Automation of Analog IC Layout – Challenges and Solutions

Juergen Scheible
Robert Bosch Center for Power Electronics
Reutlingen, Germany
juergen.scheible.de@ieee.org

Jens Lienig
Dresden University of Technology
Dresden, Germany
jens@ieee.org

ABSTRACT
Physical analog IC design has not been automated to the same degree as digital IC design. This shortfall is primarily rooted in the analog IC design problem itself, which is considerably more complex even for small problem sizes. Significant progress has been made in analog automation in several R&D target areas in recent years. Constraint engineering and generator-based module approaches are among the innovations that have emerged. Our paper will first present a brief review of the state of the art of analog layout automation. We will then introduce active and open research areas and present two visions – a "continuous layout design flow" and a "bottom-up meets top-down design flow" – which could significantly push analog design automation towards its goal of analog synthesis.

Categories and Subject Descriptors
B7.2[**Integrated Circuits**]: Design Aids

General Terms
Algorithms, Design, Verification.

Keywords
Analog design; layout; constraint engineering; design methodology; physical design; analog layout automation

1. INTRODUCTION
While physical design automation of analog IC design has seen significant improvement over the past decade, it has not advanced at anything like the rate of its digital counterpart. This shortfall is primarily rooted in the analog IC design problem itself, which is very much more complicated even for small problem sizes: it deals with a large number of specific circuit classes; it requires a customized design approach for each circuit class; and analog circuits are very susceptible to noise and process variations. In particular, the work and costs involved in producing analog layout is a serious bottleneck in IC design, despite numerous attempts at automating the process. Furthermore, the analog design problem lacks a sufficiently comprehensive and exact descriptiveness in conventional CAD approaches [1-3].

Advances in analog layout automation have been made however in recent years in many R&D target areas, such as generator-based module approaches [8,9,11-13,16]; and we have witnessed the emergence of constraint engineering to support top-down design styles [2,6,10,18,19,24].

Unfortunately, achievements made thus far fall way short of meeting the needs of advanced analog layout automation. Active new research areas needed to bridge this deficiency gap include

- The next generation of constraint engineering approaches;
- Context-aware layout design;
- Advanced methods for assisted layout design;
- The development of top-down design approaches, tailored for analog circuits, and very powerful bottom-up design procedures, such as module-generator-based and template-based design.

The purpose of this paper is to give an up-to-date overview of analog design automation, highlighting physical design, its specific characteristics and its current research areas from both an industrial and an academic perspective. Specifically, we will first review the analog layout design problem itself and discuss various aspects of today's design flows. We then introduce active and open research areas and finally present two visions, a *continuous layout design flow* and a *bottom-up meets top-down design flow*. It is our hope that these new design paradigms will significantly enhance analog design automation and bring us one step closer to the long-awaited goal of analog synthesis.

2. THE LAYOUT OF ANALOG CIRCUITS

2.1 Sources of Complexity
The majority of today's ICs are mixed signal designs, i.e., they consist of analog and digital circuits (blocks, partitions). Both analog and digital designers claim their design tasks are "highly complex", and in fact both are right, but in a different sense.

Analog designs are characterized by a much richer and more complex set of design constraints that need to be considered simultaneously and which may span several domains (e.g., electrical, electro-thermal, electro-mechanical, technological, geometrical domain). Therefore, in typical mixed signal ICs, the effort needed to design the analog part often matches or even exceeds the effort for the digital part by far. This is true despite the fact that analog modules typically contain only a small number of devices compared to digital ones.

ISPD'15, March 29–April 1, 2015, Monterey, CA, USA.
ACM 978-1-4503-3399-3/15/03.
http://dx.doi.org/10.1145/2717764.2717781

Therefore, when talking about complexity, we prefer to distinguish between (1) *quantitative* complexity, as observed in digital designs, referring mainly to the number of design elements ("More Moore"), and (2) *qualitative* complexity. The latter is rooted in the diversity of the requirements to be considered ("More than Moore"), as found in analog designs.

2.2 Problem Description

Any physical design process can be understood as a course of actions that aim at optimizing a layout with regard to design objectives while meeting *design constraints*. Constraints generally belong to one of the following four categories:

(1) *technological constraints* that enable the fabrication and are derived from technological restrictions,
(2) *functional* (or: *electrical*) *constraints* that ensure the desired electrical behavior of the design,
(3) *geometric* (or: *design-methodical*) *constraints* that are introduced to reduce the overall complexity of the design process, thus facilitating or even enabling the use of design tools, and
(4) *commercial constraints* that arise from chip area or packaging requirements, and the like.

Whereas technological and functional constraints are mandatory, geometric constraints are in principal optional (depending on the layout design tools in use). Any given constraint can be assigned to a *constraint type* that defines the algorithms for its consideration in the design.

A constraint can be *implicit* or *explicit*. An implicit constraint is not clearly expressed: it may be a plain textual note; or it may arise from assumptions intrinsically built into circuit descriptions or layout generators. Explicitly defined constraints are visible and accessible to design algorithms.

Explicitly defined functional constraints are the primary means for describing the designer's intent. On average, each design object (instance, net, path, etc.) in an analog IC design must comply with a larger and more comprehensive set of functional constraints to fulfill its intended function than is the case with digital design. The primary reason for this observation is the higher level of functional abstraction achievable (and offered) in digital designs. This allows for more robust operation requiring fewer constraints to assure the intended function, compared to the quasi low-level mode of operation in analog designs.

In addition, many constraints may still be unknown when the analog design process begins, due to the qualitative complexity described above. This often renders traditional automatic top-level design planning for analog IC designs impossible. It is one of the reasons that highly skilled design engineers are needed to plan and implement top-level design manually.

This constraint-related problem also makes algorithm and tool development for analog IC design much more difficult as the number of specific design algorithms needed may increase with each new constraint type. Considering today's conventional design algorithm development approach (one constraint type and one algorithm to handle it), its weakness becomes all too apparent when faced with new complex constraints affecting multiple design parameters simultaneously and vastly outnumbering simple constraints. This is one of the primary reasons why conventional analog design automation still lags behind its digital counterpart. We will make some suggestions on how to overcome this bottleneck in Sec. 5.

Considering and verifying all mandatory constraints in analog design automatically is currently not possible, mainly for three reasons. First, the constraints are often used implicitly, i.e., based on the designer's experience (expert knowledge), due to the lack of identical tool representation. Second, analog designers are constantly confronted with new requirements that are application specific and cannot be translated "on the fly" into functional constraints. Third, the number of constraints as well as the correlations between them are increasing continuously with more and more contradictory constraints that cannot be met simultaneously. Sections 5.1 and 5.2 suggest new design flows that address this issue of incorporating complex and conflicting constraints.

2.3 Today`s Analog Physical Design

2.3.1 Design Flow: Digital vs. Analog

While the design steps for digital circuits are mostly separated from each other and are performed sequentially, analog design steps typically overlap and several steps are performed simultaneously. For example, device generation, module placement and routing are usually executed simultaneously (Fig. 1).

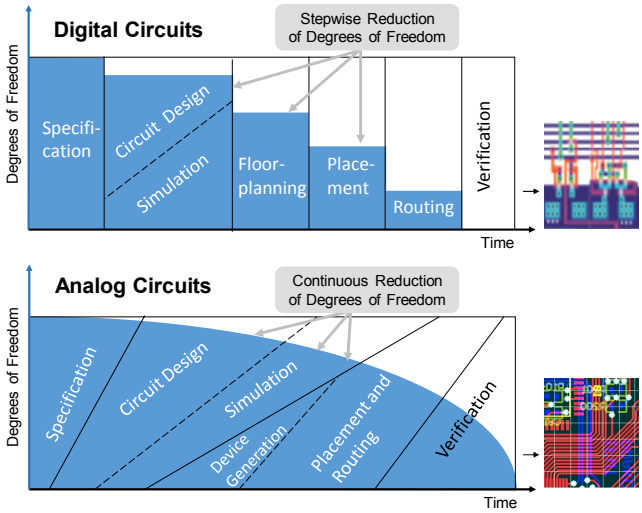

Figure 1. Simplified design flow for digital circuits (above) and for analog IC design (below) where steps typically overlap and are tightly linked. Both flows are also characterized by a reduction in the design freedom throughout the process.

Any design flow is also characterized by a chronological reduction in the *design freedom* which is reduced stepwise in digital designs and continuously reduced in analog circuit designs (see Fig. 1). In general, a feasible solution for a specific design problem is obtained by sequentially removing all degrees of design freedom by sequentially transforming functional representations with many degrees of design freedom into equivalent ones with fewer degrees. For example, one may transform a given functional specification into a netlist, which is subsequently transformed into a floorplan, a placement order, a wired layout and finally into a physical mask layout which contains no further degree of design freedom.

2.3.2 *Design Evolution: Schematic-Driven vs. Constraint-Driven*

Despite the advances that have been made over the past 10 years or so, current analog design tools cannot fully cover the entire design process for analog or mixed-signal circuits. They are either restricted to specific parts of the design flow or require the intervention of an expert designer. Thus, most analog circuits are currently designed interactively, in terms of schematics, followed by layout design ("schematic-driven") and several (iterative) verification steps.

Many experts agree that the ultimate goal of fully automated analog design (analog design automation) can only be achieved if the current schematic-driven layout (SDL) methodology first evolves into a *constraint-driven design paradigm* as a necessary intermediate step [4-7]. This is based on the belief that we first need a methodology that enables the inclusion of "expert knowledge" in the form of constraints (i.e., specifying requirements). Secondly, based on this we need the ability to verify them (i.e., checking requirements). Only then (in a third step) will we be able to tackle the task of analog layout synthesis (i.e., fulfilling requirements) in a comprehensive and consistent manner. In other words, "analyzing" and "verifying" capabilities are a precondition for "synthesizing" [4] (Fig. 2).

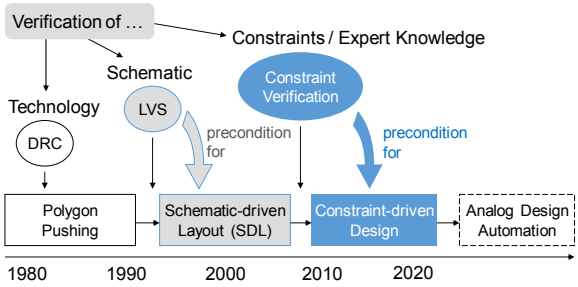

Figure 2. The evolution of analog physical design methodologies towards the goal of a fully automated analog design flow.

2.3.3 *Design Styles: Top-Down vs. Bottom-Up*

The existing layout design styles can be divided into "top-down" and "bottom-up".[1] A *top-down design* approach addresses the design problem from a high-level strategic perspective. Here it is assumed that each design object itself is (mainly) independent of its design context such as neighboring design elements and context-specific design rules. Thereby, explicitly defined design constraints are often used to enforce the designer's intent. Whenever the requirements of a design problem can be completely described by such a set of constraints, this top-down approach is the preferred design style. Examples are simple standard analog circuits, such as current mirrors, cascodes, and bandgaps.

There are however many cases where this approach doesn't lead to viable solutions. Designs that depend on design context information, such as RF designs, signal sensing designs, power-stage designs and the majority of advanced node designs, are required to embrace a *bottom-up design* style in order to include necessary external information during layout implementation. A bottom-up design approach addresses the design problem from a more tactical perspective while assuming that multiple design objects are required to cooperate in concert to achieve a desired design result. Today, these designs primarily rely on expert knowledge and manual work while using implicit constraints.

Commercial tool chains address both design styles. For example, while schematic-driven layout (SDL) tools are top-down approaches, the widely used concept of parameterized cells (see Sec. 2.3.4) uses the bottom-up style.

2.3.4 *Design Automation: Optimization vs. Procedures*

As already indicated, design automation of analog circuits is currently characterized by two different design styles – top-down vs. bottom-up. Both represent different paths towards fully automated analog design: an optimizing approach vs. a procedural approach. While so-called "optimizers" perform design automation top-down, the procedural approaches ("procedures") generate the final layout with the bottom-up style.

As illustrated in Fig. 3, the top-down approach makes use of optimization-based tools similar to conventional digital flows. Their overall structure is given by an optimization engine generating solution candidates and an evaluation engine selecting the "best" candidates based on design objectives in a loop-wise manner [23]. An optimizer is capable of producing new (genuine) design *solutions*.

In contrast, procedures re-use expert knowledge with the *result* of solutions previously conceived and captured in a procedural description by a human expert, thus imitating the expert's decisions in a straight-forward manner. Typical examples are the widely used concepts of parameterized cells as provided by Cadence's PCells [14] or Synopsys' PyCells [15].

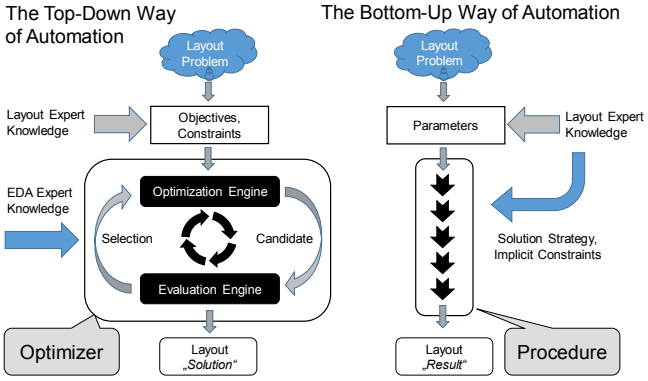

Figure 3. Top-down optimization vs. bottom-up procedures. The top-down approach makes use of optimization-based tools similar to conventional digital flows. The bottom-up way of automation reproduces a design solution previously conceived and captured in a procedural description by a human expert. The grey arrows indicate the data flow of the layout design process. Whereas optimizers are built by EDA experts, procedures are built by layout experts (blue arrows).

Most of the attempts at automatizing analog designs focus on migrating digital design strategies into the analog world. Hence, the above-mentioned top-down "optimizing approach" – successfully applied in digital designs – has been modified and deployed to address analog requirements. However, despite various initia-

[1] In this context the term "top-down" is not used in the usual sense of relating to a design hierarchy, but to denote the approach to a design solution.

tives, no viable "analog synthesis solution" has emerged as yet. In our opinion, this is mainly due to the fact that analog "expert knowledge" cannot be translated into formal expressions of high-level, abstract design requirements (constraints). Hence, we believe "bottom-up automation" based on the above mentioned procedural approach is an indispensable element in any future "analog synthesis flow" (see Sec. 5).

3. ACTIVE RESEARCH AREAS

In this section we describe some current research areas in analog design automation. We focus on aspects which we believe to have the largest impact on urgently needed improvements.

3.1 Constraint Engineering

The specification step in a design process defines certain design requirements, such as the operating frequency of a circuit or a symmetry requirement for a current mirror bank. Subsequently, these requirements are expressed as specific functional constraints. These functional constraints then have to be translated into geometric constraints in order to consider them properly in the layout design step. The required operating frequency could result in a maximum RC value, for example, which is subsequently translated into a maximum wire length of a net. The symmetry requirement for a current mirror bank would lead to a (preferably quantified) matching constraint attached to the set of related elements.

The formal constraint representation is a key requirement for a constraint-driven design flow and – as the examples have shown – extra resources are needed to transform constraints automatically from the electrical to the physical domain.

Specifically in the physical domain, the consideration of geometric constraints, such as alignment, placement pattern, orientation, during design implementation has steadily improved over the last couple of years [6,10,19]. Unfortunately, despite this progress, we are still a far cry from advanced analog layout automation. Additionally, methods for checking the completeness of a set of constraints, their self-consistency as well as the verification coverage achieved, need to be developed further to assure IC functionality, reliability, robustness, etc. [5,6].

Other key enabling research areas are improved methods for constraint handling throughout hierarchies (transistor – block – chip levels) [25] and design domains (chip – board – system levels) [26]. Given that design domains today are characterized by different and almost independent tool environments, we are convinced that this requires tool-independent constraint data management to achieve consistency over the above domain levels (*constraint propagation*) [18,24].

3.2 Context-Aware Physical Design

As in digital design, analog circuits face multiple design for manufacturability (DFM) issues due to reduced structural dimensions. Examples are context-dependent design rules, layout-dependent effects and reliability problems caused by electromigration and electrical overstress. These issues can be addressed at the design implementation stage (1) by introducing *context awareness* to the layout generation tools and (2) by using *query functionality* to identify critical structures. For instance, a PCell placing a set of matching transistors might obtain its context (e.g., the distance to the trench) as a parameter value and then act accordingly to assure device matching (e.g., by adjusting the space between the set and the trench in order to avoid local proximity effects).

3.3 Module Generators

It is widely agreed that bottom-up layout design based on module generators is well suited for basic analog circuitry. Module generators, such as the PCells concept, should therefore be upgraded to include "higher-up" design levels.

Basic layout devices are usually available as *device generators* that procedurally create the appropriate device layout based on device parameters. Augmenting this purpose, extensive research is going into the development of higher-level parameterized *module generators*. These are able to generate entire layouts for basic analog circuits by hierarchically employing other generators and creating in-between interconnections [8,11,16].

While this represents a smart way of incorporating valuable expert knowledge in the layout automation, it implies a trade-off between module re-usability and generator development effort. Thus, there are many on-going studies into improving the techniques for creating complex generators that have brought forth powerful new tools for PCell development such as PCell Designer [12,13] and others [8,11]. Accompanying this intention in the layout domain, complex procedural circuit generators have also emerged on the schematic side for circuit generation [17] and testbench generation, along with sophisticated novel approaches for their development [8,9].

The advancement of module generator approaches is essential to achieve the overarching goal of elevating the seamless schematic-driven-layout design flow to higher levels of abstraction (Fig. 4). This move accommodates a designer's way of engineering circuit functions beyond that of basic devices. It also mirrors the level of functional abstraction achieved to date in the digital domain. Providing consistent module generators both in schematic and layout not only yields an immediate increase in design productivity in both domains, but also allows for the consideration of layout parasitics when initially sizing a schematic circuit, thus obviating the need for costly design recursions.

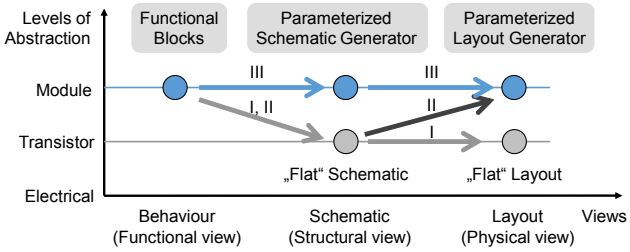

Figure 4. Analog design flows illustrated by design views and compared by levels of abstraction. (I) Today's schematic-driven flow: schematic and layout design at transistor level. (II) Layout module generators (e.g., "module-PCells") improve automation in layout design but encounter a second hierarchical break in the flow. (III) Schematic module generators improve automation in circuit design; additionally they eliminate hierarchical breaks by elevating the entire flow to a higher level of abstraction.

Further works in this context focus on the task of migrating module generators to other semiconductor technologies using templatization [11]. This effort addresses the challenges of incorporating high-level layout generators into the SDL-design flow by means of circuit structure recognition [27], and by combining PCells with other automation approaches. A particular problem

with hierarchical module generators is the necessity to verify input parameters for sub-entities of the module at the module level and efficiently pass them down through the hierarchy [13].

3.4 Assisted Physical Design

As previously mentioned, algorithmic approaches aiming for fully automated layout synthesis have not established themselves to any great extent in an industrial setting to date. Nevertheless there is great interest in algorithmic approaches that assist the layout designer in his or her daily routine. Some promising approaches are described below.

Online DRC support is already widely used. Current development efforts are aimed at improving calculation speed and functionality. Selected applications of these solutions focusing on specific design steps lead to guided placement (e.g., [20]) and guided routing (e.g., [21]) with automatic recognition of relevant design rules and constraints.

Closely related to guided placement is interactive compaction. Although online compactors already exist, there is a demand for further improvements because, as mentioned in Sec. 3.2, the distance rules are getting more and more complicated. Improving the state of the art could also lead to sophisticated decompactors which can repair incorrect layout distances or overlaps.

Although powerful routing engines are available in analog layout environments, the use of these "autorouters" is not very popular in the analog community. One reason is that they are incapable of calculating the net segments to comply with current density requirements. On-going developments aim at improving routing algorithms to understand current constraints, thus enabling *current-driven routing* [22].

Assisted physical design leads to a growing number of "assistant functions"; this tendency may worsen user-friendliness. This, in turn, requires improvements in the human-machine (HM) interface in order to keep the resulting increases in tool functionality in sync with tool usability. A solution could be the application of interactive user interfaces commonly found in other domains, such as tablet computers.

4. OPEN RESEARCH ISSUES

Analog design automation is a very vibrant research topic. In our opinion, the open research issues cited below are just a few of the issues that will have the greatest impact on design automation in the future.

Pursuing the goal of constraint-driven design [4,6,7], EDA research has put considerable research effort into constraint engineering over the past decade. Most elementary to this is the question of constraint data management, which basically specifies how constraints can be described in an abstract and uniform manner. Next-generation design flows need to drive this generalization towards the coalescence of circuit, layout and constraint data, such that constraints can be consistently used across various domains, between different applications, and throughout all design steps in the flow [6,18,24,25].

In addition to the generalization of constraints there is a need for further constraint standardization to enhance algorithmic verifiability. This is analogous with design rule checking (DRC), where all rules must be described such that they can be verified with a predefined set of graphical functions. Thus, the constitution of these functions defines the description format of the rules. Equivalently, the description format for constraints is best developed in absolute compliance with the methods that will be available for their verification. For example, [5] contributes a meta-verification environment where constraints are represented with a formal description based on Horn clauses.

Another topic in this context is the universality of constraint-description formats for the purpose of constraint extensibility. In order to promote a resolute usage of constraints in day-to-day design, designers ask for the means to introduce new, proprietary constraint types on their own account. This not only implies sufficiently flexible constraint formalization, as presented in [24], for example, but also the availability of appropriate, user-friendly tool interfaces with intuitive constraint definition capabilities.

Technological and functional constraints associated with layout solutions are verified with automated tools like DRC and LVS, which use graphical layout data. Therefore layout generators, which are in fact software, are verified by checking their layout results with these tools. For parameterized generators (e.g., PCells), a large number of instances are created for this verification purpose, each with a differently permutated set of parameter values. However, an issue arising here is that the full parameter space of a layout generator cannot be explored; hence, the extent of test coverage is unknown. Against the backdrop of the ever-increasing complexity of layout generators[2], there is an urgent need for new approaches to solve this issue. The generator's code could be investigated, for example, as part of a solution. Another approach could be to model the behavior of generators in order to efficiently find critical parameter value sub-spaces.

5. TWO VISIONS OF ANALOG LAYOUT

The aforementioned open research issues are derived from a careful analysis of the state of the art in industrial analog design. In the following, we would like to expand on this analysis by adding two visions. While the first suggestion for a continuous layout design flow could significantly enhance the current interactive layout style (Sec. 5.1), the second vision conflates the advantages of the top-down and bottom-up design styles (Sec. 5.2).

5.1 A "Continuous" Layout Design Flow

5.1.1 A Blind Spot in Today's Analog Layout Flow

As mentioned in Sec. 2.3.1 and illustrated in Fig. 1, the reduction of *degrees of design freedom* in today's interactive analog layout design style seems to occur in a continuous way. Unfortunately however, this observation is *not* an intrinsic capability of this flow, rather it is the result of a vast number of recursions! These recursions result from repeating the same design steps, notably placement, routing and device generation, again and again in order to make necessary modifications. Previously determined parameters, such as the folding characteristic of a transistor or the width of a wiring segment, have to be updated due to constraints which emerge at a later stage in the design process and therefore cannot be foreseen. These modifications account for the greatest amount of time and effort in analog layout work. The efficiency of the widely used interactive layout style can therefore be greatly improved by reducing the number of these recursions.

[2] Module generators with up to 50 parameters are in use in the automotive electronics industry of today.

Before outlining our proposal for a solution we need to discuss the root cause of this problem, which is that the *edit commands* used in today's layout editors are simple implementations of *design steps* for the purpose of interactive usage. The problem arising from this similarity can be best explained by looking at how the editor commands affect the degrees of design freedom.

Each design parameter (i.e., the property of a design element such as a wire width) can be regarded as a degree of design freedom. When assigning a value to a design parameter, the related degree of freedom is eliminated. Two typical layout tasks and their corresponding editor commands shall be considered next to exemplify their impact on the degrees of design freedom.

The task "routing a net" is usually performed by drawing paths. A path command simultaneously eliminates all degrees of freedom an electrical connection can have (i.e., layer assignment, x- and y-coordinates, Steiner nodes, wire width, and so on). This is an inevitable consequence of the command itself. The same thing happens when "placing" a device, where all related degrees of freedom (i.e., x-, y-coordinates of absolute position, orientation and, implicitly, all relations to other elements as well) are eliminated with one single mouse click.

The underlying reason for the recursions mentioned above is given by this "design-step-like" behavior of today's layout editors, which only allows combined handling of the degrees of freedom resulting in their *implicit* elimination. Thus a designer is permanently forced to make implicit decisions concerning the degrees of freedom without having the appropriate information at the decision time. This tends to lead to an unavoidable "trial and error mode" resulting in the said recursions.

Despite this deficiency, this aspect of today's layout editors is accepted by the analog-designer community as this behavior feels "natural" and everyone has got accustomed to it. We would like to raise awareness to this "blind spot" and outline a proposal for a new *continuous layout design flow* that addresses this issue.

5.1.2 Direct Access to Degrees of Design Freedom

One solution to this problem would be to sequentially remove only those degrees of design freedom that are fully defined at the current design stage. This would require that functions like place and route are decoupled from their respective *fixed* degrees of freedom such that these degrees can be accessed directly and thus managed independently. Hence, they are now eliminated *continuously* during the layout process, but, each one only when it is necessary and appropriate according to its "definition status". This intrinsic capability of the flow is performed until we reach the physical mask layout which contains no further degree of design freedom. In such a *continuous layout design flow* the layout would be generated first in an almost symbolic manner before getting more and more detailed with actual physical parameters until it finally "crystallizes" to a real physical design.

For example, a net is laid out like this: First the net routing region is assigned, afterwards the preferred routing layer is determined, and at a later stage, when the current flows are known, the appropriate wire widths are assigned to their associated net sections.

5.1.3 Constraint Handling and Re-Use Capabilities

A continuous design flow would also deliver real benefits for constraint handling, and new ways of constraint recognition. With the direct access to the degrees of freedom, a detailed representation of the dependency of the layout-specific degrees of freedom on the functional constraints becomes possible. Hence, constraint verification and, thus, constraint-driven design (Sec. 2) become viable options.

The re-use of previous layout solutions in current designs is a well-known problem. Some reasons are that (1) the design is too application-specific, (2) even small changes to the circuit may require large changes in layout, (3) a new technology node is used, and (4) the shape of a layout module does not fit. However, careful consideration reveals that the underlying reason is that the layout view does not encompass any remaining degrees of freedom.

This problem can be addressed in an elegant manner in a continuous design flow by re-using a layout at a *symbolic stage* defined as follows: A re-usable design may only contain design freedoms that do not impact constraints; hence, the remaining design freedoms are *unconstrained*. In turn, the absence of constrained degrees of freedom indicates that all constraints are met. In other words, all design decisions induced by fulfilling a constraint are maintained, which is in fact the (long-sought-after) re-use of the implemented expert design knowledge (Fig. 5).

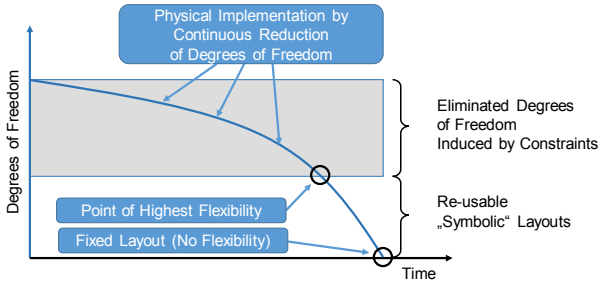

Figure 5. In the proposed design flow, the degrees of design freedom are continuously reduced. Re-using unfinished layout (i.e., symbolic level) supports the adjustment to new project-specific requirements because the symbolic level still contains degrees of design freedom needed for the adjustment.

The remaining degrees of freedom help modify the design for re-use to meet project-specific requirements, thus overcoming the problems mentioned above. The greater the number of remaining degrees of freedom, the higher the remaining flexibility, and thus, the higher the "re-usability". And the more the design problem to be solved resembles the re-use candidate, the fewer the remaining degrees of freedom that are needed for modifications and the less work is involved. This is a major advantage over current re-use methodologies which lack this ability to modify a design to suit a particular project.

5.2 A "Bottom-Up Meets Top-Down" Layout Design Flow

5.2.1 The Dilemma of Top-Down Automation

Analog layout automation has been studied and investigated intensely by EDA for over 30 years now, and almost exclusively in an attempt to solve the analog layout problem with techniques similar to those successfully deployed in the digital domain. These optimization-based "top-down" approaches require an abstraction of the design problem as a formalized mathematical model to which optimizing algorithms can be easily applied. Despite occasional successes, an industry-wide breakthrough of optimization-based approaches in the analog domain has not emerged so far.

This is due to the fact that the solutions generated by optimizers (see Fig. 3) suffer from either a weakness of the optimizing engines (if the modeling is at a low level of abstraction, which is done in an attempt to mirror physical reality as closely as possible) or from the weakness of the modeling approach (done to enable the use of efficient algorithms).

Fig. 6 illustrates this dilemma. The efficiency of optimizing algorithms, often measured in speed and the ability to find a global optimum, is generally inversely proportional to the accuracy of the underlying mathematical model which is derived from the physical world. Designs of high qualitative complexity demand high modeling accuracy while those of high quantitative complexity require high algorithmic efficiency. Hence, only design problems located below the curve can be satisfactorily solved by optimizers.

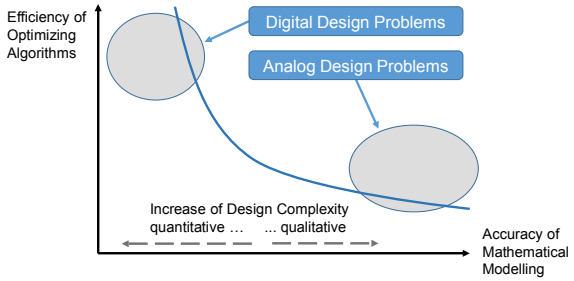

Figure 6. Illustration of the efficiency of optimizing algorithms, which is generally inversely proportional to the accuracy of the underlying mathematical model (blue curve). Analog designs of high qualitative complexity demand high modeling accuracy. Digital designs that are usually of high quantitative complexity require high algorithmic efficiency. Only design problems located below the curve can be satisfactorily solved by optimizers – which thus excludes most analog design problems.

This observation leads to the conclusion that top-down automation alone cannot solve the analog layout problem in its entirety.

5.2.2 Automated Delegation of Design Tasks to Procedures

The obvious approach to overcome the deficiencies in top-down optimization strategies is to complement them with appropriate bottom-up procedures (e.g., PCells). This is based on our observation that bottom-up procedures have the potential to provide the missing features in optimization-based approaches. A common issue with top-down automation is that algorithmically "invented" layout solutions are rejected by expert designers because they do not meet their expectations. Thus, designers prefer to re-use existing, silicon-proof design solutions which usually incorporate years of design knowledge, both from a human expert's personal experience and a company's design group portfolio.

In that regard, a particular strength of bottom-up automation is its intrinsic ability to augment the re-use of singular design solutions (i.e., "copy-paste") to a more sophisticated manner of re-using design solution strategies. This is why novel techniques are needed that enable circuit and layout designers to efficiently translate their design strategies into new automatic procedures. Only then will we see real progress with bottom-up procedures. The closer these techniques match the designer's way of thinking and the better they are adapted to his/her work style, the easier the techniques will capture the valuable expert knowledge, skills and creativity that are mainly absent from mere top-down automatisms. We believe that these techniques must be more than just novel description languages or tool wizards: They should be "schematic-like" for a circuit designer and "layout-editor-like" for a physical designer.

As mentioned above, analog design automation is severely handicapped by the qualitative complexity of analog constraints (expert knowledge). By restricting top-down optimization to "strategic constraints", such as high level design requirements, and by delegating the remaining constraints to bottom-up procedures, this problem could be eliminated. The ability of bottom-up procedures to make use of implicitly integrated expert knowledge is an ideal supplement to optimization approaches. In this regard the optimizers can be regarded as "senior tools", which delegate special tasks to their "subordinate" procedural tools.

An important step in this direction is the development of context-aware generators and of parameterized module generators as introduced in Secs 3.2 and 3.3. Context-aware generators, in particular, have the potential to play a central role, because their "sensing ability" brings a new kind of intelligence to procedural generators, which is notably helpful when taking over tasks from (senior) tools rather than from human designers.

5.2.3 Bridging the Gaps

Despite the widely held assumption that procedural automation is just a matter of handcraft and thus of little interest to academia, we are convinced that developing the techniques mentioned above is an academically appealing and practically profitable challenge for future EDA research. Conflating the resulting bottom-up procedures with existing top-down automation may be the key to finally achieving the full analog synthesis flow that has been pursued for over three decades now.

To make this vision a reality, we need at least two kinds of "bridges": Firstly, sophisticated techniques need to be developed that enable human design experts to capture easily their design know-how in bottom-up automation procedures. Secondly, technical concepts are needed that intelligently combine the different automation paradigms of optimization-based (top-down) and procedural (bottom-up) approaches.

6. SUMMARY AND OUTLOOK

In this paper we presented an overview of the analog layout design problem and the state of the art of analog layout automation with respect to active and future research areas. Despite enormous research effort in analog design automation, little progress has been made towards a fully automated design flow. We discussed some of the reasons for this, for example, the lack of uniform representation of design constraints in the analog design flow context. Thus, most of the constraints in today's analog designs are still specified and considered manually by expert designers (expert knowledge). Furthermore, analog constraints are often used implicitly (i. e., based on a designer's experience) rather than being explicitly defined – thereby preventing their effective use in design automation.

We also identified key factors relating to the next generation of analog design automation. Among them are techniques that reduce the degree of design freedom gradually rather than abruptly by providing direct and independent access to each degree of design freedom (*continuous layout design flow*). Another vision is to exploit the full potential of both bottom-up and top-down design styles. Conflating both styles to one *bottom-up meets top-down*

design flow should enable us to incorporate the aforementioned expert knowledge while also addressing high-level design requirements.

While breaking with conventional design approaches, these two proposed paradigm changes could lead to a new class of (higher level) design techniques that brings us one step closer to the goal of full-scale analog design automation.

ACKNOWLEDGEMENTS

We would like to thank Daniel Marolt, Andreas Krinke, Vinko Marolt and Göran Jerke for the numerous fruitful discussions related to the topic of this paper.

REFERENCES

[1] H. Chang, E. Charbon, U. Choudhury, et al. *A Top-down, Constraint-driven Design Methodology for Analog Integrated Circuits*, Springer Verlag, Norwell, MA (1997) ISBN: 978-0792397946.

[2] E. Malavasi, E. Charbon, E. Felt, A. Sangiovanni-Vincentelli, "Automation of IC layout with analog constraints," *IEEE Trans. CAD of Integr. Circuits and Systems*, 15, 8 (1996) 923–941. DOI= http://dx.doi.org/10.1109/43.511572

[3] R. Rutenbar, J. Cohn, "Layout tools for analog ICs and mixed-signal SoCs: a survey," *Proc. Int. Symp. on Physical Design* (ISPD) (2000) 76–83. DOI= http://dx.doi.org/10.1145/332357.332378

[4] J. Scheible, "Constraint-driven Design – Eine Wegskizze zum Designflow der nächsten Generation," *Proc. of ANALOG '08*, 2008, VDE Verlag, Berlin, Offenbach (2008) ISBN 978-3800730834.

[5] J. Freuer, G. Jerke, J. Gerlach, W. Nebel, "On the verification of high-order constraint compliance in IC design," *Proc. Design, Automation and Test in Europe* (DATE) (2008) 26–31. DOI= http://dx.doi.org/10.1109/DATE.2008.4484655

[6] G. Jerke, J. Lienig, J. B. Freuer, "Constraint-driven design methodology: A path to analog design automation," *Analog Layout Synthesis — A Survey of Topological Approaches* H. Graeb (ed.) Springer Verlag, New York, ISBN 978-1-4419-6931-6, (2011) 271-299. DOI= http://dx.doi.org/10.1007/978-1-4419-6932-3_7

[7] G. Jerke, J. Lienig, "Constraint-driven design — the next step towards analog design automation," *Proc. of the Int. Symp. on Physical Design* (ISPD) (2009) 75–82. DOI= http://dx.doi.org/10.1145/1514932.1514952

[8] J. Crossley, et al., "BAG: A designer-oriented integrated framework for the development of AMS circuit generators," *Proc. IEEE/ACM Conference on Computer-Aided Design (ICCAD)* (2013) 74-81. DOI= http://dx.doi.org/10.1109/ICCAD.2013.6691100

[9] D. Marolt, J. Scheible, G. Jerke, "PCDS: A new approach for the development of circuit generators in analog IC design," *Proc. 22nd Austrian Workshop on Microelectronics (Austrochip)* (2014). DOI= http://dx.doi.org/10.1109/Austrochip.2014.6946310

[10] V. Meyer zu Bexten, M. Tristl, G. Jerke, H. Marquardt, D. Medhat, "Physical verification flow for hierarchical analog IC design constraints," *Proc. Asian and South Pacific Design Automation Conference (ASPDAC)* (2015).

[11] IPGen 1Stone, *Internet:* http://ipgenme.de/

[12] G. Jerke, T. Burdick, P. Herth et al. "Visual PCell programming with Cadence PCell Designer," *Proc. CDNLive! EMEA 2013*, Munich, 2013.

[13] G. Jerke, T. Burdick, P. Herth et al. "Hierarchical module design with Cadence PCell Designer," *Proc. CDNLive! EMEA 2015*, Munich, 2015.

[14] R. Arora, A. Ginetti, R. Bishop, G. Lamant, S. Gangwar, "Virtuoso Express Pcells for better interoperability and performance on OA," *Proc. CDNLive! India 2007.* Internet: http://www.cadence.com/rl/Resources/conference_papers/4.5_presentationIndia.pdf

[15] Synopsys, "PyCell Studio," Internet: http://www.synopsys.com/cgi-bin/pycellstudio/req1.cgi

[16] D. Marolt, J. Scheible, G. Jerke, "A practical layout module pcell concept for analog IC design," *Proc. CDNLive! EMEA 2013*, Munich, Germany, paper nr. CUS01. Internet: http://www.cadence.com/cdnlive/eu/2013/pages/proceedingssummary.aspx

[17] P. Bhushan, R. Mitra, "Schematic Pcell implementation in Virtuoso platform," *Proc. of International Cadence Users Group Conference*, Santa Clara, 2004.

[18] A. Krinke, M. Mittag, G. Jerke, J. Lienig, "Extended constraint management for analog and mixed-signal IC design," *Proc. of the 21th European Conf. on Circuit Theory and Design* (ECCTD) (2013) 1-4. DOI= http://dx.doi.org/10.1109/ECCTD.2013.6662319

[19] A. Nassaj, J. Lienig, G. Jerke, "A new methodology for constraint-driven layout design of analog circuits," *Proc. of the 16th IEEE Int. Conference on Electronics, Circuits and Systems* (ICECS) (2009) 996-999. DOI= http://dx.doi.org/10.1109/ICECS.2009.5410838

[20] K. Krishnamoorthy, S. C. Maruvada, F. Balasa, "Topological placement with multiple symmetry groups of devices for analog layout design," *IEEE Int. Symp. Circuits and Systems*. (2007) 2032–2035. DOI= http://dx.doi.org/10.1109/ISCAS.2007.378437

[21] P.-C. Pan, H.M. Chen, Y-K. Cheng, J. Liu, W. Yi Hu, "Configurable analog routing methodology via technology and design constraint unification," *IEEE/ACM Int. Conf. Comput.-Aided Design*, (2012) 620–626.

[22] J. Lienig, "Electromigration and its impact on physical design in future technologies," *Proc. of the ACM 2013 Int. Symposium on Physical Design* (ISPD'13), Stateline, Nevada, (2013) 33-40. DOI= http://dx.doi.org/10.1145/2451916.2451925

[23] R. Rutenbar, "Design automation for analog: The next generation of tool changes," *1st IBM Academic Conf. on Analog Design, Technology, Modelling and Tools* (2006).

[24] A. Krinke, G. Jerke, J. Lienig, "Adaptive data model for efficient constraint handling in AMS IC design," *Proc. of the 20th IEEE Int. Conf. on Electronics, Circuits and Systems* (ICECS), (2013) 285-288. DOI= http://dx.doi.org/10.1109/ICECS.2013.6815410

[25] M. Mittag, A. Krinke, G. Jerke, W. Rosenstiel, "Hierarchical propagation of geometric constraints for full-custom physical design of ICs," *Proc. of Design, Autom. & Test in Europe Conf. (DATE)* (2012) 1471-1474. DOI= http://dx.doi.org/10.1109/DATE.2012.6176599

[26] C. Katzschke, M.-P. Sohn, M. Olbrich, V. Meyer zu Bexten, M. Tristl, E. Barke, "Application of mission profiles to enable cross-domain constraint-driven design," *Proc. of Design, Autom. & Test in Europe Conf. (DATE),* (2014) 1-6. DOI= http://dx.doi.org/10.7873/DATE.2014.079

[27] D. Marolt, J. Scheible, "The Application of layout module generators upon circuit structure recognition," *Proc. CDNLive! EMEA 2011*, Munich, Germany, paper nr. AC13. Internet: http://www.cadence.com/cdnlive/eu/2011/pages/proceedings.aspx