A New Methodology for Constraint-Driven Layout Design of Analog 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 integrated circuits suffers from a lack of automation due to the multitude of complex design constraints. Most of them are specified and considered manually by expert designers (expert knowledge). We present a new methodology that enables the automatic inclusion of expert knowledge in the form of layout constraints. The resulting comprehensive constraint-driven design approach allows defining and verifying the analog layout constraints by transforming them between the different design domains. In order to verify our methodology, we present a new constraint-driven placement optimization engine. It includes a deterministic decision maker. The decision maker is used to solve effectively the hard instances of the optimization problem that are resulting from complex correlations between the constraints. We have verified our methodology successfully by applying it to the placement of analog circuits in an industrial design environment.**

I. INTRODUCTION

Contrary to digital layout design, the layout design of analog circuits is often a manual and time-consuming task. This is due to the various stringent and complex design requirements (*constraints*) that must be considered simultaneously. In analog designs, most of the constraints are currently specified and considered manually by experts (so-called expert knowledge). These constraints are often used implicitly based on the designer's experience. This prevents the effective use of the constraints in design automation.

Each constraint is assigned a specific *constraint type* that represents a classification property for the same class of constraints. For example, the constraint type "IR-drop" is used by *n* individual constraints that define the detailed IR-drop limits between *n* pin pairs of the specific circuit. Constraint types have a clearly defined unit that belongs to the physical, electrical, mechanical, mathematical or geometrical domain.

Today's automatic design approaches require a specific algorithm to handle each constraint type. Hence, considering new constraint types requires the re-development of the algorithmic approaches. In order to avoid this, a new methodology is needed in which new constraint types can be defined and considered "on the fly" by introducing appropriate transformation rules.

The physical design process is characterized by layout design tools that optimize the layout with regard to the design objectives while fulfilling all constraints. State of the art optimization engines are based mainly on heuristics which search the solution space of the constraints based on trial-and-error methods. However, a trial-and-error method does not work efficiently if the constraints are numerous and correlated as it is often the case in analog layout designs. For a successful automatic layout design, the following questions should be positively answered:

- Can the methodology understand and consider automatically all required constraints throughout the whole layout design process?
- Can the methodology automatically find an optimized layout in a reasonable time even in the presence of numerous and highly correlated constraints?

In this paper, we provide a placement methodology that addresses these questions for the first time.

A. Related Work

Several approaches for automatic analog layout design have been developed such as [1][2]. One of the first placement and routing frameworks is introduced in [3]. It considers symmetry, device abutment and merging constraints. In [4], performance-driven layout techniques are presented. These approaches are only suitable for small circuits such as filters and operational amplifiers. The approach introduced in [5] directly takes into account the presence of an arbitrary number of symmetry groups during the exploration of the solution space. In [6], a heuristic to place analog devices taking into consideration symmetry and other placement constraints is described. However, a heuristic does not always succeed in finding a solution if many correlated constraints exist. In [7], a deterministic topological approach based on a hierarchically bounded enumeration of basic building blocks is published.

In [8], we presented a new approach for the placement of analog circuits. It allows the verification whether a layout solution satisfying all constraints exists at all *prior* to placement optimization. The cost function of the algorithm is evaluated based on the objectives as well as on the adaptively weighted constraints. The results are proven to be better than those of the conventional analog placement approaches. In case of numerous and highly correlated constraints, however, only few solutions that satisfy all constraints exist. As the probability of finding these solutions using a conventional trial-and-error heuristic is very low, the algorithm is sometimes trapped in invalid regions of the solution space.

B. Our Contribution

We present a new methodology to solve the above mentioned problems. It has been extensively verified in an industrial design flow. Firstly, we introduce a new concept of a comprehensive constraint-driven layout design approach. The new paradigm allows

978-1-4244-5091-6/09/$25.00 ©2009 IEEE

defining and verifying the analog layout constraints by transforming them between the different design domains based on appropriate transformation rules. Hence, *all* transformable constraints can be handled with our approach *without* the need to modify the fundamental design algorithm. Secondly, we present a new constraint-driven placement optimization engine in order to verify our methodology. The engine searches for an optimized placement while using a deterministic decision maker. This allows solving efficiently the hard instances of the placement optimization problem in case of numerous and correlated constraints. Hence, the decision maker efficiently prevents the algorithm of being trapped in invalid regions of the solution space.

Our paper is organized as follows. In Section II, we describe the fundamental components required in a constraint-driven analog layout design flow. We verify this methodology by applying it to the placement step as shown in Section III. Section IV discusses the experimental results. Finally, concluding remarks are drawn.

II. Components of the Constraint-Driven Analog Layout Design Flow

A design methodology that considers all relevant constraints in a consistent and comprehensive manner throughout the whole layout design process is denoted as *constraint-driven layout design*. In the following, we provide an overview of its essential components.

A. Constraint Representation

Formally, constraints define relations between the set of design variables that are related with each other. A relation between independent design variables represents a *simple constraint*. A relation between dependent design variables represents a *complex constraint*. In the latter case, the constraints share the same set of design parameters and are thus either directly or indirectly correlated. All constraints and all related design data must be uniformly defined. A constraint, such as a delay time constraint, must be preserved at any time during the layout design process in order to guarantee the correct processing of constraint information. Our approach uses CLP (Constraint Logic Programming [9][10]) in order to uniformly represent the constraints. CLP is an extension to logical programming languages like PROLOG. The constraints are described as predicates within Horn clauses. The following notation shows a simple example of a maximum permitted signal delay time constraint (0 ps $\leq d_t \leq$ 200 ps) between two net terminals p_1 and p_2:

$$signalDelay(p_1, p_2, d_t) :- 0 <= d_t , d_t <= 200ps.$$

The uniform representation enforces a common understanding of all constraints among all involved design and verification algorithms. Hence, it is a primary requirement for the development of multilateral layout design and verification algorithms that are able to address the current "constraint challenge".

B. Constraint Derivation (Assignment)

Constraint derivation is the process of either generating new constraints (e.g., by applying new design rules) or deriving sub-level constraints from top-level constraints. The constraint transformation presented in Section C is a special case of the constraint derivation. It derives low-level constraints (e.g., geometrical constraints) from high-level constraints (e.g., electrical constraints) and vice versa.

C. Constraint Transformation

Constraints can be transformed from a higher level to a lower (physical) level (top-down) and vice versa (bottom-up). The result of any transformation must be complete and unambiguous. As an example, transforming an electrical wire resistance constraint between two net terminals into a one-dimensional geometrical maximum distance constraint is shown (using CLP):

$$getCoordinate(x_1, x_2, RQ, DQ) :- x_2 - x_1 = DQ*R/RQ, R<10.$$

The CLP clause *getCoordinate()* defines a transformative relation between the electrical parameter wire resistance R and the geometrical net terminal coordinates x_1 and x_2. This clause also holds a resistance constraint $R<10$ (Ohm). It ensures that only variables x_1 or x_2 will be returned that fulfill the defined resistance constraint. (It is assumed here that square resistance RQ and the lengths of a wire square DQ are constants.) Using this approach, it is now possible to *automatically* consider the (transformed) geometrical constraint by a placement tool during the optimization process. For the definition of complex constraints, x_1, x_2 and R might also depend on additional clauses that may, for example, describe a temperature dependency. This simple example demonstrates the flexibility and power of our approach. Once a transformation relation between, for example, electrical and geometrical constraints can be formulated, *all* transformable electrical constraints can be handled with our approach *without* the need to modify the fundamental placement algorithm.

D. Constraint Verification

Constraint verification ensures correct application functionality as well as design quality and reliability. During the analog layout design, various constraint types must be considered simultaneously. Many of them are complex constraints. The fulfillment of these constraints cannot be verified with conventional verification approaches. This is due to the fact that these approaches require one specific verification tool or algorithm for each constraint type.

Recently, we introduced an approach that allows verifying complex linear constraints for the first time [10]. This meta-verification approach is based on dividing every complex verification problem into simpler problems, which in turn can be verified using existing verification algorithms and tools. The meta-verification does not replace existing verification and simulation tools. It rather offers a method to combine the tasks of these tools.

E. Mutual Constraint Dependencies

In order to solve mutual constraint dependencies, we implemented a linear constraint solver [10]. It is based on the simplex algorithm and provides logical conclusions on linear equations and inequations. In this work, we use this solver in order to (1) verify if either a set of linear and piecewise linear constraints can be met at all or a mutual constraint conflict exists, and (2), if possible, to find a solution that meets all constraints simultaneously. In this work, we show the usefulness of this method in the light of the analog placement problem (Section III C).

III. Constraint-Driven Placement Algorithm

We verify our constraint-driven design methodology by applying it to the placement of analog circuits. The placement procedure is divided into two main steps. First, the analytical method linear programming (LP) is used in order to find out whether the placement under the given constraints is feasible [8]. If feasible, LP constructs an initial placement satisfying all constraints. In the second step, the placement is optimized with regard to the objectives. Our optimization engine uses two methods simultaneously, a heuristic and an analytical method. Both interact and solve efficiently the optimization problem as described in Section III B.

A. Initial Placement Using Linear Programming

Using linear programming (LP), the layout information and the constraints are represented as a linear system of equations. (Please refer to [8] for a detailed description.) By using LP for constructing the initial placement, we assert if a solution for all constraints exists at all. If this is the case, LP constructs an initial placement satisfying all constraints. Assuming that each constraint has a range of permitted values, the modules are placed so that their constraint parameters are as far away as possible from their critical boundary values. While assuming a generally linear or convex placement

optimization problem, the placement has in this case maximum *constraint robustness*. Here, small perturbations of the placement solution will most probably not violate any constraint.

B. Placement Optimization Engine

The optimization engine consists of two parts. A heuristic method is used effectively in order to optimize the placement by minimizing a cost function. It interacts with a deterministic decision maker (DM). The DM is called by the heuristic whenever the optimization cannot find a solution that satisfies the numerous and correlated constraints simultaneously (solution trapping).

The heuristic starts with the initial placement which has maximum constraint robustness but is not yet optimized with regard to the design objectives. The cost function is extended with all constraints and is evaluated only if the placement perturbation satisfies all constraints. In case of correlated constraints, the optimization often traps at several stages and finds only solutions that do not fulfill all constraints. In order to solve this problem, the placement information and the constraints are delivered to the DM in order to analytically find a solution that satisfies *all* constraints.

In more detail, the optimization engine perturbates the placement and verifies the fulfillment of the constraints. For example, it moves a selected module m_i to a new position. If the perturbation does not violate any constraint, then the cost function is evaluated. Otherwise, the perturbation is rejected and a counter "*count*" is increased (Algorithm I). Then, a new perturbation is applied on m_i and the constraints are verified again. In case the number of consequent violations *count* exceeds a temperature dependent number $k(T)$, the decision maker is called. The DM analytically finds solutions that satisfy the constraints and feeds them back to the heuristic. The whole procedure is repeated until a specified stop criterion is reached.

Algorithm I: Optimization Engine

```
1: Read initial placement constructed using linear programming
2: REPEAT
3:   Select a module m_i for perturbation
4:   REPEAT
5:     Apply placement perturbation
6:     IF no geometrical constraint is violated THEN
7:       Evaluate the cost function (estimate the gain)
8:       Accept or reject the perturbation based on the gain
9:     ELSE
10:      reject the perturbation
11:      IF count++ > k(T) THEN
12:        Call the decision maker for violating module m_i
13:        IF solutions exist THEN
14:          Return solutions to SA
15:          Go to 5
16:        ELSE i = i + 1; go to 3
17:  UNTIL number of perturbations exceeds a predefined value
18:  T = α * T
19: UNTIL T < T_min or desired placement quality reached.
```

The output of the algorithm is a placement optimized with respect to its cost function. To our knowledge, this is the first analog placement algorithm having mixed heuristic and deterministic (local optimization) elements at the same time. By applying a controlled deterministic behavior when needed, it combines the advantages of both classes with a relatively small runtime (see Section IV).

C. Decision Maker

The input data to the decision maker (DM) are the current position of the circuit modules and the relevant constraints. Its task is to find a new position of the circuit module m_i that is to be perturbed while fulfilling *all* related constraints. The DM is called by our modified SA algorithm when the heuristic procedure fails for $k(T)$ number of times to find a valid position for m_i that fulfills all constraints. The term $k(T)$ is the number of failed attempts. It is linearly dependent on the SA cooling temperature T. The DM analytically searches for new valid positions for m_i that fulfill all constraints:

- If the DM does not find a valid new position for m_i, then the current position of m_i is the only valid one at this point in the design process. Consequently, SA keeps m_i in its position and applies a new placement perturbation on another module.
- If the DM finds one or more valid positions, then it returns them all to the SA algorithm for evaluation. Consequently, the SA decides which solution best optimizes the placement.

The outline of the DM algorithm is given as follows:

Algorithm II: Decision Maker

```
1: Get module m_i and add it to List L
2: Get all related constraints c_j applied to the module m_i
3: For each constraint c_j
4:    Get all involved modules m_j
5:    If m_j ∉ L
6:       Add m_j to L
7: Formulate the problem for m_i, c_j, and m_j ∈ L (see below)
8: Solve the linear system of equations
10: Return the solutions (if they exist) to SA
```

Next, we present the problem formulation as used by our approach. The objective is to minimize all P_j variables, where P_j is a measurement of the fulfillment of the constraints c_j, $\forall\, j \in J$, where J is the number of the shared constraints between m_i and m_j. The origin of m_i is defined by the coordinates (x_i, y_i) of its lower left corner. W and H are the maximal allowed width and height of the placement.

$$\text{Minimize } \sum P_j; \quad \forall j \in J; \tag{1}$$

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

$$w_i = getWidth(m_i); \quad h_i = getHeight(m_i); \tag{3}$$

$$x_i + w_i \leq W; \quad y_i + h_i \leq H; \tag{4}$$

for each constraint c_j, $\forall j \in J$

$$m_j = getModule(c_j); \tag{5}$$

$$x_j = getX(m_j); \quad y_j = getY(m_j); \tag{6}$$

$$w_j = getWidth(m_j); \quad h_j = getHeight(m_j); \tag{7}$$

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

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

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

We assume that each constraint is fulfilled in a the range $[\min_{ij}(c_j), \max_{ij}(c_j)]$. The mean of the boundary values is given by *opt*. The last three equations guarantee that the module m_i is placed within $[\min_{ij}(c_j), \max_{ij}(c_j)]$ as near as possible to the position *opt*. The distance is given as center-to-center Manhattan distance for m_i and m_j. The system of equations is solved and the solutions are passed to the SA for evaluation according to all of the placement objectives.

IV. Experimental Results

We implemented our approach in the Java programming language. Several analog circuit test benches from automotive IC applications were used to validate our approach. The test benches are real-world circuits with known optimized placement results laid out manually by experienced analog designers. All experiments were done using an AMD PC, running at 1.9 GHz with 512 MB RAM.

In order to show the advantages of our methodology, we compare it with a conventional SA optimization algorithm and with that in [8]. The latter approach is also based on the SA algorithm. However, the algorithm is driven not only by the objectives, but also by the adaptively weighted constraints. To ensure a fair comparison, we used in all approaches the same cooling schedule and inner-loop criterion for the SA. Table I lists some of the results. In the second column, the correlation degree of the constraints is given. "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 of a layout is reached. This is the case if the layout fulfills all constraints and the increase in the design objectives does not exceed 10% compared to the manually placed layout.

TABLE I. COMPARISON OF OUR PLACEMENT METHODOLOGY WITH A CONVENTIONAL APPROACH (SA) AND WITH THE APPROACH PRESENTED IN [8].

Circuit	Constraints		Desired quality reached?		
	#	Degree of Correlation (%)	SA	LP + SA [8]	LP + SA + DM
cir_x1	14	Low (<20)	**Y**	**Y**	**Y**
cir_x2	18	Med. (20-35)	**Y**	**Y**	**Y**
cir_x4	35	High (>35)	N	**Y**	**Y**
cir_x6	54	Med. (20-35)	N	N	**Y**
cir_x7	48	High (>35)	N	N	**Y**

Table I shows that a conventional optimization approach solely based on SA could not achieve the desired quality for a larger number of constraints with medium and high correlation degree. In this case, SA spends most of the time performing invalid perturbations that fulfill some constraints while violating others at the same time. The approach in [8] shows better results than that of the conventional SA approach. However, the trial-and-error heuristic fails for complex designs with numerous and highly correlated constraints as only few neighboring valid solutions exist. We overcome this problem by including the decision maker. It finds analytically valid solutions and passes them to the SA for evaluation. As a result, the design quality was reached for all test benches.

We also compare our approach with state-of-the-art analog placement tools in terms of runtime and area usage. However, this comparison is difficult. First, the approaches were tested on different computers and platforms. Second, available benchmark circuits include only few constraint types such as symmetry. Nevertheless, Table II shows the comparison between our methodology and other approaches based on two benchmark circuits. The approaches in [5][7][11] take only the symmetry constraint into consideration during placement. Hence, we also restricted our approach to this constraint in order to allow a direct comparison.

While the results of our new methodology and the state-of-the-art placers are comparable, it is important to mention that our goal was not to compete with the other approaches solely in terms of the runtime and area usage. We rather present a methodology that enables a layout designer to define new constraint transformation rules whenever needed during the design process. This allows including new constraint types without the need to modify the fundamental placement algorithm. Furthermore, our methodology can automatically consider all constraints during the optimization process even if they are highly correlated. All these properties are missing in the state-of-the-art analog placement approaches.

V. CONCLUDING REMARKS

The automatic inclusion of expert knowledge in the form of constraints is one major precondition in order to overcome the analog IC design problem with its fundamental lack of automation. We present a new constraint-driven design methodology that addresses this issue. It includes two modules. The first module is a comprehensive constraint system for capturing, transforming, and verifying the constraints. This system is accessed by the second module consisting of the layout optimization engines such as placement algorithms.

We introduce a hybrid optimization algorithm for the placement of analog circuits. It combines the advantages of heuristic and analytical approaches. A heuristic optimizes efficiently the placement with regard to the objectives. It calls an analytical decision maker in order to solve hard instances of the optimization problem resulting from multiple and correlated constraints. We have shown that our methodology is comparable to the state-of-the-art analog placement approaches in terms of area optimization and runtime. However, it is superior in terms of the flexible and comprehensive constraint consideration.

REFERENCES

[1] R. Rutenbar and J. Cohn 2000, "Layout tools for analog ICs and mixed-signal SoCs: a survey," In Proc. of ISPD, pp. 76–83, 2000.

[2] H. Graeb, F. Balasa, P.-H. Lin, R. Castro-Lopez, and M. Strasser, "Analog layout synthesis - Recent advances in topological approaches," Proc. of Design, Automation and Test in Europe (DATE), pp. 274-279, Apr. 2009.

[3] 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 JSSC, vol. 26, pp. 330–342, Mar. 1991.

[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," ICCAD, pp. 349–354, 2006.

[7] M. Strasser, M. Eick, H. Graeb, U. Schlichtmann, F. M. Johannes, "Deterministic analog circuit placement using hierarchically bounded enumeration and enhanced shape functions," Proc. of the 2008 ICCAD, pp. 306 -313, Nov. 2008.

[8] A. Nassaj, J. Lienig, G. Jerke, "A Constraint-Driven Methodology for Placement of Analog and Mixed-signal Integrated Circuits," Proc. of the 14th IEEE Int. Conf. on Electronics, Circuits and Systems (ICECS), pp. 770-773, Aug. 2008.

[9] J. Jaffar, S. Michaylov, P. Stuckey, and R. Yap, "The clp language and system," ACM Trans. on Programming Languages and Systems, vol. 14, pp. 339–395, Jul. 1992.

[10] J. Freuer, G. Jerke , J. Gerlach, and W. Nebel, "On the verification of high-order constraint compliance in IC design," Proc. ACM/IEEE Design, Automation and Test in Europe (DATE), pp. 26–31, 2008.

[11] L. Po-Hung and L. Shyh-Chang; "Analog placement based on novel symmetry-island formulation," In ACM/IEEE Design Automation Conference (DAC), pp. 465–470, Jun. 2007.

TABLE II. COMPARING THE AREA USAGE AND TIME BEHAVIOUR OF OUR METHODOLOGY WITH OTHER APPROACHES BASED ON TWO INDUSTIAL CIRCUITS. THE CIRCUITS INCLUDE ONLY SYMMETRY CONSTRAINTS. TIME IS MEASURED IN SECONDS.

Circuit Name	Modules		Approaches							
			[5]		[7]		[11]		Our Work	
	#	Area (10^3 μm^2)	Area (%)	Time	Area (%)	Time	Area (%)	Time	Area (%)	Time
biasynth_2p4g	65	100% (4.7)	114.89	246	104 96	337	104.68	22	108,36	407
lnamixbias_2p4g	110	100% (46)	109.35	726	107.68	387	105.72	43	109,84	639