設 計 自 動 化 66－4
（1993. 3. 12）

AR分割法の改良とその評価

松下 尚平　高久 進　高木 弘朗　上田 和宏
芝浦工業大学 システム工学部

AR回路分割法は、モジュール間の距離情報を力学的な吸引・反発力に対応づけて、グラフの平衡状態を求めこれを分割することにより非常に優れた分割結果を得るものである。本手法の欠点は、全てのモジュール間の最短経路を求める処理に$O(n^3)$の計算時間を必要とすることである。これを解決する手段として、クラスタリング技法を取り入れて、実験を行ない、その有効性を確認した。また、従来のAR分割法では、平衡処理を2次元的に行なっていたが、これを3次元処理することによって解が改善されることを示した。

IMPROVEMENTS ON THE AR PARTITIONING METHOD AND THEIR EVALUATION

Shohei Matsushita　Susumu Takaku　Hiroaki Takagi　Kazuhiro Ueda

Faculty of Systems Engineering

Shibaura Institute of Technology

307 Fukasaku, Omiya, Saitama, 330 Japan

The main drawback of the AR partitioning method is to take a vast amount of CPU time when the circuit size increases, because its time complexity is $O(n^3)$. To solve this problem, a clustering technique has been applied to the AR method. Experiments show that it is quite effective to shorten the CPU time. In addition, it is shown that a three-dimensional balancing process can obtain better solutions compared to a two-dimensional one.

1 はじめに

LSIやプリント基板によって電子回路を実現する際、大規模な機能回路や論理回路をより小規模な回路に分割する必要が生じる[1]。これを一般に回路分割問題(circuit partitioning problem)という。この分割問題の最も基本的な形は、与えられた回路を等しい規模に分割し、分割された回路ブロック間の配線数を最小にするものである。回路分割問題も計算複雑度の視点からNP-完全問題のクラスに分類される。つまり、厳密な最適解を求めようとすれば、回路規模の指数関数に比例する計算時間を要することになり、まず実用的な規模の問題に対しては、このようなアプローチはとれないことは明らかである。従って、各種の発見的手法が提案されてきた。KernighanとLin[2]のアルゴリズムは$O(n^2 log n)$（ここで、nはモジュール数）であることが知られており、FiduccaとMattheyses[3]は、バケットリストや効率的なデータ構造を導入することによりK&L法を大幅に高速化し、O(p)（ここでpはピン数）を実現したとしている。しかし、これらのアルゴリズムは発見的手法であるために、ほとんど局所的最適解(local optimum solution)にとらわれてしまう欠点を持っている。また、同時に分割の初期解に強く依存してしまうという欠点を持つことが知られている。われわれは、このような従来法の持つ欠点を解決する新しい手法としてAR分割法を提案してきた。本手法は、回路ネットワークの持つ固有のグラフ的性質を利用して、初期値の良否にかかわらず最適解か、あるいはこれに非常に近い局所的最適解を得ることのできるものである。しかしAR分割法は、

i) モジュールの数の2乗に比例したメモリを必要とするため大規模な回路を分割することが難しい（4000モジュールで約32MByte必要）、
ii) 必要なメモリを確保できたとしてもモジュールの3乗に比例した計算時間を必要とするため、分割する回路規模が非常に大きくなると実用的な時間で解を求めることが困難になる、
iii) 平衡処理を2次元的に行なっているため、回路ネットワークが非平面的なグラフの場合、一定の形に収束しにくい、
iv) 分割の際に一つの平面を直線でしか分割できないので、よりよい分割状態を見過ごしている可能性がある、

などの問題点があった。そこで、本報告では、i) ii)の問題点については、

・クラスタリングを導入し、処理する回路ネットワークの規模を縮小する、

iii) iv)の問題点については、

・平衡処理を3次元的に行ない、立体を平面によって、あらゆる方向から分割することにより分割結果を改善する、

などのアプローチを取ることによってこれらの問題点を解決したので以下に報告する。また、回路ネットワークを画面上で立体収束形として認識できることを利用した、人手の介入によるインタラクティブな回路分割についても述べる。

2 AR分割法

図1にAR分割法[4]の処理手順を示す。

本手法では、まず与えられた回路ネットワーク

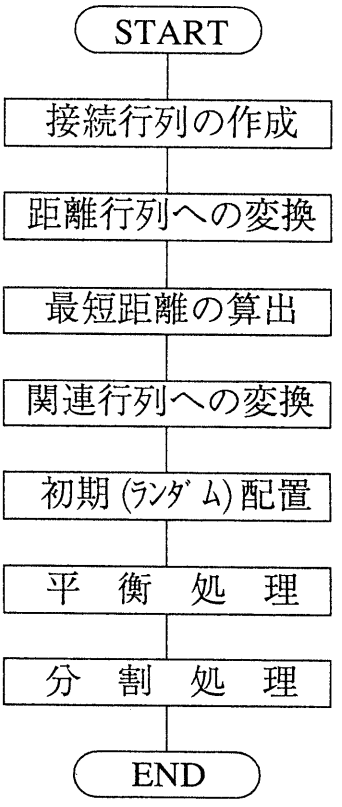


図1 AR分割法の処理手順

をグラフ表現し、グラフ上で求めたモジュール間の距離情報に基づいて、モジュールペア間の親密・疎遠関係を定義する。これを力学的吸引・反発力に対応付けて各モジュールに作用させることによって得られる平衡状態を、適当なカットラインによって分割することにより解を得るものである。各手順を以下に示す。

2.1 接続行列

接続行列Cは、モジュール間の直接的な接続情報を表す行列であり、任意の2つのモジュール間の接続の強さを表したものである。多端子ネットは重み付け完全グラフ化し、2端子ネットに変換している。1本のネットにm個のモジュールがつながっている場合、モジュール相互が信号の受け渡しをするのに必要なネットの本数はm - 1であり、完全グラフ化後のネットの本数はm(m -1) / 2であるから、完全グラフ化後のネットの重みは、

$$m - 1 = \rho m(m - 1) / 2 \qquad (2.1)$$

$$\therefore \rho = 2 / m \qquad (2.2)$$

となる。

2.2 距離行列

接続行列Cの要素はモジュール間の接続関係の強さを表しており、接続関係の強いモジュールほどその距離は短いと考えられる。そこで距離行列Dの行列要素D(i, j)を接続行列Cの行列要素の逆数で定義する。

$$D(i, j) = 1 / C(i, j) \quad (i \neq j \text{ かつ } C(i, j) \neq 0) \qquad (2.3)$$

$$D(i, j) = \infty \quad (i \neq j \text{ かつ } C(i, j) = 0) \qquad (2.4)$$

$$D(i, j) = 0 \quad (i = j) \qquad (2.5)$$

この距離は直接接続関係があるモジュール間の距離を表しており、接続関係のないモジュール間の距離(∞)は十分大きい値を入れておく。

2.3 最短距離行列

直接接続関係のないモジュール同士も、他のモジュールを経由して接続されているため、その距離を求めなくてはならない。そこで図8に示すフロイドのアルゴリズム[5]により全てのモジュールペア間の全てのパスを検索し、最短の距離をそのモジュール間の距離とする。

このようにして得られた最短距離を距離行列の行列要素に置き換えることで最短距離行列を得る。このアルゴリズムの計算複雑度が$O(n^3)$のため、AR分割法の計算機時間がn^3に比例する。

以下モジュール間の距離とは、モジュール間の最短距離を指すものである。

2.4 関連行列

最短距離行列Dの各要素から、平均距離を引いて関連行列Rに変換する。

$$R(i, j) = D(i, j) - Av \quad (i \neq j) \qquad (2.6)$$

$$R(i, j) = 0 \quad (i = j) \qquad (2.7)$$

ただし、

$$Av = \Sigma D(i, j) / (n(n - 1)) \qquad (2.8)$$

関連行列Rの各要素は各モジュール間の平均距離に比べて近距離にあるか、遠距離にあるか、あるいは等しいかでそれぞれ負、正、零の値を取ることになる。従って、集合M(i), Z(i), P(i)を、

$$M(i) = \{j \mid R(i, j) < 0\} \qquad (2.9)$$

$$Z(i) = \{j \mid R(i, j) = 0\} \qquad (2.10)$$

$$P(i) = \{j \mid R(i, j) > 0\} \qquad (2.11)$$

と定義する。M(i)はモジュールiと親密な関係にあるモジュールの集合、P(i)は疎遠な関係にあるモジュールの集合、P(i)はどちらでもないモジュールの集合である。|R(i,j)|は親密度、疎遠度を表す。

2.5 初期配置

平衡処理を行う前に、各モジュールが一点に集まらないように初期座標を与える。AR分割法は、初期配置に依存しないので、ここでは乱数を用いて定めている。

2.6 平衡処理

平衡処理では、各モジュールに対してM(i)に属するモジュールからは吸引力を、P(i)に属するモジュールからは反発力を、|R(i,j)|に比例して作用させる。1回の操作によるモジュールiの移動後の位置は、モジュールiをまずM(i)に属するモジュールによる吸引力の重心位置に移動させ、その後P(i)に属するモジュールによる反発力の合力方向に移動させて求める。この操作を各モジュールに対して行い、これを必要回数繰り返すと全モジュールにつにて平衡状態が得られる。

(1) 吸引力による移動

移動前のモジュールiの座標：(x_i, y_i)

移動後のモジュールiの座標：(x_i', y_i')

$$x_i' = \frac{\sum_{j \in Mi} |R(i, j)| \cdot x_j}{\sum_{k \in Mi} |R(i, k)|} \qquad (2.12)$$

$$y_i' = \frac{\sum_{j \in Mi} |R(i,j)| \cdot y_j}{\sum_{k \in Mi} |R(i,k)|} \quad (2.13)$$

(2) 反発力による移動

L_{ij}：モジュールi, j間の距離

$$\Delta x_i' = \frac{\sum_{j \in Pi} R(i,j) \cdot \frac{x_i' - x_i}{L_{i,j}}}{L_{i,j}^2} \quad (2.14)$$

$$\Delta y_i' = \frac{\sum_{j \in Pi} R(i,j) \cdot \frac{y_i' - y_i}{L_{i,j}}}{L_{i,j}^2} \quad (2.15)$$

(3) モジュールiの座標の更新

$$x_i \leftarrow x_i' + \Delta x_i' \quad (2.16)$$

$$y_i \leftarrow y_i' + \Delta y_i' \quad (2.17)$$

2.7 分割処理

平衡処理で得られた各モジュールのx座標を用いてソートし、２つのブロックのモジュール数が等しくなるように分割し、２つのブロック間にまたがるネットの数を算出する。この処理を0度から180度まで微小角度づつ回転させながら行い、分割されるネットの数が最小になる分割を選択する。

2.8 AR分割法の処理例

図2にAR分割法の処理例を示す。用いた回路は、8×8モジュールの格子状の回路である。

3 クラスタリングAR分割法

従来のAR分割法の前処理にクラスタリング処理を行い、グラフの規模を小さくすることによって、AR分割法の高速化を計ったものである。本手法では、SchulerとUlrichによるクラスタリング手法[6]を用いた。図3に接続度が２のモジュールペアの例を示す。これらは全て接続度が２のモジュールペアであるが、クラスタリングする際、全て同じように取り扱うことは不自然である。この場合(c)のモジュールjはi以外に接続されているモジュールはなくiとクラスタリングされるのが自然であると考えられる。これと対照的に(a)モジュールi, jは、ほかのモジュールとも接続関係があり、一概にクラスタリングされるべきではない。このように、評価するモジュールペア間の接続度だけではなく、まわりとの接続関係も評価しながらクラスタリングしないと、良い結果が得られない。そこでクラスタリング時の接続度として、式(3. 1)の関数

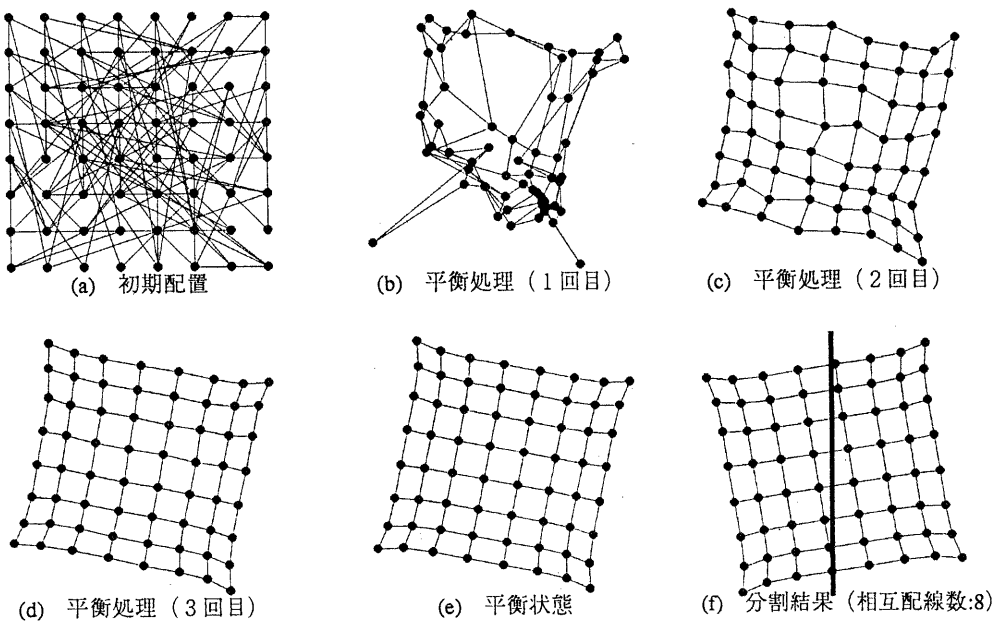


図2 AR分割法の処理例

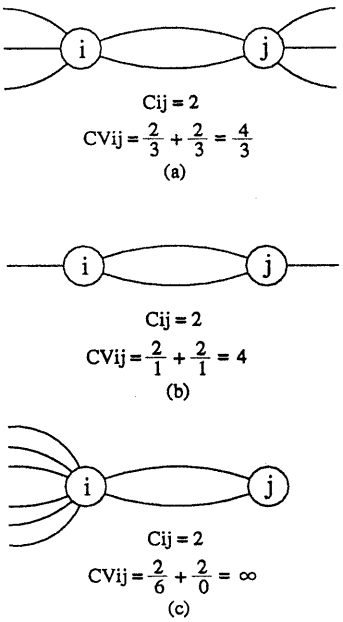


図3　CijとCVijの評価の違い

を用いる。

$$CVij = f(Si)\frac{Cij}{Ti - Cij} + f(Sj)\frac{Cij}{Tj - Cij} \quad (3.1)$$

ここで、

Cij : モュールi, j間の接続強度

Ti, Tj : モジュールi及びjにつながるネットの接続強度の合計

f(Si), f(Sj) : モジュールi及びjの面積の関数

図3　に示すように、この式を用いることで、評価するモジュールペア間の接続度だけでなく、まわりの接続状況を評価に含めることができる。ただし、この手法は、接続行列を作成するためn^2のメモリを必要とするが、接続行列の行列要素が0のところを記憶しないようにする事により、使用メモリの量を大幅に節約している。

4　3次元ＡＲ分割法

従来のＡＲ分割法では平衡処理を2次元平面上で行なっていたが、回路ネットワークが非平面的なグラフの場合、それを平面上だけで収束させようとするのは無理がある。ここで、非平面的なグラフとはネットとネットの交差が多い回路のことである。このような回路の場合、x軸、y軸に加えて新たにz軸を設けてこの3軸に対して平衡処理を行なえばよい。分割する際は、この立体収束形を平面で2分割することになる。こうすることによって平面を直線で2分割していたのに比べて、分割箇所が非常に多くなるので分割結果は良くなることが予想される。その反面、処理時間は長くなることになる。3次元AR分割法は平衡処理及び分割処理を立体的に行なう（z軸が増える）という以外はその処理手順は従来の2次元AR分割法と全く同じであるので変更部分だけを簡潔に述べる。

4.1　平衡処理

2次元AR分割法と全く同じであるが新たにz軸が追加されて次のようになる。

(1)　吸引力による移動

移動前のモジュールiの位置 : (x_i, y_i, z_i)

移動後のモジュールjの位置 : (x_i', y_i', z_i')

$$x_i' = \frac{\sum_{j \in Mi} |R(i,j)| \cdot x_j}{\sum_{k \in Mi} |R(i,k)|} \quad (4.1)$$

$$y_i' = \frac{\sum_{j \in Mi} |R(i,j)| \cdot y_j}{\sum_{k \in Mi} |R(i,k)|} \quad (4.2)$$

$$z_i' = \frac{\sum_{j \in Mi} |R(i,j)| \cdot z_j}{\sum_{k \in Mi} |R(i,k)|} \quad (4.3)$$

(2)　反発力による移動

Li,j :　モジュールi, j間の距離

$$\Delta x_i' = \frac{\sum_{j \in Pi} R(i,j) \cdot \frac{x_i' - x_i}{Li,j}}{Li,j^2} \quad (4.4)$$

$$\Delta y_i' = \frac{\sum_{j \in Pi} R(i,j) \cdot \frac{y_i' - y_i}{Li,j}}{Li,j^2} \quad (4.5)$$

$$\Delta z_i' = \frac{\sum_{j \in Pi} R(i,j) \cdot \frac{z_i' - z_i}{Li,j}}{Li,j^2} \quad (4.6)$$

(3)　モジュールiの座標の更新

$x_i \leftarrow x_i' + \Delta x_i'$　(4.7)

$y_i \leftarrow y_i' + \Delta y_i'$　(4.8)

$z_i \leftarrow z_i' + \Delta z_i'$　(4.9)

4.2 分割処理

これも考え方は2次元AR分割法と全く同じであるが処理手順は若干異なる。従来はxy平面上で分割を行なっていたのでモジュールの回転はxy平面に垂直に$0° \leqq \theta \leqq 180°$だけであったのに対して、3次元AR分割法の場合、このθ各々に対してxz平面に垂直に（又はyz平面に垂直に）$0° \leqq \phi \leqq 180°$の回転が加わる。従って1度おきに分割する場合、平面であれば180回で済んでいたものが立体の場合は$180 \times 180 = 32400$回も繰り返し分割処理を行なうことになる。これは分割箇所が180倍になるので分割結果が良くなることが予想される反面、分割処理に費やされる時間は逆に180倍、長くなるということである。従って、実用的な時間内で解を得るためには、初めは粗くカットし、良いところだけをさらに細かくカットする必要がある。このような処理をしても3次元AR分割法本来の解よりは若干悪くなるものの従来の2次元AR分割法よりは良い解が期待される。

5 インタラクテイブな回路分割

AR分割法の応用としてインタラクテイブな回路分割について述べる。AR分割法はモジュール群を図形的に2分割するのでその処理過程や分割結果が画面上で認識できるのがほかの手法にない大きな特長である。従って処理過程や分割結果を見ながら人手を介入させ、新たな処理を加えることも可能である。

図4に平衡状態のものを直接画面上で分割した例を示す。これは、多重分割の際、モジュールのかたまりを画面上で認識できるので有効な手段である。さらに、この分割結果を初期解として別の手法を行なう場合を考えてみる。このとき図4のAとBは全モジュールのなかで最も接続関係が弱いもの同士であるから、いかなる手法を用いてもこれが入れ替わることはないといえる。従って,図4のCのように両グループの境界領域のみをマウスなどで指定した後、この領域内のモジュールにだけ他の手法を用いることにより、全て自動で分割するより、短時間により良い解が得られる可能性がある。

6 実験結果

テスト回路は、MCNC(Microelectronics Center of North Carolina)のベンチマークテストより、PRIMARY1, PRIMARY2, STRUCT, FRACTを用いた。また、人工的に発生させた最適解のわかっている、格子状、木状、クラスタ状の回路を用いた。

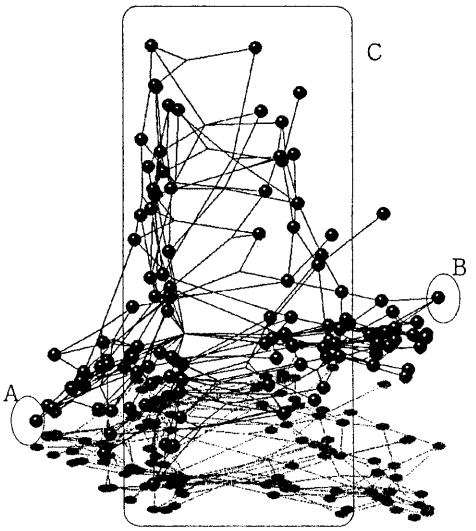


図4 インタラクティブな回路分割例

改良したAR分割法の有効性を確認するため、従来のＡＲ分割法、Ｋ＆Ｌ法、Ｆ＆Ｍ法、及びSA(Simulated Annealing)[7]分割法についても同じ回路を用いて実験を行った。

実験には、EWS(Sun SPARCstation IPX:28.5MIPS 8MFLOPS)上で、C言語を用いて実験プログラムを作成して行った。

図5~7にPRIMARY2をテスト回路として用いたクラスタリングAR分割法の処理時間、メモリサイズ、相互配線数とクラスタリングとの関係を示す。ただし、クラスタ率とは式(6. 1)で表される値である。これらの図より、クラスタリングによって、使用メモリの節約や、処理時間の短縮ができることが判る。また、クラスタ率と最終解の間には関連性があるものの、規則性はないことが判った。

表1にAR分割法及びクラスタリングAR分割法と従来手法との比較結果について示す。ただし、クラスタリングAR分割法の結果については、相互配線数が最小になったクラスタ率のものについて記してある。また、K&L法、F&M法、SA分割法につ

$$\text{クラスタ率} = \frac{\text{クラスタリングによるモジュール数の減少数}}{\text{クラスタリング前のモジュール数}} \quad (6.1)$$

いては、いくつかの初期解を与え、相互配線数が最小になったものについて記してある。この表より、

・クラスタリングAR分割法の方が、AR分割法よりも良い解が得られる場合があった。これは、クラスタリングによってネットワークが簡単になったためであると思われる、

・3次元AR分割法は2次元AR分割法に比べて、良い解が得られるが処理時間は多少長くなる、

などが判った。

図8にPRIMARY1のクラスタリング技法を導入した3次元AR分割法による処理経過例を示す。

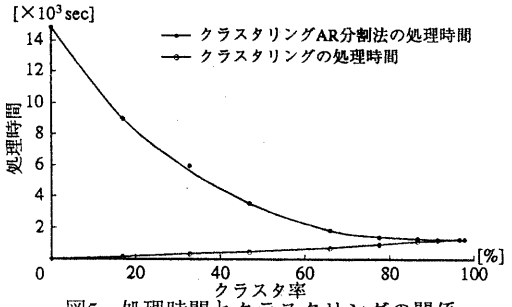


図5　処理時間とクラスタリングの関係

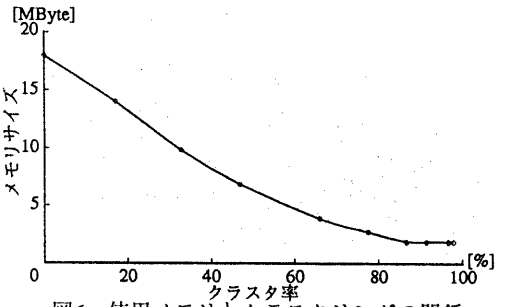


図6　使用メモリとクラスタリングの関係

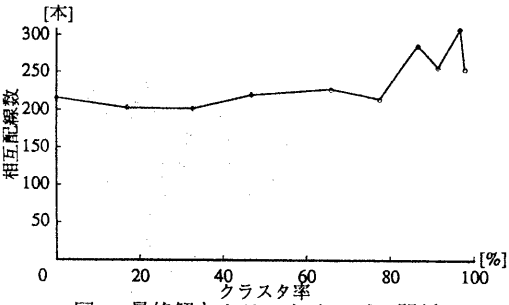


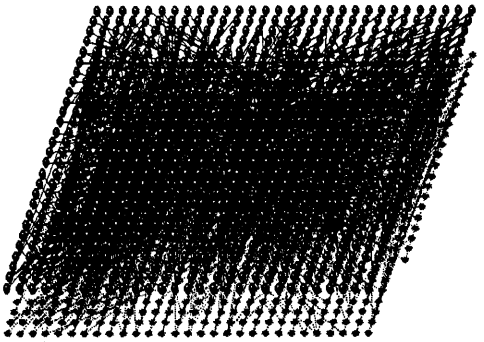
図7　最終解とクラスタリングの関係

表1　従来手法との比較

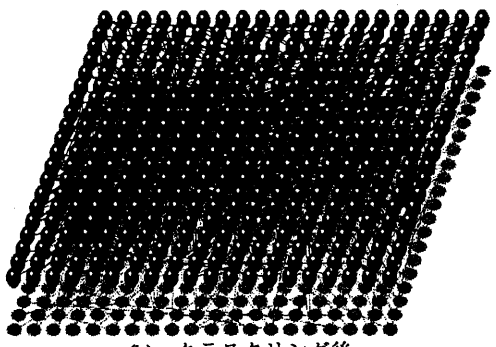
テスト回路	回路規模		AR分割法		クラスタリングAR分割法		3次元AR分割法	
	モジュール	ネット	最終解	処理時間	最終解	処理時間	最終解	処理時間
PRIMARY1	833	902	70	422.4	63	111.6	62	451.9
PRIMARY2	3014	3029	216	14799.4	202	5970.7	293	11553.9
STRUCT	1888	1888	43	2496.1	36	117.7	42	2254.2
FRACT	125	128	11	5.9	13	2.0	11	27.6
32×32 CLUSTER	1024	1161	1	350.0	1	12.7	1	986.0
32×32 GRID	1024	1984	32	580.8	34	333.6	32	1274.2
32×32 TREE	1024	1023	1	313.4	1	9.5	1	947.0
16×16 CLUSTER	256	285	1	19.4	1	2.7	1	19.4
16×16 GRID	256	480	16	22.1	18	3.3	16	22.1
16×16 TREE	256	255	1	18.6	1	2.0	1	18.6

テスト回路	クラスタリング3次元AR分割法		F&M法		K&L法		SA分割法	
	最終解	処理時間	最終解	処理時間	最終解	処理時間	最終解	処理時間
PRIMARY1	59	164.4	71	0.75	95	255.0	53	1019.0
PRIMARY2	197	1718.0	201	3.1	259	22679.0	146	32026.9
STRUCT	39	938.9	49	1.3	160	3035.8	36	15339.8
FRACT	11	36.3	11	0.05	87	1.4	11	4.0
32×32 CLUSTER	1	12.7	61	1.4	49	1305.9	6	31612.5
32×32 GRID	34	333.6	32	2.1	34	863.7	32	3156.4
32×32 TREE	1	9.5	56	1.7	33	878.3	4	21098.8
16×16 CLUSTER	1	2.7	12	0.12	7	11.0	1	3140.5
16×16 GRID	18	3.3	16	0.22	16	8.8	16	31.8
16×16 TREE	1	2.0	14	0.17	11	13.1	1	3126.8

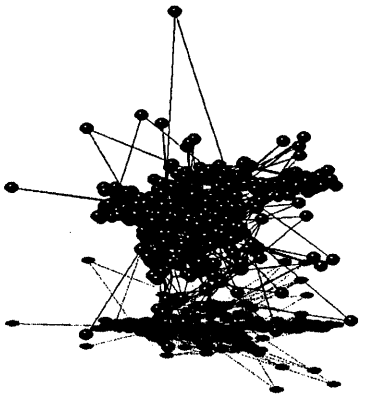
（処理時間の単位：秒）



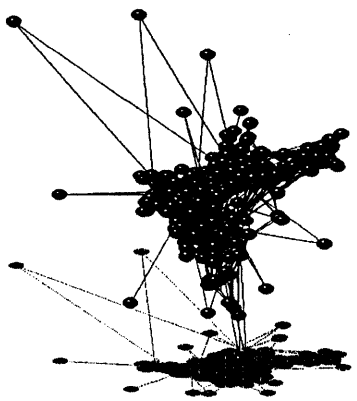
(a) 初期状態



(b) クラスタリング後



(c) 平衡処理過程



(d) 平衡状態

図8. 3次元AR分割法の処理過程

7 まとめ

AR分割法とその改良について述べた。本手法の特徴をまとめると以下のようになる。

(1) 回路グラフ上の全モジュール間の距離情報を利用するためグローバルな最適解、あるいはそれに近い解が得られる。

(2) 3次元AR分割法とクラスタリングを組み合わせることにより、短時間でより良い結果が得られる。

今後の課題として、クラスタリングの高速化、グラフ形状の最終解への影響の検討、インタラクティブなAR分割法と他手法の組み合わせなどがあげられる。

参考文献

[1] W.E.Donath, "Logic Partitioning", in Physical Design Automation of VLSI Systems, The Benjamin/Cummings Publishing Company, 1988, pp.65-86.

[2] B.W.Kernighan and S.Lin, "An Efficient Heuristic Procedure for Partitioning Graphs", Bell System Technical Journal, Vol.49, No.2, Feb. 1970, pp.291-307.

[3] C.M.Fiduccia and R.M.Mattheyses, "A Linear Time Heuristic for Improving Network Partitions", Proc. 19th Design Automation Conference, 1982, pp.175-181.

[4] 松下、島谷、上田:"吸引・反発力を利用した論理分割手法 AR分割法", DAシンポジウム'91 論文集.

[5] R.W.Floyd, "Algorithm 97, Shortest Path", Comm. ACM 5(1962).

[6] D.M.Schuler and E.G.Ulrich, "Clustering and Linear Placement", Proc. 9th Design Automation Workshop, 1972, pp.50-56.

[7] S.Kirkpatric, "Optimization by simulated annealing", Science, Vol.220, No.4958, pp.671-680.