A Procedural Approach to Automate the Manual Design Process in Analog Integrated Circuit Design

Florian Leber and Juergen Scheible
Robert Bosch Center for Power Electronics, Reutlingen University
Alteburgstr. 150, 72762 Reutlingen, Germany
{florian.leber, juergen.scheible}@reutlingen-university.de

Abstract

This paper presents a novel approach to automating the design of analog integrated circuits: (1) the Expert Design Plan (EDP), a procedural generator, and (2) the EDP Language, a high-level description language for writing an EDP. An EDP is a parameterizable, executable script, which reproduces a designer's course of action when designing a circuit. Thus, an EDP formalizes the design expert's knowledge-based strategy and makes it reusable. Since it is essential that an EDP represents a circuit designers' way of thinking and working as close as possible, the designers themselves should be enabled to create the EDP. Therefor, our approach provides a input method through a domain-specific language called EDP Language (EDPL). Using this language is intuitive and requires no special training. In an exemplary implementation of our approach, a common-source amplifier is automatically sized using a set of only 10 instructions. Even in the first usage our EDP approach has appeared to be more efficient than the manual sizing process.

1 Introduction

The majority of today's ICs are mixed-signal systems consisting of digital and analog circuit parts. The design of these circuit parts is assisted by automatisms, whereby the degrees of automation in digital and analog design have evolved differently: The digital design has been highly automated for many years, whereas the analog design is still done in a laborious manual fashion. As a result, the design effort for the analog parts exceeds the design effort for the digital parts by orders of magnitude, although the digital parts often include 90% and more of the IC's functionality.
The success of automation in digital design is based on the restriction of design freedom, such as the use of standard cells. This allows a simplified, mathematical modeling of the digital design problem, thus enabling a successful application of algorithmic optimization techniques.
The principle of optimization algorithms is shown in Fig. 1(a): an optimization algorithm repeatedly refines a solution in a loop of solution space exploration and solution candidate evaluation [1]. For example, the exploration proposes a sizing (candidate) by a binary search algorithm, while the evaluation rates it according to a formal cost function and validates if it satisfies all constraints. The advantage of such optimization algorithms is the ability to find self-intelligently new solutions. However, an optimization algorithm handles exclusively the aspects which were transformed into its abstract mathematical model. Other aspects remain unconsidered.
In contrast to digital design, in the analog design a much bigger diversity of parasitics and their physical effects must be taken into account. Therefore, analog design is still performed at the transistor level because this allows the utilization of all design freedom. This makes the mathematical modeling of the analog design problem more difficult. Thus, a complete and sufficient description of constraints and optimization objectives on this low level is very complicated and time-consuming as well.
In practice this leads to a low acceptance of optimizing algorithms in analog design. Accordingly, the design is based almost exclusively on expert knowledge and is characterized by a manual design process. This has lead us to the idea that a further automation of analog design can be achieved by automating exactly this process. Therefore, we propose to transfer the manual process into a procedural approach.
The concept of procedural approaches is illustrated in Fig. 1(b). A procedure is a sequence of design steps that are previously conceived by human experts and are captured in an executable script [2]. Thus, a procedure is emulating an expert's decisions and makes the expert knowledge re-usable. The expert can steer the execution using parameters. In contrast to an optimization algorithm, which finds a new solution during run-time, the result is previously de-

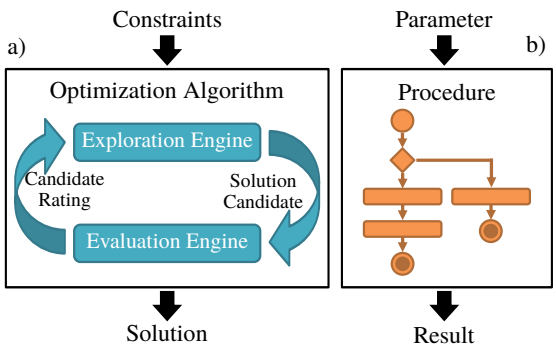

Figure 1 Automation strategies [1] [2]

ISBN 978-3-8007-4754-2

termined by the author of the procedure. The advantage of such procedures is, that they can consider all relevant aspects and constraints implicitly. A mathematical modeling is not necessary.

The success of this procedural approach is determined strongly to which extent it will be possible for experts to implement the procedure by themselves. This is often hindered by two obstacles: The implementation is (1) laborious and (2) not intuitive. We address these problems by providing a dedicated input method that match the experts' way of thinking.

Our paper presents a procedural approach for capturing the strategy of experts for analog circuit design and making it re-usable as an executable script. Our paper is structured as follows: Section II gives a brief overview of various works related to the topic of the paper. Section III describes our EDP approach and introduces our EDP language. Section IV presents an example and illustrates our concept. Section V concludes with a summary and an outlook.

2 Related Work

The automation of analog design has been investigated by EDA research for several decades. The resulting approaches can be classified into optimization-based and knowledge-based approaches. In the following we consider automation approaches for the sizing problem.

The optimization-based approaches are the prevalent approaches and can be roughly categorized according to their evaluation methods. At first the evaluation by equations has been in focus (e.g. OPASYN [3]). Since the mid-1990s evaluation by simulation emerged (e.g. FRIDGE [4], Anaconda [5], MOJITO [6], GENOM-POF [7]), whereby these are still the leading approaches. In the mid-2000s approaches for an evaluation by simulation based on simplified models have been presented (e.g. Alpaydin et al. [8]). The evaluation methods differ in terms of their performance: Evaluation by equations is fast but often too inaccurate. Evaluation by simulation is highly accurate but very slow. The evaluation based on simplified models is a compromise but it does mostly not cover all requirements in the industrial practice.

The optimization algorithms themselves have also been continuously developed. The first optimization algorithms adopted were simple, e.g. simulated annealing in FRIDGE [4] or pattern search in Anaconda [5]. Due to the fact that these approaches are unsuitable for multiple objectives the trend switched to multi-objective evolutionary algorithms like the NSGA-II in MOJITO [6] and the extended NSGA-II in GENOM-POF [7]. Other approaches attempt a hybrid approach and embed expert knowledge, like FASY [9] which uses fuzzy decisions defined by the designer.

Apart from these enhancements, in all presented optimization-based approaches the declaration of constraints is still comprehensively required. Moreover, the execution time of modern approaches is still high (a typical example is GENOM-POF with one hour for setup and several hours for execution [7]).

In addition to these optimization-based approaches, also knowledge-based approaches have been investigated by EDA research, e.g. IDAC, OASYS and OASE. In IDAC [10] the design is based on a formal description in a predefined order of steps (five steps for an OTA). A subsequent analyzer checks the specification match. In case of mismatch it automatically restarts the sizing with varied values. In OASYS [11] the approach is to divide a circuit into sub-blocks, whereby the circuit specifications are translated into sub-block parameters by a design plan ("linear sequence of executable steps"). Sub-blocks (e.g. current mirrors, differential pairs, etc.) in turn are designed through a "computation" of analytical equations. OASE [12] comprises an expert system that uses an iterative "uniform cycle" that decomposes circuits into less complex substructures and handles them by machine based reasoning. This is based on circuit-specific knowledge such as rules (e.g. selection rules) or heuristics (e.g. design strategies or problem estimations), that have been captured in a dedicated process of knowledge acquisition based on object-oriented programming. Since IDAC and OASYS are based on formulas the computer-aided calculation just requires a few seconds. But the results of such simplified formulas are too inaccurate for today's ICs with very small structure sizes. However, the OASE synthesis process includes a SPICE-like simulator that gives more accurate results. The big drawbacks of such knowledge-based approaches are (1) the long time to implement a new circuit (e.g. roughly one month in OASYS) and (2) the integration of expert knowledge by dedicated acquisition processes with inappropriate programming languages (e.g. Franz LISP in OASYS).

Despite intensive EDA research, none of these automation approaches (optimization-based and knowledge-based) for analog design could reach a significant dissemination into industrial practice. However, in other areas of the analog chip design, there are successful knowledge-based approaches: Since the 1990s, procedural generators have been successfully applied to layout design. The so-called parameterized cells (PCells) are widely used to generate layouts from simple transistors to basic circuit modules. Recently, the PCell approach has been also adapted to circuit design (Circuit PCells) for generating symbols and schematics [13]. In Circuit PCells device dimensions like W/L values as well as topological circuit variants are controlled by parameters. One reason for the success of PCells is that their implementation can be done in a very comfortable way with a high-level input based on special graphical interfaces. Examples are the PCell Designer [14] for layout and schematic PCells or gPCDS for Circuit PCells [15].

3 The EDP Approach

3.1 The Idea

Looking at the industrial practice it can be observed, that the analog circuit design is characterized by a largely manual design process consisting of these three steps:

(i) Topology: Determination of the circuit structure, displayed as a schematic which contains the components and their electrical connections

(ii) Sizing: Determination of the parameter values of the components
(iii) Verification: Functionality check of the circuit by simulation

These steps are repeated iteratively until the desired behavior is achieved. The design expert corrects and adjusts the parameters (and possibly also the topology if necessary) based on his expert knowledge and experience.

Our idea is to automate exactly this manual design process by embedding it in procedures. Thus, these procedures implicitly contain the knowledge and experience of the experts. This can be regarded as the transfer of the PCell principle to the task of analog circuit design, i.e. the work of designing an analog circuit is automated in a procedural approach. For this, the work steps must be formalized and converted into executable commands. By executing these commands the original design actions are reproduced. We call the script, formed by a sequence of suchlike commands, the *Expert Design Plan* (EDP) to point out that the expert knowledge is being transferred to the plan.

In the design flow such an EDP can be considered as the counterpart of a circuit module PCell. It has the ability to complement the PCell approach, which is already available on the structural view (with circuit PCells) and the layout view (with layout PCells), on the functional view, thus leading to a fully generator-like analog design flow based on procedural automatisms.

Since the experts themselves are best familiar with the manual design process, our idea is that the experts shall create the EDPs by their own. Therefore, it will be particularly important to map the way of thinking and working of analog circuit designers as accurately as possible in the EDP. We propose a domain-specific language, which we call *EDP Language* (EDPL) to provide a high-level input for this purpose.

3.2 Requirements

In the manual design process the expert interacts directly with the design environment via schematic editor and simulator. This interaction will now be taken over by the EDP. Therefore, a machine connection to the design environment and its functions is needed. For the control and the return / feedback of the EDP execution a user interface is required. To imitate the real manual design process, a method for formalizing this process in detail is needed. This requires to derive specific expressions for all occurring process steps.

Since an EDP must be created by experts themselves, it is important to obtain their acceptance for our approach. Therefore also some “soft” requirements must be fulfilled. It must be possible to input the design steps in an intuitive manner and the effort for creating an EDP should not exceed the effort for manual design. Because designers regularly develop new circuits by copy-pasting existing designs and varying details, an EDP should also be easily changeable.

3.3 Expert Design Plan

Our approach supports the expert during the manual design process by taking the responsibility for the process. The EDP takes over the direct interaction with the design environment and can thus reproduce the original design actions of the expert. For that we propose the structure shown in Fig. 2. The EDP is attached to the design environment with the connector. This connector handles the function calls and ensures the correct control of the simulator in the background. Using the connector, the EDP is executable directly in the design environment and can also use its functions. Thus, it is also possible to control the EDP and output results / simulations via the design environment. The expert can therefor execute the EDP over his usual environment and use the known tools. This simplifies the execution and increases the acceptance of the approach.

From the perspective of the EDP execution, the EDP translates input parameters to a schematic based on the same strategy and in the same manner as the expert would have done manually. The execution model is shown in Fig. 3. The input parameters steer the execution of the EDP. Typical input parameters are performance parameters such as gain, band width or slew rate. During execution, the EDP iterates the three typical design steps: topology, sizing and verification. The execution takes design decisions according to the conditions formalized in the EDP. An execution ends when the specified design objectives are met. Then the EDP outputs a fully sized schematic. Since an EDP can also be used to adapt a circuit topology, EDPs are in principle capable for a wide range of input parameters. However, if the execution fails, the EDP needs to be adjusted.

It should be emphasized that in a schematic only the final result of a design process is stored. Normally there is no information how this result has been achieved. In contrast to this, our approach features some significant advantages: (1) An EDP contains a detailed description of the solution path for a finally sized schematic without the need for any additional documentation. (2) Because of its executability this description is easily re-usable, thus offering a big potential to improve design productivity. (3) Additionally, an EDP is an easy way for archiving design knowledge, which can be reused for other purposes, like training new designers.

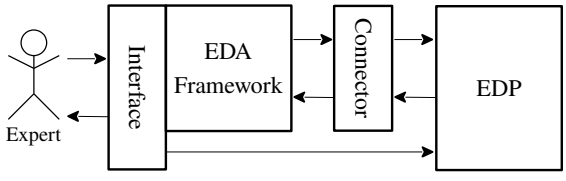

Figure 2 EDP Interaction Model

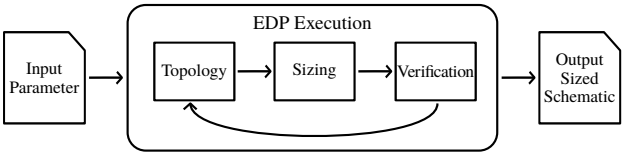

Figure 3 EDP Execution Model

3.4 EDP Language

Technically, an EDP contains a sequence of design commands written in the EDP Language (EDPL). The design commands are high level commands and correspond to real steps of a manual design process. These commands serve as the interface for the high-level knowledge input. Each EDPL-command invokes operations implemented in a logic-core. The EDPL is a kind of domain-specific language (DSL) which differs from other knowledge-based approaches that use more complex and general-purpose languages like Franz LISP in OASYS [11].

A DSL is a language that is exactly adapted to the target domain. Thus, an expert can use it "in his language". As a result, a DSL is very intuitive and requires no specific training. Another advantage is that it provides exactly the tools, i.e. the design commands, needed to solve the domain problems. Using these commands, the thinking and acting of the experts can be easily and accurately formalized.

We propose to derive the high-level design commands from an accurate modeling of the manual design flow. Other commands for control, debugging, etc. are based on typical programming languages. We divide the design commands into these five categories:

(a) *Formula commands* are used to calculate the initial sizing and to determine the dependencies between the parameters.
(b) *Simulation commands* are a set of commands to initialize simulator and test bench, auto run the simulation, qualitatively check the results and get performance parameters from the simulation.
(c) *Device commands* allow to set and get device values.
(d) *Topology commands* modify the structure, i.e. adding transistors and changing nets.
(e) *Control commands* are used for conditions, iterations, logging and breaks (for debugging).

In addition to these design commands, the EDPL also contains typical programming functionalities such as the creation of subroutines. Therefore, the EDPL facilitates a flexible coding style and the expert knowledge can be organized flat or hierarchical as desired.

3.5 Typical Use Cases

In practice the EDP can be applied in several ways. Three typical use cases are (1) to reuse an existing EDP, (2) to modify an existing EDP and (3) to create a new EDP.

In the first use case the EDP is readily parameterized (i.e. the set of supported performance parameters is implemented and sufficient for the design problem) and can be easily reused from a previous project. An expert just chooses a desired set of values for the given set of performance parameters and executes the EDP.

Of course this works only for a certain range of parameter values. For parameters falling out of this range the EDP has to be changed. This leads to the second use case, where an existing EDP can be used as basis for further implementations. Only parts of the EDP are modified and, for example, simply extended by further process steps. Thereby the already contained expert knowledge is re-used and the effort is minimized compared to a new implementation.

If a design problem differs to much from the problems already covered by existing EDPs, a new implementation is needed. This will be regularly the case, when an EDP is required for a new circuit class, i.e. a circuit with another basic functionality than those for which EDPs are available. In this third use case the EDP has to be fully implemented from scratch. To minimize this effort, the EDP includes high-level commands, such as applying predefined simulations.

Once a designer team has EDPs for all its frequently used circuit classes, the EDP approach deploys its full automation power.

4 Illustration

4.1 Prototype for a Common-Source Amplifier

In the current state of our work we have developed the EDP Language to an extent that it is able to support a typical sizing process for basic analog circuits. The actual functional capability allows the creation of EDPs for achieving typical performance parameters. In order to demonstrate the feasibility of our approach, we have implemented it for the sizing of a common source amplifier (see Fig. 4). Although this circuit is very simple (3 transistors), it is complex enough to capture and illustrate the core steps of the sizing procedure in the analog design process.

Based on this example we have collected experience, estimated the performance and showed the basic suitability of our approach. Our implementation ensued these four steps: (1) Modelling of the manual design process, (2) deriving and implementing EDP commands from this modelling, (3) creating EDPs for various scenarios, and (4) evaluating and improving the approach based on the collected experience.

4.2 Implementation

The manual sizing process of the common-source amplifier was modelled on the basis of 29 steps. Based on this modeling, we derived 10 types of design commands (see Tab.1), which we implemented with SKILL++ in Cadence Virtuoso [16].

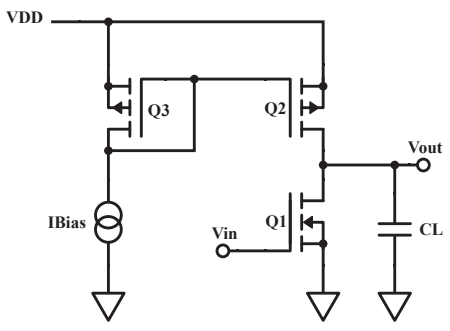

Figure 4 Common-Source Amplifier

Table 1 Derived EDPL commands

makeInstance	Initialization and test bench setup
setTarget	Specification of performance parameter
calculateStartValues	Calculate the formulas
simulate	Run simulation
checkResults	Qualitativ check of simulation results
getPParameter	Get simulated performance parameter
getPercentComplete	Get percentage deviation between specified and simulated performance parameter
set / get	Set and get of transistor values
increase /decrease	In- and decrease of transistor values
calcApproximation	Calculate proportional deviation based on formula and the parameters

The simulation loop is an important part of the sizing process to be implemented. Our EDP approach directly integrates a test bench and provides high-level commands, such as the automatic extraction of performance parameters from the simulation results. These high-level commands make it very easy to start both the simulation and the checking of results with single commands.
The listing below shows the sizing of a parameter W1 as a typical code example for the implemented EDPL commands [16]:

```
1  CS->setW1( CS->calcApproximation( "Av" "W1"
       ?percent 100 ) )
2  CS->simulate()
3  CS->checkResults()
4
5  while( CS->getPercentComplete()<100
6     CS->increaseW1(1u)
7     CS->simulate()
8     CS->checkResults()
9  )
```

In the first line the value is roughly calculated via the dependencies of the formulas and set to the parameter. Then a simulation is started (line 2) and results are checked (line 3). In the last step, W1 is adjusted in a loop (line 5-9) with small steps (line 6).
We applied these EDPL commands on three scenarios for the common-source amplifier to create EDPs with the following objectives:
(1) gain 30db, band width 1.3MHz, slew rate 120V/us
(2) gain 46db, band width 0.3MHz, slew rate 50V/us
(3) gain 30db, band width 1.0MHz, slew rate 50V/us, ultra low noise

The initial EDP implemented for the first scenario comprises a total of 42 lines of code. It is able to cover the sizing of a common-source amplifier for a variation of about 10% of the required performance parameter values.
Since the other scenarios (consciously) deviate more strongly, additional EDPs (based on the initial one) had to be created. Our experimentee required only negligible time for training. Creating the initial EDP for the first scenario took about 2.5 hours. However, the two subsequent implementations took significantly less than an hour, as the first EDP could be reused. The execution of one of the EDPs itself requires only about 5 minutes, whereby the execution is mainly occupied by the simulation.

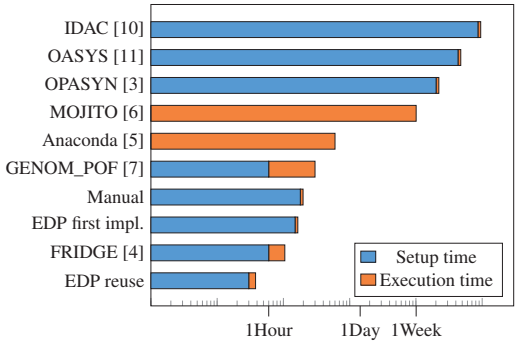

Figure 5 Comparison of EDP with other approaches

4.3 Results

The three realized implementations show that a comprehensive implementation of an EDP needs almost 2.5 hours (without learning the EDPL) plus a runtime of 5 minutes. The manual design process needs 3 hours. The comparison shows that by using our approach design effort can be saved (here 30 minutes) even in the first time of usage. This seams contradictory but details show: The effort for creating an EDP is not negligible, but the included automatisms (e.g. auto simulation setup) in EDP overcompensate the invested time. The time for implementing a second, slightly different EDP is obviously smaller than 2.5 hours, depending on the degree of difference.
These results can be compared to previous approaches based on their setup time and execution time. The setup time comprises e.g. the setting of constraints (optimization-based approaches) or the capturing of knowledge (knowledge-based approaches). In IDAC, OASYS and OPAYSN the setup time takes weeks to months and the execution time is a few seconds. In MOJITO and ANACONDA no setup times were outlined, but for MOJITO the typical execution time is known to be about 7 days and for ANACONDA several hours. In contrast, GENOM-POF and FRIDGE indicate a setup time of 1 hour. The typical execution time is a few hours for GENOM-POF and only 45 minutes for FRIDGE. A rough comparison is shown in Figure 5 (please note the logarithmic axis), but it should be noted that the approaches partly implement different circuits and use older hardware for execution.

5 Summary and Outlook

This paper presents the idea of a procedural approach for the automation of analog circuit design: the Expert Design Plan (EDP). Its corresponding EDP Language (EDPL) is intended to facilitate the capturing of design strategies of analog design experts for making them re-usable in form of executable procedures.
In contrast to many other approaches, which have not found their way into a wide industrial usage, we hope that our approach can find acceptance by industrial designers. This believe is based on these facts: (1) The EDP approach is designed to enable an imitation of an analog designer's way of work as close as possible, thus leading to an ex-

ecutable expert design strategy. (2) The capturing of this design strategy can be done intuitively by using a special domain specific language EDPL. (In future work a graphical user interface will be developed.) (3) The EDP approach does not require a change of the design flow. It can be easily used as an addition to existing design flows.

As an illustration a first version of the EDP Language and some EDPs could be successfully implemented for the sizing of a common-source amplifier circuit. This exemplatory application shows the huge potential: (1) Even for the initial implementation and usage of an EDP the time needed is shorter than doing the sizing process in a classical manual manner. (2) The sole run time of an EDP undercuts the time for the manual process by far.

Actually we work on the integration of an abstraction layer in order to make the EDP independent of concrete component designations (e.g. transistor T1 with W1) in schematics. In addition, it needs research on intelligent termination capabilities of non-convergent loops. To further increase the acceptance we also plan to implement a graphical user interface for the input of expert knowledge. Furthermore, in our future work the functionality of the EDPL will be expanded to cover also topology changes and to support more complex decision strategies.

ACKNOWLEDGMENTS

We wish to thank Philipp Lamprecht for his work on the illustration [16].

This project is funded by the German Federal Ministry for Education and Research under the support code: 13FH051PX5.

6 Literature

[1] R Rutenbar. Analog CAD: Not done yet. In *NSF Workshop*, pages 8–9, 2009.

[2] Juergen Scheible and Jens Lienig. Automation of Analog IC Layout - Challenges and Solutions. In *Proceedings of the 2015 Symposium on International Symposium on Physical Design*, pages 33–40. ACM, 2015.

[3] Han Young Koh, Carlo H Sequin, and Paul R Gray. OPASYN: A compiler for CMOS operational amplifiers. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 9(2):113–125, 1990.

[4] Fernando Medeiro, Francisco V Fernández, Rafael Domínguez-Castro, and A Rodriguez-Vazquez. A Statistical Optimization-Based Approach for Automated Sizing of Analog Cells. In *Proceedings of the 1994 IEEE/ACM international Conference on Computer-aided design*, pages 594–597. IEEE Computer Society Press, 1994.

[5] Rodney Phelps, Michael Krasnicki, Rob A Rutenbar, L Richard Carley, and James R Hellums. Anaconda: Simulation-Based Synthesis of Analog Circuits Via Stochastic Pattern Search. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 19(6):703–717, 2000.

[6] Trent McConaghy, Pieter Palmers, Michiel Steyaert, and Georges G. E. Gielen. Trustworthy Genetic Programming-Based Synthesis of Analog Circuit Topologies Using Hierarchical Domain-Specific Building Blocks. *IEEE Transactions on Evolutionary Computation*, 15(4):557–570, 2011.

[7] Nuno Lourenço and Nuno Horta. GENOM-POF: Multi-Objective Evolutionary Synthesis of Analog ICs with Corners Validation. In *Proceedings of the 14th annual Conference on Genetic and Evolutionary Computation*, pages 1119–1126. ACM, 2012.

[8] Guner Alpaydin, Sina Balkir, and Gunhan Dundar. An evolutionary approach to automatic synthesis of high-performance analog integrated circuits. *IEEE Transactions on Evolutionary Computation*, 7(3):240–252, 2003.

[9] Antonio Torralba, Jorge Chavez, and Leopoldo García Franquelo. FASY: A Fuzzy-Logic Based Tool for Analog Synthesis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 15(7):705–715, 1996.

[10] Marc GR Degrauwe et al. IDAC: An Interactive Design Tool for Analog CMOS Circuits. *IEEE Journal of Solid-State Circuits*, 22(6):1106–1116, 1987.

[11] Ramesh Harjani, Rob A Rutenbar, and L Richard Carley. OASYS: A Framework for Analog Circuit Synthesis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 8(12):1247–1266, 1989.

[12] Klaus Hoffmann, Michael Mertens, Klaus Milzner, Werner Brockherde, Gunther Hess, Roland Klinke, and Florian Krohm. Oase: A knowledge based environment for analog circuit design. In *Rechnergestützter Entwurf und Architektur mikroelektronischer Systeme*, pages 157–168. Springer, 1990.

[13] Daniel Marolt, Matthias Greif, Jürgen Scheible, and Göran Jerke. PCDS: A new approach for the development of circuit generators in analog IC design. In *Proceedings of the 22nd Austrian Workshop on Microelectronics (Austrochip)*, pages 1–6. IEEE, 2014.

[14] Cadence Design Systems. Cadence PCell Designer For Cadence Virtuoso, 2017. `https://www.cadence.com/content/dam/cadence-www/global/en_US/documents/services/cadence-vcad-pcell-ds.pdf`.

[15] Matthias Greif, Daniel Marolt, and Juergen Scheible. gPCDS: An interactive tool for creating schematic module generators in analog IC design. In *Proceedings 2016 12th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME)*, pages 1–4. IEEE, 2016.

[16] Lamprecht P. Entwicklung einer domänenspezifischen Sprache für die automatische Dimensionierung einer analogen Schaltung. *Masterthesis, Robert Bosch Zentrum für Leistungselektronik an der Hochschule Reutlingen*, 2017.