A Genetic Algorithm for Channel Routing in VLSI Circuits*

Jens Lienig† and K. Thulasiraman

Department of Electrical and Computer Engineering
Concordia University
1455 de Maisonneuve Blvd. West
Montreal, Quebec H3G 1M8, Canada

Abstract

A new genetic algorithm for channel routing in the physical design process of VLSI circuits is presented. The algorithm is based on a problem specific representation scheme and problem specific genetic operators. The genetic encoding and our genetic operators are described in detail. The performance of the algorithm is tested on different benchmarks and it is shown that the results obtained using the proposed algorithm are either qualitatively similar to or better than the best published results.

1 Introduction

In the physical design process of very large scale integrated (VLSI) circuits the logical structure of a circuit is transformed into its physical layout. Detailed routing is one of the tasks in this process. A detailed router connects pins of signal nets in a rectangular region under a set of routing constraints, such as the number of layers, the minimal space between wires and the minimum wire width. The quality of this detailed routing has a strong influence on the performance and production costs of the circuit.

The detailed routing in a rectangular region with pins exclusively located on the upper or lower boundary of the routing region is called channel routing. Channel routing is one of the most commonly occurring routing problems in VLSI circuits. A simple example of a channel routing problem and a possible routing solution is shown in Figure 1.

The channel routing problem is NP-complete [33] and therefore, there is no known deterministic algorithm to solve it in a polynomial time. Hence, although many different algorithms have been proposed (e.g. [12], [19], [28], [32], [34]), the problem of finding the globally optimized solution for channel routing is still open.

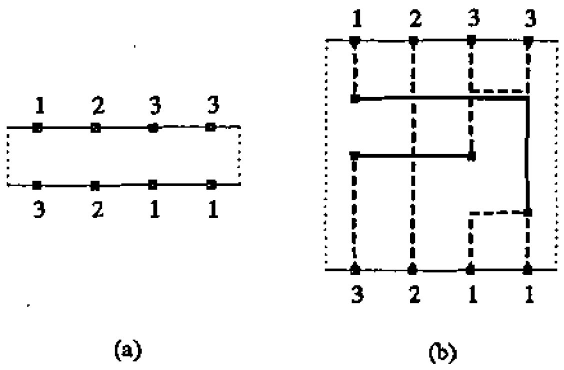


Figure 1: An example of a channel routing problem (a) and a possible routing solution (b). Solid lines represent interconnections on one layer, the poly layer; and dashed lines represent interconnections on the other layer, the metal layer.

New approaches are necessary to solve this problem. The evolution process in nature optimizes, for example, the fitness of an individual in its environment and thus, can be used as a strategy for mathematical optimization. Genetic algorithms are a new class of heuristic search methods based on the biological evolution model. During the last few years, genetic algorithms have been applied more and more successfully to find good heuristic solutions to NP-complete optimization problems [13], [14].

The strength of a genetic algorithm results from the ability to perform a fairly efficient search in the search space even if the available knowledge is limited to an evaluation procedure that can measure the quality of any point in the search space[16]. Consequently, genetic algorithms belong to the category of the so-called weak methods, i.e., problem solving methods that make few assumptions about the problem domain; hence, they usually enjoy wide applicability. However, as stated by many authors (e.g. [7], [26]), these well-theorized, binary coded, pure genetic algo-

*This article appeared in *Evolutionary Computation*, Vol. 1, No. 4, pp. 293-311. Cambridge, MA: MIT Press, 1994.

†E-mail: jensl@ece.concordia.ca

rithms cannot handle a lot of highly constrained problems. To solve this dilemma, many application-specific variations of genetic algorithms have been developed. These variations enhance the traditional genetic algorithm by incorporating problem specific knowledge in both appropriate coding schemes and genetic operators (e.g. [15], [25], [26]).

We present a genetic algorithm for channel routing that is based on such a problem specific representation scheme and problem specific genetic operators. The algorithm starts by performing a random path search to create different routing solutions of the channel. These non-optimized routing structures are seen as individuals of an initial population. They are coded in 3-dimensional chromosomes with integer representation. Based on certain quality factors, these routing structures are improved by genetic operators to eventually present a globally optimized routing result. It is shown that the resulting routing structures are either qualitatively similar to or better than the best results available in the literature.

2 Problem description

The channel routing problem is defined as follows. Consider a rectangular routing region, called *channel*, with a number of *pins* located either on the upper or the lower boundary of the channel. The pins that belong to the same net have to be connected, subject to certain constraints and quality factors. The connection has to be made inside the channel on a symbolic routing area consisting of horizontal *rows* and vertical *columns* (see Figure 1 (b)).

The constraints for the interconnections include the following:

- A net is to be routed using a Manhattan geometry, i.e., only horizontal and vertical net segments are allowed.
- Two layers are available for routing (see Figure 1).
- A net may change from one layer to another using a contact window called a *via*.
- Different nets cannot cross each other on the same layer and must respect a minimum distance rule.
- The perimeter of the channel is not used for routing.

Three quality factors are used in this work to judge the quality of the routing result:

- Minimum routing area
  The horizontal dimension of the channel along which pins are located is fixed but the vertical dimension which has no pins (expressed as the number of rows of the channel) can vary depending on the area required for routing. It is desirable to use the least area, i.e., the least number of rows.
- Net length
  The shorter the length of the interconnection nets the smaller the propagation delay.
- Number of vias
  The introduction of a via between the two interconnection layers means longer propagation delays and lower fabrication yield. Consequently, the fewer the number of vias the better the routing quality.

3 Genetic algorithms in VLSI layout design

Because of its complexity, the physical design process of VLSI circuits is usually separated into four consecutive phases, namely, partitioning, placement, routing and compaction. In the following, we will give a brief overview of genetic algorithms that have been successfully applied in these major steps of VLSI layout design.

Partitioning The task of partitioning is to divide the components of a circuit into subsets to reduce the problem size of the layout design.

In [17] and [18], different coding schemes for the problem of circuit partitioning are investigated to find the most suitable coding. The proposed genetic algorithm is tailored for the partitioning of circuits with complex bit-slice components using a special two-step coding of partitions. The genetic algorithm in [6] is based on a population structure that involves subpopulations which have their isolated evolution occasionally interrupted by inter-population communication.

Placement The placement procedure is responsible for the assignment of the circuit's components to their locations on the chip. According to variation in sizes and locations of these components, placement algorithms can be divided into algorithms for standard cell layout, macro cell layout and gate-matrix layout.

After the pioneering work of Cohoon et al. [5], further applications of genetic algorithms [29], [30] and evolution strategies [20], [21], [35] for standard cell placement have been presented. These approaches produce high quality placements at the cost of long run times. In [24], the run time has been reduced significantly by using a parallel implementation of a genetic algorithm.

We are aware of three papers in which genetic algorithms for macro cell placement are discussed [3], [8], [9]. The approach in [3] is based on a two-dimensional bitmap representation of the macro cell placement problem. Another representation scheme, a binary tree, is applied in [8]. In [9], a combination of a genetic algorithm with a simulated annealing strategy is presented. The experimental results suggest that a mixed strategy performs better than a pure genetic algorithm for the macro cell placement problem.

An application of a genetic algorithm for the placement of gate-matrix layouts has been published in [31].

Routing As already mentioned in Section 1, routing is the process of connecting pins subject to a set of routing constraints. VLSI routing is usually divided into global routing (to assign nets into certain routing regions) and detailed routing (to assign nets to exact positions inside a routing region).

To our knowledge, only one evolutionary algorithm for global routing has been reported [4].

According to the position of the pins, detailed routing can be separated into channel routing (pins are only located on two parallel sides of the routing area) and switchbox routing (pins are placed on all four sides of the routing area).

Three papers have been published in which strategies derived from the concept of genetic algorithms are applied to the channel routing problem [11], [23], [27]. In [23], a rip-up-and-rerouter is presented which is based on a probabilistic rerouting of nets of one routing structure. However, the routing is done by a deterministic Lee algorithm [22] and main components of genetic algorithms, such as the crossover of different individuals, are not applied. The router in [11] combines the so-called steepest descent method with features of genetic algorithms. The crossover operator, however, is restricted to the exchange of entire nets and the mutation procedure performs only the creation of new initial individuals. The proposed algorithm in [27] is limited to the restrictive channel routing problem. Here, all vertical net segments are located on one layer and all horizontal segments are placed on the other. Furthermore, so-called doglegs[1] are not allowed, i.e., the horizontal segments of each net must be placed on only one horizontal row. Due to these restrictions, this algorithm cannot be used for routing structures with loops in the vertical constraint graph, as is often the case in practice[2]. Moreover, the resulting routing area is generally larger than necessary.

[1]The term "dogleg" is used in VLSI literature to describe a vertical net segment that connects two horizontal segments of the same net located on different rows.

[2]A vertical constraint graph is a directed graph with its nodes representing the nets of the channel and its branches representing the relative position of the horizontal parts of a net from the top to the bottom of the channel. The forming of the vertical constraint graph is based on the assumption that each net can have at most one horizontal segment. A loop in the horizontal constraint graph indicates that a routing solution cannot be achieved with this assumption, i.e., at least one net has to be divided into different horizontal segments.

The algorithms in [11], [23] are also applied to switchbox routing.

Compaction Compaction is usually the final step in the physical layout design of VLSI circuits to transform the symbolic layout to a mask layout with the goal of minimizing the size of the resulting circuit layout.

To the best of our knowledge, the only application of a genetic algorithm for compaction has been advanced by Fourman [10]. He describes two prototypes of genetic algorithms which perform compaction of a symbolic circuit layout. Although his results are limited to very simple layout structures, he proposes a new problem specific representation for layout design that includes constraints of the compaction process.

4 Description of our algorithm

4.1 Survey

Genetic algorithms, in general, carry out optimization by simulating biological evolutionary processes. The environment in which individuals live affects their ability to survive and the individual best suited for the environment has the highest probability of survival and reproduction. The descendants that inherit desirable characteristics for survival in the environment also have a high probability of survival and reproduction, while other, less fit individuals die out. This principle is known as "the survival of the fittest" and can be used in optimization [13].

In our channel routing problem an *individual* can be defined as a channel routing result, i.e., a routing structure. The quality of this routing structure according to the above mentioned quality factors can be evaluated to produce a measurement of the individual's *fitness*. First, we generate an *initial population* of randomly created, and thus different, routing structures for a given channel routing problem. This population is subjected to a simulated evolution process consisting of three main components, namely, *selection*, *crossover* and *mutation*. If the simulation works, better and better evaluated individuals will predominate in the population because they have a higher probability of reproducing descendants which can inherit the best characteristics of their predecessors. These best evaluated individuals are the best routing solutions according to our quality factors.

An overview of the genetic algorithm presented in this paper is shown in Figure 2. The number of individuals $|\mathcal{P}_c|$ is kept constant throughout all generations. Our mutation operator is applied after the reduction procedure, i.e., the modifications caused by the mutation operator remain "unpunished" in the population during the next mate selection and crossover procedure. This separation of the crossover and mutation procedures improves the ability of our approach to overcome local optima. Since the mutation operator has access to all individuals, the best individual is saved in each generation before the mutation operator is applied. At the end of the algorithm, the best individual p_{best} that has ever existed undergoes an optimization and then constitutes our final routing solution.

```
create initial population $(\mathcal{P}_c)$
fitness_calculation $(\mathcal{P}_c)$
$p_{best}$ = best_individual $(\mathcal{P}_c)$
for $generation = 1$ until $max\_generation$
  $\mathcal{P}_n = \emptyset$
  for $offspring = 1$ until $max\_descendant$
    $p_\alpha$ = selection $(\mathcal{P}_c)$
    $p_\beta$ = selection $(\mathcal{P}_c)$
    $\mathcal{P}_n = \mathcal{P}_n \cup$ crossover $(p_\alpha, p_\beta)$
  endfor
  fitness_calculation $(\mathcal{P}_n)$
  $\mathcal{P}_c$ = reduction $(\mathcal{P}_c \cup \mathcal{P}_n)$
  $p_{best}$ = best_individual $(p_{best} \cup \mathcal{P}_c)$
  mutation $(\mathcal{P}_c)$
  fitness_calculation $(\mathcal{P}_c)$
endfor
optimize $(p_{best})$
```

Figure 2: Outline of the algorithm.

4.2 Genetic encoding scheme

In genetic algorithms, a distinction is made between the *genotype* and the *phenotype* of an individual [13]. While the genotype is the coding of the information of an individual, the phenotype is the physical appearance of the individual. Crossover and mutation are carried out on the genotype; fitness has to be expressed in terms of the phenotype.

We use for the genetic encoding of the routing structures a three-dimensional lattice-like chromosome (see Figure 3). The length of the z-axis of the chromosome is two units in accordance with the number of layers. Two horizontal adjacent chromosome positions represent the minimal distance between two adjacent, different routing connections on the phenotype.

Each individual is encoded in one chromosome. According to the position in the phenotype, chromosome positions are occupied with coding numbers of the routing connections and pins. The coding must distinguish between routing connections which can be shifted or erased during the evolution process and fixed pins. Thus, we choose the following encoding scheme (see Figure 3):

Let (x, y, z) be a chromosome position in the genotype, $G(x, y, z)$ be the value of the chromosome position and (x', y', z') be the corresponding coordinate in the phenotype.

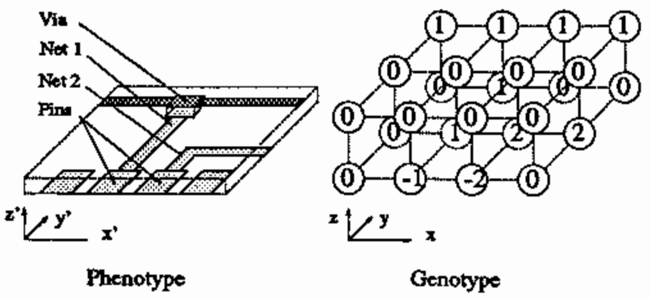


Figure 3: Genetic encoding of the routing structure.

- If $G(x, y, z) = 0$, the phenotype is not occupied at (x', y', z').

- If $G(x, y, z) > 0$, the phenotype is occupied with a routing connection at (x', y', z'). This routing connection can be shifted or erased. $G(x, y, z)$ represents the net number of the phenotype at (x', y', z').

- If $G(x, y, z) < 0$, the phenotype is occupied with a pin at (x', y', z'). This pin cannot be shifted or erased.

We chose this three-dimensional encoding scheme with integer representation after numerous experiments with other genetic encoding schemes. For example, parts of the routing structure with near-optimal routing paths (termed as good "routing islands") are often scattered over the chromosome instead of being represented in one compact building block when binary or integer string representations are used. This leads to unsatisfactory convergence behavior of the genetic algorithm. In another approach, the representation of the routing problem in a graph or tree requires decoding of the genotype into its corresponding phenotype whenever a genetic operator is applied so as to monitor the routing constraints. This results in an unacceptable run time of the algorithm.

Our three-dimensional encoding scheme ensures that good "routing islands" in the routing structure are preserved as compact high-fitness building blocks in the chromosome. Consequently, these building blocks have a high probability of being transformed intact and recombined with other high-quality building blocks in the next generation. Furthermore, this encoding scheme enables a simple monitoring of the routing constraints directly in the chromosome.

For the sake of simplicity, we will describe the genetic operators in terms of the representation of the phenotype.

4.3 Creation of an initial population

The initial population is constructed from randomly created individuals.

First, each of these individuals is assigned a random initial number y_{ind} of rows with $2 * y_{min} \leq y_{ind} \leq 4 * y_{min}$, where y_{min} represents the estimated number of rows of the best individual expected at the end of the algorithm.

Let $\mathcal{S} = \{s_1, ...s_i, ...s_k\}$ be the set of all pins of the channel which are not connected yet and let $\mathcal{T} = \{t_1, ...t_j, ...t_l\}$ be the set of all pins having at least one connection to another pin. Initially $\mathcal{T} = \emptyset$. A pin $s_i \in \mathcal{S}$ is chosen randomly among all elements in $\mathcal{S}$. If $\mathcal{T}$ contains pins $\{t_u, ...t_j, ...t_v\}$ (with $1 \leq u < v \leq l$) of the same net, a pin t_j is randomly selected among them. Otherwise a second pin of the same net is randomly chosen from $\mathcal{S}$ and transferred into $\mathcal{T}$. Both pins (s_i, t_j) are connected with a so-called "random routing". Then s_i is transferred into $\mathcal{T}$. The process continues with the next random selection of $s_i \in \mathcal{S}$ until $\mathcal{S} = \emptyset$.

The random routing of (s_i, t_j) is done as follows. A vertical line is extended from both s_i and t_j until it reaches an obstacle, e.g. the channel border or an already routed net of different potential on the same layer (see Figure 4 (a,b)). A position between the start point and the end point of both lines is randomly chosen. From these positions horizontal lines are extended in both directions (see Figure 4 (c)). The path search continues by selecting random points on the two horizontal lines and extending vertical lines at those points in both directions (see Figure 4 (d)), and so on.

The layer of each extension line is chosen as follows. Let each layer have a preferred routing direction and r_n be a random number between 0 and 1. If $r_n \leq 2/3$, the extension line is created with the layer associated with the routing direction of the extension line. Otherwise (i.e., $2/3 < r_n \leq 1$) it is created with the layer having a preferred routing direction opposite to the direction of the extension line.

The extension is stopped when

- the extension lines of both points meet each other on the same layer, or
- the extension lines of s_i touch a net point which is already connected with t_j as shown in Figure 4 (e) (or vice versa).

In the latter case, t_j (or s_i) is replaced with this meeting point (see Figure 4 (f)).

If the creation of extension lines does not succeed in one of these conditions within i iterations, all extension lines are erased and the channel is extended with an additional row on a random position y_{add} with $1 \leq y_{add} \leq y_{ind}$ (y_{ind} = current number of rows of the channel). After adjusting all previous routed nets to this new row, s_i and t_j undergo a new attempt to connect them with randomly created extension lines.

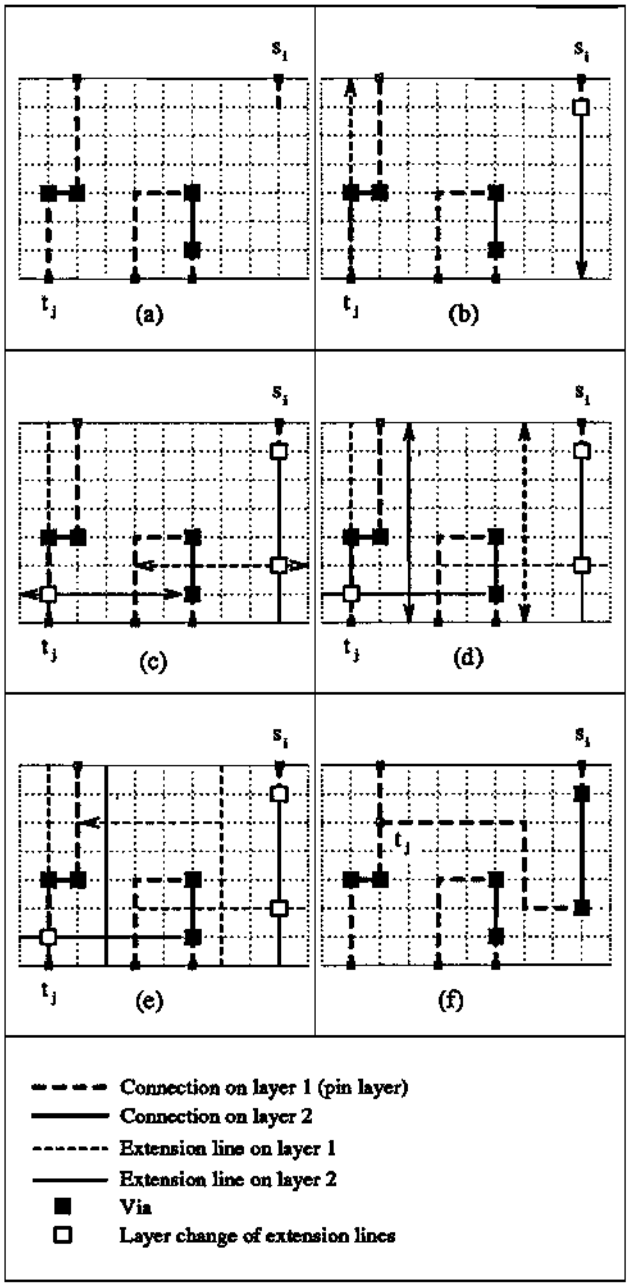

s_i
t_j
(a)
(b)
(c)
(d)
(e)
(f)
Connection on layer 1 (pin layer)
Connection on layer 2
Extension line on layer 1
Extension line on layer 2
Via
Layer change of extension lines

Figure 4: Random routing of (s_i, t_j).

The maximum number of iterations i is calculated according to

$$i = (3 * |x_s - x_t|) + y_{ind} + 10 \tag{1}$$

where x_s = column position of s_i,
x_t = column position of t_j and
y_{ind} = current number of rows of the channel.

If 10 extensions of the channel also do not lead to a connection, this individual is completely deleted and the process to create a new individual is started again right from the beginning.

The routing process of (s_i, t_j) is finished by tracing the shortest path on the extension lines from their meeting point backwards to both s_i and t_j (see Figure 4 (f)). This backtracing avoids unnecessary loops in the connection of (s_i, t_j) without limiting the randomness of the resulting routing path.

The creation of the initial population is finished when the number of completely routed channels is equal to the population size $|\mathcal{P}_c|$. As a consequence of our strategy, these initial individuals are quite different from each other and scattered all over the search space.

4.4 Calculation of fitness

The fitness F of each individual $p \in \mathcal{P}$ is calculated to assess the quality of the individual's routing structure relative to the rest of the population $\mathcal{P}$. The selection of the mates for crossover and the selection of individuals which are transferred into the next generation are based on these fitness values.

First, two functions F_1 and F_2 are calculated for each individual $p \in \mathcal{P}$ according to Equations (2) and (3).

$$F_1(p) = \frac{1}{y_{ind}} \tag{2}$$

where y_{ind} = number of rows of individual p.

$$F_2(p) = \frac{1}{\sum_{i=1}^{n_{ind}} (l_{acc}(i) + a * l_{opp}(i)) + b * v_{ind}} \tag{3}$$

where $l_{acc}(i)$ = net length of net i of net segments according to the preferred direction of the layer,
$l_{opp}(i)$ = net length of net i of net segments opposite to the preferred direction of the layer,
a = cost factor for the preferred direction,
n_{ind} = number of nets of individual p,
v_{ind} = number of vias of individual p and
b = cost factor for vias.

In order to assure that the area minimization, i.e., the number of rows, predominates the net length and the number of vias, the fitness $F(p)$ is derived from $F_1(p)$ and $F_2(p)$ as follows:

Assume that $(p_i, ...p_x, ...p_j)$ are individuals with the same number y of rows, i.e., the same value $F_1(p)$. These individuals are arranged in an ascending order according to $F_2(p)$. Then p_i is the individual with the lowest value $F_2(p)$ in this group ("worst individual with y rows"). Its fitness value $F(p_i)$ is defined by

$$F(p_i) = F_1(p_i). \tag{4}$$

The individual p_j has the highest value $F_2(p)$ in this group ("best individual with y rows"). Let $F_1(p_{j+1})$ be the F_1-value of the next ("better") group with $y-1$ rows. The fitness $F(p_j)$ is calculated as follows:

$$F(p_j) = F_1(p_{j+1}) - \frac{\Delta F_1}{j - i + 1} \tag{5}$$

where $\Delta F_1 = F_1(p_{j+1}) - F_1(p_j)$.

Now $F(p_x)$ of the remaining individuals of this group can be calculated relative to their F_2-values between the lower bound $F(p_i)$ and the upper bound $F(p_j)$:

$$F(p_x) = F(p_j) - \frac{\Delta F * (F_2(p_j) - F_2(p_x))}{\Delta F_2} \tag{6}$$

where $\Delta F = F(p_j) - F(p_i)$, and
$\Delta F_2 = F_2(p_j) - F_2(p_i)$.

After the evaluation of $F(p)$ for all individuals of the population $\mathcal{P}$ these values are scaled linearly as described in [13], in order to control the variance of the fitness in the population.

4.5 Selection strategy

The selection strategy is responsible for choosing the mates among the individuals of the population $\mathcal{P}_c$. Because of its impact on the standard deviation in the population, the selection strategy is crucial to the performance of the algorithm.

Our selection strategy is stochastic sampling with replacement in accordance with the terminology in [13]. That means any individual $p_i \in \mathcal{P}_c$ is selected with a probability

$$\frac{F(p_i)}{\sum_{p \in \mathcal{P}_c} F(p)}$$

The two mates needed for one crossover are chosen independently of each other. An individual may be selected any number of times in the same generation.

4.6 Crossover operator

During crossover, two individuals are combined to create a descendant. Let p_α and p_β be copies of the mates (Figure 5 (a,b)) and p_γ be their descendant.

First, a cut column x_c is randomly selected with $1 \leq x_c < x_{ind}$, where x_{ind} represents the number of columns of the individuals.

The individual p_α transfers its routing structure to p_γ which is

- located on (x_α, y_α, z) with $1 \leq x_\alpha \leq x_c$, $1 \leq y_\alpha \leq y_{ind\alpha}$ ($y_{ind\alpha}$ = number of rows of p_α), $1 \leq z \leq 2$ and
- not cut by the cut column x_c.

Accordingly, p_β transfers to p_γ the uncut connections located on (x_β, y_β, z) with $x_c < x_\beta \leq x_{ind}$, $1 \leq y_\beta \leq y_{ind\beta}$ and $1 \leq z \leq 2$ (see Figure 5 (c,d)).

Note that connections of p_α and p_β cut by x_c are traced until their next Steiner point or pin is reached and not transferred into p_γ.

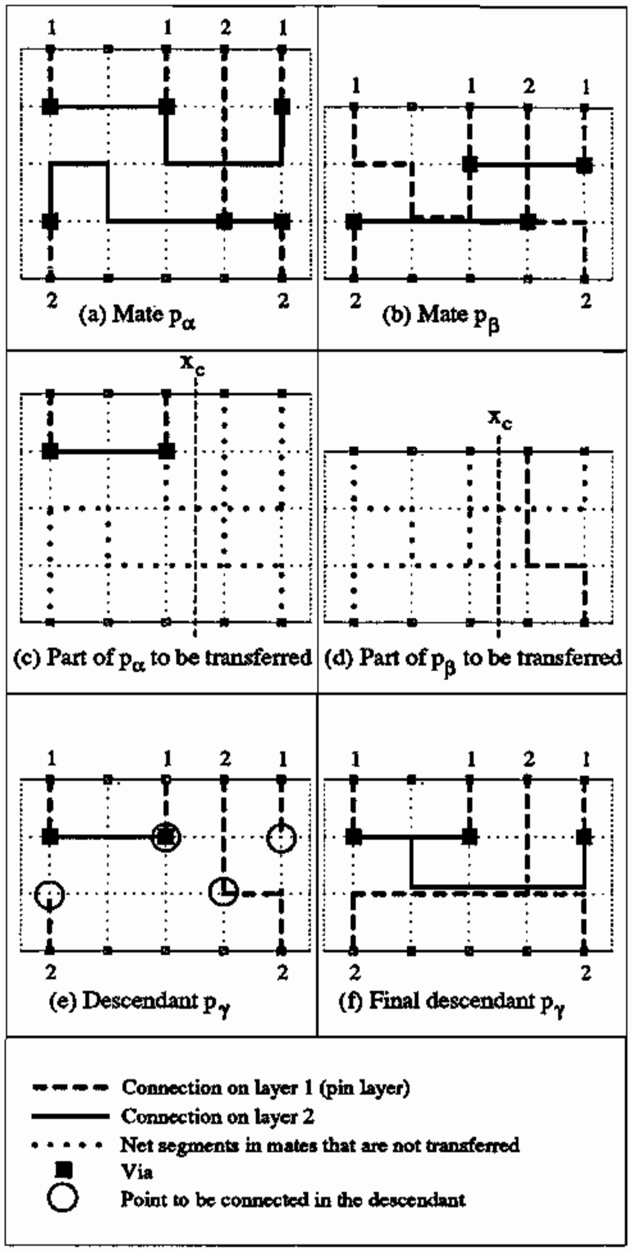
Assume that the part of p_α (or p_β) which has to be transferred into p_γ contains rows not occupied by any horizontal segments. Then the number of rows $y_{ind\gamma}$ of p_α (or $y_{ind\beta}$ of p_β) is decremented by deleting this unoccupied row until no empty row is left.

The initial number of rows $y_{ind\gamma}$ of p_γ is equal to the maximum of $(y_{ind\alpha}, y_{ind\beta})$. The mate which now contains fewer rows than p_γ is extended with additional row(s) at random position(s) before transferring its routing structure to p_γ.

The routing of the remaining open connections in p_γ is done as follows: Let $\mathcal{N}_\alpha$ be the set of all Steiner points or pins which are end points of a cut segment in p_α. Accordingly, let $\mathcal{N}_\beta$ be the set of these points in p_β. If $\mathcal{N}_\alpha$ contains more than one point of the same net, these points are connected with each other in a random order by our random routing strategy (see Section 4.3). Except for one randomly chosen point, all points of this net in $\mathcal{N}_\alpha$ are now deleted. The same "inner routing" in $\mathcal{N}_\beta$ is performed. As a result, $\mathcal{N}_\alpha$ and $\mathcal{N}_\beta$ do not contain more than one point per net. These points in $\mathcal{N}_\alpha$ are now selected randomly and compared with all points in $\mathcal{N}_\beta$. If a point of the same net is found in $\mathcal{N}_\beta$, both points are connected by means of our random routing (see Figure 5 (e,f)).

If the random routing of two points does not lead to a connection within i extension lines per point (see Equation (1)), the extension lines are deleted and the channel is extended at a random position y_{add} with $1 \leq y_{add} \leq y_{ind\gamma}$. If j extensions of the channel also do not enable a connection (j = *initial* number of rows of p_γ), p_γ is deleted entirely and the crossover process starts again with a new random cut column x_c applied to p_α and p_β.

The crossover process of creating p_γ is finished with deleting all rows in p_γ that are not used for any horizontal routing segment.



1 1 2 1
2 2
(a) Mate p_α
1 1 2 1
2 2
(b) Mate p_β
x_c
(c) Part of p_α to be transferred
x_c
(d) Part of p_β to be transferred
1 1 2 1
2 2
(e) Descendant p_γ
1 1 2 1
2 2
(f) Final descendant p_γ
Connection on layer 1 (pin layer)
Connection on layer 2
Net segments in mates that are not transferred
Via
Point to be connected in the descendant

Figure 5: Crossover of (p_α, p_β) to p_γ.

4.7 Reduction strategy

Because the population size of a genetic algorithm should be constant, a reduction strategy is necessary to decide which individuals among the current population $\mathcal{P}_c$ and the set of descendants $\mathcal{P}_n$ should survive for the next generation.

We use a deterministic reduction strategy which guarantees that high quality individuals survive in as many generations as they are superior. Our reduction strategy simply chooses the $|\mathcal{P}_c|$ fittest individuals of $(\mathcal{P}_c \cup \mathcal{P}_n)$ to survive as $\mathcal{P}_c$ into the next generation. This strategy, which is the same as that usually applied in evolution strategies [1], is derived from the characteristic of our crossover operator that a high quality mate does not necessarily produce a high quality descendant, and in such a case, the mate should survive rather than the descendant.

4.8 Mutation operators

Mutation operators perform random modifications on an individual. The purpose is to overcome local optima and to exploit new regions of the search space.

We have designed four types of mutation operators which are applied in a random order with a certain mutation probability to each of the individuals $p_i \in \mathcal{P}_c$.

mut_1 Define a surrounding rectangle with random sizes (x_r, y_r) around a random center position (x, y, z). All routing structures inside this rectangle are deleted. The remaining net points on the edges of this rectangle are now connected again in a random order with our random routing strategy.

mut_2 Define a random number of nets n_r with $1 \leq n_r < n_{ind}$ (n_{ind} = number of nets of p_i). Select n_r nets randomly, delete them and route them again in a random order by means of the random routing.

mut_3 Add at a random row position y_{add}, with $1 \leq y_{add} \leq y_{ind}$ (y_{ind} = number of rows of p_i), an additional row, select randomly net segments from the "neighbor row(s)" and place them on y_{add}.

mut_4 Remove a row at a random row position y_{del} with $1 < y_{del} < r_{ind}$. Affected net segments are traced until their next Steiner point or pin is reached and rerouted by our random routing strategy.

If any of these mutations are not feasible, the mutation operator tries new random changes of the same type until either a successful mutation is performed or no feasible mutation of this type is possible within 30 iterations.

4.9 Optimization of the best individual

Since genetic algorithms are more focused on global optimization of the overall population rather than performing finely tuned local search, it is preferable to implement a local optimization of the best individual at the end of the evolution process.

In this local optimization, all types of mutations described in Section 4.8 are applied sequentially to the best individual, p_{best}, which has ever existed throughout the evolution process. Only improvements to p_{best} are accepted. The final p_{best} constitutes the routing solution to our specific channel routing problem.

5 Implementation and experimental results

The algorithm has been implemented in FORTRAN on a SPARC workstation. The approximate size of the source code is 8000 lines.

We have collected a number of well-known benchmarks for channel routing from the literature. In the following, we present our results using these benchmarks and compare the quality of these routing results with other approaches. We also conducted an experiment to study how the initialization of the random number generator affects these results.

5.1 Measurement conditions

The routing results of the benchmarks, presented later, are the best results obtained in 10 consecutive executions of the algorithm for each benchmark. All executions are based on an arbitrary initialization of the random number generator. We always stopped the executions after 150 generations.

The values of the other parameters are as follows:

$\vert\mathcal{P}_c\vert$	= 50
max_descendant	= 30
a	= 1.001 (Equation (3))
b	= 2.000 (Equation (3))

Mutation probability	mut_1	= 0.001
	mut_2	= 0.002
	mut_3/4	= 0.01

The same parameter setting is used for all benchmarks.

We have investigated different mutation probabilities for each of the mutation types. We achieved the best convergence towards high fitness of the best individual with the above mentioned values. If one of these mutation probabilities is increased, the frequent mutations turn the evolution process into a random walk. On the other hand, reduced mutation probabilities often lead to convergence in a local optimum only.

Benchmark	System	Col.	Rows	Netlength	Vias
Yoshimura-Kuh channel	Yosh.-Kuh [34]	12	5	75	21
	Weaver [19]	12	4	67	12
	Monreale [11]	12	4	72	11
	Our work	12	4	70	11
Joo6_12	Weaver [19]	12	4	79	14
	Packer [12]	12	4	82	18
	Monreale [11]	12	4	84	13
	Our work	12	4	79	14
Joo6_13	Greedy [28]	18	8	194	38
	Weaver [19]	18	7	169	29
	Silk [23]	18	6	171	28
	Packer [12]	18	6	167	25
	Our work	18	6	165	25
Joo6_16	Weaver [19]	11	8	131	23
	Weaver[a] [19]	11	7	121	21
	Monreale [11]	11	7	120	19
	Our work	11	6	116	15
Burstein's difficult channel	Mighty [32]	13[b]	4	83	8
	Packer [12]	12	4	82	10
	Monreale [11]	12	4	82	10
	Our work	12	4	82	8

[a] interactively
[b] additional column in the middle of the channel

Table 1: Benchmark results.

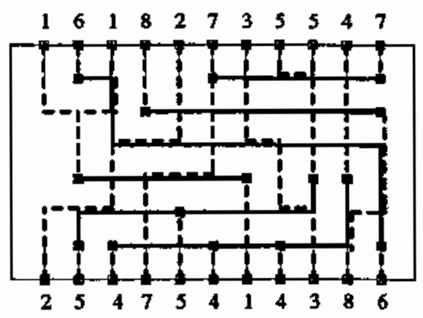


Figure 6: Our routing solution of Joo6_16.

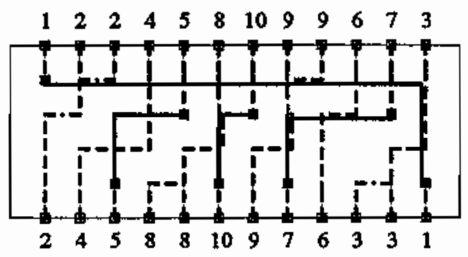


Figure 7: Our routing layout of Burstein's difficult channel.

5.2 Channel routing results

The performance of the algorithm has been tested on different benchmarks. The results obtained are presented in Table 1. It can be seen that our results are either as good as or better than the best known results from popular channel routers published for these benchmarks.

In [19, Fig. 6-16], Joobbani was able to route the so-called channel Joo6_16 which could not be routed by the Greedy algorithm [28]. This was accomplished by using his Weaver algorithm interactively and non-interactively. As is evident from Table 1, our algorithm yields better results than the Weaver algorithm even when the latter is used interactively. Figure 6 shows our routing solution.

The layout of Burstein's difficult channel achieved with our algorithm is depicted in Figure 7.

The CPU-times of the executions that obtained the results of Table 1 in 150 generations were:

Yoshimura-Kuh channel	:	5.6 min
Joo6_12	:	13.4 min
Joo6_13	:	94.2 min
Joo6_16	:	48.9 min
Burstein's difficult channel	:	9.6 min

Due to the inherent parallelism in genetic algorithms we are optimistic about reducing the runtime through the implementation of a parallel version of our algorithm.

5.3 Diversity within the population

We investigated the degree of diversity within the population during the convergence process because the population diversity is crucial to the ability of a genetic algorithm to guarantee a sufficient exploration of the search space.

Figure 8 shows the convergence behavior of the best, the average and the worst individual in the population for Burstein's difficult channel.

Similar graphs were achieved using the other benchmarks.

From these investigations we conclude that our genetic algorithm ensures sufficient diversity within the population even in an advanced stage of the evolution process.

5.4 Effect of random number generator

Since the methodology of our algorithm is probabilistic, it is important to investigate the effect of the initialization of the random number generator on the routing results.

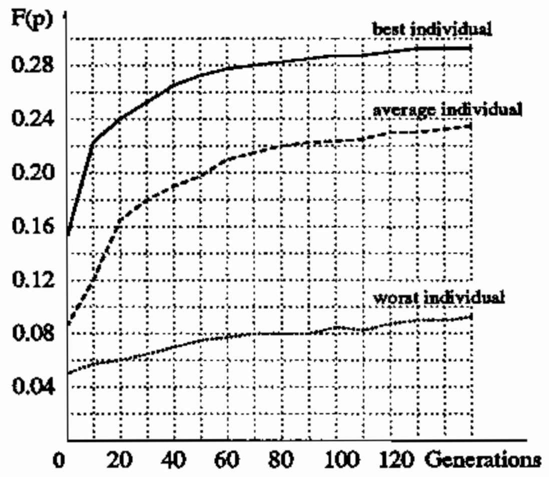


Figure 8: Average convergence behavior of the individuals for Burstein's difficult channel in 10 program executions.

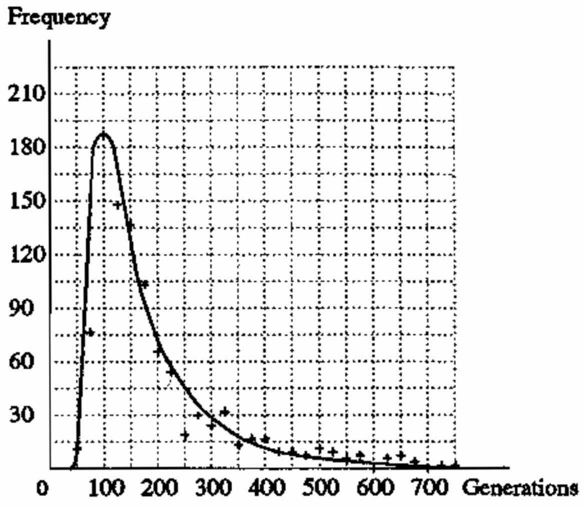


Figure 9: Distribution of the number of generations needed to achieve the best result of Burstein's difficult channel.

An experiment was conducted to study how the initial seed of the random number generator affects the number of generations needed to reach the best routing results presented in Table 1. We executed our program 1000 times with different initializations of the random number generator to route Burstein's difficult channel. Figure 9 shows the number of generations necessary to reach our best result for this channel. For example, between 88 and 112 generations were needed in 187 of the 1000 executions to achieve the result of Table 1. In one case, this result was reached after only 36 generations, in the worst case, 755 generations were necessary. On average, 197 generations are needed to obtain the result of Burstein's difficult channel as presented in Table 1.

Similar results were reached using the other benchmarks of Table 1. At this point we noticed a direct relationship between the complexity of the channel routing problem and the shape of the curve: The more complex the routing structure, the flatter is the curve and the more the curve is shifted towards a higher number of generations.

From the experiment we conclude that the initialization of the random number generator affects only the run time. Our routing results can be achieved with any initial seed of the random number generator.

6 Final remarks and conclusions

A new genetic algorithm for the channel routing problem of VLSI circuits has been presented. The algorithm is based on problem specific representation scheme and genetic operators. It has been shown that the results obtained using our algorithm are either qualitatively similar to or better than the best published results for channel routing benchmarks.

In developing this algorithm, the following conclusions have been reached:

- The representation scheme of a layout problem in a genetic algorithm should be a problem specific, three-dimensional representation rather than a one-dimensional string. Our scheme ensures that high quality parts of the layout structure are preserved as high-fitness building blocks and transferred intact with an increased probability in the next generation.

- The genetic operators of a genetic algorithm in a VLSI layout design should be adapted to the specific layout problem rather than selecting an unnatural representation that would allow the use of traditional genetic operators.

- The ability to overcome local optima is improved by separation of the crossover and mutation procedures.

- From our results we believe that genetic algorithms are promising tools for solving the channel routing problem and other optimization problems in the physical design process of VLSI circuits.

Our future work will concentrate on implementing a parallel version of the proposed algorithm in order to make it more efficient in terms of the run time.

Furthermore, additional studies are needed to investigate the effect of the genetic algorithm design on our results. For example, adopting Baker's selection algorithm [2] could lead to a more stable selection. Consequently, our reduction strategy could be changed to a probabilistic one or even be totally eliminated by replacing the current population with the population of the descendants.

Additional investigations are also needed to measure the performance of the algorithm as the size of the channel routing problem increases. Preliminary studies suggest that an exponential relationship exists between the CPU run time and the size of the channel routing problem. Further experiments are needed in this direction.

Acknowledgment

The authors would like to thank Henrik Esbensen, University of Michigan, Ann Arbor, for the fruitful discussions concerning this work.

References

[1] T. Bäck and H.-P. Schwefel, "An Overview of Evolutionary Algorithms for Parameter Optimization," *Evolutionary Computation*, Vol. 1, No. 1, pp. 1-23, 1993.

[2] J. E. Baker, "Adaptive Selection Methods for Genetic Algorithms," *Proc. of the First International Conference on Genetic Algorithms*, pp. 101-111, 1985.

[3] H. Chan, P. Mazumder and K. Shahookar, "Macro-Cell and Module Placement by Genetic Adaptive Search with Bitmap-Represented Chromosome," *Integration, the VLSI journal*, Vol. 12, No. 1, pp. 49-77, Nov. 1991.

[4] Y.-A. Chen, Y.-L. Lin and Y.-C. Hsu, "A New Global Router for ASIC Design Based on Simulated Evolution," *Proc. International Symposium VLSI Technology, Systems, and Applications*, Taipei, Taiwan, May 1989.

[5] J. P. Cohoon and W. D. Paris, "Genetic Placement," *IEEE Trans. on Computer-Aided Design*, Vol. 6, No. 6, pp. 956-964, Nov. 1987.

[6] J. P. Cohoon, W. N. Martin, and D. S. Richards, "Genetic Algorithms and Punctuated Equilibria in VLSI," *Parallel Problem Solving from Nature*, H. P. Schwefel and R. Männer, eds., Lecture Notes in Computer Science, Vol. 496, Berlin: Springer Verlag, pp. 134-144, 1991.

[7] L. Davis, "Adapting Operator Probabilities in Genetic Algorithms," *Proc. of the Third International Conference on Genetic Algorithms*, pp. 61-69, June 1989.

[8] H. Esbensen, "A Genetic Algorithm for Macro Cell Placement," *Proc. of the European Design Automation Conference*, pp. 52-57, Sept. 1992.

[9] H. Esbensen and P. Mazumder, "SAGA: A Unification of the Genetic Algorithm with Simulated Annealing and its Application to Macro-Cell Placement," *Proc. of the 7th International Conference on VLSI Design*, pp. 211-214, Jan. 1994.

[10] M. P. Fourman, "Compaction of Symbolic Layout using Genetic Algorithms," *Proc. of the First International Conference on Genetic Algorithms*, pp. 141-153, 1985.

[11] M. Geraci, P. Orlando, F. Sorbello and G. Vasallo, "A Genetic Algorithm for the Routing of VLSI Circuits," *Euro Asic '91*, Parigi 27-31 Maggio, Los Alamitos, CA: IEEE Computer Society Press, pp. 218-223, 1991.

[12] S. H. Gerez and O. E. Herrmann, "Switchbox Routing by Stepwise Reshaping," *IEEE Trans. on Computer-Aided Design*, Vol. 8, No. 12, pp. 1350-1361, Dec. 1989.

[13] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Reading, MA: Addison-Wesley Publishing Company, 1989.

[14] D. E. Goldberg, "Genetic and Evolutionary Algorithms Come of Age," To appear in *Communications of the Association for Computing Machinery (CACM)*, New York, NY: Association for Computing Machinery, 1994.

[15] J. J. Grefenstette, "Incorporating Problem Specific Knowledge into Genetic Algorithms," *Genetic Algorithms and Simulated Annealing*, pp. 42-60, Los Altos, CA: Morgan Kaufmann Publishers, 1987.

[16] J. H. Holland, *Adaptation in Natural and Artificial Systems*, Ann Arbor, MI: University of Michigan Press, 1975.

[17] M. Hulin, "Analysis of Schema Distributions," *Proc. of the Fourth International Conference on Genetic Algorithms*, pp. 204-209, 1991.

[18] M. Hulin, "Circuit Partitioning with Genetic Algorithms Using a Coding Scheme to Preserve the Structure of a Circuit," *Parallel Problem Solving from Nature*, H. P. Schwefel and R. Männer, eds., Lecture Notes in Computer Science, Vol. 496, Berlin: Springer Verlag, pp. 75-79, 1991.

[19] R. Joobbani, *An Artificial Intelligence Approach to VLSI Routing*, Boston, MA: Kluwer Academic Publishers, 1986.

[20] R. M. King and P. Banerjee, "ESP: Placement by Simulated Evolution," *IEEE Trans. on Computer-Aided Design*, Vol. 8, No. 3, pp. 245-256, March 1989.

[21] R. M. King and P. Banerjee, "Optimization by Simulated Evolution with Applications to Standard Cell Placement," *Proc. of the 27th IEEE Design Automation Conference*, pp. 20-25, 1990.

[22] C. Y. Lee, "An Algorithm for Path Connections and its Applications," *IRE-Trans. on Electronic Computers*, pp. 346-365, 1961.

[23] Y.-L. Lin, Y.-C. Hsu and F.-S. Tsai, "SILK: A Simulated Evolution Router," *IEEE Trans. on Computer-Aided Design*, Vol. 8, No. 10, pp. 1108-1114, Oct. 1989.

[24] S. Mohan and P. Mazumder, "Wolverines: Standard Cell Placement on a Network of Workstations," *IEEE Trans. on Computer-Aided Design*, Vol. 12, No. 9, pp. 1312-1326, Sept. 1993.

[25] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Artificial Intelligence, Berlin: Springer Verlag, 1992.

[26] Z. Michalewicz, "A Hierarchy of Evolution Programs: An Experimental Study," *Evolutionary Computation*, Vol. 1, No. 1, pp. 51-76, 1993.

[27] A. T. Rahmani and N. Ono, "A Genetic Algorithm for Channel Routing Problem," *Proc. of the Fifth International Conference on Genetic Algorithms*, pp. 494-498, July 1993.

[28] R. L. Rivest and C. M. Fiduccia, "A Greedy Channel Router," *Proc. of the 19th IEEE Design Automation Conference*, pp. 418-424, 1982.

[29] K. Shahookar and P. Mazumder, "GASP - A Genetic Algorithm for Standard Cell Placement," *Proc. of the European Design Automation Conference*, pp. 660-664, 1990.

[30] K. Shahookar and P. Mazumder, "A Genetic Approach to Standard Cell Placement using Meta-Genetic Parameter Optimization", *IEEE Trans. on Computer-Aided Design*, Vol. 9, No. 5, pp. 500-511, May 1990.

[31] K. Shahookar, W. Khamisani, P. Mazumder and S. M. Reddy, "Genetic Beam Search for Gate Matrix Layout," *Proc. of the 6th International Conference on VLSI Design*, pp. 208-213, Jan. 1993.

[32] H. Shin and A. Sangiovanni-Vincentelli, "A Detailed Router Based on Incremental Routing Modifications: Mighty," *IEEE Trans. on Computer-Aided Design*, Vol. 6, No. 6, pp. 942-955, Nov. 1987.

[33] T. G. Szymanski, "Dogleg Channel Routing is NP-complete," *IEEE Trans. on Computer-Aided Design*, Vol. 4, No. 1, pp. 31-41, Jan. 1985.

[34] T. Yoshimura and E. S. Kuh, "Efficient Algorithms for Channel Routing," *IEEE Trans. on Computer-Aided Design*, Vol. 1, No. 1, pp. 25-35, Jan. 1982.

[35] C. X. Zhang and D. A. Mlynski, "Ein Layout-Paket für Standardzellenschaltungen: Plazierung und Verdrahtung mit Evolutionsstrategie," *Proc. of Computer Science, Technology and Applications, COWITEAN'88*, Bonn, pp. 47-49, 1988.