

Synthesis of CMOS Analog Cells using AMIGO

Ramy Iskander*, Mohamed Dessouky, Maie Aly, Mahmoud Magdy, Noha Hassan,
Noha Soliman, Sami Moussa

Faculty of Engineering, Ain Shams University, 1 El-Saray St., 11517 Abbaseya, Cairo, Egypt

*Mentor Graphics Corporation, 51 Beirut Street, 11341 Heliopolis, Cairo, Egypt
mohamed.dessouky@ieee.org

Abstract

In this paper, a simulation-based synthesis tool, AMIGO, for analog cell sizing is presented. AMIGO is based upon genetic optimization techniques adapted to circuit sizing. A framework has been developed using TCL/TK language that allows the designer to set the optimization problem, define complex constraint functions, watch the progress of optimization, and finally view results. To increase design reliability a sizing-rule pre-processor is incorporated in the tool to automatically generate topology related constraints specific to analog building blocks. Different approaches of using circuit optimizers are demonstrated through the synthesis of three different analog cells: a latched-type comparator, a folded cascode opamp and a switched-capacitor integrator. AMIGO showed to be a successful synthesis tool that can be part of a more general synthesis/migration flow.

1. Introduction

Due to the advent of mixed-signal circuit design, and the need to integrate both digital and analog blocks on the same SOC, analog circuit synthesis has become a hot research topic during the past few years. Synthesis comprises two steps: topology selection and sizing. Topology selection means selecting the appropriate circuit topology from a library of topologies. Sizing consists of choosing appropriate transistor dimensions and biasing voltages to satisfy a given set of performance specifications. Topology selection has proven very difficult to automate due to its knowledge-intensive nature. In this paper, we focus on the sizing problem.

Many attempts have been made in order to mimic the designer's expertise and knowledge into automation tools [1-4]. Synthesis tools are categorized into two different types: the *knowledge-based stream* [1] and the *optimization-based stream* [2-4]. In the knowledge-based

stream, the designer extracts design equations and integrates them into the tool to be reused for the same topology. In the optimization-based approach, the optimizer searches the design space for the circuit that satisfies certain constraints and minimizes certain objectives. The optimization-based approach was further divided into two approaches: *equation-based optimization* [3] and *simulation-based optimization* [2]. In the equation-based optimization, circuit evaluation is done through pre-derived equations for performance specifications, initially extracted by the designer or by symbolic analysis [4]. In the simulation-based optimization, the specifications are directly measured from the output waveforms of a simulator. The simulation-based approach has two major advantages over the equation-based approach:

- Accurate simulation models are used instead of approximate equations
- No long preparatory effort to extract all the describing equations. Practically, the extraction may rely fully on the simulator capabilities.

2. AMIGO

AMIGO stands for Analog Migration for ICs with Genetic Optimization. AMIGO is a simulation-based circuit-sizing tool based on genetic optimization techniques [5]. AMIGO sizes any arbitrary topology according to a set of performance specifications. The designer chooses the devices, their parameters to be changed, and the inter-dependency between those parameters. Then, the tool explores the design space formed by those parameters and selects the design point that best meets all the required specifications.

AMIGO was initially intended for analog migration of analog IPs. Through iterative refinement and development of the tool, it became clear that the tool should target both synthesis and migration. Currently the tool performs both tasks. In this paper, we will focus on the synthesis task.

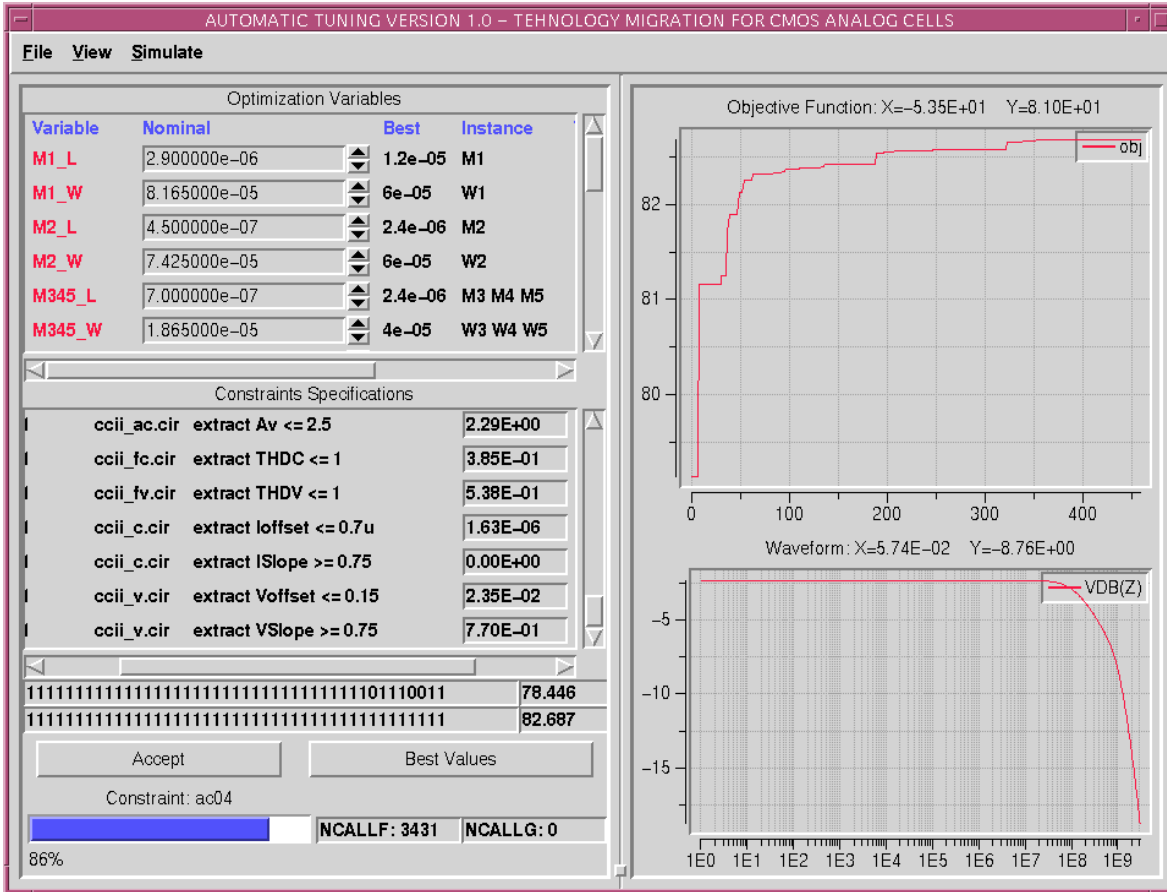


Figure 1: AMIGO analog synthesis tool.

Fig. 1 shows the tune interface of AMIGO. The *Optimization Variables* pane shows all the independent variables with their relevant information. The *Constraints Specifications* pane shows the designer's supplied constraints and detailed information for each constraint. The *Objective Function* pane shows the progression of the cost function viewed as a maximization problem. The *Waveform* pane allows the designer to view any output waveforms.

The tuner interface has two modes of operations: *manual* and *automatic*. In the manual mode, the designer specifies the variables values and then asks the tuner to evaluate all the constraints. By proceeding in this way, the designer can study the different tolerances in any given design. In the automatic mode, the designer fires the optimization engine to search for a solution in the design space. During optimization, the designer can monitor the progression of the optimization process through the previously specified panes. We tried to keep the interface as simple as possible in an attempt to simplify the complexity of the process to the designer.

3. Optimization Components

3.1. Inputs

The input to AMIGO is an optimization file that describes all the information needed to guide the synthesis process. The input information is categorized as follows:

- **Independent variables:**

The designer specifies all the devices and the parameters that should be changed during optimization. Each variable is a discrete one that changes within a range using a step increment value.

- **Variables dependency:**

Many devices, like matched transistors, need to assume exactly the same values during optimization. This dependency can be expressed in the input file.

- **Constraints:**

Constraints on performance specifications can be labeled and expressed using a simple syntax. The constraint statement describes the netlist to use during evaluation and the constraint expression to calculate. The expression

can rely on simulator capabilities, or it can be a user-defined function that is programmed by the designer to measure the specification. A framework has been developed to allow the design to define complex functions using TCL/TK language [6]. This increases the abstraction of the functions and ensures their independence from the simulator.

3.2. Simulation

The tool uses an interface developed over EXPECT [6] to encapsulate all the internal aspects of a simulation. This way any simulator can be encapsulated and incorporated into the optimization environment very easily.

Any waveform can be viewed on the tuner interface. Inspecting the outputs can be beneficial especially during the manual mode. Also, the convergence of the outputs to a given behavior can be inspected during automatic mode.

3.3. Automatic generation of topology-specific constraints

In earlier versions, the designer had to manually write all constraints in an *optimization* input file. Now, this file is generated using the *sizing rules method* [7]. Sizing rules method is the derivation of a list of constraints that must be satisfied to guarantee the function and reliability of any circuit.

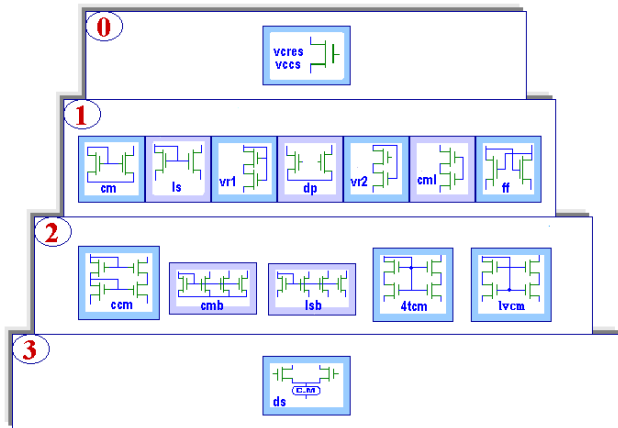


Figure 2: Library of Basic Building Blocks.

Sizing rules method consists of:

1. A hierarchical library of transistor groups defined as the basic building blocks for analog CMOS circuits. Fig. 2. shows the used library.
2. An automatic recognition algorithm of building blocks starting from the circuit netlist.

3. A hierarchical generic list of constraints that must be satisfied to guarantee the function of each block and its reliability. These constraints are automatically added to the input optimization file for each block once the corresponding block is recognized.

Results showed that using sizing rules method has provided an important automation support to analog designers, especially when consistently considering process and operating tolerance and mismatch.

4. Examples

In this section, some examples are given to show the tool efficiency.

4.1. Latched type comparator:

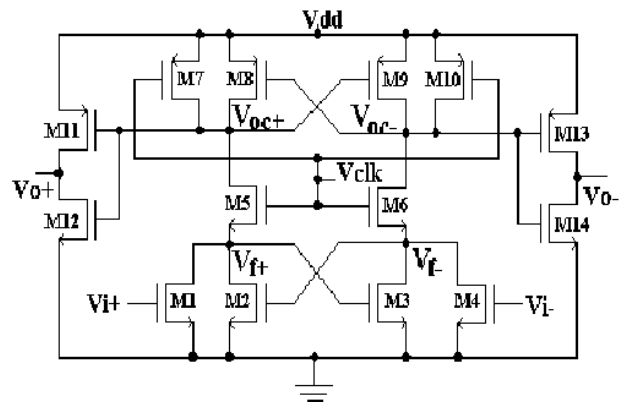


Figure 3: Schematic of the latched comparator.

The comparator circuit shown in Fig. 3 consists of 14 transistors [8]. Each 2 transistors form a pair having the same dimensions. The most important specification of the comparator is its comparison speed. Due to its highly non-linear nature, designers usually adjust the comparator speed through many simulation iterations. Simulation of the comparator was performed with a special input that allows measuring the comparison time at different conditions. Two opposite piece-wise linear inputs were applied to V_i^+ and V_i^- . The differential input ($V_i^+ - V_i^-$) and the clock signal are shown in Fig. 4a.

Optimization constraints were specified directly on the output *waveforms* to guarantee proper operation while satisfying required speed as follows:

The first constraint was used to guarantee that V_o^+ is high only when both the differential input and the clock are high. This is done using a simple logic function evaluated at each simulation time step. Another function was used to guarantee that both outputs have complementary logic levels (Fig. 4b and 4c).

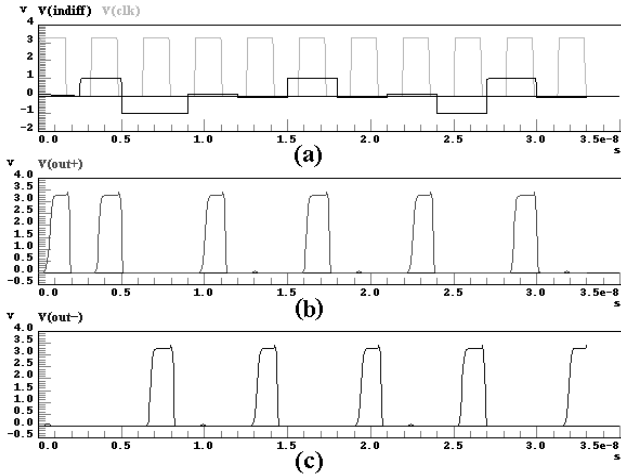


Figure 4: Latched comparator simulation.

One constraint was used to guarantee that the output, when high, would rise to the supply voltage. Another constraint was used to guarantee that V_o^+ would be low (less than 0.1V) when the clock is low or when the differential input is negative. This last constraint allows minimizing transient pulses occurring in the low-level of such kind of comparators as shown in Fig. 5.

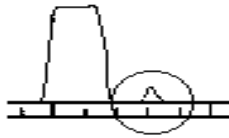


Figure 5: Low-voltage spikes in comparator output.

Finally, constraints were set to improve the performance of the circuit after optimization, which are the comparison time and the transistors area. The total number of constraints reached 30 constraints. The number of design variables was 7 variables, which are the widths of the 7 transistor pairs. The length of all transistors was fixed to simplify the optimization process. The comparator was optimized for a 0.4ns comparison time. Optimization results are shown in Table 1.

All transistors' L	0.6 μ
W (M1, M4)	13.9 μ
W (M2, M3)	65.05 μ
W (M5, M6)	65.95 μ
W (M7, M10)	85.35 μ
W (M8, M9)	86.3 μ
W (M11, M13)	48.3 μ
W (M12, M14)	40.65 μ

Table 1: Optimization results of the comparator.

4.2. Folded cascode opamp:

The folded cascode circuit shown in Fig. 6 is frequently used in switched-capacitor applications due to its relatively simple design, which often leads to a high gain and gain-bandwidth product. The bias voltages of the opamp are derived from the bias circuit shown in Fig. 7. In this example, the opamp is optimized for use in a low-voltage delta-sigma modulator. The phase margin must be chosen to provide a good settling behavior. Good slew rate (SR) performance is needed to avoid nonlinear settling due to fast changes at the output of the integrators. Low noise and low total harmonic distortion (THD) are required, especially for the first integrator stage of the modulator. When trying to optimize for good matching, differential pairs need low overdrive voltage while for current mirrors this is preferably a high one. The other hand, high-speed operation calls for high overdrive voltages and small transistor lengths. The noise performance requires wide and long transistors with high transconductances [9]. Combining also the requirement of low-voltage operation, we easily end up with a jungle of conflicting requirements because we must also take into account linear voltage swings, common-mode ranges, etc.

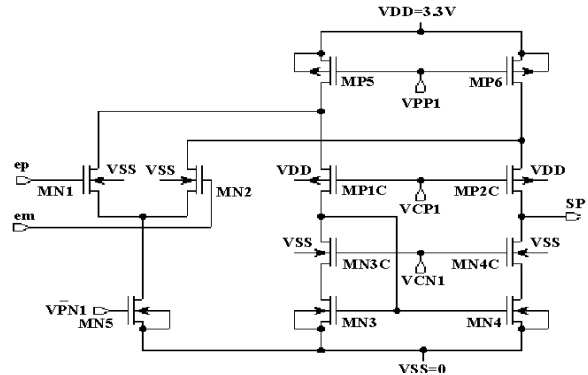


Figure 6: Schematic of the folded cascode opamp.

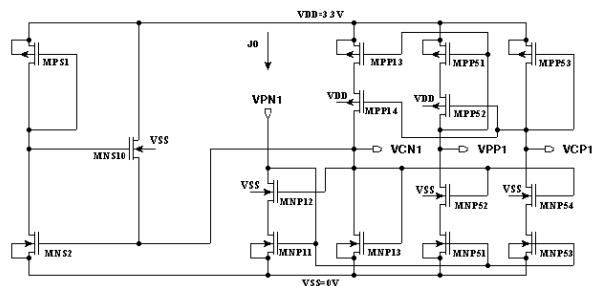


Figure 7: Circuit used to bias the opamp.

Design variables are device widths and lengths. At first glance, we expect the design variables to be twice the number of the transistors in the cell (W & L of each transistor). Fortunately, setting constraints on pairs of transistors to have the same width and length reduced the number of design variables. Furthermore, for simplicity purposes, we assumed that all the transistors in the cell, plus most of the transistors of the bias circuit, have the same length.

One more variable has been added (which is the input offset voltage) in order to avoid the computational effort of calculating a new offset voltage each time a new candidate is visited during the sizing run. By putting it as a variable, and adding a constraint for the DC output to be half the value of VDD, we ensure a correct calculation of the operating point at each simulation for each running candidate. The total number of variables reached 27 variables.

Topological constraints were automatically extracted using the sizing rule method described in section 3.3. For example, for each transistor 2 constraints were used to ensure that it is on and in the saturation region ($V_{GS} \geq V_{TH}$, and $V_{DS} \geq V_{DSAT}$). This set of constraints counts up to 52 constraints.

The other set of constraints were used to achieve the required performance. For each performance a simulation test circuit must be used to evaluate it at each visited solution during optimization. As an example, Fig. 8 shows the test circuit used to evaluate the opamp settling time. The total number of constraints reached 128 constraints. It took 200 simulations to reach a solution. The design constraints together with the results of optimization are listed in Table 2. Optimization CPU time was around three hours on an ULTRA-10, 433MHz, 128MB machine.

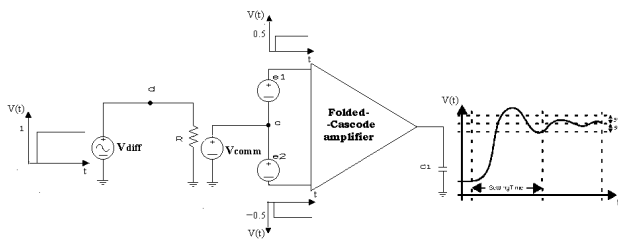


Figure 8: Settling time test circuit.

	Constraint	Result
DC output (V)	$= 1.65$	1.649
Voltage swing (V)	≥ 2	2.03
FT (MHz)	≥ 9.3	11.2
Ado (dB)	≥ 70	75
Aco (dB)	≤ -26	-46

CMRR (dB)	≥ 96	121
Phase Margin	≥ 75	76.4
Slew rate (V/ μ s)	≥ 15	15.2
Settling time (ns)	≤ 157	142
Input noise (μ V)	≤ 75	75
Output noise (μ V)	≤ 0.048	0.082
Power (mW)	≤ 1	0.34
PSRR_VDD (dB)	-	87.4
PSRR_VSS (dB)	-	90.2

Table 2: Optimization results of the opamp.

4.3. Switched capacitor integrator:

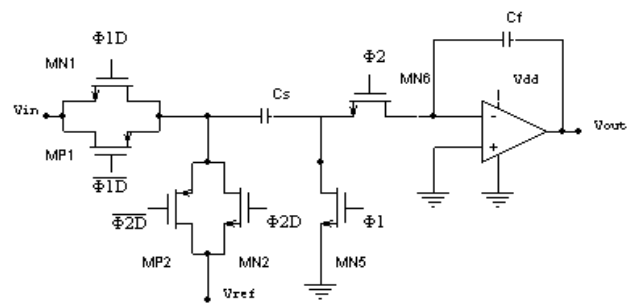


Figure 9: Switched capacitor integrator.

In this section we are going to introduce the optimization of the switched-capacitor integrator shown in Fig. 9. The design of such circuits is a challenging task especially in sizing the MOS switches. Two major mechanisms in MOS transistor operation introduce errors at the instant the switch turns off. They are channel charge injection and clock feed through [9]. Both effects lead to a trade-off between speed and precision.

Integrator optimization was done using two different approaches. First a hierarchical technique is investigated by employing an opamp macro-model, shown in Fig. 10 [10]. The amplifier model parameters (BW, Av, etc) are supplied to the optimization tool as design variables together with MOS switch sizes. Thus the performance of the amplifier is specified while optimizing the integrator. Later, the amplifier can be optimized separately as in the previous section.



Figure 10: The amplifier model.

The waveforms of the output of the integrator for the initial design and after optimization are shown in Fig. 11 for a clock frequency of 1.28 MHz. It's clear that the optimization solved the problem of channel charge injection that existed in the initial design. Optimization output is shown in Table 3. This optimization took 578 simulation calls. CPU time was around 6 hours on an ULTRA-10, 433MHz, 128MB machine.

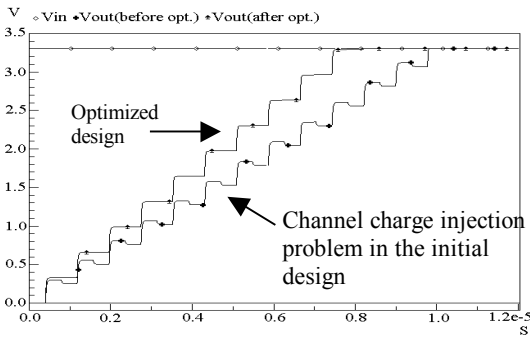


Figure 11: Integrator output using opamp model.

Parameter	Before optimization	After optimization
Wn1 (μm)	4	18.2
Wn2 (μm)	4	6.8
Wn5 (μm)	4	15.2
Wn6 (μm)	4	8.9
Wp1 (μm)	8	9.25
Wp2 (μm)	8	5.15
Av (dB)	100	105
BW (MHz)	100	90
SR (V/ μsec)	5	3.3
Cin (pF)	1.4	8
Rout (ohms)	75	105

Table 3: Optimization results of the integrator.

The second technique was to optimize the amplifier circuit (+bias) and the integrator in a flat transistor-level netlist. Here, the folded cascode opamp (Fig. 6) was used as the amplifier. Instead of optimizing on all amplifier specifications, the overall integrator settling was the only performance under consideration. The integrator output is shown in Fig. 12. In this case 43 parameters and 86 constraints were used during the optimization. It took 2230 simulations to reach a solution. Optimization CPU time was around 23 hours on an ULTRA-10, 433MHz, 128MB machine.

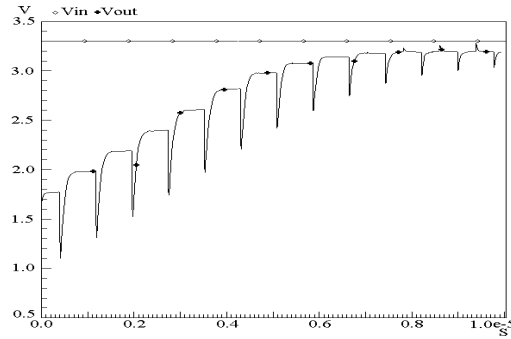


Figure 12: Integrator using transistor-level opamp.

5. Conclusions

In this paper, the analog cell optimization tool, AMIGO, was presented. The tool uses the genetic algorithm as the main optimization engine. Circuit evaluation is performed by calling the simulator at each visited circuit design point. Analog-specific topological constraints are automatically extracted using a special pre-processor. This allows the designer to focus on circuit specifications while increasing circuit reliability.

Different examples were given to show the effectiveness of the tool. Also, different approaches of using a circuit optimizer were demonstrated.

6. References

- [1] M. Degrauwe *et al.*. IDAC: An Interactive Design Tool for Analog CMOS Circuits. *IEEE J. Solid-State Circuits*, 22:1106–1115, Dec. 1987.
- [2] E. Ochotta, R. Rutenbar, and L. R. Carley. Synthesis of High-Performance analog circuits in ASTRX/OBLX. *IEEE Trans. Computer-Aided Design*, 15:273–294, Mar. 1996.
- [3] G. Van der Plas, G. Debyser, F. Leyn, K. Lampaert, J. Vandebussche, G. Gielen, W. Sansen, P. Veselinovic, and D. Leenaerts. AMGIE – A Synthesis Environment for CMOS Analog Integrated Circuits. *IEEE Trans. Computer-Aided Design*, 20(9), Sept. 2001.
- [4] Georges Gielen and Willy Sansen. *Symbolic Analysis for Automated Design of Analog Integrated Circuits*. Kluwer Academic Publishers, 1991.
- [5] David E. Goldberg. *Genetic Algorithms in Search Optimization and Machine Learning*. Addison-Wesley, 1989.
- [6] John K. Ousterhout, editor. *Tcl and the Tk tool kit*. Addison-Wesley Publishing Company, 1994.
- [7] H. Graeb, S. Zizala, J. Eckmueller, K. Antreich. The Sizing Rules Method for Analog Integrated Circuit Design. In *Proc. ICCAD*, 2001.
- [8] A. Yukawa. 1 CMOS 8-bit High-Speed Converter. *IEEE Journal Solid-State Circuits*, 20, Jun. 1985.
- [9] Behzad Razavi. *Design of Analog CMOS Integrated Circuits*. McGraw-Hill, Inc., 2000.
- [10] J. Alvin Connelly, and Pyung Choi. *Macromodeling with Spice*. Prentice Hall, 1992.