Results given by a New Evaluation System for Placement and Routing Heuristics

Reinhard Rauscher, Dieter Klawan[1], Hans-Jürgen Bandelt[2]
University of Hamburg
Dept. of Computer Science
Hamburg, Germany
e-mail: rauscher@informatik.uni-hamburg.de

Abstract

This paper presents a software system for generation of VLSI layouts. Its main advantages are minimal hardware requirements and its independence of programming languages, hardware platforms and fabrication processes. It is thus designed for use in algorithm development and universitary backgrounds.

1. Introduction

Up to now developers of VLSI layout algorithms had to choose between two possibilities:

- to handle the problem purely theoretically, i.e. to ignore practical constraints or
- to face the whole problem complexity, i.e. to deal with complex design rules and a specific fabrication process.

This led to two kinds of algorithms separated by a large gap: On one hand algorithms wgich implement novel techniques without concerning the realization in practice and on the other which are commercial tools that took years of implementation and thus are no longer state-of-the-art regarding the employed algorithms. The integration of improvements especially in the field of local-search heuristics into VLSI layout is extremely obstructed by this fact.

The goal of our project was to improve this situation by providing a software tool that allows easy implementation, evaluation, and comparison of new heuristics with randomly generated as well as with practical data. The system that evolved from this idea is called ESPRO (Evaluating System for Placement and ROuting). It consists of several stand-alone programs that interact through ASCII data files.
The basic approach of ESPRO is similiar to that in [17].

[1] D. Klawan, Univ. of Hamburg, Dept. of Psychology, Von-Melle-Park 6, D-20146 Hamburg

[2] H.-J. Bandelt, Univ. of Hamburg, Dept. of Mathematics, Bundesstr. 55, D-20146 Hamburg

The system ESPRO permits a broad comparison of different implementations of algorithms (standard techniques as well as new approaches). In contrast to the 'duels' commonly fought out in the literature here the problem model is the same for all competitors and all programs work on the same data. Therefore, such a comparison is much more meaningful and informative than the usual ones.

Last but not least, ESPRO can operate as an aid for bringing together scientists from different disciplines and stimulating the exchange of new ideas. It can be used by mathematicians working on local-search heuristics as well as by computer scientists dealing with placement and routing. In this respect this system might lead to a closer co-operation of both disciplines.

2. The underlying model

The goal of our approach is to keep the problem environment for the programs as realistic as possible while getting rid of unnecessary details. This leads to a system that produces meaningful results, but on the other hand is easy to use.

In addition, we decided to eliminate all fabrication depending details. In this way the system itself becomes independent from a specific fabrication process and yields longer lifetime without a need of adaption to changing technology.

The information about the library cells is purely geometrical. Namely, cells and pads are rectangular shapes with defined height and width and a specific number of pins but without any information about the interior function. A cell has no identical pins, and routing-over-the-cell is forbidden. This yields independence from specific cell libraries.

Wires are lines of equal width lying on a rectangular grid of nonspecified size. The wire area is discretized as a grid of constant unit distance. The components and the wiring segments are placed within two wiring layers. At every layer change within a net a via is inserted. Power supply wiring is neglected completely. The sizes of pads and cells are given as integer values with respect to the wire grid.

Feed-throughs are modeled by using the space between the cells of a cell row. The number of feed-throughs between two cells is limited only by the existing space between them. Feed-throughs must match the wire grid.

3. System overview

In the current state of progress the system contains only tools for standard-cell layout.

In defining the conceptual structure of the program system the following demands have been considered:

- The system should be extendable in an incremental manner. This means that users of the system should be able to add their own programs performing their preferred heuristics.
- The system should be available on a usual PC, but must be portable to any other hardware platform. The first requirement is necessary especially for teaching aspects, while the second one is useful if time consuming parts have to run, for example, on a workstation.

In responding to these demands, the system consists of a set of stand-alone programs. All programs can be used in an interactive mode, optionally. In this mode, the user can change a set of parameters (e.g. decreasing rate of temperature in simulated annealing [12]).

Apart from the in- and output procedures, which must be able to read and write ESPRO files, all the rest of the program including the choice of the hardware platform or the programming language is completely up to the programmer. Even old existing programs can be integrated by writing converting routines between the in- and output file formats.

The overview of the system (and the program communication flow) is shown in figure 1. In each step the user can choose one of the existing programs, or include (and evaluate) his own programs (or heuristics).

The design examples can be either generated randomly or converted from commercial tools. In the second case the input of ESPRO is contained in a so called *foreign file*.

In the second step a partitioning of the cells can be performed optionally.

The next step runs one of the implemented placement procedures. The result is stored in a so called *placement file*.

For the routing procedure the user can also choose between a set of implemented heuristics. The result is stored in a *layout file*. Additionally, some informations concerning bottlenecks in the current routing solution are stored in a *feedback file*.

Optionally, a postoptimization can be performed.

One important effect of the open system architecture of ESPRO is that there are not prespecified objective functions.

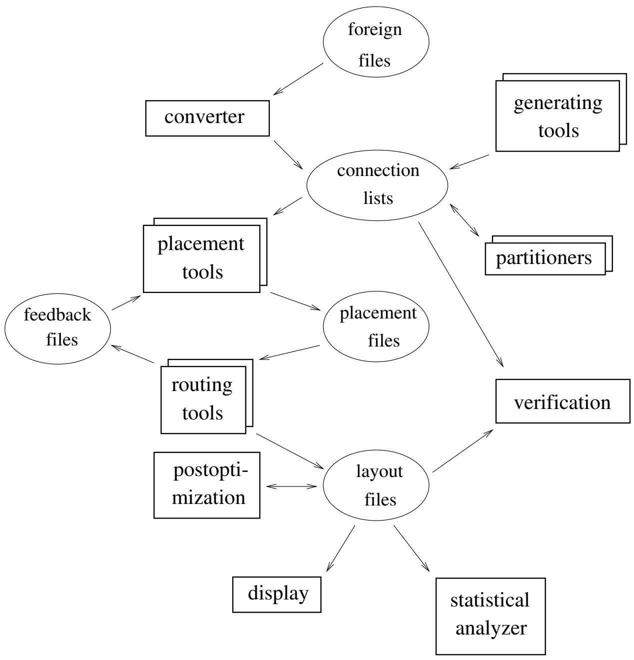

Figure 1. System overview of ESPRO

This concerns placement and routing routines as well as the postoptimization tools. In this way, alternative and innovative preferences or weightings of multiple objectives can be realized.

Another feature of the system is the option of generating a set of statistical informations. comparing two different layouts, determing the number of nets This includes data from the connection lists and the analysis of complete layouts. Employing these informations one can create artificial chips for testing purposes that resemble real world chips more closely.

The last option is to display the resulting layout on the monitor.

In the following section the different programs are described in some more detail.

4. Existing programs

Until now only a few programs are actually implemented (in a prototype form) due to the early stage of system development. For a detailed description see [13].

All programs existing at the moment are written in C and are running under MS-DOS on Intel-PC's. Using this hardware platform they are able to handle data up to a size of about 10,000 cells within acceptable time. For storage intensive tasks there is the opportunity to run the programs under UNIX.

According to figure 1 the programs are classified as follows:

4.1. Converter

The existing converting routine is a tool that converts net list files from the SOLO 1400 format into the ESPRO connection lists format. SOLO 1400 [4] is a commercial layout system by European Silicon Structures Ltd. Optionally, a file with statistical informations about the connection lists can be generated. This stores the total numbers of cells, wires, mean cell and net sizes, and average number of pins per net.

4.2. Generating tools

Two tools for generating connection lists are available. The first one requires the entire connection lists as an input from the user and therefore is appropriate only for small test instances. Each detail of the circuit may be specified by the user.

The second synthesis tool only requires a few key parameters such as the number of elements or the maximum cell and net sizes and generates the connection lists randomly based on these values. This tool can be used to generate large test instances of up to 10,000 cells or more.

4.3. Partitioners

Two partitioning heuristics have been implemented (see [5]).

The first one is a simple greedy local-search heuristic whereas the other one additionally employs a genetic algorithm. The objective in both is to divide the set of cells into several subsets of nearly equal size while minimizing the number of nets connecting different subsets. These programs can assist to reduce the problem complexity of large circuits by obtaining smaller parts that can be placed and routed separately.

4.4. Placement tools

In order to generate initial placements for local-search heuristics, ESPRO contains a random placement routine.

Next, there is a local-search placement heuristic (GDPlacer) based on the 'Great Deluge' algorithm by Dueck [2], [3]. This program starts from an initial placement stored in an ESPRO placement file as its input and produces a new placement file containing the improved placement. Additionally, this program can utilize information obtained in a preceding routing stage (e.g. about too long nets or overcrowded channels) in a new placement run. This information is contained in the feedback files mentioned in figure 1, which are produced by one of the routing modules.

Furthermore, there is a placement routine based on an algorithm that aims at minimizing the distances between connected cells (see [5]). It uses a "rubber band" model similar to those described in [18] and takes into account the positions of the pads. These can be preplaced by another tool also described in [5].

In progress is the implementation of some new placement methods recently mentioned in the literature (see [7]), e.g.:

- ML-Germinal, an approach based on the "dynamic cloud" method [1],
- an approach based on fuzzy logic [15].

Finally the well-known 'Timber Wolf' [16] package for placement and global routing (TWPlacer) has been implemented (see [9]). This allows comparing the performance of new approaches with those of standard algorithms on the same data. A first prototype of a genetic placement heuristic (GENPlacer) [11] has been implemented.

4.5. Routing tools

One new routing approach (DKRouter) has been implemented that merges global and local routing into one step (see [13]). This deterministic algorithm aims at optimizing the routing path of each net in the inner loop while optimizing the order in which the nets are placed in the main loop.

As standard algorithms the 'Timber Wolf' [16] global router, the 'Lee' global routing algorithm and a greedy channel routing procedure [19] for local routing have been implemented (see [9]).

4.6. Postoptimization

The postoptimization phase one strives for:

- reducing the length of critical nets
- reducing the number of tracks for some channels
- reducing the number of vias
- reducing the critical areas [14]

for a given complete layout. Obviously, these four tasks are not independent from each other. The user has to specify his preferences in this multiobjective task.

The basic idea of the postoptimization procedure [10] is to substitute a connection between two points of a net by another one. This can be achieved for example by burying or floating net segments.

In the current state of the system three programs are available:

1. A statistical program determing the amount of vias, the critical area, the average density of the horizontal tracks, and so on.
2. A program performing a general postoptimization procedure as e.g. reduction of chip area, critical areas, the amount of vias, and so on.

3. A program performing a minimization of a given critical net.

Some results using these tools are described in figure 9 and figure 10.

4.7. Statistical analyser

Besides the statistical output concerning the connection lists that can be generated by the converting tool (see section 5.1) there is a program analyzing complete layouts. This routine likewise produces a file with statistical information, including the portion of the circuit area used by wiring, width and mean density of the channels, number of wire segments and vias per track, net lengths, etc. Such statistics, however, are usually difficult to obtain with a commercial tool.

4.8. Display

The display routine employs a Graphical User Interface. It allows zooming into a given layout, highlighting of selected nets, and printing of scaled views.

4.9. Verification

ESPRO can be used by a scientist for a comparison of his heuristic with other (standard) heuristics. So, ESPRO is concipated as an open system, allowing to integrate "own" additional programs. Furthermore, for debugging of our components a verification procedure has been considered necessary, which automatically compares the "connection lists" with the final layout and determines whether all nets have been implemeted correctly. Possible faults could be:

1. shorts between two nets
2. missing connections between at least two points of the actual net.
3. superfluous wire segments

Within the ESPRO system a first prototype of the verification tool [8] has been implemented and tested. This tool has not only been necessary to prove the correctness of the layout, but also for debugging purposes.

5. First results

We have used the 'Great Deluge' (GD) heuristic, the Timber Wolfe heuristic (TW), and a genetic heuristic (GEN) for placement. For routing we compared the new routing heuristic (DK) and a standard heuristic based on the well known Lee-Algorithm and a Greedy-procedure for channel routing.

So, we have been able to generate alternative layouts for parts of a few chips that were previously designed using a commercial tool. Further problem instances were created randomly and then subjected the same layout procedure.

To give an example, a portion of a resulting layout for an artificial chip with 500 cells is shown in figure 2 (as shown by the display routine).

chip	t GD	t TW	t GEN
a50	22	62	26
a100	54	2270	178
a200	141	913	364
a500	113	5732	3650
a1000	445	22688	2036
r45	8.4	40	63
r246	21	754	84
r276	29	1046	58
r690	57	3101	1921

Figure 3. Times for Placement heuristics (in sec.)

In figure 3 the CPU times in seconds for the three different placement heuristics is shown.

Figure 3 contains the results for one series of experiments on small artificial and real chips; the chip code indicates the number of cells and whether it is an artificial chip (a) or part of a real world chip (r).

One could argue, that partially there has been to much time spent using the TWPlacer, but its not an easy task to determine the CPU time the algorithm will take. For GDPlacer and especially GENPlacer the time can be estimated quite easily.

In figures 4 and 5 the CPU times for routing and the total chip area for the actual layouts are described. Some of the designs could not be routed using the choosen parameter set. These examples are marked by [1]. The reason for this is, that for each of the placement heuristics another amount of rows of standard cells, amount of necessary feed-throughs, and so on, is determined. So, the examples show, that neither one of the placement heuristics nor the routing heuristics could be considered to be best suited. One can see, that at least for these examples DKRouter produces better layouts, but in average uses a factor of 4 more CPU time.

For a dedicated comparison of two different placement heuristics we used the following approach. Given a set of examples, two different placement heuristics are performed using nearly the same CPU time. After this the same routing procedure (DKRouter) is executed and the resulting layouts

[1] Results concerning changes of the system parameters. Otherwise no solution has been detected.

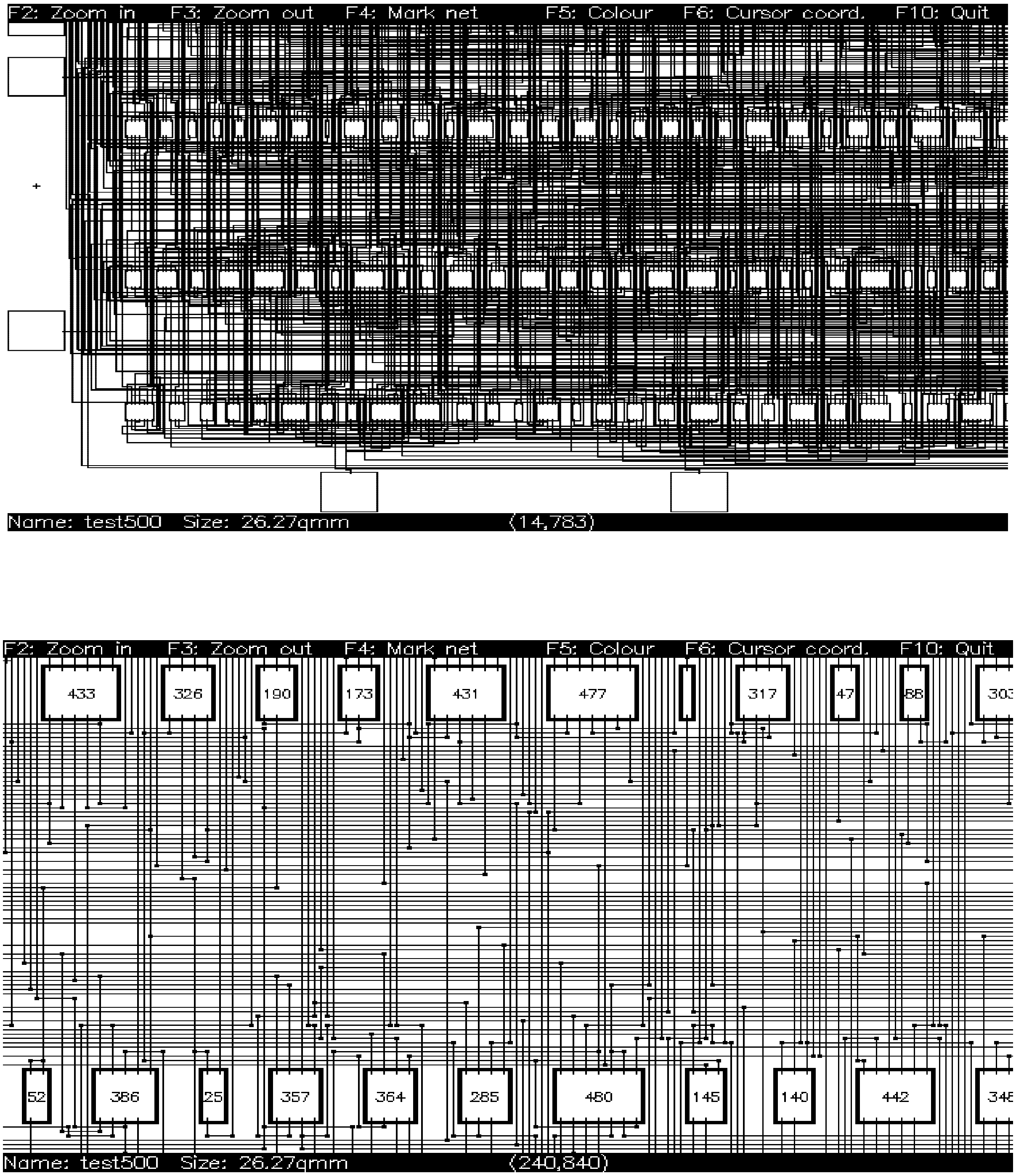

Figure 2. Minor part of a layout generated by ESPRO (prior to postoptimization), and a further zoomed section of a single channel

chip	a GD	t GD	a TW	t TW	a GEN	t GEN
a50	1.72	60	1.73	66	1.60	58
a100	3.35	171	3.24	177	3.12	171
a200	6.90	615	6.86	644	6.74	586
a500	26.12	4433	26.57	5591	25.88	4374
a1000	52.33	8800	54.11[1]	9770[1]	54.73	16794
r45	2.82	27	2.81	25	2.82	21
r246	4.31	126	3.80	74	4.31	149
r276	6.17	241	5.64	113	5.55	258
r690	32.96	1084	26.88	510	31.46	1215

Figure 4. Results (time in sec) in using DKRouter

chip	a GD	t GD	a TW	t TW	a GEN	t GEN
a50	1.75	14	1.80	60	1.84	10
a100	3.38	47	3.42	25	3.77	29
a200	7.02	80	6.73	150	7.00	83
a500	28.24[1]	604	27.89[1]	476[1]	28.78[1]	610
a1000	56.45[1]	2311[1]	57.58[1]	614[1]	58.13[1]	1310[1]
r45	2.82	7	2.81	5	2.93	4
r246	4.44	37	5.01[1]	10[1]	4.78	20
r276	6.32	69	6.78[1]	68[1]	5.89	19
r690	38.82	246	33.17[1]	986[1]	31.77	229

Figure 5. Results (time in sec) in using standard Algorithms LEE and GREEDY

are compared. In a first attempt we compared the GDPlacer with the GENPlacer.

The GENPlacer is based on the approach described in [11]. The underlying selection procedure is the Monte Carlo method, for recombination the PMX-procedure [11] is used, and reproduction is done by copying. The advantages/disadvantages of the GENPlacer are:

1. The CPU time spent can be determined a priori.
2. After each generation a valid solution is available.
3. Different criteria for termination (e.g. percentage of improvement, Number of generations, CPU time) can be integrated.
4. The disadvantages are the huge area for storage and the huge amount of time spent for the calculation of fitness.

In figure 6 a comparison of GDPlacer and the genetic heuristic is sketched. The procedure has been as follows: Using the GDPlacer (with its optimal parameter set) a placement is performed. Afterwards the GENPlacer is started given the same CPU time. At last the same router (DKRouter) is used for both procedures and the total layout is analyzed.

The results of figure 6 show, that GENPlacer is better suited for our examples than GDPlacer.

In figure 7 a comparison betweeen GDPlacer and TW-Placer has been performed. Here we tried to get the same

chip	a GEN	a GD	t GEN	t GD
a50	1.624	1.748	324	324
a100	3.115	3.387	734	745
r45	2.817	2.817	223	218
r276	5.36	5.87	970	972
a200	6.496	6.955	1086	1136
a500	25.526	26.485	2856	2688
r246	4.476	4.45	795	831
r690	27.434	28.375	1396	1360

Figure 6. Results comparing Great Deluge and a genetic heuristic

chip area, but different CPU times for both heuristics. The procedure has been as follows: Starting with the TW heuristic we analyzed the chip area and started to get (nearly) the same area by using GDPlacer and measured the time spent by GDPlacer. As a result of figure 7 a factor of 10 or more could be determined.

In practical designs oe usually distinguishes between 'prototyping layouts' and ' final layouts'. The first one is generated to give a first idea of the chip size, the second is performed to generate the final layout. Depending of these two classes we distinguish between 'fast layout generation' and 'time-consuming layout generation'.

chip	a TW	a GD	t TW	t GD
a50	1.814	1.744	216	131.54
a100	3.338	3.350	721	169.56
a200	6.937	6.78	917	297.5
a500	26.162	26.28	5758	316.5
r45	2.871	2.817	178	42.51
r246	4.014	3.95	787.09	57.8
r276	5.534	5.87	3967	98.9
r690	26.988	27.38	8648	220.03

Figure 7. Comparison of GDPlacer and TW-Placer (both combined with DKRouter)

In figure 8 a comparison between chip sets based on this distinction is described. We defined two different time areas for the placement and routing procedure. MIN indicates, that for P&R only a few minutes could be spent, HOU indicates, that several hours have been used. In figure 8 the advantage of GENPlacer in comparison to GDPlacer (in percent) is described using the usual criterions $I_{max}, I_d, and A_{ges}$.

I_{max} = maximal net length (in mm)
I_d = average net length (in mm)
A_{ges} = total chip area (incl. pads, in mm^2)

crit.	MIN	HOU
I_{max}	-11.7%	-11.6%
I_d	+9.9%	+9.6%
A_{ges}	+4.3%	+4.2%

Figure 8. Results comparing GDPlacer and GENPlacer

One result of figure 8 is, that only in the case that maximal net length is the most important feature of the design, GDPlacer is better suited that GENPlacer.

5.1. Results concerning the postoptimization procedure

After generating the chip layout an (optional) postoptimization procedure can be performed. There are two different main goals in postoptimization:

- Reduction of the amount of vias, reduction of chip area, reduction of critical area [14]
- Reduction of the maximal net length

For a first evaluation we made some experiments concerning improvement of the number of vias (impr.V), improvement of the chip area (impr. area) and the reduction of critical area (impr. CA).

chip	impr. V	impr. area	impr. CA
a50	229 / 24.8%	605 / 1.5%.	674 / 4.8%
a100	409 / 20.0%	654 / 0.7%	1634 / 3.93%
a200	599 / 15.9%	0 / 0%	2689 / 3.0%
r246	453 / 32.3%	0 / 0%	1440 / 4.3%
r276	626 / 29.7%	0 / 0%	2823 / 4.8%

Figure 9. Results of the general postoptimization procedure

In figure 9 the improvement is illustrated for some of the chips. For each item the improvement is determined absolutely and relatively. It can be seen, that the number of vias has been reduced by 20 % on average.

chip	impr. abs	impr. rel
a50	179	16.3%
a100	223	13.0%
a200	347	11.8%
r246	16	1.2%
r276	140	9.5%

Figure 10. Results of the postoptimization procedure for decreasing the length of one specified net

In figure 10 another optimization goal has been treated. Here the goal is to decrease the maximal net length. Figure 9 shows the absolute improvement (in $1/1000$ mm) and the relative improvement. The average improvement is roughly 10 %.

5.2. Overall discussion of the results

Although one should certainly not generalize from the results of a single experiment, we are confident that a tuned interplay of different strategies will eventually outperform single standard approaches.

A thorough testing of alternative heuristics for a certain phase of the layout procedure will, of course, require

- optimal implementation of all algorithms
- various parameter settings for the competing heuristics
- combination with different system components of earlier and later phases
- numerous test instances covering a wide range of potential layout problems.

6. Planned activities

The following activities are planned:

- fine-tuning and combination of implemented heuristics
- implementation of further heuristics
- evaluation of the heuristics for additional practical examples (benchmarks)
- improvement of the postoptimization procedure.
- widening of the application area to macrocell designs

7. Conclusions

The benefit for programmers joining ESPRO is the open system concept. It allows easy implementation while supporting most of the real world problem conditions. Excluded are only details depending on a specific fabrication process. With ESPRO it is possible to work with data of produced chips as well as to create artificial data with special characteristics to test the quality of a program under extreme conditions.

In a university environment, students of computer science or mathematics can therefore easily contribute to the existing program package (in form of a seminar work or diploma thesis) since creating system modules for ESPRO takes only a short training period. So, theroretical considerations [6] and practical aspects of VLSI layout need not necessarily drift apart in university teaching.

ESPRO thus provides a competitive scenario for algorithm development and further enhances thorough testing and fine-tuning of subroutines as well as improving the interplay of optimal program components.

References

[1] A. Dahmani, Y. Savaria, B. Kaminska: ML-Germinal: A New Heuristic Standard Cell Placement Algorithm, IEEE European Conference on Design Automation, pp 184-188, 1993

[2] G. Dueck, T. Scheuer, H.-M. Wallmeier: Toleranzschwelle und Sintflut: neue Ideen zur Optimierung, Spektrum der Wissenschaft, March 1993, pp 42-51

[3] G. Dueck: New Optimization Heuristics, The Great Deluge Algorithm and the Record-to-Record-Travel, Journal of Computational Physics, Vol. 104, pp 86-92, 1993

[4] European Silicon Structures Ltd., Solo 1400 Release 3.1 Reference Manual, Berkshire, 1991

[5] M. Gerloff: Clusterverfahren im VLSI-Standardzellenlayout, diploma thesis, University of Hamburg, 1995

[6] Th. Lengauer: Combinatorial Algorithms for Integrated Circuit Layout, Teubner, Stuttgart, 1990

[7] H. Jenke: Implementierung und Evaluierung von innovativen Heuristiken zum Placement & Routing, diploma thesis, University of Hamburg, 1996 (in preparation)

[8] O. Kahlbaum: Ein Tool zur Verifikation von Standardzellayouts, Diploma thesis, University of Hamburg, 1996, (in preparation)

[9] O. Kahlbaum: Implementierung und Evaluierung von Standard-Heuristiken zum Placement & Routing, Studienarbeit, University of Hamburg, Sept. 1995

[10] O. Kastek: Nachoptimierung von Standardzellentwürfen, diploma thesis, University of Hamburg, 1995

[11] W. Kinnebrock: Optimieren mit genetischen und selektiven Algorithmen, Oldenbourg, 1994

[12] S. Kirkpatrick, C. Gelatt, M. Cecchi: Optimization by Simulated Annealing, Science, Vol. 200, pp 671-680, 1983

[13] D. Klawan: ESPRO - Ein Programmpaket für Placement & Routing beim Standardzellen-Entwurf, diploma thesis, University of Hamburg, 1994

[14] S. Kuo: YOR: A Yield Optimizing Routing Algorithm by Minimizing Critical Areas and Vias, Proceedings European Conference on Design Automation, Bruxelles, pp 525-529, 1992

[15] R. Lin, E. Shragowitz: Fuzzy Logic Approach to Placement Problem, Proceedings of the 29th ACM/IEEE Design Automation Conference, pp 153-158, 1992

[16] C. Sechen, A. Sangiovanni-Vincentelli: Timber Wolf 3.2: A New Standard Cell Placement and Global Routing Package, Proceedings of the 23rd ACM/IEEE Design Automation Conference, pp 432-439, 1986

[17] A. Sherman: VLSI Placement and Routing: The PI-Project, Springer-Verlag, Berlin, 1989

[18] G. Sigl, J. Kleinhans, F. Johannes: Sea-of-Gates Placement by Simultaneous Quadratic Programming Combined with Improved Partitioning, Proceedings of VLSI '89, München, pp 445-545, 1989

[19] W. Wolf: Modern VLSI-Design: A Systems Approach, Prentice Hall, 1989