A Genetic Algorithm for Global Improvement of Macrocell Layouts

K. Glasmacher, A. Hess, G. Zimmermann

Fachbereich Informatik
University of Kaiserslautern
6750 Kaiserslautern, FRG

Abstract

The result of many floorplanning algorithms is a placement of macrocells. A novel technique for the improvement of a given placement is presented which is based on the optimization of the channel densities by refining the cell positions. We introduce a distance function for each channel representing the channel width. This width can be altered by shifting adjacent cells along each other by an offset. We present an optimization algorithm to find offsets for adjacent cells which lead to a minimal area demand of the total layout. The method is based on a genetic algorithm, an iterative improvement procedure. Finally results are presented.

1 Introduction

In our design system PLAYOUT /Zim89/ the placement is controlled by the shape of the cells and the reduction of the resulting total netlength. This is common to many floorplanning approaches e.g MASON /LaD86/. The length of each net is estimated by some easy to calculate formula, for example the half perimeter of the bounding box of a net. During placement channel densities are estimated by heuristics. Global routing refines this estimation by assigning nets to channels, thus modifying the placement but not the topology. Even after this step it is unlikely that cells have optimal positions along channels in the sense that the resulting channel densities will yield an optimal layout. It can therefore be expected that the placement can be further improved by a subsequent compaction step which refines the channels. The problem is related to two-dimensional compaction, but the spacing rules are non-constant functions. Note that the global optimum cannot be achieved by minimization of an individual channel because this can enlarge adjacent channels (see figure 1).

/She89/ proposed a method called Hardening which refines a placement by optimizing individual channels between modules. Always two modules are laterally shifted by offsets which leads to a minimal bounding area of the compound module, thus optimizing the channel.

/SSS86/ presents a technique for two-dimensional layout compaction by "zone refining". This method is based on box packing, but incorporates spacing rules between boxes. Spacing is performed by updating the actual wiring immediately when moving a box.

In our approach spacing rules are defined by non-linear distance functions: Given an initial placement it is assumed that the density of a channel can be altered by shifting adjacent cells along the channel by offsets. If we do not change the result of the global routing the number of nets passing the channel remains unchanged. Thus the channel density is only influenced by the position of the connected pins on the sides of the channel. The necessary channel width for each offset depends on the calculated maximal channel density, with respect to the detailed-routing model and the design rules. The width as a function of the offset is called the distance function. Distance functions for all channels can be estimated.

Given offset values of adjacent cells for all channels we describe a cluster technique to evaluate a valid layout. A layout is called valid if the values of the distance functions (corresponding to the actual offsets) are less than or equal to the actual distances of cells. In the PLAYOUT approach the partitioning in the floorplan is represented by a slicing structure. Clustering is performed bottom up the slicing tree with respect to the distance functions of the channels along the slicing lines. As the result of this technique, a valid layout can be calculated uniquely for any set of offset values, with an additional property: For each slicing line there exists at least one pair of adjacent cells for which the value of the distance function is equal to the cell distance. For the others the cell distances are greater than or equal to the values of the corresponding distance functions.

Channels are treated simultaneously by altering all offsets at the same time. The goal of the optimization is to find offsets which lead to an optimal valid layout with respect to an objective function. In our approach the objective function combines area demand and aspect ratio of the resulting layout. We present a new genetic algorithm that approaches the optimal solution with reasonable runtime and good results. The algorithm starts by determining a set of individuals building the initial population. Each

CH3040-3/91/0000/0306$01.00 © 1991 IEEE

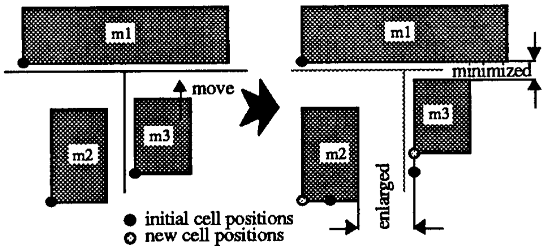


Figure 1: Local channel minimization

individual corresponds to a valid layout with random offset values for each cell. With genetic methods new individuals are generated from the old ones. The corresponding layouts of all individuals are scored with the objective function. By a natural selection strategy a portion of individuals leading to unfavorable layouts are eliminated. The remaining individuals build the new population. The last step is iterated until a solution with sufficient quality is reached. The experimentals show that this algorithm asymptotically approaches the optimum.

2 The problem

Given a set of macrocells M = {m1, ...,mn} with fixed shape and fixed pin positions and a floorplan which is defined by a fully orientated slicing tree and initial positions for all cells. Further given a channel graph CG = (V, E) where V is the set of vertices representing intersections of slicing lines and a set of edges $E \subseteq V \times V$. An edge connects two vertices which are neighbors on the same slicing line and corresponds to a channel. Further, a netlist N is given. By assigning nets to channels for each net $ni \in N$ a global route gr is defined by a set of edges of the channel graph: $gr(ni) \subseteq E$. Figure 2 illustrates a floorplan based on a slicing strucure. The slicing lines are partitioned by the vertices v1, ...,v14 of the channel graph. The boxes enclosed by the slicing lines mark the space of the modules: The white space marks the estimated routing area and the dark boxes indicate the macrocells. The polygon which connects the cells m2 and m6 shows a global route for a net. The net is assigned to the edges (v2,v6), (v6,v5), (v5,v10) and (v10,v11). The problem is finding a placement with at most slight changes of the overall topology which does not change the result of global routing and requires minimal area for a given aspect ratio.

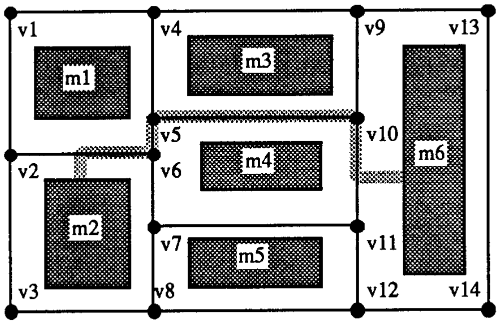


Figure 2: A floorplan with 6 cells

The term "slight changes" is illustrated in figure 3: If we propose the shown placement it would not be possible to redraw the vertical slicing line as defined by the topology. This situation is captured by introducing jogs on slicing lines as indicated by the bold line. This conserves the topology and the global routing solution. Note that exchanging cell m1 by m2 alters the orientation in the slicing tree and makes the global routing invalid.

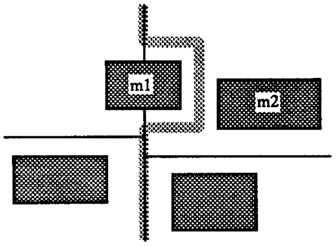


Figure 3: Slicing line with jogs

3 Measuring distances

For each channel of the channel graph, vertical and horizontal neighborhoods are defined for adjacent cells. Along vertical (horizontal) slicing lines horizontal (vertical) neighborhoods are defined. In addition to this we distinguish overlapping and non-overlapping neighborhoods depending on the actual positions of the cells. Two cells which are adjacent to a vertical (horizontal) slicing line overlap if a horizontal (vertical) line exists which cuts both cells. Figure 4 shows neighborhoods between cells.

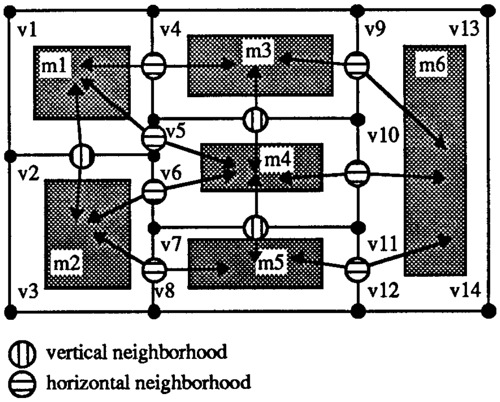


Figure 4: Neighborhoods defined by the floorplan

In order to determine channel widths we distinguish two kinds of channels: In the first case overlapping cells define a critical area of a channel. A maximal density can be calculated, which determines the minimal channel width. Note, that the maximal channel density is the maximum of the densities along the channel. In figure 5 the darkly shaded regions mark the critical area of channels a

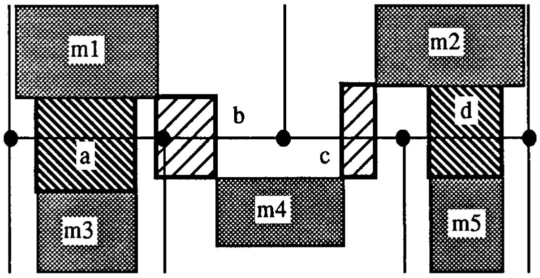


Figure 5:Two kinds of channels between neighboring cells

and d. Inside the critical area a maximal density can be calculated. The second case captures channels where neighboring cells do not overlap. In this case so called passing areas are defined (the lightly shaded regions in figure 5). The maximal density for the passing area is equal to the number of nets which pass this area in the direction of the slicing line. Note that the minimum distance is zero if no nets pass this area.

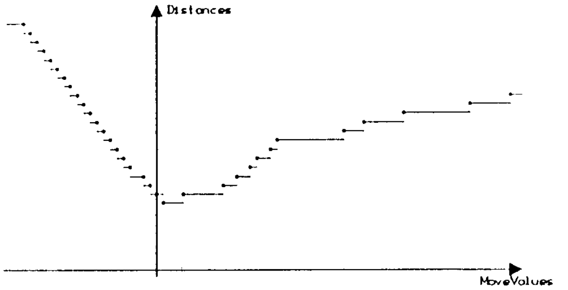


Figure 6: A distance function

We introduce a distance function d(ro) for each channel, which returns a distance value for some relative offset ro. One distance function is defined for a set of offsets for hypothetical shifts of neighboring cells along the slicing line within a limited range. The offsets are relative to the initial cell positions. An example for a distance function is shown in figure 6: If we propose an offset o1 for cell m1 and o2 for cell m2 then the relative offset between both cells is defined as ro = o1 - o2.

Figure 7a shows an example for an initial placement of neighboring cells m1 and m2. The shaded region marks the critical area. In this case the distance function returns a minimum width of three tracks. Figure 7b illustrates a lateral shift of m1 by an offset o1. In this case the distance function will return a minimum width of two tracks when assuming a manhattan routing model with two layers. The actual analysis of the densities is not further discussed. The distances which result from the densities depend on the basic routing model. We refer to /Len90a/.

Figure 8 illustrates the distance function for the passing area of channel (v1,v2) which is marked by the shaded region. Only nets n1 and n2 are passing this region. Thus the distance function returns two tracks as the minimal width. In figure 8b the distance between m1 and m2 has been reduced to two tracks by altering the detailed routing. Note that the environment was ignored for this operation. The lightly shaded cells in figure 8a are no neighbors by definition because they are not adjacent in relation to any channel. Experimental results have shown that these neighborhoods must be added in some cases. Further work is investigated on this.

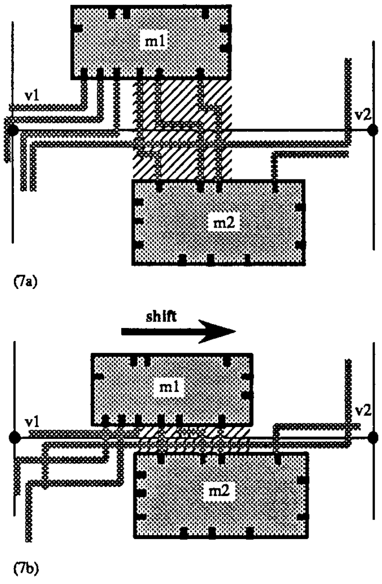


Figure 7: The critical area of neighboring cells

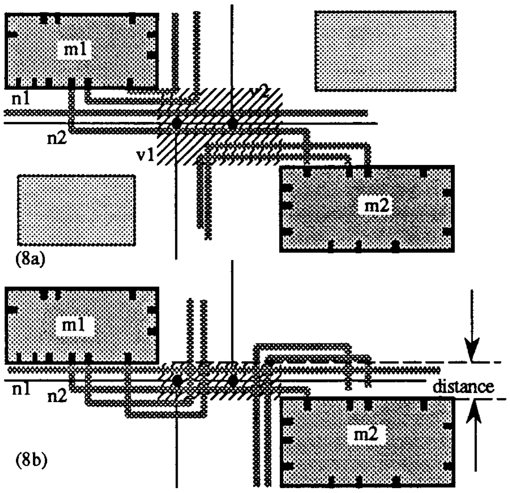


Figure 8: The passing area of neighboring cells

By assuming hypothetical offsets the distance functions can be precalculated because they only depend on the initial placement and the global routing. In order to allow fast access we generate function tables. In /TaT88/ and /Len90b/ techniques for minimization of channel widths are presented. The calculation of the distance functions must only be done once for a floorplan and remains constant during the optimization.

4 Computation of a valid layout

During the optimization an offset is proposed for each cell. Therefore it is necessary to set up a strategy which leads to a valid layout for the proposed offsets of all cells. A layout is called valid if for every defined neighborhood the actual distance of the cells is greater than or equal to the value returned by the distance function. We describe a cluster algorithm for the computation of valid layouts. Clustering is performed bottom up the slicing tree (figure 9):

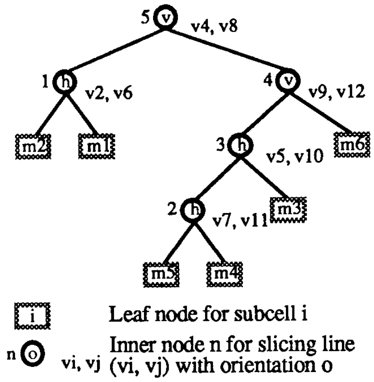


Figure 9: The slicing tree for the floorplan in figure 2

Initial step:

For all inner nodes of the slicing tree which are composed of two leaf cells, generate a cluster (nodes 1 and 2 of the slicing tree in figure 9):

- Shift the neighboring cells by the given offsets along the slicing line (note that only one channel corresponds to this slice).
- Determine the minimum distance between the cells by evaluating the distance function.
- Correct the position of the cells such that the actual distance accords to the minimum distance.

The initial step sets up partial layouts which are independent of each other. Figure 10 illustrates this step: Figure 10a shows adjacent cells on initial positions. In Figure 10b cell m2 is laterally shifted by an offset while the position of cell m1 remains unchanged. Thus the relative offset ro corresponds to the offset for cell m2. In figure 10c the width of the channel between m1 and m2 is set to the value of d(ro).

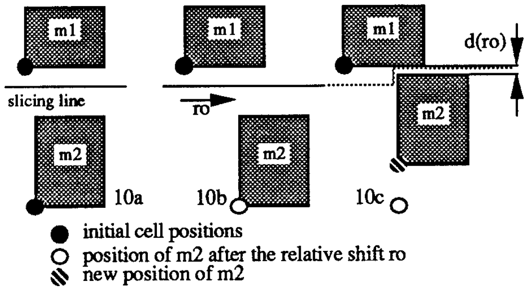


Figure 10: The initial step of the cluster algorithm

Recursion step:

If a cluster for an inner node has not been built and the partial layouts for the sons exist, then:

- Select this node.
- Select one cell from each cluster of the son nodes which are neighbors by definition.
- Adjust the partial layouts along the slicing line such that the positions of the selected cells reach their initial positions.
- Shift the partial layouts by the given offsets for the selected cells along the slicing line.
- Compute the actual offsets for all neighboring cells along the slicing line.
- Determine the minimum distances for all neighbors from their distance functions and compute the differences to the actual distances.
- Correct the position of the partial layouts by the maximum of the differences.

Termination:

By generating the cluster for the root the algorithm terminates. Note that the estimation of the border channels must be done during an additional step which is not discussed here. By omitting the border channels the layout is represented by the enclosing polygon for further steps, as shown in figure 12.

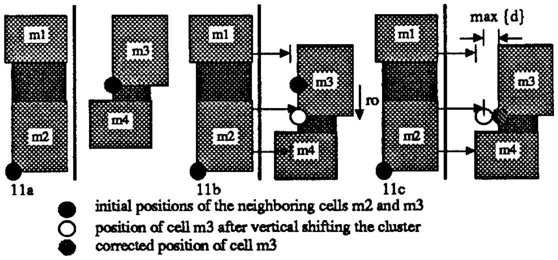


Figure 11: Recursion step of the cluster algorithm

In our example in figure 9 clustering of the inner nodes is performed in the order 3, 4, 5. Figure 11 illustrates the recursion step. Figure 11a shows two partial layouts which shall be clustered. Cells m2 and m3 are selected from the clusters. The initial positions of the selected cells determine the positions of the partial layouts. Figure 11b illustrates a lateral shift of the right cluster along the vertical slicing line, which is equal to the relative offset ro. The minimal distances for all neighborhoods along the vertical slicing line are determined from the distance functions and the differences to the actual distances are computed. In figure 11c the position of the right partial layout is corrected by the maximum of the differences max{d}.

5 The objective function

In order to compare layouts, the quality of a layout is scored by a single value. We introduce an objective function f which combines three criteria for any layout:

- the aspect ratio AR of the bounding box
- the area AP of the enclosing polygon
- the area AB of the bounding box

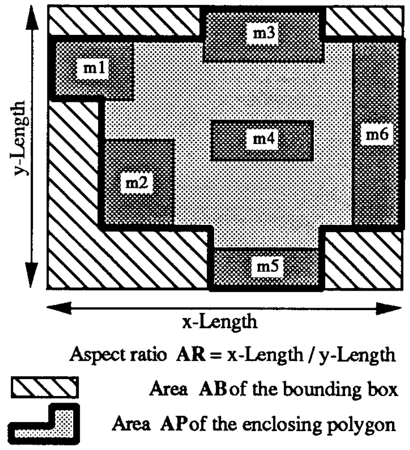


Figure 12: The objective function

The first criterion aims at the preservation of a given aspect ratio ARg: The term ARg /AR calculates to one if the given aspect ratio is reached. Otherwise a factor x < 1 is introduced. The area of the bounding box and the area of the enclosing polygon (see figure 12) are weighted against the minimal possible area. This is the sum A of the areas of the macro cells. The following objective function defines the quality of a layout:

$$f(AR, AP, AB) := (A / AP + A / AB) * x,$$

$$x := \text{if } ARg < AR \text{ then } ARg / AR \text{ else } AR / ARg$$

Short analysis of f:

$$A / AP \leq 1, A / AB \leq 1, x \leq 1 \longrightarrow f \leq 2.$$

The best quality of the layout can be reached by maximizing the objective function.

6 Genetic Algorithms

Genetic Algorithms are derived from processes of natural genetics. The principles were presented in /Hol75/. More detailed techniques for application are presented in /Gol89/.

Terms used in Genetic Algorithms are: population, generation, individual and gene. A population at a fixed time is called a generation. Every individual of a population consists of a fixed number of genes, which characterized the individual.

New individuals are built from the old ones by the genetic operators crossover and mutation. A new generation is built by reproduction: genetic operators are applied to selected individuals of the old generation. The new generation is set up from individuals of the old generation and the reproduced ones.

The outline of the algotithm:

```
Compute (Generation[0]);   {initial population}
ComputeFitness (Generation[0]);
FOR ActGeneration := 1 TO NoOfGenerations DO
    Reproduction (Generation[ActGeneration-1], Indi-
            vidual_List);
    FOR Count := 1 TO PopulationSize DIV 2 DO
        Crossover (Individual_List, Child_1, Child_2);
        Insert (Child_1, Children_List);
        Insert (Child_2, Children_List);
    ENDDO;
    Mutation (Children_List);
    Mixing (Generation[ActGeneration-1],
            Children_List, Generation[ActGeneration]);
    ComputeFitness (Generation[ActGeneration]);
ENDDO;
```

The individuals of each generation are scored by a fitness value. The result of the algorithm is the individual with the overall best fitness value of all generations. The quality of the algorithm is influenced by the size of the population, the number of generations and the probabilities for crossover and mutation. These parameters are set up at the start of the algorithm.

7 The optimization procedure

The goal of the optimization is to find the best layout according to the objective function. The optimization procedure is defined by the algorithm presented in chapter 6. The procedure starts by determining an initial population which consists of a fixed number of individuals. As we have shown a unique valid layout can be computed for each set of offset values. We call an individual the representation of a layout. An individual consists of one gene for every slicing line with offset values for one pair of cells. An example of an individual is shown in figure 13, which refers to the floorplan in figure 4.

gene	1	2	3	4	5
slice	v2, v6	v7, v11	v5, v10	v9, v12	v4, v8
channel	v2, v6	v7, v11	v5, v10	v11, v12	v6, v7
cells	m2, m1	m5, m4	m4, m3	m5, m6	m2, m4
offsets	(om2,om1)	(om5,om4)	(om4,om3)	(om5,om6)	(om2,om4)

Figure 13: An individual

Algorithm:

Step 1: INITIAL POPULATION

- Set up genes of all individuals by randomly selecting one channel along the appropriate slicing line. Generate random offsets for the neighbor cells of the selected channel.
- Compute the layouts represented by the individuals and determine the "fitness" values (values of the objective function) of each individual.

Step 2: REPRODUCTION

- Set up a mating pool by selecting individuals from the actual generation. The individuals are selected at random by prefering those with high fitness values. Thus individuals can repeatedly selected. The size of the mating pool is choosen equal to the size of the population.

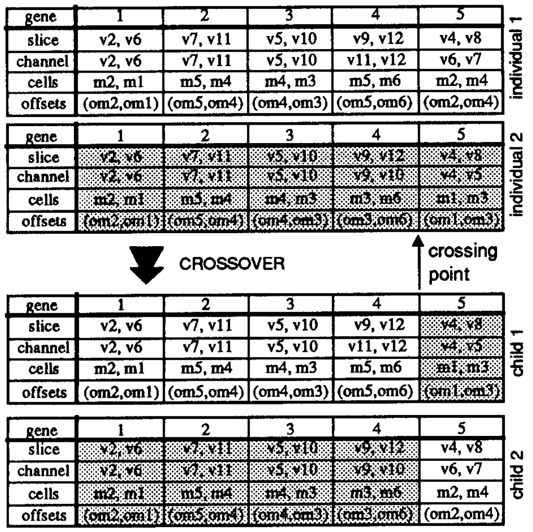
Step 3: CROSSOVER

While the mating pool is not empty do:

- Select two individuals at random and remove them from the mating pool.
- With a given probability CrP (set to 85 %) crossover is performed and the resulting individuals are inserted into the children pool. If crossover is not performed, the original individuals themselves are inserted.

Note that each individual has the same number of genes and that genes at the same position of different individuals correspond to the same slicing line.

To cross to individuals determine a crossing point at random. The crossing point divides both individuals into two segments. Two new individuals are generated by concatenating the first segment of one individual with the second segment of the other and vice versa (see figure 14).



gene	1	2	3	4	5
slice	v2, v6	v7, v11	v5, v10	v9, v12	v4, v8
channel	v2, v6	v7, v11	v5, v10	v11, v12	v6, v7
cells	m2, m1	m5, m4	m4, m3	m5, m6	m2, m4
offsets	(om2,om1)	(om5,om4)	(om4,om3)	(om5,om6)	(om2,om4)

gene	1	2	3	4	5
slice	v2, v6	v7, v11	v5, v10	v9, v12	v4, v8
channel	v2, v6	v7, v11	v5, v10	v9, v10	v4, v5
cells	m2, m1	m5, m4	m4, m3	m3, m6	m1, m3
offsets	(om2,om1)	(om5,om4)	(om4,om3)	(om3,om6)	(om1,om3)

gene	1	2	3	4	5
slice	v2, v6	v7, v11	v5, v10	v9, v12	v4, v8
channel	v2, v6	v7, v11	v5, v10	v11, v12	v4, v5
cells	m2, m1	m5, m4	m4, m3	m5, m6	m1, m3
offsets	(om2,om1)	(om5,om4)	(om4,om3)	(om5,om6)	(om1,om3)

gene	1	2	3	4	5
slice	v2, v6	v7, v11	v5, v10	v9, v12	v4, v8
channel	v2, v6	v7, v11	v5, v10	v9, v10	v6, v7
cells	m2, m1	m5, m4	m4, m3	m3, m6	m2, m4
offsets	(om2,om1)	(om5,om4)	(om4,om3)	(om3,om6)	(om2,om4)

Figure 14: The crossover

Step 4: MUTATION

Apply a mutation to the individuals of the children pool.

Mutation of an individual is done by modifying the genes. For every gene it is decided with a given probability MuP (set to 6%) if it should be modified. A gene is modified by selecting a new channel along the slice and new offsets for the corresponding cells at random.

Step 5: MIXING

- Compute the fitness values of the individuals in the children pool.
- Build the next generation by the following selection strategy:
    - Choose a fixed number of best individuals from the actual generation and the children pool (25% from each).
    - Choose a fixed number from the remaining individuals of the actual generation and the children pool at random (25% from each pool).

  The selected individuals set up the new actual generation.
- If a stopping criterion is not reached goto Step 2.

Currently a fixed number of populations is generated.

The resulting layout is represented by the individual of the last generation with the best fitness value.

8 Results

The implementation of the presented algorithm has been carried out on APOLLO/DOMAIN workstations under DOMAIN/OS /Hes90/. The implementation language is APOLLO/DOMAIN PASCAL. The tool has been integrated in the PLAYOUT design system which serves as an environment for design data and other tools (e.g. Chip Planning, Cell Synthesis, ...).

The presented example in figure 15 consists of seven macrocells. The corresponding floorplan was a result of a preceding Chip Planning phase in PLAYOUT and the macrocells were assembled by the subsequent Cell Synthesis. Figure 15a shows the floorplan with initial positions of the macrocells. Figure 15b shows the solution of the optimization procedure with the result of detailed routing. Until now the border channels have been omitted.

Table 1 shows seven runs of the optimization procedure with different sets of control parameters. The result of optimization has been measured by the aspect ratio AR, the area of the enclosing polygon AP and the area of the bounding box AB. In addition the deviation from the ini-

Table 1:
Table 1a shows the control parameters.
Table 1b shows the results of the optimization (*: After 25 populations no more changes occur and thus the algorithm terminates).
The results of run 4 are shown in figure 15 and 16.

run	PopSize	PopNum	CrP	MuP
1	50	100	0.85	0.08
2	100	100	0.9	0.1
3	100	50	0.7	0.06
4	200	100	0.9	0.08
5	50	50	0.85	0.06
6	50	50	0	1
7	50	50	1	0

(a)

run	time (ms)	AR	Δ AR (%)	AP (nm)^2	Δ AP (%)	AB (nm)^2	Δ AB (%)
1	72098	3.60	14.9	6.15 e 12	23.0	8.91 e 12	0.1
2	204059	3.53	16.5	6.20 e 12	22.4	8.73 e 12	2.1
3	101241	3.54	16.3	6.27 e 12	21.5	8.81 e 12	1.2
4	628037	3.56	15.8	6.22 e 12	22.2	8.71 e 12	2.4
5	36351	3.53	16.5	6.31 e 12	21.0	8.73 e 12	2.1
6	41453	3.83	9.5	6.85 e 12	14.3	9.47 e 12	6.2
7	15422 *	3.98	5.9	6.62 e 12	17.1	9.83 e 12	10.2
original values		4.23		7.99 e 12		8.92 e 12	

(b)

tial values is shown. In our example the algorithm leads to an improvement on an average of 22 % in polygon area by a deviation of 16 % from the aspect ratio (mean values of runs 1 to 5). Runs 6 and 7 show the result if either CrP or MuP is set to zero, which will not lead to a satisfying solution.

The correctness of the final layout in sense of routable channels depends on the distance functions. Experimental results have shown that in most cases detailed routing can be performed without enlarging channels.

The main optimization procedure is controlled by the following parameters which can be interactively set:

- The size of population: PopSize
- The number of populations: PopNum
- crossover probability: CrP
- mutation probability: MuP

The values in table 1 show typical sets of control parameters. In the given example the size of the population does not much influence the overall improvement. A common set of control parameters is unknown until now.

The illustrated diagram in figure 16 shows the fitness values of the individuals in the order of their creation time for the first set of control parameters. The outlined curve connects the best increasing fitness values. The gradient of the curve decreases with time. Normally the algorithm can be interrupted when the outlined curve becomes flat. A further improvement is unprobable.

Further work has been done on setting up only partial function tables which speeds up the computation of the distance functions. The tables include the local extrema of the distance functions. We use a heuristic approach for finding local extrema /Web89/. The heuristic is based on best abutment, minimal netlength, and a so called puzzle test but will not be further discussed. During optimization the distance values for undefined offsets are approximated. Results of this method will be reported.

9 Conclusions

An optimization algorithm for the global improvement of macrocell layouts was presented. The technique uses iterative improvement by genetic methods.

During most floorplanning approaches the global

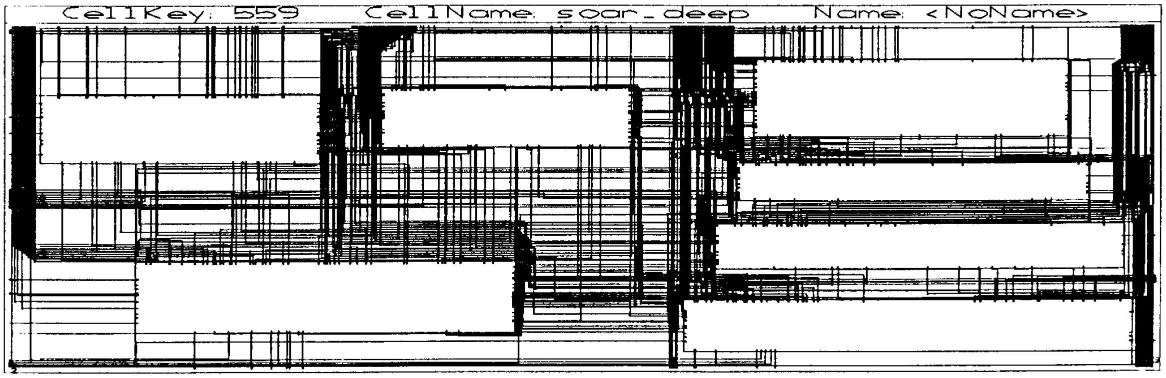


Figure 15a: The floorplan with initial positions of the macrocells and the result of detailed routing

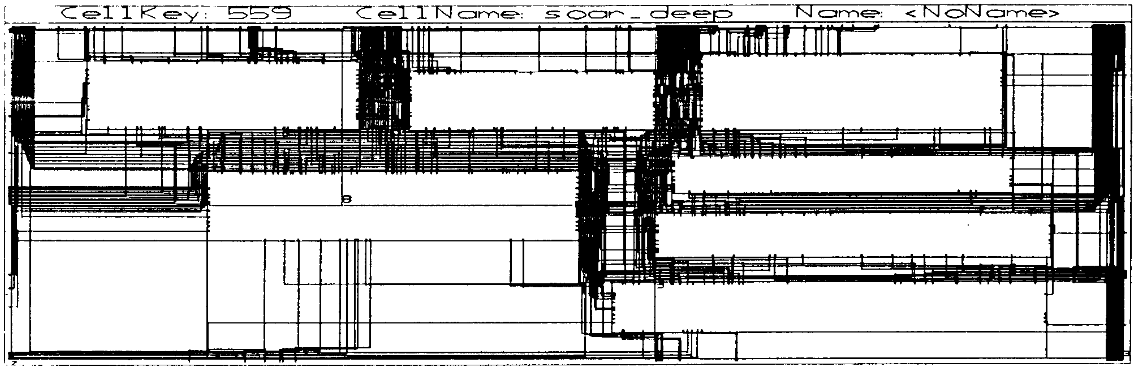


Figure 15b: The solution of the optimization procedure (run 4) with the result of the detailed routing

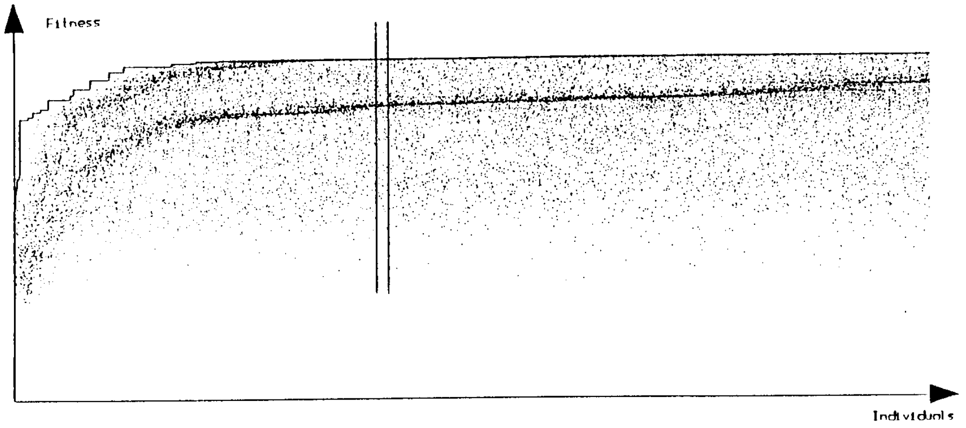


Figure 16: The fitness values of the individuals: As an example the individuals of one population are enclosed by two vertical lines.

routing determines a good estimation of the routing space and corrected cell positions. We assume that the routing space can be further reduced by looking up the channels to find new cell positions which lead to minimal channel widths. In our approach we deal with all channel widths simultaneously in order to find a global optimum according to the overall routing area.

Especially floorplanning approaches which deal with flexible cells, like Chip Planning in PLAYOUT, lead to layouts which can be further improved because the final masklayouts of cells are unknown during the planning phase. In this case our approach should improve the layout without changing the found topology and the result of global routing.

First results have shown that our approach leads to reasonable runtime and good results. Further work will be done on calculating more accurate distance functions, tuning of the control parameters and speeding up the algorithm by utilizing the implicit parallelism of this method.

References

/Gol89/ David E. Goldberg, "Genetic Algorithms in Search, Optimization, and Machine Learning, Addison Wesley, 1989

/Hes90/ Axel Heß, "Chip-Assembly Strategien bei der Layout-Synthese nach Floorplan-Vorgaben", diploma thesis, University of Kaiserslautern F.R.G., 1990

/Hol75/ J.H. Holland, "Adaption in Natural and Artificial Systems", University of Michigan Press, Ann Arbor, MI, 1975

/LaD86/ D. LaPotin and S. Director, "Mason: A Global Floorplanning Approach For VLSI Design", IEEE Transaction on CAD, vol CAD-5, no. 4, 477-489, 1986

/Len90a/ Thomas Lengauer, "Combinatorical Algorithms For Integrated Circuit Layout", John Wiley & Sons Ltd, Chapter 9.1.3, 1990

/Len90b/ Thomas Lengauer, "Combinatorical Algorithms For Integrated Circuit Layout", John Wiley & Sons Ltd, Chapter 9.6 and 9.7, 1990

/She89/ Alan T. Sherman, "VLSI Placement And Routing: The PI Project", Springer-Verlag New York Inc., Chapter 5.3, 1989

/SSS86/ Hyunchul Shin, Alberto L.Sangiovanni-Vincentelli and Carlo H. Sequin, "Two-dimensional Compaction by 'Zone-Refining'", Proceedings 23rd DAC, 1986

/TaT88/ Roberto Tamassia and Ioannis G. Tollis, "On Improving Channel Routability By Lateral Shifting Of The Shores", SIGDA Newsletter, vol 18, number 1, year 1988

/Web89/ Harald Weber, "Abschätzung der Kanaldichte zwischen Zellen und Entwicklung geeigneter Distanzoptimierungsstrategien", internal report, University of Kaiserslautern F.R.G., 1989

/Zim89/ Gerhard Zimmermann, "PLAYOUT - A Hierarchical Design System", Information Processing 89, IFIP, 1989