SWARM: A Self-Organization Approach for Layout Automation in Analog IC Design

Daniel Marolt and Jürgen Scheible
Robert Bosch Center for Power Electronics, Reutlingen University, Reutlingen, Germany
Email: {daniel.marolt, juergen.scheible}@reutlingen-university.de

Göran Jerke and Vinko Marolt
Automotive Electronics, Robert Bosch GmbH, Reutlingen, Germany
Email: {goeran.jerke, vinko.marolt}@de.bosch.com

***Abstract*—Optimization-based analog layout automation does not yet find evident acceptance in the industry due to the complexity of the design problem. This paper presents a Self-organized Wiring and Arrangement of Responsive Modules (SWARM), able to consider crucial design constraints both implicitly and explicitly. The flexibility of algorithmic methods and the expert knowledge captured in PCells combine into a flow of supervised module interaction. This novel approach targets the creation of constraint-compliant layout blocks which fit into a specified zone. Provoking a synergetic self-organization, even optimal layout solutions can emerge from the interaction. Various examples depict the power of that new concept and the potential for future developments.**

***Index Terms*—analog IC design, layout automation, constraint engineering, methodology, algorithm, particle system**

I. Introduction

Digital IC design has become highly automated thanks to optimization algorithms. In contrast, analog design is still done in a laborious manual fashion today, supported by comparably simple parameterized cells, as an adoption of algorithmic approaches in the analog domain does not find evident industrial acceptance so far. This is mainly rooted in the difficulty to account for all relevant functional design constraints, whose qualitative complexity in analog systems is known as "More than Moore". Constraints represent increasingly critical pieces of design knowledge and can basically exist either *formalized* or *non-formalized*. In principal, formalized constraints can be considered by an automatism *explicitly*, whereas non-formalized constraints can only be considered *implicitly*.

Optimization algorithms are designed to consider constraints explicitly and in a very generic fashion because they can self-intelligently *find* a solution for a particular design problem. While this flexibility makes them suitable for a wide range of use-cases, each application requires that *all* respective design constraints are formally described. Such automation approaches, which demand a complete mathematical formalization of the entire design problem, are also referred to as *top-down automation* [1].

In contrast to the flexibility of top-down automation, Parameterized Cells (PCells) are procedural generators dedicated to a very specific design purpose. At runtime, a PCell merely *reproduces* a customizable layout "result", while the cognitive layout "solution" is conceived by the PCell programmer in advance. This automation approach may seem rather ordinary, but it allows PCells to capture invaluable expert knowledge in an *informal* manner and thus to handle even most complex design constraints *implicitly*. However, an explicit consideration in PCells is barely possible with adequate flexibility because all eventualities need to be anticipated in advance. So, opposite to optimization algorithms, PCells follow a fundamentally different paradigm also called *bottom-up automation* [1].

In practice, due to the nature of analog layout design, constraints always appear formalized *and* non-formalized such that top-down and bottom-up automation approaches *alone* cannot tackle the problem in its entirety. While academia strongly focuses on the top-down paradigm and industrial flows rather pursue bottom-up automation, an effective conjunction of both paradigms is considered the key for closing the automation gap in the analog domain.

This paper presents a new layout automation approach named *Self-organized Wiring and Arrangement of Responsive Modules* (SWARM), which combines top-down and bottom-up automatisms into a novel flow of coordinated PCell *interaction* based on the concept of cellular automata. While each participating PCell autonomously exploits its individual bottom-up functionalities, a supervising control mechanism steers their aggregate activity towards a common goal from a top-down perspective, thereby allowing the interaction process to consider crucial design constraints both implicitly *and* explicitly.

SWARM is meant to be used by layout designers for the creation of block layouts which fit within a desired, sufficiently large, layout area. This is done by minimizing the available layout area until all PCells arrange

Manuscript received July 3, 2015; revised January 26, 2016.

doi: 10.18178/ijeee.4.5.374-385

themselves within that user-specified zone. The *arrangement* covers the alignment of the block's sub-circuits (as in floorplanning) and the detailed placement of the devices inside each sub-circuit. While the decentralized decision-making breaks down the complexity of the overall problem, it has the potential to show *emergent* behavior by resulting in a synergetic flow of self-organization. This phenomenon is deliberately evoked in SWARM for the arrangement. The term *wiring* denotes the internal routing of each sub-circuit which is autonomously performed and adjusted by the PCells in the course of the arrangement.

In various aspects, as will be elaborated in Section V, SWARM differs from classical optimization approaches, including swarm-intelligent methods [2] and the widely used analog placement algorithm Simulated Annealing [3]. Most notably, SWARM is absolutely deterministic.

This paper is structured as follows: Section II gives a brief overview of various works related to the topic of the paper. Section III describes the theory of the SWARM approach, while Section IV illustrates several implemented examples. Section V discusses the differences between SWARM and common optimization algorithms. Finally, Section VI concludes with a summary and an outlook.

II. Related Work

A. Optimization Algorithms

Layout design is an optimization problem which can, in principle, be solved with optimization algorithms. Optimization algorithms translate the design problem into an abstract model and search for an optimal solution regarding specific aims and constraints. When applied to analog layout, suchlike approaches usually focus on one specific design step (particularly placement, as in [4] and [5], and routing, as in [6] and [7]) and its respective difficulties. Relatively few works, such as [8] and [9], deal with the challenge of handling these two heavily correlated tasks simultaneously.

While there are various different placement algorithms such as Min-Cut and Force-Directed Placement, the most widely used one is Simulated Annealing [3], for example utilized in [4], [9]-[11]. Evolutionary approaches such as genetic algorithms have as well been applied to analog placement [12], and furthermore, [13] also investigates the use of neural networks. Some works, that disapprove the stochastic behavior found in most placement algorithms, decidedly pursue deterministic solutions [8] and [14].

Analog routing is, unlike for digital ICs, preferably not split into global routing and detailed routing, but done in one single step called area routing. The first area routers worked sequentially, such as Lee's maze router [15] and Hightower's line router [16]. Non-sequential area routers process the nets not in a truly parallel but quasi-parallel fashion. They can be divided into hierarchical approaches (e.g., [17]) or rip-up-and-reroute techniques (e.g., [18]).

Today, active research is done on methods of swarm intelligence [2], which -like evolutionary approaches- belong to the class of population-based searches. In the context of IC design, Particle Swarm Optimization and Ant Colony Optimization have already been used for the sizing, modeling and testing of analog circuits [19] and [20].

Since optimization algorithms require a mathematical description of the design problem, much effort is put into formal representations of design data. Amongst others, particular attention lies on abstract floorplan models such as TCG-S [4], sequence pairs [5], [9] and [10] and more enhanced representations [11]. Particular research effort is spent on the formal, explicit consideration of constraints. E.g., for placement many algorithms take device symmetry [5] and [11] and device proximity [10] and [14] into account. For routing, the constraints of interest include net shielding [6] and wiring symmetry [8]. Constraints must be algorithmically processible and they need to be available in the appropriate hierarchical context. Hence, substantial work deals with the generation [21], propagation [22], transformation [6], and unification [7] of constraints.

Despite these efforts, the need to completely formalize a design problem remains a huge obstacle for algorithmic approaches, because analog expert knowledge cannot be easily translated into computationally digestible expressions of high-level, abstract design requirements [1].

B. Parameterized Cells

In the industry, optimization algorithms still meet with skepticism and reluctance among analog layout designers. Instead, parameterized cells are the most frequented pieces of automation in practice. Basic device PCells are even indispensable for today's prevalent style of manual layout creation, despite their petty abilities to relieve layout engineers merely from laborious drawing work. In fact, an ongoing trend towards more powerful compound *module PCells*, as for example seen in [23] and [24], can be observed.

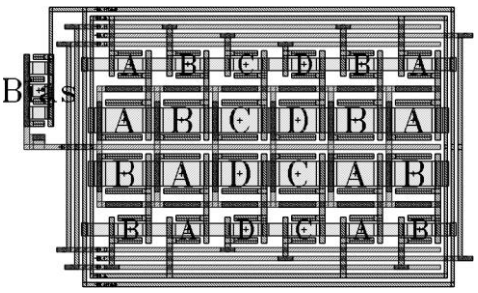

Figure 1. Example of a wide-swing current mirror PCell instance.

Being an incarnation of bottom-up automation, module PCells can create entire layouts for analog basic circuits in full-custom quality (Fig. 1). Where optimization algorithms work self-intelligently, a PCell leaves the creative part of the layout task completely to the human expert. PCells do not require a formalization of the design problem as they simply automate a designer's "best practice". This allows PCell developers to easily incorporate expert knowledge, creativity and intuition into the automatism in a non-formalized fashion. That

way, the resulting layout output is not only deterministic but even predictable.

An important mission for analog EDA, rather than the development of powerful PCells, is to provide intuitive tools such as [23] and [25] which allow layout experts to easily create their required PCells themselves. A central challenge for such tools is to match the designers' mentality as close as possible in order to capture their layout solution strategies in an informal way. A convincing example for this intent is the visual PCell programming approach taken by PCell Designer [23], which resembles the familiar design style of a human expert's daily layout work.

The advancement of powerful module PCells in the layout also sparks interest in corresponding circuit PCells on the schematic side. For example, recent works dealing with circuit PCells are BAG [24] and PCDS [26].

C. Artificial Life

From an abstract perspective, the PCell interaction in SWARM is a form of artificial life. The cellular automaton, introduced by Ulam and extended by von Neumann in the 1940s [27], can be considered the first artificial-life model. Cellular automata are a general concept for simulating discrete, dynamic systems and have applications in many fields of science [28]. A famous example of a two-dimensional cellular automaton is Conway's Game of Life, devised in 1970 [29]. It implements a grid of binary cells each of which performs transitions according to a set of simple rules, thereby considering only the cell itself and its neighbors. On this basis, interesting patterns arise after several generations, including still lifes, oscillators, "spaceships" (as in Fig. 2) and self-replicating constructs.

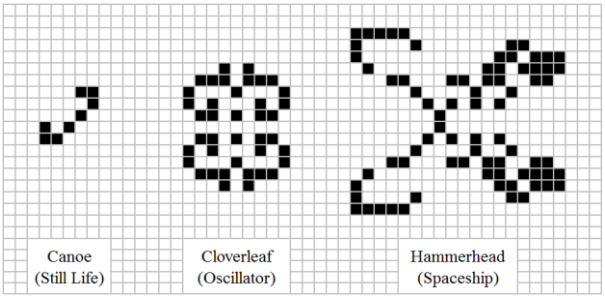

Figure 2. Examples of stable patterns in Conway's "Game of Life".

Agent-based modeling [30] is a concept specifically designed to simulate the interactions of autonomous entities in order to examine the effect of their collective behavior on the overall system. In 1986, Reynolds presented Boids [31], the first computer program to simulate the flocking behavior of birds as agents. Implementing a distributed behavioral model, the aggregate motion of the entire flock results from the interaction of the individual Boids, where each Boid's behavior follows very simple steering rules in the context of its local environment.

The evolvement of new structures from the interactions of simpler sub-units is called *emergence* [32] and is subject to many disciplines including biology, philosophy, sociology, physics, and systems theory. Emergence is closely related to *self-organization*, however the phenomena are not the same. Early principles of self-organizing systems were formulated by the cyberneticians Ashby in 1947 [33] and von Foerster in 1960 [34]. In nature, self-organization occurs in the swarming behavior of insects living in colonies, such as ants, bees and termites.

III. The SWARM Approach

SWARM is composed of three correlated concepts that cover the necessities of a layout block from its bottom-level devices up to its top-level requirements (see Fig. 3). At the lower levels, here subsumed as the *participant level*, PCells are used to implement *governing modules* (A) which perform dedicated design tasks for basic circuit parts. Above, at the so-called *surface level*, these modules enter a self-organized *module interaction* (B) which runs in a recursive cycle between the modules' activities and an *interaction control* (C) mechanism at block level. Subsequently, the theory behind these three concepts (A), (B), (C) will be discussed in detail.

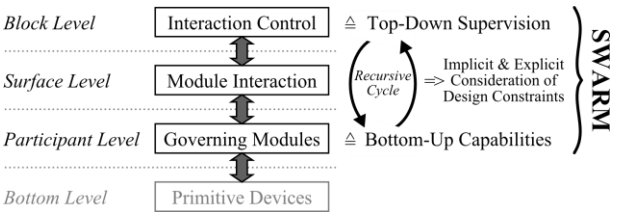

Figure 3. The three correlated concepts of the SWARM approach.

A. Governing Modules

A governing module is a small automation entity that performs one or several dedicated steps to create a specific analog basic circuit from a given set of layout devices. Of these steps, the two most fundamental ones are to position the devices and to wire them. Governing modules are suited for basic circuits with highly regular structures (e.g., identical devices in a matrix arrangement) for which optimal layouts, including the *wiring*, are already known from practical experience. Such circuits are often found on the lowest design levels, covering prime analog functions such as current mirrors and differential pairs. Fig. 4, aspect (a) shows an example of governing modules used to build a current mirror from four given transistors.

For pure positioning, a governing module is a meta-entity: it moves each device to a dedicated position without creating any physically relevant layout shapes. To connect the devices, a governing module does create wiring shapes in the layout design. Usually, the positioning strongly depends on the wiring, in which case it is suitable to perform both tasks with one single governing module. If multiple modules are used as in Fig. 4(a), one of them must be implemented as *supreme commander* over all the others and has to maintain the consistency of this entire, so-called *module association* (Fig. 4(b)). Further details are left to the module developer, but normally the module performing the primary design task is to be a supreme commander. If

only one module is used (no module association), it is by definition a supreme commander.

In analog design, a layout component –be it a primitive device or a compound entity– can have different variants, covering various aspect ratios without altering its electrical behavior. To consider this, a governing module must, in addition to the constructive tasks mentioned above, be able to tell and provide the *variability* Φ (the set of all *n* feasible layout variants V_1, V_2 …V_n) for the circuit it implements. A *variator module* as shown in Fig. 4(b) is a meta-entity with the sole purpose to provide the variability of a single, primitive device. For a transistor, the variability Φ is usually given by the possible number of fingers. The figure also shows, that SWARM allows governing modules to be hierarchically imposed onto each other. In that case, the total *cumulative variability* $\Phi_{c,m}$ of a module is the product of its own *intrinsic variability* $\Phi_{i,m}$ and the cumulative variability $\Phi_{c,s}$ of its sub-modules: $\Phi_{c,m} = \Phi_{i,m} \times \Phi_{c,s}$. This aspect is depicted in Fig. 4(c).

Every variant evaluation has to adjust its device positioning as well as its wiring and is, like the module interaction (Section III.B), led by the supreme commanders.

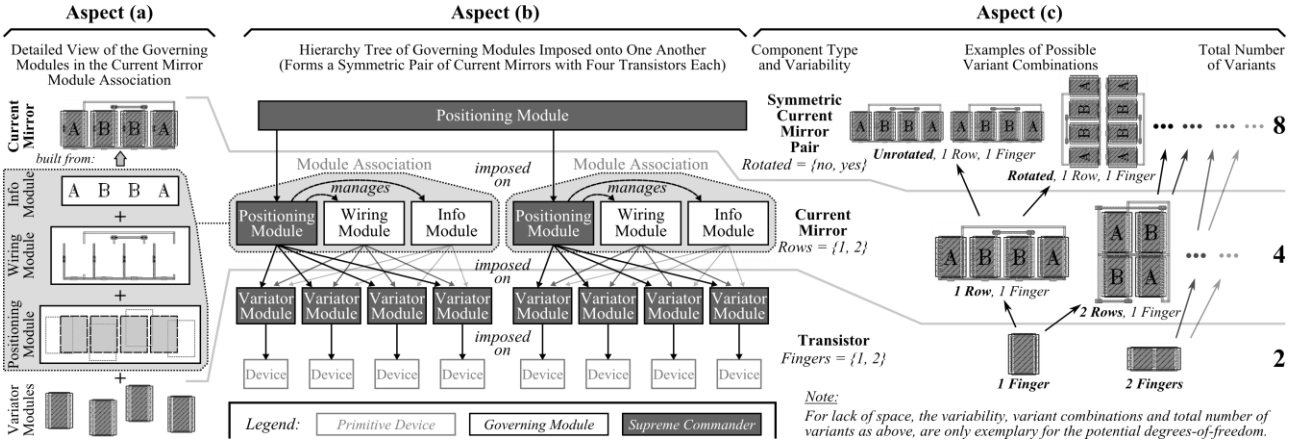

Figure 4. Governing modules in SWARM. Aspect (a) shows how an association of governing modules is used to build a current mirror from four given transistors. Aspect (b) illustrates how governing modules can be hierarchically imposed onto each other to create higher-level modules. Aspect (c) displays the combinatorial increase of layout variability resulting from a hierarchical imposition of governing modules.

B. Module Interaction

To form the desired layout block, the governing modules are to be arranged in a constellation that fits within a *user-specified zone* and explicitly satisfies all design constraints that are not yet covered implicitly by the modules themselves. At this surface level, irregular (non-matrix) constellations and arbitrary aspect ratios of the user-specified zone need to be served. The governing modules may provide sufficient variability to achieve that goal, but the enormous amount of possible constellations raises a combinatorial challenge. As a solution, the modules are impelled into a process of interaction, dedicated to letting them find a suitable *arrangement* on their own. During that process, the layout variants of the modules may be changed by the supreme commanders in order to suit the arrangement, thereby adjusting their internal *wiring*.

The module interaction in SWARM obeys a distinct principle already mentioned in Section II relating to [29] and [31]: each participating module (subsequently denoted as a *participant*) repeatedly chooses its own course *itself*, always based on very simple rules and just a local assessment of its current situation. For that purpose, the modules' abilities as described in Section III.A, which can be considered their *introversive* behavior, are extended by an *extroversive* behavior, which defines each participant's reaction to changes of its environment.

The extroversive behavior is not necessarily identical among all participants, but it always abides by a common scheme comprised of the following four measures:

- *Assessing the participant's condition,*
- *Perceiving its "free peripheral space",*
- *Exploring and evaluating possible actions,*
- *Executing the preferred action (or staying idle).*

In the remainder of this section, each of the four measures above will be covered in a subsection of its own. To better grasp their role during the interaction flow, the reader is advised to see Section III.C (especially Algorithm I).

1) Assessment of the participant's condition

The condition of a participant *P* decides whether there is a need to take action or not. Action is provoked when the condition sustains negative influence, which can be exerted by three different factors, subsequently referred to as *protrusion*, *interference* and *noncompliance*:

Protrusion is the case if the participant does not completely lie within the *user-specified zone*. Depending on the grade of protrusion, the participant is denoted as *lost* (entirely outside), *prone* (partially outside) or *safe* (entirely inside). The grade of protrusion effects the action that will be taken, as will be explained in subsections 3 and 4.

Interference is when the participant *P* overlaps one or multiple other participants called *P**. If there is no interference among them, the participant is said to be *clear*.

An *overlap* ω of *P* and *P** is their intersection area multiplied with the area of *P**. The overlap ω causes a *trouble* $\tau = \alpha_i + \omega$ where α_i is the *aversion* of *P* towards *P**. In the beginning of the interaction there is no

aversion among the participants, but an overlap ω of *P* and *P** increases the aversion of *P* towards *P** so the new aversion is $\alpha_{i+1} = (\alpha_i + \omega) \cdot (\gamma_i + 1)$ where γ_i is the previous number of *clashes* between *P* and *P**, which is then incremented to $\gamma_{i+1} = \gamma_i + 1$ due to the overlap. If there is no overlap, the aversion drops to $\alpha_{i+1} = \alpha_i \cdot \varphi$ where φ is a conciliation factor chosen within the interval [0, 1].

Every overlap also causes a *wound* on participant *P*. A wound *W* is a pair (ρ, ς) which has a specific *region* ρ on *P* and a discrete *severity* ς. A new wound *W* has ς = 1 but if it overlies an existing wound *W**, then the old severity ς_o of the intersection $\rho_W \cap \rho_{W*}$ is incremented to the new value $\varsigma_n = \varsigma_o + 1$. When ς_n exceeds a critical level ς_c the participant begins a strategy of *recuperation* and chooses his subsequent actions such that the wound fully *heals*. A wound, as long as it isn't aggravated by a new overlying wound, heals by decrementing its severity until ς is 0.

Aversion has a long-term effect in that it hinders a perpetual interference of two participants, whereas a critical wound has an immediate impact. Furthermore, as aversion correlates with the area of an overlap, wounds are more effective for preventing marginal interferences. A careful balance in the modeling of aversions and wounds is one key to a fluent progress of interaction in SWARM.

Noncompliance means that the participant, in its current position, violates at least one explicit constraint. Further details cannot be discussed here, because they depend on the constraints actually being used in the design.

If the participant is not affected by any of the above factors, then it is *contented*, else it is *discontented*. A participant strives for action if it is discontented. When contented, no action is required, but nonetheless, the participant attempts to perform a move in this case (if possible): to center itself within its *free peripheral space* (see below). If that is not possible, the participant lingers where it is. Although the centering does not ameliorate the participant's condition, it can be regarded as an improvement of its situation, imagining that the participant "feels" best when it can equalize the distances to all its neighbors.

2) Perception of the free peripheral space

The basis for a participant's action is the vacant area surrounding it (and, for actions involving other participants, also the vacant areas around *them*). While there are various possible conceptions of how this so-called *free peripheral space* could be specified, we provide the following unambiguous definition:

Given the user-specified zone *Z* as a rectilinear polygon, let *P* be a rectangular participant located (at least partially) inside *Z*, and *B* the rectangular bounding box around the part of *P* that is completely inside *Z*. For every edge *E* of *B* with length *e*, let the corridor C_E be a rectangle beginning at *E* and stretching away from *P* to infinity with a width of *e*. Let *C* be the four corridors of *P* and let *O* be a set of obstacles containing the inverse of *Z* as well as all participants except *P*. Then, the free peripheral space S_{FP} of *P* is uniquely defined as the largest possible rectangle around *P* not containing any intersection $O \cap C$.

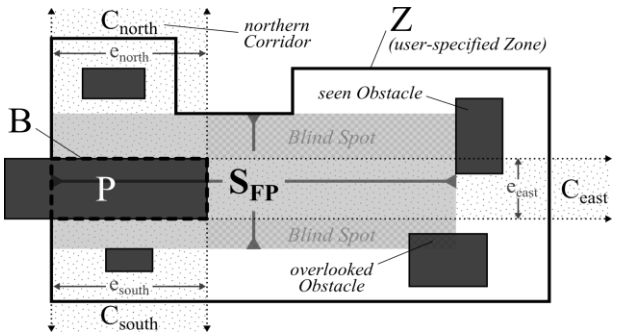

Figure 5. The free peripheral space S_{FP} as perceived by a participant P. Obstacles in "blind spots" of S_{FP} are accidentally overlooked.

Fig. 5 visualizes this definition and points out the fact that S_{FP} can contain up to four *blind spots* comprised of $S_{FP} \backslash C$, which are those parts of S_{FP} not contained in any corridor C_E. Obstacles in these blind spots are overlooked when *S* is determined. However, this is not critical since eventual collisions will be detected before *P* performs an action. Furthermore, in the course of the tightening of *Z*, the arrangement of all participants becomes increasingly compact and the area of blind spots approaches zero.

3) Exploration and evaluation of possible actions

Every action that a participant can perform is basically a set of moves for all involved participants, where each move $M = (T, R, D)$ consists of a translation *T*, a rotation *R* and a deformation *D*. The translation is a 2-dimensional vector $T = (\Delta x, \Delta y)$ that displaces the participant without altering its aspect ratio. The rotation $R \in \{-90°, 0°, 90°\}$ rotates the participant by an angle of 90 degrees around its center (thereby inverting its aspect ratio) or preserves its orientation. The deformation $D \in \{V_1, V_2, \ldots, V_n\}$ is one of the *n* layout variants the participant can assume (as given by its cumulative variability Φ_c). *D* always changes the participant's aspect ratio to that of the assumed layout variant (which always includes the internal *wiring* that is thereby adjusted) but does not dislocate its center point.

Although each participant may implement its own individual action strategy, SWARM facilitates a fundamental set of possible actions considered adequate for any participant *P* as a kind of natural, "instinctive" behavior. The actions can be divided into the following nine types, which are all illustrated in Fig. 6:

- **Re-Entering** (M_R): is performed when *P* is *lost*. It has the sole aim of catapulting *P* back into the user-specified zone *Z* at the nearest possible location and disregards both *interference* and *noncompliance* (Section III.B 1).
- **Centering** (M_C): denotes that *P* aligns its center point with the center point of its free peripheral space S_{FP}.
- **Lingering** (M_L): occurs if *P* is *contented* but cannot perform a centering (e.g., due to *interference* with an obstacle in a *blind spot*). In that case, *P* just does nothing.
- **Budging** (M_B): makes *P* probe additional room (by spotting the free peripheral spaces from the

viewpoints of each of its corners) and then tries to slip into an appropriate position within that "extended" peripheral space.

- **Pairing** (M_P): means that *P* jumps next to another participant *P** (and may even push *P** aside) such that both participants then share the free peripheral space of *P**.
- **Swapping** (M_S): lets *P* trade places with another participant *P**. That is, *P* jumps into the free peripheral space of *P** and *P** jumps into the free peripheral space of *P*.
- **Hustling** (M_H): is an action, where *P* remains as it is, but pushes away all other interfering participants as much as necessary to get rid of them and become *clear* again.
- **Evasion** (M_E): is only applicable when *P* is *prone*. Here, *P* is driven into the user-specified area by its boundary, trying to evade other participants by moving sideways.
- **Yielding** (M_Y): is only done in case *P* has interference *I* but no appropriate action can be found. Then, *P* determines the polygon $S_Y = P \cap Z \setminus I$ (the part of *P* inside *Z* excluding the interfering areas *I*) and aligns its center with the geometric centroid (i.e., the barycenter) of S_Y.

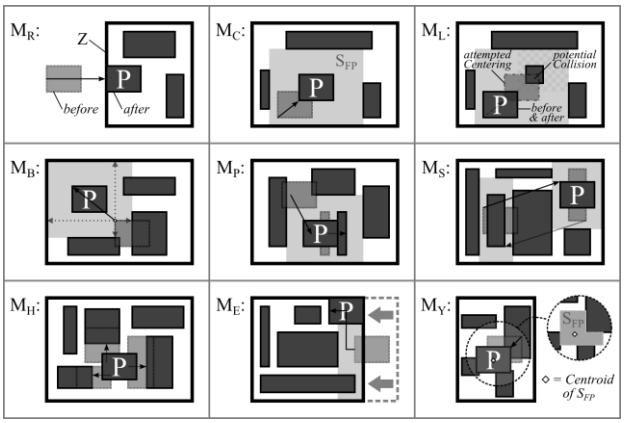

Figure 6. The nine "instinctive" actions facilitated by SWARM.

4) Execution of the preferred action

An action is discarded if it would lead an involved participant into a state of *protrusion* or *noncompliance* or if it would aggravate a wound that is currently subject to full recuperation. In contrast, *interference* can be tolerated and is the main factor that decides which action will be taken by a participant *P* if there are multiple alternatives.

If *P* is *lost*, it performs a re-entering move (M_R). In case *P* is *contented*, it performs a centering (M_C) if possible, otherwise it lingers (M_L). If *P* is *discontented* but not *lost*, then it explores the actions M_C, M_B, M_P, M_S, M_H (and, if *P* is *prone*, also M_E). For each explored action, *P* evaluates the prospective *interference*, i.e., *P* calculates the value $\tau_{i+1} = \tau_{p0} + \tau_{p1} + \ldots + \tau_{pn}$ which is the sum of the prospective *troubles* for the *n* participants involved in the action. If $\tau_{i+1} = 0$, then the action will be immediately executed without exploring other alternatives. If $\tau_{i+1} > 0$ for *all* explored actions, then the action with the least prospective *trouble* will be executed, but only if $\tau_{i+1} < \tau_i$ where τ_i is the current sum of *troubles* for the *n* participants. Such an action is called *beneficial*.

If no *beneficial* action can be found, then *P* executes a yielding move (M_Y). Yielding leaves *P* in an unsatisfactory position but has the effect that, if *P* is stuck between multiple other participants *P**, *P* will then deliberately interfere with all of them. Thus, *every P** is provoked to move away such that *P* can again try to find a *beneficial* action on its next turn. As already said in subsection 3, a participant may also implement its own additional actions and neglect some or all of those described above.

C. Interaction Control

The interaction control supervises the module interaction from a top-down perspective and carefully steers it to enforce the emergence of a constraint-compliant, compact arrangement that fits within the user-specified zone *Z*.

In the current conception of SWARM, the control mechanism exerts influence on the interacting modules only *indirectly*, by minimizing the available layout area. This recursive tightening represents a change of the environment and merely "motivates" a participant's action, which can in turn have a *latent* effect on subsequent actions too. Such an indirect coordination of individuals is also called *stigmergy* [35], a term closely related to emergence, self-organization and swarm intelligence.

Stigmergy can be observed among animals in nature. And incidentally, the SWARM approach of driving autonomous PCell modules towards a self-organized, compact arrangement is analogous to the herding of livestock. For example, a shepherd brings a group of sheep together just by encircling them, thereby leaving it up to every single animal to find its individual place among the herd.

The mechanism of SWARM and its interaction control is outlined in Algorithm I. Beginning with the primitive layout devices given by the schematic circuit, SWARM first of all instantiates *governing modules* on the components such that the basic circuits are fully layouted and provide full variability Φ_c. The initial constellation of the modules may be arbitrary, but can also be customized (e.g., according to a certain layout template). Next, the user-specified zone *Z* is centered on the bounding box of the constellation and then enlarged so that all participants can easily find a place inside *Z* without any interference. Then, the interaction starts and lets every participant perform an action (as in Section III.B). The actions in such a *round* are meant to emulate a concurrent behavior but are in fact executed sequentially. Multiple rounds of interaction are repeated until no action at all is performed within one round. This outcome is denoted as a *settlement* because each participant has settled at a definite position. If not all participants are contented, they failed in finding an appropriate arrangement. Otherwise, the constellation is said to be *viable* and *Z* is tightened (without excluding any participant) in order to stimulate another settlement. The tightening-settlement cycle is repeated until *Z*

reaches the user-specified size provided for the final layout. A completed cycle of settlements is called a *run*.

Algorithm I: Interaction Control over a SWARM Run

```
impose governing modules on the devices
center the zone Z on the initial constellation
enlarge Z such that all participants are within Z
REPEAT
  REPEAT
    FOR each participant P
      let P determine and execute an action
    ENDFOR
  UNTIL no actions have been performed
  IF not all participants are contented THEN
    abort (failure)
  ENDIF
  tighten Z such that all participants are within Z
UNTIL user-specified size of Z has been reached
end (success)
```

The composition of governing modules, module interaction and interaction control is a delicate issue. Many details that need to be considered cannot be fully covered in this paper and will be addressed only briefly below.

First of all, the order in which all the participants act always remains the same during a run, but it doesn't follow any specific metric. One might argue that the order can be of paramount importance for the convergence of a run. On the other hand, a solid and well-balanced implementation of the concept is supposed to eliminate the relevance of the order since emergent behavior is supposedly very robust with respect to such particular items. Nevertheless, this topic is subject to further research.

Another issue in this context is the tightening policy for the zone *Z*. SWARM tightens *Z* as much as possible but such that no participant has a protrusion greater than 50 % of its area. Other policies may be investigated in the future, including an adaptive and a reverse tightening.

A mandatory concern is the definition of a minimal-distance-threshold to prevent infinitesimal moves. This is not only for reasons of performance, but to prevent a group of contented participants from centering themselves ad infinitum. As our current realization shows, it is feasible to correlate the threshold with the amount of free space so it changes dynamically during a run.

Altogether, the challenge with SWARM's interaction approach is to implement a well-balanced module behavior (between tentative and vivid) and an adequate pacing policy for the zone tightening (between relaxed and aggressive). The ultimate goal is to make the overall interaction proceed as straightforward and robust as possible, letting it neither drift towards stagnation (i.e., system is too static) nor to chaos (i.e., system is too dynamic).

IV. Implementation and Examples

The SWARM approach has been implemented in the programming language SKILL for the Cadence Virtuoso design environment. Cadence PCell Designer [23] was used for the realization of the module PCells.

For the purpose of interaction, the module PCells require a *responsive* behavior which facilitates that they can react to environmental changes. However, it should be noted, that common PCells are conceptually not able to access their environment. To address this shortcoming, SWARM implements a special interface fabric to (1) feed a PCell with information *read* from its context and to (2) read information from a PCell in order to *modify* its context. In each SWARM run, the interface fabric is wrapped around the PCells to provide an artificial responsivity.

A. Synthetic Example of an Emerging Collective Motion

To illustrate the SWARM approach step by step, Fig. 7 shows a very simple example with two participants *P1* and *P2* inside a rectangular zone. For the purpose of exemplification, the given participants are only symbolic modules, without being imposed on real devices. Furthermore, rotations and deformations are omitted from the possible actions in this example, which means that the participants can only perform translational moves.

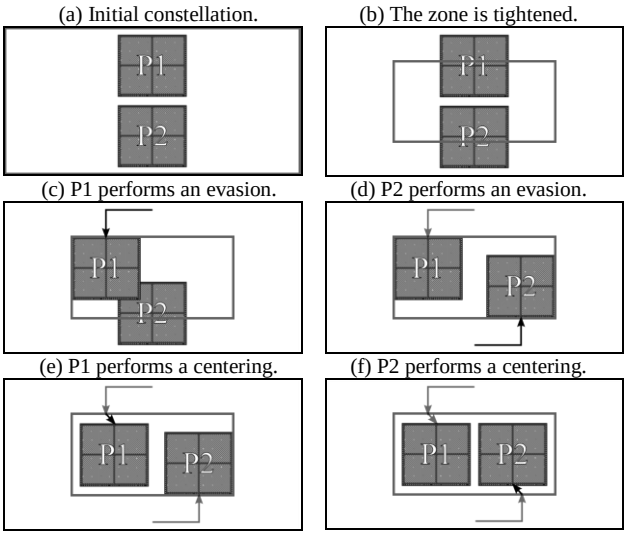

Figure 7. Steps of a simple example SWARM run. Performing basic moves, the two modules execute a pirouette motion from an above-below alignment towards a side-by-side alignment.

The various steps of the interaction are displayed in the subfigures (a) to (f). The first subfigure shows the initial constellation (a). Next, the zone is tightened (b) such that both participants become prone and need to act. *P1*, attempting to move back into the zone, performs an evasion by moving downward and to the left (c) where there is less overlap with *P2* than in the horizontal middle of the zone. Then, *P2* also executes an evasion, moving upward and to the right (d). Now, both participants are again safe and clear so first *P1* starts centering itself within its free peripheral space (e) and then *P2* also performs centering until both have settled in their final position (f).

A comparison of the initial constellation (a) with the final constellation (f) points out that *P1* and *P2* have been forced to re-arrange themselves from an above-below alignment to a side-by-side alignment. The two participants achieved this with two basic types of action: evasion and centering. From a higher-level perspective, their movements give the impression as if the participants

had performed a pirouette motion. This can be considered as an example of how a complex collective maneuver seems to emerge from more primitive moves of interaction.

B. Synthetic Example of an Emerging Optimal Solution

Fig. 8 shows another synthetic placement exercise with a rectangular zone. As in the previous example, the eight given participants are just symbolic modules and again, they only perform translational movements.

Subfigure (a) displays the initial constellation, in which all participants were placed at random positions and thus also overlap each other. After several interaction cycles, the participants have struggled themselves free from each other and settled at new locations were they are all contented (b). Next, the zone is recursively tightened as the participants settle a second time, a third time (c) and a fourth time (d). During these settlements, the overall constellation changes only slightly (i.e., the relative positions of the participants mostly remain the same). After the fourth zone tightening, multiple overlaps occur among the participants as they strive to get completely back into the zone (e). Subfigures (f) and (g) show some subsequent intermediate constellations to illustrate that the overlaps are diminished. One particular overlap, that cannot be resolved at once, wanders through the constellation as can be seen in the right-bottom corner of (f) and in the left-bottom quadrant of (g). Finally, this overlap also gets resolved and the participants settle a fifth time as shown in subfigure (h). The relative positions in that constellation represent the globally optimal solution, from which the example was initially constructed from. A last tightening of the zone leads to the sixth and final, absolutely compact settlement of the participants (i).

The presented example has also been executed with other, randomly created, initial constellations – always leading to a constellation that is the global optimum. It should further be noted, that the optimal solution here even is a non-slicing floorplan. This example demonstrates that via interaction the participants in SWARM are –basically– capable of finding even the *best* possible arrangement on their own account. Moreover, they don't even have a notion of their achievement since each participant's actions are based solely on its individual point of view, without a global conception of the overall problem. So, the optimal solution really *emerges* (quasi as a "byproduct") from the participant's coherent interaction.

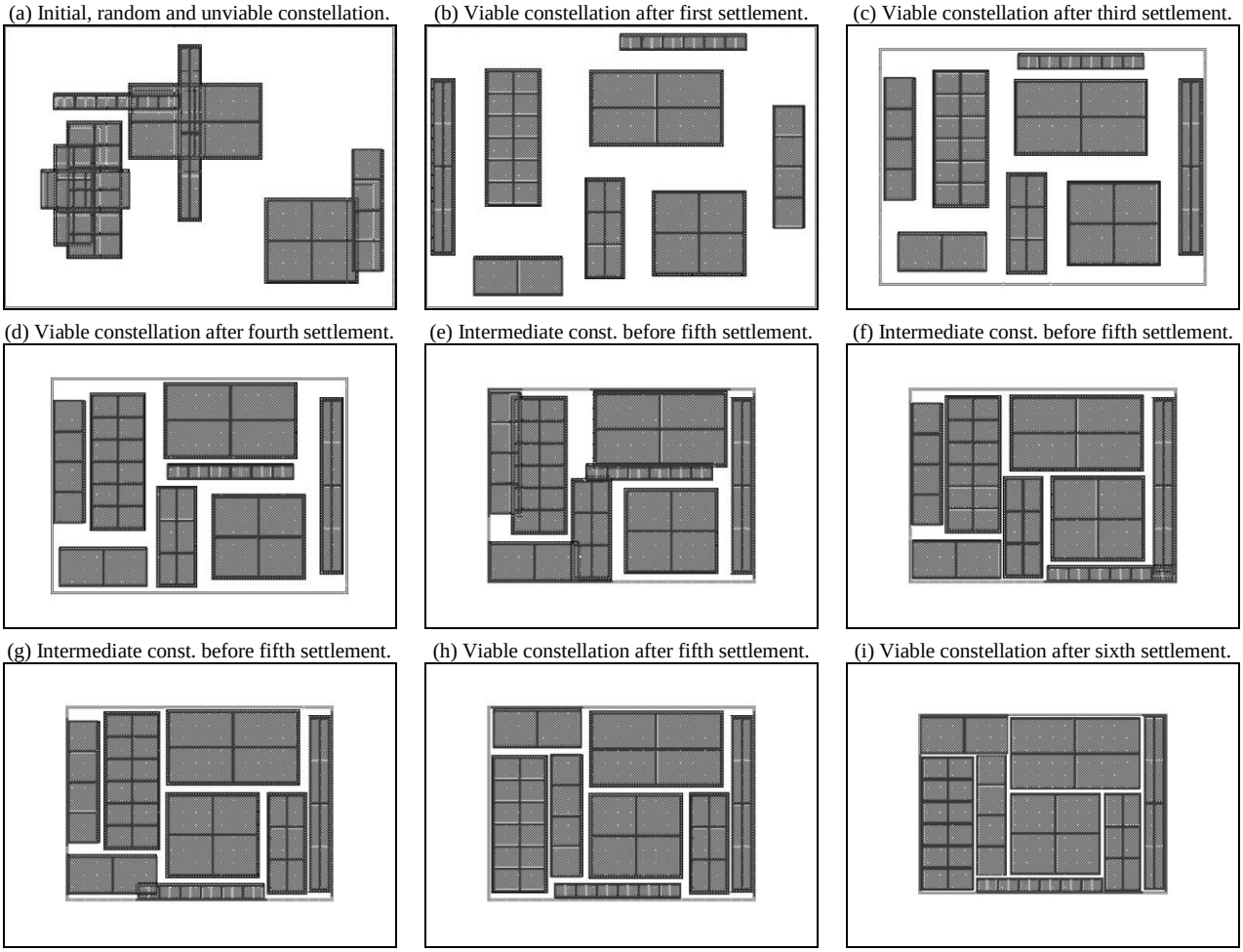

Figure 8. Constellations of an example SWARM run. Starting from a random placement (a), the modules manage to find the optimal arrangement (i).

C. Practical Example of a Symmetric P-Input OTA

As a practical example, Fig. 9 shows how SWARM is used to create different layouts for a symmetric p-input OTA (Operational Transconductance Amplifier).

Subfigure (a) presents the schematic diagram of the OTA circuit, made of one differential pair (1) and five current mirrors (2 to 6). In the layout, each of these six basic circuits is managed by an association of governing modules which enact a suitable device interdigitation, handle the detailed positioning, provide appropriate wiring, and display the respective interdigitation pattern (e.g., ABBA) as a label. The governing modules implicitly take care of the inherent design constraints, which include a close proximity of the devices, same orientation, and precise device alignment. For the current mirrors, the devices are allowed to be aligned within either one single row or spread over two rows. In contrast, the transistors of the differential pair obey a two-dimension common-centroid ("Quad") constraint, demanding that the devices are to be placed in a 2-by-2 cross-coupled AB/BA array.

Subfigures (b), (c) and (d) show three different layouts resulting from a template-based initial constellation via module interaction (including translations, rotations and deformations) for three different aspect ratios: 1:1 in (b), 1:2 in (c) and 3:1 in (d). As shown, the governing modules keep on implicitly satisfying their inherent design constraints in each resulting layout, regardless of the location, orientation and variant they have finally assumed.

In addition to the implicitly considered constraints, all modules have explicitly taken care of further, supplementary constraints during the interaction process. These constraints strictly specify the relative positions of the layout modules to ensure a vertically symmetric arrangement of the overall layout block. For that purpose, an additional dummy cap is inserted in the layout, located to the right of the central Quad. During the interaction, the cap always performs an individual *following* move (instead of the "instinctive" actions) in order to mimic the course and match the size of current mirror 6, located to the left of the Quad. This ensures that the sensitive differential pair is horizontally aligned with the center of the layout block.

The current mirrors 2 and 3 require a careful "twofold" symmetric placement, which is managed by a *symmetric pair module* (already presented in Fig. 4). Thus, in all three resulting layouts, the reference transistors (marked A) of the two current mirrors at the bottom of each layout are interdigitated within their module, but are also placed mutually symmetric with respect to the layout as a whole.

As intended, SWARM provides the *intra*-module wiring for each basic circuit part of the OTA and finds a constraint-compliant layout arrangement within a user-specified zone. However, the *inter*-module routing (i.e., the connections between the circuit parts) is still left to the designer. As Fig. 9 shows, SWARM can put artificial boundaries around each of the participating modules to preserve empty space which is then available for the inter-module routing. An inclusion of inter-module routing into the interaction is subject to further work. Not shown in the examples so far is the fact that the layout zone need not be rectangular but may be a (rectilinear) polygon.

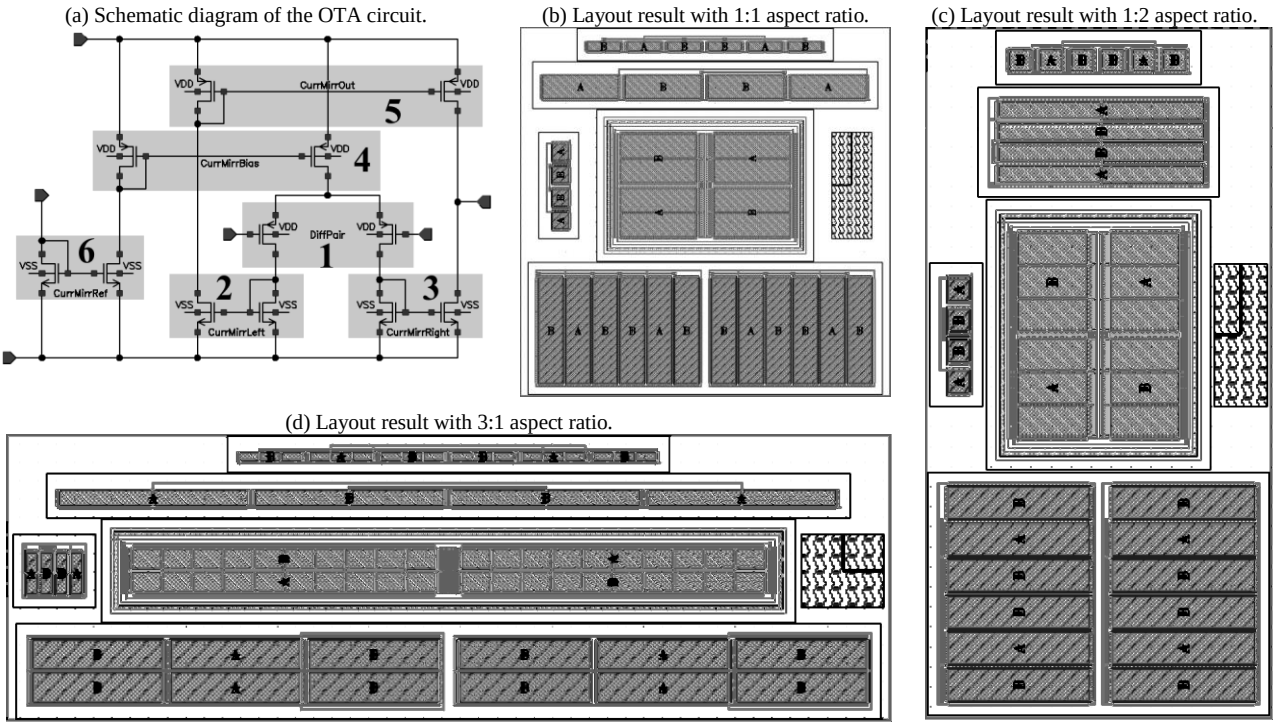

Figure 9. Example of applying SWARM to an OTA circuit. For the given schematic (a), three different layout results (b, c, d) were achieved with SWARM. Crucial design constraints are implicitly and explicitly taken into consideration during the interaction process. Amongst others, the relative positions of the participating modules are always the same among all three layouts, even though the aspect ratios are different.

V. Distinction from Optimization Algorithms

As will be treated in this section, SWARM is innovatively different from traditional optimization algorithms, including swarm intelligence and Simulated Annealing.

Methods of swarm-intelligence are algorithms which use an entire population of candidate solutions. These so-called "particles" collectively wander through the given search space to find an optimum in there. So, every particle is one potential solution to the given problem. In contrast, the SWARM approach focuses on a single solution: the layout, which is increasingly compacted via tightening. Here, the particles are the *participants* which make up that solution through interaction and self-organization.

Simulated Annealing (SA) represents a single-solution approach where the candidate solution is repeatedly modified via random perturbations and evaluated according to a general cost function. The perturbation mechanism is the same for all elements and there is only one single cost function to evaluate the layout solution in its entirety. This is substantially different in SWARM, where layout modifications are delegated to the participating modules. Every action is autonomously chosen by the individual participant, based on a personal assessment that disregards the action's effect on the fitness of the overall solution. This decisive aspect sets SWARM apart from all optimization algorithms that use a global cost metric. Compared to the primitive perturbations and probabilistic search technique used in SA and most other heuristic optimization algorithms, the pool of actions available in SWARM is considerably richer and the choice of action is far more intelligent. If necessary, a participant evaluates all possible actions in order to choose the most beneficial move. This effort per action may strain the performance more than in SA, but the total number of iterations in SWARM is smaller by several orders of magnitude. Further contrast with SA is given by the implementation of wounds and aversion in SWARM that can be regarded as a kind of memory. It makes SWARM's problem solving progress more stringent and less volatile than strategies of random perturbation in general. And unlike contemporary layout automation algorithms which mostly rely on stochastic optimization, the interaction approach in SWARM works totally deterministic and reproducible.

VI. Summary and Outlook

This paper presents a Self-organized Wiring and Arrangement of Responsive Modules (SWARM) for automatic creation of constraint-compliant block layouts that fit into a specified zone. In contrast to the sole top-down perspective of optimization algorithms, SWARM leaves critical design tasks to a coordinated PCell interaction.

A well-balanced conjunction of bottom-up automation and top-down supervision in SWARM allows constraints to be considered both implicitly and explicitly, which represents a coalescence of two fundamentally different paradigms. By provoking a synergetic self-organization, even optimal layout solutions can emerge from the PCell interaction, as demonstrated by several given examples.

Future work on SWARM targets (a) the already mentioned inter-module routing via interaction, (b) the realization of multiple supervisors that may further exert even direct influence and pursue more intelligent control strategies, (c) hierarchical SWARM runs allowing many parallel interaction flows with individual goals to proceed concurrently, and (d) the examination of efficiency, convergence and robustness criteria for the self-organization.

Acknowledgment

We wish to thank Andreas Gerlach for the OTA example (with circuit, template and constraints) as well as Peter Herth and Thomas Burdick for the support on PCells.

References

[1] J. Scheible and J. Lienig, "Automation of analog IC layout – Challenges and solutions," in *Proc. ACM Int. Symp. on Physical Design*, Monterey, CA, March 2015, pp. 33-40.

[2] J. Kennedy, R. Eberhart, and Y. Shi, *Swarm Intelligence*, San Fr., CA: Morgan Kaufmann, 2001.

[3] S. Kirkpatrick, C. Gelatt Jr., and M. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671-680, May 1983.

[4] J. Lin, G. Wu, Y. Chang, and J. Chuang, "Placement with symmetry constraints for analog layout design using TCG-S," in *Proc. Asia South Pac. Des. Aut. Conf.*, 2005, pp. 1135-1138.

[5] K. Krishnamoorthy, S. Maruvada, and F. Balasa, "Topological placement with multiple symmetry groups of devices for analog layout design," in *Proc. Int. Symp. of Circ. and Syst.*, 2007, pp. 2032-2035.

[6] Q. Gao, Y. Shen, Y. Cai, and H. Yao, "Analog circuit shielding routing algorithm based on net classification," in *Proc. ACM/IEEE Int. Symp. on Low-Power Electronics and Design*, 2010, pp. 123-128.

[7] P. Pan, H. Chen, Y. Cheng, J. Liu, and W. Hu, "Configurable analog routing methodology via technology and design constraint unification," in *Proc. Int. Conf. Comp.-Aided Design*, 2012, pp. 620-626.

[8] Y. Yang and I. Jiang, "Analog placement and global routing considering wiring symmetry," in *Proc. International Symposium on Quality Electronic Design*, 2010, pp. 618-623.

[9] L. Xiao, E. Young, X. He, and K. Pun, "Practical placement and routing techniques for analog circuit designs," in *Proc. IEEE/ACM Int. Conf. on Computer-Aided Design*, 2010, pp. 675-679.

[10] L. Zhang, S. Dong, Y. Ma, and X. Hong, "Multi-Stage analog placement with various constraints," in *Proc. Int. Conf. on Communications, Circuits and Systems*, 2010, pp. 881-885.

[11] P. Lin, Y. Chang, and S. Lin, "Analog placement based on symmetry-island formulation," *IEEE Trans. Computer-Aided Design Integr. Circuits and Systems*, vol. 28, no. 6, pp. 791-804, 2009.

[12] L. Zhang and U. Kleine, "A genetic approach to analog module placement with simulated annealing," in *Proc. IEEE International Symposium on Circuits and Systems*, 2002, pp. 345-348.

[13] M. Yu, "A study of the applicability of Hopfield decision neural nets to VLSI CAD," in *Proc. 26th Conf. on Design Automation*, 1989, pp. 412-417.

[14] M. Strasser, M. Eick, H. Gräb, U. Schlichtmann, and F. Johannes, "Deterministic analog circuit placement using hierarchically bounded enumeration and enhanced shape functions," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, 2008. pp. 306-313.

[15] C. Lee, "An algorithm for path connections and its applications," *IRE Trans. on Electr. Comput.*, vol. EC-10, no. 3, pp. 346-365, 1961.

[16] D. Hightower, "A solution to line-routing problems on the continuous plane," in *Proc. 6th Annual Design Automation Conf.*, 1969, pp. 1-24.
[17] M. Burstein and R. Pelavin, "Hierarchical wire routing," *Trans. Comp.-Aid. Des. Integr. Circ. and Syst.*, vol. 2, no. 4, pp. 223-234, 1983.
[18] W. Dees and P. Karger, "Automated rip-up and reroute techniques," in *Proc. 19th Conference on Design Automation*, 1982, pp. 432-439.
[19] P. Kumar and K. Duraiswamy, "An optimized device sizing of analog circuits using particle swarm optimization," *Journal of Computer Science*, vol. 8, no. 6, pp. 930-935, 2012.
[20] H. Gupta and B. Ghosh, "Analog circuit design using ant colony optimization," *Int. J. Electr., Comp. and Commun. Technol.*, vol. 2, no. 3, pp. 9-21, 2012.
[21] S. Bhattacharya, N. Jangkrajarng, and C. Shi, "Multilevel symmetry-constraint generation for retargeting large analog layouts," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 6, pp. 945-960, 2006.
[22] M. Mittag, A. Krinke, G. Jerke, and W. Rosenstiel, "Hierarchical propagation of geometric constraints for full-custom physical design of ICs," in *Proc. Design, Aut. & Test Eur.*, 2012, pp. 1471-1474.
[23] G. Jerke, *et al.*, "Hierarchical module design with cadence PCell designer," in *Proc. Pres. CDNLive! EMEA*, Munich, 2015.
[24] J. Crossley, *et al.*, "BAG: A designer-oriented integrated framework for the development of AMS circuit generators," in *Proc. IEEE/ACM International Conference on Computer-Aided Design*, 2013, pp. 74-81.
[25] A. Graupner, R. Jancke, and R. Wittmann, "Generator based approach for analog circuit and layout design and optimization," in *Proc. Design, Automation & Test in Europe Conference*, 2011, pp. 1-6.
[26] D. Marolt, M. Greif, J. Scheible, and G. Jerke, "PCDS: A new approach for the development of circuit generators in analog IC design," in *Proc. 22nd Austrochip Workshop*, 2014, pp. 1-6.
[27] J. V. Neumann, "The general and logical theory of automata," in *Cerebral Mechanisms in Behavior – The Hixon Symposium*, New York: John Wiley & Sons, 1951, pp. 1-31.
[28] S. Wolfram, *A New Kind of Science*, Wolfram Media, Inc., May 2002, pp. 1-1197.
[29] M. Gardner, "Mathematical games – The fantastic combinations of John Conway's new solitaire game 'life'," *Scientific American*, vol. 223, pp. 120-123, Oct. 1970.
[30] M. Niazi and A. Hussain, "Agent-Based computing from multi-agent systems to agent-based models: A visual survey," *Scientometrics*, vol. 89, no. 2, pp. 479-499, 2011.
[31] C. Reynolds, "Flocks, herds, and schools: A distributed behavioral model," in *Proc. 14th Annual Conference on Computer Graphics and Interactive Techniques*, 1987, pp. 25-34.
[32] G. Lewes, *Problems of Life and Mind (First Series)*, London: Trübner, 1875, vol. 2, p. 412.
[33] W. Ashby, "Principles of the self-organizing dynamic system," *Journal of General Psychology*, vol. 37, no. 2, pp. 125-128, 1947.
[34] H. V. Foerster, "On self-organizing systems and their environments," in *Self-Organizing Systems*, M. Yovits and S. Cameron, Eds., London: Pergamon Press, 1960, pp. 31-50.
[35] L. Marsh and C. Onof, "Stigmergic epistemology, stigmergic cognition," *Cognitive Systems Research*, pp. 1-15, 2007.

Daniel Marolt studied the interdisciplinary degree program mechatronics at Reutlingen University in Reutlingen, Germany, where he received the B.Eng. degree in 2008 and the M.Sc. degree in 2009. Currently, he is pursuing the Ph.D. degree in electrical engineering in the field of analog IC design automation at the Robert Bosch Center for Power Electronics in Reutlingen, which is a research and teaching network established in 2009 by the Bosch Group, Reutlingen University and the University of Stuttgart with support from the South-West German state of Baden-Württemberg.
Accompanying his studies from 2004 to 2009, he was with the IC layout design department of the Robert Bosch GmbH in Reutlingen as a Working Student. In 2010 he joined Reutlingen University to work for one year on a government-funded research project in cooperation with Bosch in Reutlingen. Since 2011, he is with the Robert Bosch Center for Power Electronics as an Academic Employee, where he continues his cooperative research work with Bosch. In regard to his dissertation topic, his current research interests are focused on the advancement of parameterized cells for the automation of full-custom analog circuit and layout design.

Jürgen Scheible received the Dipl.-Ing. (diploma) degree in electrical engineering (1987) and the Dr.-Ing. (Ph.D.) degree in electrical engineering (1991) both from the University of Karlsruhe, Germany (today: Karlsruhe Institute of Technology – KIT).
From 1987 to 1992, he was a Research Assistant with the Institute for Theory in Electrical Engineering (ITM) of the University of Karlsruhe. From 1992 to 2010 he was with the automotive electronics division of the Robert Bosch GmbH. Amongst other positions there he worked as Senior Engineer in ASIC design, as Project Manager in hybrid design flow, as Director for EDA tool-management and as Head of Department for IC layout design. Since 2010 he is Full Professor for electronic design automation with the Robert Bosch Center for Power Electronics at Reutlingen University, Germany. His research interests include the automation of analog IC design with a special emphasis on physical design and methodologies for electro-thermal simulation. In an invited talk on ISPD 2015, he presented his visions of new methodologies for analog IC layout automation, which are summarized in [1].
Prof. Scheible is senior member of IEEE, member of ACM's Special Interest Group for Design Automation (SIGDA), and member of the MPC (Multi Project Chip) group and BW-CAR, two academic communities for applied research in the South-West German state of Baden-Württemberg. He was recipient of the Young Scientist Award of Südwestmetall (Employers Association of the South-West German Metal and Electrical Industry) in 1992. Prof. Scheible acted as peer reviewer for the German symposiums ANALOG (CAE-methods for analog design) and ZuE (reliability and design of electronic circuits).

Göran Jerke received his Dipl.-Ing. (diploma) degree in electrical engineering from the Dresden University of Technology in 2000.
In 2000, he joined the automotive electronics division of the Robert Bosch GmbH in Reutlingen, Germany, to work as System Architect on the research and development of EDA tools for IC design.
In his current position as Senior Project Manager in EDA Research and Advance Development (since 2010), he is responsible for the research, the advance development and the adoption of constraint-driven design methods. He authored and co-authored several academic and industry papers on electro-migration avoidance in IC designs, constraint-driven design, and various other EDA topics. His research interests currently include design flow concepts in general, as well as methods for automated and interactive physical design implementation, verification and robustness validation of IC designs.
Mr. Jerke was recipient of the EdaCentrum EDA Achievement Award in 2004. The prize was awarded to Mr. Jerke for his works on the analysis and verification of current density in conductive paths.

Vinko Marolt accomplished a dual vocational training at the vocational college and the Robert Bosch GmbH in Reutlingen, Germany, between 1973 and 1976. Until 1979, he remained with the Robert Bosch GmbH in Reutlingen to work in a developmental laboratory for automotive electronic hybrid applications. In 1979, he left Bosch to augment his knowledge in a technical school in Reutlingen and finished in 1981 with the degree of state-certified technician.

In 1981, he returned to the Robert Bosch GmbH, Reutlingen, in order to join the automotive electronics division, where he has since then been working as a Development Engineer in the department for IC layout design. Throughout this time, he has also been formative tool expert for the then-upcoming design environments such as Silvar-Lisco's Princess, Mentor's IC Station and Cadence Virtuoso. His research interests include the automation of analog layout design with special dedication to parameterized cells. Further research interests are focused on systems theory, cybernetics and philosophy.

Until today, Mr. Marolt has filed a total of five patents in the field of microelectronics, four of which are hardware patents and one of which is a software patent.