繰り返し処理するために終了時間の把握ができず，処理の手続き上，適当な時間で打ち切られる．アルゴリズムを制御しているパラメータが経験に依存するという欠点もあった．別の提案として，グラフの接続状態に応じた評価値を算出し，それを枝重みとして用いることによって，大局的な分割を行う階層木分割手法[2]が提案されている．各グラフご

とに評価値を算出するために，ＲＦＭ法やＧＦＭ法よりも優れた結果が得られている．基本的には解の繰り返し改善も不要で，局所解などに陥いることもない．しかしながら，依然としてパラメータの設定が困難であった．そこで、我々は評価基準に単純なものを採用し，２種類の貪欲戦略を組み合わせる階層木分割手法を提案する．これによって、解品質と効率の点で、優れた分割を得ることができた．

2　準備

(1) 階層レイアウト設計とシステム分割

階層レイアウト設計[1)]のイメージを図 1に示す．最上位の階層となる回路システムをサブシステムに分けて，必要に応じてさらに細分して設計を行うことにより，各サブシステムごとに設計が行えるために，開発期間の短縮が行える．しかし，分割により，回路の品質が劣化しては意味が無い．そこで，品質を落さないように分割しなければならない．これがシステム分割問題[2)3)]である．

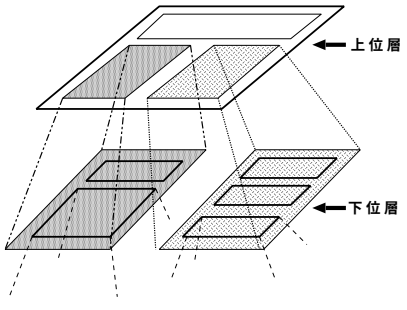

図 1: 階層レイアウト設計

表 1: 論理回路部品のサイズ比

セル名	サイズ比
入出力ピン	0.01
NOT	0.43
NAND	0.67
NOR	0.76
AND	1.10
OR	1.19
バッファ	0.86
XOR	2.87

(2) ノード重み

グラフ分割問題ではノードと枝に重みを与える．本論文ではノード重みは論理ゲートの面積値を採用するが，プロセス独立で議論するために，実サイズではなくレイアウトのサイズ比をノード重みとして用いることにする．文献 [4] に公開されているレイアウトから算出した各論理などのサイズ比を表 1に示す．なお，ＮＡＮＤ論理，ＮＯＲ論理，ＮＯＴ論理以外の論理は，この３論理の組み合わせで実現し，そのサイズ比もこれらを組み合わせたものとして算出してある．

(3) 入力データとグラフ変換

図 2に実験に使用する ISCAS85 ベンチマーク回路のｃ１７回路の記述フォーマット例を示す．また，ｃ１７回路の記述から直接再現した回路図を図 3に示す．

階層木分割手法を適用するために，図 3に示す回路図をグラフに変換する必要がある．入出力用ピンや論理素子などをノードに，配線を枝に対応させ，回路情報通りに，入力側から出力側へ向かう方向で変換を行った結果が図 4である．

```
#  c17 iscas example
# ----------------------------------------------
#    total number of lines in the netlist..... 17
#    simplistically reduced equivalent fault set size..... 22
#          lines from primary inputs gates..... 5
#          lines from primary outputs gates..... 2
#          lines from interior gate outupts..... 4
#          lines from **       3 ** fanout stems .... 6
#                  avg_fanin = 2.00 ,    max_fanin = 2
#                  avg_fanout = 2.00 ,    max_fanout = 2

    INPUT( G1gat )
    INPUT( G2gat )
    INPUT( G3gat )
    INPUT( G6gat )
    INPUT( G7gat )
    OUTPUT( G22gat )
    OUTPUT( G23gat )

    G10gat  =  nand ( G1gat , G3gat )
    G11gat  =  nand ( G3gat , G6gat )
    G16gat  =  nand ( G2gat , G11gat )
    G19gat  =  nand ( G11gat , G7gat )
    G22gat  =  nand ( G10gat , G16gat )
    G23gat  =  nand ( G16gat , G19gat )
```

図 2: ISCAS85 ベンチマーク回路記述フォーマット

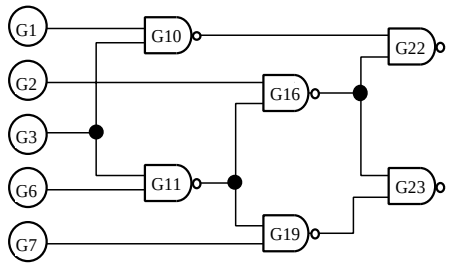

図 3: c17 回路

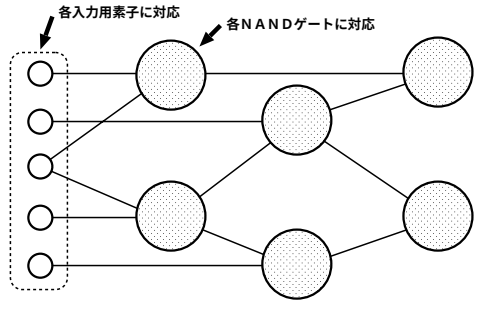

図 4: c17 回路を変換したグラフ

ここで，図3のＧ３とＧ１０，Ｇ１１を結ぶ配線では，

実際にはＧ１０とＧ１１の間に同一信号の配線が存在するが，図 4ではその配線をモデル化した枝がない．このような３個以上のノードを結ぶ多端子ネットをグラフに変換する場合は，その多端子ネットを完全グラフとして再現する．また，出力用ピンをモデル化したノードを追加する．以上を踏まえて，回路情報をグラフに変換すると図 5のようになる．

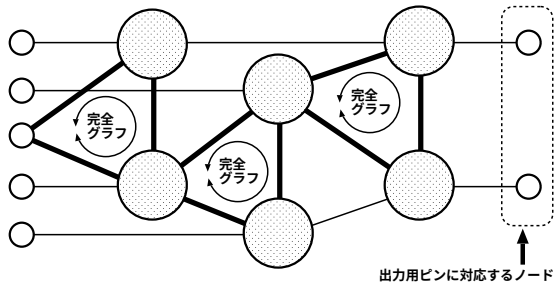

図 5: c17 回路を分割処理を考慮して変換したグラフ

3　グラフ分割アルゴリズム

(1) 評価基準 (カットエッジ)

グラフ分割問題において，分割部分が特定されると分割前のグラフからその部分を切り離していく．このとき，切断される枝をカットエッジ[2)]と呼ぶ．このカットエッジは分割部分間に跨っている枝であり，システム分割を行っている回路で考えるならば，それはサブシステム間の信号伝達を担う配線である．このようなサブシステム間の配線は，サブシステム内部の配線に比べ，必要となる配線領域が大きく配線長も長くなる．また，信号遅延や信頼性の点で望ましくない．従って，カットエッジが少なくなるように分割を行えば，信頼性を高めるばかりでなくレイアウト面積をも縮小が可能となる．それ故に，分割結果の良し悪しはカットエッジ数により評価できる．

図 6(ａ) において，グラフＧのノード集合Ｖ＝ $\{n_1, n_2, n_3, n_4, n_5, n_6\}$ からサブグラフＳのノード集合Ｖ$_S$＝ $\{n_1, n_4\}$ を分割する場合を説明する．

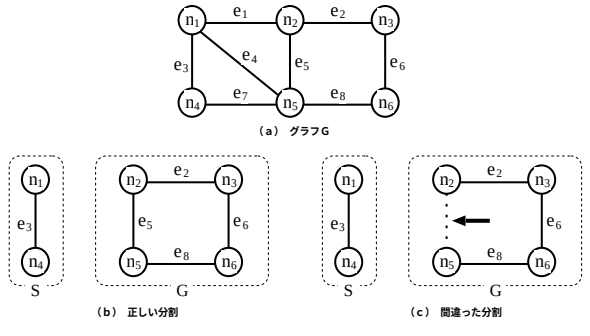

図 6: カットエッジ

枝集合 $\{e_1, e_4, e_7\}$ をグラフＧから取り除けば，サブグラフＳを分割できる (図 6(ｂ))．枝集合 $\{e_1, e_4, e_7, e_5\}$ を取り除く場合 (図 6(ｃ)) にも，結果的には分割されていることとなる．しかしながら，分割する際のカットエッジの集合はその分割を与える極小集合とする．

(2) 階層木分割手法

貪欲戦略の手法では，最適解が得られる可能性は低い．そこで，回路の接続状態に応じて適切な分割を行う階層木分割手法が提案されている．この手法では大局的な分割を実現するために，前処理として評価基準となる枝重みを算出する．この枝重みを分割処理で用いることにより，従来の貪欲戦略の手法に優ることが報告されている．ここでは準備として，この階層木分割手法の処理手順を説明する．

評価値算出処理及び分割処理の詳細は後述する．

階層木分割アルゴリズム

1. ［ 初期設定 ］
   - 希望する分割過程を階層木で表現する．
   - 各パラメータ値を決定する．
2. ［ 評価値算出処理 ］
3. ［ 分割処理 ］

(a) 評価値算出処理

この処理では，入力グラフの枝重みをある初期値に設定し，グラフの接続状態に応じて各枝重みを調整していく．この初期値を ε，調整のための増加分をペナルティ Δ と表す．これらは非常に小さな値であり，経験的に設定される．

評価値算出アルゴリズム

1. 各枝重み ε，ペナルティ Δ を設定する．
2. 以下を全てのノードを出発点として，全ノード間の最短路が確定するまで繰り返す．
   - 2.1. 出発点を任意に選択する．
   - 2.2. Dijkstra のアルゴリズムを実行し，最短路を求める．ただし，最短路に新たに１ノード付け加える度に，次頁の制約条件を満たすかどうかを調べる．
     - (true) Dijkstra のアルゴリズムを続行する．
     - (false) 最短路として確定している枝に対してのみ，その枝重みにペナルティ Δ を加えて，各枝重みを更新する．出発点は同一のノードとし，最短路情報をリセットし，ＳＴＥＰ２．２へ．

制約条件

$$\sum_{u \in \text{最短路}} (\text{出発点からノード}u\text{までの枝重み}) \times (\text{ノード}u\text{のノード重み}) \geqq \left(\begin{array}{l} \text{最短路として確定しているノードの重みの総和と} \\ \text{階層木構造から算出される計算値} \end{array} \right)$$

以上のようにして求められた枝重みは，ノード間のトポロジカルな広がりを示しており，その相対的な距離を表している．即ち，この重みが小さければそれほど近くに配置しなくてはならないことを表す．図 7に，このアルゴリズムのフローチャートを示す．

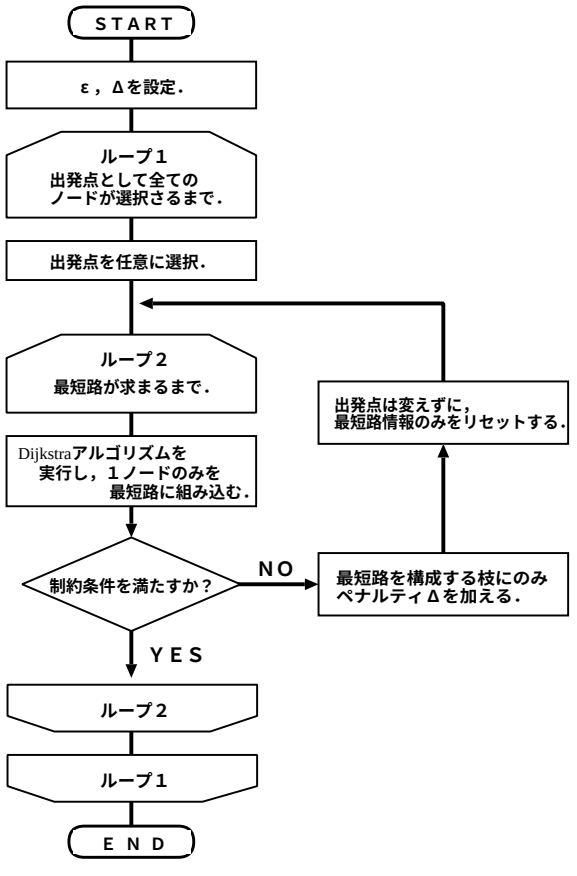

図 7: 評価値算出アルゴリズムのフローチャート

(b) 分割処理

分割処理ステップでは，前段階で算出した枝重みに関する局所的な最小木を構成することによって分割単位を特定していく．これを，階層木の構造に従って再帰的に処理を繰り返す．注意としてアルゴリズム中の「希望サイズ」とは，生成したい分割単位に含まれるノード重みの総和のことである．以下に処理手順を示す．また図 8に，このアルゴリズムのフローチャートを示す．

分割アルゴリズム

1. 入力されたグラフが高さ０の節点における希望サイズであるかどうか調べる．

(true) 入力されたグラフを高さ０の節点での分割単位と特定する．その後，１つ上階層の節点での分割処理へ戻る．

(false) 入力されたグラフのノード集合が空集合になるまで，以下を繰り返す．

f.1. 分割単位として確定していないノード集合内から，始点となるノードを任意に選択する．

f.2. １つ下の階層の節点での希望サイズに一致する局所的な最小木を構成する．

f.3. 生成した局所的な最小木を構成するノード集合を分割単位としてグラフから切り離す．

f.4. 特定した分割単位に再帰的にこの分割アルゴリズムを適用する．

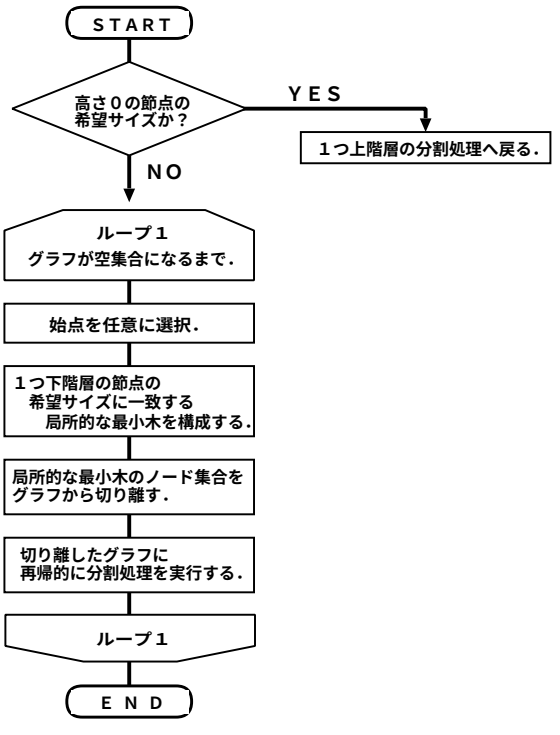

図 8: 分割アルゴリズムのフローチャート

(3) 提案する階層木分割手法

前述の階層木分割手法では，従来の貪欲戦略の手法よりもより良い分割結果が得られるとはいえ，評価値を算出するステップに膨大な処理時間を必要とした[6]．そこで処理時間の短縮化を図るために，本研究では評価値にノードの次数を用いる．これにより処理の単純化が可能である．しかしながら次数の評価方法が１通りでは，単なる貪欲戦略となってしまうために優れた結果が得られ

ることは期待できない．そこで２通りの評価方法を採用する．

まず２通りの評価方法に基づく基本処理について説明する．その後，新たな階層木分割手法を提案する．

(a) 基本処理１

基本処理１では，次数の高いノード同士を近くに配置することを目的とする．

基本処理１

1. 確定している分割単位に隣接するノードのうち，次数最大のノードを選択する．複数候補がある場合は，ノード重みが大きいものを選択する．
2. 選択したノードを分割単位Ｓに組み込む．

図 9に基本処理１の適用例を示す．図 9の状態で基本処理１の処理対象となるのは，ノードA，B，C，Dである．これらから次数最大のものを選択する．各ノードの次数は順に５，３，４，５である．ノードA，Dは同評価となる．このような場合は，さらにノード重みを比較し，大きなノードを選択する．従って，この場合はノードAが選択されることになる．

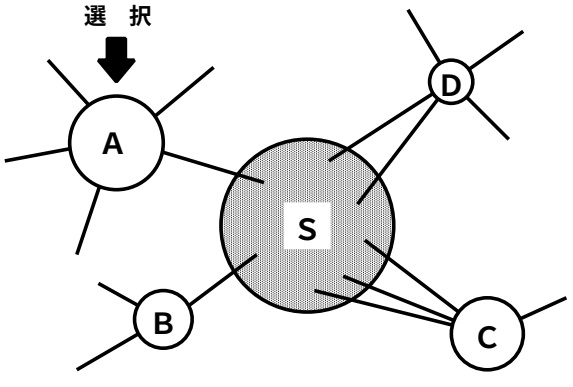

図 9: 基本処理１の適用例

(b) 基本処理２

基本処理２では，最終的に特定する分割単位のカットエッジ数を少なくすることを目的とする．これは同時に，次数の小さいノード同士を遠くに配置する役割を果たす．

基本処理２

1. 確定している分割単位に隣接するノードのうち，分割単位に組み込んだ結果のカットエッジ数が最も減少するノードを選択する．複数候補がある場合は，次数最大のものを選択する．さらに候補が複数存在するときは，ノード重みが小さいものを選択する．
2. 選択したノードを分割単位Ｓに組み込む．

図 10に基本処理２の適用例を示す．図 10の状態で基本処理２の処理対象となるのは，ノードA，B，C，Dである．図 10でのカットエッジは，集合Ｓに接続する枝になる．各ノードを集合Ｓに組み込むと，カットエッジ数はノードA，B，Cでは各１本減少，ノードDでは４本増加する．同評価であるノードA，B，Cの各次数は順に５，３，５であるので，ノードA，Cについてのみノード重みを比較する．この結果，ノード重みの小さなノードAが選択される．

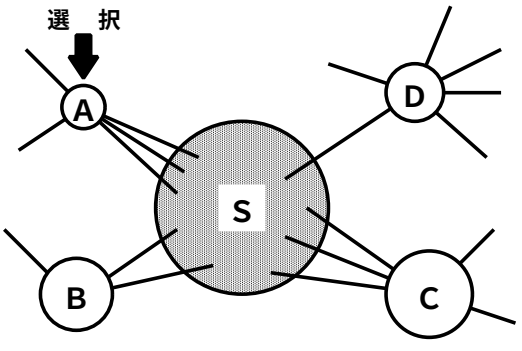

図 10: 基本処理２の適用例

(c) 提案手法

本論文で提案する新階層木分割手法は，前述の基本処理１，２の組合せである．分割単位を特定する前半過程に基本処理１を，後半に基本処理２を適用する．以下に提案する新手法の処理手順を示す．ただし，繰り返し回数 x，基本処理１，２の全実行回数の内で基本処理１を実行する割合を変化させる変化量 p は，各節点で共通して用いるパラメータ値である．図 11に提案手法のフローチャートを示す．

提案手法

1. ［ 初期設定 ］

・希望する分割過程を階層木で表現する．

・繰り返し回数 x，基本処理１実行割合の変化量 p を設定する．

2. ［ 分割処理 ］

高さ０の節点における希望サイズであるかどうか調べる．

(true)　高さ０の節点における分割単位と特定する．その後，１つ上の階層での分割処理へ戻る．

(false)　節点の全ノードが分割単位と確定するまで，以下を繰り返す．

.f.1.　以下を x 回繰り返す．

.f.1.1.　$PA = 0\%$とする．

.f.1.2.　PA が１００％を越えるまで，以下を繰り返す．

.f.1.2.1. 分割単位として確定していないノードのうち，次数最大 (同評価の場合はノード重み最大のノード) のノードを始点として分割単位に特定する．

.f.1.2.2. 希望サイズ× PA にノード重みの総和が一致するまで基本処理１を実行する．

.f.1.2.3. 希望サイズにノード重みの総和が一致するまで基本処理２を実行する．

.f.1.2.4. $PA = PA + p$ とする．

.f.2. $x \times \{\frac{100}{p} + 1\}$ 回の分割結果から，カットエッジ最小の結果を分割単位として特定し，グラフから切り離す．

.f.3. 切り離したグラフにＳＴＥＰ２以下の分割処理を再帰的に適用する．

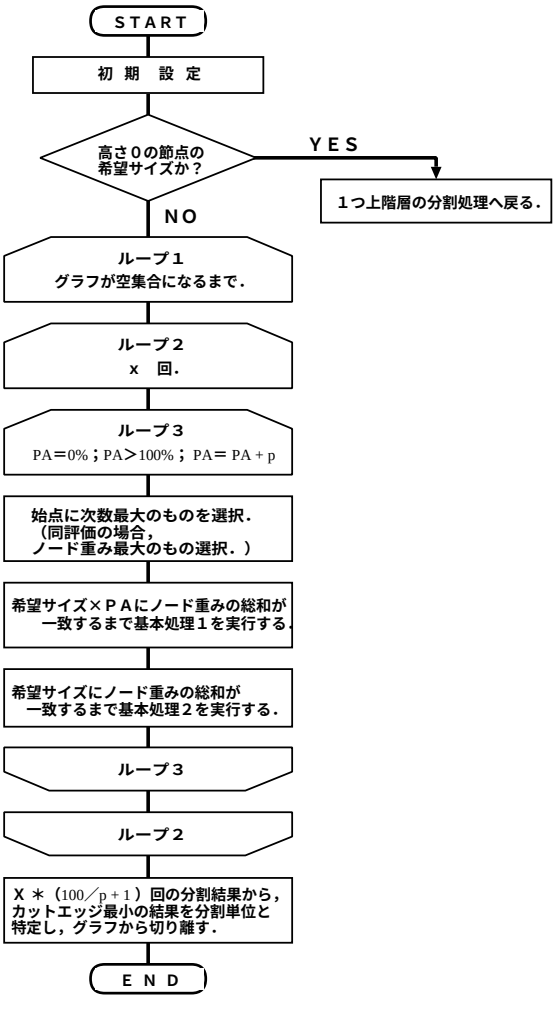

図 11: 提案手法のフローチャート

4 実験および評価

(1) 実験方法

本手法の有効性を確認するために，従来の階層木分割手法との比較実験を行った．評価するのはカットエッジ数（ＣＥ数）と，全処理時間である．実験を行った計算機環境は，Intel Pentium II 266MHz(FreeBSD-4.0)，プログラム言語にはC言語を用いた．

実験には５通りのベンチマーク回路を用いた．それぞれの回路を§２.(２) で説明した方法で変換した結果を，表 2に示す．なお，その際には各ノード重みは§２.(３) の表 1に示したサイズ比を用いる．

表 2: 実験データのグラフ情報

回路名	ノード数	枝 数
c432	２０３	６４８
c499	２７５	１１５２
c880	４６９	１３９７
c1355	６１９	２２２４
c1908	９３８	２６０９

各手法でのパラメータは以下の９通りであり，各方法３０回づつ分割処理を行って，最大，最小，平均を評価した．

従来手法

- （ε，Δ）＝（ 0.1 , 0.1 ）
- （ε，Δ）＝（ 0.05 , 0.05 ）
- （ε，Δ）＝（ 0.01 , 0.01 ）

提案手法

p ＝ １０%に対して，
- x ＝ １回
- x ＝ ５回
- x ＝ １０回

p ＝ ５%に対して，
- x ＝ １回
- x ＝ ５回
- x ＝ １０回

なお，各実験での階層木の木構造は「高さ２の完全２分木」(参考文献 [２][３]) とした．

(2) 実験結果

表 3に回路規模が最大の c1908 回路の実験結果例を示す．前述のパラメータ設定で各３０回の実行結果である．

他の回路の結果については文献 [７] を参照されたい．

各手法の性能比較をするために，図 12に平均的なカットエッジ数について，図 13に平均的な処理時間についてのグラフを示す．なお，従来手法のパラメータは基本的

に良い結果を算出する可能性が高い組合せとして,（ε，Δ）＝（0.01，0.01）を，提案手法では，p＝５%，１０%で，x＝５回とした.

比較すると，分割品質すなわちカットエッジ数は平均して３０%の削減が得られ，処理時間もはるかに高速であった.

表 3: ｃ１９０８回路の分割結果

	実験パラメータ		平均ＣＥ数（本）	最大値（本）	最小値（本）	平均処理時間（秒）
従来法	(ε，Δ)＝(0.1，0.1)		303.0	602	250	77.2
	(ε，Δ)＝(0.05，0.05)		288.6	527	237	96.2
	(ε，Δ)＝(0.01，0.01)		278.2	425	240	745.8
提案手法	p = 10%	x＝１回	169.6	181	155	7.1
		x＝５回	161.7	173	151	35.2
		x＝１０回	157.8	165	147	70.3
	p = 5%	x＝１回	167.5	181	153	14.1
		x＝５回	158.1	169	145	70.1
		x＝１０回	155.9	167	147	140.3

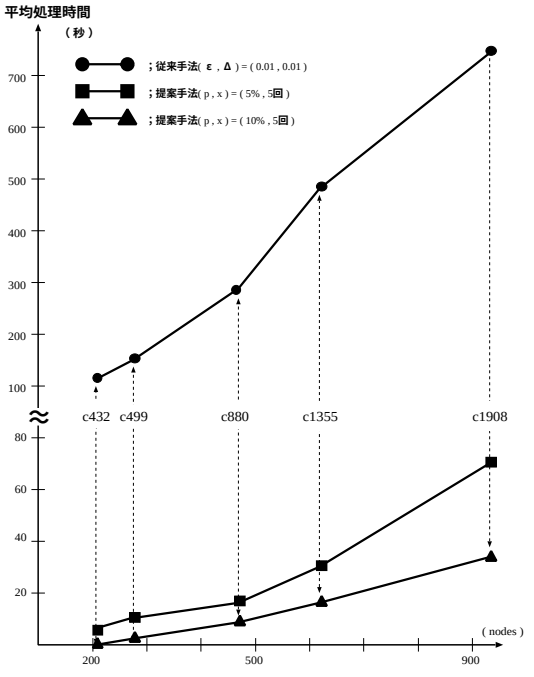

図 13: 従来手法と提案手法の平均処理時間の比較

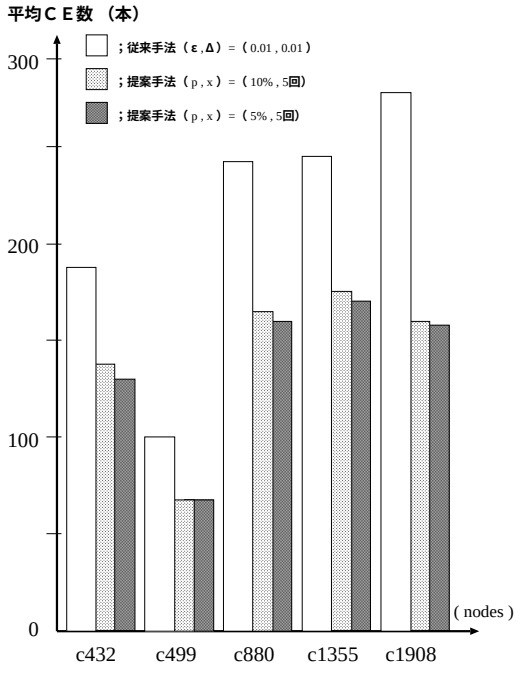

図 12: 従来手法と提案手法の平均ＣＥ数の比較

(3) 考察

提案手法では，評価値算出処理を基本処理１，基本処理２，各処理の実行回数を制御するパラメータ PA の組み合わせにより従来手法以上の品質を実現しつつ，処理時間の短縮に成功した．理由を以下に考察する.

評価値算出処理の結果により得られた枝重みを分類すると，枝重みが大きいグループ，小さいグループ，そしてその中間グループの３グループに分けられる．この結果と評価値算出処理の過程から考えると，任意のノード間に経路が数多く存在する場合，その各経路を構成する各枝重みは経路数が多いほど小さくなる．経路数は経路中に次数の大きいノードがあれば，確率的に多くなる．逆に，任意のノード間の経路が少ない場合は，多い場合に比べて経路の各枝重みは大きくなる．従って，次数の大きいノード同士を繋ぐ枝重みは小さく，次数の小さいノード同士を繋ぐ枝重みは大きく算出される確率が高くなる．つまり，従来手法では評価値算出処理で枝重みの小さい枝を特定しようとしていたことを，提案手法では次数の大きいノードを選択するだけの基本処理１で確率的に高い割合で処理している．同様に，枝重みの大きい枝の特定は基本処理２で，中間の枝の特定については基本処理１，２の実行回数を制御しているパラメータ PA で処理している．これらに加えて，基本処理２はカットエッジ数が少なくなるノードを貪欲的に選択するという役割も果たしているために，単なる階層木分割手法以上の性能を達成できたと考えられる.

この基本処理２が評価値算出処理より優れている点を図 14を例に説明する．図 14左に示すように，次数１のノードＡがノードＢに繋がっている部分をもつグラフを考える．このグラフに評価値算出処理を適用すると，ノードＡへの経路は必ずノードＢを経由しなくてはならなくなる．これはそのノード間を接続する枝重みが非常に大きく算出されることを意味する．この状態で分割

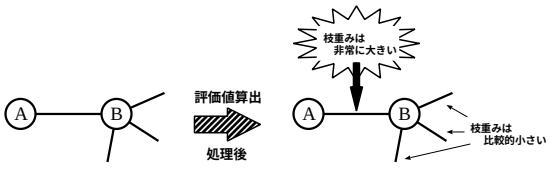

図 14: 基本処理２の説明

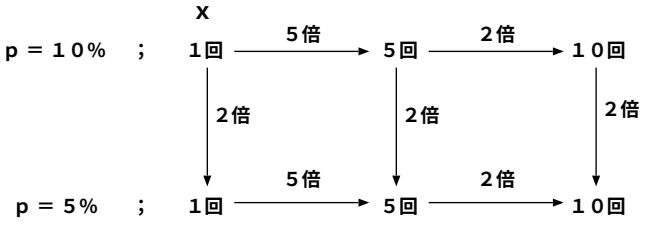

図 15: 提案手法のパラメータ設定と処理時間の関係

処理を施した場合，ノードＢが分割単位に組み込まれてもノードＡＢ間の枝重みが大きすぎるためにノードＡが分割単位に組み込まれる確率は非常に低い．それに比べ基本処理２では，ノードＢが選択されていれば，カットエッジを減少させられるようなノードも貪欲的に選択していける．この差が，階層木分割手法よりも優れている要因であると考えられる．

提案手法では設定するパラメータは p と x であった．パラメータ p が大き過ぎては大まかな分割となって，分割品質が低下する恐れがある．一方，小さくしすぎても無駄に処理回数を増加させるだけである．パラメータ x は小さ過ぎると各節点での最適な PA を見出すことが難しくなり，分割品質を低下させ，大き過ぎると，やはり無駄な処理回数の増大に繋がる．これらは，実験結果にも現れている．このような理由から，パラメータ p は５～１０%，パラメータ x は５回あたりが最適であると思われる．従来手法でのパラメータεやΔの設定に比べれば，提案手法での p や x は取りうる範囲が狭いために，パラメータ設定が比較的容易である．またパラメータ p, x と処理時間の間に図15に示すような関係が実験結果から得られた．パラメータ p が一定であれば，処理時間はパラメータ x の増減に比例する．パラメータ x が一定であるときは，パラメータ p に反比例する．これは，評価値算出処理のような複雑な繰り返し処理がないためである．従って提案手法では，パラメータを設定すれば処理の終了時間が予測できる．従来の貪欲戦略の手法では終了時間の予測は不可能であった．階層木分割手法においては，評価値算出処理での繰り返し回数を正確に特定できないことから，終了時間の予測はやはり難しい．これらに比較して，提案手法の大きな特長と言える．

5　まとめ

本研究ではシステム分割問題を解決するために対象となるシステムをグラフでモデル化し，グラフ分割問題に帰着し解決した．提案手法は従来手法に解品質，処理時間の両面で優っていた．

また，提案手法で用いるパラメータは，カットエッジ数の削減率と処理時間の関係から，最適なパラメータの範囲を特定することができた．さらに，パラメータ設定の組み合わせにより処理の終了時間を予測できるという特長を確認できた．以上から，提案手法はシステム分割問題に非常に有効であると言える．

今後の課題としては，多端子ネットのように完全グラフ，もしくはそれに近い構成となっている部分を処理前に１つのノードとしてグルーピングおくことで，さらに処理時間を短縮し，分割品質の向上を図ることが挙げられる．

参考文献

[1] 渡邊　孝博:“ＶＬＳＩレイアウト自動設計の現状と可能性，”電子情報通信学会誌 Vol.76,pp.774-782,1993-7.

[2] Ming-Ter Kuo and Chung-Kuan Cheng,”A network flow approach for hierarchical tree partitioning,”34th Design Automation Conference Proceedings,pp.512-517,1997.

[3] G.Vijayan,”Generalization of min-cut partitioning to tree structures and its applications,”IEEE Trans. on Computers Vol.40,No.3,pp.307-314,1991.

[4] W.Wolf,”Modern VLSI Design ～A SYSTEMS APPROCH～,”PTR Prentice Hall,1944.

[5] 浅野, 今井,”計算とアルゴリズム ～計算機の科学～,”オーム社,1986.

[6] 徳本　守彦,”ＶＬＳＩ配置問題における階層木分割手法,”山口大学卒業研究論文,1999.

[7] 徳本　守彦,”大規模システムの効率的な階層木分割手法,”山口大学大学院修士論文,2001.

(2001.8.31 受理)