Chapter 4
Optimization and Co-simulation of an Implantable Telemetry System by Linking System Models to Nonlinear Circuits

Yao Li, Hao Zou, Yasser Moursy, Ramy Iskander, Robert Sobot and Marie-Minerve Louërat

Abstract This chapter presents a platform for modeling, design, optimization, and co-simulation of mixed-signal systems using the SystemC-AMS standard. The platform is based on a bottom-up design and top-down simulation methodologies. In the bottom-up design methodology, an optimizer is inserted to perform a knowledge-aware optimization loop. During the process, a PEANO trajectory is applied for global exploration and the Nelder–Mead simplex optimization method is applied for local refinement. The authors introduce an interface between system-level models and their circuit-level realizations in the proposed platform. Moreover, a transient simulation scheme is proposed to simulate nonlinear dynamic behavior of complete mixed-signal systems. The platform is used to design and verify a low-power CMOS voltage regulator for an implantable telemetry system.

4.1 Introduction

With the development of system on chip (SoC), the increasing complexity of mixed-signal systems makes their simulation and validation a demanding task. There is a trend toward hierarchical analog synthesis, automation, optimization, mixed-signal systems, etc. For most systems, the simulation needs to take into

Y. Li (✉) · H. Zou · Y. Moursy · R. Iskander · M.-M. Louërat
Université Pierre et Marie Curie, Paris, France
e-mail: yao.li@lip6.fr

M.-M. Louërat
e-mail: Marie-Minerve.Louerat@lip6.fr

R. Sobot
The University of Western Ontario, London, ON, Canada
e-mail: Rsobot@uwo.ca

R. Sobot
ENSA/ETIS, University of Cergy-Pontoise, Cergy-Pontoise, France

© Springer International Publishing Switzerland 2015

M. Fakhfakh et al. (eds.), *Computational Intelligence in Analog and Mixed-Signal (AMS) and Radio-Frequency (RF) Circuit Design*,
DOI 10.1007/978-3-319-19872-9_4

account both system and circuit levels, and the challenge is to create a co-simulation environment that allows synchronization and interaction between the two levels. Recently, the Accellera Systems Initiative releases an open source SystemC-AMS [1, 2]. As an extension to the SystemC [3], SystemC-AMS provides an extended set of capabilities for system-level mixed-signal modeling.

Many existing co-simulation approaches are based on SystemC, SystemC-AMS, or SPICE. In [4], co-simulation-refined models with timed data flow (TDF) paradigm of SystemC-AMS are presented. SystemC-AMS acts as master controlling VHDL testbench. In [5], the proposed solution relies on the integration between an instruction set simulator (ISS) and the SystemC simulation kernel to analyze the performances of embedded systems. In [6], it addresses a method for simulator coupling allowing a transient time simulation of SPICE and the mixed-signal language VHDL-AMS within one simulation process. Another attempt to achieve analog and mixed-signal simulation using loose coupling between SystemC and SPICE is presented in [7]. Nevertheless, all of these attempts lack a clear implementation to establish a link between system-level description and circuit-level realization.

This chapter presents a novel co-simulation framework used for modeling, design, and verification of mixed-signal systems based on knowledge-aware optimization engine. The complete system can be described using only the AMS extension of SystemC with some parts described in SPICE netlists. With this method, we can verify the impact of a circuit block (transistor netlist) on the system level. At the same time, the circuit-level non-idealities propagate upward and affect the system-level ideal behavior. In this co-simulation environment, the SystemC-AMS simulation and the circuit SPICE simulation engines are synchronized in order to perform a nonlinear time-domain analysis and to exchange data at the end of each time step.

Moreover, the optimization engine is used to perform an automatic sizing and biasing of the circuit level. It is a fast design space exploration of analog firm intellectual properties (IP). The main contribution is to propose a knowledge-aware optimization approach, instead of knowledge-based synthesis, which assumed that performance equations are provided by the designer for the underlying topology. We replace the performance equations by traditional SPICE-like netlists that are much easier to provide. Besides, the new optimization algorithm is combined with the hierarchical sizing and biasing methodology [8].

In summary, the advantages of the system-level to circuit-level co-simulation and optimization approach can be summarized as follows:

1. Proposing a very fast sizing and biasing engine to implement the analog IPs.
2. Achieving an automation sizing and biasing based on circuit performances.
3. Presenting a transient simulation scheme to allow the simulation of system-level non-conservative ideal models along with conservative non-ideal circuit-level netlists.

4. Basing only on the C/C++ language, our approach can be used both in high-level modeling (SystemC-AMS) and low-level design (SPICE, optimization engine).

The chapter is organized as follows. Section 4.2 describes the co-simulation and optimization platform architecture by introducing the AMS extensions of SystemC and the hierarchical sizing and biasing procedure that are part of the platform. Section 4.3 gives a detailed explanation of the optimization engine. The implantable telemetry system is selected as the case study and shown in Sect. 4.4. The simulation cycle in co-simulation environment is introduced in Sect. 4.5. The simulation results of the circuit in different model are reported in Sect. 4.6. Section 4.7 concludes the demonstrated work.

4.2 Platform Architecture

Figure 4.1 represents the proposed platform architecture to link system models to nonlinear circuit. This platform is composed of a *bottom-up design* path as well as a *top-down simulation* path.

1. The bottom-up design path consists of the following:
   - A SPICE simulator © is used for sizing purpose.
   - The sizing simulator is controlled by the circuit sizing and biasing procedure Ⓐ.
   - An optimizer Ⓖ is called during the **end_of_elaboration** phase of a TDF moduleⒺ, defined by the SystemC standard.
   - The optimizer takes the circuit specifications as input parameters, calls the sizing and biasing procedure, and compares the circuit performances with specifications at each optimization iteration.
   - The whole design procedure provides an optimized, sized circuit to be used in the following top-down simulation.
2. The top-down simulation path consists of the following:
   - The *testbench* B1 instantiates the SystemC-AMS models, generates the stimuli, and monitors the simulation results.
   - The instantiation of TDF models Ⓕ.
   - The circuit simulator control engine Ⓑ is called by the **processing** phase of a TDF module and applies the stimuli to the circuit netlist.
   - A SPICE simulator is used for analyzing the complete circuit netlist behavior.

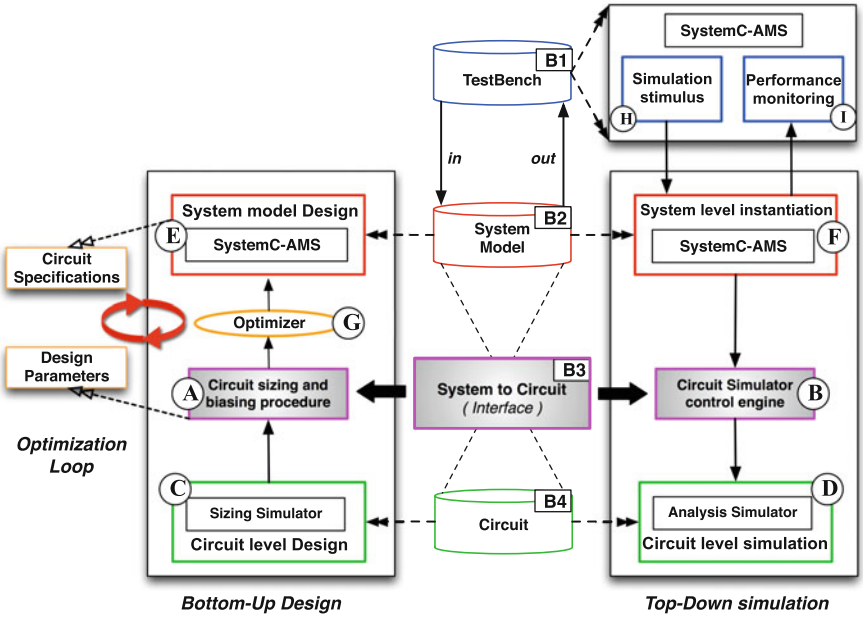

Fig. 4.1 Proposed modeling, design, optimization, and co-simulation platform architecture

As shown in Fig. 4.1, the system to circuit interface B3 consists of two main parts: *the circuit sizing and biasing procedure* Ⓐ and *the circuit simulator control engine* Ⓑ.

A complete system can be described using only the AMS extension of SystemC with some parts described in SPICE netlists. The proposed platform is capable to simulate the whole system with different levels of abstraction. Along with it, the circuit-level non-idealities propagate from upward and affect the system-level ideal behavior.

4.2.1 SystemC-AMS (Analog and Mixed-Signal System Design)

SystemC-AMS [1, 9] provides a framework for functional modeling [10], integration validation, and virtual prototyping [11] of *embedded analog and mixed-signal systems*. SystemC-AMS has three different models of computation: TDF, linear signal flow (LSF), and electrical linear networks (ELN).

Unlike the TDF modeling style, the LSF and ELN modeling styles can only be composed from their own linear primitives. Therefore, in the proposed approach, the TDF model of computation is selected. TDF is a discrete-time modeling style, which considers data as signals sampled in time. These signals are tagged at discrete

points in time and carry discrete or continuous values, such as voltages. Besides, TDF can be used with great efficiency to model complex non-conservative behaviors at system, functional, and macromodel level. Figure 4.2 shows the principle of the TDF modeling. The basic entities found in the TDF model of computation are as follows: the TDF modules, the TDF ports, and the TDF signals. The set of connected TDF modules form a directed graph, called a TDF cluster as defined below:

- TDF modules are the vertices of the graph.
- TDF signals correspond to its arcs.

Each TDF module involved in the cluster contains a specific C++ member function, named **processing()**, that computes a value at each time step.

If enough data samples are available at its input ports, depending on the involved port rates, the samples computed by a TDF module are written to the related output ports and describe continuous-time behaviors.

4.2.2 CHAMS Sizing and Biasing Engine

CHAMS [8, 12, 13] is a tool that provides assistance to the designer for the design of analog firm IP [14, 15]. It allows to generate the analog IP sizing and biasing procedure. It consists of the following three parts: *sizing and biasing operators*, *graph representation*, and *simulator encapsulation*.

4.2.2.1 Sizing and Biasing Operators and Graph Representation

To size and bias a reference transistor, a bipartite directed acyclic graph (DAG) is associated with it. The bipartite graph [16] for the sizing and biasing of the diode-connected transistor using operator OPVGD(V_{EG}) is shown in Fig. 4.3b. A set of input parameters are defined for the diode-connected transistor. The sizing and biasing operator OPVGD(V_{EG}) is then called to compute the set of output parameters.

4.2.2.2 Simulator Encapsulation

Sizing and biasing operators use a specific simulator encapsulation that allows to interface with industrial design kits to ensure very accurate computed results. The simulator encapsulation is illustrated in Fig. 4.4. At the bottom is an electrical netlist that specifies the suitable technology and contains only 2 transistors: one PMOS and one NMOS, entirely sizable and biasable through simulator interactive commands. It is loaded by the electrical simulator launched in interactive mode.

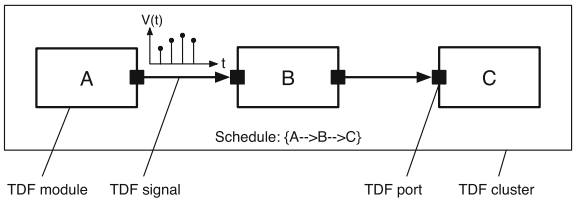

Fig. 4.2 A basic TDF model with 3 TDF modules and 2 TDF signals

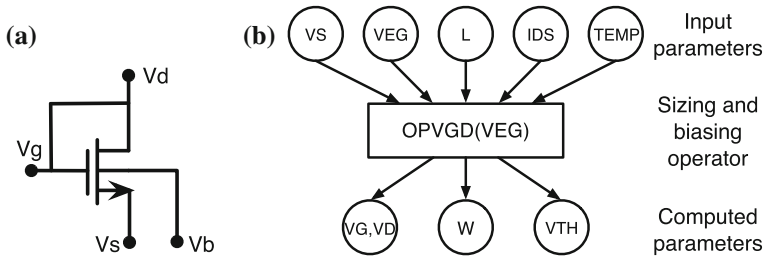

Fig. 4.3 **a** NMOS reference transistor. **b** Graph representing the input parameters and output parameters of the operator OPVGD

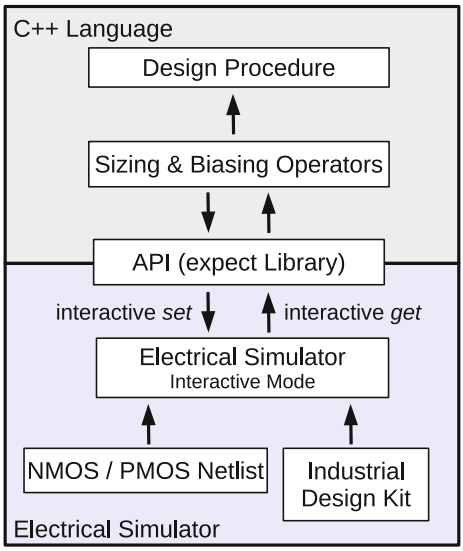

Fig. 4.4 CHAMS sizing engine: electrical simulator encapsulation within sizing and biasing operators

Three types of interactive commands are evaluated: *set*, *get,* and *run*. The first one allows to set all transistor known parameters (sizes and biases) inside the simulator. The second one enables to get all currents, voltages, and small-signal parameters computed by the simulator. After a set command, a simulation must be run using

run command, in order to compute the DC operating point of the transistor. An API is developed using *expect* library [17] to automate *set*, *get,* and *run* commands execution using simulator interactive mode. Sizing and biasing operators are optimized to minimize the number of calls to the simulator, which can reach several thousands during sizing.

4.3 Knowledge-Aware Simulation-Based Optimization Method

Simulation-based synthesis encapsulating a simulator within an optimization loop is presented in Fig. 4.5. Since the simulator is a verification tool, it starts with a set of sizes and biases (vector V2). First, it computes small-signal parameters (vector V3) by evaluating transistor models such as BSIM3v3 [18], BSIM4 [18], PSP [19], and EKV [20]. Second, linear and nonlinear performances (vector V4) are evaluated using a set of testbenches. We point that *performance evaluation* is performed by the simulator, and performances are then compared with the specifications that are specified by the designer.

Generally, the designer would like to use more meaningful design parameters (vector V1) to design analog circuits. The mapping to sizes and biases (vector V2)

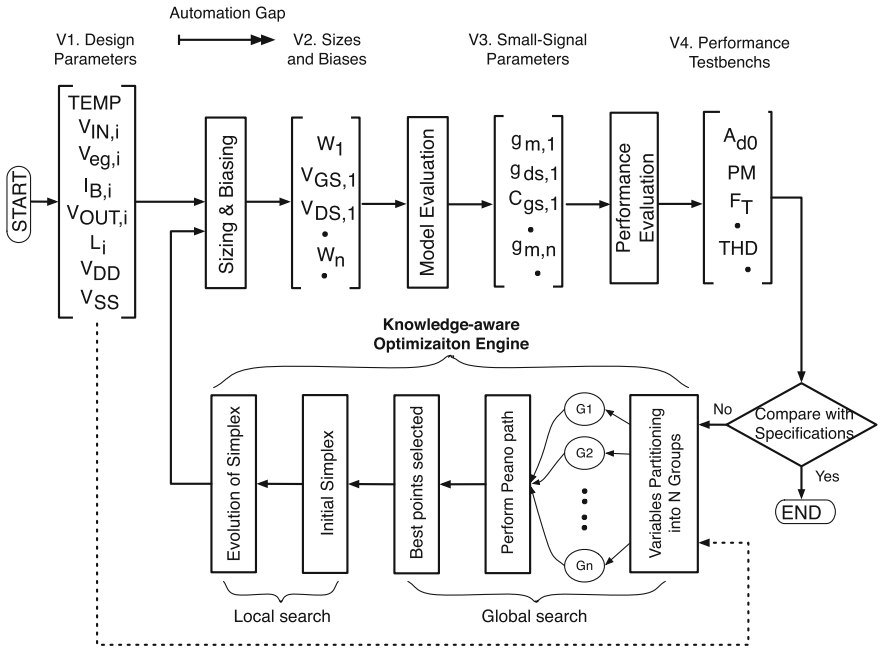

Fig. 4.5 Proposed loop for simulation-based synthesis with circuit optimization engine

becomes a laborious task that has to be repeated for each newly introduced circuit topology. This step depends mainly on the designer expertise and the complexity of circuit topology. Today, this step is not yet formalized; therefore, an *automation gap* is identified in the analog design flow, as illustrated in Fig. 4.5. This use of a formal representation favors the increase of analog design reuse, hence the reduction in design time. The automation gap is filled by generating design procedures using the *hierarchical sizing and biasing methodology*, already presented in the previous section.

Another major point is the performance evaluation. In general, performances are classified into 3 categories: *linear*, *weakly nonlinear,* and *strongly nonlinear*. Linear and weakly nonlinear performances may be easily modeled using mature symbolic analysis techniques [21, 22]. Strongly, nonlinear performances may be modeled using various techniques such as *model-order reduction* [23], *support vector machines* [24], and many others. In [8], we assumed that performance equations were mainly provided by the designer. Therefore, in this work, we propose to use the testbenches for circuit performance evaluation. Besides, we propose a very practical optimization method that is more adapted to the graph presentation as expected in [25].

The architecture of the optimizer is depicted as follows. The optimization variables comprise the set of design parameters chosen by the designer from vector V1. In order to break the curse of dimensionality, a partitioning scheme is selected where the n variables are partitioned into n/p groups of p variables each. Several variable groups are formed, and each group is globally optimized using a PEANO-like path exploration. During this global exploration, the best points are retained. Then, each point is used to start a local search by defining an initial simplex from this starting point and propagating this simplex until a convergence criterion is fulfilled. These schemes are explained in detail in the following sections.

4.3.1 Global Exploration: PEANO Trajectory

The trajectory used during the global search to compute the objective function was invented by the Italian mathematician PEANO [26] to establish a one-to-one correspondence between the number of points on a straight line and the number of points inside a square. This piecewise linear trajectory changes only 1 variable per step, helping optimization engine to converge faster since each point is taken as a prediction for the next one, based on the following Taylor expansion:

$$f(x_{1,\text{next}}, x_2, \ldots) = f(x_{1,\text{prev}}, x_2, \ldots) + \frac{\delta f}{\delta x_1} \cdot (x_{1,\text{next}} - x_{1,\text{prev}}) \tag{4.1}$$

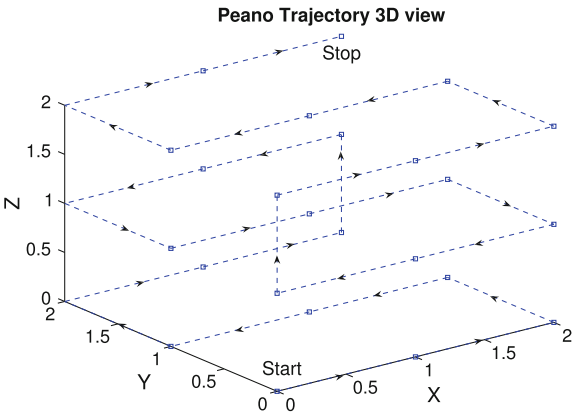

Fig. 4.6 PEANO trajectory for only 3 variables (X, Y, Z)

Figure 4.6 visualizes a PEANO trajectory for 3 variables (X, Y, Z). It is clear from the figure that moving on the PEANO path makes only 1 step change in 1 variable at a time.

4.3.2 *Global Exploration: p Variable Partitioning of an n-Dimensional Design Space ($p \ll n$)*

In order to break the *Curse of dimensionality* described by Richard Bellman in [27], a partitioning scheme for the n-dimensional space is proposed as follows: If we have n variables' optimization problem, we are interested in calculating the objective function at N points of a PEANO trajectory for each variable. In this case, the number of objective function evaluations $N_{\mathrm{OBJ\,1}}$ without partitioning is as follows:

$$N_{\mathrm{OBJ\,1}}(\mathrm{PEANO}, N, n) = N^n \tag{4.2}$$

Let us assume we make a partitioning by dividing randomly the n variables into n/p groups of p variables each. We repeat the partitioning process until a score of M is obtained for each variable. M is defined as the total number of times a given variable appears in all groups. In this case, the number of objective function evaluation $N_{\mathrm{OBJ\,2}}$ with partitioning is

$$N_{\mathrm{OBJ2}}(\mathrm{PEANO}, N, M, n, p) = M \cdot \frac{n}{p} \cdot N^p \tag{4.3}$$

Equations 4.2 and 4.3 indicate that the number of function evaluation provided by the p variable splitting is greatly reduced.

4.3.3 Local Exploration: Nelder–Mead Simplex

The Nelder–Mead simplex algorithm is the most widely used direct search method for solving the unconstrained optimization problem.

$$\min f(x) \tag{4.4}$$

where $f(x)$ is called the objective function. A simplex is a geometric figure in n dimensions that is the convex hull of $n + 1$ vertices. We denote a simplex with vertices $x_1, x_2, \ldots, x_{n+1}$. The vertices satisfy the following relation:

$$f(x_1) \leq f(x_2) \leq \cdots \leq f(x_{n+1}) \tag{4.5}$$

where x_1 refers as the best vertex, and x_{n+1} refers as the worst vertex. We eliminate the worst point of the simplex by using the four possible operations: *reflection*, *expansion*, *contraction*, and *shrink*, which are well defined in [28, 29].

The purpose of the global search is to extract lowest possible value points of the objective function to start the simplex search in a better area of interest. An initial simplex [30] placed symmetrically over these variables is an intuitive and reasonable choice.

4.3.4 The Cost Function

The objective function measures the deviation of the current solution with respect to objectives to minimize and constraints to meet. In our proposed formulation, the objective function is not a weighted function. It is the sum of two 2 types of contributions: *hard constraints* and *soft constraints*. A hard constraint must be satisfied to produce a feasible solution. A soft constraint has no guarantee to be satisfied. It may be minimized as best as possible.

A *hard constraint* is put off through a Heaviside function $H(X)$ whenever it is exceeded, while a *soft constraint* is always active, as long as at least one *hard constraint* is exceeded.

We define the expression for a hard constraint as follows:

$$\text{Hard\_}C = (1 - H(\text{spec}(i) - \text{spec}_{\text{lim}}(i))) \cdot \left(\frac{\text{spec}(i) - \text{spec}_{\text{lim}}(i)}{\text{spec}_{\text{lim}}(i)}\right)^2 \tag{4.6}$$

$H(X) = 1$ if $X \geq 0$, and $H(X) = 0$ if $X < 0$.

We define the expression for a soft constraint as follows:

$$\begin{aligned} \text{Soft\_}C &= \alpha \cdot \left(\frac{\text{spec}(i) - \text{spec}_{\text{lim}}(i)}{\text{spec}_{\text{lim}}(i)} \right)^2 \\ \text{with} \quad \alpha &= \frac{1}{n_{\text{hard}}} \sum_{i=1}^{n_{\text{hard}}} H(\text{spec}(i) - \text{spec}_{\text{lim}}(i)) \end{aligned} \tag{4.7}$$

The general expression for the objective function F_{obj} is as follows:

$$\begin{aligned} F_{\text{obj}} = \alpha \cdot & \sum_{i=1}^{n_{\text{soft}}} \left(\frac{\text{spec}(i) - \text{spec}_{\text{lim}}(i)}{\text{spec}_{\text{lim}}(i)} \right)^2 \\ & + \sum_{i=1}^{n_{\text{hard}}} \left[(1 - H(\text{spec}(i) - \text{spec}_{\text{lim}}(i)) \cdot \left(\frac{\text{spec}(i) - \text{spec}_{\text{lim}}(i)}{\text{spec}_{\text{lim}}(i)} \right)^2 \right] \end{aligned} \tag{4.8}$$

Note: The spec represents the performance extracted from SPICE simulator, while spec_{lim} is the target specification.

n_{soft} is the number of soft constraints, while n_{hard} is the number of hard constraints. The objective function is a summation of squared terms; therefore, its value is minimized when the specification spec reaches its target spec_{lim} at $F_{\text{obj}} = 0$.

4.4 Case Study: Implantable Telemetry System

Recently, methodologies for energy harvesting received extensive attention in the research community and gained significant momentum. Especially, in the case of small animal subjects, rats and mouses in particular, the coupling inductive of RF energy has become the primary method to transmit energy to the implantable telemetry system. However, the level of available internal energy varies by several orders of magnitude at the receiving side because of the subject's movement. Implication is that some form of AC/DC regulation is required for implantable telemetry systems [31].

The case study selected is an analogue IC of an implantable telemetry system. It is a RF-based CMOS voltage regulator for electromagnetic (EM) energy harvesting, which consists of a rectifier/charge pump, a folded-cascode amplifier, and a bandgap voltage reference sub-circuit, as shown in Fig. 4.7.

As the input RF power from the receiver is limited and constantly changing as the receiver moves, it requires the rectifier to be power efficient and the regulated supply to be stable when the given power supply changes. Consequently, it is important to design an efficient implantable voltage regulator that also consumes a minimal amount of energy for its own operation while providing continuous power to the load.

For a wireless power transmission system, the RF power to DC power conversion is realized in a rectifier. The generated steady DC voltage level depends on input RF signal.

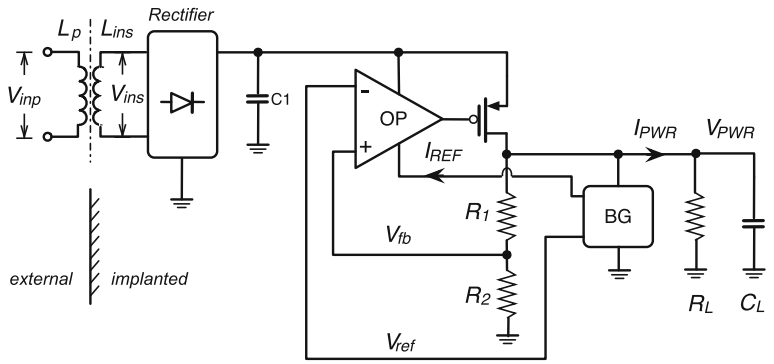

Fig. 4.7 Block diagram of energy-harvesting front-end circuit showing the inductors, rectifier/charge pump, and closed-loop regulation with folded-cascode amplifier (OP in the figure) and bandgap voltage reference (BG in the figure)

The regulation feedback loop is formed by the folded-cascode amplifier, the PMOS driver M_0 and the voltage divider R_1 and R_2 network, which sets the ratio between V_{pwr} and V_{ref} voltages as

$$V_{\text{pwr}} = \left(1 + \frac{R_1}{R_2}\right) \cdot V_{\text{ref}} \tag{4.9}$$

High DC gain in folded-cascode amplifier helps to suppress the difference between the V_{ref} and the feedback voltage.

There are two main purposes to present this circuit:

1. Design: two blocks, bandgap voltage reference, and folded-cascode amplifier are extremely important in order to guarantee the feedback system to work as expected. Hence, we use the optimization engine to design and verify each blocks by meeting their specifications.
2. Simulation: we want to show our proposed platform can be used to co-simulate and verify a complete system that contains a feedback loop by propagating circuit non-idealities to system performances.

4.4.1 Design Process and Model Evolution

Figure 4.8 presents the design process and model evolution of the voltage regulator. It is a top-down structure and can be done in 3 steps:

- **Step 1**: Demonstrating the SystemC-AMS modeling environment, a set of TDF modules (**rectifier, PMOS, bandgap voltage reference, folded-cascode amplifier**) are organized to build the voltage regulator. Each module is

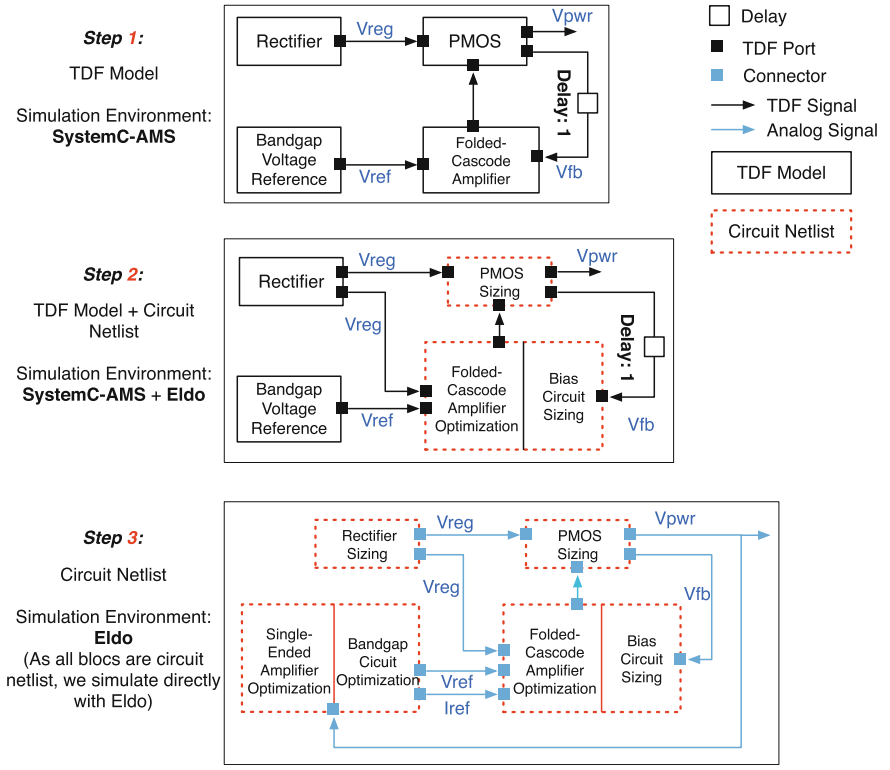

Fig. 4.8 Synthesis flow of the voltage regulator, it is based on design concept, model evolution, optimization, combination, and co-simulation

integrated in a separate file. As shown in Fig. 4.8, the model contains a loop; therefore, a mandatory port delay assignment with delay value 1 (D: 1) has been performed on the output port of *PMOS*. This assignment allows the **folded-cascode amplifier + PMOS** loop to adjust the output signals V_{pwr} and keep it constant. The impact of the insertion of one delay can be neglected as a result of the sampling frequency is very high (500 MHz).

- **Step 2**: Establish a SystemC-AMS, Eldo co-simulation environment. Firstly, we optimize the folded-cascode amplifier by meeting their specifications. And then, we replace the ideal **folded-cascode amplifier** and **PMOS** model with circuit netlist and keep the **rectifier** and **bandgap voltage reference** as ideal model. At last, with the co-simulation platform, we simulate the whole system and propagate the nonlinearities of the folded-cascode amplifier at system level.
- **Step 3**: Optimize the bandgap voltage reference circuit and replace **Rectifier** and **Bandgap voltage reference** model with circuit netlist. Since all the blocs are in circuit netlist, the simulation can be done directly with Eldo.

4.4.2 Folded-Cascode Amplifier

The high DC gain of the operational amplifier can be achieved by using a single-stage folded-cascode structure. The diagram of folded-cascode amplifier is shown in Fig. 4.9. It contains two parts: folded-cascode amplifier and its bias circuit. Biasing voltages, V_2, V_3 and V_4 need to be carefully calculated to ensure that the associated devices operate in the saturation region over the load variation. They are generated by the bias circuit which is associated with the left part of Fig. 4.9.

The sizing procedure of the whole folded-cascode amplifier circuit can be separated into two parts:

1. Firstly, we apply the optimizer engine to optimize the folded-cascode amplifier.
2. Secondly, with the desired biasing voltage (V_2, V_3 and V_4), we size the bias circuit to meet these biasing voltages.

Instead of optimizing the whole circuit, we optimize the core part of the circuit and size the remain part using the sizing and biasing procedure without optimization. Such kind of optimization procedure can dramatically reduce the optimization complexity by decreasing the number of variables.

The sizing and biasing procedure of the folded-cascode amplifier is shown in Fig. 4.10 for a 130-nm process (sizing procedure of the bias circuit is shown in Fig. 4.11). It is a *bipartite graph* [13] that contains the designer's knowledge to size and bias the amplifier. The folded-cascode amplifier is composed of five devices: D_1, D_2, D_3, D_4, D_5 and a transistor M_b. The designer's knowledge is represented by

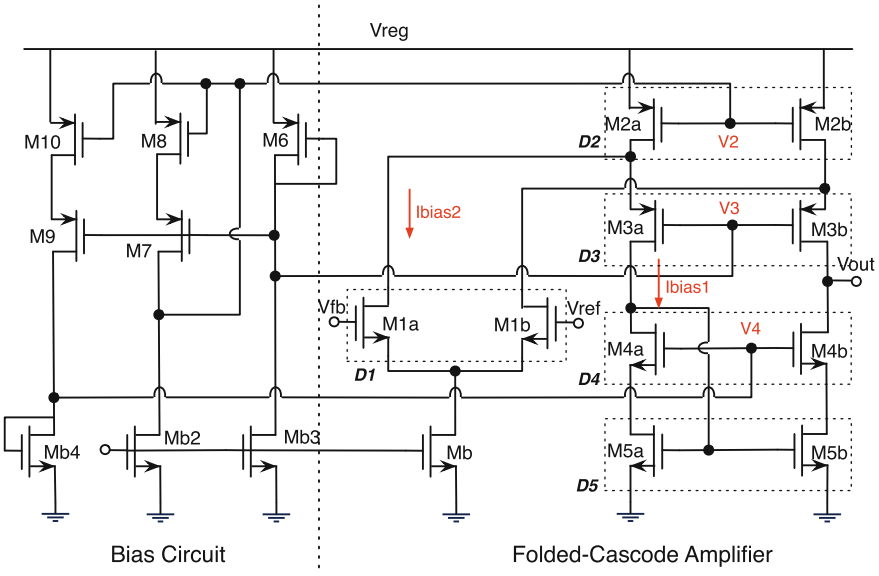

Fig. 4.9 Schematic diagram of the folded-cascode amplifier

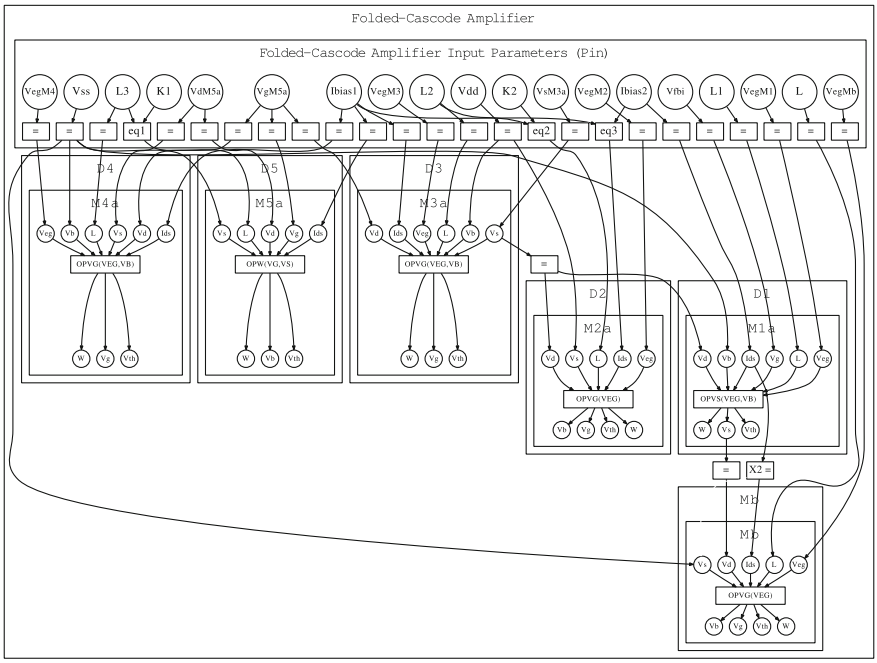

Fig. 4.10 The bipartite graph (i.e., the design procedure) associated with the folded-cascode amplifier. Sizing and biasing operators are part of the bipartite graph

P_{in} set of input parameters (at the top of the graph). Parameters in P_{in} (see in Tables 4.3 and 4.4, these present fixed variables and optimized variables, respectively) are spread in the graph and used by the sizing and biasing operators to compute unknown sizes and biases. Rectangle nodes named "eq" represent designer's defined equations, and an example of equation is given with eq3: $I_{\text{Bias\_}D2} = I_{\text{Bias 1}} + I_{\text{Bias 2}}$, ($I_{\text{Bias 1}}$ and $I_{\text{Bias 2}}$ are given from input parameters, and the result $I_{\text{Bias\_}D2}$ is passed to device D_2). The resulting output parameters P_{out} are listed in Table 4.5. The bipartite graph is a sequence of sizing and biasing operators, and it is evaluated from top to bottom to get the sizes and biases of all transistors.

Figure 4.11 shows the bipartite graph of the bias circuit of the folded-cascode amplifier. The input parameters of the bias circuit are related to the output parameters of the folded-cascode amplifier. In other words, the whole design procedure can be seen as a hierarchical sizing and biasing method.

We want a high gain to avoid any discrepancy in the DC input voltage on positive and negative terminals of the amplifier. Table 4.1 gives the specifications to be met and displays the optimized performances. The global search boundaries for optimizing folded-cascode amplifier are shown in Table 4.2, and nine variables are optimized: 3 lengths (L_{M1a}, L_{M3a}, and L_{M4a}); 4 overdrive voltages ($V_{\text{EG},M4a}$, $V_{\text{EG},M3a}$, $V_{\text{EG},M2a}$, and $V_{\text{EG},M1a}$); 2 currents ($I_{\text{BIAS 1}}$ and $I_{\text{BIAS 2}}$). The search boundary for each variable has been selected arbitrarily.

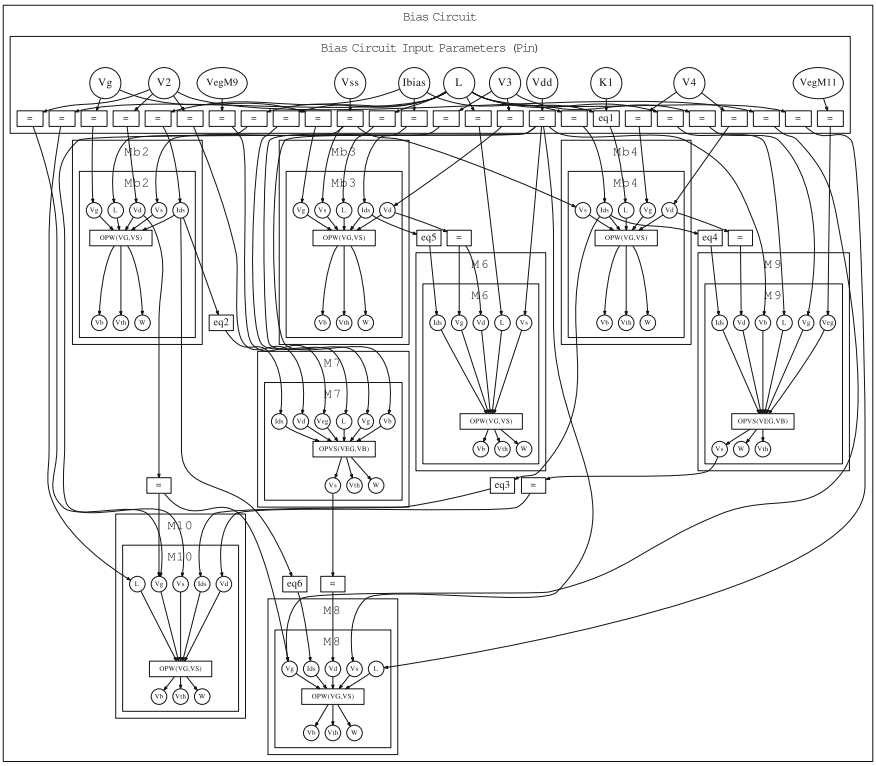

Fig. 4.11 The bipartite graph (i.e., the design procedure) associated with the bias circuit of the folded-cascode amplifier

Table 4.1 Specifications for folded-cascode amplifier circuit in 130-nm technology

Specification	Requirement	Constraint type	Performances
Gain	≥75 dB	Hard	75.2 dB
Unity-gain frequency	≥1 MHz	Hard	1.421 MHz
Phase margin	≥80°	Hard	87.9°
Power (10 kΩ load)	≤10 μW	Soft	10.5 μW

Table 4.2 Global search boundaries for optimizing folded-cascode amplifier

Parameter	Boundaries values	Parameter	Boundaries values
L_{M1a} (μm)	$0.5 \leq L_{M1a} \leq 3$	L_{M3a} (μm)	$0.5 \leq L_{M3a} \leq 3$
L_{M4a} (μm)	$0.5 \leq L_{M4a} \leq 3$	I_{BIAS1} (μA)	$2.0 \leq I_{\mathrm{BIAS1}} \leq 5.0$
I_{BIAS2} (μA)	$2.0 \leq I_{\mathrm{BIAS2}} \leq 5.0$	$V_{\mathrm{EG},M1a}$ (V)	$0.06 \leq V_{\mathrm{EG},M1a} \leq 0.15$
$V_{\mathrm{EG},M2a}$(V)	$0.06 \leq V_{\mathrm{EG},M2a} \leq 0.15$	$V_{\mathrm{EG},M3a}$ (V)	$0.06 \leq V_{\mathrm{EG},M3a} \leq 0.15$
$V_{\mathrm{EG},M4a}$(V)	$0.06 \leq V_{\mathrm{EG},M4a} \leq 0.15$		

Table 4.3 Input parameters (P_{in}) of the folded-cascode amplifier (fixed variables)

Parameter	Value	Parameter	Value	Parameter	Value
V_{DD} (V)	1.2	$V_{D,M5a}$ (V)	0.3	L_{M_b} (μm)	1.0
V_{SS} (V)	0.0	$V_{D,M4a}$ (V)	0.6	V_{EG,M_b} (V)	0.10
V_{REF} (V)	0.585	$V_{S,M3a}$ (V)	0.9	$K_2 = L_{M2a}/L_2$	5
$K_1 = L_{M5a}/L_3$	5				

Table 4.4 Input parameters (P_{in}) of the folded-cascode amplifier (optimized variables)

Parameter	Value	Parameter	Value	Parameter	Value
L_{M1a} (μm)	0.655	L_{M3a} (μm)	0.595	$V_{EG,M2a}$ (V)	0.0766
L_{M4a} (μm)	0.880	$I_{BIAS\,1}$ (μA)	2.071	$V_{EG,M3a}$ (V)	0.0695
$I_{BIAS\,2}$ (μA)	2.198	$V_{EG,M1a}$ (V)	0.0812	$V_{EG,M4a}$ (V)	0.0545

Table 4.5 Computed width for the folded-cascode amplifier (P_{out})

Transistor	Width	Transistor	Width	Transistor	Width
W_{D1} (μm)	1.325	W_{D4} (μm)	4.785	W_{D3} (μm)	3.730
W_{D2} (μm)	38.380	W_{D5} (μm)	5.855	W_{M_b} (μm)	1.405

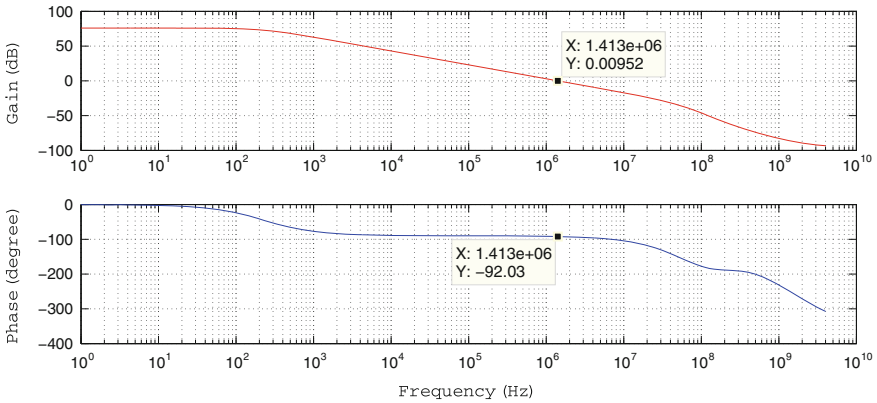

Fig. 4.12 Simulated gain and phase margin of the folded-cascode amplifier

Figure 4.12 illustrates the AC simulation results from optimized circuit. The DC gain is equal to 75.2 dB, phase margin is equal to 87.9°, the transition frequency is 1.421 MHz, and the power consummation is about 10.5 μW with a 10 kΩ loaded resistor. The load capacitance is set to 50 pF.

4.4.3 Bandgap Voltage Reference Circuit

A key target for an integrated voltage reference is to provide adequate temperature stability and high rejection to power supply variations. These features are typically achieved by using a bandgap-based reference.

In this section, we will present a low-voltage low-power temperature-insensitive voltage reference. The schematic diagram of the circuit is presented in Fig. 4.13. To be more specific, an amplifier implements the weighted sum between a complementary to absolute temperature (CTAT) voltage (generated by means of a forward-biased diode) and a proportional to absolute temperature (PTAT) voltage. In the CMOS 0.13 μm process, with this implementation, it is possible to work with power supply voltage as low as 1.0 V. The bandgap voltage reference circuit consists of a single-ended two-stage amplifier, which is detailed in the next sub-section. The optimization of the bandgap voltage reference circuit is done in 2 steps:

1. The first step consists in optimizing the single-ended two-stage amplifier by meeting some specifications.
2. The second step aims at optimizing the bandgap voltage reference circuit structure using the previously optimized amplifier, by sizing the remaining bandgap voltage reference circuit transistors.

4.4.3.1 Single-Ended Two-Stage Amplifier

A single-ended two-stage amplifier is used inside the bandgap voltage reference circuit, as shown in Fig. 4.14. As the emitter–base voltage of Q1 varies from 0.5 to 0.8 V over the full temperature range, an NMOS input differential pair is used in input stage of the amplifier.

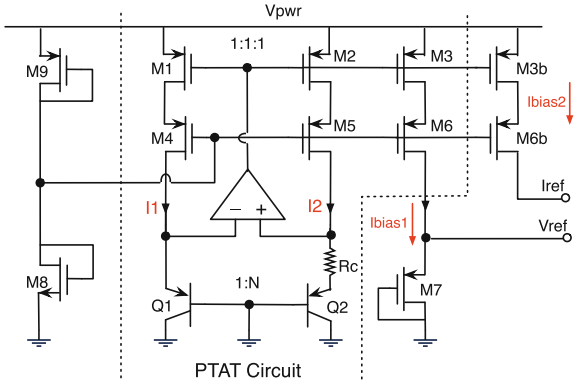

Fig. 4.13 Schematic diagram of bandgap voltage/current reference circuit that consists of PTAT core, V_{ref} generator, I_{ref} generator, and a biasing voltage generator

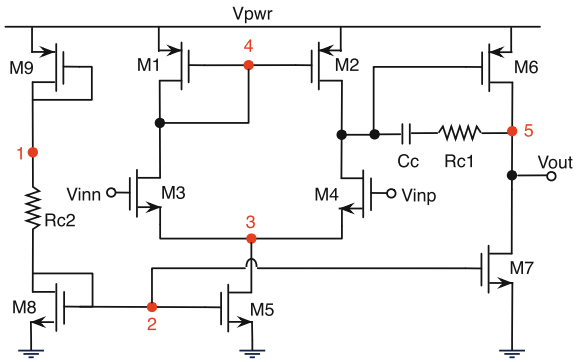

Fig. 4.14 Schematic diagram of the single-ended two-stage amplifier

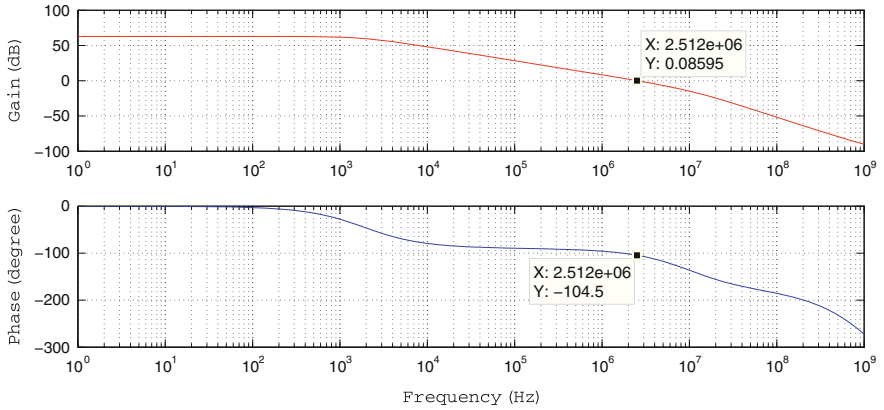

Fig. 4.15 Simulated gain and phase margin of the two-stage amplifier in the PTAT circuit

We use the same methodology presented in Sect. 4.4.2 to optimize this amplifier. With a 50-pF load capacitance, optimization results show that the DC gain is 67.8 dB, phase margin 75.5° with the transition frequency of 2.518 MHz. Figure 4.15 presents the AC simulation results.

4.4.3.2 Temperature Independent of Bandgap Voltage Reference Circuit

In the analysis of bandgap voltage reference circuit, shown in Fig. 4.13, assuming for simplicity that (M_1–M_2–M_3), (M_4–M_5–M_6) are identical pairs, where $I_1 = I_2$, yielding the same behavior for $I_{\mathrm{BIAS\,1}}$, we note that the transistor M_7 works in the saturation region. The current $I_{\mathrm{BIAS\,1}}$ therefore equals

$$I_{\mathrm{BIAS\,1}} = I_{\mathrm{SD}} = \frac{1}{2}\mu_p C_{\mathrm{ox}} \frac{W_7}{L_7} \left(V_{\mathrm{SG}} - |V_{\mathrm{TP}}|\right)^2 \tag{4.10}$$

we get

$$V_{\mathrm{ref}} = V_{\mathrm{SG}} = |V_{\mathrm{TP}}| + \sqrt{\frac{2I_{\mathrm{SD}}}{\mu_p C_{\mathrm{ox}} \frac{W_7}{L_7}}} \tag{4.11}$$

Note: $|V_{\mathrm{TP}}|$ is the PMOS threshold voltage, μ_p is the carrier mobility, Cox is the unit gate oxide capacitance, I_{SD} is the bias current, and (W/L) is the gate width to length ratio. Therefore, a temperature-independent voltage/current reference is required.

In this equation, the I_{SD} and μ_p are two parameters related to the temperature and hence:

$$\begin{aligned} \frac{\partial V_{\mathrm{ref}}}{\partial T} &= \frac{\partial |V_{\mathrm{TP}}|}{\partial T} + \frac{\partial \sqrt{\frac{2I_{\mathrm{SD}}}{\mu_p C_{\mathrm{ox}} \frac{W_7}{L_7}}}}{\partial T} \\ &= \frac{\partial |V_{\mathrm{TP}}|}{\partial T} + \sqrt{\frac{1}{2\, C_{\mathrm{ox}} \mu_p I_{\mathrm{SD}} \frac{W_7}{L_7}}} \cdot \frac{\partial I_{\mathrm{SD}}}{\partial T} - \sqrt{\frac{I_{\mathrm{SD}}}{2\, C_{\mathrm{ox}} \mu_p^3 \frac{W_7}{L_7}}} \cdot \frac{\partial \mu_p}{\partial T} \end{aligned} \tag{4.12}$$

For a bipolar device, we can write $I_C = I_S \exp(V_{\mathrm{BE}}/V_T)$, where $V_T = kT/q$, thus:

$$\begin{aligned} I_2 &= \frac{\Delta V_{\mathrm{EB}}}{R_C} = \frac{V_{\mathrm{BE\,2}} - V_{\mathrm{BE\,1}}}{R_C} \\ &= \frac{V_T \ln \frac{I_{C2}}{I_{S2}} - V_T \ln \frac{I_{C1}}{I_{S1}}}{R_C} = \frac{V_T \ln N}{R_C} = \frac{KT}{q R_C} \ln N \end{aligned} \tag{4.13}$$

Now, returning to Eq. 4.11 and including $\partial I_{\mathrm{SD}}/\partial T$, we have

$$\frac{\partial V_{\mathrm{ref}}}{\partial T} = \frac{\partial |V_{\mathrm{TP}}|}{\partial T} + \frac{1}{g_{m7}} \frac{K \ln N}{q R_C} - \sqrt{\frac{I_{SD}}{2\, C_{\mathrm{ox}} \mu_p^3 \frac{W_7}{L_7}}} \cdot \frac{\partial \mu_p}{\partial T} \tag{4.14}$$

To get a temperature-independent voltage, it should have a positive temperature coefficient as well as a negative temperature coefficient. The above analysis helps to select the global search boundaries for optimizing bandgap voltage reference circuit, as shown in Table 4.6, and five variables are optimized: $I_{\mathrm{BIAS\,1}}$, N, R_C, L, and $V_{M7,\mathrm{VS}}$.

We choose the specifications of the bandgap voltage reference circuit. Firstly, we expect the reference voltage is restricted to a very narrow range between 0.57 and 0.61 V. Secondly, the variation of V_{ref} with temperature (between 0 and 100 °C) in

Table 4.6 Global search boundaries for optimizing bandgap voltage reference circuit

Parameter	Boundaries values	Parameter	Boundaries values
L (μm)	$0.5 \leq L \leq 3$	N	$6 \leq N \leq 15$
R_C (kΩ)	$10 \leq R_C \leq 50$	$V_{M7,\mathrm{VS}}$ (V)	$0.5 \leq V_{M7,\mathrm{VS}} \leq 0.7$
I_{BIAS} (μA)	$1 \leq I_{\mathrm{BIAS}} \leq 4$		

Table 4.7 Specifications for bandgap voltage reference circuit in 130-nm technology

Specification	Mode	Performances	Constraint type
$V_{\max}$ of V_{ref}	Typical	≥0.57 (V)	Hard
$V_{\min}$ of V_{ref}	Typical	≤0.61 (V)	Hard
ΔV_{ref}	Typical	≤0.0005 (V)	Hard
ΔV_{ref}	Corner	≤0.0015 (V)	Hard
Power (10 kΩ load)	Typical	≤10 (μW)	Soft

Table 4.8 Input parameters (P_{in}) of the bandgap voltage reference circuit (fixed variable)

Parameter	Value	Parameter	Value	Parameter	Value
$I_{\mathrm{BIAS\,2}}$ (μA)	4.15	V_{SS} (V)	0.0	V_{M6b,V_S} (V)	0.46
$V_{M6,V_{\mathrm{EG}}}$ (V)	−0.12	V_{DD} (V)	1.0	$K = L_{M8,M9}/L$	5
$V_{M6b,V_{\mathrm{EG}}}$ (V)	−0.12	$V_{M3,V_{\mathrm{EG}}}$ (V)	−0.12		

Table 4.9 Input parameters (P_{in}) of the bandgap voltage reference circuit (optimized variable)

Parameter	Value	Parameter	Value	Parameter	Value
L (μm)	0.325	N	9	$I_{\mathrm{BIAS\,1}}$ (μA)	1.2198
R_C (Ω)	26,610	V_{M7,V_S} (V)	0.5137		

typical mode and corner mode should be less than 0.5 and 1.5 mV, respectively. Thirdly, as we design a low-power circuit, the power dissipation should be less than 10 mW with a 10 kΩ load. Table 4.7 gives the all the specifications to be met.

The sizing and biasing procedure of the bandgap voltage reference circuit is presented in Fig. 4.16, note that the input parameters $I_{\mathrm{BIAS\,2}}$ and V_{M6b,V_S} are inherited from the sizing and biasing procedure of the folded-cascode amplifier, which are equal to $I_{M_b,\mathrm{BIAS}}$ and $V_{M_b,\mathrm{VG}}$' respectively. The computed width values for the bandgap voltage reference circuit are listed in Table 4.10.

4.4.3.3 Simulation Results of the Bandgap Voltage Reference Circuit

The optimization is performed using 3 SPICE netlists to simulate each of the corner cases for the 130-nm technology. In our bandgap voltage reference circuit, we have 3

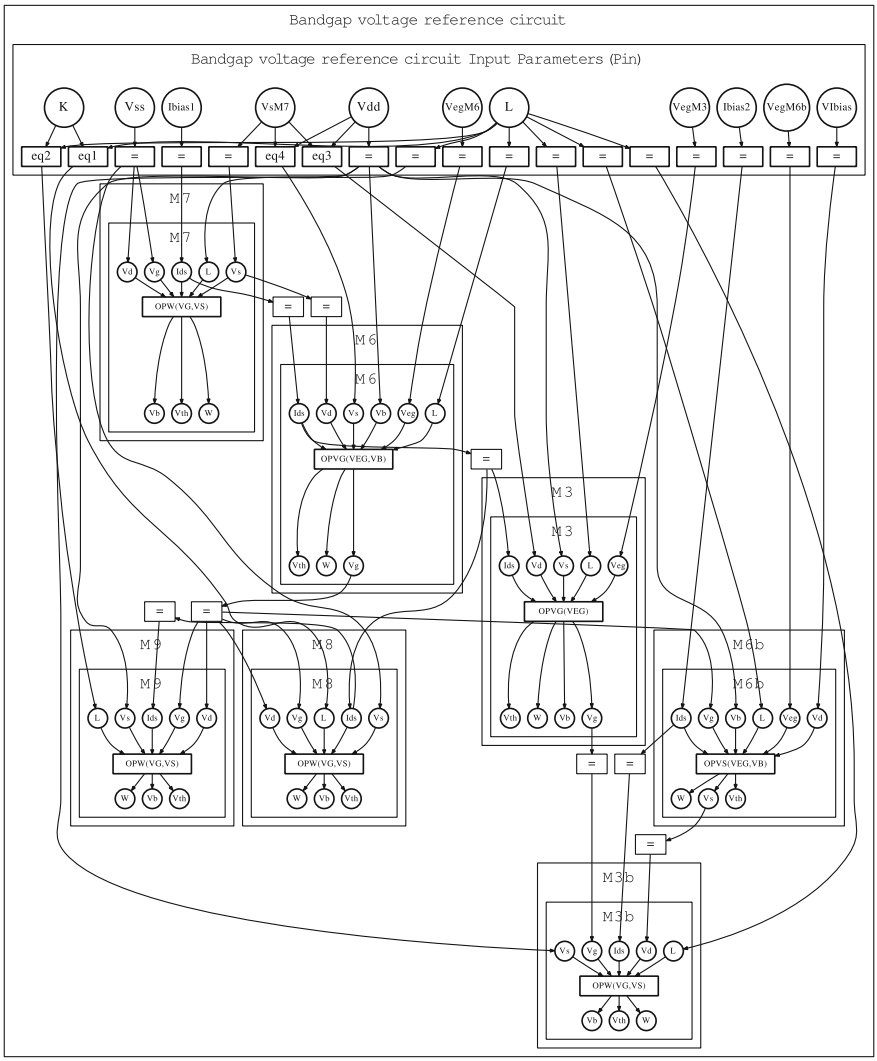

Fig. 4.16 The bipartite graph (i.e., the design procedure) associated with the bandgap voltage reference circuit. Sizing and biasing operators are part of the bipartite graph. Input parameters p_{in} (see Tables 4.8 and 4.9) are on the top of the graph

Table 4.10 Computed width for the bandgap voltage reference circuit (P_{out})

Transistor	Width	Transistor	Width	Transistor	Width
W_{M3} (μm)	0.525	W_{M6} (μm)	0.525	W_{M6b} (μm)	1.695
W_{M7} (μm)	3.235	W_{M8} (μm)	32.055	W_{M3b} (μm)	0.230
W_{M9} (μm)	0.320				

types of components: *N*-type transistor (typical, slow, fast), *P*-type transistor (typical, slow, fast), and bipolar (typical, b_{min}, b_{max}), respectively. Therefore, we chose the SPICE netlists to simulate the corners of these components as follows: the first netlist for (typical, typical, typical), the second one for (fast, fast, bmax), and the third one for (slow, slow, b_{min}). We use SPICE netlist to load specific corners, in order to optimize the circuit process deviation. Actually, there are 27 (3^3) combination of the corner netlist. Here, we keep only three cases, all typical, all slow and all fast.

Figure 4.17a–c represents, respectively, a SPICE DC temperature sweep simulation from 0 to 100 °C. Figure 4.17a represents 3 curves corresponding to 3 sets of parameters (typical, bmin, bmax) for bipolar, while the *P*-type transistor and *N*-type transistor are set to the typical case. Figure 4.17b represents 3 curves corresponding to 3 sets of parameters (typical, bmin, bmax) for bipolar, while the *P*-type transistor and *N*-type transistor are set to the slow case. Figure 4.17c represents 3 curves corresponding to 3 sets of parameters (typical, bmin, bmax) for bipolar, while the *P*-type transistor and *N*-type transistor are set to the fast case. This combination is generated to further verify the electrical behavior of the bandgap voltage reference circuit.

The simulated curves of V_{ref} versus temperature show that in condition of NMOS/PMOS corners are typical, there is a very clear compensation, and the lowest reference voltage point is around 65 °C. In condition of NMOS/PMOS corners are fast, the compensation phenomenon is less obvious. In condition of

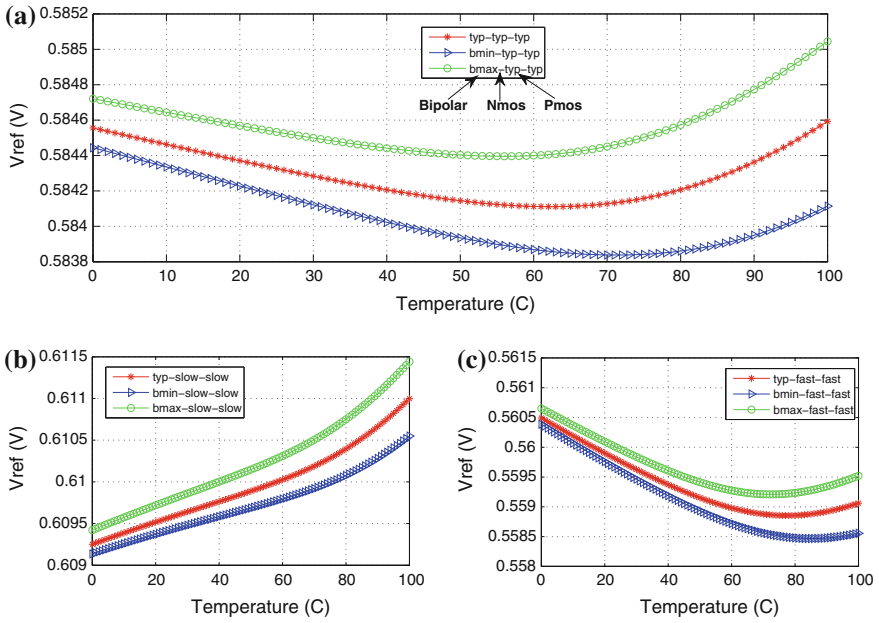

Fig. 4.17 Reference voltage versus temperature: **a** 3 extreme sets of bipolar parameters (typical for NMOS and PMOS). **b** 3 extreme sets of bipolar parameters (slow for NMOS and PMOS). **c** 3 extreme sets of bipolar parameters (fast for NMOS and PMOS)

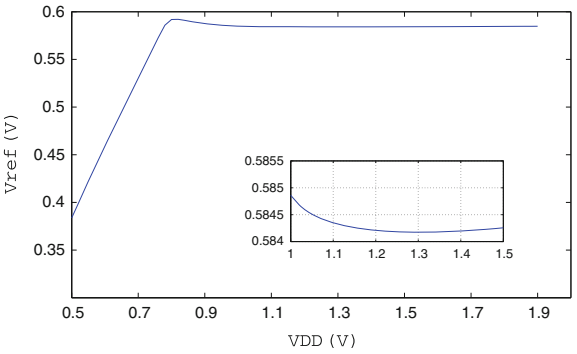

Fig. 4.18 The reference voltage dependence over V_{DD} (Typical, temperature = 37 °C)

NOMS/PMOS corners are slow, there is no compensation phenomenon, but the voltage variation from 0 to 100 °C is still less than 1.5 mv.

The variation of the reference voltage curve when V_{DD} changes from 0.5 to 1.9 V at 37 °C is shown in Fig. 4.18. The inserted plot shows the zoom-in view for V_{DD} between 1 and 1.5 V, which confirmed the circuit can work as low as 1 V.

4.5 Simulation Cycle in Co-simulation Environment

In this section, we explain in detail the simulation cycle in SystemC-AMS, Eldo co-simulation environment, which refers to **step 2** in Fig. 4.8. As shown in Fig. 4.19a, the co-simulation interface is related to the TDF module with circuit netlist. It involves three member functions: **end_of_elaboration()**, **initialize()**, and **processing()**. The **end_of_elaboration()** function calls the optimization engine, which invokes the design procedure at each optimization iteration, and the design procedure computes sizes and biases parameters (W, V_G etc.) from the design parameters such as V_{EG}, I_D, and L. The **initialize()** function sets these sizes and biases variables to circuit netlist. The signal processing function **processing()**, where the circuit netlist into the SPICE simulator is loaded, performs circuit-level transient simulation.

Note that in the member functions **end_of_elaboration()** and **processing()**, call two different simulators, named sizing simulator and analysis simulator. Both of them encapsulate an electrical simulator, mentioned in Sect. 4.2.2.2. The only difference between sizing simulator and analysis simulator is the transistor netlist loaded by the electrical simulator. The sizing simulator contains only two transistors: one PMOS and one NMOS, while the analysis simulator includes the complete circuit netlist.

To further describe the integration methodology, a flowchart is represented in the Fig. 4.19b, which introduces the algorithm to implement the design process and co-simulation in the standard simulation cycle of SystemC-AMS. In this algorithm, each step is defined by a number that corresponds to either a TDF module (①) or a

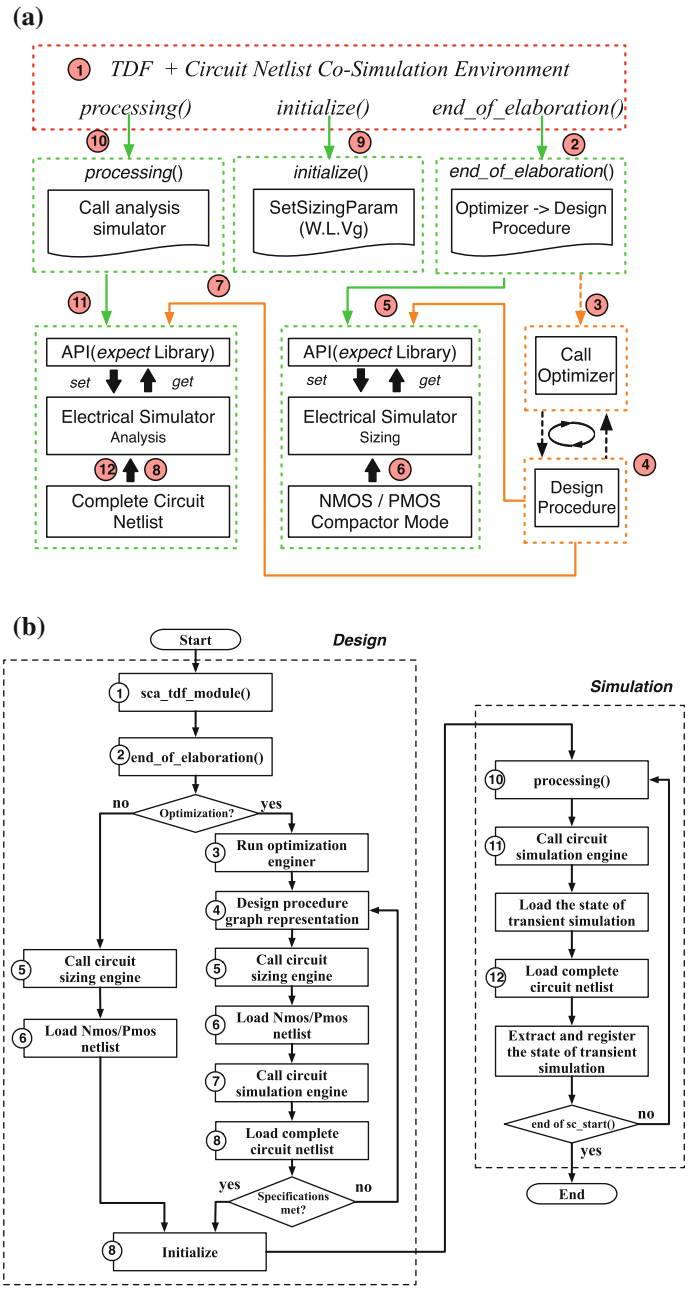

Fig. 4.19 **a** SystemC-AMS, Eldo co-simulation environment. **b** Algorithm that permits to realize circuit sizing interface from system level to circuit level

function call (②–⑫) shown in Fig. 4.19a. The number of each step is the same for Fig. 4.19a and b.

This algorithm can be divided into two parts, which are *system design* and *system simulation,* respectively:

1. The *system design* part corresponds to the sizing and biasing of the circuit within the complete system-level description, and it is the bottom-up design part in Fig. 4.1.

   - In step ②, it defines all the required parameters used for circuit design procedure, such as the configuration of the optimizer, the specifications of the circuit. The sizing and biasing procedure is executed by using a sizing simulator (Ⓒ in Fig. 4.1).
   - In case of performing optimization, an optimizer is called in step ③, just before calling the sizing and biasing procedure.
   - In step ④, the optimizer invokes the sizing and biasing procedure, which is presented by a graph as shown in Sect. 4.2.2.1.
   - In step ⑤, the sizing simulator loads the suitable electrical netlist NMOS/PMOS. Both transistors refer to a transistor compact model, entirely sizable and biasable through simulator interactive commands.
   - At each iteration of the optimization, the sizing simulator computes the sizes and bias values based on different design parameters.
   - The optimizer is closed in case the specifications are successfully met.
   - At the end of the optimization loop, the optimized sizes and bias values will be restored and transmitted in step ⑨.
   - This sizing simulator is closed before the starting step ⑩.

2. From step ⑩, until the end of execution, the steps correspond to the system simulation include the circuit-level propagation. It is the top-down simulation path in Fig. 4.1.

   - In steps ⑩, ⑪, and ⑫, at each *time step*, the signal interface passes the input samples and evaluates the simulated output samples. These steps are executed until the last input sample is processed.
   - At the first execution of step ⑩, an analysis simulator (Ⓓ in Fig. 4.1) is opened, it calls the complete circuit netlist at the step ⑫, and it is closed at the end of the system simulation.
   - During the simulation, a loading and registration of the state of the circuit are performed, respectively, before and after step ⑫ at each time step. These two operations refer to the initial condition of the circuit transient simulation (see Sect. 4.5.1 for more details).

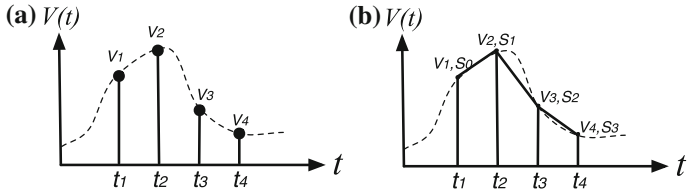

Fig. 4.20 **a** TDF signal with sampled values. **b** Transient simulation with a set of pulsewise linear signals

4.5.1 Transient Analysis Method

The TDF model of computation is not conservative, and it considers values that are discrete in time and value. However, we aim at performing conservative nonlinear simulations for the components described in SPICE netlist. To be able to handle such problem, we convert the TDF input signal shown in Fig. 4.20a to the piecewise linear version shown in Fig. 4.20b. This conversion will be considered as the stimuli signal during SPICE simulation.

The pulse width is set to the sampling period, and a transient analysis is performed during each period. At the beginning of the transient analysis, the voltages at nodes 1, 2, 3, 4, and 5 (The five nodes connect to all the small-signal capacitances in the circuit.) marked in Fig. 4.14 are, respectively, set to previous statement. At the end of current simulation, the value of each node is retrieved and used as the initial conditions for the simulation of the next time step.

[Input TDF signal] [Transient simulation]

To construct a piecewise linear signal and perform the transient simulation from t_n to t_{n+1}, we should firstly know both the sample value V_n and V_{n+1}. Then, we consider the previous statement as the initial condition of this period. Finally, with the command .**TRAN** $\mathbf{t_n}$ **dt uic** in SPICE netlist, it activates the transient analysis. Note that **dt** is the sampling period, Eldo automatically initializes all the node voltages itself as well as the option **uic** included in a .**TRAN** command.

Using the above approach, the unified platform for mixed-signal system design can mix non-conservative system-level behavior with conservative nonlinear circuit simulation.

4.6 Simulation Results

System responses against model responses for two different tests are given in Figs. 4.21 and 4.22. For testing the functionalities of this feedback system, we keep the most two sensitive blocs (PMOS, folded-cascode amplifier) in circuit netlist and model the others modules in SystemC-AMS (bandgap voltage reference circuit, rectifier), as shown in Fig. 4.8.

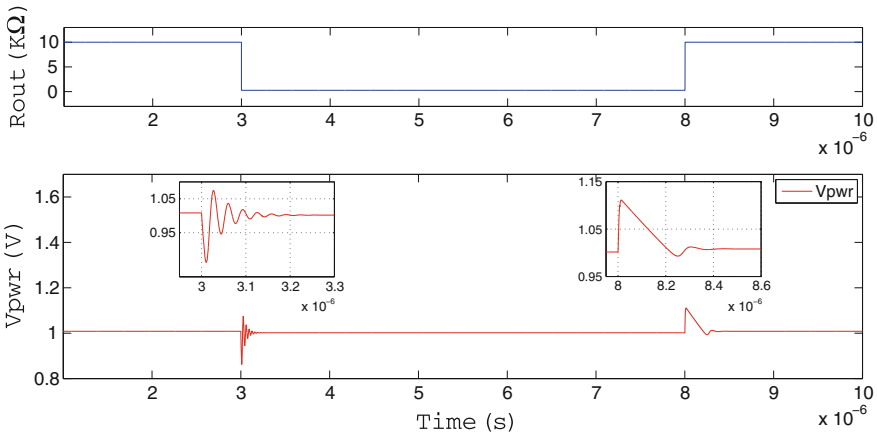

Fig. 4.21 SystemC-AMS, Eldo co-simulation results, and output voltage waveform (V_{pwr}) when the load changes from 10 kΩ to 250 Ω

Figure 4.21 shows the transient waveform of the regulated voltage when the output load switches between 250 Ω and 10 kΩ. The line transient response is measured in condition of the V_{reg} is equal to 1.5 V. The difference between the voltage levels at the two stable states is equal to 6.2 mV. We notice that when the output load decreases form 10 kΩ to 250 Ω, there is an oscillation at the beginning before getting stable. This indicates that there might be stability issues in this configuration.

Figure 4.22 shows the transient waveform of the regulated voltage when the input voltage V_{reg} switches from 1.2 to 1.7 V within 250 ns. This line transient response is measured for load condition (5 kΩ, we choose the mean value between

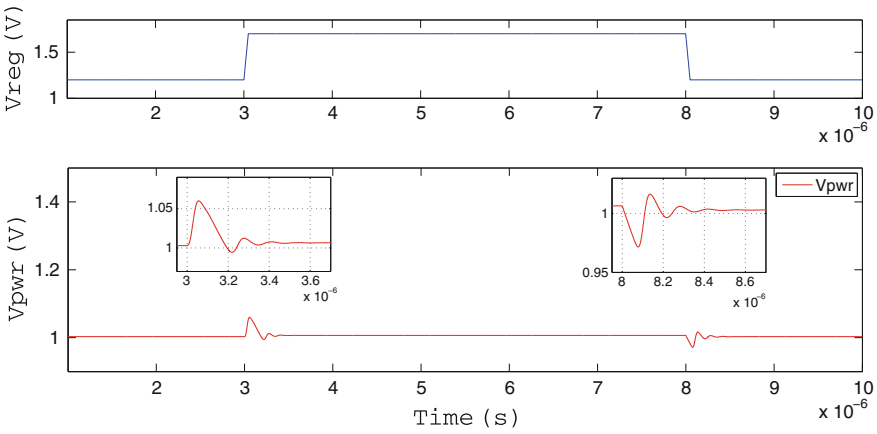

Fig. 4.22 SystemC-AMS, Eldo co-simulation results, and output voltage waveform (V_{pwr}) when the input voltage switches from 1.2 to 1.7 V

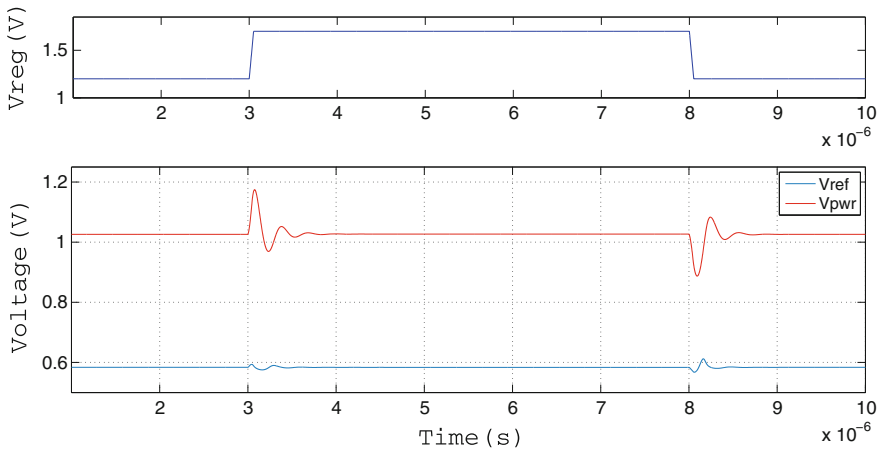

Fig. 4.23 Eldo simulation results, output voltage (V_{pwr}), and reference voltage waveform (V_{ref}) when the input voltage switches from 1.2 to 1.7 V

10 kΩ and 250 Ω, the load capacitance is set to 50 pF). The zoom part of the simulation confirms that it takes about 0.5 µs to settle within 1 % of its final value. The difference between the voltage levels at the two stable states is equal to 3.5 mV.

Another simulation is shown in Fig. 4.23. It presents the simulation result of **step 3** in Fig. 4.8, where the whole circuit netlist is simulated only in Eldo. To compare with the co-simulation environment as shown in Fig. 4.22, we applied the same configuration to simulate the whole circuit. We notice that it takes 0.8 µs to settle within 1 % of its final value. The difference between the voltage levels at the two stable states is equal to 1 mV. Besides, the transient response of V_{ref} indicates the optimized bandgap voltage reference circuit generates a very stable reference voltage for the regulator circuit.

All of the results can be seen in two aspects. Firstly, we propagate the circuit non-idealities and performances from circuit level to system level by using our platform. Secondly, the proposed platform works well in a feedback system, where the feedback loop is applied by introducing a delay. These observations demonstrate the effectiveness and reliability of our proposed modeling, design, optimization, and co-simulation methodology.

4.7 Conclusion

We present a platform for modeling, design, optimization, and co-simulation of mixed-signal systems. It is based on C/C++ language which can be used with SystemC-AMS. In this platform, an optimization engine is introduced for simulation-based hierarchical sizing and biasing using CHAMS. This optimization

engine meets both linear and nonlinear specifications. It is a fast design exploration of analog firm IP, where global exploration following the PEANO curves and Nelder–Mead simplex optimization is performed to realize local exploration.

The co-simulation principles make it possible to link circuit performances to system models, perform conservative nonlinear transient simulation for TDF model of computation, and enable feedback of non-functional properties in the functions models. The proposed approach is used to design and verify an implantable telemetry system. The simulation results prove the efficiency and correctness of our platform.

We foresee that the environment SystemC-AMS will be a common industry platform for modeling, design, optimization, and verification of mixed-signal systems. Compiling of the different level of abstractions to reach this goal, researchers should focus on the design aspects of the mixed-signal systems in SystemC-AMS.

References

1. Accellera Systems Initiative, SystemC AMS 2.0 Standard. http://www.accellera.org/downloads/standards/systemc/ams/ (2013)
2. SystemC AMS PoC Library Beta 2, May 2011, Fraunhofer Institute, Dresden. http://www.systemc-ams.org/ (2011)
3. IEEE Computer Society. 1666-2005 IEEE Standard SystemC Language Reference Manual, pp. 1666–2005. IEEE
4. Zaidi, Y., Grimm, C., Hasse, J.: On mixed abstraction, languages, and simulation approach to refinement with SystemC AMS. EURASIP J. Embed. Syst. (2010). doi:10.1155/2010/489365
5. Formaggio, L., Fummi, F., Pravadelli, G.: A timing-accurate HW/SW co-simulation of an ISS with SystemC. In: IEEE/ACM/IFIP, pp. 152–157 (2004)
6. Frank, F., Weigel, R.: Co-simulation of SPICE Netlists and VHDL-AMS models via a simulator interface. In: Signals, Systems and Electronics, pp. 75–78 (2007)
7. Kirchner, T., Bannow, N., Grimm, C.: Analogue mixed signal simulation using SPICE and SystemC. In: Design, Automation Test in Europe Conference Exhibition, pp. 284–287 (2009)
8. Iskander, R., Louërat, M.-M., Kaiser, A.: Hierarchical sizing and biasing of analog firm intellectual properties. Integr. VLSI J. **46**(2), 123–148 (2013). doi:10.1016/j.vlsi.2012.01.001
9. Vachoux, A., Grimm, C., Einwich, K.: Extending SystemC to support mixed discrete-continuous system modeling and simulation. In: IEEE International Symposium on Circuits and Systems, pp. 5166–5169 (2005)
10. Mu, Z., Van Leuken, R.: SystemC-AMS model of a dynamic large-scale satellite-based AIS-like network. In: Forum on Specification and Design Languages, pp. 1–8 (2011)
11. Cenni, F., Scotti, S., Simeu, E.: Behavioral modeling of a CMOS video sensor platform using Systemc AMS/TLM. In: Forum on Specification and Design Languages, pp. 1–6 (2011)
12. Iskander, R., Louërat, M.-M., Kaiser, A.: Automatic DC operating point computation and design plan generation for analog IPs. Analog Integr. Circ. Sig. Process. J. **56**, 93–105 (2008). doi:10.1007/s10470-007-9075-3
13. Javid, F., Iskander, R., Louërat, M.-M.: Simulation-based hierarchical sizing and biasing of analog firm IPs. In: IEEE International Behavioral Modeling and Simulation Conference, pp. 43–48 (2009)
14. Levi, T., Lewis, N., Tomas, J., Fouillat, P.: IP-based methodology for analog design flow: application on neuromorphic engineering. In: IEEE International NEWCAS-TAISA Conference, pp. 343–346 (2008)

15. Saleh, R., Wilton, S., Mirabbasi, S., Hu, A., Greenstreet, M., Lemieux, G., Pande, P.P., Grecu, C., Ivanov, A.: System-on-chip: reuse and integration. In: Proceedings of the IEEE, pp. 1050–1069 (2006)
16. Javid, F., Iskander, R., Louërat, M.-M., Dupuis, D.: Analog circuits sizing using bipartite graphs. In: IEEE International Midwest Symposium on Circuits and Systems, pp. 1–4 (2011)
17. Libes, D.: Exploring Expect: A Tcl-Based Toolkit for Automating Interactive Programs. O'Reilly, Sebastopol (1995)
18. Liu, W.: MOSFET Models for SPICE Simulation: Including BSIM3v3 and BSIM4. Wiley, New York (2001)
19. NXP. MOS Model PSP level 103. http://www.nxp.com/models/mos/_models/psp/ (2011)
20. Enz, C., Krummenacher, F., Vittoz, E.: An analytical MOS transistor model valid in all regions of operation and dedicated to low-voltage and low-current applications. Analog Integr. Circ. Sig. Process. J., 83–114 (1995)
21. Gielen, G.E., Wambacq, P., Sansen, W.: Symbolic analysis for analog circuits: a tutorial overview. In: Proceedings of the IEEE 82, pp. 287–304 (1994)
22. Gielen, G.E., Sansen, W.: Symbolic Analysis for Automated Design of Analog Integrated Circuits. Kluwer Academic Publishers, The Netherlands (1991)
23. Dong, N., Roychowdhury, J.: General-purpose nonlinear model-order reduction using piecewise-polynomial representations. IEEE Trans. Comput. Aid. Des. **27**, 249–264 (2008). doi:10.1109/TCAD.2007.907272
24. Bernardinis, F.D., Jordan, M., Sangiovanni-Vincentelli, A.: Support vector machines for analog circuit performance representation. Proceedings of the Design Automation Conference, pp. 964–969 (2003)
25. Malak, A., Li, Y., Iskander, R., Durbin, F., Javid, F., Guebhard, J.-M., Louërat, M.-M., Tissot, A.: Fast multidimensional optimization of analog circuits initiated by monodimensional global peano explorations. Integr. VLSI J. **48**, 198–212 (2014). doi:10.1016/j.vlsi.2014.04.002
26. Poivey, C., Durbin, F., Haussy, J.: Méthodes d'optimisation Globale pour la CAO de Circuits Intégrés. Interface avec le simulateur Spice-PAC: Optimisation des performances de circuit non linéaires. Rapport CEA-R-5465 (1988)
27. Rust, J.: Using randomization to break the curse of dimensionality. Econometrica: J. Econometric Soc. **65**, 487–516 (1997)
28. Lagarias, J.C., Reeds, J.A., Wright, M.H., Wright, P.: Convergence properties of the Nelder-Mead simplex algorithm in low dimensions. SIAM J. Optim. **9**, 112–147 (1998)
29. Nelder, J.A., Mead, R.: A simplex method for function minimization. Comput. J. **7**, 308–313 (1965)
30. Spendley, W., Hext, G.R., Himsworth, F.R.: Sequential application of simplex designs in optimisation and evolutionary operation. Technometrics **4**(4), 441–461 (1962)
31. Sobot, R.: Implantable RF telemetry for cardiac monitoring in the murine heart: a tutorial review. EURASIP J. Embed. Syst. (2013). doi:10.1186/1687-3963-2013-1