SWARM: A Multi-agent System for Layout Automation in Analog Integrated Circuit Design

Daniel Marolt, Jürgen Scheible, Göran Jerke and Vinko Marolt

Abstract Despite 30 years of Electronic Design Automation, analog IC layouts are still handcrafted in a laborious fashion today due to the complex challenge of considering all relevant design constraints. This paper presents *Self-organized Wiring and Arrangement of Responsive Modules* (SWARM), a novel approach addressing the problem with a multi-agent system: autonomous layout modules interact with each other to evoke the emergence of overall compact arrangements that fit within a given layout zone. SWARM's unique advantage over conventional optimization-based and procedural approaches is its ability to consider crucial design constraints both explicitly and implicitly. Several given examples show that by inducing a synergistic flow of self-organization, remarkable layout results can emerge from SWARM's decentralized decision-making model.

Keywords Integrated circuits · Electronic design automation · Analog layout · Constraint-driven design · Parameterized cells · Self-organization

1 Introduction

Microelectronic products are increasingly controlling, connecting, and changing our world, and today, market demands for low-cost, multifunctional, and densely integrated circuits (ICs) drive the trend to systems-on-chip with both analog and digital

D. Marolt (✉) · J. Scheible
Robert Bosch Center for Power Electronics,
Reutlingen University, Reutlingen, Germany
e-mail: daniel.marolt@reutlingen-university.de

J. Scheible
e-mail: juergen.scheible@reutlingen-university.de

G. Jerke · V. Marolt
Robert Bosch GmbH, Automotive Electronics, Reutlingen, Germany
e-mail: goeran.jerke@de.bosch.com

V. Marolt
e-mail: vinko.marolt@de.bosch.com

G. Jezic et al. (eds.), *Agent and Multi-Agent Systems: Technology and Applications*, Smart Innovation, Systems and Technologies 58,
DOI 10.1007/978-3-319-39883-9_2

content. But while the task of *digital* IC design follows highly automated synthesis flows based on optimization algorithms, Electronic Design Automation (EDA) is still struggling to apply such approaches in the *analog* domain. For over three decades, such attempts have repeatedly failed to find industrial acceptance and thus, most design steps are manually accomplished by expert designers with very little support by automation. In particular, the step of *layout design*, where a circuit schematic has to be turned into a graphical description of the detailed circuit geometries—needed for the photolithographical manufacturing of the IC—remains a laborious and severely time-consuming bottleneck in the design flow.

This paper presents a novel layout automation approach for analog IC design: *Self-organized Wiring and Arrangement of Responsive Modules* (SWARM) [1]. To our knowledge, SWARM is the first approach to address the design problem by implementing a multi-agent system, in which the individual agents are layout modules that interact with each other. Steered by a supervising control organ, the self-interested modules autonomously move, rotate and deform themselves inside an increasingly tightened layout zone, vying for the available space. By inducing a synergistic flow of self-organization, the decentralized decision-making of the modules is supposed to provoke the *emergence* of compact layout arrangements.

The presented approach is especially interesting in two aspects. First, it joins ideas from several different disciplines such as cybernetics, game theory, biology, geometry, and electrical engineering. Second, SWARM addresses a serious practical problem which could not yet be satisfactorily solved with layout algorithms equivalent to those that are successfully used for digital circuit designs—despite a substantial amount of EDA work. In contrast to these achievements, SWARM's decisive asset is that it facilitates an explicit *and* implicit consideration of crucial *design constraints*, as both are essential for achieving the degree of layout quality that is uncompromisingly demanded in the analog domain. Since SWARM is an interdisciplinary and quite extensive approach, this paper focuses on providing (1) an elementary introduction to the problem of IC layout design, (2) a presentation of the SWARM approach cast in the light of multi-agent systems, and (3) a couple of examples demonstrating what results can be achieved with SWARM.

The paper is organized as follows. Section 2 gives a basic introduction to the problem of IC layout design. Section 3 discusses existing works on analog layout automation, discerning algorithmic and generator approaches. Then, Sect. 4 covers the multi-agent system of SWARM from a high-level point of view, not going into all details. Section 5 exemplifies the application of SWARM and shows some achieved results. Section 6 finishes the paper with a summary and an outlook.

2 The Problem of Integrated Circuit Layout Design

As shown in Fig. 1, any IC layout design problem expects three inputs: (1) a set of *design rules*, inherently given by the semiconductor technology, (2) a structural description of the circuit, i.e., a *schematic* diagram or a *netlist*, and (3) a set of

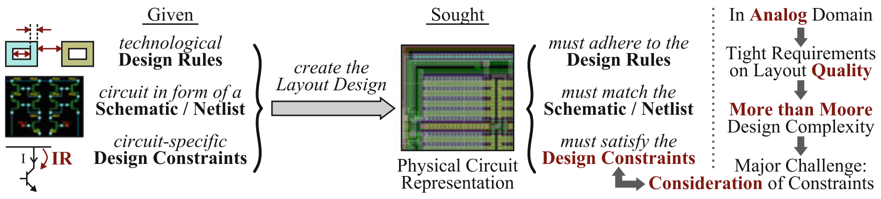

Fig. 1 The problem of IC layout design: turn a circuit into a physical representation

circuit-specific *design constraints*. Sought is a physical representation describing the detailed chip geometries of the circuit's devices and their interconnections on all *process layers*, used as photolithography masks in the manufacturing process. To ensure manufacturability and functionality, such a layout must (1) adhere to the design rules, (2) match the given circuit, and (3) satisfy all design constraints.

In terms of problem complexity, one has to discern *digital* from *analog* design. In digital design, with its primary objective to cram more and more components (nowadays millions and billions of logic gates) onto a chip—a desire referred to as *More Moore* [2]—the design problem is mainly a matter of *quantity*. In contrast, the major difficulty in the continuous-valued analog domain is to maintain signal integrity in the face of nonlinearities, parasitic effects, thermal gradients, high voltages, external physical influences and other *More than Moore* [2] challenges. This makes analog design complexity rather an issue of *quality*, and thus, obtaining a functional layout involves many, diverse, and correlated design constraints.

Analog layout design includes several tasks. As detailed in Table 1, the main tasks are: *floorplanning* (to specify positions, aspects ratios, and pin positions for the top-level layout blocks of a chip), *placement* (to set the position, orientation, and layout variant of all electronic devices, e.g., transistors, resistors, capacitors, inside a layout block), and *routing* (creating electrical wires, using so-called *vias* to connect wires across different metal layers). Overall, layout design is an *optimization problem* defined by restrictions (e.g., minimal wire widths and spaces, given by the design rules) and objectives (usually: reducing area and wirelength). For each specific layout problem, function-relevant restrictions and objectives are captured by the respective design constraints. This set of constraints (and thus, a definition of the optimization problem itself) in turn depends on a multitude of attributes, including the chosen semiconductor technology as well as the type, application, mission profile, and particular reliability requirements of the circuit.

While some design constraints relate to high-level aspects, e.g., the available layout space, most constraints deal with low-level details, in particular to achieve electrical symmetry for layout modules that perform critical electrical functions. This so-called *matching* usually requires equal sizes and consistent orientations of the module devices in a compact, interdigitated, common-centroid placement [3]. Such restrictions reduce the degrees of freedom, but—as shown by the examples in Fig. 2—they typically still leave much layout *variability*. This is a characteristic trait of analog design which often has to be exploited to satisfy all requirements.

Table 1 The main tasks in analog layout design: floorplanning, placement and routing

	(a) Floorplanning	(b) Placement	(c) Routing
Considered components	Circuit blocks (with sub-hierarchy)	Basic devices (atomic primitives)	Wire segments + Vias (to cross metal layers)
Quantities to be set by the design task	Block positions	Device positions	Wire paths, segment
	Aspect ratios	Device orientations	Layers and widths +
	Pin positions	Layout variants	Via positions and sizes
Typical restrictions	Layout boundary	Layout boundary	Available metal layers
	Maximum distances	Space for routing	No wires above devices
Primary objectives	Minimize total area	Device matching	Minimize number of vias
	Minimize wirelength	Overall symmetry	Homogenize wire density

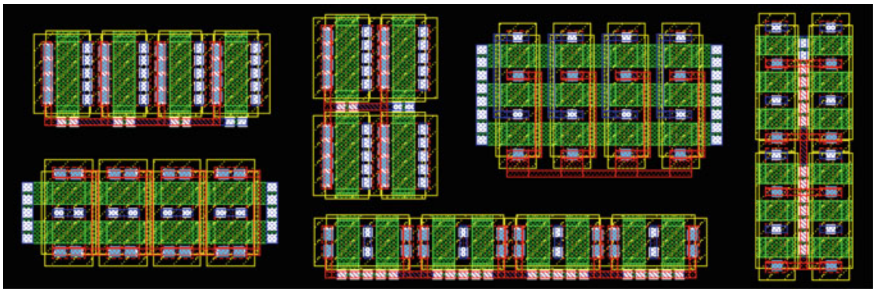

Fig. 2 Different variants of an analog layout module with equal electrical function

In today's flows of manual layout design, two basic forms of constraint consideration are found: high-level constraints usually need to be *explicitly* formulated to be really taken into account, while low-level matching requirements are often *implicitly* taken care of by experienced layout engineers. But as will be explained in Sect. 3, so far no automatism supports *both* forms of constraint consideration.

3 State of the Art in Analog Layout Automation

Although EDA research has put forth a rich variety of analog layout automation approaches over the past decades, these can be divided into two fundamental categories [4]: optimization algorithms (Fig. 3a, Sect. 3.1) and procedural generators (Fig. 3b, Sect. 3.2). They follow entirely different automation strategies, while SWARM (Fig. 3c, Sect. 4) is an attempt to conflate the two paradigms.

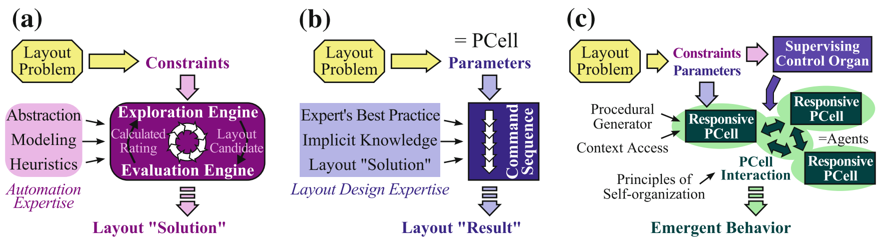

Fig. 3 Layout automation strategies: **a** algorithmic, **b** procedural, **c** SWARM

3.1 Optimization-Based Layout Automation Algorithms

Referring to [5], the use of optimization algorithms in EDA largely concentrates on a canonical form depicted in Fig. 3a, where a single candidate layout is repeatedly refined in a loop of solution space *exploration* and solution *evaluation*. For example, the most widely used algorithm for analog *placement* is Simulated Annealing [6] where exploration is done via a random modification of the current candidate placement, while every evaluation rates the new placement according to a formal cost function and validates if it satisfies all design restrictions. As an example for *routing*, the exploration-evaluation style of optimization is particularly apparent in techniques that *rip-up and reroute* existing wires as in [7].

For layout automation, a particular strength of such approaches is their ability to take into account formal expressions of high-level design constraints. However, optimization algorithms work by translating the optimization problem into an abstract mathematical model, in some cases based on a physical analogy (in the case of SA: the annealing of solids), and specifically optimize just the modeled aspects. This implies, that *all* relevant solution requirements and optimization goals *must* be *explicitly* expressed in a formal, comprehensive, unambiguous, and consistent representation of constraints that can be processed by the algorithm.

In analog layout however, it is tremendously intricate and still not possible today to *efficiently* and *sufficiently* describe the entire diversity, various impacts and correlated dependencies of *all* crucial design constraints in an *explicit* fashion due to the *More than Moore* complexity (see Sect. 2). The inherent quality loss is the major reason for the rejection of optimization-based automation in practice. Another shortcoming is that most of these approaches are not deterministic, thus yielding nonreproducible solutions. Furthermore, algorithmic layout automation typically concentrates on just one particular design step (e.g., placement *or* routing), although—in practice—these steps are heavily interrelated with each other.

In spite—or because—of these issues, analog EDA research continues to work on a vast range of topics in the context of optimization algorithms. These include (a) the consideration of different constraint types such as boundary, fixed-outline, or symmetry-island [8] constraints, (b) topological floorplan representations like SKB-trees [9], (c) performance improvements as in [10], (d) the simultaneous execution

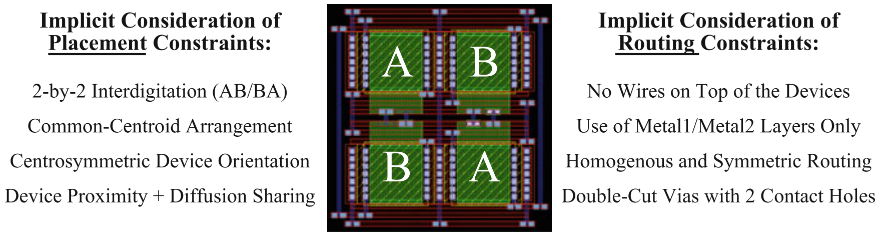
Implicit Consideration of Placement Constraints:
2-by-2 Interdigitation (AB/BA)
Common-Centroid Arrangement
Centrosymmetric Device Orientation
Device Proximity + Diffusion Sharing
A
B
B
A
Implicit Consideration of Routing Constraints:
No Wires on Top of the Devices
Use of Metal1/Metal2 Layers Only
Homogenous and Symmetric Routing
Double-Cut Vias with 2 Contact Holes

Fig. 4 PCell instance and implicitly considered placement and routing constraints

of different design tasks [11], and (e) constraint propagation [12]. Please note that the given references are not comprehensive, but merely representative. Approaches beyond the canonical form of Fig. 3a will be covered in Sect. 3.3.

3.2 *Procedural Generators (Parameterized Cells, PCells)*

The predominantly manual design style in practice employs an entirely different kind of automation at the lowest design levels: procedural generators, also called *Parameterized Cells* (PCells). A PCell is an instantiable library component: instantiated in a design, it takes a set of parameter values and follows a predefined, successive sequence of commands to produce a customized layout result inside. Primarily, PCells automate the layout creation of *primitive devices* (e.g., transistors), but recently, industrial flows can be observed to pursue an advancement of PCells towards more powerful *hierarchical modules* which generate entire analog basic circuits (e.g., current mirrors). For such circuits, feasible layout variants are often already known from practical experience, and PCells simply represent instruments that embody this invaluable *expert knowledge* as layout generators. For example, Fig. 4 shows a “Quad” PCell instance that creates a *differential pair* layout for two transistors A and B, split into a cross-coupled AB/BA array.

In academia, PCells have been of little interest as they merely replicate a human expert’s best practice: they generate a layout *result*, but the actual *solution* was preconceived by the design expert who implemented the PCell (Fig. 3b). However, this trait gives module PCells the advantageous ability to consider design constraints *implicitly*, i.e., without the need to formalize them. For example, as pointed out in Fig. 4, the Quad module innately takes care of all requirements crucial to achieve a high matching, without having been *explicitly* told to do so.

In contrast to the weaknesses of optimization algorithms mentioned before, PCells elude the need for abstract models and formal floorplan representations, work fast and deterministic, and can naturally do both placement and routing (as in Fig. 4). These assets are based on the fact that every PCell targets one specific type of device (or module), but the immanent downside is the effort required to implement a PCell for each desired type, and the inability to consider constraints explicitly. Due to this tradeoff, the use of PCells in industrial environments has not yet shown to be profitable enough above the level of analog basic circuits.

Amongst other topics, PCell-related work deals with (a) implementing higher-level layout modules as in [13], (b) developing advanced tools for PCell programming [14], (c) circuit PCells [15], and (d) auxiliary concepts such as hierarchical parameter value editing [16]. Again, the given references are only representative.

3.3 Other Approaches

After 30 years of EDA, research on layout automation still keeps a strong focus on the single-candidate style of optimization in Fig. 3a. However, population-based approaches using a pool of candidate solutions have also been applied, including genetic algorithms (e.g., for floorplan area optimization in [10]) and methods of swarm intelligence (such as the hybrid ant colony and particle swarm optimization algorithm in [17]). Still, existing approaches are not as sophisticated as optimization techniques found for example in computer science, e.g., the combination of multi-objective evolutionary learning and fuzzy modeling in [18] and the random forest based model of [19]. Compared to layout automation, a much richer spectrum of optimization techniques has been used for *circuit sizing* [20]. Fuzzy logic, despite gaining popularity in various types of applications (e.g., for the tuning of servo systems in [21]), has been of almost no interest for IC layout design so far, with just very few exceptions such as the placement approach [22].

One might argue, that many layout automation tools incorporate both algorithmic and procedural aspects (in the sense of approaches described in Sect. 3.1 and Sect. 3.2, respectively). That is right, but on closer examination, such tools always reveal a clear emphasis on one of the two automation strategies and the respective form of constraint consideration, which is either explicit or implicit. The same is true for template-based approaches such as [23] that extract implicit design knowledge from existing layout designs. Although this looks like a hybrid approach, the design knowledge is in fact turned into a formal representation and processed by a conventional optimization algorithm in a purely explicit fashion.

It can be said, that there is no layout automation approach which facilitates a balanced consideration of both explicit and implicit constraints, although today's manual design style suggests that both forms are indispensable to cope with the *More than Moore* design complexity in the analog domain. This is exactly where the multi-agent approach of SWARM is supposed to make a difference. Here, it should be noticed, that SWARM is not a method of swarm intelligence (in the algorithmic sense), for it does not employ a pool of candidate solutions. Instead, every agent in SWARM is a layout module and—as such—a *part* of the solution. This should become clear from the discussion of SWARM that follows in Sect. 4.

4 The Multi-agent System of SWARM

Although layout *algorithms* (again, referring to the prevalent, canonical form of optimization from Sect. 3.1) and procedural *generators* (as covered in Sect. 3.2) follow entirely different EDA strategies, both share a common drawback: to work in a totally *centralized* fashion. Either the algorithm must solve the entire design problem, or the solution must be completely preconceived by the design expert who develops the generator. SWARM addresses this problem by implementing a multi-agent system in which *module PCells* for analog basic circuits interact with each other to constitute an overall layout block in a flow of self-organization. This *decentralization* not only breaks down the complexity of the design problem, but allows low-level design constraints to be *implicitly* considered by each PCell, while high-level constraints can be *explicitly* considered during the interaction.

After referring some relevant agent-related works in the next subsection, the remaining subsections first give an overview of the SWARM system and its flow of PCell interaction, and then delve into some detailed features. Although the approach cannot be covered in its entirety here, the subsequent discussion should illustrate that SWARM is a relatively elaborate multi-agent system—mainly due to the following aspects: the interaction territory continually changes throughout the flow (Sect. 4.2), the agents don't obey a simple payoff function (Sect. 4.3), and the agents' potential actions always depend on their current situation (Sect. 4.4). Some notes on the design and application of SWARM are also given (Sect. 4.5).

4.1 Related Work

The SWARM approach is encouraged by the observation that multi-agent systems can outperform classical approaches in handling complexity [24]. For the design of such systems, principles of self-organization (like the *edge of chaos* [25]) can be accounted for to obtain operating points that are both stable and dynamic enough to effectively stir up *emergent behavior*. This phenomenon of *emergence* [26], which can be observed in many natural processes (e.g., crystallization), but also in artificial systems such as Conway's cellular automaton Game of Life [27], has become a topic of interest in many fields of science, philosophy, and art.

SWARM's idea of PCells that perform areal movements across a geometrical layout plane, is closely related to Reynold's Boids [28]. Both approaches feature a vivid biological analogon: while Boids simulates the motion of a flock of birds, SWARM imitates the roundup of an animal herd. As will be shown, the interaction in SWARM is not one of direct communication, but of *stigmergy* [29]—the kind of indirect coordination found among animal societies such as ant colonies. Existing agent simulators are discarded by SWARM due to its specific purpose so instead, the system has been conceived and implemented from the ground up.

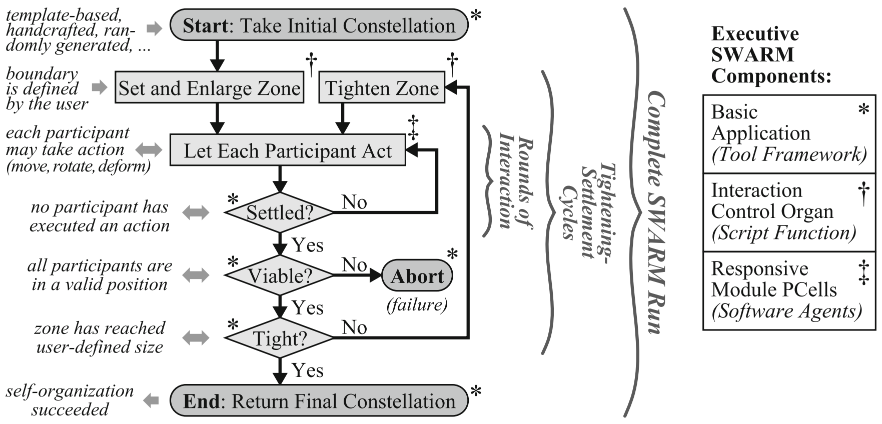
template-based, handcrafted, randomly generated, ...
Start: Take Initial Constellation *
boundary is defined by the user
Set and Enlarge Zone †
Tighten Zone †
each participant may take action (move, rotate, deform)
Let Each Participant Act ‡
no participant has executed an action
* Settled?
No
Yes
all participants are in a valid position
* Viable?
No
Abort *
(failure)
Yes
zone has reached user-defined size
* Tight?
No
Yes
self-organization succeeded
End: Return Final Constellation *
Rounds of Interaction
Tightening-Settlement Cycles
Complete SWARM Run
Executive SWARM Components:
Basic Application (Tool Framework) *
Interaction Control Organ (Script Function) †
Responsive Module PCells (Software Agents) ‡

Fig. 5 Control flow in the multi-agent system SWARM for analog layout automation

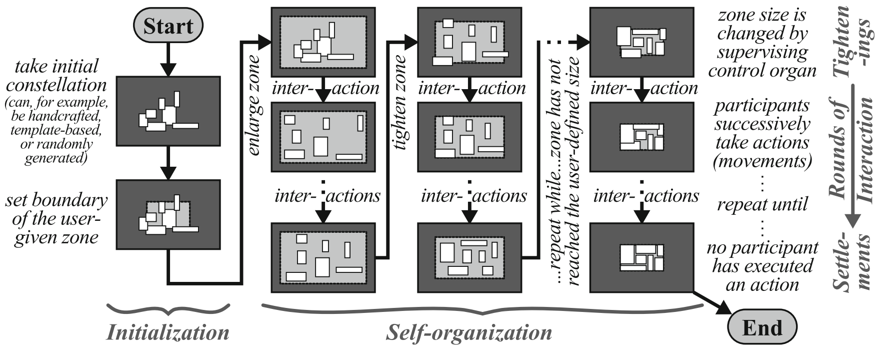
Start
take initial constellation
(can, for example, be handcrafted, template-based, or randomly generated)
set boundary of the user-given zone
enlarge zone
tighten zone
inter- action
inter- actions
...repeat while...zone has not reached the user-defined size
zone size is changed by supervising control organ
participants successively take actions (movements)
repeat until
no participant has executed an action
Tighten -ings
Rounds of Interaction
Settle- ments
End
Initialization
Self-organization

Fig. 6 Exemplary depiction of SWARM's PCell interaction and self-organization flow

4.2 Overview of SWARM and Its Interaction Flow

Due to conceptual limitations, PCells cannot access their design context. So, to use PCells as agents (as in Fig. 3c), SWARM enhances their native, "introversive" purpose (i.e., creating an internal layout) with an "extroversive" behavior by which a PCell can take an action (move, rotate, or deform itself) in response to changes of its environment. On this basis, SWARM allows a set of such *responsive PCells* to interact with each other and arrange themselves inside a given rectilinear zone. SWARM also implements a *supervising control organ* that recursively tightens the zone to steer the interaction towards a compact layout arrangement without depriving the PCells of the leeway needed to organize themselves. The overall flow of control in such a so-called SWARM *run* is depicted in Fig. 5 and exemplarily illustrated in Fig. 6. A more algorithmic presentation is given in [1].

Given a set of responsive PCells, SWARM takes an initial constellation (that may be template-based, handcrafted or randomly created) and centers the user-defined zone Z on it. Next, the supervising control organ enlarges Z so its area is significantly greater than the sum of the PCell areas. Then, the self-organization starts with a *round* of interaction, where each PCell (subsequently called a *participant*) performs an action. Multiple rounds are executed until no participant moves within one round. If the constellation, denoted as a *settlement* because the participants have *settled*, is *viable* (i.e., all participants are in a legal and satisfying position), then Z is tightened to induce another settlement (otherwise, the SWARM run failed and is aborted). The tightening-settlement cycle continues until Z reaches the user-defined size. The last settlement ends the SWARM run and represents the final solution. Regarding the biological analogon of Sect. 4.1, the PCells can be thought of as sheep, with the control organ being the shepherd.

4.3 A Participant's Condition and Its Influencing Factors

In terms of game theory, SWARM can be considered a noncooperative, infinitely-repeated, imperfect-information game in extensive form, with an unknown number of *stage games* [30]. In each stage game (here: a *round* of interaction), every participant acts in a self-interested, *utility-theoretic* way to improve its personal situation (here: the participant's *condition*). But unlike typical *utility functions* that map an agent's preferences to a real number, SWARM implements a more sophisticated decision model composed of several influencing factors which affect a participant's condition and thus stimulate its actions, as illustrated in Fig. 7.

The currently implemented influencing factors largely rely on the fact that all participating PCells are geometric objects with a rectangular bounding box and thus have an area, in contrast to dot-like particles found in some other systems. A participant P is forced to take an action if it suffers at least one of the following factors:

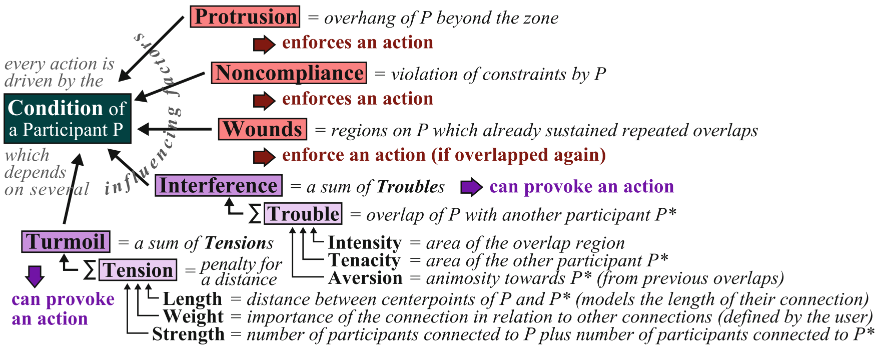
Protrusion = overhang of P beyond the zone
enforces an action
every action is driven by the
Noncompliance = violation of constraints by P
enforces an action
Condition of a Participant P
Wounds = regions on P which already sustained repeated overlaps
enforce an action (if overlapped again)
which depends on several influencing factors
Interference = a sum of Troubles
can provoke an action
∑ Trouble = overlap of P with another participant P*
Intensity = area of the overlap region
Tenacity = area of the other participant P*
Aversion = animosity towards P* (from previous overlaps)
Turmoil = a sum of Tensions
∑ Tension = penalty for a distance
can provoke an action
Length = distance between centerpoints of P and P* (models the length of their connection)
Weight = importance of the connection in relation to other connections (defined by the user)
Strength = number of participants connected to P plus number of participants connected to P*

Fig. 7 A participant (PCell agent) is influenced by five factors with several subfactors

protrusion (overhang beyond the zone boundary), *noncompliance* (violation of constraints), or *wounds* (regions on *P* which repeatedly overlapped with other participants in previous rounds of interaction). Further factors, which *can* (depending on their magnitude) provoke an action, are *interference* and *turmoil.*

Interference is a sum of *troubles*, each of which is caused by an overlap of *P* with another participant *P**. As detailed in Fig. 7, each trouble again involves several subfactors. One of these is the *aversion* of *P* towards *P**. Like wounds, aversions are inflicted (or aggravated) by overlaps and can be seen as concepts of memory. Aversion has a long-term effect that hinders a perpetual interference of two participants, while a wound has an immediate impact and is more effective for preventing marginal interferences. Both are essential aspects of the decision model and a careful balance of these is crucial for a fluent progress of interaction.

Turmoil is a concept relevant if participants should move close to each other because they need to be electrically connected afterwards, thus minimizing the total wirelength. Turmoil is a sum of *tensions*, with each tension penalizing the distance between the acting participant *P* and another one to which it is directly connected. Like trouble, tension involves multiple subfactors as Fig. 7 points out. For some tasks, turmoil is essential (e.g., for the floorplanning task in Fig. 10), but for others it may be ignored (as in the place-and-route problem of Fig. 11).

4.4 How a Participant Determines and Performs an Action

Many classes of problems, to which multi-agent systems have been applied, involve agents whose possible actions are simply given as a predefined, discrete, limited set of alternative options, from which the agent has to choose one. In SWARM, the situation is more complicated because the participants can freely move across a continuous layout plane. So, to take an action, a participant must first determine its set of possible actions depending on the current constellation. A participant's decision-making always follows a common *action scheme* which consists of the following four measures: (1) assessing the participant's condition, (2) perceiving its so-called *free peripheral space*, (3) exploring and evaluating all possible actions, and (4) executing the preferred action (or staying idle). Each of these measures, exemplarily illustrated in Fig. 8, will be subsequently described.

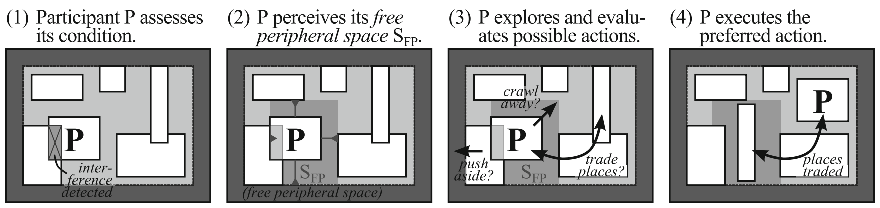
(1) Participant P assesses its condition.
(2) P perceives its *free peripheral space* S_{FP}.
(3) P explores and evaluates possible actions.
(4) P executes the preferred action.
P
inter-ference detected
S_{FP}
(free peripheral space)
crawl away?
push aside?
trade places?
places traded

Fig. 8 A participant's actions follow an action scheme consisting of four measures

Considering a constellation of six participants as in Fig. 8 (1), assume that it is participant P's turn to take an action. Following the action scheme, P begins by assessing its condition. While P is not in a state of *protrusion* nor *noncompliance* here, P detects *interference* with another participant. For that reason, P is said to be *discontented* and strives for an action that improves its condition.

To do so, P perceives the vacant area around it, since most of a participant's possible actions are based on this so-called *free peripheral space* S_{FP}. As shown in Fig. 8 (2), SWARM determines S_{FP} as a rectangle obtained by extending each of P's four edges in its respective direction until another participant—or the zone boundary—is encountered. Surely, other definitions of S_{FP} are also conceivable.

Next, as indicated in Fig. 8 (3), P explores all actions available in the current situation. Although a participant could implement its own actions, SWARM provides nine different elementary actions that are automatically explored (see [1]). Six of these actions (called *centering*, *budging*, *evasion*, *yielding*, *re-entering*, and *lingering*) only affect the acting participant itself (or none at all), while three actions (called *pairing*, *swapping*, *hustling*) also involve other participants. For the latter class of actions, only translational moves are considered. Otherwise, rotations and deformations are also explored. In this context, deformation means to assume another layout variant with equal electrical behavior (such as in Fig. 2).

The actions are evaluated so the one which improves P's condition the most can be chosen and executed. In Fig. 8 (4), the executed action is even synergistic: P trades places with another participant and *both* get rid of their interference.

4.5 Final Remarks About the Design and Application of SWARM

The participants' desire to improve their condition and the repeated layout zone tightenings are the driving forces behind SWARM's self-organization. It should again be emphasized that the participants act in a *utilitaristic* way and that each tightening represents just a *stigmergic* change of their environment. So, the inherent optimization of the overall layout solution is not incited directly (like with a global cost function, as is typical for EDA algorithms), and is thus not actively pursued via *collaborative* agents but emerges from their *competitive* interaction. This effect can also be observed in nature, where animals within a population seek cover behind other conspecifics to not get caught: such behavior inevitably leads to aggregations with reduced predation risk (see *selfish herd theory* [31]). To yield suchlike emergent behavior, SWARM requires a well-balanced taring.

One the one hand, there is the basic SWARM system itself (the overall flow, the model of influencing factors, the action scheme, the elementary actions, etc.). It has been developed in cooperation with layout experts from the industry, but also reflects several principles of self-organization such as the edge of chaos [25] (addressed with wounds and aversion), Ashby's law of requisite variety [32] (i.e., exploiting a PCell's layout variants during the interaction), and the reduction of friction [24] (by inducing synergistic moves as in Fig. 8 (4)). That basic system is supposed to be generally feasible, independent of the particular layout problem.

On the other hand, the system involves numerous actuating variables which allow a layout designer to fine-tune SWARM when applying it to a specific problem. These variables, described in detail in [1], mostly pertain to the influencing factors (Fig. 7) and the zone tightening. They can thus be used to adjust the participants' decision-making (between tentative and vivid) and to set an adequate tightening policy (between relaxed and aggressive). A more sensitive approach would be to make the tightening—and also the participants' behavior—*adaptive* so it changes during the interaction. Going one step further, this might even be achieved using an overlay network of synchronized agents as presented in [33].

Such advanced techniques, as well as the dependence of the self-organization on the actuating variables, on the initial module constellation, and on the order in which the participants act, are subject to further investigation. Another idea not examined so far is to equip the participants with different *attitudes* (between dominant and subordinate). In these regards, SWARM provides many opportunities for future enhancements. Still, remarkable results can already be achieved with the current implementation of SWARM, as will be demonstrated in Sect. 5.

5 Implementation and Examples

With the exclusive aim of layout automation, targeting the IC design framework *Cadence Virtuoso*®, SWARM has been implemented from scratch in Virtuoso's own script language SKILL, using PCell Designer [16] for developing the PCells.

Figure 9 has a theoretic example with eight participants (PCell agents) inside a rectangular zone (a) disregarding PCell connections. The PCells are just symbolic (i.e., they contain no real layout) and perform only translational movements (b), (c). The final constellation (d) demonstrates, that—without having knowledge of the overall placement problem—the agents are, in principle, able to find even the *optimal* solution (from which this example was initially constructed from).

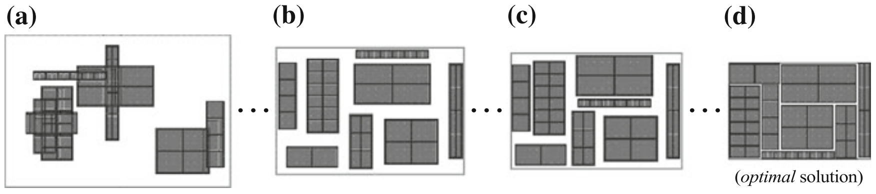

Fig. 9 Example of applying SWARM to a hypothetical analog layout design problem. **a** Initial SWARM constellation. **b** Intermediate constellation. **c** Intermediate constellation. **d** Final constellation

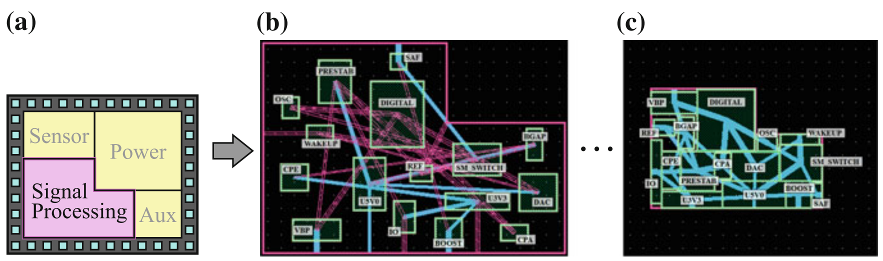

Fig. 10 Example of applying SWARM to a practical IC layout floorplanning problem. **a** Section of an IC to be floorplanned. **b** Initial SWARM Constellation. **c** Final constellation

Figure 10 uses SWARM for *floorplanning* a mixed-signal chip section (a) with digital and analog *blocks*. Each block is realized as a simple "hull" PCell, leaving the internal layout to subsequent layout steps. As indicated in Table 1 (a), the aspect ratios of such blocks may cover *full* variability (not discrete variants as in Fig. 2) and furthermore, the layout zone can be *rectilinear* (not just rectangular). SWARM supports both, and also takes block distances into account here. In the initial constellation (b), 19 out of 30 connections violate a maximum-length constraint (see patterned lines). In the compact arrangement of SWARM's final constellation (c), these constraints are all satisfied (solid lines) and all blocks lie inside the given layout zone. In floorplanning, the final routing of the blocks is usually done in layers above them, so no space has to be reserved between them.

The 'dead' space and runtime (*quantitative* criteria that stem from digital design automation) are comparable to other works, but more importantly, SWARM surpasses existing approaches in a *qualitative* way, reflecting the notion of quantitative and qualitative complexity in the digital and analog IC domain (Sect. 2): to our knowledge, SWARM is the first floorplanner that (1) minimizes both area and wirelength, (2) supports *nonslicing* floorplan structures, (3) allows for *fully* variable block dimensions, (4) pays respect to a user-definable *rectilinear* outline, *and* (5) works completely *deterministic* because no randomization is involved.

The primary purpose of SWARM is *placement* and *routing* (Table 1 (b), (c)). As an example, Fig. 11a shows the schematic of an amplifier circuit found in high-precision automotive IC designs. It consists of several analog basic circuit modules for which a desired arrangement (b) is defined via constraints to even out parasitic effects with overall layout symmetry. The final layouts show SWARM's results for a square zone (c) and for a 4:1 boundary (d). By allocating additional space around every participating module during interaction, electrical wires are created between the participants via a subsequent PCell-based routing step [34].

For the *differential pair* (1), the layout is created by the "Quad" PCell known from Fig. 4. Modules (2)–(5) are different types of *current mirrors*, each of which is also covered by a respective (and responsive) PCell agent. In fact, the flow is closer to that in manual design: first, the circuit's *primitive devices* are instantiated in the layout, and then the PCell agents are imposed on them as so-called *governing modules* which manage the devices by placing and routing them [1].

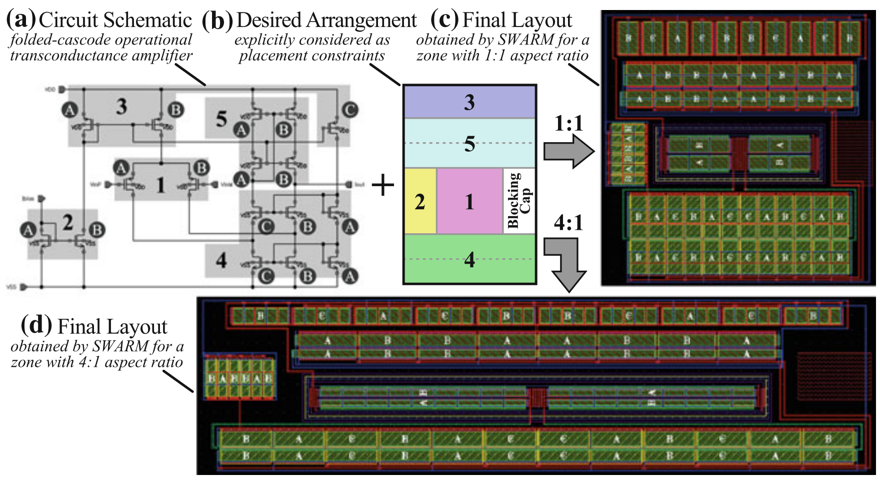
(a) Circuit Schematic
folded-cascode operational transconductance amplifier
(b) Desired Arrangement
explicitly considered as placement constraints
(c) Final Layout
obtained by SWARM for a zone with 1:1 aspect ratio
(d) Final Layout
obtained by SWARM for a zone with 4:1 aspect ratio
3
5
2
1
Blocking Cap
4
1:1
4:1

Fig. 11 Example of applying SWARM to a practical place-and-route layout problem

Table 2 Comparison of SWARM with algorithmic and procedural layout automation

	Algorithmic (Optimizer)	Procedural (Generator)	The SWARM approach (Multi-agent system)
Layout solution	Found by the engine	Preconceived by human expert	Emerges from Interaction of PCell Modules (= Agents)
Constraint considera-tion	Explicit (formalized)	Implicit (nonformalized)	Explicit and implicit (formalized and nonformalized)
Paradigm	"Top-down"	"Bottom-up"	"Bottom-up Meets Top-down"

This example displays SWARM's *qualitative* asset: while existing approaches are bound to either an explicit *or* implicit consideration of constraints, SWARM supports *both*. So, high-level requirements, such as the zone boundary and desired arrangement are *explicitly* enforced during interaction. Also, in this example the *Quad* module (1) is explicitly prevented from changing its AB/BA layout into a single-row variant. Furthermore, the *Blocking Cap* obeys a *following* move to mimic the actions of *current mirror* (2) and thus obtain overall layout symmetry. Simultaneously, each PCell module *implicitly* takes care of all detailed low-level *matching* restrictions and objectives, such as those listed in Fig. 4 for the *Quad*.

Besides all of these technical merits, Table 2 evaluates SWARM from a more strategic point of view. While algorithmic and procedural layout automatisms (in the sense of Sects. 3.1 and 3.2) follow different paradigms referred to as *top-down* and *bottom-up* automation [4], SWARM can be regarded as a combination of the two. And in the long run, such *bottom-up meets top-down* approaches may be one essential key to finally close the automation gap in analog layout design.

6 Summary and Outlook

The novel layout automation approach SWARM implements a multi-agent system where autonomous module PCells interact with each other to attain compact arrangements inside a recursively tightened layout zone. In contrast to optimization algorithms and procedural generators, the decentralized decision-making in SWARM allows it to consider design constraints both explicitly and implicitly.

Our approach is currently being implemented and tested in an industrial IC design environment, and early assessments indicate that SWARM is much closer to a human expert's manual design style than existing automation strategies. As shown in several given examples, remarkable layout results can emerge from the aggregate PCell interaction by inducing a synergistic flow of self-organization.

The presented results have been achieved with the very first implementation of SWARM. So, considering that the approach is still in its infancy, there is enormous potential for further developments. Future work on SWARM includes (a) the realization of multiple concurrent control organs, (b) adaptive control policies, (c) hierarchically nested interaction flows, (d) modules with learning aptitude, (e) improvements of convergence and robustness, (f) parallelization of module activity via multi-threading, and even (g) real-time human intervention.

References

1. Marolt, D., Scheible, J., Jerke, G., Marolt, V.: SWARM: a self-organization approach for layout automation in analog IC design. Int. J. Electron. Electr. Eng. **4**(5), 374–385 (2016)
2. Arden, W., Brillouët, M., Cogez, P., Graef, M., Huizing, B., Mahnkopf, R.: 'More-than-Moore' white paper. International Technical Roadmap for Semiconductors (2010)
3. Hastings, A.: The Art of Analog Layout, 2nd edn. Prentice Hall (2005)
4. Scheible, J., Lienig, J.: Automation of analog IC layout—challenges and solutions. In: Proceedings of the ACM International Symposium on Physical Design, pp. 33–40 (2015)
5. Rutenbar, R.: Analog CAD: not done yet. Pres. NSF Workshop: Electronic Design Automation—Past, Present, and Future, Slide 6 (2009)
6. Kirkpatrick, S., Gelatt Jr., C., Vecchi, M.: Optimization by simulated annealing. Science **220**(4598), 671–680 (1983)
7. Bowen, O.: Rip-up, reroute strategy accelerates routing process. IEEE Potentials **25**(2), 18–23 (2006)
8. Lin, P., Chang, Y., Lin, S.: Analog placement based on symmetry-island formulation. IEEE Trans. Comp.-Aid. Des. Integr. Circ. Syst. **28**(6), 791–804 (2009)
9. Lin, J., Hung, Z.: SKB-tree: a fixed-outline driven representation for modern floorplanning problems. IEEE Trans. VLSI Syst. **20**(3), 473–484 (2012)
10. Tang, M., Lau, R.: A parallel genetic algorithm for floorplan area optimization. In: 7th International Conference on Intelligent Systems Design and Applications, pp. 801–806 (2007)
11. Ou, H., Chien, H., Chang, Y.: Simultaneous analog placement and routing with current flow and current density considerations. In: Proceedings of the ACM/IEEE 50th Design Automation Conference, pp. 1–6 (2013)
12. Mittag, M., Krinke, A., Jerke, G., Rosenstiel, W.: Hierarchical propagation of geometric constraints for full-custom physical design of ICs. In: Proceedings of the Design, Automation and Test in Europe Conference, pp. 1471–1474 (2012)

13. Reich, T., Eichler, U., Rooch, K., Buhl, R.: Design of a 12-bit cyclic RSD ADC sensor interface IC using the intelligent analog IP library. In: ANALOG, pp. 30–35 (2013)
14. Graupner, A., Jancke, R., Wittmann R.: Generator based approach for analog circuit and layout design and optimization. In: Design, Automation and Test in Europe Conference Exhibition, pp. 1–6 (2011)
15. Marolt, D., Greif, M., Scheible, J., Jerke, G.: PCDS: a new approach for the development of circuit generators in analog IC design. In: Proceedings of the 22nd Austrian Workshop on Microelectronics (Austrochip), pp. 1–6 (2014)
16. Jerke, G., Burdick, T., Herth, P., Marolt, V., Bürzele, C., et al.: Hierarchical module design with Cadence PCell Designer. In: Pres. CDNLive! EMEA 2015, Munich, CUS02
17. Kaur, P.: An enhanced algorithm for floorplan design using hybrid ant colony and particle swarm optimization. Int. J. Res. Appl. Sci. Eng. Technol. **2**(IX), 473–477 (2014)
18. Gacto, M., Galende, M., Alcalá, R., Herrera, F.: METSK-HD^e: a multiobjective evolutionary algorithm to learn accurate TSK-fuzzy systems in high-dimensional and large-scale regression problems. Inf. Sci. **276**, 63–79 (2014)
19. Tomin, N., Zhukov, A., Sidorov, D., Kurbatsky, V., Panasetsky, D., Spiryaev, V.: Random forest based model for preventing large-scale emergencies in power systems. Int. J. Artif. Intell. **13**(1), 211–228 (2015)
20. Rocha, F., Martins, R., Lourenço, N., Horta, N.: Electronic Design Automation of Analog ICs combining Gradient Models with Multi-Objective Evolutionary Algorithms. Springer International Publishing (2014)
21. Precup, R., David, R., Petriu, E., Preitl, S., Rădac, M.: Fuzzy logic-based adaptive gravitational search algorithm for optimal tuning of fuzzy-controlled servo systems. IET Control Theory Appl. **7**(1), 99–107 (2013)
22. Lin, R., Shragowitz, E.: Fuzzy logic approach to placement problem. In: Proceedings of the ACM/IEEE 29th Design Automation Conference, pp. 153–158 (1992)
23. Chin, C., Pan, P., Chen, H., Chen, T., Lin, J.: Efficient analog layout prototyping by layout reuse with routing preservation. In: International Conference on Computer-Aided Design, pp. 40–47 (2013)
24. Gershenson, C.: Design and control of self-organizing systems. Ph.D. Dissertation, Vrije Universiteit Brussel (2007)
25. Langton, C.: Computation at the edge of chaos: phase transitions and emergent computation. Physica D: Nonlinear Phenomena **42**(1–3), 12–37 (1990)
26. Johnson, S.: Emergence: The Connected Lives of Ants, Brains, Cities, and Software. Scribner, New York, NY, USA (2001)
27. Gardner, M.: Mathematical games—the fantastic combinations of John Conway's new solitaire game 'Life'. Sci. Am. **223**, 120–123 (1970)
28. Reynolds, C.: Flocks, herds, and schools: a distributed behavioral model. In: Proceedings of 14th Annual Conference on Computer Graphics and Interactive Techniques, pp. 25–34 (1987)
29. Marsh, L., Onof, C.: Stigmergic epistemology, stigmergic cognition. Cogn. Syst. Res. Elsevier B.V. 1–15 (2007)
30. Shoham, Y., Leyton-Brown, K.: Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations. Cambridge University Press (2009)
31. Hamilton, W.: Geometry for the selfish herd. J. Theor. Biol. **31**(2), 295–311 (1971)
32. Ashby, W.: An Introduction to Cybernetics. Wiley, New York, NY, USA (1956)
33. Bojic, I., Podobnik, V., Ljubi, I., Jezic, G., Kusek, M.: A self-optimizing mobile network: autotuning the network with firefly-synchronized agents. Inf. Sci. **182**, 77–92 (2012)
34. Marolt, D., Scheible, J., Jerke, G., Marolt, V.: Analog layout automation via self-organization: enhancing the novel SWARM approach. In: Proceedings of the 7th IEEE Latin American Symposium on Circuits and Systems, pp. 55–58 (2016)