社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2008－MPS－68 (21)
2008／3／5

PCクラスタにおけるアントコロニー最適化法を用いたFPGA配置ツールの並列化

小森 龍志 †, 尼崎 太樹 †, 飯田 全広 †, 末吉 敏則 †

† 熊本大学大学院自然科学研究科
〒 860-8555 熊本県熊本市黒髪 2-39-1

FPGA 自動設計フローの中で配置処理は最も時間を費やす工程の一つである．近年では，FPGA の性能向上によって実装回路の大規模化が進んでおり，配置処理の高速化は重要な課題となっている．現在，その配置は SA に基づいて行われるのが一般的である．SA の並列化は多く研究されてきたが，SA は逐次性の強いアルゴリズムであり並列化にはあまり適していない．そこで，近年注目されているアントコロニー最適化法を FPGA 配置問題に適用する．アントコロニー最適化法は複数のエージェントが解の探索を独立して行う点で，並列化による高速化などの効果が期待できる．本研究では，PC クラスタ上でアントコロニー最適化法を用いた FPGA 配置ツールの並列化を行い，その評価と検討を行う．

Parallelization of FPGA placement tool using Ant Colony Optimization on PC cluster

Ryushi Komori † Motoki Amagasaki † Masahiro Iida † Toshinori Sueyoshi †

† Graduate School of Science and Technology, Kumamoto University
2-39-1 Kurokami Kumamoto-shi, 860-855 Japan

Placement processing is one of the processes which spends time most in FPGA design automation flow. Accordingly FPGAs have improved circuit performance, the circuit scale that is implemented by FPGAs becomes larger. So the computation time devoted to placement has grown dramatically. Now, the placement based on SA is performed in VPR. SA is a strong algorithm of sequentiality and is not suitable for parallelization. Therefore, Ant Colony Optimization which attracts attention in recent years is applied to FPGA placement problem. In this research, FPGA placement tool which used Ant Colony Optimization on PC cluster is parallelized, and the evaluation and examination are performed.

1 はじめに

工学分野における 2 次割当問題などの組合せ最適化問題は NP 完全問題であるため，短時間で厳密解を得ることは難しい．そのため，近似解法の利用や並列実装，専用回路への実装など高速化のための様々な研究が行われている．近似解法にはシミュレーテッドアニーリング（Simulated Annealing : SA）などがあるが，その中でも蟻の採餌活動を模倣したアントコロニー最適化法（Ant Colony Optimization : ACO）は比較的新しい近似解法でその並列性から近年注目されている．

書き換えが可能なデバイスである FPGA（Field Programmable Gate Array）は近年，性能，集積度，価格等が飛躍的に改善されており，ディジタル民生機器などの量産にも採用されてきている．しかし，開発技術の向上と共に，回路の自動設計に要する時間も増加し，その中でも配置に要する時間は回路規模に対して指数関数的に増加するため，並列処理などによる高速化が期待されている．現在，研究分野で広く用いられている FPGA 配置配線ツールの VPR（Versatile Place and Route）[2] では SA を採用しているが，SA は逐次性が強いため並列化による性能向上が得られにくいと考えられる．そこで本研究は，FPGA 配置問題に対し ACO を並列実装することにより，その並列処理能力を十分に活かすことで配置問題の効率的な解決を目的とする．

以下，2 章でアントコロニー最適化法，3 章で FPGA 配置問題について述べる．4 章で ACO を

配置問題に実装する際に使用したVPRと，実際に実装したACOベースVPR，5章でその並列化法について述べる．6章で評価を示し，最後に7章でまとめとする．

2 アントコロニー最適化法

アントコロニー最適化法は，蟻の採餌行動を模倣したアルゴリズムである．蟻が巣と餌までの間の最短経路を導き出すメカニズムを基に1991年にM. Dorigoらによって最初のACOアルゴリズムであるアントシステム（Ant System : AS）[1]が提案された．

2.1 最短経路生成メカニズム

蟻は餌を探す際，独立して探索を行う．しかし，蟻はフェロモンと呼ばれる道標を通過する経路上に置き，それをもとに他の蟻との情報交換を行いながら，餌と巣穴間の最短経路を生成し，効率良く餌を巣に持ち帰る．

このメカニズムを元にDorigoらが提案したアルゴリズムの簡単な例を以下に示す．

ACOアルゴリズム

1. メモリ・フェロモンの初期化
2. 終了条件まで繰り返し
   - 2-1. 解の生成
   - 2-2. 解の統計・評価
   - 2-3. フェロモンの更新
3. 最良解の出力

3 FPGA配置問題

FPGAとは，配置配線情報を変更することによって自身のハードウェア構成を変更可能なLSIである．最も一般的であるSRAMを用いたLUT（Look-Up Table）に基づくアイランドスタイルFPGAでは，論理ブロックおよびIOブロックが2次元アレイ状に配置され，その周囲を取り囲むように配線領域を持つ．論理ブロックの基本的な構成例としては，LUTとフリップフロップ（flip flop : FF），マルチプレクサからなり，LUTの入力数までの論理を自由に表現することができ，これらの論理ブロック間を信号線を通じて接続することで任意の回路を表現可能となる．配線領域上には，スイッチブロック，および，コネクションブロックというプログラマブルなスイッチが配されており，接続が変更可能である．

FPGA配置問題

図1に示すように，FPGA配置問題は，論理ブロックおよびI/Oブロックとそれらの接続関係が記されたネットリストを入力として，各ブロックをFPGA上のどの位置に割当てるかを決定する問題である．一般的に，FPGAの論理要素は2次元アレイ状に規則正しく整列しており，配置問題は2次割当問題として定式化できる．したがって，実用規模の回路への厳密解法の適用は困難である．

4 VPR

VPRはトロント大学で開発されたFPGA配置配線ツールで，研究等に広く使用されている．その配置を行うアルゴリズム（VPlace）はSAに基づき設計されている．VPlaceの配置フローを図2に示し，そのアルゴリズムを簡単に説明する．

初期配置はランダムに生成する．次に移動元ブロックと移動先のブロックをランダムに選択し，配線長を見積もるコスト関数Cを元に受理判定を行う．Cは回路中の全てのネットiの，バウンディングボックスのマンハッタン距離 "$bb_x(i)+bb_x(i)$" にネットに属するブロック数$numpins(i)$に応じた重みqを加えた値を計算し，それらの総和としてコストCが定義される．

$$C = \sum_{i=1}^{Nnum} q(numpins(i))\,[bb_x(i) + bb_y(i)] \quad (1)$$

バウンディングボックスは，ネットに属する全てのブロックを内包する最小の矩形領域である．

選択した移動（交換）は，以下の式で受理もしくは棄却される．

$$Accepet(\Delta C, T) = \begin{cases} 1 & (\Delta C \leq 0) \\ e^{-\frac{\Delta C}{T}} & otherwise \end{cases} \quad (2)$$

ブロックの選択および受理判定を一定期間繰り返し，終了判定を満たしたら処理を終了する．

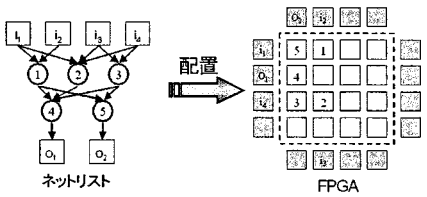

Fig. 1 FPGA配置

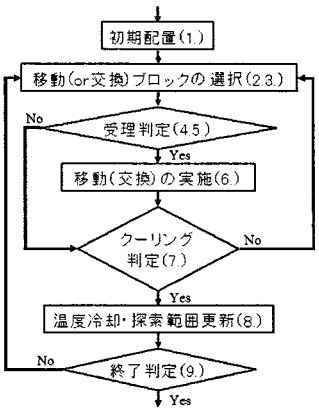

Fig. 2　FPGA 配置フロー

4.1　ACO ベース VPlace アルゴリズム

ACO に変更する際に必要なフェロモン構造の追加，解探索におけるフェロモンの使用，反復・終了判定の変更を主に行った．

初期フェロモン付加

まず，全ブロック間とネットリストから得られた接続情報を元に接続されるブロック間に初期フェロモンを付加する．

初期配置

初期配置はランダムに配置する．

移動（交換）ブロックの選択

ランダムに 1 つのブロックを選び，そのブロックの上下左右のブロックの中で最もフェロモン関係が弱いブロックを移動元ブロックとする．次に，移動先ブロックの選択では，最初に選択したブロックとよりフェロモン関係が強いブロックの探索を数回行い，決定した 2 つのブロックの交換を行う．

解の評価，フェロモン更新

配置結果からコストを計算し，結果が最良の配置に関してフェロモンを強化する．ACO と同様にフェロモン蒸発も行う．

以上を終了判定を満たすまで繰り返し行う．

終了判定

終了判定はユーザが指定した回数行った時点で終了．

5　FPGA 配置の並列化手法

本研究では，作成したプログラムの並列化をするにあたり，領域分割を行うことで並列実装を行う．

領域分割では，対象回路の物理的領域を分割し，各ノードに分割した領域を割当てる．以下に手順を示す．

(1) 全ノードが同一の初期状態でスタート
(2) 領域を分割し，各ノードに割当てる
(3) 各ノードが担当領域内部で解の交換を実施
(4) 定期的に全体で同期を取って結果を結合

以降，終了条件を満たすまで (2)〜(4) を繰り返す．

領域分割による問題は，ブロックの移動可能な範囲が分割した領域内部に限定されることである．常に固定された領域では早期に局所解へと陥ってしまうため，領域の分割方法を切り替え，FPGA 全域に渡る移動を許容する必要がある．したがって，移動可能な範囲が大きく変更されるように図 3 に示すうに水平分割と垂直分割を交互に入れ替える．

6　評価

作成した ACO ベース VPR を実際に以下の評価環境で単体・並列実行し，SA ベース VPR と実行速度，配置コストの比較を行う．

6.1　評価環境

評価回路・並列環境

評価回路には，5 つの MCNC Largest ベンチマーク回路 [3] を用いる．回路の情報を表 1 に示す．また，実験を行った並列環境を表 2 に示す．

6.2　評価結果

作成した ACO ベース VPR を 1，4，8，16 台の計算機を使用して単体・並列実行した際の配置速度向上率，配置コスト増加率をそれぞれ表 3，表 4 に示す．また，SA ベース VPR を並列実行した場合の結果も同様に示す．表 3 の 1 ノードの値と 16 ノードの括弧内の値は実行時間である．

今回作成した ACO ベース VPR の単体実行で SA と同程度の結果を得るには十数倍の時間がか

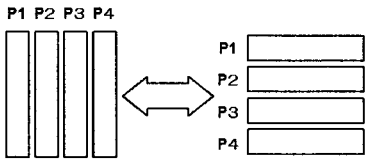

Fig. 3　領域切り替え方法の一例

Table 1　MCNC Largest ベンチマーク回路

回路名	論理ブロック数	I/O ブロック数	配列サイズ
tseng	1,047	174	33x33
apex2	1,878	42	44x44
frisc	3,556	136	39x39
ex1010	4,598	20	40x40
s38417	6,406	135	81x81

かった．今回の比較には約 10 倍程度で終了するように設定した結果を使用する．

表 3 から，ACO ベース VPR では s38417 の場合を除いてノード数に比例して速度が向上している．多少バラつきはあるが，4 つの回路では理想的な速度向上を得ている．s38417 で速度向上が得られなかった理由としては，ブロック数の増加によるフェロモン等に使用するメモリの影響，または計算量の増加が考えられる．SA ベース VPR で速度向上があまり得られていないのは，単体での速度が十分速いために並列化による影響が小さいと考えられる．計算機 16 台ではある程度両 VPR 間の実行時間の差がなくなっている．よって，コストはやや悪いが，ACO の場合実行時間を長くすると改良の余地があることから，計算機の台数を増やすことで実行時間，配置コスト共に改善できると考えられる．

配置コストに関しては，SA ベース VPR では並列実行するとややコストの劣化が生じるが，ACO ベース VPR ではあまり見られず，逆に良くなっている場合が多い．これは探索領域が小さくなったことで解の収束が改善されたためと考えられる．この結果から，並列化による影響がより良く働いている ACO の方が並列実装に適していると考えられる．

7　まとめ

今回の実験において，FPGA 配置問題に ACO の並列実装を行った場合，コスト劣化が起きず高い速度向上率を得ることができた．SA に比べ単体の実行速度は遅いが並列度を上げることにより SA を上回る可能性がある．

今後の課題として，ACO のアルゴリズムを改良することにより単体実行の速度を改善すること，他の並列手法の実装などがあげられる．

参考文献

1) Marco Dorigo, Mauro Birattari, *Ant Colony Optimization*, Technical Report No. TR/IRIDIA/2006-023, 2006.

2) *Vpr and t-vpack: Versatile packing, placement and routing for fpgas package ver 4.30*, `http://www.eecg.toronto.edu/vaughn/vpr/vpr.html`, 1997.

3) S. Yang, Logic synthesis and optimization benchmarks user guide version, Tech. Report, Microelectronics Center of North Carolina, 1991.

4) PC Cluster Consortium, `http://www.pccluster.org/`.

Table 2　HP ProLiant DL380 G3

クラスタコンピュータ	HP ProLiant DL380 G3 × 16 台
CPU	Intel Xeon 2.8GHz × 2
OS	Fedora Core 3
並列プログラミング環境	SCore[4] version 5.8.3

Table 3　速度向上率

	ACO ベース VPR			
ノード数	1(s)	4	8	16
tseng	174	4.8	9.0	14.5(12)
apex2	470	5.1	10.0	17.7(27)
frisc	1928	5.3	10.4	19.1(101)
ex1010	3204	5.5	10.9	19.5(164)
s38417	7086	2.0	2.3	2.5(2791)
	SA ベース VPR			
ノード数	1	4	8	16
tseng	23	1.6	1.7	1.6(13)
apex2	26	1.4	1.6	1.7(28)
frisc	181	1.4	1.6	1.7(94)
ex1010	297	1.7	2.0	2.0(142)
s38417	688	1.7	1.9	2.1(305)

Table 4　コスト増加率

	ACO ベース VPR			
ノード数	1	4	8	16
tseng	128	-0.04%	0.03%	-0.08%
apex2	294	-0.01%	0.01%	0.00%
frisc	640	-0.07%	-0.08%	-0.05%
ex1010	707	-0.01%	-0.00%	-0.02%
s38417	768	0.01%	-0.02%	-0.02%
	SA ベース VPR			
ノード数	1	4	8	16
tseng	118	0.4%	0.4%	1.0%
apex2	275	0.3%	0.4%	0.4%
frisc	554	1.2%	2.0%	2.0%
ex1010	660	0.2%	0.5%	1.0%
s38417	701	0.5%	1.9%	2.8%