Results in Comparing Innovative Placement Heuristics

Reinhard Rauscher, Dieter Klawan
University of Hamburg
Dept. of Computer Science
Hamburg, Germany
e-mail: rauscher@informatik.uni-hamburg.de

Abstract

This paper presents new results achieved with ESPRO, a software system for generating 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 evaluation of new P&R heuristics in algorithm development and universitary backgrounds.

1. Introduction

The ESPRO (Evaluating System for Placement and ROuting) system for VLSI layout heuristics (described in more details in [5]) was created to provide the possibility to test new algorithmic approaches with artificial or real world benchmarks and to compare them with standard placement & routing heuristics on the same data.

In addition the system allows easy implementation and even the integration of existing programs. With this properties, ESPRO can bring together scientists from different disciplines and speed up the integration of new ideas into professional software tools. It consists of several stand-alone programs exchanging data via ASCII files.

In this paper, first we will give a short system overview. In the following we will report the latest system extensions as well as recent results not included in the preceding paper [5].

2. System overview

The goal of our approach was to keep the problem environment for the programs as realistic as possible while getting rid of unnecessary details, e.g. fabrication depending parameters. Thus the information about the layout components is purely geometrical. Placement and routing takes place on a rectangular grid of nonspecified size. The wiring segments are placed within two wiring layers.

A important feature of the system is the possibility to integrate any objective function into the system heuristics. In this way the common concentration upon chip area minimization can be expanded to include e.g. maximal net length, the amount of vias, or critical regions [8]. Also individual preferences or weightings of multiple objectives can be realized.

Another advantage of the open system structure is the option to generate any kind of statistical information required, e.g. channel densities or number of wire segments. Using commercial tools often it is impossible to get these kind of data.

System components can be written in any programming language and run on any hardware platform. The only requirement is to communicate with the other programs through ESPRO in- and output files.

In the current state of progress the system contains only tools for standard-cell layout. The overview of the system (and the program communication flow) is shown in figure 1. For each design step new programs can be integrated.

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

3. Existing programs

Up to now quite a couple of programs have been implemented. All programs are written in C and are running under MS-DOS on Intel-PC's. Using this hardware platform they are able to handle layouts with 10,000 cells within acceptable time. For storage intensive tasks there is the possibility to run the programs under UNIX.

According to figure 1 the programs are classified as follows:

The existing converting routine is a tool that converts net list files from the SOLO 1400 format [2] into the ESPRO connection lists format.

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

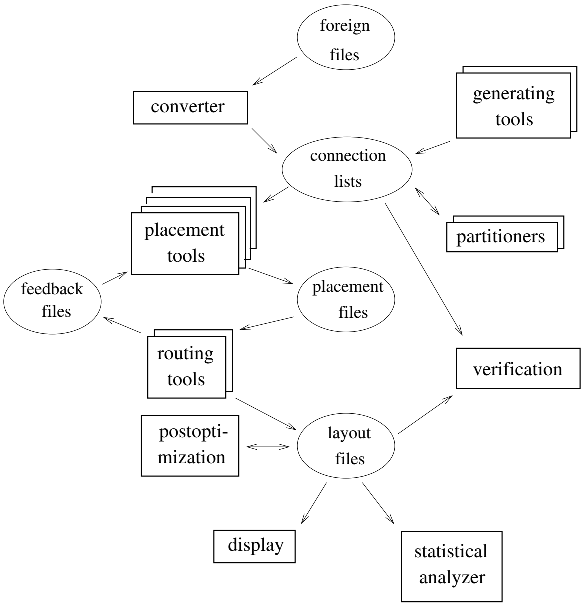

Figure 1. System overview of ESPRO

instances. On the other hand each detail of the circuit may be specified by the user. The second synthesis tool requires only a few key parameters such as the number of components or the maximum component sizes and generates the connection lists randomly based on these values.

Two partitioning heuristics have been implemented (see [3]).

In order to generate initial placements for local-search heuristics, ESPRO contains a random constructive placement routine. Further, a couple of placement tools is available, including recent approaches like genetic algorithms [4] or fuzzy logic heuristics [4] as well as standard tools like the well-known 'Timber Wolf' package [13]. Other placement tools are based on the 'Great Deluge' algorithm by Dueck [1] or the Min-Cut-heuristic, first published in [14].

One new routing approach (DKRouter) has been implemented that combines global and local routing into one step (see [11]). As standard algorithms the 'Timber Wolf' [13] global router, the 'Lee' [10] global routing algorithm and a greedy channel routing procedure [15] for local routing have been implemented (see [7]).

The basic idea of the two implemented post optimization procedures [8] is to substitute a connection between two points of a net by another one (see [5] for more details).

In the current state of the system the converting routine mentioned above can also be used to produce statistical output concerning the generated connection lists. In addition a program is available which determines for a given layout several statistical informations.

The existing display routine provides a Graphical User Interface. It supports zooming into a given layout, highlighting of selected nets, and printing of scaled views. A detailed description can be found in [5].

For debugging of new components as well as for correctness proving purposes an additional verification tool has been implemented [6].

4. Results

We have used the following heuristics for placement:

1. the 'Great Deluge' (GD) heuristic,
2. the Timber Wolf heuristic (TW),
3. a genetic heuristic (GEN),
4. the Fuzzy based heuristic (FL),
5. the hierarchical ordering heuristic (HALO).

For routing we compared the new routing heuristic (DK) and a standard heuristic based on the Lee-Algorithm for global routing and a Greedy-procedure for channel routing.

The GEN-placer is based on the approach described in [9]. The underlying selection procedure is the Monte Carlo method, for recombination the PMX-procedure [9] is used, and reproduction is done by copying. The advantages/disadvantages of the GEN-placer 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.

Thus, we were 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.

The first comparison concerns the CPU time of the placers under consideration. Each heuristic is executed on nine small artificial and real problem instances. In figure 2 the chip code indicates the number of cells and whether the chip is artificial (a) or part of a real world chip (r). A hyphen in the table indicates that the concerned problem was too large for the program. Comparing the CPU time values listed in figure 2, we conclude that the Great Deluge placer is fastest, the Timber Wolf placer takes much more time, whereas the genetic algorithm is in between. The Fuzzy logic approach as well as the HALO heuristic seem to consume at least the time used for Timber Wolf.

In order to evaluate the quality of the generated placements routing is executed on each instance with two alternative strategies: One uses the new DKRouter and the other employs the Lee algorithm for global routing and a Greedy

chip	GD	TW	GEN	FL	HALO
a50	22	62	282	29	28
a100	54	2270	500	295	232
a200	141	913	423	2978	3579
a500	113	5732	1427	56950	47481
a1000	445	22688	2036	-	-
r45	8.4	40	217	12	11
r246	21	754	752	605	747
r276	29	1046	763	1447	3300
r690	57	3101	1921	25987	19773

Figure 2. CPU time for placement heuristics (in sec.)

procedure for channel routing. In this way we could compare all combinations of placers and routers with respect to different optimization goals as e.g. minimizing the chip area, minimizing the maximal net length, and so on.

For reducing the results to few fundamental facts we chose the following procedure: First, a discussion and comparison of the placement heuristics named GD, TW, and GEN is performed with respect to the dependency of the underlying routing procedure.

In a second step a comparison of all innovative approaches is presented with respect to the resulting chip area and the average net length.

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

Figure 3. Results (area in mm^2, time in sec) for DKRouter

Figures 3 and 4 show the first results of these experiments: In each case the total chip area of the layout is measured in square mm and the CPU time (in sec.) is recorded that the corresponding router needed departing from the placements generated by GD, TW, and GEN respectively.

Some of the designs, however, could not be routed by the employed heuristic with the initially chosen set of parameters. These instances are marked by [1] in the figures.

[1]Results with necessary changes of the system parameters. Amount of feed-throughs has been increased.

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

Figure 4. Results (area in mm^2, time in sec) for standard algorithms LEE and GREEDY

Indeed, the placers delivered quite different placements (concerning the number of of standard cell rows, the number of necessary feed-throughs, etc.) the router apparently had to be adjusted to. Since we used neither uniform nor optimally taylored parameter settings, one should be cautious in judging the quality of the heuristics from these experiments.

Comparing figure 3 with figure 4 it becomes clear that routing with Lee and Greedy is faster than with DKRouter in most of the instances, but at the expense of a somewhat larger resulting chip area. There is certainly no clear winner neither among the three placement heuristics nor among the routing heuristics.

Next, we analyzed the innovative placement heuristics (FL and HALO) with respect to the required area and the average net length and compared them with GD and GEN. Figures 5, 6 and 7 illustrate the comparison of these placement heuristics.

chip	GD	GEN	FL	HL
a50	1.70	1.53	1.55	1.72
a100	3.28	3.13	2.96	3.20
a200	7.04	6.65	6.48	6.89
a500	26.80	25.69	23.72	25.88
r45	2.82	2.82	2.82	2.82
r246	3.80	4.30	3.21	4.40
r276	4.83	5.57	3.80	6.63
r690	27.37	27.40	23.54	28.82

Figure 5. Results comparing areas for GD, GEN, FL, and HALO (area in mm^2) based on DKRouter

Figure 5 refers to the resulting chip area. Obviously, the FL-placer seems to be best suited, but also it requires a huge amount of calculation time (see figure 2). The HALO-placer doesn't seem to be a favorite because of the huge amount of calculation time while achieving average results.

chip	GD	GEN	FL	HALO
a50	2.28	1.93	1.85	2.36
a100	3.95	3.59	3.19	4.00
a200	7.80	7.35	7.01	8.09
a500	16.65	16.33	15.26	17.99
r45	0.70	0.81	0.79	0.99
r246	0.97	1.35	0.46	1.33
r276	1.45	1.81	0.65	2.35
r690	4.10	4.38	1.37	4.71

Figure 6. Results comparing the values for average net length for GD, GEN, FL, and HALO based on DKRouter

chip	GD	GEN	FL	HALO
a50	4.82	5.30	4.60	5.60
a100	9.06	8.42	7.35	8.77
a200	14.89	15.59	12.87	14.95
a500	39.45	36.88	34.52	37.20
r45	3.33	4.75	3.24	3.63
r246	8.26	8.01	3.28	7.28
r276	7.27	8.53	3.71	7.76
r690	23.82	22.60	10.44	26.32

Figure 7. Results comparing the values for maximal net length for GD, GEN, FL, and HALO based on DKRouter

Figure 6 shows the comparison of the different placement heuristics with respect to the average net length. A remarkable result has been, that the GEN-placer seems to be suited better for artificial examples than the GD-placer. On the other hand this seems to be false for real world examples, Again, the FL-placer is best suited. Partially, the reduction of average net length achieves a factor of two or more. Concerning this criterion the HALO-placer seems to be a quite bad approach also.

Figure 7 shows the results concerning a highly important criterion (e.g. for determining the maximal clock frequency), namely the maximal net length. Once more the FL-placer seems to be best suited, while the HALO-placer again seems to be a quite bad approach. According to these results the HALO-placer is not discussed further in this paper.

Summarized, the FL-placer seems to give best results in most of the cases, but it requires a much calculation time. So, the best candidates for large designs could be seen in GD-placer and GEN-placer. Further results in comparing the different heuristics can be found in [4].

5. Conclusions

The evaluations presented in this paper are another evidence of the successful system concept of ESPRO. It allows easy comparison of layout procedures producing instructive results for further heuristic development.

The results itself are supporting our thesis that there is no perfect placement and routing heuristic that is suitable best for all problems. Instead we recommend a proceeding taking in account the problem size, the known characteristics, and the computing time available. According to these facts a suitable combination of heuristics and parameter settings should be chosen.

References

[1] 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

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

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

[4] H. Jenke: Implementierung und Evaluierung von innovativen Heuristiken zum Placement & Routing, diploma thesis, University of Hamburg, 1997

[5] R. Rauscher, D. Klawan, H.-J. Bandelt: Results given by a New System for Placement & Routing, in: M.E. Kavanaugh (ed.): Proceedings of EUROMICRO '96, IEEE Computer Society Press, Los Alamitos, pp. 259-266, 1996

[6] O. Kahlbaum: Ein Tool zur Verifikation von Standardzellayouts, diploma thesis, University of Hamburg, 1997

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

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

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

[10] C. Lee: An Algorithm for Path Connection and its Applications, IRE Trans. Electronic Computers, EC10, 1961, 346-365

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

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

[13] 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

[14] Y. Yang, C. Kyung: HALO - An Efficient Global Placement Strategy for Standard Cells, IEEE Transact. on CAD, S. 1024-1031, 1992

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