TINA*: Analog Placement Using Enumerative Techniques Capable of Optimizing Both Area and Net Length

Tobias H. Abthoff† and Frank M. Johannes
Institute of Electronic Design Automation
Technical University Munich

Abstract

In this paper a new enumerative algorithm called TINA is introduced that generates slicing placements optimal in both area and overall net length. It is designed to automate the task of placement for analog circuits given a set of modules with multiple realizations, a corresponding net list, and neighborhood relations. TINA reduces the overall net length to nearly one fifth compared to a net length unaware enumeration algorithm while using only 1.6% more area and twice the CPU-time. TINA is based on enhanced shape functions that are capable of carrying net length information. TINA can be used either as a fully automatic analog placement tool, or as an interactive tool for creating extremely dense placements from a loose placement provided by the designer to establish neighborhood relations. In the first case, another tool, e.g. PLACEBO, is used to compute neighborhood relations. TINA is able to handle all major analog constraints like clustering, framing, fixed orientations, fixed realizations, and symmetries.

1 Introduction and Motivation

Analog placement is a much more complex task than digital placement mainly because a large number of constraints have to be considered. The task is further complicated due to the fact that for a device usually many layout realizations are available, i.e. a large transistor has several alternative layout realizations depending on how many times the transistor is folded. For analog layout design only very few automatic tools were available in the past. This has changed with the upcoming of powerful module generators. In analog design it is common to have a huge number of different modules, e.g. a very big number of transistors with different W/L ratios and capacitors with different shapes. These modules are very likely to contain errors if drawn manually by designers. To cope with this problem many companies have developed module generators, programs that are able to generate modules like transistors from given specifications. These module generators guarantee error free module realizations and are usually able to create many different realizations (up to 100) for a given specification. Today, most of the modules in a given circuit are created by module generators.

Human beings perform surprisingly well in packing modules with fixed shapes but unfortunately are not very good in choosing a shape out of a set of given shapes to achieve an optimal packing. Thus, there is a great need for placement tools to exploit the design space which is offered by the use of module generators. Analog circuits are rarely bigger than 100 modules. This results on the one side from the ability of a human designer to cope with the design complexity, and on the other side from the fact that there are nearly no analog function blocks consisting of more than 100 modules.

Taking all this into account, a placement tool for today's needs must be able to deal with about 100 modules with an average of 20

* TINA: TINA is no acronym

† This work was partly funded by TEMIC, Germany within the JESSI project.

EURO-DAC '96 with EURO-VHDL '96
0-89791-848-7/96 $4.00 ©1996 IEEE

different realizations. An analog placement tool should not only be able to compute a layout of a whole circuit automatically, but also be able to place modules interactively to assist the designer. The placement tool must be able to handle the relevant analog constraints. A list of these constraints is presented in Section 3.5.

1.1 Review of Previous Work

The placement of analog circuits is in some aspects similar to the traditional floorplanning problem. Both the placement algorithms for analog circuits and floorplanning algorithms have to deal with a fairly small number of elements that differ dramatically in size and may have several realization alternatives. Many researchers have studied various aspects of area minimization of slicing floorplans, which is very similar to our problem. For slicing floorplans, Otten [7] showed for the first time that the minimization problem can be solved efficiently. Stockmeyer [6] presented his well known and widely used algorithm of time complexity O(nd) where n is the number of modules (basic blocks) and d is the depth of the resulting slicing tree. Shi [8] presented an algorithm with a time (and space) complexity of O(n log n) and proved that this is the lower bound for the minimization of slicing floorplans. Zimmermann [9] combined the algorithm of Stockmeyer with a wiring area estimation. More information can be found in [4].

1.2 Target of Our Work

Our algorithm TINA performs a combined area and net length optimization. It does no wiring area estimation, but computes the overall net length of every realization using half perimeters. Wiring area is not taken into account since we have focused on BiCMOS technology where over-the-cell routing is possible.

2 Overview

The placement procedure TINA is based on enumerative techniques. The input to TINA consists of a net list, a constraint database, a shape function for each module to be placed and a set of neighborhood relations between all module pairs.

TINA enumerates all slicing configurations that are compatible with the given neighborhood relations thereby computing for every realization the aspect ratio, area, and overall net length. The net length is measured using the half perimeter model. For every possible slicing configuration, the resulting shape and the overall net length is presented to the designer, who chooses the configuration that fits best to his or her needs. For the chosen configuration, the positions, orientations and realization alternatives are determined for each module.

In the next Section some basic definitions are given including a complete list of geometrical analog constraints that was, to our knowledge, never presented before in such completeness. In Section 4 the new algorithm and its data structures are presented. Section 5 contains results regarding CPU-time, memory consumption and improvement of layout quality.

3 Definitions

3.1 Realization

A realization φ is a geometrical representation of one or more modules. It consists of its bounding box and the set of all terminals. The size of a realization (bounding box) φ is given by $(\bar{x}_\varphi, \bar{y}_\varphi)$. The internal net length of a realization is given by l_φ. If a realization φ represents only one module, l_φ is always zero, if φ represents several modules, l_φ is the length of all connections between these modules modeled as half perimeter. In most cases there are several geometrically different but electrically equivalent realizations for a module, or a set of modules. Two realizations φ_1 and φ_2 are treated differently if they have different dimensions, or if the coordinates or dimensions of their terminals are different.

3.2 Terminal

A terminal is a rectangle positioned inside the area of a realization. A terminal is used to connect a realization with a net. The lower left corner of a terminal τ positioned on a realization φ has the coordinates (x_τ, y_τ) relative to the lower left corner of φ. The dimensions of the terminal τ are $(\bar{x}_\tau, \bar{y}_\tau)$. Please note that terminals are not restricted to realizations of modules. A realization of a set of modules also has terminals that are not identical with the terminals of its modules (this is discussed in Section 4.2).

3.3 Modules

A module is an element of an electrical circuit. In this paper a module μ denotes the set of all realizations $\varphi \in \mu$ of a single circuit element delivered by its module generator.

3.4 Nets

A net ν is a set of terminals $\tau \in \nu$ that have to be connected. A net ν must not contain more than one terminal from one module.

3.5 Geometrical Analog Constraints

The analog constraints presented in this paper have been formulated within the JESSI AC12 project. All JESSI AC12 partners have agreed that these rules are sufficient to express all necessary electrical constraints.

1) Pre-placement:
   Pre-placement of modules is essential. No further common definition was made. Currently pre-placement is done by specifying a certain slicing structure for the modules.
2) Orientation/Realization:
   - A module has a fixed orientation/realization, or one orientation/realization out of a set of allowed orientations/realizations.
   - Two or more modules have the same orientations/realizations.
3) Distance
   - Two modules must be placed as close as possible (clustering).

- Two modules must have a certain minimum/maximum distance.

4) Symmetry
- Two modules of the same type must be symmetrical to each other with a symmetry axis of 0° or 90°.
- In between the two given symmetrical modules a given group of other modules (middle group) may be placed.
- Several symmetrical modules must share a common axis and middle group.

5) Nesting of all previous rules is allowed.

3.6 Neighborhood Relationships

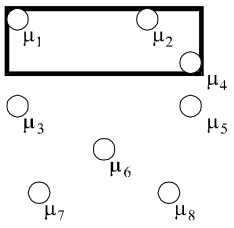

Figure 1 - Global Placement

Neighborhood relations are a set of constraints defining whether the placement of a certain module is forbidden in a certain direction of another module. In Figure 1 neighborhood relations of modules $\mu_1 \ldots \mu_8$ are shown as a global placement, each point corresponding to a module. Module μ_6 is not allowed to be placed on top or to the left of module μ_3. In Figure 2, all allowed slicing configurations and placements for a cluster of module μ_3, μ_6 and μ_7 are shown (each module has only one realization, no rotation allowed).

This neighborhood information is either delivered by another placement algorithm, e.g. PLACEBO [1, 2]., or by the designer. PLACEBO computes global placement information by quadratic optimization. The neighborhood relations have to be in accordance with all specified analog constraints, including the symmetry constraints. Two parts of a symmetry must have identical but mirrored neighborhood relations. If the module pairs μ_3, μ_5 and μ_7, μ_8 are specified to be symmetrical to a common vertical axis in the final placement, the global placement must reflect this. In the global placement shown in Figure 1 the corresponding points are placed symmetrical and therefore comply with such a specification. If a clustering of certain modules is specified, the neighborhood relations of all modules have to be set so that a rectangle can be drawn that covers only the points corresponding to the clustered modules. In Figure 1, the neighborhood relations are compatible with a clustering of the modules μ_1, μ_2, and μ_4 but e.g. not with a clustering of μ_1 and μ_4 because a rectangle covering μ_1 and μ_4 will always cover μ_2.

4 Algorithm

The new enumeration algorithm TINA enumerates all possible placement alternatives for a given set of modules. TINA computes the shape function of the circuit and the overall net length of the corresponding realization of each point of the shape function. The choice of "best" realization is done either by the designer or by applying a simple heuristic to trade off desired aspect ratio, area, and overall net length. Then TINA computes the position, orientation and realization for every module to obtain the chosen realization.

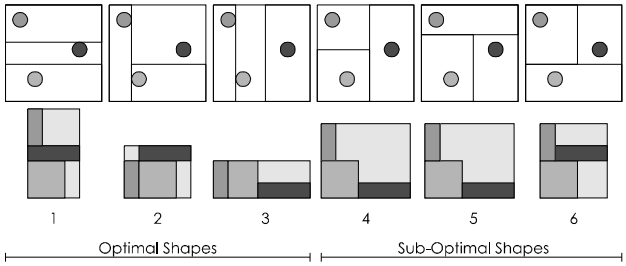

Figure 2 - Realizations Compatible with Global Placement

4.1 Data Structures

The presented algorithm TINA is based on the principles of slicing [7], and shape functions [5, 6] and a slicing enumeration algorithm [3]. Shape functions are an efficient data structure to store several rectangular shapes for a given module or set of modules. For every shape one point is stored. Two modules or set of modules having their realization represented as shape functions can be packed very efficiently. In Figure 3, a shape function for a module with a quadratic and a rectangular realization is drawn. This shape function has three points, one for the quadratic shape, one for the rectangular shape and one for the rotated rectangular shape. All shapes with sub-optimal area do not appear in the shape functions (sub-optimal shapes can be seen in Figure 2). Shape functions and packing of objects represented by shape functions are discussed in detail in [5, 9].

A shape function stores information about different shapes for one object. In order to perform a combined area and overall net length optimization the following additional information is stored for every shape (point of the shape function):

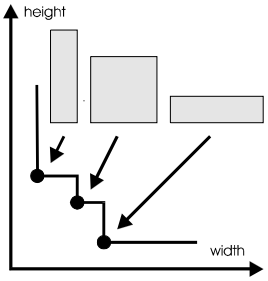

Figure 3 - Shape Function

- The position and bounding box of every terminal τ of φ.
- Overall accumulated net length inside the realization φ. If φ consists of only one module this is zero.

A shape and this additional information is denoted to be a realization as defined in Section 3.1.

In Figure 4, the shape functions for three sets of modules $\mathcal{M}_1$, $\mathcal{M}_2$, and $\mathcal{M} = \mathcal{M}_1 \cup \mathcal{M}_2$ are drawn. The sets $\mathcal{M}_1$ and $\mathcal{M}_2$ consist of only one module each. Underneath the shape functions the terminal positions and shapes are drawn for two different realizations (for $\mathcal{M}_1$ the same realization is drawn twice). Please note, that a single point in the shape functions of one module corresponds to up to

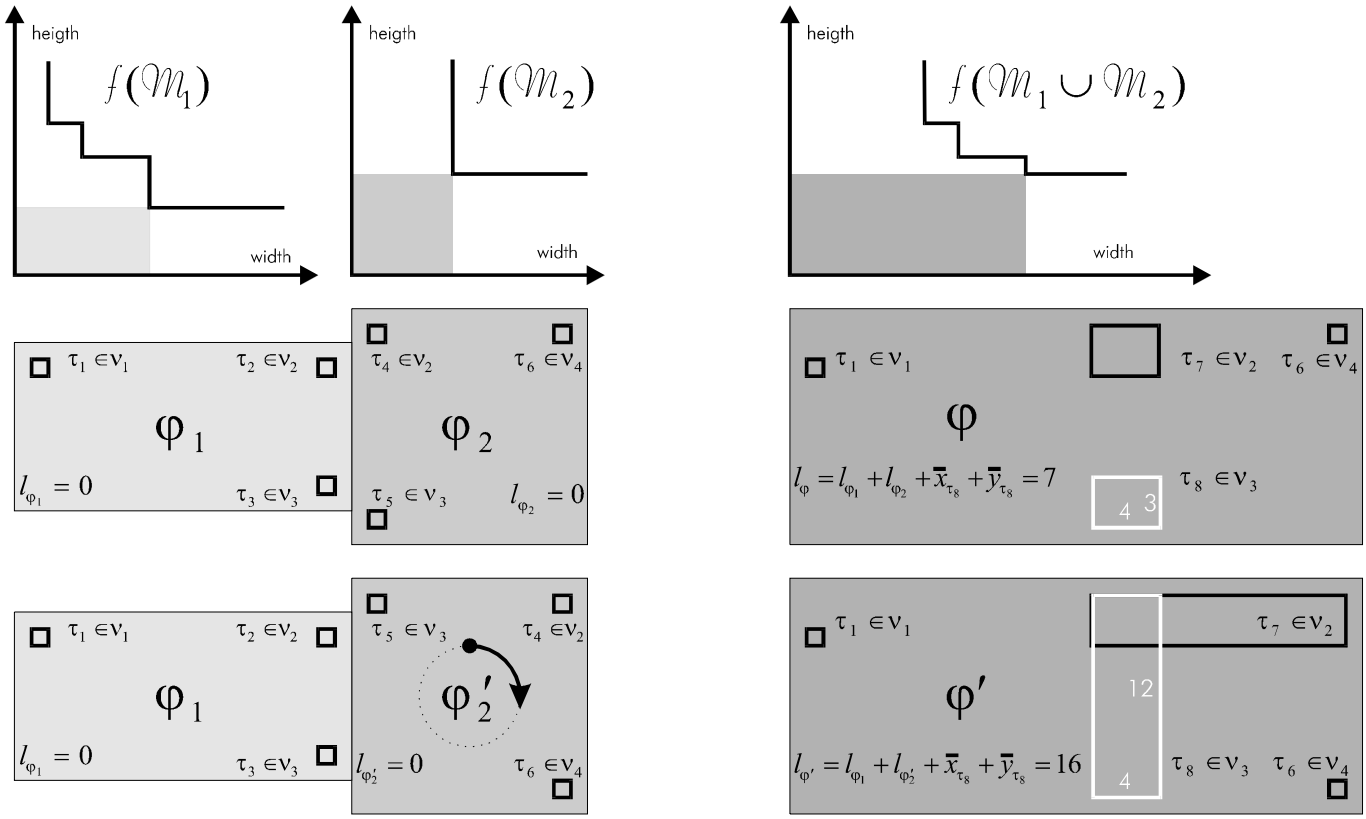

Figure 4 - Adding Two Realizations

eight[‡] different realizations due to different orientations. For sets of modules there may be even more different realizations. One single module has no inside nets, thus the total accumulated net length of all realizations of $\mathcal{M}_1$ and $\mathcal{M}_2$ is zero. $\mathcal{M}$ consists of more than one module. The half perimeter of all nets connecting only modules $\mu \in \mathcal{M}$ is stored as the internal net length of the realizations of $\mathcal{M}$. The construction of realizations of $\mathcal{M}$ is discussed in Section 4.2.

4.2 Basic Algorithm For Adding Two Realizations

Two sets of modules $\mathcal{M}_1$ and $\mathcal{M}_2$ are combined by adding every realization of $\mathcal{M}_1$ with every realization of $\mathcal{M}_2$. For this problem effective algorithms have been proposed, for example in [8], which avoid the computation of realizations that are not area-optimal. We use these algorithms and focus on area optimal shapes (see Figure 2 and Section 4.3). The remaining problem is how to add two realizations and how to store the net length information efficiently. We are attacking this problem by the enhanced definition of terminals: If a set of modules contains two or more modules that have terminals connected to the same net, a new terminal is created using the bounding box of all terminals of the modules that are connected to the net. This new terminal is used for outside connections of the module set.

The algorithm performs the following steps to compute a realization φ of a set of modules $\mathcal{M} = \mathcal{M}_1 \cup \mathcal{M}_2$ from one realization φ_1 of module set $\mathcal{M}_1$ and a realization φ_2 of module set $\mathcal{M}_2$ (see Figure 4):

1. Compute the shape of φ from the shapes of φ_1 and φ_2.
2. Compute the overall accumulated net length $l_\varphi = l_{\varphi_1} + l_{\varphi_2}$.
3. Determine all terminals of module $\mathcal{M}_1$ that are connected to nets not connected to $\mathcal{M}_2$ and vice versa. Create these terminals in φ. In Figure 4, terminal τ_1 of $\mathcal{M}_1$ is connected to ν_1. Net ν_1 is not connected to any terminal or sub-net of $\mathcal{M}_2$, thus terminal τ_1 of φ is created the with the shape and position of terminal τ_1 of φ_1. The same is done for terminal τ_6 of φ_2, but the position of τ_6 of φ is offset by the width of φ_1.
4. Determine all terminals of $\mathcal{M}_1$ and $\mathcal{M}_2$ that are connected to the same net and merge them into new terminals of φ. In Figure 4, terminal τ_2 of φ_1 is connected to the same net ν_2 as terminal τ_4 of φ_2. These two terminals are merged into a

[‡] There might be less than eight realizations if the object is self-symmetric, e.g. a module with a single quadratic terminal located in its center has one realization per shape function point.

new terminal τ_7 of φ with the shape and position of the smallest covering rectangle of the two terminals τ_2 of φ_1 and τ_4 of φ_2. The same is done with terminal τ_3 of φ_1 and terminal τ_5 of φ_2, they are merged into τ_8 of φ, since they are both connected to net ν_3.

5. Delete all terminals that cover a whole net ν and add their half perimeter to the overall accumulated net length l_φ of φ.
   It is assumed in the example that net ν_3 consists of only the two terminals τ_3 and τ_5 that have been merged into terminal τ_8 of φ. Since no other terminals are connected to net ν_3 terminal τ_8 of φ is not stored but the half perimeter of its shape is added to the total accumulated net length of φ $\left(l_\varphi := l_\varphi + \bar{x}_{\tau_8} + \bar{y}_{\tau_8}\right)$.

In Figure 4 there also shows an addition of φ_1 with another realization φ_2' of $\mathcal{M}_2$. The realization φ_2' corresponds to φ_2 except that it is rotated by 90°. Although the shapes of φ_2' and φ_2 are identical, the position and bounding boxes of their terminals are quite different. The internal net length is also different. Since no decision can be made whether φ_2 or φ_2' will result in better placements both have to be stored.

The memory needed for storing net length optimization is minimal since completed nets do not require extra memory. A good placement will have most of the nets completed on a low level; thus the total memory consumption is low.

4.3 Identifying Sub-Optimal Realizations

When combining area and net length optimization, it is not possible to decide whether a certain realization is sub-optimal compared to other realizations. In the case of area-only optimization a realization φ is sub-optimal compared to another realization φ' if the shape of φ is bigger or equal in both dimensions. Examples are shown in Figure 2.

If both area and net length optimization are performed, the previous sub-optimal realization may have a better internal net length or distribution of its terminals so that it cannot be decided which realization will lead to a better placement. For simplicity and complexity we decided to take only area-optimal realizations into account. Sub-optimal realizations in terms of area will never lead to placements with optimal area usage. The goal of work however is to deliver placements with very good area usage and very good net length.

4.4 Handling Analog Constraints

The following analog constraints can be currently handled by TINA:

Fixed orientations/implementation alternative. Every orientation/implementation combination of a module is treated by TINA as different realization of the module. Thus, deleting certain module realizations before the enumeration process starts limits the module to certain orientations or implementation alternatives.

Clustering / Framing. A cluster is treated by TINA as one object. All realization alternatives of the cluster are computed before the rest of the circuit is enumerated. This concept is very flexible and allows nested clusters. If framing is desired the shape of every realization of the cluster is enlarged by the frame size in every direction before the enumeration process starts.

Symmetries. A symmetry consists of three parts, a left-hand ($\mathcal{L}$), a right-hand ($\mathcal{R}$) side and a center ($\mathcal{C}$) part, as shown in Figure 5. The symmetry is processed in 5 steps, which is shown for a symmetry in x-direction, for the y-direction this is done accordingly. If $\mathcal{C}$ is empty, the steps one and two are omitted.

1. All realizations of the center part $\mathcal{C}$ are computed (the middle part is processed like a cluster).
2. A new virtual module $\mu_\mathcal{C}$ is created. The shape of every realization of $\mu_\mathcal{C}$ corresponds to a realization of the center part $\mathcal{C}$ except that the shape is divided by two in x-direction (the right half is discarded). The shape and positions of the sub-nets and terminals of $\mu_\mathcal{C}$ remain unchanged to $\mathcal{C}$, so that some terminals (or sub-nets) of $\mu_\mathcal{C}$ may be positioned outside the shape of $\mu_\mathcal{C}$. In Figure 5 this is shown in the middle drawing: the gray area shows the new shape of $\mu_\mathcal{C}$ and the thick line the area where terminals of $\mu_\mathcal{C}$ may reside.
3. The module $\mu_\mathcal{C}$ is added to $\mathcal{L}$ ($\mathcal{L} := \mathcal{L} \cup \{\mu_\mathcal{C}\}$) and all realizations of $\mathcal{L}$ are computed, with the constraint that $\mu_\mathcal{C}$ must always be the rightmost module. A resulting realization of $\mathcal{L}$ looks like the middle drawing in Figure 5, $\mu_\mathcal{C}$ is at the right side, and some terminals of $\mu_\mathcal{C}$ are located outside the shape of $\mathcal{L}$.
4. The same is done for $\mathcal{R}$ accordingly; see the right picture of Figure 5.
5. Now $\mathcal{L}$ and $\mathcal{R}$ are combined by adding every shape of $\mathcal{L}$ with the *corresponding* shape of $\mathcal{R}$. After this all terminals are positioned inside the shapes of the symmetries realizations.

5 Results

5.1 Layout Quality

A comparison of the achieved layout quality is very difficult. First, our algorithm, and all other enumerative algorithms, enumerate the whole design space that is compatible to the global placement information so that the found solution will always be optimal. Sec-

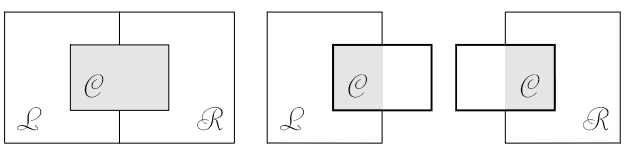

Figure 5 - Symmetry Handling

ond, it is not sufficient to compare a single placement determined by TINA with a single placement obtained by another algorithm, instead the quality of all presented solutions should be compared, since the preferences of the designer are not previously known.

To estimate the improvement of layout quality, a comparison to Stockmeyers algorithm was done, which is well known in literature (ST). ST delivers 42 different shapes for an industrial OP-amp example. The net length aware algorithm (TINA) delivers the same 42 shapes plus an average of 25 different realizations per shape all differing in their overall net length. TINA will always implement the realization with the shortest overall net length for a given shape. ST, however, has no information about net length, so we assume that it implements a realization for a given shape with random net length. The net length of this realization is assumed to be the average net length of all possible realizations of that shape. Comparing the average net length of all 42 shapes implemented by TINA and ST, the average net length of the realizations computed by TINA is only half of the average net length of ST. If a tolerance of the desired aspect ratio of 10% is allowed, TINA can choose the best realization of several shapes within this tolerance, leading to a net length of only 22% compared with ST. On the other hand ST will always choose the shape with the least area inside the given aspect ratio range, so that the average area needed by TINA is 1.6% higher.

One of the most important features of TINA is the ability to give the designer the chance to do a trade-off between area and net length. While this trade-off is not that important in digital circuit design, it is crucial in the design of analog circuits. The designer has the ability to choose small area for non-critical parts of his circuit and switch to small net length in critical high-performance parts of his design. It is also possible to assign weights to nets so that critical nets can be handled with special care.

5.2 Run Times

In Table 1, the CPU times are shown for industrial circuits between 4 and 13 modules. The memory consumption was about 10 MB for the 13 modules. It can be seen that the CPU consumption is quite low, however the complexity is exponential. TINA is able to enumerate circuits with up to 20 modules completely, but is configured to split circuits with more than 10 modules in groups that are enumerated separately. A set of modules consisting of 10 or more modules can be realized with many different shapes, so that several of those module groups can be combined with nearly no slack. Our experiments have shown nearly no loss of layout quality compared to enumeration of groups with about 10 modules to a full enumeration. Thus the complexity of the whole program is nearly linear, so that a circuit with 100 modules can be placed in less than 20s.

Number of Modules	CPU-Time without Net Length	CPU-Time with Net Length
13	2.14 s	8.08 s
12	1.09 s	4.19 s
11	0.626 s	2.32 s
10	0.331 s	1.21 s
9	0.193 s	0.708 s
8	0.0818 s	0.271 s
7	0.0578 s	0.174 s
6	0.0314 s	0.0714 s
5	0.0185 s	0.0325 s
4	0.0136 s	0.0202 s

Table 1 - CPU Times on an Intel PPro 150 Processor

6 Conclusion

We have presented a new approach called TINA to enumerate placement alternatives efficiently. Our approach performs a global area and overall net length optimization in reasonable time for interactive work. Compared to a net length unaware enumeration algorithm TINA reduces the overall net length to nearly one fifth if an aspect ratio tolerance of 10% is allowed. This is done at the cost of 1.6% area and twice the CPU time. Furthermore, TINA is able to a trade overall net length against area and to handle the major analog constraints including nested symmetries. TINA can be embedded in an automatic placement tool or can be used as an interactive and incremental tool to assist a designer at his daily work.

7 Acknowledgment

The authors want to thank Telefunken microelectonics GmbH (TEMIC) Ulm, Germany and all other JESSI AC12 partners for their cooperation and valuable discussions.

8 References

[1] Abthoff, T.H.; Johannes, F.M.: "Analogue Placement using Guided Enumeration“, International Journal of Circuit Theory and Applications, Vol. 23, 453-473 (1995)

[2] Abthoff, T.H.; Johannes, F.M.: "PLACEBO: Analog Placement with Efficient Symmetry Support“, International Journal of Electronics and Communications, Vol. 49, No. 2, 55-63 (1995)

[3] van Ginneken, L.P.P.P.; Otten, R.H.J.M.: "Optimal Slicing of Plane Point Placements", Proc. European Conference on Design Automation, pp. 322-326, 1992

[4] Lengauer, T.: "Combinatorial Algorithms for Integrated Circuit Layout", New York, Wiley, 1990

[5] Otten, R.H.J.M.: "Efficient Floorplan Optimization“, Proc. IEEE International Conference on Computers and Design, 1983, pp. 499-501

[6] Stockmeyer, L.: "Optimal Orientations of Cells in Slicing Floorplan Designs", Information and Control, Vol. 57, pp. 322-326, 1990

[7] Szepieniec, A.; Otten, R.H.J.M.: "The Genealogical Approach to the Layout Problem", Proc. ACM/IEEE 17th Design Automation Conference, pp. 164-170, 1980

[8] Shi, W.: "An Optimal Algorithm for Area Minimization of Slicing Floorplans“, Proc. IEEE/ACM International Conference on Computer Aided Design, 1995, pp. 480-484

[9] Zimmermann, G.: "A New Area and Shape Function Estimation Technique for VLSI Layouts“, Proc. ACM/IEEE 25th Design Automation Conference, 1988, pp. 60-65