AIDA-C: Evolutionary Optimization Techniques applied to Analog IC Design

André Ferreira
Instituto Superior Técnico - ULisbon
Lisboa, Portugal

***Abstract*—This paper presents an approach to automatically generate circuit-level design constraints to a layout-aware sizing approach. The proposed approach is an enhanced version and implementation of an established method, based on pattern recognition and symmetry detection, and is integrated in the AIDAsoft electronic design automation (EDA) environment. The generation of constraints increases the automation of the design process and reduces the risk of errors, assisting the project designer during the design specification setup. The validity and effectiveness of the proposed approach is illustrated for the synthesis of classical circuit structures in the AIDAsoft environment.**

Keywords—Analog Integrated Circuit; Electronic Design Automation; Constraints Generation.

I. INTRODUCTION

Analog integrated circuit (IC) design is still a critical task as the number of mixed-signal ICs, where analog circuitry play a decisive role, is increasing. Many EDA approaches have been proposed in the analog circuit domain to assist in the design process, but it still lags in the automation level, as such, analog design is still highly manual and a labor-intensive task.

As analog IC design automation is dominated by optimization-based sizing approaches that use circuit simulators as evaluation engines [1, 2, 3]. The quality of a design is determined by the degree to which compliance constraints are met and design optimization goals achieved. With the exception of basic constraints, e.g., saturation/linear region, defining such constraints requires an experienced designer to manually craft them. Moreover, specifying design goals, e.g., DC gain, bandwidth, power consumption, etc., is not enough to prevent optimizers from finding a solution meeting those goals, but not reaching a true IC design solution, or when a practical IC design solution is found, it often shows high sensitivity to noise, or process/operating variations.

The work presented in this paper automatically adds constraints on the transistor level that further automate the design task, while reducing the risk of errors and the compliance of the obtained solutions. It consists of enhancing an established method for constraint generation, based on pattern recognition and symmetry detection, and integrate it in the AIDAsoft EDA environment [3].

The paper is organized as follows. In section II, the background and contributions are presented. In section III, the implemented constraint generation method is reviewed. In section IV, the results achieved by the integration within AIDAsoft EDA are discussed. Finally, in section V, the conclusions are drawn.

II. BACKGROUND AND CONTRIBUTIONS

Circuit sizing is in its essence a multi-objective multi-constraint problem where the designer explores tradeoffs between performance measures. The correct specification of design constraints is critical for conducting the optimization process through the solution space.

Even experienced circuit designers, who possess the knowledge to properly define the constraints for a given circuit, may overlook some constraints, either because the designer is not aware of the constraint, or does not consider it to be relevant, or simply by a lapse in the setup. This way the optimizers can take longer to find a solution, become unable to find a solution, or allowing the optimizer to find unsuitable solutions, i.e., solutions that meet the specified constraints but fail in later verifications, forcing expensive redesigns.

A. State-of-the-Art on Automatic Constraint Generation

Automatically adding a complete set of constraints not only reduces setup time, but would make the circuit more robust when considering process and operating variations. With this target in mind, some constraint generation approaches have been proposed by the EDA community. In the Sizing Rules Method (SRM) [4], design patterns are found in the circuit, so that, some sizing rules can be generated without any input from the designer. This reduces setup time and effort, as well as optimization times, as mentioned earlier. The SRM can efficiently capture design knowledge on the technology specific level of transistor groups. Further increasing the SRM capabilities, the Hierarchical Placement Rules (HPR) [5] generates a structural signal flow graph (SSFG) to detect symmetries in the circuit. This SSFG works as a qualitative model of the circuit. Proximity constraints according to the building blocks are also considered. In Matching-Driven Analog Sizing (MARS) [6] further improves the HPR symmetry detection by partitioning the circuit in a core and bias sub-circuits, behavioral signal flow analysis, and by including some single-ended passive devices.

B. Contributions

The goals of this work are set to enhance and implement a variation of the SRM and HPR methods, and also, validate the resulting constraint generation module within AIDA [3]. AIDA is an analog IC design automation framework composed of two main tools: AIDA-C and AIDA-L. AIDA-C targets automatic circuit sizing and optimization. It is based in state-of-the-art multi-objective, multi-constraint optimization techniques and targets highly robust designs [7]. AIDA-L target automatic layout generation, both standalone and in-loop layout-aware circuit sizing. Its optimization based floorplanner [8], uses designer specified knowledge such as symmetry, matching and current flow constraints, that are here automatically generated.

Sizing constraints will be generated and provided to AIDA-C to be considered during the optimization process, proximity symmetry, and current-flow constraints will be provided to AIDA-L for use during the layout generation. Although the proposed method for constraint generation is based on SRM and HPR methods, there are several differences in the implemented approach, such as: (a) a simpler and faster method to circuit pattern recognition; (b) a signal and current flow hybrid graph rather than just signal flow graph; (c) detection of higher priority self-symmetries; and (d) ability to add new patterns or remove/edit existing patterns in the library.

III. Constraint Extraction from a Netlist

The automated constraint generation process is illustrated in Fig. 1. The process starts by parsing the netlist, then a pattern detection phase is executed, which lead to the definition of the pattern constraints (e.g. overdrive voltages, V_{DS} matching between devices, etc.), and also, in this phase the design variables are set as netlist parameters. The next phase is the generation of the circuit's graph that is used for symmetry detection, which lead to the set symmetry constraints to be used in the layout generation. The circuit's graph is also used to generate the current flow constraints for the layout generation. The outputs of the tool are the parameterized netlist, a set of functional constraints (with the respective measure statement to be included in the testbench), the symmetry between devices and the current flows. Both phases are detailed in the next subsections.

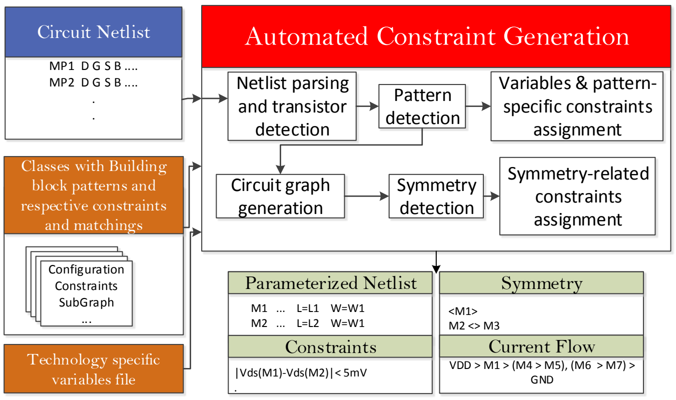

Circuit Netlist
MP1 D G S B ....
MP2 D G S B ....
Classes with Building block patterns and respective constraints and matchings
Configuration
Constraints
SubGraph
...
Technology specific variables file
Automated Constraint Generation
Netlist parsing and transistor detection
Pattern detection
Variables & pattern-specific constraints assignment
Circuit graph generation
Symmetry detection
Symmetry-related constraints assignment
Parameterized Netlist
M1 ... L=L1 W=W1
M2 ... L=L2 W=W1
Symmetry
<M1>
M2 <> M3
Constraints
|Vds(M1)-Vds(M2)|< 5mV
Current Flow
VDD > M1 > (M4 > M5), (M6 > M7) > GND

Figure 1. Automated Constraint Generation architecture

A. Pattern Detection

After reading the netlist and establishing the transistor database, a search for patterns is executed following the 2-level approach in SRM, generating the first level of building blocks of the circuit, as illustrated in Fig. 2. Determining the second level of building blocks is done through a similar step, matching each level 1 building block either to a transistor, or to another level 1 building block to a level 2 pattern.

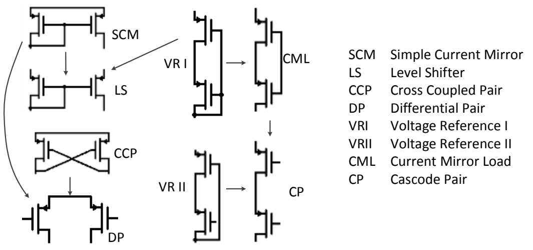


Figure 2. Pattern searching order hierarchy

The design variables and the constraints are determined from the building blocks and the patterns' associated constraints. Proximity constraints are attributed mainly to transistors that are contained in the same building blocks. In addition, the implemented Constraint Generation allows the designer to implement new patterns to describe personalized and/or specific transistor configurations, or change the constraints associated with already existing patterns. Defining a new pattern requires assigning a list of port connections between devices/building blocks, constraints associated with the pattern, and the pattern's current and signal graph.

B. Symmetry Detection

After determining all the building blocks, the current and signal flow graphs are created by merging the subgraphs of the building blocks. Once the graph is generated, a search similar to the HPR approach for symmetric nodes is made. To start a search, a pair of symmetric nodes has to be previously determined. To find a pair of nodes that are considered symmetric, the Differential Pair and Cross-Coupled Pair are considered to be symmetric patterns, and the transistors' gates are the starting pair of symmetric nodes. To find further symmetries, each given symmetric pair is searched for similar edges pointing to (or from) another pair of nodes. Edges are considered similar if they are originated from the same pattern and are of the same transistor "type" (p-mos or n-mos). The process repeats iteratively for the newly found symmetric pair. The method used includes symmetry search in the opposite direction of the edges, which was not implemented in HPR. This aids in determining further symmetries that would otherwise not be detected.

Figure 3 shows the fully differential two stage amplifier found in [9] with the detected patterns and symmetries overlaid. In the shaded regions, and the symmetric pairs in dashed line boxes (boxes with a single transistor represent self-symmetry). The constraint generation processing time took less than one second, and the output was the expected, consisting of the constraints from each detected pattern and symmetry.

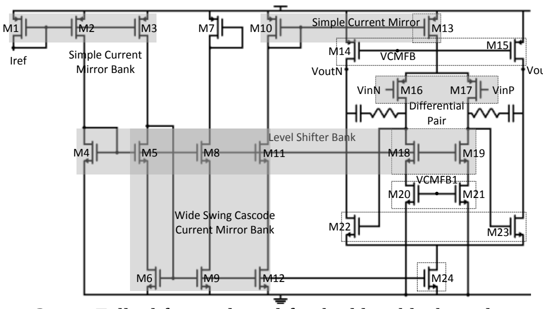


Figure 3. Fully diferential amplifier building blocks and symmetries.

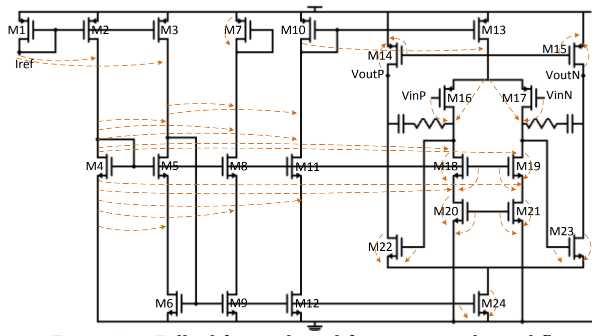


Figure 4. Fully diferential amplifier current and signal flow.

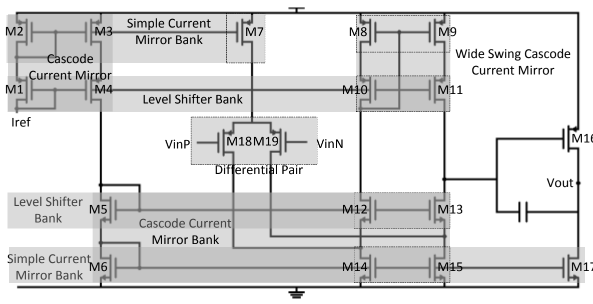


Figure 5. Single ended two-stage folded cascode

C. Integration with AIDA

Adding a circuit to AIDA is a two-step operation. First, the circuit netlist needs to be provided, and then, using AIDA-C's setup assistant to accelerate the process, a file based design structure is created, where the file *design.xml* is the main description [3]. The Automated Constraint Generator reads the original netlist and generates a parameterized netlist, the set of relevant structural constraints (overdrive voltages, deltas, active areas) and the respective measure statements for the simulator being used, symmetries and current flow constraints. The *design.xml* is then completed with the constraints associated to the identified patterns, and, the layout template with the detected symmetry and current flow constraints.

IV. Results

The focus on this paper is to generate constraints based on a given circuit netlist in an automated manner. The results shown focus on the detected patterns, generated constraints and reduced setup time, rather than the computation time used for constraint generation. This is mostly because the design of a circuit is a process that takes days or even weeks, and speeding a module that takes less than one second gives no actual benefit to the designing process. Different types of constraints are also not considered since the designer can choose to customize the constraints associated to the symmetries and to each pattern, along with the fact that previous design projects already include constraints introduced manually by the designer.

A. Optimization results with AIDA-C

The circuit shown in Figure 5 is a two stage folded cascode amplifier, the detected building blocks, in the shaded regions, and the symmetric pairs in dashed line boxes. This example shows that some transistors are part of different building blocks, namely transistors M10 and M11, which belong to the Wide Swing Cascode Current Mirror, and a Level Shifter bank, and thus have their W's and L's matched accordingly, while not creating any conflict in the constraints. The circuit was optimized in AIDA-C project with the automatically generated constraints, and also, using a previous setup with manual constraints that only guarantee saturation and overdrive of the transistors. The optimization goals in both runs were set to maximize gain-bandwidth (GBW) and minimize current consumption (Idd).

Figure 6 shows the Pareto Optimal Fronts (POFs) resulting from both optimizations, where it's clear that the POF from manual setup is wider that the one obtained when considering the automatically generated constraints. This result was expected, as the extra constraints generated by the constraint generation module will create further limitations in the feasible space.

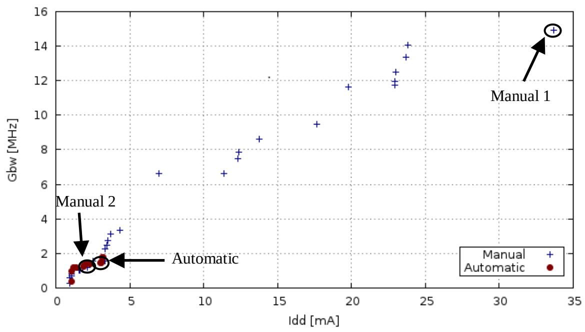


Figure 6. POF achieved with manual and automatic constraints

To compare the quality and robustness of the obtained solutions, the solutions from each POF with larger GBW (Manual 1 and Automatic) were selected and a Monte Carlo (MC) analysis was applied to them. One additional solution from the manual POF was picked, to have similar Idd and GBW characteristics to the automated solution, for the sake of fairness in the analysis. Table I shows the nominal current and gain-bandwidth, the nominal and average offset voltage, the performance that shows larger variability in the MC simulations, and the variance of each of them. With respect to the offset voltage, the most problematic performance in this project after MC, in both manual designs the degradation takes the circuit outside the limit, 5mV, showing both higher average and variance.

TABLE I. Monte Carlo Analysis

	Current cons.		**Gain-Bandwidth**		**Offset Voltage**	
Simulation	**Nominal [mA]**	**Variance [μA]**	**Nominal [MHz]**	**Variance [KHz]**	**Nominal (Average) [mV]**	**Variance [μV]**
Manual 1	33,71	46,22	14,9	109.173,7	0,147(21,79)	288,13
Manual 2	2,486	0,212	1,55	174,02	0.037(9,757)	57,19
Automated	3,175	0,2725	1,77	69,327	0.009(3,032)	5,097

Figure 7 shows the resulting histograms for the 450 MC simulations. The X-axis values were normalized using the performance's nominal values. The histograms show that the circuits sized with manually written constraints are more sensible to variations when compared to the automatically generated constraints.

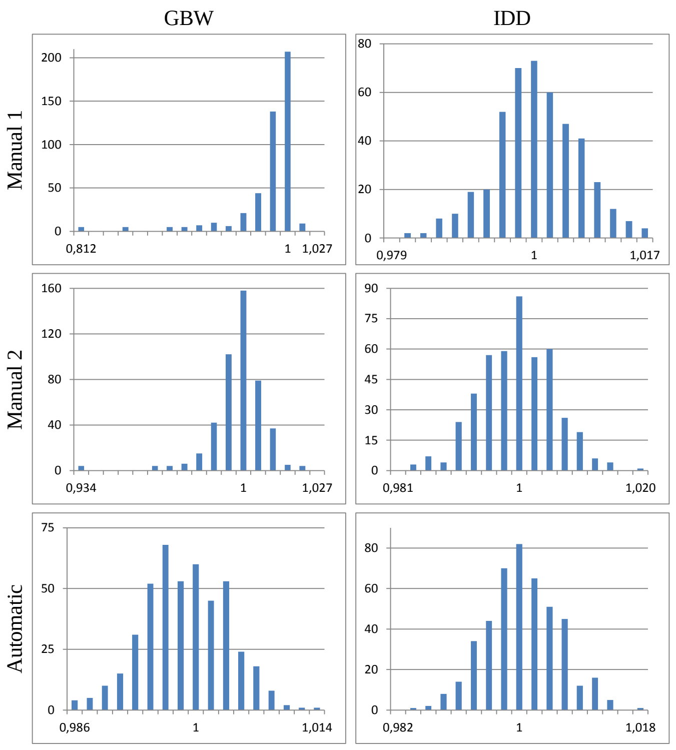
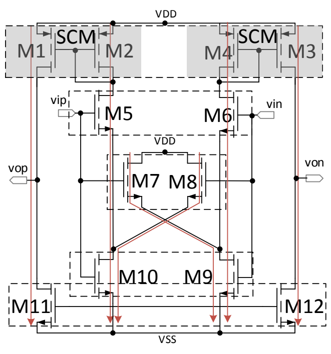


Figure 7. GBW and IDD histograms for 450 Monte Carlo simulations.

B. *Layout Generation with AIDA-L*

The fully deferential OTA from [10], whose schematic is show in Figure 8 (a), has a differential structure without the differential pair, thus making the Differential Pair pattern undetectable, leading to the lack of detection of any symmetric pattern. However, even if no symmetry from building blocks was found, there are several ways to determine symmetric nodes (indicated by the designer, parsed by the module, specific node names, etc.). In this example, both input nodes were considered symmetric, and once the two nodes are determined to be symmetric, the circuit symmetries are found from the transistors (the basic building block) sub-graph. Figure 8 (a) also shows the detected symmetries and current flows. In Figure 8 (b), a floorplan obtained from AIDA-L optimization-based Placer using the detected symmetries and current flow constraints [8].



(a)

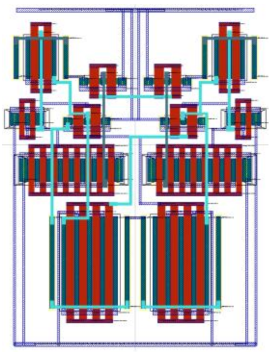

(b)

Figure 8. Fully Differential OTA (a) shcematic and detected symetries and patterns; (b) floorplan obtained from AIDA-L optimization-based Placer, with routing.

V. CONCLUSIONS

This paper presents an enhanced method in automatic analog IC constraint determination. The developed module finds patterns efficiently and accurately, finds symmetries and self-symmetries, generates constraints associated with each pattern and symmetry, which are suitable to be sent to the sizing and layout automation tools. The methodology was tested, and it showed capability in determining the building blocks and symmetries and in generating constraints, even in circuits with a very low ratio of transistors with assigned patterns. One of the circuits was run for a full optimization project, to compare previously written constraints with the newly generated ones, and despite reduced range of solutions in the nominal POF obtained from the more stringent set of constraints, they were more robust when considering variability. Furthering the relevance of automatic constraints, one other example was considered for fully automatic layout generation using the automatically extracted symmetries and current flow constraints together with AIDA-L.

ACKNOWLEDGEMENT

This work was supported in part by the Instituto de Telecomunicações (Research project OPERA PEST-OE/EEI/LA0008/2013) and by the Fundação para a Ciência e Tecnologia (Research project DISRUPTIVE EXCL/EEI-ELC/0261/2012, Research project UID/EEA/50008/2013 and SFRH/BPD/104648/2014**).**

REFERENCES

[1] E. Roca, M. Velasco-Jiménez, R. Castro-López and F. V. Fernández, “Context-dependent transformation of Pareto-optimal performance fronts of operational amplifiers”, *Analog Integrated Circuits and Signal Processing*, vol. 73, no. 1, pp. 65-76, 2012.

[2] H. Gupta, and B. Ghosh, “Analog Circuits Design Using Ant Colony Optimization”, *International Journal of Electronics, Computer & Communications Technologies*, vol. 2, no. 3, pp. 9-21, 2012.

[3] R. Martins, N. Lourenço, A. Canelas, R. Póvoa and, N. Horta, “AIDA: Robust Layout-Aware Synthesis of Analog ICs including Sizing and Layout Generation”, *Int. Conference on SMACD*, pp. 1-4, 2015.

[4] T. Massier, H. Graeb and U. Schlichtmann, "The Sizing Rules Method for CMOS and Bipolar Analog Integrated Circuit Synthesis," *IEEE Transactions on Compututer-Aided Design of Integrated Circuits and Systems (IEEE TCAD)*, vol. 27, no. 12, pp. 2209-2222, December 2008.

[5] M. Eick, M. Strasser, K. Lu, U. Schlichtmann and H. Graeb, "Comprehensive Generation of Hierachical Placement Rules for Analog Integrated Circuits," *IEEE Transactions on Compututer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 2, pp. 180-193, Feb. 2011.

[6] M. Eick and H. Graeb, "MARS: Matching-Driven Analog Sizing," *IEEE TCAD*, vol. 31, no. 8, pp. 1145-1158, August 2013.

[7] N. Lourenço, A. Canelas, R. Póvoa, R. Martins and N. Horta, "Floorplan-aware analog IC sizing and optimization based on topological constraints," *Integration, the VLSI Journal, Elsevier*, p. (in press), 2014.

[8] R. Martins, R. Póvoa, N. Lourenço, and N. Horta, "Exploring Design Tradeoffs in Analog IC Placement with Current-Flow & Current-Density Considerations", *International Conference on SMACD*, pp. 1-4, 2015.

[9] E. Santin, L. Oliveira, B. Nowacki, and J. Goes, “A Fully Integrated and Reconfigurable Architecture for Coherent Self-Testing of High Speed Analog-to-Digital Converters”, *IEEE Transactions on Circuits and Systems – I*, vol. 58, no. 7, pp. 1531-1541, Jul. 2011

[10] R Póvoa, N. Lourenço, N. Horta, R Santos-Tavares, J. Goes, “Single-Stage Amplifirers with Gain Enhancement and Improved Energy-Efficiency employing Voltage-Combiners”, *Very Large Scale Integration (VLSI-SoC), 2013 IFIP/IEEE 21st International Conference on*, pp. 19-22, Oct 2013.