設 計 自 動 化 67－8
（1993．6．25）

遅延考慮配置アルゴリズム

袖　美樹子　　宮沢 義幸＊
日本電気（株）　C＆Cシステム研究所
＊NEC技術情報システム開発（株）
〒216 川崎市宮前区宮崎4丁目1-1
＊〒213 川崎市高津区坂戸3-2-1 KSP かながわサイエンスパークビルR&Dビルd棟

あらまし

製造技術の進歩に伴いファンクションブロックのサイズは、小さくなってきている。その結果、回路の遅延時間の中で配線遅延が占める割合が大きくなってきている。また、配置はレイアウトの性能を左右する重要な部分である。そこで、遅延を考慮した高速で高性能な配置アルゴリズムが望まれている。我々はこれまで階層クラスタリング手法を用いた配置アルゴリズムについて発表を行なってきた。本稿では、階層クラスタリング手法を用いた配置アルゴリズムに対し、コストにより遅延を考慮する方法について述べる。特に、各ネットのパスへの影響度を求めるアルゴリズムについて述べる。計算機実験の結果、階層クラスタリング手法を用いた配置アルゴリズムで遅延を満足することができなかった例題に対しても、配線長の増加なしに遅延を満足する結果を得ることができることを確認した。

和文キーワード

遅延、配置、グラフ

A Performance driven Placement Algorithm

Mikiko Sode Tanaka　　Yoshiyuki Miyazawa*
NEC Corporation
*NEC Scientific Information System Development Ltd
1-1,Miyazaki 4-chome,Miyamae-ku, Kawasaki,Kanagawa, 216 Japan
*Kanagawa-Science-Park Building D-5F, 3-2-1, Sakato, takatu-ku, Kawasaki, Kanagawa, 213 Japan

Abstruct

As technology advances, the size of modules in integrated circuits becomes smaller. Consequently, while the effect on intra-module delay becomes less significant, the effect on inter-module wire delay becomes more prominent. Thus, minimizing interconnecting wire delay becomes an important issue in improving VLSI circuit performance. We have proposed the hierachical clustering with min-cut exchange method. In this paper, we present a net weighting approach for timing driven placement in the hierachical clustering with min-cut exchange method. Computational results show that our algorithm is able to find a solution to satisfy the delay constraint.

key words

delay, performance driven layout, placement, graph

1 はじめに

製造技術の進歩に伴いファンクションブロックのサイズは、小さくなってきている。その結果、回路の遅延時間の中で配線遅延が占める割合が大きくなってきている。また、配置はレイアウトの性能を左右する重要な部分である。そこで、遅延を考慮した高速で高性能な配置アルゴリズムが望まれている。

遅延考慮配置に関しては、今まで多くの研究が行なわれてきた [7][8][9][10][11]。それらの方法は、以下のように分類できる。

1. ネットウエイトによる方法 [10]
2. ネットの配線容量制限を守る方法 [11]
3. パスを考慮した方法 [7][8]

1の方法は、遅延制約をネットのコストに反映させるため、遅延情報を必ず守れるとは限らない。しかし、従来の配置アルゴリズムに対しコストにより遅延の考慮を追加することは容易である。

2の方法は、ネットの制限長を求め、これを守るように配置する方法である。この方法では、あるネットのネット長が変更されると、実際はこのネットを通過するパス上にある他のネットの制限長が変化するが、これについての十分な考慮がされていない。また、一般に制約を付加したすべてのネットのネット長を制限以下に抑えるのは難しいという問題がある。

3の方法では、クリティカルパス (遅延制限が厳しいパス) を幾つか列挙し、そのクリティカルパスに対し制限を満足する配置結果を得ることは可能であるが、制限パス数が増加した場合、それらのパスを同時に考慮し、配置を行なうことは、問題が複雑となり難しい。なぜなら、配置で遅延を考慮した改良を加えていった場合、あるパス1を短くすると他のパス2が長くなり、パス2を短くするとパス1が長くなるというようにクリティカルパスが絡み合っていることが多いからである。また一般に、クリティカルパスを列挙することは難しい。なぜなら、配置前、配置途中の仮想配線長を元に求めたクリティカルパスが、配置の各フェーズを通してクリティカルパスであることは保証されないからである。

我々はこれまで階層クラスタリング手法を用いた配置アルゴリズム [1] について発表を行なってきた。階層クラスタリングによる min-cut 法とは、各カットに対して、ネットの接続度により各クラスタが同じ大きさになるようにクラスタを形成する。この処理を階層的に行ない、最終的にはクラスタが2つになるまで処理を行なう。この後、2つのクラスタに対し min-cut 法を適用しクラスタの位置を決定する。この後、クラスタを一つ崩し、再度 min-cut 法を適用する。この処理をクラスタがなくなるまで繰り返す (図1)。これによりブロックをカットラインの両側から1つ取り出していたのでは、最適解を得ることができなかった問題 (図2(a)) にたいしても最適解を得ることができる (図2(b)) というものであった。

本稿では、階層クラスタリング手法を用いた配置アルゴリズム [1] に対し、コストにより遅延を考慮する方法について述べる。

配置では、配線長を短くするために無理に配置改良を行なうことはあまり望ましくない。そこで、効果的に遅延を満足する配置結果を得るためには、あるネットを短くすることにより最も遅延を短くする効果のあるネットを探し、そのネットを短くする処理を行なうべきである。ネットを短くすることにより最も遅延を短くする効果のあるネットとは、遅延制約を満足していないパス上にあるネットである。しかし、遅延制約を満足していないパス上には多くのネットが存在する。この中でネットを短くすることにより最も遅延を短くする効果のあるネットとは、どのようなネットであろうか？

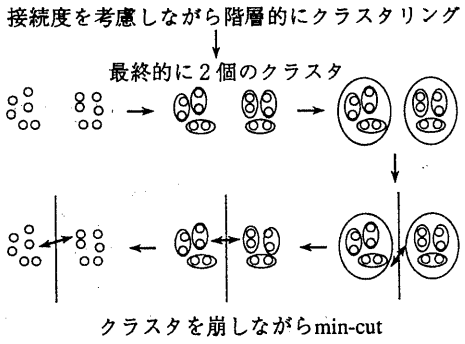


図 1: 階層クラスタリング手法による min-cut 法

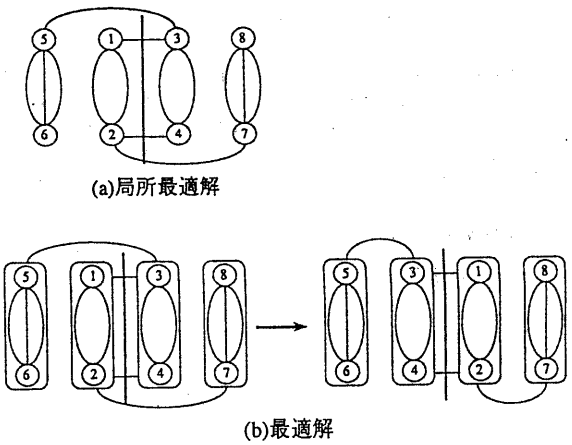


図 2: 階層クラスタリング手法の効果

各ネットを考えた場合ある1つのネットの長さを変えることにより、多くのパスに影響を与えるものと、ほとんど影響を与えないものが存在する。例えば、遅延制約を満たしていないパスのすべてがあるネットを通過している場合、そのネットを1つ短くすることでそのネットを通過するすべての遅延を満たしていないパスを短くすることができる。

以上考察したように、各ネットに対し、ネットの長さを変化させることにより、幾つのパスに影響を与えるかを求め、この値、即ち、パスへの影響度の高いネット (回路的に遅延のキーとなる部分) に対しコストを与え、配置の各フェーズで考慮することにより効果的に配置を行なうことができる。パスへの影響度の高いネットをクリティカルアークと呼ぶこととする。

本稿では、階層クラスタリング手法を用いた配置アルゴリズムに対し、コストとして遅延を考慮する方法について述べる。特に、各ネットのパスへの影響度を求めるアルゴリズムについ

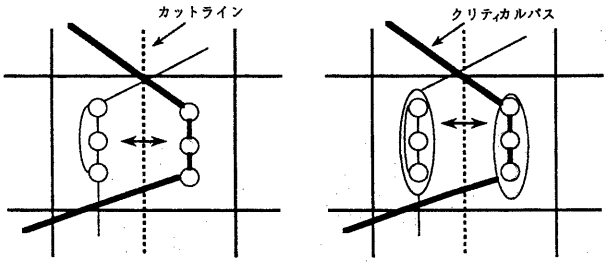


図 3: 遅延考慮クラスタリング手法による効果

て述べる。

本アルゴリズムは、ネットの仮想配線長を求め、仮想配線長を元に配線遅延を求める。次に、遅延解析 [2] を行ない、ゼロスラックアルゴリズム [3] によりスラック値の分配処理を行ない各ネットの余裕度を求める。また、各ネットのパス影響度 (クリティカルアーク) を求める。この各ネットのパス影響度と余裕度を、初期クラスタリング時、階層クラスタリング時、ペア交換時に考慮する方法である。

本アルゴリズムは、階層クラスタリング時に遅延を考慮してクラスタリングを行なうことにより、従来カットラインの両側からブロックを 1 つずつ取り出して、遅延を考慮し交換を行なっていた方法では遅延を満足する結果を得ることができなかった例題 (図 3(a)) に対しても、遅延を満足する結果 (図 3(b)) を得ることができる。即ち、階層クラスタリング時に遅延を考慮してクラスタリングを行なうことにより、パスを考慮しているのと同じ効果を期待することができる。また、各ネットのパス影響度を加味し、クラスタリング, ペア交換を行なうので、効果的に遅延制約 (回路の動作速度制限) を満足する結果を得ることができる。

計算機実験 (EWS4800/350 を使用) の結果、 [1] の手法で遅延を満足することができなかった例題に対し、配線長の増加なしに遅延を満足する結果を得ることができることを確認した。

また、パスへの影響度コストの効果を計算機実験により評価した。計算機実験結果より、パスへの影響度コストが有効に作用していることを確認した。

2 全体フロー

入力として回路の動作速度制限と、 FF(フリップフロップ)-FF 間の組が与えられるとする。本アルゴリズムでは、それらを入力とし、与えられた全 FF-FF 間に対し、回路の動作速度制限 (遅延制約) を満足する配置を行なうものである。

基本的には、各カットライン毎に予想配線長を求め、エルモアの式により配線遅延を求める。次に、遅延解析を行ない、ゼロスラックアルゴリズムにより各ネットの余裕度を求める。この余裕度とパスへの影響度をクラスタリング時のコスト、ブロックのペア交換時のコストに反映させることにより遅延考慮配置アルゴリズムを実現する。

全体フローを以下に示す。

全体フロー

1. ネットのパスへの影響度を求める。
2. ネットの仮想配線長を求め、エルモアの式により各ネット配線遅延を求める。
3. 遅延解析 [2] を行ない、各ネットのスラック値を求める。
4. ゼロスラックアルゴリズム [3] によりスラック値の分配を行なう。
5. 初期クラスタリングを行なう。

```
for(各カットラインに対して){
    6. ネットの仮想配線長を求め、エルモアの式により各
       ネットの配線遅延を求める。
    7. 遅延解析 [2] を行ない、各ネットのスラック値を求
       める。
    8. ゼロスラックアルゴリズム [3] によりスラック値の
       分配を行なう。
    9. 階層クラスタリング
    for(各階層に対して){
        10. ペア交換
    }
}
```

7. 下地へのマッピング

まず、ネットのパスへの影響度を求める。次に、ネットの余裕度を求める。即ち、ネットの仮想配線長を求め、エルモアの式により各ネット配線遅延を求める。次に、遅延解析を行ない、各ネットのスラック値を求め、余裕度の分配処理を行ない各ネットの余裕度を求める。この後、ネットのパスへの影響度と余裕度を考慮して初期クラスタリング処理を行なう。

次に、各カットラインに対して、各ネットの余裕度を求める。この後、ネットのパスへの影響度と余裕度を考慮してクラスタが 2 つになるまで、階層クラスタリングを行なう。次に、各階層に対して、ネットのパスへの影響度と余裕度を考慮してカットラインを横切るネットの数が最小になる様にペア交換を行なう。

この後、下地へのマッピング処理を行なう。

この様に各カットライン毎にネットの余裕度を再計算することで、正確に遅延を反映できる。

3 クリティカルアーク

各ネットのパスへの影響度を求めるアルゴリズムについて述べる。

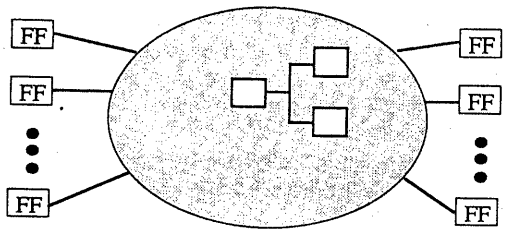
有向グラフ $G = (V, E)$ を考える。ここで V はノードの集合とソースノードとシンクノードを表す。また E は、各ネットの入力端子から出力端子への有向枝とソースノードから入力 FF への有向枝と、出力 FF からシンクノードへの有向枝の集合を表す。例えば図 4(a) の回路は、図 4(b) と表される。このグラフは、組合せ回路のグラフであるため、サイクルを含まない。拡張可能であるため、ここでは、ノード i, j に接続する枝は 1 つしかないと仮定する。

このグラフ上でノード u を通過するソースノードからシンクノードへのパスの数を、枝 $e(u, v)$ のパスへの影響度と呼ぶこととする。パス影響度が大きいほど、この枝の長さを変化させることによりパスの到達時間に影響を受けるパスの数が多いことを表す。パスへの影響度が大きい枝をクリティカルアークと呼ぶこととする。

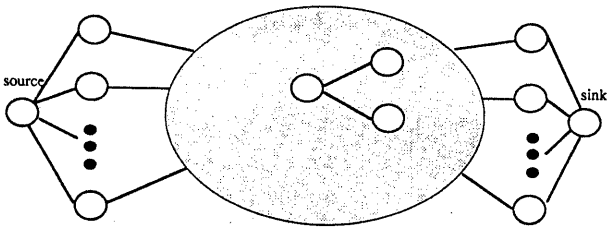
以下では各枝のパスへの影響度を求めるアルゴリズムについて述べる。

「系」 パスは同じノードをただ 1 度しか通らない。

「定理 1」 ソースノードからノード v へのパスの数は、ソースノードから $(u, v) \in E$ であるすべてのノード u へのパス



(a)回路表現



(b)グラフ表現

図 4: 回路のグラフ表現

の数を足したものである。

「証明」 パスを頂点の系列として表すこととする。$(u,v) \in E$ である u の集合を $\{u_1, u_2, \cdots, u_n\}$ と表すこととする。

ソースノードから各 $u_1, u_2, \cdots, u_n$ のすべてのパスが列挙できたと仮定する。この時、ソースノードからノード v へのパスは容易にすべて列挙できる。すなわち、ソースノードからノード v へのパスは、ソースノードから各 $u_1, u_2, \cdots, u_n$ へのパスに枝 $e(u_i, v)$ 通りノード v へ至るパスであるため、各パスの系列に v を追加することによりソースノードからノード v へのパスは容易に列挙できる。

すなわち、ソースノードからノード v へのパスの数は、ソースノードから $(u,v) \in E$ であるすべてノード u へのパスの数を足したものである。

□

「定理 2」 シンクノードからノード v へのパスの数は、シンクノードから $(v,w) \in E$ であるすべてのノード w へのパスの数を足したものである。

「証明」 省略

「定理 3」 ノード v を通過するパスの数は、ソースノードからノード v へのパスの数とシンクノードからノード v へのパスの数を掛け合わせたものである。

「証明」 パスを頂点の系列として表すこととする。$(u,v) \in E$ である u の集合を $\{u_1, u_2, \cdots, u_n\}$ と表すこととする。また、$(v,w) \in E$ である w の集合を $\{w_1, w_2, \cdots, w_m\}$ と表すこととする。

ソースノードから各 $u_1, u_2, \cdots, u_n$ のすべてのパスが列挙できたと仮定する。また、シンクノードから各 $w_1, w_2, \cdots, w_m$ のすべてのパスが列挙できたと仮定する。この時、ソースノードからシンクノードへのパスは容易にすべて列挙できる。すなわち、ソースノードからシンクノードへのパス

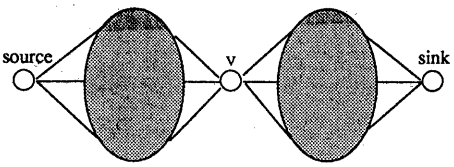


図 5: ノード v を通過するパスの数

は、ソースノードから各 $u_1, u_2, \cdots, u_n$ へのパスに枝 $e(u_i, v)$ を通りノード v へ至り、枝 $e(v, w_i)$ を通り、各 $w_1, w_2, \cdots, w_m$ に至り、各 $w_1, w_2, \cdots, w_m$ からへシンクノードへのパスであるため、各ソースノードから各 $u_1, u_2, \cdots, u_n$ のすべてのパスの系列に v を追加し、各 v からシンクノードに至るすべての系列を加えれば良い。

即ち、ノード v を通過するパスの数は、ソースノードからノード v へのパスの数とシンクノードからノード v へのパスの数を掛け合わせたものである。

□

定理 1,2,3 より各ノードを通過するパスの数は、ソースノードから各ノードまでのパスの数、シンクノードから各ノードまでのパスの数を求め、この値を各ノードで掛け合わせることにより求めることができる。以下にアルゴリズムを示す。

「定理 4」 アルゴリズム count() は、ソースノード (シンクノード) から各ノードまでのパスの数を求める。

「証明」 略

定理 4 より count() は、シンク (ソース) ノードから各ノードへのパスの数を求める。また calculate() は定理 3 より各ノードを通過するパスを求める。よって、アルゴリズム count_path() は、各ノードを通過するパスの数を求める。アルゴリズム count_path() の計算複雑度はノードの数を $|V|$、枝の数を $|E|$ とした場合、$O(|V|+|E|)$ である。

クリティカルアーク算出アルゴリズム

```
procedure count_path(source,sink)
{

  /*sink から各ノードへのパスの数 */

    for(すべての v ∈ V){

     lab(v)=num.output(v)=0;
     }
  num.output(sink)=1;
  i=0;
  count(source,num.output);

  /*source から各ノードへのパスの数 */

    for(すべての v ∈ V){
```

```
        lab(v)=num.output(v)=0;
        }
    num.input(source)=1;
    i=0;
    count(sink,num.input);

    /* 各ノードを通過するパスの数 */

      for(すべての v ∈ V){

       lab(v)=0
       }
    i=0;
    calculate(source);
}

procedure count(v,num)
{
    i=i+1;
    lab(v)=i;

      for ((v,u) ∈ E であるすべての頂点 u に対して){

       if( lab(u)=0 ){
        num(v)+=count(u,num);
        }
        else{
        num(v)+=num(u);
        }
    }
    return(num(v));
}


procedure calculate(v)
{
    i=i+1;
    lab(v)=i;
    cal(v)=num.input(v) * num.output(v);

      for ((v,u) ∈ E であるすべての頂点 u に対して){

       if( lab(u)=0 ){
         calculate(u);
         }
    }

}
```

4 余裕度の分配処理

ネットの余裕度を求める処理を余裕度の分配処理と呼ぶ。以下に、フローを示す。

余裕度分配処理フロー

1. 各ネットの予想配線長を求める。
2. エルモアの式により配線遅延を求める。
3. 遅延解析 [2] を行なう。
4. ゼロスラックアルゴリズム [3] によりスラック値の分配を行なう。

まず、ネットの仮想配線長を求める。これは、配置がまだ、行なわれていない場合は、各ネットのピンペア数にある定数を掛け合わせることにより、配置処理の途中では、ブロックは現在属する領域の中心にあるものと仮定して、疑似的にスタイナー木を作成することにより仮想配線長を求める。その後、エルモアの式により各ネットの配線遅延を求める。次に、遅延解析 [2] を行ない各ネットのスラック値を求め、ゼロスラックアルゴリズム [3] によりスラックの分配処理を行ない、各ネットの余裕度を求める。余裕度が負の場合は、仮想配線長よりも余裕度分ネットを短くしなくては、遅延を満足しないことを表し、余裕度が正の場合は、仮想配線長よりも余裕度分ネットを長くしても遅延を満足することを表す。

5 遅延考慮クラスタリング

初期クラスタリング、階層クラスタリングにおける遅延考慮の方法について述べる。

遅延を満足する配置結果を得るために、余裕度が小さい値、特に負のネットに接続するブロックは、なるべく近くに配置する必要がある。また、余裕度が小さな値で、パスへの影響度が高いネットは特に近くに配置する必要がある。そこでクラスタリング処理において、ネットの結合度に加え、ネットの余裕度とパスへの影響度を考慮することにより、クラスタリングするブロックを決定する。

まず、ブロック i,j のクラスタリングコスト $cost(i,j)$ を以下の様に定義する。

$$cost(i,j) = \alpha \times connect(i,j) + \beta \times \sum_{l \in E1} (afford(l) \times (-1) \times path(l))$$

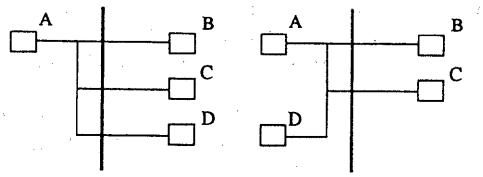
ここで、$connect(i,j)$ は、ブロック i, ブロック j 間のネットの結合度を表す。また、$E1$ は、ブロック i, ブロック j 間を接続するネットの集合を表し、$afford(l)$ は、ネット l の余裕度、$path(l)$ は、ネット l のパスへの影響度を表す。また、α, β は任意の定数。

このクラスタリングコストをすべてのブロックの組に対して計算し、この中で最もコストの高いブロックをクラスタリングする。即ち、クラスタリング目標数とクラスタリング面積和制限 (ペア交換時にブロックの大きさが同じ方が交換しやすいため) が与えられるとする。まず、すべてのブロック組 i,j に対して、もしブロック i,j の面積和がクラスタリング面積和制限を満たしていれば、クラスタリングコストを計算する。次に、この中で、クラスタリングコストが最も大きいもののクラスタリングを行ない、再度すべてのブロックの組に対して、もしクラスタリング面積和制限を満たしていれば、クラスタリングコストを計算する。この処理を、クラスタ数がクラスタリング目標数に達するまで行なう。

6 遅延考慮ペア交換

各階層でペア交換を行なう場合の遅延考慮の方法について述べる。

カットラインを横切るネットの数について考える。あるブロックにつながるある 1 本のネットに注目する。ブロックが移動することにより、このネットに接続するブロックがすべて同じ領域に存在する様になれば、カットラインを横切るネットの数が 0 本なり、この交換によりカット数が減少したことになる。しかし、ネットには複数のブロックが接続しており、ブロックが 1 つ移動することによりカットラインを横切るネットの数が減



(a)ネット形状1　　(b)ネット形状2

図 6: 形状コスト

少することは少ない。そこで、ネットに接続するブロックの位置により、どれだけカットラインを横切るネットの数を減少させる方向にブロックが移動したかを評価することを考える。例えば、カットラインを横切るネット数が同じでも、ネットに接続するブロックの位置としては、図 6(b) より図 6(a) の方が望ましい。なぜなら、図 6(a) では、1 つブロックを移動させることによりカットラインを横切るネットの数を 0 本とすることができるからである。そこでこれを形状コストと呼び、以下の様に定義する。

$$form(i,j) = 1 - \frac{pin(i,j)}{tpin(i)}$$

ここで $tpin(i)$ は、ネット i に接続するピンの総数を表す。また、$pin(i,j)$ は、ネット i のピン j を現在いる領域の反対側の領域に移動させた後の移動元領域に存在するネット i のピン数を表す。図 6(b) の例におけるブロック D の形状コストは 1-1/4=3/4 である。

遅延考慮ペア交換では、各階層でカットラインを横切るネットの数を最小とすると同時に、カットラインを横切る余裕度が負 (小さな値) のネット数、特に、余裕度が負でパスへの影響度が高いネットの数を最小にすることを目的としカットラインの両側からブロックを 1 つずつ取り出し、ブロックの交換を行なう。

即ち、ブロック i とブロック j を交換する場合のコスト $exchange(i,j)$ は、ブロック i とブロック j に接続するネットの集合を $E2$ とすると、以下のように定義される。

$$exchange(i,j) = \sum_{l \in E2} \Big(\sum_{i \in P(l)} \alpha \times form(l,i) \times (1 + afford(l) \times (-1) \times path(l)) \Big)$$

ここで、$afford(l)$ は、ネット l の余裕度を表し、$form(l,i)$ は、ネット l のピン i の形状コストを表し、$path(l)$ は、ネット l のパスへの影響度を表す。また、$P(l)$ は、ネット l に接続するピンの集合を表す。α は任意の定数。

$exchange(i,j)$ をすべてのブロックの組に対し計算し、最もコストの高いものが交換の対象となり、交換が行なわれる。

7 計算機実験結果

提案した手法を、EWS4800/350(95MIPS) 上に c 言語でプログラミングし、計算機実験を行なった。[1] で遅延を満足することができなかった例題に対し、配線長の増加なしに遅延を満足する結果を得ることができることを確認した。結果を表 1,2 に示す。表中の illegal path は、制約を満足できなかったパス数 / 制約パス数を表す。また、illegal arc は、パス解析後スラック値が負のネットの数 / 制約パス数上のネットの数を表す。

また、パスへの影響度コストの効果を計算機実験により評価した。遅延を考慮しない場合、遅延を考慮したが、パスへの影響度コストは考慮しない場合、パスへの影響度コストを含め考慮した場合の結果を表 3 に示す。表 3 の illegal start pin は、制約を満足することができなかったパスのスタート FF の数 / 制約パスのスタート FF の数を表す。この結果より、パスへの影響度コストが有効に作用していることを確認した。また、遅延を考慮したが、パスへの影響度コストは考慮しない場合は、illegal start pin 数が減少しているのに illegal path 数が多く残っていることがわかる。これは、あるネットが長く残ってしまい、このネットを通過する多くのパスが残ったものと考えられる。表 4 にメディアディレイの値の変化を示す。パスへの影響度コストを含め遅延を考慮した場合のメディアディレイ値の最大値、最長値、平均値は、遅延を考慮しない場合にくらべ減少していることがわかる。

表 1: 遅延考慮配置評価 (データ 1)

	遅延あり	遅延なし
処理時間	161s	166s
仮想配線長	650408880	674471540
density	(800,633)	(841,645)
illegal path	0/411	357/411
illegal arc	0/1128	213/1128

表 2: 遅延考慮配置評価 (データ 2)

	遅延あり	遅延なし
処理時間	148s	147s
仮想配線長	635384740	642235420
density	(652,615)	(576,752)
illegal path	0/236	55/236
illegal arc	0/378	48/378

表 3: 遅延考慮配置評価 (データ 3)

	遅延考慮なし	影響度なし	影響度あり
処理時間	123s	147s	145s
仮想配線長	674471540	649449480	624990560
density	(841,645)	(735,718)	(778,698)
illegal start pin	15/47	3/47	0/47
illegal path	352/411	311/411	0/411
illegal arc	124/1128	75/1128	0/1128

8 おわりに

本稿では、階層クラスタリング手法を用いた配置アルゴリズムに対し、コストにより遅延を考慮する方法について述べた。特に、各ネットのパスへの影響度を求めるアルゴリズムについ

表 4: 遅延考慮配置評価 (データ 3)

	MAX	MIN	AVERAGE
遅延なし	791787	3775	73091
影響度なし	791787	3255	72953
影響度あり	310375	2114	58104

て述べた。計算機実験の結果、本アルゴリズムが有効であることを確認した。

9 謝辞

貴重な助言をくださった日本電気 (株)C&C システム研究所の中村氏に深く感謝します。また、遅延解析部のプログラムを提供して頂いた日本電気 (株) 第 1 コンピュータ事業部多和田氏に深く感謝します。

参考文献

[1] 袖 美樹子, 枝広 正人, 吉村 猛, " ゲート敷き詰め型ゲートアレイ用配置アルゴリズム ", 第 4 回 回路とシステム軽井沢ワークショップ, pp.367–372, (1991).

[2] Robert B. Hitchcock, Gordon L. Smith, David D. Cheng, "Timing Analysis of Computer Hardware", *IBM J. Res. Develop. vol 1 January 1982*, pp.100–105, (1982).

[3] Peter S. Hauge, Ravi Nair, Ellen J. Yoffa, "Circuit Placement for Predictable Performance", *Proc. ICCAD-87.*, pp.88–91, (1987).

[4] Habib Youssef, Eugene Shragowitz, "Timing Constraints for Correct Performance", *Proc. ICCAD-90.*, pp.24–27, (1990).

[5] Wing K. Luk, "A Fast Physical Constraint Generator for Timing Driven Layout", *Proc. 28th DA Conf.*, pp.626–631, (1991).

[6] Ting-Hai Chao, Yu-Chin Hsu, "Rectilinear Steiner Tree Construction by Local and Global Refinement", *Proc. ICCAD-90.*, pp.432–435, (1990).

[7] Wilm E. Donath, Reini J. Norman, Bhuwan K. Agrawal, Stephen E. Bello, Sang Yong Han, Jerome M. Kurtzberg, Paul Lowy, Poger I. McMillan, "Timing Driven Placement Using Complete Path Delays", *Proc. 27th DA Conf.*, pp.84–89, (1990).

[8] Ichiang Lin, David H. C. Du, "Performance-Driven Constructive Placement", *Proc. 27th DA Conf.*, pp.103–106, (1990).

[9] Arvind Srinivasan, Kamal Chaudhary, E. S. Kuh, "RITUAL:A Performance Driven Placement Algorithm for Small Cell IC's", *Proc. ICCAD-91.*, pp.48–51, (1990).

[10] Ren-Song Tsay, Juergen Koehl, "An Analytic Net Weighting Approach for Performance Optimization in Circuit Placement", *Proc. 28th DA Conf.*, pp.620–625, (1991).

[11] Masayuki Terai, Kazuhiro Takahashi, Koji Sato, "A New Min-cut Placement Algorithm for Timing Assuarance Layout Design Meeting Net Length Constraint", *Proc. 27th DA Conf.*, pp.96–102, (1991).

[12] Masato Edahiro and Takeshi Yoshimura, "New Placemnet and Global Routing Algorithms for Standard cell Layouts", *Proc. 27th DA Conf.*, pp.642–645, (1990).

[13] 伊理正夫, 白川功, 梶谷洋司, 篠田庄司, " 演習グラフ理論 ", コロナ社.