A Behavioral Simulation Tool for Continuous–Time $\Delta\Sigma$ Modulators

K. Francken, M. Vogels, E. Martens* and G. Gielen
Katholieke Universiteit Leuven, Dept. of Electrical Engineering, ESAT-MICAS
Kasteelpark Arenberg 10, B-3001 Leuven–Heverlee, Belgium
email: francken|vogels|emartens|gielen@esat.kuleuven.ac.be

Abstract— **Circuit–level simulation of $\Delta\Sigma$ modulators is a time–consuming task (taking one or more days for meaningful results). While there are a great variety of techniques and tools that speed up the simulations for discrete–time (DT) $\Delta\Sigma$ modulators, there is no rigorous methodology implemented in a tool to efficiently simulate and design the continuous–time (CT) counterpart. Yet, in todays low–power, high–accuracy and/or very high–speed demands for A–to–D converters, designers are often forced to resort to the use of CT $\Delta\Sigma$ topologies. In this paper, we present a method for the high–level simulation of continuous–time $\Delta\Sigma$ modulators that is based on behavioral models and which exhibits the best trade–off between accuracy, speed and extensibility compared to other possible techniques that are reviewed briefly in this work. A user–friendly tool, implementing this methodology, is then presented. Nonidealities such as finite gain, finite GBW, output impedance and also nonlinearities such as clipping, harmonic distortion and the important effect of jitter are modeled. Finally, experiments were carried out using the tool, exploring important design trade–offs.**

I. Introduction

A very popular and efficient way to perform A–to–D conversion for a wide range of applications is $\Delta\Sigma$ modulation [1]. Due to the oversampling nature of $\Delta\Sigma$ modulators, a large number of clock cycles are needed for one output sample [2]. The long CPU times needed to simulate these modulators with a circuit–level simulator has led to the use of high–level (behavioral) simulation to speed up the analysis ([3], [4], [5]) and even the synthesis ([6], [7]) phase. All these efforts have, however, been concentrated on the discrete–time case. The continuous–time case has been the focus of much less attention [8]. Nonetheless, designers often need to resort to the use of CT modulators, especially for low–power [9], high–accuracy and/or high–bandwidth applications [10], [11]. More information on the pros and cons of CT versus DT modulators can be found in [8].

None of the DT tools presented up till now, however, can handle the CT case. This paper addresses the lack of high–level simulation tools for CT $\Delta\Sigma$ modulators. To this end, we first survey and compare different techniques (section II). One method will then be selected that offers the best trade–off between simulation speed, accuracy and the feasibility to implement additional nonidealities. The selected methodology will be elaborated in section III. In section IV, an overview is given of both linear and nonlinear nonidealities that have been modeled. In section V, a discussion of a full–custom tool is presented that implements the chosen methodology in a very user–friendly way. Moreover, postprocessing and graphical visualisation are all seamlessly integrated in the tool, making it – to the best of our knowledge – the first one of its kind presented in open literature that addresses CT $\Delta\Sigma$ modulators.[1] Of course, successful CT modulators have been implemented on chip [12], but these rely more on *ad hoc* methods. Using the developed tool, experimental results are then demonstrated in section VI. Finally, in section VII, conclusions are drawn.

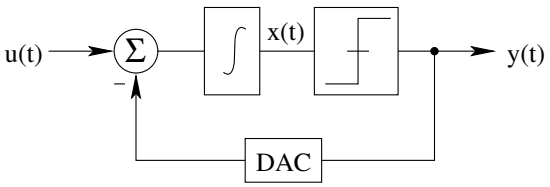

Fig. 1. A first–order CT $\Delta\Sigma$ converter.

II. High–level simulation of CT $\Delta\Sigma$ modulators

As already mentioned, circuit–level (SPICE) simulation of $\Delta\Sigma$ modulators is too time consuming in most practical cases. Especially in the case where parametric analyses (sweeps) are needed or when a complete synthesis loop is performed, a more efficient alternative needs to be found. Of course, this can be realized in different ways and different methods will show a different trade–off between the various performance metrics of concern. The most important metrics are simulation speed and accuracy. In addition, the extensibility of the method to include new nonidealities is also an important factor. In the following subsections three approaches are presented and compared that tackle this problem of extremely high (circuit–level) simulation time by increasing the abstraction level at which the circuit is simulated (see also [13]).

A. Network approach

In the network or macromodel approach the modulator is replaced by an electrical network that models its behavior. From this network, it is then possible to derive the differential equation:

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{f}[\mathbf{x}(t), u(t), y(t)] \quad (1)$$

$$y(t) = g[\mathbf{x}(t), u(t)] \quad (2)$$

which can then be solved numerically (using time–marching methods). The $\mathbf{x}$, u and y symbols represent the state vector, input and output of the modulator. This method has the disadvantage that a lot of calculations have to be done within one sampling period. Of course, the simulation is faster than a circuit–level simulation. Still, most dominant nonidealities can be modeled (e.g. finite gain and transconductance of the OTA, excess

*Research Assistant of The Fund for Scientific Research – Flanders (Belgium)

[1] See also: [8]: "... but to our knowledge there is no CT $\Delta\Sigma$ equivalent of a program like TOSCA." ([3]).

0-7803-7607-2/02/$17.00 ©2002 IEEE

TABLE I

COMPARISON OF DIFFERENT SIMULATION TECHNIQUES.

	Method		
	Network approach	**Behavioral model**	**DT Equivalence**
Simulation speed	slow	fast	fastest
Simulation accuracy	highest	high	low
Extensibility	high	reasonable	low
Available signals	all signals in 1 period	all sampled signals	in/output quantizer

loop delay, clock jitter, ...) and also implemented without too much difficulties.

B. Behavioral model

A second approach is to view the modulator as a system implementing a set of mathematical equations. This leads to a behavioral model of the modulator and its nonidealities. For the first–order system of figure 1 for instance, this gives:

$$x(kT) = x((k-1)T) + \frac{1}{T}\int_{(k-1)T}^{kT} u(\tau)\,d\tau - \frac{1}{T}\int_{(k-1)T}^{kT} y(\tau)\,d\tau \tag{3}$$

The simulation speed of this approach is comparable to the simulation of DT modulators. Of course, there is some speed penalty due to the calculations of the integrals, but this can be kept minimal if they are calculated analytically instead of numerically. With this approach sampled versions of both output and internal state variables are obtained.

C. Equivalent discrete–time model

Yet another approach is based on the fact that the clocked quantizer of a CT modulator implies that a discrete–time equivalent exists in such a way that the quantizer input is equal in both CT and DT cases on the sampling instants. There are two methods to derive the filter transfer function $H(z)$ of the equivalent DT modulator. In the **empirical** variant, one states that:

$$X(z) = G(z)H(z)U(z) - H(z)Y(z) \tag{4}$$

with $G(z)$ a prefilter, $X(z)$ the quantizer input and $U(z)$ and $Y(z)$ the input and output respectively. For a number of clock cycles, one of the above simulation approaches is used (e.g. the network approach as in [13], or even a circuit–level simulation). Using a least–squares approximation, a best–fit difference equation is then calculated. This can then be used in one of the available DT simulators. A big disadvantage here is the lack of a straightforward way to find the right terms to include in the best–fit difference equation [13].

In the **mathematical** approach, the impulse–invariant transformation is used ([14], [8]):

$$H(z) = \mathcal{Z}\left\{\mathcal{L}^{-1}\left\{H(s)\cdot H_{DAC}(s)\cdot\left(\sum_{k=0}^{\infty}\delta(t-kT)\right)\right\}\right\} \tag{5}$$

where $\mathcal{Z}$ and $\mathcal{L}$ are the Z–transform and Laplace–transform respectively and $H_{DAC}(s)$ is the transfer function of the D/A converter in the feedback path of the CT $\Delta\Sigma$ converter.

Both of the DT equivalence methods share the impossibility to model certain important nonidealities (e.g. clock jitter) with sufficient accuracy.

D. Comparing the approaches

To summarize this section, table I lists the various alternatives to circuit–level simulation of CT $\Delta\Sigma$ converters together with their impact on key simulation issues. As can be seen, the network approach gives the best accuracy, although this is also dependent on the integration method, but has the lowest speed. The DT equivalence method is the fastest (with a dependency on the chosen method for the initial samples and their number), but offers less accurate results and is unable to cover some important nonidealities. A very nice trade–off is offered by the behavioral model approach. The simulation speed is comparable to that of discrete–time simulators and most important nonidealities can be implemented (see section VI). This is the reason why we chose to implement this method in our simulator. Furthermore, all the sampled signals are available, which gives insight in the integrator output swings.

III. OVERVIEW OF THE PROPOSED SIMULATION APPROACH

The transfer function of an ideal integrator in the frequency domain is given by:

$$I = \frac{K}{s} \tag{6}$$

where K is the integrator gain (e.g. g_m/C). In the case of linear nonidealities this transfer function can be changed to e.g. a second–order one:

$$I = \widetilde{K}\frac{s-z}{(s-\lambda)(s-\mu)} \tag{7}$$

Here, we have 2 poles (λ and μ) and one zero (z). When the integrator approaches an ideal one, we have $\widetilde{K} \approx K$, $\lambda \approx 0$, $\mu \approx -\infty$ and $z \approx \pm\infty$.

To implement the behavioral model approach, we need a set of equations in the time–domain that permit to calculate the outputs of the integrators at each sampling moment based on the modulator input and the previous integrator outputs. A second–order model can be described in the time domain in a generic way with two state variables ($w(t)$ and $q(t)$) and one output variable ($x(t)$). For the first integrator (with poles λ and μ and with two input signals $i_1(t)$ and $i_2(t)$) the following set of equations can be derived:

$$\begin{bmatrix} w(t) \\ q(t) \end{bmatrix} = \mathbf{A}(t)\begin{bmatrix} w(t_0) \\ q(t_0) \end{bmatrix} + \mathbf{B}\begin{bmatrix} \mathcal{E}^{i_1}_{\lambda,0}(t) \\ \mathcal{E}^{i_2}_{\lambda,0}(t) \\ \mathcal{E}^{i_1}_{\mu,0}(t) \\ \mathcal{E}^{i_2}_{\mu,0}(t) \end{bmatrix}$$

$$x(t) = r_0 w(t) + r_1 q(t) + r_2 i_1(t) + r_3 i_2(t) \tag{8}$$

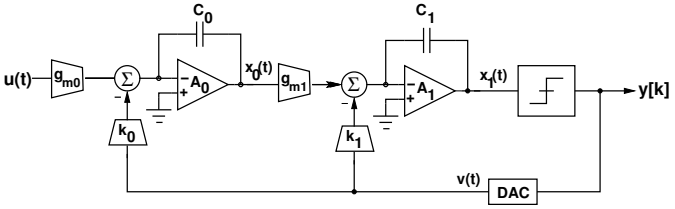

Fig. 2. Example of a second–order modulator model where the opamps have a finite gain.

where **A** describes the influence of the states at t_0 on the states at t of the first integrator:

$$\mathbf{A}(t) = \begin{bmatrix} a_0 & b_0 \\ f_0 & g_0 \end{bmatrix} e^{\lambda(t-t_0)} + \begin{bmatrix} a_1 & b_1 \\ f_1 & g_1 \end{bmatrix} e^{\mu(t-t_0)} \quad (9)$$

and **B** describes the influence of the inputs:

$$\mathbf{B} = \begin{bmatrix} c_0 & d_0 & c_1 & d_1 \\ h_0 & m_0 & h_1 & m_1 \end{bmatrix} \quad (10)$$

and where the function $\mathcal{E}^f_{\lambda,j}(t)$ is defined as:

$$\mathcal{E}^f_{\lambda,j}(t) = \int_{t_0}^{t} \frac{(t-\tau)^j}{j!} e^{\lambda(t-\tau)} f(\tau)\, d\tau \quad (11)$$

One can now map a specific integrator model including certain nonidealities to these generic coefficients $(a_i, b_i, \cdots, h_i, m_i)$ and to the poles (λ and μ). For a number of models this mapping has been coded symbolically. Given the values of finite gain, output impedance, ..., the tool automatically calculates the corresponding coefficients. For other models the user has the possibility to define the coefficients himself/herself. Similar expressions can be derived for a system with n integrators (n: 2, 3, ...), be it that the states of all integrators depend on the states and inputs of the preceding sample. Given a certain topology, one can calculate which combination of the (generic) coefficients must be used to derive these dependencies. For a number of topologies this is already coded.

As an example, the following equations describe a second–order modulator (see figure 2) where the opamps have a finite gain (A_0, A_1) and the D/A pulse is ideal (ranging from αT to βT, $0 \le \alpha < \beta \le 1$, T = sampling period). The states of the system are represented by $x_0[k]$ and $x_1[k]$.

$$x_0[k] = a_0^0\, x_0[k-1] e^{\lambda T} + c_0^0\, \mathcal{E}^u_{\lambda,0}[k] + d_0^0\, \mathcal{E}^v_{\lambda,0}[k] \quad (12)$$

$$x_1[k] = a_0^1\, x_1[k-1] e^{\lambda T} + a_0^0\, c_0^1\, T x_0[k-1] + c_0^0\, c_0^1\, \mathcal{E}^u_{\lambda,1}[k] + d_0^1\, \mathcal{E}^v_{\lambda,0}[k] + d_0^0\, c_0^1\, \mathcal{E}^v_{\lambda,1}[k] \quad (13)$$

with $u(t)$ the system input, $v(t)$ the feedback signal and the following definitions:

$$a_0^j = 1; \quad c_0^j = \frac{g_{m_j}}{C_{j,eq}}; \quad d_0^j = -\frac{k_j}{C_{j,eq}}; \quad others: 0 \quad (14)$$

where the superscript of the coefficients denotes the integrator.

$$C_{j,eq} = C_j \frac{(1+A_j)}{A_j} \quad (15)$$

$$\lambda = -\frac{g_{g_{m0}} + g_{k_0}}{(1+A_0)C_0} = -\frac{g_{g_{m1}} + g_{k_1}}{(1+A_1)C_1} \quad (16)$$

with g_{g_m} and g_k the output conductance of the g_m and k blocks respectively. Also note that we made no assumptions about the filter type (lowpass or bandpass) in our methodology. However, the previous derivations were presented for lowpass filters.

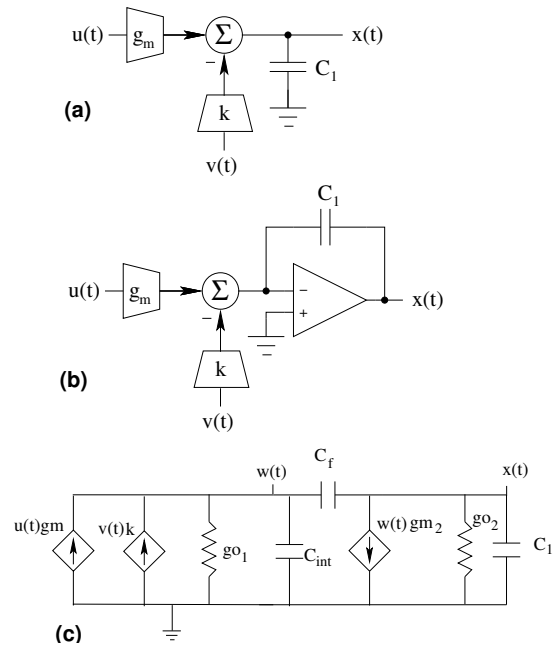

Fig. 3. Gm–C (a), OPAMP–C (b) implementations of the integrator and an example of a Gm–C macromodel (c).

IV. Implemented nonidealities

All major nonidealities are incorporated in the current version of the tool. The next three subsections give an overview of the implemented linear and nonlinear nonidealities, as well as jitter. Experimental simulation examples are presented in section VI.

A. Linear nonidealities

For the integrators, we have included several linear nonidealities, including finite gain, finite GBW, output impedance, parasitic capacitances, etc. Moreover, we have implemented models for different integrator types, namely Gm–C, OPAMP–C (see figure 3), OTA–C and current integrators. This list can, of course, be extended. The user can also plug in his own models by mapping them to the generic coefficients as shown in the previous section. All linear nonidealities can easily be included in the models.

Also, thermal noise and basic comparator nonidealities (offset, hysteresis) are covered. Moreover, any kind of DAC feedback pulse (ideal, finite rise and/or fall time) can be included using the behavioral modeling approach we have implemented.

B. Nonlinear nonidealities

Not only linear, but also nonlinear effects can be taken into account with the proposed method. This is not possible when using the discrete–time equivalent.

B.1 Clipping

The effect of clipping (finite output swing of the integrators) has been implemented as follows. Whenever the calculated voltage at the output of an integrator exceeds the maximum output swing V_{max} at a certain sampling instance t_n, the following calculations are done (see figure 4). First, the time instance (between two sampling moments) where the integrator output voltage reaches the clipping level is estimated. This value, t'_{cross}, is

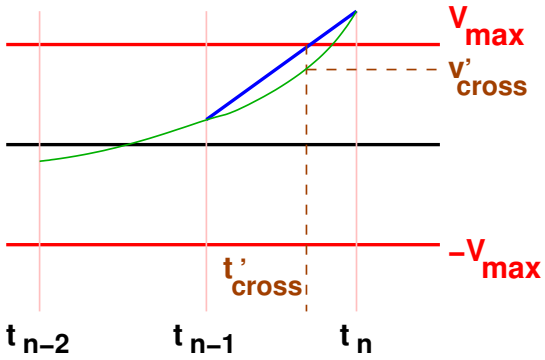

Fig. 4. Handling of integrator clipping.

based on a linear approximation as follows:

$$t'_{cross} = \frac{V_{max} - v_{n-1}}{v_n - v_{n-1}} \cdot (t_n - t_{n-1}) \tag{17}$$

The real voltage at this timepoint, v'_{cross}, is then calculated by integrating between t_{n-1} and t'_{cross}. The error, being the difference between the calculated value and V_{max}, can be kept smaller than a maximum allowed relative error percentage by iteratively repeating the calculation of (t'_{cross}, v'_{cross}). The state vector is then stored in sample n. Then, the integrator voltages are calculated at time t_n by integrating between t'_{cross} and t_n starting from the previous sample which was temporarily stored in sample n and the clipping integrator output fixed at V_{max}. This result is then the correct state vector, including the effect of clipping, and is overwritten on the nth sample.

B.2 Harmonic distortion

Harmonic distortion can be induced by several nonlinearities, most notably a nonlinear transconductance or nonlinear capacitors. To extend the models with the effect of harmonic distortion due to nonlinear capacitors, an approach similar to [15] is followed. Let the voltage–dependent capacitance be described by a second order Taylor series:

$$C(v) = C_0(1 + a_1 v + a_2 v^2) \tag{18}$$

where a_1 and a_2 are much smaller than unity. The current $i_{tot}(t)$ through this nonlinear capacitor can be calculated by

$$i_{tot}(t) = \frac{dQ(t)}{dt} = C_0\frac{dv}{dt} + 2C_0a_1v(t)\frac{dv}{dt} + 3C_0a_2v(t)^2\frac{dv}{dt} \tag{19}$$

where $Q(t)$ is the charge on the capacitance at time t. Assume that the voltage change on the capacitor between two adjacent samples (t_{n-1} and t_n) can be approximated by a straight line ($v(t) = \rho_{tot}t + v(t_{n-1})$). Without the nonlinearity present, the current through the capacitor would be $i_{lin}(t) = \rho_{lin}C_0$. This means that the effect of the nonlinearity can be modeled by an additional current i_{nl} (on top of the linear current i_{lin}) that is injected in the capacitor. This current can in a first order approximation be calculated to be:

$$i_{nl}(t) = i_{tot}(t) - i_{lin}(t) = C_0\rho_{nl} + 2C_0a_1(\rho_{tot}t + v(t_{n-1}))\rho_{tot} + 3C_0a_2(\rho_{tot}t + v(t_{n-1}))^2\rho_{tot} \tag{20}$$

with:

$$\rho_{nl} = -\frac{2a_1 + 3a_2v(t_{n-1})}{1 + 2a_1v(t_{n-1}) + 3a_2(v(t_{n-1}))^2} \cdot v(t_{n-1})\rho_{lin} \tag{21}$$

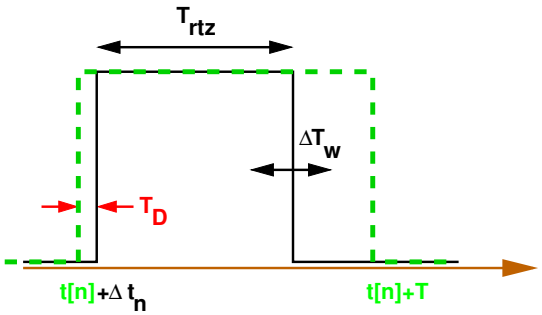

Fig. 5. Effect of jitter on the feedback pulse.

The procedure to calculate the influence of this nonideality is as follows. First, the states and outputs of the system are calculated assuming the system is linear. Then the extra currents are calculated (under the assumption that the response of the linear part can be approximated by straight lines) and the system's response to these currents is added to the states and outputs.

A similar procedure can be followed to include other weak nonlinearities (e.g. due to the nonlinear transconductance).

C. Jitter

Since the derived expressions (e.g. equations (8)–(11)) are dependent on time, the effect of jitter can easily be incorporated with sufficient accuracy. Different mechanisms of jitter exist (see figure 5) *Sampling jitter* has an effect on both the instant at which the input is sampled and on the beginning of the feedback pulse ($t[n] + \Delta t_n$). In general, the feedback pulse can be NRTZ (non return to zero) or RTZ (return to zero). In the RTZ case, the pulse width T_{rtz} equals a certain percentage of the sampling period T augmented by the *pulse width jitter* ΔT_w.
The feedback pulse can also be delayed by an amount T_D, which can vary by ΔT_D due to *delay jitter*. All these effects are important in CT $\Delta\Sigma$ modulator design and are incorporated into our models.

V. Full–custom CT $\Delta\Sigma$ simulation tool

The developed tool features a fully–fledged user interface and simulates rapidly due to the simulation strategy explained in section III. The simulator was coded in C. We will illustrate the tool's usefulness with some examples in the next section. Inputs to the simulator include topology, sampling frequency, input amplitude and other modulator specifics, but also simulator settings (e.g. no. of points, decimation, windowing, ...). In figure 6 we show a screenshot of the power spectral density plot of a simulation with the inputs of table II. The result for this ideal

TABLE II
Chosen inputs for the experiments.

Parameter	Value
Topology	CT single–loop, third order
OSR	32 / 48
sampling frequency	50 MHz
input frequency	0.1 MHz
input amplitude	0.33 V
reference voltage	1 V
integrator type	OPAMP–C / Gm–C

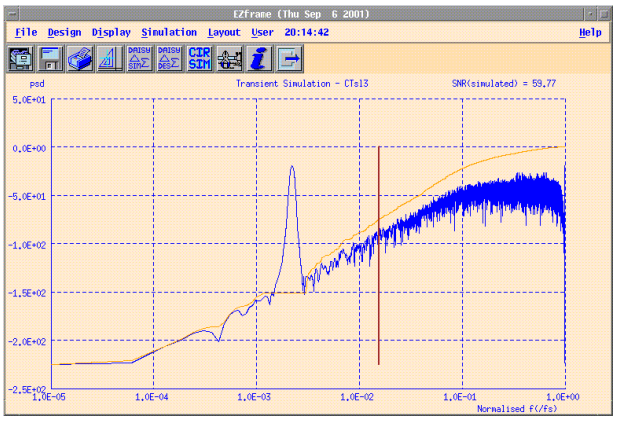

Fig. 6. An example of a power spectral density simulation.

case is similar to that of a DT $\Delta\Sigma$ modulator. One can clearly observe the noise shaping. The peak represents the input signal and the vertical line indicates the considered signal bandwidth. The upper (light-grey) curve is the accumulated noise spectral density. The resulting signal–to–noise ratio (SNR) is 59.77 dB which contains the integrated noise in the considered bandwidth.

The time for one simulation including postprocessing (FFT, visualization) is about 10 seconds. In order to interact with other tools or simulators, it is possible to control the inputs of the simulation and to read in the results (ASCII format) for further processing (e.g. in Matlab). In that way, our tool can be part of a larger design flow that also addresses other analog blocks.

VI. Experiments

In this section, we will use the tool to derive several interesting results. Figures in this section will use the tool's encapsulated postscript drivers for reasons of picture quality (and size) as opposed to the screenshot of the previous section. We will take the same input settings as in the previous section as a basis (see table II). In the first experiment, a NRTZ (non return to zero) type of DAC was used, while in the second experiment a RTZ (return to zero) type was chosen.

A. Experiment 1: Multi–variable sweep

The first example considers the effect of finite gain (1000), parasitic input capacitance (1 pF), output impedance (1K, 3K and 10K) and finite gain–bandwidth for the opamps. This last parameter was swept from 10 MHz to 140 MHz, resulting in the plot of figure 7.

The top curve is for an output impedance of 10K; the lowest curve corresponds to a 1K output impedance. Two conclusions can be drawn for this example: first, the output impedance should be sufficiently high (between 3K and 10K) in order not to degrade the performance, and secondly, the GBW of the opamps should be higher than about 50 MHz (where the SNR curves in figure 7 start to level off). This illustrates one of the major advantages of continuous versus discrete–time: the requirement on the opamp GBW is lower. Indeed, for discrete–time topologies the GBW should be at least 2 to 3 times the sampling frequency ([7]), while for continuous–time $\Delta\Sigma$ modulators it can be close to the sampling frequency.

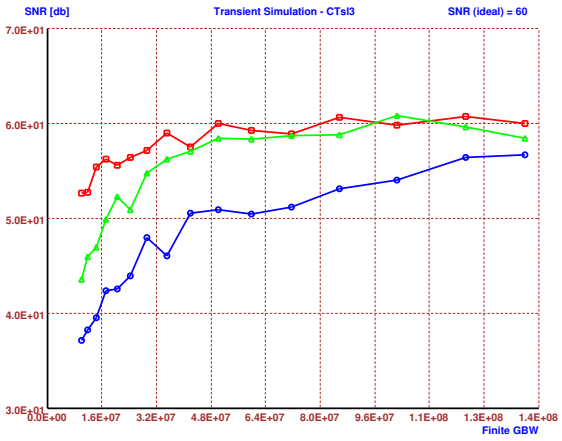

Fig. 7. The effect on SNR of a finite opamp GBW for different output impedances.

B. Experiment 2: The effect of jitter

Here, a Gm–C type of integrator (with a 2–stage OTA for the gm) and an oversampling ratio of 48 were selected. The finite gain, the first–stage gain and first–stage GBW were set to 1000, 10 and 1 GHz respectively. The integrator capacitance was 1 pF. A pulse width of 80 % (of the sampling period) was chosen for the RTZ DAC. The result in figure 8 shows a sweep of the pulse width jitter standard deviation from 1E-5 to 5E-3 (relative to T) for different values of the sampling jitter. The top two curves are for sampling jitter standard deviations of zero and 1E-3. It is clearly visible that the *sampling jitter* doesn't have a very big influence when using a RTZ DAC. Only for the lower curve, which has a relative standard deviation of 1E-2 already, the sampling jitter has an effect. In this example, the pulse width jitter standard deviation should be smaller than 0.01 % in order not to degrade the performance. As can be seen from this figure, the *pulse width jitter* has a large impact on modulators using a RTZ type of DAC.

C. Experiment 3: The effect of clipping

Figure 9 shows the result of a simulation with the model of clipping implemented as described in subsection IV-B.1. A sweep is performed with different settings for the clipping level. The results are compared to a simulation of a VHDL–AMS

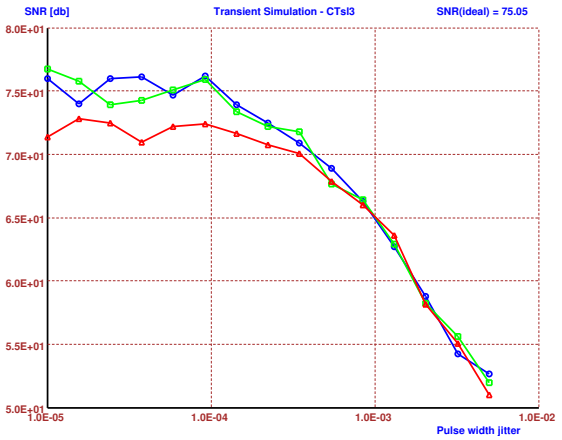

Fig. 8. The effect on SNR of a pulse width jitter for different values of the sampling jitter.

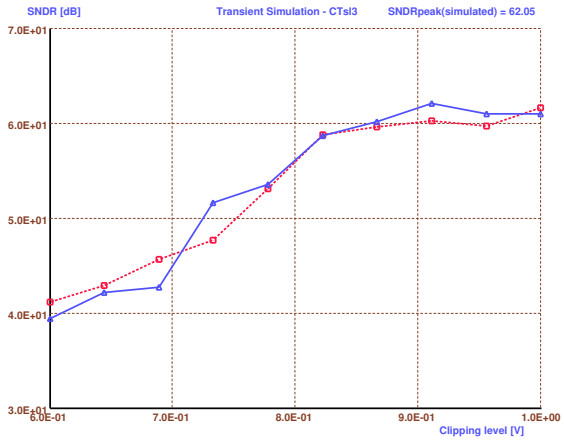

Fig. 9. The effect on SNR of integrator clipping. The dashed line is for the VHDL–AMS simulation.

model (using the network approach decribed in subsection II-A), where hard clipping is implemented by simply limiting the allowed output level of the capacitor. The two curves show good agreement. The VHDL–AMS simulation takes an average time of about 15 minutes for a single point, where our method needs only 10 seconds per point, a speed–up factor of about 90.

D. Experiment 4: The effect of harmonic distortion

Figure 10 shows the result of a simulation with a nonlinear capacitance such as described by equation (18), where a sweep of a_1 is performed. The quadratic voltage coefficients a_2 is set to zero for this simulation. The results are again compared to a simulation of a VHDL–AMS model of the same complexity, but with the nonlinearity implemented as is. This corresponds to the network approach as stated in section II. The two curves are in good agreement, validating our proposed method. The VHDL–AMS simulation takes about 10 minutes for a single point, where our method needs only 10 seconds, a speed–up factor of about 60.

All the above experiments show that the tool can produce useful results that give a designer insight in the required specifications of the building blocks.

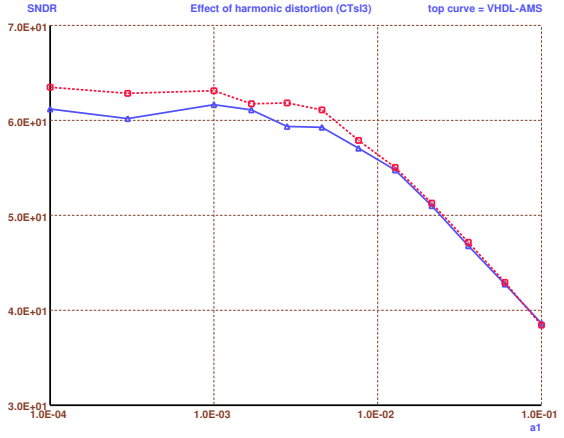

Fig. 10. The effect on SNR of harmonic distortion. The dashed line is for the VHDL–AMS simulation.

VII. Conclusions

Although a wide range of tools are available for the high–level analysis and even synthesis of discrete–time $\Delta\Sigma$ modulators, there is no efficient alternative for continuous–time $\Delta\Sigma$ modulators that can take all important nonidealities into account. Yet, for low–power, high–accuracy and/or high–bandwidth applications (e.g. VDSL), designers often have to resort to the use of continuous–time topologies. Noticing this trend, we have first given an overview of alternative simulation methods for the extremely long circuit–level simulations. Following this, a comparison and trade–off based on accuracy, speed and implementation of additional nonidealities was made. The behavioral model approach was shown to be the best strategy for the development of an accurate, extensible and fast simulator. We then implemented this method in a user–friendly tool that is (to the best of our knowledge) the first one for CT $\Delta\Sigma$ modulators published in open literature. Finally, using the developed tool, design–relevant experiments were carried out, showing the usefulness of the approach and illustrating the straightforward way of obtaining and exploring design trade–offs.

References

[1] Steven Norsworthy, Richard Schreier, and Gabor Temes, *Delta–Sigma Data Converters: theory, design, and simulation*, IEEE Press, 1996.

[2] K. Francken, M. Vogels, and G. Gielen, "Dedicated System–Level Simulation of $\Delta\Sigma$ Modulators," in *Proceedings Custom Integrated Circuits Conference (CICC)*, May 6 – 9 2001, pp. 349 – 352.

[3] V. Dias, V. Liberali, and F. Maloberti, "TOSCA: A User–Friendly Behavioural Simulator for Oversampling A/D Converters," in *Proceedings IEEE International Symposium on Circuits and Systems*, 1991, pp. 2677–2680.

[4] Augusto Marques, *High Speed CMOS Data Converters*, Ph.D. thesis, ESAT–MICAS, K.U.Leuven, Belgium, January 1999.

[5] K. Francken and G. Gielen, "Optimum System–Level Design of $\Delta\Sigma$ Modulators," in *Proceedings IEEE Benelux Workshop on Circuits, Systems and Signal Processing (ProRISC)*, November 25–26 1999.

[6] Fernando Medeiro, B. Perez-Verdu, and A. Rodriguez-Vazquez, *Top–Down Design of High–Performance Sigma–Delta Modulators*, Kluwer Academic Publishers, 1999.

[7] K. Francken, P. Vancorenland, and G. Gielen, "DAISY: A Simulation–Based High–Level Synthesis Tool for $\Delta\Sigma$ Modulators," in *Proceedings IEEE/ACM International Conference on Computer–Aided Design (ICCAD)*, November 5 – 9 2000, pp. 188 – 192.

[8] James Cherry and Martin Snelgrove, *Continuous–Time Delta–Sigma Modulators for High–Speed A/D Conversion*, Kluwer Academic Publishers, 2000.

[9] Eric van der Zwan, "A 2.3mW CMOS $\Sigma\Delta$ Modulator for Audio Applications," in *Proceeding International Solid State Circuits Conference (ISSCC)*, February 6–8 1997, pp. 220–221, 461.

[10] A. Hussein and W. Kuhn, "Bandpass Modulator Employing Undersampling of RF Signals for Wireless Communication," *IEEE Transactions on Circuits and Systems II*, vol. 47, no. 7, pp. 614–620, 2000.

[11] J. van Engelen, R. van de Plassche, E. Stikvoort, and A. Venes, "A 6th-Order Continuous-Time Bandpass $\Sigma\Delta$ Modulator for Digital Radio IF," in *Proceeding International Solid State Circuits Conference (ISSCC)*, February 15–17 1999, pp. 56–57.

[12] Lucien J. Breems, Eric J. van der Zwan, E. Carel Dijkmans, and Johan H. Huijsing, "A 1.8mW CMOS $\Sigma\Delta$ Modulator with Integrated Mixer for A/D Conversion of IF Signals," in *Proceeding International Solid State Circuits Conference (ISSCC)*, 1999, pp. 52–53.

[13] J. Cherry and M. Snelgrove, "Approaches to Simulating Continuous-Time Delta Sigma Modulators," in *Proceedings IEEE International Symposium on Circuits and Systems*, 1998, vol. 1, pp. 587–590.

[14] P. Benabes, M. Keramat, and R. Kielbasa, "Synthesis and Analysis of Sigma–Delta Modulators Employing Continuous–Time Filters," *Analog Integrated Circuits and Signal Processing*, vol. 23, no. 2, pp. 141–152, May 2000.

[15] Piet Wambacq and Willy M.C. Sansen, *Distortion Analysis of Analog Integrated Circuits*, Kluwer Academic Publishers, 1998.