設 計 自 動 化 77－9
（1995. 10. 19）

演算器系論理回路における等遅延配置アルゴリズム

中村 猛　小島 章道　古木 勝也

日本電気(株)
〒211 神奈川県川崎市中原区下沼部 1753
Tel: (044) 435-1518　Fax: (044) 435-1888
Email: t-naka@lsi.tmg.nec.co.jp

あらまし　本論文ではデータ信号が順次伝播するような演算器系論理回路、特に加算、乗算等のレイアウト設計において、論理段毎の配線長のばらつきを小さくした、等遅延配置アルゴリズムについて提案する。

ここで提案するアルゴリズムは、論理演算回路を構成するファンクションブロックに対して、伝播するデータ信号の流れに沿って順次トレースを行い、ビット毎の系列、論理段を決定して相対位置を求める。この配置方法は、論理段毎の配線経路パターンを一様にし、配線長のばらつきを小さくすることにより遅延差の小さい配置の実現を目的とする。提案したアルゴリズムに対しての評価を行い、有効性を示す。

キーワード　LSI CAD、演算器、論理回路、配置、レイアウト

A Timing Driven Placement Algorithm in Arithmetic Logic Circuits

Takeshi Nakamura　Akimichi Kojima　Katsuya Furuki

NEC Corporation
1753 Shimonumabe, Nakahara-ku, Kawasaki, Kanagawa 211, Japan
Tel: (044) 435-1518　Fax: (044) 435-1888
Email: t-naka@lsi.tmg.nec.co.jp

Abstract　This paper proposes a placement algorithm mainly dedicated to datapath elements and minmized a difference of delay time between bits.

This algorithm decides a relative placement by tracing functional blocks in logic circuits based on the data signal path utilizing bit width and depth of logic stages.

The placement using this algorithm shows that a difference of wire length in every logic stage is smaller than current auto placement and that this method is effective.

key words　LSI CAD, Arithmetic, Logic Circuits, Placement, Layout

1 はじめに

LSIのレイアウト設計における自動配置の多くはネットのカット数を最適化するMincut手法を用いて行われている。Mincut手法は、ネットのカット数を最小になるように回路を分割し、最適な配置を行う。

Mincut手法を用いた場合、ネットのカット数を最適にすることで、ネットの混雑度が一様で、しかもネット結合の強いセル同士が近くに配置され、配線長も最適なものとなる。

しかし、近年のLSIの動作高速化によって、データ信号が順次伝播するような論理演算回路の高速なデータ処理が重要な課題となっている。そのために、Mincut手法を用いたカット数を最適にする手法だけではなく、配線長、およびタイミングの最適化、各ビット毎の同一信号伝播経路パターンにおける同一伝播遅延も考慮した配置が要求される。

Mincut手法と配線長やタイミングを考慮した手法として、クラスタリングやタイミングドリブン、配置改善等を用いた手法がいくつか提案されている[3][4][5][6]。また、論理演算回路の配置を、複数の部分回路群に分解して、互いを結ぶ総配線長を短くするような手法[7]も提案されているが、いずれも、演算するデータのビットの並びや、演算を行うための論理回路の順序、演算データの流れを考慮した、論理演算回路を構成する各ファンクションブロックの配置を行っていない。そのため、各信号ネット毎の配線長、タイミングは最適化されるが、データが伝播される論理段毎の配線長に『ばらつき』がどうしても生じてしまい、その結果、各処理ビットの論理段毎の遅延に差が出てしまう。この論理段毎の配線長の『ばらつき』を抑えることによって、ビット系列毎の信号伝播遅延差が抑えられ、等遅延配線が可能となれば、その結果として高速な演算回路のレイアウトが可能であると考えられる。

そこで本稿では、演算データが順次伝播し、しかもそのタイミングが重要になる論理演算回路のレイアウト設計において、論理段毎の配線長『ばらつき』を少なくすることにより、等遅延配置を可能とするファンクションブロックの相対位置決定アルゴリズムを提案する。また、その効果について報告する。

以下、第2章では一般的な論理演算回路構成の説明を、第3章では、本稿のレイアウト設計手順、および提案するファンクションブロックの相対位置決定アルゴリズムを説明し、第4章では適用評価結果を示し、第5章でその考察述べ、第6章でまとめを行う。

2 論理演算回路

本報告で取り扱う論理演算回路の一例を図1に示す。

演算器系回路は、インバータ、アンド、オア等のファンクションブロックを用いて構成されており、しかも演算するデータのビット数の幅方向、論理段数の深さ方向にそれぞれ規則的な回路の繰り返し構造をとることが一般的である。

また、各ビット毎のデータ信号は、入力端子よりファンクションブロックを順次伝播されて各ビット毎に出力される。

このような特徴をもった論理演算回路は、同じビットの系列に属し、伝播信号に関して入力と出力の関係にあるブロック同士を論理段の深さ方向に隣接して配置を行う。この様な配置を行うことで、同一系列で互いを結ぶレイアウト上の配線長を短くすることが可能である。さらにこのことに加え、ネットの結合強度、ネットの混雑度に注目した配置ではなく、ビット毎に順次伝播される信号をデータの流れに従い、ファンクションブロックをトレースすることで、規則性のある配置を行うことが出き、論理段毎のデータ信号配線経路パターンが一様になる。

本配置を実現することで、ビット毎に順次伝播処理される論理段毎のデータ信号配線長を短く、『ばらつき』が少なくなることにより、論理段毎の伝播遅延が小さく、かつ一様な等遅延配置が可能と考えられる。

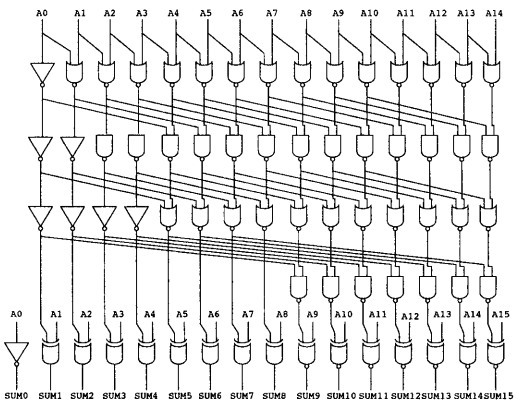


図1 論理演算回路の一例

3 レイアウト手法

3.1 レイアウト設計手順

今回提案する論理演算回路における等遅延配置を可能とするファンクションブロックの相対位置決定アルゴリズムを使用したレイアウト設計の概要を図2に示す。また、レイアウト設計の各ステップは以下の様になっている。

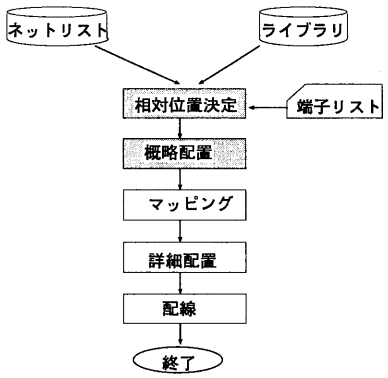


図2 レイアウト設計手順

1. 相対位置決定

   今回提案するアルゴリズムを用いて、論理演算回路のファンクションブロックをデータ信号の流れに添ってトレースを行い、ビット系列、論理段を決定する。

   入力はネットリスト、ライブラリ、出力端子名リストである。出力端子名リストは、端子に接続されるネット名として定義することも可能である。

   論理演算回路のデータ信号における各ファンクションブロックのトレースは、出力端子名リストで定義された出力端子名、あるいはネット名の順序で行うため、出力端子名、あるいはネット名は上位ビットから下位ビットの順序で定義を行う。

   トレースによって決定されたビット系列と論理段より各ファンクションブロックの相対位置を決定する。

   このとき、各ファンクションブロックの位置は、ビット系列と論理段毎に揃える様に配置される。ここで、系列幅は論理段に同じ系列を持つ各ファンクションブロックの合計幅とする。

   本アルゴリズムについては、次項で述べる。

2. 概略配置

   相対位置情報と、実際に配置を行う領域により相対位置情報を位置情報に変換する概略配置を行う。位置情報は、配置領域の左下座標と相対位置情報の左下位置を合わせることにより整合を行う。

3. マッピング

   ゲートアレイの様に下地を予め定義されている配置では、位置情報と下地情報よりマッピングを行いながらビット系列や論理段を守りながら絶対位置の配置を行う。

   また、位置情報よりビット系列、論理段を守りながら配線性、配置禁止情報を考慮して配置改善を行う。

   1チップレベルを考えると他のファンクションブロックが入り込むと特性悪化の原因となるので、論理演算ブロックを配置後、領域に対して配置禁止の発生を行う。

4. 配線

   配置情報に対して概略配線、詳細配線を行う。

   ファームマクロタイプ†として1チップレベルでのレイアウト設計における配線を実行する場合、他の部分の配線に影響を受け、迂回などによる特性悪化の可能性が生じる。このことを解決するために優先配線等が効果的であると考えられる。

3.2 相対位置決定方法

今回提案する等遅延相対配置決定アルゴリズムは、指定した端子、あるいは端子に接続するネットから論理演算回路の伝播データ信号をトレースして行き、トレースにより認識された各ファンクションブロックに、ビット系列数と論理段数の情報を付加し、これをレイアウトする際の2次元配置の行、列に対応した相対位置情報を算出する。

ここでは論理段毎にビット系列を決定していく方法とビット系列毎に論理段を決定していく方法の2つのアルゴリズムを提案する。

3.2.1 Trace Circuit アルゴリズム 1

論理段毎にビット系列を決定していく方法のアルゴリズムは以下の通り。

†ファームマクロタイプとは、相対位置情報を持ち、絶対位置情報を持たないマクロをいう。

Algorithm1 *Trace Circuit*

```
j=1 ;   /* Initial stage number */
/* Phase I */
S_j=ø;  /* Set of cells belonged stage j */
for each cell c_i connected i bit out put terminal O_i
            /* decrese i from higher bit */
  Serige( c_i )=i ; /* Bit serige number of cell c_i */
  Stage( c_i )=j ; /* Logic stage number of cell c_i */
  s_{i,j}=s_{i,j} ∪ { c_i } ;
    /* Set of cells belonged serige i and stage j */
  if c_i is not marked
    S_j=S_j ∪ { c_i } ;
    c_i mark ;
  endif
endfor
/* Phase II */
While set S_j is not empty
  for each set s_{i,j}
    for each cell c_i belonged s_{i,j}
      for each fan-in cell ĉ_k of c_i
        if ĉ_k is not marked
          Serige( ĉ_k )=i ;
          Stage( ĉ_k )=j+1 ;
          ĉ_k mark ;
        else
          Serige( ĉ_k )=i ;
          Stage( ĉ_k )=MAX( Stage( ĉ_k ) , j+1 ) ;
        endif
        s_{i,j}=s_{i,j} ∪ { ĉ_k } ;
        if ĉ_k is not marked
          S_j=S_j ∪ { ĉ_k } ;
          ĉ_k mark ;
        endif
      endfor
    endfor
  endfor
  j=j+1 ;
endwhile
```

End *Trace Circuit*

例えば、図3の論理回路のトレースを考えた場合、アルゴリズム1のPhase Iの処理により、A3(3,1)††、A2(2,1)、A1(1,1)、A0(0,1)、$s_{3,1}$={A3}、$s_{2,1}$={A2}、$s_{1,1}$={A1}、$s_{0,1}$={A0}、S_1={A3,A2,A1,A0}となる。トレースは必ず上位ビットから下位ビットの順序で、出力端子方向から行う。

††(系列番号，論理段)とする。

Phase IIの処理に移り、まず$s_{3,1}$に属するA3のfan-inファンクションブロックB3、B2に対しては、B3(3,2)、B2(3,2)となる。次に、$s_{2,1}$に属するA2のfan-inファンクションブロックB2、B1に対しては、B2(2,2)、B1(2,2)となるが、B2については上位ビットA3の処理でマークされているのでB2(2,2)となる。以下、同様に論理段jに属するファンクションブロック集合S_jが空集合になるまでトレースを繰り返す。結果として各ファンクションブロックの相対位置は表1のように決定される。

上記Algorithm1は出力端子方向からのトレース方法を示したものであるが、入力端子方向からのトレースも可能である。

処理として、Phase Iではiビット入力端子I_iに接続たれたファンクションブロックc_iを下位ビットから上位ビット方向にトレースを行う。次にPhase IIに移つり、c_iのfan-outファンクションブロック$\hat{c}_k$に対しトレースを行うという処理を論理段の深さ方向に最終論理段まで繰り返し行い、ビット系列、および論理段を決定する。

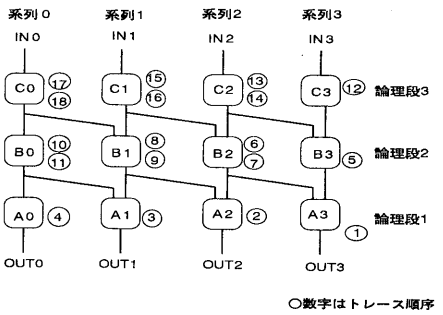


図3 模擬論理演算回路1

表1 相対位置1

C0(0,3)	C1(1,3)	C2(2,3)	C3(3,3)
B0(0,2)	B1(1,2)	B2(2,2)	B3(3,2)
A0(0,1)	A1(1,1)	A2(2,1)	A3(3,1)

3.2.2 Trace Circuit アルゴリズム2

ビット系列毎に論理段を決定していく方法のアルゴリズムは以下の通り。

Algorithm2 *Trace Circuit*

```
for each cell c_i connected i bit out put terminal O_i
            /* decrese i from higher bit */
  j=1 ;   /* Initial stage number */
  Serige( c_i )=i ; /* Bit serige number of cell c_i */
  Stage( c_i )=j ; /* Logic stage number of cell c_i */
  if fan-in cell ĉ_k of c_i exist
    for each cell ĉ_k
      TRACE ( c_i , Serige( c_i ) , Stage( c_i ) )
      /* TRACE is a subfunction defined below */
    endfor
  else
    next c_i do ;
  endif
endfor
subfunction TRACE ( c , i , j )
  for each fan-in cell ĉ of c
    if ĉ is not marked
      Serige( ĉ )=i ;
      Stage( ĉ )=j+1 ;
      ĉ mark ;
    else
      Serige( ĉ )=i ;
      Stage( ĉ )=MAX( Stage( ĉ ) , j+1 ) ;
    endif
    if fan-in cell e of ĉ exist
      for each cell e
        TRACE ( e , Serige( e ) , Stage( e ) )
      endfor
    else
      next ĉ do ;
    endif
  endfor
end subfunction TRACE
```

End *Trace Circuit*

アルゴリズム２の処理による論理演算回路のトレース順序は図４となる。アルゴリズム１と比較するとトレース回数は多くなるが、アルゴリズム１と同じ相対位置情報が得られる。

上記 Algorithm2 は出力端子方向からのトレース方法を示したものであるが、Algorithm1 と同様に入力端子方向からのトレースも可能である。

処理として、i ビット入力端子 I_i に接続たれたファンクションブロック c_i を各ビット系列毎に fan-out ファンクションブロック $\hat{c}_k$ に対して論理段の深さ方向に最終論理段までトレースを行うという処理を下位ビットから上位ビットまで繰り返し行い、ビット系列、および論理段を決定する。

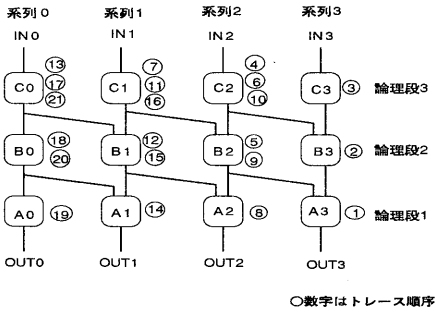


図４ 模擬論理演算回路２

3.2.3 相対位置決定

前述のトレースによって得られたビット系列と論理段の情報を基に、各ファンクションブロックを２次元的に配置する位置を決定する。

各ビット系列幅は、それぞれの系列内に属するファンクションブロック幅の最大幅に加え、配線性を考慮して決定する。

また、図３の論理演算回路は、１論理段における１系列に含まれるファンクションブロック数が１以下の場合であるが、図５の論理演算回路のように、１論理段に同一系列をもつファンクションブロックが複数ある場合、論理段毎の系列内に属するファンクションブロック合計幅に加え配線性を考慮してビット系列幅を決定する。

さらに、それら複数のファンクションブロックは、論理段を守りながら、そのビット系列幅内で、配線性を考慮して配置の入れ換え等を行う。

図５の論理演算回路の各ファンクションブロックは図６のように論理段、系列に配置される。

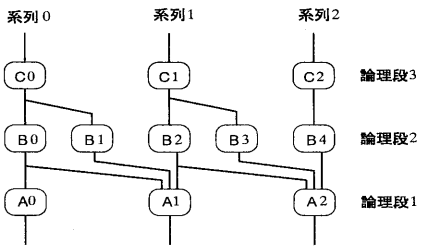


図５ 模擬論理演算回路３

表 2 相対位置 2

C0(0,3)	C1(1,3)	C2(2,3)
B0(0,2)	B1(1,2) B2(1,2)	B3(2,3) B4(2,3)
A0(0,1)	A1(1,1)	A2(2,3)

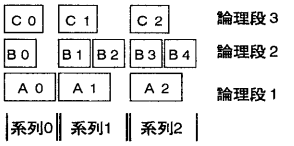


図6 ファンクションブロック配置

4 評価

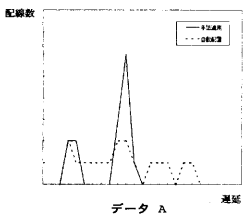
今回提案した論理演算回路のファンクションブロックのトレースを行うアルゴリズムを適用して相対位置を決定してレイアウトを行った評価結果を述べる。

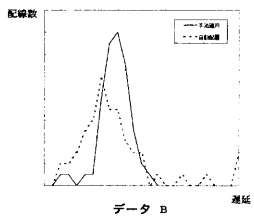
評価データはADDER、INCREMENTER、MULTIPLIRE等を使用した。表3、およびグラフ1はそれらをレイアウトした際の配線長、及び分布の結果である。自動配置は本手法を適用した際の配置面積と同面積で実行をしたものであり、面積小自動というのは、セル占有率を十分高くなるように制約を付けて面積を決定し自動配置したものである。データのうち面積小自動がないものは、本手法適用配置が既にセル占有率が十分高い配置結果になるものである。

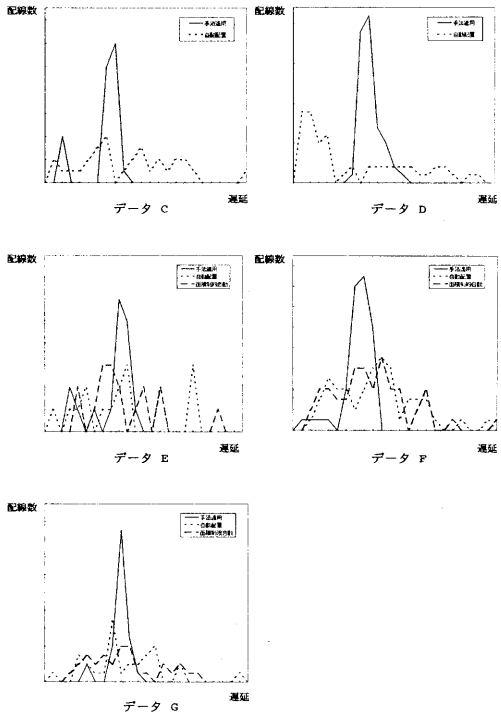
自動配置は Mincut 法を用いたもので行った結果である。

本アルゴリズムを適用して相対配置を行った場合、グラフ1や表3からも分かる通り、自動配置に比べ、論理段毎の配線長のばらつきが小さく抑えており、データ信号の遅延差も少ないという結果が得られた。これは、論理演算回路を規則的に配置を行うことができ、論理段毎のデータ信号配線経路パターンが一様になったためである。

また、総配線長に関してもトレース処理を行うことにより、入力と出力の関係にあるファンクションブロックを系列を守りながら論理段の深さ方向に規則的に隣接して配置を行っているため、互いを接続する配線を短く均等に行うことが可能となり、自動配置による配線結果よりも短く配線されるという結果が得られた。







グラフ 1 遅延分布

表 3 評価結果

データ	実行方法	配線分布 標準偏差	総配線長
A	本手法	34.1 μm	5.630mm
	自動配置	65.0 μm	8.796mm
B	本手法	28.1 μm	68.558mm
	自動配置	110.6 μm	82.298mm
C	本手法	36.4 μm	18.612mm
	自動配置	71.7 μm	19.370mm
D	本手法	46.26μm	287.34 mm
	自動配置	199.38μm	356.395mm
E	本手法	42.6 μm	8.066mm
	自動配置	97.7 μm	15.749mm
	面積小自動	70.4 μm	13.279mm
F	本手法	28.8 μm	75.144mm
	自動配置	104.4 μm	125.196mm
	面積小自動	111.7 μm	105.509mm
G	本手法	15.6 μm	25.248mm
	自動配置	82.3 μm	50.976mm
	面積小自動	47.7 μm	31.660mm

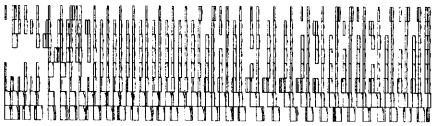


図 7 32 ビット ADDER 回路配置図

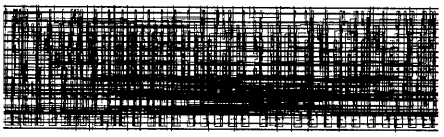


図 8 32 ビット ADDER 回路配線図

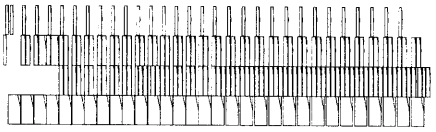


図 9 32 ビット INCREMENTER 回路配置図

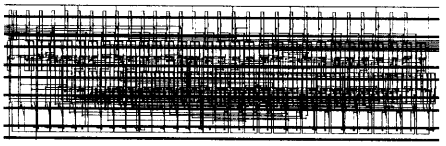


図 10 32 ビット INCREMENTER 回路配線図

5 考察

今回の評価はトレース方向として、出力端子方向からのファンクションブロックのトレースを行った結果である。

下位ビットで、かつ入力端子からのトレースにおいては、系列、論理段は求められるが、図１１の論理演算回路ように下位ビットと上位ビットにおける系列毎の論理段数が異る場合、論理段の少ない下位ビット系列の論理段が入力端子方向論理段にシフトされてしまう。これは先に論理段数の少ない下位ビットがトレースされ、その後に論理段数の多い上位ビットがトレースされるためである。そのために、本来最終論理段にあるべきファンクションブロックが入力方向に詰まった状態に配置され、下位ビットと上位ビットの系列毎における入力端子から出力端子へのデータ信号伝播速度差が大きくなってしまう。それに対して、上位ビットで、かつ出力端子からのトレースにおいては、多い論理段数が先に決定され、各ファンクションブロックが本来の論理段位置に配置されるため、入力端子から出力端子までの各ビット系列毎の論理段数が同一になり、入力端子から出力端子へのデータ信号の伝播速度に差が小さくなる。

配線長、遅延分布について、論理段によっては、配線分布の山が２つに分かれて現れる。これは、同一論理段であっても、次の論理段にあるファンクションブロックに接続される fan-out 数が、それぞれのビット系列におけるファンクションブロックによって異る場合によるものである。そのために論理段毎の出力配線パターンが、fan-out 条件によって２種存在し『ばらつき』分布が２分化された。

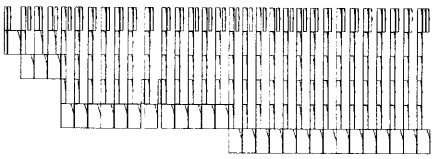


図 11 入力端子トレース回路配置図

6 まとめ

本稿では、論理演算回路における等遅延配置方法の一アルゴリズムを提案した。

データの伝播する方向を考慮してファンクションブロックをトレースし、論理回路に近い配置を行い、配線長のばらつきを抑えることによって、等遅延配置の実現しようとした本アルゴリズムは、評価の結果から有効な一手法だということがいえる。

しかしながら、本アルゴリズムは概略配置を決定するものであり、今回は、その概略配置の状態を保存した、言わば、概略配置を行っただけのレイアウト結果である。この概略配置を生かした詳細配置方法については、まだ十分とは言えず今後の研究課題である。有効な詳細配置方法を行うことで、より高い効果が期待される。

参考文献

[1] G.Odawara, T.Hiraide, and O.Nishina,"Partitioning and placement technique for cmos gate arrays." *IEEE Trans.CAD*,vol.CAD-6.pp.355-363,May 1987.

[2] S.Goto,"An efficient algorithm for the two-dimensional placement problem in electrical circuit

layout." *IEEE Trans. Circ.Syst.* ,vol.CAS-25,pp.12-18,Jan 1981.

[3] S.Dey, F.Brglez, and G.Kedem,"Circuit partitioning for logic synthesis." *IEEE J. Solid-State Circ.* ,Vol.26,pp. 350-363, Mar. 1991.

[4] D.M. Schuler and E.G. Ulrich, "Clustering and linear placement." *Proc. 9th DAC*,pp. 412-419, 1972.

[5] J.Cong and M.Smith, "A Parallel Bottom-Up Clustering Algorithm with Application to Circuit Partitioning in VLSI Design", *Proc. 30th DAC*,pp. 755-760, 1993.

[6] T.Gao, P.M.Vaidya, and C.L.Liu, "A New Performance Driven Placement Algorithm", *Proc. ICCAD*, pp. 44-47, 1991.

[7] Yu-Wen Tsay and Youn-Long Lin,"A Row-Based Cell Placement Method That Utilizes Circuit Structural Properties.", *IEEE Trans.CAD* ,vol. 14 N0.3,Mar 1995.

[8] 品川正行, 田丸啓吉," 改良多重波面型乗算器のモジュールジェネレータ", 信学技法, ICD94-224,pp.53-60,1995