

# DAISY

## $\Delta\Sigma$ Analysis and Synthesis

A simulation and design environment for  $\Delta\Sigma$  modulators

A SysConv IC&D deliverable

Kenneth Francken

December 21, 2001

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Goal . . . . .	1
1.2	Oversampling A/D Converters . . . . .	1
1.3	Simulation Strategies . . . . .	1
1.4	Modeling . . . . .	2
<b>2</b>	<b>The DAISY tool</b>	<b>4</b>
2.1	Program Details . . . . .	4
2.2	Supported Topologies . . . . .	4
2.3	Features . . . . .	4
2.4	Starting the program . . . . .	7
<b>3</b>	<b>High–Level Behavioural Simulation of <math>\Delta\Sigma</math> ADCs</b>	<b>8</b>
3.1	Dynamic Analysis . . . . .	8
3.2	Parametric Analysis . . . . .	9
	3.2.1 Sweeping nonidealities . . . . .	9
	3.2.2 Sweeping input parameters . . . . .	10
3.3	Time Domain Analysis . . . . .	10
<b>4</b>	<b>High–Level Synthesis of <math>\Delta\Sigma</math> ADCs</b>	<b>14</b>

# 1 Introduction

## 1.1 Goal

The goal of this document is to become familiar with high-level simulation and synthesis of a specific analog building block, namely a  $\Delta\Sigma$  A/D converter. To this end, we will first explain in a few words what we should know about this type of converter. Then, possible alternatives are discussed for simulating  $\Delta\Sigma$  converters. In a next subsection the modeling of the nonidealities is covered by means of an example.

In section 2, an introduction is given on the dedicated tool that implements the methodology explained further in this section. The following two sections then deal with a practical exploration of the tool for both simulation and synthesis respectively. First, in section 3, we will explore the advanced simulation capabilities of the tool. The last section is dedicated to a synthesis example.

## 1.2 Oversampling A/D Converters

It is assumed that the reader is familiar with the general concept of A/D converters. Also, it is beyond the scope of this text to discuss the possible architectural alternatives for implementing A/D converters. Instead, we will focus on oversampling and noise-shaping converters, also referred to as  $\Delta\Sigma$  converters.<sup>1</sup> Conventional methods often require high-precision circuit components and are sensitive to circuit nonidealities and component mismatches [2]. Oversampling and noise shaping techniques combined with DSP allow to trade-off resolution versus time. This typically allows to use relatively imprecise circuit components at the cost of extra digital filtering, which is not a problem in modern VLSI technologies.

As an illustration, figure 1 shows a cascade 2–1–1 topology without the digital noise cancellation logic. Figure 2 shows the same topology with the digital filtering included. The modeling of the different blocks will be briefly discussed in subsection 1.4.

## 1.3 Simulation Strategies

There are several places in a design flow where it is necessary to have a fast simulator. In our case, we specifically need the fast speed to iteratively synthesize the modulator at the architecture level. But also when simulating a number of nonidealities to verify some assumptions or when doing sweep analyses the speed is an important factor. Since  $\Delta\Sigma$  converters are oversampled architectures, the large number of samples at the output involves a long transient analysis if done with a conventional circuit simulator like SPICE, i.e. at the electrical level. Clearly, an alternative has to be taken here. Common approaches are macromodels, still too slow for a general design flow, or behavioral simulations. Although there is some penalty in accuracy, behavioral models can produce quite acceptable results considering the level of abstraction. The speed of behavioral simulation is, however, extremely faster than electrical or even macromodel simulation.

---

<sup>1</sup>For the interested reader, an excellent book is “Delta-Sigma Data Converters: theory, design, and simulation” [1].

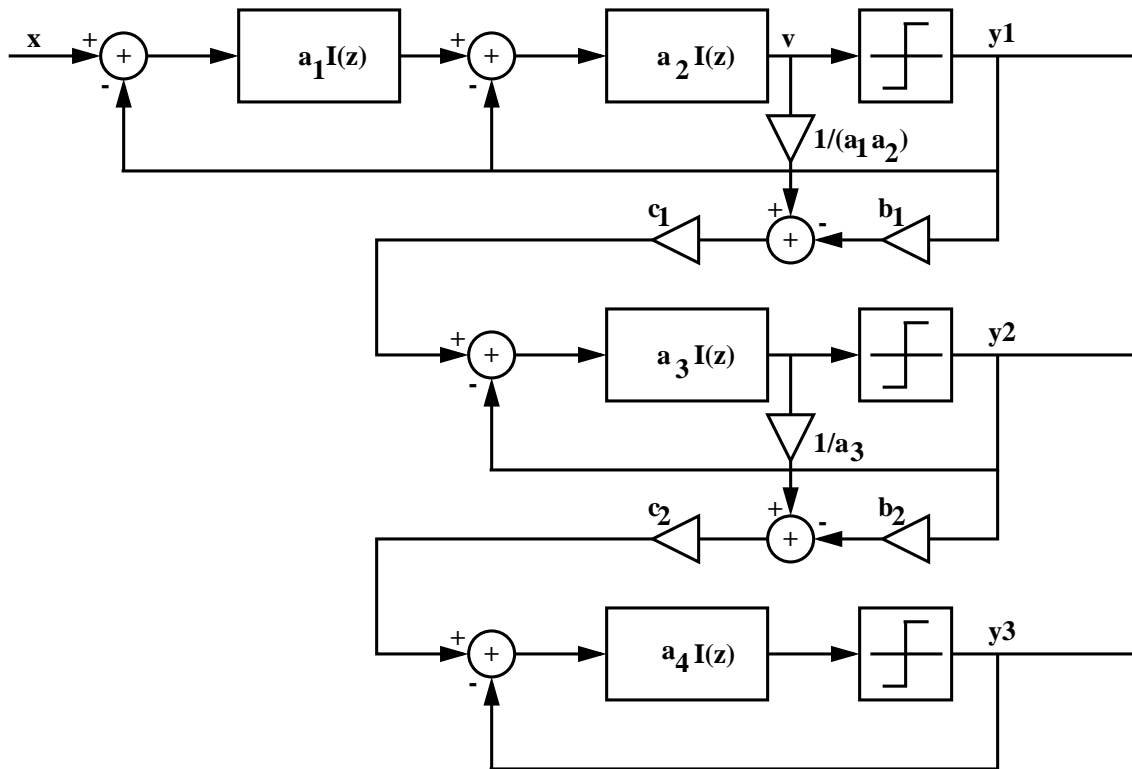


Figure 1: Block Diagram of a Cascade 2-1-1  $\Delta\Sigma$  ADC.

## 1.4 Modeling

In order to efficiently and successfully design a  $\Delta\Sigma$  ADC at a high level, an idea of the impact of the building block nonidealities is necessary. The models used in the behavioral simulation are derived by means of time-domain descriptions of the building blocks. The nonidealities are incorporated into these  $z$ -domain behavioral models. As an illustration, equation (1) shows the transfer function of an ideal integrator (see figure 3). A 1-clock-cycle delay is considered.

$$I(z) = \frac{z^{-1}}{1 - z^{-1}} \quad (1)$$

In general, the different physical causes of linear degradation change this ideal integrator transfer function in two ways. Firstly, they cause a gain error  $\alpha$  (smaller than 1) that is multiplied in the numerator of  $I(z)$ . Secondly, they cause a pole error  $\beta$  (smaller than 1). Ideally the pole is located at  $z$  equal to 1 (see equation (1)). Both these changes in the integrator transfer function will eventually introduce low-frequency quantization noise leakage. In addition, some nonidealities can also introduce an extra pole in the transfer function of the integrators. The resulting transfer function is then given by:

$$I(z) = \frac{\alpha z^{-1}}{1 - \beta z^{-1} + \gamma z^{-2}} \quad (2)$$

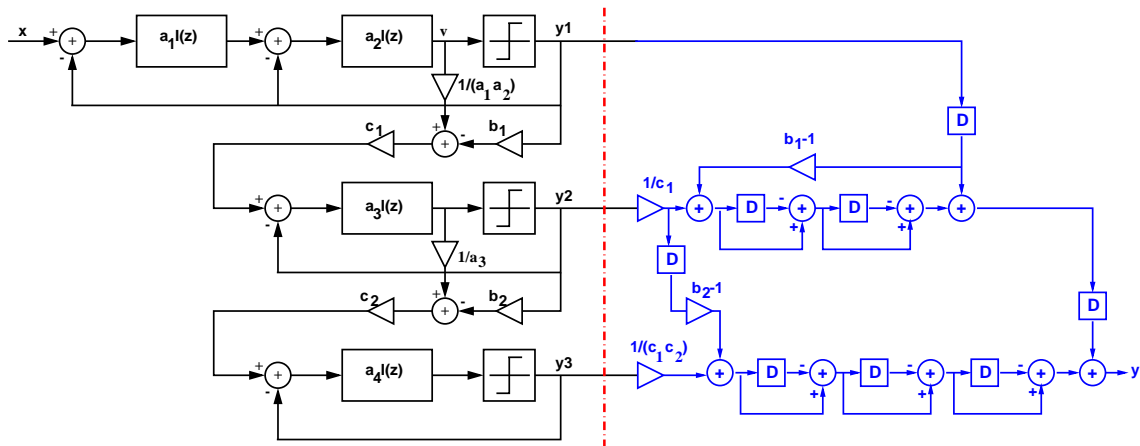


Figure 2: Same as figure 1, but now with the digital noise cancellation logic.

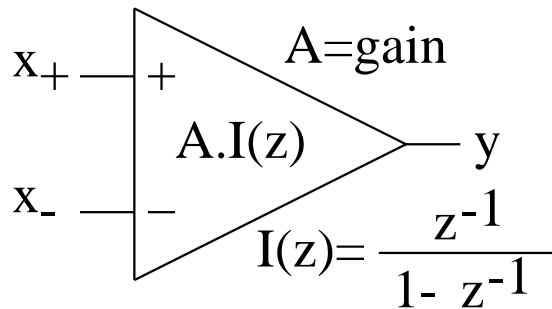


Figure 3: Ideal representation of an integrator.

It is beyond the scope of this document to go into the details of  $\Delta\Sigma$  ADC nonidealities and their exact modeling equations. For this we refer to deliverable [3]. The implemented nonidealities are summarised in the following subsection.

## 2 The DAISY tool

Lets now take a look at the specific implementation details of the tool. The subsequent two sections will then guide you through some practical examples.

### 2.1 Program Details

For the programming language we used standard C, where the models and simulator functions are a standalone core library of functions. The compiled C code ensures fast execution of the simulator algorithms while Motif<sup>®</sup> is used for offering a highly convenient and flexible user interface. Motif<sup>®</sup> is the industry standard graphical user interface (as defined by the IEEE 1295 specification) for UNIX, used on more than 200 hardware and software platforms.

The following platforms and operating systems are supported:

- Sun Solaris
- Intel Linux
- HP-UX

### 2.2 Supported Topologies

We will only consider converters based on switched-capacitor circuits, which are the majority of the implementations, and not continuous-time based converters. There are several general topologies for  $\Delta\Sigma$  ADCs that are supported in the tool:

- single bit, single loop topologies
- single bit, cascaded topologies
- multi bit topologies (both single loop and cascaded)

All these topologies are implemented with feedback loops. No feedforward topologies are supported at this moment. In this document we will mainly use single bit topologies for the examples.

### 2.3 Features

The DAISY tool has an easy to use graphical user interface, which will be shown in the following sections. Let's first take a look at the input window (figure 4) of the simulator. Pushing the left button (*Analysis Type*) displays a frame that allows to select the requested analysis type. The following types are supported:

- **time domain:** bitstream output waveforms



Figure 4: Input window of the  $\Delta\Sigma$  ADC simulator.

- **dynamic:** a frequency analysis resulting in a power spectral density plot of the signal, noise and accumulated noise
- **parametric:** a sweep analysis as a function of any input parameter or nonideality

In the following section we will experiment with the different types of analysis.

The second button (*Input Settings*) displays a frame allowing to set some basic parameters like the topology (see further), the oversampling ratio, input amplitude and frequency, the input type (DC or sinusoidal), etc. It is also possible to edit the coefficients ( $a_1$ ,  $a_2$ , etc.) of the integrator gains (see figure 1 for a cascade 2–1–1 topology). Furthermore, the nonidealities of the building blocks can be set as well (see further).

The following topologies are supported in the current version:

- sl1: single bit, single loop, first order
- sl2: single bit, single loop, second order
- sl3: single bit, single loop, third order
- sl4: single bit, single loop, fourth order
- c21: single bit, cascaded, 3rd order
- c31: single bit, cascaded, 4th order
- c22: single bit, cascaded, 4th order
- c211: single bit, cascaded, 4th order

Also the multi bit equivalents are supported and the same nomenclature is used as above with an 'm' prepended before the name.

The nonidealities that are implemented in the current version are:

- finite OTA gain
- finite OTA GBW
- finite OTA output swing
- non zero switch resistance
- capacitor mismatch
- comparator offset
- comparator hysteresis
- thermal noise

- DAC nonlinearity (for multi bit topologies)

The third button (*Simulator Options*) displays a frame where it is possible to select the number of points to simulate, whether or not the output should be decimated and which windowing function should be used to calculate the power spectral density.

## 2.4 Starting the program

The tool can be started from one of the supported computer platforms by issuing the following command:

```
/your_installation_dir/daisy
```

If you start the tool for the first time and no configuration file is present, one will be copied to your home directory. The name of the run-command file is `.Daisyrc` and it allows to specify your personal defaults for most options by changing it with any ASCII editor. If you don't want the file to appear in your home directory, you can move it to any other directory as long as you set the environment variable `KFCONFDIR` equal to this directory path. At this moment the online help (available from the help menu) is rather limited.

The following command-line options are supported:

Option (short)	Default	Explanation
-version		Prints the program name and version, then exits.
-help (-h)		Prints information on command line options, then exits.
-info		Prints information about this program, then exits.
-nogui	X	Don't use the graphical user interface.
-nolog	X	Disable logging to file (speed-up).
-noconfirmonexit	X	Don't ask confirmation on exit.
-config (-c)		Specify the rc file.
-exec (-x)	X	Specify the command file (no GUI).
-lesscolors	X	Reduce color hungry pixmaps(0..3).
-nosplash	X	Don't show the splash screen on startup.
-valid	X	Print expiration date.
-bgcolor (-bg)	X	Specify the background color.
-fgcolor (-fg)	X	Specify the foreground color.

### 3 High-Level Behavioural Simulation of $\Delta\Sigma$ ADCs

We will now explore some examples that illustrate the capabilities of the DAISY tool for high-level simulation of  $\Delta\Sigma$  modulators. In each of the following subsections a short description of the goal is formulated together with the output results. You can use the tool to reproduce the output results and try some experiments of your own.

#### 3.1 Dynamic Analysis

A dynamic analysis is in fact a time-domain analysis followed by a Fourier Transform. The result is a power spectral density plot as a function of the (normalised) frequency.

**Example 1** Let's say we want to view the simulation results corresponding to the following inputs:

- topology: cascaded 2–1–1, single bit
- oversampling ratio (OSR): 32
- sampling frequency: 50 MHz
- input frequency: 100 kHz
- input amplitude: 0.25 V
- reference voltage: 1 V
- number of transient points: 32768

After entering the above inputs, you should see a plot like the one given in figure 5. The lower line gives the noise power spectral density. The peak is the input signal. The fourth order noise shaping is clearly visible.

The resulting signal-to-noise ratio (SNR) is printed in the upper right corner. In our example the SNR is 88.63 dB.

In the above example, an ideal modulator was simulated. In practice, several nonidealities will influence the outcome of the SNR. Take, for instance, the thermal noise resulting from the sampling capacitor. This will give rise to a certain noise floor (opposed to the continuous noise shaping for low frequencies in figure 5).

**Example 2** Include the effect of the thermal noise by selecting the proper nonideality in the input window (figure 4). If a value equal to zero is filled in, the simulator will automatically calculate the thermal noise from the sampling capacitor ('kT/C noise'). It is also possible to give the 'sigma' of the thermal noise distribution, irrespective of the originating source (capacitor, switch thermal noise, ...). The result should be as shown in figure 6. Here we can clearly see the resulting noise floor and its impact on the SNR, which drops from 88.63 dB to 84.41 dB.

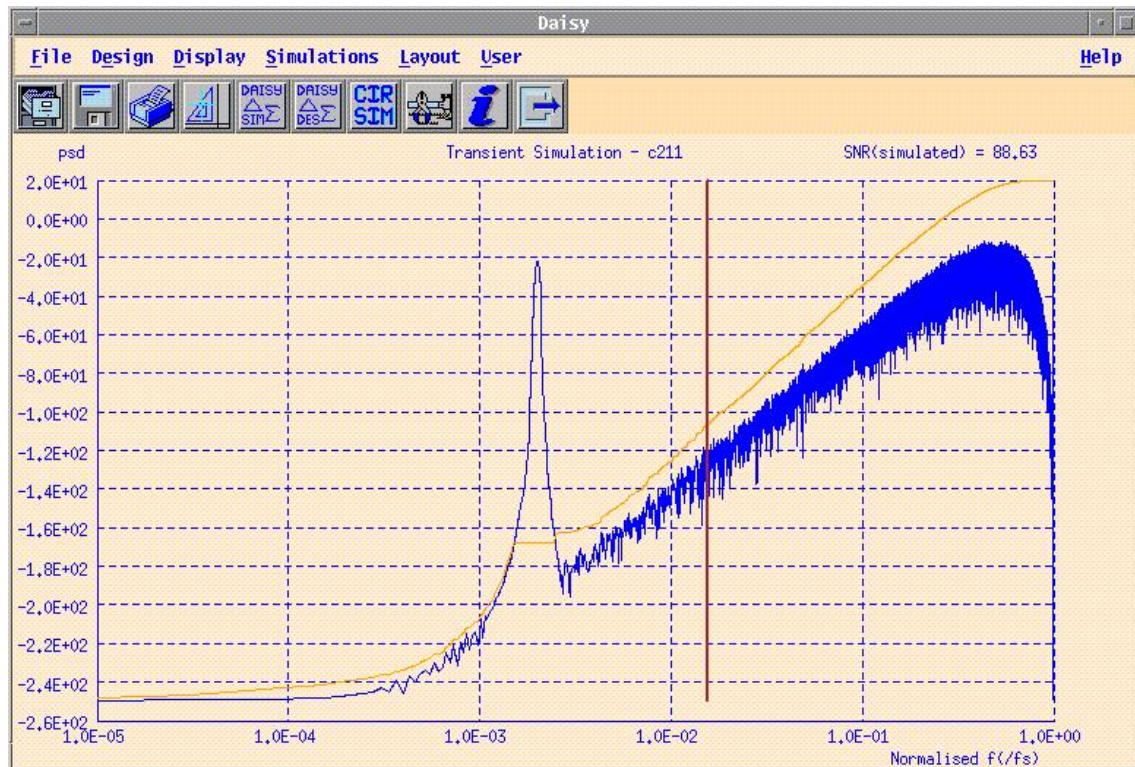


Figure 5: Output of the dynamic analysis for our example.

In the above examples, we assumed ideal integrators and comparators. In practice, this is of course not true. Figure 7 illustrates the effect of having an OTA with a finite gain of 1000 (60 dB) and a finite GBW (gain bandwidth product) of 100 MHz. We see that the SNR has dropped slightly down to 84.94 dB. Lower values of the gain can introduce harmonics.

## 3.2 Parametric Analysis

In order to get a good impression of the impact of a parameter on the performance one usually resorts to doing a sweep of the parameter value. This can be done in the tool by selecting '*Parametric Analysis*' in the input window as analysis type. Both sweeps of input parameters and circuit nonidealities are possible. We will show an example of both.

### 3.2.1 Sweeping nonidealities

In the previous subsection we simulated the topology with a nonideal OTA. We entered some values for the finite gain and finite GBW. It would, however, be interesting to see the effect of decreasing GBW for a whole range of values on one plot. In other words, we would like to see a parametric plot of the SNR as a function of the GBW. Figure 8 shows such a plot for a finite

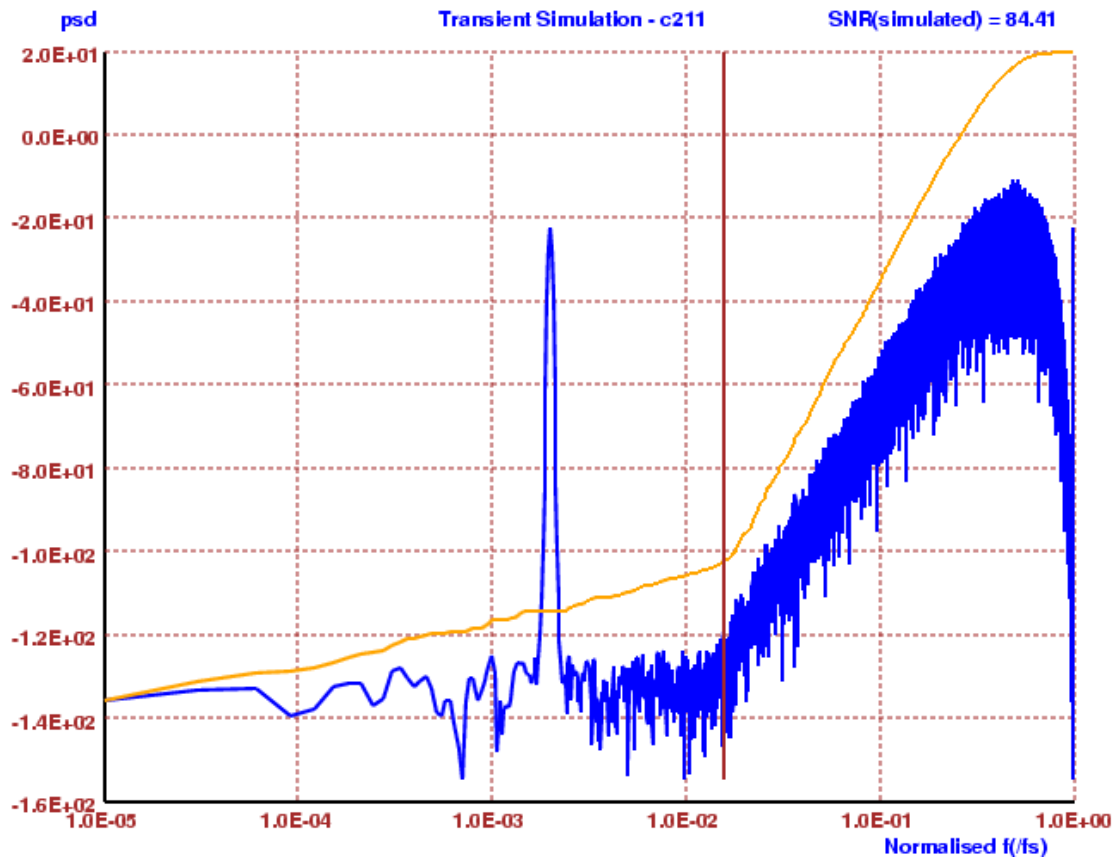


Figure 6: Equivalent of figure 5, now with  $kT/C$  noise included.

OTA gain of 1000 and the finite OTA GBW going from 100 kHz to 1 GHz (30 points). It can be seen that, in order not to have a SNR degradation due to the limited OTA GBW, we should select or design an OTA with a GBW of minimum 100 MHz for this example.

### 3.2.2 Sweeping input parameters

One of the typical performance parameters of a  $\Delta\Sigma$  modulator is its dynamic range (DR) or peak SNR. This can be easily derived by sweeping one of the input parameters.

The resulting typical overload curve of the modulator is shown in figure 9.

## 3.3 Time Domain Analysis

Selecting time domain analysis enables the user to enter a filename to save the (unfiltered) data. The waveform data being unfiltered means that in the case of cascaded topologies no digital noise

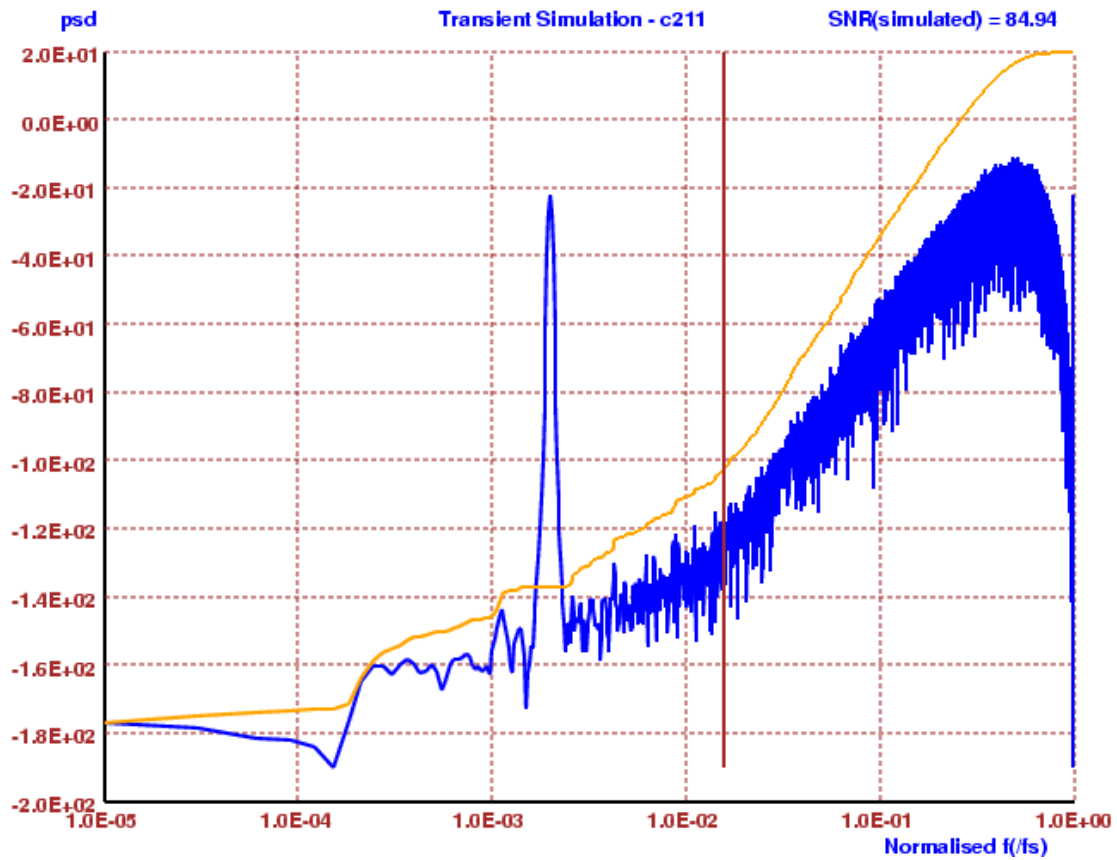


Figure 7: Same as figure 5 with finite OTA gain of 1000 and finite OTA GBW of 100 MHz.

cancellation is performed. The resulting square wave can then be plotted but this is currently not implemented in the tool. You can, however, examine the ASCII data and/or read it into another tool with plotting capabilities.

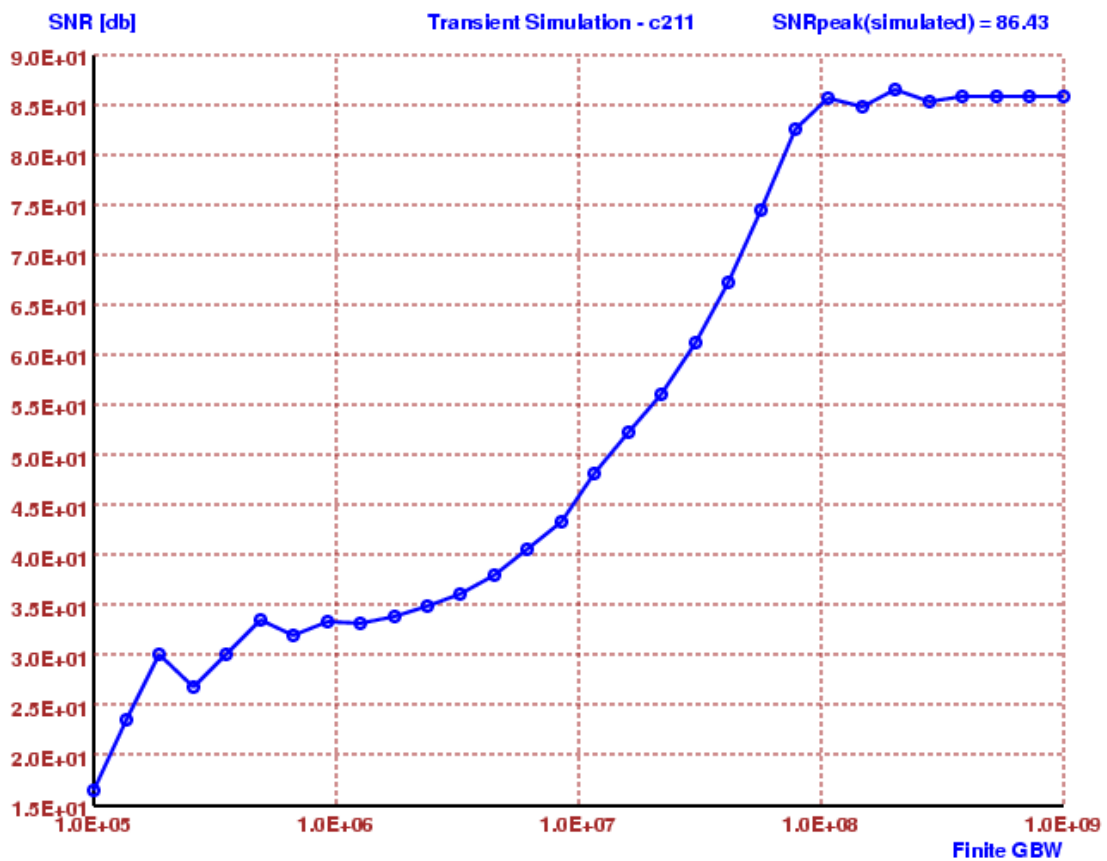


Figure 8: Parametric plot of the SNR as a function of GBW (finite OTA gain of 1000).

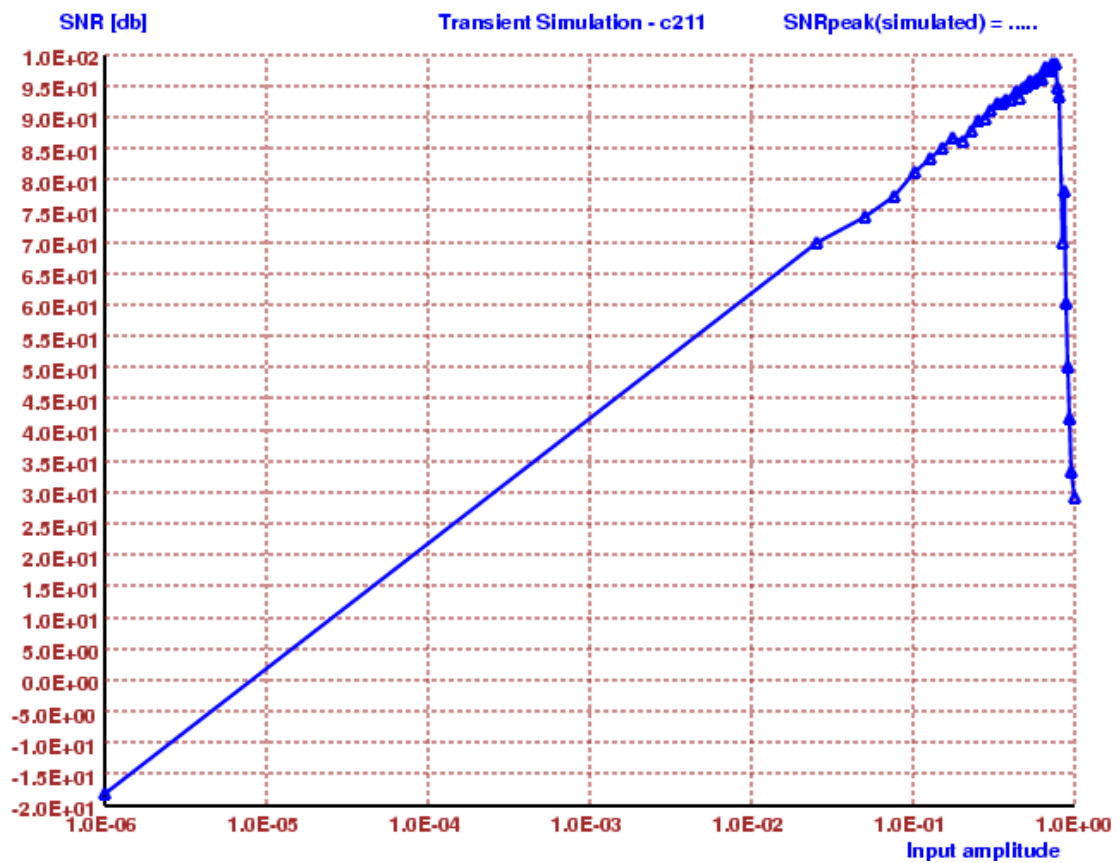


Figure 9: Typical overload curve of the ideal modulator from example 1.

## 4 High-Level Synthesis of $\Delta\Sigma$ ADCs

Now that we have extensively explored the simulator capabilities in the previous section, we are ready for the next step: the synthesis of a  $\Delta\Sigma$  ADC. If we review the literature, we find that  $\Delta\Sigma$  converters are being used for a wide range of applications, going from instrumentation to telecom. Because of this wide application range the specifications of the converter are also very different in value. Therefore the implemented methodology (see figure 10) selects the optimum topology together with the building block specifications satisfying the application requirements. In our case, the optimum topology is based on a relative power merit. Due to the exploration

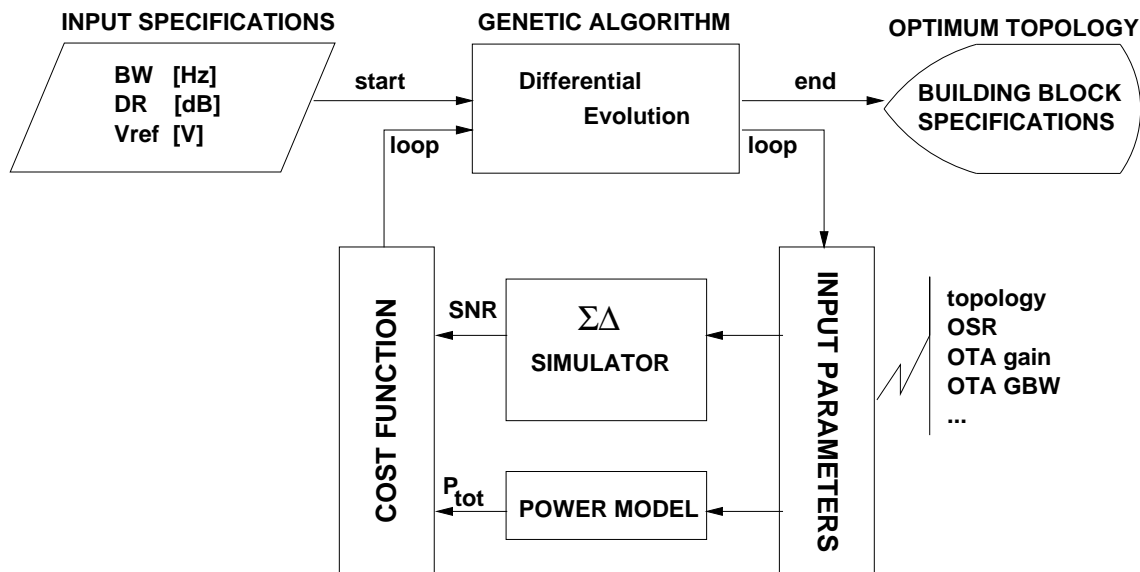


Figure 10: Flow of the implemented methodology.

process of the genetic algorithm, a fast simulator is necessary (see subsection 1.3). We will not go into more details of the methodology – more information can be found in [4].

The design parameters we will take into account are:

- topology
- oversampling ratio
- finite OTA gain
- finite OTA GBW
- finite OTA output swing
- non zero switch on resistance
- comparator offset

- comparator hysteresis

This means that the final outcome of the procedure will be the optimized topology and oversampling ratio, together with specifications for the building blocks. The main specifications of a  $\Delta\Sigma$  ADC are:

- dynamic range [dB]
- signal bandwidth [Hz]
- reference voltage [V]

Figure 11 illustrates the specification window. The reference voltage is set to 1 V and we don't consider any other nonidealities than the ones itemized above. It is now possible to reproduce the results published in [4] by optimizing the topology and building block specifications for the parameters dynamic range and signal bandwidth as defined in table 1.

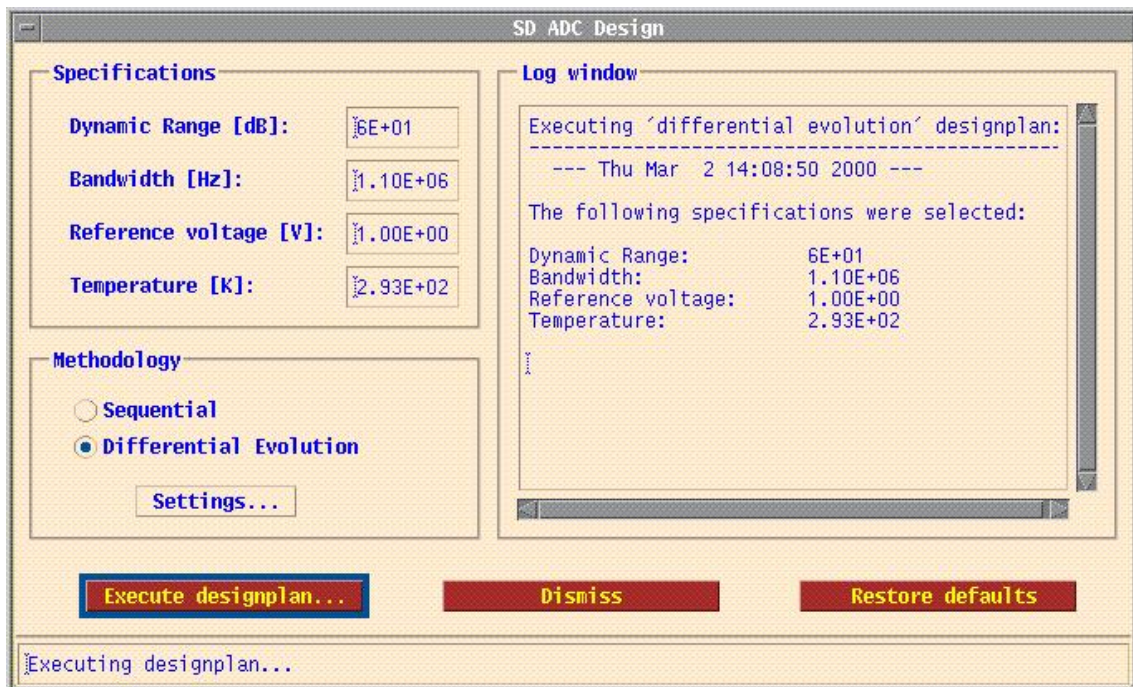


Figure 11: Input specification window for the  $\Delta\Sigma$  ADC design procedure.

Table 1: Parameters for the  $\Delta\Sigma$  ADC synthesis experiments.

<b>dynamic range: signal bandwidth</b>	<b>30 dB</b>	<b>45 dB</b>	<b>60 dB</b>	<b>75 dB</b>
<b>25 kHz</b>	<i>topology:</i> <i>OSR:</i> <i>A<sub>OTA</sub>:</i> <i>GBW<sub>OTA</sub>:</i> <i>OS<sub>OTA</sub>:</i> <i>R<sub>switch</sub>:</i> <i>Offset<sub>comp</sub>:</i> <i>Hyst<sub>comp</sub>:</i> <i>t<sub>exec</sub>:</i>	<i>topology:</i> <i>OSR:</i> <i>A<sub>OTA</sub>:</i> <i>GBW<sub>OTA</sub>:</i> <i>OS<sub>OTA</sub>:</i> <i>R<sub>switch</sub>:</i> <i>Offset<sub>comp</sub>:</i> <i>Hyst<sub>comp</sub>:</i> <i>t<sub>exec</sub>:</i>	<i>topology:</i> <i>OSR:</i> <i>A<sub>OTA</sub>:</i> <i>GBW<sub>OTA</sub>:</i> <i>OS<sub>OTA</sub>:</i> <i>R<sub>switch</sub>:</i> <i>Offset<sub>comp</sub>:</i> <i>Hyst<sub>comp</sub>:</i> <i>t<sub>exec</sub>:</i>	<i>topology:</i> <i>OSR:</i> <i>A<sub>OTA</sub>:</i> <i>GBW<sub>OTA</sub>:</i> <i>OS<sub>OTA</sub>:</i> <i>R<sub>switch</sub>:</i> <i>Offset<sub>comp</sub>:</i> <i>Hyst<sub>comp</sub>:</i> <i>t<sub>exec</sub>:</i>
<b>100 kHz</b>	<i>topology:</i> <i>OSR:</i> <i>A<sub>OTA</sub>:</i> <i>GBW<sub>OTA</sub>:</i> <i>OS<sub>OTA</sub>:</i> <i>R<sub>switch</sub>:</i> <i>Offset<sub>comp</sub>:</i> <i>Hyst<sub>comp</sub>:</i> <i>t<sub>exec</sub>:</i>	<i>topology:</i> <i>OSR:</i> <i>A<sub>OTA</sub>:</i> <i>GBW<sub>OTA</sub>:</i> <i>OS<sub>OTA</sub>:</i> <i>R<sub>switch</sub>:</i> <i>Offset<sub>comp</sub>:</i> <i>Hyst<sub>comp</sub>:</i> <i>t<sub>exec</sub>:</i>	<i>topology:</i> <i>OSR:</i> <i>A<sub>OTA</sub>:</i> <i>GBW<sub>OTA</sub>:</i> <i>OS<sub>OTA</sub>:</i> <i>R<sub>switch</sub>:</i> <i>Offset<sub>comp</sub>:</i> <i>Hyst<sub>comp</sub>:</i> <i>t<sub>exec</sub>:</i>	<i>topology:</i> <i>OSR:</i> <i>A<sub>OTA</sub>:</i> <i>GBW<sub>OTA</sub>:</i> <i>OS<sub>OTA</sub>:</i> <i>R<sub>switch</sub>:</i> <i>Offset<sub>comp</sub>:</i> <i>Hyst<sub>comp</sub>:</i> <i>t<sub>exec</sub>:</i>
<b>500 kHz</b>	<i>topology:</i> <i>OSR:</i> <i>A<sub>OTA</sub>:</i> <i>GBW<sub>OTA</sub>:</i> <i>OS<sub>OTA</sub>:</i> <i>R<sub>switch</sub>:</i> <i>Offset<sub>comp</sub>:</i> <i>Hyst<sub>comp</sub>:</i> <i>t<sub>exec</sub>:</i>	<i>topology:</i> <i>OSR:</i> <i>A<sub>OTA</sub>:</i> <i>GBW<sub>OTA</sub>:</i> <i>OS<sub>OTA</sub>:</i> <i>R<sub>switch</sub>:</i> <i>Offset<sub>comp</sub>:</i> <i>Hyst<sub>comp</sub>:</i> <i>t<sub>exec</sub>:</i>	<i>topology:</i> <i>OSR:</i> <i>A<sub>OTA</sub>:</i> <i>GBW<sub>OTA</sub>:</i> <i>OS<sub>OTA</sub>:</i> <i>R<sub>switch</sub>:</i> <i>Offset<sub>comp</sub>:</i> <i>Hyst<sub>comp</sub>:</i> <i>t<sub>exec</sub>:</i>	<i>topology:</i> <i>OSR:</i> <i>A<sub>OTA</sub>:</i> <i>GBW<sub>OTA</sub>:</i> <i>OS<sub>OTA</sub>:</i> <i>R<sub>switch</sub>:</i> <i>Offset<sub>comp</sub>:</i> <i>Hyst<sub>comp</sub>:</i> <i>t<sub>exec</sub>:</i>
<b>1000 kHz</b>	<i>topology:</i> <i>OSR:</i> <i>A<sub>OTA</sub>:</i> <i>GBW<sub>OTA</sub>:</i> <i>OS<sub>OTA</sub>:</i> <i>R<sub>switch</sub>:</i> <i>Offset<sub>comp</sub>:</i> <i>Hyst<sub>comp</sub>:</i> <i>t<sub>exec</sub>:</i>	<i>topology:</i> <i>OSR:</i> <i>A<sub>OTA</sub>:</i> <i>GBW<sub>OTA</sub>:</i> <i>OS<sub>OTA</sub>:</i> <i>R<sub>switch</sub>:</i> <i>Offset<sub>comp</sub>:</i> <i>Hyst<sub>comp</sub>:</i> <i>t<sub>exec</sub>:</i>	<i>topology:</i> <i>OSR:</i> <i>A<sub>OTA</sub>:</i> <i>GBW<sub>OTA</sub>:</i> <i>OS<sub>OTA</sub>:</i> <i>R<sub>switch</sub>:</i> <i>Offset<sub>comp</sub>:</i> <i>Hyst<sub>comp</sub>:</i> <i>t<sub>exec</sub>:</i>	<i>topology:</i> <i>OSR:</i> <i>A<sub>OTA</sub>:</i> <i>GBW<sub>OTA</sub>:</i> <i>OS<sub>OTA</sub>:</i> <i>R<sub>switch</sub>:</i> <i>Offset<sub>comp</sub>:</i> <i>Hyst<sub>comp</sub>:</i> <i>t<sub>exec</sub>:</i>

## References

- [1] Steven Norsworthy, Richard Schreier, and Gabor Temes, *Delta-Sigma Data Converters: theory, design, and simulation*, IEEE Press, 1996.
- [2] A. Marques, *High Speed CMOS Data Converters*, Ph.D. thesis, K.U.Leuven, 1999.
- [3] Kenneth Francken and Georges Gielen, “D1.1.1: Analysis of major nonidealities of multibit oversampling converters,” deliverable, K.U. Leuven, 1999, URL: <http://www.esat.kuleuven.ac.be/sysconv/>.
- [4] 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.