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

2005－SLDM－118（13）
2005／1／25

シーケンスペアに基づく交叉専用アーキテクチャの設計

金光　亮輔†，尾藤　彰訓†，吉川　雅弥†，寺井　秀一†

†立命館大学 理工学研究科　〒525-0058 滋賀県草津市野路東 1-1-1

E-mail:　ro002006@se.ritsumei.ac.jp,

あらまし　VLSI レイアウト設計におけるフロアプランニング問題において，シーケンスペアに基づく遺伝的アルゴリズム（GA）を解探索手法に用いたフロアプランニング手法が提案され良好な結果を得ている[3]。しかし GA には多点探索アルゴリズムから来る処理時間の問題が内在する。そのため，GA を実用的なアプリケーションに適用することを考えた場合，専用ハードウェアによる GA が重要になってくる。これまでに GA に基づくフロアプランニングにおいて 20MHz の動作周波数を想定すると，ステップ数による試算で約 160 倍の処理速度の見通しを得ている（ブロック数 250、世代数 10000）。本研究では GA をハードウェア化し，遺伝オペレーションの１つである交叉専用のアーキテクチャを提案する。

キーワード　フロアプランニング，シーケンスペア，GA，交叉，

Architecture for crossover operation based on sequence pair

Ryousuke Knamitsu†, Akinori Bito†, Masaya Yoshikawa†, Hidekazu Terai†

†Ritsumeikan University　1-1-1 Nojihigashi, Kusatu-city, Shiga, 525-0058 Japan

E-mail:　ro002006@se.ritsumei.ac.jp,

Abstract　The floor planning technique that uses GA based on the sequence pair for the solution search is proposed and it obtains an excellent result. However, the problem at the processing time that comes from the multipoint search algorithm exists inside GA. Therefore, GA with hardware is important when thinking GA is applied to the application. In the comparison with software, the prospect of about 160 times the processing speed is obtained (blocks number 250 and generation number 10000).Then, we made GA with hardware and propose the architecture of crossover that is one of the inheritance operations.

Keyword　Floorplanning, sequence pair, GA, Crosssover,

1. はじめに

集積回路の微細化・高集積化に伴い，ゲート遅延に比べて，システム性能に対する配線遅延の影響が大きくなっている[1]。正確な配線遅延は配線が終了しないと分からないが，フロアプラン設計（フロアプランニング）を行ない，機能ブロックの概略配置を決定しておけば遅延見積もりの精度を上げることが出来る。そのためＶＬＳＩ設計におけるフロアプランニングの重要性が増加している[2]。特に，システムＬＳＩや SoC のように，多くの機能を搭載したＬＳＩにおいては，システム性能に対するフロアプランニングの影響は大きい。

このフロアプランニング問題は矩形モジュールを互いに重なることなく，チップ面積を最小化するように配置する矩形パッキング問題に帰着することができる。この問題に対しては、従来から様々なヒューリスティック手法が提案されている[3][4][5]。

近年，この矩形パッキング問題に対する解の表現方法の１つとして，非スライシング構造[3]の表現が可能であるシーケンスペアが提案されている[4]。この解表現を用いると，与えられた矩形の集合に対し，有限な解空間に最適なフロアプランを含むことが保証されるため，この解表現に基づく遺伝的アルゴリズム[6]（Genetic Algorithm:以後 GA と略記する）を解探索手法に用いたフロアプランニング手法が提案され良好な結果を得ている[7]。

GA は，生物進化のメカニズムを工学的にモデル化したアルゴリズムであり，選択・交叉・突然変異の３つの遺伝オペレーションを適用し，世代を進化させてゆく。世代を経るごとに環境に適応した個体の集合が形成されていき，そこから最適解を導くことができる。GA の特徴としては，集団を進化させるという多点探索を行うアルゴリズムであり，探索範囲が広く，局所解に陥りにくいという点が挙げられる。一方，ＧＡには多点探索アルゴリズムから来る処理時間の問題が内在する。そのため，ＧＡを実用的なアプリケーションに適用することを考えた場合，専用ハードウェアによるＧＡが重要となる。

本研究では，ＧＡをベースとした高速なフロアプランニングを実現する専用システムの開発を目的としている。本稿では，2 章でフロアプランニングの概要について述べ，3 章で具体的なフロアプランシステムについて述べる。4 章ではＧＡの遺伝オペレーションの１つである交叉のアーキテクチャについて述べ，5 章ではその評価と考察，最後に 6 章でまとめと今後の課題について述べる。

2. フロアプランニング問題

本論文で取り扱うフロアプランニング問題は，与えられたｎ個の矩形ブロックを互いに重なることなく最小面積の矩形内部に配置する問題である。各矩形モジュールは幅と高さが与えられ，すべてのモジュールを囲む最小矩形をチップと呼ぶ。

個体の表現方法にはシーケンスペア[9]（Sequence pair）を用いた。シーケンスペアは配置の対象となるモジュール名で構成する系列（Γ＋，Γ－）でモジュールの配置を表す方法である。シーケンスペアでは，各モジュール名は（Γ＋，Γ－）の各系列において１個ずつ含まれ，（Γ＋，Γ－）の順列に基づいて任意の２つのモジュールに対して相対的な位置関係を指定する。そのため，どのような配置に対しても，それを表現するシーケンスペア（Γ＋，Γ－）が存在するから，n 個のブロックに対して$(n!)^2$個のシーケンスペアが存在する。従って，それらの中から最適なシーケンスペアを見出す問題がフロアプランであるといえる。

図１にシーケンスペアの表現例を，図２に実際のチップレイアウトを示す。チップレイアウトは，図３に示す水平制約グラフと垂直制約グラフから得ることができる。ブロックの幅，高さをそれぞれ水平，垂直制約グラフの経路の重みとして与えると，それぞれのグラフの最長経路から，チップの幅，高さを求めることができる。

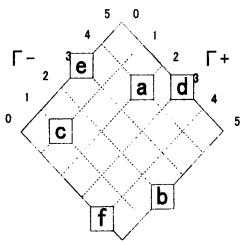

Fig.1. Sequence-pair Expression
図１．シーケンスペア表現

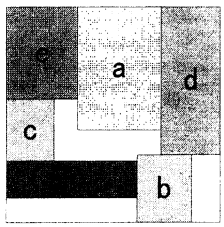

Fig.2. chip layout
図２．チップレイアウト

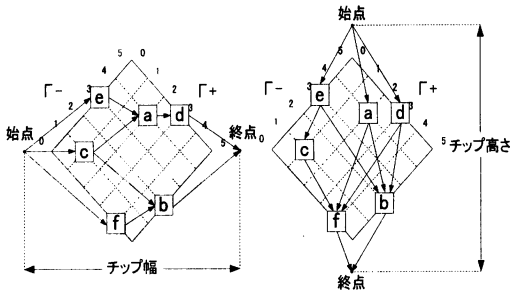

Fig3. The horizontal / vertical constraint graph
図３．水平・垂直制約グラフ

3. フロアプランシステムの構成

構成するフロアプランシステムは，図４に示すように，パラメータの設定や初期個体群の生成など，初期設定の部分をソフトウェアで行い，評価，終了判定，選択，交叉，突然変異といった進化処理オペレーションをハードウェアで行う。ＧＡの世代モデルとしては，全ての個体が一斉に子孫を生成し，次世代集合を作る離散世代モデルと，現在の世代の何割かを入れ替える連続世代モデルのどちらかを選択することが可能となっている。以下に，個体の表現方法，評価方法，各遺伝オペレーション（選択・交叉・突然変異）の詳細を示す。

3.1. 個体表現方法

提案するアーキテクチャでは，解表現にシーケンスペアを用いているため，個体のコーディング方法は図５のように行う。ここでＮはブロック数を表す。まず，ブロックの向き θ（回転）は 0° 90° 180° 270°を２ビットで，ブロックの aspect 比は６ビットで表現している。Γ+，Γ-にはシーケンスペアで表されたブロック相対位置を 8bit×N で表現している。よって実際の個体データは 8bit×3N となる。

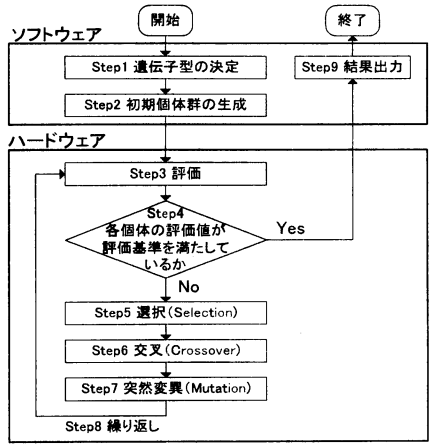

Fig.4. Floorplan system
図４．フロアプランシステム

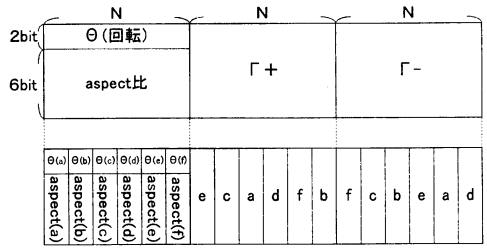

Fig.5. Individual coding method
図５．個体のコーディング方法

3.2. 評価

個体の評価項目としては，面積，チップアスペクト比，クリティカルパス，総配線長を考慮する。クリティカルパスや総配線長の計算に用いる仮想配線長は，図6に示すようなネットの最小外形矩形の半周長を用いる。面積をS，チップアスペクト比をASP，クリティカルパスをLmax，t番目のネットの仮想配線長をnetLength (t)とすると評価値f (t)は下式で表される。ただし，tを個体，a，b，c，dを実数係数とする。

$$f(t)=a\times S+b\times ASP+c\times Lmax+d\times\sum_{t=1}^{allnet}netLength(t)$$

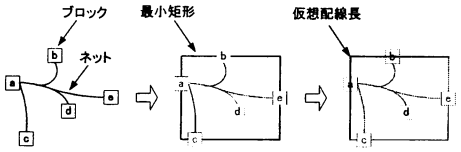

Fig.6. Virtual wiring length
図 6. 仮想配線長

3.3. 選択

選択手法としてはトーナメント戦略，適応度比例戦略，ランク戦略の3つの手法を用いる。また，適応度比例戦略とランク戦略の組合せ以外は，選択の5割をトーナメント戦略で行い，残りの5割をランク戦略で行うといったように，一度の解探索処理で2つの選択手法を併用することができる。さらに，次世代に評価の高い個体（解）を残すことができるエリート保存戦略も行う。以下に，それぞれの選択手法について説明する。

①トーナメント戦略
個体群からランダムに個体を任意の数（あらかじめトーナメントサイズとしてパラメータで与える）だけ選択し，その中で最も評価の高い個体を選択することによって実現する。

②適応度比例戦略
各個体の選択確率として，全個体の評価値（適応度）の総和に対する各個体の評価値の占める割合を用いる手法。各個体の子孫はその評価値に比例した確率で選択されるので，評価の高い個体ほど選択されやすい。評価値が中心角に比例するような円グラフ（ルーレットホイール）と，'1'～'全個体の評価値の積算値'の範囲の値をとる乱数を用いて選択する。

③ランク戦略
評価値によって各個体をランク付けし，あらかじめ各ランクに対して決められた確率（ルーレットホイールの中心角）で子孫を残すことができるようにする手法。この戦略では，評価値そのものの値は重要ではなく，評価値の大小関係だけが考慮される。適応度比例戦略と同様に乱数を生成し，個体を選択する。

例として，個体A，B，C，Dそれぞれの評価値が100，60，400，40で，ランク戦略の選択確率が評価値の良い順から50%，25%，17%，8%である場合，適応度比例戦略，ランク戦略それぞれのルーレットホイールは図7のようになる。

3.4. 交叉

交叉手法には配置依存部分交換交叉（ＰＰＥＸ）を用いた[2]。この交叉手法は，連続した四角形の部分領域(窓領域)をランダムに作成し，窓領域に含まれるブロックを対象として交叉を行う。以下に詳細を説明する。

まず，図8のように親個体1，親個体2に対してそれぞれランダムな窓領域を生成する。窓領域外のデータについては交叉対象外なので，親個体1，親個体2の窓領域外の遺伝子座のデータをそのまま子個体1，子個体2の遺伝子座にコピーする。

Γ＋，Γ－については，親個体1，親個体2の窓領域内の遺伝子座のデータを他方のシーケンスペアにおける出現順序で子個体1，子個体2の遺伝子座にコピーする。回転(θ)・aspect 比については，親個体1，親個体2の窓領域内の遺伝子座のデータを他方の同じ遺伝子座のデータと交換し，子個体1，子個体2のデータとする。

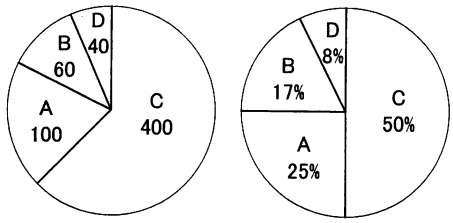

Fig.7. Roulette wheel
図 7. ルーレットホイール
（左：適応度比例戦略　右：ランク戦略）

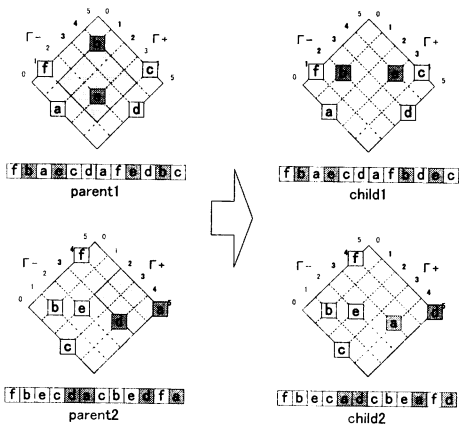

Fig.8. Crossover method
図 8. 交叉手法

3.5. 突然変異

突然変異は，あらかじめパラメータで与えられている選択確率で１つの方法を選択し，行う。各突然変異方法は以下の６種類である。

①アスペクト比一点突然変異
１つのブロックをランダムで選択し，アスペクト比を変異させる一点突然変異法を用いる。アスペクト比はあらかじめ定められた最大値と最小値の間の値となるよう変異させる。
②回転（θ）一点突然変異
１つのブロックをランダムで選択し，θを変異させる一点突然変異法を用いる。
③Γ+、Γ-突然変異
１つのブロックをランダムで選択し，Γ＋とΓ-を変異させる。変異後に重複するブロックについては，重複しないように突然変異対象ブロックの変異前の座標に変異させる。
④Γ+突然変異
１つのブロックをランダムで選択し，Γ＋のみを変異させる。③と同様に、変異後に重複するブロックについても変異させる。
⑤Γ-突然変異
１つのブロックをランダムで選択し，Γ-のみを変異させる。③と同様に，変異後に重複するブロックについても変異させる。
⑥ペア交換
２つのブロックをランダムで選択し，Γ+，Γ-それぞれの値を入れ替える。

4. 交叉アーキテクチャ

今回は，ＧＡの遺伝オペレーションの１つである交叉専用のアーキテクチャを提案し，試作を行った。以下に，アーキテクチャの詳細を示す。

4.1. 回路構成とブロック図

3.4 で述べた交叉方法に基づく交叉回路のブロック図を図９に示す。今回は動作テスト用の回路も挿入しており，回路はメモリ部分と，窓の生成，メモリの制御を担う crossover_core 部，回路の動作テストを行う crossover_test 部に大別できる。

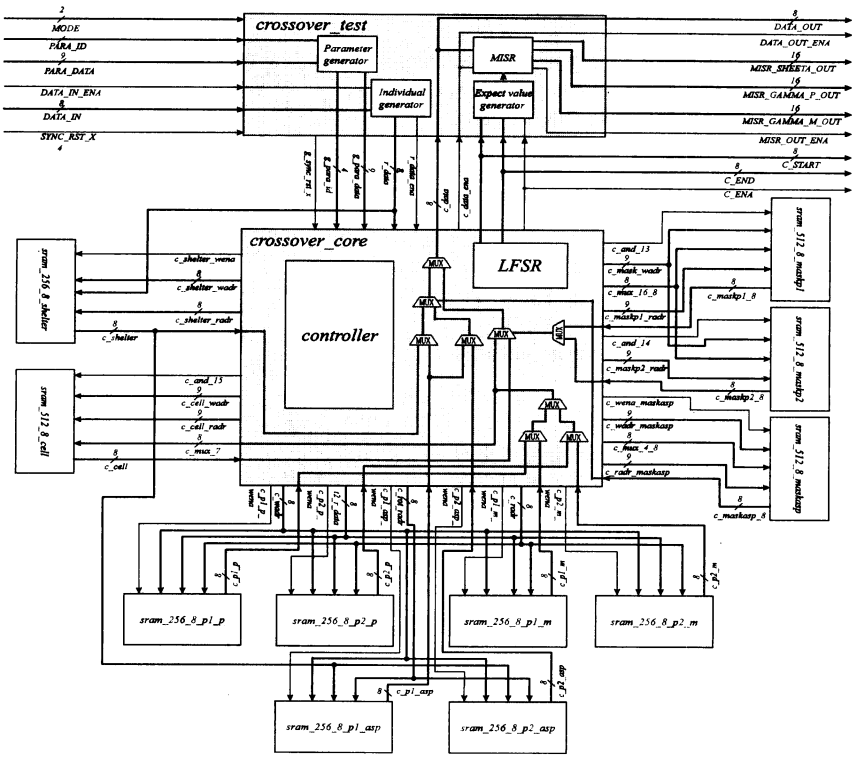

Fig.9. Block chart of crossover circuit
図９．交叉回路ブロック図

4.2. crossover_test 部

まず入力として，ノーマルモードまたはテストモードを選択する信号を与え，それがテストモードである場合に回路の動作テストを行う。

交叉回路は，窓というランダムなパラメータを用いているので，ランダムな個体をテスト用に与えると期待値を生成することが困難であり，回路規模も大きくなってしまう（期待値生成部分が crossover_core と同等の規模になってしまう）。そこで，図 10 に示すような，ブロックが一列に並んだ単純な個体をテスト用の親個体として用いることにした。

テスト用パターンとして２つの親個体を生成し，同時にブロック数，個体数，窓幅などのパラメータも乱数で発生させる。crossover_core 回路からの窓データをもとに子個体の期待値を生成し，期待値と実際の子個体データを比較し，比較結果出力によりテストを行うことができる。crossover_core 回路と crossover_test 回路の関係を図 11 に示す。

4.3. crossover_core 部

rossover_core 部は，２つの親個体データに対して交叉を行い，２つの子個体を生成し，出力する。具体的にはメモリの制御と窓の生成を行う。

まず,最大値をブロック数 － 窓幅 ＋1 として窓の基準点を乱数で発生させる。こうすることにより，窓がブロック数を超えて生成されることを防ぐことができる。基準点を開始点，基準点＋窓幅を終了点として開始点から終了点の間を 1,それ以外を 0 とするような 1 ビットの信号をΓ＋の窓データとして交叉マスク RAM に送る。ここで，交叉マスクとは，窓内ブロックと窓外ブロックを識別する 1 ビットのマスクを言う。

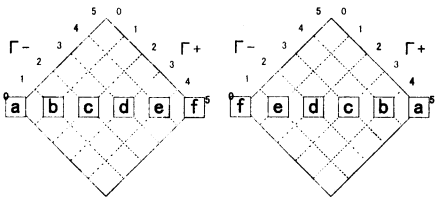

Fig.10. Parent individual for test
図 10. テスト用親個体

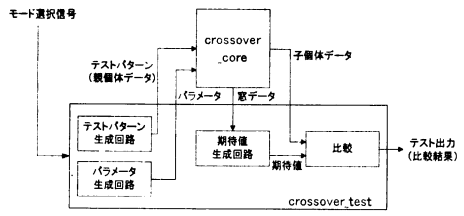

Fig.11. Relation between
core and test
図 11. core 回路と test 回路の関係

Γ－についても同様に窓データを生成し，Γ＋，Γ－ともに窓データが 1 であるブロックを 1，それ以外を 0 になるようにして，交叉マスクを生成する。

この交叉回路のアーキテクチャでは，複数のメモリを用いてそれらが同時にアクセスできるようにし，パイプライン処理を実現した。パイプラインを図 12 に，子個体出力までのデータの流れを図 13 に示し，以下説明する。

① P1(θ・aspect 比)に入力データを格納する。

② P1 (Γ＋，Γ－)に入力データを格納する。
窓生成回路からの窓データをライトデータ，入力データ（この時点では P1 のΓ＋，Γ－）をライトアドレスとして P1 のマスクデータを交叉マスク（P1）に書き込む。

③ P2(θ・aspect 比)に R_DATA を格納する。 P1 のマスクデータを交叉マスク(θ・aspect 比)に書き込む。

④ P2(Γ＋，Γ－)に入力データを格納する。
窓生成回路からの窓データをライトデータ，入力データ（この時点では P2 のΓ＋，Γ－）をライトアドレスとして P2 のマスクデータを交叉マスク（P2）に書き込む。
P1 のマスクデータをライトイネーブル，入力データ（この時点では P2 のΓ＋，Γ－）をライトアドレスとして，交叉セル(P2 格納用)を生成し，交叉セル RAM に書き込む。

⑤ P2 のマスクデータを交叉マスク(θ・aspect 比)に書き込む。
交叉マスク(θ・aspect 比)を制御信号として，P1(θ・aspect 比)と入力データ（この時点では P2 のθ・aspect 比のデータ）をマルチプレクサに入力し，子 1 のθ・aspect 比を出力する。

⑥ 交叉マスク（P1）を制御信号として，P1(Γ＋，Γ－)と交叉セル(P2 格納用)をマルチプレクサに入力し，子 1 のΓ＋，Γ－を出力する。P2 のマスクデータをライトイネーブル，P1(Γ＋，Γ－)をライトアドレスとして交叉セル(P1 格納用)を生成し，交叉セル RAM に書き込む。

⑦ 交叉マスク(θ・aspect 比)を制御信号として，P1(θ・aspect)と P2(θ・aspect)をマルチプレクサに入力し，子 2 のθ・aspect 比を出力，続いて交叉マスク（Γ＋，Γ－）を制御信号として，P2(Γ＋，Γ－)と交叉セル(P1 格納用)をマルチプレクサに入力し，子 2 のΓ＋、Γ－を出力する。

5. 評価・考察

5.1. ソフトウェア／ハードウェア速度比較

提案アーキテクチャの処理速度の有効性を確かめるために，フロアプランシステムにおけるソフトウェア（CPU: PentiumIV 2.4GHz, OS: Vine Linux, メモリ: 1GHz）との比較を行った。ハードウェアは 20MHz の動作周波数を想定し，ステップ数による試算を行った。表 1 にその結果を示す。

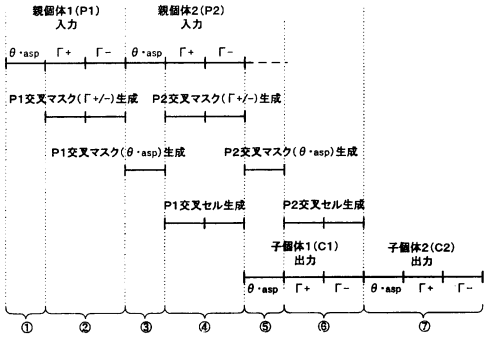

Fig.12. Crossover pipeline
図 12. 交叉パイプライン

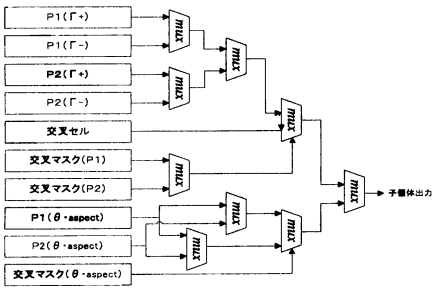

Fig.13. Flow of data until child individual outputting
図 13. 子個体出力までのデータの流れ

Table1. Processing speed comparison with software
表 1. ソフトウェアとの処理速度比較

ブロック数	処理時間(秒)		速度比
	ソフトウェア	ハードウェア	
10	12	1.1	10.9
30	21	2.4	8.75
50	56	3.9	14.3
100	286	7.7	37.1
200	1707	15.2	112
250	3095	18.7	165

速度比がブロック数に比例しているのは，ソフトウェアでは個体の評価を行うのに要する時間がブロック数の二乗に比例するのに対し，ハードウェアではブロック数に比例するためであると考えられる。この結果，ブロック数が多い場合は速度比が良く，ハードウェアが有効であるといえる。

5.2. 逐次処理／パイプライン処理速度比較

ブロック数をNとすると，逐次処理の場合，2つの親個体入力で6×Nステップ，P1・P2それぞれの交叉マスク生成に6×Nステップ，P1・P2それぞれの交叉セル生成に4×Nステップ，子個体出力で6×Nステップかかるので，合計22×Nステップを要する。一方、今回のパイプライン処理を施した場合，2つの親個体の入力終了後直ちに子個体の出力を行うことができるので，親個体入力で6×Nステップ，子個体出力で6×Nステップ，合計12×Nステップで処理を行うことができ，約1.8倍の処理速度を得ることができた。

6. まとめ

ＧＡを解探索手法に用いたシーケンスペアに基づくフロアプランシステムを提案し，ＧＡの遺伝オペレーションの一つである交叉部分のアーキテクチャを設計した。処理速度の有効性を確かめるため，ソフトウェアとの比較実験を行い，250 ブロックにおいては約165倍の処理速度の見通しを得ることができた。また，交叉部分に関しては表 2 のスペックに示すように，ASIC に実装することによってさらに動作周波数を高めることができた。

今後の課題としては，より高速なフロアプランシステム構築のため，選択，突然変異，評価部分の ASIC 化や，柔軟性と速度のトレードオフを考えた FPGA-ASIC 混載フロアプランシステムの構築等が挙げられる。

Table2. Spec of crossover circuit
表 2. 交叉回路のスペック

動作周波数		33.3 MHz
ゲート数		13896.5
セル数		5550
dual port RAM	512×8	5
	256×8	6
チップサイズ		6.71 ㎜ × 6.81 ㎜
I/O ピン数		103 pin / 208 pin

参考文献

[1]J.Cong, Z.Pan, L.Hei, C.K.Koh, and K.Y.Khoo, "Interconnect design for deep submicron ICs,"Dig.Tech.Papers ICCAD, pp.478-485,1997.
[2]K.Bazargan, S.Kim, and M.Sarrafzadeh,"A floorplanner of uncertain designs,"IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, Vol.18, No.4,pp.389-397,1997.
[3]Sait,S.M. and Youssef, H.:VLSI Physical Design Automation, IEEE Press(1995).
[4]H.Murata, K.Fujiyoshi, S.Nakatake, and Y.Kajitani,"VLSI module placement based on rectangle-packing by the sequence-pair,"IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems, Vol.15, No.12,pp.1518-1524, 1996.
[5]Koichi Hatta, Shin'ichi Wakabayashi, Tetsushi Koide ,"Solving the rectangular packing problem by an adaptive GA based on sequence pair," Proc. Asia-South Pacific Design Automation Conference, pp.181-184(1999).
[6]J.Holland:"Adaptation in Natural Artificial Systems", the University of Michigan Press(Second edition; MIT Press)(1992).
[7]中矢真吾，小出哲士，若林真一，“適応的遺伝的アルゴリズムに基づく VLSI フロアプランニングの一手法” 情報処理学会論文誌 pp.1361-1371,2002