Chapter 31

TOP–DOWN DESIGN METHODOLOGY FOR ANALOG CIRCUITS USING MATLAB AND SIMULINK

Naveen Chandra and Gordon W. Roberts
Microelectronics and Computer Systems Laboratory, McGill University

31.1. Introduction

This chapter presents a new design methodology for the creation of analog or mixed signal integrated circuit components. Through the use of top–down design techniques in conjunction with an optimization process, circuit design can take place at the highest level of abstraction. As a result, the requirements of the building blocks will be specified prior to the undertaking of transistor level simulations, thereby saving much valued design time. This method has the added advantage that designs can be implemented with currently used, and widely available tools. For illustration, a design example featuring a third-order, switched capacitor (SC) delta–sigma $(\Delta\Sigma)$ modulator will be presented.

A top–down design methodology consists of the division of labor in a progressive manner, among several levels of abstraction. On the conception of an *idea* and the setting of performance goals, design at the *system* level can commence. It is at this stage that the functionality of the proposed system is tested, the design or selection of a high-level architecture is made, and the analog building blocks necessary for implementation are determined. Following this, the implementation of the building blocks at the *circuit* transistor level is performed, followed by the *layout* implementation of the transistors and other components, and this is then completed by *fabricating* the design in silicon. A top–down design cycle is illustrated in Figure 31.1.

In particular, the design of analog systems primarily consists of three obstacles:

1 The selection of an architecture

2 The determination of the specifications for the analog building blocks necessary to implement the chosen architecture

3 The minimization of the effects due to circuit non-idealities.

These are usually dealt with separately, and more often than not, most of the building block specifications and non-idealities are explored through

C. Toumazou et al. (eds), Trade-Offs in Analog Circuit Design: The Designer's Companion, 923–952.

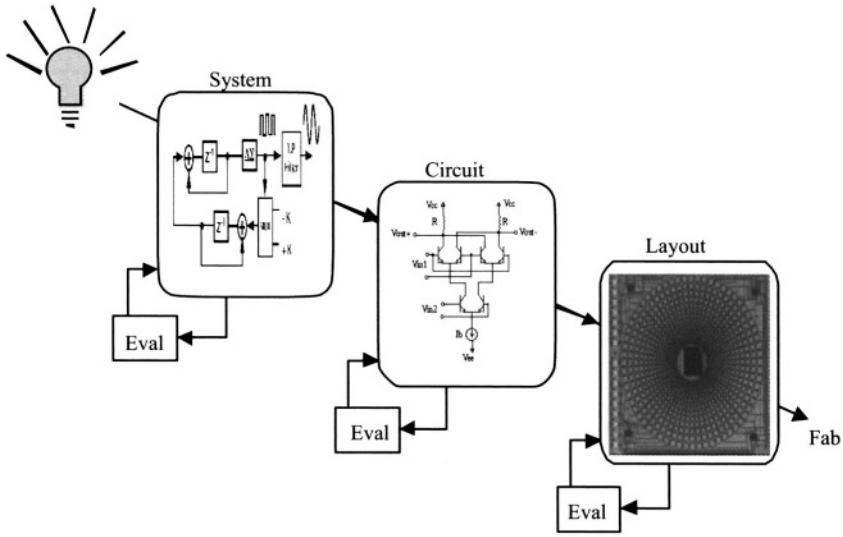

Figure 31.1. Top–down design flow.

simulation at the transistor level using circuit simulation programs such as SPICE. To verify the circuit with respect to changes in passive component (resistor, capacitor, etc.) and transistor characteristics requires simulation over the full range of these variations. Obtaining performance curves, when considering a single architecture, let alone multiple architectures, also requires multiple simulations. Attempting to perform this type of analysis at a low level can be problematic due to long simulation times [1]. Most system simulations require days to run for each case given the fastest workstations. As a result, the design cycle will take too long to practically meet the market demands for the technology, especially given the rapid progression seen today. Table 31.1 shows the Semiconductor Industry Association (SIA) predictions made in 1999. As is evident, Very Large Scale Integrated (VLSI) circuit technology feature sizes are expected to shrink, reenforcing the need for shorter design cycles.

On top of this problem, the results obtained from SPICE may be limited by rounding and truncation errors that accumulate over the large number of time steps needed for accurate simulation. Furthermore, it is often difficult to focus on key design parameters when working with many integrated devices at the transistor level. In addition to this, the entire procedure must be carried out for each architecture selected, necessitating even further simulation. Therefore, the designer has to be very careful when choosing an architecture for fear of losing much time in the initial design stages.

Table 31.1. Long-term technology roadmap [2].

Feature	*1999*	*2002*	*2005*	*2008*	*2011*	*2014*
Line width (μm)	0.18	0.13	0.10	0.07	0.05	0.035
Transistors/chip (millions)	23.8	47.6	190	539	1523	4308
Frequency (MHz)	1200	1600	2000	2500	3000	3600
Metal layers (max)	7	8	9	9	10	10
Supply voltage (volts)	1.5–1.8	1.2–1.5	0.9–1.2	0.6–0.9	0.5–0.6	0.3–0.6

In order to reduce the number of design iterations and to better explore the design options, it is beneficial to perform the analysis at the system architectural level before starting transistor level design. This allows for a feasibility analysis in which all the design considerations are treated at the highest level of abstraction. The ultimate goal of this procedure is to have the low level circuit parameters be dictated by the selected architecture and desired performance. Therefore, a top–down design procedure using optimization will be presented which makes use of Simulink modeling and Matlab optimization [3]. More specifically, in Section 31.2, the motivation for the design methodology is presented and discussed. The work is further explained in Sections 31.3 and 31.4 with a design procedure featuring a $\Delta\Sigma$ modulator involving Simulink modeling. This is followed by a discussion on the Matlab optimization setup in Section 31.5, and supported by simulation results in Section 31.6. A simple, fully detailed example is presented in Section 31.7, with some concluding remarks being made in Section 31.8.

31.2. Design Methodology Motivation

Choosing the specifications for an analog circuit, given a set of design objectives such as minimal area and power, is potentially a very complicated and time-consuming process. This task is further complicated by the fact that the number of specifications to deal with is usually very large, and varies from one application to another.

There exist tools that are aimed at fully automating the design process, however, they are limited to a small number of fixed schematics [4–7]. These tools, which are not designed to be reproduced, remove the designer from the process, and does nothing to increase the designer's knowledge. Furthermore, the techniques used in these programs are hidden and cannot be applied to other designs.

One of the main drawbacks to analog design is the uncertainty that arises due to a change in technology. As a result, it is more amenable to consider a design process that can begin without a dependence on a specific technology. This

would allow for a large portion of the design to be carried out independently of the technology, and could lead to a high degree of reusability. High-level optimization geared design avoids the reliance on a specific technology the longest, and in this case provides the designers with values for familiar parameters (e.g. g_m and r_{out} in the case of an operational transconductance amplifier or OTA). These are quantities that are very familiar and understood by designers and provide excellent guidelines for the construction of a device. Furthermore, symbolic analysis is a method by which designers can obtain an understanding of a circuit's behavior. Therefore, relating the behavior of the circuit to several descriptive equations or expressible rules of thumb, and moreover to a few key parameters, can reduce the complexity of a problem and lends itself to a solution through optimization.

The main focus of this chapter is to provide designers with a means of tackling the design problem by presenting a simple to implement design methodology that involves widely used and available tools. As a result, the procedures can be implemented and reused with little difficulty or expense. In order to make use of widely available tools which are already known to many designers [8,9], Simulink is used to implement the system architecture and model non-idealities, while the Matlab optimization toolbox [10] is used to create routines to optimize the circuit parameters.

31.2.1. Optimization Procedure

Optimization in general concerns the minimization or maximization of a function. In the case of analog systems, the objective is to maximize or minimize a given set of performance criterion, such as the Signal to Noise and Distortion Ratio (SNDR), power, area or settling time. To do so, one must establish a trackable set of variables (g_m, r_o, I_{bias} or C_{load}) that can be used to model the system while simultaneously providing a link to the performance criterion. An illustrative flow of this procedure can be seen in Figure 31.2. It consists of four main components: a property measurement block, a summing block with the desired property as one input, an optimization procedure and finally the circuit that is under development. The property measurement block detects particular attributes of the output signal from the circuit and compares it with the desired behavior. The error signal is then used to adjust the key parameters of the circuit such that the error is minimized. It is using this very simple principle that the proposed top–down methodology is executed.

Due to its ability to model nonlinear devices, there already exists much work in a variety of fields that use Simulink modeling, and as such it is not difficult to find pre-existing models, or to create new models specific to a design [11–15]. In order to best explain the procedure, the proposed methodology will be illustrated using a $\Delta\Sigma$ analog-to-digital converter (ADC). It is important

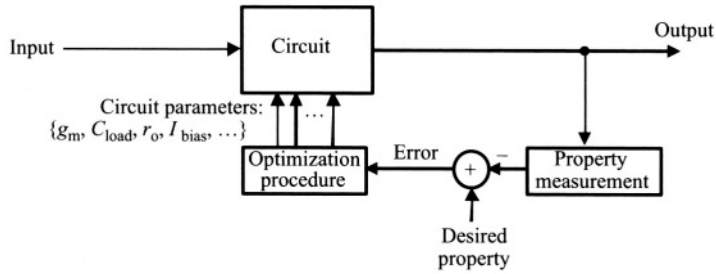

Figure 31.2. Optimization procedure.

to realize that this methodology is not limited to $\Delta\Sigma$ modulators. It may be used to help design any device or system, in which key parameters, formulas or rules of thumb can be identified.

31.3. Switched Capacitor Delta–Sigma Design Procedure

$\Delta\Sigma$ ADCs are widely used and well suited for high-resolution conversion. One dominant factor in their popularity stems from their high tolerance to component mismatch and circuit non-idealities. Despite its high tolerance to non-idealities, the $\Delta\Sigma$ modulator is still governed by the limitations of its analog building blocks. In particular, SC $\Delta\Sigma$ modulators are sensitive to circuit non-idealities at the input stage where no noise shaping has yet taken place [16]. Specifically, sampled capacitor (kT/C) noise and OTA characteristics (noise, clipping, finite gain, finite bandwidth and slew rate (SR)) at the front-end limits the achievable dynamic range and therefore performance. A high-level diagram of a Simulink realization of a third-order $\Delta\Sigma$ single-loop modulator can be seen in Figure 31.3. It is important to realize that upon implementation, each component block has to be translated into its circuit equivalent (analog building blocks) and then implemented.

In order to use an optimization procedure, it is first necessary to isolate the key parameters that influence the system's performance. In the design of high-resolution SC $\Delta\Sigma$ modulators, it can appear that there is a large set of parameters to optimize. However, if one examines the building blocks of the system, it can be seen that most of the factors that affect the system's performance (kT/C noise, OTA noise, voltage clipping due to a finite output range, finite gain, finite bandwidth and SR) can be related to a small set of parameters. These parameters will now be explored, and since $\Delta\Sigma$ modulators are sensitive to circuit non-idealities at the input stage (the first integrator seen in Figure 31.3) where no noise shaping has yet taken place, this section of the modulator will be the focus of the analysis.

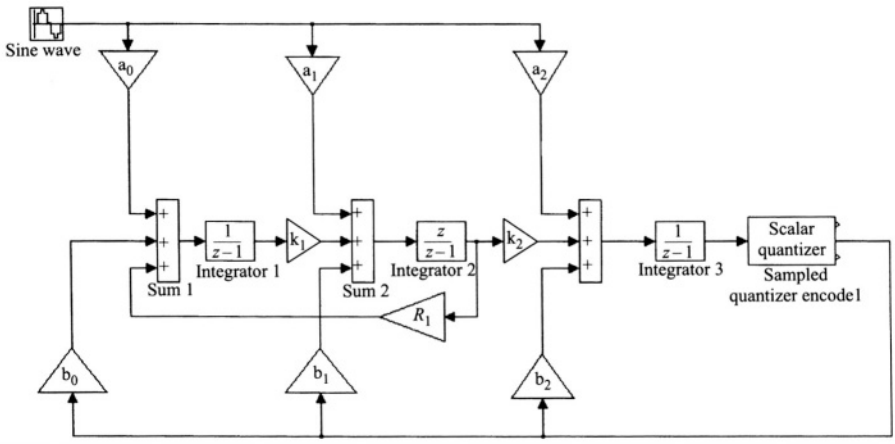

Figure 31.3. High-level diagram of the modulator.

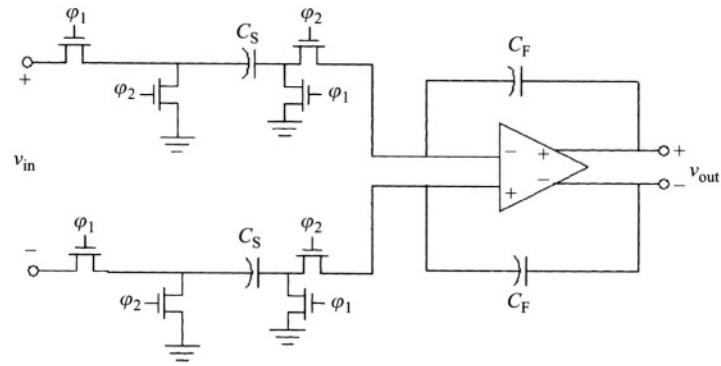

Figure 31.4. SC implementation of the first integrator.

31.3.1. Switched Sampled Capacitor (kT/C) Noise

A critical source of noise is the kT/C noise injected into the first integrator of the modulator because it directly increases the noise level at the output. The kT/C noise component results from the sampled noise stored on the capacitor due to switch resistances. As a result, the input capacitors must be large enough to counter this effect. Therefore, the first key parameter, which can be seen in Figure 31.4, is the input sampling capacitor C_S.

Table 31.2. OTA non-idealities.

Non-ideality	*Result*
Finite DC gain	Rise in quantization noise
Saturation	Possible harmonic distortion
Bandwidth limiting	Noise due to incomplete settling
SR limiting	Noise due to incomplete settling and harmonic distortion
Thermal noise	Added white noise

31.3.2. OTA Parameters

The OTA within the modulator is the most critical component, as the non-idealities that it exhibits causes an incomplete transfer of charge leading to nonlinearities. By briefly examining this device, key parameters that govern its non-ideal behavior can be isolated and examined. Shown in Table 31.2 are the corresponding non-idealities that must be considered in an OTA.

There are three OTA topologies commonly used in the design of $\Delta\Sigma$ modulators [17], the folded cascode amplifier (Figure 31.5(a)), the two-stage class A amplifier (Figure 31.5(b)), and the two-stage class AB amplifier (Figure 31.5(c)). Table 31.3 highlights the key properties that govern the operational performance of these OTAs. These properties allow for behavioral modeling of the device and, more importantly, can be dissected to obtain the key parameters of the device. It is important to note that r_{o1} refers to the output resistance of the first stage. We can see from this table, that the main sources for non-idealities in the modulator can be related to a few key OTA parameters (g_m, r_o, C and I_{bias}). As a result, the key parameters for each of the OTA topologies that can be used in the modeling of each structure are listed in Table 31.4.

31.4. Modeling of $\Delta\Sigma$ Modulators in Simulink

In order to perform behavioral simulations of $\Delta\Sigma$ modulators while taking into account most of the non-idealities, it is necessary to create models for them. A set of models has been proposed that attempt to simulate the non-idealities in Simulink [15]. The following work is intended as modifications and extensions to those models for the purpose of providing reference to the key parameters highlighted in Table 31.4, and to make them more suitable for an optimization environment.

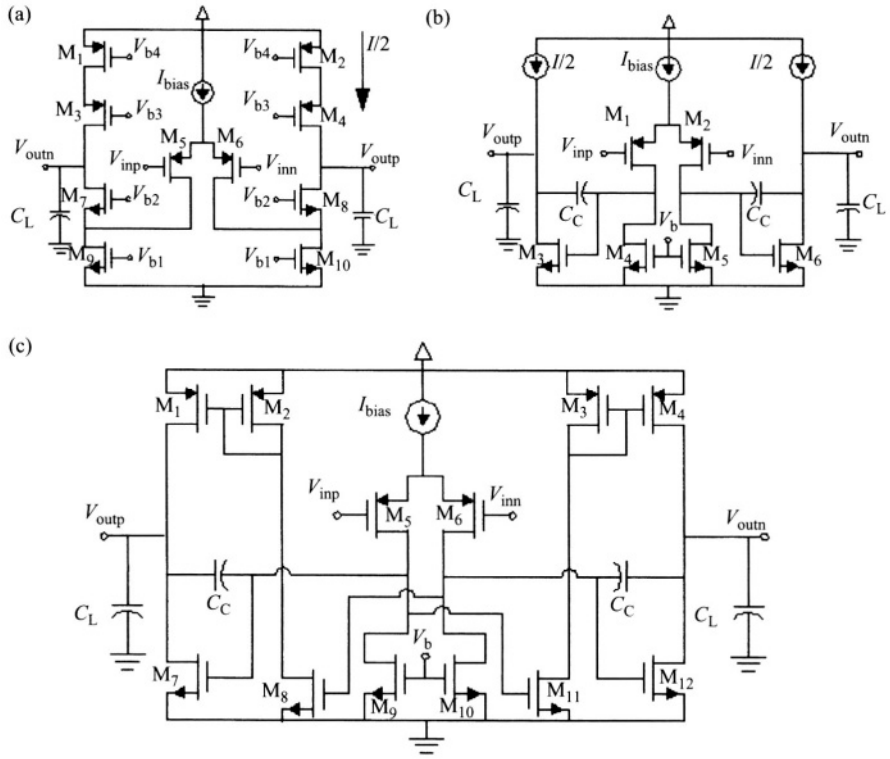

Figure 31.5. OTA topologies: (a) folded cascode, (b) two-stage class A, (c) two-stage class AB.

Table 31.3. Key properties in OTAs.

Property	*Folded cascode*	*Two-stage class A*	*Two-stage class AB*
DC gain	$g_{m5} * r_o$	$g_{m1} * ro1$	$g_{m5} * r_{o1}$
Unity gain Frequency	g_{m5}/C_L	g_{m1}/C_C	g_{m5}/C_C
SR	I/C_L	$\min\{I_{bias}/C_C, I/(C_L + C_C)\}$	I_{bias}/C_C

31.4.1. Sampled Capacitor (kT/C) Noise

The thermal noise associated with the switching due to input sampling onto the first integrator can be modeled as follows:

$$y(t) = k_{int} \cdot [x(t) + n_{switch}(t)] \tag{31.1}$$

Table 31.4. Key parameters for modeling.

Folded cascode	*Two-stage class A*	*Two-stage class AB*
$g_{m5}, r_o, C_L, I_{bias}, C_S$	$g_{m1}, r_{o1}, C_L, C_C, I_{bias}, I, C_S$	$g_{m5}, r_{o1}, C_C, I_{bias}, C_S$

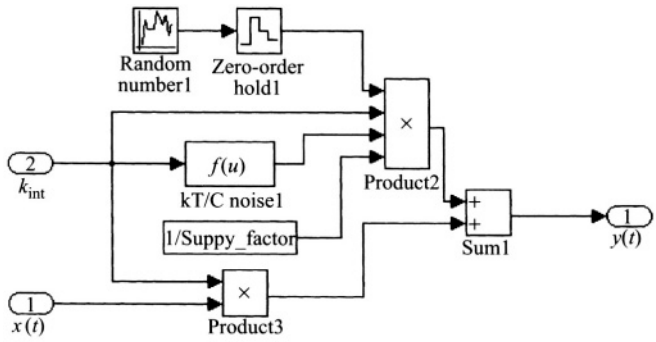

Figure 31.6. kT/C noise model in simulink.

where

$$n_{switch}(t) = \sqrt{\frac{k \cdot T}{C_S}} \cdot \mathrm{RN}(t), \tag{31.2}$$

where k_{int} is the integrator gain, C_S is the input sampling capacitor, k is Boltzman's constant, T is temperature and RN(t) is a Gaussian random number with zero mean and unity standard deviation. This has been implemented in the Simulink block diagram shown in Figure 31.6. It is important to note that the formula used in the $f(u)$ block with its input k_{int}, implements the $n_{switch}(t)$ formula given in equation (31.2). A Supply_factor block can also be seen in the diagram. This is present for simulations in which it is desired to normalize the supply voltage. If no such normalization is required, then Supply_factor should be set to 1.

31.4.2. OTA Noise

The input referred noise of an OTA can be modeled as follows:

$$y(t) = k_{int} \cdot [x(t) + n_{OTA}(t)] \tag{31.3}$$

where

$$n_{OTA}(t) = V_n \cdot \mathrm{RN}(t) \tag{31.4}$$

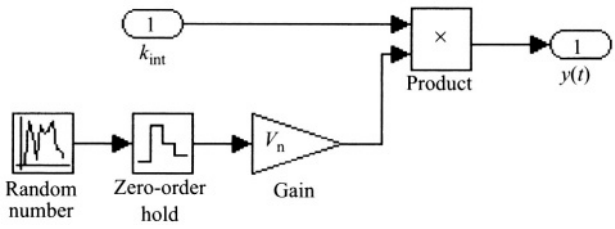

Figure 31.7. OTA noise model in simulink [15].

where k_{int} is the integrator gain, V_n is the OTA RMS noise voltage, and RN(t) is a Gaussian random number with zero mean and unity standard deviation. The model can be seen in Figure 31.7.

31.4.3. Switched Capacitor Integrator Non-Idealities

SC integrators are composed of non-ideal analog building blocks, the foremost of which is an OTA. As a result, a model needs to be created to take into account the noise and/or distortion generated by these components.

One of the main non-idealities is caused by having finite gain in the OTA. As a result, the transfer function of the integrator changes depending on the amount of DC gain provided by the amplifier. The ideal transfer function is given by

$$H(z) = \frac{k_{int}}{z-1} \tag{31.5}$$

In reality, this model is not sufficient for the complete description of the integrator's characteristics. Examining Figure 31.8 reveals that the transfer function of the integrator changes depending on the amount of DC gain provided by the amplifier.

As a result, the transfer function takes on a gain dependency given by:

$$H(z) = \frac{k_{int}}{z-\alpha} \tag{31.6}$$

where k_{int} is the integrator gain,

$$\alpha = \frac{g_m \cdot r_o - 1}{g_m \cdot r_o}$$

and g_m as well as r_o are the parameters in the amplifier that define its gain. The deviation caused by finite gain, and represented by α, must be accounted for in any modeling.

Another source of non-ideality comes about because an OTA is subject to clipping. Clipping occurs when the OTA is asked to produce an output voltage

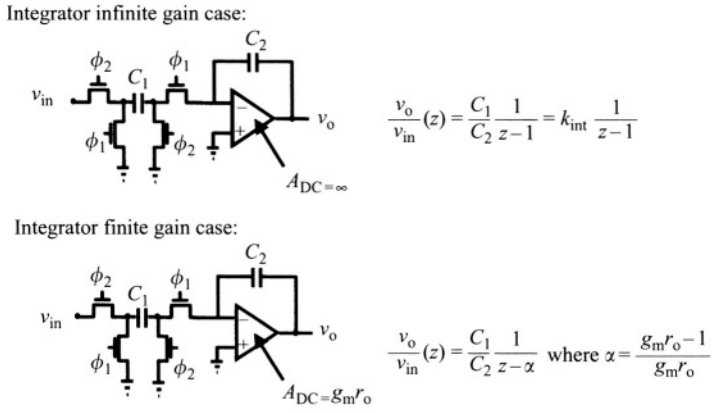

Figure 31.8. Transfer function of integrator with finite gain.

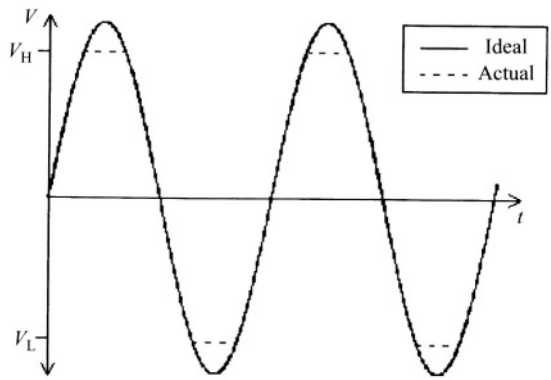

Figure 31.9. Distortion effect of clipping.

higher than the voltage supply rails. Should this occur in the system, the output would cease to follow the ideal output voltage waveform, and instead a distorted waveform would be produced. This is illustrated in Figure 31.9, where V_H and V_L are the upper and lower power supply rails. This effect must be integrated into the model as well, and is accomplished through the use of a saturation block in Simulink.

A final source of non-ideality that will be dealt with is slew and bandwidth limiting. The slew and bandwidth behavior of an OTA was modeled based on an analysis in which their effects in a SC integrator are interpreted as a nonlinear gain [18]. There are three basic mutually exclusive conditional states that can

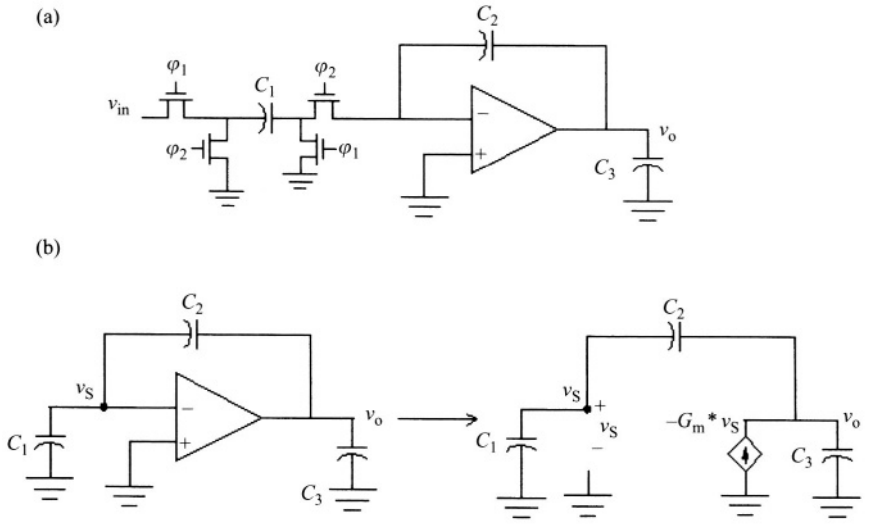

Figure 31.10. Analysis of transient integrator behavior.

occur, based on the present input voltage and the previous output voltage:

1 complete bandwidth limiting,

2 complete SR limiting, and

3 first SR followed by bandwidth limiting.

In order to understand and model this effect properly, it will now be explored in some detail. The primary goal is to capture the input–output transient behavior of the circuit in mathematical form. Figure 31.10(a) shows the single-ended SC integrator, while Figure 31.10(b) reveals the circuit representation of the integrator on the driving phase ϕ_2. It is important to recall that in a sampled-data system such as this, there is a finite amount of time given for all the phases to execute. In a two-phase system, that time corresponds to $T/2$ (where T corresponds to the clock period). For the purpose of this analysis, assume that capacitor C_3 is a lumped capacitor containing the output parasitics and the loading of the next stage. It is also important to realize that the analysis was done on a single-ended amplifier to simplify the understanding of the procedure. If one takes advantage of the symmetry present in a differential circuit, the same analysis is valid [19].

The transient associated with the output voltage of the SC integrator can be represented in the following form:

$$v_o(t) = v_o(0^-) - G \cdot v_S(0) \cdot (1 - e^{-t/\beta \cdot w_t}) \tag{31.7}$$

where $\beta = C_2/(C_1 + C_2)$, $w_t = G_m/C_L$, $C_L = C_3 + C_1 \cdot C_2/(C_1 + C_2)$, $G = C_1/C_2$, $v_S(0)$ is the input OTA node voltage at the beginning of this cycle, and $v_o(0^-)$ refers to the output voltage present at the end of the previous cycle. This means that effectively, the charging/discharging time constant τ of the circuit can be defined as:

$$\tau = \frac{1}{\beta \cdot w_t} = \frac{C_1 \cdot C_2 + C_2 \cdot C_3 + C_1 \cdot C_3}{G_m \cdot C_2} \tag{31.8}$$

It is important to note that $v_S(0) = -v_{in}(0)$, and as a result, equation (31.7) can be written as

$$v_o(t) = v_o(0^-) + G \cdot v_{in}(0) \cdot \left(1 - e^{-t/\beta \cdot w_t}\right) \tag{31.9}$$

The system behaves as described by equation (31.9) when limited purely by the charging/discharging time constant, and is commonly referred to as being bandwidth limited.

The system may also suffer from SR limitations, which is illustrated in Figure 31.11. Since the amplifier can only supply a maximum finite current (I_{bias}) to the capacitive load (C_L), an inherent limit to the instantaneous change in output voltage is present. Therefore, if the change in voltage is too large, this limit is exceeded, and the output will be subject to a linear charging (with slope SR) described by

$$\mathrm{SR} = \frac{\mathrm{d}}{\mathrm{d}t} v_o(t)\Big|_{max} = \frac{I_{bias}}{C_L} \tag{31.10}$$

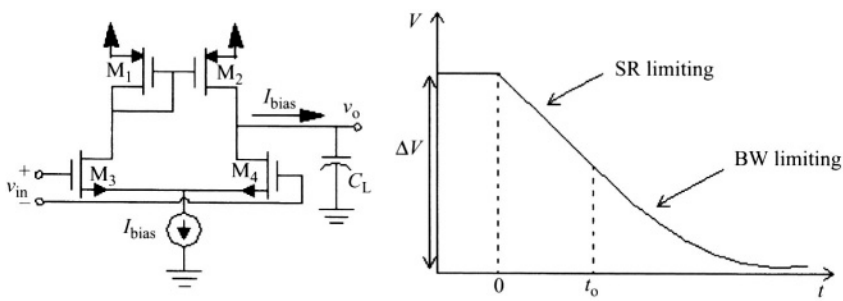

Figure 31.11. The presence and effect of SR limiting.

If it is assumed that the system is SR limited for a portion of the time t_o, the output voltage can be written as

$$v_o(t) = \begin{cases} v_o(0^-) + \mathrm{SR} \cdot t & \text{if } t \le t_o \\ v_o(0^-) + \mathrm{SR} \cdot t_o + (G \cdot v_{in}(0) - \mathrm{SR} \cdot t_o) \cdot (1 - e^{(t_o - t)/\tau}) & \text{if } t_o < t \end{cases} \tag{31.11}$$

At time t_o, the system ceases to be SR limited and returns to its "natural" charging/discharging state and becomes bandwidth limited. It follows that at only the point corresponding to t_o, the bandwidth and SR limiting will be identical. This arises because bandwidth limiting produces a monotonically increasing/decreasing exponential voltage function. Such a response has a different slope at every point, and thereby will be equal to SR only once (i.e. at $t = t_o$, $dv_o(t)/dt = \mathrm{SR}$). As a result, one can solve for the time the system remains SR limited by evaluating the slope of the bandwidth-limited response at time t_o and equating it with the slope of a SR limited system, that is,

$$\mathrm{SR} = \frac{d}{dt} v_o(t)\bigg|_{t_o} = \left(\frac{-\mathrm{SR} \cdot t_o}{\tau} + \frac{G \cdot v_{in}(0)}{\tau} \right) \cdot e^{(t_o - t)/\tau} \tag{31.12}$$

After evaluating the function at time t_o, one is left with the following equation:

$$\mathrm{SR} = \frac{-\mathrm{SR} \cdot t_o}{\tau} + \frac{G \cdot v_{in}(0)}{\tau} \tag{31.13}$$

This can be easily rearranged to provide,

$$t_o = \frac{G \cdot v_{in}(0)}{\mathrm{SR}} - \tau \tag{31.14}$$

Therefore, SR limiting will occur if $dv_o(t)/dt > \mathrm{SR}$ and last for a length of time given by t_o.

When creating a model involving SR and bandwidth limiting, the actual transient behavior of the integrator is not crucial since SC circuits depend only on the output at the end of each cycle. Therefore, it is important to obtain an expression for the output voltage of the integrator at time $T/2$. The appropriate behavior at this time can be summarized as follows:

$$v_o\left(\frac{T}{2}\right) = \begin{cases} v_o(0^-) + G \cdot v_{in}(0) \cdot (1 - e^{-T/2\tau}) & \text{if } t_o \le 0 \\ v_o(0^-) + \mathrm{SR} \cdot t_o + (G \cdot v_{in}(0) - \mathrm{SR} \cdot t_o) \cdot \left(1 - e^{(t_o - T/2)}\right) & \text{if } 0 < t_o \le T_2 \\ v_o(0^-) + \mathrm{SR} \cdot T/2 & \text{if } \frac{T}{2} < t_o \end{cases} \tag{31.15}$$

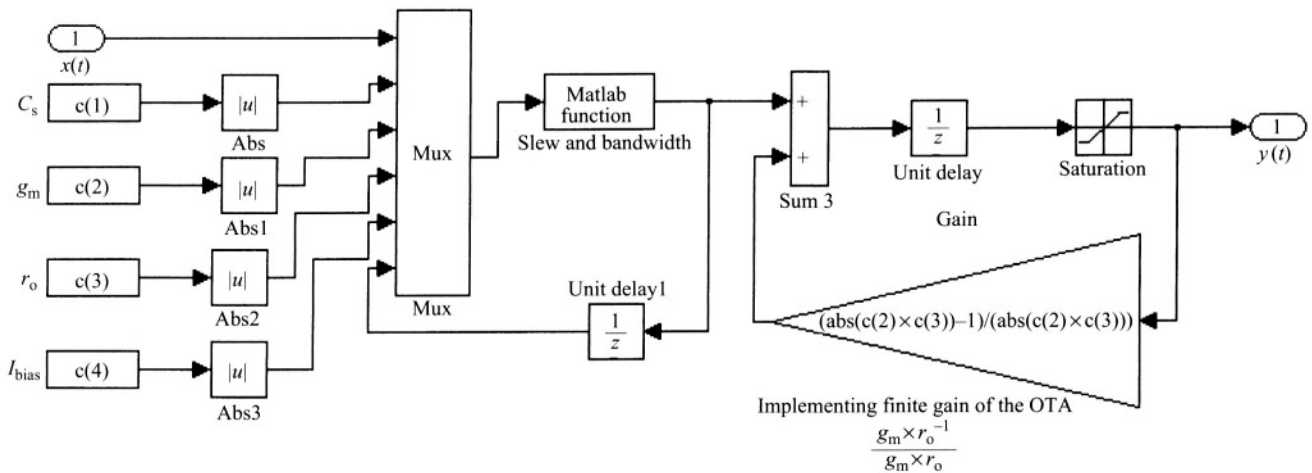

Figure 31.12. Non-ideal integrator model in simulink.

The finite gain, the clipping levels, the finite bandwidth and the SR limiting of the OTA are incorporated into a Simulink model of the SC integrator as shown in Figure 31.12. The formulas used in this block depend on the type of OTA implemented. In this case, the formulas for the folded cascode amplifier were used, hence parameters C_S, g_m, r_o and I_{bias} are required (Table 31.4). From the diagram, it can be seen that a Slew and Bandwidth Matlab function block exists that interfaces with a multiplexer. Due to its conditional nature, the modeling was much easier accomplished through the use of a Matlab function, which implemented the previously described equations. Furthermore, it was necessary due to the nature of SR limiting, to create a block that is capable of retaining "memory" of its previous state. This is accomplished by feedback through the multiplexer, which also takes in the key parameter values (C_S, g_m, r_o, I_{bias}) fed to it by the optimization routine.

31.5. Optimization Setup

In this section, a set of guidelines that clarify the Matlab design space will be presented to help explain and carry out the design methodology. Some familiarity with the Matlab optimization toolbox will be assumed.

The optimizer is essentially split up into three parts and is illustrated in Figure 31.13. Each block is described below:

1 The Matlab optimizer block receives an error value and accordingly adjusts the key parameters given to it in order to minimize any further

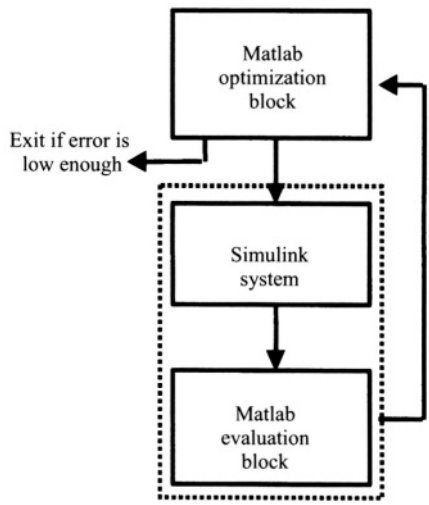

Figure 31.13. Optimization setup.

error. On the first run, the optimizer begins by using the initial conditions supplied to it for the key parameters.

2 The Simulink system, with its built in non-idealities, is called by the optimizer so that data for property measurements can be obtained (i.e. getting the required data to produce SNDR calculations).

3 The Matlab evaluation block receives data from the Simulink System, evaluates the results based on constraints (bounds) on the key parameters, and produces a value for the error based on the designer's criteria. When the error is low enough, the process will stop and deliver its results.

For the specific case of the $\Delta\Sigma$ modulator, the folded cascode OTA was chosen, and as a result, the key parameters that necessitated optimization were C_S, g_m, r_o and I_{bias}. The next step is to define both the error measurement criteria and the constraints. It is important to note that these criteria can be as simple or as complicated as the designer wishes. The more criteria that are given translate into a longer optimization routine, but a final result that is closer to the overall goal. In the case given here, the following three criteria were used:

1 *Minimization of area.* This was formulated on the basis of capacitance. Since the capacitance required in single-loop modulators usually accounts for at least half of its area, the area constraint placed an emphasis on minimizing this capacitance.

2 *Minimization of power.* The main power consumption in the modulator comes from the op-amp used in the first-stage integrator. As a result, constraints were placed on the bias current in order to place an upper bound on power consumption, and to minimize the current used.

3 *Maximization of SNDR.* Since SNDR is a dominant measure of a modulator's performance, lower limits were placed on its value, with an emphasis placed on its maximization.

The Simulink system was used in order to get the data for the SNDR measurements. All these measurements were affected by the key parameters, as the modeling for the system was based around them. An example of a $\Delta\Sigma$ Simulink system, with all of the non-ideality modeling present, can be seen in Figure 31.14. This is the same system that was presented in Figure 31.3, except that in this diagram, the Simulink models have been added to account for kT/C and OTA noise, and to account for the use of a non-ideal integrator in the first stage of the modulator.

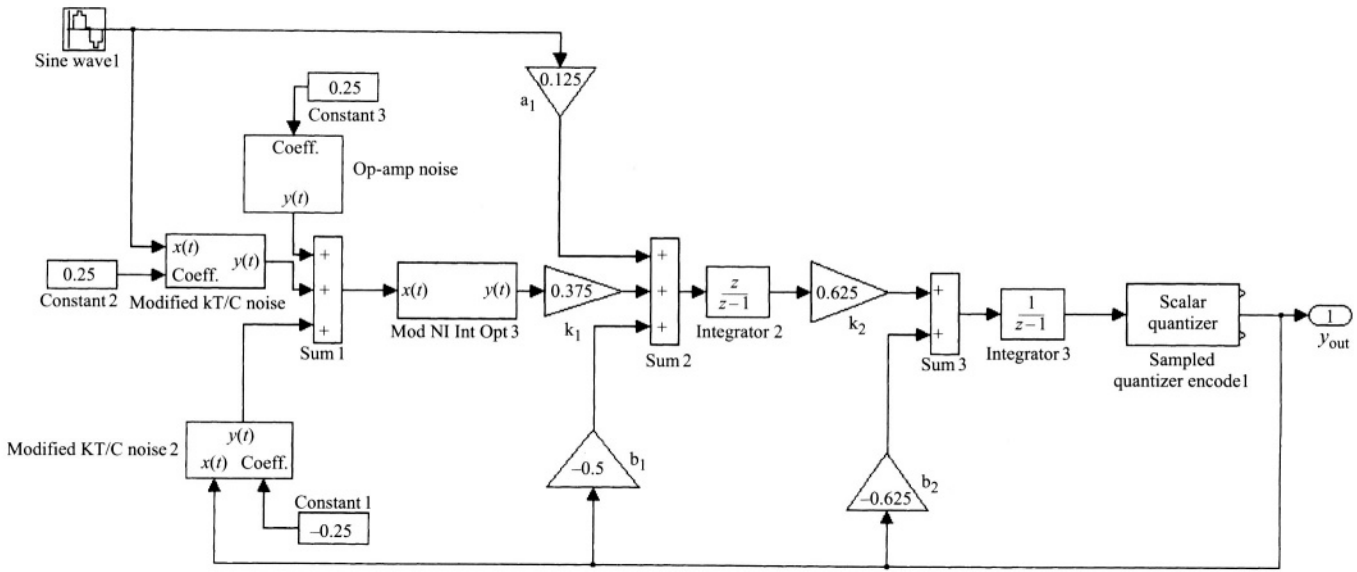

Figure 31.14. Non-ideal system in simulink.

31.5.1. Implementation in Matlab

Example code for the third-order $\Delta\Sigma$ modulator is presented in Figures 31.15 and 31.16. The first function called "optim.m" (Figure 31.15), basically implements the Matlab optimizer block seen in Figure 31.13. This function takes in performance specifications from the user (such as the sampling frequency (f_S), and the desired SNDR (SNDR_goal)) and is also used to set up the optimization options such as the step size and tolerance of the program. This program strives to minimize the coefficients that are specified, in this case C_S, g_m, r_o and I_{bias}. By minimizing these parameters, the minimal requirements for the OTA will result, and power should also be minimized.

```
% File: optim.m

function [c]  =  optim(Npoints,M,Supply_factor,fs,bw,simtime1,SNDR_goal);

% Calculation of Constants

SNDR_weight = 10^(SNDR_goal/10);
number_of_tau = -log(10^(-1*(SNDR_goal/20)));
maxtime = 1e9*(1/fs)*(1/2)/number_of_tau;
window1 = kaiser(Npoints,30);
band_edge = floor((bw/fs) * Npoints)+1;

% Starting values of Coefficients
% Cs    1e-12 (pF)
% gm    1e-4  (100's uV/Amp)
% ro    1e6   (Mohm)
% Ibias 1e-4  (100's uAmps)
coef(1) = 4.5;
coef(2) = 33;
coef(3) = 1;
coef(4) = 5;

% The optimizer will try to minimize Cs, gm ro, and Ibias

% Lower and Upper Bounds on the Coefficients
VLB  = [3.5 10 .2 .5];
VUB  = [5 34 1 10];

% Set up Optimization Options: Used to Define Tolerances and Step Sizes
options = foptions;
options(1) = 1;
options(2) = .1;
options(3) = .01;
options(4) = .1;
options(14) = 1000;
options(16) = .1;
options(17) = 1;

[C,dummy]= constr('param',coef,options,VLB,VUB,[],Npoints,M,Supply_factor, ...
                  simtime1,SNDR_weight,maxtime,SNDR_goal,window1,band_edge);
```

Figure 31.15. Matlab optimization block: *optim.m.*

```
% File: param.m

function [L2,g] = param3o(coeff,Npoints,M,Supply_factor,simtime1,SNDR_weight, ...
                          maxtime,SNDR_goal,window1,band_edge);

% Running Simulink to get the stream
  disp('Simulating...');

inputs = simset('SrcWorkspace','current','DstWorkspace','base');

%Creating the Variables for Interface with Simulink

c = [(coeff(1)*1e-12) (coeff(2)*1e-4) (coeff(3)*1e6) (coeff(4)*0.5e-4)];
Cs = c(1)
gm = c(2)
ro = c(3)
Ibias = c(14)

[tsim,xsystem,yout]=sim('Non_ideal_modulator',simtime1,inputs);

simtime = length(yout);
hy = (2*(abs(fft(yout(simtime-Npoints+1:simtime).*window1,Npoints))).^2)/((Npoints)*(Npoints-1));

% Additional Factors for use in Time Constant Calculation

Cp1 = 1e-12;
Ck1 = .75e-12;
Clp1 = 2.346e-12;

% Time Constant on Each Phase

tau11 = ((4*c(1)*Cp1) + Clp1*(4*c(1) + Cp1))/(c(2)*4*c(1));
tau12 = (Clp1 + Ck1 + 4*c(1)*(2*c(1) + Cp1)/(6*c(1) + Cp1))/((c(2)*4*c(1))/(6*c(1) + Cp1));

tau = 1e9*max(tau11,tau12);

% Calculate SNDR with Kaiser Window

noise1 = sum(hy(1:(M+1-floor(19/2)-1)));
noise2 = sum(hy((M+1+floor(19/2)+1):band_edge));
noise = noise1 + noise2;
noise = noise./(((.402)\^2);
peak = (hy(M+1))/.2276^2);
sndr = 10*log10(peak/noise);
mag = peak/noise;

L2 = (abs(1/mag))*SNDR_weight

disp('Done.');

% ******* set up constraints ************
% constr <= 0

constr1 = tau - maxtime        % Tau is less than the maximum allowable time
constr2 = -tau                 % Tau is greater than zero
constr3 = -sndr + SNDR_goal    % SNDR is greater than the SNDR goal

g = [constr1 constr2 constr3];
```

Figure 31.16. Matlab evaluation block: *param.m.*

It is in this function where the initial conditions along with the bounds on the variables are set, which will be discussed in Subsections 31.5.2 and 31.5.3. It is also in this function where all preliminary calculations are done. For example, the windowing function for the Fast-Fourier Transform (FFT) used to calculate the SNDR is created here. This is created only once and the vector is passed to the Matlab evaluation block each time an SNDR is calculated. By creating the

window in this function rather than repeatedly recalculating it in the evaluation block, valuable CPU time is saved.

The function "param.m" (Figure 31.16) basically implements the Matlab evaluation block and the Simulink system block. In fact, the Simulink System Block is implemented through a function call to the Simulink file "Non_ideal_modulator" seen in Figure 31.14. The Matlab evaluation block, which is the remainder of the file, receives the information gathered from the Simulink simulation, and calculates an error based on how far this is from the desired result. Furthermore, several constraints can be placed on any of the variables or values derived from them. For example, the settling time of the integrator is calculated in this function. A constraint is placed on it, forcing the optimization to continue until the settling time is less than the maximum settling time allowable to still achieve the desired performance. Further, constraints in this function force the settling time to be greater than zero (a condition that prevents the use of negative numbers), and force the optimizer to continue until the desired SNDR performance is achieved.

The final function "slew.m" (Figure 31.17), represents the code that was used to implement the Matlab function which accounts for bandwidth and SR limiting. This code is specifically tailored to the third-order $\Delta\Sigma$ modulator seen in Figure 31.14, however, it can be easily adapted for other designs. This function takes in the inputs C_s, g_m, r_o, I_{bias}, u and v (as seen in Figure 31.12), where u is the current input to the integrator, and v is the previous output of the integrator. This function is basically used to properly implement equation (31.15) in its entirety. It is important to note that the input u is already scaled by a factor of 1/4 (as seen in Figure 31.14), and therefore the quantity G in equation (31.15) is set to one.

31.5.2. Initial Conditions

A good set of initial conditions, or an adequate bounding of the optimized variables is needed, for improved optimization efficiency, and to increase the chances of obtaining a convergent solution.

The best set of initial conditions and bounds will usually come from the designer, as a designer usually has a better intuitive understanding and more knowledge concerning the feasibility of certain values. For example, a simulator may solve for a capacitance of 1 nF, but a designer knows that such a value is infeasible or undesired. The task then shifts to the designer for obtaining a good set of initial conditions and bounds.

In addition, the knowledge of transistor parameters such as r_o and g_m can help to obtain all of the required information necessary for setting bounds on variables. Preliminary simulations in SPICE can solve this problem quickly and efficiently. For example, the measurement of r_o for a single transistor

```
function [out]= slew(u,Cs,gm,ro,Ibias,v)

% u is current input
% v is previous output

% Sampling Frequency
Fs = 6.144e6;
Ts = (1/Fs);
Tsn = 1;              % All Equations are Normalized to Ts

% Capacitors Used in the Integrator
% Note: Gain of integrator is 1/4

Clp = 2.4e-12;        % OTA Output Capacitance
Cp  = 1e-12;          % UTA Input Capacitance
Ck1 = .75e-12;        % Capacitance of Next Stage

% Effective Load
CL = Clp + Ck1 + (4*Cs*(2*Cs + Cp))/(6*Cs + Cp);

% Slew Rate of the OTA (V/sec)
Sr = Ibias/CL;
SR = (Sr/Fs);

% Unity Gain Bandwidth of the OTA
ugbw = gm/(2*pi*CL);
UGBW = (ugbw/Fs);

% Scale Input to Integrator
d = u - v;

% Time Constant
tau = 1/((2*pi*UGBW*4*Cs)/(6*Cs + Cp));

% Slew Time: Implement Slewing Equations Here
to = ((abs(d))/SR) - tau;

f = abs((d/tau));
if abs((d/tau)) < = SR

    outpt = v + d + (1 - exp(-Tsn/(2*tau)));
else

  if to < (Tsn/2)

      output = v + d + (SR*to - d)*exp((to-(Tsn/2))/tau);

  else
      if d > 0
         outpt = v + (Tsn/2)*SR;
      else
         outpt = v - (Tns/2)*SR;
      end
  end
end

out = output;
```

Figure 31.17. Matlab function: *slew.m*.

is a trivial matter, and based on this value, a range of values for the output resistance of an OTA can be constructed. Furthermore, feasible bounds for g_m can be found by simulating a differential pair, and sweeping the bias current. This process is very simple and quick, and only needs to be done once for each new technology. Bounds can also be created based on the designer's preference. For example, if a capacitance is desired, then upper and lower bounds can be set so that no solution will contain too low (put the lower bound equal to that of the minimum sized matchable capacitor allowable in the technology) or too high a capacitance for the designer's liking. Once bounds have been chosen for each of the key parameters, choosing the initial conditions is not so arduous a task, as any values within those bounds can be taken as a starting point.

31.5.3. Additional Factors

There are additional factors that can alter the behavior of the system. As examples, the input and output capacitances of the amplifier along with the loading capacitance of the common-mode feedback circuit (CMFB) can affect the settling times of the integrators. These parameters can be accounted for by lumping them in with other key parameters. For example, C_L can be altered to reflect all the additional loading mentioned. The inclusion of these additional factors can be left to the discretion of the designer, or can be part of a larger iterative design procedure.

31.6. Summary of Simulation Results

This optimization procedure was applied to the design of a 16-bit (SNDR = 98 dB), third-order $\Delta\Sigma$ ADC, in which the primary objective was to design a lowpass, audio-band (bandwidth of 24 kHz), single bit, modulator, with an oversampling ratio of 128. The choice of coefficients used in the Simulink diagrams is purely an architectural design consideration and will not be discussed here. The focus remains on the performance of the system with the modeled non-idealities.

An ideal Simulink simulation of this modulator was run in order to determine the maximum achievable performance of such a system. The simulation yielded a peak SNDR of 105.72 dB, and a dynamic range of 112 dB, which can be seen in Figure 31.18(a). After the addition of the models that were presented in Section 31.4, the Simulink diagram was constructed (Figure 31.14), and the optimization routine was run. The process resulted in feasible design values ($g_m = 3.3\,\text{mA/V}$, $r_o = 963\,\text{kW}$, $I_{bias} = 458\,\text{mA}$ and $C_S = 4.5\,\text{pF}$, for an effective load of 8.2pF). An SNDR curve, which reflected the modeled non-idealities was produced, and can be seen in Figure 31.18(b). The simulation of the non-ideal modulator yielded a peak SNDR of 99.45 dB, and a dynamic range of 102 dB. This system was verified in SPICE through a series of simulations.

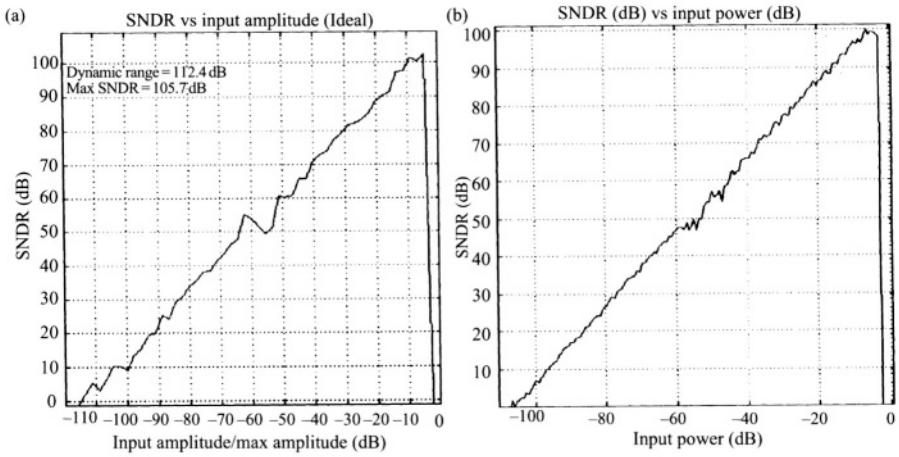

Figure 31.18. SNDR plots for third-order modulator in simulink. (a) Ideal and (b) non-ideal.

31.7. A Fully Coded ΔΣ Modulator Design Example

A brief example will be presented to further clarify the design process. This example is fully reproducible in Matlab and Simulink, and can be used to get an understanding of the methods needed for the design. The same third-order $\Delta\Sigma$ modulator will be used to design for kT/C noise requirements. The modulator will operate at 6.144 MHz, with a bandwidth of 24 kHz, have 16-bit performance, with a power supply of 2.5 V.

In order to begin, one can break the process down into several simple steps to be followed:

1 Identify/select the parameters to be optimized.

2 Create/use Simulink models to implement the non-idealities causes by the chosen parameters.

3 Select the performance criteria, and formulate relevant constraints. Choose the initial conditions, upper and lower bounds, and integrate any relevant additional factors.

For this design, the effect that will be examined is kT/C noise. This noise relates to the input capacitors C_S (as explained in Subsection 31.4.1), and as such is the parameter that will be optimized.

The step that follows involves designing the kT/C noise block for the integrator. The model used is based on the one seen in Figure 31.6, but requires some customization for this design. The first issue to deal with concerns the $f(u)$ block, and its variables (see Subsection 31.4.1). For this example, only one temperature point will be examined, so a fixed value of 300 degrees Kelvin

can be used for *T*. On top of this, C_S is needed in the formula. For the purpose of this optimization routine, C_S will be defined as *c*(1). Therefore, *c*(1) must appear in the equation. This allows for the optimization routine to directly interface with Simulink. Furthermore, since a normalized system was constructed in Simulink (i.e. the power supply is normalized to +1 V and –1 V), a value for *Supply_factor* should be chosen. Assuming that the power supply is 2.5 V, leads to a selection of 1.25 for *Supply_factor*. Upon the customization of the Simulink model, a full system diagram should be built. For this example, the diagram can be seen in Figure 31.19, and includes the Simulink kT/C modeling.

The next step that must be taken into account involves deciding what performance criteria will be used to determine a successful completion of the optimization routine. For simplicity, SNDR will be used as the sole performance indicator. In order to calculate this value, a Simulink system must be run, its output data points collected, and an FFT taken. For this modulator, 16 bits of performance is required (98 dB), and as a result, a constraint is placed on SNDR forcing it to be above 98 dB.

Finally, it is necessary to determine the initial conditions and bounds on the parameter C_S. Depending on the technology used, there is a minimum value of capacitance that can be manufactured with a certain pre-known degree of accuracy (usually 20%). Therefore, this value should be used for the lower bound on the capacitance. In this case 250 fF was chosen. The upper bound is usually limited by area constraints. In order to minimize area, one wants to use the smallest capacitance possible. In this case, the upper bound on capacitance was limited to 5 pF. A final decision remains concerning the selection of an initial condition. Basically, one can choose any value within the previously determined bounds.

At this point, there is enough information to create the optimization files. In the first file, seen in Figure 31.20, the system specifications are listed, the Kaiser window for the FFT is calculated, the initial conditions and finally the upper and lower bounds are set. The values listed correspond to numbers that are needed to make various calculations. For example, *Supply_factor* is needed for the kT/C noise block. *Npoints* refers to the number of points to be used in the FFT, while *fs* and *bw* correspond to the sampling frequency and bandwidth of the modulator. The value *band_edge* is a constant used to determine the last in-band point to be considered when calculating the FFT. The value *M*, determined by coherency, represents the bin in which the input signal lies. In this case, a frequency of 4312.5 Hz corresponds to an *M* of 23. Also defined in this file are the optimization options. The *options* vector contains the parameters used in defining the display format, coefficient accuracy, termination accuracy, constraint accuracy, the maximum number of iterations and the maximum step-size. Definitions and guidelines for choosing these values can be found in [10].

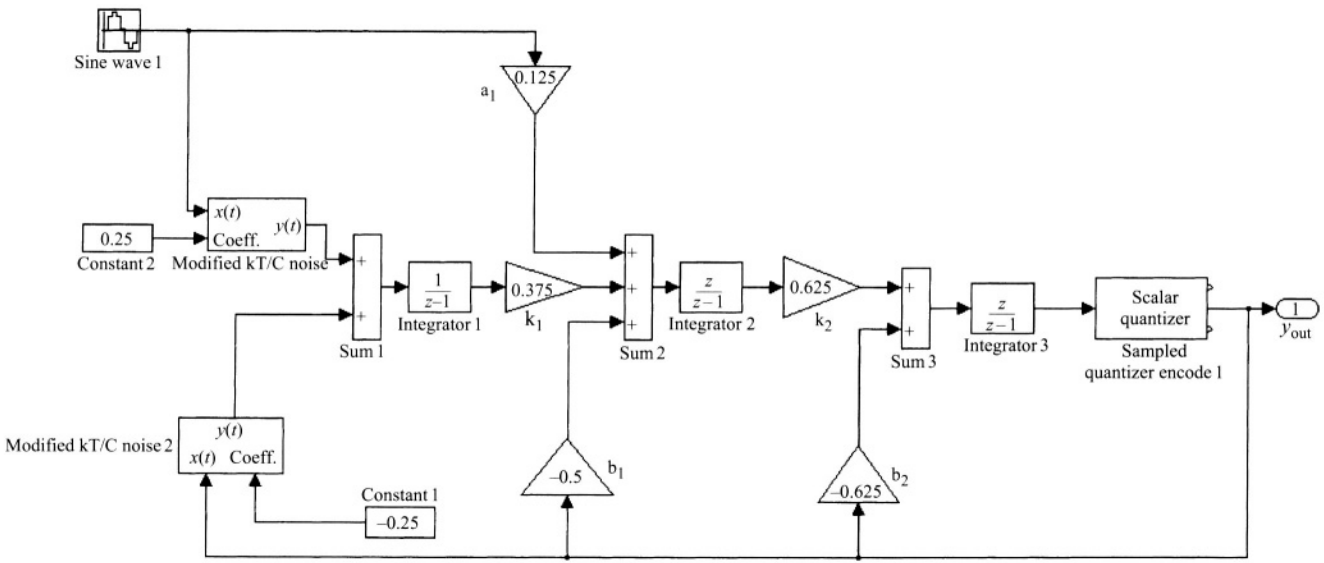

Figure 31.19. **Simulink system diagram with kT/C noise models ("*KTC_optim*").**

```
% File: optim_kTC.m

function [c] = optim_kTC();

warning off;

% Calculation of Constants

M = 23;              % Frequency of 4312.5 Hz
Npoints= 2^15;
Supply_factor = 1.25;
fs = 6.144e6;
bw = 24e3;
window1 = kaiser(Npoints,30);
band_edge = floor((bw/fs)*Npoints)+ 1;

% Starting values of Coefficient
% Cs    1e-12 (pF)
coef(1)= 3.0;

% Lower and Upper Bounds on the Coefficients
VLB = [0.25];
VUB = [5.0];

% Set up Optimization Options: Used to Define Tolernaces and Step Sizes
options = foptions;
options(1) = 1;
options(2) = .1;
options(3) = .01;
options(4) = .1;
options(14) = 1000;
options(16) = .1;
options(17) = 1;

[C,dummy]=constr('param_KTC',coef,options,VLB,VUB,[],Npoints,M,Supply_factor,window1,band_edge);
```

Figure 31.20. File "optim_kTC".

The second file, which is called in the last line of the previous one, can be seen in Figure 31.21. This file contains the call to Simulink model "KTC_optim", which was seen in Figure 31.19, the error calculation, and the constraints. The error condition is chosen to get the SNDR as close to 98 dB as possible. Dividing the V/V representation of 98 dB by the current SNDR in V/V does this. Ideally, this will give a value of one. As a result, one is subtracted to indicate that there is no error when this is achieved. This formulation penalizes SNDRs greater than 98 dB, but this is necessary to ensure that the smallest C_S possible is used. The constraint forcing the SNDR to be greater than 98 dB, can also be seen in this file.

By creating these two files along with the simulink diagram, this system can be simulated in Matlab. Once run, the optimization routine should quickly find a value for the minimum capacitance needed at the input stage. In this case, the result should be approximately 3.25 pF, and should take approximately 130 seconds of CPU time.

Based on this example involving kT/C noise along with the information gathered from the discussions in earlier sections, one has the necessary tools and background to easily expand upon these principles. Further, parameters that influence the non-idealities of the system, along with additional constraints to

```
% File: param_kTC.m

function [L2,g] = param_kTC(coeff,Npoints,M,Supply_factor,window1,band_edge);

% Running Simulink to get the stream
  disp('Simulation...');

inputs = simset('SrcWorkspace','current','DstWorkspace','base');

%Creating the Variables for Interface with Simulink

c(1) = [(coeff(1)*1e-12)];
Cs = c(1)

[tsim,xsystem,yout]=sim('KTC_optim',45700,inputs);

simtime = length(yout);
hy=(2*(abs(fft(yout(simtime-Npoints+1:simtime).*window1,Npoints))).^2)/((Npoints)*(Npoints-1));


%Calculate SNDR with Kaiser Window

noise1 = sum(hy(1:(M+1-floor(19/2)-1)));
noise2 = sum(hy((M+1+floor(19/2)+1):band_edge));
noise = noise1 + noise2;
noise = noise./((.402)^2);
peak = (hy(M+1))/(.2276^2);
sndr = 10*log10(peak/noise)
mag = peak/noise;

L2 = abs(((10{\^}(98/10))/mag) - 1)

disp('Done.');

% ******* set up constraints ***********
% constr <= 0

constr1 = -sndr + 98    % SNDR is greater than the SNDR goal

g = [constr1];
```

Figure 31.21. File "param_kTC".

more thoroughly define performance, can easily be added to develop a more accurate and comprehensive design.

31.8. Conclusion

A new design methodology for analog or mixed-signal integrated circuit components was presented, along with the benefits of a top–down optimization procedure. Foremost among these benefits was a shorter design cycle, along with ease of implementation and reproducibility. A major advantage of having adopted such a design strategy was its universal applicability to any design problem, provided that one has the ability to obtain formulas or rules of thumb to help guide the process. Once these formulas and rules of thumb have been obtained or decided upon, Matlab and Simulink could be used to model them, and an optimization procedure could be conceived, as per the guidelines presented in this chapter. In order to more easily understand and apply this procedure, Simulink modeling along with several design procedures and considerations were presented. Furthermore, the design of a $\Delta\Sigma$ modulator using

this methodology was carried out to more concretely illustrate the benefits of a top–down design methodology using Matlab and Simulink.

References

[1] S. R. Norsworthy, R. Schreier and G. C. Temes, *Delta–Sigma Data Converters: Theory, Design, and Simulation.* New York: IEEE Press, 1997.

[2] Semiconductor Industry Association, *The National Technology Roadmap for Semiconductors: 1999 Edition.* Austin, Texas: International SEMATECH, 1999.

[3] *SIMULINK and MATLAB User's Guides,* MathWorks Inc., 1997.

[4] R. Harjani, R. A. Rutenbar and L. R. Carley, "OASYS: a framework for analog circuit synthesis", *IEEE Transactions on Computer Aided Design,* vol. 8, pp. 1247–1266, December 1989.

[5] F. El-Turky and E. E. Perry, "BLADES: an artificial intelligence approach to analog circuit design", *IEEE Transactions on Computer Aided Design,* vol. 8, pp. 680–692, June 1989.

[6] M. Degrauwe et al., "IDAC: an interactive design tool for analog CMOS circuits", *IEEE Journal of Solid-State Circuits,* vol. 22, pp. 1106–1114, December 1987.

[7] F. Medeiro, B. Perez-Verdu and A. Rodriguez-Vazquez, "A vertically integrated tool for automated design of $\Sigma\Delta$ modulators", *IEEE Journal of Solid-State Circuits,* vol. 30, no. 7, pp. 762–772, July 1995.

[8] T. E. Dwan and T. E. Hechert, "Introducing SIMULINK into systems engineering curriculum", *Proceedings of Frontiers in Education Conference, 23rd Annual Conference,* pp. 627–671, 1993.

[9] A. Azemi and E. E. Yaz, "Utilizing SIMULINK and MATLAB in a graduate non-linear systems analysis course", *Proceedings of Frontiers in Education Conference, 26th Annual Conference,* vol. 2, pp. 595–598, 1996.

[10] A. Grace, *Optimization Toolbox for use with MATLAB,* MathWorks Inc., 1990.

[11] E. Baha, "Modeling of resonant switched-mode converters using SIMULINK", *IEE Proceedings on Electronic Power Applications,* vol. 145(3), pp. 159–163, May 1998.

[12] A. S. Bozin, "Electrical power systems modeling and simulation using SIMULINK", *IEE Colloquium on the Use of Systems Analysis and Modeling Tools: Experiences and Applications* (*Ref. No. 1998/413*), pp. 10/1–10/8, 1998.

[13] A. Dumitrescu, D. Fodor, T. Jokinen, M. Rosu and S. Bucurenciu, "Modeling and simulation of electrical drive systems using MATLAB/ SIMULINK environments", *International Conference IEMD '99,* pp. 451–453, 1999.

[14] H. Hanselmann, U. Kiffmeier, L. Koster, M. Meyer and A. Rukgauer, "Production quality code generation from SIMULINK block diagrams", *Proceedings of the IEEE 1999 International Conference on Computer Aided Control System Design,* vol. 1, pp. 355–360, 1999.

[15] S. Brigati, F. Francesconi, P. Malcovati, D. Tonietto, A. Baschirotto and F. Maloberti, "Modeling sigma–delta modulator non-idealities in SIMULINK", *Proceedings of the IEEE Symposium on Circuits and Systems,* vol. 2, pp. 384–387, ISCAS '99.

[16] B. E. Boser and B. A. Wooley, "The design of sigma-delta modulation anaolg-to-digital converters", *IEEE Journal of Solid-State Circuits,* vol. 23, no. 6, pp. 1298–1308, December 1988.

[17] S. Rabii and B. A. Wooley, "A 1.8-V digital-audio sigma–delta modulator in $0.8\,\mu m$ CMOS", *IEEE Journal of Solid-State Circuits,* vol. 32, no. 6, pp. 783–796, June 1997.

[18] F. Medeiro, B. Perez-Verdu, A. Rodriguez and J. L. Huertas, "Modeling opamp induced harmonic distortion for switched-capacitor $\Sigma\Delta$ modulator design", *Proceeding of the IEEE International Symposium on Circuits and Systems,* vol. 5, pp. 445–448, London, UK, June 1994.

[19] G. Temes and LaPatra, *Introduction to Circuit Synthesis and Design.* New York: McGraw-Hill, 1977.