# データパスレイアウトにおける機能ブロック配置の一手法†

2H－2

若林真一　中嶋幸治　境敏親　小出哲士　吉田典可
広島大学工学部

## 1. まえがき

今日，マイクロプロセッサの高性能化・多機能化により，チップ内のデータパスも複雑になっており，人手によるレイアウト設計は困難となってきている．そのため，データパスレイアウトの自動設計手法の開発が強く望まれている．

著者らは文献[2]において，クラスタリングに基づく機能ブロックの配置を行ったあとブロックの対交換によって解の改良を行う手法を提案した．しかし，対交換による改良ではブロック数が増加するにつれて局所解に陥りやすい傾向があった．そこで本稿では，対交換のかわりに３つのブロックを交換するように手法を拡張した．実験の結果，本手法は局所解に陥りにくくなっており，優れた解を得ることがわかった．

## 2. 問題の定式化

通常，データパスのレイアウトは，図１のようにセルが格子状に並ぶ．水平方向には同型のリーフセルがビット数分並び，１つの機能ブロックを構成する．垂直方向には機能の異なるブロックが１列にならぶ．機能ブロック間の配線はいくつかのリーフセルをまたぐことがあるが，リーフセル上を通すことのできる配線数（トラック容量）は限られているため，トラック容量を超える場合にはリーフセルとリーフセルの間にスペーサセルを置き，スペーサセル上に配線しなければならない．このような配線をオーバーフロー配線と呼ぶ．このため，機能ブロックの幅はオーバーフロー配線数に依存して増大する．さらにデータパス全体の幅は，幅最大の機能ブロックに依存するため，データパス部の面積を小さくするためにはオーバーフロー配線の最大数を最小にしなければならない．また総配線長はデータパスの性能に大きく影響するため，総配線長も考慮しなければならない．以上よりデータパスのフロアプランニングはトラックオーバーフローの最大数と総配線長を最小とするような機能ブロックの１次元配置問題としてモデル化できる[1]．本稿で考察する機能ブロック１次元配置問題の定式化を以下に示す．

【入力】１．論理回路（ネットリスト）
　　２．各ブロックのサイズ（高さ×幅）
　　３．各ブロックのトラック容量
【出力】目的関数が最小となる機能ブロックの１次元配置
【目的関数】＜オーバーフロー配線数，総配線長＞の辞書式順序付き２項組

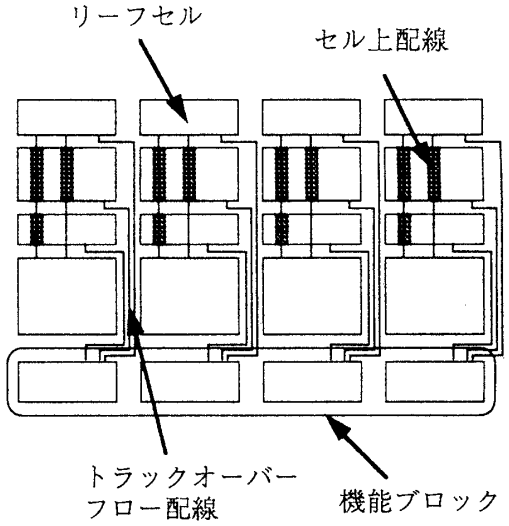


図１　データパスレイアウト

## 3. 配置アルゴリズム

### 3.1. アルゴリズムの概要[2]

以下に機能ブロック１次元配置手法の概要を示す．

ステップ1：機能ブロックの集合をクラスタに分割する．各クラスタに含まれる機能ブロック数があらかじめ決められた数（MAX）を超える場合は，クラスタ内に階層的にクラスタを作る．

†A Functional Building Block Placement Method for Datapath Layout
Shin'ichi WAKABAYASHI, Koji NAKASHIMA, Toshichika SAKAI, Tetsushi KOIDE, Noriyoshi YOSHIDA, Faculty of Engineering, Hiroshima University.

ステップ2：クラスタ単位で目的関数最小な配置をA*アルゴリズムを適用して求める．その後，トップダウンに同様の操作を行い，全ての機能ブロックが1列に並ぶまで繰り返す．

ステップ3：ステップ2で得られた1次元配置に対し，3つのブロックを互いに交換することによる解の改良を行う．

## 3.2. クラスタの構成方法

3.1.で示した1次元配置手法のステップ1において，以下のボトムアップなクラスタリングを行う．

ステップ1.1：すべてのブロックのペアに対しネットによる結合度を計算し，結合度の大きい順に並べる．

ステップ1.2：結合度の大きい順にペアを選び出し，クラスタリングを行う．ただし，トラック容量が極端に小さいブロック（定数CONST以下）は，単独に1つのクラスタとする．

ステップ1.3：最上位の階層のクラスタ数がMAX以下になるまで階層的にステップ1.1，1.2を繰り返す．

## 3.3. A*アルゴリズム

提案アルゴリズムのステップ2において，各クラスタ内でのブロックやクラスタの最適な1次元配置を求めるために人工知能において知られているA*アルゴリズムを用いる．A*アルゴリズムは分枝限定法に基づいており，未探索部分における目的関数の下界値を見積ることにより最適解を失うことなく探索時間を削減する．下界値がコストの真値に近いほど高速に最適解を求めることができる．

## 4. 実験

提案アルゴリズムをサン・マイクロシステムズ社のSPARC station 2上でC言語を用いて実現し，シミュレーション実験を行った．実験データとして，ランダムデータ(a,b,c)と実データ(d)を用いた（表1）．提案アルゴリズム(three)に対し，ブロックの交換数を4に増やした手法(four)，文献[2]の手法(two)，反復改良(PI)法およびシミュレーティドアニーリング(SA)法と比較した結果を表2に示す．ここでoverはオーバーフロー配線数，lengthは総配線長，timeは計算時間(秒)である．PI法は乱数により発生させた10個の初期配置に対し，逐次改良(対交換)を行った場合の解の平均値である．本手法はSA法と同等な解をPI法と同程度の短い計算時間で求め，また文献[2]の手法よりも優れた解を得ており，提案手法の有効性が確認された．また，ブロックの交換数を4つに増やしても解の精度はほとんど変わらず，計算時間が大きくなりすぎるため，交換数を3とすることが実用的であることも確認できた．

表1　データサイズ

| | ブロック数 | ネット数 |
|---|---|---|
| data a | 10 | 15 |
| data b | 30 | 30 |
| data c | 50 | 50 |
| real d | 12 | 135 |

表2　実験結果

| | | over | length | time |
|---|---|---|---|---|
| data a | three | 4 | 2829.8 | 0.2 |
| | four | 4 | 2829.8 | 0.7 |
| | two | 4 | 2816.1 | 0.2 |
| | PI | 4.0 | 2853.4 | 0.3 |
| | SA | 4 | 2816.1 | 0.7 |
| data b | three | 10 | 18752.3 | 38.1 |
| | four | 10 | 16435.9 | 2960.4 |
| | two | 11 | 16885.5 | 11.8 |
| | PI | 11.7 | 17056.1 | 13.4 |
| | SA | 11 | 17681.1 | 385.3 |
| data c | three | 18 | 53594.5 | 502.0 |
| | four | 18 | 42985.4 | 89830.8 |
| | two | 20 | 41891.2 | 95.5 |
| | PI | 21.1 | 43861.5 | 108.1 |
| | SA | 19 | 43216.1 | 1472.4 |
| real d | three | 0 | 12766.3 | 0.2 |
| | four | 0 | 12766.3 | 0.2 |
| | two | 0 | 12766.3 | 1.1 |
| | PI | 0 | 14053.8 | 1.3 |
| | SA | 0 | 14190.0 | 22.8 |

## 5. あとがき

今後の課題として，クラスタリングの改良や総配線長を短縮することが挙げられる．

### 文献

[1] H. Cai, et al. :"A data path layout assembler for high performance DSP circuit," Proc. 27th DAC, pp.306-311 (1990).

[2] 境，他：" データパスレイアウトにおける機能ブロック配置アルゴリズム" ，情処学研報，Vol. 92, No. 83, 92-DA-64, pp. 49-56 (1992).