ACCURATE VHDL-BASED SIMULATION OF ΣΔ MODULATORS

R. Castro-López[1], F. V. Fernández[1,2] F. Medeiro[1,2] and A. Rodríguez-Vázquez[1,2]
[1]IMSE-CNM, Edif. CICA, Avda. Reina Mercedes s/n, E-41012 Sevilla, SPAIN
Phone: +34 955056666, FAX: +34 955056686, e-mail: castro@imse.cnm.es
[2]Escuela Superior de Ingenieros, Isla de la Cartuja s/n, E-41092 Sevilla, SPAIN
Phone: +34 954487378

ABSTRACT

Computational cost of transient simulation of ΣΔ modulators (ΣΔMs) at the electrical level is prohibitively high. Behavioral simulation techniques arise as a promising solution to this problem. This paper demonstrates that both, hardware description languages (HDLs) and commercial HDL simulators, constitute a valuable alternative to traditional special-purpose ΣΔ behavioral simulators. In this sense, a library of HDL building blocks, modeling a complete set of circuit non-idealities which influence the performance of ΣΔMs, is presented. With these blocks, ΣΔM architectures can be described in two different ways, which are analyzed in detail. Experimental results are provided through several simulations of a fourth-order 2-1-1 cascade multi-bit ΣΔM.

1. INTRODUCTION

Among the architectures performing signal conversion between the analog and digital worlds, ΣΔMs have become very popular thanks to their ability to solve problems exhibited by other architectures. For instance, ΣΔMs do not need high-accuracy analog antialiasing filtering and have relatively large insensitivity to circuit imperfections or noisy environments [1],[2].

During the design process of ΣΔMs (actually of any integrated circuit), accurate emulation of the circuit's behavior, by using appropriate simulation techniques, becomes an essential step previous to circuit fabrication. The most accurate one amid available simulation techniques is, undoubtedly, electrical simulation. Unfortunately, although theoretically possible, it may become impractical for circuits like ΣΔMs. For instance, days or weeks of CPU time can be required to estimate the signal-to-noise ratio of a typical ΣΔM architecture [3],[4].

Macromodel descriptions of building blocks, i.e., opamps, can only simplify the problem slightly. This is because the resulting system of differential equations must still be solved numerically. Thus, these techniques are not fast enough for simulation of ΣΔMs. To separate the analog and digital parts of the circuit and simulate them dependently is the basis for mixed-signal (multilevel) simulation. But as long as numerical resolution of differential equations is used for the analog part, the extraction of ΣΔM performances will still be too costly in terms of CPU time.

Event-driven behavioral simulation is a smart solution to further reduce computational cost. This kind of simulation technique involves a circuit partitioning into basic behavioral blocks with fully independent functionality. This means that an instantaneous block output cannot be related to itself, or, in other words, either there is no global feedback loop, or in case such loop exists, there is a delay that avoids the instantaneous dependence. A behavioral simulation of the complete circuit requires, then, a behavioral model for each building block of the circuit. This model takes the form of explicit expressions relating the output variable with the input and internal state variables.

Different approaches to behavioral simulation of ΣΔM have been reported. Special-purpose approaches have been described in [4]-[10]. These are, as their name suggests, simulation techniques specifically devoted to behavioral description and simulation of ΣΔ converters. Among them, the main differences are in the number of topologies/basic blocks included, the accuracy of the models used, the postprocessing capabilities and the friendliness of the user interface. ASIDES [9] is a representative example of this group. In this tool, both, behavioral models of the building blocks composing the ΣΔMs and the simulation engine, are written in C language. This enhances the simulation speed, making ASIDES very appropriate, not only for ΣΔM validation, but also for ΣΔM synthesis (i.e., in simulation-based optimization approaches). A drawback is that ASIDES, like the rest of special-purpose ΣΔ behavioral simulators, is restricted to the simulation of this class of systems.

Simulation approaches based on HDLs (i.e., *VHDL* [11] or *Verilog* [12] and their analog extensions *VHDL-AMS* [13] and *Verilog-AMS*) can overcome the limitation of special-purpose approaches since HDL simulators are found in many commercial design environments. In this way, ΣΔMs modeled with *VHDL* or Verilog can be simulated together with other *VHDL*-modeled blocks and even continuous-time descriptions of blocks, modeled using *VHDL-AMS* or *Verilog-AMS*. Furthermore, different description levels (extracted layout, transistor, macromodel or behavioral levels) can be combined into a single simulation. This is clearly advantageous when compared with special-purpose simulators.

In this paper, the *VHDL* language is used to model high-performance ΣΔMs. Special care has been put to capture all major non-idealities affecting the nominal behavior of these circuits. The paper is organized as follows. An overview of the non-idealities considered and modeled with *VHDL* is provided in Section 2. Section 3 describes two different techniques adopted to describe ΣΔM topologies in detail. Section 4 shows experimental results of a ΣΔM simulation. Concluding remarks are given in Section 5.

2. HDL MODELING OF ΣΔM BUILDING BLOCKS

A basic block representation of a ΣΔM is shown in Figure 1. The non-idealities degrading its behavior can be implemented separately by including imperfections in the performance of each of the three

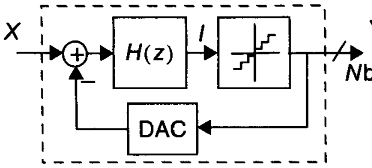


Figure 1. Basic structure of a ΣΔ modulator.

building blocks: the discrete-time filter, $H(z)$, the quantizer and the DA converter (DAC). The ideal behavior is as follows: the output y is subtracted from the input signal, x, which has been sampled at a rate larger than the Nyquist frequency. The result, after passing through $H(z)$, is the input to the quantizer. The goal is to reduce the quantization power spectral density in the low-frequency range. The simplest way to do it is to implement $H(z)$ as a SC integrator, thus performing a shaping on the quantization error.

2.1. SC Integrators

The integrator is the fundamental block because its non-idealities largely affect the performance of ΣΔMs. Figure 2 shows a typical SC integrator used in ΣΔMs.

The set of non-idealities considered and modeled with *VHDL* are: thermal noise, finite and non-linear dc gain, output range, opamp slew-rate and dynamics, switch resistance and capacitor non-linearity and mismatching [2]. These non-ideal effects are included in the complete integrator model, illustrated with the flow graph in Figure 3. During the integration phase, an iterative procedure is started to calculate the integrator output voltage, including the effects of the finite and non-linear opamp, the non-linear capacitors, transient response and the output range limitation [2],[14]. While most integrator models limit the opamp dynamics to the integration phase, introducing these effects also in the sampling phase may become important, especially for high-speed applications. During the sampling phase, the final value of the voltages stored in the input capacitors are calculated considering the values of these and the ON resistance of the switches. The input equivalent thermal noise of the integrator is calculated and added to the sampled voltage. Settling errors are then evaluated.

There is a problem regarding the implementation of the settling error model with *VHDL*. For ΣΔMs of order larger than 1, the calculation of the opamp input and output voltages requires to know the voltage stored in the sampling capacitors of the following integrator during the previous phase. Since each block has to be modeled independently (e.g., its description cannot be related to variables of other blocks), one solution is to accommodate such voltages in the so-called dynamic external ports of the block model with *VHDL*. As these ports can only have only fixed values during a simulation, they have to be transformed into input ports, thus increasing the number of input ports of the integrators. Consequently, block interconnection for complex ΣΔ architectures may become a rather complex task.

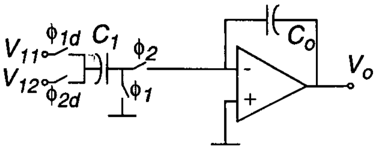


Figure 2. SC integrator.

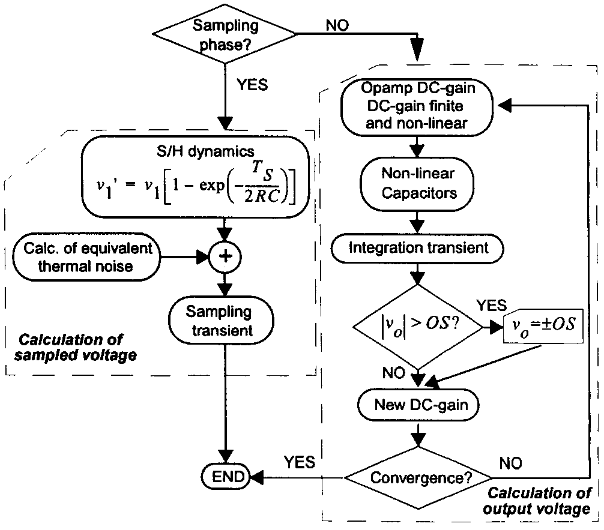

Sampling phase?
NO
YES
Opamp DC-gain DC-gain finite and non-linear
S/H dynamics
$v_1' = v_1\left[1 - \exp\left(-\frac{T_S}{2RC}\right)\right]$
Non-linear Capacitors
Calc. of equivalent thermal noise
+
Integration transient
Sampling transient
YES
$|v_o| > OS?$
$v_o = \pm OS$
Calculation of sampled voltage
NO
New DC-gain
END
YES
Convergence?
NO
Calculation of output voltage

Figure 3. Complete integrator model.

2.2. Quantizers and DA converters

Single-bit architectures incorporate a simple comparator to perform the internal quantization. A simple DA converter is then used in the feedback loop, which does not introduce any non-linearity error. Some comparators also exhibit a hysteresis of random nature. This is illustrated in Figure 4(a), where the transfer curve of such a comparator is shown. The corresponding behavioral model of the comparator with hysteresis is described with the flow diagram in Figure 4(b). It presents three well-differentiated operation zones: in the first of them, in response to an input, the output value (V_{max} or V_{min}) is a function of the sign of $|v_i - V_{off}|$, provided that $|v_i - V_{off}| > h/2$ is fulfilled, where V_{off} and h represent the offset and hysteresis of the comparator, respectively. Otherwise, the output is randomly determined in the dynamic hysteresis model and simply does not change in the deterministic model. For illustration's sake, the *VHDL* implementation of the comparator model is shown in Figure 5.

For multi-bit DA converters, characterized by an offset *off*, a gain error γ, and an integral non-linearity (INL), the model contains first an ideal DA converter, as shown in Figure 6. The result goes through a non-linear block with a third-order non-linearity and a gain error block. Finally, the offset is added [2]. An analogous scheme is used for multi-bit quantizers.

3. HDL MODELING OF ΣΔM ARCHITECTURES

ΣΔM architectures can be described by interconnecting the building blocks above. This can be accomplished by following two different procedures in *VHDL*. The first one consists in defining the ΣΔM architecture through *VHDL* instances [11].

The second procedure, more user-friendly, involves the use of schematic capture tools. With these resources, it is possible to build ΣΔMs just by placing and routing pre-defined symbols. Figure 7 shows the schematic of a 2-1-1 cascade multi-bit ΣΔM. The same architecture built with Mentor Graphics's HDL Designer® sche-

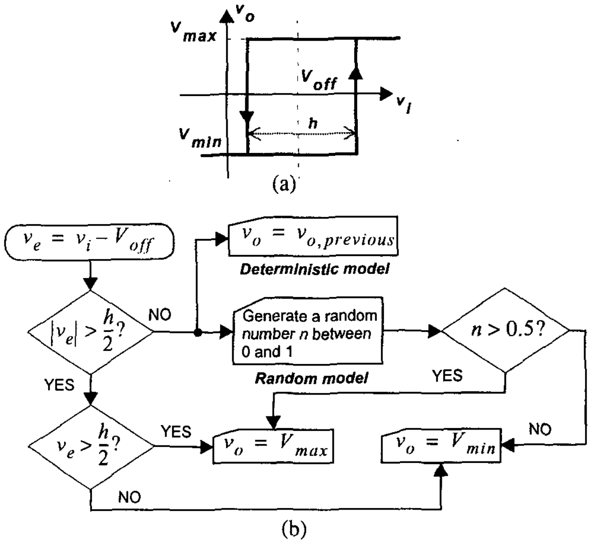
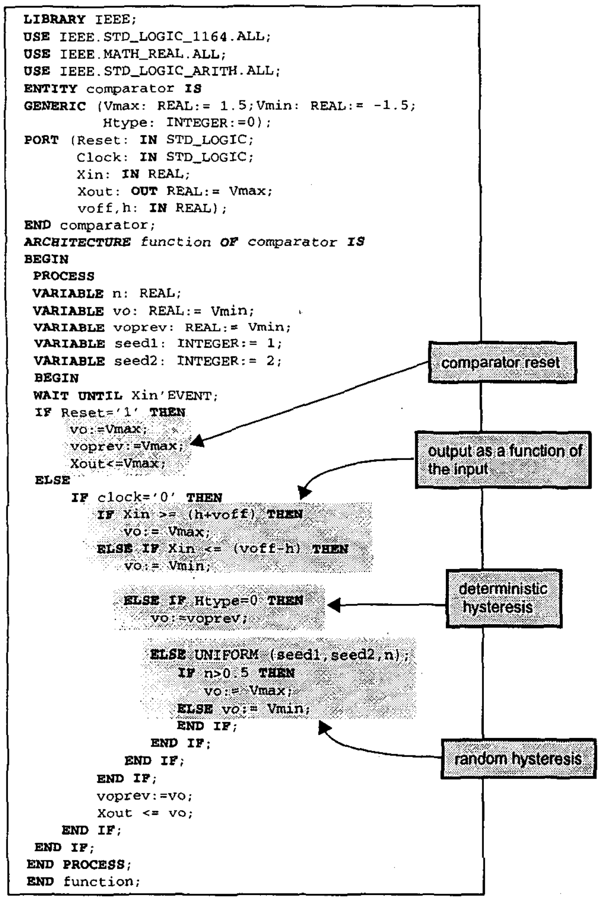


Figure 4. (a) Transfer curve of a comparator with hysteresis and (b) its model flow graph.


comparator reset
output as a function of the input
deterministic hysteresis
random hysteresis

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.MATH_REAL.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
ENTITY comparator IS
GENERIC (Vmax: REAL:= 1.5;Vmin: REAL:= -1.5;
         Htype: INTEGER:=0);
PORT (Reset: IN STD_LOGIC;
      Clock: IN STD_LOGIC;
      Xin: IN REAL;
      Xout: OUT REAL:= Vmax;
      voff,h: IN REAL);
END comparator;
ARCHITECTURE function OF comparator IS
BEGIN
 PROCESS
 VARIABLE n: REAL;
 VARIABLE vo: REAL:= Vmin;
 VARIABLE voprev: REAL:= Vmin;
 VARIABLE seed1: INTEGER:= 1;
 VARIABLE seed2: INTEGER:= 2;
 BEGIN
 WAIT UNTIL Xin'EVENT;
 IF Reset='1' THEN
      vo:=Vmax;
      voprev:=Vmax;
      Xout<=Vmax;
 ELSE
     IF clock='0' THEN
        IF Xin >= (h+voff) THEN
           vo:= Vmax;
        ELSE IF Xin <= (voff-h) THEN
           vo:= Vmin;

          ELSE IF Htype=0 THEN
             vo:=voprev;

             ELSE UNIFORM (seed1,seed2,n);
                IF n>0.5 THEN
                   vo:= Vmax;
                ELSE vo:= Vmin;
                END IF;
             END IF;
          END IF;
        END IF;
        voprev:=vo;
        Xout <= vo;
    END IF;
 END IF;
END PROCESS;
END function;
```

Figure 5. *VHDL* model of comparator with hysteresis.

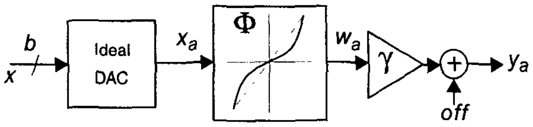


Figure 6. Behavioral model of DA converters.

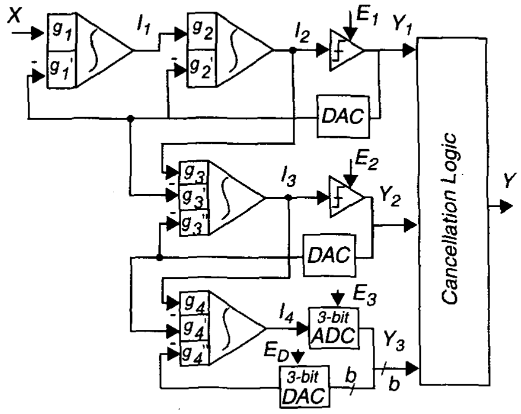


Figure 7. Fourth-order 2-1-1 cascade multi-bit ΣΔ modulator.

matic capture tool is depicted in Figure 8.

In both alternatives, the user must know the number of parameters and inputs of every building block, including the extra ports corresponding to voltages in the sampling capacitors, as explained in Section 2.1. To avoid this extra burden, an optional facility has been developed. It allows the user to introduce just the *VHDL* code or create the schematic of the modulator architecture (with no additional input ports). Then, circuit connectivity is automatically traced and a correct *VHDL* code, with all extra information, is generated.

For subsequent data processing, a number of capabilities have also been implemented as *VHDL* functions. This enable full exploitation of the behavioral simulation results and allows to obtain the output spectrum, the SNDR as a function of the input level or frequency, the in-band error power and the effective resolution. Other *VHDL* functions have been created to perform several types of analysis, such as Monte Carlo or parametric analysis.

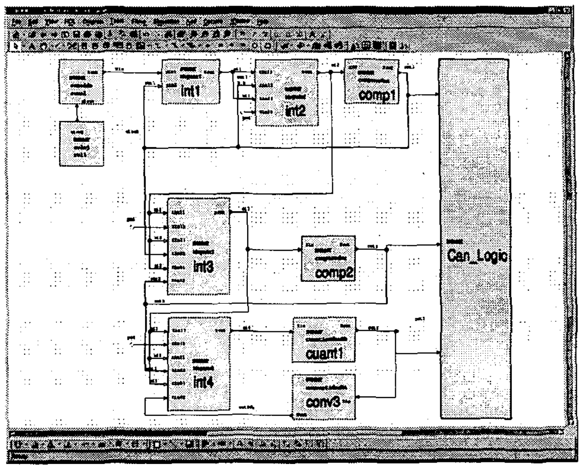


Figure 8. Snapshot of a 2-1-1 cascade multi-bit ΣΔ modulator drawn with HDL Designer®.

4. EXPERIMENTAL RESULTS

HDL simulation of ΣΔM behavior presented here, will be illustrated via behavioral simulations of the 2-1-1 cascade multibit ΣΔM of Figure 7. The result of a *VHDL*-based behavioral simulation is shown in Figure 9. There, the influence of the non-linearity of the DA converter in the in-band error power is analyzed by performing a parametric simulation. Compilation of the *VHDL* ΣΔM architecture took 7.5s. of CPU time and simulation[1] took 203.5s, using Mentor Graphics' Advance-MS®. The same simulation required 25s. in ASIDES [2] to provide identical results (note that block models were exactly the same in both cases). Thus, ASIDES is more suitable for processes where repetitive simulation is required, i.e., in simulation-based iterative optimization tasks. However, the capability to simulate *VHDL*-modeled ΣΔMs together with other *VHDL*- or *VHDL-AMS*-modeled subsystems suggests this as a more flexible methodology for system-level verification.

It is very interesting to evaluate which is the accuracy of the behavioural simulation and how it compares with a conventional electrical simulator. The power spectral density (PSD) plots in Figure 10 were obtained from three different sources. First, it was computed, from 65536 samples, by simulating the *VHDL*-modelled ΣΔM in Figure 7 with Mentor Graphics' Advance-MS®. Simulation time was 3.6s. Second, the PSD was also obtained with HSPICE. The simulation took 5 days of CPU time to get only 8192 samples. It can be observed that HSPICE computes a lower error power because thermal noise cannot be included in the transient simulation. Finally, the PSD was measured directly from a chip prototype (a different signal frequency has been chosen for better visualization)

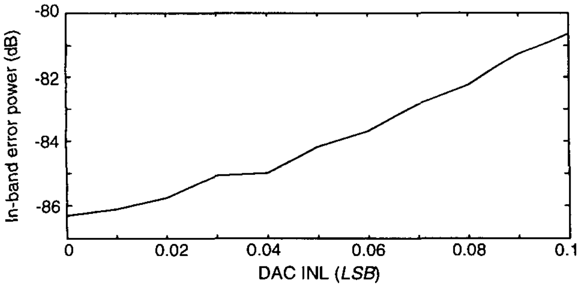


Figure 9. Influence of the DA converter non-linearity on the in-band error power.

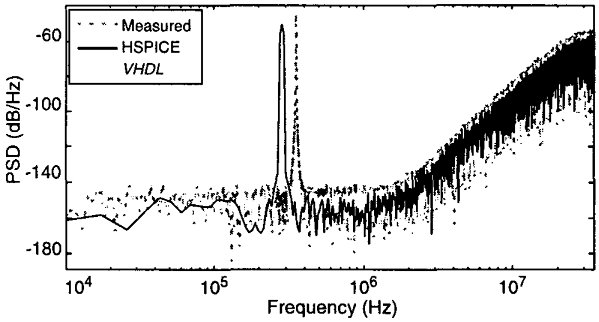


Figure 10. Simulated and measured PSD.

5. CONCLUSIONS

HDLs together with commercial behavioral simulators are efficient resources to perform accurate verification of ΣΔMs. They also allow to combine ΣΔMs with *VHDL* descriptions of digital blocks or *VHDL-AMS* descriptions of other analog blocks (even a *VHDL-AMS* description of any block of the ΣΔM can be used). The capability to simulate ΣΔMs within more complex systems and the integration into commercial design environments makes HDLs valuable alternatives to special-purpose ΣΔ behavioral simulators.

6. REFERENCES

[1] J. Candy and G.C. Temes, "Oversampling Methods for A/D and DA Conversion," in *Oversampling ΣΔ converters*, pp. 1-25, IEEE Press, 1992.

[2] F. Medeiro, B. Pérez-Verdú and A. Rodríguez-Vázquez, *Top-Down Design of High-Performance Sigma-Delta Modulators*. Kluwer Academic Publishers, Boston, 1999.

[3] V. F. Dias, V. Liberali and F. Maloberti: "Design Tools for Oversampling Data Converters: Needs and Solutions", *Microelectronics Journal*, Vol. 23, pp. 641-650, 1992.

[4] C. H. Wolff and L. Carley: "Simulation of Δ-Σ Modulators Using Behavioral Models", *Proc. IEEE Int. Symp. Circuits and Systems*, pp. 376-379, 1990.

[5] V. Liberali, V.F. Dias, M. Ciapponi and F. Maloberti, "TOSCA: a Simulator for Switched-capacitor Noise-shaping A/D Converters," *IEEE Trans. Computer-Aided Design*, Vol. 12, No. 9, pp. 1376-1386, Sept. 1993.

[6] S. R. Norsworthy, I. G. Post and H. S. Fetterman: "A 14-bit 80-kHz Sigma-Delta A/D Converter: Modeling, Design and Performance Evaluation", *IEEE J. Solid-State Circuits*, Vol. 24, pp. 256-266, April 1989.

[7] A. Opal: "Sampled Data Simulation of Linear and Nonlinear Circuits", *IEEE Trans. on Computer-Aided Design*, Vol. 15, pp. 295-306, March 1996.

[8] T. Ritoniemi, T. Karema, H. Tenhunen and M. Lindell: "Fully-differential CMOS Sigma-Delta Modulator for High-Performance Analog-to-Digital Conversion with 5 V Operating Voltage", *Proc. IEEE Int. Symp. Circuits and Systems*, pp. 2321-2326, 1988.

[9] F. Medeiro, B. Pérez-Verdú, A. Rodríguez-Vázquez and J.L. Huertas "A Vertically Integrated Tool for Automated Design of ΣΔ Modulators" *IEEE J.Solid-State Circuits*, pp. 762-777, July 1995.

[10] K. Francken, P. Vancorenland and G. Gielen, "DAISY: a Simulation-based High-level Synthesis Tool for ΔΣ Modulators," *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, pp. 188-192, 2000.

[11] *IEEE VHDL Language Reference Manual*, IEEE Std 1076-2002.

[12] *IEEE Standard Verilog hardware description language*, IEEE Std 1364-2001.

[13] *IEEE VHDL 1076.1 Language Reference Manual*, IEEE Std 1076.1-1999.

[14] R del Río, F. Medeiro, B. Pérez-Verdú and A Rodríguez-Vázquez "Reliable Analysis of Settling Errors in SC Integrators: Application to ΣΔ Modulators," *IEE Electronics Letters*, pp. 503-504, Vol. 36 Iss. 6, March 2000.

[1] All simulations were performed on a SunFire 3800 @ 750 MHz.