A Constraint-Driven Methodology for Placement of Analog and Mixed-Signal Integrated Circuits

Ammar Nassaj, Jens Lienig
Dresden University of Technology
Institute of Electromechanical and Electronic Design
Dresden, Germany
nassaj@ifte.de, jens@ieee.org

Göran Jerke
Robert Bosch GmbH, AE/EIM
Reutlingen, Germany
goeran.jerke@ieee.org

Abstract— **Layout design of analog and mixed-signal circuits is often a manual and time-consuming, trial-and-error task. Stringent constraints that must be considered simultaneously are a major reason why layout design is often not automated. To overcome this bottleneck in the design process, we present a new *constraint-driven* design methodology. We have verified our methodology by applying it to the placement of both analog and mixed-signal circuits. Our approach allows us to verify whether a solution that satisfies all constraints exists prior to the time consuming optimization process. If a solution exists, an initial placement with a maximum constraint robustness is constructed. Next, the initial placement is optimized. Unlike the optimization engines known so far, our implementation is driven not only by the placement objectives, but also by the adaptively weighted constraints. This allows efficient searching in the solution space of the constraints.**

I. Introduction

The increasing complexity of integrated circuits (ICs) often demands integrating both analog and digital functions on a single chip. The design and synthesis of the analog parts represent a bottleneck in the design flow. This is due to the stringent analog design requirements like minimization of crossovers and isolation of sensitive nets. Moreover, analog circuits are more sensitive to the fluctuations of the manufacturing process. Thus, various constraints such as device matching, symmetry, parasitics, and thermal effects must be taken into account [1]. The combined constraints define a solution space (Fig. 1), where each candidate solution satisfies all constraints.

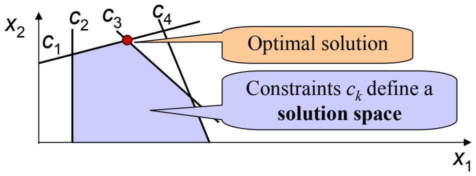


Figure 1. Graphical representation of a solution space defined by a set of four linear constraints c_k on two variables x_1 and x_2 . The optimal solution of the respective layout problem results from minimizing (maximizing) an objective function and thus, is located on the boundary of the solution space.

The analog layout designer is increasingly faced with new application-specific requirements represented by numerous new constraints. Hence, a new methodology is needed that takes these constraints into account such that an automatic layout approach is also feasible for analog and mixed-signal circuits.

A. Related Work

One of the first successful placement and routing frameworks for analog circuits is introduced in [2]. Symmetry as well as device abutment and merging are handled. Techniques for performance-driven layout of analog integrated circuits are introduced in [3] and [4]. These approaches are only suitable for small circuits such as filters and operational amplifiers where general performance requirements such as gain and bandwidth are given.

Balasa *et al.* [5] directly take into account the presence of an arbitrary number of symmetry groups during the exploration of the solution space. Other constraints are not discussed. In [6], an approach to place analog devices taking into consideration symmetry and other placement constraints is described. Here, the constraints are simultaneously satisfied in a candidate solution. However, finding such a candidate based only on a heuristic method is not always guaranteed if many correlated constraints exist.

Current commercial analog layout tools often fail to state whether a solution that fulfills all constraints exists at all, or whether there are some contradictory constraints. Furthermore, even with non-contradictory constraints, conventional optimization methods are severely hampered due to the numerous and correlated constraints that have to be taken into account simultaneously.

B. Our Contribution

We developed a constraint-driven methodology to address the above mentioned problems for the first time. As a result, the *automatic* layout generation of analog and mixed-signal circuits with multiple and correlated constraints becomes feasible. This methodology is verified by applying it to placement of analog and mixed-signal circuits.

We present a straightforward constraint verification step. It allows verifying whether a layout solution which satisfies all constraints exists at all prior to layout generation (i.e., contradictory constraints that prevent any valid solution do not exist). This is done using linear programming (LP). If feasible, LP constructs an initial placement with maximum constraint robustness (i.e., a small perturbation of the placement will most probably not violate any constraint).

Constraints are not only used in the optimization process in order to verify the validity of a layout solution. Additionally, they are actively steering this process. This is done by applying a constraint-driven simulated annealing (SA) algorithm on the initial placement. SA was chosen because it can efficiently handle the different objectives and numerous constraints. The cost function of our SA is evaluated not only based on the objectives but also on the adaptively

The present work was partially supported by the BMBF project LEONIDAS+ (conveyor sign: 01M3074). We thank Jan Freuer, Dr. Jürgen Scheible and Ralf-Eckhard Stephan from Robert Bosch GmbH for the valuable discussions.

This paper has been published in Proc. of the
14th IEEE Int. Conf. on Electronics, Circuits and Systems (ICECS)
Malta, pp. 770-773, August 2008.

978-1-4244-2182-4/08/$25.00 ©2008 IEEE.

weighted constraints. This results in efficient searching the solution space of the constraints.

II. Terms And Definitions

We differentiate between the so-called *geometrical* and the *no-overlap* constraints which are implemented and considered differently:

- *No-overlap constraints:* Constraints that define minimal distances in order to prevent module overlaps.
- *Geometrical constraints:* Constraints that result from the mapping process of "conventional" constraints such as electrical and thermal constraints into geometrical constraints.

Generally, each constraint can have a range of permitted values. The range represents an allowed tolerance and is defined between a minimal and a maximal value, *min* and *max*, respectively (e.g., defining a minimal allowed distance between a module and a heat source). In this case, *max* is defined, for example, by the maximal allowed dimensions of the layout. Moreover, we define a desirable (optimal) value *opt* within the range [*min*, *max*]. *Opt* can be assigned by the user and can take any value within the range [*min*, *max*]. If not defined, *opt* is the arithmetic mean and is calculated internally by the design algorithms. If no tolerance is allowed, then *min* = *max* = *opt*.

As a module may have many constraints, it is possible to estimate a position that fulfills simultaneously all constraints as good as possible with regard to *opt*. This is true for all modules. Hence,

- *Optimal placement with respect to all constraints:* All modules are placed so that they satisfy all related geometrical constraints as good as possible with regard to their respective *opt* value. Hence, we obtain a layout with *maximum constraint robustness.*

III. Constraint-Driven Placement Methodology

Our approach is divided into two steps. First, the analytical method linear programming (LP) is used to verify whether the placement under the given constraints is feasible. If feasible, LP constructs an initial placement with maximum constraint robustness. In the second step, a modified simulated annealing (SA) algorithm is applied based on the initial placement. SA is driven not only by the objectives, but also by the adaptively weighted constraints (AWC). Fig. 2 depicts the main steps of our approach.

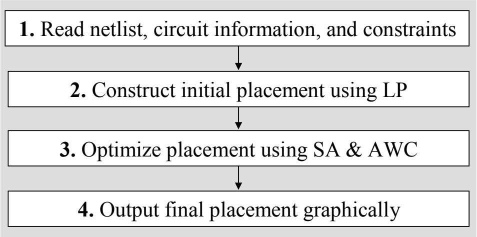

1. Read netlist, circuit information, and constraints
2. Construct initial placement using LP
3. Optimize placement using SA & AWC
4. Output final placement graphically

Figure 2. Outline of our constraint-driven placement approach.

A. Initial Placement Using Linear Programming

Using linear programming (LP), the placement is represented as a linear system of equations. LP is used in several works to minimize the area and/or the wirelength ([7]-[9]). Here, we do not aim to minimize the area or wirelength. Our aims are:

- Finding whether a solution that satisfies all geometrical constraints exists at all.
- If solution(s) exist(s), then we construct an initial placement with the maximum constraint robustness.

These goals are of special importance for the analog and mixed-signal layout design where multiple constraints are to be taken into consideration simultaneously. If no solution exists, the user is notified to change the initial setup such as the maximum dimensions of the layout. Please note that in this step the no-overlap constraints are not considered due to complexity issues.

Our LP formulation consists of the cost function, equations to place the circuit modules within the allowed dimensions of the layout, equations that describe the dimensions of the circuit modules, and equations that describe the geometrical constraints. The origin of each module m_i is defined by the coordinates (x_i, y_i) of its lower left corner ($i \in n$, where n is the number of modules). The width and height of each module are w_i and h_i, respectively. W and H describe the maximal allowed width and height of the layout, respectively.

In the following, we present the LP formulation as used by our approach. The objective is to minimize all P_k variables, where P_k is a measurement of the robustness of the constraint $k; \forall k \in K$, where K is the number of constraints.

$$\text{Minimize } \sum P_k; \quad \forall k \in K; \tag{1}$$

$$x_i \geq 0; \quad y_i \geq 0; \tag{2}$$

$$x_i + w_i \leq W; \quad y_i + h_i \leq H; \tag{3}$$

$$w_i = width(m_i); \quad h_i = height(m_i); \tag{4}$$

for each constraint c_k, $\forall k \in K$

$$min_{ij}(c_k) = min\_allowed\_dist(m_i, m_j); \forall i \neq j; i, j \in n \tag{5}$$

$$max_{ij}(c_k) = max\_allowed\_dist(m_i, m_j); \forall i \neq j; i, j \in n \tag{6}$$

$$opt_{ij}(c_k) = opt\_dist(m_i, m_j); \forall i \neq j; i, j \in n \tag{7}$$

$$\left|x_i - x_j\right| + \left|y_i - y_j\right| + \frac{(w_i - w_j) + (h_i - h_j)}{2} \geq \min_{ij}(c_k); \tag{8}$$

$$\left|x_i - x_j\right| + \left|y_i - y_j\right| + \frac{(w_i - w_j) + (h_i - h_j)}{2} \leq \max_{ij}(c_k); \tag{9}$$

$$\left|x_i - x_j\right| + \left|y_i - y_j\right| + \frac{(w_i - w_j) + (h_i - h_j)}{2} - opt_{ij}(c_k) = P_k; \tag{10}$$

Equations (2) and (3) place the modules within the allowed dimensions of the layout. The widths and heights of the modules are defined by (4). Equations (5)-(7) define the minimal, maximal and most desired relative position of the modules m_i and m_j with respect to constraint c_k. Equations (8) and (9) guarantee that m_i and m_j are placed within [$min_{ij}(c_k)$, $max_{ij}(c_k)$]. The distance is given as Manhattan distance between two points. Here, the center-to-center distance between the modules m_i and m_j is used. Equation (10) defines the P_k variables illustrated above. Minimizing these variables results in a placement with a maximum constraint robustness (excluding no-overlap constraints).

B. Placement Optimization Using a Constraint-Driven Simulated Annealing

In a standard implementation of SA algorithm, a cost function which consists only of the objectives is minimized. Other implementations include penalty terms for violated constraints. These implementations are not suitable if many correlated constraints exist. In this case, many perturbations will be invalid. To overcome this problem, we developed a modified SA algorithm with the following properties:

- SA starts with the initial placement which has maximum constraint robustness but it is not yet optimized with regard to the design objectives.
- The cost function is evaluated *only* if the placement perturbation fulfills all constraints.
- The cost function is extended with all constraints.

All no-overlap constraints have a preset high weight. Hence, the final placement has practically no overlaps. The geometrical constraints are adaptively weighted as a function of the temperature and the current position of the modules as explained next.

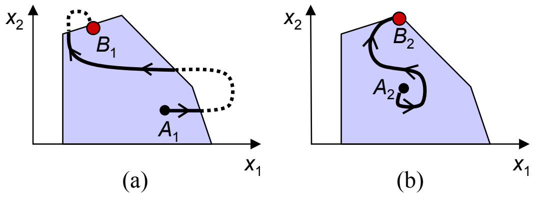
1) Adaptive Constraint Weighting: Every geometrical constraint is linked to a weight. The weight consists of a temperature-dependent component and a position (quality)-dependent component:

$$weight(c_k) = s(t) \cdot d_k \tag{11}$$

where *weight(c_k)* is the adaptive total weight of the geometrical constraint c_k, *s(t)* is the temperature-dependent component and d_k is the quality-dependent component of the constraint c_k.

The temperature-dependent component *s(t)* gradually reduces its weight during the annealing process for all geometrical constraints. The quality-dependent component d_k is dynamically calculated based on the actual relative position of the modules. If a module is placed so that a corresponding constraint c_k is (sub-)optimally fulfilled, then the quality-dependent component d_k is given a small value. If the constraint is "almost violated", then d_k is given a high value (penalty).

Using the adaptive constraint weighting, the optimization process is driven as follows: At the beginning of the process, i.e. at high annealing temperatures, the weights of the geometrical constraints are high. Hence, they strongly influence the cost function in comparison to the objectives. Thus, SA is more likely to accept the perturbations that have higher constraint robustness. These solutions are near the initial solution A_2 as shown in Fig. 3 (b). As the temperature decreases, the constraint weights tend to take lower values. Hence, their importance in the cost function gets lower. In this case, SA is more likely to accept intermediate solutions with optimized objectives. At the end of the process, it is only required to fulfill the constraints rather than to optimize the design robustness according to its constraints.



(a) (b)

Figure 3. Possible progress of a conventional SA algorithm (a), where the optimization is driven only by the objectives. In contrast, (b) shows a possible progress of our algorithm which does not penetrate the boundaries of the solution space. The starting point A_2 is the solution with the maximum constraint robustness constructed using LP.

2) Placement Optimization: Based on the initial placement with the maximum constraint robustness, the placement is iteratively optimized. The cost function of SA includes the adaptive constraint weights *weight(c_k)* of all geometrical constraints c_k; $\forall k \in K$:

$$Cost\ function = f(objectives, no\text{-}overlap\ constraints) + \sum weight(c_k)$$

The possible placement perturbations for a module are 'move', 'rotate', and 'flip'. If the perturbation does not violate any geometrical constraint, the cost function is evaluated according to the objectives as well as to the adaptive constraint weights as shown in Algorithm I. Otherwise (if the perturbation violates a geometrical constraint), then it is rejected and a counter *count* is increased. If the number of rejected perturbations exceeds a temperature-dependent value $k(t)$, then SA randomly chooses another module for perturbation. The output of SA is an optimized placement which fulfills all constraints.

Algorithm I: Placement optimization

```
Read initial placement constructed using LP
REPEAT
  REPEAT
    Apply placement perturbation
    IF no geometrical constraint is violated THEN
      Estimate adaptive weights
      Evaluate cost function (estimate the gain)
      Accept or reject the perturbation based on the gain
    ELSE
      reject the perturbation
      IF $count$++ > $k(t)$ THEN
        choose new module
  UNTIL stop criterion
  $t = \alpha * t$
UNTIL $t < t_{min}$ or desired quality reached
End.
```

IV. Experiments and Discussions

We implemented our approach using the Java programming language. Fig. 4 shows the GUI of our placement tool with a demonstrative example. The white-colored modules have no geometrical constraints. Gray (green) colored modules fulfill their geometrical constraints (sub-)optimally (e.g., the top-alignment constraint c_1). Light gray (orange) means that the constraints are far from being optimally fulfilled, but still not violated. Black (read) indicates that at least one related geometrical constraint is violated.

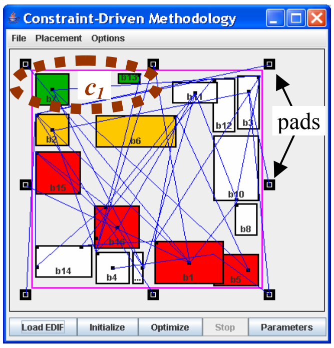
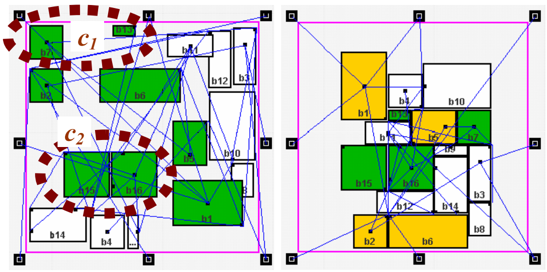


Figure 4. GUI of the placement tool with a simple design example.

The placement is initialized using the LP formulation, where only constrained modules are relevant. The initial placement has overlapping modules (Fig. 5 (a)). Here, two of the given constraints are indicated: The top-alignment constraint c_1 and the symmetry constraint c_2. Next, SA optimizes the placement with the help of the adaptively weighted constraints (AWC) as shown in Fig. 5 (b).



(a) (b)

Figure 5. Initial placement with maximum constraint robustness (a), and the area and wirelength optimized placement while fulfilling all constraints (b). The indicated variables c_1 and c_2 denote top-alignment and symmetry constraints.

We verified our approach using several testbenches within an industrial flow of analog and mixed-signal circuits for automotive applications (Robert Bosch GmbH, Germany). These circuits are real-world testbenches with known optimized placement results laid out manually by experienced analog layout designers.

We compared our results with those of a conventional SA. To ensure a fair comparison, we used in our approach and the conventional SA the same cost function (excluding the adaptive constraint weights for the conventional SA algorithm) and the same cooling schedule and inner-loop criterion. Table I lists some of the test results. In the second column, the number and the correlation degree of the constraints is given for each circuit. E.g., "medium correlation" means that 20-35% of all constraints are correlated to at least one other constraint. The third column indicates whether the desired quality is reached. The desired quality of a layout is reached if it fulfills all constraints and the increase in the design objectives does not exceed 10% compared to the manually placed layout.

TABLE I. COMPARISON BETWEEN OUR PLACEMENT APPROACH AND A CONVENTIONAL APPROACH (SA)

Circuit name	**Constraints**		**Desired quality reached?**	
	Num.	**Correlation degree**	**SA**	**Our approach (LP, SA+AWC)**
CA01	14	Low (<20%)	**Y**	**Y**
CA02	18	Med. (20-35%)	**Y**	**Y**
CB03	41	Med. (20-35%)	N	**Y**
CB04	35	High (>35%)	N	**Y**

Table I shows that a conventional optimization approach based on SA could not achieve the desired quality for larger number of constraints with medium and high correlation degree. In this case, SA spent most of the time performing invalid perturbations that fulfill some constraints but violate simultaneously others (however, the total number of violated constraints was reduced). Table II shows the time behavior of our approach. The run time of LP is negligible compared to that of SA. This allows time efficient constraint verification and feasibility study prior to any layout optimization process.

TABLE II. TIME BEHAVIOR OF THE PLACEMENT TOOL (PC,1.9 GHZ CPU, 512 MB RAM)

Circuit name	**Num. of modules**	**Num. of nets**	**Runtime (sec.)**		
			LP	**SA+AWC**	**Total**
CA01	32	620	0.1	15	15.1
CA02	62	1106	0.1	83	83.1
CB03	92	1398	0.2	196	196.2
CB04	101	1464	0.2	287	287.2
CB05	137	1781	0.2	405	405.2

We integrated our tool in the industrial Cadence DFII design environment [10]. After layout initialization and optimization, the resulting placement was fed back into the Layout Editor.

Fig. 6 shows the automatically generated layout of the circuit "CB05" in the Cadence DFII design environment using our methodology. The space between modules has been preserved for routing. The manual placement of the different circuits is very time consuming and takes several weeks for production designs (due to the multiple and often contradictory constraints). In contrast, our approach delivers in a time efficient manner (Table II) an optimized solution fulfilling the wide range of constraints. It is important to mention that our approach *does not* aim to replace or challenge the high performance layout tools for digital circuits. We rather provide a concept that helps the analog layout designers finding a solution to complex designs which need to fulfill many correlated constraints.

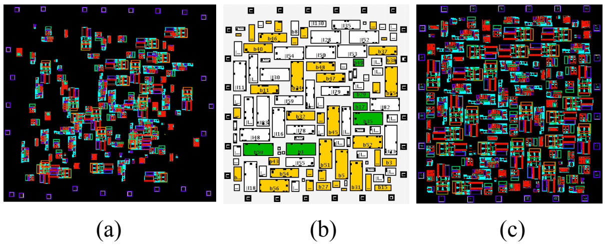

(a) (b) (c)

Figure 6. The circuit CB05 before placement (a), exported and placed automatically using our tool (b), and fed back into Virtuoso Layout Editor (c).

V. CONCLUDING REMARKS

With the increasing complexity of analog and mixed-signal circuits, it is required to fulfill multiple application-specific requirements (constraints). Taking into consideration all given constraints, state-of-the-art analog layout design tools do not verify the feasibility of the layout prior to the time-consuming optimization process. Furthermore, the optimization engine of these approaches is driven only by the design objectives. This is disadvantageous if multiple and correlated constraints exist. Here, many of the steps will be invalid as some constraints are repeatedly violated.

In this work, we present a solution to these problems by introducing a constraint-driven design methodology. An analytical method based on linear programming verifies prior to the design step if a solution that satisfies the constraints exists. If solutions exist, we find the one with the maximum constraint robustness. Based on that, a layout optimization process follows. Here, the process is steered not only by the optimization objectives but also by the adaptively weighted constraints. This allows intelligently searching the solution space by considering the ranges and boundaries of the various constraints in order to prevent constraint violations. We have verified our approach successfully in an industrial design flow.

REFERENCES

[1] G. Gielen and R. A. Rutenbar, "Computer-aided design of analog and mixed-signal integrated circuits," Proc. of the IEEE, vol. 88, no. 12, pp. 1825–1852, Dec. 2000.

[2] J. M. Cohn, D. J. Garrod, R. A. Rutenbar, and L. R. Carley, "Koan/Anagram II: New tools for device-level analog placement and routing," IEEE J. of Solid–State Circuits, vol. 26, pp. 330–342, Mar. 1991.

[3] E. Malavasi, E. Charbon, E. Felt, and A. Sangiovanni-Vincentelli, "Automation of IC layout with analog constraints," IEEE Trans. on CAD, vol. 15, no. 8, pp. 923–942, Aug. 1996.

[4] K. Lampaert, G. Gielen, and W. Sansen, "Analog layout generation performance and manufacturability," Springer, 1999.

[5] F. Balasa, S. C. Maruvada, and K. Krishnamoorthy, "On the exploration of the solution space in analog placement with symmetry constraints," IEEE Trans. on CAD, vol. 23, pp. 177–191, Feb. 2004.

[6] Y. Tam, E. F. Young, and C. Chu, "Analog placement with symmetry and other placement constraints," Proc. of the 2006 ICCAD, pp. 349–354, 2006.

[7] S. Sutanthavibul, E. Shragowitz, and J. B. Rosen, "An analytical approach to floorplan design and optimization," IEEE Trans. on CAD, vol. 10, no. 6, pp. 761–769, Jun. 1991.

[8] S. Reda and A. Chowdhary, "Effective linear programming based placement methods," Proc. of the 2006 ISPD, pp. 186–191, Apr. 2006

[9] S. Koda, C. Kodama, and K. Fujiyoshi, "Linear programming-based cell placement with symmetry constraints for analog IC layout," IEEE Trans. on CAD, vol. 26, no. 4, pp. 659–668, Apr. 2007.

[10] www.cadence.com.